Learning-Based Arabic Word Spotting Using a Hierarchical Classifier

Muna Al-Khayat

A Thesis

In The Department of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy in Computer Science

Concordia University

Montréal, Québec, Canada

December 2013

Learning-Based Arabic Word Spotting Using a Hierarchical Classifier

Muna Al-Khayat

Concordia University, 2013

Abstract

The effective retrieval of information from scanned handwritten documents is becoming essential with the increasing amounts of digitized documents, and therefore developing efficient means of analyzing and recognizing these documents is of significant interest. Among these methods is word spotting, which has recently become an active research area. Such systems have been implemented for Latin-based and Chinese languages, while few of them have been implemented for Arabic handwriting. The fact that Arabic writing is cursive by nature and unconstrained, with no clear white space between words, makes the processing of Arabic handwritten documents a more challenging problem.

In this thesis, the design and implementation of a learning-based Arabic handwritten word spotting system is presented. This incorporates the aspects of text line extraction, handwritten word recognition, partial segmentation of words, word spotting and finally validation of the spotted words.

The Arabic text line is more unconstrained than that of other scripts, essentially since it also includes small connected components such as dots and diacritics that are usually located between lines. Thus, a robust method to extract text lines that takes into consideration the challenges in the Arabic handwriting is proposed. The method is evaluated on two Arabic handwritten documents databases, and the results are compared with those of two other methods for text line extraction. The results show that the proposed method is effective, and compares favorably with the other methods.

Word spotting is an automatic process to search for words within a document. Applying this process to handwritten Arabic documents is challenging due to the absence of a clear space between handwritten words. To address this problem, an effective learning-based method for Arabic handwritten word spotting is proposed and presented in this thesis. For this process, sub-words or pieces of Arabic words form the basic components of the search process, and a hierarchical classifier is implemented to integrate statistical language models with the segmentation of an Arabic text line into sub-words.

The holistic and analytical paradigms (for word recognition and spotting) are studied, and verification models based on combining these two paradigms have been proposed and implemented to refine the outcomes of the analytical classifier that spots words.

Finally, a series of evaluation and testing experiments have been conducted to evaluate the effectiveness of the proposed systems, and these show that promising results have been obtained.

Thesis Supervisor: Ching Y. Suen
Title: Professor

Thesis Supervisor: Louisa Lam
Title: Affiliate Professor

Dedication

I would like to dedicate this thesis to my dearest parents, who installed in me the value of science and gave me the means and tools to attain it. I am deeply grateful for my parents' love, care and support.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Ching Y. Suen, for the great efforts of taken the time to read drafts of my thesis and papers, for his persistent guidance, continuous support and kind encouragement. Special thanks to my co-supervisor Dr. Louisa Lam, for the great efforts put forth by her taking the time to read so many drafts of my papers and thesis and providing me with corrections, wise advices, constructive feedback and countless suggestions. Her insights have significantly strengthen this thesis. Working with her is an honor. I am extremely grateful to my hosting supervisor in Chine, Dr. Cheng-Lin Liu. Who hosted me during my research fellowship to the National Institute of Pattern Recognition (NIPR), Beijing, China. Many thanks for his constructive discussions and feedbacks, the great research environment he provided, and also for his encouragement and support.

I would like to thank my examination committee, for their evaluations and valuable comments and feedbacks, which I highly appreciate.

I wish to thank all my friends and colleagues, also staff in the Centre of Pattern Recognition and Machine Intelligence (CENPARMI) for providing a happy and encouraging research environment. Special thanks to Graciela Meza Lovon, David Yu, Dr. Chun Lei He, Jehan Janbi, Mrouj Almuhajri and Fariba Haghbin. I am grateful to our research manager Mr. Nicola Nobile who is always available for any technical support and help. Finally, special thanks to our administrative assistant Ms. Marleah Blom, for her cheerfulness administrative assistant.

I would like to thank also all the visiting researchers who passed by CENPARMI. Special thanks to Dr. Andreas Fischer who provided a nice and corporative research environment, also for his inspiration, valuable comments and advices and constructive discussions.

Thank you my parents, sisters, brother, nieces and nephews. Great thanks to my family who inspired, enlightened and gave me the strength and support to carry on with this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

A great number of handwritten documents have been digitized, to preserve, analyze, and disseminate them. These documents are of different categories, being drawn from fields as diverse as history, commerce, finance, and medicine. As the sheer number of handwritten documents being digitized continues to increase, the need for indexing them becomes vital. Word spotting is an approach that allows a user to search for keywords in spoken or written textS. While initially developed for use in Automatic Speech Recognition (ASR), word spotting has since been applied to the growing number of handwritten documents for the purpose of indexing. Even though speech is analog in nature, while handwritten documents are spatial, word spotting of handwritten documents has been able to adopt the methods of speech recognition for its use. Eventually, techniques and algorithms specific to handwritten documents have been developed.

Early indexing work started by applying conventional Optical Character Recognition (OCR) techniques, and the results are passed to special search engines to search for words. However, Manmatha et al. [2] designed the first handwritten word spotting system in 1996 because they found that applying traditional OCR techniques to search for words is inadequate. Using OCR in indexing words fails for the following reasons [3, 4]: 1) handwriting analysis suffers from low recognition accuracies; 2) the

associated indexing systems are hampered by having to process and recognize all the words of a document, and then apply search techniques to the entire result; and 3) the training of OCR systems requires that a huge database be constructed for each alphabet.

Word spotting methods are based on two main approaches: template matching and learning-based. Manmatha et al. [2] proposed the first indexing or word spotting system for single writer historical documents. The proposed method was based on matching word pixels. Zhang et al. [5] proposed a template matching approach based on extracting features from word images. Dynamic Time Warping (DTW) [3, 6, 7] was successfully applied as an efficient template matching algorithm. Learning-based word spotting systems were introduced to adapt to muli-writers with promising results. However, sufficiently large databases are needed to train these systems.

Several successful handwritten word spotting systems have been proposed in the literature. Most of these systems are applied to Latin-based and Chinese scripts, while less attention has been devoted to Arabic handwritten word spotting systems. Arabic writing is a cursive horizontal script whose words consist of sub-words or Pieces of Arabic Words (PAWs), each of which consists of one or more letters. In general, the white spaces between words and PAWs may be of similar sizes, so that the boundaries of words are not clearly indicated. This, together with the naturally cursive structure of Arabic writing which tends to be more unconstrained than in other languages, make word spotting in the Arabic language a challenging problem in need of further research.

Many studies [8, 9, 10] were oriented towards viewing an Arabic text line as a sequence of PAWs instead of words. This is because a PAW consists of one major connected component and some or no minor connected components, which makes it easy to segment a text line into PAWs. Saabni and El-Sana [9] favored spotting PAWs instead of Arabic handwritten words. Sari and Kefali [8] segmented the document into connected components (sub-words or PAWs) each of which was represented by global features such as loops, ascenders, etc. Then an approximate string matching algorithm was used to search for the subwords based on the Levenshtein edit distance.

Moghaddam and Cheriet [10] proposed a system for word spotting in historical documents by matching the shapes of query words to those of the document images through comparing the skeletons of connected components.

Some Arabic word spotting systems [11] are based on segmentation-free approaches, since these approaches have shown promising results when applied to Latin-based word spotting systems. However, these approaches have not produce promising results on Arabic handwritten documents, so further research is needed to implement segmentation-free approaches to Arabic handwritten word spotting.

1.2 Objective

The Arabic language is one of the most widely spoken languages in the world. It is spoken by more than 256 million people, and it is one of the six formal languages of the United Nations. Nevertheless, little work has been done on Arabic handwriting analysis and recognition until the recent years.

Many document analysis studies try to improve the performance of the text line segmentation methods and algorithms, since errors resulting from a text line extraction algorithm would be carried on to the document recognition systems. Extracting text lines from handwritten documents is more difficult than from printed documents, because handwritten text lines are unconstrained, can be overlapping and skewed. Arabic handwriting is more unconstrained than other scripts. It also consists of small connected components called diacritics and dots, which are usually located between the text lines. These diacritics and dots are often misplaced when Arabic text lines are automatically segmented. All these factors make Arabic text line segmentation more challenging.

Many Arabic document analysis and recognition systems are modeled based on PAWs, where a PAW usually contains one complete major connected component which is easy to extract and some or no minor connected components. However, because of different writing styles, PAWs may be overlapping, touching or disconnected. This makes it crucial to find an effective algorithm to segment Arabic text lines or

word images into PAWs.

Arabic handwriting is cursive by nature and each letter in the Arabic language has two to four forms. In addition, many letters in the Arabic language share the same shape and differ only by the number and/or location of dots. These facts introduce difficulties to the Arabic word recognition systems. Different feature extraction methods and many classifiers have been proposed in the literature of handwriting recognition, but not all of them are suitable for recognition of handwritten Arabic words. This increases the need for experimenting with different features and classifiers that can be applied to recognize Arabic handwritten words and/or PAWs.

Word spotting is defined as a process of searching for the visual appearance of a word within a document or a text line. While word spotting systems can be application dependent, they can be divided into two broad classes depending on the lexicon: open lexicon and closed lexicon. An open lexicon is often applied to historical documents, but it can have broad applications, since it allows the user to search for any word within a document. On the other hand, a closed lexicon is more suitable for domain specific applications, and can be either static or dynamic. Whereas static lexicons such as those for commercial or medical applications have a fixed number of words, dynamic lexicons have large and varying vocabularies. A closed lexicon for a handwritten Arabic commercial application will be used to validate our word spotting system.

Words in Arabic handwritten text have no clear boundaries, and the Arabic script is horizontal, cursive and more unconstrained than other scripts. PAWs are complete connected components that can be easily extracted. Thus, many Arabic word spotting systems tend to spot PAWs rather than words. However, the development of a complete word spotting system that can search for words in the Arabic text, still requires more attention and research.

The matching techniques of Hidden Markov Models (HMM) and Dynamic Time Warping (DTW) are frequently used to spot words. Meanwhile, learning based word spotting system can successfully adapt to multi-writer word spotting systems. Some holistic classifiers such as Support Vector Machines (SVM), Modified Quadratic Dis-

criminant Function (MQDF) and Regularized Discriminant Analysis (RDA) can have high classification accuracies, in addition to their abilities to discriminate between classes; however they have rarely been used for word spotting.

Given all the above considerations, we propose a learning-based word spotting system that is based on PAWs rather than words. For the first time, language models will be used in conjunction with holistic classifiers to spot Arabic handwritten words. Thus, the outcome of the thesis will be a work that attempts to overcome the difficulty of finding the boundaries of Arabic handwritten word within documents.

Finally, different word spotting systems have been proposed, while reducing the number of the false positive may significantly increase the performance of a word spotting system. Thus, we investigated and applied a post-processing technique based on combining two classifiers of different nature to reduce these false positives .

1.3 Contributions

In this thesis we present a coherent learning-based word spotting system for multi-writer Arabic handwritten script. This system aims to solve the lack of boundaries problem which appears in Arabic handwriting, so that complete Arabic handwritten words can be spotted. Because the PAW model has shown promising results in Arabic handwritten word spotting, we propose a word spotting system that integrates the partial segmentation into PAWs, with PAW language models to spot words. Thus, an input document is first segmented into text lines, each of which is then segmented into PAWs. These PAWs are recognized using a hierarchical classifier, and then language models are used to reconstruct words from PAWs. Figure 1-1 summarizes the proposed systems.

A robust method for handwritten text line extraction is proposed. Morphological dilation with a dynamic adaptive mask for line extraction are used, while line separation occurs because of the repulsion and attraction between connected components. The characteristics of the Arabic script are considered to ensure a high performance of the algorithm. This algorithm was evaluated and it outperformed
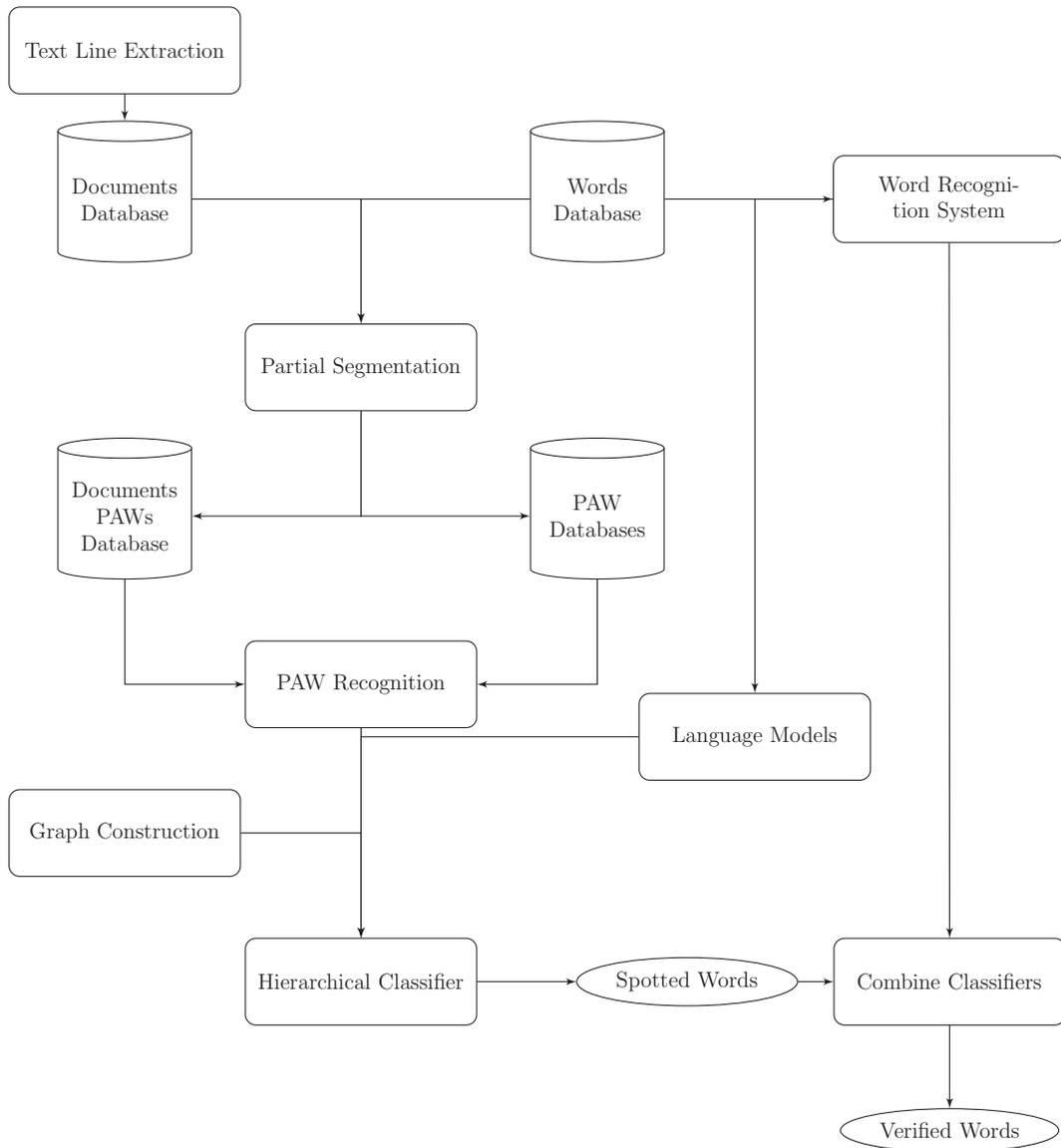
Figure 1-1: A complete overview of the proposed systems

other state-of-the-art algorithms for text line extraction.

Different feature extraction methods are implemented and experimented on Arabic handwritten words. In addition, different promising classifiers are used to construct an Arabic word recognition system. Experiments have been conducted to evaluate the performance and the effectiveness of these systems on different Arabic handwritten words databases.

A two-pass partial segmentation algorithm is proposed to segment word images or text lines into PAWs. This algorithm applies heuristics based on the characteristics of Arabic handwriting. The algorithm was evaluated and promising results were produced.

In this thesis we propose an effective method to spot words from Arabic handwritten documents. This method takes into consideration the fact that Arabic handwriting consists of PAWs. Consequently, PAWs form the basic components of this search process, and a hierarchical classifier (consisting of a set of classifiers each trained on a different part of the input pattern) is implemented. For the first time in Arabic word spotting, language models are incorporated into the process of reconstructing words from PAWs. The method was tested and promising results have been achieved.

Finally, we propose three verification methods for Arabic word spotting systems based on a holistic approach. The first method is based on matching the results of the word spotting system with those of the holistic classifier, while the other two methods derive new score evaluation criteria based on the results of an analytical word spotting classifier and a holistic word recognition classifier. The results show that verifying a word spotting system using these methods can significantly improve the performance of the system.

1.4   Thesis Overview

This thesis is organized as follows: Chapter 2 discusses word spotting approaches, describes the characteristics of the Arabic handwritten text, and reviews the work that has been done on Arabic handwritten word spotting.

Chapter 3 contains a brief overview of Arabic text lines extraction, explains the proposed algorithm for segmenting an Arabic unconstrained handwritten document into text lines, presents the experiments and results to evaluate the proposed algorithm, followed by a conclusion.

Chapter 4 presents a complete word recognition system, discusses the preprocessing methods and presents three sets of features that are different in nature. This chapter also describes different classification methods that have been proposed namely, Support Vector Machines, Modified Quadratic Discriminant Function and Regularized Discriminant Analysis. Experiments and results are shown together with a comparison with other Arabic word recognition systems.

Chapter 5 defines Pieces of Arabic Words (PAWs), and then presents in detail the proposed algorithm to partially segments Arabic handwritten words or text lines into PAWs. Experiments on the proposed partial segmentation algorithm are also presented.

Chapter 6 describes in detail the proposed hierarchical classifier. This classifier is proposed to overcome the lack of boundaries problem which appears in Arabic handwriting, so that Arabic handwritten words can be spotted. Different experiments have been conducted using the hierarchical classifier to spot Arabic handwritten words; these experiments are presented together with the results.

Chapter 7 describes an ensemble classifier which combines two different paradigms to verify the resulting words of a word spotting system. Two classifiers to spot Arabic handwritten words are introduces, and then three different verification models are proposed. These are followed by experiments and their results.

Finally, we conclude with some observations and directions for future work in Chapter 8.

Chapter 2

Literature Review

Word spotting is an efficient approach for document retrieval, with the result that many studies favor word spotting over word recognition to retrieve words from documents. Most of these studies have addressed word spotting in documents based on Latin or Chinese scripts, while the cursive nature of Arabic script makes Arabic handwritten word spotting more challenging. Different proposed Arabic word spotting systems are presented in this chapter.

This chapter defines word spotting in section 2.1.1, and discusses different approaches of word spotting in section 2.1.3. The performance measures of word spotting systems are described in section 2.1.4, the characteristics of Arabic handwriting are described in section 2.2.1, and finally a detailed literature review on word spotting for Arabic handwriting is presented in section 2.2.2.

2.1   Word Spotting

2.1.1   Definition

Handwritten word spotting, also called indexing or searching within documents, is the task of detecting keywords from documents by segmenting the document into word images (clusters) based on their visual appearance. Word spotting systems aim to recognize all occurrences of the specific keyword within a document. The input

```
                    ┌─────────────────┐
                    │  Word Spotting  │
                    │   Approaches    │
                    └─────────────────┘
                             │
           ┌─────────────────┼─────────────────┐
           ▼                 ▼                 ▼
  ┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐
  │ Similarity      │ │ Shape Code      │ │ Learning Based  │
  │ Matching        │ │ Matching        │ │                 │
  └─────────────────┘ └─────────────────┘ └─────────────────┘
```
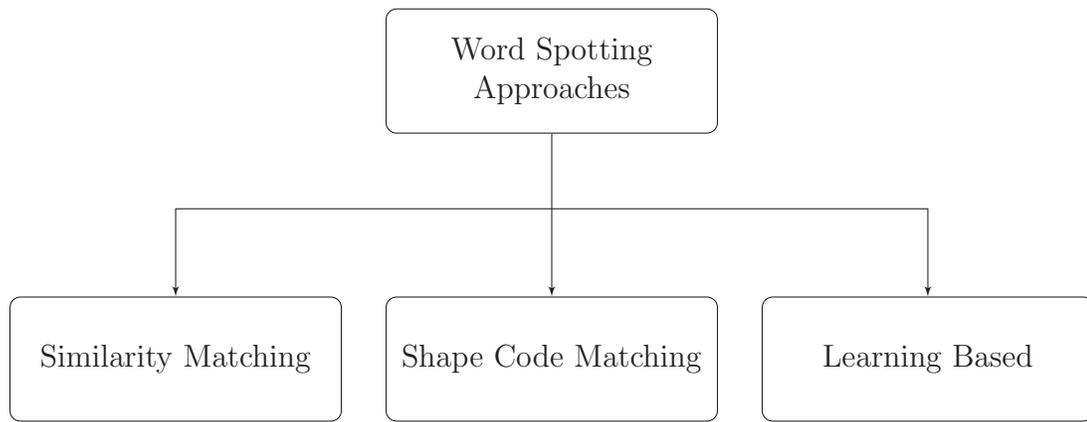
Figure 2-1: Word Spotting Approaches

to the word spotting system is a keyword query, which can be either query by string or query by example. Query by string is a string of letters entered on the keyboard, while query by example uses an image of a word. Initially, most of the word spotting systems start by clustering documents into words. This can be done using different clustering techniques. Afterwards, the word can be described as a whole or it can be segmented into a set of components such as letters, strokes or graphemes. Finally, different algorithms and methods are used to spot words. These methods include learning-based, template matching, and shape code mapping. Figure 2-1 illustrates different word spotting approaches.

2.1.2   Input Queries

In word spotting systems, both query by string and query by example are used to input keywords. Each of these approaches has its pros and cons. Query by string requires learning the alphabet of the language, and then concatenating the letters to form the word model for later matching with the words in the document [9, 12, 13, 14]. These systems alleviate some of the drawbacks of traditional handwriting recognition systems, which require huge databases for training. These word spotting systems perform well for lexicon-free approaches [15], where there are no restrictions on the size of the lexicon.

On the other hand, for query by example, the pixel by pixel or the extracted

features of the template image are passed to the system, which is then detected in the document using word spotting techniques. These systems suffer from the drawback that they can be applied only on closed lexicons [16, 17, 18, 19].

2.1.3 Word Spotting Approaches

Segmenting or clustering the document into words is considered the first step in many word spotting systems. This can be done using state-of-the-art word segmentation techniques. Various techniques are proposed to establish a threshold for the gap distance between the words in the document, to decide if the gap is within or between words [16, 17, 20]. Other techniques apply vertical projections and profiles to the lines of the document to find optimal segmentation points, and the document can also be clustered into words using classifiers such as artificial neural networks [21]. However, Leydier et al. [19] found that it is impossible to achieve accurate line or word segmentation. Thus, many successful segmentation-free approaches have been proposed, in which classifiers integrate segmentation with recognition, such as Hidden Markov Models (HMM) [22] and recurrent neural networks [23].

Handwritten word spotting is a technique which detects words selected by the user in a document without any syntactic constraints [19]. Many methods are used in the literature to detect words. These methods are based on three approaches: template matching, shape code mapping and learning-based.

Similarity Matching methods are applied in many different studies to spot words. These methods have successful applications with systems of few writers and are also lexicon-free. These methods measure the similarity or dissimilarity between either the pixels of the images or the features that are extracted from the images. Manmatha et al. [2] proposed the first indexing or word spotting system for single writer historical documents. The proposed method was based on matching word pixels. Subsequently, different template matching approaches based on features extracted from word images have been proposed [5, 7, 18, 21]. Dynamic Time Warping (DTW) [3, 6, 16, 24] has been successfully applied as an efficient template matching algorithm based on dynamic programming.

Shape code mapping techniques use the character shape code in which each character is mapped into a shape code. Ascenders, descenders, loops and other structural descriptors are used to form the shape code. Each word forms a sequence of shape codes, and query words are mapped into word shape codes. Then, string matching algorithms can be applied to perform the mapping and detect words [8].

Learning based word spotting systems were introduced to adapt to muli-writers with promising results. However, sufficiently large databases are needed to train the system. HMM is the most common classifier applied to word spotting systems [13, 20, 25]. Other approaches have also been developed; for example, Frinken et al. [23] proposed a word spotting system that uses a bidirectional Long Short-Term Memory (LSTM) Neural Network together with the Connectionist Temporal Classification (CTC) Token Passing algorithm to spot words, and this system has shown high performance.

2.1.4 Performance Measure

To evaluate any system, some performance metrics are needed. There are two ways to measure the performance of a word spotting system, either viewing it from the correctly spotted samples or from the incorrectly spotted ones. In the former view, both the recall rate and the precision rate are determined and often the precision-recall curve is plotted to give a visual view on the performance of the system [20, 21]. The following formulas are used to measure the performance of a word spotting system.

Recall Rate (RR): measures the ratio of actual positives, or the successful retrieval of the relevant target sample,

$$RR = \frac{TP}{TP + FN} \tag{2.1}$$

TP (*True Positive*): total number of correctly spotted target samples,

FN (*False Negative*): total number of target samples which are not spotted,

Precision Rate (PR): the probability that the retrieved image is a target word,

$$PR = \frac{TP}{TP + FP} \qquad (2.2)$$

FP (*False Positive*): total number of spotted samples which are misrecognized.

The precision-recall curve is also used to calculate the Mean Average Precision (MAP) represented by the area under the curve, and the $R - Prec$ which gives the rate at which the recall and precision graphs intersect.

The other way of measuring the performance is adopted from spoken word spotting [13, 16]. This approach is based on the error rate where the following formulas are used.

Word Error Rate (WER): the proportion of the words that were not recovered exactly as they were in the manual transcript,

Out Of Vocabulary words (OOV): words that occur only in the testing pages and not in the training pages or words,

False Alarm Rate (FAR): an erroneous image target detection decision. The percentage of how many times the word was falsely spotted,

$$FAR = \frac{FP}{FP + TN} \qquad (2.3)$$

TN (*True Negative*): Total number of the OOV image that were not spotted.

2.2 Word Spotting in Arabic Handwritten Documents

2.2.1 Characteristics of Arabic Handwriting

Arabic script is always cursive even when printed, and it is written horizontally from right to left. In Arabic writing, letter shapes change depending on their location in the word. This fact distinguishes Arabic writing from many other languages. In addition, dots, diacratics, and ligatures are special characteristics of Arabic writing.

