

Volker Märgner
Haikal El Abed *Editors*

Guide to OCR for Arabic Scripts



Guide to OCR for Arabic Scripts

Volker Märgner • Haikal El Abed
Editors

Guide to OCR for Arabic Scripts



Springer

Editors

Volker Märgner
Institute for Communications Technology
Braunschweig Technical University
Braunschweig, Niedersachsen
Germany

Haikal El Abed
Institute for Communications Technology
Braunschweig Technical University
Braunschweig, Niedersachsen
Germany

ISBN 978-1-4471-4071-9

ISBN 978-1-4471-4072-6 (eBook)

DOI 10.1007/978-1-4471-4072-6

Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2012941223

© Springer-Verlag London 2012

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

It is with great pleasure that I accepted to write the foreword for this book.

In our society, we use computers increasingly for all kinds of communication. This means that for a language to work well for communication, we need to have computer tools—language technology—for that language. Language technology makes communication more efficient, and it enables many more users to have access to information in various forms.

Arabic is an important language, spoken in a large region of the world, and it is important to support the use of Arabic—in business, in public administration, in private homes, and even in the exchange of information with the rest of the world.

As not all Arabic information is already available in electronic form, one of the basic requirements in such a scenario is the ability of computers to “read” images of written Arabic, be it handwritten or printed, be it online or offline. This kind of technology (OCR—optical character recognition) is difficult to create even for languages in Latin script, and written Arabic adds to the complexity.

This book is a comprehensive guide to the field of Arabic OCR, offering thorough descriptions of the data sets for training and testing, several different OCR methodologies, and the evaluation and assessment thereof.

Before closing, I would like to congratulate the editors in having assembled such an important collection of contributions to a complex problem for which the Arabic world, and the rest of the world, needs solutions.

Copenhagen, Denmark

Bente Maegaard

Preface

Arabic Scripts

The Internet is a source of information which is used by approximately 2 billion users worldwide (www.internetworldstats.com). Two aspects affect the way in which the Internet is used, especially in searching for documents. One is the availability of more and more scanned documents with varying amounts of metadata for searching these documents on the Internet, and the other is the appearance of more and more languages with non-Latin characters. Both aspects show the importance of developing recognition technology for all types of characters and languages to make the content of scanned images of text available to the Internet users. A worldwide-used acronym for any type of text recognition is OCR, which means optical character recognition. OCR is used not only for recognizing printed characters, but it is often also used for cursive handwriting, even when words instead of single characters are recognized. Some alternative acronyms are used for the case of handwritten words, like HWR (handwritten word recognition) but these are not in common use today.

Knowing that about 200 million people in the world use Arabic as their first language it is obvious that a growing interest of that huge group of Arabic-speaking Internet users is to search for documents in their mother tongue. In parallel to this situation is in the past few years a growing interest in Arabic word and text recognition has been observed. During that time two events have been important landmarks in Arabic text recognition technology development. In 2002 a database on Arabic handwritten words (*IFN/ENIT-database*)¹ was made available to the community and has served as a reference for competitions since 2005 (ICDAR 2005).² In September 2006 a summit on Arabic and Chinese Handwriting Recognition was held at College Park, MD in the USA (SACH2006),³ where experts from both re-

¹<http://www.ifnenit.com>

²<http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=10526>

³<http://www.umiacs.umd.edu/lamp/meetings/SACH06/>

Table 1 Arabic characters (www.ethnologue.com)

IPA	Value	Name	Final	Medial	Initial	Isolated	IPA	Value	Name	Final	Medial	Initial	Isolated
[d̪]	d̪	dād̪	ض	ض	ض	ض	[?]	'(a)	alif	ا	-	-	ا
[t̪]	t̪	tā'	ط	ط	ط	ط	[b]	b	bā'	ب	ب	ب	ب
[z̪]	z̪	zā'	ظ	ظ	ظ	ظ	[t̪]	t̪	tā'	ت	ت	ت	ت
[f̪]	f̪	fā'	ع	ع	ع	ع	[θ̪]	th̪	thā'	ث	ث	ث	ث
[ɣ̪]	gh̪	ghayn	غ	غ	غ	غ	[ʒ̪]	j̪	jīm	ج	ج	ج	ج
[f̪]	f̪	fā'	ف	ف	ف	ف	[h̪]	h̪	hā'	ح	ح	ح	ح
[q̪]	q̪	qāf̪	ق	ق	ق	ق	[χ̪]	kh̪	khā'	خ	خ	خ	خ
[k̪]	k̪	kāf̪	ك	ك	ك	ك	[d̪]	d̪	dāl̪	د	-	-	د
[l̪]	l̪	lām	ل	ل	ل	ل	[ð̪]	dh̪	dhāl̪	ذ	-	-	ذ
[m̪]	m̪	mīm̪	م	م	م	م	[r̪]	r̪	rā'	ر	-	-	ر
[n̪]	n̪	nūn	ن	ن	ن	ن	[z̪]	z̪	zāy̪	ز	-	-	ز
[h̪]	h̪	hā'	ه	ه	ه	ه	[s̪]	s̪	sīn̪	س	س	س	س
[w̪]	w̪	wāw̪	و	-	-	و	[ʃ̪]	š̪	shīn̪	ش	ش	ش	ش
[j̪]	y̪	yā'	ي	ي	ي	ي	[s̪]	š̪	šād̪	ص	ص	ص	ص

Fig. 1 Example of an Arabic printed text

السلام عليكم
بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيمِ
كتاب التعرف على الخط العربي
شكراً لمشاركتكم

search fields presented their actual work. From that time intensive research on Arabic script recognition started and has resulted in a big step forward today.

Arabic script is the second most widespread script in the world; it is used not only for Arabic but also for the Persian, Urdu, and Pashto languages, for example. Today 14 languages use Arabic script worldwide, which shows its importance. Characteristics of Arabic script are a writing direction from right to left, characters within a word being mostly connected, 28 characters with different shapes for different positions in a word, and dots and diacritical signs above and below characters. Table 1 shows all the shapes of the 28 Arabic characters.

For different languages some additional characters may be used. Typical for Arabic script is also the variation of a word in length by elongation of the connecting lines between the characters. Figure 1 shows an example.

Table 2 Examples of ligatures

without ligatures	ligatures
مُحَمَّد	محمد
أَلَّيْ	ألي

A further important special Arabic script style is the possibility to write characters as vertical or horizontal ligatures. These ligatures modify the shape of the characters significantly. Some examples of ligatures are shown in Table 2. All these typical Arabic script characteristics influence the processing and recognition of Arabic script in different ways and make it clear that a simple adaptation of Latin character-based processing is not possible.

This book presents the state of the art of OCR for Arabic scripts presented from most active and successful groups. The parts of the book show that a lot of work still has to be done on Arabic script recognition. But the techniques and algorithms used are of general interest; many problems are typical not only for Arabic but for many other scripts. We believe that the collection of Arabic OCR related work is also an inspiration for other scripts and vice versa.

The book is divided into four parts. Part I, Pre-processing, presents different aspects of pre-processing and feature extraction for Arabic OCR systems. Part II, Recognition, includes chapters with details about different recognition approaches. Part III collects chapters describing the important aspects of how to assess the performance of a recognition system. The final Part IV, Applications presents system solutions for selected application fields.

Part I: Pre-processing

Part I presents different approaches for the pre-processing of OCR systems for Arabic. It starts with an overview of Arabic handwriting recognition technology. Srihari and Ball present in their chapter the parts of a recognition system from pre-processing to classification. Finally they discuss application fields and challenges. Chapters 2–6 deal with pre-processing tasks of an OCR system. Bukhari, Shafait, and Breuel discuss layout analysis methods, Setlur and Govindaraju pre-processing issues, Belaid and Ouwayed segmentation of ancient Arabic documents, and Likforman-Sulem et al. features for word recognition systems.

Part II: Recognition

Chapters 7–15 present different approaches for the recognition of Arabic script. The first six chapters all use HMM-based approaches. Borovikov and Zavorin present a multi-stage approach to document analysis, Ahmed, Mahmoud, and Parvez a recognizer for printed Arabic text, Pechwitz, El Abed, and Märgner an offline handwritten

Arabic word recognizer, Dreuw, Rybach, Heigold, and Ney a large vocabulary optical character recognition system, Alkhoury, Gimenez, and Juan a Bernoulli-based handwriting recognition system, Jifroodian and Suen a handwritten Farsi word recognition system, Kessentini, Paquet, and Ben Hamadou a multi-stream Markov model recognizer.

Two chapters discuss further approaches. Graves presents a recognition system based on multidimensional recurrent neural networks, and Mozaffari discusses the application of fractal theory for document analysis and recognition. Khemakhem and Belghith discuss an OCR system based on the combination of complementary systems in Chap. 15.

Part III: Evaluation

The subject of Part III is the evaluation of recognition systems. In Chap. 16 Zavorin and Borovikov discuss data collection and annotation, and Arabic handwriting recognition competitions are described in Chap. 17 by Märgner and El Abed. In Chap. 18, Slimane et al. describe benchmarking strategies for Arabic word recognition.

Part IV: Applications

The final Part IV presents different applications using Arabic script recognition technology. In Chap. 19 Cheriet and Moghaddam present a robust word spotting system for historical Arabic manuscripts. Natarajan discusses, in Chap. 20, script-independent methods for Arabic handwriting recognition, and Kundu and Hines present an Arabic handwriting recognition system using over-segmentation in Chap. 21. Boubaker et al. discuss online Arabic databases and applications using these data in Chap. 22, and Abdelazeem et al. present, in Chap. 23, techniques for using online and offline features for Arabic handwriting recognition.

Target Audience

This book provides an overview of the state-of-the-art research in the field of OCR for Arabic scripts. Different aspects and solutions have been addressed by the authors, and we hope that this comprehensive collection of ideas, problems, and solutions motivates researchers to continue this work. In that sense this book shall serve as a reference for researchers and graduate students studying OCR technology and methodology in general and for Arabic script in particular.

Braunschweig, Germany

Volker Märgner
Haikal El Abed

Acknowledgements

We thank all the chapter authors for their contributions to this book, which make it an invaluable resource to researchers in the area of OCR for all kinds of script using Arabic characters. A total of 67 authors prepared their contributions for the 23 chapters of the book and supported us during all steps from concept to draft and finalized chapter. We would like to thank Bente Maegaard, Director of the Centre for Language Technology at Copenhagen University, Denmark, for writing the foreword and for her continuing support in developing Arabic language resources and a network of researchers in this field. We would also like to thank Springer for their encouragement and persistence in driving us to complete this work in a timely manner.

Braunschweig, Germany

Volker Märgner
Haikal El Abed

Contents

Part I Pre-processing

- 1 An Assessment of Arabic Handwriting Recognition Technology** 3
Sargur N. Srihari and Gregory Ball
- 2 Layout Analysis of Arabic Script Documents** 35
Syed Saqib Bukhari, Faisal Shafait, and Thomas M. Breuel
- 3 A Multi-stage Approach to Arabic Document Analysis** 55
Eugene Borovikov and Ilya Zavorin
- 4 Pre-processing Issues in Arabic OCR** 79
Zhixin Shi, Srirangaraj Setlur, and Venu Govindaraju
- 5 Segmentation of Ancient Arabic Documents** 103
Abdel Belaïd and Nazih Ouwayed
- 6 Features for HMM-Based Arabic Handwritten Word Recognition Systems** 123
Laurence Likforman-Sulem, Ramy Al Hajj Mohammad, Chafic Mokbel, Fares Menasri, Anne-Laure Bianne-Bernard, and Christopher Kermorvant

Part II Recognition

- 7 Printed Arabic Text Recognition** 147
Irfan Ahmed, Sabri A. Mahmoud, and Mohammed Tanvir Parvez
- 8 Handwritten Arabic Word Recognition Using the IFN/ENIT-database** 169
Mario Pechwitz, Haikal El Abed, and Volker Märgner
- 9 RWTH OCR: A Large Vocabulary Optical Character Recognition System for Arabic Scripts** 215
Philippe Drew, David Rybach, Georg Heigold, and Hermann Ney

10 Arabic Handwriting Recognition Using Bernoulli HMMs	255
Ihab Alkhoury, Adrià Giménez, and Alfons Juan	
11 Handwritten Farsi Word Recognition Using Hidden Markov Models	273
Puntis Jifroodian Haghighi and Ching Y. Suen	
12 Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks	297
Alex Graves	
13 Application of Fractal Theory in Farsi/Arabic Document Analysis	315
Saeed Mozaffari	
14 Multi-stream Markov Models for Arabic Handwriting Recognition	335
Yousri Kessentini, Thierry Paquet, and AbdelMajid Ben Hamadou	
15 Toward Distributed Cursive Writing OCR Systems Based on a Combination of Complementary Approaches	351
Maher Khemakhem and Abdelfattah Belghith	

Part III Evaluation

16 Data Collection and Annotation for Arabic Document Analysis	375
Ilya Zavorin and Eugene Borovikov	
17 Arabic Handwriting Recognition Competitions	395
Volker Märgner and Haikal El Abed	
18 Benchmarking Strategy for Arabic Screen-Rendered Word Recognition	423
Fouad Slimane, Slim Kanoun, Jean Hennebert, Rolf Ingold, and Adel M. Alimi	

Part IV Applications

19 A Robust Word Spotting System for Historical Arabic Manuscripts	453
Mohamed Cheriet and Reza Farrahi Moghaddam	
20 Arabic Text Recognition Using a Script-Independent Methodology: A Unified HMM-Based Approach for Machine-Printed and Handwritten Text	485
Premkumar Natarajan, Rohit Prasad, Huagu Cao, Krishna Subramanian, Shirin Saleem, David Belanger, Shiv Vitaladevuni, Matin Kamali, and Ehry MacRostie	

21 Arabic Handwriting Recognition Using VDHMM and Over-segmentation	507
Amlan Kundu and Tom Hines	
22 Online Arabic Databases and Applications	541
Houcine Boubaker, Abdelkarim Elbaati, Najiba Tagougui, Haikal El Abed, Monji Kherallah, and Adel M. Alimi	
23 On-line Arabic Handwritten Word Recognition Based on HMM and Combination of On-line and Off-line Features	559
Sherif Abdelazeem, Hesham M. Eraqi, and Hany Ahmed	
Index	587

Contributors

Sherif Abdelazeem Electronics Engineering Dept., The American University in Cairo (AUC), Cairo, Egypt

Hany Ahmed Electronics Engineering Dept., The American University in Cairo (AUC), Cairo, Egypt

Irfan Ahmed Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

Adel M. Alimi REGIM Group, National School of Engineers (ENIS), University of Sfax, Sfax, Tunisia; Research Group on Intelligent Machines (REGIM), National School of Engineers ENIS, University of Sfax, Sfax, Tunisia

Ihab Alkhoury DSIC, Universitat Politècnica de València, València, Spain

Ramy Al Hajj Mohammad Lebanese International University, Beqaa Valley, Lebanon

Gregory Ball Center of Excellence for Document Analysis and Recognition (CEDAR), University at Buffalo, State University of New York, Amherst, NY, USA

Abdel Belaïd LORIA, Vandoeuvre, France

David Belanger BBN, Cambridge, MA, USA

Abdelfettah Belghith HANA Research Group, ENSI, University of Manouba, Manouba, Tunisia

AbdelMajid Ben Hamadou Laboratoire MIRACL, Université de Sfax Tunisie, Sfax, Tunisia

Anne-Laure Bianne-Bernard Telecom ParisTech, CNRS UMR 5141, Paris, France; A2iA, Paris, France

Eugene Borovikov Knowledge and Information Management Division, CACI, Lanham, MD, USA; Research and Development, CACI International Inc, Lanham, MD, USA

Houcine Boubaker Research Group on Intelligent Machines (REGIM), National School of Engineers ENIS, University of Sfax, Sfax, Tunisia

Thomas M. Breuel Technical University of Kaiserslautern, Kaiserslautern, Germany

Syed Saqib Bukhari Technical University of Kaiserslautern, Kaiserslautern, Germany

Huaigu Cao BBN, Cambridge, MA, USA

Mohamed Cheriet Synchromedia Laboratory for Multimedia Communication in Telepresence, École de Technologie Supérieure, Montréal, QC, Canada

Philippe Drew Human Language Technology and Pattern Recognition, RWTH Aachen University, Aachen, Germany

Haikal El Abed Institute for Communications Technology (IfN), Technische Universität Braunschweig, Braunschweig, Germany

Abdelkarim Elbaati Research Group on Intelligent Machines (REGIM), National School of Engineers ENIS, University of Sfax, Sfax, Tunisia

Hesham M. Eraqi Electronics Engineering Dept., The American University in Cairo (AUC), Cairo, Egypt

Adrià Giménez DSIC, Universitat Politècnica de València, València, Spain

Venu Govindaraju Department of CSE, University at Buffalo, SUNY, Buffalo, NY, USA

Alex Graves Technical University of Munich, Munich, Germany

Puntis Jifroodian Haghghi CENPARMI (Center for Pattern Recognition and Machine Intelligence), Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec, Canada

Georg Heigold Human Language Technology and Pattern Recognition, RWTH Aachen University, Aachen, Germany

Jean Hennebert DIVA Group, Department of Informatics, University of Fribourg, Fribourg, Switzerland

Tom Hines MITRE, McLean, VA, USA

Rolf Ingold DIVA Group, Department of Informatics, University of Fribourg, Fribourg, Switzerland

Alfons Juan DSIC, Universitat Politècnica de València, València, Spain

Matin Kamali BBN, Cambridge, MA, USA

Slim Kanoun National School of Engineers (ENIS), University of Sfax, Sfax, Tunisia

Christopher Kermorvant A2iA, Paris, France

Youssi Kessentini Laboratoire LITIS EA 4108, Université de Rouen France, Saint-Etienne du Rouvray, France

Maher Khemakhem MIRACL Lab, The Higher Institute of Management University of Sousse, Sousse, Tunisia

Monji Kherallah Research Group on Intelligent Machines (REGIM), National School of Engineers ENIS, University of Sfax, Sfax, Tunisia

Amlan Kundu MITRE, McLean, VA, USA

Laurence Likforman-Sulem Telecom ParisTech, CNRS UMR 5141, Paris, France

Ehry MacRostie BBN, Cambridge, MA, USA

Sabri A. Mahmoud Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

Volker Märgner Institute for Communications Technology (IfN), Technische Universität Braunschweig, Braunschweig, Germany

Fares Menasri A2iA, Paris, France

Reza Farrahi Moghaddam Synchromedia Laboratory for Multimedia Communication in Telepresence, École de Technologie Supérieure, Montréal, QC, Canada

Chafic Mokbel UOB, El-Koura, Lebanon

Saeed Mozaffari Electrical and Computer Department, Semnan University, Semnan, Iran

Premkumar Natarajan BBN, Cambridge, MA, USA

Hermann Ney Human Language Technology and Pattern Recognition, RWTH Aachen University, Aachen, Germany

Nazih Ouwayed LORIA, Vandoeuvre, France

Thierry Paquet Laboratoire LITIS EA 4108, Université de Rouen France, Saint-Etienne du Rouvray, France

Mohammed Tanvir Parvez Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

Mario Pechwitz Institute for Communications Technology (IfN), Technische Universität Braunschweig, Braunschweig, Germany

Rohit Prasad BBN, Cambridge, MA, USA

David Rybach Human Language Technology and Pattern Recognition, RWTH Aachen University, Aachen, Germany

Shirin Saleem BBN, Cambridge, MA, USA

Srirangaraj Setlur Department of CSE, University at Buffalo, SUNY, Buffalo, NY, USA

Faisal Shafait German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany

Zhixin Shi Department of CSE, University at Buffalo, SUNY, Buffalo, NY, USA

Fouad Slimane DIVA Group, Department of Informatics, University of Fribourg, Fribourg, Switzerland

Sargur N. Srihari Center of Excellence for Document Analysis and Recognition (CEDAR), University at Buffalo, State University of New York, Amherst, NY, USA

Krishna Subramanian BBN, Cambridge, MA, USA

Ching Y. Suen CENPARMI (Center for Pattern Recognition and Machine Intelligence), Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec, Canada

Najiba Tagougui Research Group on Intelligent Machines (REGIM), National School of Engineers ENIS, University of Sfax, Sfax, Tunisia

Shiv Vitaladevuni BBN, Cambridge, MA, USA

Ilya Zavorin Knowledge and Information Management Division, CACI, Lanham, MD, USA; Research and Development, CACI International Inc, Lanham, MD, USA

Part I

Pre-processing

Chapter 1

An Assessment of Arabic Handwriting Recognition Technology

Sargur N. Srihari and Gregory Ball

Abstract Automated methods for the recognition of Arabic script are at an early stage compared to their counterparts for the recognition of Latin and Chinese scripts. An assessment of the technology for Arabic handwriting recognition is provided based on the published literature. An introduction to the Arabic script is given followed by a description of algorithms for the processes involved: segmentation, feature extraction, classification, and search. Existing corpora for Arabic are described together with a design for corpus collection. The paper is concluded by identifying technology gaps and providing a bibliography of the recent literature on Arabic recognition.

1.1 Introduction

While automated handwritten text recognition technology for Latin script languages has been the subject of significant research, Arabic script has received far less attention. Existing recognition technology is still in its relative infancy. This paper is a survey and qualitative assessment of the state of the art of technology for handwritten Arabic recognition techniques. It covers recent articles related to the subject that have been published in the International Conference on Document Analysis and Recognition (ICDAR), Symposium on Document Image Understanding Technology (SDIUT), International Workshop on Frontiers in Handwriting Recognition (IWFHR), The International Society for Optical Engineering-Document Recognition and Retrieval (SPIE-DRR), and International Journal of Document Analysis and Recognition (IJDAR) proceedings and journals. We first provide a general presentation of the Arabic script and its main features relevant for automated recognition, followed by an overview of the recognition process itself. We then illustrate the pre-processing tasks involved in the process and discuss recognition and document search methods.

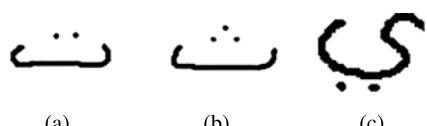
S.N. Srihari (✉) · G. Ball

Center of Excellence for Document Analysis and Recognition (CEDAR), University at Buffalo,
State University of New York, Amherst, NY 14228, USA
e-mail: srihari@cedar.buffalo.edu

Character	alone	end	middle	begin	Character	alone	end	middle	begin
Alif	ا	أ			Dhad	ض	ض	ض	ض
Ba	ب	ب	ب	ب	Taa	ط	ط	ط	ط
Ta	ت	ت	ت	ت	Dha	ظ	ظ	ظ	ظ
Tha	ث	ث	ث	ث	Ayn	ع	ع	ع	ع
Jim	ج	ح	ج	ج	Ghayn	غ	خ	خ	خ
Ha	ح	ح	ح	ح	Fa	ف	ف	ف	ف
Kha	خ	خ	خ	خ	Qaf	ق	ق	ق	ق
Dal	د	د			Kaf	ك	ك	ك	ك
The	ذ	ذ			Lam	ل	ل	ل	ل
Ra	ر	ر			Mim	م	م	م	م
Zai	ز	ز			Nun	ن	ن	ن	ن
Sin	س	س	س	س	He	ه	ه	ه	ه
Chin	ش	ش	ش	ش	Waw	و	و	و	و
Sad	ص	ص	ص	ص	Ya	ي	ي	ي	ي

Fig. 1.1 The Arabic alphabet and the shapes of its letters

Fig. 1.2 The shapes of
 (a) teh, a letter with two dots
 above, (b) theh, a letter with
 three dots above, (c) yeh, a
 letter with two dots below



1.1.1 Features of Arabic Script

Arabic is a semi-cursive language with an alphabet of 28 letters, 22 cursive and 6 non-cursive. Cursive letters can have up to four different shapes depending on their position within a sub-word. The shapes correspond to being placed at the start of a sub-word (initial form), in the middle (medial form), and at the end (final form), as well as a separate letter form (isolate form). Figure 1.1 shows the letters with their different shapes. Non-cursive letters have one unique shape which does not depend on their position. There is also no concept of case in Arabic letters.

Arabic uses various kinds of dot components to distinguish some characters. One letter can have up to three dots which can be placed (depending on the letter) either above or below the main form. Figure 1.2 illustrates the use of dots in some Arabic letters.

Arabic also uses diacritical marks (vocalized text) to control the pronunciation of words, but these rarely appear in handwritten documents since the context generally makes the pronunciation unambiguous. They are used mostly in formal documents

Fig. 1.3 Sub-words occurring in the Arabic equivalent of “the situation”

Fig. 1.4 Laam-heh, laam-meem, and laam-alef combinations



and where the context is ambiguous. Most Arabic handwriting research focuses on non-vocalized Arabic text.

Since the non-cursive letters lack a connecting form, their presence in a word necessitates a break in the word, making for a predictable pattern of sub-words. Figure 1.3 shows the different components for the word corresponding to the English “the situation.”

When some combinations of letters appear, they have unique forms. These combinations are mostly pairs of letters, such as “laam-alef,” “laam-meem,” or “laam-heh,” all common occurrences in handwritten text. Some, such as “laam-alef” are even written the same way in machine printed text. Figure 1.4 shows examples of connected letters.

Unlike Latin script languages, handwritten and machine printed Arabic are very similar. While letter shapes have minor variations due to style in handwritten text, the way letters are connected is always the same. Figure 1.5 shows an example of the same text with a machine printed version and a handwritten version.

Arabic script is not used only for Arabic itself. Farsi and Urdu, for example, use similar scripts with small differences. Figure 1.6 shows the Unicode table for Arabic script as well as extended symbols including letters and numerals used in Middle Eastern countries and some letters used in other languages. The actual range for Arabic letters is from 0x0622 to 0x064A.

1.1.2 Related Surveys/Assessments

Some recent publications have focused on Arabic recognition technology. Lorigo and Govindaraju [29] give an overview of the off-line handwriting recognition techniques. Amara and Bouslama [8] focused on the hybrid methods in recognition (methods which use more than one source of information).

1.2 Overview of the Recognition Process

While different algorithms vary in their implementation details, most follow a general path from document to Unicode text (Fig. 1.7 shows an overview of Ara-

النقى رئيس اللقاء الوطنى للحوار الفكرى الشيخ صالح بن عبدالرحمن الحصين مساء اليوم مع نخبة من المثقفين والملقىات تبادل الآراء ووجهات النظر حول الحوار الوطنى وسبل تطويره وذلك بمركز الامير سلمان الاجتماعى فى الرياض.

وقد استهل اللقاء بعرض فيلم عن مركز الملك عبدالعزيز للحوار الوطنى وما مر به من مراحل .

بعد ذلك استعرض النشيف صالح الحصين مسيرة مركز الملك عبدالعزيز للحوار الوطنى مؤكداً أن هذا المركز الشئء لم يكون لبلة في بناء المجتمع من خلال ترسيخ مفهوم الحوار وثقافة الحوار وجعلها قيمة مثل في نفوس أفراد المجتمع وتكرير الوحدة الوطنية في إطار العقيدة الإسلامية ورسالة خطاب إسلامي صحيحة مبنية على الوسطية والاعتدال ومعالجة جميع القضايا الوطنية المختلفة من خلال الحوار والياته .

(a)

النقى رئيس اللقاء الوطنى للحوار التركى الذى صاحبته عبادات
الحين ماداً اليوم مع فنقة من المثقفين والذئب انتقام لتبادل
آراء ووجهات النظر حول الحوار الوطنى وسبل تطويره وذلك
برئاسة الامير سلمان الاصفهانى فى الرياض .

وفى استهل اللقاء بعرض فيلم عن مركز الملك عبد العزىز للحوار
الوطني وما مر به من مراحل .

بعد ذلك استعرض النشيف صالح الحصين مسيرة مركز الملك عبد العزىز
للحوار الوطنى مؤكداً أن هنا الحركى اثنى لكون لبيت فى بناء
المجتمع من خلال ترسيخ مفهوم الحوار وثقافة الحوار وجعلها قيمة
مبنية في تعنى أفراد المجتمع وتكرس الوحدة الوطنية في إطار العقيدة
الإسلامية ورسالة خطاب إسلامي صحيحة مبنية على الوسطية والاعتدال
وتحافظ جميع القضايا الوطنية المختلفة من خلال الحوار والياته .

(b)

Fig. 1.5 Machine printed (a) and handwritten (b) versions of the same Arabic text

bic handwriting recognition system). A document is generally input as a scanned grayscale image. Pre-processing first converts the document to a black and white image (binarization) and then converts to some representation which is easier to process than the raw image itself, such as a chain code or a skeleton representation. From here, additional pre-processing steps such as noise reduction, removing the slant angle, and smoothing are done. This cleaned image is then passed to a segmentation algorithm.

060	061	062	063	064	065	066	067	068	069	06A	06B	06C	06D	06E	06F
۰ 0600	۱ 0610	۲ 0620	۳ 0630	۴ 0640	۵ 0650	۶ 0660	۷ 0670	۸ 0680	۹ 0690	۰A 06A0	۰B 06B0	۰C 06C0	۰D 06D0	۰E 06E0	۰F 06F0
۱ 0601	۲ 0611	۳ 0621	۴ 0631	۵ 0641	۶ 0651	۷ 0661	۸ 0671	۹ 0681	۰A 0691	۰B 06A1	۰C 06B1	۰D 06C1	۰E 06D1	۰F 06E1	۱ 06F1
۲ 0602	۳ 0612	۴ 0622	۵ 0632	۶ 0642	۷ 0652	۸ 0662	۹ 0672	۰A 0682	۰B 0692	۰C 06A2	۰D 06B2	۰E 06C2	۰F 06D2	۱ 06E2	۲ 06F2
۳ 0603	۴ 0613	۵ 0623	۶ 0633	۷ 0643	۸ 0653	۹ 0663	۰A 0673	۰B 0683	۰C 0693	۰D 06A3	۰E 06B3	۰F 06C3	۱ 06D3	۲ 06E3	۳ 06F3
۴ 0614	۵ 0624	۶ 0634	۷ 0644	۸ 0654	۹ 0664	۰A 0674	۰B 0684	۰C 0694	۰D 06A4	۰E 06B4	۰F 06C4	۱ 06D4	۲ 06E4	۳ 06F4	
۵ 0615	۶ 0625	۷ 0635	۸ 0645	۹ 0655	۰A 0665	۰B 0675	۰C 0685	۰D 0695	۰E 06A5	۰F 06B5	۱ 06C5	۲ 06D5	۳ 06E5	۴ 06F5	
۶ 0626	۷ 0636	۸ 0646	۹ 0656	۰A 0666	۰B 0676	۰C 0686	۰D 0696	۰E 06A6	۰F 06B6	۱ 06C6	۲ 06D6	۳ 06E6	۴ 06F6		
۷ 0627	۸ 0637	۹ 0647	۰A 0657	۰B 0667	۰C 0677	۰D 0687	۰E 0697	۰F 06A7	۱ 06B7	۲ 06C7	۳ 06D7	۴ 06E7	۵ 06F7		
۸ 0628	۹ 0638	۰A 0648	۰B 0658	۰C 0668	۰D 0678	۰E 0688	۰F 0698	۱ 06A8	۲ 06B8	۳ 06C8	۴ 06D8	۵ 06E8	۶ 06F8		
۹ 0629	۰A 0639	۰B 0649	۰C 0659	۰D 0669	۰E 0679	۰F 0689	۱ 0699	۲ 06A9	۳ 06B9	۴ 06C9	۵ 06D9	۶ 06E9	۷ 06F9		
A 062A	۱ 063A	۲ 064A	۳ 065A	۴ 066A	۵ 067A	۶ 068A	۷ 069A	۸ 06AA	۹ 06BA	۰A 06CA	۰B 06DA	۰C 06EA	۰D 06FA		
B 060B	۱ 061B	۲ 062B	۳ 063B	۴ 064B	۵ 065B	۶ 066B	۷ 067B	۸ 068B	۹ 069B	۰A 06AB	۰B 06BB	۰C 06CB	۰D 06DB	۰E 06EB	۰F 06FB
C 060C	۱ 061C	۲ 062C	۳ 063C	۴ 064C	۵ 065C	۶ 066C	۷ 067C	۸ 068C	۹ 069C	۰A 06AC	۰B 06BC	۰C 06CC	۰D 06DC	۰E 06EC	۰F 06FC
D 060D	۱ 061D	۲ 062D	۳ 063D	۴ 064D	۵ 065D	۶ 066D	۷ 067D	۸ 068D	۹ 069D	۰A 06AD	۰B 06BD	۰C 06CD	۰D 06DD	۰E 06ED	۰F 06FD
E 060E	۱ 061E	۲ 062E	۳ 063E	۴ 064E	۵ 065E	۶ 066E	۷ 067E	۸ 068E	۹ 069E	۰A 06AE	۰B 06BE	۰C 06CE	۰D 06DE	۰E 06EE	۰F 06FE
F 060F	۱ 061F	۲ 062F	۳ 063F	۴ 064F	۵ 065F	۶ 066F	۷ 067F	۸ 068F	۹ 069F	۰A 06AF	۰B 06BF	۰C 06CF	۰D 06DF	۰E 06EF	۰F 06FF

Fig. 1.6 Unicode table for Arabic script languages. Includes letters for Arabic and other languages (Urdu, Farsi, Uighur, Kazakh, Kirghiz, Kurdish) with the numerals and symbols used in these languages

A segmentation algorithm splits a large image into smaller regions of interest. For example, a page segmentation algorithm segments an unconstrained page of text into component lines. Lines are segmented into words, and words into characters or sub-characters. The segmentation algorithm can then pass its output directly to a recognition algorithm. If the segmentation algorithm is completely distinct from the

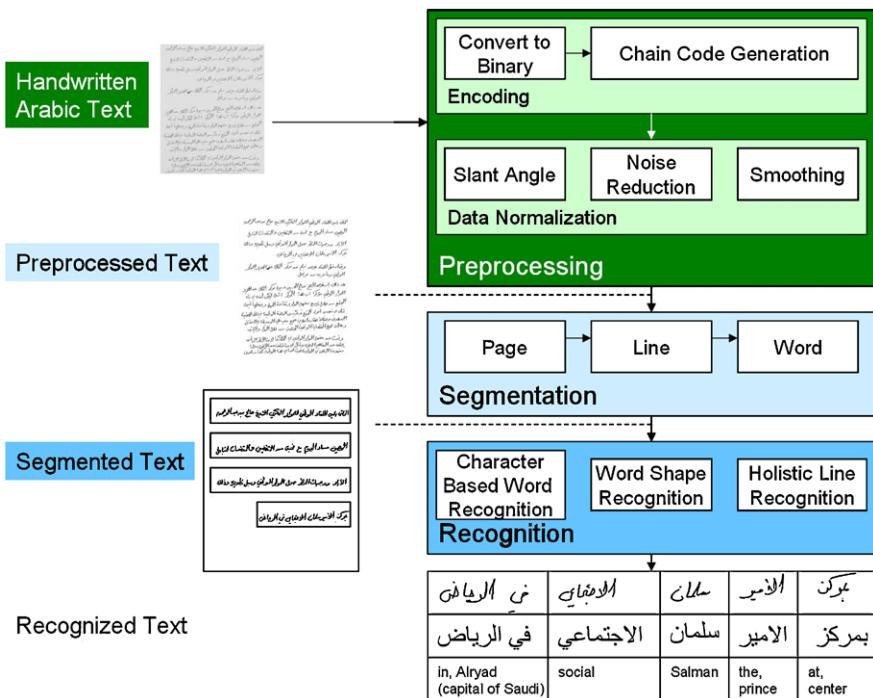


Fig. 1.7 Overview of recognizing Arabic handwriting

recognition, its performance can be a bottleneck for recognition. If the segmentation is incorrect, the lines, words, or characters might not be present in a single image, or multiple ones might be present in the same image. For this reason, some algorithms (segmentation-free algorithms) might make only partial use of a segmentation tool. They might use the output of page segmentation, but not line, for example. Another approach is to use a segmentation algorithm as a tool to make suggestions rather than to dictate choices.

Word recognition algorithms fall into three broad main categories. Character-based word recognition attempts to segment or over-segment words into pieces and maximize the score of the component characters against some candidate word in the lexicon. Word shape recognition matches whole word features against entire words. Holistic line recognition attempts to maximize the score of a given line of text rather than act on individual words (for an overview see Fig. 1.8).

1.3 Document Pre-processing, Segmentation, and Candidate Generation

Before recognition is performed on handwritten documents, pre-processing must first be performed. The choice of pre-processing steps ultimately depends on

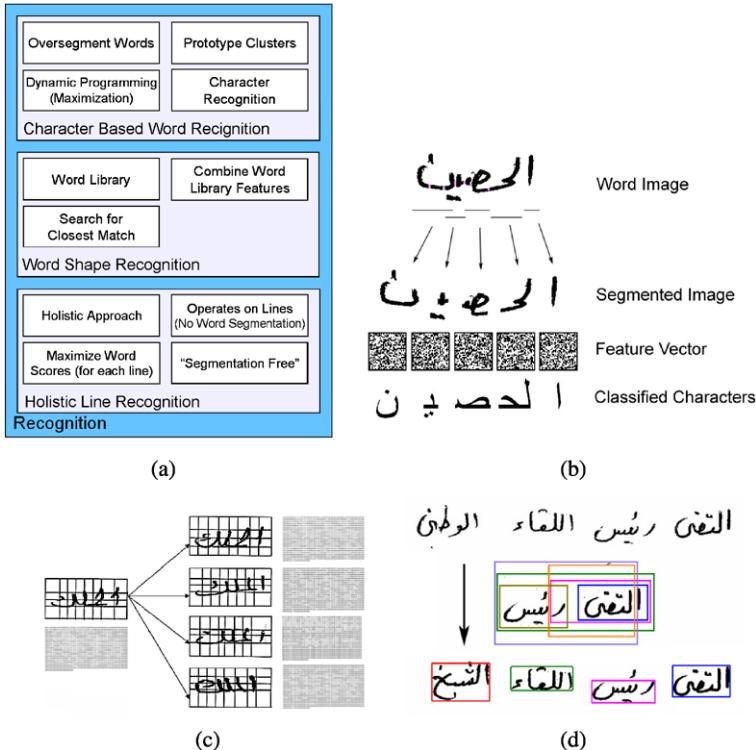


Fig. 1.8 Detailed breakdown of handwritten Arabic recognition. (a) Major methods, (b) character-based recognition, (c) word shape recognition, (d) holistic line recognition

the desired recognition algorithm, but some common tasks include binarization, skew/slant correction, and line detection and separation. Line separation often is postponed to the segmentation step. Segmentation algorithms attempt to split a document into pieces: pages into lines, lines into words, words into characters. These algorithms generate candidate regions for recognition.

1.3.1 Pre-processing

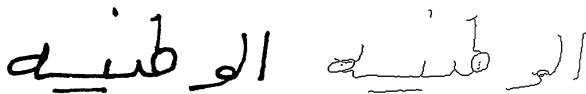
As mentioned, the choice of pre-processing steps ultimately depends on the desired recognition algorithm, but some common tasks include binarization, skew or slant correction, and line detection and separation. One approach specifically used for Arabic text to eliminate skew is to find the baseline of each word and rotate it on its center of gravity so that the baseline becomes horizontal.

Some work has focused specifically on pre-processing Arabic handwritten documents for recognition. Farooq et al. [20] used the IFN/ENIT dataset to develop

Fig. 1.9 Contour-based representation



Fig. 1.10 The Arabic word “al wataniya” (nationalism) and its skeleton



and test their pre-processing methods by combining them into document-like images, in order to simulate skew, line separation, and other common features of such images. They were able to recognize the correct baseline 78.5 % of the time with their method based on the local minima points of words. However, they fail to find a baseline in situations where the diacritics are large relative to the word, and suggest removing diacritics as a potential solution. They did not do work on separating overlapping handwritten lines. Methods for slant, skew correction, and separating a line into words are discussed, but no specific results are presented.

1.3.2 Representation

Two main types of representation have been used in handwritten Arabic recognition research. The chain code representation [22] traces the exterior and interior contours of 8-connected groups of pixels. It preserves the positional and directional information of adjacent pixels (an example is shown in Fig. 1.9). Skeletonization uses various thinning algorithms to extract skeletons, generally single-pixel-wide forms representing the “core” form of the writing. It is often used to generate structural features from characters. An example of a word skeleton is shown in Fig. 1.10.

1.4 Discrimination of Handwriting and Machine Print

Several groups have worked on classifying regions or words of Arabic documents that contain both handwriting and machine print into their own separate classes. This is particularly difficult in the case of Arabic because of the cursive nature of both the machine printed and handwritten forms.

Sridharan et al. hypothesize that in Arabic handwriting, horizontal runs and gradients are not as uniform as in machine print [46]. Their method is trainable, and thresholds were not selected empirically, unlike some previous approaches. Their approach was based on Gabor filters followed by classification using an expectation maximization-based probabilistic neural network. An overall precision of about 95 % was found with this method on an internally collected dataset. Training was done on 50 words each of handwritten and machine printed images, and the testing data consisted of documents containing a total of 286 machine printed and 104 handwritten words.

Femiani et al. [21] used data in the form of 430 “almost-randomly chosen” degraded 150 ppi 8-bit grayscale Arabic language document images that had been extracted from compressed PDF files, suffering compression and scanning artifacts. They described a set of 2D and 3D stroke attributes, and generated a classifier using the JRip propositional rule learner. They were able to correctly identify about 95 % of the components.

1.5 Segmentation

Unconstrained pages of text need to have some separation take place, grouping text into regions. This is the process of segmentation, generating candidate regions for recognition. Segmentation algorithms often attempt to split a document into pieces: pages into lines, lines into words, and words into characters.

1.5.1 Line Segmentation

Several methods have been successfully used for line separation. Hough transform methods are able to detect broken lines but are computationally slow. Horizontal projection-based methods use the projection profile of the image with its local maxima and minima. Vectorization algorithms extract vectors from pieces of broken lines and merge them together according to rules to form directional single connected chains (DSCCs) which can represent lines in the text. This DSCC approach was described further by Sridharan et al. [46].

A recent algorithm described by Arivazhagan et al. [11] starts by obtaining an initial set of candidate lines from the piecewise projection profile of the document. The lines traverse any obstructing handwritten connected component by associating it to the line above or below. A decision of associating such a component is made by (i) modeling the lines as bivariate Gaussian densities and evaluating the probability of the component under each Gaussian or (ii) evaluating the probability obtained from a distance metric. The proposed method is robust to handle skewed documents and those with lines running into each other. Experimental results show that on 720 documents (including English, Arabic, and children’s handwriting) containing a total of 11,581 lines, 97.31 % of the lines were segmented correctly. On an experiment over 200 handwritten images with 78,902 connected components, 98.81 % of them were associated to the correct lines.

1.5.2 Word Segmentation

One method specific to automatic word segmentation of Arabic was presented recently by Srihari et al. [51]. The process of automatic word segmentation begins

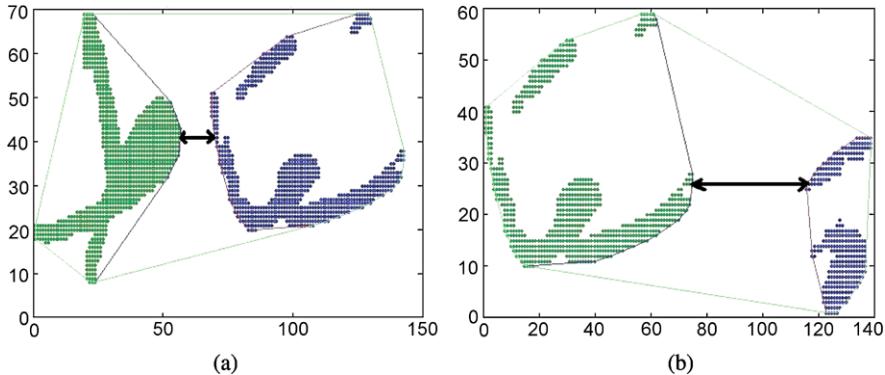


Fig. 1.11 Automatic segmentation gaps **(a)** between characters and **(b)** between words

with obtaining the set of connected components for each line in the document image. Figure 1.11(a) and Fig. 1.11(b) show the connected components (exterior and interior contours) of a small region of a document image. The interior contours or loops in a component are ignored for the purpose of word segmentation as they provide no information for this task. The connected components are grouped into clusters by merging minor components such as dots above and below a major component.

Also particular to Arabic, many words start with the Arabic character “Alef.” The presence of an “Alef” is a strong indicator that there may be a word gap between the pair of clusters. The height and width of the component are two parameters used to check if the component is the character “Alef.” Figure 1.12 shows samples of the Arabic character “Alef.” All pairs of adjacent clusters are candidates for word gaps.

Nine features are extracted for these pairs of clusters, and a neural network is used to determine if the gap between the pair is a word gap. The features are: width of the first cluster, width of the second cluster, difference between the bounding box of the two clusters, flag set to 1 or 0 depending on the presence or absence of the Arabic character “Alef” in the first cluster, the same flag for the second cluster, number of components in the first cluster, number of components in the second cluster, minimum distance between the convex hulls enclosing the two clusters, and the ratio between the sum of the areas enclosed by the convex hulls of the individual clusters to the total area inside the convex hull enclosing the clusters together. The minimum distance between convex hulls is calculated by sampling points on the convex hull for each connected component and calculating the minimum distance of all pairs of such points. Some of the differences noted between the tasks of segmenting Arabic script and segmenting Latin script are the presence of multiple dots above and below the main body in Arabic and the absence of upper case letters at the beginning of sentences in Arabic. The method presented was found to have an overall correctness of about 60 %.

Another method, the Arabic Character Segmentation Algorithm (ACSA) discussed by Sari et al. [43], relies on local minima detection in connected components (or sub-words). A preliminary step detects all local minima, which are then

Fig. 1.12 Samples of Arabic character “Alef.” The height and width are two parameters that are used to detect the presence of “Alef” in the clusters



Fig. 1.13 Segmentation of the first component of the word “meezan.” (a) All local minima, (b) valid segmentation points



classified into valid or non-valid segmentation points, depending on a set of six acceptance rules. Points that are not accepted go through a rejection test using three rejection rules. The decision rules are based on the features of Arabic script in which ascenders, descenders, and loops have a key role. ACSA also uses a character separation method that extracts the outer contour between segmentation points and then rebuilds the character with a filling procedure after adding the loops. Figure 1.13 shows an example of segmentation points.

The algorithm fails to segment words when it encounters special connections between two letters or components that touch each other unexpectedly. The method reports an 86 % success rate with 9 % over-segmentation and 5 % rejection.

Another similar approach described by Olivier et al. [39] was based on a simplification made on the upper contour of the processed words. A Freeman code representation is first extracted from the contour. The algorithm then removes the small variations in direction so that the contour used for segmentation is represented only by horizontal segments, which will help avoid incorrect local minima. Local minima are detected and filtered according to a set of three rules. The method reports a success rate of 97.41 % with a sub-segmentation rate of 2.24 % and 0.35 % of unnecessary segments. The method fails when an overlapping is encountered (a combination of letters merged together). A solution is given which tries to find artificial local minima in the overlapping area to separate the merged characters.

1.5.3 Character Segmentation

Ligatures are strong candidates for segmentation points in cursive scripts. If the distance between y -coordinates of the upper half and lower half of the outer contour for an x -coordinate is less than or equal to the average stroke width, then the x -coordinate is marked as an element of a ligature. Concavity features in upper contour and convexities in the lower contour are also used to generate candidate segmentation points, which are especially useful for distinct characters which are

touching, as opposed to being connected. A ligature will cause any overlapped concavity features to be ignored. For a given x -coordinate, if a concavity and convexity overlap, a segmentation point is added for that x -coordinate. This ligature-based method was used by Ball et al. in [12] for their spotting recognition step.

Lorigo and Govindaraju [28] describe a character segmenter which over-segments each word and then removes extra breakpoints using some prior knowledge of letter shapes. On 200 images from the IFN/ENIT dataset, they found 92.3 % of the segmentation points correctly with 5.1 % instances of over-segmentation. They discuss the fact that they need to do additional work to correctly add pre-recognition information on shadda and hamza.

1.5.4 Segmentation-Free Line Processing

Ball et. al [50] put forth a segmentation-free algorithm that processes entire lines rather than relying on pre-segmented words. Rather than attempt to parse the line into distinct words, it uses a sliding window approach in order to consider overlapping regions of interest. By considering overlapping candidates, this algorithm can avoid some of the bottleneck of straightforward word segmentation.

The algorithm can be viewed as a sequence of steps. First, the image is processed into component lines. Candidate segmentation points are generated for a given line. The line is scanned with a sliding window, generating candidate words and scoring them, as well as filtering out nearly equivalent candidates. The segmentation algorithm used on the line is essentially the same as the one used to generate candidate character segmentation points in candidate words in the actual spotting step. It is performed via a combination of ligatures and concavity features on an encoded contour of the components of the image. The average stroke width is estimated and used to determine the features. The segmentation-free line processing method uses a character segmentation algorithm applied to the split of entire lines, the motivation being to generate candidate word regions on the line. Arabic has predictable breaks in a word based on non-connective characters. Therefore, the number of connected components in a word is predictable as well.

Line Scanning

The method utilizes a sliding window, as illustrated in Fig. 1.14. The direction of the scan is unimportant because all realistic combinations of connected components will be considered.

Each character class c in Arabic is associated with a minimum and a maximum durational length ($\text{minlen}(c)$ and $\text{maxlen}(c)$, respectively). These lengths are generated by segmenting a representative dataset of characters with the same segmentation algorithm, and taking the min and max for each character. Due to the nature of the Arabic character set, the upper bound for all characters is 5, not 4 as in [27].

Fig. 1.14 Candidate word regions



The scanning algorithm will scan for candidate words consisting of a range of segments. For a given search word W consisting of n characters, c_0 to $c_{n-1} \in W$, the minimum number of segments $\text{minlen}(W)$ considered is $\sum_{i=0}^{n-1} \text{minlen}(c_i)$ and the maximum considered length $\text{ maxlen}(W)$ is $\sum_{i=0}^{n-1} \text{ maxlen}(c_i)$.

The scanning algorithm considers each segment s on a line generated by the segmentation algorithm acting on the line. For a given segment s_i , if $i = 0$ or if $s_i.\text{left} > s_{i-1}.\text{right}$ (i.e., there is horizontal space to the left of the segment), it is considered a valid start point. Similarly, for a given segment s_i , if $i = \text{max}(s)$ or if $s_i.\text{right} < s_{i+1}.\text{left}$ (i.e., there is horizontal space to the right of the segment), it is considered a valid endpoint. The algorithm considers candidate words to be all ranges of segments starting with some valid start point s_i , ending with a valid endpoint s_j , such that $\text{minlen}(W) \leq j - i + 1 \leq \text{ maxlen}(W)$.

While this generally results in more candidate words than the other segmentation method, it does not result in a dramatic decrease in performance since each Arabic word is only broken into a few pieces separated by whitespace.

Filtering

Often, a candidate word influences neighboring candidate words' scores. Neighboring candidate words are those words with overlapping segments. Often, a high scoring word will also have high scores for neighboring candidates. One issue arises when the high scoring word is in fact an incorrect match. In this case, the incorrect choice and several of its neighboring candidates may receive similarly good scores, pushing the rank of the actual word lower in the list. Another issue is if the target word appears multiple times in a document. The best matching words' neighboring candidates can depress the second occurrence's rank. Various ways of dealing with the overlap meet with different degrees of success.

The approach taken in the current implementation of the algorithm is to keep the candidate word that has the highest score out of the overlapping words. The sliding window method was able to increase performance significantly, offering 91 % raw classification accuracy. Using five writers for providing prototypes and the other five for testing, using manually segmented documents, 55 % precision was obtained at 50 % recall for the word shape method alone. The character-based method achieved 75 % precision at the same recall rate. The combined method is consistently better, resulting in about 80 % precision.

1.6 Features for Recognition

1.6.1 Isolated Characters

The features that are used most often for Arabic character recognition are either structural, such as loops, endpoints, cross points, branch points, and ascenders/descenders, or statistical, such as moments, Fourier descriptors, and projections. While some of these features may be good discriminative features as they offer direct correlation to Arabic structures, some cannot be extracted easily from training images. Loops appearing in characters might not always be closed, for instance, which makes their detection difficult.

An example of a statistical feature set is the WMR (Word Model Recognizer) feature set, which consists of 74 features (described in [47] among others). Two are global features—aspect and stroke ratio of the entire character. The remaining 72 are local features. Each character image is divided into 9 subimages. The distribution of the 8 directional slopes for each subimage form this set (8 directional slopes \times 9 subimages = 72 features). $F_{l_{i,j}} = s_{i,j}/N_i S_j$, $i = 1, 2, \dots, 9$, $j = 0, 1, \dots, 7$, where $s_{i,j}$ = number of components with slope j from subimage i , where N_i = number of components from subimage i , and $S_j = \max(s_{i,j}/N_i)$. These features are the basis of comparison for the character images derived from the segmentation of words to be recognized.

In [27], a lexicon-based method for recognizing English words is discussed, and in [12] a similar method is applied directly to Arabic for the purpose of word spotting.

An example of a structural feature set is one used which relies on the relationships between the basic stroke of the characters' shape described in [9]. Each character is thinned and its skeleton is extracted during pre-processing. It is then simplified to include only basic strokes of a predetermined set. Those basic strokes are lines in four directions, open curves in four directions, and loops. Once characters are simplified, connections between these basic strokes are identified and classified into classes. The character is represented by a propositional logic form that is specific to the classifier used in the system which is based on inductive logic programming.

1.6.2 Word Shape Features

Word level features describe features of the overall word.

One feature set used recently is the Generalized Shape Context (GSC) feature set, described in [53]. It has two types of features, distribution and local concavity, extracted from sub-frames of the word image or cells. The word image is pre-processed in order to extract the upper and lower baselines. The lower baseline is the “traditional” baseline where most ligatures occur, and the upper baseline is the height where many Arabic letter ascenders terminate. The image is also divided into vertical frames of equal width which are themselves divided into cells of equal

height. Features vectors are then extracted from each frame, each vector containing 24 features. Distribution features are essentially based on pixel densities. They are extracted using different density ratios: number of black pixels in one cell, ratio of black to white pixel in cell columns, positions of the centers of gravity, and a derivative feature that gives differences between positions of the centers of gravity of different frames. Local concavity features represent directions in turning points detected in the cells, some of them involving specifically pixels that are between the two baselines.

In [51] and [49] a word shape-based method is utilized for word spotting.

Another feature set generates a simplified representation of the word's skeleton by taking an approximation of line strokes and loops. A word is then represented by a two-dimensional feature vector for each line segment, the two components of the vector being the distance between the segment's endpoints and its orientation angle. From that representation, each segment is classified into one of the 60 classes of basic strokes using a clustering algorithm that outputs a sequence of basic predefined strokes which is then used for recognition. A chain code-based approach divides the image into vertical frames and each frame into five zones. For each zone a histogram of chain code slope is built with 45 steps. The histogram is normalized by dividing it by the height of the zone. A Kohonen self-organizing feature map (SOFM) algorithm is then used to limit the size of the feature set for the training phase.

1.6.3 Dealing with Dots

There are several approaches to dealing with the dots which distinguish Arabic characters from one another. Since words are rarely distinguished from one another exclusively by dots (as opposed to characters), some approaches simply ignore them, reducing the character classes. Other approaches try to assign the dot groups to specific subsegments. Yet another approach is to act on the word level, and look at dot groups independently of the main word itself.

1.7 Classification

1.7.1 Isolated Characters

Much work has been done on isolated characters. Several algorithms have been described for the recognition of numerals used in Farsi postal codes. In one such algorithm [37], the number is first smoothed and the skeleton is extracted using a thinning algorithm. It is then used to generate a set of feature points. Features points are intersection and terminal points. The character is then decomposed into primitives (segments that join feature points). Another algorithm attempts recognition using a quadtree-based fractal representation and an iterated function system. Both

of these systems attempt to recognize 8 of 10 of the numerals since only these numbers are used in Iranian zip codes; the other two numerals have appearances which can cause confusion due to similarity. A recognition rate of 94.44 % was obtained for numerals on the test sets they produced internally. These sets were gained from more than 200 people with different ages and different educational backgrounds. There are 480 samples per digit in the database for a total of 3840 digit images.

Another method for recognizing such digits was presented [36] where a quadtree is first built from the raw image of a character and then compressed using fractal encoding. Feature extraction uses vectors that have a variable number of components. Principal component analysis is then used to normalize the length of the vectors. After feature vectors are extracted and normalized, classification is done using two methods. The first method is a nearest neighbor classifier using the feature vectors and the signal to noise ratio between the fractal codes of the query image and the database used for training. The other method measures the minimum Euclidean distance between the query after one iteration of the coding algorithm and the decoded images of the database. The authors report a performance of 86.3 % on test sets with the nearest neighbor classifier and 90.6 % with the fractal transformation method.

Earlier, an algorithm was presented [1] where pre-segmented characters are represented by fuzzy direction sets. After a skeleton is extracted from the character image, a tree representation is built using four types of components present in the skeleton: single dot, directions, single loops, and double loops. A fuzzy number representation is built out of the tree structure and matched against a character model during the recognition step. The system had a recognition rate of 73.6 % with 9.4 % error and 17 % rejection.

A Bayesian classifier was used to classify isolated characters as well as characters arising from word segmentation. A preclassification step was performed on secondary character components, such as dots, based on heuristic thresholds for components' size and dimensions. The distance used for classification is shown below. It represents the distance between the observation vector x and the group i , m being the mean value of the group i and σ_i being its covariance matrix.

$$D(x)_i^2 = (x - m_i)^T + \sigma_i^{-1}(x - m_i) + \ln |\sigma_i| \quad (1.1)$$

Recognition of secondary characters in this method allows the authors to bring the number of classes from 100 to 64. The results of the two recognition steps are associated to give a final recognition score. The reported performance for isolated characters was 99.5 %.

1.7.2 Isolated Words

A competition was held for ICDAR 2005 [32] which compared a variety of contemporary systems in progress. The competition used the IFN/ENIT database of handwritten Arabic town names made available in 2002. This dataset was developed for the sake of advancing Arabic handwritten word recognition systems. Until

recently, such a large, public dataset was not available for Arabic, in contrast to the long and widely available English databases [41].

Five systems were submitted to the contest (see Sect. 17.3.1). Although one participant wanted all details about their system to remain confidential, the four remaining systems gave some rudimentary details about their approach to the problem. ICRA was a two-tiered neural net approach using a derived lexicon. TH-OCR was an extension of a multilingual recognition system utilizing statistical pattern recognition. UOB [17] is a pure hidden Markov model (HMM) system using a toolkit called HCM (details are in [17]). The system for handwritten word recognition uses the feature extraction discussed in [35]. REAM uses a hybrid planar Markov model to partition handwritten words into five logical horizontal bands. The approach was presented in [52]. The presenters of the competition also added the results of their system, presented in 2003, for comparison. Their system was based on a semicontinuous one-dimensional HMM [40].

Of the systems tested, the highest recognition rate on a novel dataset similar to IFN/ENIT was for UOB with a recognition rate of nearly 76 %, closely followed by the authors' own method, which achieved a recognition rate of about 75 %. The other systems' performances varied widely, ranging from 15–66 %, several suffering dramatically from apparent over-training. While the systems that performed the best were HMM based with a neural net approach, other systems using an HMM or neural network approach had very low recognition rates. This led the authors presenting the competition results to conclude that features, normalization, and recognition method all play critical parts in recognition performance.

The ICDAR 2005 competition results are as follows.

System Name	Top 1	Top 5	Top 10
ICRA	65.74	83.95	87.75
SHOCRAN	35.70	51.62	51.62
TH-OCR	29.62	43.96	50.14
UOB	75.93	87.99	90.88
REAM*	15.36	18.52	19.86
ARAB-IFN	74.69	87.07	89.77

*(tested with 3000 names)

A follow-up competition in ICDAR 2007 [33] met with much more participation, testing 14 systems by 8 groups, likely reflecting the increased interest with which Arabic recognition is being investigated. Märgner noted that more than 54 research groups are working with the IFN/ENIT database alone. In the 2007 competition, the systems were compared on overall recognition rate as well as speed. The winning system by Siemens had the highest recognition rate and was based on a hidden Markov recognizer. The system by CEDAR had the shortest average processing

time. While the range of performance reported in this competition was large, this reflected the wide range of techniques and feature sets that were utilized.

Nadir et al. [38] tried using two types of features with a number of artificial neural networks (ANNs) in order to recognize isolated words. Their neural networks were based on a three-layer perceptron. The number of neurons in the hidden layer is calculated by a heuristic, and the overall system is made from two identical ANN subsystems, each containing two ANNs. They have a statistical feature set based on pixel information. A word image is cut into multiple pieces with various shapes, and the features are the density of lit pixels in the regions. They also use a structural feature set expressed as a composition of structural units—they calculate such features as the number of ascenders, descenders, and loops. They used a restricted lexicon of 48 words used by Arabic writers when writing the literal amount field on a check. They had 4800 word images, the 48 words written by 100 different writers. The rejection criterion was chosen to keep a reliability of at least 99 %. They showed several combinations of their neural networks and had recognition rates in the low to mid-90 %'s with reject rates of around 5 %.

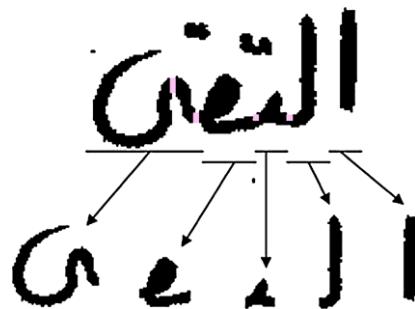
Yousef Al-Ouali et al. described a check recognition system in [3]. Checks include numerical information which can be compared with written amounts and therefore improve recognition. While recognition is performed, sub-words are extracted separately and combinations are tested and compared with the available numerical values using a context-sensitive grammar. A database was built to show the distribution of the different sub-words using a human tagging process which would separate different components (or objects) into four classes as a preliminary step (sub-words, numerical amounts, courtesy amount blocks and legal amount blocks). The limited number of possible sub-words allows a limited number of grammar rules to be used.

Lexicon-Based Approaches

“Lexicon-based” approaches make use of the sequences of Arabic characters for words appearing in a collection of words (lexicon) in order to select sets of prototype images representing the characters forming them. The word will be categorized as one of the words in the lexicon and receive a relevance score (how closely the word appears to be that prototype word). If the image's real word does not appear in the lexicon, it cannot be recognized correctly. In this case, although the word will be assigned to one in the lexicon, it is hoped that the relevance score will indicate that the word is, in fact, unrecognized.

Pre-processing steps often include line and word segmentation; it is most common for lexicon-based approaches to act on candidate words. In one algorithm [12], the candidate word image is first split into a sequence of segments (as in Fig. 1.15). The best possible sequence corresponds to the individual characters in the candidate word being separated. In general, it is better to “over-segment” the words rather than “under-segment” them because it is often easier to combine fragments of characters together than to recognize unsegmented characters. Segments are then rejoined and

Fig. 1.15 Arabic word “king” with segmentation points and corresponding best segments shown below



features extracted, which are in turn compared to features of prototype images of the characters. Further issues of under-segmenting unique to Arabic (such as characters which are not horizontally segmentable) can be dealt with using compound character classes.

A score that represents the match between the lexicon and the candidate word image is then computed. The score relates to the individual character recognition scores for each of the combined segments of the word image. Adjacent segments are compared to the character classes dictated as possibilities by a given lexicon entry. In the first phase of the match, the minimum Euclidean distance between the WMR features of candidate super-segments and the prototype character images is computed. In the second phase, a global optimum path is obtained using dynamic programming based on the saved minimum distances obtained in the first matching phase. The lexicon is ranked, the entries with the lowest total scores being the closest matches.

One HMM approach, intended for the recognition of city names, is based on dividing each word image into a given number of vertical frames from which feature vectors are extracted. Each entry in the lexicon will then have its own two-directional HMM representation which gives estimated transition probabilities between the different frames. The maximum number of forward jumps considered in the HMM was experimentally chosen between two and four.

For a given word model λ_c , if O is the set of K training samples, then

$$O = O^{(1)}, O^{(2)}, \dots, O^{(K)} \quad (1.2)$$

Each training word is represented by a set of observation symbols:

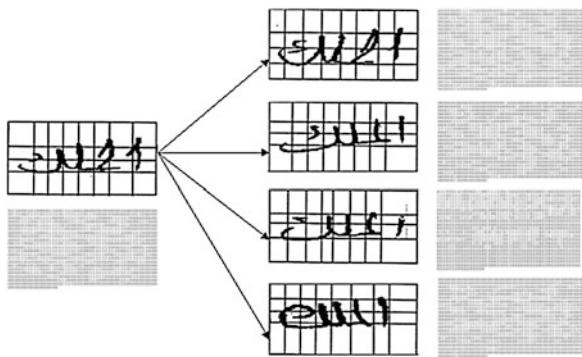
$$O^{(k)} = O_1^{(1)}, O_2^{(2)}, \dots, O_{T_k}^{(K)} \quad (1.3)$$

The probability of generating a word k with that model is then

$$P_k = P(O^{(k)} | \lambda_c) = \sum_{i=1}^N \alpha_{T_k}^{(k)}(i) \gamma_i \quad (1.4)$$

where $\alpha^{(k)}$ is the forward variable for a given word sample and γ_i are the last state distribution probabilities.

Fig. 1.16 A query word image (*left*) matched against four selected prototype images. The 1024 binary GSC features are shown next to the images



The authors used different training schemes with and without a smoothing factor for the HMM transition probabilities. The recognition rate increases dramatically when a high smoothing factor is applied and reaches 95 % for the top 20 results using a smoothing factor of 0.01. The performance is lower for a higher factor.

Word Shape Matching

In one word shape matching algorithm, binary features are compared using the correlation similarity measure 1.7.2 to obtain a similarity value between 0 and 1. This similarity score represents the extent of match between two word images. The smaller the score, the better the match. For word spotting, every word image in the test set of documents is compared with every selected prototype, and a distribution of similarity values is obtained. The distribution of similarity values is replaced by its arithmetic mean. Now every word is sorted in rank in accordance with this final mean score. Figure 1.16 shows an example query word image compared with a set of four selected prototypes.

Similarity Measure

The method of measuring the similarity or distance between two binary vectors is essential. The correlation distance performed best for GSC binary features [53] which is defined for two binary vectors X and Y , as in Eq. (1.5):

$$d(X, Y) = \frac{1}{2} \left(1 - \frac{s_{11}s_{00} - s_{10}s_{01}}{[(s_{10} + s_{11})(s_{01} + s_{00})(s_{11} + s_{01})(s_{00} + s_{10})]^{\frac{1}{2}}} \right) \quad (1.5)$$

where s_{ij} represent the number of corresponding bits of X and Y that have values i and j .