Figure 2-2 shows two Arabic handwritten documents.
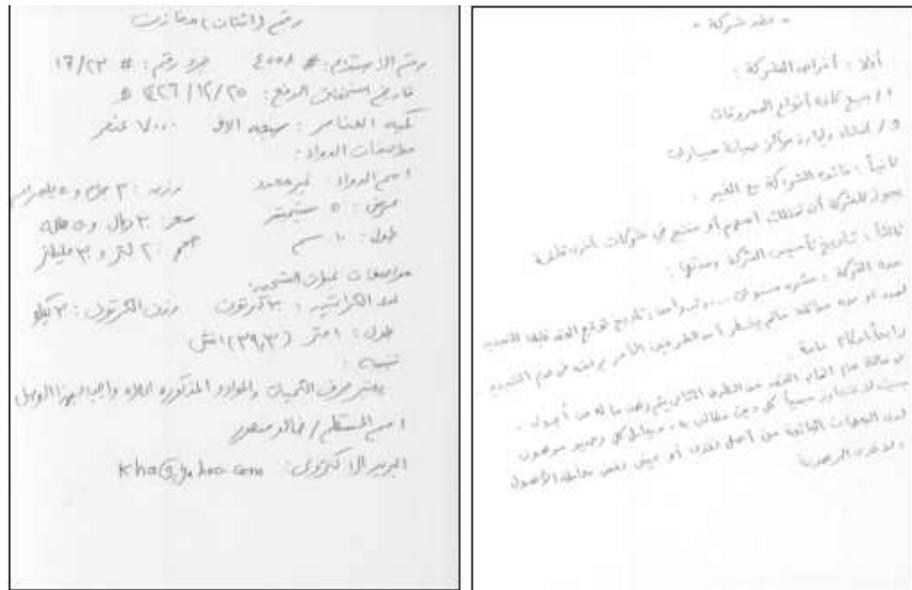
Figure 2-2: Two Arabic handwritten documents

The Arabic handwriting system evolved from a dialect of Aramaic which has fewer phonemes than Arabic. Aramaic uses only 15 letters but Arabic uses 28 letters. The letters in Arabic are formed by adding one, two or three dots above or below the Aramaic letters to generate different sounds [15]. Thus, many letters share a primary common shape and only differ in the number and/or location of dots. This means dots play an important role in Arabic writing and other languages that share the same letters such as Farsi (Persian) and Urdu. It is also worth mentioning that more than half of the Arabic letters (15 out of 28) are dotted. In printed documents, double and triple dots are printed as separate dots, while in handwritten documents there are different ways to write them.

In addition, shapes of letters change depending on their position in the word. Therefore, each Arabic letter has between two and four shapes. Letters can be isolated (28 letters), beginning (22 letters), middle (22 letters), and ending (28 letters). However, Arabic letters do not have upper and lower case. There are six letters in Arabic that are only connected from the right side; therefore, when they appear in the word they cause a disconnection resulting in sub-words or Pieces of Arabic Words

14

(PAWs). This fact makes word spotting and document segmentation into words more challenging.

Ligatures are used to connect Arabic letters, making it difficult to determine the boundaries of the letters, since ligatures are not added according to any writing rule. Ligatures in Arabic can only be found on the baseline because letters are only connected on the baseline, as opposed to Latin-based languages in which letters can be connected from the ascenders and descenders.

In Arabic words there are small markings called "diacritical markers"; these markers represent short vowels, double consonants and other marks [26] that are added to the letters. There are no Arabic letters with upper and lower diacritics together. Adding these diacritics to the Arabic script is not obligatory, so they are not always added.

2.2.2 Word Spotting in the Arabic Language

The naturally cursive structure of Arabic writing is more unconstrained than in other languages. This, coupled with the fact that the boundaries between words are arbitrary and often non-existing, makes word spotting in the Arabic language challenging problem in need of further research.

Attempts have been made to construct a language independent word spotting system, but these have encountered problems when handling Arabic script. Srihari and Ball [21] proposed a language independent word spotting system, in which they extracted gradient features from words since these features are language-independent. However, for Arabic handwritten word spotting, they found it necessary to apply manual word segmentation (clustering). In this way, they circumvent a main problem of the Arabic language — that there are no clear boundaries between words. Leydier et al. [19] proposed a segmentation-free language independent word spotting system which may overcome this problem. However, they faced difficulties with words from the same root. Even though the system was validated for Arabic using only one simple query consisting of a single PAW, the precision rate of 80.00% for Arabic was lower than that of the two Latin databases that were tested. Similarly, Wshah et

al. [11] proposed a script independent segmentation-free word spotting system based on HMMs, and this system was compared to a concurrent word spotting system [25] also utilizing HMMs. Both systems have found that the lowest results were obtained when applying the system on the Arabic language.

DTW has been extensively used for word matching in Arabic handwritten word spotting. Moghaddam and Cheriet [27] applied Euclidean distance enhanced by rotation, together with DTW, to measure the similarity between two connected components or PAWs of historical documents. Moreover, Self-Organizing Maps were used to initially cluster PAWs depending on the shape complexity of each PAW. Rodriguez-Serrano and Perronnin [28] proposed a model-based similarity measure between vector sequences. Each sequence is mapped to a semicontinuous Hidden Markov Model, and then a measure of similarity is computed between the HMMs. This computation of similarity was simplified using DTW. They applied the measure to handwritten word retrieval in three different datasets including the IFN/ENIT database of Arabic handwritten words (see Section 4.4.3), and concluded that their proposed similarity outperforms DTW and ordinary continuous HMMs. Saabni and Bronstein [29] implemented an Arabic word matching approach by extracting contour features from PAWs, then embedding each PAW into an Euclidean space to reduce the complexity; finally they used an Active-DTW [30] to determine the final matching result of a PAW.

Content-based retrieval using a codebook has been used for Arabic word spotting [31, 32, 8]. In these systems, meaningful features are extracted to represent codes of symbols, characters, or PAWs. Then similarity matching or distance measure algorithms between the codes and the codebook are applied to perform the final match.

Latin script is basically based on two models (character and word). However, Arabic script is based on three models: Character, PAW and Word models. The three models are used for Arabic word spotting, while the PAW model is extensively used, since a line of the Arabic text can be viewed as a sequence of PAWs instead of words; also there are no differences between the spaces separating PAWs and those

separating words. Nevertheless, a few segmentation-free systems have been proposed for Arabic handwritten word spotting, in which segmentation is embedded within the classification process. These systems are either implemented using HMMs based on the character model [11], or an over-segmentation is applied based on the PAW model [10].

Attempting to segment Arabic documents into candidate words may not be an appropriate approach for Arabic word spotting systems. This is because Arabic words are composed of PAWs that are easy to extract, while there are no clear boundaries between words. This latter aspect would introduce difficulties in segmenting a document into words. Srihari et al. [33] tried to cluster words by segmenting the line into connected components and merging each main component with its diacritics. Nine features were extracted from each pair of clusters and the features were passed to a neural network to decide whether the gap between the pairs is a word gap. However, with ten writers each writing ten documents, the overall performance was only 60% when the word segmentations were correct, and this significantly affected the spotting results.

Many studies favored segmenting documents into PAWs rather than words due to the problem of not having clear boundaries for words. Sari and Kefali [8] preferred to segment the document into major connected components, to circumvent the problem of word segmentation in Arabic documents. Thus, they decided to favor Arabic PAWs processing instead of words. They converted the PAW into Word Shape Tokens (WST) and represented each PAW by global structural features such as loops, ascenders and descenders. Similarly, input queries were coded and then a string matching technique was applied. They validated their word spotting system using both printed and handwritten Arabic manuscripts and historical documents. This approach is promising because it uses open lexicons and avoids pre-clustering. Saabni and El-Sana [9] also segmented the documents into PAWs; they used DTW and HMM for matching in two different systems, and then additional strokes were used by means of a rule-based system to determine the final match. Similarly, Khayyat et al. [34] proposed a learning-based word spotting system for Arabic handwritten documents;

17

this system has also favored the PAW model, in which words are spotted using a hierarchical classifier where PAWs are recognized, and then words are re-constructed from their PAWs. Language models are incorporated into this system to present the contextual information.

In Arabic, word spotting using an analytical approach to segment words into letters is challenging due to several reasons. Firstly, the Arabic language has 28 letters but each letter has a different shape (form) depending on its location within a word. This results in more than 100 shapes of letters, many of which are extremely similar and only differ in the number or location of the dots. Secondly, writers may elongate ligatures and letters in order to highlight a keyword or for aesthetic reasons. Thirdly, vertical overlapping between letters often occurs. Finally, in Arabic there are many writing styles in which a letter in the same position of a word can be written in different ways. These facts make segmenting a document into words challenging. Toufik et al. [35] proposed an analytical approach for handwritten Arabic letter segmentation. They extracted some structural features that occur in Arabic letters such as holes, turning points, double local minima, ascenders, descenders, and one, two and three dots. They applied their segmentation algorithm to an omni-scriptor database, and the results show that 5% of the characters were under-segmented, 9% of the characters were over-segmented and 86% of the characters were well segmented.

Attempting to spot words after segmenting them into letters, PAWs or words may increase the error rate, due to segmentation errors. Ball et al. [36] over-segmented the words hoping not to have more than one letter in a segment, then a dynamic programming algorithm was applied to find the candidate letters. However, because of the difficulties in segmentation, a segmentation-free approach can be applied to spot Arabic words [11]; this approach has shown promising results in Latin handwritten word spotting.

The recall and precision rates of the some Arabic handwritten word spotting systems are summarized in Tables 2.1, 2.2, 2.3 and 2.4.

#	Author/s	Data	Model	Features	Method	Recall	Precision
1	Khayyat et al. [34]	Multiwriter - 137 documents - CENPARMI Arabic handwritten documents database	PAW	Gradient Features [37]	Hierarchical classifier with pruning (SVM)	50.00%	84.56%
2					(RDA)	50.00%	66.04%
3	Khayyat et al. [38]	Multiwriter - 47 documents - Subset of CENPARMI Arabic handwritten documents database	Combination of PAW and Word models	Gradient Features	ensemble of classifiers - SVM	50.00%	84.4%
4	Cheriet and Moghaddam [10]	Single writer - Historical Documents	PAW	Topological and geometric features extracted from the CC's of skeleton	Similarity Matching	54.29%	71.26%
5	Wshah et al. [11]	Multiwriter - AMA Arabic dataset	Segmentation-free (Character)	Sequence of overlapped windows, with gradient and intensity features	HMM	50.00%	60.0%

Table 2.1: Recall and Precision rates on Arabic handwriting spotting (1)

#	Author/s	Data	Model	Features	Method	Recall	Precision
6	Khayyat et al. [39]	Multiwriter - 47 documents - Part of CENPARMI Arabic handwritten documents database	PAW	Gradient Features	Hierarchical classifier - SVM	53.00%	65.0%
7	Saabni and	Different writers - 5 documents - 10 Pages	Word-Part (PAW)	Structural features that captures local, semi-global and global behaviors [40]	HMM		74.00%
8	El-Sana [9]				DTW		86.00%
9	Toufik Sari and Abderrahmane Kefali [8]	132 Pages of Arabic manuscripts different sources covering different fields and diverse queries	Word shape token (WST)	Diacritics, Descenders, Ascenders, and Loops	Searching for consecutive WSTs within the document	56.62%	77.78%
10	Leydier et al. [19]	Arabic Manuscript - Single writer - One query of one PAW	Segmentation Free	Gradient Features	Cohesive Elastic Matching	80.00%	80.00%

Table 2.2: Recall and Precision rates on Arabic handwriting spotting (2)

Author/s	Data	Model	Features	Method	Recall	Precision
11 Shahab et al. [32]	60 pages - single writer	PAW (subword) using code book	concentric circles features and width features	Similarity measure using Angular Separation		72.5%
12			DFT of subword profiles - Angular lines features, and height features	Angular Separation		72.5%
13			DFT of subword profiles - Angular lines features, and concentric circles features	Product of Manhattan Distance		72.5%
14 Chan et al. [15]	20 pages of a 12th Century document	Segmentation based (Character)	Modeling Ink	gHMM	50.00%	24.70%

Table 2.3: Recall and Precision rates on Arabic handwriting spotting (3)

	Author/s	Data	Model	Features	Method	Recall	Precision
15	Ball et al. [36]	CEDARABIC (10 writers)	Segmentation based (Character)	Word Model Features	Minimum Euclidean distance then Dynamic Programming	50.00	28.00%
16			Segmentation Free (Word)		Sliding Window - scoring - Filtering	50.00%	34.00%
17			Manual Segmentation (Word)			50.00%	65.00%
18	Sargur Srihari et al, [33]	CEDARABIC (8-writers)	Manual Segmentation	Global Word Shape features- 1024 binary features	Word shape matching -similarity measure	50.00%	70.00%

Table 2.4: Recall and Precision rates on Arabic handwriting spotting (4)

Chapter 3

Text Line Segmentation

Text line extraction is a crucial preprocessing step for document analysis and recognition applications. Compared to printed documents, line extraction in handwritten documents is more challenging because of irregular spacing between lines, curved and multi-skewed lines, varying skew within the same line, touching and overlapping lines.

Arabic handwritten script is naturally cursive, unconstrained and horizontal. This makes the extraction of Arabic handwritten lines challenging. Many script independent line extraction methods have been proposed in the literature. However, Arabic handwritten text lines extraction algorithms have either not been evaluated or have reported higher error rates than other languages. This is because Arabic script is more unconstrained than other scripts. Furthermore, the Arabic script consists of small connected components called diacritics. These diacritics are usually located above or below the major connected components of the scripts. In handwriting, these diacritics are often located between the text lines. Accordingly, they are often misplaced in script independent text line extraction methods.

This chapter briefly discusses some methods for Arabic text line extraction in section 3.1, and then presents our method for Arabic handwritten text line extraction, using a dynamic adaptive mask [41]. This method is explained in section 3.2, experiments and results are presented in section 3.3, and a conclusion is drawn in section 3.4.

3.1 Arabic Handwritten Text Line Extraction

Numerous methods have been proposed to extract text lines from handwritten documents. These methods can be classified into the following six major categories: projection profile based, smearing methods, Hough transformation based, clustering or grouping methods, repulsive attractive methods that uses energy minimization systems, and stochastic methods which make use of stochastic learning algorithms.

Many methods have been proposed for Arabic handwritten text lines. Kumar et al. [42] proposed a graph based approach to extract text lines from Arabic unconstrained handwritten documents. This method is fast since it is based on connected components. However, it does not perform well in the presence of touching components. Shi et al. [43] extracted Arabic handwritten text lines by applying a direction filter; then an adaptive thresholding algorithm was applied to adaptive local connectivity maps to form connected components. Finally, a clustering algorithm was used to group connected components so that text lines are extracted.

Many script independent text line extraction algorithms have been proposed. Yin and Liu [44] presented a clustering method using Minimum Spanning Trees (MST) to extract lines from both Chinese and Latin-based documents. The results show that their method performs well on multi-skewed and curved text lines in handwritten documents. Bukhari et al. proposed [45] a script independent line extraction algorithm that uses ridges over smoothed images to estimate the central line of text lines parts. An active contour was applied over ridges to segment the lines. Li et al. [46] proposed a script independent algorithm which applies the level set to segment lines. These two approaches perform well on Arabic handwritten documents, but they suffer from the high computational cost.

Ziaratban and Faez [47] applied a bottom up algorithm to segment a document into adaptive blocks; after which the skew of each block is estimated. Three parameters were defined so that the method can adapt to different writers. Different techniques were combined to improve the results, also to adapt the method to scripts with special characteristics. Ouwayed et al. [48] implemented a text extraction system

using various local techniques including snakes (Repulsive Attractive Methods), to create a contour which segments the lines into local zones. Then the orientation of each zone was detected using special projection profile histograms.

3.2 Our Method for Arabic Text Line Extraction

Arabic handwritten script is a horizontal cursive script by nature. Based on this fact, we use a horizontal dynamic mask to perform appropriate smearing to separate Arabic text lines in a document. Algorithm 1 summarizes the flowchart of our method. The mask keeps adapting to the document to find the best mask size and shape to separate text lines. Moreover, this mask may have different shapes for different zones in the document. The power of this mask becomes apparent as it segments the document into big blobs that give the potential layout of the lines. The text within blobs repulse or attract depending on the characteristics of the Arabic script. Applying special techniques to disconnect touching lines as a preprocessing step is computationally expensive. Thus, touching components are detected and separated in an intermediate step within the algorithm, which is computationally more efficient. All thresholds utilized by the proposed algorithm are empirically chosen. Figure 3-1 illustrates the major steps of our method: (a) shows a preprocessed binarized document, (b) is a potentially smeared document, (c) is the document after the dynamic adaptive smearing, (d) shows the final smeared blobs, (e) is the final text line segmentation, and finally (f) illustrates the line separation step.

3.2.1 Document Analysis and Preprocessing

A simple preprocessing is applied to an input document, and then the properties of the document are learned. These properties are needed to tune the parameters and the thresholds of the text line segmentation algorithm.

A 3×3 Gaussian filter is applied to the document to remove noise, after which the document is binarized by applying the Otsu binarization algorithm [49].

Average height and width of the major connected components in the document are
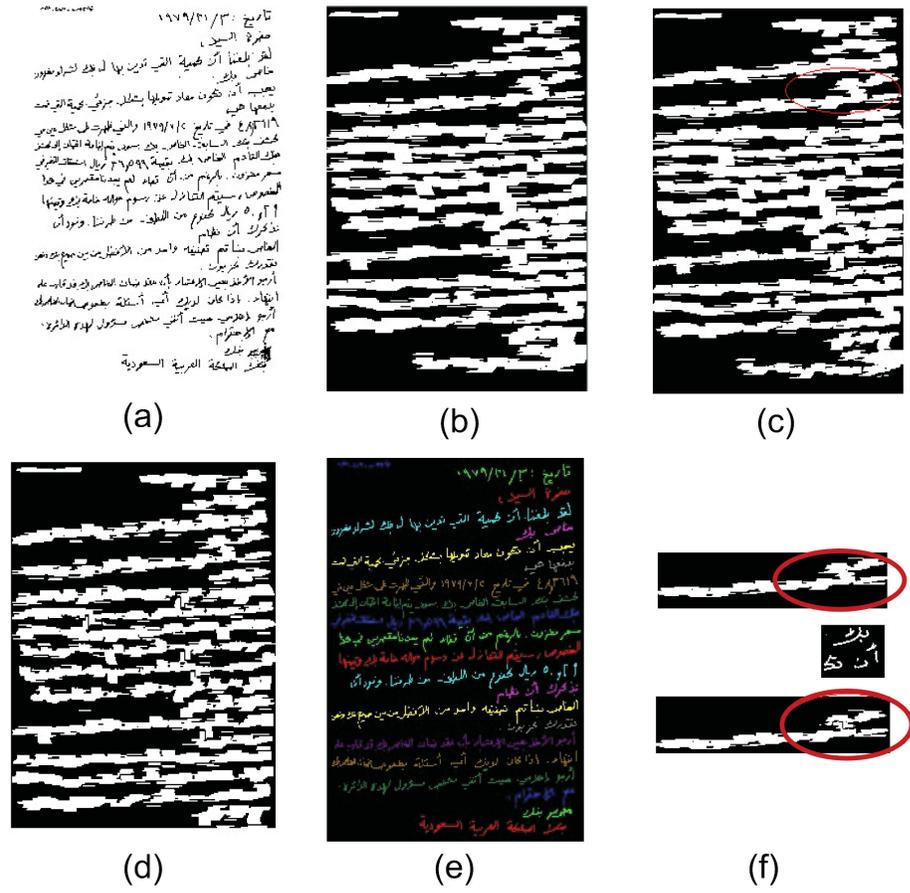
Figure 3-1: Steps of the text line segmentation algorithm.

then calculated. These parameters are needed to initialize the width of the dynamic mask wd. The horizontal projection profile $f(y, p(y))$ of the document is found, which reflects the nature of the document and the distribution of the text lines. From $f(y, p(y))$ the algorithm finds significant peaks and valleys, and then calculates the slope between each peak and its neighboring valley as follows

$$slope_{prof} = tan(\theta) = \frac{p(y_{peak}) - p(y_{valley})}{y_{peak} - y_{valley}} \qquad (3.1)$$

$p(y)$ is the number of white pixels in row y, while y_{peak} and y_{valley} are the coordinates of the line where the peak and the valley are detected.

Calculated slopes reflect the lines skew in different zones, and these slopes would determine the slope (shape) of the smearing mask within different zones.

Algorithm 1 Line Segmentation

$h_{avgc} \leftarrow$ Average height of major connected components
$w_{avgc} \leftarrow$ Average width of major connected components
$wd \leftarrow \frac{w_{avgc}^2}{h_{avgc}}$
repeat
  $blobs \leftarrow smearDocument(wd)$
  for all $blob_i \in blobs$ **do**
    if $h_{blob} > h_{blob-threshold}$ **then**
      $separateLines(blob)$
    end if
  end for
  $wd \leftarrow wd - 3$
until $\forall blob_i \in blobs \ (h_{blob} \leq h_{blob-threshold})$
blobs re-labeling
for all $blob_i \in blobs$ **do**
  if $blob_i$ is small **then**
    $mergeBlobs(blob_i, blob_j)$ where $blob_j$ is the closest big blob to $blob_i$
  end if
  if $checkKaf(blob_i)$ **then**
    $mergeBlobs(blob_i, blob_j)$ where $blob_j$ is the closest big blob to $blob_i$
  end if
  if $checkDiacritics(blob_i)$ **then**
    $mergeBlobs(blob_i, blob_j)$ where $blob_j$ is the closest big blob to $blob_i$
  end if
end for
for all $blob_i \in blobs$ **do**
  $doHorizontal(blob_i)$
end for

Figure 3-2 shows the horizontal projection profile of two different documents. The horizontal projection profile in Figure 3-2 (a) shows that the lines are not well separated and words are sparse all over the document. In addition, there are no deep valleys in many parts of the profile, which gives an indication that the lines are skewed and close to each other. The profile of Figure 3-2 (b) shows that the lines in the documents are nicely separated, since it has deep valleys.

3.2.2 Morphological Dilation and Dynamic Mask

Document A is dilated using a dynamic mask (structuring element) B to produce a new smeared document S. This document consists of a number of big connected
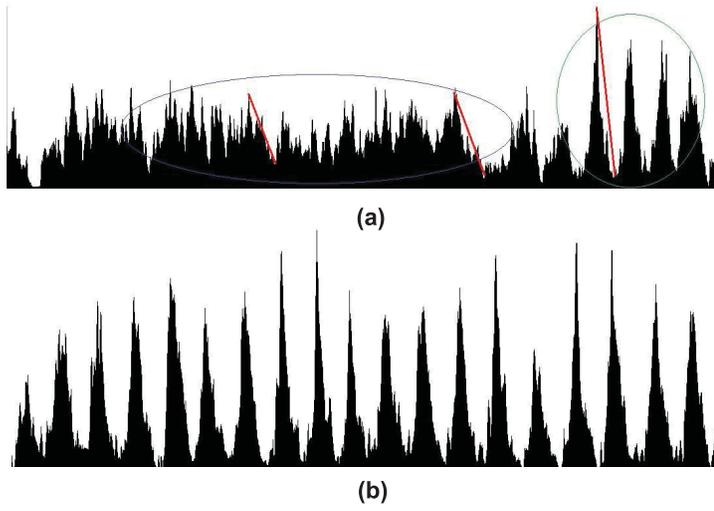
Figure 3-2: Projection profiles of two different documents.

components (blobs) as shown in Figure 3-1 (b), using the following formula:

$$S = A \oplus B = \{z | [(\hat{B})_z \cap A] \subseteq A\} \tag{3.2}$$

$\hat{B} = \{w | w = -b, b \in B\}$, and A, B and S are sets in Z^2.

Initially, the document is dilated with a binary mask B of only 1's with height $h_m = 1$ and width $w_m = \frac{w_{avgc}^2}{h_{avgc}}$, where w_{avgc} and h_{avgc} are the average width and height of the connected components of a document respectively.

The blobs of the smeared documents are analyzed to perform the suitable smearing to the document. If any blob has a height greater than $4 \times h_{avgb}$, where h_{avgb} is the average height of the potentially smeared document blobs, then the slopes between the peaks and valleys within a zone where the blob is located are calculated using Equation 3.1. If the absolute value of the calculated slope $tan(\theta)$ is greater than a predefined threshold, then the height of the mask h_m changes to a maximum of 3 pixels, and the width of the mask decreases by 3 pixels accordingly. The mask for that zone will be a slanted line, with a slope equal to that of the lowest blob in the zone. The slope of a blob is computed using the mean square method. Finally, the zone of that blob is re-smeared with the new mask. This step is repeated for all blobs until the document is smeared. Figure 3-1(b) and (c) show the potentially

smeared document and the smeared document after dynamically changing the shape, dimensions and inclination of the mask for each zone, respectively.

3.2.3 Splitting Lines

The actual height of a blob, which is the maximum number of white pixels in a column (see Figure 3-3), is found. If the height is greater than $2.5 \times h_{avgc}$, then the blob is passed to a recursive function for line separation.
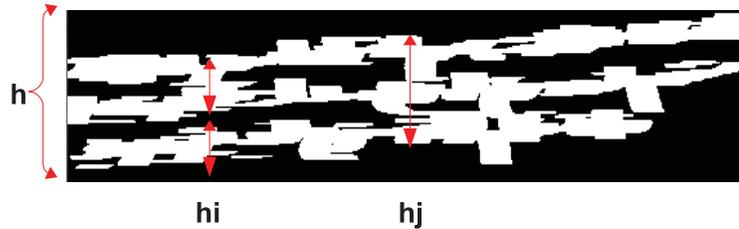
Figure 3-3: Actual blob height.

The line separation function $separateLines(blob)$ looks for a point (x, y), where x is the column with the maximum run of white pixels, while y is the row with the minimum horizontal projection around x. A block of size $3h_{avgc} \times 3w_{avgc}$ centralized at (x, y) is taken, and the connected components of the text in this block are extracted. This block is detected as shown in Figure 3-4 (a). The connected components closer to the upper bound of the given block are attracted to each other, and those below them are repulsed. This is performed by removing all pixels below the lower profiles of the upper connected components as shown in Figure 3-4 (b) and (c).

Components with height greater than 1.8 of the average height are considered two vertically touching components, and the algorithm disconnects them at the middle row that has the least number of white pixels. This is illustrated in Figure 3-4 and Figure 3-1 (f). The function keeps iterating recursively until the width of the blob is less than a predefined threshold. To avoid infinite loops or bad splitting, the function stops and returns a negative flag after seven iterations. The width of the mask is dynamically reduced by 3 pixels each iteration, and the document is re-smeared accordingly.
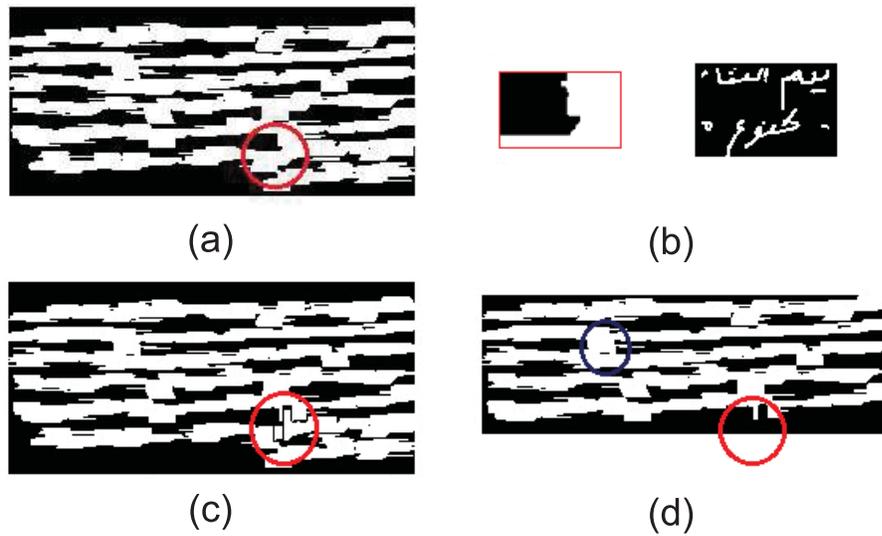
Figure 3-4: Line Separation Algorithm

3.2.4 Diacritics Affinity and Merging Horizontal Lines

In Arabic script, diacritics and dots are small connected components that are located above or below the words. Thus they are often located between text lines. The diacritics are sometimes not merged within the big blobs because of their locations. This results in having small blobs with a small height. The affinity of these diacritics will grow between the nearest big blobs, the Euclidian distances between a diacritic blob and its neighboring big blobs are used to find the nearest blob. Some diacritics are relatively wide, such as the diacritics of the "Kaf" in Arabic as shown in Figure 3-5. These diacritics may not be merged in this step. Accordingly, after passing this step the algorithm looks for the connected components of relatively sizeable width and distinguish between them and the words' parts. "Kaf" is always located above the words, so even if it appears closer to the upper blob, it is always merged to the blob below it. This step will result in merging all diacritics to the appropriate lines.

Some blobs from the same line may not touch because of the width of the mask. After merging the diacritics and separating the lines, the algorithm looks for blobs with left and right ends in the same horizontal region. The algorithm will connect those blobs horizontally and group them into one line, so that they will attract horizontally.

3.3 Evaluation and Experimental Results

The proposed algorithm was tested on the CENPARMI Arabic handwritten documents database (Section 3.3.1), which contains touching and unconstrained lines. The algorithm has successfully separated these touching components and interfering lines. Figure 3-5 shows some examples of separating touching and interfering lines using the proposed algorithm. The results show that the proposed algorithm outperformed the well known MST algorithm [44] that is based on connected components and it performed very well on Chinese and Latin-based scripts.

Figure 3-5: Examples for the performance of the proposed segmentation algorithm on touching and interfering lines.

3.3.1 Databases

The performance of the proposed method was evaluated using the CENPARMI Arabic unconstrained handwritten documents database. This database consists of 146 Arabic handwritten documents containing a total of 2,137 lines written by different writers. The documents contain multi-skewed and touching lines, and were digitized with a resolution of 300dpi.

3.3.2 Evaluation

The *precision*, *recall* and $f1_{score}$ (defined later in this sub-section) were used to evaluate the proposed text line segmentation method. An $M \times N$ confusion matrix

is found between the M ground truth lines and the N result lines. Given that g_i is a ground truth line, r_i is a result line, $P(x)$ is a black pixel in the line x, and $T(x)$ counts the number of pixels in line x, the matching score (MS) between result and ground truth documents is computed as follows

$$MS(r_i, g_i) = \frac{T(P(r_i) \cap P(g_i))}{T(P(r_i) \cup P(g_i))} \tag{3.3}$$

A confusion matrix was filled with the MS scores between lines. For each result line, if the score is above a predefined threshold then the line is considered as true positive TP. Result lines that are not matched are considered false positive FP. Finally, ground truth lines that are not matched are considered as false negative FN. The $precision$, $recall$ and $f1_{score}$ are computed as follows:

$$precision = \frac{TP}{TP + FP} \tag{3.4}$$

$$recall = \frac{TP}{TP + FN} \tag{3.5}$$

and

$$f1_{score} = \frac{2 \times precision \times recall}{precision + recall}. \tag{3.6}$$

3.3.3 Results

Table 3.1 shows the $precision$, $recall$ and $f1_{score}$ of the proposed algorithm and the MST algorithm [44] for line segmentation. The results show that our method outperformed the MST algorithm.

Table 3.2 compares our method with the Kumar et al. [42] method using the handwritten Arabic proximity database [1] with $MS = 0.95$. The methods produce similar results. However, the adaptive mask in our method introduces a new technique to identify a potential layout of the handwritten text lines. The results of our algorithm can be significantly improved by applying state of the art methods to disconnect touching components, and to separate the blobs from the critical regions detected by the algorithm. Moreover, training on some Arabic handwritten

Method	$Precision$	$Recall$	$F1_{score}$
MS		0.95	
Proposed Method	0.96319	0.967228	0.965421
MST	0.816784	0.871951	0.843466
MS		0.90	
Proposed Method	0.975746	0.979859	0.977799
MST	0.84051	0.89728	0.867967

Table 3.1: Experimental Results on CENPARMI Arabic Handwritten Documents Database.

documents to establish the thresholds may improve the results.

Method	$Precision$	$Recall$	$F1_{score}$
Proposed Method	0.90309	0.91536	0.909185
Kumar et al. [42]	0.9161	0.9017	0.909

Table 3.2: Experimental Results on Database [1] with $MS = 0.95$

3.4   Summary of Text Line Segmentation

We proposed a robust Arabic handwritten text line extraction algorithm that uses a dynamic mask, and is based on document smearing. Usually smearing does not perform well with overlapping and touching lines. However, the proposed dynamic mask and line splitting criterion, which depends on the attraction and repulsion of the connected components, overcame the aforementioned drawback and made the algorithm robust to touching and overlapping lines.

Moreover, our algorithm introduces a new way to identify a potential layout of the text lines and detect the critical regions to break up text into lines. Thus, different techniques can be proposed for text repulsion and attraction at these regions to improve the text line segmentation results.

Chapter 4

Arabic Handwriting Recognition

Handwriting recognition is the task of determining the identities of the handwritten letters, digits, symbols or words that are present in a digital image. The importance of handwriting recognition is increasing, since it can performs a major role in the automatic processing of document analysis tasks, such as document recognition and word spotting.

Different classifiers have been implemented for handwriting recognition including Multi-Layer Perceptron (MLP), Support Vector Machines (SVM) and Hidden Markov Models (HMM) [50, 51]. Arabic handwriting is receiving more attention in recent years, and many studies has been conducted to address this problem. Different classifiers have been also implemented for Arabic handwriting recognition systems such as SVM and MLP [52, 53], however most of the Arabic handwriting recognition systems are based on segmentation-free approaches such as Hidden Markov Model (HMM). Different features were experimented using HMM to recognize unconstrained Arabic handwriting [54, 55, 56, 57, 58, 59], also combinations of multiple classifiers have been implemented to improve the recognition rates of Arabic handwriting [60].

A complete word recognition system based on a discriminant classifier is presented in this chapter. Three feature sets are examined, in addition to three classifiers of different natures. This chapter is organized as follows: Section 4.1 explains the preprocessing steps of the proposed system. Sections 4.2 and 4.3 present the extracted features and the classification techniques respectively, and recognition results are

presented in section 4.4. The performance of our recognition system is compared
with other successful Arabic handwritten word recognition systems in Section 4.5,
and concluding remarks are contained in section 4.6

4.1   Preprocessing

A Gaussian filter is applied to the word images for noise removal, after which the
word images are binarized using the Otsu binarization algorithm [49], and then the
images are size normalized to 120×50 pixels. This normalization was chosen with
consideration for the nature of the Arabic script, which is horizontal and cursive.
Finally, a smoothing algorithm [61] is applied to the images.

4.2   Feature Extraction

Three sets of features were extracted from the word images: Gradient features [37],
Gabor filter features [52] and Frequency features using Discrete Fourier transform [13].
The feature vector sets have dimensions 400, 392, and 42 respectively. The procedures
of extracting local features of the first two sets are similar [62]. However, directional
features are extracted in the former, while a local narrow band pass filter with se-
lectivity to both orientation and spatial frequency is applied to the latter. Different
down-sampling and normalization techniques were applied in these two procedures.
We also implemented the discrete Fourier transform on three different profiles. In
our experiment, a recognition system based on Regularized Discriminant Analysis
(RDA) is used. Since this classifier would be more efficient with lower dimensionality,
a dimensionality reduction method is implemented for this classifier.

   The following sections describe the extracted features and the method used to
reduce the dimensionality.

4.2.1 Gradient Features

A mean filter is applied to an image for converting it to gray scale. The gray scale image is normalized so that the mean and maximum of the image intensities are 0 and 1 respectively. The Roberts filter is applied to each pixel $g(i, j)$ of the normalized image. Then the gradient is calculated as follows

$$Direction : \theta(i, j) = tan^{-1}(\frac{\Delta v}{\Delta u}) \tag{4.1}$$

$$Strength : f(i, j) = \sqrt{(\Delta u)^2 + (\Delta v)^2} \tag{4.2}$$

where

$$\Delta u = g(i + 1, j + 1) - g(i, j) \tag{4.3}$$

$$\Delta v = g(i + 1, j) - g(i, j + 1) \tag{4.4}$$

.

The direction of the gradient is quantized to 32 levels with $\pi/16$ intervals. Then the image is divided into 9×9 blocks, and a down-sampling is performed to reduce the dimensionality using a Gaussian filter. The directional resolution is reduced from 32 to 16 with a weight vector $[1\ 4\ 6\ 4\ 1]^T$. Finally, the variable transformation $(y = x^{0.4})$ is applied to make the distribution of the features Gaussian-like [37]. This results in a 400-dimensional feature vector (5 horizontal, 5 vertical, 16 directional resolution).

4.2.2 Gabor Filter Features

The Gabor filter is a harmonic linear filter composed of two parts. The 2-dimensional Gabor filter $h(x, y)$ is composed of a complex sinusoid called carrier $s(x, y)$, and a Gaussian function also called the Gaussian envelope $g(x, y)$, as shown in the following equation

$$h(x, y, \lambda, \theta) = s(x, y)g(x, y) \tag{4.5}$$

The complex carrier consists of two separate functions that are allocated in the real and the imaginary parts of a complex function. This carrier is the impulse response function and is defined as

$$s(x, y) = e^{j(2\pi(u_0 x + v_0 y) + P)} \tag{4.6}$$

with the following real and imaginary carriers representing an even-symmetric cosine component and an odd-symmetric sine component

$$real(s(x, y)) = cos(2\pi(u_0 x + v_0 y) + P) \tag{4.7}$$

$$imaginary(s(x, y)) = sin(2\pi(u_0 x + v_0 y) + P) \tag{4.8}$$

P is the phase of the sinusoid, (u_0, v_0) are the optimal spatial frequencies such that $u_0 = f_0 cos\theta$ and $v_0 = f_0 sin\theta$, where f_0 is the oscillation frequency, and $j = 0, 1, ...$ is the scale index (in equation 4.6).

The following is the Gaussian envelope

$$g(x, y) = \frac{1}{2\pi\sigma} e^{\frac{-(ax')^2 + (by')^2}{2\sigma^2}} \tag{4.9}$$

where $x' = xcos\theta + ysin\theta$, and $y' = -xsin\theta + ycos\theta$. This produces the following harmonic function

$$h(x, y, \lambda, \theta) = g(x, y)e^{j2\pi(u_0 x + v_0 y)} \tag{4.10}$$

σ can be related to λ, and $f_0 = 1/\lambda$. Applying the Gabor filter to an image $F(x, y)$ produces the image $I(x, y, \lambda, \theta)$ by the following convolution

$$I(x, y, \lambda, \theta) = \sum_m \sum_n F(m, n).h(x - m, y - n, \lambda, \theta) \tag{4.11}$$

$\theta = \frac{\pi k}{d}$ is the oscillation orientation, d is the number of directions (8 directions), and λ is the wavelength. After the convolution with each direction θ, the image I is divided into 7×7 blocks. For each block, the mean intensity is calculated and the

average intensity above the mean will be a feature. This will produce a feature vector of 392 features (8 directions $\times 7 \times 7$ blocks).

4.2.3 Fourier Features

Fourier features are extracted by generating three time series based on the projection profile, upper profile and lower profile. A projection profile is generated by counting the number of black pixels in each column. Upper and lower profiles are generated by calculating the distance from the word to the top and bottom of the word bounding box respectively. The Discrete Fourier Transform (DFT) is applied to the time series $f = f_0, ..., f_{n-1}$ to obtain the frequency space representation $F = F_0, ..., F_{n-1}$ using

$$F_k = \sum_{l=0}^{n-1} f_l e^{-2\pi lk/n}, 0 \le k \le n - 1. \tag{4.12}$$

From the DFT representation, the first seven real (cosine) components and imaginary (sine) components are extracted from each of the three profiles, resulting in a 42-dimensional feature vector. The seven components were determined as optimal using cross validation.

4.2.4 Dimensionality Reduction

In our system, a classifier based on Regularized Discriminant Analysis (RDA) is used. Since this classifier would be more efficient with lower dimensionality, we reduce the dimension of feature vectors from 400 to 80 by applying Principal Component Analysis (PCA) which is a powerful tool for dimensionality reduction [63].

Given a vector $x \in R^N$ with zero mean and covariance matrix Σ_x, then $y = \phi x$, where $y \in R^M$, $M < N$, is a linear transformation of the vector x to y. ϕ is the matrix of the orthonormal eigenvectors with the M highest eigenvalues in λ, called Principal Components, of the covariance matrix Σ_x. This results in reducing the dimensionality from N to M, while maximizing the variance in the lower dimensional space.

4.3 Classification

Three classifiers were used in this experiment, one of which is a discriminant classifier and the other two are statistical classifiers. These classifiers are Support Vector Machines (SVMs), Regularized Discriminant Analysis (RDA), and Modified Quadratic Discriminant Function (MQDF). The following sections briefly describe the three classifiers.

4.3.1 Support Vector Machines - SVM

Support Vector Machines [64] are useful for learning and classifying data. An SVM maps input vectors into a high dimensional feature space Z through a non-linear mapping. The target of this classifier is to find the optimal hyperplane for separable classes, which is a linear discriminate function with maximal margin between the vectors of the two classes. To construct such a hyperplane, only small numbers of the training data that can determine this margin should be taken into account. These training data are called *support vectors*.

Given a set of labeled training patterns $(y_1, x_1), ..., (y_l, x_l)$, where $y_i \in \{-1, 1\}$ and $x_i \in R^n$, SVMs tend to solve the following optimization problem

$$\underset{w,b,\xi}{\text{minimize}} \quad \frac{1}{2}||\mathbf{w}||^{\mathbf{2}} + C \sum_{i=1}^{l} \xi_i$$

$$\text{subject to} \quad y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

(4.13)

The function ϕ maps the training vectors $\mathbf{x}_i$ to a higher dimensional space, ξ_i are slack variables that permit margin failure, and C is the parameter which trades off wide margins between classes for a small number of margin failures. The idea of constructing support vector machines comes from considering the general form of the dot-products:

$$\phi(u).\phi(v) = K(u, v)$$

$K(u, v)$ is called the kernel function. Many functions can be used as kernels for the SVM such as linear, polynomial, Radial Basis Function (RBF) and sigmoid function.

Our experiments were conducted using LibSVM[1] as classifier, and an RBF was chosen with kernel

$$K(x_i, x_j) = e^{(-\gamma \|x_i - x_j\|^2)} \tag{4.14}$$

where x_i and x_j are the support vector and testing data point respectively and γ is the kernel parameter. In this experiment, these parameters were optimally chosen by cross-validation via parallel grid search on the validation set. After this step, the validation set was added to the training set to form an expanded set for training the SVM.

4.3.2 Regularized Discriminant Analysis (RDA)

RDA [65] has been applied to implement a parametric classifier with Gaussian density function, and it uses a classification rule based on the normal distribution

$$f_k(\overline{X}) = (2\pi)^{-\frac{N}{2}} |\Sigma_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(\overline{X} - \mu_k)^T \Sigma_k^{-1}(\overline{X} - \mu_k)} \tag{4.15}$$

where $\overline{X} = (x_1, x_2, ..., x_N)$ is the feature vector of a sample with N features, μ_k and Σ_k are the class k $(1 \leq k \leq K)$ mean vector and covariance matrix respectively, and K is the number of classes.

The RDA classifier is an improvement over the Quadratic Discriminant Function (QDF) classifier, which is also a parametric Gaussian classifier. When QDF is applied, the discriminative score of the kth class is calculated as follow

$$d_k(\overline{X}) = \sum_{i=1}^{N} \frac{[\phi_{ik}^T(\overline{X} - \mu_k)]^2}{\lambda_{ik}} + \sum_{i=1}^{N} \ln \lambda_{ik} - 2 \ln \pi_k \tag{4.16}$$

where π_k is the unconditional probability of class k, λ_{ik} and ϕ_{ik} denote respectively the eigenvalues of class w_k and the corresponding eigenvectors for $i = 1, 2, ..., N$. Eq. 4.16 shows that the discriminative score is heavily weighted by the smallest eigenvalues

---

[1]Chih-Chung Chang and Chih-Jen Lin, LIBSVM: a library for support vector machines, 2001, http://www.csie.ntu.edu.tw/ cjlin/libsvm

and the directions associated with the corresponding eigenvectors.

RDA smooths the covariance matrix of each class with the pooled covariance matrix and the identity matrix I multiplied by its average eigenvalues, which eliminates the bias toward commonality. Thus, RDA applies the following formula to calculate the smoothed regularized covariance matrix

$$\hat{\Sigma}_i(\beta, \gamma) = (1 - \gamma)[(1 - \beta)\Sigma_i + \beta\Sigma_0] + \gamma\hat{\Sigma}_i I \qquad (4.17)$$

where Σ_0 the pooled average covariance matrix, $\hat{\Sigma}_i = \frac{1}{N}trace(\Sigma_i)$, while β and γ in $[0, 1]$ are regularization parameters that control shrinkage towards a multiple of the identity matrix. RDA assigns a test sample x to the class $\hat{k}$ of minimum distance $d_{\hat{k}}(x)$ to x.

The probability density function of RDA outputs discriminant scores that represent distances between samples and classes densities. These discriminant scores cannot be used as confidence values or an approximation to a posteriori probabilities. Accordingly, a confidence transformation is applied to the output of the RDA classifier [66]. This confidence transformation method combines scaling and activation functions.

A global normalization is applied to the output of the classifier using

$$z_k(x) = \frac{d_k(x) - \mu_0}{\sigma_0} \qquad (4.18)$$

where μ_0 and σ_0 are the mean and the variance of the classifier output. Then $z_k(x)$ is scaled using the Gaussian function

$$f_k(x) = w_k z_k(x) - b_k \qquad (4.19)$$

where w_k and b_k are the weight and bias of the kth class respectively. These values are calculated from the stochastic gradient descent algorithm [67] used by artificial neural networks to estimate probabilities.

Finally, a sigmoid measure is used to approximate the likelihood of the class

posterior probability

$$P^{sg}(w_k, x) = \frac{1}{1 + e^{f_k(x)}} \tag{4.20}$$

4.3.3   Modified Quadratic Discriminant Function - MQDF

The MQDF improves the performance of the QDF (explained in section 4.3.2) by reducing the complexity and improving the generalization performance. This is achieved by replacing the minor eigenvalues λ_{ji}, $j > t$ (Equation 4.16) with a large constant δ_i. The MQDF discriminant score is then calculated as follows

$$d_k(\overline{X}) = \sum_{i=1}^{t} \frac{[\phi_{ik}^T(\overline{X} - \mu_k)]^2}{\lambda_{ik}} + \sum_{i=t+1}^{N} \frac{[\phi_{ik}^T(\overline{X} - \mu_k)]^2}{\delta_i} + \sum_{i=1}^{t} [\log \lambda_{ik} + (N-t) \log \delta_i] \tag{4.21}$$

This modification reduced the bias of minor eigenvalues. Also because the projections to minor axes are not computed, the computational complexity is significantly reduced, as well as the storage requirements.

The confidence transformation in Equations 4.18, 4.19 and 4.20 is similarly applied to $d_k(\overline{X})$.

4.4   Experiments and Results

Several experiments were performed on the CENPARMI Arabic handwritten words database (Section 4.4.2), to evaluate the performance of three sets of features: Gradient, Gabor and Fourier features. The performances of different classifiers were also evaluated and reported in Section 4.4.4 based on experiments performed on two Arabic handwritten words databases. These databases are different in nature as described in the following section.

4.4.1   Databases

Two databases were used to evaluate our word recognition system, namely CENPARMI Arabic handwritten words database (Section 4.4.2) and IFN/ENIT database

(Section 4.4.3). The former database has a small lexicon of 69 word classes, while the latter has a relatively large lexicon of 937 word classes.

4.4.2   CENPARMI Arabic Handwritten Words Database

CENPARMI has developed a database for Arabic off-line handwriting recognition in 2008 (Alamri et al. [68]). The database contains Arabic handwritten words. The data were collected from 328 participants in Canada and Saudi Arabia. The participants were Arabic writers of different nationalities, ages, genders and educational levels.

The isolated words database contains 69 Arabic words that have not been studied before. These include some commercial terms, together with words used in weights, measurements, and currencies of Saudi Arabia. Each word class has a ground truth data file that contains information about each sample: image name, content, number of connected components, writer number, age, gender and hand-orientation. This database contains 17007 and 4233 training and testing samples respectively.

4.4.3   IFN/ENIT Database

The Institute of Communications Technology (IFN) and the École Nationale d'Ingénieurs de Tunis (ENIT) have developed, the advanced Arabic handwritten words database (IFN/ENIT)[69]. This database was collected using over 2200 form-pages of (937) Tunisian town/village names. The database was written by 411 writers and contains 26000 Arabic words (Tunisian town/village names). The data consist of 5 sets (a - e); the samples are distributed almost equally among the sets, where every set contains about 6000 samples. To evaluate our system sets a, b , c and d are used for training and set e is used for testing. Among the data about 56 words are badly written, and these words are distributed among the sets. In this database some town/village names rarely appear, while some of them appear only once.

4.4.4 Results

Table 4.1 shows the recognition results obtained by the SVM classifier that was trained on three sets of features extracted from word samples of CENPARMI database. The SVM was chosen to examine the features sets, since discriminant classifiers are known to yield high recognition rates if trained with sufficient numbers of samples, and this is the case for CENPARMI Arabic handwritten words database. The results show that applying gradient features outperforms the other two sets of features and achieves a recognition rate of 96.51%.

Features	Recognition Rate (%)
Gradient Features	96.51
Gabor Features	88.52
Fourier Features	85.35

Table 4.1: Experimental results on CENPARMI database using different sets of features

Since gradient features have shown promising recognition rates when applied to CENPARMI Arabic handwritten words database, these features were chosen to evaluate the performance of SVM, RDA and MQDF classifiers on CENPARMI database. Table 4.2 shows results of the three classifiers with recognition rates higher than 96.5% when trained on CENPARMI database.

The classifiers used in this experiment are based on the holistic approach (Word Model), where MQDF and RDA are statistical classifiers and SVM is a discriminant classifier. The performances of the three classifiers were compared with those of another two classifiers of different natures. These classifiers are segmentation-free and are considered global approaches (based on Character Model): Hidden Markov Model (HMM) [70] and Bidirectional Long Short Term Memory Neural Networks (BLSTM-NN) [50]. Seven structural features [70] were extracted from each column of a word image when HMM and BLSTM-NN were used.

Table 4.2 shows the results of applying the aforementioned five classifiers to both CENPARMI and IFN/ENIT (sets (a-d) for training and set e for testing) databases.

Word based approaches have higher recognition rates than segmentation-free classi-
fiers on CENPARMI Arabic handwritten words which has a relatively small lexicon
(69 word classes). On the other hand, BLSTM-NN and HMM which are segmentation-
free classifiers outperformed the other classifiers when applied to IFN/ENIT database
which has a large lexicon (937 word classes). The discriminant classifier SVM has
poor performance on IFN/ENIT database when tested on set (e) and trained on the
other sets, since many classes have insufficient number of samples (i.e. one or two
training samples), while the statistical classifiers have better performances on this
database with over 14% increase in the recognition rate; the reason is that statistical
classifiers are more stable with respect to the training samples size. The strength
of the holistic approach appears in fixed and static lexicon scenarios with enough
training samples. However, with large and dynamic lexicons the ability of the holistic
classifiers to distinguish between classes is diminished [71]. This is the main reason
for the poor performance of the holistic classifier on the IFN/ENIT database. Nev-
ertheless, the BLSTM-NN classifier produces very promising results regardless of the
size of the database.

Classifier	Recognition Rate %	
	CENPARMI	IFN/ENIT
SVM	96.51	38.41
RDA	96.77	52.22
MQDF	96.89	55.54
HMM	88.61	73.45
BLSTM-NN	96.13	91.39

Table 4.2: Recognition results using different classifiers

4.5 Comparison of Results on Arabic Word Recog-
nition

The three classifiers used in this system are considered holistic classifiers, and are
compared with other classifiers that were implemented for Arabic word recognition.
To enable consistent performance evaluation, the classifiers used in our word recog-

nition system were evaluated using a four-subset version of IFN/ENIT in which sets a, b, and c are used for training, while set d is used for testing. This is in accordance with the practice adopted by all other systems to which our systems are compared.

Table 4.3 shows the recognition rates of the three classifiers used for our word recognition systems, in addition to some successful Arabic word recognition systems applied on the Arabic handwritten words database IFN/ENIT. In fact, all of these Arabic word recognition systems are based on HMMs. Pechwitz and Märgner [72] have extracted two sets of features and used them to train and test characters HMMs. These sets are gradient features and baseline dependent features. Two methods have been used to estimate the baseline for the latter features set: projection and skeleton, where the method based on the skeleton has better performance. Dreuw et al. [58] proposed one of the most promising HMM based Arabic word recognition systems, they examined different models of the white-space, where the between PAWs white-space model has the best performance.