Summary of Approaches

Lorigo and Govindaraju [29] included a table of recognition engines in their survey. To put the state-of-the-art methods in some context, we have built upon their work to provide extended and reorganized versions, also comparing training, and testing set information as well as reported results.

The most common approaches to recognition generally involve some usage of hidden Markov models (HMMs) or artificial neural networks (ANNs).

Approaches involving HMMs are as follows:

Publication	Description/Results
El-Hajj et al., 2005 [16]	Character HMMs, 1D HMM system (analytical approach) IFN/ENIT database words with 8 or more instances (21,500 images), up to 87.20 % recognition
Safabakhsh, Adibi, 2005 [42]	Continuous-density variable-duration HMM with novel pre-processing, segmentation (targeting avoidance of under-segmentation); three Fourier descriptor features, five structural and discrete features Nastaaligh style words, 50 word lexicon, up to 91 % recognition rate on some words
Alma'adeed et al., 2004 [6]	Skeleton-based features extracted, rule-based classifier for global recognition, HMM used for trial classification 47 word lexicon, 4,700 images rejecting 10 % images in pre-processing, 100 writers, 45 % recognition rate
Khorsheed, 2003 [26]	Trains a single HMM with structural features; uses multiple character HMMs Historical manuscripts, 87 % recognition accuracy with spell check, 72 % without
Pechwitz, Märgner, 2003 [40]	Semicontinuous 1D HMM, character HMM for each variation of characters IFN/ENIT database (937 word lexicon), 89 % accuracy
Dehghan et al., 2001 [14]	Holistic approach, HMM for each word; uses right-left discrete HMM and Kohonen self-organizing vector quantization 198 Iranian city names, 17,000 images, 60 % training, 40 % testing, 65.0 % top 1, 76.1 % top 2, 95.0 % top 20

Approaches involving ANNs:

Publication	Description/Results
Farah et al., 2004 [19]	Holistic approach; ANN (multilayer perceptron with supervised training), K-NN, fuzzy K-NN using classifier combination 48 word lexicon written by 100 writers, 1,200 words for training, 3,600 for testing; 91 % recognition for neural network, 89.08 % for K-NN, 92.16 % for fuzzy K-NN, after classifier combination, 96 % recognition rate overall
Snoussi Maddouri, 2002 [31]	Transparent neural network, local/global vision modeling (GVM-LVM) at the word level; GVM uses structural features, LVM uses Fourier descriptors 70 word (from bank checks) lexicon, 2,070 images, 97 % recognition
Fahmy, Ali, 2001 [18]	Uses skeleton representation, features include locating endpoints, junctions, turning points, loops, generating frames (segmentation step), and detecting strokes 69.7 % word recognition rate on 600 words written by one writer
Souici-Meslati, 2004 [44, 45]	Knowledge-based ANN, perceptual features analysis for creating knowledge base, translation into ANN 55 word lexicon, 92 % recognition

HMMs and ANNs have been applied to individual character recognition as well:

Publication	Description/Results
Haraty, 2001, 2004 [24, 25]	Conventional initial segmentation, generation of pre-segmentation points, neural network verification About 4000 words written by students and faculty, about 70 % accuracy (later paper 73 %)
Dehghani, 2001 [15]	Two types of feature vectors based on regional projection contour transformation, two-stage recognition, multiple HMMs 92.76 % on training set, 71.82 % on testing (set size not specified)
Miled, 2001 [34]	Planar HMMs, architecture designed for printed Arabic sub-words, testing underway for handwritten “Encouraging” results
Amin, 1996 [10]	Skeleton-based tracing, neural network approach; combines rule-based (structural) and classification tests Trained with 2,000 characters, tested with 1,000 more by 10 writers, 92 % recognition

A variety of other interesting methods have been explored for recognizing individual characters over the years:

Publication	Description/Results
Mozaffari et al., 2005 [36]	Quadtree, fractal encoding, nearest neighbor, minimum Euclidean distance 8 digits (Iranian postal code digits), 280 image training, 200 testing (per digit), 86.3 % fractal nearest neighbor, 90.6 % fractal transformation
Mozaffari et al., 2005 [37]	Statistical method embedded with statistical features, principal component analysis, 94.44 % accuracy 8 digits (Iranian postal code digits), 280 image training, 200 testing (per digit), nearest neighbor, 94.44 % accuracy
Al-Shaher, 2003 [4]	Training point distribution models using the expectation-maximization algorithm Tested on 7 character classes, 100 samples of each character, 98.3 %
Amin, 2003 [9]	Rule-based, stroke types and relationship extracted automatically, inductive learning program generates first-order Horn clauses for characters 120 characters, 40 samples (30 training, 10 testing), 86.65 % accuracy
Clocksin, 2003 [13]	Discriminative support vector machine with 10-fold cross-validation; variety of segmentation methods/combinations of features, both on characters and words Syriac script, 91 % accuracy
Abuhaiba, 1998 [2]	Fuzzy sequential machine character recognition 13 pages, 13 writers for training, 20 pages, 20 writers testing Authors asked to write in a specific style, single stroke, etc. 55.4 % sub-word, 51.5 % character recognition rates
Abuhaiba, 1994 [1]	Noise-independent, produces skeletons reflecting structural relationships, converted to tree structure, uses fuzzy constrained character graph models and rule-based matching test data by 4 people, results varying from 73.6 % to 100 % with tuning based on input
Al-Yousefi, 1992 [5]	Statistical approach; primarily separated into dots/zigzags, secondary characteristics then identified; using quadratic discriminant classification 10 handwritten samples from database of 50, 81.0 %–98.79 % accuracy
Goraine, 1992 [23]	Eight-direction code used for stroke representation/classification at primary/secondary levels, contextual postprocessor for detecting errors and correction 180 words written neatly by three writers in a specific font, 90 % success
Almuallim, 1987 [7]	Skeleton representation, structural features, rules to join strokes into characters 400 samples, 2 writers, 91 % success

1.8 Document Search

1.8.1 Word Spotting

Recently, work has been started on spotting words in Arabic handwritten documents [48, 49, 51], and [12]. Given a set (or a page) of handwritten words, a common query for a user to ask is whether or not a specific word is among that set. More generally, given a page of handwritten text, a common query is whether or not a word or words of interest appear on the page. Such a query allows a user to sift through a set of documents for a subset of documents that is of most interest. This is the motivation behind the word spotting problem: given a scanned image of a handwritten document and a Unicode sequence of characters (the query word), the word spotting problem asks if the image contains a handwritten image of the query word and, if so, at what coordinates in the image does the word exist.

An algorithm and a system for searching handwritten Arabic documents to locate such key words was presented. In this approach, the system had three main components: a word segmenter, a shape-based matcher for words, and a search interface. Two steps are involved in the search. First, the query is used to obtain a set of handwritten prototype samples of that word from a known set of writers. In the second step, the prototypes are used to spot each occurrence of the words in the documents to be searched. The performance of the system was tested on a database of 20,000 word images contained in 100 scanned handwritten Arabic documents written by 10 different writers. On average, if five authors were used for providing prototypes and the other five for testing, using manually segmented documents, 55 % precision was obtained at 50 % recall, with increased performance if more writers were used for training.

1.8.2 Versatile Search

The algorithm was later extended and incorporated into a framework for versatile search [50]. Versatile search is a framework by which the query can be either text or image and the retrieval method is a fusion of text and image retrieval methods. A Unicode and an image query are maintained throughout the search, with the results being combined by a neural network. Preliminary results show positive results that can be further improved by refining the component pieces of the framework (text transcription and image search). The conclusion was that processing image- and text-based queries in parallel can result in higher performance than either alone, boosting precision of the same queries from roughly 55 % to 80 % at 50 % recall.

1.9 Databases of Handwritten Arabic Text

Until recently, a large, public dataset was not available for Arabic, in contrast to the long and widely available English databases. The IFN/ENIT database of handwrit-

ten Arabic town names was the first to be made available in 2002. Other databases were used like the database of Arabic checks of the Saudi Al Rajhi bank, but were not made available. Following is a list of the databases used by different recognition systems either as training data or as lexicons:

Database	Description/Results
Dehghan 2001 [14]	More than 17,000 images of 198 Iranian city names
Fahmy 2001 [18]	600 words by one writer (300 different words, each written twice)
Alma'adeed 2004 [6]	10,000 words used on bank checks by 100 writers
Amin 2003 [9]	4,800 samples of isolated characters
Khorsheed 2003 [26]	Ancient historical manuscript
Pechwitz 2003 [40]	26,459 images of Tunisian city names, by 411 writers
Haraty 2001, 2004 [24, 25]	Ligatures and characters, handwritten in shapes they would have in words
Mozaffari 2005 [36]	Database of numerals, 480 samples per digit written by more than 200 people

1.10 Construction of a Handwritten Arabic Corpus

1.10.1 *Corpus Collection*

Most available corpora for Arabic consist of single word images. Perhaps the largest bottleneck to continued research in Arabic handwriting recognition is that of a lack of available large-scale corpora. Without such resources, developing and testing algorithms for Arabic documents is highly problematic since there is difficulty in judging the generality and effectiveness of results.

1.10.2 *Handwritten Documents*

One much-needed corpus needs to be created consisting of truthed, unconstrained document images written by several different authors. Many interesting and challenging problems relate to the processing of real-world documents, which often take this form. Thus far, some authors have resorted to creating pseudo-documents out of single word images. Such approaches allow for theoretical testing only. Not only is the generation of such documents inconvenient, researchers have to resort to generating methods by testing on artificial data, and to report results that have never actually been tested on real documents. While some methods may not suffer from being developed on pseudo-documents, others rely on the natural flow of writing from a single author, or rely on more complicated language models that are difficult to approximate.

Character Images

Another needed corpus is a large set of truthed handwritten character images, taken from naturally written words. A large database of such images is useful in extracting generic features of Arabic handwriting used in character-based recognition approaches. Such approaches are among the highest performing in English recognition, and recent results show promise for Arabic as well.

A large database is necessary to generate sufficient variation in the writing of the characters for each author. A variety of authors is necessary to create generalized features of different writing styles. Since individual character images written alone may have significant differences with those written in the natural flow of a sentence, it is necessary to extract them from words and not simply have them written alone. In addition samples where the authors are making an effort to write neatly have some use, but it is more important to get natural writing since the goal is to process naturally written documents.

1.10.3 Execution Plan

Since there are very few databases applicable to Arabic, there are many areas of need. However, the following set of data is the most needed.

1. Full, Handwritten Pages—Examples of full handwritten pages should be obtained and truthed. Though difficult, the needed number is at least in the hundreds.
2. Character Data—A minimum of several hundred examples of each variation of each Arabic letter.
3. Representative of Many Styles—It is essential to encompass the writing styles of a variety of authors, preferably representing all common variations of writing. This means the documents from which the samples are drawn should include a diverse set of authors (in terms of age, gender, geographic origin, etc.).
4. Hierarchical—For page data, truth should be available in a hierarchical fashion, tagging each page, paragraph, line, word, and character with truth values. Documents often contain multiple layers, incorporating such items as logos, printed text, signatures, handwritten notes, and stamps. Each of these layers should be isolated individually when possible.
5. Real World—Real-world documents are necessary to avoid the inherent issues in collecting data specifically for the purpose of corpora, such as authors writing too neatly.

One difficulty is finding a source for such documents, due to such issues as privacy. Very good sources include handwritten notes and forms. Full page corpus generation can be greatly aided by transcription mapping.

Document Representation

XML is one efficient way to address the hierarchical requirement listed above. XML also has benefits such as data interoperability. One such document representing XML structure was recently used in storing CEDARABIC documents by CEDAR, with the following DTD:

```
<!ELEMENT File (Document,Writer,Page,Words)>
<!ATTLIST File Name CDATA #REQUIRED>
<!ELEMENT Document (#PCDATA)>
<!ELEMENT Writer (#PCDATA)>
<!ELEMENT Page (#PCDATA)>
<!ELEMENT Words (word+)>
<!ELEMENT word (top,left,right,bottom,Pronunciation,
               Meaning,Alphabets)>
<!ATTLIST Word Line_Number CDATA #REQUIRED>
<!ATTLIST Word Word_Number CDATA #REQUIRED>
<!ELEMENT top (#PCDATA)>
<!ELEMENT left (#PCDATA)>
<!ELEMENT right (#PCDATA)>
<!ELEMENT bottom (#PCDATA)>
<!ELEMENT Alphabets (#PCDATA)>
<!ELEMENT Pronunciation (#PCDATA)>
<!ELEMENT Meaning (#PCDATA)>
```

The above DTD describes the following tags:

- <File>—Encloses the contents of the entire file. Contains the file name in the attribute File Name. Encloses the tags <Document> <Writer> <Page> <Words> . There can be only one <File> tag per XML file.
- <Document>—Contains the value of the document number. There can be only one <Document> tag per XML file.
- <Writer>—Contains the unique number assigned to the writer who wrote the document. There can be only one <Writer> tag per XML file.
- <Page>—Contains the page number of that file. Each file contains only a single image of the handwritten document, and hence there can be only one <Page> per XML document, which specifies if the image corresponds to the first or second page.
- <Words>—This tag encloses all the words in the document using a <word> tag for each word.
- <word>—The word tag gives the following details about that particular word. The Word Number and Line Number are represented as attributes of this tag. It encloses the <top>, <left>, <right>, <bottom>, <Alphabets>, <Pronunciation> and <Meaning> tags.
- <top>, <left>, <right>, <bottom>—These tags give the *x*-, *y*-coordinates of the top, left and right, and bottom of the bounding box that encloses that word.

- <Alphabets>—This tag gives the alphabet sequence of the word, which contains the root alphabets separated by order.
- <Pronunciation>—This gives the pronunciation of the word in English.
- <Meaning>—This tag gives the meaning of the word in English.

The following is the structure of a sample XML file:

```
<File Name="001_1_1.png"
<Document>1</Document>
<Writer>001</Writer>
<Page>1</Page>
<words>
<word Line_Number="1" Word_Number="1">
<top>299</top>
<left>2049</left>
<right>2187</right>
<bottom>387</bottom>
<alphabets>Alef|Lam|Teh|Qaf|Alef maksura|</alphabets>
<pronunciation>eltaqa</pronunciation>
<meaning>met</meaning>
</word>
<word Line_Number="1" Word_Number="2">
<top>303</top>
<left>1862</left>
<right>2017</right>
<bottom>395</bottom>
<alphabets>Reh|Yeh+hamza|Yeh|Seen|</alphabets>
<pronunciation>raees</pronunciation>
<meaning>leader-Head</meaning>
</word>
<word Line_Number="1" Word_Number="3">
<top>283</top>
<left>1679</left>
<right>1840</right>
<bottom>379</bottom>
<alphabets>Alef|Lam|Lam|Qaf|Alef|Hamza|</alphabets>
<pronunciation>alleqa?</pronunciation>
<meaning>meeting</meaning>
</word>
</words>
</File>
```

This structure can be extended to allow for layers mentioned, such as logos, printed text, signatures, handwritten notes, and stamps, simply by adding corresponding tags.

1.11 Technology Gaps and Conclusions

1.11.1 Character and Word Recognition

Section 1.7 discussed many of the recent developments in character and word recognition performance. While some results are encouraging, lack of a standard benchmarking dataset has made it difficult to compare reported results. Additionally, since Arabic is a cursive script, isolated character recognition has limited utility in recognizing words. Word recognition needs significant improvements to approach even the current levels of recognition for other languages, such as English.

1.11.2 Word Spotting

Word spotting has only recently received any attention. Reported performance results are steadily rising, but further progress need be made before it is on the level of modern English algorithms. Section 1.8 discussed some of the recently presented algorithms for document search. This result oriented research is especially interesting because it is focused on generating useful information rather than the general task of recognition.

1.11.3 Document Decomposition

A real-world document image may consist of text (handwritten or machine printed), line drawings, tables, diagrams, pictures, icons, etc. To efficiently recognize the entire image, it is necessary to decompose a document into component parts. Since Arabic recognition is still in a relative infancy period, such document decomposition is not well developed.

1.11.4 Hand Drawn Diagram Recognition

Hand drawn diagrams in Arabic have many features in common with such diagrams drawn in other languages. For this reason, little research has been done explicitly on the problem. One interesting feature of analyzing such diagrams may be the differences, if any, caused by the right-to-left script.

1.11.5 Phone Number, Date, and Address Recognition

As mentioned earlier in this document, one of the most extensively researched areas in Arabic involves address and number recognition. This is at least in part due

to the availability of publicly available datasets involving Tunisian city names and bank check amounts. Phone number and date recognition has received less attention. Number recognition has some issues in common with recognizing Arabic, since Arabic has its own set of digits (as opposed to the numerals used in languages such as English).

1.11.6 Transcription and Transcription Mapping

Transcription of handwritten documents allows for easier and more accurate document processing, such as searches. Transcript mapping is the alignment of words in a text file with image regions in a document, allowing for speedier corpus building. Transcript mapping is relatively new to Arabic; the first such method is to appear shortly [30].

References

1. Abuhaiba, I.S.I., Mahmoud, S.A., Green, R.J.: Recognition of handwritten cursive Arabic characters. *IEEE Trans. Pattern Anal. Mach. Intell.* **16**(6), 664–672 (1994)
2. Abuhaiba, I., Holt, M., Datta, S.: Recognition of off-line cursive handwriting. *Comput. Vis. Image Underst.* **77**, 19–38 (1998)
3. Al-Ohali, Y., Cheriet, M., Suen, C.Y.: Databases for recognition of handwritten Arabic cheques. In: Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition (2000)
4. Al-Shaher, A., Hancock, E.: Learning mixtures of point distribution models with the EM algorithm. *Pattern Recognit.* **36**, 2805–2818 (2003)
5. Al-Yousefi, H., Udupa, S.: Recognition of Arabic characters. *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 853–857 (1992)
6. Almaadeed, S., Higgens, C., Elliman, D.: Off-line recognition of handwritten Arabic words using multiple hidden Markov models. *Knowl.-Based Syst.* **17**, 75–79 (2004)
7. Almuallim, H., Yamaguchi, S.: A method of recognition of Arabic cursive handwriting. *IEEE Trans. Pattern Anal. Mach. Intell.* **9**, 715–722 (1987)
8. Amara, N.E.B., Bouslama, F.: Classification of Arabic script using multiple sources of information: State of the art and perspectives. *Int. J. Doc. Anal. Recognit.* **5**(4), 195–212 (2005)
9. Amin, A.: Recognition of hand-printed characters based on structural description and inductive logic programming. *Pattern Recognit. Lett.* **24**, 3187–3196 (2003)
10. Amin, A., Al-Sadoun, H., Fischer, S.: Hand-printed Arabic character recognition system using an artificial network. *Pattern Recognit.* **29**, 663–675 (1996)
11. Arivazhagan, M., Srinivasan, H., Srihari, S.: A statistical approach to line segmentation in handwritten documents. In: Proceedings of SPIE (2007)
12. Ball, G., Srihari, S., Srinivasan, H.: Segmentation-based and segmentation-free methods for spotting handwritten Arabic words. In: IWFHR (2006)
13. Clocksin, W., Fernando, P.: Towards automatic transcription of Syriac handwriting. In: Proc. Intl Conf. Image Analysis and Processing (2003)
14. Dehghan, M., Faiez, K., Ahmadi, M., Shridhar, M.: Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM. *Pattern Recognit.* **34**, 1057–1065 (2001)

15. Dehghani, A., Shabani, F., Nava, P.: Off-line recognition of isolated Persian handwritten characters using multiple hidden Markov models. In: Proc. Intl. Conf. Information Technology: Coding and Computing (2001)
16. El-Hajj, R., Likforman-Sulem, L., Mokbel, C.: Arabic handwriting recognition using baseline dependent features and hidden Markov modeling. In: Proc. Intl. Conf. Document Analysis and Recognition (2005)
17. El-Hajj, R., Likforman-Sulem, L., Mokbel, C.: Arabic handwriting recognition using baseline dependent features and hidden Markov modeling. In: ICDAR '05: Proceedings of the Ninth International Conference on Document Analysis and Recognition. IEEE Comput. Soc., Seoul (2005)
18. Fahmy, M., Ali, S.A.: Automatic recognition of handwritten Arabic characters using their geometrical features. *Studies in Informatics and Control* **10** (2001)
19. Farah, N., Souici, L., Farah, L., Sellami, M.: Arabic words recognition with classifiers combination: an application to literal amounts. In: Proc. Artificial Intelligence: Methodology, Systems, and Applications (2004)
20. Farooq, F., Govindaraju, V., Perrone, M.: Pre-processing methods for handwritten Arabic documents. In: ICDAR '05: Proceedings of the Ninth International Conference on Document Analysis and Recognition, vol. 1. IEEE Comput. Soc., Seoul (2005)
21. Femiani, J.C., Phielipp, M., Razdan, A.: A system for discriminating handwriting from machine print on noisy Arabic datasets. In: SDIUT '05: Proceedings of the Symposium on Document Image Understanding Technology, College Park, Maryland (2005)
22. Freeman, H.: Techniques for the digital computer analysis of chain-encoded arbitrary plane curves. In: Proceedings of the National Electronics Conference, vol. 17 (1961)
23. Goraine, H., Usher, M., Al-Emami, S.: Off-line Arabic character recognition. *Computer* **25**, 71–74 (1992)
24. Haraty, R., Ghaddar, C.: Arabic text recognition. *Int. Arab J. Inf. Technol.* **1**, 156–163 (2004)
25. Haraty, R., Hamid, A.: A neuro-heuristic approach for segmenting handwritten Arabic text. In: ACS/IEEE International Conference on Computer Systems and Applications (2001)
26. Khorsheed, M.: Recognising handwritten Arabic manuscripts using a single hidden Markov model. *Pattern Recognit. Lett.* **24**, 2235–2242 (2003)
27. Kim, G., Govindaraju, V.: A lexicon driven approach to handwritten word recognition for real time applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **19**(4), 366–379 (1997)
28. Lorigo, L., Govindaraju, V.: Segmentation and pre-recognition of Arabic handwriting. In: ICDAR '05: Proceedings of the Ninth International Conference on Document Analysis and Recognition, vol. 2. IEEE Comput. Soc., Seoul (2005)
29. Lorigo, L., Govindaraju, V.: Off-line Arabic handwriting recognition: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(5), 712–724 (2006)
30. Lorigo, L.M., Govindaraju, V.: Transcript mapping for handwritten Arabic documents. In: Proceedings SPIE (2007, to appear)
31. Maddouri, S.S., Amiri, H., Belaid, A., Choisy, C.: Combination of local and global vision modeling for Arabic handwritten words recognition. In: Proc. Intl Conf. Frontiers in Handwriting Recognition (2002)
32. Märgner, V., Pechwitz, M., Abed, H.: ICDAR 2005 Arabic handwriting recognition competition. In: ICDAR '05: Proceedings of the Ninth International Conference on Document Analysis and Recognition, vol. 1. IEEE Comput. Soc., Seoul (2005)
33. Märgner, V., Pechwitz, M., Abed, H.: ICDAR 2007—Arabic handwriting recognition competition. In: ICDAR '07: Proceedings of the Tenth International Conference on Document Analysis and Recognition. IEEE Comput. Soc., Los Alamitos (2007)
34. Miled, H., Amara, N.B.: Planar Markov modeling for Arabic writing recognition: advancement state. In: Proc. Intl. Conf. Document Analysis and Recognition (2001)
35. Mokbel, C., Akl, H.A., Greige, H.: Automatic speech recognition of Arabic digits over telephone network. In: Proceedings of RTST (2002)
36. Mozaffari, S., Faez, K., Ziaratban, M.: Character representation and recognition using quad tree-based fractal encoding scheme. In: ICDAR '05: Proceedings of the Ninth International

- Conference on Document Analysis and Recognition, vol. 2. IEEE Comput. Soc., Seoul (2005)
- 37. Mozaffari, S., Faez, K., Ziaratban, M.: Structural decomposition and statistical description of Farsi/Arabic handwritten numeric characters. In: ICDAR '05: Proceedings of the Ninth International Conference on Document Analysis and Recognition, vol. 1. IEEE Comput. Soc., Seoul (2005)
 - 38. Nadir, F., Abdellatif, E., Tarek, K., Mokhtar, S.: Benefit of multiclassifier systems for Arabic handwritten words recognition. In: ICDAR '05: Proceedings of the Ninth International Conference on Document Analysis and Recognition, vol. 1. IEEE Comput. Soc., Seoul (2005)
 - 39. Olivier, C., Miled, H., Romeo, K., Lecourtier, Y.: Segmentation and coding of Arabic handwritten words. In: Proceedings of the International Conference on Pattern Recognition (1996)
 - 40. Pechwitz, M., Märgner, V.: HMM based approach for handwritten Arabic word recognition using the IFN/ENIT—database. In: ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition. IEEE Comput. Soc., Edinburgh (2003)
 - 41. Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H., et al.: IFN/ENIT-database of handwritten Arabic words. In: Proc. CIFED 2002, Hammamet, Tunisia, October 21–23, 2002
 - 42. Safabakhsh, R., Adibi, P.: Nastaaligh handwritten word recognition using a continuous-density variable-duration HMM. *Arab. J. Sci. Eng.* **30**, 95–118 (2005)
 - 43. Sari, T., Souici, L., Sellami, M.: Off-line handwritten Arabic character segmentation algorithm: ACSA. In: Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (2002)
 - 44. Souici, L., Farah, N., Sari, T., Sellami, M.: Rule based neural networks construction for handwritten Arabic city-names recognition. In: Proc. Artificial Intelligence: Methodology, Systems, and Applications (2004)
 - 45. Souici-Meslati, L., Sellami, M.: A hybrid approach for Arabic literal amounts recognition. *Arab. J. Sci. Eng.* **29**, 174–194 (2004)
 - 46. Sridharan, K., Farooq, F., Govindaraju, V.: Classification of machine print and handwriting in mixed Arabic documents. In: SDIUT '05: Proceedings of the Symposium on Document Image Understanding Technology College Park, Maryland (2005)
 - 47. Srihari, S.N., Tomai, C.I., Zhang, B., Lee, S.: Individuality of numerals. In: ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition. IEEE Comp. Soc., Washington (2003)
 - 48. Srihari, S., Srinivasan, H., Babu, P., Bhole, C.: Handwritten Arabic word spotting using the CEDARABIC document analysis system. In: SDIUT '05: Proceedings of the Symposium on Document Image Understanding Technology College Park, Maryland (2005)
 - 49. Srihari, S.N., Srinivasan, H., Babu, P., Bhole, C.: Handwritten Arabic word spotting using the CEDARABIC document analysis system. In: Proc. Symposium on Document Image Understanding Technology (SDIUT-05), College Park, MD (2005)
 - 50. Srihari, S., Ball, G., Srinivasan, H.: Versatile search of scanned Arabic handwriting. In: SACH'06: Summit on Arabic and Chinese Handwriting (2006)
 - 51. Srihari, S., Srinivasan, H., Babu, P., Bhole, C.: Spotting words in handwritten Arabic documents. In: Proceedings SPIE, San Jose, CA (2006)
 - 52. Touj, S., Amara, N.B., Amiri, H.: Arabic handwritten words recognition based on a planar hidden Markov model. *Int. Arab J. Inf. Technol.* **2**(4), 318–325 (2005)
 - 53. Zhang, B., Srihari, S.N.: Binary vector dissimilarity measures for handwriting identification. In: Document Recognition and Retrieval X, vol. 5010. SPIE, Bellingham (2003)

Chapter 2

Layout Analysis of Arabic Script Documents

Syed Saqib Bukhari, Faisal Shafait, and Thomas M. Breuel

Abstract Layout analysis—extraction of text lines from a document image and identification of their reading order—is an important step in converting the document into a searchable electronic representation. Projection methods are typically employed for extraction of text lines in Arabic script documents. Although projection methods achieve good accuracy on clean, skew-free documents, their performance drops under challenging situations (border noise, skew, complex layouts, etc.). This chapter presents a layout analysis system for extracting text lines in reading order from scanned Arabic script document images written in different languages (Arabic, Urdu, Persian, etc.) and different styles (Naskh, Nastaliq, etc.). The presented system is based on a suitable combination of different well-established techniques for analyzing Latin script documents that have proven to be robust against different types of document image degradations.

2.1 Introduction

Layout analysis deals with text line detection and their reading order determination in document images. The wide variety of layouts in large-scale document digitization projects poses stern challenges to document image analysis. A document image may contain different types of contents like text, graphics, halftones, etc. The goal of optical character recognition (OCR) is to extract text from a document image. This is achieved in two steps. The first step, geometric layout analysis, locates text lines in the image and identifies their reading order. In the second step, text lines

S.S. Bukhari (✉) · T.M. Breuel
Technical University of Kaiserslautern, Kaiserslautern, Germany
e-mail: bukhari@informatik.uni-kl.de

T.M. Breuel
e-mail: tmb@informatik.uni-kl.de

F. Shafait
German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
e-mail: faisal.shafait@dfki.de

وفي نهاية اللقاء شكرت هديل الوفد والشابات صاحبات المشاريع على الجدية والحماسة في الدفاع عن مشاريعهن وتعهدت بتقديم يد العون والمساعدة على مختلف الأصعدة ضمن وزارة شؤون المرأة ودعمهن في مختلف القطاعات الأخرى للوصول إلى الهدف المنشود في سبيل الحفاظ على الاستمرارية في المشاريع من خلال التدريب والتثقيف والتواصل والتشبيك مع الجهات المعنية للوصول إلى المرحلة الانتاجية.

(a) Sample Arabic document written in Naskh script.

مجھ سے بے محابا سن لو! اور میرے اپر لوگوں کے نام پیش کر کے جھٹ جھٹ کرنا کر کہنے لگو۔ یہ رئے یوں کہا اور ابراہیم بن ادھم نے یوں فرمایا۔ کیونکہ جس نے حضور صلی اللہ علیہ وسلم اور صاحبہ کرام سے جھٹ پچھٹی اس کی دلیل اور جھٹ سب سے قوی اور مضبوط ہے۔

(b) Sample Urdu document written in Nastaliq script.

Fig. 2.1 An example of printed Arabic text in Naskh script and Urdu text in Nastaliq script. Text lines in Nastaliq have very little spacing between them compared to Naskh script

identified by the layout analysis step are fed to a character recognition engine which converts them into text in an appropriate format (ASCII, UTF-8, etc.).

The Arabic script is used for writing several languages of Asia and Africa, e.g., Arabic, Urdu, Persian, Pashto, Kurdi, and Jawi. After Latin script, it is the second most widely used script in the world. It is a cursive script; i.e., individual characters are usually combined to form ligatures. Although there are many styles for writing Arabic script, the most widely used styles are Naskh and Nastaliq. The Naskh writing style is dominant in Arabic and Pashto languages, whereas Nastaliq is the standard style adopted for writing Urdu and Persian. Examples of printed Arabic text written in Naskh script and Urdu text written in Nastaliq script are shown in Fig. 2.1. From a layout analysis point of view, the main differences of Nastaliq script as compared to Naskh script are: (i) very small interline and interword spacing and (ii) tall ascenders and descenders that overlap into adjacent text lines.

Research on Arabic script OCR has primarily been focused on word recognition [1], and very few approaches have been proposed for text line extraction from machine printed Arabic script document images. Since Arabic is generally written in Naskh script, text line segmentation using horizontal projections works quite well on machine printed documents due to the large interline spacing [22]. Segmentation of a page image into individual lines by horizontal projection is a primitive approach and works only on clean, single-column documents with large interline spacing. To handle multi-column documents, either the $x-y$ cut method [29] is used, or morphological operations are employed to get text blocks [38], which can then be further subdivided into individual text lines by horizontal projection. More sophisticated approaches for text line extraction have been presented in the domain of segment-

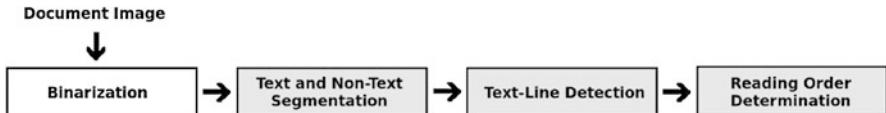


Fig. 2.2 Processing flow of a high performance generic layout analysis system. *Filled blocks* show the areas that this chapter discusses in detail

ing handwritten Arabic documents [4, 41]. However, the key problem addressed in these approaches is handling the local nonlinearity of text lines.

Over the last two decades, several layout analysis algorithms have been proposed in the literature (for a literature survey, please refer to [11, 28]) that work for different layouts and are quite robust to the presence of noise in the document. Many of these algorithms have come into widespread use for analyzing document images in different scripts. Kumar et al. [24] have evaluated the performance of six algorithms for page segmentation on Nastaliq script: the $x-y$ cut [29], the smearing algorithm [40], whitespace analysis [2], the constrained text line finding algorithm [5], Docstrum [30], and the Voronoi-diagram based approach [23]. These algorithms work very well in segmenting documents in Latin script, as shown in [37]. However, when Kumar et al. applied these algorithms to segment Nastaliq script documents, none of these algorithms was able to achieve an accuracy of more than 70 % on their test data, which had simple book layouts with no font size variations within each page.

Contrary to Arabic OCR, there has been very little work in the area of Urdu or Persian document analysis. Husain et al. [19] proposed an Urdu character recognition system for the Nastaliq script. Urdu is written in Nastaliq script using more than 20,000 ligatures [16]. Husain et al. skipped the layout analysis step to concentrate more on the OCR part. Pal et al. [32] presented an approach for recognizing printed Urdu documents. First, they perform skew correction of the document using a Hough transform. Text lines in the skew corrected document are then segmented by horizontal projection. A similar approach is used by Jelodar et al. [20] to extract text lines from printed Persian documents.

Shafait et al. [36] have presented an adaptation of the layout system described in [6] to Urdu script documents. First, they evaluate empty whitespace rectangles as candidates for column separators or gutters. Text lines are then detected by modifying a RAST-based (Recognition by Adaptive Subdivision of Transformation Space) text-line finding algorithm [5] where column separators are introduced as “obstacles.” Finally, text lines are analyzed for determining the reading order using constraints on the geometric arrangement of text line segments on the page. Particular advantages of their system are that it is nearly a parameter-free approach and robust to the presence of noise in document images.

In this chapter we present a layout analysis system that is an extension of the approach presented in [36] and is applicable to a wide variety of Arabic script binary document images. A grayscale document image can be first converted into binary form using an appropriate binarization approach such as those in Otsu [31] and Sauvola [35], which are commonly used state-of-the-art binarization approaches.



Fig. 2.3 A sample newspaper image printed in Arabic Naskh script and its corresponding layout analysis result: *gray region* represents non-text components, *color-coded labeling* represent segmented text lines, and *magenta line* shows text line reading order. [Note: *left* image has been taken from the website [17]]

A possible flow of a generic layout analysis system is shown in Fig. 2.2. Here, we will discuss text and non-text segmentation, text line detection, and reading order determination. For a sample Arabic script document image, the output of the layout analysis system is shown in Fig. 2.3.

The rest of the chapter is organized as follows. In Sect. 2.2, the multiresolution morphology-based text and non-text segmentation algorithm [3, 10] is described. State-of-the-art $x-y$ cut [29] and ridge-based [7] text line finding methods are explained in Sect. 2.3. A topological sorting-based reading order determination algorithm [36] is discussed in Sect. 2.4, followed by the conclusion in Sect. 2.5.

2.2 Text and Non-text Segmentation

Text and non-text segmentation is the process of separating text and non-text elements in document images. It is an important initial step in document image processing like optical character recognition (OCR) systems. A character recognition engine is designed for recognizing text elements, and it produces garbage for non-text elements.

Different approaches have been proposed in the literature for text and non-text segmentation. Wong et al. [40] presented a classical smearing-based page segmentation approach. Bloomberg [3] introduced a method for text and halftone segmentation using multiresolution morphology. Other state-of-the-art text and non-text segmentation approaches can be generally categorized as classification-based

approaches such as the pixel [27], connected component [9], or zone [21, 39] classification-based text and non-text segmentation methods. In general, the accuracy of classification-based text and non-text segmentation approaches heavily depends on the training samples.

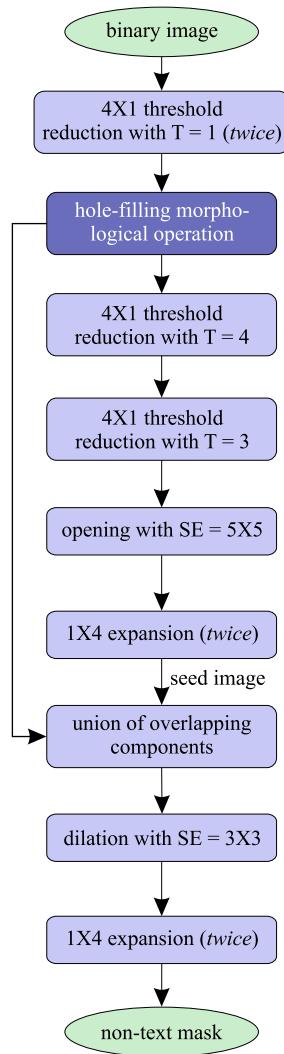
The multiresolution morphology-based method [3] was specifically designed for separating halftones from document images. It works well for halftone segmentation, but not for other types of non-text elements like drawings, maps, etc. It is a simple approach, based on the assumption that the size of non-text elements is larger than text elements in document images. We presented an improvement to the multiresolution morphology-based text and non-text segmentation method in [10] which can also segment drawing type non-text elements. A data flow diagram of the improved version of Bloomberg's text and non-text segmentation method is shown in Fig. 2.4. A brief description of Bloomberg's multiresolution morphology-based text and non-text segmentation method [3] and our improved version [10] is described below.

Bloomberg introduced the concept of *threshold reduction* for subsampling of document images, which is defined as follows. Consider a binary document image where a foreground pixel is represented by 1 and a background pixel is represented by 0. Each 2×2 pixel block in the document image is replaced by a single value, either 1 or 0, in a corresponding subsampled image. The value is set to 1 if the sum of values in a particular 2×2 pixels block is greater than or equal to some predefined threshold value; otherwise the value is set to 0. This threshold reduction operation mimics the process of image dilation for a threshold value equal to 1 and erosion for a threshold value equal to 4 that follows subsampling of each 2×2 pixel block by its upper left pixel. The threshold reduction is also referred as multiresolution morphology.

Bloomberg used the concept of threshold reduction for implementing the text and halftone segmentation method that is shown in Fig. 2.4. An input image is first processed by two threshold reduction operations, both with threshold value equal to 1. These threshold reduction operations produce a subsampled image. The subsampled image is further processed by two threshold reductions with threshold values equal to 4 and 3, respectively, and a morphological opening operation. The output image is referred to as a seed image. The seed image, after expansion, is compared with the subsampled image for generating a halftone-mask image. The halftone-mask image is composed of fully or partially overlapped components between the seed image and the subsampled image. The halftone-mask image is finally processed by a morphological dilation operation.

The performance of the multiresolution morphology-based text and halftone segmentation method depends upon the residual portions of halftones of an input document image in its corresponding seed image. In a document image, non-text elements (like drawings, maps, graphs, and even halftones) may also be composed of line art. Threshold reduction operations wipe out these types of non-text elements in the corresponding seed image.

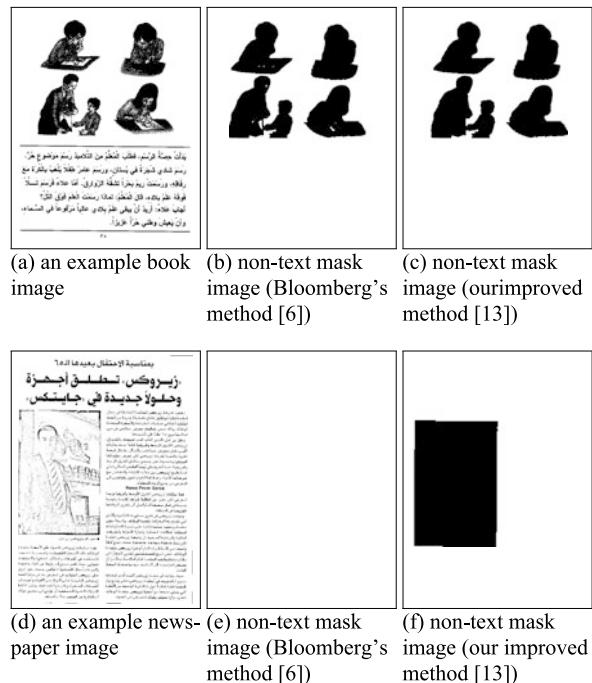
Fig. 2.4 Data flow diagram of improved version of Bloomberg's text and non-text segmentation algorithm [10]. The original Bloomberg's text and non-text algorithm [3] is equivalent to the given data flow diagram without the hole-filling operation. (Note: T: threshold; SE: structuring element)



We introduced an improved version of the multiresolution morphology-based segmentation method that can handle non-text elements like halftones, drawings, and graphics, etc. In the improved version, which is shown in Fig. 2.4, the sub-sampled image is first processed by a hole-filling morphological operation. The hole-filling operation fills drawing type non-text elements, with a better possibility of keeping the residual portions of these non-text elements in the seed image.

Figure 2.5 shows sample Arabic script document images and their text and non-text segmentations for the original version of the multiresolution morphology-based text and non-text segmentation method [3] and its improved version [10]. Our im-

Fig. 2.5 Results of non-text mask using Bloomberg's multiresolution morphology-based text and non-text segmentation method [3] and our improved version [10]. Top figure shows a simple case where non-text components are larger than text components and can also be separated by using median size of connected components analysis. Bottom figure shows a challenging condition where non-text components are comparable to or even smaller than text components. In contrast to [3], improved multiresolution morphology-based text and non-text segmentation algorithm gives correct result for both simple and challenging conditions



proved version performs well in these examples compared to the original version. In Arabic script document images, like Latin script images, the size of text elements is usually smaller than that of non-text elements, which fits well the assumption of multiresolution morphology-based text and non-text segmentation. Therefore, this approach also works well for Arabic script document images.

2.3 Text Line Detection

Text line detection is an important layout analysis step in document image processing. It is often used before feeding a page to a character recognition engine. The performance of the text line detection operation directly influences the accuracy of the recognition engine.

In the literature, a large number of text line detection approaches are proposed for Arabic document images. Among them, projection profile analysis is a widely used algorithm for detecting text lines in Arabic script document images [22]. It works well for clean document images with large interline spacing, but fails for document images which contain noise, multi-column formats, and small interline spacing. The $x-y$ cut [29] is a state-of-the-art page segmentation approach that is based on project profile analysis. It can handle multi-column documents with small interline spacing. However, it fails for skewed document images and images with large amounts of noise. We presented a ridge-based text line detection approach for

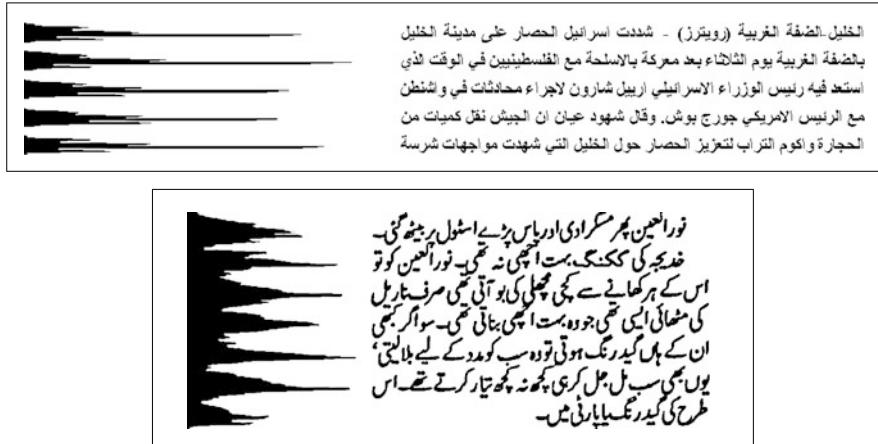


Fig. 2.6 Horizontal projection of Arabic scripts. The *top* figure shows the case of larger, well-defined interline spacing in Naskh script; there are between-line zero valleys in the projection profile. The *bottom* figure shows the case of small interline spacing in Nastaliq script (Urdu); there are no between-line zero valleys in the projection profile

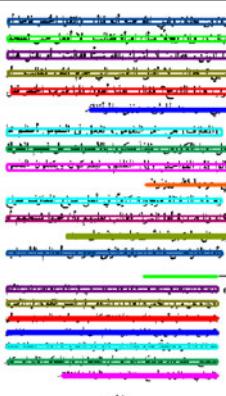
warped camera-captured document images [7, 8]. The ridge-based text line finding method is robust to the presence of noise, skew, and small interline spacing. It can also be used equally for text line detection in different types of Arabic script document images. The $x-y$ cut and ridge-based text line detection methods are described in more detail below.

2.3.1 $x-y$ Cut Text Line Detection Method

The $x-y$ cut page segmentation method [29] is a tree-based algorithm. An input document image is considered as a rectangular block. The $x-y$ cut algorithm recursively cuts a block into smaller blocks, until no block can be cut further. For splitting a block, first its horizontal and vertical projection profiles are computed. The noise removal thresholds t_n^x and t_n^y are then used for computing valleys in the projection profiles. The bins of horizontal and vertical projection profiles are set to zero if they contain values less than linearly scaled threshold t_n^x and t_n^y , respectively, with respect to the width and height of the block. The valleys of the horizontal (v_x) and vertical (v_y) projection profiles are compared with the predefined thresholds t_x and t_y , respectively. The block is split into two blocks at the midpoint of the wider of v_x and v_y , which are larger than t_x and t_y , respectively.

Horizontal projection profiles of sample paragraphs of Naskh and Nastaliq scripts are shown in Fig. 2.6. There are clear zero valleys in the projection profile of the Naskh script corresponding to interline gaps between text lines. In contrast, there is no zero valley in the projection profile of the Nastaliq script. The $x-y$ cut method

(a) Naskh Script



(b) X-Y cuts

(c) Labeled Text-lines



(d) Nastaliq Script



(e) X-Y cuts



(f) Labeled Text-lines

Fig. 2.7 The $x-y$ cut algorithm produces correct text line segmentation results for sample Arabic script document images

can be used to segment Nastaliq script documents. In such cases, the noise thresholds are set to a high value for finding the main body of text lines. Afterwards, the remaining portions of text lines are assigned to them through a simple post-processing step. Sample document images of Nastaliq script and their correctly segmented text lines are shown in Fig. 2.7. The following values of thresholds are used for generating these results: $t_n^x = 100$, $t_n^y = 100$, $t_x = 100$, and $t_y = 10$.

The x - y cut algorithm usually fails on documents with a large amount of border noises and reports the whole page as one segment. It also produces wrong text line segmentations for skewed document images. Failed cases of the x - y cut text line segmentation method are shown in Fig. 2.8. The ridge-based text line finding algorithm [7, 8] described next can be used in such cases.

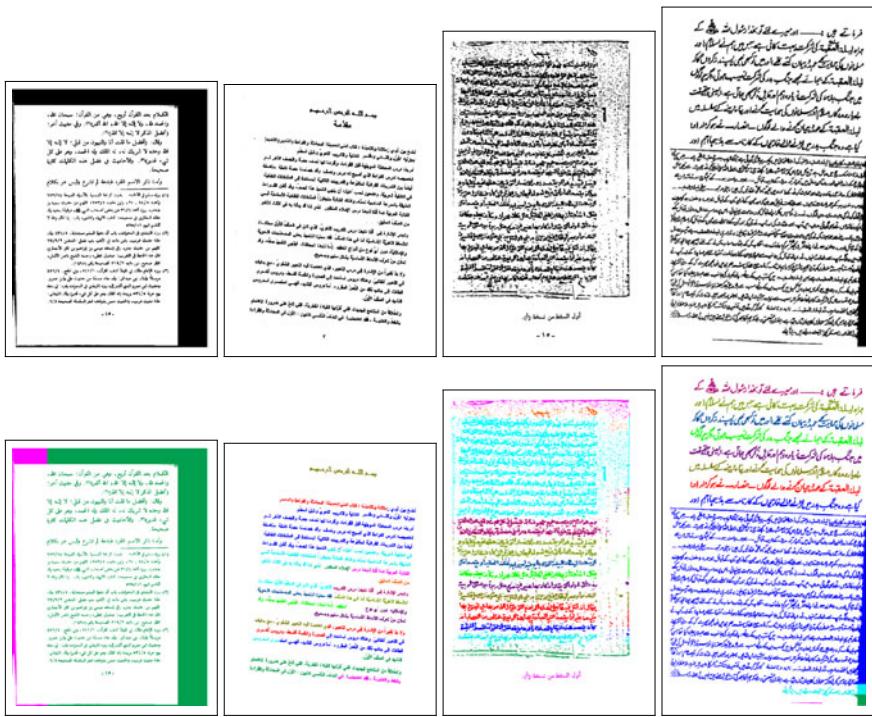


Fig. 2.8 The $x-y$ cut method produces text line segmentation failures for sample document images which contain challenging conditions, like border noise, skew, and a large number of joined characters

2.3.2 Ridge-Based Text Line Detection Method

We introduced a ridge-based text line finding algorithm [7, 8] for warped, Latin script camera-captured document images. The ridge-based text line finding method can be equally applied on different types of document images with respect to digitization methods (scanned or camera-captured), intensity values (binary or grayscale), scripting languages (like Latin, Chinese, Arabic, etc.), and writing styles (typed-text or handwritten). The method consists of two standard image processing techniques: (i) oriented anisotropic Gaussian filter bank smoothing and (ii) ridge detection. A detailed description of the ridge-based text line detection algorithm is presented next.

Step 1: Anisotropic Gaussian Filter Bank Smoothing Here, Gaussian filter bank smoothing is employed for enhancing text line structure in document images, such that it fills intraline gaps and maintains interline spaces. The general formula for an oriented, anisotropic Gaussian filter is shown in Eq. (2.1). It contains three well-defined parameters: σ_x : x -axis standard deviation, σ_y : y -axis standard deviation, and θ : angle of orientation.

$$g(x, y; \sigma_x, \sigma_y, \theta) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left\{-\frac{1}{2}\left(\frac{(x \cos \theta + y \sin \theta)^2}{\sigma_x^2} + \frac{(-x \sin \theta + y \cos \theta)^2}{\sigma_y^2}\right)\right\} \quad (2.1)$$

Input: Image I

Output: Smoothed Image I_S

Set $I_S := 0$;

foreach pixel location x, y **do**

for $\sigma_x := w_{\text{start}}$ **to** w_{end} **do**
for $\sigma_y := h_{\text{start}}$ **to** h_{end} **do**
for $\theta := \theta_{\text{start}}$ **to** θ_{end} **do**
 $val := g(x, y; \sigma_x, \sigma_y, \theta);$
/* The formula for $g(x, y; \sigma_x, \sigma_y, \theta)$ is given in
Equation 2.1 */
if $val > I_S(x, y)$ **then**
 $I_S(x, y) := val$
end

end
end
end

Algorithm 1: The text line structure enhancement algorithm using multioriented and multiscale anisotropic Gaussian filter bank smoothing

A diverse collection of document images is composed of a wide variety of font sizes, text line orientations, and interline and intraline spaces. Thus, a single isotropic ($\sigma_x = \sigma_y$) or anisotropic ($\sigma_x \neq \sigma_y$) Gaussian filter may either fill interline gaps (for a high value of standard deviation) or leave intraline gaps unfilled (for a small value of standard deviation). In contrast, a set of Gaussian filters, with varying values of standard deviations and orientations, is applied to a document image, and a maximum response is selected for each pixel for the corresponding smoothed image. For generating a filter bank, first the range of values is defined for σ_x , σ_y , and θ . These ranges can either be selected empirically or automatically by analyzing the statistics of connected components in document images. Then, a set of Gaussian filters is generated; each filter is composed of a different combination of values for σ_x , σ_y , and θ . The text line structure enhancement algorithm is shown in Algorithm 1. In order to speed up this algorithm, we used the fast anisotropic Gaussian filter implementation [15, 25]. A sample document image and its corresponding smoothed text line image are shown in Fig. 2.9(a) and 2.9(b), respectively. Now, text lines can be extracted from the smoothed image using the *ridge detection* method, which is described in the second step.

Step 2: Ridge Detection The ridge detection approach is used for representing the shape of objects in digital and speech signal processing. Here, we use the ridge detection approach for finding the main body of text lines in the smoothed document images. Researchers have introduced and analyzed different approaches for



Fig. 2.9 Steps of the ridge-based text line finding algorithm. (a) An example image of Arabic Nastaliq script. (b) Smoothed text line (text lines enhanced) image, which is generated by using oriented anisotropic Gaussian smoothing filter bank approach. (c) Detected ridges from smoothed image using Riley-based ridge detection [33, 34] algorithm. There are over-segmentation errors because of multi-column format. (d) Column separators that were detected through whitespace analysis; these separators help in correcting over-segmentation errors. (e) Processed ridges using whitespace separators; each ridge covers a complete region of a particular text line. (f) *Color-coded labeled* text lines result using detected ridges

ridge detection [13, 14, 26, 34]. Riley [33, 34] introduced the concept of a differential geometry-based ridge detection approach for detecting spectral peaks in speech signals. We use a ridge detection approach that has been derived from Riley's work. An open source version of the ridge detection method is made available as part of the OCROpus OCR system [18]. The Riley-based ridge detection method is described here in detail.

The ridge detection method finds ridge points in an input signal by analyzing its gradient vectors and the greatest downward curvatures. Let us consider a 2D image

I in a Cartesian coordinate system. For a point (x, y) , the content of I is represented by $I(x, y)$, and the corresponding gradient vector and the greatest downward curvature are represented by $\nabla I(x, y)$ and $C(x, y)$, respectively. We chose symbol C to represent the greatest downward curvature. The gradient vector $(\nabla I(x, y))$ is defined as:

$$\nabla I(x, y) = \left(\frac{\partial I(x, y)}{\partial x}, \frac{\partial I(x, y)}{\partial y} \right) \quad (2.2)$$

and the greatest downward curvature ($C(x, y)$) is defined as:

$$C(x, y) = \frac{\mathbf{e}_s(x, y)}{|\mathbf{e}_s(x, y)|}, \quad (2.3)$$

where $\mathbf{e}_s(x, y)$ is the eigenvector of the small eigenvalue of the Hessian matrix at point (x, y) ; $\lambda_s(x, y)$ represents the small eigenvalue and $\lambda_l(x, y)$ represents the large eigenvalue. Each point in image I is analyzed by Eq. (2.4) for checking whether it is a ridge point or not.

$$R(x, y) = \begin{cases} 1 & \text{if } \begin{cases} 1. \lambda_s(x, y) < 0 \text{ and} \\ 2. (\nabla I(x, y) \cdot C(x, y)) = 0 \end{cases} \\ 0 & \text{else} \end{cases} \quad (2.4)$$

The condition in Eq. (2.4) is sufficient for finding ridges in continuous signals. In order to make it applicable for discrete signals, more tests are needed for analyzing whether or not a point is a ridge's point. The ridge detection method that we have used here is derived from Riley's work of ridge detection for discrete signals. In this case, each point is analyzed for a ridge's point with each of its neighboring points based on the rules mentioned in Eq. (2.5), and is considered a ridge point if the output of Eq. (2.6) is 1. A complete description of the ridge detection method is shown in Algorithm 2.

$$R(x, y, dx, dy) = \begin{cases} 1 & \text{if } \begin{cases} 1. \lambda_s(x, y) < 0 \text{ and } |\lambda_s(x, y)| > |\lambda_l(x, y)| \text{ and} \\ 2. \lambda_s(x + dx, y + dy) < 0 \text{ and} \\ |\lambda_s(x + dx, y + dy)| > |\lambda_l(x + dx, y + dy)| \text{ and} \\ 3. \nabla I(x, y) \cdot \nabla I(x + dx, y + dy) < C(x, y) \\ \cdot C(x + dx, y + dy) \text{ and} \\ 4. (\nabla I(x, y) \cdot C(x, y))(\nabla I(x + dx, y + dy) \\ \cdot C(x + dx, y + dy)) \\ (C(x, y) \cdot C(x + dx, y + dy)) < 0 \end{cases} \\ 0 & \text{else} \end{cases} \quad (2.5)$$

$$R(x, y) = \max_{(dx,dy) \in \{(0,1), (1,0), (0,-1), (-1,0)\}} R(x, y, dx, dy) \quad (2.6)$$

The ridge detection output for the smoothed image of Fig. 2.9(b) is shown in Fig. 2.9(c). A ridge covers a complete region of a particular text line for single-column document images. For multi-column documents, the filter bank may fill small gaps between text lines between different columns. Therefore, a single ridge may cover either a single text line or multiple text lines at the same height (Fig. 2.9(c)). This situation causes over-segmentation errors. This type of over-segmentation error can be corrected by a *whitespace analysis*, as described below.

```

Input: The Smoothed Image  $I$ 
Output: Detected Ridges Image  $I_R$ 
Calculate gradient vectors image  $\nabla I$ ;
Calculate greatest downward curvature image  $C : \mathbf{e}_s, \lambda_s, \mathbf{e}_l, \lambda_l$ ;
Set  $I_R := 0$ ;
foreach pixel location  $x, y$  do
  if  $R(x, y, 0, 1)$  or  $R(x, y, 1, 0)$  or  $R(x, y, 0, -1)$  or  $R(x, y, -1, 0)$ 
    then /* The formula for  $R(x, y, dx, dy)$  is given in Eq. (2.5) */
      |  $I_R(x, y) = 1$ 
    else
      |  $I_R(x, y) = 0$ 
    end
end

```

Algorithm 2: The Riley [33, 34]-based ridge detection algorithm

Step 3: Whitespace Analysis Whitespace analysis aims to find a set of maximal whitespace rectangles in a document image such that the union of these rectangles completely covers the document's background. It is usually used for page segmentation [2] or multi-column separation [5]. An algorithm for finding maximal whitespace rectangles is presented in [5]. The main idea behind that algorithm is similar to the quick-sort or branch-and-bound methods. The whitespace rectangles are evaluated as candidates for column separators or gutters based on their statistics, like aspect ratio, width, etc. Whitespace cuts that correspond to column separators are shown in Fig. 2.9(d). Now, over-segmentation errors (as shown in Fig. 2.9(c)) can be corrected by cutting detected ridges at those points which lie over whitespace rectangles. The output ridges are shown in Fig. 2.9(e), where a single ridge covers a single text line. These ridges are considered as detected text lines.

Step 4: Text Line Labeling A text line labeling method assigns a unique label to all of the connected components of a text line. As shown in Fig. 2.9(e), each detected ridge represents the main region of a particular text line. Let us consider a simple case where each connected component in a document image overlaps with a single ridge. First, each ridge is assigned a unique label. Then, each connected component is assigned the label of its corresponding ridge. Each unlabeled connected component in the proximity of text lines is assigned the label of its nearest text line. It is also possible that a connected component overlaps with more than one ridge, which usually happens in the case of inter-line touching or overlapping. In such a case, a connected component is cut in the center of each pair of consecutive ridges, and then each portion is assigned the label of the particular ridge. The result of text line labeling in color-coded form is shown in Fig. 2.9(f).

For some of the challenging problems like document skew and noise, the text line detection results of the ridge-based text line extraction method are shown in Fig. 2.10. As can be seen in the figure, the ridge-based text line detection method is robust to document skew, small interline gaps, border noise, and interline touching and/or overlapping as compared to the $x-y$ cut method (whose results are shown in Fig. 2.8).



Fig. 2.10 The ridge-based text line finding method produces correct text line segmentation results for sample document images with border noise, skew, and a large number of joined characters

2.4 Text Line Reading Order Determination

A reading order determination method tries to find the order of text lines with respect to their corresponding reading flow. The reading order can be determined by applying some ordering criteria over the positioning of the text lines. In contrast to Latin script, the reading order of Arabic script is from right to left. A reading order determination method is presented in [6] for Latin script, which is modified for Nastaliq Arabic script in [36]. The ordering criteria that were presented in [36] are stated as follows:

- “Text-Line ‘a’ comes before text-line ‘b’ if their ranges of x -coordinates overlap and if text-line ‘a’ is above text-line ‘b’ on the page”.
- “Text-Line ‘a’ comes before text-line ‘b’ if a is entirely to the right of ‘b’ and if there does not exist a text-line ‘c’ whose y -coordinates are between ‘a’ and ‘b’ and whose range of x -coordinates overlaps both ‘a’ and ‘b’ ”.

The reading order determination method [6, 36] finds the partial ordering of text lines through the above-defined ordering criteria, and then finds a complete order using a topological sorting algorithm [12].



Fig. 2.11 Example images illustrating results of reading order for a newspaper and a book page. *Thin horizontal lines* with different colors indicate detected text line segments, and the *magenta lines* running down and diagonally across the image indicate reading order

Examples of reading order determination on sample document images are shown in Fig. 2.11. The performance of the reading order determination method decreases with a decrease in text line detection accuracy and/or the presence of noise in document images.

2.5 Discussion

In this chapter, we have presented a generic layout analysis system which can be used for a variety of typed-text (like Naskh and Nastaliq), handwritten, and ancient Arabic script document images. Our layout analysis system first performs text and non-text segmentation, then text line detection, and finally reading order determination. The output of our layout analysis system in conjunction with an efficient OCR engine can be used for the digitization of a wide variety of typed-text Arabic script document images.

We have demonstrated that the improved version of the multiresolution morphology-based text and non-text segmentation approach is suitable for Arabic script

document images. It gives correct text and non-text segmentation results, unless a document image contains very large text element(s) and/or very small non-text element(s).

The projection profile-based x - y cut method gives correct text line segmentation results for clean document images. It produces bad segmentation results for document images containing large amounts of noise and/or document skew. We have proposed a robust text line detection approach using standard image processing methods (filter bank Gaussian smoothing and ridge detection) for Arabic script document images. Our ridge-based text line finding approach is robust to large amounts of border noises, handwritten marks, and document skew and curl and is equally applicable on both binary and grayscale document images. It has well-understood free parameters, such as ranges of orientation angles, and x - and y -axis standard deviations for oriented anisotropic Gaussian filter bank smoothing. These parameter values can be easily tuned for a variety of document images with respect to the size statistics of connected components.

The performance of the reading order determination algorithm heavily depends on the text line detection accuracy. Manual inspection of the results showed the following types of errors: (1) if two text lines from different text columns are merged, they are interpreted as a separator, and hence the algorithm gives a wrong reading order, and (2) in some cases, the separation between different sections of a multi-column document is not represented by a text line spanning the columns, but instead a ruling (thick horizontal black line) is used. In that case the algorithm fails to detect the start of a new section.

In general, our generic layout analysis system is quite robust to different types of challenging problems in complex document image layouts, like books, newspapers, and ancient Arabic script document images. To the best of our knowledge, there is no public dataset available for the evaluation of different layout analysis systems for Arabic document images. As a future goal, there is a need for a public dataset with a large variety of Arabic script documents with text line level ground truth for benchmarking existing layout analysis systems for Arabic scripts.