Author	Method	Classifier	Recognition Rate %
Alkhateeb et al. [73]	Overlapping sliding window	HMMs	86.73
Al-Hajj et al. [74]	sliding window	HMMs	87.60
Benouareth et al. [75]	un-uniform sliding window	semi-continuous HMMs	83.79
Dreuw et al. [58]	Overlapping sliding window	HMMs	90.71
Pechwitz and Märgner [72]	Baseline projection Baseline skeleton	HMMs	81.84 83.56
Current Systems	gradient features	SVM RDA **MQDF**	80.98 84.08 **86.49**

Table 4.3: Experimental results on IFN/ENIT database. Classifiers are trained on sets (a - c) and tested on set d

Table 4.3 compares our word recognition systems with other Arabic handwritten word recognition systems. The word recognition system implemented using MQDF classifier has promising results which are comparable to the other systems in the

literature. This shows that this holistic statistical classifier can cope with different sizes of classes and training data.

4.6   Conclusion

Three sets of features are tested on Arabic handwritten words database with a lexicon of 69 word classes. Gradient features produced more promising results than Gabor and Fourier features.

Discriminant classifiers based on a holistic approach out-performed segmentation-free classifiers when applied to Arabic handwritten words database of a small lexicon, while segmentation-free classifiers have better performances when applied to a database of a large lexicon. However, statistical classifiers such as MQDF have comparable and promising performances compared to those of HMM classifiers.

Chapter 5

Partial Segmentation of Arabic Handwritten Words into Pieces of Arabic Words

Many document analysis and word recognition systems tend to segment text lines or word images into characters or words. In Latin based languages, words are delimited by space which makes it easier to extract words to model a system (using word model) [21, 4, 3]. In the Chinese language words consist of one or more characters, so text lines are usually segmented into Chinese characters (using character model), after which words are reconstructed [76, 77, 78]. The Arabic script is cursive, and many letters differ only by the location or number of dots, which makes it difficult to segment text lines into letters. Meanwhile, the lack of boundaries problem that appears in the Arabic script introduces difficulties in segmenting a text line into words. Therefore, for the Arabic language many studies favoured segmenting a text line into PAWs (using PAW model) over letters or words [8, 9, 27], particularly since PAWs can be easily extracted from a text line.

This chapter defines the Pieces of Arabic Words (PAWs) in section 5.1, and then our method to segment lines into PAWs is explained in section 5.2. Section 5.3 reports on the experiments conducted and results, and conclusions are presented in section 5.4.

5.1 Pieces of Arabic Words

The Arabic script is written in a cursive style, under which letters are connected to form words. There are six letters in the Arabic alphabet that cause disconnections in the word resulting in PAWs. Each word in the Arabic script consists of one or more PAWs, each of which contains only one major connected component (CC) and some or no minor CCs. In the literature on Arabic handwriting recognition and document analysis, these minor CCs are often called diacritics and dots. Major and minor CCs can be distinguished by their size and location. However, due to some writing styles, it may be difficult to distinguish between them, and some writers may misplace minor CCs, which introduces additional difficulties to the segmentation process. Fig. 5-1 shows an Arabic word of three PAWs.

Figure 5-1: An Arabic word and its three PAWs. The heads of the arrows point to major connected components, while the rest are minor connected components.

PAWs play an important role in many Arabic handwritten document analysis and recognition applications. They have been used in lexicon reduction applications for Arabic handwriting, in which using the PAW model resulted in promising performance [59]. Consequently, many Arabic document recognition and retrieval systems tend to segment a text line into PAWs rather than words, to improve the performance of these systems [9, 32].

5.2 Segmentation into PAWs

This section an presents an automatic segmentation of Arabic handwritten words or text lines into PAWs using a two-pass partial segmentation algorithm. This algorithm applies heuristics (based on the height, width, number of pixels, and locations of CCs) to extract PAWs within the text line or word images.

Connected components are extracted from a word or a text line image using a one-pass algorithm [79](Chapter 3). A binary image is the input of the algorithm. The foreground pixels are negated by being assigned a value of -1 to indicate unprocessed pixels. The algorithm starts by searching for a pixel with a value of -1 to assign to it a unique label, and then all unprocessed foreground eight-connected neighboring pixels will be assigned the same label. The algorithm recursively continues until no pixel has a value of -1.

Given a text line or a word image T, let $\{cc\}$ be the sequence of all CCs extracted from T, such that for any two CCs cc_i and cc_{i+1}, cc_i is on the right of cc_{i+1}. Let $major(cc_i)$ denote that cc_i is a major component, and analogously for $minor(cc_i)$. Then $cc_i \rightarrow cc_j$ indicates that $minor(cc_i)$ belongs to $major(cc_j)$.

For each connected component cc_i, the total number of pixels within the component is denoted by $count(cc_i)$, while if there are connected components below or above cc_i (and vertically overlapping with cc_i), then these would be denoted by cc_{blw} and cc_{abv} respectively. If there are no components above or below the component cc_i, then it is considered a major connected component ($major(cc_i)$) even if it is not large enough; for example, periods and commas in the line.

The first pass of the algorithm searches for potential major CC candidates. This is based on ($count(cc_i)$), location of the connected components and cc_{abv}. The second pass assigns minor CCs to their major CCs, and it may also correct wrongly assigned major CCs. This is based on connected components above and below cc_i. Finally, for each $major(cc_j)$ a PAW image is constructed by combining $major(cc_j)$ and all $minor(cc_i)$ belonging to cc_j. The output of this algorithm is a list of all PAW images within the input image (word or text line). Algorithm 2 shows the two-pass algorithm for extracting PAWs.

The words in the lexicon under study are segmented into their PAWs, to form a complete set $\mathcal{S}$ of PAWs extracted from the words database. Each PAW $s \in \mathcal{S}$ is assigned to a unique class. Each word consists of one to n PAWs, while PAWs can appear in more than one word and in different positions. This partial segmentation results in n new databases of PAWs $S^1, ..., S^n$, and for $1 \leq i \leq n$, S^i contains the

---
Algorithm 2 Partial Segmentation

$cc \leftarrow extractConnectedComponents(T)$
$h_{avgcc} \leftarrow$ Average height of CC
$w_{avgcc} \leftarrow$ Average width of CC
$count_{avgcc} \leftarrow$ Average number of pixels for CC
for all $cc_i \in cc$ **do**
   set $major(cc_i)$ based on the $count(cc_i)$ and location of cc_i relative to cc_{abv}
end for
for all $cc_i \in cc$ **do**
   set $minor(cc_i)$ based on the $count(cc_i)$ and location of cc_i relative to cc_{abv} and cc_{blw}
end for
for each $major(cc_i) \in cc$ **do**
   $\forall cc_j \rightarrow cc_i$ combine with $major(cc_i)$
end for

PAWs which appear in position i of words, and $\mathcal{S} = \bigcup_{i=1}^{n} S^i$. Similarly, test documents are segmented into PAWs.

5.3 Experiments and Results

The proposed method was evaluated using CENPARMI off-line words database (Section 4.4.2), and tested on CENPARMI documents database (Section 3.3.1).

5.3.1 Partial Segmentation Results

The partial segmentation algorithm which segments words into PAWs was applied to the CENPARMI documents database. A total of 33,765 PAW images were obtained from segmenting the text lines of 137 documents. For the 112 testing documents, 301 out of 2590 lexicon words were incorrectly segmented due to disconnections, touching PAWs, or segmentation errors from the algorithm. Table 5.1 shows a total of 4.93% segmentation error rate with 11.66% error rate on the lexicon words.

	Touching	Disconnected	Segmentation Errors	Total
Key Words (All Documents)	82	86	277	445
Key Words (Testing Documents)	51	62	188	301
All PAWs	343	327	995	1664
Percentage (%)	1.02	0.97	2.95	4.93
Error(%) on Lexicon Words	1.98	2.4	7.28	11.66

Table 5.1: Segmentation results

The results also show that almost 30% of the errors are due to touching and disconnected components, while the rest of the errors result from the segmentation algorithm. The segmentation errors occur mainly in the Arabic letters "Alef", "Ra'a", "Dal", and the diacritic of the "Kaf" (these letters are shown in Figure 5-2). The diacritic of the letter "Kaf" is usually placed above the major connected component to the right of the PAW to which it belongs, which increases the number of false positives. The other letters are often written in small sizes and overhang the component to the right of the letters, which increases the number of false negatives.

Arabic	English	Code
ا	A	Alef
ر	R	Ra'a
د	D	Dal
ک	K	Kaf

Figure 5-2: Arabic letters Alef, Ra'a, Dal and Kaf.

Table 5.2 shows the performance of the segmentation algorithm on the CEN-PARMI documents database. The results are promising with high recall and precision rates of 96.0% and 95.4% respectively. Disconnected samples would increase the total number of false positives, because disconnected components would result in additional PAWs that do not match the ground truth data. On the other hand, touching PAWs would have no match in the ground truth, which would increase the number of false negatives.

	FP	FN
No. of PAWs	1537	1356
Percentage (%)	4.55	4.02
	PR	**RR**
Percentage (%)	95.4	96.0

Table 5.2: Performance of the segmentation algorithm on CENPARMI documents Database

5.3.2 PAW recognition results

The result of applying partial segmentation Algorithm 2 to the isolated words database is a new database S of PAWs instead of words. The PAWs database consists of 92 classes with 33025 and 8035 training and testing samples respectively. Some PAWs appear in more than one word in different locations. This database was re-grouped into four different databases $S^1, ..., S^4$ depending on the locations of the PAW classes within the words. Since some PAWs may appear in more than one word in different locations, these PAWs may appear in more than one database.

Five different databases of PAWs are constructed. The first database (S^1) consists of all PAWs appearing at the beginning (right most) of the lexicon word, the second database (S^2) contains all PAWs following those in the first database as the second PAWs of the lexicon words. Similarly for the third (S^3) and fourth (S^4) databases. The database S contains the PAWs of all four classes. Gradient features (explained in Section 4.2.1) were extracted from the PAW images, then each database was trained using two different classifiers SVM (Section 4.3.1) and RDA (Section 4.3.2). Table 5.3 shows the number of classes, training and testing samples, and the recognition rates of the five PAW databases. While SVM and RDA have similar recognitions rates when applied to PAWs databases $S^1 - S^4$. SVM outperforms RDA when applied to the databases containing all PAWs.

Classifier	No of Samples		No of	Classifier Name	
Database	Testing	Training	Classes	SVM	RDA
S	7790	32051	92	90.37	85.2
S^1	4710	19295	54	93.38	93.4
S^2	4570	18930	36	92.7	89.87
S^3	2512	10445	18	92.48	87.91
S^4	761	3272	3	94.22	92.92

Table 5.3: PAW classifiers Recognition Results

A confusion matrix was produced for each of the above recognition systems. The results show that some similar PAWs were often confused, in particular when they may only differ by the number and/or the location of dots. Many people omit the dots because of their writing styles. This may make it difficult even for a native

reader to distinguish some PAWs of different classes in the absence of the context. Table 5.4 shows some samples of misclassified PAWs, together with their printed images. The most frequently confused classes share many similarities in appearance. Some of these samples were misclassified with very high posterior probabilities which sometimes exceeds 0.9.

Correct PAW	Confused With	Sample of Misclassified PAWs
ت	ن	
ر	د	
د	و	
ن	ة	
د	ر	
و	د	
صــيـل	ميـل	
ثـه	ئـه	

Table 5.4: Some most frequently misclassified PAWs.

5.4   Conclusion

We presented an automatic partial segmentation algorithm to segment Arabic handwritten word images or text lines into PAWs. This algorithm shows promising segmentation results on CENPARMI Arabic handwritten documents database with 95.4% and 96.0% precision and recall rates respectively. The algorithm was applied also to CENPARMI Arabic handwritten words database, which was segmented into new databases of PAWs. A recognition rate of 90.37% on this PAWs database was obtained when gradient features are extracted from the PAW images, and then trained and tested using SVM.

Chapter 6

Hierarchical Classifier for Arabic Handwritten Word Spotting

Many word spotting systems have been proposed, to retrieve words from digitized handwritten documents in a fast and efficient manner. These systems are either template matching based or learning-based. Template matching word spotting systems have been often applied to documents written by single writer, while learning-based word spotting system have been successfully applied to documents written by many writers.

The Arabic script can be seen as a sequence of PAWs rather than words. This is because the white spaces between and within words is of similar size. Many Arabic handwritten word spotting systems have been proposed. Most of them are based on PAW model, since PAWs are easy to extract. Khayyat et al. [39] proposed a hierarchical classifier that recognizes PAWs rather than words, and then PAW language models have been used with this classifier to re-construct words from their PAWs. This system has shown promising word spotting results and is able to spot words with different numbers of PAWs.

This chapter presents a learning-based system for multi-writer Arabic handwritten word spotting. The system aims to overcome the problem of not having clear boundaries between words in Arabic handwriting. In this chapter the proposed hierarchical classifier is described in Section 6.1, while Section 6.2 presents a state-of-the-the-

art reference system, this system has been implemented widely and successfully for Latin-based word spotting. Section 6.3 presents the conducted experiments and the results, Section 6.4 presents time complexity analysis of the system, and Section 6.5 examines the causes of errors. Section 6.6 compares the hierarchical classifier with a state-of-art word spotting system based on an HMM recognizer, while Section 6.7 contains comparison between the proposed system and other systems in the literature of Arabic handwritten word spotting. Finally, we conclude our work on this aspect in Section 6.8.

6.1 Hierarchical Classifiers

A hierarchical classifier is proposed to search for and recognize words within an Arabic handwritten document. This classifier aims to find boundaries of the lexicon words, and then to classify each word into one of the lexicon classes. The classifier consists of a sequence of classifiers $\{C_1, ..., C_n\}$, where n is the maximum number of PAWs within the lexicon words. The training and testing data are represented by the lexicon L and the testing documents respectively.

Training images are first partially segmented into PAW images and then gradient features (Section 4.2.1) are extracted. Similarly, testing documents are segmented into text lines and then text lines are segmented into PAW images in which gradient features are extracted as well.

The testing process starts by segmenting each text line of the testing document into PAW images as shown in Figure 6-1(a). Most of these PAW images are Out Of Vocabulary (OOV) PAWs. Each PAW image s is passed to classifier $C1$. The three PAW classes of the highest recognition scores given by classifier $C1$, are assigned to the three nodes n_{11}, n_{12} and n_{13} together with their confidence values. A graph is then created with a root node connected to the three nodes n_{11}, n_{12} and n_{13}. Section 6.1.3 presents the details of constructing the graph.

The classifier recognition results do not include contextual or semantic information, so the recognition results are not sufficient to evaluate the paths of a graph and

(a) Partial Segmentation

0.5 (ا, 15)
0.3 (د, 3)
0.2 (ه, 19)

0.6 (نتما, 16)
0.4 (نتها, 30)

0.5 (د, 15)
0.4 (ء,31)
0.1 (ر, 3))
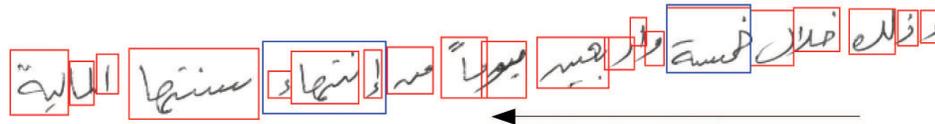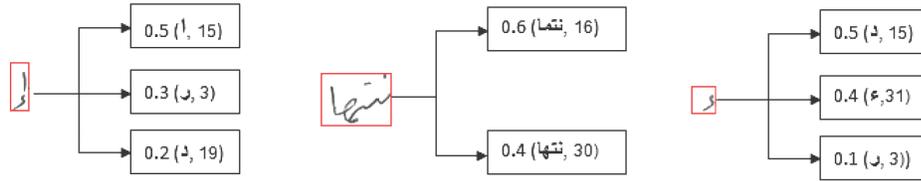
(b) Classification Results
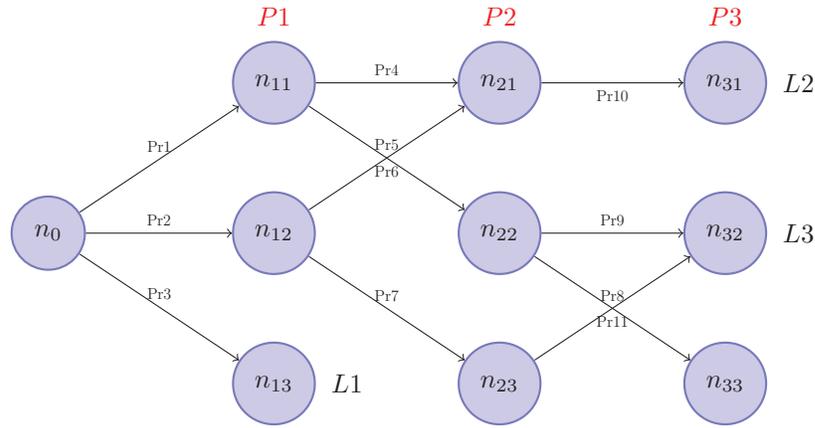
$P1$ $P2$ $P3$

n_{11} Pr4 n_{21} Pr10 n_{31} $L2$

Pr1 Pr5 Pr6

n_0 Pr2 n_{12} n_{22} Pr9 n_{32} $L3$

Pr3 Pr7 Pr8 Pr11

n_{13} $L1$ n_{23} n_{33}

(c) Graph Construction

(d) Language Models

Figure 6-1: The testing process of the hierarchical classifier

determine the optimal one. Consequently, language models (Section 6.1.2) are integrated with the PAW classifiers' confidence values to evaluate the paths. Figure 6-1 illustrates the word spotting process of the hierarchical classifier.

The classifiers $C_1, C_2, ..., C_n$ were trained based on three different methods: Support Vector Machines (SVM), Regularized Discriminant Analysis (RDA), and Modified Quadratic Discriminant Function (MQDF). These classification methods are

explained in sections 4.3.1, 4.3.2, and 4.3.3 respectively.

The probabilities produced by the SVM can be used as an approximation to a posteriori probabilities, so that they can be used as confidence values. However, the probability density function of both RDA and MQDF outputs discriminant scores that represent distances between samples and classes densities. These discriminant scores cannot be used as confidence values. Thus, a confidence transformation is applied to the output of these classifiers using Equations 4.18, 4.19, and 4.20 in Chapter 4. The RDA classifier is more efficient with lower dimensionality, so the dimensionality of the gradient features is reduced to 80 using the method described in Section 4.2.4.

6.1.1 Partial Segmentation

Partial segmentation refers to the process of segmenting the word images in $\mathcal{W}$ into PAWs in $\mathcal{S}$, where

$\mathcal{W}$ is the set of all word images from the lexicon L,

W_i is the subset of all words in class i, ($\mathcal{W} = \bigcup_{i=1}^{n} W_i$ and n is the number of word classes).

$\mathcal{S}$ is the set of all PAW images obtained from segmenting the images in $\mathcal{W}$ (S represents sub-words),

S_j is the subset of all PAWs in class j, ($\mathcal{S} = \bigcup_{j=1}^{m} S_j$ and m is the number of PAW classes).

Thus, each word image $w \in \mathcal{W}$ is represented as a sequence of PAWs in $\mathcal{S}$. Each PAW $s \in S$ is assigned to a unique class S_j. Each word can consist of one to four PAWs (the maximum number of PAWs within the lexicon under study is four).

This partial segmentation results in a new database LW of PAWs instead of words. LW is segmented into LW_i ($1 \leq i \leq 4$), where i is the location of the PAW according to the writing sequence ($LW = \cup_i LW_i$).

Similarly, each text line of a testing document is partially segmented into a sequence of PAW images, starting from the right most PAW image of a text line.

6.1.2 Language Models

Language models have been integrated with many handwriting recognition applications and word spotting systems. Chowdhury et al. [80] integrated a Weighted Finite State Transducent (WFST) based language model for online Indic script to improve the word recognition results. Wang et al [81] studied the effect of integrating different language models to handwritten Chinese script, and obtained significant improvements on their word recognition system. Jiang et al [82] integrated bigram models for a word lexicon to match the segmentation and recognition results. They found that contextual information plays a crucial role in Chinese character segmentation and recognition, and its usage can definitely improve the segmentation and recognition results. Fischer et al [83] integrated character $n-gram$ language models into their segmentation-free word spotting system, which significantly improved the system performance.

Language models [84] are integrated into our system to determine the probability of a sequence of PAWs within the lexicon words. Integrating both *bigram* and *trigram* PAW language models is appropriate for the the Arabic language [39], and these models are implemented to convey the contextual information.

Suppose s_i is a PAW for any natural number i. Then the conditional probability of the bigram $s_{i-1}s_i$ is defined as

$$P(s_i|s_{i-1}) = \frac{C(s_{i-1}s_i)}{C(s_{i-1})} \tag{6.1}$$

and that of the trigram $s_{i-2}s_{i-1}s_i$ is:

$$P(s_i|s_{i-2}s_{i-1}) = \frac{C(s_{i-2}s_{i-1}s_i)}{C(s_{i-2}s_{i-1})} \tag{6.2}$$

where $P(s_i|s_{i-n+1}...s_{i-1})$ are the n-gram probabilities that define the language model, and $C(s_i...s_n)$ is the number of occurrences of the sequence of PAWs $s_i...s_n$ in the isolated words lexicon.

6.1.3 Graph Construction

Let s_t be a PAW image in an Arabic handwritten text line, where $1 \leq t \leq m$ and m is the number of PAWs in a text line. Then s_1 is the right most PAW image in a text line, and s_{t+1} is the PAW image to left of s_t, since the direction of the Arabic handwriting is from right to left.

Suppose a PAW image s_t is passed to classifier C_1 to be recognized. The best three candidate PAW classes produced by the classifier C_1, together with their confidence values given by C_1 are assigned to three nodes n_{11}, n_{12}, and n_{13}. These nodes are then added to a graph, and connected to a root node denoted n_0.

If there is at least one non-leaf node, then the PAW image s_{t+1} will be passed to classifier C_2, similarly the best three candidate PAW classes with their confidence values given by classifier C_2 are assigned to the three nodes n_{21}, n_{22}, and n_{23}, which are added to the next level of the graph. This process is repeated for the following classifiers (i.e. C_3 and C_4), so that new nodes are added to the graph for each iteration, until all external nodes are leaf nodes. This is illustrated in Figures 6-1(b) and 6-1(c). Hence, a leaf node is a node that cannot be extended, because the assigned PAW class represents the last PAW in a word. Finally, all valid paths in a graph are evaluated, and the shortest path (path with lowest cost) is considered a lexicon word with a score equal to the path cost. Path evaluation is explained in Section 6.1.4.

The sequence of classifiers $\{C_1, ..., C_n\}$ forming the internal representation of the hierarchical classifier, has been trained and experimented using three different structures: default structure, pruning structure and multi-layer pruning structure. These structures are explained in the following sections.

Default Structure

For the default training of the hierarchical classifier, the words in lexicon L are partially segmented into a set $(\mathcal{S})$ of PAWs to construct a database LW of all PAWs, which is re-grouped into n PAW databases, denoted by LW_i where $(1 \leq i \leq n)$ and n is the maximum number of PAWs within a word in lexicon L. The first database

contains the rightmost PAWs of the lexicon words, the second database contains the second PAWs of the lexicon words, and similarly for the rest of the databases.

Each level i of the hierarchy contains a classifier C_i which is trained on the sub-database LW_i. Thus, a PAW s tested by classifier C_i can be assigned only to one of the PAW classes trained by the classifier C_i.

Pruning Structure

This structure is very similar to the default one; however, the first level of the hierarchical classifier (i.e. C_1) is replaced by a classifier trained on the set $\mathcal{S}$ (database LW), which contains all PAW images resulting from the partial segmentation of the words database (lexicon L).

Using classifier C_1 which is trained on LW to recognize a PAW image, may result in a candidate PAW class that is not necessarily located at the beginning (rightmost) of any of the lexicon words. This will prune out irrelevant PAW candidates, that are not located at the beginning of the word (If a PAW s_t is tested by classifier C_1, and none of the three candidate PAW classes appear at the beginning of a lexicon word, then no graph will be constructed, and the next PAW s_{t+1} will be passed to classifier C_1 instead of C_2 to be recognized). This will significantly reduce the number of graphs created to spot and find the boundaries of an Arabic handwritten word.

Multi-layer Pruning Structure

In this structure, the internal PAW classifiers $\{C_1, ..., C_n\}$, of the hierarchical classifier are all trained on the same set $\mathcal{S}$ (database LW) of all PAWs, so that $C_1 = C_2 = ... = C_n$. Subsequently, if a PAW s_t tested by classifier C_i ($1 \leq i \leq n$), and none of the three candidate PAW classes assigned to s_t are located in the ith location of the lexicon words, then the graph will not be extended, and the next PAW s_{t+1} will be passed to classifier C_1 instead of C^{i+1} (the next internal classifier).

Similar to the pruning structure, when a PAW image s_t is tested by classifier C_1, and the three candidate PAW classes assigned to s_t and recognized by classifier C_1, are not located at the beginning of any of the lexicon word, then no graph will be

created. After which the following PAW s_{t+1} will be passed to classifier C_1.

This structure does not only reduce the number of graphs created, but also prunes the length of the paths within a graph.

6.1.4 Path Evaluation

Nodes in the graph denote candidate PAW classes, while links are created between PAWs depending on the probabilities given by the PAW language models.

Each node is assigned to a candidate PAW class S_i for a PAW s tested by classifier C_j where j is the classifier level ($1 \leq j \leq 4$), and the cost of the node is obtained from the confidence value $P(s|S_j^i)$ given by classifier C_i and assigned to class S_j. Links between nodes are evaluated using the probabilities $P(S_i)$ given by the PAW language models.

The strength α_i of s_i is defined as follows:

Suppose s_i occurs in the n lexicon words $\{W_i^1, W_i^2, ..., W_i^n\}$,

l_i^j denotes the location of s_i in word W_i^j ($1 \leq j \leq n$),

f_i^j is a factor determined by the number of PAWs following s_i in word W_i^j, then

$$\alpha_i = \sum_{j=1}^{n} l_i^j f_i^j \tag{6.3}$$

All paths between the root node and leaf nodes are evaluated using the following formula

$$R = \frac{1}{p} \sum_{i=1}^{p} \alpha_{Sj} |\log P(s|S_j^i)| + \frac{1}{q} \sum_{l=1}^{q} |\log P(S_l)| \leq t \tag{6.4}$$

where R is the path cost, p and q are the numbers of nodes and paths respectively within a given path, j is the class number to which the node is assigned, and t is a user defined threshold to accept or reject the path.