References

1. Abed, H.E., Märgner, V.: ICDAR 2009—Arabic handwriting recognition competition. *Int. J. Doc. Anal. Recognit.* **14**, 3–13 (2011)
2. Baird, H.S.: Background structure in document images. In: Bunke, H., Wang, P., Baird, H.S. (eds.) *Document Image Analysis*, pp. 17–34. World Scientific, Singapore (1994)
3. Bloomberg, D.S.: Multiresolution morphological approach to document image analysis. In: *Proceedings 1st International Conference on Document Analysis and Recognition*, St. Malo, France, pp. 963–971 (1991)
4. Boussellaa, W., Zahour, A., Abed, H.E., Benabdellahid, A., Alimi, A.M.: Unsupervised block covering analysis for text-line segmentation of Arabic ancient handwritten document images. In: *Proceedings 20th International Conference on Pattern Recognition*, Istanbul, Turkey, pp. 1929–1932 (2010)
5. Breuel, T.M.: Two geometric algorithms for layout analysis. In: *Proceedings Document Analysis Systems*, Princeton, NY, USA, Aug. 2002, pp. 188–199 (2002)

6. Breuel, T.M.: High performance document layout analysis. In: Symposium on Document Image Understanding Technology, Greenbelt, MD, USA, April 2003
7. Bukhari, S.S., Shafait, F., Breuel, T.M.: Ridges based curled textline region detection from grayscale camera-captured document images. In: Computer Analysis of Images and Patterns. Lecture Notes in Computer Science, vol. 5702, pp. 173–180 (2009)
8. Bukhari, S.S., Shafait, F., Breuel, T.M.: Script-independent handwritten textlines segmentation using active contours. In: Proceedings 10th International Conference on Document Analysis and Recognition, Barcelona, Spain, pp. 446–450 (2009)
9. Bukhari, S.S., Shafait, F., Breuel, T.M.: Document image segmentation using discriminative learning over connected components. In: Proceedings 9th IAPR Workshop on Document Analysis Systems, Boston, Massachusetts, USA, pp. 183–190 (2010)
10. Bukhari, S.S., Shafait, F., Breuel, T.M.: Improved document image segmentation algorithm using multiresolution morphology. In: Proceedings SPIE Document Recognition and Retrieval XVIII, San Jose, CA, USA, Jan. 2011
11. Cattoni, R., Coianiz, T., Messelodi, S., Modena, C.M.: Geometric layout analysis techniques for document image understanding: a review. Technical Report 9703-09, IRST, Trento, Italy (1998)
12. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms, pp. 485–488. MIT Press, Cambridge (1990). Chapter 23
13. Damon, J.: Properties of ridges and cores for two-dimensional images. *J. Math. Imaging Vis.* **10**, 163–174 (1999)
14. Eberly, D., Gardner, R., Morse, B., Pizer, S., Scharlach, C.: Ridges for image analysis. *J. Math. Imaging Vis.* **4**, 353–373 (1994)
15. Geusebroek, J.M., Smeulders, A.W.M., Weijer, J.V.D.: Fast anisotropic Gauss filtering. *IEEE Trans. Image Process.* **12**, 2003 (2003)
16. http://en.wikipedia.org/wiki/Nastaliq_script
17. <http://www.ijma3.org/Templates/InsideTemplate.aspx?PostingId=427>
18. <http://code.google.com/p/ocropus/>
19. Husain, S.A., Amin, S.H.: A multi-tier holistic approach for Urdu Nastaliq recognition. In: IEEE International Multi-topic Conference, Karachi, Pakistan, Dec. 2002
20. Jelodar, M.S., Fadaieieslam, M.J., Mozayani, N., Fazeli, M.: A Persian OCR system using morphological operators. *World Acad. Sci., Eng. Technol.* **4**, 141 (2007)
21. Keysers, D., Shafait, F., Breuel, T.M.: Document image zone classification—a simple high-performance approach. In: 2nd International Conference on Computer Vision Theory and Applications, Barcelona, Spain, Mar. 2007, pp. 44–51 (2007)
22. Khorsheed, M.S.: Off-line Arabic character recognition—a review. *Pattern Anal. Appl.* **5**(1), 31–45 (2002)
23. Kise, K., Sato, A., Iwata, M.: Segmentation of page images using the area Voronoi diagram. *Comput. Vis. Image Underst.* **70**(3), 370–382 (1998)
24. Kumar, K.S., Kumar, S., Jawahar, C.: On segmentation of documents in complex scripts. In: 9th International Conference on Document Analysis and Recognition, Curitiba, Brazil, Sep. 2007, pp. 1243–1247 (2007)
25. Lampert, C.H., Wirjadi, O.: An optimal nonorthogonal separation of the anisotropic Gaussian convolution filter. *IEEE Trans. Image Process.* **15**(11), 3501–3513 (2006)
26. Lindeberg, T.: Edge detection and ridge detection with automatic scale selection. *Int. J. Comput. Vis.* **30**, 465–470 (1996)
27. Moll, M.A., Baird, H.S., An, C.: Truthing for pixel-accurate segmentation. In: Proceedings 8th IAPR International Workshop on Document Analysis Systems, Sep. 2008, pp. 379–385 (2008)
28. Nagy, G.: Twenty years of document image analysis in PAMI. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(1), 38–62 (2000)
29. Nagy, G., Seth, S., Viswanathan, M.: A prototype document image analysis system for technical journals. *Computer* **7**(25), 10–22 (1992)

30. O’Gorman, L.: The document spectrum for page layout analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **15**(11), 1162–1173 (1993)
31. Otsu, N.: A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.* **9**(1), 62–66 (1979)
32. Pal, U., Sarkar, A.: Recognition of printed Urdu script. In: 7th International Conference on Document Analysis and Recognition, Edinburgh, UK, Aug. 2003, pp. 1183–1187 (2003)
33. Riley, M.D.: Beyond quasi-stationarity: Designing time-frequency representations for speech signals. In: Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, Apr. 1987, vol. 12, pp. 657–660 (1987)
34. Riley, M.D.: Time-frequency representation for speech signals. Ph.D. Thesis, MIT (1987)
35. Sauvola, J., Pietikainen, M.: Adaptive document image binarization. *Pattern Recognit.* **33**(2), 225–236 (2000)
36. Shafait, F., Hasan, A., Keysers, D., Breuel, T.M.: Layout analysis of Urdu document images. In: 10th IEEE International Multi-topic Conference, INMIC’06. Islamabad, Pakistan, Dec. 2006
37. Shafait, F., Keysers, D., Breuel, T.M.: Performance evaluation and benchmarking of six page segmentation algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(6) (2008)
38. Shirali-Shahreza, S., Manzuri-Shalmani, M.T., Shirali-Shahreza, M.H.: Page segmentation of Persian/Arabic printed text using ink spread effect. In: SICE-ICASE International Joint Conference, Busan, Korea, Oct. 2006, pp. 259–262 (2006)
39. Wang, Y., Haralick, R., Phillips, I.: Improvement of zone content classification by using background analysis. In: Proceedings Document Analysis Systems, Rio de Janeiro, Brazil, Dec. 2000
40. Wong, K.Y., Casey, R.G., Wahl, F.M.: Document analysis system. *IBM J. Res. Dev.* **26**(6), 647–656 (1982)
41. Zahour, A., Taconet, B., Mercy, P., Ramdane, S.: Arabic hand-written text-line extraction. In: Proceedings 6th International Conference on Document Analysis and Recognition, Seattle, WA, USA, Sep. 2001, pp. 281–285 (2001)

Chapter 3

A Multi-stage Approach to Arabic Document Analysis

Eugene Borovikov and Ilya Zavorin

Abstract We approach the analysis of electronic documents as a multi-stage process, which we implement via a *multi-filter document processing framework* that provides (a) flexibility for research prototyping, (b) efficiency for development, and (c) reliability for deployment. In the context of this framework, we present our multi-stage solutions to multi-engine Arabic OCR (MEMOE) and Arabic handwriting recognition (AHWR). We also describe our adaptive pre-OCR document image cleanup system called ImageRefiner. Experimental results are reported for all mentioned systems.

3.1 Introduction

The analysis of electronic documents is usually a multi-stage process that typically includes document image cleanup, content detection and segmentation, recognition, and correction, possibly followed by named entity recognition (NER), automatic summarization, and machine translation (MT). In this kind of system, every processing stage naturally relies on the output of prior stages accompanied by confidence measures, which could come directly from an algorithm that implements a particular processing stage or from a confidence computation based on additional types of evidence.

Each individual processing stage could potentially incorporate several substages being run sequentially or in parallel with others. For instance, given multiple optical character recognition (OCR) engines, a combined OCR stage can execute those engines in parallel on a single document with individual results combined into a single output text stream. Alternatively, a recognition stage can be implemented via a sequence of heterogeneous classifiers working at different levels of data granularity

E. Borovikov (✉) · I. Zavorin

Research and Development, CACI International Inc, 4831 Walden Lane, Lanham, MD 20706,
USA

e-mail: yborovikov@caci.com

I. Zavorin

e-mail: izavorin@caci.com

(e.g., characters versus words) with each classifier providing disambiguation of output of a preceding classifier. Furthermore, an advanced document analysis system may include a feedback loop that would allow it to learn something about an earlier processing step by examining reliable evidence obtained at a later stage.

Here, we discuss the major stages of Arabic OCR and handwriting recognition (HWR), address their goals and challenges, review related approaches, and present a multi-stage solution that we have developed. We discuss the algorithms we developed and utilized for Arabic OCR and HWR in Sect. 3.2, including pre/post-OCR processing as well as the recognition stages. In Sect. 3.3, we discuss in detail our *multi-filter* framework, which was utilized and extended during the implementation of two specific systems for Arabic document analysis: *Arabic Handwritten Word Recognizer* described in Sect. 3.2.2 and *Multi-evidence Multi-OCR Engine* described in Sect. 3.2.3.

We conclude with a discussion of the results we were able to obtain with our multi-stage solution, and we also identify some interesting R&D directions that could be taken with the multi-stage approach as well as other modules that could be integrated into the multi-filter document processing framework.

3.2 Arabic Document Processing Algorithms

In this section, we describe our Arabic document processing efforts related to pre-OCR processing (adaptive document image cleanup), optical character and word recognition (focusing on handwriting), and post-OCR accuracy boosting using single and multiple OCR engines. We also discuss possible extensions, e.g., combining pre-processing, recognition, and post-processing into one system using the framework.

3.2.1 Pre-OCR Processing

Modern OCR engines are expected to handle document imagery that can have wide variations in noise level, page layout, image quality, and pixel depth. While most such engines perform some type of pre-recognition image enhancement on their own, this enhancement is usually generic in nature and thus may not take into account specific types of artifacts that given OCR engine users often encounter in the data that they are processing.

Many image enhancement methods that correct particular types of noise (such as despeckling, filling broken lines, etc.) can actually degrade images without these noise sources. Our experience suggests that many individual methods are likely to harm OCR results at least as often as they improve them. Thus, selecting the right image improvement method or methods to apply constitutes an important step in using OCR to make the text of noisy document images available for searching or further processing.

In operational settings, the usual options are (a) to select some fixed set of image transformations that are deemed likely to improve the bulk of the documents in a large batch or (b) to assign a human operator to select appropriate techniques to apply to each scanned image. Neither choice is fully satisfactory. The fixed choice is subject to the problem described above, and individual decisions by a human require time and effort—and the result that looks best to a human may not reflect the best choice for OCR results.

When the task is to process a large volume of document images, it is necessary to automate the pre-OCR cleanup stage by employing an intelligent image enhancement system to determine and apply an optimal image transformation individually for each document image in the OCR queue. Evidently, such a system needs to be optimized for a given OCR engine (while treating the engine as a black box that ingests a document image and outputs recognized text) and the varieties of expected image noise in a typical document set.

In what follows, we discuss the problem of pre-OCR document image cleanup and enhancement, and describe a system that solves this problem adaptively with respect to the given OCR engine(s) and the document corpus [32, 34, 37]. The system is called *ImageRefiner*, and it is based on machine learning. Using an artificial neural net (or other multi-value classifiers), the system learns which image enhancement transformations are best suited for a given document image type with respect to the given OCR engine. The input feature set is based on various image measurements (mostly noise characteristics). The method has been successfully applied to both bitonal and grayscale Arabic document images, resulting in improved OCR accuracy.

ImageRefiner

We have developed an approach to automate image pre-processing that is based on machine learning. This approach has been implemented in a system called *ImageRefiner*. Our work was originally inspired by the Quality Assessment, Restoration, and OCR (QUARC) system [5] developed at Los Alamos National Labs and related work [23]. Like any system based on machine learning, *ImageRefiner* operates in two modes: training and application (image refining, in our case).

During *training* (given an OCR engine and a set of document images together with the corresponding ground truth text), *ImageRefiner* applies a set of transformations to each input image and passes the resulting transformed images to the OCR engine. The resulting OCR output text is compared against the ground truth, the OCR accuracy is computed, and the transformation which yields the highest accuracy is chosen as the best transformation for this kind of image. The schema of the training process is shown in Fig. 3.1.

For each original image, a feature vector is computed via a set of scalar measurements on this image. Thus, each training image yields a single feature vector and a single (integer) class label that represents its best transformation. This data is then used by *ImageRefiner* to train one of its classifiers.

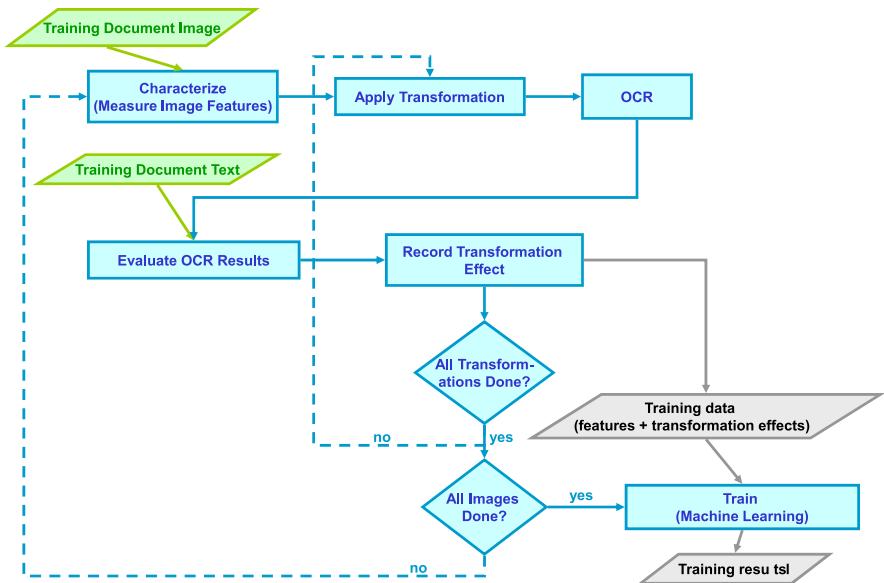


Fig. 3.1 Training mode of ImageRefiner

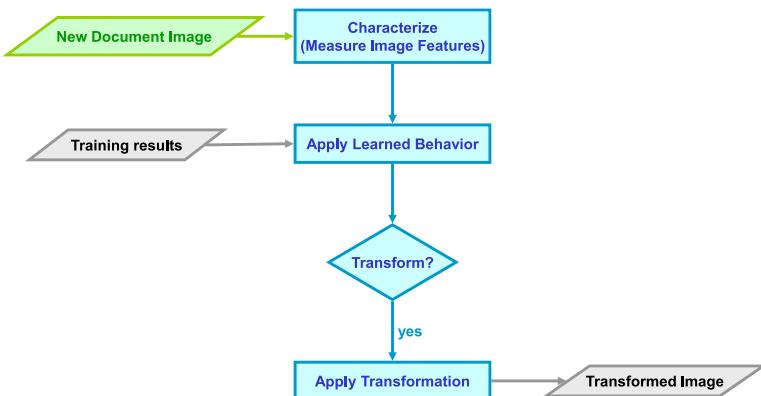


Fig. 3.2 Refining modes of ImageRefiner

During *refining* (given a pretrained classifier and a set of images that need to be cleaned up), ImageRefiner computes a feature vector for each image which is then used by the classifier to recommend an image transformation for the image. The image may then be transformed and optionally passed to an OCR engine. See Fig. 3.2 for the schema of the refining phase.

Our initial effort was geared toward computing best transformations that would be applied to entire bitonal images [32]. While at that stage we only experimented with English documents, the resulting system was language independent and could

easily be trained on document collections in any other language, as long as an OCR engine for that language was available.

We later extended this work in two parallel directions. *First*, we stayed in the bitonal image domain but considered automatic segmentation of individual images followed by finding separate best enhancement transformations for different segments [34]. *Second*, we looked into processing of grayscale images without segmentation [37]. In both cases, we have applied the resulting extended systems to Arabic document images.

Application to Bitonal Images

In this effort, we used the full set of specific image *characteristics* in QUARC, e.g., small and large speckle factors, and touching and broken character factors. In addition, we implemented several more general characteristics based on various statistics such as mean and standard deviation for size, height, width, and other measures of connected components. For *machine learning*, we used the standard backpropagation neural network. For *image transformations*, we used nine QUARC transformations, e.g., a moderate despeckling filter, morphological opening and closing, and the kFill filter with different grid sizes.

Application to Segmented Bitonal Images

In the same way that different images may have different best transformation methods, it is possible that there may be different best image transformation methods for different regions in the same image. We developed an approach to this problem that combines a novel segmentation method and the neural network classification method. The system functions in essentially the same way during both training and refining as the original system described, except that the segmentation method is applied to an input image to split it into characteristically homogeneous segments and then each segment is treated as a separate image.

Segmentation consists of two recursive steps. The first step intentionally over-segments, and the second step merges similar adjacent segments to compensate for the over-segmentation.

In Step 1, we build a quadtree by recursive splitting. Initially, the image consists of a single region; we then recursively split each region into four smaller segments of equal size until either the smallest acceptable segment size is reached, or the region is characteristically homogeneous. To measure if a segment is homogeneous, several characteristics of the image segment are used, e.g., the size of black connected components. For all these characteristics, the sample standard deviations are computed. A segment is regarded as homogeneous in the process of splitting if each sample standard deviation of these four characteristics is less than its corresponding threshold. These thresholds are selected based on empirical testing.

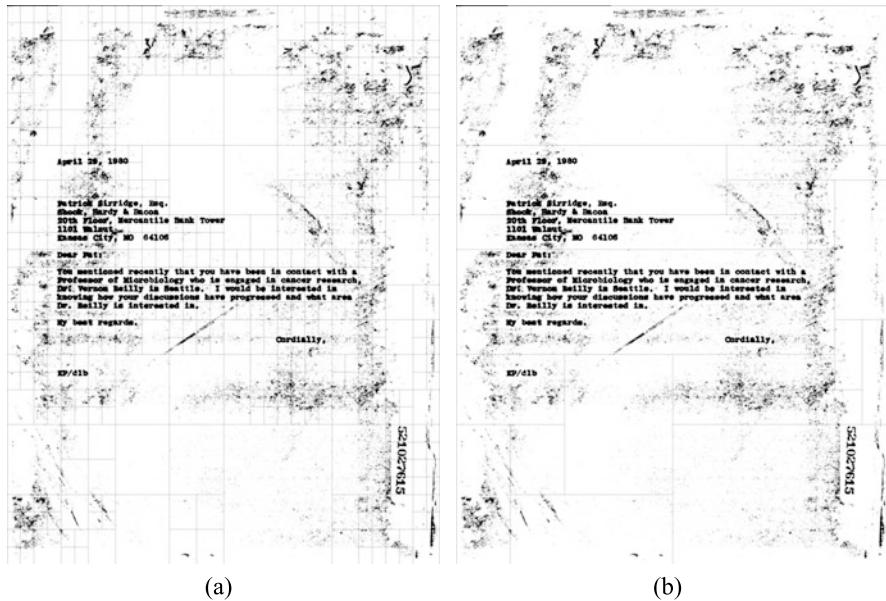


Fig. 3.3 Sample document image after Step 1 **(a)** and Step 2 **(b)** of the segmentation process

In Step 2, to reduce the number of homogeneous segments, we repeatedly merge two adjacent rectangular segments that can form a single homogeneous segment, until no further merges are possible. Two adjacent segments are considered homogeneous if any of the following three criteria is satisfied:

1. The difference of the sample means and the difference of sample standard deviations of the two segments are smaller than certain separate thresholds (this criterion works when the black connected components in these regions are mainly noise).
2. The total connected components in both segments are less than certain numbers (this criterion works when these regions are small and black connected components in these regions are mainly characters)
3. The means of the size of black connected components in the two segments are considered the same by a hypothesis test, and the means of the width of the black connected components in the two segments are considered the same by a similar hypothesis test (this criterion works when the black connected components are mainly characters).

Normally, Step 1 produces a large number of image segments. Therefore, there are many different possible sequences in which to consider merging the adjacent segments during Step 2. Since this effort was focused on English and Arabic, in which characters form horizontal lines, the algorithm looks for as many horizontal merges as possible first, then as many vertical merges as possible; it then repeats the process until no more merging can occur. This approach usually generated few



Fig. 3.4 Sample segmented Arabic documents

homogeneous segments in the final stage of the segmentation, which was helpful in reducing the amount of downstream processing. Figures 3.3(a) and 3.3(b) show a sample document image after, respectively, Step 1 and Step 2 of the segmentation process.

We trained the system on a small collection of about 100 Arabic documents using a commercial Arabic OCR engine, and then applied it to another collection of test documents. The resulting accuracy for segmented and enhanced documents was on average 35 % higher than that for the original documents. Figure 3.4 shows some of the segmented Arabic documents.

Application to Grayscale Images

In this effort, we focused on Arabic grayscale documents. We implemented two different types of image *transformations*: denoising and thresholding. The grayscale image characteristics are all derived from the image histogram of an original grayscale image, and are all byproducts of various thresholding methods. When determining which transformations to include, we used the following criteria:

- The final selection should be representative of the various classes of transformations that have been developed to process document grayscale images.

- The transformation should be well known and based on a solid mathematical foundation.
- Although there are numerous transformations that can potentially be included [30], since a limited amount of training data was available to us, the final selection had to be small.

In addition to the identity transformation, which does not alter the original image at all, we have chosen the following transformations: Otsu thresholding, which is a classical global thresholding algorithm [27]; Niblack thresholding, which is a local thresholding algorithm [25]; and median filtering with a 3×3 window, which is a classical denoising transformation. Furthermore, if during training, none of the above filters yields an OCR accuracy above a certain predetermined threshold, the image is assigned the *rejection* class. Thus, a grayscale image classifier was trained with training data assigned to five different classes.

There are various ways in which image *characteristics* may be selected. Since they are to be used to determine the most appropriate transformation for a particular image, one approach is to work backwards, i.e., to choose those characteristics that are somehow relevant to the types of transformations that will be applied. An important requirement when choosing a characteristic is that it has to be cheap to compute; otherwise, it would render ImageRefiner inefficient.

We have chosen five measurements that are computed by various thresholding methods [28] when determining the best gray level that separates foreground from background:

- *Minimum entropy fuzziness* is the smallest value of a fuzziness measure that is based on Shannon's entropy function and that is computed over a fuzzy set of foreground pixels.
- *Minimum Yager fuzziness* is the smallest value of an alternative fuzziness measure developed by Yager.
- *Johansen minimum F* is the smallest value of a measure of interdependence between gray levels separated into foreground and background.
- *Otsu maximum η* is the largest value of a measure of between-class scatter of the foreground and background pixels for the given grayscale image. The threshold at which this largest value is attained is selected as the optimal threshold for the image by the Otsu thresholding method.
- *Pun maximum F* is the largest value of a measure of entropy of thresholded black and white pixels of the given grayscale image.

We also experimented with several types of *machine learning* algorithms that could be separated into two types: neural network based and all-pairs discriminants [6].

In our experiments, we used two performance measures: the *matching ratio* and the *cumulative accuracy improvement*. The former metric is the fraction of the test images for which trained ImageRefiner correctly selected the best transformation. The rationale behind the latter measure was that the ultimate goal of ImageRefiner is not just to pick the best transformation, but to improve the OCR accuracy as much

as possible. Even if ImageRefiner picks a second-best transformation, the resulting accuracy improvement might still be substantial. Therefore, for a given set of test document images, and a given set of transformations applied to these respective images, we compute the cumulative change in the OCR accuracy over all the transformed images in the set compared to the OCR accuracy applied to the original images. This value can actually be negative in the case when transformations computed by the system for many documents result in worse OCR performance.

We used two different corpora of Arabic documents, one of which was the Arabic News corpus [35], and performed cross-validation tests. These showed that, if measured in terms of the matching ratio, the overall performance of the different ML methods was comparable. However, when we took into account OCR accuracy change, the single neural network that was originally implemented in the system outperformed all other methods.

We also noticed that performance differed significantly between the two datasets. For one of them, the image transformations that we evaluated were fairly adequate at improving OCR quality, which was not the case for the other set. This implies that the set of available image cleaning transformations has to be greatly extended to account for the diversity that exists in real-world document imagery.

We also observed that there was quite a bit of feature correlation in both datasets. This should probably be expected to hold in general since all the characteristics are auxiliary quantities that are computed by different algorithms that belong to the same family of global thresholding methods. Therefore, more work can be done to extend the set of available characteristics.

3.2.2 Arabic Script Recognition

Research and development in the area of Arabic OCR resulted in a number of high quality commercial recognition engines for printed Arabic text, any of which can be embraced by our framework. These tools produce high quality output in general, although the general problem of Arabic document recognition is not quite solved yet. Among the challenges (partially solved in some systems) that still remain are: multi-column reading order, multi-language/script recognition, layout retention, very noisy documents, and non-printed-text artifacts (stamps, handwriting). Here we address various research aspects of converting Arabic document images to text with an emphasis on handwriting recognition.

Printed Arabic Text Recognition

Arabic OCR has its specific challenges. To begin with, Arabic text is written, typed, and printed cursively in blocks of interconnected characters. A word may consist of several character blocks. Arabic characters in addition to their *isolated* form can take different shapes depending on their position inside the block of characters: *initial*, *medial*, or *final*, as shown for the letter “ain” in Fig. 3.5.

Fig. 3.5 Four shapes of the Arabic letter “ain” (right to left): isolated, initial, medial, and final

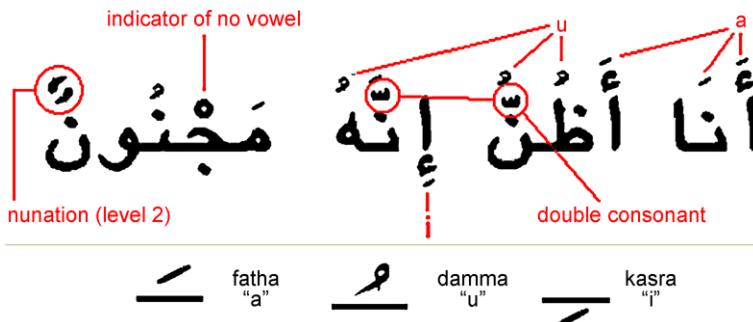


Fig. 3.6 Arabic diacritics

Arabic characters can also be written stacked one on top of another, which may lead to character blocks having more than one base line. Additionally, Arabic uses many types of external objects such as dots, *Hamza* and *Madha*. Optional diacritics (shown in Fig. 3.6, courtesy of lexicorient.com) also add to the set of external objects.

Finally, Arabic font suppliers do not always follow a common standard. Given the peculiarities of Arabic fonts and the characteristics of the Arabic language, building an omni-font Arabic OCR becomes a difficult undertaking.

Our efforts in the Arabic OCR area have focused on adapting existing OCR engines to handling (very) noisy documents, by building pre-OCR image cleaners (e.g., ImageRefiner described in Sect. 3.2.1) and post-OCR accuracy boosters (e.g., MEMOE described in Sect. 3.2.3).

Arabic Handwriting Recognition

General handwriting recognition is a challenging and interesting problem. The main difficulty here apparently stems from the large amount of variability that general handwriting may exhibit. Two different writers are bound to write the same glyph differently, which may be roughly analogous to font differences in printed text, but even the same hand writes two instances of the same glyph somewhat differently, and this introduces additional difficulty in general-purpose recognition.

Arabic script presents additional challenges for handwriting recognition systems due to its highly connected nature, numerous forms of each letter, presence of ligatures, and regional differences in writing styles and habits. Having been interested

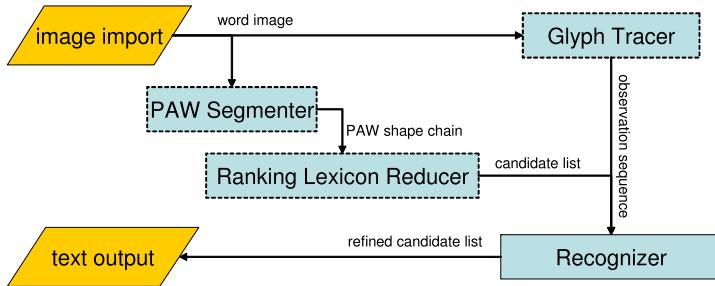


Fig. 3.7 A configuration of our AHWR system

in off-line Arabic handwriting recognition for several years, we researched and developed several handwriting recognition methods that may be applicable to Arabic handwriting recognition. Below we mention a few research papers that we perceive to be particularly relevant to our work.

AbdulKader [1] describes the ICRA system for off-line Arabic handwriting recognition. The system is a two-tier recognizer. The methodology exploits the fact that, because six Arabic letters do not connect to other letters from the left, all Arabic words consist of PAWs (Parts of Arabic Words). The number of unique PAWs grows sublinearly with the number of words. Thus, the first tier of the system consists of a PAW recognizer that processes connected components of a given word image using a neural network and a PAW-to-letter lexicon. The second tier uses a word-to-PAW lexicon to produce the final result using a variation of the best-first search algorithm called beam search. This system was a participant of the competition at ICDAR 2005 [22] and was reported to have one of the best performance levels when applied to the IFN/ENIT corpus [29].

Govindaraju [10] outlines the two currently accepted approaches to handwriting recognition: holistic and analytical. Holistic approaches attempt to identify whole words at once, while analytical methods try to build words from recognized characters. In order to unify the two, Govindaraju attempts to find a middle ground between the two for AHWR by proposing a recognition tool that utilizes PAWs lexicons.

Lorigo [20] summarizes the state of the AHWR field and the direction in which research appears to be trending. He also outlines attempts at the University of Buffalo to analyze ancient Arabic documents using existing recognition tools. After reviewing some basic image processing techniques related to AHWR, the author offers a broad look at the use of neural networks, hidden Markov models (HMMs), and combinations of the two in handwriting recognition. The author concludes by discussing a practical application of these tools in analyzing ancient documents.

Off-line Recognition An off-line handwriting recognition system typically consists of multiple stages of information processing: starting with image handling, and then continuing to feature extraction, pattern recognition, and text correction. The implementation of such stages may vary, but the overall processing pattern appears

to be common [21]. Here we present our *multi-filter* approach to the off-line AHWR problem, and describe a solution using our filter-based document processing framework described in Sect. 3.3. The system combines several heterogeneous classifiers, which we implemented as filters. Here we briefly describe the major ones, whose methodologies are discussed in detail in our prior publications [36, 39].

Glyph Tracer represents a word image as a sequence of discrete topological and geometric features (such as loops, x-cross, T-cross, end points, turns, etc.) and generates a sequence of the corresponding discrete observations.

*PAW Segmente*r splits input word/phrase into PAWs using horizontally connected components and encodes each PAW as a *PAW shape chain*, with each element corresponding to one of the 16 character shapes identified by *character shape detector*. For instance, the class ٩ corresponds to the characters ٩, ف, ق, and 9.

Ranking Lexicon Reducer evaluates each PAW shape chain against a list of candidate models, eliminates those models that do not match the chain, and ranks the remaining ones in the range [0, 1]. While it can function as a full recognizer, its primary purpose here is in reducing the number of candidate terms to consider and also providing initial scores that may be used as weights at later stages of recognition.

Recognizer is a stochastic sequence labeler that uses output of the Lexicon Reducer and/or the Glyph Tracer filters to make the final decision in the word, PAW, or character recognition process. We typically use an HMM-based classifier that evaluates an observation sequence against a collection of word models corresponding to a predetermined lexicon. As an alternative, we have also employed a conditional random field (CRF)-based recognizer designed to label PAW sequences.

We experimented with various combinations of the classifiers. Figure 3.7 shows a typical filter configuration with the dashed boxes corresponding to optional filters. In the case of an HMM recognizer, the system processes an input word image via the Glyph Tracer and the PAW Segmente followed by the Lexicon Reducer. The latter processing branch produces a reduced lexicon that is fed, together with the observation sequence generated by the former branch, to the HMM-based classifier, producing the final list of candidate lexicon entries. In the case of PAW-based CRF, the Glyph Tracer was not used and the recognition relied solely on features from PAW shape chains.

At the time of our experiments, there were not very many publicly (or even commercially) available Arabic handwriting datasets with segmented ground truth. The best known standard set was the IFN/ENIT database of Tunisian villages [12] conveniently presegmented and cleaned up. Part of our research effort was to create some additional Arabic handwriting corpora with rich, XML-based ground truth, which resulted in *AMA corpus* [35], consisting of handwritten notes and lists. We also have created a small on-line dataset [35] using the IFN/ENIT database as a source of words and ground truth.

In our *LexiconReducer* → *HMM* combination approach [39], we identified the following three configurations as the most promising, based on the experimental results using both seen and unseen data [35]:

Table 3.1 Recognition rates on IFN/ENIT corpus: training on A, testing on B

Configuration	PAW rate	Term rate
$PAWSeg \rightarrow RLR \rightarrow HMM$	N/A	0.42
$PAWSeg \rightarrow CRF$	0.64	0.51

DHMM showed good generalization to unknown data.

$DHMM \rightarrow RLR$ showed improvements in top-*N* rates and robust overall recognition performance.

$RLR \rightarrow DHMM$ showed improved performance when tested on known data and helped resolve some of the ambiguity that arises when HMM models are trained using data that contains a significant number of outliers.

Here DHMM stands for *Discrete HMM* modified to produce top-N most likely chains, and RLR stands for *Ranking Lexicon Reducer*. Overall, when tested on the IFN/ENIT corpus, the system achieved a 73 % top-1 word recognition rate on seen test data and 52 % on unseen data. While there is certainly room for improvement, what is encouraging is that combining different classification approaches produced better results than using the same approaches individually, which suggests that this methodology should be developed further.

An interesting direction for experimenting with our AHWR system was to use conditional random fields (CRFs) as an alternative to the HMM-based recognizer. The CRF-based recognizer would consume the output of our PAW Segmente, extracting features from the PAW shape chains and learning sequential labeling based on partially recognized glyph sequences. We have compared the performance of our CRF-based recognizer to that of an HMM-based classifier at PAW sequence recognition using the IFN/ENIT corpus. Notice that the HMM classifier used an HMM per *term* (whole village name) while the CRF recognizer was entirely PAW-based and general purpose (not term-based).

We trained on subset A and tested on subset B. The PAW-based ground truth was generated utilizing our PAW Segmente (for images) and PAW Splitter (for text) filters. The segmentation was imperfect and rather noisy, yet it was consistent (as similar glyph shapes were recognized as such by the same filters employed during the inference) and thus usable for testing. Table 3.1 shows the recognition rates separately for PAWs and for terms. The adaptation of the CRF algorithm was by no means complete or optimal for the task, and the training phase took substantial time (about 52 hours) to complete on a 3 GHz Intel Pentium D PC with 4 GB of RAM. The testing, however, was fairly quick and efficient, completing in a matter of minutes.

We have also tested our CRF-based off-line recognizer on several parts of AMA corpus (described in detail separately in this manuscript [35]) that we have collected as part of our Arabic handwriting recognition project. The PAW recognition rates on synthetically generated PAW shape chains appeared in a direct negative correlation with the level of noise introduced; i.e., starting with perfect accuracy and zero noise, we observed a 10 % decrease in accuracy for each 10 % increase in noise. On the

real (non-synthetic) seen data, our CRF recognition rate was about 75 % on, while on the unseen data it averaged at about 30 %. This was expected as AMA corpus exhibits a much greater variety of the written contents, layout, scanning quality, and kinds of paper used.

On-line Recognition For our on-line data collection task we have built a hand-writing capture application working in the tablet PC environment. When a small corpus based on IFN data was collected, we integrated our HMM-based AHWR system into the handwriting capture application and tested it on the captured data. The overall performance of the on-line recognition was respectable enough to make it into an interactive demonstration system for AHWR.

Figure 3.8 shows an example of the on-line handwriting recognition. The handwriting sample (on the right) is passed to the recognition engine as digital ink, and then the ten recognition candidates are displayed (on the left) in descending order of their similarity score. As we can see in the top example, the best candidate is separated from the rest by a fair score margin. In the bottom example, the best candidate has a similar shape to the second best, which is reflected by the score.

3.2.3 Post-recognition Accuracy Boosting

Although modern recognition technology is capable of handling a wide variety of document images, there is no single recognition engine that performs equally well on all input documents. This is especially true in cases of recognition problems or scripts/languages for which recognition technology has not yet fully matured. Examples of the former are handwriting recognition and processing of highly degraded documents; examples of the latter are recognition of African and Arabic script languages.

Since each recognition engine has its strengths and weaknesses, the same engine can vary in accuracy on different documents, and different engines can have different error rates on the same document image. Because recognizers are usually used as black-box software components, the problem of improving the accuracy of a single engine or a set of engines without being able to explicitly tune or modify them has attracted considerable attention in recent years. However, most of the existing systems do not go beyond variations on majority voting.

While voting may work well in many cases, it has limitations. For instance, a typical situation in which one would be inclined to use a multi-engine system is that of a language for which recognition technology is still maturing, when there are few independent engines available and some of these engines are expected to perform poorly on many documents. When majority voting is applied in this situation, its results are often skewed considerably by individual engine errors.

A more sophisticated approach would be to combine, in an optimal or near-optimal way, output streams of one or more engines together with various types of evidence extracted from these streams, original document images, statistics collected about the engines as well as knowledge of the specifics of the target script or



Fig. 3.8 On-line Arabic handwriting recognition

language, to produce higher quality final output. Besides added robustness, another advantage of such an approach over simple voting is that it can be applied to the case when only one or two engines are available.

Different types of evidence can be divided into two major categories with respect to the system processing a set of documents: static and dynamic. *Static* evidence is usually collected per language or script (e.g., language models) or per engine (e.g., character confusion map) using some learning corpus. Static evidence is used on all processed documents. *Dynamic* evidence is collected per document (e.g., im-

age speckle factor, engine disagreement) using both learning and testing corpora. Dynamic evidence is used immediately on the processed document.

In recent years, the problem of combining classifiers for accuracy boosting has been studied extensively in the context of many different classification problems [11, 16, 26, 40]. Abed and Märgner [7] give an overview of several papers published within the last ten years that study this problem for the specific case of Arabic text recognition. The methods presented in these papers attempt to perform classifier fusion both at the feature level [31] and at the classifier level [3, 9], as well as by using hybrid approaches [8]. Using the IFN/ENIT corpus, the authors [7] also evaluated several classifier-level combination schemes.

Although there has been a significant amount of work on combining classifiers to improve OCR that produced promising results [2, 13, 17, 19], most systems that are currently available still employ some variations of majority voting [18]. More sophisticated heuristically driven voting schemes [14] may suffer less from the above limitations, since they usually take into account additional sources of evidence, but they rarely assess the statistical optimality of the corrected OCR output.

In what follows, we describe our approach in some detail as it was applied to the problem of Arabic OCR [4, 38]. However, it is important to realize that this approach can easily be extended to other similar problems such as handwriting recognition and other scripts and languages. We approached the multi-engine OCR problem as a statistically optimal combination of one or more OCR streams given one or more types of evidence.

Our system is structured as shown in Fig. 3.9. Document images are processed with one or more OCR engines. The resulting output sequences of characters, together with the original document images, are passed to the Multi-Evidence Processor Engine. The Multi-Evidence Processor Engine aligns the output text from the engines [24] and also measures a number of other features, discussed in the section “Evidence Types” below. These features include both image-based features that are measured on the original input and text-based features that reflect the OCR engines’ output. They can serve as evidence in selecting the preferred *engine* output for a particular document and/or in directly selecting the most likely *text*, words or characters. The output streams and calculated features go to the Evidence Combiner and File Generator Engine, which combines these various inputs into a single output sequence of characters that it deems most likely. More specifically, its Evidence Analyzer Engine component identifies the significance of the measured features, turning them into evidence for particular choices; the Combiner and File Generator Engine resolves conflicts between the indicated results and produces a single output sequence of characters. Possible implementations of the Evidence Analyzer Engine and the Combiner are discussed in more detail in the sections “Single-Engine OCR Accuracy Boosting” and “Multi-Engine OCR Accuracy Boosting” below.

Evidence Types

We investigated several types of OCR-related evidence to be later combined in the post-OCR correction stage:

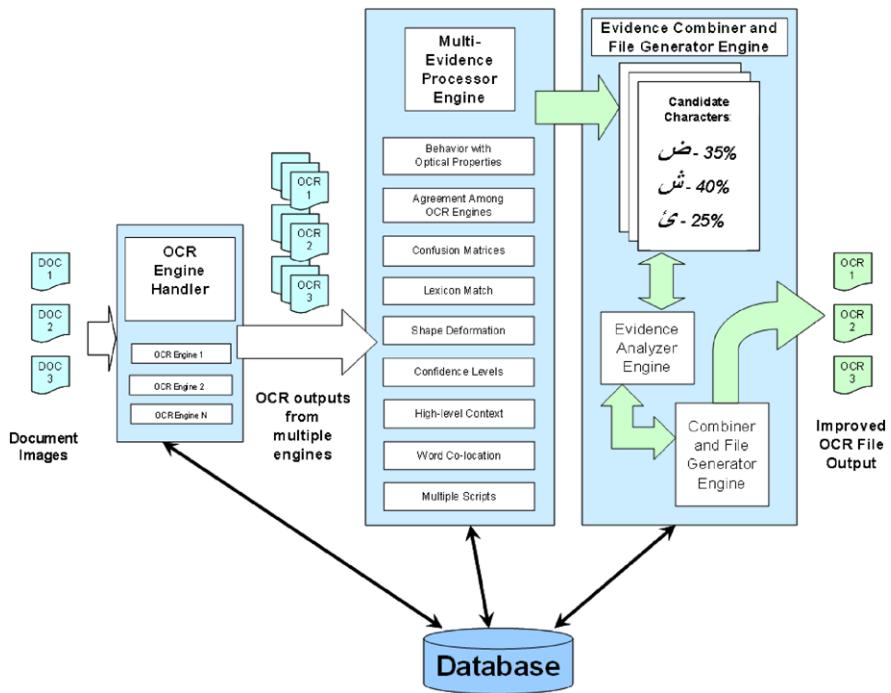


Fig. 3.9 Overview of the multi-engine multi-evidence OCR system

- *Image optical properties* measured directly from a document image.
- *Disagreement among OCR engines* at the character and word level.
- *Character confusion matrices* that record information about single characters as well as groups of characters.
- *Language models* on the character level (e.g., bigram statistics) and word level (e.g., dictionaries).
- *Characters as deformable shapes*, a measure that functions like a confusion matrix, but uses a measurement of the energy required to transform a skeleton of one character into that of another.
- *OCR confidence indicators*, such as the presence of unrecognized character markers in OCR output.
- *High-level context* such as language models tuned to specific subject domains.
- *Co-location*, using other words in a document as sources of corrections for a word likely to be incorrect.
- *Multiple scripts in a document* that help select an appropriate engine, or engines, for a given document.

For each evidence type, we identified three key characteristics:

- Whether it may serve as a basis for prediction of OCR output accuracy, its improvement, or both.

- Whether it may be attributed to single characters, words, or the entire document, i.e., whether it may contain potentially useful information at these different levels of output granularity.
- The extent to which it depends on the script or language of the document.

Most of these evidence types were found to provide at least some useful information that could be exploited to boost the OCR accuracy.

Single-Engine OCR Accuracy Boosting

The first application of this approach was to accuracy boosting of the output of a single OCR engine [4]. The major focus was on developing an OCR accuracy booster based on the hidden Markov model (HMM). The HMM filter modeled OCR engine noise as a two-layer stochastic process. The OCR correction problem is formulated as follows. Given a lexicon $\Lambda = \{w_1, \dots, w_N\}$ that contains all known words w_j , and an OCR output word sequence $O = \langle o_1, \dots, o_K \rangle$, find the known word sequence $W_{BEST} = \langle w_1, \dots, w_K \rangle$ with $w_j \in \Lambda$ which best fits O , i.e.,

$$\begin{aligned} W_{BEST} &= \operatorname{argmax}_W P(W|O) = \operatorname{argmax}_W \left(\frac{P(W)P(O|W)}{P(O)} \right) \\ &= \operatorname{argmax}_W P(W)P(O|W), \end{aligned} \tag{3.1}$$

where the terms of the right-hand side of (3.1) can be further decomposed into finer components [15].

$P(W)$ probabilistically models the language, e.g., via word bigram statistics extracted from a large corpus of ground truth text, while $P(O|W)$ models the given OCR engine, e.g., via a confusion matrix [33]. The OCR output O is corrected by W_{BEST} .

Our approach is similar to the above generic framework, but it works mostly on the character level. That is, the lexicon Λ is replaced by an alphabet $A = \{c_1, \dots, c_N\}$ that contains all known characters c_j , while O represents an OCR output character sequence. This sequence, as well as others derived from it, are scored in terms of probabilities $P(C)$, which is determined from character bigram statistics computed from a large corpus of text, and $P(O|C)$, which characterizes a given OCR engine via a character confusion matrix precomputed over the corpus. This method is not purely character-based, but may include some word-level information. When character bigram statistics are computed, the number of times a given bigram appears in a given word may be multiplied by the number of times this word appears in the corpus.

Experiments of the resulting system revealed its versatility in applications to different languages (it was able to handle both English and Arabic) as well as its robustness and generalization power (e.g., in correcting words on which the filter was not trained).

Multi-engine OCR Accuracy Boosting

The system described in the section “*Single-Engine OCR Accuracy Boosting*” above is an example of a *single-filter* approach to implementing a multi-evidence boosting system in which all types of evidence are combined together into a single mathematical apparatus like HMM. The major advantage of this approach is that it is possible to assess its optimality in terms of a formulation like (3.1). The disadvantage is that, when combining several possibly heterogeneous evidence types, it yields a large and complex filter that is difficult to implement, maintain, and reuse. Therefore, when building a multi-engine boosting system, we adopted a *multi-filter* approach where the system is built as a collection of filters connected sequentially and/or in parallel, with each filter based on a single evidence type. Below we describe several filters that we have developed.

Lexicon look-up is applied to a single stream and performs word spell checking based on a lexicon collected from a large body of text. Words from a stream are selected for correction by the filter if they contain unrecognized character markers that some OCR engines produce.

Multiple stream processor does not correct anything by itself. Instead, it aligns multiple output streams using character-level synchronization implemented in the `synctext` utility of the ISRI OCR toolkit (open source software) by Nartker et al. [24]. Synchronization results are used, for instance, as input to the voting corrector filter described next.

Voting corrector is a trainable filter applied to multiple OCR streams that uses patterns of disagreement among engines as well as various heuristics to resolve the cases when simple voting will likely fail, e.g., when three OCR streams each produce a different candidate character for the same position in the document.

Triage selects one or more better-quality streams using various text statistics collected from a stream. These statistics include the total output size in bytes, the total number of correct words, median and mean lengths of correct words, and other similar metrics. The filter makes a decision based on analysis of these statistics computed for the given OCR streams.

These individual filters were combined in several different configurations, and the resulting OCR boosting systems were evaluated. One of the configurations, which consisted of the lexicon look-up applied to individual output streams of three commercial Arabic OCR engines followed by triage and in turn followed by the voting corrector filter, performed better than majority voting or any of the engines individually.

3.3 Filter-Based Document Processing

Automatic document analysis typically involves multiple stages. Here we present a multi-stage approach to the document analysis problems, and describe the filter-

based document processing framework we developed to some problems of Arabic text recognition, namely Arabic handwriting recognition (AHWR) and multi-engine OCR accuracy boosting, both of which have been described above. The framework defines and operates in terms of the following core classes of objects:

Document can be thought of as a virtual page binder, that various filters can look at, add new pages to, or update the existing *pages*.

Page is a catalog of named *streams* carrying the source contents and some derived information that various *filters* would act on.

Stream is a sequence of generic tokens (e.g., image patches, characters, or words) that filters can generate and/or process. A stream can encapsulate arbitrary data structures that are consumed/produced by one or more document *filters*.

Filter is an individual action unit that can process any of its non-obscured streams or the whole document, if necessary.

As illustrated by Fig. 3.10(a), a filter would typically process one or more information streams from the last page of the document and append a new page with its results also stored in one or multiple streams. By default, document pages are *transparent*, referencing contents of previous pages. Filters can augment (modify, erase, or obscure) the page contents or introduce some new information on a page. Such manipulations are usually performed on the last page, leaving all previous pages intact. This conservative architecture preserves the document's history and allows for process backtracking and partial process completion (e.g., in case of an error), with the possibility of a subsequent restart from any partial result point.

Filters can be combined into filter assemblies. A *filter assembly* is a macro-filter that maintains multiple sub-filters. Our framework implements two basic filter assembly patterns: *serial* and *parallel*. The filter assemblies run their constituent filters in separate processing threads. The serial filter assembly (a.k.a. *pipeline*) is particularly useful for efficient processing of document batches by a sequence of filters, and is typically used to represent application *use cases* and frequently used filter layouts, e.g.,

$$\text{ImageCleanup} \rightarrow \text{OCR} \rightarrow \text{TextCorrection} \quad (3.2)$$

The *parallel* assembly is efficient at spawning independent sub-filters that run in parallel, each producing or acting on independent streams. Examples of parallel assemblies are arrays of independent recognition engines, single stream word correctors, independent image feature extractors, etc.

Figure 3.10(b) presents a sample use case of filter-based document processing involving both optical (OCR) and intelligent (ICR) character recognition. The high-level assembly consists of three major stages (image processing, recognition, and post-processing) implemented as filter assemblies (pipeline, parallel and hybrid, respectively). A collection of document images is loaded by the Image Processing assembly, where document images are processed in a sequential pipeline of respective image filters. Then the Recognition assembly runs various recognition filters on the document image segments. The Post-processing stage then runs *triage* and

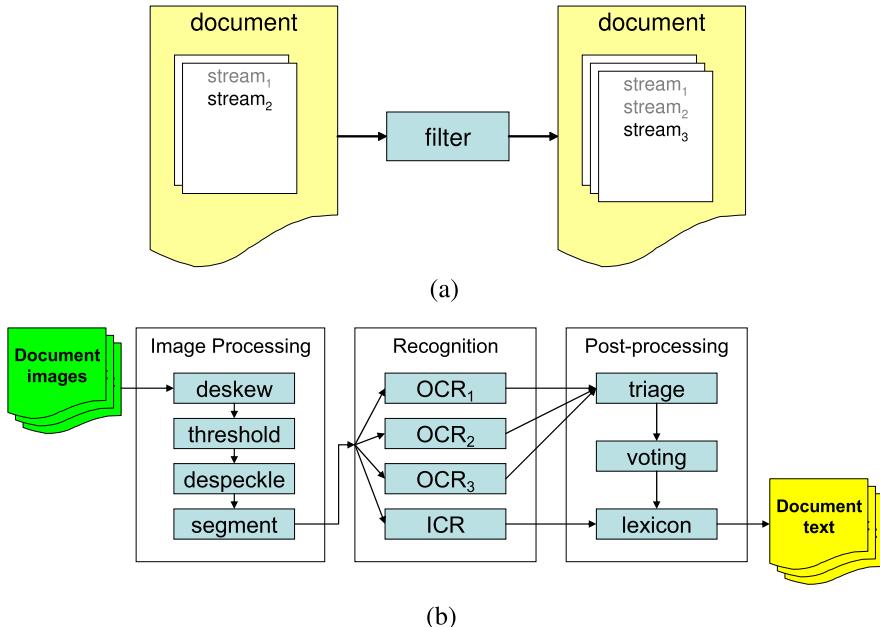


Fig. 3.10 Filter-based document processing: a typical step (a) and sample use case (b)

voting filters on multiple OCR streams and lexicon corrector runs in parallel on both OCR and ICR streams. The output text (e.g., main document text and handwritten annotations) is stored in XML format carrying document-level information for each resulting stream, e.g., recognition confidence values, detected reading order, and detected fonts.

Filters can compute, accumulate and use *batch-level* information (also referred to as batch evidence), e.g., lexicons, confusion matrices, or recognition models. Documents, on the contrary, usually carry *document-level* information (document evidence), e.g., measures of image noise, recognition disagreement measures, or dominant script/language. Trainable filters can “learn” from the accumulated evidence and adapt to the kinds of data in a given corpus/batch. Hence each filter has two basic modes: training and application.

All components of the system can serialize themselves from/to XML streams, thus allowing for great flexibility in dynamic component creation and configuration. For example, filters can store/load their state in/from the configuration files, thus allowing for incremental training. Documents can serialize to/from XML streams, thus allowing for partial results input/output. In a typical run, the complete input of the system consists of the initial filter configuration and an input document batch, and the complete output consists of the output document batch (with all derived per-document evidence) and the final filter configuration after the batch (with all the batch-level evidence).

3.4 Conclusion

In our view, Arabic document analysis is a multi-stage process involving document image cleanup, content segmentation, recognition, and correction. We have discussed the major stages of Arabic OCR and handwriting recognition (HWR), addressed their goals and challenges, and presented a multi-stage solution that we have developed, as discussed in Sect. 3.2. Section 3.3 introduced our *multi-filter* framework, which was utilized during the implementation of two specific systems for Arabic document analysis:

Multi-evidence Multi-OCR Engine (MEMOE) system, which uses multiple OCR engines to recognize a document and uses some extra information about the document and resulting OCR streams to produce an output that is more accurate than any of the individual OCR outputs. This system accounts for various types of OCR evidence extracted from its multiple OCR streams and from the original document images [38]. The system was designed with the goal of improving the OCR accuracy on images that were likely to result in low recognition rates, which in turn could significantly impact downstream processing (categorization, NEE, MT). The system was tested on (very) noisy Arabic documents and produced some encouraging results, beating majority voting and each individual OCR engine it worked with.

Arabic Handwritten Word Recognizer (AHWR), which combines several classifiers and lexicon reducers in both sequential and parallel fashion to improve recognition results relative to the individual classifiers [39]. The set of individual components integrated as filters into the combined recognizer includes both internally developed classifiers and components developed externally (not exclusively for this integration), thus demonstrating that the proposed framework is well suited for implementation of systems based on collaborative efforts. The recognition accuracy of any system is inevitably biased by the training data (ours was no exception), but we were able to build a general-purpose on/off-line handwriting recognizer that was fairly accurate on datasets representative of its training corpora.

The proposed multi-filter framework can be extended into an advanced document processing system with feedback that would combine pre-recognition processing of Arabic document images (e.g., *ImageRefiner* described in Sect. 3.2.1), printed text and/or handwriting recognition (e.g., AHWR described in Sect. 3.2.2), and post-recognition error correction (e.g., MEMOE described in Sect. 3.2.3). The user in the loop approach (also implemented as a filter) can be taken to incorporate things like relevance feedback to expedite a system’s adaptability to the given data and to narrow the perceptual gap between machine and human recognition apparatus.

Acknowledgements We would like to express our thanks to David Doermann (UMCP) for providing a nice GUI tool for Arabic handwriting ground truth preparation, Sargur N. Srihari (CEDAR) for providing an efficient solution for document image segmentation, Volker Märgner (TU Braunschweig) for providing access to the IfN database and for evaluating our Arabic handwriting recognizer, Leila Saidi (CACI) for helping with Arabic documents ground truth creation, Ericson Davis (CACI) for providing technical solutions to document image processing, Anna

Borovikov (CACI) for coordinating Arabic corpora creation and for mathematical advice, Yaguang Yang (CACI) for providing custom document image transform and for software coding, Kristen Summers (CACI) for providing technical expertise and leadership in the area of document understanding in general and in Arabic OCR in particular, Mark Turner (CACI) for administrative project oversight, and Luis Hernandez (ARL) for access to various Arabic datasets and GOTS software.

References

1. AbdulKader, A.: A two tier Arabic handwriting recognition based on conditional joining rules. In: Proceedings of the Summit on Arabic and Chinese Handwriting, pp. 121–128 (2006)
2. Al-Ani, A., Deriche, M.: A new technique for combining multiple classifiers using the Dempster–Shafer theory of evidence. *J. Artif. Intell. Res.* **17**, 333–361 (2002)
3. Alma'aheed, S., Higgins, C., Elliman, D.: Off-line recognition of handwritten Arabic words using multiple Hidden Markov Models. In: Twenty-third SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, vol. 17, pp. 75–79 (2004)
4. Borovikov, E., Zavorin, I., Turner, M.: A filter based post-OCR accuracy boost system. In: Proceedings of the 1st ACM Workshop on Hardcopy Document Processing (2004)
5. Cannon, M., Hochberg, J., Kelly, P.: Quality assessment and restoration of typewritten document images. *Int. J. Doc. Anal. Recognit.* **2**(2–3), 80–89 (1999)
6. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. Wiley-Interscience, New York (1973)
7. El Abed, H., Märgner, V.: Comparison of combination methods of Arabic handwritten word recognizers. In: 5th International Multi-Conference on Systems, Signals and Devices, pp. 1–6 (2008)
8. Essoukri, N., Amara, B., Bouslama, F.: Classification of Arabic script using multiple sources of information: State of the art and perspectives. *Int. J. Doc. Anal. Recognit.* **5**(4), 195–212 (2003)
9. Farah, N., Souici, L., Sellami, M.: Classifiers combination and syntax analysis for Arabic literal amount recognition. *Eng. Appl. Artif. Intell.* **19**(1), 29–39 (2006)
10. Govindaraju, V.: Paradigms in handwriting recognition. In: Proceedings of the Summit on Arabic and Chinese Handwriting, pp. 171–175 (2006)
11. Ho, T.K.: Multiple classifier combination: Lessons and next steps. In: Bunke, H., Kandel, A. (eds.) Hybrid Methods in Pattern Recognition, pp. 171–198. World Scientific, Singapore (2002)
12. <http://www.ifnenit.com>
13. Jaeger, S.: Informational classifier fusion. In: Proceedings of the International Conference on Pattern Recognition, vol. I, pp. 216–219 (2004)
14. Klein, S.T., Kopel, M.: A voting system for automatic OCR correction. In: Proceedings of the SIGIR 2002 Workshop on Information Retrieval and OCR: From Converting Content to Grasping Meaning, University of Tampere, August 2002
15. Kolak, O., Resnik, P., Byrne, W.: A generative probabilistic OCR model for NLP applications. In: Proceedings of HLT-NAACL, May 2003
16. Kuncheva, L.: Combining Pattern Classifiers. Methods and Algorithms. Wiley, New York (2004)
17. Lee, D.-S.: A theory of classifier combination: the neural network approach. Ph.D. thesis, State University of New York at Buffalo (1995)
18. Lin, X.: Reliable OCR solution for digital content re-mastering. In: Proceedings of SPIE Conference on Document Recognition and Retrieval IX (2002)
19. Lin, X.: DRR research beyond COTS OCR software: A survey. Technical Report HPL-2004-167, Imaging Systems Laboratory, HP Laboratories Palo Alto, CA (2004)
20. Lorigo, L.M.: Arabic handwriting recognition and application to ancient documents. In: Proceedings of the Summit on Arabic and Chinese Handwriting, pp. 111–120 (2006)

21. Lorigo, L.M., Govindaraju, V.: Off-line Arabic handwriting recognition: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(5), 712–724 (2006)
22. Märgner, V., Pechwitz, M., El-Abed, H.: ICDAR 2005 Arabic handwriting recognition competition. In: Proceedings of the International Conference on Document Analysis and Recognition, Seoul, Korea, pp. 70–74 (2005)
23. McNamara, J., Casey, D., Smith, R., Bradburn, D.: A classifier for evaluating the effects of image processing on character recognition. In: Image Algebra and Morphological Image Processing, pp. 109–120 (1993)
24. Nartker, T.A., Rice, S.V., Lumos, S.E.: Software tools and test data for research and testing of page-reading OCR systems. In: SPIE Conference Document Recognition and Retrieval, San Jose, CA, January 2005, vol. 5676, pp. 37–47 (2005)
25. Niblack, W.: An Introduction to Image Processing. Prentice-Hall, Englewood Cliffs (1996)
26. Nomoto, T.: Predictive models of performance in multi-engine machine translation. In: MT Summit IX, pp. 269–276 (2003)
27. Otsu, N.: A threshold selection method for gray level histograms. *IEEE Trans. Syst. Man Cybern.* **9**, 62–66 (1979)
28. Parker, J.R.: Algorithms for Image Processing and Computer Vision. Wiley, New York (1996)
29. Pechwitz, M., Snoussi Maddouri, S., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT-database of handwritten Arabic words. In: Proceedings of CIFED, pp. 129–136 (2002)
30. Sezgin, M., Sankur, B.: Survey over image thresholding techniques and quantitative performance evaluation. *J. Electron. Imaging* **13**, 146–165 (2004)
31. Snoussi Maddouri, S., Amiri, H., Belaid, A., Choisy, Ch.: Combination of local and global vision modeling for Arabic handwritten words recognition. In: 8th IWFHR, pp. 128–135 (2002)
32. Summers, K.: Document image improvement for OCR as a classification problem. In: Document Recognition and Retrieval X, vol. 5010 (2003)
33. Tong, X., Evans, D.A.: A statistical approach to automatic OCR error correction in context. In: 4th Workshop on Very Large Corpora, Copenhagen, Denmark, August 1996, pp. 88–100 (1996)
34. Yang, Y., Summers, K., Turner, M.: A text image enhancement system based on segmentation and classification methods. In: Proceedings of the 1st ACM Workshop on Hardcopy Document Processing (2004)
35. Zavorin, I., Borovikov, E.: Data collection and annotation for Arabic document analysis. In: Märgner, V., El Abed, H. (eds.) Guide to OCR for Arabic Scripts. Springer, Berlin (2012)
36. Zavorin, I., Borovikov, E., Turner, M.: Initial results in offline Arabic handwriting recognition using large-scale geometric features. In: Proceedings of the Symposium on Document Image Understanding Technology, December 2005
37. Zavorin, I., Borovikov, E., Hernandez, L.: Adaptive pre-OCR cleanup of grayscale document images. In: Proceedings of SPIE/IS&T Electronic Imaging Conference on Document Recognition & Retrieval XIII. SPIE, Bellingham (2006)
38. Zavorin, I., Borovikov, E., Borovikov, A., Hernandez, L., Summers, K., Turner, M.: A multi-evidence, multi-engine OCR system. In: Proceedings of SPIE/IS&T Electronic Imaging Conference on Document Recognition & Retrieval XIV (2007)
39. Zavorin, I., Borovikov, E., Davis, E., Borovikov, A., Summers, K.: Combining different classification approaches to improve off-line Arabic handwritten word recognition. In: Proceedings of SPIE/IS&T Electronic Imaging Conference on Document Recognition & Retrieval XV (2008)
40. Zuo, Y., Zong, C.: Multi-engine based Chinese-to-English translation system. In: The INTERSPEECH/ICSLP-2004 Satellite Workshop: International Workshop on Spoken Language Translation (2004)

Chapter 4

Pre-processing Issues in Arabic OCR

Zhixin Shi, Srirangaraj Setlur, and Venu Govindaraju

Abstract Pre-processing of scanned documents is a necessary first step in the process cycle of any document processing application. While pre-processing methods are generally language independent, the effectiveness of downstream OCR processes can often be improved by language/script specific adaptations, particularly in the case of non-Latin scripts such as Arabic and Indic scripts. In this chapter, we present some techniques that have proven effective for the pre-processing of handwritten Arabic documents.

4.1 Introduction

The input to most optical character recognition (OCR) systems in use or under development today consists of off-line document images. These images are usually scanned from page documents of handwritten text written on various types of paper including paper with pre-printed rule-lines, logos, graphics, and printed text. The scanned images could be color, grayscale, or binarized images. Factors that could have a significant impact on downstream OCR performance include the *scanning resolution*, which could be as low as 200 dpi, *binarization artifacts* due to poor built-in binarization algorithms used in many low-end scanners, and various types of *noise*, ranging from dark backgrounds and salt-and-pepper noise (holes and small blobs) to non-text objects such as form lines or rule-lines, logos, and other graphics that need to be detected and removed. Figure 4.1 shows some examples of binary document images illustrating some of these pre-processing issues.

Another source of document images are historical manuscripts often found in libraries such as the United States Library of Congress. These collections of handwritten historical document images are typically digitized by photographing the original

Z. Shi (✉) · S. Setlur · V. Govindaraju
Department of CSE, University at Buffalo, SUNY, Buffalo, NY, USA
e-mail: zshi@buffalo.edu

S. Setlur
e-mail: setlur@buffalo.edu

V. Govindaraju
e-mail: govind@buffalo.edu

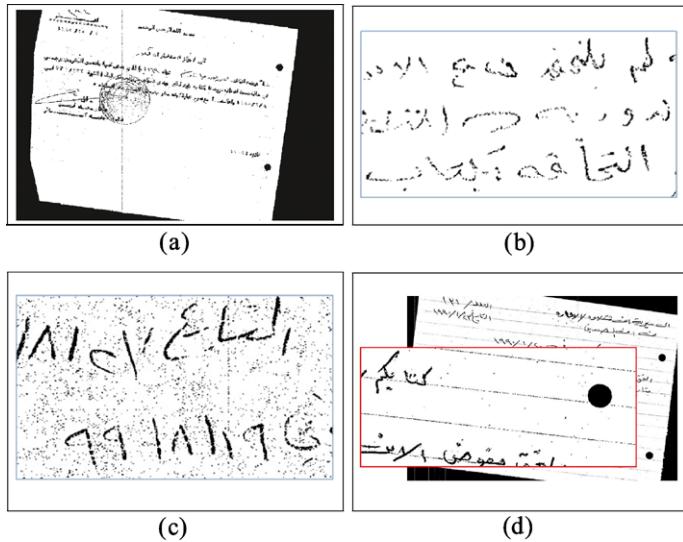


Fig. 4.1 Examples illustrating problems requiring pre-processing. (a) Irregular clutter noise, (b) salt noise resulting in broken strokes, (c) pepper noise, and (d) broken rule-lines, punch holes, and other problems

documents. The poor quality of historical document images is a result of multiple factors. Historical documents degrade due to aging and handling. Bleed-through of ink, dirt, and other types of damage are also frequently seen in very old documents. Since historical documents are also likely to be fragile, they are usually imaged using digital cameras instead of platen scanners. This causes the image to have uneven intensities. Enhancement methods are required to improve the readability of historical documents for various purposes such as visual inspection by historians and for OCR applications.

Figure 4.2 shows a typical historical handwritten document image, and Fig. 4.3 depicts the scanline view, which shows uneven background intensities across the document. It is therefore not trivial to separate the foreground text from the background. Ideally, the thresholding along the scanline for the separation should be a curve, rather than a straight line or lines determined by traditional global or locally adaptive thresholding algorithms [20].

Targeted image enhancement algorithms for a few noise categories are available in the literature. For example, surrounding noise is addressed in [1, 17], and filters such as modified median filters [8], the kFill operator [3], optimal Boolean filters [10], and modified directional morphological filters (MDMFs) [18] have been used to handle salt-and-pepper noise. However, when multiple noise types are present in a single image, these methods sometimes result in unsatisfactory results.

In this chapter, we present techniques for three important pre-processing tasks: (i) enhancement techniques for grayscale images, (ii) enhancement techniques for binary images, and (iii) line separation for binary handwritten images.

Fig. 4.2 Historical handwritten document image with uneven background

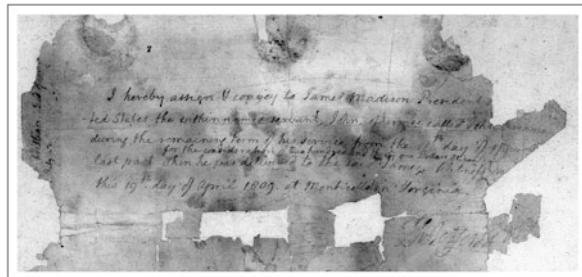
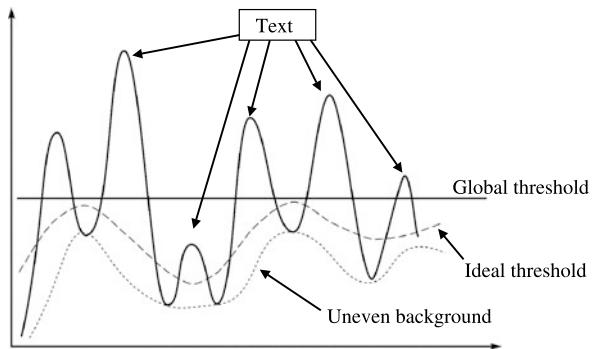


Fig. 4.3 Scanline view for a typical photographic image of a historical handwritten document



The enhancement techniques for grayscale images focus on enhancing the quality to enable better readability and better binarization.

The enhancement algorithms for binary images are designed to improve image quality for automatic OCR. The proposed algorithms are executed in sequence. Clutter noise, if any, in the document is first removed, and then a statistical approach is designed for determining whether the image contains any pepper noise. A pivotal text pixel identification process precedes a localized filtering algorithm to remove the pepper noise from the background. A region growing algorithm is then applied to enhance holes within text and broken text strokes (salt noise) by interpolation. Documents with form lines and rule-lines are detected if present. These are then removed using a stroke following approach without breaking the text strokes.

Generally, text line separation algorithms first locate the lines and then segment and label them in their original logical reading order. We describe some methods that have been very effective for line separation of handwritten Arabic documents.

4.2 Pre-processing for Grayscale Images

Three popular thresholding algorithms in the literature for text segmentation [11] are Otsu's thresholding technique, the entropy techniques proposed by Kapur et al., and the minimal error technique by Kittler and Illingworth. Another entropy-based method specifically designed for historical document segmentation is [13], which

deals with the noise inherent in paper quality, such as double-sided documents. Techniques for dealing with bleed-through text have been addressed in [28] (using direct image matching) and [29] (using directional wavelets).

In this section, a novel technique for historical document image binarization is described. This method, which is targeted toward enhancing images with an uneven background (Fig. 4.2), extends our earlier work [24] in which a linear model is used to adaptively approximate the paper background by using a nonlinear model for the approximation. The nonlinear model uses a combination of local and global line fitting approaches to find the best straight line segment that fits all the points within the neighborhood of each point on the scanline.

A linear approximation line at each scanline point is used to estimate the background intensity level at that point. A transformation based on a normalized approximation is used to even out the background, resulting in enhanced contrast with the foreground text. This method can also be adapted easily to color images.

4.2.1 Background Normalization for Variable-Intensity Background Grayscale Images

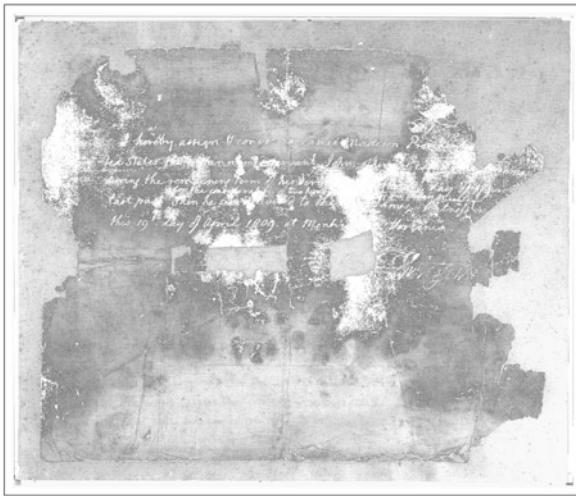
Many of the scanned document images encountered in application scenarios such as historical documents cannot be binarized using a simple global threshold. Even adaptive approaches that use thresholds over local patches do not work well on historical documents, since the background intensities vary significantly across the images.

An image can be looked at as a three-dimensional object whose positional coordinates are in the x - y plane and the pixel gray values are in the z direction. Considering the extreme case of a document image that does not have any textual content, the image will be approximately a two-dimensional plane which represents the background surface of the paper. A *traditional thresholding* model would find a plane H parallel to the x - y plane above the paper surface that can separate the z value representing the textual content. In historical documents with an uneven background, it is typically not easy to find a single plane surface H parallel to the x - y background surface that can separate the textual content.

Our initial approach to address this issue is to find a plane K above *most* of the background pixels with *almost all of* the foreground pixels above the plane. We can achieve this using a simple histogram. The objective of this initial approximation is to keep *most* of the foreground pixels above K . The pixels below this threshold plane K will contain some foreground pixels in addition to the background pixels.

Figure 4.4 is an image of the incomplete background, where the foreground pixels already removed by the approximate threshold plane are shown in white. The next step is to better approximate the background in small local regions.

Fig. 4.4 Majority of background pixels are below the thresholding plane



Background Separation by Linear Approximation

Since the leftover (residual) background in Fig. 4.4 may still have pixels which are likely to be part of the foreground text, we make a second finer pass over the image for separation of the foreground text and background.

To accomplish this, our first approach is to use a linear model to approximate the background in small local regions. The residual background image is first partitioned into m by n smaller regions, each of which approximates a flat surface in that local neighborhood. In each such region we find a linear function in the form

$$Ax + By - z + D = 0 \quad (4.1)$$

Pixels in this residual image are represented as points in the form (x_i, y_i, z_i) where (x_i, y_i) is the position of a pixel and z_i is the pixel value. We apply the minimal sum of distances,

$$\min \sum_i (Ax_i + By_i - z_i + D)^2 \quad (4.2)$$

where the sum is taken for all the available points in the residual image. The minimization gives a “best fit” linear plane (4.1) because the distance from any point to the plane in (4.1) is a constant proportional to $|Ax_i + By_i - z_i + D|$.

The solution for A , B , and D is obtained by solving a system of linear equations, which are derived by taking the first derivatives of the sum function in (4.2) with respect to the coefficients, and setting the derivative functions to zeros. Therefore, in each small partition we find a plane that is a best fit to the image background in the partition. The pixel value of the plane is evaluated by

$$z = Ax_i + By_i + D \quad (4.3)$$

for each pixel located at (x, y) .

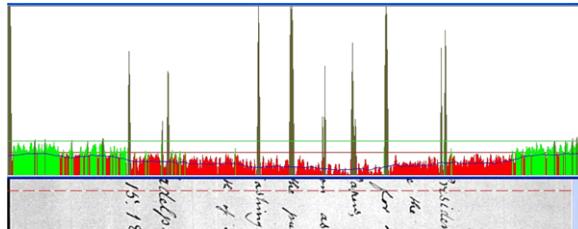


Fig. 4.5 Scanline histogram and background approximation. The histogram of black pixel intensity along the selected scanline for a grayscale document. The *horizontal line* is the average intensity level. The *curve* is the approximation of the background

Background Separation by Nonlinear Approximation

A second approach is to use a nonlinear curve for approximating the document background. For efficiency, the nonlinear approximation is computed along each scanline (Fig. 4.5).

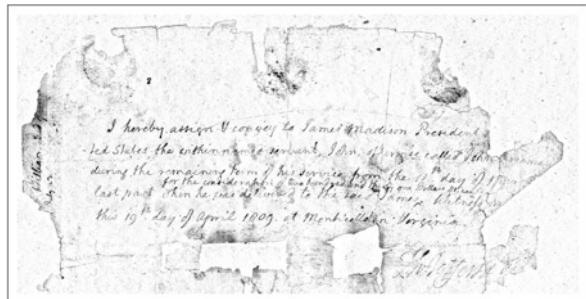
The histogram corresponding to text line regions shows taller peaks with large variations; background regions with no text pixels show a lower and less variant distribution. Also, the number of background pixels in a document image is typically significantly larger than the number of foreground text pixels. Our technique takes advantage of these observations.

We first compute the mean or average intensity level from the histogram. This average is used as a reference to set a background level at each pixel position along the scanline. The scanline is traversed from left to right. If the intensity level at the current position is less than the mean, then we will take this value for the next computation of our approximation and update a variable *previousLow* with the value of the current level. If the current level is higher than the mean, we will use the value in *previousLow* as the background intensity level at the current location for the subsequent computation of our approximation.

Thus, we have approximated a background intensity level for each pixel position on the scanline. This is just a rough approximation and is not very accurate for the following reasons. First, at the foreground pixel location, the foreground level is set based on a previous background level which may be used multiple times for a consecutive run of foreground pixels. Second, for low quality images, even the true background pixels may be assigned an intensity level that is very different from the ideal representation of the paper background. This roughly selected and estimated background (SEB) is an intermediate step for a better approximation.

Using the SEB pixel levels on a scanline, the approximation of the paper background can be done in two ways. The first uses a sliding window approach where, at every pixel position, the average of the SEB values in its local neighborhood is used to compute the approximated background intensity level. The second approach to find an approximated value on the curve at a point location is to use the SEB values in a neighborhood of fixed size, centered at the point, to find a best fitting line using squared distance minimization. The approximated value is the value on the

Fig. 4.6 Normalized historical document image showing an even background



line corresponding to the point location. The final approximation of the background calculated in this manner usually results in a nonlinear curve.

The original grayscale image can be normalized by using the linear or the nonlinear approximation. Assume a grayscale image with pixel values in the range 0 to 255 (0 for black and 255 for white). For any pixel at location (x, y) with pixel value z_{orig} , the normalized pixel value is then computed as

$$z_{\text{new}} = z_{\text{orig}} - z + c \quad (4.4)$$

where z is the corresponding pixel value on the approximated background; c is a constant fixed to some number close to the white color value 255. An example of a normalized image is shown in Fig. 4.6.

4.2.2 Experiment and Results

The primary target documents for the techniques described are handwritten historical or other degraded document images digitized by using either conventional platen scanners or digital cameras. The goal of the algorithms is to enhance the readability of the document images both for humans as well as automated recognition systems. Some images which are difficult to read for humans can be made at least human readable for digital library applications. For images which are unable to be read by automatic document processing systems due to problems with traditional binarization methods, our algorithm provides a better image enhancement method so that the enhanced images can be easily binarized using a global binarization algorithm.

The evaluation of image enhancement and binarization algorithms can be a challenging task, since the ideal achievable result is hard to define.

There are two possible approaches. One is through visual inspection to compare the resultant images (after processing) against the original images to make a qualitative judgment of the efficacy of the algorithm. Another approach is to provide a quantitative metric by evaluating the improvement in performance in a downstream process such as OCR results from a document recognition system. A quantitative metric can also be obtained by using synthetic noisy image data generated from clean binary images and evaluating the processed image against the clean, ideal image.

Fig. 4.7 An example Arabic historical document. *Above:* Color image of original document with uneven background. *Below:* Image after background normalization



Since our target images for the grayscale enhancement algorithms were aged, handwritten, historical document images, quantitative evaluation was not feasible. This is due to the following reasons. First, there is no readily available document recognition system for handwritten historical document images, which makes it difficult to measure the algorithm using OCR performance. Second, it is very difficult to define the ground truth for a document image in terms of quality. Although measures such as the number of readable words or characters in an image could be candidates, it is still a subjective determination. And last, finding an algorithm to simulate the aging process to generate synthetic images is equally challenging; therefore, using synthetic image data is also not practical.

For the reasons outlined above, a qualitative evaluation was used to determine the efficacy of our algorithm. Improving image quality for human readability was the primary goal of the evaluation.

The results demonstrate that the enhanced images show a marked improvement in image quality for human reading. Figures 4.7 and 4.8 show an Arabic historical document image and intermediate results using the techniques described in this chapter.

There are many document binarization algorithms available in the literature. As discussed in the introduction, most of the methods look for a threshold value globally or locally. When the threshold(s) is/are found for a document image, the image is then segmented into foreground and background based on the threshold. The difference in our method is that we find a curve to approximate the background of the image, which allows segmentation of the foreground and background of an image

Fig. 4.8 *Above:* Binarized image after background normalization. *Below:* Text line separation using the binarized image



along a curved plane instead of a straight plane. Our test is designed to adjust the claim. We downloaded 100 historical handwritten document images from the Library of Congress website. These images were selected because they all have obvious uneven background problems. From these images, we chose 20 random images for the test. These images could not be segmented easily with a global threshold. The images were successfully binarized using a global threshold after application of our normalization algorithm. Example images can be found in Figs. 4.9 and 4.10.

4.3 Pre-processing for Binary Document Images

Scanned documents, after binarization, often produce a number of artifacts that hinder automatic recognition systems. Removing non-text objects and improving the quality of the text are primary objectives for pre-processing. In this section, we describe methods to overcome four different types of issues.

1. *Clutter noise*, which we define as large connected components including large solid black areas of various shapes. These typically result from improper scanning.
2. *Pepper noise*, which consists of small connected components mostly due to over-thresholding of document images scanned from text written on dark paper (or other poor contrast settings) or dirty surfaces.

Fig. 4.9 The best possible binary image obtained from the original document image in Fig. 4.2 using a global threshold. The binarized image shows significant parts of the text obliterated

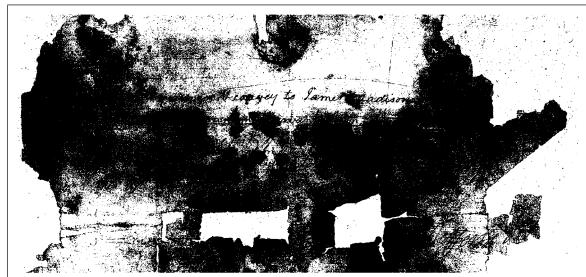
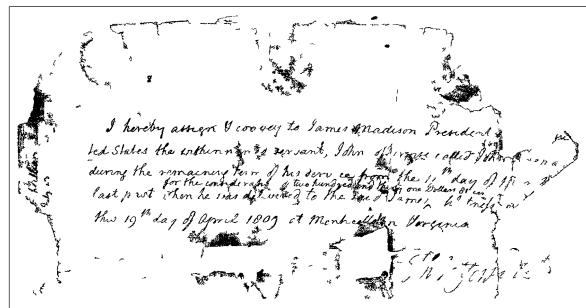


Fig. 4.10 The image in Fig. 4.2 normalized using the method described in this chapter can be binarized better using a global threshold



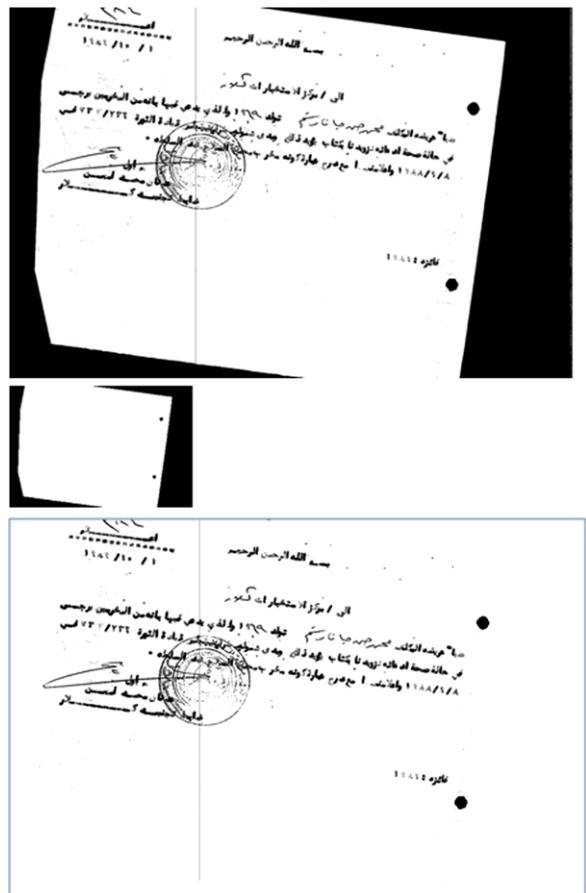
3. *Form lines and rule-lines*, where the objective is to detect and remove the pixels from the lines without removing any of the interfering text pixels. It is typical to find handwritten text in form fields cutting across the form lines.
4. *Salt noise and broken strokes* or holes or gaps within text strokes, which usually are a by-product of under-thresholding or pre-processing steps such as line removal. Light uneven writing, such as pencil or marker writing, may also result in holes within strokes after binarization.

4.3.1 Clutter Noise

Clutter noise is usually a result of improper scanning processes. This type of noise is characterized by dark strips or irregular dark solid areas along the borders of a document. These areas could also touch and interfere with text strokes near the border. We use a multi-resolution approach for the detection and removal of clutter noise. A biased downsampling algorithm is used to obtain a rough texture map of the document. The input binary document image is reduced to a fraction of the original size. This downsampling ratio is estimated based on a statistical sampling of randomly selected training images. The biased downsampling approach is an effective method for efficiently excluding the text components and focusing on the clutter noise.

The bias in the downsampling is achieved by using an n by n window where the pixel in the downsampled image corresponding to the center of the window is set

Fig. 4.11 Removal of clutter noise. *Above:* Original image. *Middle:* Downsampled image as a mask with text filtered out. *Below:* Document image with clutter noise removed



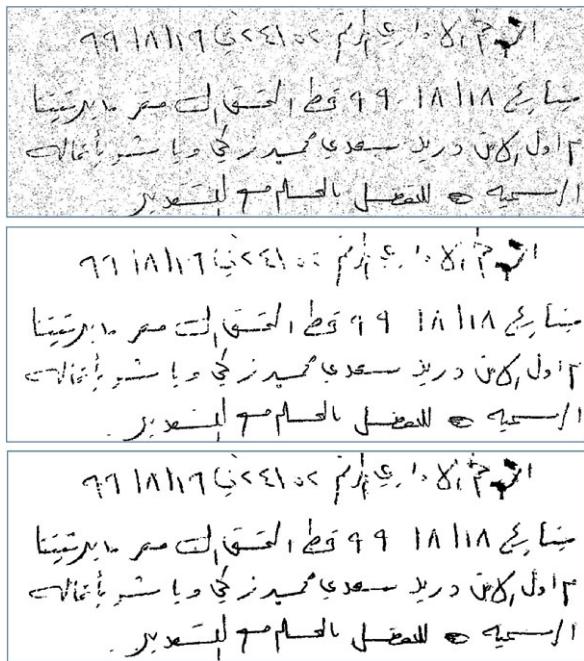
to black only if all the pixels in the window are black. This results in the region of clutter noise being retained as black pixels, whereas the pixels corresponding to the text strokes will be labeled white. The noise pixels thus identified can then be removed from the original image (Fig. 4.11).

4.3.2 Pepper Noise

Pepper noise is characterized by small connected components of black pixels that are not part of the foreground text.

Pepper noise can be identified by the size as well as the relative density of the connected components. A window of n by n pixels is used to determine a distribution sampling of small connected components. The window size and noise threshold are determined from randomly sampled training images.

Fig. 4.12 Removal of pepper noise by text pivoting. *Above:* Original image including pepper noise. *Middle:* Text mask image. *Below:* Result from pepper noise removal



Morphology operators and filter-based methods have been used in the literature for pepper noise removal [3, 8, 18]. Our method is different from these approaches in that it tries to preserve small components that are part of the text and are not noise and it also handles pepper noise over a wide range of sizes.

Intuitively, the algorithm attempts to aggressively remove pepper noise that is not close to text components; a more conservative approach is used if the noise is near text elements. This is achieved by first using a morphology operator and a size-based filter to roughly identify noise pixels. The remaining pixels form a mask image which consists of primarily the text region. This mask image is then used to generate what we term a *noise evaluation image*. Each black pixel which is in the original image but not in the mask image is considered as a potential noise candidate. At each noise candidate pixel, we apply a filter using a window of n by n pixels and calculate the approximate text pixel density by counting the number of black pixels in the mask image. If there are no black pixels in the region corresponding to this window in the mask image, then the current noise candidate pixel is removed as being true noise. Otherwise, the pixel value of the current noise candidate is set to a grayscale value between 0 and 255 based on the text pixel density corresponding to the window in the mask image. The new grayscale values are saved into a new image buffer that we call the noise evaluation image. A simple thresholding algorithm is then used to binarize this noise evaluation image. The final output of the pepper noise removal is a combination of the mask image with the thresholded result from the noise evaluation image (Fig. 4.12).

4.3.3 Rule-Lines

A directional local profiling approach is used for the detection of the rule-line locations. This is followed by a refined adaptive vertical runlength search for removing the rule-line pixels. This approach minimizes breaks in text strokes when the rule-line intersects the text.

In an ideal image, it would be trivial to detect the location of the rule-line with a simple projection profile which would show a distinctive peak along the center of the rule-line. However, the projection profile is very sensitive to skew in the image. In many realistic scenarios, there is significant skew, and many times binarization issues will result in broken rule-lines. In such cases, a simple projection profile would not be helpful in detecting the rule-lines accurately.

In order to accurately detect rule-lines, we apply an adaptive local connectivity transform [26] to a document image. This connectivity measure can be intuitively understood as the likelihood of a pixel belonging to a line, thus acting as the localized version of a projection profile. We first transform the document image into a connectivity map using the concept of a fuzzy runlength.

The fuzzy runlength at each foreground pixel is the length of the cumulative horizontal run in either direction. The *fuzzy* nature of the computation is introduced by the fact that small gaps in the run are ignored when computing the runlength. The maximum length of the accumulated gaps is capped at a predetermined threshold. The connectivity map is a two-dimensional matrix that is the size of the original binary image. Each entry in the matrix is the fuzzy runlength at that pixel position (Fig. 4.13).

We use this matrix as an image and binarize it by using a modified local adaptive thresholding algorithm to reveal the locations of the rule-lines. The fuzzy runs amplify the pixel intensities for the pixels that are on the rule-lines.

The rule-lines encountered in the real-world document images in our test set show a wide variation in thickness, sometimes even within the same image. Figure 4.13 illustrates this variation. Although almost all of the pixels on the rule-lines are covered by the detected line patterns, these line patterns cover pixels belonging to text strokes also. This problem is particularly accentuated in Arabic documents, where the text strokes run along the rule-lines. In the areas where the rule-lines intersect with the handwritten text, the detected line patterns in Fig. 4.13 are generally thicker than the real rule-lines. In the case of very thin or disconnected rule-lines, the corresponding line pattern may also be disjointed.

To overcome this problem, we use the detected line patterns in Fig. 4.13 to reconstruct the true rule-lines by estimating the best fitting line using linear regression. See Fig. 4.14 and Fig. 4.15. Using the reconstructed rule-lines, we trace the vertical runs in the original document image. If a vertical run is entirely overlapped by a reconstructed rule-line, the run is removed from the original document image. If a vertical run is longer than the width of a rule-line, we keep the run. Figure 4.16 and Fig. 4.17 show the result of rule-line removal for the original image in Fig. 4.13. This ensures that pixels belonging only to the rule-lines are removed, while the pixels belonging to intersecting text strokes are retained.

Fig. 4.13 Rule-line detection using fuzzy runlength. *Above*: Original image including rule-lines. *Middle*: The fuzzy runlength image. The fuzzy runlength image is a grayscale showing the connectivity of the foreground pixels. *Below*: Binarized fuzzy runlength image reveals the approximate locations of the rule-lines

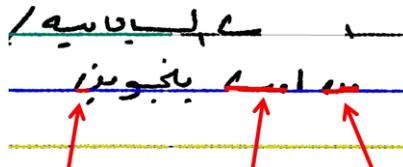
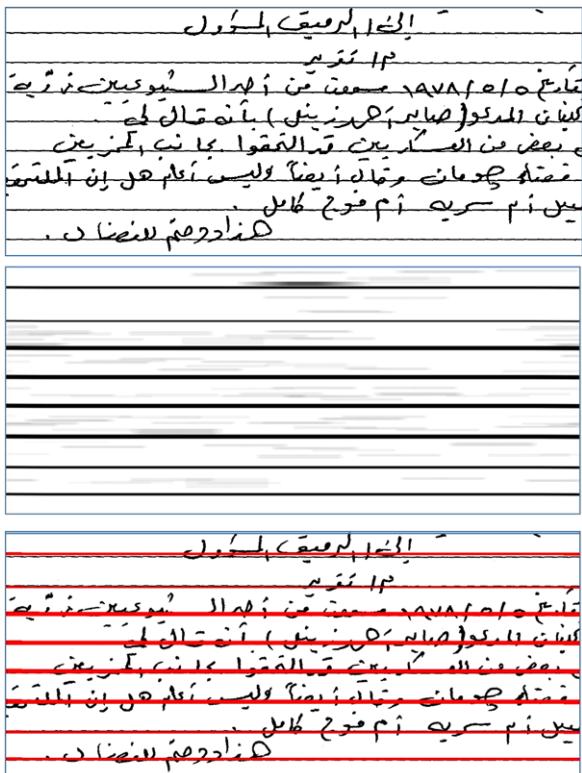


Fig. 4.14 Using the average thickness of a rule-line estimated from its line pattern, the thick areas are marked out (pointed by arrows). The rest of the pixels in the line pattern will be used in the linear regression estimation of a best fitting line

4.3.4 Salt Noise or Holes

Salt noise refers to holes and gaps in text strokes. While the fix for pepper noise is to remove the noise pixels, the fix for salt noise is to add black pixels. Salt noise can be an unintended by-product of binarization, pepper noise removal, or rule-line removal. We use a modification of the binary image enhancement algorithm presented in [22] to address this problem. This method uses a region growing approach to fix broken strokes.

Fig. 4.15 Using the linear regression method, a best fitting line is estimated (above). Then the entire rule-line is reconstructed by filling in pixels around the best fitting line (below)

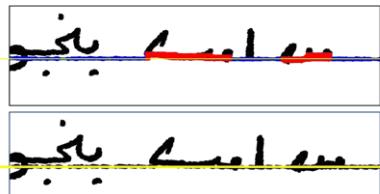


Fig. 4.16 Removing rule-line pixels by removing vertical runs covered by the reconstructed lines

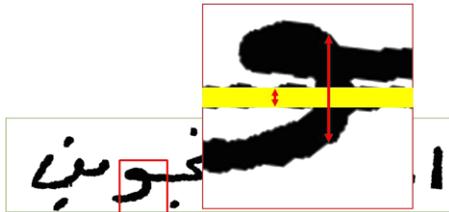
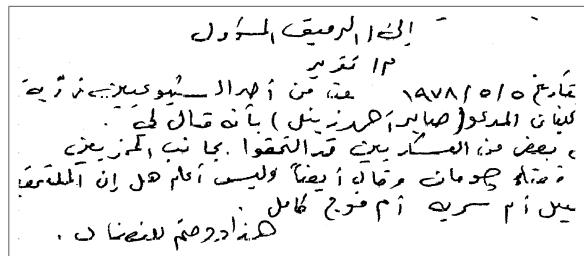


Fig. 4.17 Rule-line removal result for image in Fig. 4.13



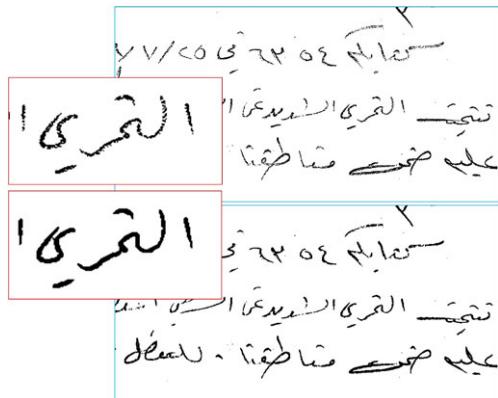
The algorithm uses a deformable filter window of n by n pixels. Starting at the black pixel on which the window is centered, it is sheared to the left or right until the parallelogram shape includes the maximum number of black pixels in the window neighborhood. Within the window, gaps are filled row by row (and column by column) using the algorithm in [22] to enhance the strokes by filling the gaps. This region growing algorithm continues to enhance the strokes until all the pixels in the image are visited. Figure 4.18 shows an example of the enhanced image.

4.3.5 Experiments and Results

The pre-processing algorithms have been tested using a set of 204 handwritten Arabic document images from the DARPA MADCAT data.

Visual examination of the experimental results show that the algorithm for removal of clutter noise is efficient as well as effective on all images containing clutter noise. The pepper noise removal algorithm is evaluated based on the results of text line separation since the efficacy of the text line separation is severely influenced by the presence of pepper noise. Using our algorithm for pepper line removal boosts

Fig. 4.18 Binary image enhancement by region growing for fixing salt noise and broken strokes



the line separation rate to 95 % from the 48 % line separation rate before using the pre-processing algorithm.

To evaluate the rule-line removal, we decide that a rule-line is detected and removed if 90 % of the pixels on the rule-line are removed. We count the total number of removed rule-lines in the images include rule-lines. The successful rule-line removal rate is 91 %.

Since downstream OCR results were not feasible, the region growing algorithm to fix salt noise was evaluated by visual examination of the images exhibiting the problem. The resulting images after our enhancement process showed significantly improved text image quality due to a reduction in the number of broken strokes, smoothing of the boundary of the written strokes, and “even”ing of the width of the strokes.

4.4 Text Line Separation

Finding text lines is an important step in the pre-processing of a document recognition system. A text line separation process segments a page document image into isolated text lines which are organized according to the natural reading order. The performance of a handwriting recognition system depends heavily on the results of the text line extraction process. Although line separation is usually trivial in the case of machine printed documents, text line extraction poses many challenges for handwritten document images. The typical challenges include (i) variability in text line orientation between different text lines, (ii) varying skew within the same text line (undulating lines), (iii) overlapping text in pages with crowded writing where characters of adjacent text lines have overlapping bounding boxes, (iv) characters in one line touching text in adjacent lines, and (v) the presence of small symbols such as those seen in Arabic which float between text lines (see Fig. 4.19).

There are a number of text line extraction algorithms in the literature. The projection profile method [4, 16, 21] produces a histogram along the direction of the text lines in a text block. The valleys of the histogram represent inter-line gaps,



Fig. 4.19 Challenges in handwritten Arabic document images

which are located to segment the text lines. Hough transform-based methods are theoretically identical to the projection profile methods [27]. A set of selected directions are used, and straight lines along the direction are determined to fit the text elements. The best fit gives the general orientation and the location of the text lines. The Hough transform can be applied on all black pixels, on reduced data from horizontal and vertical runlength computations [7], or only on the bottom pixels of connected components [9]. A nearest neighbor clustering of connected components can also be used for text line separation [15]. Most of these methods are designed primarily for machine printed documents and provide good results on printed documents only. They cannot usually be directly adapted to handwritten documents due to their reliance on global features in an image and are therefore more suited for well-structured documents.

Compared to machine printed documents, handwritten documents have more complex local structures such as undulating text lines, skewed text lines, overlapping characters, and touching and/or crossing text lines. There are methods that have been designed specifically for handwritten documents addressing these challenges. Due to the local rather than global structure characteristics in handwritten documents, the methods in the literature for handwritten documents are generally “bottom-up” and based on local analysis. Most methods extract text lines by grouping the basic text elements such as pixels, connected components [12], or local minima detected from a chain code structure [5]. These grouping algorithms are designed based on heuristic rules [12], learning [19], or searching in structures [14]. The local-global algorithm in [2] first partitions a document image into vertical strips. In each of these strips, the algorithm applies a projection profile algorithm with the assumption that the lines in a strip are almost all parallel to each other.

Most of these methods are dependent on the isolation of text elements such as strokes or connected components. When adjacent text lines touch each other, split-

ting of connected components, which is often difficult, must be performed before the location of the text lines can be detected. One other problem is that these methods use local decisions in the grouping process. They sometimes tend to be “trapped” by strong local features. For example, in a page image with very crowded text, connected components close to each other may not necessarily belong to the same line.

In this chapter, we describe a text line extraction algorithm for handwritten documents. The algorithm is based on an *adaptive local connectivity map* (ALCM) generated using a steerable direction filter. The text line extraction method is specifically designed for addressing the complex problems in handwritten documents.

4.4.1 Line Separation Methodology

Humans can identify text lines in a document image by detecting the text line zones as patterns on a reduced scale of the image. On a downsampled scale the text lines appear distinct as zones or patterns, and the touching between lines loses prominence.

Based on these observations, we have developed an adaptive local connectivity feature [26] to change the scale of a document image to reveal the distinctive text line patterns. At each pixel, we define a connectivity measure by cumulatively collecting its neighboring pixels’ intensities along a predetermined direction. Intuitively, the connectivity measure can be understood as the likelihood of a pixel belonging to a line. As measured by the connectivity, the pixels in between text lines are less likely to have an influence on the location of the text lines.

A fuzzy runlength [25] concept as a relaxed version of runlength computed from background pixels in a binary image was also considered. It emphasizes using background features for text line extraction and it can efficiently extract text lines for complex documents including mixed objects of graphics, handwritten, and printed text. The drawback in the method [25, 26] is that it cannot adequately handle fluctuating lines and lines with large skew.

We present in this chapter a generalization of the method in [26]. Instead of using a rectangular window filter along the fixed horizontal direction to generate the ALCM, we propose a steerable directional filter which calculates local connectivity features from multiple directions. The most likely text line direction is then captured by the maximum directional connectivity from the multiple directions.

Our proposed text line extraction algorithm consists of the following steps. (1) Convert a downsampled version of the input image into an ALCM by using a steerable directional filter. The resulting ALCM is represented as a grayscale image. (2) Binarize the ALCM using a local adaptive thresholding algorithm. The text line patterns are revealed as connected components in the binary ALCM. (3) Group the connected components into location masks, one for each text line. (4) Extract the text lines by collecting the connected components corresponding to the location

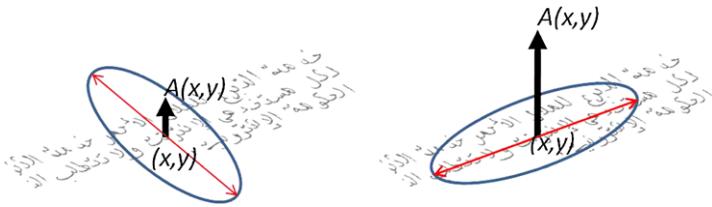


Fig. 4.20 ALCM $A(x, y)$ using the same size filter aligned along different directions. Using the filter along the direction of the text (*right*) the ALCM has a greater response (connectivity) than when using the filter in any other direction

masks in the original input binary image. (5) Group the leftover small components into the spatially closest text lines. (6) When a connected component touches more than one text line pattern in the ALCM, apply a splitting algorithm to split the component into pieces, and group each of them with the closest text lines.

ALCM Using Steerable Filter

Let $f : R^2 \rightarrow R$ represent a document image, which can be either binary or grayscale. The adaptive local connectivity map (ALCM) is defined as a transform

$$\text{ALCM} : f \rightarrow A$$

by the convolution

$$A(x, y) = \int_{R^2} f(x, y) G_{a,b}^{\theta_0}(x - t, y - s) dt ds \quad (4.5)$$

where

$$G_{a,b}^{\theta_0}(x, y) = \begin{cases} 1 & \text{if } (x, y) \in E_{a,b}^{\theta_0} \\ 0 & \text{otherwise} \end{cases} \quad (4.6)$$

where $E_{a,b}^{\theta_0}$ is an ellipse with semi-minor axis a, b and rotated by an angle θ_0 :

$$E_{a,b}^{\theta_0} = \left\{ (x, y) \mid \begin{array}{l} x < a \cos(\theta - \theta_0) \\ y < b \cos(\theta - \theta_0) \end{array} \text{ and } 0 \leq \theta < 2\pi \right\}$$

When we choose a longer than b , the ellipse $E_{a,b}^{\theta_0}$ is an elongated mask aligned with its long axis in the θ_0 direction. ALCM is a convolution that aggregates the pixel intensities within the mask centered at (x, y) . When the long axis of the filter is aligned with the direction of a text line, the ALCM value $A(x, y)$ at the pixel location inside the text line will be greater than the value along any other direction (see Fig. 4.20).

The transform can be implemented in many ways. We present a simple implementation in this chapter. For convenience, we first reverse the input image so that 255 represents the strongest level of intensity for foreground text. Often handwritten document images are scanned with resolutions ranging from 200 to 300 dpi or

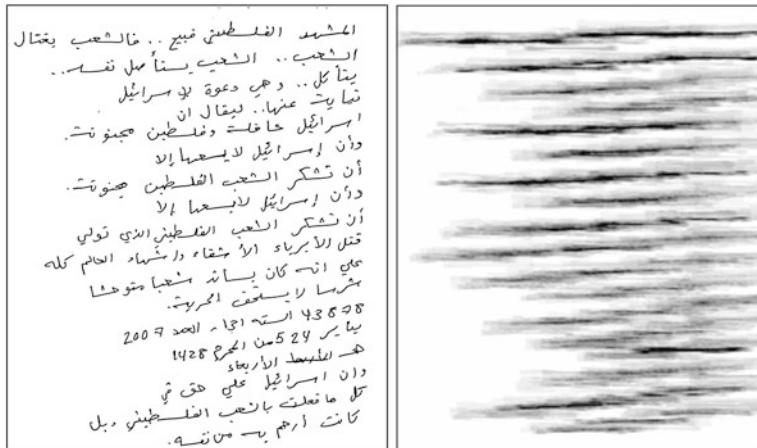


Fig. 4.21 A handwritten Arabic document image with varying skew, and its ALCM

higher, which is much higher than we need in text line location. A lower resolution image is enough to retain the necessary information. So, we downsample the image to 1/4 of its original size (1/2 in each direction). Values a and b , which determine the size of the steerable filter, are generally chosen as follows. b is chosen to be a value less than the height of the text, and a is chosen to be long enough to capture the location profile aggregate. Our experiments showed that 5 times the text height is a reasonable value for a . We estimate the text height dynamically. Our experiments show that our method tolerates a large range of a and b .

Choosing different values for θ_0 for the use of multiple directions in the filter allows the extraction of text lines with changing skew and undulation. In our experiments, for efficiency, we choose five directions: horizontal, slope ratio of 1 in 10 and 1 in 20, and their negations. Finally, we rescale the resulting ALCM values to the range from 0 to 255 to obtain a true grayscale image (see Fig. 4.21). A binarization algorithm can then be used to convert this grayscale image to a binary image.

Location of Text Lines

The value of each pixel in an ALCM image is the cumulative intensity of the foreground pixels in the elliptical neighborhood around the pixel in the original document image. A higher pixel value in the ALCM implies that the pixel is in a dense text region. We therefore classify the pixels in the ALCM into two classes, one for text regions and one for less likely text. A binarization algorithm is adapted for the classification.

Otsu's global thresholding algorithm is used in [26] to binarize the ALCM. The algorithm works well for most English handwritten documents, including historical manuscript images from the Library of Congress. However, we have found that it is hard to use any global thresholding algorithm in locating distinctive text line patterns

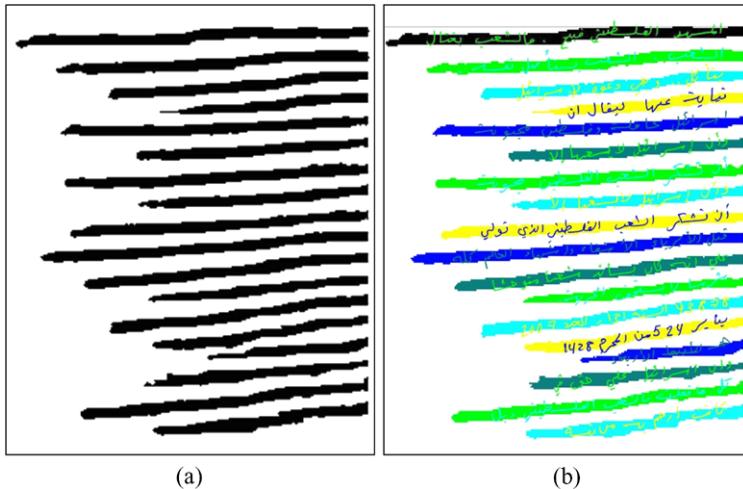


Fig. 4.22 Binarization of ALCM showing the patterns of text lines

when we have to deal with images with crowded writing, variability in thickness of the strokes, and irregular line spacing.

A local adaptive thresholding algorithm similar to that presented in [6] is modified and implemented. The algorithm determines a pixel's binary value by considering the pixel intensity distribution in a 5 neighborhood block structure. The 5 neighborhood blocks are 5×5 windows with one in the middle centered at the pixel under consideration and 4 other blocks adjacent to the corner of the center block. A weighted difference between the average pixel intensity in the middle block and that in the other 4 blocks is used to decide the center pixel's binary value. See [6] for the general algorithm. Our modification is in the implementation of the algorithm using a configurable value n for the block size. Figure 4.22(a) shows the binarization result using the adaptive thresholding algorithm. In order to use the connected components to form the complete line pattern, we do some filtering and reconstruction as follows. First, we filter out the small pieces. Based on our experiments, we observe that some pieces have a width that is significantly smaller than most other components. Usually, they are also short. These pieces are not required and should be filtered out. Second, for each connected component, we calculate its upper and lower profiles and also the center points. All the pixels within each pair of upper and lower profile points are filled to eliminate holes and gaps inside the line patterns.

In the ideal case, each connected component represents a complete text line. But sometimes a text line pattern is made up of two to three components and requires grouping. Grouping is done based on the horizontal alignments and on a determination of whether or not a group of two neighboring components is too wide to form a line.

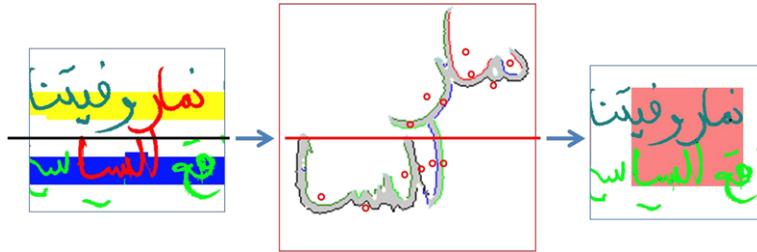


Fig. 4.23 Splitting a connected component crossing multiple text lines. *Left:* Image showing detected multi-line characters. *Middle:* Splitting of the contours with the center of mass marked. *Right:* Character images are split and grouped to the closest text line

Extraction of Text Lines

After grouping of the connected components in the binary ALCM, we have a group of connected components that can be used as a mask for an actual text line. Then the text lines in the original document image are extracted by a second round of connected component analysis as follows. (1) Generate connected components for the text in the original document image. (2) Upsample ALCM to the scale of the original image and superimpose the line patterns on the original document image. (3) For each text line pattern (the mask), collect all the connected components of text touching the pattern and group these components together to form the text line. (4) Group the small connected components that do not touch any line pattern with the closest neighboring line. Figure 4.22(b) shows the line patterns that are superimposed on a document image; the extracted text lines are shown in different colors.

If a text connected component touches more than one line pattern, then it represents characters that cross multiple text lines (the red color components in Fig. 4.23). These crossing pieces can be easily detected, but it is a nontrivial task to split them correctly and group them with the right text lines to which they belong. The touching connected components can be split using the splitting algorithm presented in [23]. For each touching component, we first draw a reference line between the line patterns. The splitting algorithm segments the contours of the piece into contour segments. Based on the relative location of the center of mass of the contour segments to the reference line, we group them with the corresponding text lines. The text images are then recovered using the contour segments (see Fig. 4.23).

4.4.2 Experiment

A set of 45 randomly chosen handwritten Arabic document images from the DARPA MADCAT data were used for our experiments. These images are binary images and were scanned at 300 dpi.

For performance evaluation, we used a connected component-based approach in which connected components are used as the basic text objects. The numbers

of connected components classified into the correct text lines are counted for the performance numbers.

There was a total of 1,022 text lines in the 45 test pages. The total number of text connected components was 32,936. There were two instances where two lines were merged incorrectly and 144 isolated components were incorrectly classified by the system. There were 178 connected components that touched more than one line and 14 of them were incorrectly split or grouped. The error for the incorrectly split components is counted twice. The overall performance measured in terms of correctly classified connected components was $(32936 - 144 - 2 \times 14) / 32936 = 99.5\%$.

Acknowledgements Part of this material is based upon work supported by the Defense Advanced Research Projects Agency DARPA/IPTO (PLATO: A System for Taming MADCAT: Multilingual Automatic Document Classification Analysis and Translation) ARPA Order No. X103 Program Code No. 7M30 issued through a subcontract from BBN Technologies Corp. under DARPA/CMO Contract # HR0011-08-C-0004.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency or the U.S. Government.

References

1. Agrawal, M., Doermann, D.S.: Clutter noise removal in binary document images. In: ICDAR, pp. 556–560 (2009)
2. Bruzzone, E., Coffetti, M.C.: An algorithm for extracting cursive text lines. In: ICDAR '99: Proceedings of the Fifth International Conference on Document Analysis and Recognition, p. 749. IEEE Comput. Soc., Los Alamitos (1999)
3. Chinnasarn, K., Rangsanseri, Y., Thitimajshima, P.: Removing salt-and-pepper noise in text/graphics images. In: IEEE APCCAS 1998, pp. 459–462. IEEE Comput. Soc., Los Alamitos (1998)
4. Ciardiello, G., Scafuro, G., Degrandi, M.T., Spada, M.R., Roccatelli, M.P.: An experimental system for office document handling and text recognition. In: Proc. 9th Int. Conf. on Pattern Recognition, pp. 739–743 (1988)
5. Feldbach, M., Tonnes, K.D.: Line detection and segmentation in historical church registers. In: ICDAR '01: Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR '01), pp. 743–747. IEEE Comput. Soc., Los Alamitos (2001)
6. Giuliano, E., Paitra, O., Stringa, L.: Electronic character reading system. U.S. Patent No. 4047152, Sep. 1977
7. Hinds, S.C., Fisher, J.L., D'Amato, D.P.: Document skew detection method using run-length encoding and the Hough transform. In: Proceedings of 10th International Conference on Pattern Recognition, pp. 464–468 (1990)
8. Kal Vin Toh, K., Isa, N.A.M.: Noise adaptive fuzzy switching median filter for salt-and-pepper noise reduction. IEEE Signal Process. Lett. **17**(3), 281–284 (2010)
9. Le, D.S., Thoma, G.R., Wechsler, H.: Automated page orientation and skew angle detection for binary document image. Pattern Recognit. **27**, 1325–1344 (1994)
10. Lee, W.-L., Fan, K.-C.: Document image pre-processing based on optimal Boolean filters. Signal Process. **80**(1), 45–55 (2000)
11. Leedham, G., Varma, S., Patankar, A., Govindaraju, V.: Separating text and background in degraded document images—a comparison of global thresholding techniques for multi-stage

- thresholding. In: Proceedings Eighth International Workshop on Frontiers of Handwriting Recognition, September 2002, pp. 244–249 (2002)
- 12. Likforman-Sulem, L., Faure, C.: Extracting text lines in handwritten documents by perceptual grouping. In: Faure, C., Keuss, P., Lorette, G., Winter, A. (eds.) *Advances in Handwriting and Drawing: A Multidisciplinary Approach*, pp. 117–135. Europia, Paris (1994)
 - 13. Mello, C.A.B., Lins, R.D.: Image segmentation of historical documents. In: Visual2000, Mexico City, Mexico, September 2000, pp. 244–249 (2000)
 - 14. Nicolas, S., Paquet, T., Heutte, L.: Text line segmentation in handwritten document using a production system. In: IWFHR '04: Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR'04), pp. 245–250. IEEE Comput. Soc., Los Alamitos (2004)
 - 15. O'Gorman, L.: The document spectrum for page layout analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **15**(11), 1162–1173 (1993)
 - 16. Pavlidis, T., Zhou, J.: Page segmentation by white streams. In: Proc. 1st Int. Conf. Document Analysis and Recognition (ICDAR), Int. Assoc. Pattern Recognition, pp. 945–953 (1991)
 - 17. Peerawit, W., Kawtrakul, A.: Marginal noise removal from document images using edge density. In: Proc. Fourth Information and Computer Eng. Postgraduate Workshop (2004)
 - 18. Ping, Z., Lihui, C.: Document filters using morphological and geometrical features of characters. *Image Vis. Comput.* **19**(12), 847–855 (2001)
 - 19. Pu, Y., Shi, Z.: A natural learning algorithm based on Hough transform for text lines extraction in handwritten documents. In: Proceedings Sixth International Workshop on Frontiers of Handwriting Recognition, pp. 637–646 (1998)
 - 20. Sauvola, J., Pietikainen, M.: Adaptive document image binarization. *Pattern Recognit.* **33**(2), 225–236 (2000)
 - 21. Seth, S.C., Nagy, G., Stoddard, S.D.: Document analysis with expert system. In: Proceedings of Pattern Recognition in Practice II, June 1985 (1985)
 - 22. Shi, Z., Govindaraju, V.: Character image enhancement by selective region-growing. *Pattern Recognit. Lett.* **17**(5), 523–527 (1996)
 - 23. Shi, Z., Govindaraju, V.: Segmentation and recognition of connected handwritten numeral strings. *Pattern Recognit.* **30**(9), 1501–1504 (1997)
 - 24. Shi, Z., Govindaraju, V.: Historical document image enhancement using background light intensity normalization. In: 17th International Conference on Pattern Recognition, Cambridge, United Kingdom, 23–26 August 2004, pp. 23–26 (2004)
 - 25. Shi, Z., Govindaraju, V.: Line separation for complex document images using fuzzy runlength. In: DIAL '04: Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL'04), p. 306. IEEE Comput. Soc., Los Alamitos (2004)
 - 26. Shi, Z., Setlur, S., Govindaraju, V.: Text extraction from gray scale historical document images using adaptive local connectivity map. In: ICDAR '05: Proceedings of the Seventh International Conference on Document Analysis and Recognition, pp. 794–798. IEEE Comput. Soc., Los Alamitos (2005)
 - 27. Srihari, S.N., Govindaraju, V.: Analysis of textual images using the Hough transform. *Mach. Vis. Appl.* **2**, 141–153 (1989)
 - 28. Wang, Q., Tan, C.L.: Matching of double-sided document images to remove interference. In: IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, USA (2001)
 - 29. Wang, Q., Xia, T., Li, L., Tan, C.L.: Document image enhancement using directional wavelet. In: Proceedings of the 2003 IEEE Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA, June 2003, pp. 18–20 (2003)

Chapter 5

Segmentation of Ancient Arabic Documents

Abdel Belaïd and Nazih Ouwayed

Abstract This chapter addresses the problem of ancient Arabic document segmentation. As ancient documents have neither a real physical structure nor a logical one, the segmentation will be limited to textual areas or to line extraction in the areas. Although this type of segmentation appears quite simple, its implementation remains a challenging task. This is due to the state of many old documents; the image is of low quality, and the lines are not straight, but sinuous and connected. Given the failure of traditional methods, we proposed a method for line extraction in multi-oriented documents. The method is based on an image meshing that allows one to detect the orientations locally and safely. These orientations are then extended to larger areas. The orientation estimation uses the energy distribution of Cohen's class, which is more accurate than the projection method. Then, the method exploits the projection peaks to follow the connected components forming text lines. The approach ends with a final separation of connected lines, based on the exploitation of the morphology of terminal letters.

5.1 Introduction

Ancient handwriting is inherently complex because of its irregularity due to the manual aspect of the script. Rarely, did the writers use line support (or layers) to write, which resulted in sinuous lines of writing. Moreover, because of the calligraphic style of the writing, ligatures were easily introduced between the parts of words, and attachment occurred between the words of successive rows. Adding to this, as the document existed only on paper, updating was done directly on the text itself, which led either to extending the lines in the margins or adding entire blocks of lines in the margins.

All these artifacts complicate the problem of line segmentation which is essentially contextual in old documents, whereas most segmentation techniques of mod-

A. Belaïd (✉) · N. Ouwayed
LORIA, Campus Scientifique, 54500 Vandoeuvre, France
e-mail: abdel.belaïd@loria.fr

N. Ouwayed
e-mail: nazih.ouwayed@loria.fr



Fig. 5.1 Examples of four categories of handwritten ancient documents: (a) Mono-oriented, (b) multi-oriented, (c) multi-scripts, and (d) heterogeneous

ern documents are rather “natural,” seeking essentially parallel alignments of connected components. The problem of “contextual” segmentation is new and complex, and has been a challenge in research over the last decade.

The literature suggests many techniques for extracting lines, and some are more suitable than others. We will expose, in the first part of this chapter, a classification of these techniques. Several classification choices are possible, by the focus type, by the script type, or finally by the method procedure, bottom-up or top-down. The second section will be devoted to the segmentation of a class of ancient Arabic documents. A further difficulty arises when dealing with Arabic, corresponding mostly to the calligraphic aspect being more accentuated for Arabic (see Fig. 5.1, which shows different document classes with different kinds of orientation).

Given the failure of traditional methods, we proposed a method for line extraction in multi-oriented documents [33–38]. The technique has been studied for Arabic documents but can be generalized to any other script for which the writing is linear. The method starts by an image meshing allowing us to progressively and locally determine the orientations. The orientation is estimated using the energy distribution of Cohen’s class on the projection histogram profile. This local orientation is then enlarged to extract the orientation areas. Afterwards, the text lines are extracted locally in each area based on the connected components follow-up. Finally, the connected components that touch in adjacent lines are separated.

The chapter is organized as follows. In Sect. 5.2, we briefly describe the state of the art concerning the segmentation approaches. The multi-skew detection algorithm is detailed in Sect. 5.3. We present some experimental results in Sect. 5.4, and the conclusion, along with future trends of this work, will be given in Sect. 5.5.

5.2 Previous Work

The literature proposes many approaches for document line segmentation. Some of them are top-down, while others are bottom-up.

Top-down methods start from the whole image and iteratively subdivide it into smaller blocks to isolate the desired part. They use either a priori knowledge on the documents such as inter-line or inter-column spaces, or a document model to reach such a segmentation. The localization of white separations is generally done by analyzing the projection histogram profile, either by analyzing vertical stripes like in [3], by shredding the interline surface with local minima tracers like in [30], or finally by using a vector distance between histogram peaks and pixels, as in [1]. To face the inclination problem, one uses a Hough transform that considers the whole image composed of straight lines [44]. Given the failure of these global methods, other researchers are trying to use a knowledge model, such as the DMOS model proposed in [9] which consists of a grammatical formalism position to model the document structure, or the model of [51] corresponding to a vectorization-based algorithm, parametrized by some line features such as angle and length, etc. Nicolas, Paquet, and Heutte [31] propose an AI (artificial intelligence) problem solving framework using production systems.

Bottom-up methods deal with noise problems and writing variation. Most methods of line extraction in handwritten documents are bottom-up. The connected component-based methods are the mainstay of the bottom-up approaches. They are clustered into bigger elements such as words, lines, and blocks. In each research, simple rules are used in a different way. These rules are based on the geometric relationships between neighboring blocks, such as distance, overlap, and size compatibility. The difference between the different works lies in their capabilities to cope with space variation and influence of the script and writer peculiarities. Several approaches for clustering connected components have been proposed in the literature, such as K_NN, the Hough transform, smoothing, repulsive-attractive networks, the minimal spanning tree (MST), and deformable models.

Clustering methods related to the notion of mutual neighborhood have been considered in the clustering literature, as in [48] where the clustering is operated on different kinds of textual blocks extracted from vertical strips, in [22] where a perceptual grouping based on the “Gestalt theory” principles, such as proximity and similarity, is operated, or in [13] where the grouping is based on the text line alignments. The Hough transform is also used in bottom-up approaches. The main questions are related to the voting points, the most representative of the text lines. In [23], the voting points correspond to the center of gravity of connected components. In Pu and Shi [41], they correspond to the minima of the connected components, located in a vertical strip on the left side of the image. In [24], the voting points correspond to character blocks whose size is estimated from the average of the character sizes in the document.

The smoothing technique (run length smoothing or RLS) is to darken the small spaces between the consecutive black pixels in the horizontal direction, which leads to connecting them. The boxes which include the successive connected components in the image form the lines. In [45], a fuzzy run length algorithm is used. In [19], lines are extracted by applying an RLS algorithm (RLSA), adapted to a grayscale image. Instead of connecting a series of white and black pixels, the gradient of the image is expanded in the horizontal direction with a tilt angle that varies between $\pm 30^\circ$.

The repulsive attractive network is a dynamic system to minimize energy, which interacts with the textual image by attractive and repulsive forces defined on the network components and the document image [39]. Experimental results indicate that the network can successfully extract the baselines under significant noise in the presence of overlaps between the ascending and descending parts of characters in two lines.

Considering the connected components in a document as the vertices of a graph, we can obtain a complete undirected graph. A spanning tree of a connected graph is a tree that contains all the vertices of this graph. A minimal spanning tree (MST) of a graph is that spanning tree for which the sum of the edges is minimal among all the spanning trees of this graph. An MST of a graph can be generated with the Kruskal algorithm. In this algorithm, the tree is built by inserting the remaining unused edge with the smallest cost until all the vertices are connected [47].

The deformable model is an analytical approach which can act interactively on the modeling. It allows one to change (in time and space) the model representation of the model toward the solution of the minimization problem introduced in the modeling. Concretely, this leads to the introduction of a term of time evolution in the minimization criterion, which allows one, each time, to influence the prior model when necessary and to readjust to a better solution. Early work in this area includes that of Kass, Witkin, and Terzopoulos [16]. In the case of a two-dimensional image, the deformable contour model is used to find an existing object. The process is iterative. From an initial contour, a mechanism of deformable contour is applied to change this form so that it is the target area. The evolution mechanism is an energy function. The target area will be found by minimizing this energy. Several deformable contour models exist in the literature. A few examples are parametric active contour models (snake [16], the geometric snake [5], the level-set method [32, 40, 43], the B-spline or B-snake [20], and the Mumford–Shah model [42]).

Table 5.1 summarizes all the methods mentioned, divided according to 15 criteria: line types (straight, oriented, and cursive), material types (printed, handwritten, multi-oriented, interval orientation (IO), Latin, Chinese, Indian, Arabic, Persian, Urdu, image level (C: Color, G: Gray, and B: Binary) and mesh. All these approaches are either too general, proceeding by projection or by alignment search, or too local, operating by connected component following. They reach their limits when challenged by the poor quality and multi-orientation of documents. Most of these techniques have been applied to documents with a single orientation. The adaptation of these approaches is impossible if we want to extract all directions.

5.3 Overview of the Proposed System

Given the failure of traditional methods, we have proposed a method for line extraction in multi-oriented documents. The technique has been studied for Arabic documents but can be generalized to any other script in which the writing is linear. The method is based on an image meshing that allows it to detect the orientations

Table 5.1 Line segmentation bibliography

Class	Category	Authors	Line Type			Material Type	Multi-written oriented	IO Latin Chinese Indian Arabic Persian Urdu Level	Meshing
			Straight	Oriented	Cursive				
Top-Down	Projection based	Nicolaou and Gatos [29]	x	x		x	x		G, B
		Bennasri et al. [3]	x	x		x	x		B x
		Shapiro et al. [44]	x	x		x	x		B
		Antonacopoulos and Karatzas [1]	x		x	x	x		B
	Document Model	Nagy and Falsafi [28]	x		x	x	x		B x
		Lu et al. [25]	x	x	x	x	x		B x
	Bottom-Up	Likforman-Sulem and Faure [22]	x	x	x	x	x		G, B
		Zahour et al. [48]	x	x	x	x	x		B
		Zahour et al. [49]	x	x	x	x	x		B
		Zahour et al. [50]	x	x	x	x	x		B
	Feldbach and Tönnies [13]		x	x		x	x		G

Table 5.1 (Continued)

locally and safely. These orientations are then extended to larger areas. The only assumption is that initially the central part of the paper is horizontal. The orientation estimation uses the energy distribution of Cohen's class, which is more accurate than the projection method. Then, the method exploits the projection peaks to follow the connected components forming text lines. The approach ends with a final separation of connected lines, based on the exploitation of the morphology of terminal letters.

5.3.1 Image Meshing

In this step, the document image is partitioned into small meshes. The mesh size is generated, based on the idea that a mesh must approximately contain 3 lines, so as to produce a projection histogram profile that is representative of the writing orientation. To find the lines, the active contour model (or snake) is applied. The traditional external energy has some limitations such as the edge initialization near the contour and poor convergence to regions with concavities. For that reason, Xu et al. [46] developed a new kind of external energy that permits the snake to start far from the object, and forces it into boundary concavities. This energy is called gradient vector flow (GVF).

In our application, the major axis (equal to the first harmonic of the Fourier descriptor) of the connected components is used as the initial snake. We used the GVF as external energy and a null internal energy. To detect the alignments, some morphological operations such as dilation and erosion are first applied to the initial image (see Fig. 5.2b) to expand the edges. Then, the major axis of each connected component is determined using the Fourier descriptors [27] (see Fig. 5.2c). Finally, the energy minimization mechanism is operated on the snake to deform and push it to the text edge, more or less similarly to the connected component skeleton (see Fig. 5.2d). To ensure that the lines will be detected, we increment the size of the major axis by a threshold equal to a quarter of the average width of the connected components. This threshold is obtained by experiments. Finally, the connected components that belong to the same line are grouped to form the lines (see Fig. 5.2e). Figure 5.4b shows the results of the automatic meshing of the document presented in Fig. 5.4a.

In order to reduce the running speedup, we discard the meshes containing few pixels because their inclination is insignificant. If a mesh contains some text (i.e., few connected components) and thus no noise, it is automatically merged with the neighbor meshes.

5.3.2 Orientation Area Extraction

Orientation Estimation

As the lines are wavy, the orientation is first searched in small meshes where it is more likely to have fragments of straight lines. Traditionally, the projection his-

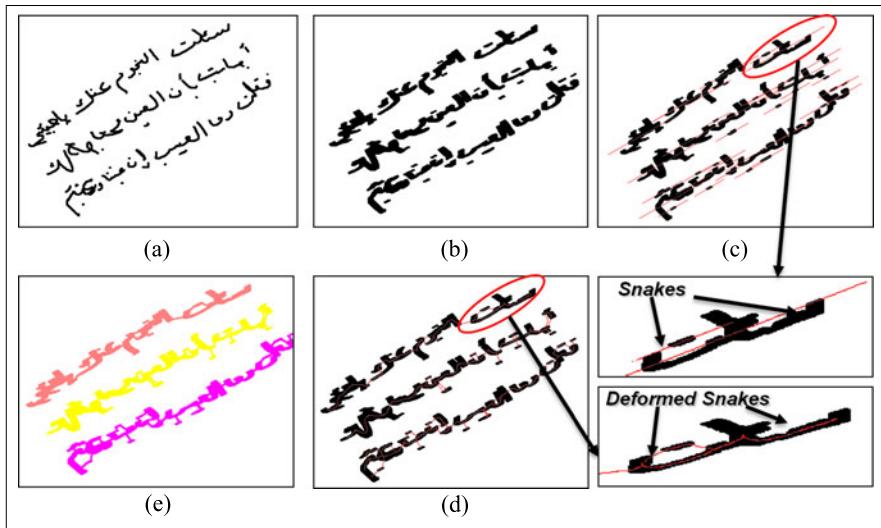


Fig. 5.2 Application of the snake for line detection: (a) initial image, (b) dilation and erosion of the image, (c) major axis drawn for each connected component of the lines. The *ellipse* encapsulates the initial connected component of (b). (d) shows the distorted snake of (c). (e) gives the final result showing the connected components gathered in each line

togram profile is employed along different orientation angles to determine the local orientation by the calculation of the difference between peaks and valleys. However, we observed that this technique fails for Arabic, in which individual parts of words (PAWs) can be oblique while the global word is horizontal. To face this problem, we have examined other features to better analyze the histogram function. We then used the energetic time–frequency distributions on the histogram as a signal.

Time–Frequency Distributions

To obtain a more robust estimator, we considered using a time–frequency representation of the histogram projection. This distribution best relates that projection to the peaks generated by the lines, translating their presence in high energy. It is less sensitive to false maxima. We used the Cohen’s class distributions which are quadratic and verify the invariance property by temporal or frequency translation. Each member of this class is distinguished by a kernel which has a determinant role in the quality of the provided images and in the properties it verifies. We limited it to the Wigner–Ville distribution (WVD), whose properties allow it to be more reactive to the presence of histogram peaks than the other distributions of the Cohen class [37].

The traditional approaches of signal processing such as the Fourier transform cannot study the signal variation over time and frequency. The energetic time–

frequency distributions go beyond what these approaches allow by analyzing the nonstationarity of a signal and distributing the signal energy in time and frequency.

According to [15], the energy E_x of a signal $x(t)$ is defined as

$$E_x = \int_{-\infty}^{+\infty} |x(t)|^2 dt = \int_{-\infty}^{+\infty} |\hat{x}(f)|^2 df, \quad (5.1)$$

where $\hat{x}(f)$ is the Fourier transform of the signal $x(t)$. The value E_x is quadratic. For this reason, the time–frequency distributions must keep this property.

Cohen’s Class In 1966, Cohen [2, 10, 12] proved that a significant number of time–frequency distributions can be seen as particular cases of the following general expression:

$$C_x(t, f) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \phi_{dD}(\tau, \xi) A_x(\tau, \xi) e^{j2\pi(t\xi - f\tau)} d\xi d\tau, \quad (5.2)$$

where $A_x(\tau, \xi)$ is the ambiguity function defined by

$$A_x(\tau, \xi) = \int_{-\infty}^{+\infty} x(t + \tau/2) x^*(t - \tau/2) e^{-j2\pi\xi t} dt.$$

Cohen’s class contains all the time–frequency distributions that are covariant under time and frequency shifts. The members of this class are identified by a particular kernel $\phi_{dD}(\tau, \xi)$ (expressed here in the delay-Doppler plane dD), which determines their theoretical properties [6–8, 14, 15] and their practical readability.

We want to use these distributions on the signal representing the histogram projection profile in each mesh in order to estimate its orientation. The Cohen’s class distributions are used to estimate the orientation because when computing the projection histogram of a document along one direction of projection, we obtain, if this direction is the real orientation of the document, a histogram in which each line leads to a clearly localized local maximum. Each block of the document leads in the projection histogram to a succession of periodic peaks and valleys, whose period is relatively constant. This periodic succession is delimited by the block size (“time” support) and oscillates at a frequency determined by the space width between the lines. As all the pixels are accumulated in the same positions, the local maxima have higher energy levels than with other projection directions. This explains why we can estimate the orientation of a document by seeking the projection angle for which the time–frequency distribution localizes a large energy level on a small area of the time–frequency plane. For example, Fig. 5.3 shows the increase of the maximum of the WVD when the number of peaks and valleys increases and when the valleys become wider.

To estimate the orientation angle, we use the analytic signal $x_a(t)$ of the centered squared root of the projection histogram $x(t)$ of the document. The analytic signal is the signal $x(t)$ without its negative frequencies. The histogram $x(t)$ is determined by projecting each document with a chosen orientation. To calculate all possible

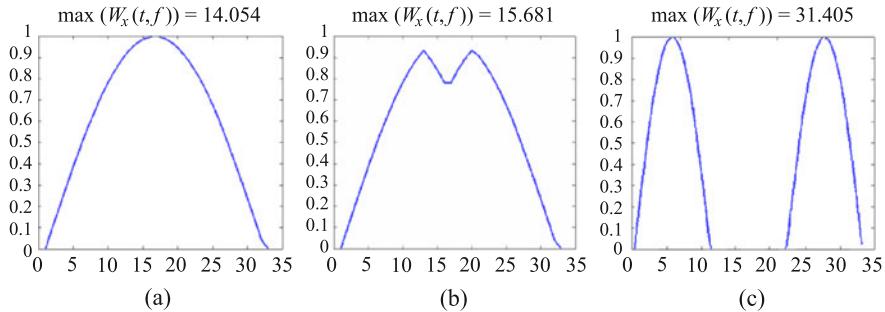


Fig. 5.3 Examples of maximum value of the Wigner–Ville distribution obtained for different projection histogram profiles when the number of peaks and valleys increases and when the valleys become wider

projection histograms, we turn the image around its center of gravity (which gives us a point deduced from the image content and not from its size and framing) and we choose the horizontal axis as an arbitrary reference for the zero degree angle. Then, we compute a time–frequency representation for the squared root of each projection histogram, whose average has been removed. The angle corresponding to the histogram with the highest maximum value of its time–frequency representation is chosen as the estimated angle of the document.

Kavallieratou et al. [17, 18] had already used the WVD to estimate the overall direction of Latin documents printed or handwritten. In this work, we first determined the properties of time–frequency representations that seemed desirable for such an application, then we established a list of performances, and then the performances were evaluated.

Orientation Area Extension

To extend the areas of orientation, we examine the orientations in neighboring meshes and proceed to an extension or a correction. Considering the writing direction in Arabic, we examine pairs of neighbors along three right-left directions: straight, sloping upward, and sloping down. The two neighbor meshes are merged if the orientation of the global mesh is equal to one of them; otherwise the orientations are maintained in both meshes. The operation is repeated for all the document meshes. After this step, the zones are constructed.