6.2   Reference System

The Hidden Markov Model (HMM) statistical classifier (recognizer) has been widely and successfully used for Latin-based handwritten word spotting [25, 85, 86], and it has also been used to spot words from Arabic handwritten documents [9, 15]. This classifier is used to spot words based on the character model, and it adapts to cursive handwriting. Thus, we chose an HMM implementation similar to the one presented in [25] as reference system, as it is a widely used reference system for word spotting.

The reference system [25], including preprocessing, feature extraction and recognition, is explained briefly in the following sections.

6.2.1   Preprocessing and Feature Extraction

Using the HMM statistical classifier requires adequate preprocessing, accordingly the following preprocessing steps were applied to the text line images:

1. Skew correction based on the linear regression of the lower black pixel of each column [70].

2. Pepper noise removal is applied, in which connected components containing less than 5 pixels are removed.

3. Salt noise removal is applied based on morphological closing operation (i.e. dilate then erode).

4. Images are then size normalized so that all text line images are 122 pixels high.

5. Horizontal mirroring is applied to the images, so that it adapts to the writing direction of the Arabic script which is from right to left (the opposite direction of Latin-based script).

A one pixel width sliding window is passed through each word image or text line, and nine geometric features [70] are extracted from each sliding window. These features contain three global features representing the density, center of gravity and second order moment of the image. The rest of the features are local in nature,

consisting of the upper and lower contours, the gradient of the upper and lower contours, the black and white transitions, and the number of black pixels between the upper and lower contours.

6.2.2 HMM Classifier

In the Arabic language, each letter has between two and four shapes. Each of these shapes is modeled using a left-to-right linear topology HMM, with a certain number m of hidden states $s_1, \ldots, s_m$ for each HMM. The parameter m is empirically chosen using a validation set, and its value increases according to the width of the character. For this experiment, m was chosen to be 30% of the mean width; which is determind using HMM-based forced alignment. A space character model with 10 hidden states was added, in addition to the Arabic characters (shapes) appearing in the 69 keywords. Figure 6-2a illustrates the character-based HMMs. This system was implemented using the popular Hidden Markov Model Toolkit (HTK)[1].

The states s_j with $1 \leq j \leq m$ emit observable feature vectors $\mathbf{x} \in \mathcal{R}^n$ with output probability distributions $p_{s_j}(\mathbf{x})$. The Gaussian Mixture Model (GMM) is used to calculate $p_{s_j}(\mathbf{x})$ as follows

$$p_{s_j} = \sum_{j=1}^{G} w_{jk} \mathcal{N}(\mathbf{x}|\mu_{jk}, \Sigma_{jk}) \tag{6.5}$$

where w_{jk} are positive weights that sum up to one, G is the number of the Gaussian Mixtures and μ_{jk} and Σ_{jk} are the mean and the covariance respectively of the normal distribution $\mathcal{N}(\mathbf{x}|\mu_{jk}, \Sigma_{jk})$. For this experiment, the number of Gaussian mixtures is 17. This number is optimized based on preliminary single word recognition experiments among the 69 lexicon words, on the validation set of the single word recognition task

The character model HMMs are trained with the transcription of each word image in lexicon L. The Baum-Welch algorithm [87] maximizes the probability of the

---

[1]http://htk.eng.cam.ac.uk/.
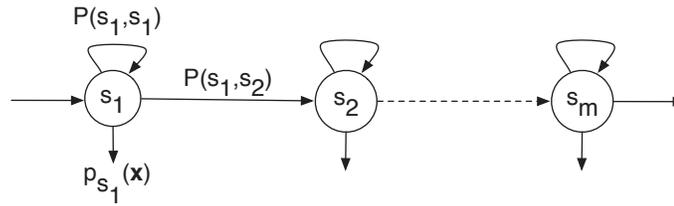
(a) Letters HMM
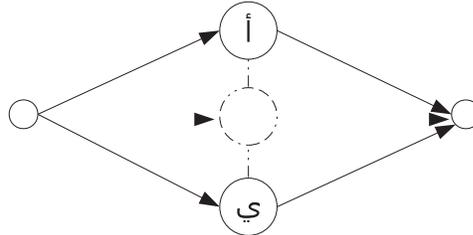
(b) Filler HMM
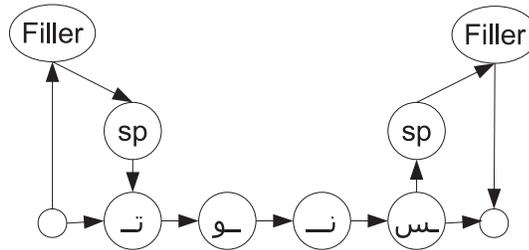
(c) Word HMM

Figure 6-2: Hidden Markov models.

word image model to emit the observed feature vector sequence $x_1, ..., x_N$ with an initial output probability distribution $p_{s_j}(x)$ and transition probabilities $P(s_j, s_j)$ and $P(s_j, s_{j+1})$ between states s_j and s_{j+1}.

Given a text line, the Viterbi algorithm [87] is used to calculate the likelihood of the observed feature vector sequence $x_1, ..., x_N$. This outputs the most likely character sequence, in addition to the beginning and the end positions of the characters.

Finally, the score $s(X, W)$ is calculated based on the log-likelihood ratio between a lexicon word text line model (K) and a filler text line model (F). This is shown in

Figure 6-2c and Figure 6-2b respectively, and the score is calculated as follows

$$s(X, W) = \frac{\log P(X|K) - \log P(X|F)}{L_K} \qquad (6.6)$$

where L_K is the number of characters in a keyword. Threshold T is applied to spot a word when $s(X, W) \geq T$. This method is explained in [25]. The difference between spotting words in this HMM word spotting system and that in [25], is that the log-likelihood difference is divided by the number of characters in the keyword for this experiment, instead of the number of assigned HMM states in [25].

6.3   Experiments and Results

The proposed system was trained on CENPARMI Arabic handwritten words database and evaluated on CENPARMI Arabic handwritten documents database, described in Section 6.3.2. The results of the proposed systems are discussed in Section 6.3.3, in which different structures of the hierarchical classifier are implemented, then evaluated and compared. Finally, Section 6.3.4 compares the performance of the proposed system with a system for which the PAWs are manually and correctly segmented.

6.3.1   Experimental Setup

Training data are prepared by a semi-automatic procedure which results in the construction of four new databases of PAWs (instead of words) in addition to one database containing all PAWs. The databases are obtained by applying the partial segmentation algorithm (explained in Chapter 5) to the words database, and then automatically labeling the PAWs of each word according to their locations within that word. In addition, incorrectly segmented or labeled PAWs are manually corrected.

The precision-recall curve is used to show the performances of the word spotting systems. This curve is also used to calculate the Mean Average Precision (MAP) represented by the area under the curve, and the *R-Prec* which gives the rate at which the recall and precision graphs intersect. Also shown are the Precision Rate (PR),

Recall Rate (RR), and $f1_{score}$ of the system when words of one, two, three or four PAWs are evaluated.

The proposed word spotting system was tested using three different internal structures: default, pruning and multi-layer pruning (explained in Section 6.1.3), each of which was trained using an SVM. In addition, three different hierarchical classifiers (SVM, RDA and MQDF) were used to implement the internal classifiers of the hierarchical classifier containing all PAWs (i.e. multi-layer pruning structure).

6.3.2 Databases

The CENPARMI Arabic off-line handwritten words database (described in Section 4.4.2) contains words from 69 classes, with each word consisting of one to four PAWs. This database represents the lexicon under study.

The CENPARMI Arabic unconstrained handwritten documents database which was designed for commercial applications (described in Section 3.3.1) is also used for this experiment. These documents were written according to 12 different templates by 24 writers, with each template adopted by 8 to 13 writers. In this experiment 137 documents divided into 112 testing documents and 25 validation documents were used. The documents contain 2590 and 678 lexicon words in the testing and validation documents respectively.

6.3.3 Performance Evaluation

Table 6.1 summarizes the statistical results of implementing three systems using the three different internal structures of the hierarchical classifier. All internal classifiers in these three systems were trained using SVM. The result shows that adding more PAWs to the internal classifiers in any level has improved the performance of the system. The default structure has the lowest MAP, and this is because each of the internal classifiers is trained only on a subset of the PAWs database. Training all the internal classifiers on all PAWs appearing in the words database (multi-layer pruning structure), outperforms the other two structures and achieves promising results with
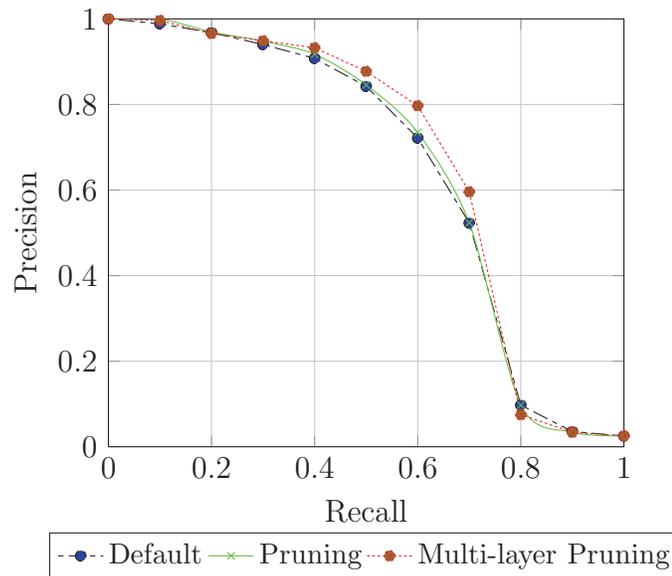
Figure 6-3: Precision-Recall Curves

a precision rate of 87.73% at 50% recall.

Summary Statistics			
	Default	Pruning	Multi-layer Pruning
MAP	0.6586	0.6602	**0.6753**
R-Prec	0.6433	0.6485	**0.6718**
$PR(\%)$ at $RR = 50\%$	84.22	84.56	**87.73**

Table 6.1: Summary statistics of results

Figure 6-3 shows the Precision-Recall curves for the three word spotting systems. All systems are based on the hierarchical classifier but each with different internal structure (default structure, pruning structure and multi-layer pruning structure). All systems produced promising results, with the systems using the multi-layer structure showing better precision-recall curves than the other two systems, by about 2% increase in the MAP. The default and pruning structures have very similar behaviours with a slight increase in the MAP for the pruning structure.

Table 6.2 summarizes the results of the three systems under study, i.e. the SVM hierarchical classifier, the RDA hierarchical classifier and the MQDF hierarchical classifier, each of which is implemented using the multi-layer structure for the internal classifiers. All systems produced promising results, with the systems using the SVM
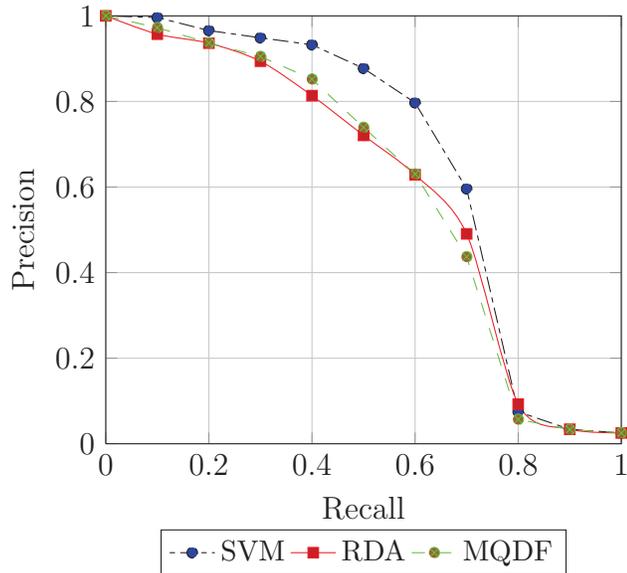
Figure 6-4: Precision-Recall Curves of the hierarchical classifier implemented using SVM, RDA and MQDF

classifier showing better precision-recall curves than both RDA and MQDF classifiers, by about 10% increase in the MAP.

	SVM	RDA	MQDF
MAP	**0.6753**	0.6079	0.6022
R-$Prec$	**0.6718**	0.6105	0.6109
$PR(\%)$ at $RR = 50\%$	**87.73**	72.07	73.93

Table 6.2: The performance of the hierarchical classifier implemented using SVM, RDA and MQDF

Similarly, Figure 6-4 shows the Precision-Recall curves of the three systems. The RDA and MQDF systems perform similarly with a slight difference between them, while the SVM system performs better than these two. This is because the confidence values were estimated in the case of RDA and MQDF, but not for the SVM.

The lexicon words under study in this work contain different numbers of PAWs ranging from one to four. We group the words according to the number of PAWs they contain. The three systems (SVM, RDA and MQDF) were implemented using the multi-layer pruning internal structure. The results are presented in Figure 6-5 which shows the precision rate, recall rate and $f1_{score}$ respectively, against the path cost R. Graphs (a) to (c), graphs (d) to (f), and graphs (g) to (i) show the results of

69

SVM

(a) (b) (c)

RDA

(d) (e) (f)

MQDF

(g) (h) (i)

- - •- - 1-PAW - - ■- - 2-PAWs ── •── 3-PAWs - -✶- - 4-PAWs
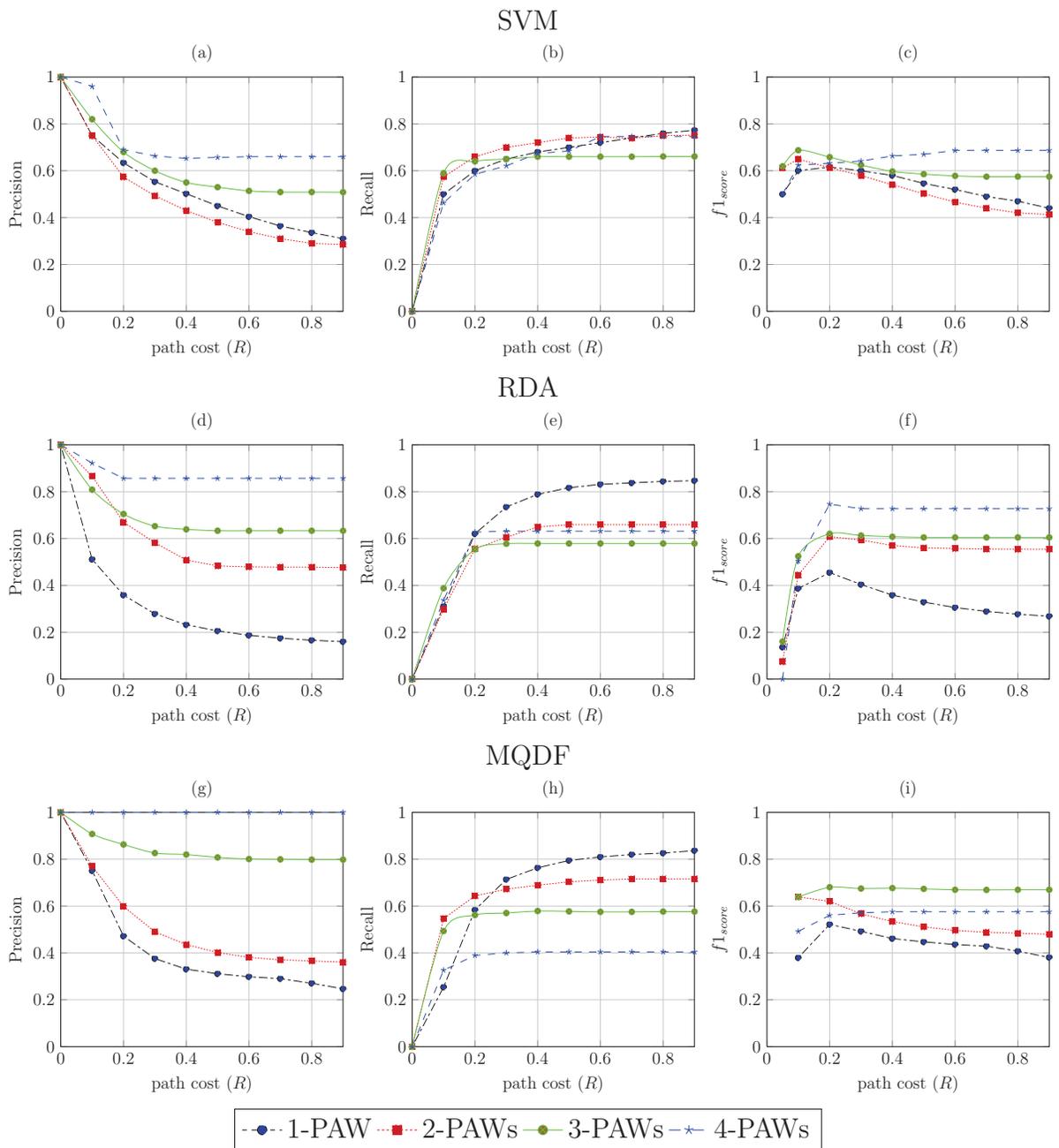
Figure 6-5: Comparison between words containing different numbers of PAWs

the SVM, RDA and MQDF systems respectively.

Graphs (a) to (c) of Figure 6-5 show that when SVM is used, all curves have a similar pattern of behaviour for different numbers of PAWs, even though words of four PAWs have higher precision rate and accordingly higher $f1_{score}$. This is because having longer paths adds more restrictions on a word, which would increase the

probability of correctly spotting the word.

When the RDA or the MQDF classifiers are used for the various hierarchical classifier levels, the precision rate increases monotonically with the number of PAWs within a word. The reason is that having more PAWs would result in a longer path with higher confidence.

It can be observed that the recall rates tend to be similar for words with different numbers of PAWs, while the precision rate increases monotonically with the number of PAWs within a word, for all classifiers used. If a word of one PAW was incorrectly recognized by the first classifier with a high probability, this can lead to a false positive. For this reason, the confidence of the classifier has high impact on the proposed system. Thus, improving the confidence estimation of the RDA and the MQDF may significantly improve the performance of the systems.

6.3.4   Comparison with Correct Segmentation

To compare the performance of the proposed system with one in which the segmentation is completely correct, we also implemented a semi-automatic segmentation process that resulted in correctly segmented PAWs. Thirteen documents containing 291 lexicon words were prepared by 13 writers. The documents were segmented into PAWs using the proposed segmentation algorithm. A total of 25 lexicon words were incorrectly segmented, either because two PAWs are touching, or because a PAW is disconnected or incorrectly segmented by the algorithm. These errors were manually corrected and then the system was evaluated.

	Algorithm	Manually Corrected
MAP	0.6414	**0.7322**
$R\text{-}Prec$	0.6357	**0.6804**
$PR(\%)$ at $RR = 50\%$	86.90	**91.41**

Table 6.3: Comparison of the word spotting system performance using segmentation algorithm and manual segmentation

Table 6.3 shows the performances of two word spotting systems when applied to the thirteen documents. Both systems are based on the proposed hierarchical classi-

fier trained with SVMs with the muli-layers pruning internal structure, with different segmentation processes. The first system applied the proposed automatic segmentation algorithm, and the second system manually corrects the incorrectly segmented PAWs produced by the segmentation algorithm. The results show that manually correcting the segmentation errors outperforms the system integrated with the proposed segmentation algorithm, with 0.09 higher MAP. This shows the performance of the system can be enhanced by an improved segmentation algorithm.
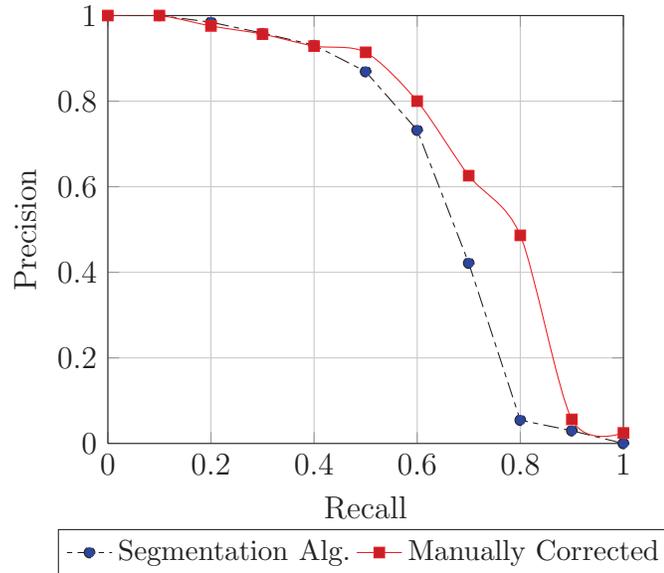
Figure 6-6: Precision-Recall Curves for Word Spotting Results of Segmentation Algorithm and Manually Corrected Segmentation Errors

Figure 6-6 illustrates the performance of the two systems. The systems show similar performance at high precision rates. The system with the manually corrected samples performs better at high recall rates, because the PAWs of the corrected samples were recognized by the internal classifiers, but with less confidence.

6.4   Time Complexity

All experiments were conducted on a PC with 4.0GB RAM, 2.66GHz processor, Windows OS, and C++ code. The system under study was divided into 3 main procedures, namely: segmentation, feature extraction and word spotting which includes

the recognition and graph construction. Table 6.4 shows the time complexity for the proposed system including SVM hierarchical classifier, RDA hierarchical classifier and MQDF hierarchical classifier. The time complexity is calculated per document and also per 100 PAWs. The average number of PAWs per document in the CENPARMI Arabic handwritten documents database is around 238.

The results show that the RDA hierarchical classifier operates at approximately 2.3 times the speed of the SVM hierarchical classifier. For the SVM hierarchical classifier, 70% of the time is consumed in classification and graph construction, while the RDA hierarchical classifier uses 50% of the time for segmentation. The MQDF hierarchical classifier consumes less time than the RDA hierarchical classifier for feature extraction, since no dimensionality reduction is performed for MQDF. However, MQDF consumes more time than RDA in classification and graph construction, and this is because feature vectors passed to MQDF are of higher dimensionality than those passed to RDA.

In conclusion, the RDA and MQDF hierarchical classifiers are significantly faster than the SVM hierarchical classifier. This means improving the confidence estimation of the RDA and the MQDF can produce fast and reliable word spotting systems.

Classifier	Segmentation	Feature Extraction	Word Spotting	Total
Average time in seconds per document				
SVM	4.78	1.62	14.58	20.97
RDA	4.78	2.32	2.13	9.22
MQDF	4.78	1.62	5.25	11.55
Average time in seconds per 100 PAW				
SVM	2.00	0.76	6.12	8.88
RDA	2.00	1.05	0.89	3.95
MQDF	2.00	0.76	2.25	5.01

Table 6.4: Time complexity of the systems

6.5 Error Analysis

When the errors produced by the system were analysed, it was noted that most of the false negatives were due to segmentation errors and some particular writing

styles which created touching or disconnected PAWs. A compilation of disconnected, touching and incorrectly segmented PAWs show that 4.7% of the lexicon words were touching or disconnected, and 7.3% of the words had segmentation errors. This means that about 12% of the recall was caused by segmentation errors. Consequently, it can be inferred that improving the segmentation process may increase the number of false negatives and the recall rate. Moreover, improving the performance of the RDA and the MQDF classifier is highly dependent on the estimated confidence values, and this can significantly affect the number of false negatives.

Many false positives resulted from words having the same root. For example, the Arabic word for forty was often spotted as four and the word eighty spotted as eight. In other cases, words representing numbers may have more than one connected component in common. Similarly, the singular and plural forms of a lexicon word may be too difficult to distinguish, since these forms have most of the PAWs in common and in the same sequence. In addition, many words not in the lexicon can be so similar to the lexicon words that even an Arabic reader may not be able to distinguish between them in the absence of context. Table 6.5 shows some examples of false positives.

Table 6.5: Samples of false positives and their assigned classes

Lexicon Words		False Positives
English	Arabic	Handwritten
Four	اربعة	أربعين
Expire	انتهاء	أ نتهاء
Liter	لتر	لشر
Hundred	مائه	هالة
Eight	ثمانية	ثمانين
Article	بند	حت
Six	سته	لسنة

74

6.6 Comparison with Reference Word Spotting System

Three different implementations (SVM, RDA and MQDF) of the hierarchical classifier based on the multi-layer pruning structure, are compared with the HMM implementations described in Section 6.2 to spot Arabic handwritten words. The results are shown in Table 6.6. The three different implementation of the hierarchical classifier based word spotting have better performance than the HMM based word spotting. The SVM implementation outperformed all the other systems with a MAP of 67.53% and a precision rate of 87.73% at 50.00% recall rate.

	Hierarchical Classifier			HMM
	SVM	RDA	MQDF	
MAP	**0.6753**	0.6079	0.6022	0.2557
R-$Prec$	**0.6718**	0.6105	0.6109	0.3195
$PR(\%)$ at $RR = 50\%$	**87.73**	72.07	73.93	13.50

Table 6.6: Word spotting performances of the hierarchical classifiers (SVM, RDA, MQDF) and the HMM system

The reference system used in [25] was implemented for Latin-based word spotting, and it was trained on documents containing all the characters appearing in the testing documents, so there were no OOV characters in the testing documents. However, for this implementation of the reference system, the HMM classifier is trained on handwritten word images, and tested on handwritten text lines. As mentioned earlier in Section 6.2.1; using HMM classifier requires sufficient and good text line normalization, so that the within class (character) variation is reduced. The normalization we applied is suitable for Latin-based text lines, but not for Arabic ones which has different characteristics, with the result that the characters appearing in the training images have slightly different shapes than those in the testing text lines.

Figure 6-7 presents the precision-recall curves of the four word spotting systems. The curves show that the HMM system did not perform well on the Arabic database under study. This could be because the skew correction procedure applied to the text lines is applicable to the Latin script, but not necessarily applicable to the

Arabic script. In addition the geometric extracted features are also appropriate for Latin-based script, while the Arabic script is cursive by nature and contains different geometric information; so that changing the features may improve the results. On the other hand, the three different classifiers (SVM, RDA and MQDF) used to recognize PAWs for the internal structure of the hierarchical classifier have promising results. This is because the extracted gradient features are suitable for the Arabic handwriting, and the chosen holistic classifiers have strong recognition ability, for which only basic preprocessing is needed.
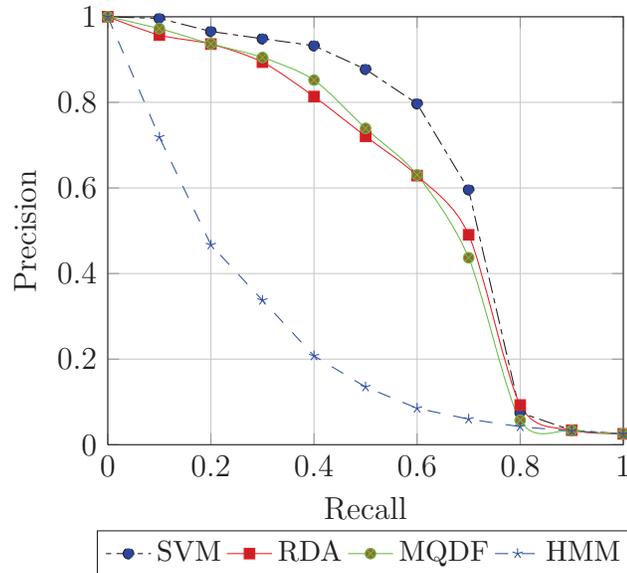
Figure 6-7: Precision-Recall Curves for three hierarchical classifiers implemented using SVM, RDA and MQDF respectively, and an HMM based classifier

6.7 Comparison of Results on Arabic Word Spotting

Table 6.7 compares some Arabic word spotting methods with our method. These comparisons would support the validity of our approach.

Ball et al. [36] presented three Arabic handwritten word spotting systems based on different approaches to segmentation: segmenting a document into characters, into words, and a manual segmentation of the document into words. The systems were

Author	Database	Recall (%)	Precision (%)
		50.0	28.0
Ball et al. [36]	CEDARABIC		34.0
			65.0
Leydier et al. [19]	Single writer	–	72.0
			80.0
Shahab et al. [32]	Single writer	–	75.5
Srihari and Ball [21]	CEDARABIC	50.0	70.0
Wshah et al. [11]	AMA Arabic	50.0	60.0
Fischer et al. [25]	AMA Arabic	20.0	18.0
Fischer et al. [25]	CENPARMI	50.0	13.5
Khayyat et al. [34]	**CENPARMI**	**50.0**	**87.73**

Table 6.7: Comparison with other Arabic word spotting systems from the literature

evaluated using the CEDARABIC documents written by 10 writers. With a recall rate of 50%, the systems reported precision rates of 28%, 34%, and 65% respectively. Leydier et al. [19] and Shahab et al. [32] presented Arabic word spotting systems that were evaluated on documents provided by a single writer, and only one query of one PAW was tested in the former. These approaches resulted in precision rates of 72.5% and 80% respectively.

Srihari and Ball [21] had proposed a language independent word spotting system that was tested on Devangari, Arabic and Latin scripts. The lowest performance had resulted from the Arabic script, even though these documents were manually segmented into words. When the system was trained using documents provided by eight writers, the precision was 70% at 50% recall rate. Wshah et al. [11] proposed a word spotting system that out-performed a state-of-the-art word spotting system that utilized Hidden Markov Models [25] on Arabic, English and Devanagari. For both systems, the lowest performance resulted from the Arabic script. The Wshah et al. system had 60% precision with 50% recall, while the Fischer et al. [25] system produced the highest recall of 20% with precision 18.0%. The hierarchical classifier reported in [39] produced 74% precision with 50% recall, when this system was tested on a subset of 43 document from CENPARMI handwritten Arabic documents. The implemented hierarchical classifier is based on two thresholds to prune out weak candidates and to tolerate different writing styles.

Our hierarchical classifier using SVM with a multi-pruning structure was designed to spot complete words in documents, and it was tested on 112 documents freely written by 24 writers in their own styles. This system has promising results of 87.73% precision rate with 50% recall rate.

6.8 Conclusion

This chapter presents a coherent, learning based and multi-writer Arabic word spotting system. The system is based on a PAW or sub-word model, in which words are spotted based on PAWs. Word spotting is implemented using a hierarchical classifier consisting of a sequence of classifiers, each of which recognizes PAWs rather than words. Language models are proposed to integrate contextual information with the confidence values given to the PAWs by the classifier sequence.

The proposed word spotting system was able to overcome the lack of boundaries problem in Arabic handwriting, so that words composed of different number of PAWs could be spotted with similar performance. The system using SVM and trained with a database containing all PAWs (multi-layer pruning structure) resulted in the best performance, with 87.73% precision rate at 50.0% recall rate and 67.53% MAP. Nevertheless, other implementations of the hierarchical classifiers have produced promising results as well.

Finally, the systems trained using RDA and MQDF classifiers are efficient time wise, so improving the confidence values of this system can be expected to result in a fast and reliable Arabic word spotting system.

Chapter 7

Ensemble of Classifiers for Word Spotting

Text lines in any script can be viewed as a sequence of words, while many word spotting systems favour characters or sub-words modeling rather than words modeling. The majority of the successful Latin-based handwritten word spotting systems are segmentation-free, based on the letter (character) model. Chinese handwritten word spotting systems favour the Chinese character model, and Arabic handwritten word spotting systems favoured PAW or sub-word modeling.

Thus, for multi-writer learning-based word spotting systems, analytical approaches are widely used, and lines are usually explicitly or implicitly segmented into graphemes such as letters, sub-words etc, instead of words. Then words can be reconstructed using special algorithms such as Viterbi algorithm [87] for HMMs, or using language model probabilities. However, for holistic classifiers one feature vector representing the most significant information of a word image can be extracted from a word, and then passed to the classifier. Thus a word can be viewed as an entire unit, and words of few or many graphemes (i.e. characters) have feature vectors of an identical dimensionality.

Both the holistic approach and the analytical (grapheme-based modelling) approach have their pros and cons, while they can complete each other. Combining the two approaches has shown promising results for handwriting recognition [88]. In this

chapter we propose three verification models for postprocessing handwritten word spotting systems. These models are based on combining a holistic classifier with an analytical classifier, in which the holistic classifier (words classifier) is used to verify the spotted words. The first verification model rejects all the spotted words in which the holistic and the analytical classifiers do not agree, while the other two verification models integrate a new score evaluation procedure for spotted words based on the scores given by both the analytical and the holistic classifiers.

The rest of this chapter is organized as follows, Section 7.1 compares the holistic and the analytical models, Section 7.2 describes the word spotting systems used in this experiment, Section 7.3 describes the verification methods, and Section 7.4 presents the experimental results, and the chapter is concluded in Section 7.5.

7.1 Holistic Versus Analytical Paradigm

The holistic paradigm is an approach that recognizes the word as a whole or uses shape features to describe a word. On the other hand, the analytical paradigm treats a word as a collection of sub-units or graphemes such as characters, and segments the word into these units [89]. In fact, some approaches attempt to systematically divide an image into many overlapping pieces or windows without regard to the contents, and then the final classification decision is made based on the integration between segmentation and recognition. In the literature of handwriting recognition, these approaches have been often called *"segmentation-free"*. Casey and Lecolinet [90] consider the term segmentation-free to be misleading, since any method involving either explicit or implicit segmentation is referred to as an analytical approach, also because no global features are extracted and the images are not treated as an entire unit. Yet other studies consider this approach to be a global holistic one. In this chapter we consider segmentation free approaches and any approach based on explicit or implicit segmentation as an analytical approach.

Both the holistic and the analytical paradigms were investigated by reading psychologists [91, 92, 93]. Some theories suggested that words are identified from their

global shapes, while others favor identifying words from their components (graphemes or letters). The former theory of reading find that predicting words written in lower case is easier and faster than those written in uppercase, since upper case has no shape features such as descenders and ascenders etc. Also, people are still able to read words even if the letters of the word are not in the correct order, and degraded words with some missing or completely degraded characters can be read because of the holistic paradigm in reading. Readers are also able to guess words in a sentence if they are given enough shape information about that word. However, the latter theory argue that the letters allow words to be recognized.

The holistic approach has achieved higher recognition accuracy given that the lexicon is static and limited; however for large and dynamic lexicons, searching for letters and then for words may be more applicable. Analytical approaches have been successfully applied to many handwritten word spotting systems, especially to cursive handwriting for which there is no clear white space between words, where segmenting a text line into words can be problematic. This is due to the fact that the analytical approach integrates segmentation with classification to determine the boundaries of the words.

The analytical approach can find the boundaries of a word by recognizing small units, such as letters or sub-words, after which probabilistic models such as language models or transition probabilities are used to connect these units. This combination can produce a strong model to form words from their unit or graphemes. Given all the above, both the holistic and the analytical approaches have their advantages and drawbacks, thus having a hybrid system which combines both approaches can lead to better performance, and verifying one model with the other can also improve the performance of the system.

7.2 Word Spotting Systems

Given a text line $\ell = \{w_1 w_2 ... w_n\}$, where w_i is the ith word in ℓ and n is the number of words, and a line ℓ can be segmented into smaller units or graphemes, such that

$$\ell = \{g_1 g_2 ... g_m\}$$

where g_j is a grapheme and m is the number of graphemes within ℓ, then a word in line ℓ ($w_i \in \ell$) can be denoted as $w_i = \{g_k ... g_{k+d}\}$, where $d \geq 1$ is the number of graphemes in the word w_i.

Let w be a word in a lexicon L, where w consists of $d \geq 1$ graphemes, and $w = \{g_1 ... g_d\}$. Spotting the word w is the task of searching for this word within a text line ℓ, while Arabic handwriting does not have boundaries between words. Thus text lines can be segmented into graphemes g_i's that can be easily extracted, and then the word w can be spotted by searching for an occurrence of the sequence $w = \{g_1 ... g_d\}$ within a sequence of a text line graphemes $\ell = \{g_1 g_2 ... g_m\}$.

In this section two different analytical classifiers for word spotting are presented, each of which is based on different graphemes for modelling. The first system integrates the partial segmentation into PAWs with language models to spot Arabic handwritten words (hierarchical classifier), while the other system utlizes HMMs based on the character model. Later, different methods are used to re-construct the words.

The hierarchical classifier and the HMM word spotting system are trained on a database of Arabic handwritten word images of a limited lexicon. On the other hand, Arabic handwritten documents are passed to the word spotting systems for testing. These documents are segmented into text lines; each text line may contain out of lexicon or OOV graphemes i.e. PAWs for the former and letters for the later.

Four analytical word spotting classifiers are used in this chapter. Three of these implement the multi-layer pruning structure (Section 6.1.3) to train the internal classifiers of the hierarchical classifier which is explained Section 6.1. Each implementation of the hierarchical classifier utilizes a different classification method or recognizer (SVM, RDA or MQDF) to train the internal classifiers. The last classifier

is implemented based on an HMM recognizer and this implementation is described in Section 6.2. Here, the hierarchical classifier aims to solve the lack of boundaries problem which appears in Arabic handwriting. This classifier is only used to spot Arabic-based handwritten words, since it is based on the partial segmentation of the words into sub-word or PAWs that only appear in the Arabic-related scripts such as Farsi, Pashto and Urdu.

7.3   Verification Models

Word spotting systems in this chapter are implemented using two models: character and PAW; the former is implemented using character HMMs which the word is spotted based on a score between the filler and the keyword models (see Figure 6-2 (b) and (c) respectively), while the latter is implemented using the hierarchical classifier which integrates PAWs with language models. These two approaches can be considered analytical approaches, since the word is spotted based on searching for the graphemes of that word, i.e. characters or PAWs; in addition they are based on implicit or explicit segmentation of a text line into small units. This means that the word cannot be seen as one unit, because the classifiers (recognizers) are trained and tested on graphemes instead of words. The proposed verification approaches are based on the following assumptions.

Suppose L is a lexicon of n word classes, $\mathcal{W}$ is the set of all lexicon word images (described in Section 4.4.2), and W_i is the set of all words in class i, where $1 \leq i \leq n$ and $W_i \subseteq \mathcal{W}$.

Sample images in the isolated words database $\mathcal{W}$ are size normalized to 50×120 pixels. Gradient features are extracted and passed to a holistic classifier for training. Three different classification methods were used to implement the holistic classifier which verifies the spotted words. These methods are SVM, RDA and MQDF.

Testing documents are passed to an analytical word spotting classifier. Suppose $score_A$ is the score calculated by the analytical classifier, where $score_A = R$ (Equation 6.4 in Chapter 6) or $score_A = s(X, W)$ (Equation 6.6) for the hierar-
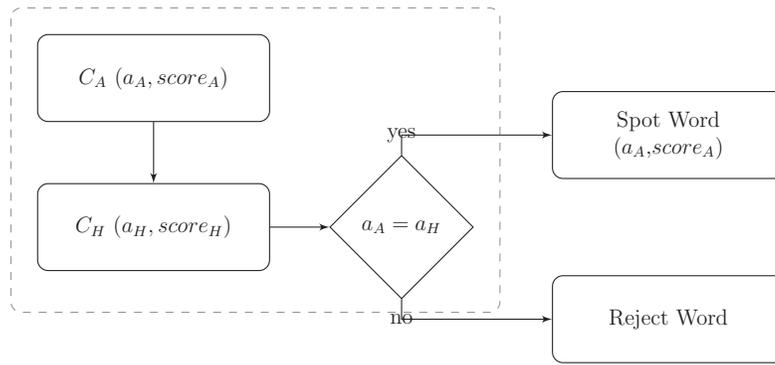
(a) Word Matching Model
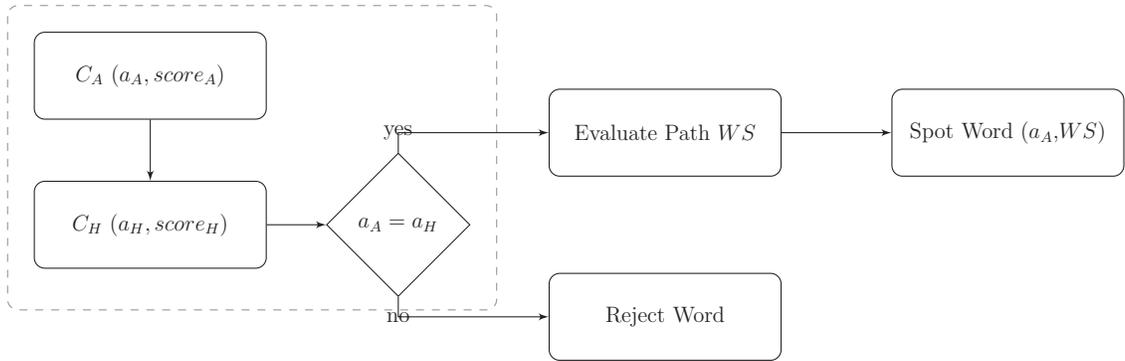


(b) Improved Score Word Matching
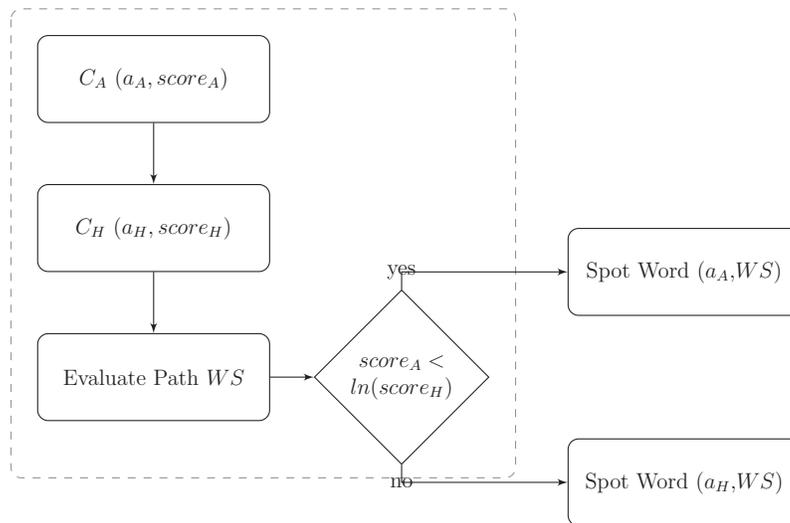
(c) Score Evaluation Model

Figure 7-1: Verification Models

chical classifier and the character HMMs respectively. All spotted words with path cost $score_A \leq t$ are size normalized as the training images, then gradient features are extracted from spotted word images, and passed to a holistic classifier for verification.

Three different models are proposed to verify spotted words, Figure 7-1 illustrates the proposed models, which are described in the following sections.

7.3.1 Word Matching Model

This model is based on two classifiers; a word spotting classifier based on an analytical model (C_A) and a word classifier based on a holistic model (C_H). Each classifier will assign a class (within the lexicon word classes) to each spotted word w, and these assigned classes are denoted by a_A and a_H respectively. If $a_A = a_H$, then the word is accepted and assigned to class a_A with a score equal to $score_A$. However, if $a_A \neq a_H$, then the word is be rejected. This is shown in Figure 7-1a.

7.3.2 Improved Score Word Matching

The word matching model aims to reduce the number of false positives by rejecting a spotted word, if the analytical classifier C_A and the holistic classifier C_H assigned the word w to different classes. This model takes into consideration only the score or the likelihood probability assigned by the analytical word spotting classifier, while the holistic classifier also produces a confidence value. Using both scores to evaluate the likelihood probability for a word w may improve the overall performance of the system.

Thus, if both classifiers agreed on the class, so that $a_A = a_H$, then w is spotted and assigned to class a_A, after which both $score_A$ assigned by the analytical classifier C_A and the confidence value $score_H = P(w|W_i)$ assigned by the holistic classifier C_H are used to calculate a new word score WS as follows

$$WS = \beta score_A - \gamma \ln(score_H), \tag{7.1}$$

where β and γ are load factors assigned to the cost $score_A$ of classifier C_A and the con-

fidence value $score_H$ of classifier C_H respectively. These load factors were empirically calculated using the validation documents. This process is illustrated in Figure 7-1b.

7.3.3   Score Evaluation

For both the word matching and the improved score word matching approaches, the analytical word spotting classifier C_A may give a very low score $score_{SA} \ll 1$ (i.e. low path cost) to a spotted word w and assign it to class a_{SA}. However, the words classifier C_H may assign this same word with low confidence to class a_H, where $a_A \neq a_H$. The opposite scenario may also happen. These disagreements may result in a failure to spot strong candidate words because the two classifiers could not agree on the assigned class for the spotted word.

To address this problem, the word class of the classifier with higher confidence will be assigned to the spotted word w; so if $score_A > score_H$ then the word w will be assigned to class a_A, otherwise it will be assigned to class a_H. As in the case with the improved score word matching model 7.3.2, both of the scores $score_A$ and $score_H$ are used to calculate a new word score WS using equation 7.1. This process is illustrated in Figure 7-1c.

7.4   Experiments and Results

We conducted several experiments to evaluate the three proposed verification models. These experiments are based on considering different combinations of analytical and holistic classifiers, before arriving at a conclusion.

This section reports on all the experiments conducted. Section 7.4.1 presents the experimental setup, Section 7.4.2 reports and compares the results of two word spotting systems with and without validation, where different combination of a holistic and a analytical classifiers are used for each experimental setup. Section 7.4.3 reports and compares the word recognition results from the individual classifiers.

7.4.1 Experimental Setup

Four word spotting systems were implemented as described in Section 7.2. Three of these were implemented based on the hierarchical classifier (PAW model) using the multi-layer pruning structure and is trained for the implementations SVM, RDA, and MQDF respectively, while the fourth classifier is implemented on an HMM recognizer (character model). The results of each of the word spotting systems are verified using the three proposed verification models (Section 7.3).

The analytical classifiers were trained on CENPARMI Arabic handwritten words database (Section 4.4.2), which represents the lexicon of the system under study. This database was also used to train the holistic classifiers (SVM, RDA and MQDF). Following the training process, the word spotting systems were evaluated on the CENPARMI Arabic handwritten documents database (Section 3.3.1). This database is divided into the two validation and testing sets; the validation set is used to empirically determine the load factors β and γ for the path evaluation (equation 7.1), while the rest of the documents were used to evaluate the system.

The precision-recall curve is used to show the performance of the word spotting systems with and without validation, while the Mean Average Precision (MAP) is calculated to evaluate the systems. Some precision-recall curves are shown in Section 7.4.2, while the other curves are included in Appendix A.

Finally, the word recognition rates for each of the classifiers are presented in Section 7.4.3. These rates are calculated from the number of correctly spotted and recognized keywords (CS), and also the keywords which the word spotting classifier was able to spot by finding their boundaries even if they were incorrectly recognized (SP). Then the accuracy is calculates as follows

$$accuracy = \frac{CS}{SP} \tag{7.2}$$

.

7.4.2 Performance Evaluation

Table 7.1 shows the MAP of the four word spotting systems without verification ($default$), as well as the MAPs of these systems after they were verified using the word matching model described in Section 7.3.1. The results show that verifying a word spotting system using the word matching model may improve the performance of a word spotting system. Combining the RDA implementation of the hierarchical classifier to spot words, with the three holistic classifiers (SVM, RDA and MQDF) using the word matching model has resulted in a significant increase in the MAP. However, for the SVM and MQDF implementations of the hierarchical classifier combined with the holistic classifier, the MAP's of the systems have resulted in only minor changes. Verifying the HMM word spotting system with the $MQDF$ or RDA has improved the results with a significant increase of about 5% in the MAP on the other hand, verifying the system using SVM has lowered the performance.

Hierarchical Classifier	default	Validation Classifier		
		SVM	RDA	MQDF
SVM	0.6753	0.6772	0.6663	0.6670
RDA	0.6079	0.6333	0.6379	0.6400
MQDF	0.6022	0.6016	0.6030	0.6047
HMM	0.2557	0.2297	0.3073	0.3055

Table 7.1: Performances of combinations of classifiers using word matching validation model

Figure 7-2 shows the precision-recall curves of the hierarchical classifier implemented using RDA, and the word spotting system implemented using HMM. Each of these is verified using the word matching model implemented by the holistic classifiers (SVM, RDA, and MQDF), in addition to the $default$ model. These curves give more detailed view of the results. Figure 7-2 shows that applying any holistic classifier to the RDA implementation of the hierarchical classifier results in an improved precision-recall curve. The curves are mainly improved at low recall rates because the strong rejection of the word matching model significantly reduces the number of false positives, which in turn significantly increases the precision rate based on equation 2.2. The precision-recall curve of the HMM word spotting system verified with

the SVM and without verification (default) are very similar at low recall rates, while later the verified curve significantly drops at higher recall rates. This is due to the fact that strong rejections did not significantly reduce the number of false negatives, while some true positives were rejected.
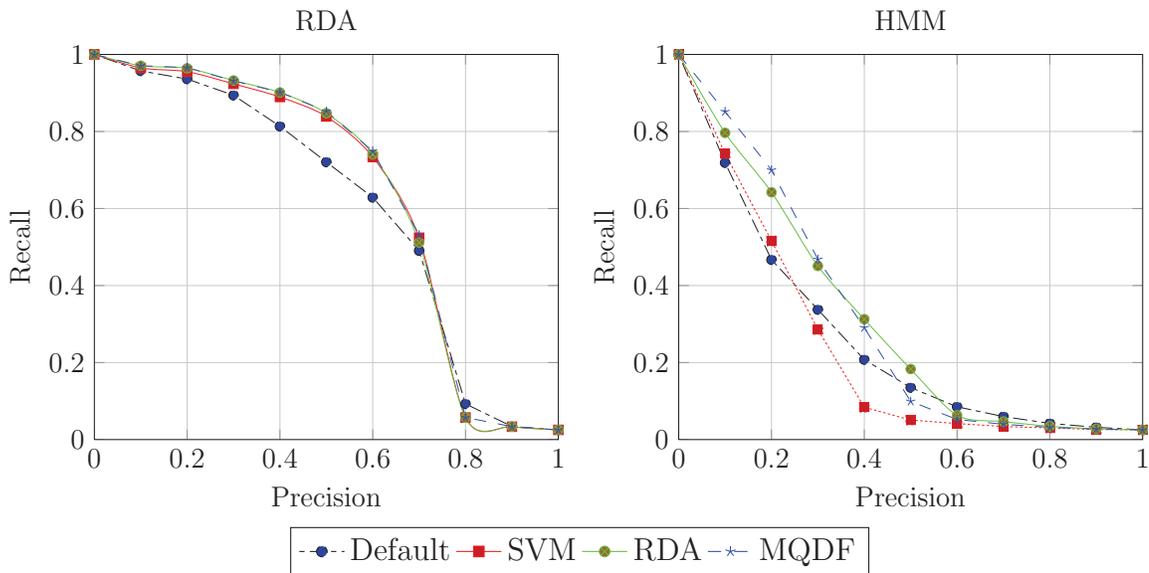

Figure 7-2: Precision-Recall Curves of hierarchical classifier implemented using RDA and HMM, and verified using the word matching model with SVM, RDA and MQDF

Figure A-1 in Appendix A shows the precision-recall curves of all combinations of classifiers based on the word matching verification model. The results show that at high precision rates the curve is always improved when the word matching model is applied. Nevertheless, the precision-recall curves of the hierarchical classifier implemented using SVM and MQDF, as well as those of the HMM word spotting system, are not always improved. This means that strong rejections may also reduce the number of true positives resulting from the non-verified word spotting system, and this will reduce the recall rate (Equation 7.2). That is the reason for the poor performance of the system at high recall rates.

Table 7.2 presents the MAPs of of the four word spotting systems with and without verification. The improved score word matching model is used to verify these word spotting systems. The results show that for all the different combinations of the analytical and holistic classifiers using the improved score word matching model, the

default performance of the word spotting system is improved. This model performs better than the word matching model, since the score calculated after the strong rejection is based on the results of two classifiers (analytical and holistic).

Hierarchical Classifier	default	Validation Classifier		
		SVM	RDA	MQDF
SVM	0.6753	0.6822	0.6748	0.6766
RDA	0.6079	0.6607	0.6640	0.6615
MQDF	0.6022	0.6322	0.6274	0.6270
HMM	0.2557	0.2580	0.3324	0.3448

Table 7.2: Performances of combinations of classifiers using improved score word matching verification model

Figure 7-3 shows the precision-recall curves of the hierarchical classifier implemented using SVM and verified using three different holistic classifiers based on the improved score word matching verification model. All the verified curves perform similarly, in that they all outperform the non-verified model. This is because each verification requires the two classifiers should agree on the assigned class, and also the resulting score is re-evaluated based on the score or the probabilities calculated from the two approaches (analytic and holistic). This will re-rank weak and strong candidates based on both the holistic and analytical scores.