When a mesh contains several orientations, the mesh orientation will be erroneous. To detect this phenomenon, we observe the orientation of the horizontal (resp. vertical) surrounding meshes which have different angles. Since this case arises inside the main horizontal (resp. vertical) writing, the vertical (resp. horizontal) projection profile is used to resolve this case. We look for the first minimum value in the projection profile from the right representing the end of the first inclination (I_m minimum index). Then the mesh is divided at I_m into two meshes.

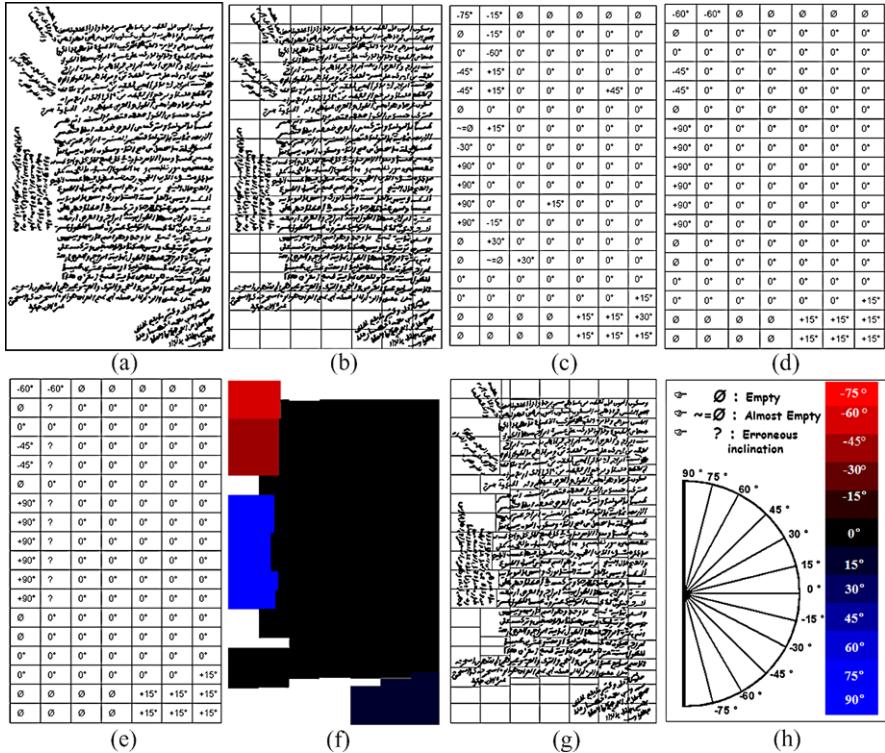


Fig. 5.4 The results for the different steps of the multi-skew detection approach

Since they are applied automatically, the initial paving edges can cross the connected components, creating problems (false maxima) in text line detection. The incorrect paving exists only in the horizontal and the vertical zones. We need to correct the position of these edges by performing a horizontal or vertical shift so that the local paving covers the local connected components. In the horizontal (resp. vertical) area, the edge that divides two consecutive rows (resp. columns) is moved to the nearest position in these rows (resp. columns) when the horizontal (resp. vertical) projection vector for each of their two consecutive meshes has a minimum value (see Fig. 5.4g and see Fig. 5.4f).

5.3.3 Text Line Extraction

The text line follow-up starts in the first window on the right side of the page. The algorithm starts by looking for the new maxima (see Fig. 5.5a). Each peak represents the starting point P_s of the orientation line bl_j . The ending point P_e of the orientation line is calculated using the P_s , the orientation, the width, and the height

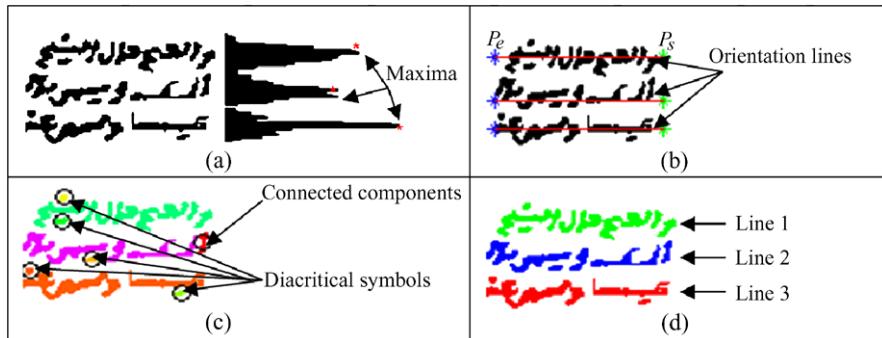


Fig. 5.5 Text line detection steps for a window. (a) Maxima detection, (b) orientation lines estimation, (c) assignment of each connected component and diacritical symbol to its appropriate line, (d) extracted lines

Table 5.2 Four types of connection observed in Arabic handwritten documents

Type	Area of connexion	Example
a		
b		
c		
d		

of each window (see Fig. 5.5b). The orientation line bl_j is calculated based on the two points (P_s , P_e) and the orientation of the window. The connected components that belong to a baseline are sought to construct the text line (see Fig. 5.5c).

A step of text line correction follows the text line detection to assign the non-detected components and the diacritical symbols to the appropriate text line (see Figs. 5.5c and d). A distance method is used to address this problem. First, the distance between the centroid of the nondetected component or diacritical symbol C_i and the text line is calculated. C_i is assigned to the text line l_j if $d_{ci,lj} < d_{ci,lj+1}$ else to $l_j + 1$.

5.3.4 Connected Line Separation

The connections occur between two successive lines when their characters touch. Often, these connections are made between ascenders in the lower line and descenders in the higher line. Table 5.2 lists the four categories of connection in Arabic: (a) a descender with right loop, connects a vertical ascender, (b) a left descender with a loop touches a vertical ascender, (c) a right descender touches the higher part of the loop of a character, and (d) a left descender connects the higher part of the lower curve of a letter.

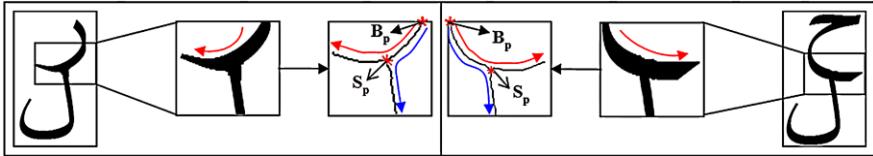


Fig. 5.6 Connection areas and direction of descenders (the right direction indicated by the red arrow, the erroneous direction by the blue arrow)

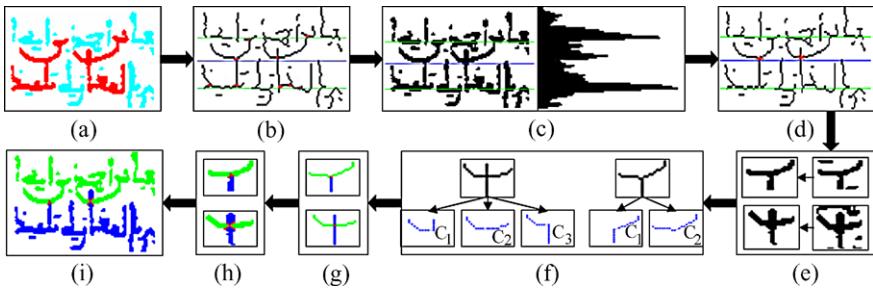


Fig. 5.7 Different steps of the separation of connected components

In all connection cases, we note the presence of a descender connecting a lower end letter. The descenders are grouped into two categories: (a, c) where the descender of the line starts from the right, and (b, d) where the descender of the line starts from the left. To streamline the work, the analysis focuses on the connection areas (see Fig. 5.6).

The method starts by extracting, in the two lines, the connected component created by the connection between the two successive lines (see Fig. 5.7a). Then, the intersection points of each connected component are detected (see Fig. 5.7b, the points are in red). An intersection point is a pixel that has at least three neighboring pixels. In the case chosen, the connection occurs at a single point of intersection S_p close to the minimum axis (valley between two lines, see Fig. 5.7c). Thus, the point S_p is the nearest point of the minimum axis (see Fig. 5.7d). We then look for the starting point of the ligature, B_p , which is generally the highest point, near the baseline of the top line. Then, from this point, the method is to follow the descending character (i.e., its skeleton, see Fig. 5.7f). The following continues beyond the intersection point respecting an angular variation corresponding to the curvature of the descending character.

Due to the symmetry of the curve branches, the value of the orientation angle must always be positive. For example, in Fig. 5.8, the angular variances are $\text{Var}(C_{1+2}) = 703,19$, $\text{Var}(C_{1+3}) = 299$, $\text{Var}(C_{1+4}) = 572,37$. In this example, the minimum angular variance $\text{Var}(C_{1+3})$ is given by the correct direction to follow. Figure 5.9 illustrates the effectiveness of the algorithm on a representative sample of 12 arbitrarily chosen connected components from 640 occurrences found in 100 documents.

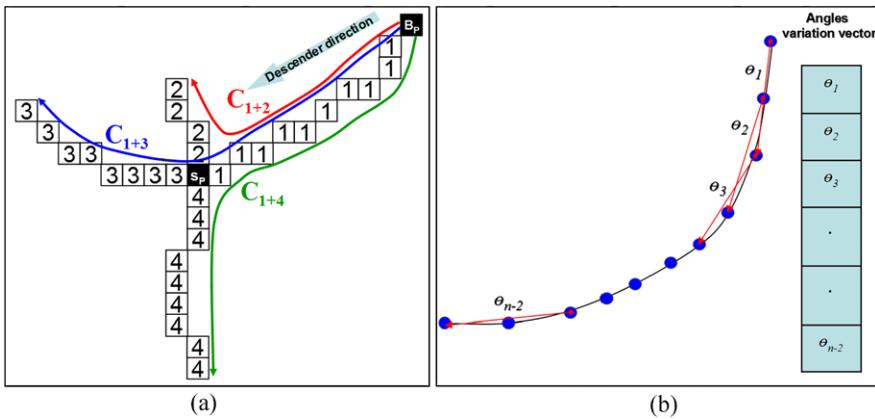


Fig. 5.8 (a) Example of Arabic connected components (the Arabic letter “ra” is connected with the letter “alif”), (b) estimation algorithm of the angular variation

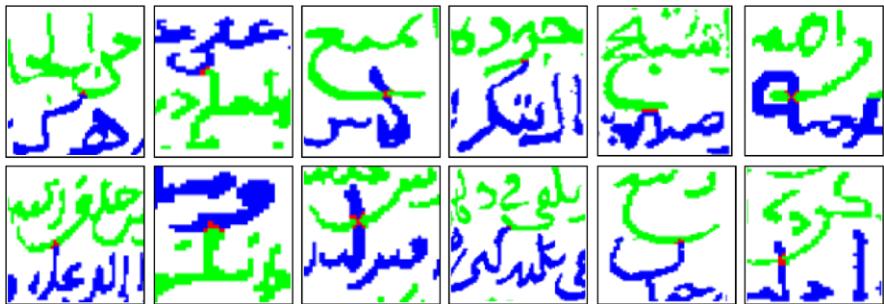


Fig. 5.9 Results of some connected line separation

5.4 Experiments and Discussion

To study the effectiveness of our approach, we have experimented on 100 Arabic ancient documents containing 2,500 lines. These documents belong to a database obtained from web sites of the Tunisian National Library, the National Library of Medicine in the USA, and the National Library and Archives of Egypt. The tests were prepared after a manual area and line labeling step of each document. The rotation angle examined during these experiments ranged from -75° to $+90^\circ$. The execution time is measured from the meshing phase until the line separation phase, and it depends on the document and the mesh sizes. The tests were performed on a PC with a Pentium M 1.4 GHz and a cache of 1 GB in Windows XP. The application was developed with MATLAB completed by the time-frequency toolbox *tftb* [2].

The approach is composed of two main steps: multi-oriented area detection and text line extraction. Our results are measured according to these algorithms.

The multi-oriented algorithm is composed of three main steps: image meshing, orientation estimation, and orientation extension and paving correction. A global

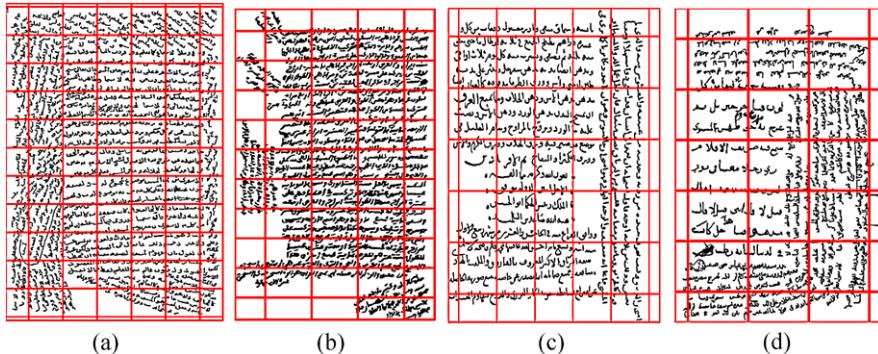


Fig. 5.10 Meshing results of four different documents

accuracy rate of 97 % is reached. The 3 % error is shared by the three stages of treatment: 1 % is due to the image paving, 1.3 % is due to the orientation estimation, and 0.7 % is due to the orientation extension and paving correction.

In image meshing, three text lines are needed to obtain a projection profile representing the orientation in each mesh. So, if this criterion is not obtained by the paving algorithm, some errors may occur in the area detection. The error rate of 1 % is divided into two cases: 0.7 % is due to the adjacent line connection and 0.3 % is due to the small oriented areas. In the first case, the connection between lines is very frequent in ancient Arabic documents. When the active contour model (snake) is applied in a mesh to extract alignments, it is possible that it will connect two adjacent lines. This will increase the alignment height and consequently the mesh height. A large mesh may include different oriented areas. In the second case, the oriented areas are composed of a few small lines. These areas can be gathered by the paving in other meshes and naturally will not be extracted. Figure 5.10 shows the image meshing results of four different documents. We note in these documents the presence of at least three lines in each mesh.

The meshes in our documents have, in some cases, more than one orientation or cursive lines. In these two cases, the orientation estimation is wrong and will be wrong for the orientation extension and consequently for the area detection. The error rate of 0.9 % is due to the meshes with multi-orientation. The 0.4 % error is due to the Arabic curvature lines in Arabic ancient documents. Figure 5.11 shows the results of the first orientation estimation of four documents selected in our database. Each color represent an orientation (see Fig. 5.4 for the color legend). We notice in these documents the presence of meshes with erroneous orientation (multi-orientations or cursive lines (gray color)).

Four extension rules are applied for mesh extension having the same orientation. In the extension phase, any error is happening because all possible orientations in the documents are considered. The error rate of 0.7 % is due to the paving correction. As this paving is rectangular, the correction can be applied just along the horizontal and vertical directions. In some cases (oblique areas), the paving correction cannot be applied, which will yield some segmentation errors. Figure 5.12 shows the results

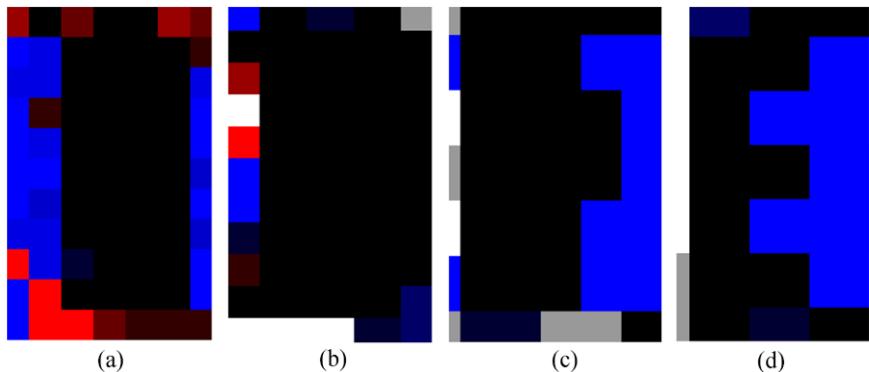


Fig. 5.11 Results of the first orientation estimation of the four selected documents

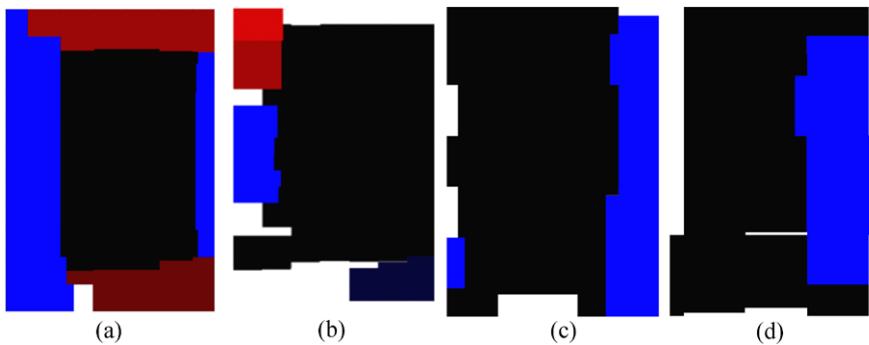


Fig. 5.12 Results of the multi-oriented areas of the four selected documents

Table 5.3 Results of the multi-skew estimation for the four documents

Figure	Document size	Resolution (dpi)	$w \times h$ of paving (pixels)	Execution time	Zone number	
					True	Detected
First document	572 × 800	72	75 × 75	35 s	5	5
Second document	410 × 625	72	75 × 75	30 s	5	5
Third document	750 × 941	72	120 × 120	34 s	2	2
Fourth document	362 × 500	72	90 × 90	30 s	2	2

of the multi-oriented area extraction of the four selected documents. Each area is visualized by a color. In these documents, all the multi-oriented areas are extracted correctly.

Table 5.3 summarizes the results of the four representative documents chosen arbitrarily from the 100 documents selected. These results show the effectiveness and the performance of the multi-oriented area detection algorithm.

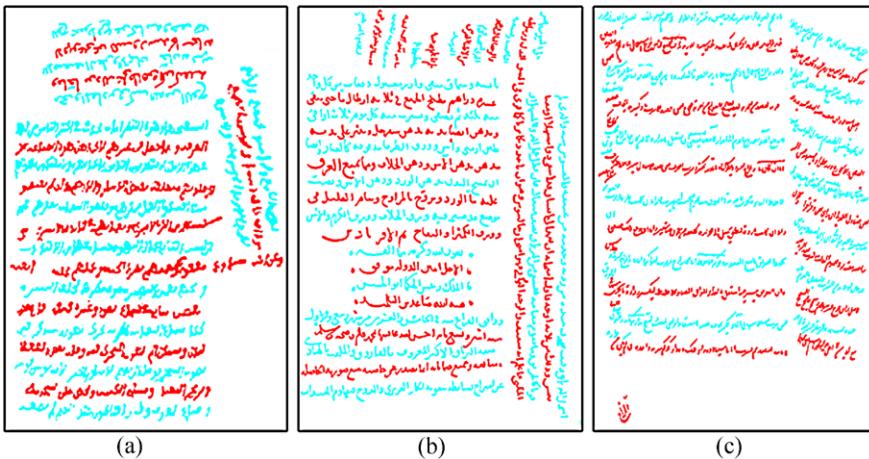


Fig. 5.13 Result examples of the text line extraction

For line segmentation, the extraction rate reaches 98.6 %. The 0.9 % of non-detected lines is due to the detection area algorithm. The error rate of 0.5 % is due to the presence of diacritical symbols in the beginning of lines that create false maxima. Figure 5.13 illustrates the effectiveness of our algorithm on a sample of three documents chosen randomly among the 100 documents processed. To identify the lines, each pair of consecutive lines is presented in two different colors.

5.5 Conclusion

A multi-oriented text line extraction approach is proposed in this chapter based on the local orientation estimation. To extract the lines, the approach first performs an image paving of the document. Then, the orientation in each mesh is estimated, extended, and corrected. Finally, the text lines are extracted and separated.

The mesh size is estimated using the active contour model (snake) approach. This size is fixed once three lines in the mesh are extracted. The skew detection approach uses the Cohen's class distributions applied on the projection histogram profile in each mesh and considered as a signal. The Wigner–Ville distribution (WVD) from this class is retained for our application due to its interesting properties adapted to the properties of our signals. The mesh area is extended to similar oriented meshes to obtain the largest orientation areas using four rules. These rules depend on the orientations presented in the documents. The text lines are extracted in each mesh using a follow-up connected components algorithm. The lines are separated based on the analysis of the terminal Arabic letters.

Experimental results on various types of handwritten Arabic documents show that the proposed method has achieved a promising performance for text line extraction. This approach will be generalized to other document types (Latin, Urdu, Farsi, etc.) and to heterogeneous documents with text and images.

References

1. Antonacopoulos, A., Karatzas, D.: Document image analysis for World War II personal records. In: International Workshop on Document Image Analysis for Libraries, pp. 336–343 (2004)
2. Auger, F., Doncarli, C.: Quelques commentaires sur des représentations temps-fréquence proposées récemment. *Trait. Signal* **9**(1), 3–25 (1992)
3. Bennasri, A., Zahour, A., Taconet, B.: Extraction des lignes d'un texte manuscrit Arabe. In: Vision Interface'99, pp. 42–48 (1999)
4. Bukhari, S.S., Shafait, F., Breuel, T.M.: Segmentation of curled textlines using active contours. In: The Eighth IAPR Workshop on Document Analysis Systems (DAS 2008), pp. 270–277 (2008)
5. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. In: International Conference on Computer Vision, pp. 694–699 (1995)
6. Classen, T.A.C.M., Mecklenbrauker, W.F.G.: The Wigner distribution—a tool for time frequency analysis, Part I. *Philips J. Res.* **35**(3), 217–250 (1980)
7. Classen, T.A.C.M., Mecklenbrauker, W.F.G.: The Wigner distribution—a tool for time frequency analysis, Part II. *Philips J. Res.* **35**(4/5), 372–389 (1980)
8. Classen, T.A.C.M., Mecklenbrauker, W.F.G.: The Wigner distribution—a tool for time frequency analysis, Part III. *Philips J. Res.* **35**(6), 372–389 (1980)
9. Coasnon, B., Camillerapp, J.: DMOS, une méthode grique de reconnaissance de documents: valuation sur 60 000 formulaires du XIXe siècle. In: Actes du Colloque International Franco-phonique sur l'Crit et le Document (CIFED'02), Hammamet (2002)
10. Cohen, L.: Generalized phase-space distribution functions. *J. Math. Phys.* **7**(5), 781–786 (1966)
11. Du, X., Pan, W., Bui, T.D.: Text line segmentation in handwritten documents using Mumford–Shah model. *Pattern Recognit.* **42**(12), 3136–3145 (2009)
12. Escudé, B., Gréa, J.: Sur une formulation générale de la représentation en temps et en fréquence dans l'analyse des signaux d'énergie finie. *C. R. Acad. Sci. Paris* **283**, 1049–1051 (1976)
13. Feldbach, M., Tönnies, K.D.: Line detection and segmentation in historical church registers. In: ICDAR'01: Proceedings of the Sixth International Conference on Document Analysis and Recognition, pp. 743–748 (2001)
14. Flandrin, P.: Time–Frequency/Time-Scale Analysis. Academic Press, San Diego (1999)
15. Hlawatsch, F., Boudreax-Bartels, G.F.: Linear and quadratic time–frequency signal representation. *IEEE Signal Process. Mag.* **9**(2), 21–67 (1992)
16. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: active contour models. In: Proceedings of 1st ICCV, pp. 259–268 (1987)
17. Kavallieratou, E., Fakotakis, N., Kokkinakis, G.: Skew angle estimation in document processing using Cohen's class distributions. *Pattern Recognit. Lett.* **20**, 11–13 (1999)
18. Kavallieratou, E., Fakotakis, N., Kokkinakis, G.: Skew angle estimation for printed and handwritten documents using the Wigner–Ville distribution. *Image Vis. Comput.* **20**, 813–824 (2002)
19. Le Bourgeois, F., Emptoz, H., Trinh, E., Duong, J.: Networking digital document images. In: 6th International Conference on Document Analysis and Recognition, Seattle (2001)
20. Leitner, F., Cinquin, P.: From splines and snakes to SNAKE SPLINES. In: Selected Papers from the Workshop on Geometric Reasoning for Perception and Action, pp. 264–281. Springer, Berlin (1993)
21. Li, Y., Zheng, Y., Doermann, D., Jaeger, S.: Script-independent text line segmentation in freestyle handwritten documents. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(8), 1313–1329 (2008)
22. Likforman-Sulem, L., Faure, C.: Extracting lines on handwritten documents by perceptual grouping. In: Faure, C., Keuss, P., Lorette, G., Winter, A. (eds.) Advances in Handwriting and Drawing: Multidisciplinary Approach, pp. 21–38 (1994)

23. Likforman-Sulem, L., Hanimyan, A., Faure, C.: A Hough based algorithm for extracting text lines in handwritten document. In: Proc. of ICDAR'95, pp. 774–777 (1995)
24. Louloudis, G., Gatos, B., Pratikakis, I., Halatsis, C.: Text line and word segmentation of handwritten documents. *Pattern Recognit.* **42**(12), 3169–3183 (2009)
25. Lu, Y., Wang, Z., Tan, C.L.: Word grouping in document images based on voronoi tessellation. In: *Document Analysis Systems*, pp. 147–157 (2004)
26. Malleron, V., Eglin, V., Emptoz, H., Dord-Crouslé, S., Régnier, P.: Text lines and snippets extraction for 19th century handwriting documents layout analysis. In: Proceedings of the Tenth International Conference on Document Analysis and Recognition (ICDAR), pp. 1001–1005 (2009)
27. Montagnat, J., Delingette, H., Ayache, N.: A review of deformable surfaces: topology, geometry and deformation. *Image Vis. Comput.* **19**(14), 1023–1040 (2001)
28. Nagy, G., Falsafi, A.: Using vanishing points to locate objects with six degrees of freedom. In: Gelsema, E.S., Kanal, L. (eds.) *Pattern Recognition and Artificial Intelligence*, pp. 123–139. North-Holland, Amsterdam (1988)
29. Nicolaou, A., Gatos, B.: Handwritten text line segmentation by shredding text into its lines. In: *International Conference on Document Analysis and Recognition*, pp. 626–630 (2009)
30. Nicolas, S., Paquet, T., Heutte, L.: Text line segmentation in handwritten document using a production system. In: *International Workshop on Frontiers in Handwriting Recognition*, pp. 245–250 (2004)
31. Nicolas, S., Paquet, T., Heutte, L.: Text line segmentation in handwritten document using a production system. In: *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9)* (2004)
32. Osher, S., Paragios, N.: *Geometric Level Set Methods in Imaging, Vision, and Graphics*. Springer, New York (2003)
33. Ouwayed, N.: Segmentation en lignes de documents anciens: application aux documents Arabes. These de doctorat. Université Nancy 2 (2010)
34. Ouwayed, N., Belaid, A.: Multi-oriented text line extraction from handwritten Arabic documents. In: *International Workshop on Document Analysis Systems*. IAPR, Nara (2008)
35. Ouwayed, N., Belaid, A.: Separation of overlapping and touching lines within handwritten Arabic documents. In: *13th International Conference on Computer Analysis of Images and Patterns (CAIP'2009)*, pp. 237–244. IEEE Press, New York (2009)
36. Ouwayed, N., Belaid, A.: Une approche générale pour l'extraction des lignes des documents Arabes anciens multi-orientés. In: *12e Colloque International sur le Document Electronique (CIDE.12)* (2009)
37. Ouwayed, N., Belaid, A., Auger, F.: Estimation de l'inclinaison d'un document Arabe manuscrit numérisé par analyse temps-fréquence des histogrammes de projection. *Trait. Signal* **26**(4), 307–319 (2009)
38. Ouwayed, N., Belaid, A., Auger, F.: General text line extraction approach based on locally orientation estimation. In: *Document Recognition and Retrieval XVII*, San Jose, California (2010)
39. Oztop, E., Mulayim, A.Y., Atalay, V., Yarman Vural, F.: Repulsive attractive network for baseline extraction on document images. *Signal Process.* **75**, 1–10 (1999)
40. Pluemptiwiriyawej, C., Moura, J.M.F., Wu, Y.J.L., Ho, C.: STACS: new active contour scheme for cardiac MR image segmentation. *IEEE Trans. Med. Imaging* **24**(5), 593–603 (2005)
41. Pu, Y., Shi, Z.: A natural learning algorithm based on hough transform for text lines extraction in handwritten documents. In: *Proceedings of the 6th International Workshop on Frontiers in Handwriting Recognition*, Taejon, Korea, pp. 637–646 (1998)
42. Ramlau, R., Ring, W.: A Mumford-Shah level-set approach for the inversion and segmentation of X-ray tomography data. *J. Comput. Phys.* **221**(2), 539–557 (2007)
43. Sethian, J.A.: Curvature and the evolution of fronts. *Commun. Math. Phys.* **101**(4), 487–499 (1985)

44. Shapiro, V., Gluchev, G., Sgurev, V.: Handwritten document image segmentation and analysis. *Pattern Recognit. Lett.* **14**(1), 71–78 (1993)
45. Shi, Z., Govindaraju, V.: Line separation for complex document images using fuzzy run length. In: International Workshop on Document Image Analysis for Libraries, pp. 306–313 (2004)
46. Xu, C., Prince, J.L.: Gradient vector flow: a new external force for snakes. In: Proc. IEEE Conf. on Comp. Vis. Patt. Recog. (CVPR), pp. 66–71 (1997)
47. Yin, F., Liu, C.-L.: Handwritten text line segmentation by clustering with distance metric learning. In: Proc. 11th ICFHR, pp. 229–234 (2008)
48. Zahour, A., Taconet, B., Ramdane, S.: Contribution la segmentation de textes manuscrits anciens. In: Conference Internationale Francophone sur l'Ecrit et le Document, CIFED'04 (2004)
49. Zahour, A., Likforman-Sulem, L., Boussellaa, W., Taconet, B.: Text line segmentation of historical arabic documents. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 138–142 (2007)
50. Zahour, A., Taconet, B., Likforman-Sulem, L., Boussellaa, W.: Overlapping and multi-touching text-line segmentation by Block Covering analysis. *Pattern Anal. Appl.* **12**(4), 335–351 (2009)
51. Zheng, Y., Li, H., Doermann, D.: A model-based line detection algorithm documents. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR) (2003)

Chapter 6

Features for HMM-Based Arabic Handwritten Word Recognition Systems

**Laurence Likforman-Sulem, Ramy Al Hajj Mohammad, Chafic Mokbel,
Fares Menasri, Anne-Laure Bianne-Bernard, and Christopher Kermorvant**

Abstract HMM-based systems need observation sequences as input. These observations consist of discrete values or vectors extracted from word images or text lines. In this chapter we explore various types of features which are popular for Arabic cursive handwriting recognition. Some of these features are statistical, based on pixel distributions or local directions. Others are structural, based on the presence of loops, ascenders, or descenders. We show how these features can be efficient within HMM-based systems based on sliding windows or grapheme segmentation.

6.1 Introduction

The recognition of Arabic writing has many applications such as mail sorting, bank check reading, and modern and historical handwritten document recognition. Arabic and Latin handwriting share similarities (see Fig. 6.1): the core of the hand-

L. Likforman-Sulem (✉) · A.-L. Bianne-Bernard
Telecom ParisTech, CNRS UMR 5141, Paris, France
e-mail: likforman@telecom-paristech.fr

R. Al Hajj Mohammad
Lebanese International University, Beqaa Valley, Lebanon
e-mail: rami.elhajj@liu.edu.lb

C. Mokbel
UOB, El-Koura, Lebanon
e-mail: chafic.mokbel@balamand.edu.lb

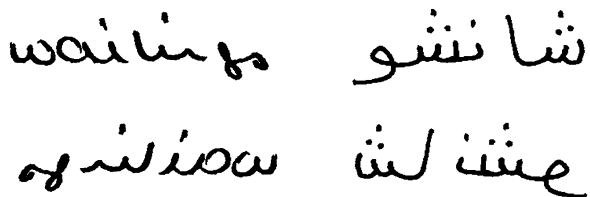
F. Menasri · A.-L. Bianne-Bernard · C. Kermorvant
A2iA, Paris, France

F. Menasri
e-mail: fm@a2ia.com

A.-L. Bianne-Bernard
e-mail: alb@a2ia.com

C. Kermorvant
e-mail: ck@a2ia.com

Fig. 6.1 A flipped Latin word shows similarities with an Arabic word and vice versa



writing lies between two baselines; ascenders and descenders lie above and under these baselines, respectively. However; Arabic writing has specificities which make it highly challenging for off-line recognition systems [2]. The handwriting is very round-shaped which makes it difficult to deslant. It also includes various small-sized marks which modify the meaning of letters. Moreover, there is a large number of graphical forms (ligatures) which group several characters [3]. Last, the number of Arabic words is potentially very large, since many distinct words can be formed from word roots, with prefixes and suffixes [15].

Different approaches have been proposed for recognizing isolated words. Using the holistic approach, words are modeled as a whole without segmenting them into smaller units. A feature vector is extracted directly from either an image, the skeleton, or the contour of a word's image [17]. This feature vector may include statistical and/or structural features. Hidden Markov model (HMM)-based holistic approaches extract a sequence of feature vectors instead of a single feature vector along the word image [10]. However, these frames are not associated to word subunits. The holistic approach is convenient when the size of the lexicon is small. Under the analytical approach, word models result from the concatenation of character models. The analytical strategy is convenient for enlarging a vocabulary with new words, since new vocabulary words can be described through their compound letters, without providing their images.

The HMM-based approach is very convenient for implementing the analytical approach. Analytical systems may or may not pre-segment words into smaller units such as characters or graphemes (subparts of characters). In both cases, HMM systems extract a feature vector from each pre-segmented unit or, by sliding a window on the word image [21, 23], obtain a sequence of feature vectors (or frames). These systems can be easily applied on both Latin and Arabic words and achieve state-of-the-art performance [18, 25].

In the following, we focus on the feature extraction step within HMM systems. We survey the features in use within such systems and present our results obtained using three types of HMM-based systems: a context-independent sliding-window system, a context-dependent system [1, 6], and a hybrid HMM/neural network system which segments words into graphemes.

6.2 Features for Sliding-Window Systems

The principle of sliding-window systems is based on extracting a sequence of observed feature vectors by moving (sliding) a window from right to left over an im-

Fig. 6.2 Sliding windows decomposed into 20 cells

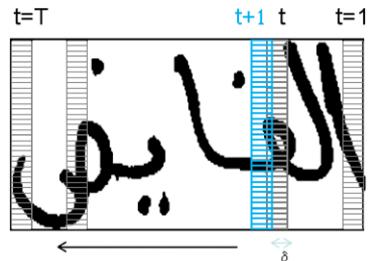
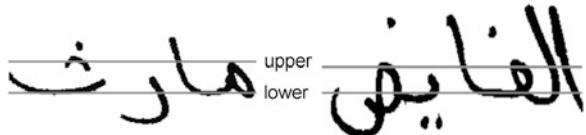


Fig. 6.3 Upper and lower baseline extraction



age word. The window starts at the extreme right of the word (at position or “time” $t = 1$), and is regularly shifted by δ until $t = T$, as shown in Fig. 6.2. At each position, a feature vector (or frame) is extracted, capturing writing shape and local pixel distribution. Each frame is divided into a fixed number of cells in order to cope with different word heights. HMM decoding will consist of associating consecutive frames to states. Sections 6.2.2 and 6.2.3 describe the features we use in our context-independent and context-dependent systems for Arabic recognition. These features are enhanced by deriving from the initial features a set of dynamic ones (see Sect. 6.2.4). The features’ values depend on the position of the word’s baselines which are extracted as an initial step (see Sect. 6.2.1). We also survey other popular features for sliding-window systems in Sect. 6.2.5.

6.2.1 Baseline Extraction

Like Latin, Arabic handwriting uses two baselines: an upper baseline and a lower one. Children at school learn how to write between one or two preprinted ruled lines. Such baselines define three zones within a word: the core zone, the upper zone where ascenders can be found, and the lower zone for descenders. Many systems thus start by finding these main lines in order to extract baseline-dependent features. Such features indicate the presence of ascenders, descenders, or both within a frame. The quality of the feature extraction step depends on that of the baseline extraction step. This step is therefore significant, and several variants have been proposed: extraction from whole lines or from connected components using projection profiles, or characteristic points fitted through a line. Figure 6.3 shows the result of a baseline extraction step. The approach relies on the algorithm described in [7] with few alterations. It is based on the vertical projection profile obtained by computing the sum of pixel values along the horizontal axis for each y word image value. First, the

peak corresponding to the maximum of the projection profile curve is determined: the position of the maximum identifies the lower baseline. It is justified by the fact that, in Arabic handwriting, most letters have many pixels on the lower baseline. Then, the algorithm scans the image from top to bottom to find the upper baseline. The position of the upper baseline corresponds to the position of the first line with a projection value greater than the average row density. Inaccurate baselines may be found in the case of very short words because of the great influence of diacritical points. We note that in the IFN/ENIT database most of the words are horizontal: the average orientation of ground truth baselines is only about 1.36 %.

6.2.2 Distribution Features

The set of distribution features consists of 16 features that characterize the density of foreground pixels within frames and frame cells. Let H be the height of the frame in an image, h the variable height of a cell, w the width of a frame, and n_c the number of cells in a frame. Feature f_1 is the density of foreground pixels within the frame. Feature f_2 is the number of black/white transitions between two consecutive frame cells:

$$f_2 = \sum_{i=2}^{n_c} |b(i) - b(i-1)| \quad (6.1)$$

where $b(i)$ is the density level of cell i ; $b(i)$ is equal to one if the cell contains at least one foreground pixel and is equal to zero otherwise. Feature f_3 is a derivative feature defined as the difference between the y -coordinate g of the center of gravity of foreground pixels of two consecutive frames t and $t-1$. g is given by

$$g = \frac{\sum_{j=1}^H j \cdot r(j)}{\sum_{j=1}^H r(j)} \quad (6.2)$$

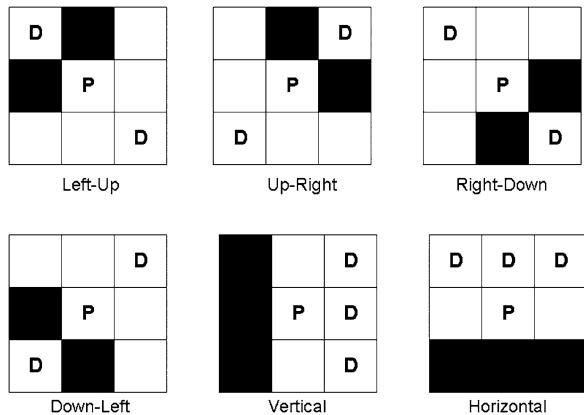
where $r(j)$ is the number of foreground pixels in the j th row of a frame. The eight features f_4 to f_{11} represent the densities of black (foreground) pixels for each vertical column of pixels in each frame (in our case the width of the frame is 8 pixels). Let L be the position of the lower baseline. Feature f_{12} is the vertical distance from the lower baseline to the center of gravity of foreground pixels, normalized by the height of the frame:

$$f_{12} = \frac{g - L}{H} \quad (6.3)$$

Feature f_{13} (resp. f_{14}) represents the density of foreground pixels over (resp. under) the lower baseline:

$$f_{13} = \frac{\sum_{j=L+1}^H r(j)}{H \cdot w}, \quad f_{14} = \frac{\sum_{j=1}^{L-1} r(j)}{H \cdot w} \quad (6.4)$$

Fig. 6.4 Masks for computing concavity features



Feature f_{15} is the number of transitions between two consecutive cells of different density levels above the lower baseline:

$$f_{15} = \sum_{i=k}^{n_c} |b(i) - b(i-1)| \quad (6.5)$$

where k is the cell that contains the lower baseline.

Feature f_{16} represents the zone to which the gravity center of black pixels belongs, with respect to the upper and lower baselines. Actually, the two baselines divide a frame into three zones: above the upper baseline ($f_{16} = 1$), a core zone ($f_{16} = 2$), and below the lower baseline ($f_{16} = 3$).

6.2.3 Concavity Features

Concavity features provide local concavity information and stroke directions within each frame. Each of the concavity features f_{17} to f_{22} represents the (normalized) number of white pixels (background) that belong to one of six types of concavity configurations. These features are explored by using a 3×3 window (mask) as shown in Fig. 6.4.

The concavity features are calculated as follows: Let N_{lu} (resp. N_{ur} , N_{rd} , N_{dl} , N_v , and N_h) be the number of background pixels that have neighboring black pixels in the following directions: left and up (resp. up-right, right-down, down-left, vertical, and horizontal). In our implementation, image border pixels are excluded. Thus, the six normalized concavity features are defined as

$$\begin{aligned} f_{17} &= \frac{N_{lu}}{H}, & f_{18} &= \frac{N_{ur}}{H}, & f_{19} &= \frac{N_{rd}}{H} \\ f_{20} &= \frac{N_{dl}}{H}, & f_{21} &= \frac{N_v}{H}, & f_{22} &= \frac{N_h}{H} \end{aligned} \quad (6.6)$$

By using the information coming from the detection of the two baselines (upper and lower), we generate six new features f_{23} – f_{28} , describing the concavities in the core zone of a word, that is, the zone bounded by the two upper and lower baselines. Let CNZ_{lu} (resp. CNZ_{ur} , CNZ_{rd} , CNZ_{dl} , CNZ_v , and CNZ_h) be the number of background pixels in the core zone that have neighboring black pixels in the configuration left-up (resp. up-right, right-down, down-left, vertical, and horizontal).

Thus, the six additional and baseline-dependent concavity features related to the core zone are defined as

$$\begin{aligned} f_{23} &= \frac{CNZ_{lu}}{d}, & f_{24} &= \frac{CNZ_{ur}}{d}, & f_{25} &= \frac{CNZ_{rd}}{d} \\ f_{26} &= \frac{CNZ_{dl}}{d}, & f_{27} &= \frac{CNZ_v}{d}, & f_{28} &= \frac{CNZ_h}{d} \end{aligned} \quad (6.7)$$

where d is the distance between the two baselines (upper and lower). This results in a 28-feature vector per frame; 17 of them are baseline independent (f_1 – f_{11} , f_{17} – f_{22}), whereas the 11 remaining ones are calculated with respect to baseline positions. Actually, those features are convenient to any script that can be decomposed into three zones (core, ascending, and descending zones), such as the Latin cursive script.

6.2.4 Dynamic Features

We introduce context at the feature extraction level through derivative features. Such features represent the dynamics of features around the current window. The feature vector at horizontal pixel position p contains information on the frame at position p but also on the context of this frame from windows at positions $p - \delta * K$ to $p + \delta * K$ (δ is the shift of the window and K is the number of windows participating to the derivative feature). The derivation is computed with a regression. In the speech recognition domain the first and second order regressions are known as delta and delta-delta coefficients.

Let \mathbf{o}_k be the feature vector at the horizontal pixel position p and \mathbf{o}_{k+i} (resp. \mathbf{o}_{k-i}) the feature vector of the sliding window shifted by $i * \delta$ (resp. $-i * \delta$) pixels from the current window, at pixel position $p + i * \delta$ (resp. $p - i * \delta$). The first order regression feature vector \mathbf{o}_k is the slope of the regression line around \mathbf{o}_k . It is written

$$\Delta \mathbf{o}_k = \frac{\sum_{i=1}^K i(\mathbf{o}_{k+i} - \mathbf{o}_{k-i})}{2 \sum_{i=1}^K i^2} \quad (6.8)$$

K is the chosen depth of the regression, giving the number of surrounding feature vectors ($2 * K$) used for computing the dynamic features. The second order regression is simply derived from Eq. (6.8) by replacing \mathbf{o}_k by $\Delta \mathbf{o}_k$:

$$\Delta \Delta \mathbf{o}_k = \frac{\sum_{i=1}^K i(\Delta \mathbf{o}_{k+i} - \Delta \mathbf{o}_{k-i})}{2 \sum_{i=1}^K i^2} \quad (6.9)$$

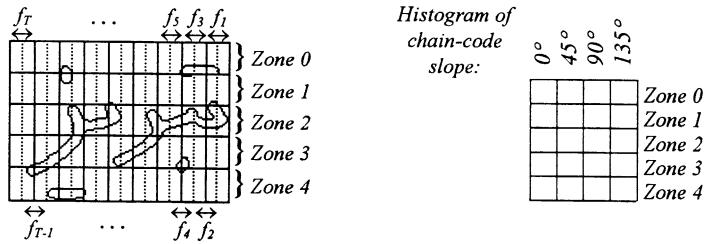


Fig. 6.5 Word divided into strips. In each strip, the histogram of the chain code direction is computed (from [10])

The final feature vector is thus the concatenation of the original vector \mathbf{o}_k , its first order regression vector $\Delta\mathbf{o}_k$, and possibly its second order regression vector $\Delta\Delta\mathbf{o}_k$.

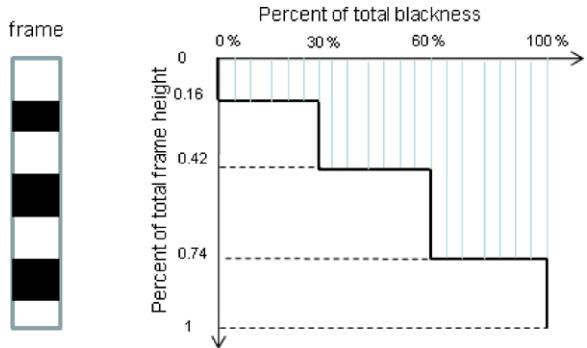
6.2.5 Other Features for Sliding-Window HMM Systems

There are many ways of extracting features from sliding windows. We have presented in this chapter a set of distribution and concavity features (see Sects. 6.2.3 and 6.2.2). To model the distribution of foreground pixels, raw pixel values (sometimes called appearance-based features) can also be used such as in [11, 23]. Moreover, several neighboring windows consisting of pixel values can be concatenated into one frame in order to include contextual information around the current window. The number of pixel values (features) within one frame is thus high and can be reduced by a principal component analysis (PCA) or a Karhunen–Loeve transformation.

Besides raw features, directional features are popular for both Arabic and Latin. These features may be extracted from word contour points and their local direction expressed within chain codes [10]. The contour image of the word is subdivided into zones corresponding to the core zone and two upper and two lower zones for ascenders, descenders, and diacritical marks (see Fig. 6.5). Structural features are extracted in [8, 27] within windows which are also divided into zones. The presence of diagonal, vertical, horizontal, and curved strokes and their orientation is collected into the feature vector. Derivative features such as the difference vector between two frames can also be included [11].

Percentile features are extracted within the frames of binary images [22, 26]. At each y -position, the number of black pixels from the frame's top to y is computed. This function of y is then normalized from 0 to 100, which means 0 % to 100 % of the total blackness of the frame. The y -axis is also normalized from 0 to 1, which means 0 % to 100 % of the total height of the frame. The total blackness is then divided into 20 percentiles, and the corresponding percentile values (percentile height of the frame) are the percentile features (see Fig. 6.6).

Fig. 6.6 Percentile features (on the y-axis) (adapted from [22])



6.3 Features for Grapheme-Based Systems

Grapheme-based systems are based on an explicit segmentation of the word into subparts called graphemes. The segmentation is an over-segmentation, which means that a grapheme is either a character or a subpart of a character.

6.3.1 Segmentation into Graphemes

In the hybrid HMM/NN system detailed in Sect. 6.4.3, the grapheme segmentation process is composed of the following steps:

- Detection of the connected components and of the internal and external contour information.
- Extraction of the skeleton of each connected component and representation as a graph.
- Detection of segmentation points (bottom part of concavities, extremities of horizontal segments). Graphemes are defined as vertical or diagonal arcs which can be linked without crossing a potential segmentation point.
- Attribution of the remaining arcs to a neighboring grapheme or identifying them as a ligature.
- Retrieval of the image of each potential grapheme from the labeled arcs.

6.3.2 Features Extracted from Graphemes

The features extracted for each grapheme are relatively simple and can be computed very quickly, as follows:

- Height and width of the bounding box of the grapheme (2 values).
- Height-width ratio (1 value).

Fig. 6.7 Baseline and grapheme extraction

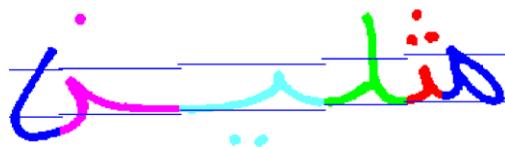


Fig. 6.8 Grapheme divided into strips. In each strip, the local direction is computed and encoded (from [28])



- Position of the top and bottom of the bounding box with respect to the baseline (2 values).
- Position of the gravity center of the grapheme in the bounding box (2 values).
- Black pixel density in the bounding box (1 value).
- Black pixel density in three zones: above, under, and inside the baseline (3 values).
- Surface of the loops in the three previous areas (3 value).
- Value of the top, bottom, left and right profiles of the grapheme taken in five points ($4 * 5$ values).
- Cumulated thickness of the grapheme, horizontally, vertically, and along the two diagonals. For each direction, the thickness is computed in 5 parallel areas ($4 * 5$ values).
- Number of intercepts along the four same directions, in the 5 parallel areas ($4 * 5$ values).

The features are normalized with respect to the baseline height; the total number of features is 74.

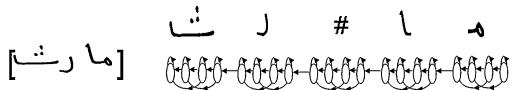
6.3.3 Other Features Extracted from Graphemes

We have presented above different features related to the grapheme position and pixel density. Other types of features can be extracted. Similarly to sliding windows, directional and structural features can be extracted on graphemes (see Fig. 6.7).

In [28], graphemes are divided into a fixed number of horizontal strips. Within each strip, the direction of the grapheme is extracted with the Hough transform (see Fig. 6.8). Thus, the local directions of the grapheme can be encoded. In [4], directions are extracted by dividing graphemes into 3×3 zones and computing the direction histograms of contour points.

From the word's skeleton, structural elements such as loops and segments are extracted from words [16]. Length and orientation of segments, types of loops and

Fig. 6.9 Modeling of a word by its compound character models



presence of turning points, and branch and cross points are computed. All elements are turned into discrete symbols.

6.4 Application to Arabic Handwriting Recognition

The features presented in Sect. 6.2 are evaluated on Arabic word recognition tasks within state-of-the-art hidden Markov model (HMM) systems. We conduct experiments with two sliding-window systems and one grapheme-based system. Sliding-window systems are both based on the segmentation-free analytical strategy.

6.4.1 Context-Independent Sliding-Window System

In the context-independent system, a word is modeled by the concatenation of its compound character models. This is illustrated in Fig. 6.9. For Arabic, we define character models which can be letters (with diacritical marks or not), ligatures, or numerals: 120 character models are defined for the IFN/ENIT database and 150 for the OpenHart database. Since the lexicon corresponding to the OpenHart database is much larger than the lexicon of the IFN/ENIT database, more character models are needed.

All character models share the same HMM topology: the same number S of emitting states, left-right transitions with one skip allowed. The observation probability density for each state is a mixture of N_G Gaussian distributions. This mixture is obtained by incrementing step by step the number of Gaussian distributions in each state until a convenient HMM topology is reached. The number of Gaussian distributions is increased as follows:

- The mixture has n components and it is to be increased to $n + m$.
- For the k th mixture to be added, $k \in [1 \dots m]$, find the mixture with the largest weight and split this mixture:
 - Divide the weight into two halves
 - Clone the mixture
 - Add perturbation to each mean vector cloned by adding (resp. subtracting) a normal distribution centered at zero and with a small standard deviation σ_{split} (default σ_{split} is 0.2) to each term.

The HMM models are initialized with one Gaussian distribution per state and are trained according to the Baum–Welch algorithm. Then, for each Gaussian mixture



Fig. 6.10 Illustration of the influence of context for handwriting. The three words العالم والقصادات الصين have been written by the same writer. However, characters laB, aaE, and saM yield different shapes

incrementing step ($n \rightarrow n + 1$), HMM models are retrained with this same algorithm.

We optimized the features extraction parameters on a validation database: window width w , overlap between two sliding windows δ , number of cells per window n_c . We start from binary images and extract the set of features presented in Sects. 6.2.2 through 6.2.4.

Decoding is performed with the Viterbi algorithm. We use the Hidden Markov Model Toolkit (HTK) for training and recognition.

6.4.2 Context-Dependent Sliding-Window System

The previous context-independent system can be enhanced by refining character models. For a given character, we have considered the influence of neighboring characters on its shape (see Fig. 6.10). We have used our knowledge of ligatures and the shapes of leftmost or rightmost parts of neighboring characters to assist the modeling of individual letters. In the context-dependent system, we build different character models according to different contexts, i.e., the characters on the right and the left of a central character. Contextual approaches lead to an excessive growth in the number of models, since one model is needed for each pair of adjacent characters. Parameter estimation may be unreliable since, for practical applications, a restricted set of training data is generally available. It is thus desirable to reduce the number of models and model parameters while preserving model refinement. Hence, model sharing and parameter tying are necessary to reduce the number of parameters. State tying determines which states can share the same Gaussian distributions. The state position-based principle is that, for a given central letter, all states corresponding to the same position in an HMM model are subject to agglomerative clustering. Our approach consists of building expert-based rules and decision trees to perform this state clustering.

The merging or splitting of state clusters is driven by a binary tree whose nodes correspond to rhetorical questions on the characteristics of the models. Such decision trees have been designed for speech recognition at the phone level by experts [9].

In our case, decision trees are based on a set of questions on the behavior of left and right contexts, and are applied to states (see the [Appendix](#)). Based on the same initial set of questions, one tree is built for every state position of all trigraphs with

the same central letter. Starting at the root node, all the states corresponding to the same position and the same central letter are gathered into a single cluster. Then, the binary question which maximizes the likelihood of the two children clusters it would create is chosen and the split is made, creating two new nodes. This splitting continues until the increase in likelihood falls below a threshold or no questions are available to create nodes with a sufficient state occupancy count. An example of a decision tree is shown in Fig. 6.11.

Let us consider a node containing the set of states S to be split in a given tree. The set S corresponds to the set of training frames $\{o_f\}_{f \in F}$. As all states in S are tied in the node, they all share the same mean $\mu(S)$ and variance $\Sigma(S)$. The likelihood of S generating the set of frames is hence given by

$$L(S) = \sum_{f \in F} \sum_{s \in S} \log(\Pr(o_f; \mu(S), \Sigma(S))) \gamma_s(o_f) \quad (6.10)$$

where $\gamma_s(o_f)$ is the a posteriori probability of frame o_f being generated by state s . Based on the work of Young [29] and assuming that we adopt Gaussian probability density functions, $L(S)$ can be rewritten

$$L(S) = -\frac{1}{2} (\log[(2\pi)^n |\Sigma(S)|] + n) \Gamma(S) \quad (6.11)$$

$\Gamma(S)$ is the accumulated state occupancy of the node, $\Gamma(S) = \sum_{f \in F} \sum_{s \in S} \gamma_s(o_f)$, and n is the dimension of the feature vectors.

Then, we introduce ΔL_q :

$$\Delta L_q = L(S_{q+}) + L(S_{q-}) - L(S) \quad (6.12)$$

The split of the state set into two subsets S_{q+} (answer to q is *yes*) and S_{q-} (answer to q is *no*) is made by question q^* which maximizes ΔL_q , provided that $\Gamma(S_{q+})$ and $\Gamma(S_{q-})$ are over the minimal state occupancy threshold, and that ΔL_q is above the threshold of minimal increase in likelihood. This condition can be reformulated [31]:

$$q^* = \operatorname{argmin}_q \left\{ \frac{1}{2} [\Gamma(S_{q+}) \log(|\Sigma(S_{q+})|) + \Gamma(S_{q-}) \log(|\Sigma(S_{q-})|) - \Gamma(S) \log(|\Sigma(S)|)] \right\} \quad (6.13)$$

The parameters ensuring efficient sizes of state clusters, namely the minimal state occupancy threshold and the minimal increase in likelihood, are tuned on the validation database. Trees reduced to their only root can be observed. They correspond to monographs with few examples which aim to tie all their corresponding trigraphs into a single model.

Decision trees have the ability of modeling unseen trigraphs with existing ones. This property is useful when test and training dictionaries differ. Each state of a new trigraph is positioned at the root node of the tree corresponding to the same state position and the same central letter. Then each state descends its belonging tree, answering questions on the trigraph contexts, until it reaches a node where a cluster

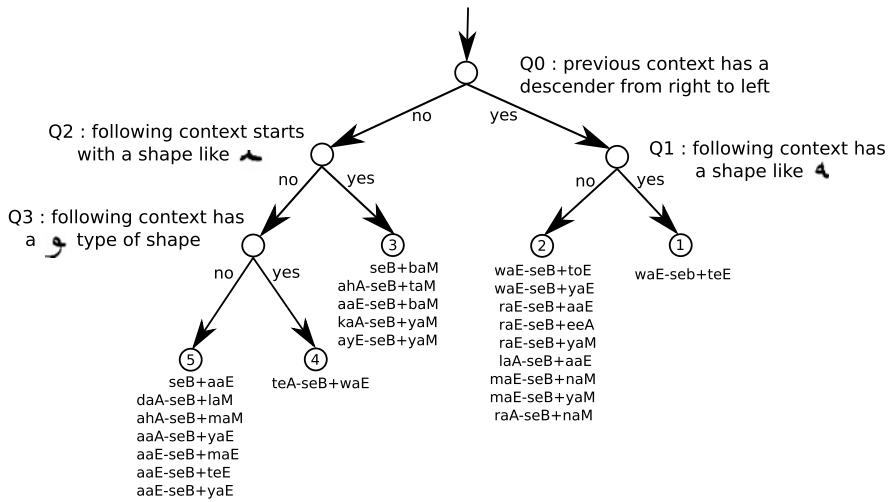


Fig. 6.11 Example of a decision tree for state clustering: questions and clusters are shown for the second state of all $^*-\text{seB}+^*$ () trigraphs

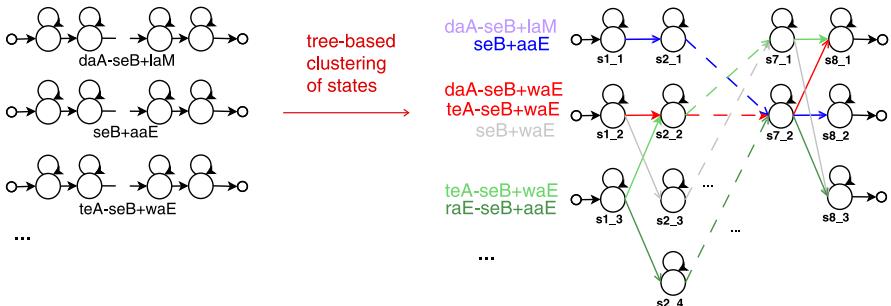


Fig. 6.12 Illustration of state clustering for the trigraphs centered on character

is positioned. The state model representing the cluster will be the model assigned to the considered state number of the trigraph for its recognition.

The training of trigraphs is performed as follows:

- We start with trained monographs with one Gaussian distribution associated to each state. Trigraphs are initialized by copying monographs. All the trigraphs associated to a given central letter are listed in the training database, and the initialized monograph model of the central letter is given as a first model for all those trigraphs.
- Then, a first and rough estimation of the trigraph parameters is obtained with a single iteration of the Baum–Welch estimation algorithm on all the different trigraphs.
- State tying is performed at each state position as described above.

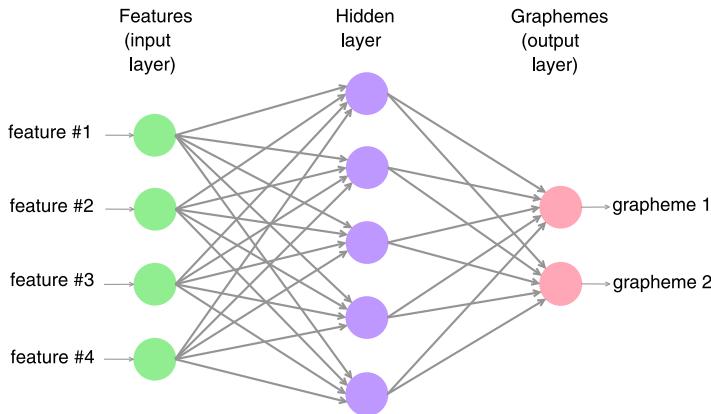


Fig. 6.13 The neural network used to predict the probability of each grapheme given a feature vector

- Using the Baum–Welch algorithm for model re-estimation, we increase the number of Gaussian distributions per state up to a chosen number NG.

An illustration of state clustering is shown in Fig. 6.12.

6.4.3 Grapheme-Based HMM

In this system, a neural network (NN) is trained to evaluate the posterior probability of each grapheme with respect to the feature vector. The neural network is a multi-layer perceptron (MLP) with as many input neurons as features (74), one layer of hidden neurons (500 neurons), and as many output neurons as grapheme classes (200). The transfer function is a softmax function. The neural network was trained in a supervised way with a stochastic backpropagation training algorithm. A simplified schema of the neural network is shown in Fig. 6.13.

Training and Recognition Hidden Markov models (HMMs) are used to model the decomposition of words into letters and then each letter into graphemes. The topology of the model is a left-to-right topology with four states for each letter HMM (see Fig. 6.14), each state corresponding to a grapheme. The hybrid NN/HMM was trained using the following procedure:

1. Decode the training set with the hybrid NN/HMM recognizer in order to create an annotated base of feature vectors.
2. Train the NN on the annotated base of feature vectors.
3. Compute the sequences of observation probability with the new NN for all words in the training set.
4. Train the HMM on the sequences of observation probability using the Baum–Welch algorithm.

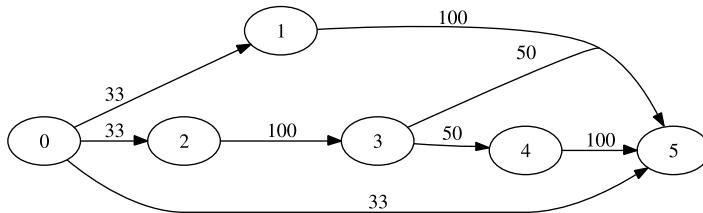


Fig. 6.14 HMM topology of a letter. The weights are initialized uniformly

5. Go back to step 1 until no improvement is observed on the recognition rate.

This procedure needs a basic recognizer to bootstrap the process; the convergence is usually observed after a few iterations.

6.5 Experiments

The features presented in Sect. 6.2 were experimented on several databases and within the systems presented in Sect. 6.4. These systems have also been combined, which increased performance.

6.5.1 Results on Arabic Databases

Systems were first tested on the benchmark IFN/ENIT database of Arabic city names [24]. The total amount of binary images of handwritten Tunisian town/village names is 26,459. Those names were written by 411 writers and they were labeled according to 937 name classes. Ground truth information is added to each entry of the database including character shape sequence and corresponding postcode. More details about ground truth data can be found in [23]. The database is separated into four sets, a, b, c, and d, in order to perform 4-fold cross-validation experiments. For the ICDAR 2005 competition, systems were tested on a novel data set, e, of 6,033 word images.

First, competing systems are single systems which are further enhanced by system combination. The first combined systems are context-independent systems which differ in the way that the feature extraction sequence is extracted. Sliding windows are slanted, and each system corresponds to a different slant angle. Combination is performed by an NN.

Then, different HMMs can be combined: we obtain a context-independent, a context-dependent, and a hybrid MLP-HMM system. The sliding-window systems (context-dependent and context-independent) are trained with more parameters (number of Gaussian distributions associated to states) than the previous systems. A combination of HMM systems always increases performance.

The systems presented above deal with a closed-vocabulary task. The results presented here correspond to a word recognition task, but the position of the words

Table 6.1 Results on Arabic databases

Primary system	Parameters	Competition	Test set	Voc. size	W. rec. in %
Context-independent HMM	3G/state, 4states/charact.	ICDAR 2005	IFN/ENIT set e	936	75.93
Context-independent HMM	3G/state, 4states/charact.	ICDAR 2007	IFN/ENIT set e	936	81.8
Combination 3 context-independent HMMs	HMM + NN param.	ICDAR 2007	IFN/ENIT set e	936	85.13
Combination 3 context-independent HMMs	HMM + NN param.	ICDAR 2007	IFN/ENIT set f	936	81.93
Combination 3 context-independent HMMs	HMM + NN param.	ICDAR 2009	IFN/ENIT set f	936	83.98
Combination context-independent + grapheme MLP-HMM	20G/state 11states/charact. + NN param.	ICDAR 2009	IFN/ENIT set f	936	89.42
Context-independent HMM + LM	20G/state 11states/charact.	OpenHart 2011	Eval_Phase2	20 K	44.9
Context-dependent HMM + LM	20G/state 11states/charact.	OpenHart 2011	Eval_Phase2	20 K	54
Combination (CD + CI + grapheme MLP-HMM) + LM	HMM + NN param.	OpenHart 2011	Eval_Phase2	20 K	62.3

within text lines is known. Moreover, the size of the dictionary has been increased, since the vocabulary is open. The systems thus include a language model (LM) based on trigrams. The results shown in Table 6.1 are the official results of the OpenHart competition (see [13]). A language model has been trained on a subset of the training documents. The lexicon has been limited to the 20,000 most frequent words and a bigram, trigram model is used. The combination of the three systems above, context-independent, context-dependent, and hybrid systems, has been achieved by training an NN as described in [6].

6.5.2 Results on Latin Databases

As mentioned in Sect. 6.1, Latin handwriting shares similarities with Arabic handwriting. Thus, our features and systems have now been tested on publicly available Latin databases such as the English IAM database [19] and the French RIMES [12] databases. The IAM database provides 9,862 handwritten text lines segmented into words. We extract from the correctly segmented and annotated text line images [32]

Table 6.2 Results on Latin databases

Primary system	Parameters	Competition	Test set	Voc. size	W. rec. in %
Grapheme MLP-HMM		ICDAR 2009		5,334	75.7
Context-dependent HMM	11states/charact. 20G/state	ICDAR 2009	RIMES 7,464	5,334	76.1
Context-independent HMM	4states/charact. 5G/state	ICDAR 2009	RIMES 7,464	5,334	72.0
Best combination (CI + CD + grapheme)	HMM + NN param.	ICDAR 2009	RIMES 7,464	5,334	86.9
Context-independent HMM	20G/state 10states/charact.	–	IAM 13,752	10.5 K	64.6
Context-dependent HMM	20G/state, 10states/charact.	–	IAM 13,752	10.5 K	67.3
Best combination (CI + CD + grapheme)	HMM + NN param.	–	IAM 13,752	10.5 K	78.1

a training set of 46,901 word images, two distinct validation sets containing respectively 6,442 and 7,061 word images, and a 13,752-word test set. The RIMES database was used during the ICDAR 2009 handwriting recognition competition (word recognition WR3 task). The database is composed of French words which include a number of diacritical marks (accents); the meaning of words is changed according to these marks. For instance “annule” means “cancel” and “annulé” means “canceled.” 44,197 word images are given for training, 7,542 word images for validation, and 7,464 word images for testing. The training lexicon includes 4,500 words, and the validation lexicon includes 1,600 words, as does the test lexicon. Note that the lexicons are case and accent sensitive and that lexicons are different from training/validation to test. The dictionary sizes explain the different results between both databases. Results of the different systems on these databases are reported in Table 6.2.

6.6 Conclusion

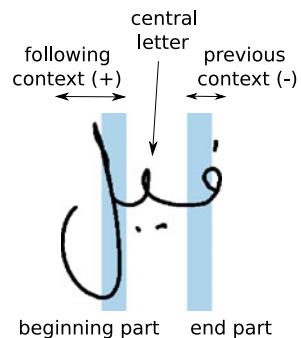
We have presented a number of features based on pixel distributions, concavities, and local directions currently imbedded within state-of-the-art HMM-based systems for Arabic word recognition. These features can be enhanced by dynamic features; alternatively, one can continue to discover new efficient features.

Automatic construction of features from pixel images can also be achieved within neural network-based architectures, deep-belief nets, or recurrent nets.

With the development of text databases, HMM systems will need to process text lines rather than, or in addition to, words. The features which are presented in this chapter for word recognition tasks can also be useful, with minor modifications, for processing text lines.

Fig. 6.15 Example of previous and following contexts for letter $\underline{\text{ا}}$ in word

فیل



Acknowledgements The authors wish to dedicate this chapter to the memory of our colleague and friend Adnan Amin, who was a pioneer and contributed extensively to the development of Arabic recognition systems. He also encouraged many researchers in their contributions in this domain, for which the authors are indebted to him.

Appendix

As we have stated in previous work [5, 6], it is quite obvious that the way of writing a character within a word is affected by adjacent letters; therefore lies the justification of using context models to improve the accuracy of modeling. We have formerly used the terms “left context” and “right context” to designate the adjacent letters. In order to avoid any confusion due to the Arabic writing direction, we will now refer to “previous context” and “following context” instead.

We use HMM Toolkit (HTK) syntax [30] to designate trigraphs, and IFN/ENIT transliteration [24] to make it writable with ASCII characters. For example, in the Arabic word فیل, the letter $\underline{\text{ا}}$ is surrounded by the letters $\dot{\text{ي}}$ (previous context) and $\dot{\text{l}}$ (following context); see Fig. 6.15. Using HTK notation, previous context is defined by “-” and following context is defined by “+”, which gives the trigraph: faB-yaM+laE.

The construction of our question sets is driven by the two following hypotheses:

- Letters with similar ending strokes will have a tendency to affect the following central letter in a similar manner.
- Letters with similar beginning strokes will have a tendency to affect the previous central letter in a similar manner.

According to these hypotheses, Arabic letters which share the same shapes [20] should be good candidates to build question sets (QSSs). Those QSSs will be later used in our clustering decision trees.

Similar endings will lead to groupings in previous-context question sets (P_QS), whereas similar beginnings will lead to groupings in following-context question sets (F_QS).

For example, the beginning (right part) of letters {ق ف قه} is very similar. They will be regrouped in the same F_QS: {*+faM, *+kaM, *+faE, *+kaE}.

The same is true for $\{\cup, \cdot\} \rightarrow \{^{*+raE}, ^{*+zaE}\}$.

In {ب ش ت م س ش ش س ش ب ت ب ش ت ب ش ش س ش س ش}, all letters do not share the same shapes, but again their beginning (right part) looks similar. This should lead to the creation of the following F_QS: {*+seM, *+shM, *+seE, *+shE, *+naM, *+baM, *+taM, *+thM, *+baE, *+taE, *+thE}.

The same is true for the ending (left part) of letters, used to build P_QS:

$$\{\text{a, b}\} \rightarrow \{\text{ayM}^{-*}, \text{ghM}^{-*}\}$$

{ز ز ر ر و و} → {waE-*, waA-*, raE-*, raA-*, zaE-*, zaA-*}

The full list of QSSs can be found at [14].

References

1. Al-Hajj Mohamad, R., Likforman-Sulem, L., Mokbel, C.: Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **31** (2009)
 2. Amin, A.: Off-line Arabic character recognition: the state of the art. *Pattern Recognit.* **31**(5), 517–530 (1998)
 3. BenAmara, N., Bouslama, F.: Classification of Arabic script using multiple sources of information: state of the art and perspective. *Int. J. Doc. Anal. Recognit.* **5**, 195–212 (2003)
 4. Benouareth, A., Ennaji, A., Sellami, M.: HMMs with explicit state duration applied to handwritten Arabic word recognition. In: *ICPR*, vol. 2, pp. 897–900 (2006)
 5. Bianne, A.L., Kermorvant, C., Likforman-Sulem, L.: Context-dependent HMM modeling using tree-based clustering for the recognition of handwritten words. In: *Proceedings of Electronic Imaging-Document Recognition and Retrieval XVII—DRR 2010 SPIE*, vol. 7534 (2010)
 6. Bianne-Bernard, A.L., Menasri, F., Al-Hajj, R.M., Mokbel, C., Kermorvant, C., Likforman-Sulem, L.: Dynamic and contextual information in HMM modeling for handwritten word recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(10), 2066–2080 (2011)
 7. Blumenstein, M., Cheng, C.K., Liu, X.Y.: New pre-processing techniques for handwritten word recognition. In: *Proceedings of the Second IASTED International Conference on Visualization, Imaging and Image Processing*, pp. 480–484 (2002)
 8. Caesar, T., Gloger, J.M., Mandler, E.: Pre-processing and feature extraction for a handwritten recognition system. In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 408–411 (1993)
 9. Chelba, C., Morton, R.: Mutual information phone clustering for decision tree induction. In: *Proceedings of the International Conference on Spoken Language Processing—ICSLP02* (2002)

10. Dehghan, M., Faez, K., Ahmadi, M., Shridhar, M.: Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM. *Pattern Recognit.* **34**(5), 1057–1065 (2001)
11. Drewu, P., Rybach, D., Gollan, C., Ney, H.: Writer adaptive training and writing variant model refinement for offline Arabic handwriting recognition. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 21–25 (2009)
12. Grosicki, E., El-Abed, H.: ICDAR 2009 handwriting recognition competition. In: ICDAR, pp. 1398–1402 (2009)
13. <http://www.nist.gov/itl/iad/mig/hart2010.cfm>
14. <http://perso.telecom-paristech.fr/lauli/ArabicScriptRules/>
15. Kanoun, S., Alimi, A.M., Lecourtier, Y.: Natural language morphology integration in off-line Arabic optical text recognition. *IEEE Trans. Syst. Man Cybern., Part B, Cybern.* **41**(2), 579–590 (2011)
16. Khorsheed, M.S.: Recognising handwritten Arabic manuscripts using a single hidden Markov model. *Pattern Recognit. Lett.* **24**(14), 2235–2242 (2003)
17. Lorigo, L.M., Govindaraju, V.: Offline Arabic handwriting recognition: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(5), 712–724 (2006)
18. Märgner, V., Abed, H.E.: ICFHR 2010—Arabic handwriting recognition competition. In: International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 709–714 (2010)
19. Marti, U., Bunke, H.: A full English sentence database for off-line handwriting recognition. In: Proceedings of the Fifth International Conference on Document Analysis and Recognition—ICDAR99, pp. 705–708 (1999)
20. Menasri, F., Vincent, N., Cheriet, M., Augustin, E.: Shape-based alphabet for off-line Arabic handwriting recognition. In: International Conference on Document Analysis and Recognition vol. 2, pp. 969–973 (2007)
21. Mohamad, R.A.H., Likforman-Sulem, L., Mokbel, C.: Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(7), 1165–1177 (2009)
22. Natarajan, P., Lu, Z., Schwartz, R.M., Bazzi, I., Makhoul, J.: Multilingual machine printed OCR. *Int. J. Pattern Recognit. Artif. Intell.* **15**(1), 43–63 (2001)
23. Pechwitz, M., Märgner, V.: HMM-based approach for handwritten Arabic word recognition using the IFN/ENIT database. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 890–894 (2003)
24. Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT database of handwritten Arabic words. In: CIFED (2002)
25. Plotz, T., Fink, G.: Markov models for offline handwriting recognition: a survey. *Int. J. Doc. Anal. Recognit.* **12**, 269–298 (2009)
26. Saleem, S., Cao, H., Subramanian, K., Kamali, M., Prasad, R., Natarajan, P.: Improvements in BBN’s HMM-based offline Arabic handwriting recognition system. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 773–777 (2009)
27. Schambach, M.P., Rottland, J., Alary, T.: How to convert a Latin handwriting recognition system to Arabic. In: Proceedings of ICFHR’08 (2008)
28. Touj, S.M., Amara, N.E.B., Amiri, H.: A hybrid approach for off-line Arabic handwriting recognition based on a planar hidden Markov modeling. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 964–968 (2007)
29. Young, S.J., Odell, J.J., Woodland, P.C.: Tree-based state tying for high accuracy acoustic modelling. In: Proceedings of the Workshop on Human Language Technology (HLT94), pp. 307–312 (1994)
30. Young, S., et al.: The HTK Book V3.4. Cambridge University Press, Cambridge (2006)

31. Zen, H., Tokuda, K., Kitamura, T.: Decision tree based simultaneous clustering of phonetic contexts, dimensions, and state positions for acoustic modeling. In: Proceedings Eurospeech, pp. 3189–3192 (2003)
32. Zimmermann, M., Bunke, H.: Automatic segmentation of the IAM off-line database for handwritten English text. In: Proceedings of the 15th International Conference on Pattern Recognition—ICPR2000, vol. 4, pp. 35–39 (2000)

Part II

Recognition

Chapter 7

Printed Arabic Text Recognition

Irfan Ahmed, Sabri A. Mahmoud, and Mohammed Tanvir Parvez

Abstract This chapter addresses automatic printed Arabic text recognition. Arabic text recognition has its own difficulties due to the cursive nature of the scripts, overlapping characters, large number of dots and diacritics, etc. In this chapter, we present a general framework for a printed Arabic text recognition system. We then discuss different phases of such a system, e.g., pre-processing, feature extraction, and classification. We present different reported techniques for each phase. In addition, different databases for printed Arabic text recognition are discussed here. We conclude this chapter by presenting several experimental results for hidden Markov model (HMM)-based printed Arabic text recognition.

7.1 Introduction

Since the advent of writing as a form of communication, paper prevailed as the writing medium. However, electronic media are replacing paper with time. Because they conserve space and are quickly accessed, electronic media are constantly gaining popularity. The convenience of paper, its widespread use for communication and archiving, and the amount of information already on paper call for quick and accurate methods to automatically read that information and convert it into electronic form [6].

The potential application areas of automatic reading machines are numerous. One of the earliest and most successful applications is sorting checks in banks, as the volume of checks that circulates daily has proven to be too enormous for manual entry. Other applications are detailed in [45, 64].

I. Ahmed (✉) · S.A. Mahmoud · M.T. Parvez

Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran,
Saudi Arabia

e-mail: irfanics@kfupm.edu.sa

S.A. Mahmoud

e-mail: smasaad@kfupm.edu.sa

M.T. Parvez

e-mail: tparvez@kfupm.edu.sa

The machine replication of human reading, i.e., optical character recognition (OCR), has been the subject of extensive research for more than five decades. Character recognition is a pattern recognition application with the ultimate aim of simulating the human reading capabilities of both machine printed and handwritten cursive text. The currently available systems may read faster than humans, but they cannot reliably read such a wide variety of text or consider context. One can say that a great amount of further effort is required to at least narrow the gap between human reading and machine reading capabilities. The practical importance of OCR applications, as well as the interesting nature of the OCR problem, has led to great interest and measurable advances in this field. Now, commercial OCR systems for Latin characters are widely available on personal computers and can achieve recognition rates above 99 % [65, 91]. Furthermore, systems on the market can now read a variety of writing styles (e.g., handwritten, printed omnifont) and character sets including Chinese, Japanese, Korean, Cyrillic, and Arabic.

Since the 1950s, researchers have carried out extensive work and published many papers on character recognition. Most of the published work on OCR has been on Latin, Japanese, or Chinese characters. This started in the mid-1940s for Latin and in the mid-1960s for Chinese and Japanese. The following are some useful surveys and reviews on Latin character recognition. Reference may be made to [68] for a historical review of OCR research and development. The survey of [45] includes surveys of other languages, [64] has an overview of character recognition methodologies, [49] reviews commercial OCR systems, [89] machine printed OCR, and [88, 90] on-line handwriting recognition. Suen et al. [87] has a survey on automatic recognition of hand printed characters (viz., numerals, alphanumeric, Fortran, and Katakana), while [69] produced a review of the recognition of hand printed (non-cursive) characters and conducted beta tests on a commercial system. Bozinovic and Srihari [27] and Simon [81] surveyed off-line cursive word recognition, Jain et al. [51] reviewed statistical pattern recognition techniques, and [73] is a comprehensive survey of on-line and off-line handwriting recognition. Bibliographies of the fields of OCR and document analysis appeared in [52, 55]. Stallings [85] and Mori et al. [67] produced surveys on recognition of Chinese machine printed and hand printed characters, respectively, and Liu et al. [61] addressed the state of the art of on-line recognition of Chinese characters.

7.2 Issues in Arabic Printed Text Recognition

In this section, we present some characteristics of Arabic script. We also discuss some issues related to the recognition of printed Arabic text.

Arabic text is written cursively from right to left. The Arabic alphabet has 28 basic characters, as shown in Fig. 7.1. An Arabic character may have up to four basic different shapes depending on the position of the character in a word: isolated, beginning, ending, or middle form (see Table 7.1). Characters of a word may overlap vertically with or without touching. Different Arabic characters have different sizes

ا ب ت ث ج ح خ ذ ز س ش ص ض ط ظ ع غ ف ق ل ك ل م ن ه ر ي

Fig. 7.1 Basic characters in Arabic

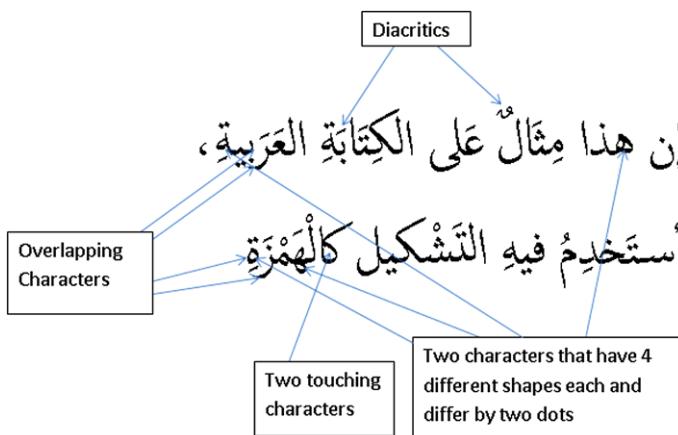


Fig. 7.2 An example of an Arabic sentence indicating some characteristics of Arabic text

(height and width). Characters in a word can have short vowels (diacritics). These diacritics are written as strokes, placed either on top of or below the letters. A different diacritic on a character may change the meaning of a word. Each diacritic has its own code as a separate character when it is considered in a digital text. Readers of Arabic are accustomed to reading unvocalized text by deducing the meaning from context. Figure 7.2 shows some of the characteristics of Arabic text related to character recognition. It shows a baseline, overlapping letters, diacritics, and three shapes of the Meem character (ending, middle, and beginning).

Regularities present in printed text offer some advantages in the recognition of printed Arabic text compared to Arabic handwriting recognition. The shapes of the characters, the spacing between words and lines, etc., are more regular in printed text than in handwritten text. The baseline in printed Arabic text is straighter compared to handwritten text. Thus, baseline-dependent features can be used more reliably in printed Arabic text recognition. However, there are several issues that researchers need to consider while developing techniques for printed Arabic text recognition.

- **Cursive Text:** Arabic is written cursively both in printed and in handwritten form. This means that the segmentation of printed Arabic text is not straightforward compared to Latin printed text. Moreover, overlapping of characters is common in printed Arabic text. Thus, segmentation of words into characters is less trivial.
- **Ligatures:** Ligatures are very common in Arabic text, both in printed and handwritten form. Some of the ligatures are optional and depend on the type of fonts being used. Ligatures are difficult to segment and are generally modeled by considering each ligature as a separate character.

Table 7.1 Shapes of Arabic characters

General	Contextual forms			
	Isolated	End	Middle	Beginning
ا	ا	ا		
ب	ب	ب	ب	ب
ت	ت	ت	ت	ت
ث	ث	ث	ث	ث
ج	ج	ج	ج	ج
ح	ح	ح	ح	ح
خ	خ	خ	خ	خ
د	د	د		
ذ	ذ	ذ		
ر	ر	ر		
ز	ز	ز		
س	س	س	س	س
ش	ش	ش	ش	ش
ص	ص	ص	ص	ص
ض	ض	ض	ض	ض
ط	ط	ط	ط	ط
ظ	ظ	ظ	ظ	ظ
ع	ع	ع	ع	ع
غ	غ	غ	غ	غ
ف	ف	ف	ف	ف
ق	ق	ق	ق	ق
ك	ك	ك	ك	ك
ل	ل	ل	ل	ل
م	م	م	م	م
ن	ن	ن	ن	ن
هـ	هـ	هـ	هـ	هـ
وـ	وـ	وـ	وـ	وـ
يـ	يـ	يـ	يـ	يـ

- *Large Number of Fonts:* Despite the regularities in printed Arabic text, the large number of available fonts for Arabic makes the recognition task challenging. Shapes of letters may vary between fonts. In addition, spaces between words/subwords, overlapping of characters, number of ligatures, etc., can vary widely in different fonts. A robust printed Arabic text recognition system should be trained on as many different fonts as possible.

7.3 Databases for Arabic OCR

In this section, we discuss different printed Arabic text databases reported and/or used by researchers of printed Arabic text recognition.

Several printed Arabic text databases are reported in the literature; however, we mention that there is no generally accepted database for printed Arabic text recognition that is freely available for researchers. Hence, different researchers of printed Arabic text recognition have used different data, and hence the recognition rates of the different techniques may not be comparable.

The ERIM database was developed in early 1995 by the Institute of Environmental Research, Michigan [80]. This database consists of over 750 pages of printed Arabic texts containing 1,000,000 characters and 200 ligatures. However, this database is no longer available.

The DARPA Arabic Machine Print (DAMP) document corpus was collected by SAIC [29]. The corpus consists of 297 images scanned from newspapers, books, magazines, etc. The corpus was partitioned into three sets: 60 images for development, 60 images for testing, and 177 images for training the OCR system.

The Arabic Gigaword database is collected by Linguistic Data Consortium (LDC) at the University of Pennsylvania [46]. This database contains 1,500 million Arabic words, collected from different news agencies over several years.

Abdelraouf et al. [4] presented a database containing 6 million Arabic words. This database is collected from a wide variety of selected sources covering old Arabic, religious texts, traditional language, modern language, different specializations, and very modern material from chat rooms.

The Arabic Printed Text Images (APTI) database was presented by Slimane et al. [83]. The database is synthetically generated using a lexicon of 113,284 words, 10 Arabic fonts, 10 font sizes, and 4 font styles. The database contains 45,313,600 single word images totaling to more than 250 million characters. Ground truth annotation is provided for each image.

Several other printed Arabic text databases are reported in different works. Ben Amor et al. [23] used a database of 85,000 sample Arabic words in five different fonts: Arabic transparent, Badr, Alhada, Diwani, and Koufi. Slimane et al. [82] used a synthetic database of word images composed of 20,630 Arabic word images. They generated the images by a Java procedure using the font Times, 24 points. Prasad et al. [75] used the DARPA database along with 380 synthetically generated images of printed Arabic text. These 380 pages were created by printing 100 newswire text passages in multiple font types and font sizes.

Khorsheed used a data corpus that includes Arabic text of more than 100 A4-size sheets typewritten in Tahoma font to assess the performance of the reported mono-font system [58]. In [59], Khorsheed used a database of more than 600 A4-size pages of Arabic text typewritten in six different computer-generated fonts: Tahoma, Simplified Arabic, Traditional Arabic, Andalus, Naskh, and Thuluth. The data corpus consists of 116,743 words and 596,931 letters, not including spaces.

Al-Muhtaseb et al. [9] used a database of 2766 lines of Arabic text, consisting of 46,062 words totaling 224,109 characters, including spaces. They have generated the database for eight fonts: Arial, Tahoma, Akhbar, Thuluth, Naskh, Simplified Arabic, Andalus, and Traditional Arabic. Al-Hashim and Mahmoud reported a printed Arabic text database of 6954 pages [8]. The pages were scanned from different sources like book chapters, advertisements, magazines, newspapers, and reports, scanned with 200, 300, and 600 dpi resolutions.

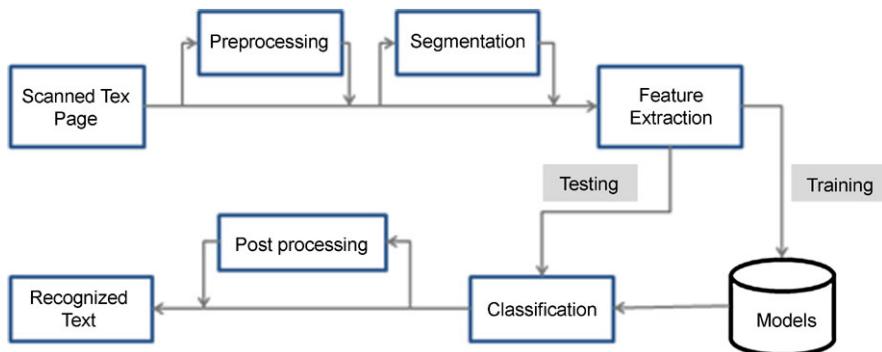


Fig. 7.3 Block diagram for an Arabic optical text recognition system

7.4 Optical Arabic Text Recognition

In this section, we present a general model for a printed Arabic text recognition system. We also discuss the different stages of the system, and we present different techniques used by researchers for each of the phases in the printed Arabic text recognition system.

7.4.1 General Model for an Arabic Optical Text Recognition System

The process of recognizing Arabic text can be broadly broken down into five stages: (1) pre-processing, (2) segmentation, (3) feature extraction, (4) classification, and (5) post-processing. Figure 7.3 shows the different stages of an Arabic optical text recognition system. In the following subsections, we discuss stages (1) to (4) in detail.

7.4.2 Pre-processing

Pre-processing focuses on enhancing the acquired image to increase the ease of feature extraction. It also aims to compensate for the eventual poor quality of the scanned documents [77]. Scanned pages differ in, e.g., quality, resolution, and source. Text images can acquire noise and/or be skewed when printed, handwritten, and/or scanned. The recognition accuracy of OCR systems greatly depends on the quality of the input text and noise, even more so than with humans. Baird [20] reports that even OCR methods that perform well on some images perform much worse on images that are only slightly harder for human readers.

When text is scanned and digitized, the raw data may carry a certain amount of noise. If the acquired image contains noise, it is subjected to a stage where “denoising” of the image takes place. Furthermore, when a document is fed to the scanner either mechanically or by a human operator, a few degrees of skew (tilt) is unavoidable. Skew correction aims at detecting the deviation of the document orientation angle from the horizontal or vertical direction. Moreover, text aligned along different directions is not uncommon. The subsequent stages of OCR systems mainly depend upon the accuracy of the pre-processing stage. For instance, if an OCR system underestimates or overestimates the skew angle, then the OCR system, which is utilizing projection-based techniques, will fail miserably.

In the pre-processing phase, we address image smoothing and skew correction. In the following sections, we discuss both of these issues in more detail.

Smoothing

The goal of smoothing is to remove or reduce the noise present in the scanned text image. An example of a noise removal algorithm is the statistical smoothing algorithm presented in [62]. The algorithm tries to eliminate small areas and to fill little holes that occur due to the regularization of the character contour [26]. This simple and efficient technique is based on a statistical decision criterion. Given a binary image of an Arabic text, the algorithm modifies each pixel based on its initial value and the values of its neighboring pixels (see Fig. 7.4). The rules are as follows:

$$\begin{aligned} \text{if } P_0 = 0 \\ \text{then } P'_0 = \begin{cases} 0, & \text{if } \sum_{i=1}^8 P_i < T \\ 1, & \text{otherwise} \end{cases} \\ \text{else } P'_0 = \begin{cases} 1, & \text{if } (P_i + P_{i+1}) = 2 \quad \text{for at least one } i = 1, \dots, 8 \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

Here, P_0 is the current pixel, P'_0 is the new pixel value, and T is the threshold. Experimentally, a threshold of 5 was found to yield acceptable results. Lower thresholds result in filling character holes and concave boundaries, consequently changing the topology of the characters. Higher thresholds result in very little or no smoothing.

Skew Estimation and Correction of Text

Various techniques have been proposed in the literature to estimate and correct the skew of document images. Comprehensive surveys can be found in [53, 54]. Most of the skew estimation techniques can be divided into classes according to the basic

Fig. 7.4 Pixel labeling scheme for statistical average-based smoothing

P4	P3	P2
P5	P0	P1
P6	P7	P8

approach they adopt [53, 54]. These approaches include projection profiles [5, 24, 25, 50, 60, 74], principal component analysis [84, 86], and Hough transforms.

The traditional projection profile approach was proposed by Postl [74]. In this approach, the input document is rotated through a range of angles, and a projection profile is calculated at each angle. Features are then extracted from each projection profile to determine the skew angle. This is computationally expensive, as it is performed directly on the original document images. Moreover, it is sensitive to the layout of the document image. Another projection profile approach was proposed by Bloomberg and Kopec [24], in which the original document image is downsampled before the projection profile is computed. Therefore, the image data to be processed is reduced, and the computational cost is reduced significantly. However, a major weakness is that its detection accuracy is influenced by the document image layout. It often fails on document images with multiple font styles and sizes or on those that contain a large amount of non-text regions.

The second class of skew correction is based on principal component analysis, in which the most significant eigenvector is calculated which leads to the skew angle of distribution. The problem with this method is that each eigenvector is constructed with support from projections of every point, which is expensive in terms of time. In addition, they are least squared estimation techniques and hence fail to account for outliers, which are common in images.

We now discuss an algorithm for skew correction that is based on finding the peaks and valleys to estimate the skew angle [79]. The baseline is the part of the script having the majority of black pixels. This property is used to allocate the baseline of the script. Then the tilt angle is found using the projection profile technique. The algorithm is summarized as follows.

1. The input image is divided into two vertical parts. Dividing the image into halves makes the lines more efficiently readable. Images having long lines may be divided into more than two parts. This enables the detection and correction of large skew angles.
2. The right half of the bisected input image is projected horizontally to get the sum of the black pixels in each row. The same procedure is repeated for the left half of the image.
3. Peaks and valleys are analyzed for each portion of the image.
4. The first valley of the right portion histogram shows the starting point of the first line. Similarly, the second valley of the left portion histogram shows the ending point of the first line.

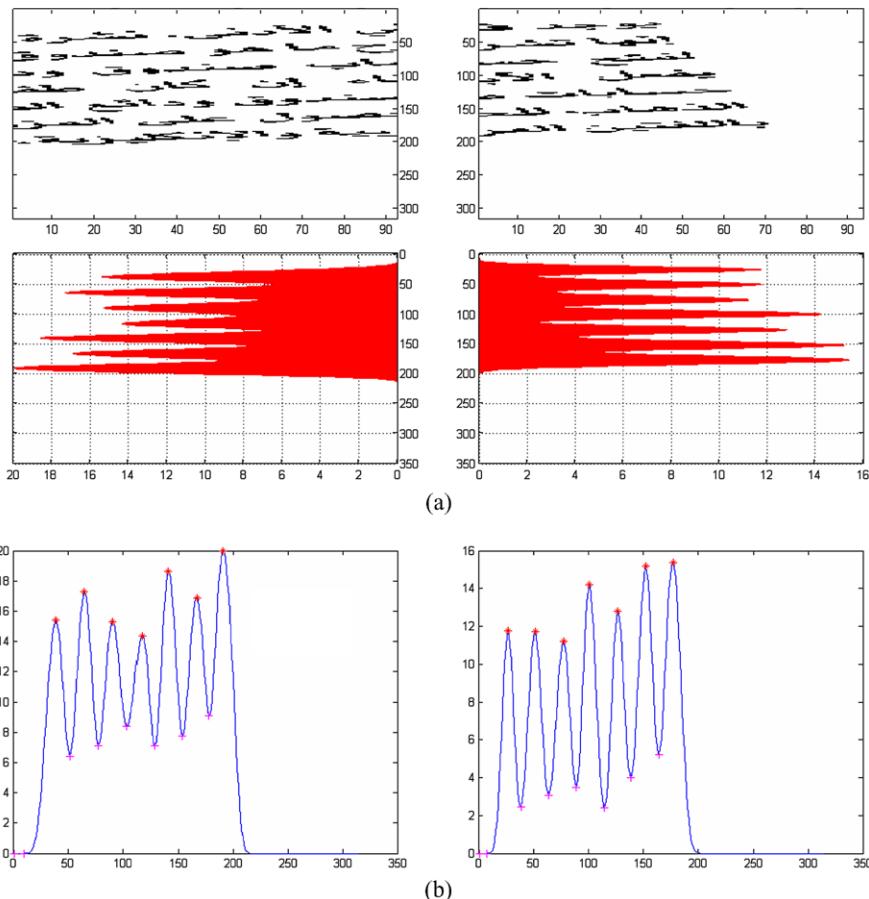


Fig. 7.5 Illustration of the skew correction algorithm. **(a)** The *top left* and *right* figures give the *left* and *right* halves of the image, respectively. The *lower left* and *right* figures give the projection profile histograms of the image for the *left* and *right* halves, respectively. **(b)** The peaks and valleys of the *left* and *right* halves of the images in **(a)**

5. The lines are projected from these points until they intersect at a point and then the perpendicular distance between these points is found.
6. Steps 4 and 5 are repeated for the remaining lines and the distance between the left and right projections of the line is calculated.
7. The skewed angles are calculated using trigonometric ratios.
8. The average of all the angles is taken.

Figure 7.5 illustrates the steps of the skew correction algorithm. Figure 7.5(a) exhibits Steps 1 and 2 of the algorithm. Step 3 of the algorithm is illustrated in Fig. 7.5(b). It is clear from the histogram in Fig. 7.5(b) that the peaks are well defined and prominent. We can easily find the orientation of the lines and thus the

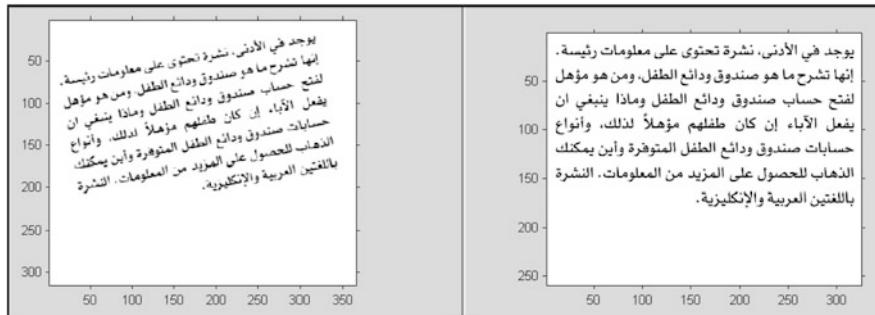


Fig. 7.6 Illustration of a skew corrected image of text. (*Left*) Before skew correction. (*Right*) After skew correction

skewed angle. The estimated angle is found to be 10.1 degrees. Figure 7.6 illustrates a skew corrected text image after applying the algorithm.

7.4.3 Segmentation Algorithms

After the pre-processing stage, most OCR systems for cursive text segment the page into lines. Then, depending on the technique used in the recognition, the lines are segmented to words and characters. Line segmentation problems can be formulated as follows: given an image of some text, line segmentation is the assignment of each component of the text to a particular group (line) that is intended to be read in sequence. The line segmentation helps in revealing the order of lines on a page, the order of words within a line, and the order of characters within words.

We now discuss an adaptive line segmentation algorithm designed for printed Arabic text pages, based on the algorithm in [40]. The algorithm uses horizontal projection (HP) to find possible cut points (PCPs) between lines. Horizontal projection is the number of foreground pixels per row. These projections have small counts between lines, which under some conditions, are judged as PCPs. PCPs are later refined to final cut points (FCPs). Every connected component (called a blob) in the page is then examined and associated with one line.

The main phases of the algorithm are as follows. Gray level images are binarized and then noise is removed by using median filtering. The horizontal projection is typically computed through one pass of the image, in which each row is examined and the number of foreground pixels is recorded. Lower values in the horizontal projection profile indicate possible locations of gaps (called valleys) between lines. However, a low value in the projection profile may be due to a small line or due to the large number of dots and diacritics. Therefore, to detect the valleys robustly, the following novel approach is used.

The horizontal projection profile of a binarized text image is considered as a two-dimensional curve C . In this curve, each point (x, y) corresponds to the row number

x with y being the number of the foreground pixels in that row. Note that curve C is an open curve. In curve C , points with low y values may indicate the locations of valleys. To avoid possible false valleys, the curve C goes through a process called collinear-points suppression [71]. Collinear-points suppression removes redundant points from a curve. A point P on the curve C is considered redundant if the distance from P to the line joining the two neighboring points of P on C is greater than some threshold d . We start with a small value of d (say 0.5). Then collinear-points suppression is applied iteratively on C , with increasing values of d at each iteration. The process is stopped when the length of the curve C reduces below some threshold. The y coordinates of the remaining points on C are taken as the new horizontal projection of the original image. This profile has fewer false valleys. Therefore, the detection of valleys can be done more reliably.

Now, we adaptively estimate a threshold to decide on the rows that are valleys. A local minimum is defined as a row containing less foreground pixels than both of its neighbors. An adaptive local threshold (LT) is then computed as the average pixel count of all local minima. We then tune the adaptive threshold by an estimation-maximization (EM) approach. The LT computation is repeated several times until convergence. Only a subset of the rows participates in calculating the LT at each iteration. The subset is recursively defined based on the current LT. Valleys are defined as the contiguous sequence of rows having an HP less than the LT. Within a valley, the row with the minimum HP is declared as a PCP. In the case of a tie, the center of the longest run of contiguous PCPs is taken as the PCP. The final cut points are marked using the statistical information of all the valleys. The average valley width (AVW) is computed from the located valleys. A global adaptive threshold (GT) is taken as half of the AVW. The GT is used to mark FCPs. Valleys narrower than the GT are merged with their nearest valleys.

Now, each blob in the image is associated to a line based on the y coordinate of its center of gravity (COG). All blobs having this coordinate value falling between two FCPs are assigned to the same line. This blob-wise approach sometimes results in associating different blobs intersecting an FCP to different lines.

The dashed bi-headed arrow in Fig. 7.7(a) demonstrates a valley. The vertical line crossing the projections is the LT. The thin solid arrow pointing toward the horizontal line that runs until mid-page represents a PCP. The thick solid arrow pointing toward the horizontal line that spans the whole width of the page shows an FCP. Figure 7.7(b) shows local maxima and local minima by enclosing them with solid and dashed circles, respectively.

7.4.4 Feature Extraction

In this section, we discuss different types of features used for printed Arabic text recognition. Many researchers have also used these features for Arabic character recognition.

Statistical features describe a pattern in terms of a set of characteristic measurements extracted from the pattern. Here, the pattern is represented as a fixed-length

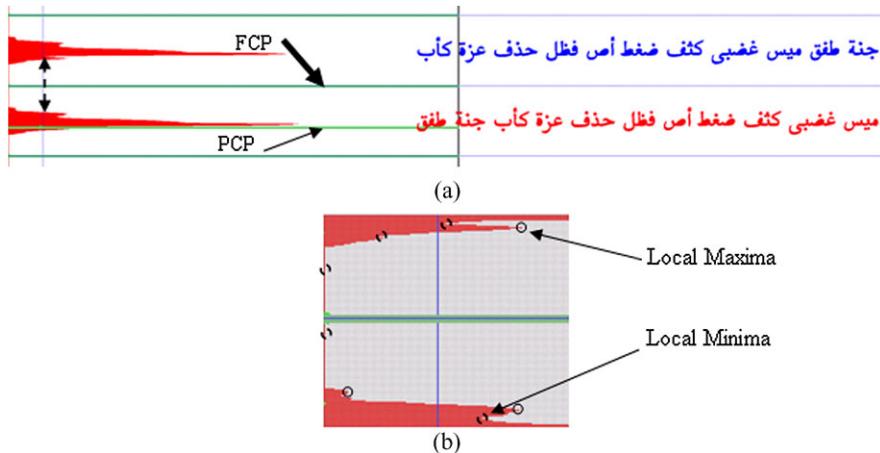


Fig. 7.7 Illustration of the line segmentation algorithm. (a) Example of PCP and FCP. (b) Enlarged part of (a) illustrating local minima and local maxima [40]

vector of ordered values and is interpreted as a point in a multidimensional space. Statistical features include zoning of the character array (i.e., dividing it into overlapping or non-overlapping regions) where the densities of pixels in these regions form the features, computing the moments of the black pixels of the character, the n-tuples of black or white or joint occurrence, the characteristic loci, and crossing distances. The set of features of all patterns defines the feature space of the recognition system. In an appropriately designed feature space, all patterns of the same class map to a unique partition of the space [33, 76].

Researchers of printed Arabic text recognition have used different statistical features. The work [13] used zoning of pixels as features. The characteristic loci method counts the number of zero/one segments a vertical line crosses in the pattern and the length of each segment [3, 43]. The crossing method counts the number of times a set of radial lines at different angles (e.g., 16 lines at 0, 22.5, 45, ... degrees) crosses the pattern [11]. This method tolerates distortions and small variations, and is fast to calculate [49]. The moment method is one of the most popular statistical approaches used for pattern recognition. The moments of a pattern about its center of gravity are invariant to translation and can be normalized to be invariant to rotation and scale [12, 33, 34, 76].

One of the simplest transformations is representing the skeleton or contour of a pattern as a chain of direction codes. Direction codes can correspond to the eight major directions as in Freeman codes [17, 19, 39, 62, 77], to six major directions in the case of hexagonal sampling [56, 57], or to unequal angle increments [44].

Fakir and Sodeyama use the Hough transform to represent the skeleton of a character as a set of line segments, and then use the length, location, and slope of the line segments as features [41]. Several researchers used Fourier descriptors, which are invariant to translation, rotation, and scaling and can tolerate moderate boundary variations, derived from the contour points of a segmented character

[10, 37, 38, 56, 57, 62]. In general, transformation schemes can be easily applied and tolerate noise and variation. However, they sometimes require the use of additional features in conjunction with Fourier descriptors to obtain high recognition rates [37, 56, 57, 62].

El-Wakil and Shoukry used a three-level classification scheme. While the first level is a dictionary lookup, the second is a 1-nearest neighbor classifier, and the third is a k-nearest neighbor [39]. One of the most critical issues in those methods is choosing an efficient and accurate distance or similarity measure.

Another common statistical method is to use Bayesian classification. A Bayesian classifier computes the a posteriori probability of each pattern class based on the detected features, the conditional probability of the features given a class, and the a priori probability of the class [1–3, 10, 12, 32].

Instead of examining all the features at once, decision tree classifiers arrange tests in a tree structure fashion. Each node of the tree is a test on a feature, and each outcome of the test leads to another node in the tree. The leaves of the tree are labeled with class identities. When a series of tests on a pattern leads to a leaf, the pattern is labeled with the label of the leaf [7, 14]. To improve accuracy, some systems use four decision trees, one for each connectivity form of a character (isolated, right-connected, etc.) [15, 16, 18].

The main advantage of statistical classifiers is that they can be automatically trained. The literature includes some simpler methods that do not fall under either paradigm, like dictionary lookup [33, 77], rule-based classification [70], and hand-crafted tree classifiers [35, 36].

HMM-Based Features

The general trend for cursive text recognition is to use the hidden Markov model (HMM). The use of other classifiers requires the segmentation of cursive text into characters, which is implicitly done by the HMM. In the following paragraphs, we discuss the use of HMMs in the recognition of printed Arabic text recognition.

Due to the advantages of HMMs, researchers have used them for speech and text recognition. HMMs offer several advantages. There is no need for segmenting the cursive text, HMMs are resistant to noise, they can tolerate variations in writing, and the HMM tools are freely available. Some researchers have used HMMs for handwritten word recognition [66, 72, 78], and others have used it for text recognition [9, 22, 28, 48]. HMMs have been used for off-line Arabic handwritten digit recognition [63] and for character recognition [31, 47].

The techniques used in [48] are based on extracting different types of features of each digit as a whole, not on the sliding window principles used by the majority of researchers using HMM. For their technique to be applicable to Arabic text recognition, it has to be preceded by a segmentation step which is error-prone. Bazzi et al. presented a system for bilingual text recognition (English/Arabic) [21, 22] using the sliding window principles and different types of features. Dehghani et al. used it for on-line handwritten Persian characters [31] and for handwritten Farsi (Arabic)

Fig. 7.8 Illustration of a seven-state, left-to-right HMM

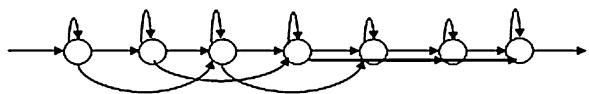


Fig. 7.9 Arabic writing line with line regions, sliding window, and feature segments

word recognition [30]. A variation of sliding window features based on hierarchical windows was used in [9].

In order to use HMMs, several researchers computed the feature vectors as a function of an independent variable [21, 22]. This simulates the use of HMMs in speech recognition where sliding frames/windows are used (see Fig. 7.9). The sliding window technique bypasses the need for segmenting Arabic text. The same technique is applicable to other languages. In this work, we utilize the HMM classifier as implemented in [42, 92]. However, we have defined our own HMM parameters and allowed transitions to the current, the next, and the following states only. This structure allows nonlinear variations in the horizontal position. We have used the Baum–Welch algorithm for training and the Viterbi algorithm for recognition, which searches for the most likely sequence of characters given the input feature vector.

We have used a left-to-right HMM for printed Arabic text recognition. Figure 7.8 illustrates a seven-state HMM, showing the allowed transition to the current, the next, and the following states only. This model allows relatively large variations in the horizontal position of the Arabic text. The sequence of state transition in the training and testing of the model is related to each text segment feature observation. In our work, we have experimented with using different numbers of states and dictionary sizes and have selected the best performing ones. Although each character model could have a different number of states, we have adopted the same number of states for all characters in a font. However, the numbers of states and dictionary sizes for each font, in relation to the best recognition rates for each font, are different.

To extract features from Arabic text, we have used the sliding window principle. A window with variable width and height has been used. We have experimented with horizontal and vertical overlapping windows, trying different values for the window width and height, and vertical and horizontal overlapping. Then different types of windows are utilized to obtain more features of each vertical segment and to decide on the most proper window size and the number of overlapping cells vertically and horizontally. The direction of the text line is considered as the feature extraction axis. In addition, different types of features are tested.

Starting from the first pixel of the text line image, a vertical segment of 3 pixels width and the height of the text line is taken. Note that the window size and vertical and horizontal overlapping are made settable, and hence different features may be extracted using different window sizes and vertical and horizontal overlapping.

A Markov model is a finite state machine that changes its state at each time (frame) unit (w). With each change of state (moving from state i to state j), a character vector O_w is generated from the probability density $b_j(o_f)$. Moreover, the transition from state i to state j is governed by the discrete probability a_{ij} . The start state and the final state of this model are non-emitting states to allow the building of composed models.

An HMM assumes that the sequence of observed text vectors, $O = o_1, o_2, \dots, o_w$, where o_w is the vector observation at frame w , representing each text line is generated by a Markov model. The probability of generating the text observation vector, O , by the model λ through the state sequence S is the product of the probabilities of the outputs and the probabilities of the transitions:

$$P(O, Q|\lambda) = \pi_1 b_1(o_1) a_{12} b_2(o_2) a_{23} b_3(o_3) \dots \quad (7.1)$$

where Q is the state sequence; $\lambda = (A, B, \pi)$; π_1 is the initial state transition; a_{ij} is the transition probability from state i to state j ; b_i is the output probability at state i . Both states i and j are between the model's first and last state, T .

As the state sequence is unknown, the probability is computed by summing over all possible state sequences. Since this is a time-consuming step, it is approximated by the following equation:

$$P(O|\lambda) = \max_Q \prod_{i=1}^T a_{q_{i-1}q_i} b_{q_i}(O_i) \quad (7.2)$$

where $Q = q_1, q_2, q_3, \dots$ is the state sequence of the model. This equation is usually computed through recursion with the assumption that the parameters a_{ij} and b_{ij} are known for each model λ_i . The model parameters are estimated in the training phase using the Baum–Welch algorithm. The sequence of states S that gives the highest probability is determined by the Viterbi algorithm.

Each text line image is represented by a sequence of text line vectors or observations. The recognition problem can be regarded as that of computing

$$\operatorname{argmax}_i \{P(C_i|O)\} \quad (7.3)$$

where C_i is the i^{th} character in the text line. This probability is computed using Bayes's rule:

$$P(C_i|O) = \frac{P(O|C_i)P(C_i)}{P(O)} \quad (7.4)$$

Thus, for a given prior probability $P(C_i)$ of each character, the most probable character depends only on the likelihood $P(O|C_i)$. The joint conditional probability $P(o_1, o_2, \dots | C_i)$ could be estimated by using a parametric model such as a Markov model. Hence, the problem of computing $P(O|C_i)$ is replaced by the problem of estimating Markov model parameters, which is a much simpler problem.

The probability of generating O by the model M through the state sequence S , $P(O, S|M) = P(O|C_i)$ is the product of the probabilities of the outputs and the probabilities of the transitions: $(O|C_i) = a_{12}b_2(o_1)a_{22}b_2(o_2)a_{23}b_3(o_3) \dots$. However, the state sequence S is unknown, and this is why the model is called the hidden Markov model. $P(O|C_i)$, now represented by $P(O|M)$, can be calculated as follows.

As the state sequence is unknown, the probability is computed by summing over all possible state sequences $S = s(1), s(2), s(3), \dots, s(F)$:

$$P(O|M) = \left\{ \sum_S a_{s(0)}a_{s(1)} \prod_{f=1}^F b_{s(f)}(o_f)a_{s(f)}a_{s(f+1)} \right\} \quad (7.5)$$

where $s(0)$ is the entry state and $s(f + 1)$ is the exit state. The last equation can be approximated as

$$\hat{P}(O|M) = \max_S \left\{ a_{s(0)}a_{s(1)} \prod_{f=1}^F b_{s(f)}(o_f)a_{s(f)}a_{s(f+1)} \right\} \quad (7.6)$$

This equation is usually computed by recursion with the assumption that the parameters a_{ij} and $b_j(o_f)$ are known for each model M_i .

The power of HMM appears here. Given a sufficient number of representative training examples of each character, the parameters of the model could be determined by a re-estimation procedure. The model represents implicitly different sources of variations inherited in character vectors.

7.5 Classification

In the classification stage, new printed Arabic text pages are scanned. These images pass through denoising in the pre-processing phase. Then, the line segmentation algorithm extracts the lines. From these lines, different features are extracted and sent to the classification phase. In the classification phase, the corresponding Arabic textual text is generated. This is the output of the system which corresponds to the scanned input data.

In the following subsections, we present some experimental results for the recognition of printed Arabic text.

7.5.1 Dataset

As mentioned before, there exists no general adequate database for printed Arabic text recognition that is freely available. The selected Arabic text is printed with different fonts and then scanned with 300 dots per inch resolution. These scanned text

Table 7.2 Important statistics on the training and testing dataset

	Number of Pages	Number of Lines	Number of Words	Number of Characters
Training	32	1000	12650	55529 (67145 with spaces)
Testing	8	258	3252	14723 (17717 with spaces)
Total	40	1258	15902	70252 (84862 with spaces)

images have been used to train and test the presented technique. We have scanned images of 40 printed Arabic text pages. These pages are segmented into lines, and this has resulted in 1258 lines. Out of these lines, 1000 have been used for training, and the remaining 258 have been used for testing. A summary of the important characteristics related to the dataset is provided in Table 7.2.

Experiments

We have conducted several experiments using different features extracted from these line images of Arabic text.

We have experimented with different types of features. All the features are based on the sliding window concept where a window frame of a given column height and width is used to extract features from the line image (the best result was obtained with 8 pixels height and 3 pixels width). This window frame is moved from the top of the image to the bottom and from the beginning of the text line to the end, with optional overlaps both horizontally and vertically. For every frame, the average number of ink pixels is calculated and saved as a feature.

A modification to the above features is the use of the derivations of horizontal and vertical edge of the text line image. The Sobel operator is used for edge detection of the text line images. Thus, we have extracted features from the original image as well as from the horizontal and vertical edge derivatives of the image.

In addition, a hierachal sliding window feature as presented in [9] has also been used. Once the features are extracted, they are quantized into different numbers of clusters and experimented with using the HMM classifier. Moreover, HMM models with different numbers of states have also been tested.

We have obtained a correctness rate of 73.78 % and an accuracy rate of 71.74 % for window features using horizontal and vertical edge derivatives of the image. Correctness accounts for errors due to substitution and deletions but does not consider insertion errors, whereas accuracy does consider insertion errors along with substitution and deletion errors. Using the hierachal window features, the best results obtained are a correctness rate of 81.05 % and an accuracy rate of 78.04 %.

Some of the reasons for the low recognition rates are the presence of skew in the segmented lines and the poor quality of the images. In addition some words in the text line have different writing baselines.

To improve the low recognition rate, a new feature extraction algorithm which is adaptive in nature and has a variable window size is devised. The line level skew is

corrected using the writing line as a reference. The new algorithm takes care of even minor skews at the line level. Additionally, the window size and positions are made adaptive to the writing line of the Arabic text. The windows are of variable size to account for differences in importance for different regions of the Arabic text line. Using this new feature extraction algorithm we have obtained a correctness of close to 98 %, a significant improvement compared to the best result of 81.45 % using the hierarchical features.

7.6 Conclusions

A robust printed Arabic text recognition system can be very useful for improving the man-machine interface. It can enable the conversion of huge amounts of scanned printed Arabic documents into editable form. In this chapter, we have discussed a general framework for the recognition of printed Arabic text. The different phases of a printed Arabic text recognition system are presented as well as examples of the techniques used in each phase. We have discussed pre-processing, feature extraction, and classification techniques. In addition, we have presented a concise theory behind the hidden Markov model (HMM) and some experimental results for HMM-based printed Arabic text recognition. The experimental results are encouraging. However, still more effort is needed to improve the method and make it practical.

References

1. Abdel-Azim, H.Y., Hashish, M.A.: A hidden Markov modelling approach to the recognition of signatures: a feasibility study. In: Proceedings of the First Kuwait Computer Conference, March 1989, pp. 402–425 (1989)
2. Abdel-Azim, H.Y., Hashish, M.A.: Automatic recognition of handwritten Hindi numerals. In: Proceedings of the 11th Saudi National Computer Conference, March 1989, pp. 287–298 (1989)
3. Abdel-Azim, H.Y., Mousa, A.M., Saleh, Y.L., Hashish, M.A.: Arabic text recognition using a partial observation approach. In: Proceedings of the 12th National Computer Conference, October 1990, pp. 427–437 (1990)
4. Abdelraouf, A., Higgins, C.A., Khalil, M.: A database for Arabic printed character recognition. In: Proceedings of the 5th International Conference on Image Analysis and Recognition (ICIAR), pp. 567–578 (2008)
5. Akiyama, T., Hagita, N.: Automatic entry system for printed documents. *Pattern Recognit.* **23**(11), 1141–1154 (1990)
6. Al-Badr, B., Mahmoud, S.A.: Survey and bibliography of Arabic optical text recognition. *Signal Process.* **41**(1), 49–77 (1995)
7. Al-Emami, S., Usher, M.: On-line recognition of handwritten Arabic characters. *IEEE Trans. Pattern Anal. Mach. Intell.* **12**(7), 704–710 (1990)
8. Al-Hashim, A.G., Mahmoud, S.A.: Benchmark database and GUI environment for printed Arabic text recognition research. *WSEAS Trans. Inf. Sci. Appl.* **4**(7), 587–597 (2010)
9. Al-Muhtaseb, H.A., Mahmoud, S.A., Qahwaji, R.: Recognition of off-line printed Arabic text using hidden Markov models. *Signal Process.* **88**(12), 2902–2912 (2008)

10. Al-Qaisy, E.K., Naser, H.L.: Using probabilistic functions for the recognition of handwritten Arabic numerals. In: Proc. of the First Kuwait Computer Conference (in Arabic), pp. 109–120 (1989)
11. Al-Tikriti, M.N., Al-Ramchi, S.K.: Fuzzy approach for some Arabic handwritten characters computer recognition. In: Proc. Comput. Processing Arabic Language Workshop Papers, pp. 1–14 (1985)
12. Al-Yousefi, H., Udupa, S.: Recognition of handwritten Arabic characters. In: Proc. of the SPIE 32nd Annual Technical Symposium on Optical and Opto-Electronics Applied Science and Engineering, San Diego, CA, USA, vol. 974, pp. 330–336 (1988)
13. Ali, S.A., Al-Saadoun, M.S.: A parallel algorithm for image thinning. In: Proc. of the First Kuwait Computer Conference (in Arabic), Kuwait, pp. 121–140 (1989)
14. Amin, A.: Arabic handwriting recognition and understanding. In: Proc. of Computer Processing and Transmission of the Arabic Language Workshop, Kuwait, vol. 1, pp. 1–37 (1985)
15. Amin, A.: OCR of Arabic texts. In: Proc. of the 9th Int. Conf. on Pattern Recognition, pp. 616–625. University of Cambridge, Cambridge (1988)
16. Amin, A., Al-Fedaghi, S.: Machine recognition of printed Arabic text utilizing natural language morphology. *Int. J. Man-Mach. Stud.* **35**(6), 769–788 (1991)
17. Amin, A., Mari, J.F.: Machine recognition and correction of printed Arabic text. *IEEE Trans. Syst. Man Cybern.* **19**(5), 1300–1306 (1989)
18. Amin, A., Masini, G.: Machine recognition of multifont printed Arabic texts. In: Proc. of the 8th IEEE Int. Joint Conf. on Pattern Recognition, Paris, France, pp. 392–395 (1986)
19. Badie, K., Shimura, M.: Machine recognition of Arabic handprinted scripts. *Trans. Inst. Electron. Commun. Eng. Jpn., Sect. E* **65**(2), 107–114 (1982)
20. Baird, H.S.: Calibration of document image defect models. In: Proceedings of the 2nd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, pp. 1–16 (1993)
21. Bazzi, I., LaPre, C., Makhoul, J., Raphael, C., Schwartz, R.: Omnifont and unlimited-vocabulary OCR for English and Arabic. In: Proceedings of the 4th International Conference on Document Analysis and Recognition, August 1997, vol. 2, pp. 842–846 (1997)
22. Bazzi, I., Schwartz, R., Makhoul, J.: An omnifont open-vocabulary OCR system for English and Arabic. *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(6), 495–504 (1999)
23. Ben Amor, N., Ben Amara, N.E.: A hybrid approach for multifont Arabic characters recognition. In: Proc. 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, pp. 194–198 (2006)
24. Bloomberg, D.S., Kopec, G.E.: Method and Apparatus for Identification and Correction of Document Skew (1993)
25. Bloomberg, D.S., Kopec, G.E., Dasari, L.: Measuring document image skew and orientation. In: Proc. SPIE, vol. 2422, 302–316 (1995)
26. Borghesi, P.: Digital image processing techniques for object recognition and experimental results. In: Proceedings of The Digital Signal Processing Conf., Florence, Italy, pp. 764–768 (1984)
27. Bozinovic, R., Srihari, S.: Off-line cursive script word recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **11**(1), 68–83 (1989)
28. Bunke, H., Bengio, S., Vinciarelli, A.: Off-line recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**, 709–720 (2004)
29. Davidson, R.B., Hopley, R.L.: Arabic and Persian OCR training and test data sets. In: Proc. SDIUT, Annapolis, MD, pp. 303–307 (1997)
30. Dehghan, M., Faeza, K., Ahmadi, M., Shridhar, M.: Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM. *Pattern Recognit.* **34**(5), 1057–1065 (2001)
31. Dehghani, A., Shabani, F.: Off-line recognition of isolated Persian handwritten characters using multiple hidden Markov models. In: International Conference on Information Technology: Coding and Computing, April 2001, pp. 506–510 (2001)
32. Duda, R.O., Hart, P.E.: *Pattern Classification and Scene Analysis*. Wiley, New York (1973)

33. El-Dabi, S.S., Ramsis, R., Kamal, A.: Arabic character recognition system: a statistical approach for recognizing cursive typewritten text. *Pattern Recognit.* **23**(5), 485–495 (1990)
34. El-Khaly, F., Sid-Ahmed, M.A.: Machine recognition of optically captured machine printed Arabic text. *Pattern Recognit.* **23**(11), 1207–1214 (1990)
35. El-Sheikh, T.S.: Recognition of handwritten Arabic mathematical formulas. In: Proc. of the UK IT 1990 Conf., pp. 344–351. University of Southampton, Southampton (1990)
36. El-Sheikh, T.S., El-Taweel, S.G.: Real-time Arabic handwritten character recognition. In: Proc. of the Third Int. Conf. on Image Processing and Its Applications, Warwick, UK, pp. 212–216 (1989)
37. El-Sheikh, T.S., Guindi, R.M.: Automatic recognition of isolated Arabic characters. *Signal Process.* **14**(2), 177–184 (1988)
38. El-Sheikh, T.S., Guindi, R.M.: Computer recognition of Arabic cursive scripts. *Pattern Recognit.* **21**(4), 293–302 (1988)
39. El-Wakil, M.S., Shoukry, A.A.: On-line recognition of handwritten isolated Arabic characters. *Pattern Recognit.* **22**(2), 97–105 (1989)
40. Elarian, Y., Mahmoud, S.A.: An adaptive line segmentation algorithm (ALSA) for Arabic. In: Proc. International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV'08), pp. 735–739 (2008)
41. Fakir, M., Sodeyama, C.: Recognition of Arabic printed scripts by dynamic programming matching method. *IEICE Trans. Inf. Syst.* **E76-D**(2), 235–242 (1993)
42. Farah, N., Ennaji, A., Khadir, T., Sellami, M.: Benefits of multi-classifier systems for Arabic handwritten words recognition. In: Proc. Intl Conf. Document Analysis and Recognition, pp. 222–226 (2005)
43. Fayek, M.B., Al-Basha, B.: A new hierarchical method for isolated typewritten Arabic character classification and recognition. In: Proc. of the 13th Nat. Computer Conf., Riyadh, Saudi Arabia, pp. 750–760 (1992)
44. Goraine, H., Usher, M., Al-Emami, S.: Off-line Arabic character recognition. *Computer* **25**(7), 71–74 (1992)
45. Govindan, V.K., Shivaprasad, A.P.: Character recognition—a review. *Pattern Recognit.* **23**(7), 671–683 (1990)
46. Graff, D., Chen, K., Kong, J., Maeda, K.: Arabic Gigaword, 2nd edn. Linguistic Data Consortium, University of Pennsylvania, Philadelphia (2006)
47. Hassin, A., Tang, X., Liu, J., Zhao, W.: Printed Arabic character recognition using HMM. *J. Comput. Sci. Technol.* **19**(4), 538–543 (2004)
48. Hu, J., Lim, S., Brown, M.: Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognit.* **33**, 133–147 (2000)
49. Impedovo, S., Ottaviano, L., Occhinegro, S.: Optical character recognition—a survey. *Int. J. Pattern Recognit. Artif. Intell.* **5**(1), 1–24 (1991)
50. Ishitani, Y.: Document skew detection based on local region complexity. In: Proc. 2nd Internat. Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan, pp. 49–52 (1993)
51. Jain, A.K., Duin, R.P.W., Mao, J.: Statistical pattern recognition: a review. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(1), 4–37 (2000)
52. Jenkins, F., Kanai, J.: A keyword-indexed bibliography of character recognition and document analysis (revision 2.0). Technical Report TR-93-07, Information Science Research Institute, University of Nevada, Las Vegas, April 1993
53. Jonathan, J.H.: Document image skew detection: survey and annotated bibliography. In: Document Analysis Systems, pp. 40–64. World Scientific, Singapore (1998)
54. Jung, K., Kim, K.I., Jain, A.K.: Text information extraction in images and video: a survey. *Pattern Recognit.* **37**(5), 977–997 (2004)
55. Kasturi, R., O’Gorman, L.: Document image analysis: a bibliography. *Mach. Vis. Appl.* **5**, 231–243 (1992)
56. Khella, F.: Analysis of hexagonally sampled images with application to Arabic cursive text recognition. Ph.D. thesis, University of Bradford, Bradford, England (1992)

57. Khella, F., Mahmoud, S.A.: Recognition of hexagonally sampled Arabic characters. *Arab. J. Sci. Eng.* **19**(4A), 565–586 (1994)
58. Khorsheed, M.S.: Mono-font cursive Arabic text recognition using speech recognition system. In: Structural, Syntactic, and Statistical Pattern Recognition. Lecture Notes in Computer Science, vol. 4109, pp. 755–763 (2006)
59. Khorsheed, M.S.: Offline recognition of omnifont Arabic text using the HMM ToolKit (HTK). *Pattern Recognit. Lett.* **28**(12), 1563–1571 (2007)
60. Liolios, N., Fakotakis, N., Kokkinakis, G.: On the generalization of the form identification and skew detection problem. *Pattern Recognit.* **35**, 253–264 (2002)
61. Liu, C.-L., Jaeger, S., Nakagawa, M.: Online recognition of Chinese characters: the state-of-the-art. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(2), 198–213 (2004)
62. Mahmoud, S.A.: Arabic character recognition using Fourier descriptors and character contour encoding. *Pattern Recognit.* **27**(6), 815–824 (1994)
63. Mahmoud, S.A.: Recognition of writer-independent off-line handwritten Arabic (Indian) numerals using hidden Markov models. *Signal Process.* **27**(6), 815–824 (2008)
64. Mantas, J.: An overview of character recognition methodologies. *Pattern Recognit.* **19**(19), 425–430 (1986)
65. McClelland, D.: OCR: teaching your Mac to read. *Macworld November*, 169–178 (1991)
66. Mohamed, M., Gader, P.: Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. *IEEE Trans. Pattern Anal. Mach. Intell.* **18**(5), 548–554 (1996)
67. Mori, S., Yamamoto, K.Y., Yasuda, N.: Research on machine recognition of handprinted characters. *IEEE Trans. Pattern Anal. Mach. Intell.* **6**(4), 386–405 (1984)
68. Mori, S., Suen, C.Y., Yamamoto, K.: Historical review of OCR research and development. *Proc. IEEE* **80**(7), 1029–1057 (1992)
69. Nouboud, F., Plamondon, R.: On-line recognition of handprinted characters: survey and beta tests. *Pattern Recognit.* **23**(9), 1031–1044 (1990)
70. Nurul-Ula, A., Nouh, A.S.: Automatic recognition of Arabic characters using logic statements—part I: system description and pre-processing. *J. King Saud Univ., Eng. Sci.* **14**(2), 343–353 (1988)
71. Parvez, M.T., Mahmoud, S.A.: Polygonal approximation of digital planar curves through adaptive optimizations. *Pattern Recognit. Lett.* **31**(13), 1997–2005 (2010)
72. Pechwitz, M., Märgner, V.: HMM based approach for handwritten Arabic word recognition using the IFN/ENIT-database. In: Proc. Seventh International Conference on Document Analysis and Recognition (ICDAR), Edinburgh, Scotland, August 2003, pp. 890–894 (2003)
73. Plamondon, R., Srihari, S.N.: On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(1), 63–82 (2000)
74. Postl, W.: Detection of linear oblique structures and skew scan in digitized documents. In: Proc. 8th Internat. Conf. on Pattern Recognition, Paris, France, pp. 687–689 (1986)
75. Prasad, R., Saleem, S., Kamali, M., Meermeier, R., Natarajan, P.: Improvements in hidden Markov model based Arabic OCR. In: Proc. 19th International Conference on Pattern Recognition (ICPR) (2008)
76. Ramsis, R., El Dabi, S.S., Kamel, A.: Arabic character recognition system. Report KSC027, IBM Kuwait Scientific Center, Kuwait (1988)
77. SaadAllah, S., Yacu, S.G.: Design of an Arabic character reading machine. In: Proc. of Computer Processing and Transmission of the Arabic Language Workshop, Kuwait (1985)
78. Safabakhsh, R., Adibi, P.: Nastaaligh handwritten word recognition using a continuous-density variable-duration HMM. *Arab. J. Sci. Eng.* **30**, 95–118 (2005)
79. Sarfraz, M., Mahmoud, S.A., Rasheed, Z.: On skew estimation and correction of text. In: Proc. Computer Graphics, Imaging and Visualisation (CGIV), Bangkok, Thailand, pp. 308–313 (2007)
80. Schlosser, S.: ERIM Arabic Document Database. <http://documents.cfar.umd.edu/resources/database/> (1995). Environmental Research Institute of Michigan (ERIM)
81. Simon, J.-C.: Off-line cursive word recognition. *Proc. IEEE* **80**(7), 1150–1161 (1992)

82. Slimane, F., Ingold, R., Alimi, A.M., Hennebert, J.: Duration models for Arabic text recognition using hidden Markov models. In: Proc. of the International Conferences on Computational Intelligence for Modelling, Control and Automation; Intelligent Agents, Web Technologies and Internet Commerce; and Innovation in Software Engineering, pp. 838–843 (2008)
83. Slimane, F., Ingold, R., Kanoun, S., Alimi, A.M., Hennebert, J.: A new Arabic printed text image database and evaluation protocols. In: Proc. 10th International Conference on Document Analysis and Recognition, pp. 946–950 (2009)
84. Smith, L.I.: A Tutorial on Principal Components Analysis. Cornell University, Ithaca (2002)
85. Stallings, W.: Approaches to Chinese character recognition. *Pattern Recognit.* **8**, 87–98 (1976)
86. Steinherz, T., Intrator, N., Rivlin, E.: Skew detection via principal components analysis. In: Proc. 5th International Conference on Document Analysis and Recognition (ICDAR), pp. 153–156 (1999)
87. Suen, C.Y., Berthod, M., Mori, S.: Automatic recognition of handprinted characters—the state of the art. *Proc. IEEE* **68**(4), 469–487 (1980)
88. Tappert, C.C., Suen, C.Y., Wakahara, T.: The state of the art in on-line handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **12**(8), 787–808 (1990)
89. Tian, Q., Zhang, P., Alexer, T., Kim, Y.: Survey: omnifont printed character recognition. In: Visual Communications and Image Processing, pp. 260–268 (1991)
90. Wakahara, T., Murase, H., Odaka, K.: On-line handwriting recognition. *Proc. IEEE* **80**(7), 1181–1194 (1992)
91. Welch, E.M.: Can you read this? OCR software. *MacUser* **9**(8), 169–178 (1993)
92. Young, S., Jansen, J., Odell, J., Ollason, D., Woodland, P.: The HTK Book (HTK Version 3.4). Cambridge University Engineering Department, Cambridge (2006)

Chapter 8

Handwritten Arabic Word Recognition Using the *IFN/ENIT-database*

Mario Pechwitz, Haikal El Abed, and Volker Märgner

Abstract A well-structured and comprehensive dataset is the most important part in the development of a handwritten word recognizer. The *IFN/ENIT-database*, a well-organized set of images of Arabic handwritten town names, is widely used as a basis to develop handwritten Arabic word recognition systems. We describe in detail the *IFN/ENIT-database*, the form used to collect the data, the ground truth information, and the statistics of the data. The recognizer developed using this database is presented in the main part of this contribution. The pre-processing of the name images, e.g., baseline estimation, normalization, and feature extraction as well as the hidden Markov model (HMM)-based recognizer together with the results achieved are presented and discussed in detail in this chapter.