Figure A-2 in Appendix A shows more curves where regardless which combination of classifiers is used the curves are improved. A comparison can be made between the HMM curves verified using the word matching model with those using the improved score word matching model. The latter model has more significant improvement at low recall rates, while the performances of the two model are very similar at high recall rates. This is because the scores calculated using HMM have very large negative values at high recall rates, so that the score re-evaluation equation can not significantly improve on these values.

Table 7.3 presents the calculated MAPs when a word spotting system based on analytical approach is combined with a holistic classifier using the score evaluation model. The previously described combinations are also tested for this model. The results show that using this model will significantly improve the performance of a
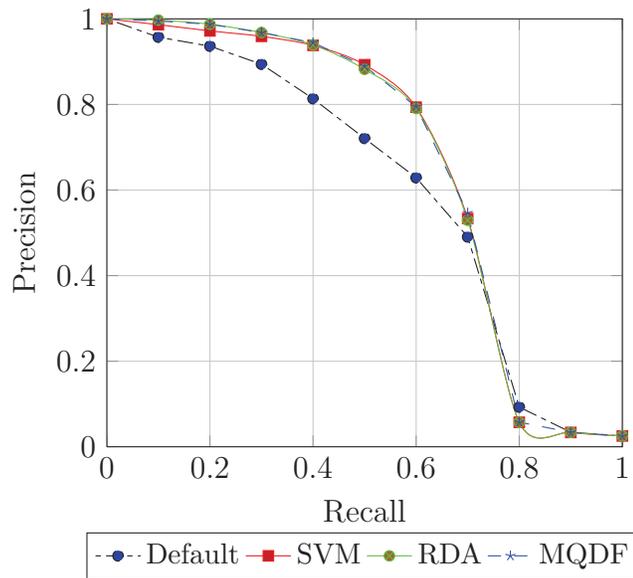
Figure 7-3: Precision-Recall Curves of hierarchical classifier implemented using SVM and verified using the improved score word matching model with SVM, RDA and MQDF

word spotting system. A comparison of Table 7.3 with Tables 7.1 and 7.2 shows that the score evaluation model outperformed the two other proposed models. The reason is the score evaluation model does not reject any spotted word. Instead it re-evaluates the score by combining the results of an analytical word spotting system with that of a holistic classifier, so that weak samples given low scores by both classifiers will have an even lower resulting score, while the samples that are not correctly classified by one of the classifiers can be classified based on another classifier with higher confidence.

It is worth noting that the scores given by HMM word spotting system and the holistic classifier are not comparable. For this reason, when the HMM word spotting system is verified using the score evaluation model, only the score is re-evaluated while the spotted word is always assigned to class determined by the HMM system. The result is that the improved score word matching model has higher performance over the score evaluation model when the HMM classifier is verified by RDA and $MQDF$.

Figure 7-4 shows the precision-recall curves of a hierarchical classifier implemented using RDA when combined with each of the three holistic classifiers (SVM, RDA, and

Hierarchical	default	Validation Classifier		
Classifier		SVM	RDA	MQDF
SVM	0.6753	0.6936	0.6916	0.6937
RDA	0.6079	0.6758	0.6732	0.6792
MQDF	0.6022	0.6415	0.6374	0.6344
HMM	0.2557	0.2737	0.2961	0.2998

Table 7.3: Performances of combinations of classifiers using score evaluation validation model
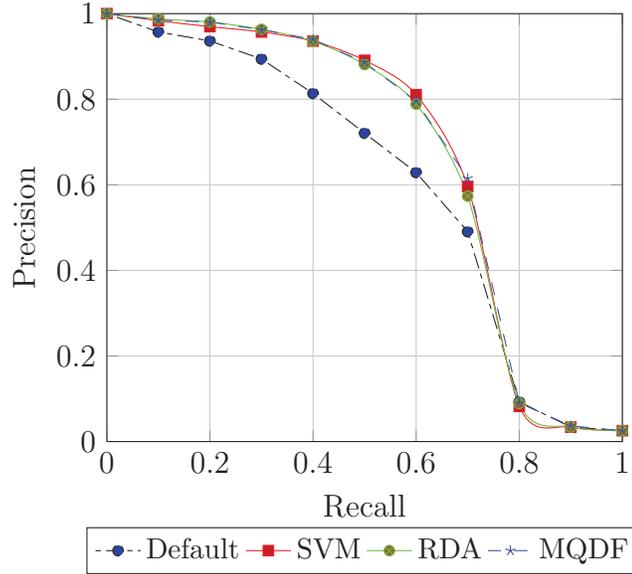

Figure 7-4: Precision-Recall Curves for the hierarchical classifier implemented using RDA and the verified using the score evaluation model with different holistic classifier

MQDF) based on the score evaluation model. The curves show that verifying the word spotting classifier with any holistic classifier significantly improves the precision-recall curve, with more than 7% and 17% increase in the MAP and the precision at 50.0% recall respectively. Figure A-3 in Appendix A shows that this model can always significantly increase the performance of any word spotting system.

A series of experiments have been conducted to compare the performances of the three verification models with the non-verified (default) model, to determine the effectiveness of these approaches. Figures A-4, A-5 and A-6 in Apendex A show the precision-recall curves of the four models for each combination of an analytical and a holistic classifier. The results of four of such systems are shown in Figure 7-5. The SVM-MQDF plot shows the precision-recall curves when the SVM classifier is used

to spot words, that are later verified using an MQDF classifier. The three verification models were compared with the non-verified (default) model. The precision-recall curves show that all the verified models have better performance at low recall rates, and at 50% recall the precision increases from 87.73% for the non-verified model to 90.00%, 92.00%, 91.87% for the word matching, improved word matching and score evaluation verification models respectively. However, the MAP of the word matching verification model drops slightly from 0.6753 to 66.70, while it increases for the other models.
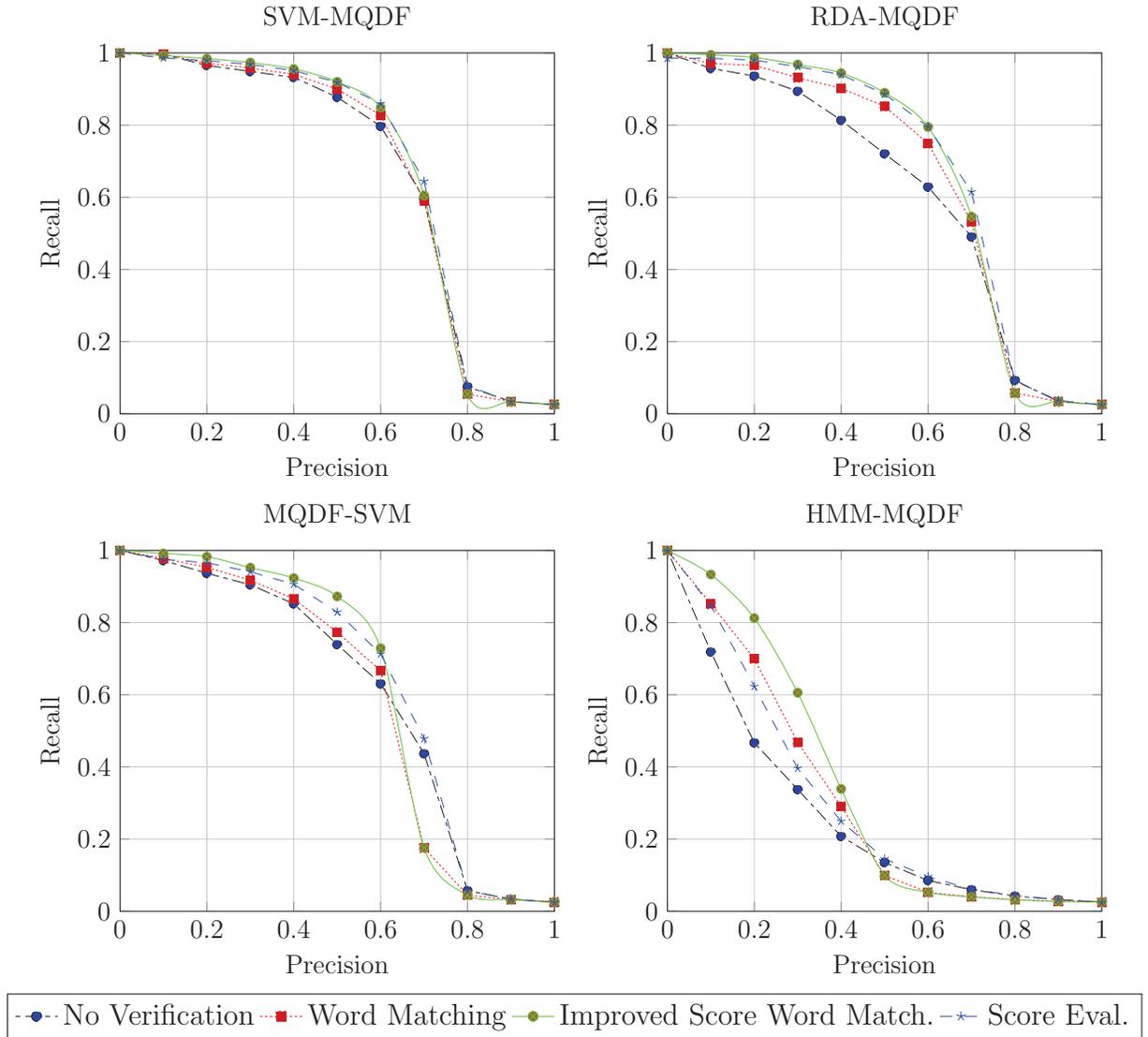

Figure 7-5: Comparison of the three validation models and the non-validated word spotting using selected combinations of classifiers

The RDA-MQDF plot compares four models based on the hierarchical classifier implemented with the RDA and verified using the MQDF holistic classifier. The results show that all the verification models can significantly increase the performance of the non-verified (default) word spotting system; the MAP increases by 3.21%, 5.36% and 7.13% when the word spotting system is verified using the word matching, improved word matching and score evaluation verification models respectively.

The MQDF-SVM plot shows that implementing the hierarchical classifier using MQDF and verifying the system using an SVM classifier, improves the performance of the system at low recall rates. At 50% recall rate the precision increases from 73.93% to 77.32%, 87.32% and 82.90% respectively for the word matching, improved score word matching and score evaluations models. As in the case of the SVM-MQDF model the MAP of the word matching model decreases slightly by 0.06%, while the MAP increases significantly with the other two models where the score is re-calculated based on results of both classifiers.

The HMM-RDA plot compares the performance of the HMM based word spotting system when it is verified with the three different models based on an RDA holistic classifier and the non-verified (default) model. The results show that verifying the HMM classifier with the RDA classifier will always improve the performance, and the MAP increases by 7.67% when the system is verified using the improved score word matching model. This verification model using RDA classifier also has a significant increase in performance at low recall rates, where the precision increases from 46.70% to 70.00%, 81.32% and 62.32% at 20% recall for the word matching, improved score word matching and the score evaluation models respectively.

Finally, it should be noted that for all models, the improved score word matching model outperforms the other models at low recall rates. This is because the model can reduce the number of false positives, and it also re-calculates the score of each word and re-rank the accepted words based on two classifiers of different natures.

7.4.3 Analysis of Word Recognition Results

This experiment was applied to the word spotting systems based on the hierarchical classifier. To further analyze the performance of the combined classifiers, the number of misclassified samples of each classifier on the spotted words has been determined. Consequently, only the words spotted by the analytical classifier are taken into account, since the analytical classifier is the one that spots words, while the holistic classifier is mainly used to verify the results. Table 7.4 presents numbers of misclassified samples of the word classifiers on the spotted words based on Equation 7.2.

Classifier	No. of Spotted Words	Number of misclassified samples			
		Hierarchical Classifier	Holistic Classifier		
			SVM	RDA	MQDF
SVM	2073	130	10	11	9
RDA	2169	269	13	19	15
MQDF	2085	275	18	18	16

Table 7.4: Word recognition rates of the hierarchical classifier vs. holistic classifier

The results show that the holistic classifiers always outperform the analytical classifiers. This proves that the holistic model has higher recognition performance, yet we need the analytical classifiers for the word spotting process.

7.5 Conclusions

In this chapter, three validation models for two learning-based word spotting systems that spots Arabic handwritten words from documents were proposed. One system integrates partial segmentation with a hierarchical classifier to spot words according to a path evaluation procedure, while the other is based on character HMMs.

The two systems first spot and find the boundaries of the lexicon words, and then each system uses a holistic classifier trained on the lexicon words (instead of PAWs or characters) to validate the spotted words. Thus, the word spotting system will accept and reject words by combining an analytical classifier with a holistic one.

Our validation models produced promising results when the two Arabic word spotting systems (hierarchical classifier and HMMs) were validated. The score evaluation model has shown significant improvement on the system when it is applied to validate the hierarchical classifier, in which the model outperformed the other models. While, the improved score word matching model has a consistent improvement on the results of all the validated word spotting systems.

Chapter 8

Conclusions and Future Work

It is the purpose of this thesis to design and implement a multi-writer learning-based word spotting system, that can overcome the lack of boundaries problem that appears in Arabic handwritten text. Several Arabic handwritten document recognition and analysis systems have been proposed and implemented in this thesis to address this problem, and the results have been presented. While the proposed system has addressed the main problem, some challenges nevertheless do remain. For this reason, several suggestions for improvement and future research are presented in Section 8.2.

8.1   Concluding Remarks

The automatic processing, analysis and recognition of documents handwritten in Arabic are challenging tasks, for reasons already described in this thesis. In order to achieve satisfactory performance of these tasks, we have designed and implemented a learning-based word spotting system that is capable of processing Arabic script from multiple writers. This system incorporates the processes essential for an accurate extraction of text lines from a document image (Chapter 3), partial segmentation of each text line into Parts of Arabic Words (PAWs) and the recognition of these PAWs (Chapter 5). In Chapter 6, a hierarchical classifier is proposed to spot Arabic words, and detailed experimental results of this process are reported.

A hierarchical classifier is proposed to solve the lack of boundaries problem that

occurs in Arabic handwritten text. This classifier integrates the partial segmentation of Arabic handwritten words (to sub-word or PAWs) with statistical language models to spot words. Different internal structures have been proposed and implemented using SVM, RDA and MQDF classifiers. Among the findings is that, training the internal classifier with more PAW classes significantly can improve the performance of the hierarchical classifier. The results obtained are promising, and it shows the proposed system can outperform a state-of-the-art word spotting system [25].

In order to verify the results of word spotting, an ensemble of classifiers based on different paradigms is utilized (Chapter 7). For the verification process, three different models (word matching, improved score word matching, and score evaluation) have been designed and implemented (Section 7.3). The experimental results, detailed performance evaluations and analyses are presented in Section 7.4. The finding is that, implementing a word spotting system based on an analytical approach, and then verifying the results of the word spotting system using a holistic approach have resulted in improved performance.

8.2 Future Work

This section discusses directions and methods for future refinements and improvement, and also suggests some proposals for future work.

A learning-based method can be applied to refine the thresholds and parameters of the proposed text line extraction algorithm. It is also possible to improve the performance of the algorithm, by applying an effective technique to separate the text lines at the detected critical regions.

The proposed partial segmentation algorithm can also be refined and improved by applying a learning-based approach to segment words into PAWs instead of applying heuristics. This is particularly pertinent for Arabic document analysis and recognition systems, as many of them are based on the PAW model.

Segmentation-free approaches have produced promising results in the literature of handwriting recognition, where implicit segmentation is integrated with the recogni-

tion to search for words. This approach can be applied to our system to replace the explicit segmentation of a text line into PAWs where some false negatives would result from erroneous segmentations. Thus, over-segmenting a text line into small strokes, then applying a dynamic programming algorithm to re-construct words based on PAWs, while keeping the PAW based hierarchical structure of the system, can possibly improve the performance.

Statistical classifiers based on linear discriminant analysis have shown a potential for effective performance in term of recognition rate and processing time. It would be logical to devise procedures to refine and improve the transformed probabilities of the MQDF and RDA, in an attempt to improve the performance of the proposed system.

Other types of features and classifiers can be applied to the internal structure of the proposed classifier, which may result in improving the performance of the system. Different features that are more applicable to the Arabic handwriting such as gradient features, could be applied to the reference system to improve the performance. Effective procedures to normalize Arabic handwritten text lines could also be examined in the process.

The proposed verification methods can be applied to other scripts such as Latin-based ones.

Appendix A

Precision-Recall Curves

Figure A-1 compares the non-validated model with the word matching validation model using three different holistic classifiers (SVM, RDA, and MQDF).
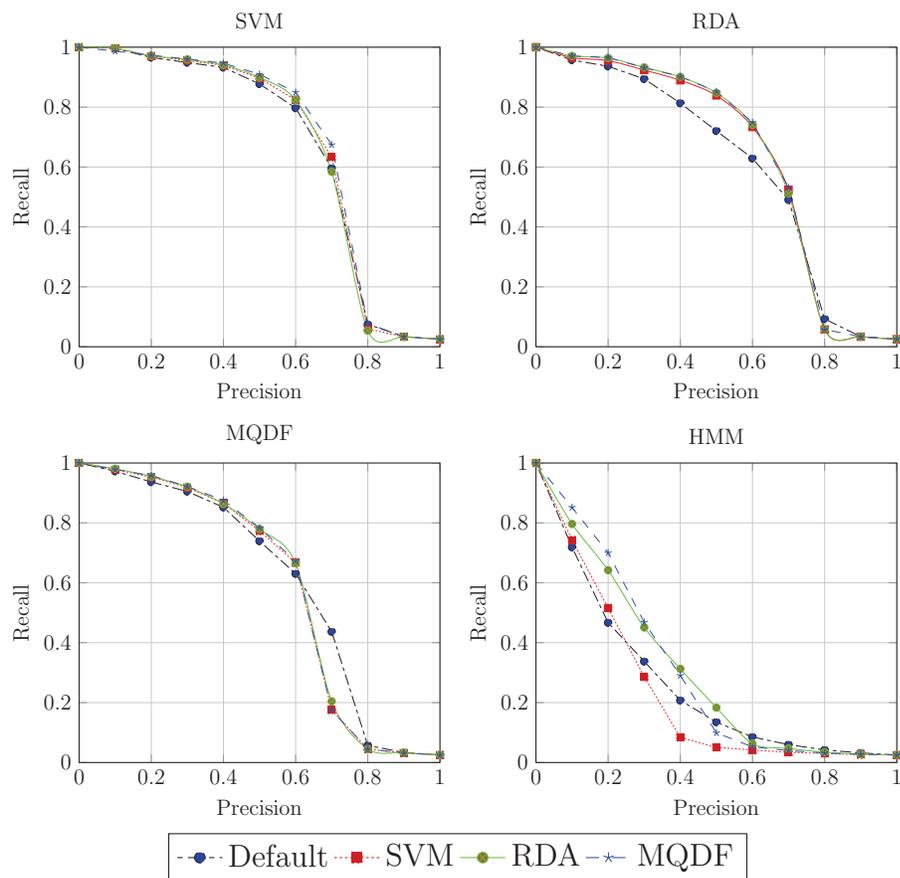

Figure A-1: Comparison of word spotting system with and without verification using the word matching model

Figure A-2 shows the performances of the non-validated model with the improved score word matching validation model incorporating three different holistic classifiers (SVM, RDA, and MQDF).
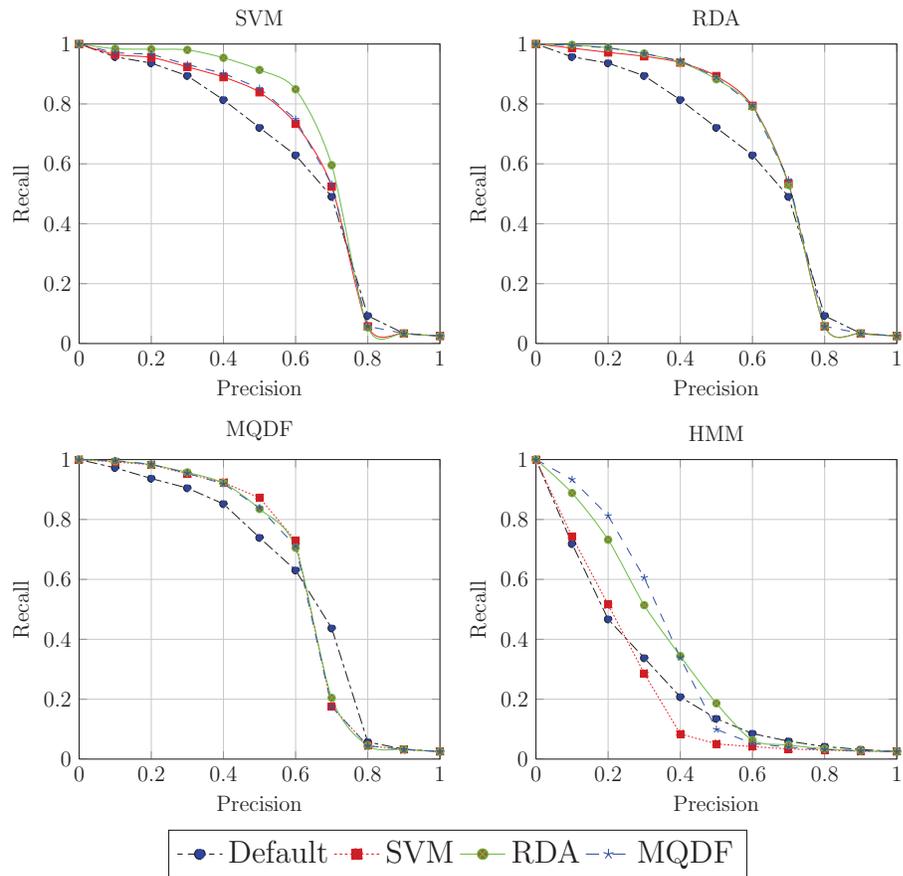

Figure A-2: Comparison of word spotting system with and without verification using the improved score word matching model