8.1 Introduction

Automatic recognition of handwritten words still remains a challenging task. Especially concerning the automatic recognition of Arabic handwritten text, a lot of work still has to be done. In the case of recognizing scanned images of written words we talk of an offline recognition system, as opposed to writing with a stylus on a device like a tablet and recognizing the stream of coordinates, which is called an online recognition system. The topic of our research is offline word recognition. The most important requirement for the development and comparison of recognition systems is a large database combined with ground truth (GT) information. The *IFN/ENIT-database*, published at the CIFED 02 conference [25], was the first dataset of Arabic handwritten name images that was freely available for university research purposes, and it is used today by approximately all groups working on Arabic word recognition. Additionally these data were used to compare recognition systems in

M. Pechwitz · H. El Abed · V. Märgner (✉)

Institute for Communications Technology (IfN), Technische Universität Braunschweig,
38106 Braunschweig, Germany
e-mail: v.maergner@tu-bs.de

H. El Abed
e-mail: elabed@tu-bs.de

competitions organized during the International Conference on Document Analysis and Recognition (ICDAR) starting in the year 2005 until 2011. During that time a continual improvement of the recognition results could be observed (see Chap. 17: Arabic Handwriting Recognition Competition).

Usually in the case of cursive handwriting the characters of words are connected, and this is exactly what makes it impossible to easily adapt recognition methods for printed words to handwritten words. A solution to the problem of connected characters is the methods based on hidden Markov models (HMMs), which have been very successfully used for recognizing spoken words and have recently also been adapted to handwritten words. A knowledge of the character set used and the writing style variation of different writers is very important for the development not only of the recognition system but also of the pre-processing and feature extraction. The different shapes of the Arabic characters compared to Latin characters makes character- and language-specific processing modules very important. Not only pre-processing and normalization but also the HMM recognizer must be adopted to the Arabic handwriting style, along with post-processing that uses language-dependent syntax and semantics.

8.1.1 Characteristics of the Arabic Script

For the reader who is not familiar with Arabic writing a short overview of the most important characteristics of Arabic is given. Arabic is written from right to left, and the characters within a word in the printed and handwritten versions are connected on a baseline, but six characters are an exception. If one of these characters appears in a word, the word is split into two parts (we call it PAW: part of Arabic word). If more than one of these characters appear in a word, it breaks into more than two parts. Arabic uses 28 characters which are not case sensitive; upper and lower cases are not known. The shape of an Arabic character depends on the position of the character. The shape of an isolated character may differ from the same character written in the beginning, the middle, or the end of a word. Additionally, to the basic shape of a character one to three dots may appear below, above, or in the middle of the character, and diacritical signs like Hamza or Madda are also used. Tables 8.1 and 8.2 show examples of character shapes and some characters with diacritical signs in printed Arabic. As an example of written words, Fig. 8.1 shows a name from the *IFN/ENIT-database* printed and written by two different writers.

8.1.2 Overview of the Recognition System

The general concept of the online part of a word recognition system is shown in Fig. 8.2. The system starts with the conversion of the paper document into a digital form, usually grayscale or color image data. In the next step pre-processing tasks

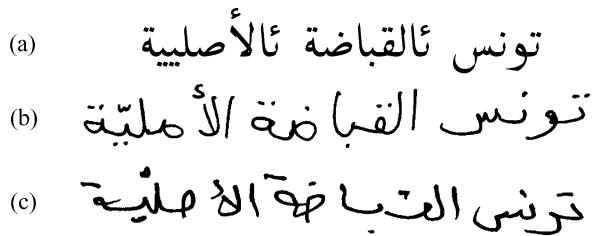
Table 8.1 The Arabic alphabet: 28 characters and up to four different forms; six letters exist only in the insulated form and in the end form (appropriate fields are left empty)

Character	Isolated	End	Middle	Begin	Character	Isolated	End	Middle	Begin
Alif	أ	ـ			Dhad	ض	ضـ	ضـ	ضـ
Ba	ب	بـ	بـ	بـ	Taa	طـ	طـ	طـ	طـ
Ta	تـ	تـ	تـ	تـ	Dha	ظـ	ظـ	ظـ	ظـ
Tha	ثـ	ثـ	ثـ	ثـ	Ayn	عـ	عـ	عـ	عـ
Jim	جـ	جـ	جـ	جـ	Ghayn	غـ	غـ	غـ	غـ
Ha	حـ	حـ	حـ	حـ	Fa	فـ	فـ	فـ	فـ
Kha	خـ	خـ	خـ	خـ	Qaf	قـ	قـ	قـ	قـ
Dal	دـ	دـ			Kaf	كـ	كـ	كـ	كـ
The	ذـ	ذـ			Lam	لـ	لـ	لـ	لـ
Ra	رـ	رـ			Mim	مـ	مـ	مـ	مـ
Zai	زـ	زـ			Nun	نـ	نـ	نـ	نـ
Sin	سـ	سـ	سـ	سـ	He	هـ	هـ	هـ	هـ
Chin	شـ	شـ	شـ	شـ	Waw	وـ	وـ	وـ	وـ
Sad	صـ	صـ	صـ	صـ	Ya	يـ	يـ	يـ	يـ

Table 8.2 Selection of special Arabic characters, together with the supplements “*Hamza*” and “*Madda*.” Additionally the common ligature *LamAlif* and the “*Ta’marbuta*” are presented

like noise reduction, segmentation, and binarization (conversion of the input image into a bilevel image) are performed. As the system should deal with a large number of different unknown writers, the next two blocks normalize the writing style with the goal to make it more robust against size, slant, skew, and line width variations of a word. The next block extracts a number of features from the normalized word image. An important precondition for this step is the estimation of baselines of each word to make the feature extraction more effective. Based on models and their parameters, which were fixed during an offline training process, and a word lexicon, the recognition process is performed by searching for models that fit best with a

Fig. 8.1 Tunisian town/village (تونس القباضة الأصلية) in three styles: printed Arabic using the Naskh style (a); handwritten by two different writers (examples from the IFN/ENIT-database) (b), (c)



given feature vector sequence. The output of the recognizer is one or more word hypotheses.

8.1.3 Pre-processing and Normalization

A very important part of the recognition system is obviously the pre-processing and normalization. Any error occurring in this part of the system may result in an unrecoverable recognition error. The first step is the noise reduction and the segmentation and binarization. These system parts are not discussed in this chapter. Here we start with the binary image of a name. Figure 8.3 shows exemplarily the normalization steps for one handwritten name. Based on a connected component analysis, the skew, slant, and baselines are estimated on the basis of the respective name. Using these parameters the name is normalized and finally converted to a grayscale image which is used to calculate the corresponding feature vector sequence. In the following we present the details of our recognition system focusing on the data (Sect. 8.2.1), the baseline and topline estimation (Sect. 8.3), normalization and feature extraction (Sect. 8.4), and the recognizer (Sect. 8.5). We end by presenting and discussing the results achieved (Sect. 8.6).

8.2 IFN/ENIT-database: Database of Arabic Handwritten Words

8.2.1 Overview of the Database Conception

Most recognition systems for handwritten words which are in use today are developed for applications with a restricted lexicon of words. These systems are focused on certain applications, such as the reading of check amounts or postal addresses, which are proven to be realistic and profitable. The further development of recognition systems, however, requires a large amount of data to train and test the system, especially if statistical methods like HMM are used. The implementation of a system requires real-world data, but data from the bank or the postal system are often confidential and inaccessible for non-commercial research. As the amount of data is crucial for a reliable training of recognition systems, we decided to use similar

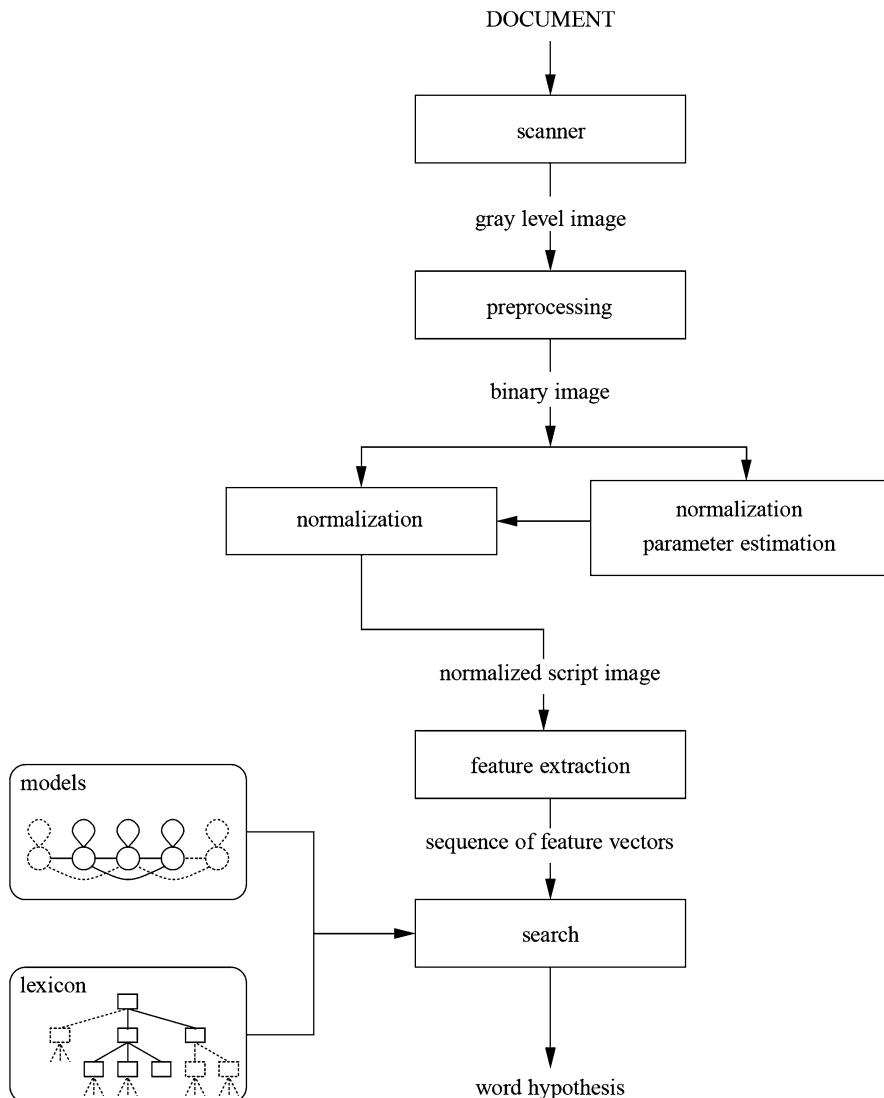


Fig. 8.2 Block diagram of the HMM-based recognition system

artificial data collected on special forms instead of scarce real-world data. Despite the disadvantage of using artificial data, the process to produce obligatory ground truth (GT) from the data is made much simpler due to the fact that the forms can be adopted to the automatic labeling process.

For a test environment, we chose a scenario related to postal address reader applications [24, 25]. We decided to collect handwritten data from the 946 Tunisian

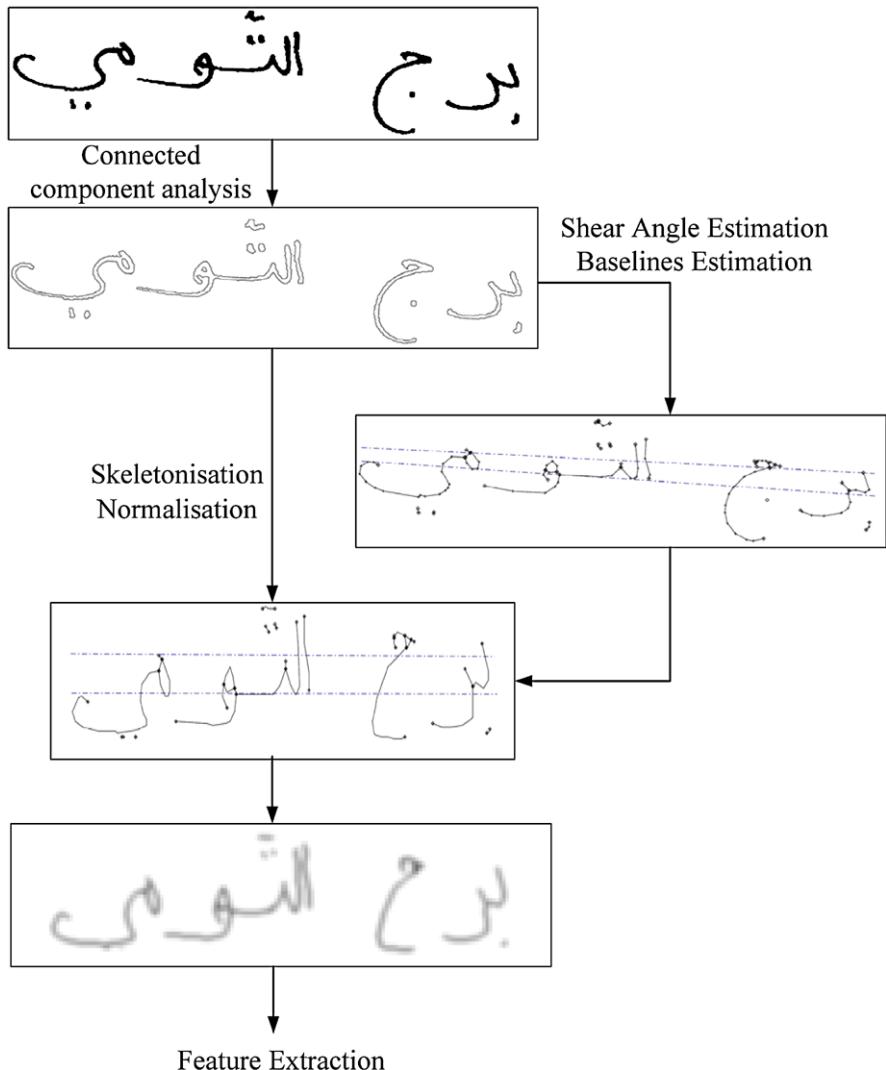


Fig. 8.3 Example of the pre-processing and normalization steps

towns/villages¹ together with their postcodes. Most of the people who contributed to the *IFN/ENIT-database* were familiar with the vocabulary, because they belong to the narrower range of the École Nationale d'Ingénieurs de Tunis (ENIT) in Tunisia.

¹From the 946 Tunisian town/village names we obtain only 937 really different words (some have only a different postcode). The classification task has to deal with a middle-sized lexicon.

In the following we describe in more detail how we collected and processed the data (Sect. 8.2.2). We also present the available ground truth (Sect. 8.2.3) and show some statistics for the database (Sect. 8.2.4).

8.2.2 Development and Realization

Our aim was to collect Tunisian handwritten town/village names written in a similar way as town names of an address on a letter. The form was designed to:

- Encourage writing without strong constraints
- Collect writing similar to writing on a letter
- Be easy to process automatically
- Provide additional information about the person who completed it

Based on these aims we developed a form for collecting handwritten data.

The Form

A completed example of this form is shown in Fig. 8.4. The form consists of three columns and a text block at the bottom. Embodied in the column on the right-hand side of the form are 12 lines with printed Tunisian town/village names and their respective postcodes, which are automatically selected² from the possible 937 names. The sample writers were expected to write the postcode in the left column and the town/village name in the middle column in their individual writing style. We did not print a line to write on or a box to write in, since we wanted to make the processing of the scanned data as simple as possible. To provide a light writing guidance we printed dark black rectangles on the backside of each page, which are shining through to the front side and thus mark roughly where to write. In the scanning process these rectangles can be removed using a simple threshold. Further segmentation operations are not necessary. A page number is used as a form identifier for the subsequent processing. In the block at the bottom additional information about age, profession, and identity of the writer is given. Each writer was asked to complete 5 forms, so one had to write up to 60 names.

Form Processing

All form pages were scanned with 300 dpi and converted to black and white (binary) images. Due to the fact that the paper was white and the words were written with a

²The criteria for the selection are explained in Sect. 8.2.4.

CODE ↓	PLACE ↓	
6132	حِمَّام بِيَاعَةٍ	6132 حمام بياعة
2056	رِوَاد	2056 رواد
2014	مُقْرَبَات الرِّيَاض	2014 مقربات الرياض
4283	نَفَّةٌ	4283 نفة
2064	جِبَل الرِّعَامِ	2064 جبل الرعاص
1200	الْعَصَمِين	1200 العصمين
7030	سَاطِر	7030 ساطر
1251	الشَّرَابِيع	1251 الشرياب
3233	فَطْوَفَةٌ	3233 فطوفة
2112	سَيِّدِي ١ حَمْدَ زَرْقَ	2112 سيدي إحمد زرق
1110	الْمَرْنَاقِيَّةٌ	1110 المرناقية
2261	سَبْعَة آبَار	2261 سبعه آبار
<u>Age:</u> 20 < <input type="checkbox"/> 21 - 30 <input checked="" type="checkbox"/> X 31 - 40 <input type="checkbox"/> 40 > <input type="checkbox"/>	<u>Profession :</u> Étudiant/élève <input checked="" type="checkbox"/> Enseignant <input type="checkbox"/> Administratif <input type="checkbox"/> Autre <input type="checkbox"/>	<u>Nom:</u> <input type="text" value="Noulat Nizar"/> <u>Ville:</u> <input type="text" value="Ariana"/>
<u>Responsable:</u> <input type="text" value="Samia Sf"/>		<u>Numéro:</u> c71.

Fig. 8.4 An example of a completed form for collecting handwritten Tunisian town/village names for the IFN/ENIT-database

Table 8.3 Examples from the *IFN/ENIT-database*: a Tunisian village name (أولاد حفوز) written by 12 different writers

أولاد حفوز	أولاد حفوز	أولاد حفوز
أولاد حفوز	أولاد حفوز	أولاد حفوز
أولاد حفوز	أولاد حفوز	أولاد حفوز
أولاد حفوز	أولاد حفوز	أولاد حفوز

black or dark blue pen, the binarization was not a problem. While scanning a page the page number and the additional information³ were keyed in manually.

A page slope correction was performed automatically using the extra bold black line at the bottom of the page as a horizontal reference. An advanced projection method was performed to extract the word and the postcode images on the page automatically.

Examples of extracted words are shown in Table 8.3. Additionally, the example gives an idea about the variety of the collected data, because the displayed Tunisian village name is written by 12 different writers.

8.2.3 Database Ground Truth

A database for training and testing recognitions systems requires not only the images with the script; it also needs to know with as much detail and accuracy as possible what is on these images. That information is called ground truth (GT). Probably the most important information of the GT is the sequence of the characters of the word. Due to the Arabic writing style (cf. Sect. 8.1.1), where the shape of the character changes depending on the neighboring characters, the concrete used shape must also be labeled. Clearly, to gather this detailed information is a very time-consuming and error-prone process. In the following we present the GT of the *IFN/ENIT-database* and we also provide insight into how we managed to get the required information.

³The additional information like name, residence, age, and profession were vital to keep track of who filled out which form. This information is needed to build writer disjunct sets from the collected data. Otherwise, the same writer could probably contribute to the learn and to the test set; this would contort the tests and had to be avoided. To gather statistical parameters about the writers who contributed to the database, the additional information from the form was also very helpful.

Table 8.4 Two examples for dataset entries of the *IFN/ENIT-database*. The symbols *M*, *B*, *A*, *E*, *L* stand for the used character shape (middle, begin, alone, end, ligature)

Image	الحَمَّامَات	قرْنَة
Ground truth:		
Postcode	8050	3070
Global Word	الحَمَّامَات	قرْنَة
Character shape sequence	ا E م ـM ـL ـB ـح ـM ـل ـA ـT ـا ـT ـا ـE ـم ـB	ـق ـB ـE ـق ـN ـE ـT ـا ـR ـق ـB ـE ـق ـB
Baseline y1,y2	52,47	77,83
Baseline quality	B1	B1 (B1=OK; B2=bad)
Quantity of words	1	1
Quantity of PAW's	4	2
Quantity of chars	8	5
Writing quality	W1	W1 (W1=OK; W2=bad)

***IFN/ENIT-database* Ground Truth in Detail**

Each handwritten town name comes with image and GT information. The following GT information is available for each name image:

- Postcode (automatically generated)
 - Arabic word in ISO 8859-6 code set (automatically generated)
 - Arabic word as character sequence with shape index (automatically generated and manually verified)
 - Number of words, PAWs, and characters in the town/village name (automatically generated and manually verified)
 - Baseline (automatically generated and manually verified)
 - Baseline quality (manually labeled)
 - Writer identifier, age, profession, and writing quality (manually labeled, indirectly part of the GT).⁴

Table 8.4 gives two examples of dataset entries of the *IFN/ENIT-database*. Because we were aware of promising research topics, we added some “special” features, which are not as common in other databases; e.g., the information about the writing line (baseline) position and some quality flags are included in the GT of the *IFN/ENIT-database*.

Pre-label and Pre-baseline

With the knowledge of the form page number each word is automatically assigned a pre-label. The pre-label of the word consists of the postcode, the word in Arabic

⁴For example, the writer is coded in the image file name, and the age, profession, and writing quality were used to arrange writers into groups and divide them uniformly over the sets (Sect. 8.2.4).

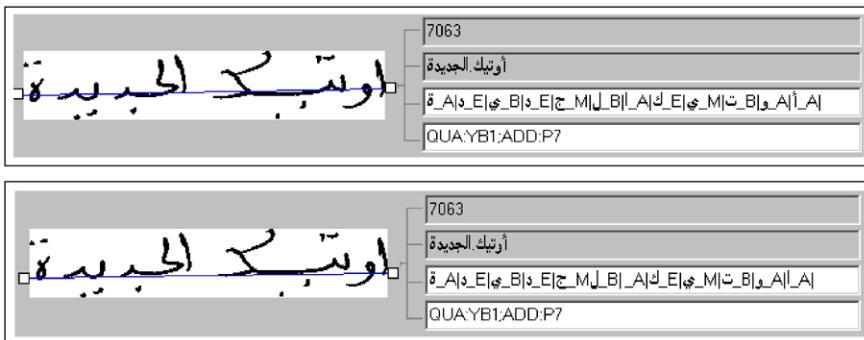


Fig. 8.5 Interface of label verification tool of the database *IFN/ENIT-database*. Top: On the left you see the image of the handwritten word with the pre-baseline, on the right we have the pre-label, and in the last row some quality and additional marks are shown. Bottom: Correct label after the verification step (ligature of the Arabic character Lam (ل_B) and Jim (ج_M) was well noted); the baseline position was slightly adjusted

code set ISO 8859-6, and a code that describes the sequence of the Arabic character shapes. This character shape code is generated automatically using a simple glyph-shaping algorithm. This is an important step for the labeling of character shapes, because the Arabic code set ISO 8859-6 does not provide shape occurrence information. To make the code unambiguous we added additional Latin characters as indexes. “B” stands for beginning, “M” for middle, “E” for end and “A” for alone/isolated character shapes. An “L” marks the “Chadda.” These descriptors are linked to the Arabic code with the “_” (cf. Table 8.4). The codes for each character are separated by “|”. Additionally, a pre-baseline estimation was performed on each name image automatically. This straight line should give a good estimation of the baseline (writing line) of the name.

Verification

Due to the variability of handwriting or simply due to writing errors, the pre-labels do not always match the handwritten word. Therefore, manual verification is needed to obtain a label that matches the handwritten word character sequence. For example, when ligatures were used and “Chadda” were not copied, the label had to be corrected. Words with writing errors or written with special ligatures, which appear only sporadically, were not included in the *IFN/ENIT-database*. During this verification procedure the automatically generated pre-baseline was also corrected.⁵ The interface for this verification procedure is shown in Fig. 8.5. The corrected label and baseline of the image are also presented in Fig. 8.5.

⁵If it was not possible for whatever reason to give an “acceptable” baseline during the verification task, the quality flag for the baseline was set to “bad.” Also, the writing quality could be marked as “bad.”

Table 8.5 Distribution of the words with respect to the quantity of characters that they consist of

Quantity of characters	Frequency in %	Quantity of characters	Frequency in %
3	2.24	4	12.21
5	10.24	6	10.03
7	11.92	8	14.85
9	10.45	10	9.00
11	5.26	12	4.47
13	3.54	14	1.37
15	1.50	16	0.07
17	2.85	≥ 18	÷

8.2.4 Details and Statistics of the Database

The vocabulary of the *IFN/ENIT-database*⁶ consists of the 937 Tunisian town/village names. The names printed on each form were selected randomly with the condition that each character shape should occur at a minimum more than 300 times.⁷ Therefore, those names with rare character shapes occur more often than names with frequent ones. That leads to a quite unequal distribution of the town village/names; e.g., 22 names appear more than 300 times, whereas 2 occur only 3 times. In the end we obtained 2273 forms filled out by 411 writers with 26459 handwritten Tunisian town/village names, made up of about 115000 parts of Arabic words (PAWs) and about 212000 characters.

In the following we present some statistics summarized in tables.

Distribution of Words and PAWs

Table 8.5 shows the distribution of the words in relation to the quantity of the characters; Table 8.6 shows it in relation to the quantity of PAWs.

More than three-quarters of the words consist of four to ten characters. The shortest word in the data collection consists of three characters, and the longest of 17 characters. A quarter of the words are composed of four PAWs.

The majority (70 %) of the PAWs consist on average of two characters. With about 15 % follow PAWs which consist of one or three characters.

⁶All information refers to *IFN/ENIT-database* version 1.0p2 (www.ifnenit.com). The database is in ongoing development.

⁷To optimize the database in relation to equally distributed town/village names was not an option, because the effort would be too huge. For example, if we only assume 100 times for each town/village name, we would need about 1600 writers!

Table 8.6 Distribution of the words with respect to the quantity of PAWs that they consist of

Quantity of PAWs	Frequency in %	Quantity of PAWs	Frequency in %
1	2.99	2	15.35
3	17.60	4	24.84
5	14.67	6	8.24
7	7.32	8	6.04
9	1.55	10	1.40

Table 8.7 Age and profession of writers who contributed to the *IFN/ENIT-database*

	Student	Academic staff	Technician	Others	Σ
≤ 20	29.0	0	0	0	29.0
21 to 30	35.6	4.2	3.9	3.9	47.6
31 to 40	0.2	3.4	4.9	2.0	10.5
> 40	0	4.1	5.4	3.4	12.9
Σ	64.8	11.7	14.2	9.3	100

Distribution of the Writers

Table 8.7 gives an overview of the age and profession of the writers who have contributed to the database. Actually, about three-quarters of the 411 writers were younger than 31 years, and about two-thirds were students.

Since the collection of the writing took place predominantly in the surroundings of the university, this result is not surprising.

Distribution of the Four Sets

The collected data was split into four sets (a, b, c, d). The characteristics of these sets are shown in Table 8.8.

Miscellaneous Details

Table 8.9 gives examples which indicate the large variety within the *IFN/ENIT-database*. Shown are the two most frequently occurring names (about 380 times) with their longest and their shortest representatives. The mean height of the script all over the *IFN/ENIT-database* calculates to 96.6 ± 19.2 pixels (8.2 ± 1.6 mm: 300 dpi), where the mean stroke width is 6.2 ± 2.5 pixels (0.5 ± 0.2 mm:300 dpi).

Table 8.8 Distribution of the four sets of the *IFN/ENIT-database*

Set	Quantity of words	Quantity of writers (of this, quant. of “bad” ^a writers)
a	6537	102 (14)
b	6710	102 (13)
c	6477	103 (15)
d	6735	104 (14)
Σ	26459	411 (56)

^aWhile verifying the GT, there was the possibility to mark a writer as “bad” if the word was hardly readable even for a native speaker

Table 8.9 Variance of the length of words within the *IFN/ENIT-database*; shows for two examples the longest and the shortest representatives (all numbers in pixels)

$\bar{X} \pm S_X$	$X \times Y$	longest representative	$X \times Y$	shortest representative
762 \pm 121	1069 \times 158	سید بی دبی امیر این همار	508 \times 158	سید یا میرا هیم از همار
253 \pm 65	603 \times 162	المرضاع	124 \times 111	الرفاع

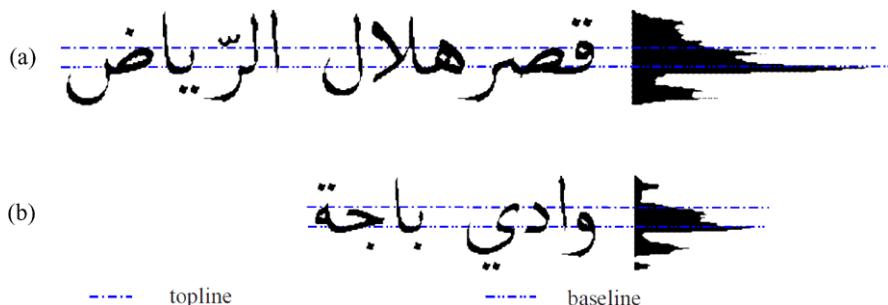


Fig. 8.6 Arabic printed words with horizontal profile and marked baseline and topline. The maximum peak in the horizontal profile defines the position of the baseline; the topline position is determined by the maximum gradient of the horizontal profile above the baseline

8.3 Baseline and Topline Estimation

The most important parameter for a script normalization procedure is the position of the writing lines or references lines within a word. We will call these lines the baseline and topline. Figure 8.6 gives two examples of printed Arabic words with their corresponding baseline and topline defined by the widely used horizontal projection approach. A prime example of the statement that some characters cannot be recognized without information about their relative vertical position within a word is the “g/9”-problem in relation to Latin handwritten character recognition. For Arabic characters we find similar issues, e.g., the characters “د (Dal)” and “ر (Ra)” or

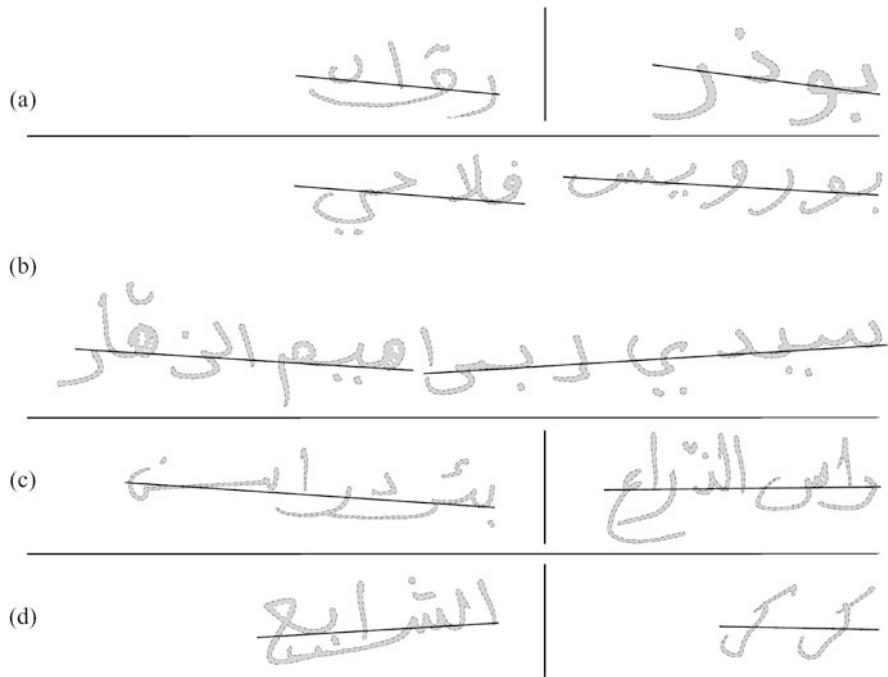


Fig. 8.7 Examples (taken from the *IFN/ENIT-database*) of handwritten words where the estimation of the baseline seems to be very challenging. The optimal position of the baseline is given for each word by a *solid line* (for further explanation see the text)

“ب” (Ba) and “ن” (Nun) (disregarding the punctuation). It becomes obvious that the baseline of a word contains essential information for an automatic recognition system. If the lines are detected, a reasonable normalization of the skew angle and the height of the word can be applied.

Challenges of the Baseline Estimation Task for Handwritten Arabic Words
Due to the fact that the writing and thus the writing line varies strongly between different writers, the estimation of the baseline⁸ is considered a very challenging task. We discovered four main issues regarding the baseline estimation procedure, as illustrated in Fig. 8.7:

- (1) Very short words, which consists of many PAWs (Fig. 8.7a)
- (2) Long words (or groups of words) with a “jumping” baseline resulting from non-compliance with rules of writing (Fig. 8.7b)
- (3) Words which consist of characters with long bottom curves (Fig. 8.7c)
- (4) Other unfavorable constellations (Fig. 8.7d)

⁸The topline estimation is discussed separately in Sect. 8.3.4.

Figure 8.7a shows two typical representatives for issue (1). The words are very short (e.g., four characters), whereby frequently characters arise which are, owing to the Arabic writing, written in an isolated manner from each other. Both characteristics are very unfavorable to a successful baseline estimation. Writing isolated characters often leads to a less consistently distinct baseline.

Examples related to issue (2) are given in Fig. 8.7b. Similar to issue (1) we have to deal with non-consistent baselines; however, the cause is entirely different. The words consist of several, partially longer PAWs (or words)⁹ which are compounded from many characters and thus become long. This circumstance means that there is a higher risk that each part of the word will be situated on a different horizontal position. Thus one observes that, e.g., often for the beginning and for the end of the word physically separate baselines exist.¹⁰ There is no doubt that the correct way to deal with that issue would be to determine each baseline separately. By the way, even the assumption that the baseline is a straight line can be questioned.¹¹ On the other hand, it is a bit dissatisfying, but there is no plan to implement a robust estimation of, e.g., a polygonal line as a baseline position, and there is currently no corresponding GT which would make it possible to evaluate such an approach.

With certain character combinations one habit of writers is observed frequently which disturbs the estimation of the baseline substantially, as described in issue (3). Figure 8.7c shows two examples. The habit consists of writing the last character of a word or PAW with great momentum. If there are many PAWs within a word consisting of characters that have descenders (see Table 8.1), the descenders become very significant and can negatively affect the baseline estimation process.

Issue (4) includes different artifacts which do not belong to one of the other issue categories. Figure 8.7d gives two examples. First, due to the carelessness of the writer, two PAWs are linked together in a very unfavorable way and, second, the writer has written even a very short word with the use of ligatures. Actually, there is nothing wrong with that writing, it is simply a matter of handwriting. But these examples raise the question of how much an effect such artifacts may have on the assumptions of baseline estimation approaches in general.

Intensive meaningful tests are desirable and, due to the GT of the *IFN/ENIT-database*, feasible. We next introduce our baseline error measurement criteria (Sect. 8.3.1). In Sect. 8.3.2 we evaluate a widely used approach for baseline estimation. This approach is based on analyses of the horizontal projection histogram. A different approach, completely based on polygonally approximated skeleton processing of a word, is described in Sect. 8.3.3. Section 8.3.4 gives a short overview on methods to estimate the topline in Arabic handwriting. We will end with a short summary in Sect. 8.3.5.

⁹Some town/village names consist of several isolated words which differentiate themselves only slightly from “normal” PAWs.

¹⁰Figure 8.7b shows examples where the baseline is marked.

¹¹If the baseline quality mark within the GT is set to “bad,” it signals that there was a problem in defining a sufficient baseline position, at least with a straight line.

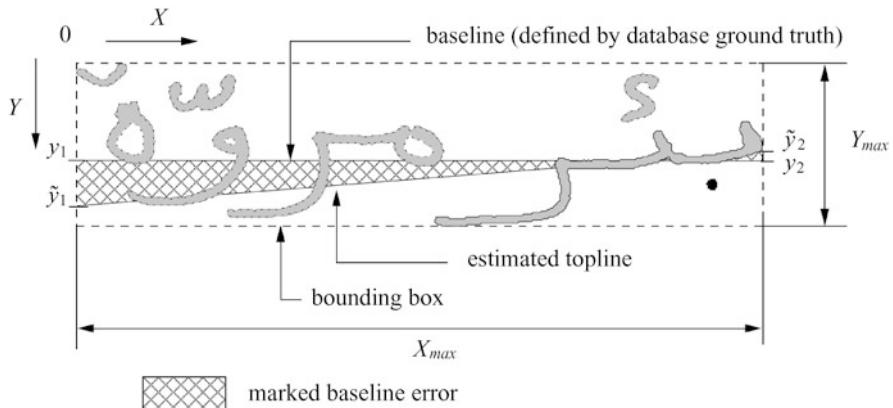


Fig. 8.8 Example of the baseline measurement definition. The error measurement is defined by the area between the optimal and the estimated baseline, divided by the length of the word (X_{max})

Table 8.10 Experimentally found thresholds of the baseline error measurement

Error threshold in pixels	Mark
0–5	excellent
5–7	acceptable
>7	insufficient

8.3.1 Definition of the Baseline Error Measurement

As mentioned before, the GT of the *IFN/ENIT-database* contains the baseline position. This baseline GT is very useful, because it puts us in a position to evaluate a baseline detection algorithm on the basis of a quite large dataset (*IFN/ENIT-database*) automatically. As an error measurement we have calculated the averaged distance between the estimated baseline and the baseline GT for each word. An example is given in Fig. 8.8. This distance value is used as a quality indicator for an objective performance evaluation. It is obvious that if the estimated baseline position is congruent to the baseline GT, the error value gives 0 pixel and that value becomes the objective of all optimizations. But up to what degree can the divergence between the estimated baseline and baseline from GT be rated as acceptable or as insufficient? To answer that question, we prepared sheets of papers with some hundred Arabic words (taken from the *IFN/ENIT-database*) together with a marked previously estimated baseline. Then we asked a group of Arabic native speakers to look quickly through the sheets with words and to tag all those words that did not agree with the marked baseline position. The result of this little visual experiment is summarized in Table 8.10. We knew that the baseline for the same word determined by three different individuals will result in three different baseline positions. As a

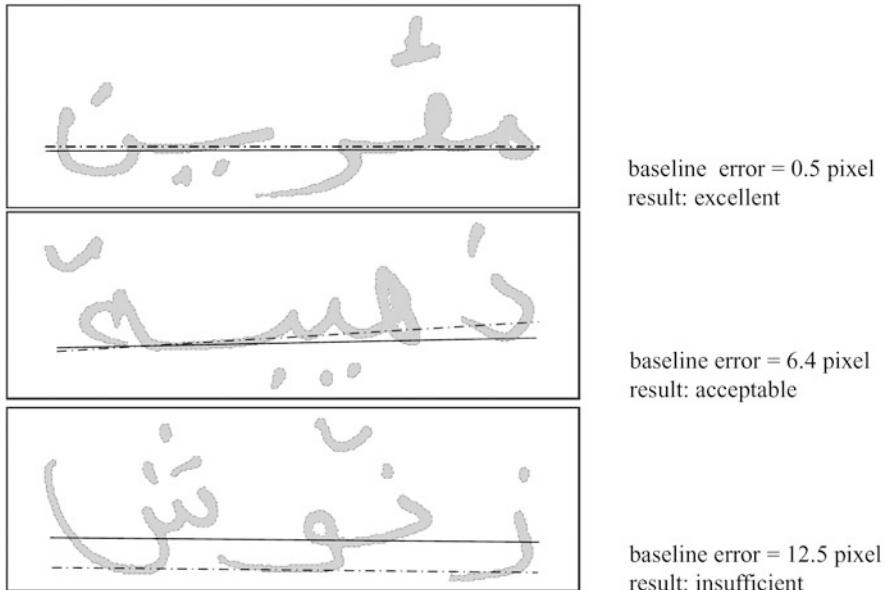


Fig. 8.9 Examples of baseline errors. The optimal baseline taken from the ground truth of the database (*solid line*) and the estimated baseline (*dot-dash line*) are marked for each word

consilient result, we observed that for up to a 5 pixel¹² vertical position error the baseline was evaluated as excellent, for up to a 7 pixel vertical position error everybody could still agree with the baseline position, and for an error of more than 7 pixels the baseline estimation was considered as insufficient. Figure 8.9 exemplifies the baseline error criteria. Of course, for a final evaluation the baseline error measurement has to be tested in a complete recognition system. However, optimizing one parameter in a baseline finding algorithm while running the whole recognition environment (normalization, feature extraction, HMM training and testing (cf. Sect. 8.5)) or asking a human reviewer again and again for thousands of words about the baseline quality is even less practical. For this purpose our quality measurement can be a helpful tool.

8.3.2 Baseline Estimation: Horizontal Projection Method

The horizontal projection method is a widely used method to estimate the baseline. Line by line the black pixels are counted; the maximum number of pixels indicates the position of the baseline (cf. Fig. 8.6). The projection method is robust and easy to implement, but it requires straight lines and long words, which is often not the

¹²Average word image height was approximately 100 pixels, 8.5 mm; for details see Sect. 8.2.4.

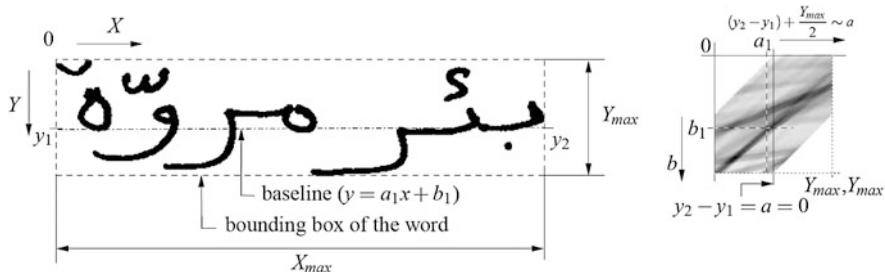


Fig. 8.10 Definition of the *modified* Hough parameter space

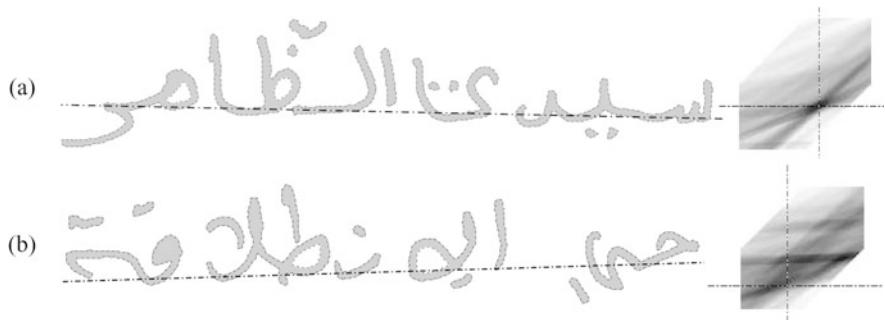


Fig. 8.11 Arabic words with correspondent Hough space; the position of the optimal baseline is marked. (a) A very distinct maximum in the Hough space marks the position of the baseline. (b) Hough space with several ranges with high entries, which represents different baseline positions

case for single handwritten words. We enhanced this approach by transforming the binary word image into a Hough parameter space, where the dark regions indicate line directions with many black pixels on a straight line in the word image. Figure 8.10 gives an example. In the simplest case only one range with high entries appears in the Hough space, and its maximum is easy to determine. In accordance with the idea of the horizontal projection method, the maximum in Hough space represents the baseline position. Figure 8.11a shows an example of a problem-free application of this procedure. In the Hough space only a distinct range is formed. The maximum value corresponds to the position of the desired baseline. By contrast, Fig. 8.11b shows an example of a more problematic case which can frequently be found. Several ranges with distinct local maxima result in the Hough space. Nevertheless, only the position of the straight lines resulting from the global maximum of the Hough space is taken for a first detailed investigation. In the following the baseline error measurement introduced in Sect. 8.3.1 is used for evaluation. The results of this first test are arranged in Table 8.11. The complete *IFN/ENIT-database* was used for this test. In 78.1 % of all words within the dataset an *acceptable* baseline (error ≤ 7) was found with the initial settings (cf. Table 8.11).

An improvement of the results was found by smoothing the Hough space with a simple median filter. Applying a median filter of size 3×3 results in a shift of

Table 8.11 Baseline estimation error determined with the basic method of horizontal projection; i.e., only the global maximum in the Hough space is evaluated. The entire data collection *IFN/ENIT-database* was used

Baseline error [pixels]	Part of the dataset [%]	Part of the dataset [words]
≤ 1	8.1	2275
≤ 5	65.3	17233
≤ 7	78.1	20672
≤ 10	87.9	23257
≤ 15	95.2	25178
≤ 50	100.0	26459

Table 8.12 Baseline estimation error determined with the basic method of horizontal projection after median filtering with filters of different window sizes; i.e., only the global maximum in the Hough space is evaluated. The whole *IFN/ENIT-database* was used

Baseline error in pixels	Part of the dataset [%]			
	Window 3×3	Window 5×5	Window 7×7	Window 9×9
≤ 1	7.6	6.5	5.3	3.0
≤ 5	71.2	71.6	70.7	58.8
≤ 7	81.4	82.7	82.8	77.5
≤ 10	89.6	90.7	91.2	89.8
≤ 15	95.7	96.4	96.2	96.9
≤ 50	100.0	100.0	100.0	100.0

the position of the global maximum in the Hough space in such a way that we could measure an improvement of 3 % for the reference value *baseline is acceptable*. For a filter size 5×5 and 7×7 this result could still be improved, whereby after a filtering with a filter of size 9×9 a degradation was found. The best result (82.8 %) was reached with the filter of size 7×7 . That corresponds quite well with the estimated averaged line thickness of *IFN/ENIT-database* (see Sect. 8.2.4). Table 8.12 shows the results in more detail.

Discussion on the Validity of the Assumption The findings are very remarkable: Within approximately a fifth of all words of the *IFN/ENIT-database* the assumption fails that the global maximum in the Hough space corresponds to the position of the baseline in the word.

Therefore, a further investigation seems to be very interesting, which is described in the opposite way: To what extent does the assumption apply at all, that the baseline is found where there is a significant cluster of word image pixels on a straight line, which corresponds to high values in the Hough space? Using the baseline GT of the *IFN/ENIT-database*, the corresponding position in the Hough space can be computed. Now it remains to examine if there is at least a “local” maximum in the

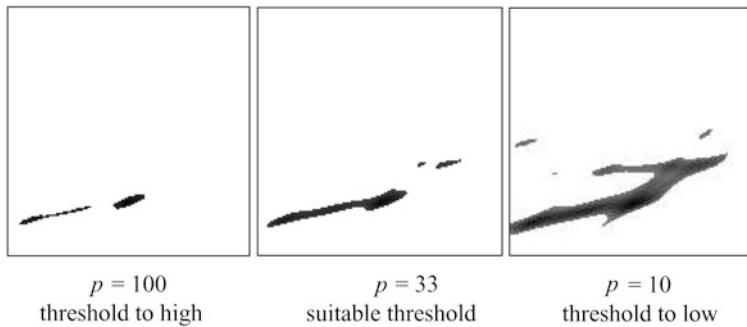


Fig. 8.12 Thresholding the Hough space using the *p-tile thresholding* method

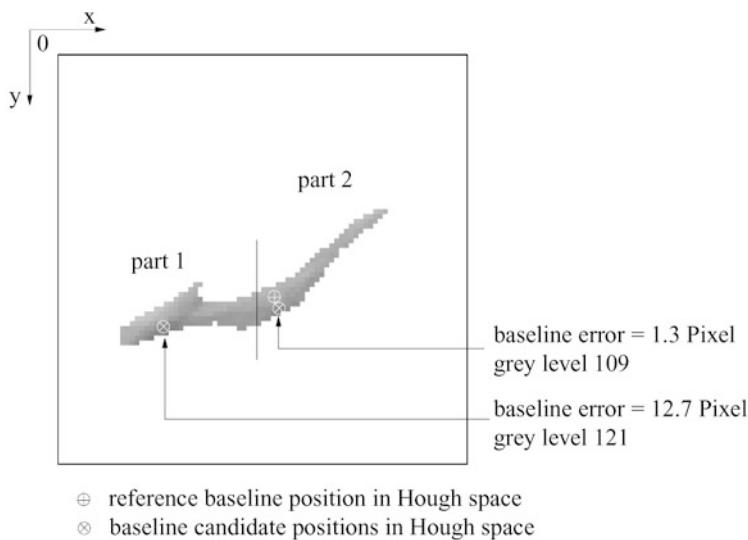


Fig. 8.13 Hough space: splitting larger clusters into equally large parts

periphery (≤ 7 pixel criteria) of the optimal baseline position in the Hough space. If that is not the case, then the basic assumption fails in principle for these words. For these tests first all local maxima in the Hough space must be determined. For this purpose a simple threshold value setting is used to cluster the region of interest of the Hough space. Afterwards these clusters are labeled using a *connected component labeling* method [28, 29]. The threshold value is determined by the *p-tile thresholding* method. The threshold value is specified in such a way that the extracted cluster area is a p th of the total area of the whole Hough space. Figure 8.12 shows some examples of a clustered Hough space using the *p-tile thresholding* method. Setting the parameter $p = 33$ leads to the best result. Thus the threshold is set in such a way that about 3 % of the background pixels remain standing. Figure 8.12 shows an example of a thresholded Hough space ($p = 33$), which results in three clusters. Often only

Table 8.13 Results of the baseline estimation using the “extended” (cf. text) horizontal projection method in relation to parameter p . The whole dataset of the *IFN/ENIT-database* was used

Baseline error in [pixels]	Part of the dataset [%]		
	$p = 100$	$p = 33$	$p = 10$
≤ 1	8.8	10.6	11.8
≤ 5	72.8	79.0	81.5
≤ 7	86.5	91.5	93.2
≤ 10	94.7	97.6	98.4
≤ 15	98.7	99.5	99.7
≤ 50	100.00	100.00	100.00

an expanded cluster is found, which is divided into several equally large parts. For clarification Fig. 8.13 gives an example, where the position of the reference baseline and the positions of the “local” maxima are marked. Every “local” maximum stands for a baseline position candidate and can be compared with the reference baseline position. We used the baseline measurement criteria (cf. Sect. 8.3.1) to evaluate these baseline candidates. It was found that at best¹³ 93.2 % of all words from the *IFN/ENIT-database* belong¹⁴ to a detected “local” maximum¹⁵ in the Hough space. To put it the other way around, in 7 % of the words even the extended¹⁶ horizontal projection method fails, and no *acceptable* baseline can be found. Table 8.13 gives the results in detail.

Discussion on the Results in Relation to the Impact of the Dataset This result is remarkable, even though it is only valid in regard to the *IFN/ENIT-database*. To get an idea of to what extent the result belongs to the dataset, we performed the following test. Using ArabTeX we generated a “printed version”¹⁷ of the *IFN/ENIT-database*. We ran the baseline estimation algorithm¹⁸ as described. Table 8.14 shows the results, including the results for the topline estimation (the underlying approach will be discussed in Sect. 8.3.4). For about 95 % of the dataset an acceptable baseline position was found and for the topline about 89 %, which leads to the conclusion that the assumptions made by the approaches are quite adequate, at least in relation to Arabic printed words. On the other hand, it makes it clear that the lower accurate rate of the baseline position finding for the *IFN/ENIT-database* is not due to the

¹³The results depend on some parameters (see Fig. 8.12 and Fig. 8.13).

¹⁴Baseline error ≤ 7 pixels.

¹⁵The parameter for thresholding and the clustering were set to values that result in not more than ten “local” maxima (baseline position candidates) in the Hough space.

¹⁶Considering “local” maximum in the Hough space.

¹⁷With regard to the vocabulary and considering the frequency and the approximate font size of the words; using the common *Naskh* style.

¹⁸Based on global maximum detection within the filtered Hough space.

Table 8.14 Baseline estimation error for Arabic printed words (for details cf. text)

Baseline error in [pixels]	Part of the printed dataset [%]	
	Baseline	Topline
≤ 1	34.3	1.7
≤ 5	90.2	80.3
≤ 7	95.2	88.7
≤ 10	96.4	92.7
≤ 15	99.6	94.7
≤ 50	100.00	100.00

vocabulary of the dataset. It seems to be the more general influence of handwriting, and so it can be supposed that the results are valid for Arabic handwriting in general.

Back to the approach which uses the local maxima in the Hough space: If an algorithm could always select the right baseline candidate (1 out of 10), an improvement of about 11 % in comparison to the median filtered Hough space method (see Table 8.12) would be possible. One approach was tested [24] based on the assumption that according to the Arabic script each PAW of a word has to intersect the baseline. It turns out that this assumption is, in practice, not the case, particularly for the “problematic” words.

In the following section we discuss another baseline estimation approach which tries to benefit from a deeper understanding of the characteristics of Arabic handwriting.

8.3.3 Baseline Estimation: Skeleton-Based Method

The method we are presenting here is not directly based on the pixels of the word image but on the skeleton of the lines [4, 14]. The advantage of this method is not only more flexibility in the line estimation but also a more flexible and easily accessible data structure of the polygonal approximated skeleton lines. Moreover, the pen-dependent line width is removed, and the connected components are obtained (Fig. 8.14).

Overview of the Skeleton Baseline Estimation Approach

The concept of our baseline detection algorithm is shown in Fig. 8.15. At first baseline-relevant features, e.g., diacritical points, are extracted from the polygonally approximated skeleton. In the next step the connected components that are not relevant for the baseline detection are deleted. Subsequently, a first estimation of the baseline is calculated. The final step is a regression analysis of the relevant

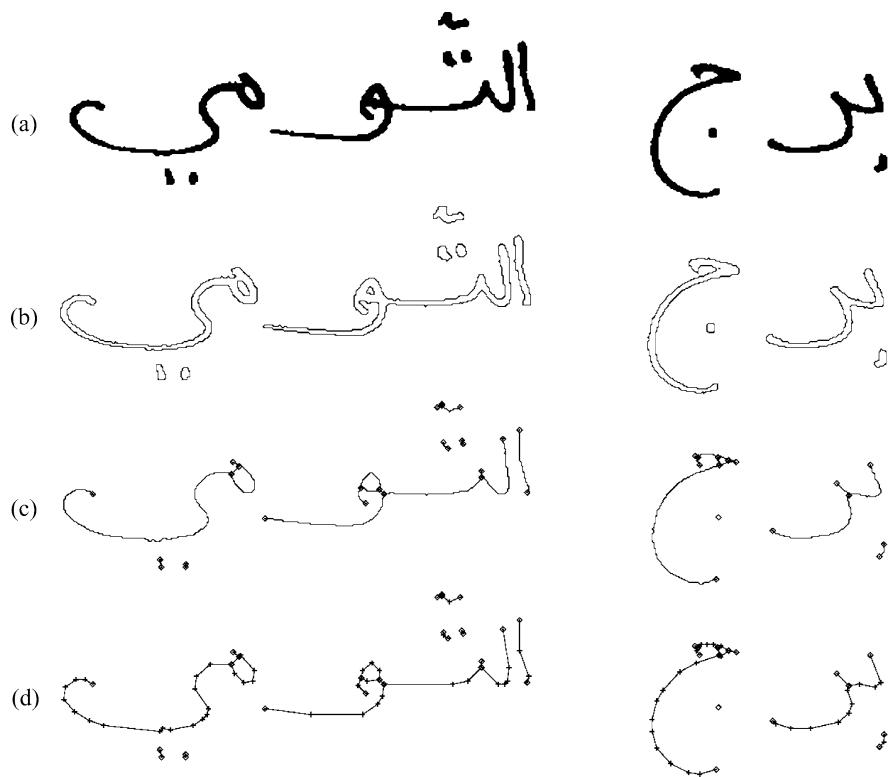
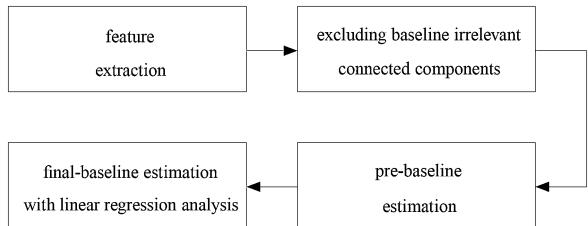


Fig. 8.14 (a) Binary scanned image; (b) contour; (c) skeleton; (d) polygonally approximated skeleton word

Fig. 8.15 Block diagram: overview of the realized approach



points in the neighborhood of this first estimation to find the final baseline position. In the following sections we give a detailed description of the proposed algorithm.

Baseline Feature Extraction

The feature extraction is based on the polygonal skeleton data of each word image (see Fig. 8.14d). This is a fast and effective way to extract baseline-relevant features.

As most parameters of handwriting vary in a wide range, we are collecting many different features of each skeleton graph (connected component) and its bounding box within a word image.

- The following features are calculated for each skeleton graph:
 - Width and height of bounding box
 - Area of bounding box
 - Aspect ratio of bounding box
 - Sum of edge lengths of skeleton graph (skeleton length)
 - Number of vertices of the graph
 - Number of edges of the graph
 - Degree of each vertex
 - Length of each edge
 - Distance and angle of vertices connected by an edge
- The following features are calculated for all skeleton graphs of each word:
 - Average bounding box area
 - Deviation of bounding box area
 - Average skeleton length
 - Deviation of skeleton length

Selection of the Baseline-Relevant Features

In the second step of our baseline estimation approach a reduction of the whole feature set of the word image is done by selecting only features of baseline-relevant connected components. This selection is done on the basis of the knowledge of Arabic writing style. Diacritical marks, e.g., points or “chadda,” are not baseline relevant. Furthermore, the features of isolated written Arabic characters with a simple graph structure, as for example the characters “Ra” or “Zai” (see Table 8.1), and the features of long bottom curves are also irrelevant and not used. In the following the features that are indicators for the presence of such baseline-irrelevant connected components (object graphs) are itemized:

- Features of erasable points are:
 - Object graphs with sum of all edge lengths less than 20 % of average length of all object graphs in the word image
 - Object graphs with bounding box area less than 20 % of average bounding box area
- Features of erasable “chadda” marks:
 - Object is not a point
 - Object graph with bounding box positioned in the top third of the word box
 - Object graph with an aspect ratio of the bounding box more than 1.4
 - Object graph has more than 2 vertices
- Features of a “simple structure” are:
 - Object is not a point
 - Object graph has exactly 2 vertices

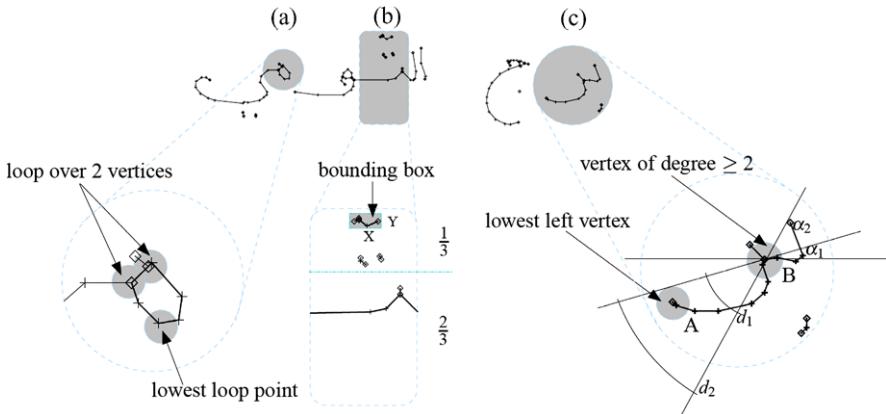


Fig. 8.16 Examples of the detection of baseline-irrelevant objects. (a) Detecting a loop and defining the lowest loop point; (b) detecting “chadda”; (c) detecting bottom curve

- Features of a “long bottom curve” are:
 - Object graph is not a point
 - Object graph has more than 3 vertices
 - Vertex A is the lowest and leftmost point of the object graph
 - Vertex A is of degree 1 and connects vertex B
 - Vertex B is of degree ≥ 2
 - The distance between A and B is d with ($d_1 \leq d \leq d_2$; $d_1 = 20$; $d_2 = 150$)
 - The angle of the straight line connecting A and B is α ($\alpha_1 \leq \alpha \leq \alpha_2$; $\alpha_1 = 15^\circ$; $\alpha_2 = 50^\circ$)

Figure 8.16 gives some examples of the described features.

Pre-baseline Estimation

Based on these features a first rough estimation of the baseline is calculated. This first estimation of the baseline is called the pre-baseline, and it is a horizontal line. Based on the features described in Sect. 8.3.3 we use only object graphs that are not part of lines that were detected as irrelevant for the baseline estimation. From these graphs the more or less horizontal edge segments are considered. For each edge segment the length and the position of its center point are calculated. The y-coordinate of the center point of each edge is weighted by the product of the following values:

1. Length of the considered edge segment
2. Sum of the length of all edges connected with this edge segment
3. Aspect ratio of the bounding box of all these edges

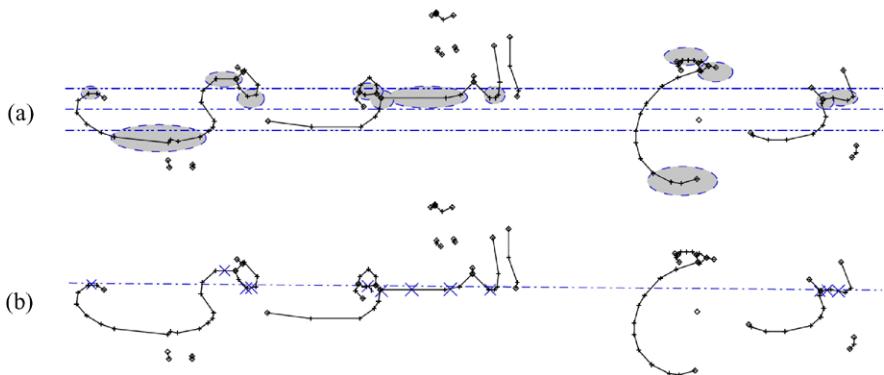


Fig. 8.17 (a) Fields with horizontal lines: marked *light gray*; pre-baseline: *dash-dot line*; range of interest: between *dash-dot-dot lines*. (b) Considered points for the linear regression analysis: marked with *crosses*; final baseline: *dash-dot line*

The sum of all the considered weighted edge segments gives us the y -coordinate of the horizontal pre-baseline. An example of a pre-baseline together with a range of interest is shown in Fig. 8.17a.

Final Baseline Estimation

In the final baseline estimation step we have another optimization step. But now we use only edge segments that are more or less horizontal with a center point inside the range near the pre-baseline. In addition to these points two further baseline-relevant robust feature points are selected:

1. If a closed loop partially overlaps the range, the lowest point of a closed loop is selected
2. If the upper end point of a long button curve is inside the range, it is selected

Based on all these points a linear regression determines the parameters of the linear equation $y = m \cdot x + b$. Figure 8.17b gives an example of the resulting baseline together with the considered points. The height of the range was set to a third of the word image height.

8.3.4 Topline Estimation

The horizontal projection histogram is often used to estimate the topline too. We again use the Hough space, but now we calculate the vertical gradient of the Hough space to determine the topline by selecting the maximal gradient value within a search area above the position of the baseline. Figure 8.18 gives an example of the

Fig. 8.18 Topline estimation approach using the vertical gradient of the Hough parameter space

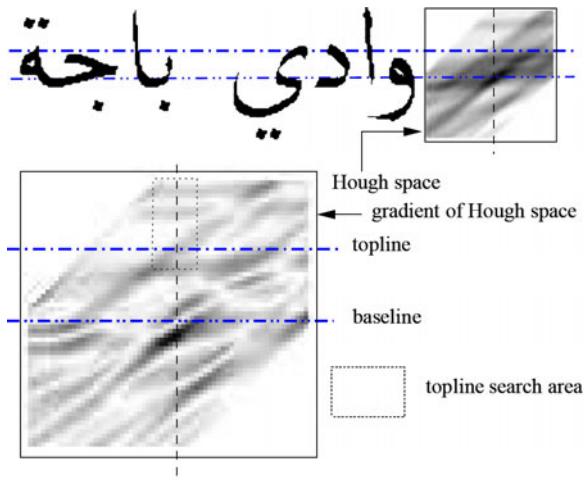


Table 8.15 Evaluation of topline finding methods

Topline error in pixels	Hough space method	Simple method
≤ 5	46.3 %	56.8 %
≤ 7	59.6 %	74.3 %

topline estimation approach. Due to the fact that the topline is part of the *IFN/ENIT-database* (*set a*), an evaluation of the topline detecting algorithm was performed. The result is disappointing (see Table 8.15). Only for about 60 % of the handwritten Arabic words was an acceptable topline determined.

It is interesting that even a straightforward method provides better results. In this approach the topline was set parallel to the baseline at a distance of 36 %¹⁹ of the distance between the writing line (baseline) and the top of the word.

These experiments have shown that the topline estimation should not be used within an Arabic handwritten word recognition process. Therefore, two different word normalization methods and two different feature sets used by the HMM-based recognition system are presented (cf. Sect. 8.4) with low and with no dependency of the estimated topline position.

8.3.5 Conclusive Remarks on Baseline Estimation

The baseline GT of the *IFN/ENIT-database* made it possible to evaluate intensively two different baseline estimation approaches. The approach based on the usage of

¹⁹It is strongly dependent on the dataset.