Figure A-3 compares the non-validated model with the score evaluation validation model using three different holistic classifiers (SVM, RDA, and MQDF).
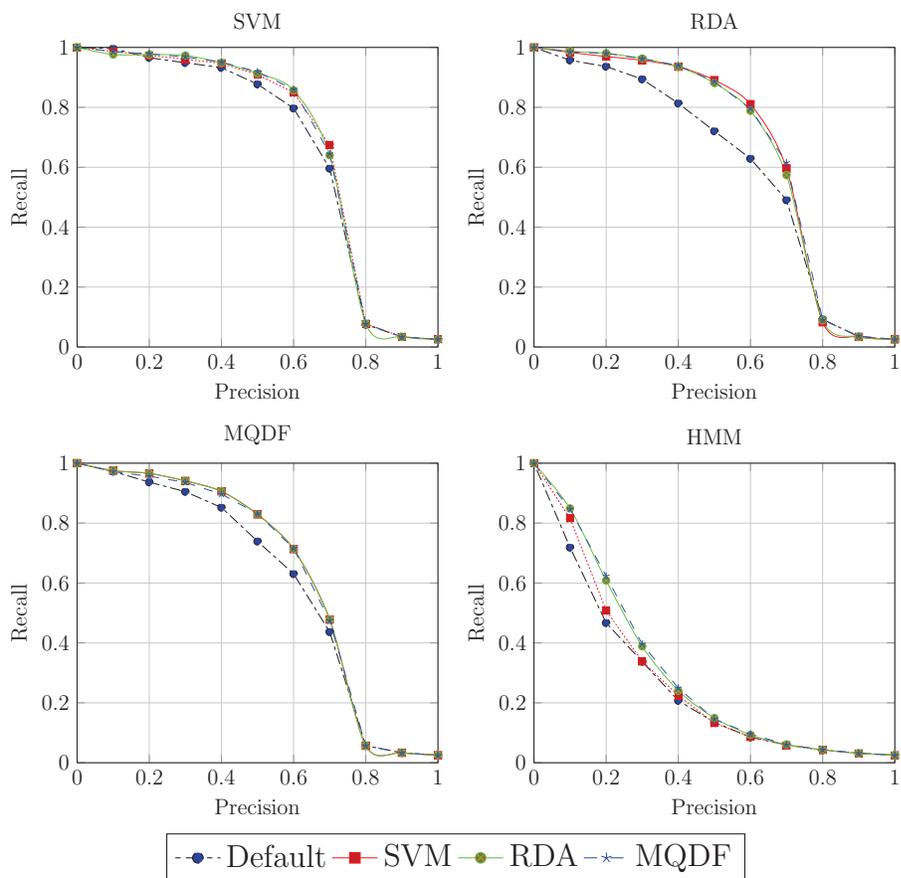
Figure A-3: Comparison of word spotting system with and without verification using the score evaluation model
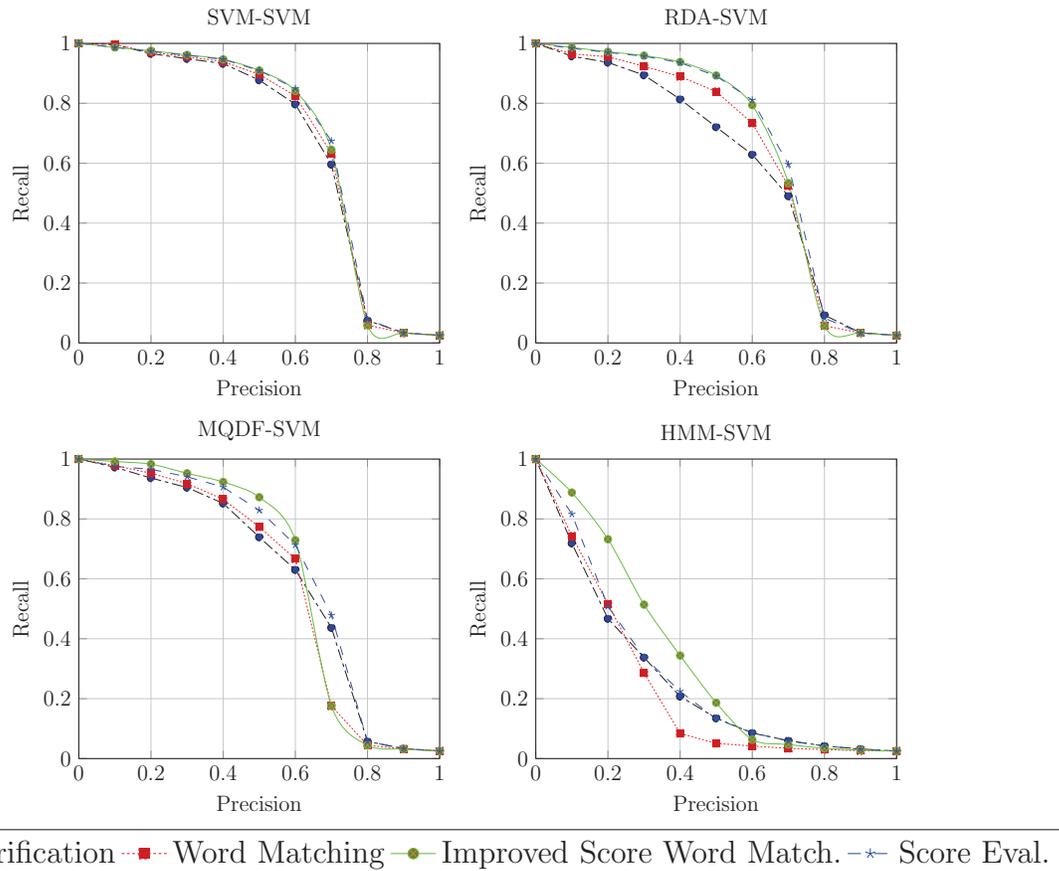
Figure A-4: Comparison of the three validation models with the non-validated word spotting system using combinations verified using SVM classifier
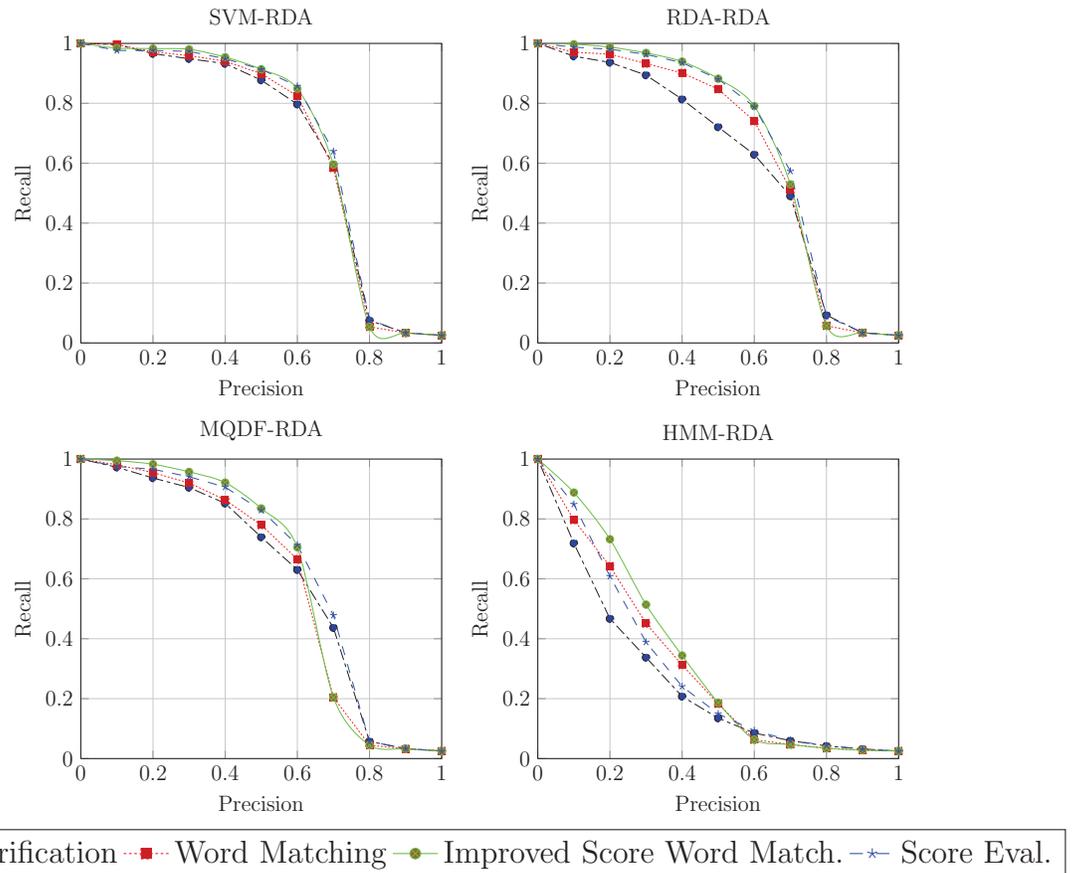
Figure A-5: Comparison of the three validation models with the non-validated word spotting system using combinations verified using RDA classifier
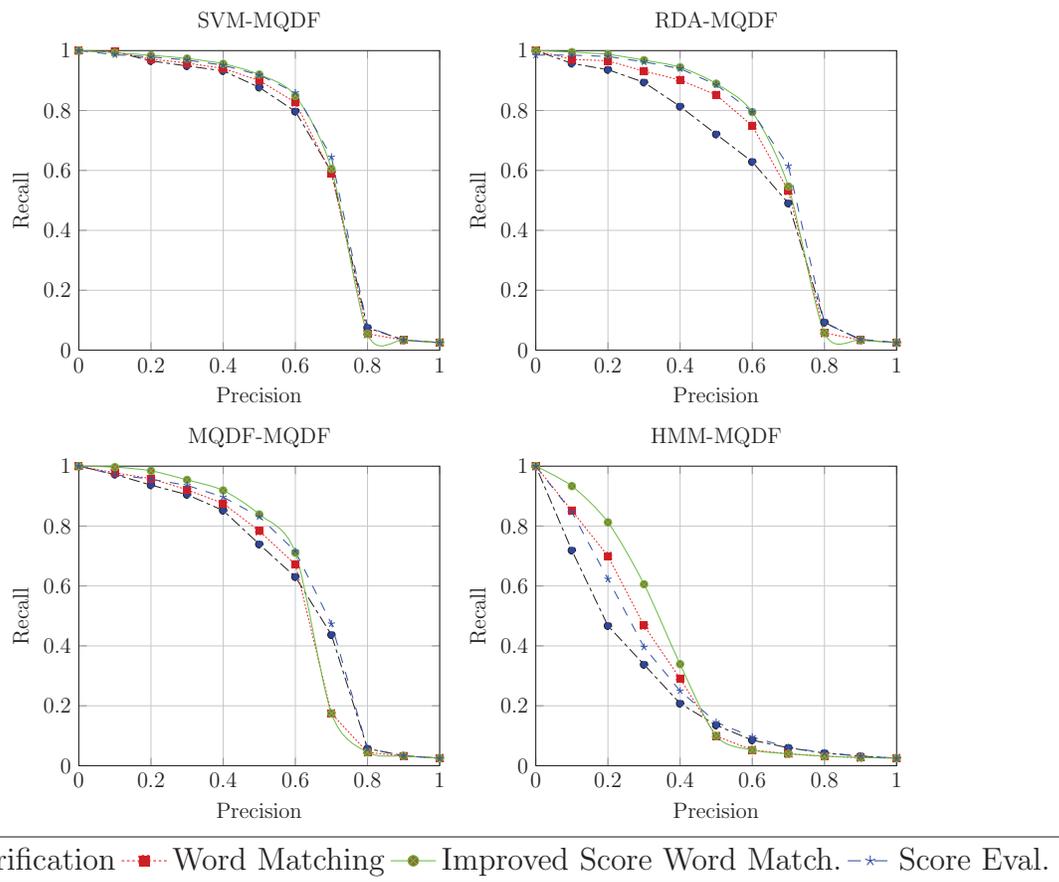
Figure A-6: Comparison of the three validation models withs the non-validated word spotting system using combinations verified using MQDF classifier

Appendix B

List of Abbreviations

ASR	Automatic Speech Recognition
BLSTM-NN	Bidirectional Long Short Term Memory Neural Networks
CC	Connected Components
CTC	Connectionist Temporal Classification
DFT	Discrete Fourier Transform
DTW	Dynamic Time Warping
FAR	False Alarm Rate
FN	False Negative
FP	False Positive
HMM	Hidden Markov Models
LSTM	Long Short-Term Memory
MAP	Mean Average Precision
MLP	Multi-Layer Perceptron
MQDF	Modified Quadratic Discriminant Function
MS	Maching Score
MST	Minimum Spanning Trees
OCR	Optical Character Recognition
OOV	Out Of Vocabulary
PAW	Pieces of Arabic Word

PCA   Principal Component Analysis

PR   Precision Rate

QDF   Quadratic Discriminant Function

RBF   Radial Basis Function

RDA   Regularized Discriminant Analysis

RR   Recall Rate

SVM   Support Vector Machines

TN   True Negative

TP   True Positive

WER   Word Error Rate

WFST   Weighted Finite State Transducent

WST   Word Shape Token

References

[1] Database. Handwritten Arabic proximity database, language and media processing laboratory, Univeristy of Maryland, College Park, Washington D.C. http://lampsrv02.umiacs.umd.edu/projdb/project.php.

[2] R. Manmatha, Chengfeng Han, and Edward M. Riseman. Word spotting: A new approach to indexing handwriting. In *CVPR conference*, pages 631–637, 1996.

[3] José A. Rodríguez-Serrano and Florent Perronnin. Local gradient histogram features for word-spotting in unconstrained handwritten documents. In *Proc. of the 11th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pages 7–12, 2008.

[4] José A. Rodríguez-Serrano and Florent Perronnin. Score normalization for HMM-based word spotting using universal background model. In *Proc. of the 11th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pages 82–87, 2008.

[5] Bin Zhang, Sargur N. Srihari, and Chen Huang. Word image retrieval using binary features. In *Document Recognition and Retrieval*, pages 45–53, 2004.

[6] Toni M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *CVPR conference (2)*, pages 521–527, 2003.

[7] Tomasz Adamek, Noel E. O'Connor, and Alan F. Smeaton. Word matching using single closed contours for indexing handwritten historical documents. *IJDAR*, 9(2-4):153–165, 2007.

[8] Toufik Sari and Abderrahmane Kefali. A search engine for Arabic documents. In *Actes du dixième Colloque International Francophone sur l'Écrit et le Document*, pages 97–102, 2008.

[9] Raid Saabni and Jihad El-Sana. Keyword searching for Arabic handwritten documents. In *Proc. 11th International Conference on Frontiers in Handwriting Recognition, (ICFHR)*, pages 271–277, 2008.

[10] Mohamed Cheriet and Reza Farrahi Moghaddam. *Guide to OCR for Arabic Scripts*, chapter A Robust Word Spotting System for Historical Arabic Manuscripts, pages 453–484. Springer, 2012.

[11] Safwan Wshah, Gaurav Kumar, and Venu Govindaraju. Script independent word spotting in offline handwritten documents based on hidden Markov models. In *13th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pages 14–18, 2012.

[12] Anurag Bhardwaj, Damien Jose, and Venu Govindaraju. Script independent word spotting in multilingual documents. In *Proceedings of the 2nd International Workshop on Cross Lingual Information Access*, pages 48–54, 2008.

[13] Victor Lavrenko, Toni M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL)*, pages 278 – 287, 2004.

[14] Jaety Edwards, Yee Whye, Teh David, Forsyth Roger, Bock Michael Maire, and Grace Vesom. Making Latin manuscripts searchable using GHMM's. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems*, pages 385–392, 2005.

[15] J. Chan, C. Ziftci, and D. Forsyth. Searching off-line Arabic documents. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, CVPR, pages 1455–1462, 2006.

[16] Aleksander Kolcz, Joshua Alspector, and M. Augusteijn. A line-oriented approach to word spotting in handwritten documents. *Pattern Anal. Appl.*, 3(2):153–168, 2000.

[17] R. Manmatha, Chengfeng Han, E. M. Riseman, and W. B. Croft. Indexing handwriting using word matching. In *Proceedings of the first ACM international conference on Digital libraries*, DL '96, pages 151–159, 1996.

[18] R. Manmatha and T. Rath. Indexing handwritten historical documents - recent progress. In *Proceedings of the Symposium on Document Image Understanding*, SDIUT-03, pages 77–85, 2003.

[19] Yann Leydier, Frank Lebourgeois, and Hubert Emptoz. Text search for Medieval manuscript images. *Pattern Recognition*, 40(12):3552–3567, 2007.

[20] José A. Rodríguez-Serrano and Florent Perronnin. Handwritten word-spotting using hidden Markov models and universal vocabularies. *Pattern Recognition*, 42(9):2106–2116, 2009.

[21] Sargur N. Srihari and Gregory R. Ball. Language independent word spotting in scanned documents. *Lecture Notes in Computer Science - LNCS*, 5362:134–143, 2008.

[22] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. HMM-based word spotting in handwritten documents using subword models. In *ICPR*, pages 3416–3419, 2010.

[23] Volkmar Frinken, Andreas Fischer, R. Manmatha, and Horst Bunke. A novel word spotting method based on recurrent neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(2):211–224, 2012.

[24] Khurram Khurshid, Claudie Faure, and Nicole Vincent. A novel approach for word spotting using merge-split edit distance. In Xiaoyi Jiang and Nicolai Petkov, editors, *CAIP*, volume 5702 of *Lecture Notes in Computer Science*, pages 213–220. Springer, 2009.

[25] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. Lexicon-free handwritten word spotting using character HMMs. *Pattern Recogn. Lett.*, 33(7):934–942, 2012.

[26] I. S. I. Abuhaibaa, M. J. J. Holtb, and S. Dattab. Recognition of off-line cursive handwriting. *Computer Vison and Image Understanding*, 71(1):19–38, 1998.

[27] Reza Moghaddam and Mohamed Cheriet. Application of multi-level classifiers and clustering for automatic word spotting in historical document images. In *Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 511–515, 2009.

[28] José A. Rodríguez-Serrano and Florent Perronnin. A model-based sequence similarity with application to handwritten word-spotting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2108–2120, 2012.

[29] Raid Saabni and Alex Bronstein. Fast key-word searching via embedding and Active-DTW. In *Proc. of the 11th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 68–72, 2011.

[30] Muralikrishna Sridha, Dinesh Mandalapu, and Mehul Patel. Active-DTW : A Generative Classifier that combines Elastic Matching with Active Shape Modeling for Online Handwritten Character Recognition. In *Proc. of the 10th Int. Workshop on Frontiers in Handwriting Recognition*, pages 193–196, 2006.

[31] Ediz Şaykol, Ali Kemal Sinop, Uğur Güdükbay, Özgür Ulusoy, and A. Enis Çetin. Content-based retrieval of historical Ottoman documents stored as textual images. *IEEE Trans. on Image Processing*, 13(3):314–325, 2004.

[32] S.A. Shahab, Wasfi G. Al-Khatib, and Sabri A. Mahmoud. Computer aided indexing of historical manuscripts. In *Proceedings of International Conference on Computer Graphics, Imaging and Vision (CGIV)*, pages 151–159, 2006.

[33] Sargur Srihari, Harish Srinivasan, Pavithra Babu, and Chetan Bhole. Handwritten Arabic word spotting using the CEDARABIC document analysis system. In

Proc. Symposium on Document Image Understanding Technology (SDIUT-05), College Park, MD, pages 123–132, 2005.

[34] Muna Khayyat, Louisa Lam, and Ching Y. Suen. Learning-based word spotting system for Arabic handwritten documents. *Pattern Recognition*, 47(3):1021 – 1030, 2014.

[35] Toufik Sari, Labiba Souici, and Mokhtar Sellami. Off-line handwritten Arabic character segmentation algorithm: ACSA. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, pages 452–456, Washington, DC, USA, 2002.

[36] G. Ball, S. N. Srihari, and H. Srinivasan. Segmentation-based and segmentation-free approaches to Arabic word spotting. In *Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 53–58, 2006.

[37] Y. Fujisawa M. Shi, T. Wakabayashi, and F. Kimura. Handwritten numeral recognition using gradient and curvature of gray scale image. *Pattern Recognition*, 35(10):2051–2059, 2002.

[38] Muna Khayyat, Louisa Lam, and Ching Y. Suen. Verification of hierarchical classifier results for handwritten Arabic word spotting. In *Proc. 12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 572–576, 2013.

[39] Muna Khayyat, Louisa Lam, and Ching Y. Suen. Arabic handwritten word spotting using language models. In *Proc. of the 13th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pages 43–48, 2012.

[40] Fadi Biadsy, Raid Saabni, and Jihad El-Sana. Segmentation-free online Arabic handwriting recognition. *IJPRAI*, 25(7):1009–1033, 2011.

[41] Muna Khayyat, Louisa Lam, Ching Y. Suen, Fei Yin, and Cheng-Lin Liu. Arabic handwritten text line extraction by applying an adaptive mask to morphological dilation. In *Document Analysis Systems*, pages 100–104, 2012.

[42] Jayant Kumar, Wael Abd-Almageed, Le Kang, and David Doermann. Handwritten Arabic text line segmentation using affinity propagation. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS '10, pages 135–142, 2010.

[43] Zhixin Shi, Srirangaraj Setlur, and Venu Govindaraju. A steerable directional local profile technique for extraction of handwritten Arabic text lines. In *Proceedings of the 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 176–180, 2009.

[44] Fei Yin and Cheng-Lin Liu. Handwritten text line extraction based on minimum spanning tree clustering. *Pattern Recognition*, 42(12):3169–3183, 2009.

[45] Syed Saqib Bukhari, Faisal Shafait, and Thomas M. Breuel. Script-independent handwritten textlines segmentation using active contours. In *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 446–450, 2009.

[46] Yi Li, Yefeng Zheng, David Doermann, Stefan Jaeger, and Yi Li. Script-independent text line segmentation in freestyle handwritten documents. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(8):1313–1329, 2008.

[47] Majid Ziaratban and Karim Faez. An adaptive script-independent block-based text line extraction. In *ICPR*, pages 249–252, 2010.

[48] Nazih Ouwayed, Abdel Belaïd, and François Auger. General text line extraction approach based on locally orientation estimation. In *proc. SPIE7534 Document Recognition and Retrieval XVII*, volume 75340B, pages 1–10, 2010.

[49] Nobuyuki Otsu. A Threshold Selection Method from Gray-level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[50] Alex Graves and Jrgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information*

Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, pages 545–552, 2008.

[51] José Ruiz-Pinales, René Jaime-Rivas, and María José Castro-Bleda. Holistic cursive word recognition based on perceptual features. *Pattern Recogn. Lett.*, 28(13):1600–1609, October 2007.

[52] Jin Chen, Huaigu Cao, Rohit Prasad, Anurag Bhardwaj, and Prem Natarajan. Gabor features for offline Arabic handwriting recognition. In *Proceedings of the 9th International Workshop on Document Analysis Systems (DAS)*, pages 53 – 58, 2010.

[53] Sofiene Haboubi, Samia Maddouri, Noureddine Ellouze, and Hailkal El-Abed. Invariant primitives for handwritten Arabic script: A contrastive study of four feature sets. *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, pages 691–697, 2009.

[54] Haikal El Abed and Volker Märgner. Comparison of different preprocessing and feature extraction methods for offline recognition of handwritten Arabic words. *Document Analysis and Recognition, International Conference on*, 2:974–978, 2007.

[55] Marc-Peter Schambac, Jörg Rottlan, and Théophile Alar. How to convert a Latin handwriting recognition system to Arabic. In *Procceedings of the 11th Int. Conference on Frontiers in Handwriting Recognition*, pages 265–270, 2008.

[56] Haikal El Abed and Volker Märgner. How to improve a handwriting recognition system. In *ICDAR*, pages 1181–1185, 2009.

[57] Abdelkarim Elbaati, Houcine Boubaker, Monji Kherallah, Abdellatif Ennaji, Haikal El Abed, and Adel M. Alimi. Arabic handwriting recognition using restored stroke chronology. In *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, pages 411–415, 2009.

[58] Philippe Dreuw, Stephan Jonas, and Hermann Ney. White space models for offine Arabic handwriting recognition. In *Proceedings of the International Conference on Pattern Recognition ICPR*, pages 1–4, 2008.

[59] Safwan Wshah, Venu Govindaraju, Yanfen Cheng, and Huiping Li. A novel lexicon reduction method for Arabic handwriting recognition. In *Proceedings of the 20th International Conference on Pattern Recognition ICPR*, pages 2865–2868, 2010.

[60] Mahdi Hamdani, Haikal El Abed, Monji Kherallah, and Adel M. Alimi. Combining multiple HMMs using on-line and off-line features for off-line Arabic handwriting recognition. In *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, pages 201–205, 2009.

[61] Cheng-Lin Liu, Y.J. Liu, and R.W. Dai. Preprocessing and statistical/structural feature extraction for handwritten numeral recognition. *Progress of Handwriting Recognition, A.C. Downton and S. Impedovo (Eds.), World Scientific*, pages 161–168, 1997.

[62] C.-L. Liu, M. Koga, and H. Fujisawa. Gabor feature extraction for character recognition: Comparison with gradient features. In *Proceedings of 8th Int. Conference on Document Analysis and Recognition*, pages 121–125, 2005.

[63] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2 edition, 2002.

[64] John Shawe-Taylor and Nello Cristianini. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[65] Jerome H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[66] Cheng-Lin Liu. Classifier combination based on confidence transformation. *Pattern Recognition*, 38(1):11–28, 2005.

[67] Tom M. Mitchell. *Machine learning.* McGraw Hill series in computer science. McGraw-Hill, 1997.

[68] Huda Alamri, Javad Sadri, Chen Y. Suen, and Nicola Nobile. A novel comprehensive database for Arabic off-line handwriting recognition. In *Proceedings 11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 664–669, 2008.

[69] Mario Pechwitz, Samia Snoussi Maddouri, Volker Märgner, Noureddine Ellouze, and Hamid Amiri. IFN/ENIT - database of handwritten Arabic words. In *Proceedings of CIFED 2002*, pages 129–136, 2002.

[70] U.-V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 15:65–90, 2001.

[71] Sriganesh Madhvanath and Venu Govindaraju. The role of holistic paradigms in handwritten word recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(2):149–164, February 2001.

[72] Mario Pechwitz and Volker Maergner. HMM based approach for handwritten Arabic word recognition using the IFN/ENIT - database. ICDAR '03, pages 890–894, 2003.

[73] Jawad H AlKhateeb, Jinchang Ren, Jianmin Jiang, and Husni Al-Muhtaseb. Offline handwritten Arabic cursive text recognition using hidden Markov models and re-ranking. *Pattern Recogn. Lett.*, 32(8):1081–1088, June 2011.

[74] Ramy Al-Hajj Mohamad, Laurence Likforman-Sulem, and Chafic Mokbel. Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(7):1165–1177, July 2009.

[75] A. Benouareth, A. Ennaji, and M. Sellami. Semi-continuous HMMs with explicit state duration for unconstrained Arabic word modeling and recognition. *Pattern Recogn. Lett.*, 29(12):1742–1752, September 2008.

[76] Katsumi Marukawa, Tao Hu, Hiromichi Fujisawa, and Yoshihiro Shima. Document retrieval tolerating character recognition error - evaluation and application. *Pattern Recognition*, 30(8):1361 – 1371, 1997.

[77] Cheng-Lin Liu, Masashi Koga, and Hiromichi Fujisawa. Lexicon driven segmentation and recognition of handwritten character strings for Japanese address reading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1425–1437, 2002.

[78] Heng Zhang and Cheng-Lin Liu. A lattice-based method for keyword spotting in online Chinese handwriting. In *Proc. 11th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1064–1068, 2011.

[79] Kenji Suzuki, Isao Horiba, and Noboru Sugie. Linear-time connected-component labeling based on sequential local operations. *Computer Vision and Image Understanding*, 89(1):1–23, 2003.

[80] Suhan Chowdhury, Utpal Garain, and Tanushyam Chattopadhyay. A Weighted Finite-State Transducer (WFST)-based language model for online Indic script handwriting recognition. In *Proc. 11th International Conference on Document Analysis and Recognition (ICDAR)*, pages 599–602, 2011.

[81] Qiu-Feng Wang, Fei Yin, and Cheng-Lin Liu. Integrating language model in handwritten chinese text recognition. In *Proc. 10th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1036–1040, 2009.

[82] Yan Jiang, Xiaoqing Ding, Qiang Fu, and Zheng Ren. Context driven Chinese string segmentation and recognition. *LNCS*, 4109:127–135, 2006.

[83] Andreas Fischer, Volkmar Frinken, Hosrt Bunke, and Ching Y. Suen. Improving HMM-based keyword spotting with character language models. In *Proc. 12th In-*

117

ternational Conference on Document Analysis and Recognition (ICDAR), pages 506–510, 2013.

[84] Jashua T. Goodman. A bit of progress in language modeling: Extended version technical report. Technical Report MSR-TR 2011-72, Microsoft Research, 2001.

[85] Yann Leydier, Frank Lebourgeois, and Hubert Emptoz. Omnilingual segmentation-freeword spotting for ancient manuscripts indexation. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, ICDAR '05, pages 533–537, Washington, DC, USA, 2005. IEEE Computer Society.

[86] José A. Rodríguez-Serrano and Florent Perronnin. Handwritten word-spotting using hidden Markov models and universal vocabularies. *Pattern Recogn.*, 42(9):2106–2116, September 2009.

[87] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[88] Francisco Zamora-Martínez, María José Castro-Bleda, Salvador España Boquera, and Jorge Gorbe. Improving isolated handwritten word recognition using a specialized classifier for short words. In *Proceedings of the Current topics in artificial intelligence, and 13th conference on Spanish association for artificial intelligence*, CAEPIA'09, pages 61–70, Berlin, Heidelberg, 2010. Springer-Verlag.

[89] Sriganesh Madhvanath and Venu Govindaraju. The role of holistic paradigms in handwritten word recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(2):149–164, February 2001.

[90] Richard G. Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(7):690–706, July 1996.

[91] James L Mcclelland. Preliminary letter identification in perception of words and nonwords. *Journal of Experimental Psychology: Human Perception and Performance*, 2:80–91, 1976.

[92] G. M. Reicher. Perceptual recognition as a function of meaningfulness of stimulus material. *Journal of Experimental Psychology*, 81:275–280, 1969.

[93] G. W. Humphreys, L. J. Evett, and P. T. Quinlan. Orthographic processing in visual word identification. *Cognitive Psychology*, 22(4):517–560, 1990.

[94] Meng Shi, Yoshiharu Fujisawa, Tetsushi Wakabayashi, and Fumitah Kimura. Handwritten numeral recognition using gradient and curvature of gray scale image. *Pattern Recognition*, 35(10):2051–2059, 2002.

[95] F. Kimura, Kenji Takashina, Shinji Tsuruoka, and Yasuji Miyake. Modified quadratic discriminant functions and the application to Chinese character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):149–153, 1987.

[96] Haikal El Abed and Volker Märgner. ICDAR 2007 Arabic handwriting recognition competition. In *Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR)*, volume 2, pages 1274–1278, 2007.