Table 8.16 Baseline estimation error, using the skeleton-based method (explanation in text). The whole *IFN/ENIT-database* was used

Baseline error in [pixels]	Part of the dataset [%]	
	Skeleton-based approach	Horiz. projection approach
≤ 1	11.7	5.3
≤ 5	76.7	70.7
≤ 7	87.5	82.8
≤ 10	94.1	91.2
≤ 15	97.5	96.2
≤ 50	100.0	100.0

the horizontal projection histogram including some extensions enables us to find an acceptable baseline in 82.8 % of the dataset. The skeleton-based approach, which takes more account of the characteristics of Arabic handwriting, performs better with 87.5 % acceptable baselines (cf. Table 8.16). The assumptions for both baseline estimation approaches are proved to be appropriate for Arabic script, especially for printed Arabic script (cf. Table 8.14). Although there is no structural difference between handwritten Arabic and printed Arabic words, at least the variety which comes with a large dataset makes the difference. Two things become clear: If there is “bad” handwriting, the baseline estimation errors increase. Also the combinations of characters within the words do have strong effects on the baseline estimation errors. These are criteria which are part of characteristics of a dataset.

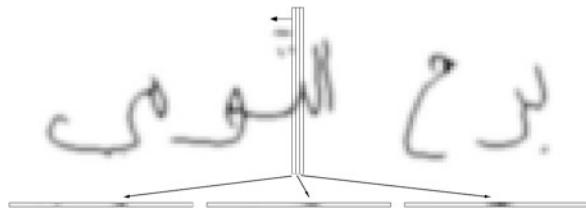
Regarding the topline estimation, our experience was that, at least within Arabic handwriting, the standard approach fails in many cases. The main reason seems to be the lack of Arabic characters that give a significant and robust detectable clue for the position of the topline. Nine out of 28 Arabic characters do not even touch the topline position. Combined with the variety due to the handwriting, there is less chance for a robust topline estimation. This results in the recommendation to avoid using the topline position for normalization or feature extraction methods.

8.4 Normalization and Feature Extraction

The extraction of the features which are used for the recognition process is a very difficult task [22, 26]. In the following we will present two different feature sets²⁰ which we used within our HMM-based recognition system. Because the feature extraction, image pre-processing, and normalization tasks strongly depend on each other, we describe these tasks accordingly in combination.

²⁰We will call the feature sets *A* and *B*.

Fig. 8.19 Extraction of pixel features using a sliding window with three columns (method A)



8.4.1 Normalization and Feature Extraction Method A

The first normalization step is a rotation of the word resulting in a horizontal baseline. Subsequently a vertical height normalization is done using a linear characteristic between topline and baseline and a nonlinear characteristic elsewhere. This results in constant heights for ascender and descender regions and thus in a fixed total height of the resulting skeleton graph. Next a horizontal width normalization with a linear characteristic is performed, yielding a word with constant average character width. The line thickness is normalized during the generation of the skeleton. Finally a rethickening is done by a Gaussian filtering of the normalized skeleton image, resulting in a gray level image. Figure 8.3 shows an example of the normalization process and the normalized word image.

Feature extraction method A is directly based on an image representation of the script using pixel values as basic features. A rectangular window is shifted with respect to the Arabic writing direction from right to left across the normalized gray level script image and generates a feature vector (frame). This results in a vast amount of features for each frame. In order to reduce the number of features, a Karhunen–Loeve transformation (KLT) is performed on the gray values of each frame. KLT is a standard statistical method to reduce a feature set to only the most relevant features. The transformation matrices are computed from the training data.

Figure 8.19 gives an example of the sliding window feature extraction method. The three columns of the sliding window are concatenated to one feature vector. The sequence of these KLT transformed feature vectors are the input for the HMM recognizer.

8.4.2 Normalization and Feature Extraction Method B

The normalization steps in methods A and B are almost identical. Instead of the vertical height normalization used in A, we now enlarge virtually the word skeleton graph image with some blank lines to bring the baseline of the word into the middle of the image. A topline estimation is not needed. The normalized word skeleton graph is now used for the feature extraction process.

The feature extraction process starts by splitting the word image into a set of frames with fixed width in the vertical direction. Each frame has an overlap of 50 % with its neighbors. Each frame is split horizontally into five zones with equal height (see Fig. 8.20). The choice of five zones is intuitive; it has yielded the best recognition results and corresponds with the work in [6]. As features the length of all lines

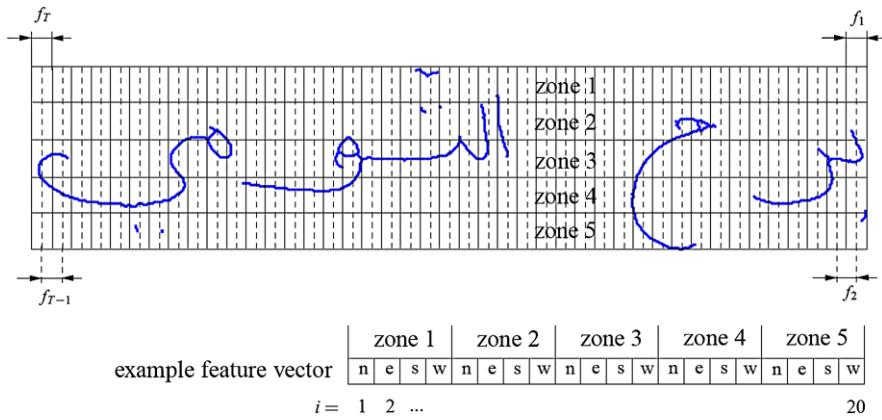


Fig. 8.20 Extraction of skeleton direction features in five zones using overlapping frames (method B)

contained in a zone calculated in the four directions north, east, south, west are used. A normalization of these values with the height of the zone ensures the invariance of the height of the word without the need of a topline estimation. Each frame is represented by a 20-dimensional feature vector. To overcome the problem of very different value ranges of each feature i , a normalization is performed (see Eqs. (8.1) to (8.6)), where i is the number of the feature in the feature vector j (i.e., in this case north, east, south, west) and $x(i)_j$ is the value of the feature vector of frame j . w_{min} is the minimum cutoff value, and w_{max} is the maximum cutoff value. V_σ is a parameter initially set to a value of 0.5. $\bar{x}(i)$ and $s(i)$ are previously calculated using N feature vectors from the training set. Finally the value $\check{x}(i)_j$ gives the normalized value of $x(i)_j$.

$$\bar{x}(i) = \frac{1}{N} \cdot \sum_{j=1}^N x(i)_j \quad (8.1)$$

$$s(i) = \sqrt{\frac{1}{N-1} \cdot \sum_{j=1}^N (x(i)_j - \bar{x}(i))^2} \quad (8.2)$$

$$w_{min}(i) = \begin{cases} \bar{x}(i) - V_\sigma \cdot s(i) & \text{if } z_1 \\ \min(x(i)_j) \forall 1 \leq j \leq N & \text{else } z_1 := \min(x(i)_j) < \bar{x}(i) - V_\sigma \cdot s(i) \end{cases} \quad (8.3)$$

$$w_{max}(i) = \begin{cases} \bar{x}(i) + V_\sigma \cdot s(i) & \text{if } z_2 \\ \max(x(i)_j) \forall 1 \leq j \leq N & \text{else } z_2 := \max(x(i)_j) > \bar{x}(i) + V_\sigma \cdot s(i) \end{cases} \quad (8.4)$$

$$\hat{x}(i)_j = \begin{cases} w_{\min}(i) & \text{if } x(i)_j < w_{\min}(i) \\ x(i)_j & \text{else } \forall 1 \leq j \leq N \\ w_{\max}(i) & \text{if } x(i)_j > w_{\max}(i) \end{cases} \quad (8.5)$$

$$\check{x}(i)_j = \frac{\hat{x}(i)_j - w_{\min}}{w_{\max} - w_{\min}} \cdot 2^8 \quad \forall 1 \leq j \leq N \quad (8.6)$$

8.5 HMM Recognizer

The problem of recognizing a handwritten word as a whole can now be considered as a sequence of decisions in which feature vectors are grouped into smaller “decision units” and sequentially recognized. The sequence of these “decision units” represents the unknown word. To solve such a recognition problem, hidden Markov models (HMMs) are widely used, in the beginning to recognize speech and later to recognize cursive written words.

Details of HMMs will not be discussed further in this paper. For more information about HMMs, refer to, e.g., [27] and [21]. In the following section the special solutions used in this system will be discussed. The first step is to define the HMM model that will be employed.

8.5.1 General Definition

HMMs can be described with the parameter set $\lambda = (A, B, \Pi)$:

- Matrix A : Transitions probabilities from one state to another, with $A = \{a_{ij}\}$ and $a_{ij} = p(X_t = j | X_{t-1} = i)$
- Matrix B : Distributes probabilities of observations, with $B = \{b_j(o)\}$
- Matrix Π : For probabilities to reach a state from the initial state, with $\Pi = \{\pi_i\}$

The goal is to determine the probability of an unknown sequence of observations $P(o_1, \dots, o_T | \lambda)$ and maximize the likelihood $\widehat{o}_i = \arg \max p(\underline{o} | \lambda_j)$.

8.5.2 Principal Structure

The following tasks must be completed to develop a cursive word recognizer:

1. Choose the states and the corresponding observations
2. Choose a topology of the states
3. Choose a strategy to segment the word into observations (manually–automatically)
4. Select training and testing data

5. Run the training of the HMM parameters
6. Test the system on the test data

In the following section, the solutions of tasks 1–6 will be briefly described as they were realized in our HMM recognizer.

8.5.3 Initialization and Preconfiguration of the Recognizer

Observations

Handwritten words are interpreted as a sequence of character shapes, which are concatenated to build the appearance of an individual handwritten word. Each character shape is interpreted as the observation output of a state of the HMM. Especially in the case of Arabic handwriting, a character's shapes differ depending on its position in a word. It follows then that the number of states is more than triple the number of characters in the Arabic alphabet.

HMM Topology

Many different model topologies have been discussed using HMM systems. The simplest and most used topology is the left-right Bakis topology (Fig. 8.21). Each state has three different paths, a recursive self-transition, a transition to the next state, and a transition that skips the next state. For the topology shown in Fig. 8.21 we have the following transition matrix:

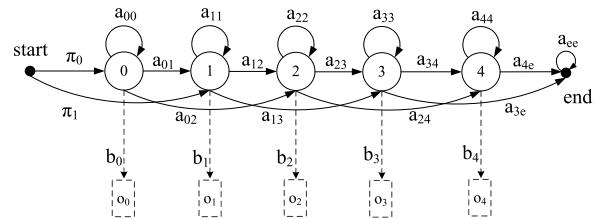
$$A = \begin{pmatrix} 0 & \pi_0 & \pi_1 & 0 & 0 & 0 & 0 \\ 0 & a_{00} & a_{01} & a_{02} & 0 & 0 & 0 \\ 0 & 0 & a_{11} & a_{12} & a_{13} & 0 & 0 \\ 0 & 0 & 0 & a_{22} & a_{23} & a_{24} & 0 \\ 0 & 0 & 0 & 0 & a_{33} & a_{34} & a_{3e} \\ 0 & 0 & 0 & 0 & 0 & a_{44} & a_{4e} \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{ee} \end{pmatrix} \quad (8.7)$$

Using this simple model for the observation of a character we chose a model with seven states for each character. This number, of course, is a parameter to optimize, which depends on the size and the quality of the data used.

Initialization

Using character models requires training data that are segmented into character shapes. As this is a very time-consuming, difficult, and error-prone job, we implemented an algorithm to segment a word into characters automatically. The initialization of a segmentation into $n \times 7$ segments of a word with n characters is

Fig. 8.21 The Bakis model: a simple left-right model with start and end states. (The Bakis model allows transitions to the same state, the next, and the one after the next)



done by a dynamic programming clustering procedure. This procedure minimizes an appropriate cost function, which ensures a maximum uniformity of feature vectors belonging to one state of a model. The minimization of the mean square error of the feature vectors belonging to the same segment with respect to its mean value is obtained recursively. Finally the initial segments are obtained by applying the backpropagation method.

As a second step the states of the multimodal distributions have to be initialized, which is called initialization of the codebook estimation. This initialization is done in two steps. The first step is the LBG-algorithm [17], which uses the Euclidian distances only. In a second step the EM-algorithm is used to optimize the codebook initialization [21].

These initialization steps are the basis of the subsequent training of the HMM parameters.

Data

The selection of training and testing data is also a very important task. The data must be relevant to the task and sufficient to train all parameters of the HMM and also—with another set—test the quality of the realized system. For this case the IFN/ENIT-database was used for training and testing our system. Each word in this database is labeled not only with the Arabic word but also with a string, which describes the sequence of character shapes of this word (see Table 8.4). This enables the automatic initialization of the segmentation as described earlier in Sect. 8.5.3. The words in the database are not equally distributed, but the words are chosen so that each character shape appears more than 30 times in the training dataset. This ensures a minimum amount of data to make training on character shape level possible.

8.5.4 Training of the HMM

The training of the HMM parameters is done by means of the Viterbi algorithm using a segmental k-means algorithm. The initial codebook is incorporated into the training procedure; that is, in each iteration only the state vector assignment

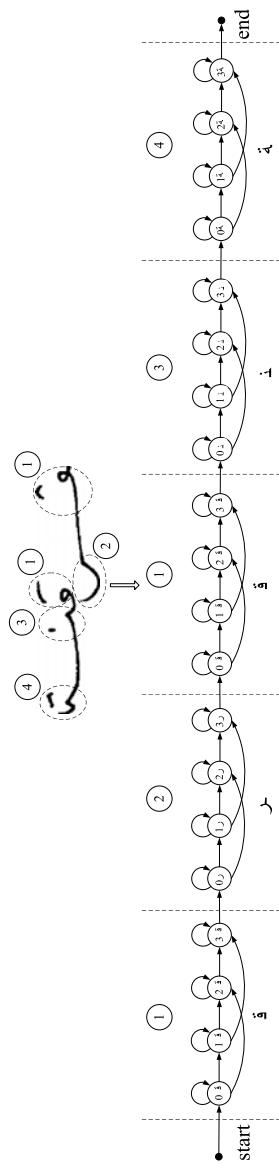
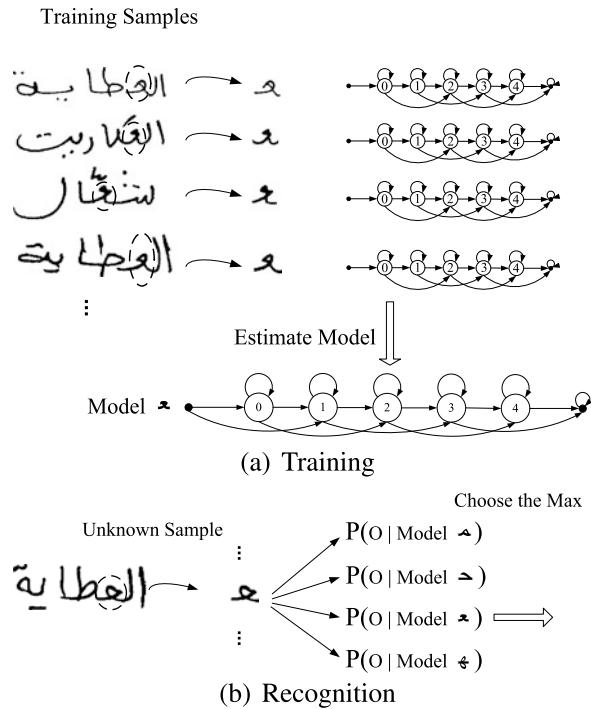


Fig. 8.22 The HMM model for the word Kerkennah (كَرْكَنَة) with a duplication of the 1. character and four models for each character

Fig. 8.23 (a) An HMM is trained for each “mode” using a number of examples of that “mode” from the training set, and (b) is trained to recognize some unknown sequence of “modes.” The likelihood of each model generating that sequence is calculated, and the most likely (maximum) model identifies the sequence



resulting from best path obtained from applying the Viterbi algorithm is used to re-estimate model parameters. As mentioned before, the character shapes are modeled with HMM and for the recognition process concatenated to valid words of the lexicon used. Figure 8.22 shows an example of the concatenation of character models to build a word model. It can be seen that one character model is used twice in the word model. Figure 8.23a gives an example of the training, showing that each character shape of the same type, independent of the word where it was written, contributes to the statistical character shape model. This enables a statistical training with less training data than in the case of word-based models.

8.5.5 Recognition

For recognition, again basically a standard Viterbi algorithm is used. The recognition process has to perform the task to assign to an unknown feature sequence a valid word from the lexicon. The basic way to do this is to calculate the probability that the observation was produced by a state sequence for each word of the lexicon. The sequence with the highest probability gives the correct word. As this procedure is too time consuming, two actions were implemented:

1. A tree-structured lexicon representing valid words is built. This leads to a significant search space reduction and unambiguous assignment of a word to each leaf
2. A beam search strategy is used to reduce the search space. The number of hypotheses generated at each step is controlled by a constant score threshold relative to the currently best score and a maximum number of allowed hypotheses to be active at the same time

Both procedures cause an acceleration of the recognition process but may also lead to a suboptimal solution only. If the parameters of these procedures are selected carefully a good result can be achieved.

8.5.6 Parameter for the HMM-Based Classifier

In the following we summarize the main parameter for our HMM-based classifier. We implemented a semi-continuous hidden Markov model (SCHMM). The code-book size is set to 128. The topology we use is a simple left-right model with start and end states (Bakis model, Fig. 8.21). It allows transitions to the same state, the next, and the one after the next. For example, the number of states and the feature vector dimension depend on the used feature set (cf. Sect. 8.4). We provide these settings here:

	Feature set A	Feature set B
Height of the word after normalization	45	\div^a
Observation window height \times width, shift	$45 \times 5, 2$	$\div \times 6, 3$
Dimension of the feature vector	25	20
Number of states HMM per character	6	4
Entries of the classifier dictionary	937 ^b	937
Feature scale parameter V_σ	\div	0.9

^aIt does not take place at any explicit height normalization

^bMany Tunisian town/village names are due to different ways of writing several times in the dictionary, which results in 2023 entries. The output of the classifier differentiates only 937 place names (on the basis of postcode)

8.6 Results and Discussion

By using the *IFN/ENIT-database* many series of tests are feasible, and some have already been partially made [24]. We focus on and summarize only some of these

Table 8.17 Recognition results: the baseline from GT was used for the normalization procedure (“GT-BLN-FEATURE-A”)

TEST	Training sets	Test set	1-best	2-best	10-best
1	a, b, c	d	91.8	95.2	98.4
2	b, c, d	a	92.2	95.3	98.4
3	c, d, a	b	92.2	95.1	98.6
4	d, a, b	c	92.4	95.3	98.2
–	∅		92.1	95.2	98.4

results in the following sections. In Sect. 8.6.1 we point out which part of the recognition error is caused by a position error of the baseline and which part is caused by an error of the recognizer or something else. Section 8.6.2 discusses the two different normalization and feature extraction methods, while Sect. 8.6.3 deals with the dataset characteristics. Section 8.6.4 takes a closer look at common recognition problems.

8.6.1 Results in Relation to the Estimated Baseline

In this section we focus on the recognition results in relation to the estimated baseline. The idea is to discover to which extent the baseline estimation affects the recognition. Therefore, we generated from the *IFN/ENIT-database* tree datasets. The dataset “GT-BLN-FEATURE-A” was made using the baseline from the database GT for normalization and using feature extraction method A (cf. Sect. 8.4.1). Set “HPROJ-BLN-FEATURE-A” is a set normalized using the horizontal projection histogram baseline estimation approach, while the set “SKL-BLN-FEATURE-A” uses the baseline estimated with the skeleton-based approach. Both sets uses normalization and feature extraction method A.

The results are given in Tables 8.17, 8.18, 8.19, where also the 1-best, 2-best, and 10-best are shown as well as the average recognition rate over all test and training set combinations.

The recognition rate reached with the set “GT-BLN-FEATURE-A” gives the maximal possible recognition rate (92.1 %) in relation to the described system. For set “HPROJ-BLN-FEATURE-A” and set “SKL-BLN-FEATURE-A” a loss in the recognition rate of about 5 % and 4 % respectively can be detected. The loss is significant, and the result correlates with the result for the baseline estimation error rates, where the skeleton-based approach performs slightly better. However, notice that the loss in recognition rate is significant smaller than the loss we have measured in relation to the baseline estimation quality (cf. Sect. 8.3.1). As a result we realize that not each word with an “insufficiently” estimated baseline leads automatically to a wrong word recognition. In Sect. 8.6.4 we will have a closer look at the reasons for word recognition errors.

Table 8.18 Recognition results: the baseline estimated using the horizontal projection approach was used for normalization (“HPROJ-BLN-FEATURE-A”)

TEST	Training sets	Test set	1-best	2-best	10-best
1	a, b, c	d	86.9	90.5	95.2
2	b, c, d	a	87.1	91.1	95.5
3	c, d, a	b	87.1	90.8	95.2
4	d, a, b	c	87.3	90.9	94.9
–	∅		87.3	90.8	95.2

Table 8.19 Recognition results: the baseline estimated using the skeleton-based approach was used for normalization (“SKL-BLN-FEATURE-A”)

TEST	Training sets	Test set	1-best	2-best	10-best
1	a, b, c	d	87.9	91.4	95.6
2	b, c, d	a	88.0	91.5	95.8
3	c, d, a	b	88.0	91.5	95.9
4	d, a, b	c	88.8	91.8	95.5
–	∅		88.2	91.5	95.7

Table 8.20 Recognition results: Tests 1 and 3 give the results obtained using feature set B; Tests 2 and 4 show the corresponding results using feature set A

TEST	Datasets	1-best	2-best	10-best
1	“GT-BLN-FEATURE-B”	90.1	94.2	97.7
2	“GT-BLN-FEATURE-A”	92.1	95.2	98.4
3	“SKL-BLN-FEATURE-B”	87.9	91.0	95.1
4	“SKL-BLN-FEATURE-A”	88.2	91.5	95.7

8.6.2 Results in Relation to the Feature Sets

To determine to which extent the feature extraction method may influence the recognition result, we generated two new datasets from the *IFN/ENIT-database*: dataset “GT-BLN-FEATURE-B”, using the baseline based on the GT together with feature extraction method B, and “SKL-BLN-FEATURE-B”, which uses the baseline estimated by the skeleton-based method. With the dataset “GT-BLN-FEATURE-A” we achieved slightly better results in comparison to dataset “GT-BLN-FEATURE-B”. If we compare “SKL-BLN-FEATURE-B” and “SKL-BLN-FEATURE-A” we cannot find any significant difference (cf. Table 8.20).

We figure out that the features extracted by the different methods (cf. Sect. 8.4) perform more or less the same. Using the baseline from GT for normalization, the

Table 8.21 Recognition results in relation to the used quantity and quality of the training set (“REF-BLN-FEATURE-A”)

TEST	Training sets	Test set	1-best	2-best	10-best
1	a, b, c	d	91.8	95.2	98.4
2	a	d	89.0	93.2	97.7
3	b	d	89.3	94.2	98.1
4	c	d	90.6	94.2	98.1
5	a, b	d	91.5	94.7	98.3
6	a, c	d	91.1	94.7	98.6
7	b, c	d	91.6	95.3	98.2
8	a/2	d	85.7	90.6	96.8

difference is about 2 %; using the estimated baseline, the difference is even less. The last we expected due to the independence of the used normalization method from the topline, which we are not able to estimate reliably.

8.6.3 Results in Relation to the Datasets

The following tests were made to determine to what degree the recognition result depends on the set used for training the HMM-based classifier. For testing the classifier we always used the set d from the *IFN/ENIT-database* normalized and feature extracted with method A and with baseline position from GT (“GT-BLN-FEATURE-A”). As shown in Table 8.21, we trained the HMM with different training sets from the *IFN/ENIT-database*. If we only use one-third of the available sets for training²¹ we notice a loss in recognition rate of about 2 %; using two-thirds of the data it is only about 0.4 %. These results lead us to conclude that the quantity of the *IFN/ENIT-database* is sufficient to train an HMM-based classifier, and the four sets are quite equally divided, so the recognition rates are more or less independent of the training set(s) used (cf. Table 8.21). If we use only one-eighth of the available training set we discover the expected loss in recognition rate.

8.6.4 Discussion of Common Recognition Errors

The best recognition rate was about 92 % using the baseline from database GT for normalization tasks together with feature extraction method A (“GT-BLN-FEATURE-A”). In this section we will have a closer look at the words that represent

²¹The *IFN/ENIT-database* version consists of four equally sized sets, a, b, c, d (cf. Sect. 8.2.4).

the leftover 8 % recognition rate and try to figure out why our supposed recognition system failed for these words. Baseline errors are generally not the main reason for recognition errors due to the used GT, although there is a strong correlation between the “bad” marked baselines in the GT (cf. Sect. 8.2.3) and recognition errors. Looking at the words that tend to be false classified, we have tried to put them into groups. In the following we discuss four main issues which often lead to false classifications:

- (1) Strongly overlapping characters within words or PAWs
- (2) Long connecting lines between characters within words or PAWs
- (3) Words in which all characters do not have ascenders
- (4) Unusual (wrong) ways of writing

Table 8.22 gives some examples of these four main issues.

Vertically overlapping characters often occur in Arabic script. In some cases they produce “new” characters, which are known as ligatures (cf. Table 8.2). The issues which we summarized in (1) are extreme cases which strongly involve the writer (cf. Table 8.22, Nos. 1–8). There are examples where one character overlaps up to five neighboring characters. Due to the fact that features are collected column by column, it becomes obvious that this issue leads to recognition errors.

Long connecting lines between characters are common in Arabic script, where most of the time the characters are combined on their baseline. Issue (2) represents the more extreme examples of this widely used habit (cf. Table 8.22, Nos. 9–14). The main problem seems to be that this writing characteristic occurs too seldom in the *IFN/ENIT-database*, so that the HMM-based recognizer does not have enough relevant training data to adjust the models accordingly.

Issue (3) is a self-made problem and belongs directly to not applying an assumption for the topline estimation, which is used for the height normalization procedure. As discussed in Sect. 8.3.4, we assume the position of the topline in a fixed percentage above the baseline and the top of the word. In these cases, where all characters do not have ascenders, the topline is equivalent with the top of the word (cf. Table 8.22, Nos. 15–18). Normalization and feature extraction method B shows in this case better results, due to the independence from the topline.

Finally issue (4) stands for unusual or simply wrong ways of writing. Even for humans the reading of these words (cf. Table 8.22, Nos. 19–26) turns out to be a difficult task, and some background information about the expected vocabulary (Tunisian town/village names) is needed. Occasionally there are real writing mistakes, like missing characters or a wrong order of characters.

Accidentally disconnected or connected characters within words or PAWs, diacritical marks which are slightly displaced, and similar issues are quite common in Arabic handwriting. Fortunately, these issues seem to be well mapped within the recognition system and do not have any priority in relation to recognition errors.

In the following we focus on the recognition errors more globally. A total of 411 writers have contributed to the *IFN/ENIT-database*. Only 18 of them cause no recognition errors. Also, the error rates are unequally distributed; e.g., 25 % of the writers cause about 50 % of the recognition errors and 50 % of the writers cause

Table 8.22 Recognition errors: examples of wrongly recognized words

No.	handwriting	printed	No.	handwriting	printed
Examples for strongly overlapping characters					
1	منزل بورقيبة المتاجع	منزل بورقيبة المتاجع	2	الروحية	الروحية
3	أم العظام	أم العظام	4	حرق	حرق
5	بن عروس	بن عروس	6	صحيح	صحيح
7	قصر المراة	قصر المراة	8	ربيانة	ربيانة
Examples for very long connecting lines between characters					
9	فندق مارينا	فندق مارينا	10	شعاع	شعاع
11	حي الانطلاقة	حي الانطلاقة	12	كسرى	كسرى
13	تونس ثامر	تونس ثامر	14	التلasse	التلasse
Examples of words, where all characters do not have ascenders					
15	سيدي مسعود	سيدي مسعود	16	دغمون	دغمون
17	ذهيبة	ذهيبة	18	سيسب	سيسب
Examples of unusual (wrong) ways of writing					
19	سوسة ابن خلدون	سوسة ابن خلدون	20	الخليج	الخليج
21	منزل حشاد	منزل حشاد	22	حي المهرجان	حي المهرجان
23	رباعي سيدى ظاهر	رباعي سيدى ظاهر	24	الفحص	الفحص
25	برج المظيلة	برج المظيلة	26	الفايض	الفايض

about 80 % of the errors. That leads to the conclusion that there are some styles of handwriting which are very different from “standard” handwriting.

After looking at the length of words and comparing this information with the relative error rate, we noticed that short words are disproportionately responsible for recognition errors. Words with only one PAW have a relative error rate of about 20 %, words with two PAWs have an error rate of about 15 %, and words with five PAWs reach a relative error rate of less than 3 %. There are probably two main reasons for this: first, short words contain indisputably less information than longer words and, second, there are many quite similar words in the dictionary of the recognition system. Obviously, both reasons complicate the classification task.

In the final analysis one notices that the vast majority of the recognition errors belong to a suboptimal adaptation to certain characteristics of Arabic handwriting. Apart from this the recognition system shows that it is able to cope with many difficulties quite impressively.

8.7 Conclusion

Handwritten word recognition is still a challenging task. The development of a recognition system needs to optimize many parts or modules to obtain good or even acceptable results. A very important role is played by the database that is used. We have presented the *IFN/ENIT-database*, a well-organized database with very detailed information about character shape, ligatures, and baseline position. This information helps to evaluate the quality of different modules on their output and not on the recognition result only. The database, normalization, and baseline estimation were described in detail. Using the information in the database together with a distance metric, two different baseline estimation methods were developed and evaluated. We showed that the skeleton-based approach performs much better than the frequently used projection approach. Two totally different feature sets were tested using the HMM-based recognizer. It was interesting to see that both approaches reach more or less the same results and that these results are comparable to those of state-of-the-art systems (ICDAR 2005 competition [23]).

References

1. Abandah, G., Jamour, F.: Recognizing handwritten Arabic script through efficient skeleton-based grapheme segmentation algorithm. In: Proceedings of the 10th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 977–982 (2010)
2. Abandah, G., Malas, T.: Feature selection for recognizing handwritten Arabic letters. *Dirasat Eng. Sci.* **37**(2), 242–256 (2010)
3. Al-Hajj, R., Likforman-Sulem, L., Mokbel, C.: Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(7), 1165–1177 (2009)
4. Capson, D.W.: An improved algorithm for the sequential extraction of boundaries from a raster scan. *Comput. Vis. Graph. Image Process.* **28**, 109–125 (1984)
5. Chen, J., Cao, H., Prasad, R., Bhardwaj, A., Natarajan, P.: Gabor features for offline Arabic handwriting recognition. In: Proceedings of the 9th International Workshop on Document Analysis Systems (DAS), pp. 53–58 (2010)
6. Dehghan, M., Faez, K., Ahmadi, M., Shridhar, M.: Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM. *Pattern Recognit.* **34**(5), 1057–1065 (2001)
7. Ding, X., Jin, J., Wang, H., Peng, L.: Printed Arabic document recognition system. In: Proceedings of SPIE—Document Recognition and Retrieval XII, vol. 5676, pp. 48–55 (2005)

8. Dreuw, P., Jonas, S., Ney, H.: White-space models for offline Arabic handwriting recognition. In: Proceedings of the International Conference on Pattern Recognition (ICPR), pp. 1–4 (2008)
9. Dreuw, P., Heigold, G., Ney, H.: Confidence-based discriminative training for model adaptation in offline Arabic handwriting recognition. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), pp. 596–600 (2009)
10. El Abed, H., Märgner, V.: Comparison of different pre-processing methods for offline recognition of handwritten Arabic words. In: Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR), vol. 2, pp. 974–978 (2007)
11. El Abed, H., Märgner, V.: Improvement of Arabic handwriting recognition systems—combination and/or reject? In: Proceedings of the Document Recognition and Retrieval XVI. SPIE Proc., vol. 7247, pp. 10 (2009)
12. El Abed, H., Märgner, V.: ICDAR 2009—Arabic handwriting recognition competition. *Int. J. Doc. Anal. Recognit.* **14**(1), 3–13 (2011). Special Issue on Performance Evaluation
13. Elbaati, A., Kherallah, M., El Abed, H., Ennaji, A., Alimi, A.M.: Arabic handwriting recognition using restored stroke chronology. In: International Conference on Document Analysis and Recognition (ICDAR) (2009)
14. Ferreira, A., Ubeda, S.: Ultra fast parallel contour tracking with application to thinning. *Pattern Recognit.* **27**(7), 867–878 (1994)
15. Graves, A.: Supervised sequence labelling with recurrent neural networks. Ph.D. thesis, Fakultät für Informatik—Technische Universität München (2007)
16. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in Neural Information Processing Systems, vol. 21 (2009)
17. Gray, R.A.: Vector quantization. *IEEE ASSP Mag.* **1**, 4–28 (1984)
18. Hamdani, M., El Abed, H., Kherallah, M., Alimi, A.M.: Combining multiple HMMs using on-line and off-line features for off-line Arabic handwriting recognition. In: Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR), pp. 201–205 (2009)
19. Hamdani, M., El Abed, H., Hamdani, T.M., Märgner, V., Alimi, A.M.: Improving a HMM-based off-line handwriting recognition system using MME-PSO optimization. In: Proceedings of the Document Recognition and Retrieval XVIII (2011)
20. Heigold, G., Deselaers, T., Schlüter, R., Ney, H.: Modified MMI/MPE: a direct evaluation of the margin in speech recognition. In: Proceedings of the International Conference on Machine Learning, pp. 384–391 (2008)
21. Huang, X.D., et al.: Hidden Markov Models for Speech Recognition. Edinburgh Universal Press, Edinburgh (1990)
22. Impedovo, S., Ottiviano, L., Occhinegro, S.: Optical character recognition—a survey. *Int. J. Pattern Recognit. Artif. Intell.* **5**(1), 1–24 (1991)
23. Märgner, V., Pechwitz, M., El-Abed, H.: ICDAR 2005 Arabic handwriting recognition competition. In: Eighth International Conference on Document Analysis and Recognition (ICDAR'05), vol. 1, pp. 70–74 (2005)
24. Pechwitz, M.: Automatische Erkennung handgeschriebener arabischer Wörter. Ph.D. thesis, Technische Universität Braunschweig, Germany (2005)
25. Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT—database of handwritten Arabic words. In: Proc. of Colloque International Francophone sur l'Ecrit et le Document, CIFED 2002, Hammamet, Tunisia, October 21–23, 2002, pp. 129–136 (2002)
26. Plamondon, R., Srihari, S.N.: On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(1), 63–84 (2000)

27. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. In: Waibel, A., Lee, K.-F. (eds.) *Readings in Speech Recognition*, pp. 267–296. Morgan Kaufmann, San Mateo (1990)
28. Zampneroni, P.: *Methoden der Digitalen Bildsignalverarbeitung*. Vieweg, Braunschweig (1989)
29. Zampneroni, P., Klette, R.: *Handbuch der Operatoren der Bildverarbeitung*. Vieweg, Braunschweig (1995)

Chapter 9

RWTH OCR: A Large Vocabulary Optical Character Recognition System for Arabic Scripts

Philippe Dreuw, David Rybach, Georg Heigold, and Hermann Ney

Abstract We present a novel large vocabulary OCR system, which implements a confidence- and margin-based discriminative training approach for model adaptation of an HMM-based recognition system to handle multiple fonts, different handwriting styles, and their variations. Most current HMM approaches are HTK-based systems which are maximum likelihood (ML) trained and which try to adapt their models to different writing styles using writer adaptive training, unsupervised clustering, or additional writer-specific data. Here, discriminative training based on the maximum mutual information (MMI) and minimum phone error (MPE) criteria are used instead. For model adaptation during decoding, an unsupervised confidence-based discriminative training within a two-pass decoding process is proposed. Additionally, we use neural network-based features extracted by a hierarchical multi-layer perceptron (MLP) network either in a hybrid MLP/HMM approach or to discriminatively retrain a Gaussian HMM system in a tandem approach. The proposed framework and methods are evaluated for closed-vocabulary isolated handwritten word recognition on the *IFN/ENIT-database* Arabic handwriting database, where the word error rate is decreased by more than 50 % relative to an ML trained baseline system. Preliminary results for large vocabulary Arabic machine-printed text recognition tasks are presented on a novel publicly available newspaper database.

P. Dreuw (✉) · D. Rybach · G. Heigold · H. Ney

Human Language Technology and Pattern Recognition, RWTH Aachen University, Ahornstr 55,
52056 Aachen, Germany

e-mail: Dreuw@cs.rwth-aachen.de

D. Rybach

e-mail: Rybach@cs.rwth-aachen.de

G. Heigold

e-mail: Heigold@cs.rwth-aachen.de

H. Ney

e-mail: Ney@cs.rwth-aachen.de

9.1 Introduction

In this work, we describe our novel large vocabulary optical character recognition (OCR) system. Our hidden Markov model (HMM)-based RWTH ASR system is based on a publicly available state-of-the-art large vocabulary continuous speech recognition (LVCSR) framework which has been designed for the special requirements of research applications and support for grid computing.

The aim of this work is to analyze for Arabic handwriting and machine-printed text recognition tasks the effect of discriminative maximum mutual information (MMI)/minimum phone error (MPE) training and the incorporation of a margin and a confidence term into discriminative criteria. Therefore, none of the pre-processing steps commonly applied in handwriting recognition like binarization, deskewing, deslanting, or size normalization are used.

The focus of this work shall be on offline handwriting recognition of closed-vocabulary isolated Arabic words and large open-vocabulary machine-printed Arabic text recognition tasks in combination with n-gram language models. More explicitly, the novelties of our investigation are as follows:

1. Conversion of a state-of-the-art large vocabulary speech recognition framework for handwritten and machine-printed OCR.
2. Analysis of offline handwritten and machine-printed Arabic text recognition.
3. Direct evaluation of the utility of the margin term in MMI/MPE-based training. Ideally, we can turn on/off the margin term in the optimization problem.
4. Direct evaluation of the utility of an additional confidence term. Ideally, we improve over the best trained system by retraining the system with unsupervised labeled test data.
5. Evaluation on state-of-the-art systems. Ideally, we directly improve over the best discriminative system, e.g., conventional (i.e., without margin) MMI/MPE for handwriting recognition.
6. Evaluation of hybrid multi-layer perceptron (MLP)/HMM and discriminatively retrained MLP-Gaussian HMM (GHMM) tandem approaches.

The remainder of this chapter is structured as follows. First, the background is described in Sect. 9.2. Next, Sect. 9.3 gives a system overview, and then the RWTH ASR software framework is presented in Sect. 9.4. The datasets we used for evaluating the proposed framework are explained in Sect. 9.5; in particular our ongoing work in creating a publicly available database for Arabic machine-printed text recognition is presented in Sect. 9.5.2. Experimental results are presented in Sect. 9.6, and the chapter is concluded in Sect. 9.7.

9.2 Background

From a system point of view, many approaches for Arabic handwriting recognition [19] in the past were HMM-based systems using the Hidden Markov Model Toolkit

(HTK) [78]. BBN’s glyph HMM system “Byblos” [44, 50, 67] has been extended to “PLATO” [49] within the MADCAT [56] project, and is used for handwriting and machine-printed OCR tasks. SIEMENS [70] showed how to convert a Latin OCR system to Arabic handwriting. Other projects like OCropus¹ or Tesseract² currently do not support the recognition of Arabic scripts, and apparently only a few commercial applications like Readiris³ and NovoDynamics VERUS⁴ can support those cursive scripts.

Many commercial machine-printed OCR products or systems described in the published literature developed their recognition algorithms on isolated characters [43]. These systems usually assumed that characters can be segmented accurately as a first step, and made hard decisions at each stage which resulted in an accumulation of errors; thus broken and touching characters were responsible for the majority of the errors. Obviously, these assumptions are too strong for degraded or handwritten documents, or font-free approaches [35].

Such approaches were surpassed by late-decision systems, e.g., tools developed by the speech recognition community, such as hidden Markov models (HMMs). In these systems, multiple hypotheses about both segmentations and identities are maintained, and the final decisions are made at the end of an observation sequence by tracing back the local decisions which led to the best global hypothesis [32]. Similar to the framework presented in [50, 67] our novel RWTH ASR system is able to recognize Arabic handwritten *and* machine-printed text.

State-of-the-art speech recognition systems are based on discriminative Gaussian HMMs (GHMMs), where major points of criticism of this conventional approach are the indirect parameterization of the posterior model, the nonconvexity of the conventional training criteria, and the insufficient flexibility of the HMMs to incorporate additional dependencies and knowledge sources [30]. State-of-the-art handwritten text recognition systems are usually based on HMMs too [5, 21, 70], but are typically trained using the maximum likelihood (ML) criterion. Hybrid neural network-based systems like RNN/CTC [25] and MLPs/HMM [20], or tandem-based approaches like MLP-GHMM [71] were recently very successful in online and offline handwriting recognition. However, most of the tandem-based approaches use an ML-based training criterion to retrain the GHMMs.

Typical training criteria for string recognition like, for example, minimum phone error (MPE) and maximum mutual information (MMI) in speech recognition are based on a (regularized) loss function. In contrast, large margin classifiers—the de facto standard in machine learning—maximize the separation margin. An additional loss term penalizes misclassified samples.

The MMI training criterion has been used in [54] to improve the performance of an HMM-based offline Thai handwriting recognition system for isolated characters. The authors propose a feature extraction based on a block-based principal

¹<http://code.google.com/p/ocropus/>

²<http://code.google.com/p/tesseract-ocr/>

³<http://www.irislink.com/readiris/>

⁴<http://www.novodynamics.com/>

component analysis (PCA) and composite image features, which are reported to be better at discriminating Thai confusable characters. In [6], the authors apply the minimum classification error (MCE) criterion to the problem of recognizing on-line unconstrained-style characters and words, and report large improvements on a writer-independent character recognition task when compared to an ML trained baseline system.

Similar to systems presented in [10, 49, 53], we apply the MMI/MPE criterion, but modified by a margin term. This margin term can be interpreted as an additional observation-dependent prior weakening the true prior [33], and is identical with the support vector machine (SVM) optimization problem of log-linear models [27].

The most common method for unsupervised adaptation is the use of the automatic transcription of a previous recognition pass without the application of confidence scores. Many publications in automatic speech recognition (ASR) have shown that the application of confidence scores for adaptation can improve recognition results. However, only small improvements are reported for maximum likelihood linear regression (MLLR) adaptation [23, 59, 62] or confidence-based constrained MLLR (CMLLR) adaptation [3]. In addition to the margin concept, the MMI/MPE training criteria are extended in this work by an additional confidence term [14] to allow for novel unsupervised model adaptation.

9.3 System Overview

In offline handwriting recognition, we are searching for an unknown word sequence $w_1^N := w_1, \dots, w_N$, for which the sequence of features $x_1^T := x_1, \dots, x_T$ fits best to the trained models. We maximize the posterior probability $p(w_1^N | x_1^T)$ over all possible word sequences w_1^N with unknown number of words N . This is modeled by the Bayes decision rule:

$$x_1^T \rightarrow \hat{w}_1^N(x_1^T) = \arg \max_{w_1^N} \{ p^\kappa(w_1^N) p(x_1^T | w_1^N) \} \quad (9.1)$$

with κ being a scaling exponent of the language model.

Especially in Arabic handwriting with its position-dependent glyphs [42], large white spaces can occur between isolated-, beginning-, and end-shaped characters (see Fig. 9.1(a)). As a specific set of characters is only connectable from the right side, such words have to be cut into parts (part of Arabic word (PAW)). Due to the ligatures and diacritics in Arabic handwriting, the same Arabic word can be written in several writing variants, depending on the writer's handwriting style.

In this work, we use a writing variant model refinement [15] of our visual model,

$$p(x_1^T | w_1^N) = \max_{v_1^N | w_1^N} \{ p_{\Lambda_v}^\alpha(v_1^N | w_1^N) p_{\Lambda_{e,t}}^\beta(x_1^T | v_1^N, w_1^N) \} \quad (9.2)$$

with v_1^N a sequence of unknown writing variants, α a scaling exponent of the writing variant probability depending on a parameter set Λ_v , and β a scaling exponent of

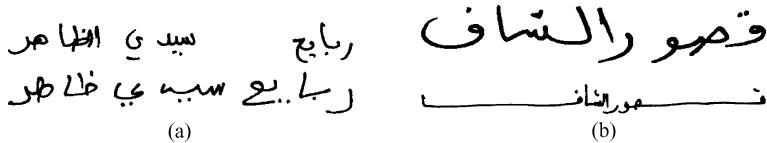


Fig. 9.1 Two examples where each column shows the same Tunisian town name: large white spaces (a) and a large stretching of long drawn-out characters (b) occur often in Arabic handwriting. Therefore, an adequate modeling of white spaces and state-transition penalties must be part of an HMM-based Arabic handwriting recognition system

the visual model depending on a parameter set $\Lambda_{e,t}$ for the emission and transition model.

During training, a corpus and lexicon with supervised writing variants instead of the commonly used unsupervised writing variants can be used; during decoding, the writing variants can only be used in an unsupervised manner. Obviously, the supervised writing variants in training can lead to better trained glyph models only if the training corpora have a high annotation quality. Usually, the probability $p(v|w)$ for a variant v of a word w is considered as uniformly distributed [13]. Here we use the count statistics as probability,

$$p(v|w) = \frac{N(v, w)}{N(w)} \quad (9.3)$$

where the writing variant counts $N(v, w)$ and the word counts $N(w)$ are estimated from the corresponding training corpora, and represent how often these events were observed. Note that $\sum_{v'} \frac{N(v', w)}{N(w)} = 1$. The scaling exponent α of the writing variant probability of Eq. (9.2) can be adapted in the same way as is done for the language model scale κ in (9.1).

9.3.1 Feature Extraction

The images are scaled down to a fixed height while keeping their aspect ratio. We extract simple appearance-based image slice features x'_t at every time step $t = 1, \dots, T$ which are augmented by their spatial derivatives in the horizontal direction $\Delta = x'_t - x'_{t-1}$. Note that many systems divide the sliding window itself into several subwindows and extract different features within each of the subwindows [4, 34, 54, 70].

In order to incorporate temporal and spatial context into the features, we concatenate seven consecutive features in a sliding window with maximum overlap, which are later reduced by a PCA transformation matrix to a feature vector x_t of dimension 30 (see Fig. 9.2).

Without any pre-processing of the input images, the simple appearance-based image slice features $x_t = [x'_t, \Delta]$ together with their corresponding state alignments can then be processed by a hierarchical MLP framework originally described in

Fig. 9.2 Right-to-left sliding PCA window over input images without any pre-processing for Arabic handwriting

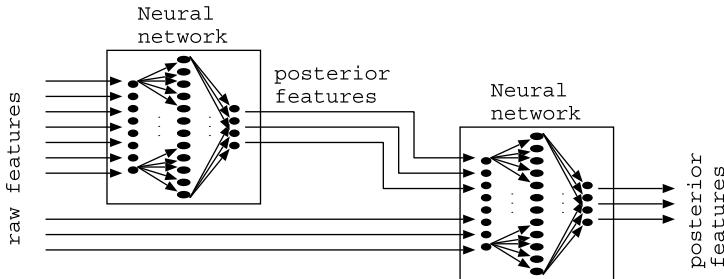
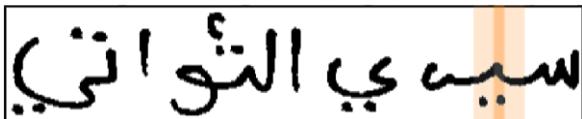


Fig. 9.3 Hierarchical MLP network for discriminative feature extraction in OCR

[76]. Depending on the MLP hierarchy, pre-processing, and post-processing operations, several feature sets can be generated. In order to incorporate temporal and spatial context into the features, we concatenate consecutive features in a sliding window, where the MLP outputs are later reduced by a PCA or a linear discriminant analysis (LDA) transformation (see Fig. 9.3). Two different MLPs are trained, raw and TRAP-DCT networks; the network details are given in Sect. 9.6.

Instead of using log-PCA/LDA reduced MLP posterior features for retraining a Gaussian HMM system, log-posterior features can be directly used without any reduction in a hybrid MLP/HMM framework [7], as briefly described in Sect. 9.3.2.

9.3.2 Visual Modeling

Arabic Handwriting Depending on the position in an Arabic word, most of the 28 characters can have up to four different shapes [42]. Here we use position-dependent glyph models to model the different presentation forms, and due to ligatures, a total of 120 glyph models and one white-space model have to be estimated for the *IFN/ENIT-database* tasks (see Sect. 9.6). Additionally, a large stretching of long drawn-out glyphs occurs often in Arabic handwriting (see Fig. 9.1(b)). Therefore, we use very low loop penalties but higher skip penalties for our HMM state transitions (see Fig. 9.4(a)).

Arabic Machine-Printed Text As for Arabic handwriting, there are no distinct upper and lower case letter forms in machine-printed texts. Both printed and written Arabic are cursive. Unlike cursive writing based on the Latin alphabet, the standard Arabic style has substantially different shapes depending on the glyph context. Standard Arabic Unicode character encodings typically do not indicate the form

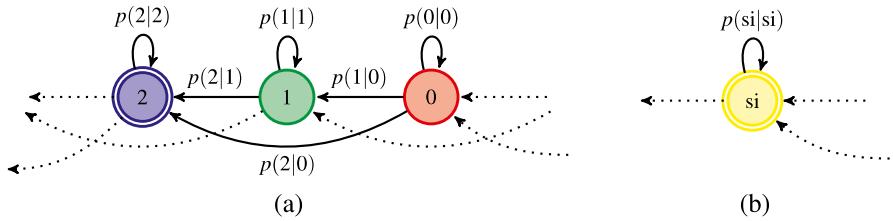


Fig. 9.4 Different HMM topologies and transition probabilities are used for character models (a) and white-space models (b) in Arabic and Latin handwriting recognition

each character should take in context, so it is left to the rendering engine to select the proper glyph to display for each character.

The basic Arabic range encodes mainly the standard letters and diacritics. For our novel large vocabulary Arabic machine-printed text database described in Sect. 9.5.2, about 200 position-dependent glyph models have to be trained.

GHMM Our hidden Markov model (HMM)-based OCR system is Viterbi trained using the maximum likelihood (ML) training criterion and a lexicon with multiple writing variants as proposed in [13, 15].

Each glyph is modeled by a multi-state left-to-right HMM with skip transitions and separate Gaussian mixture models (GHMMs) with globally pooled variances. The parameters of all Gaussian mixture models (GMMs) are estimated with the ML principle using an expectation maximization (EM) algorithm, and to increase the number of densities in the mixture densities, successive splitting of the mixture densities is applied. Different HMM topologies and transition probabilities are used for glyph models (see Fig. 9.4(a)) and white-space models (Fig. 9.4(b)) in Arabic text recognition, where the white-space model itself is always modeled by a single GMM in all systems.

The ML trained GMMs are refined using a discriminative training approach based on the margin-based M-MMI/M-MPE criteria [28] as briefly presented in Sect. 9.3.3.

Hybrid MLP/HMM The MLP posterior probabilities $p(s_t|x_t)$ are divided by the prior state probabilities $p(s_t)$ in order to approximate the observation probabilities of an HMM, i.e. $p(x_t|s_t) \approx \frac{p(s_t|x_t)}{p(s_t)}$ as described in [7].

MLP-GHMM The MLP-GHMM system is trained from scratch using the MLP log-posterior features as described in Sect. 9.3.1 (also known as the tandem approach [71]). Again, ML/M-MMI/M-MPE training criteria can be used for GMM training. Note that the MLP network itself can also be trained using different alignments generated by the correspondingly trained GHMM systems.

9.3.3 Discriminative Training: Incorporation of the Margin and Confidence Term

In this work, we use a discriminative training approach based on the maximum mutual information (MMI) and minimum phone error (MPE) criteria as presented in [26, 27, 29]. In addition to the novel confidence-based extension of the margin-based MMI training presented in [14], the confidence concept has been incorporated in the margin-based MPE criterion in this work.

The proposed approach takes advantage of the generalization bounds of large margin classifiers while keeping the efficient framework for conventional discriminative training. This allows us to directly evaluate the utility of the margin term for OCR. So, our approach combines the advantages of conventional training criteria and of large margin classifiers.

This section briefly reviews how the MMI/MPE training criteria can be extended to incorporate the margin concept, and that such modified training criteria are smooth approximations to support vector machines with the respective loss function [27].

In OCR, the two-dimensional representation of an image is turned into a string representation $X = x_1, \dots, x_T$ where x_t is a fixed-length array assigned to each column in the image (see Sect. 9.3.1 for further details). The word sequence $W = w_1, \dots, w_N$ is represented by a character string.

Assume the joint probability $p_\Lambda(X, W)$ of the features X and the symbol string W . The model parameters are indicated by Λ . The training set consists of $r = 1, \dots, R$ labeled sentences, $(X_r, W_r)_{r=1, \dots, R}$. According to the Bayes rule, the joint probability $p_\Lambda(X, W)$ induces the posterior

$$p_{\Lambda, \gamma}(W|X) = \frac{p_\Lambda(X, W)^\gamma}{\sum_V p_\Lambda(X, V)^\gamma} \quad (9.4)$$

The likelihoods are scaled with some factor $\gamma > 0$, which is a common trick in speech recognition to scale them to the “real” posteriors [29]. The approximation level γ is an additional parameter to control the smoothness of the criterion.

Let $p_\Lambda(X, W)$ be the joint probability and L a loss function for each training sample r :

$$L[p_\Lambda(X_r, \cdot), W_r] \quad (9.5)$$

with \cdot representing all possible hypotheses W for a given lexicon, and W_r representing the correct transcription of X_r .

The general optimization problem can be formulated as a minimization of the total loss function:

$$\hat{\Lambda} = \arg \min_{\Lambda} \left\{ C \|\Lambda - \Lambda_0\|_2^2 + \sum_{r=1}^R L[p_\Lambda(X_r, \cdot), W_r] \right\} \quad (9.6)$$

and includes an ℓ_2 regularization term $\|\Lambda - \Lambda_0\|_2^2$ (i.e. a prior over the model parameters), where the constant C is used to balance the regularization term and the loss term including the log-posteriors. Here, the ℓ_2 regularization term is replaced by I-smoothing [65], which is a useful technique to make MMI/MPE training converge without over-training, and where the parameter prior is centered for initialization at a reasonable ML trained model Λ_0 (see Sect. 9.3.2).

Maximum Mutual Information

In automatic speech recognition (ASR), MMI commonly refers to the maximum likelihood (ML) for the class posteriors. For MMI, the loss function to be minimized is described by

$$L^{(\text{MMI})}[p_\Lambda(X_r, \cdot), W_r] = -\log \frac{p_\Lambda(X_r, W_r)^\gamma}{\sum_V p_\Lambda(X_r, V)^\gamma} \quad (9.7)$$

This criterion has proven to perform reasonably as long as the error rate on the training data is not too low, i.e., generalization is not an issue.

Margin-Based Maximum Mutual Information

Conventional MMI is based on the true posteriors in Eq. (9.4). The margin-based MMI (M-MMI) loss function to be minimized is described by

$$L_\rho^{(\text{M-MMI})}[p_\Lambda(X_r, \cdot), W_r] = -\log \frac{[p_\Lambda(X_r, W_r) \exp(-\rho A(W_r, W_r))]^\gamma}{\sum_V [p_\Lambda(X_r, V) \exp(-\rho A(V, W_r))]^\gamma} \quad (9.8)$$

which has an additional margin term including the word accuracy $A(\cdot, W_r)$ based on the approximate word error [65]. Note that the additional term can be interpreted as if we had introduced a new posterior distribution. In a simplified view, we interpret this as a pseudo-posterior probability which is modified by a margin term.

Compared with the true posterior in Eq. (9.4), the M-MMI loss function includes the margin term $\exp(-\rho A(V, W_r))$, which is based on the string accuracy $A(V, W_r)$ between the two strings V, W_r . The accuracy counts the number of matching symbols of V, W_r and will be approximated for efficiency reasons (see Sect. 9.3.3).

As explained in [29], the accuracy is generally scaled with some $\rho > 0$, and this term weighs up the likelihoods of the competing hypotheses compared with the correct hypothesis [66]. However, this term can be equally interpreted as a margin term.

Minimum Phone Error

The MPE criterion is defined as the (regularized) posterior risk based on the error function $E(V, W)$ like, for example, the approximate phone error [64], which

is probably the training criterion of choice in large vocabulary continuous speech recognition (LVCSR). For MPE, the loss function to be minimized is described by

$$L^{(\text{MPE})}[p_A(X_r, \cdot), W_r] = \sum_{W \in \cdot} E(W, W_r) \frac{p_A(X_r, W_r)^\gamma}{\sum_V p_A(X_r, V)^\gamma} \quad (9.9)$$

In OCR, a phoneme unit usually corresponds to a glyph if words are modeled by glyph sequences.

Margin-Based Minimum Phone Error

Analogously, the margin-based MPE (M-MPE) loss function to be minimized is described by

$$\begin{aligned} L_\rho^{(\text{M-MPE})}[p_A(X_r, \cdot), W_r] \\ = \sum_{W \in \cdot} E(W, W_r) \frac{[p_A(X_r, W_r) \exp(-\rho A(W, W_r))]^\gamma}{\sum_V [p_A(X_r, V) \exp(-\rho A(V, W_r))]^\gamma} \end{aligned} \quad (9.10)$$

It should be noted that due to the relation $E(V, W) = |W| - A(V, W)$ where $|W|$ denotes the number of symbols in the reference string, the error $E(V, W)$ and the accuracy $A(V, W)$ can be equally used in Eqs. (9.9) and (9.10). The accuracy for MPE and that for the margin term do not need to be the same quantity [26].

Finally, it should be pointed out that other posterior-based training criteria (e.g. MCE as used in [6]) can be modified in an analogous way to incorporate a margin term (for more details, cf. [27, 29]).

Optimization

In [27] it is shown that the objective function $\mathcal{F}_\gamma^{(\text{MMI})}(\Lambda)$ converges pointwise to the SVM optimization problem using the hinge loss function for $\gamma \rightarrow \infty$, similar to [79]. In other words, $\mathcal{F}_\gamma^{(\text{M-MMI})}(\Lambda)$ is a smooth approximation to an SVM with hinge loss function which can be iteratively optimized with standard gradient-based optimization techniques like Rprop [27, 79].

In this work, the regularization constant C , the approximation level γ , and the margin scale ρ are chosen beforehand and then kept fixed during the complete optimization. Note that the regularization constant C and the margin scale ρ are not completely independent of each other. Here, we kept the margin scale ρ fixed and tuned the regularization constant C . Previous experiments in ASR have suggested that the performance is rather insensitive to the specific choice of the margin [27], and the results in [16] furthermore suggest that the choice of the I-smoothing constant C has less impact in an Rprop-based optimization than in an extended Baum-Welch (EBW) environment [65]. An I-smoothing regularization constant $C = 1.0$ is used in all results presented in Sect. 9.6.

In large vocabulary OCR, word lattices restricting the search space are used to make the summation over all competing hypotheses (i.e. sums over W) efficient. The exact accuracy on character or word level cannot be computed efficiently due to the Levenshtein alignments in general, although it is feasible under certain conditions as shown in [26]. Thus, the approximate character/word accuracy known from MPE/MWE [64] is used for the margin instead. With this choice of accuracy, the margin term can be represented as an additional layer in the common word lattices such that efficient training is possible. More details about the transducer-based implementation used in this work can be found in [26].

As in ASR, where typically a weak unigram language model is used for discriminative training [72, 73], we use a unigram language model in our proposed discriminative training criteria.

Confidences for Unsupervised Discriminative Model Adaptation

Sentence or word confidences can be incorporated into the training criterion by simply weighing the segments with the respective confidence. This is, however, not possible for state-based confidences. Instead of rejecting an entire sentence or word, the system can use state confidence scores to select state-dependent data in an unsupervised manner. State confidence scores are obtained from computing arc posteriors from the lattice output from a previous decoder pass.

Rprop is a gradient-based optimization algorithm. The gradient of the training criterion under consideration can be represented in terms of the state posteriors $p_{rt}(s|x_1^{T_r})$. These posteriors are obtained by marginalization and normalization of the joint probabilities $p_A(x_1^{T_r}, s_1^T, w_1^{N_r})$ over all state sequences through state s at frame t . These quantities can be calculated efficiently by recursion, e.g., forward/backward probabilities. Then, the state-based confidences $c_{r,s,t}$ are incorporated by multiplying the posteriors with the respective confidence before the accumulation. In summary, each frame t contributes $\cdot p_{rt}(s|x_1^{T_r}) \cdot c_{r,s,t} \cdot x_t$ to the accumulator acc_s of state s .

Another way to describe the incorporation of the confidence term into the margin pseudo-posteriors is from a system point of view. The accumulator acc_s of state s can be described by

$$acc_s = \sum_{r=1}^R \sum_{t=1}^{T_r} \omega_{r,s,t} \cdot x_t,$$

where the weight $\omega_{r,s,t}$, which is equal to $\delta(s_t, s)$ in ML training, is replaced for the proposed M-MMI-conf/M-MPE-conf criteria (with $\rho \neq 0$) by the mar-

gin modified pseudo-posteriors of the corresponding loss functions. The additional confidence term for the proposed M-MMI-conf criterion can be described as follows:

$$\omega_{r,s,t} := \underbrace{\frac{\sum_{s_1^{T_r}:s_t=s} [p(x_1^{T_r}|s_1^{T_r}) p(s_1^{T_r}) p(W_r) \cdot e^{-\rho\delta(W_r, W_r)}]^\gamma}{\sum_V \sum_{s_1^{T_r}:s_t=s} [p(x_1^{T_r}|s_1^{T_r}) p(s_1^{T_r}) p(V) \cdot e^{-\rho\delta(V, W_r)}]^\gamma}}_{\text{posterior}} \underbrace{\cdot \delta(c_{r,s,t} \geq c_{\text{threshold}})}_{\text{confidence selection}} \quad (9.11)$$

Here, the selector function $\delta(c_{r,s,t} > c_{\text{threshold}})$ with the parameter $c_{\text{threshold}}$ controls the amount of adaptation data. The M-MPE-conf criterion can be defined in a similar manner. Note that due to the quality of the confidence metric, thresholding the confidence scores after feature selection can often result in an improved accuracy, as reported in [23]. On the one hand, the experimental results for word confidences in Fig. 9.9 and state-based confidences in [16] suggest that the confidences are helpful, but on the other hand it seems that the threshold itself has little impact due to the proposed M-MMI-conf/M-MPE-conf approaches, which are inherently robust against outliers.

Analogously, the weight $\omega_{r,s,t}$ would correspond to the true posterior (Eq. (9.4)) in an MMI-conf/MPE-conf criterion. According to [11, 16, 17] these criteria lead to no robust improvements; i.e. only the combination of margin *and* confidences makes the proposed approaches robust against outliers.

9.3.4 Writer Adaptive Training

Writer variations are compensated by writer adaptive training (WAT) [15] using constrained maximum likelihood linear regression (CMLLR) [22] to train writer-dependent models. The available writer labels of the *IFN/ENIT-database* are used in training to estimate the writer-dependent CMLLR feature transformations. The parameters of the writer adapted Gaussian mixtures are trained using the CMLLR transformed features. During decoding, unsupervised writer clustering with a Bayesian information criterion-based stopping condition for a CMLLR-based feature adaptation during a two-pass decoding process is used to cluster different handwriting styles of unknown test writers (see Sect. 9.3.5). It can be seen from the writer statistics in Table 9.1 that the number of different writers in set e is higher than in all other subsets; thus the variation of handwriting styles. In machine-printed text recognition, the same approach could be applied to font labels available in the RAMP-N corpora (see Sect. 9.5.2).

9.3.5 Decoding Architecture

The recognition is performed in multiple passes. For model adaptation toward unknown data or unknown writing styles, the output of the first recognition pass (best word sequences or word lattices) can be either used for discriminative model adaptation or writer adaptation. Although the automatically generated transcript may contain errors, adaptation using that transcript generally results in accuracy improvements [23]. The adaptation techniques used are explained in the following sections.

Discriminative Model Adaptation

The model adaptation can be carried out by discriminatively training writer-dependent models using the word sequences obtained by the first recognition pass. Additionally, the confidence alignments generated during the first-pass decoding can be used on a sentence, word, or state level to exclude the corresponding features from the discriminative training process for unsupervised model adaptation.

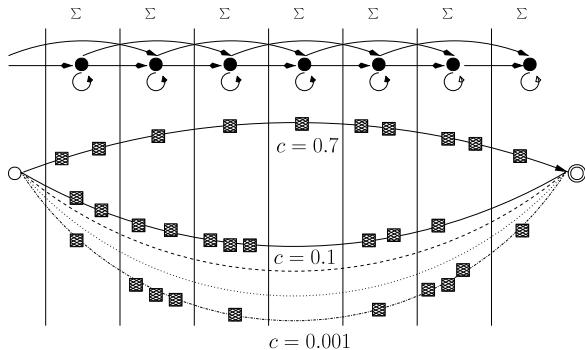
Out-of-vocabulary (OOV) words are also meant to be harmful for adaptation [62], but even when a word is wrong, the pronunciation or most of the pronunciation can still be correct, suggesting that a state-based and confidence-based adaptation should be favored in such cases.

Word Confidences As we are dealing with isolated word recognition on the *IFN/ENIT-database*, the sentence and word confidences are identical. The segments to be used in the second-pass system are first thresholded on a *word level* by their word confidences: only complete word *segments* aligned with a high confidence by the first-pass system are used for model adaptation using discriminative training.

State Confidences Instead of rejecting an entire sentence or word, the system can use state confidence scores to select state-dependent data (see Sect. 9.3.3). State confidence scores are obtained from computing arc posteriors from the lattice output of the decoder. The arc posterior is the fraction of the probability mass of the paths that contain the arc from the mass that is represented by all paths in the lattice. The posterior probabilities can be computed efficiently using the forward-backward algorithm as, for example, described in [37]. Then, the word frames to be used in the second-pass system are first thresholded on a *state level* by their state confidences: only word *frames* aligned with a high confidence by the first-pass system are used for model adaptation using discriminative M-MMI-conf/M-MPE-conf training (see Sect. 9.3.3).

An example for a word graph and the corresponding 1-best state alignment is given in Fig. 9.5: during the decoding, the ten feature frames (the squares) can be

Fig. 9.5 Example for a word graph and the corresponding 1-best state alignment: word confidence of the 1-best alignment is $c = 0.7$. The corresponding state confidences are calculated by accumulating state-wise over all other word alignments



aligned to different words (long arcs) and their states. In this example, the word confidence of the 1-best alignment is $c = 0.7$ (upper arc). The corresponding state confidences are calculated by accumulating state-wise over all competing word alignments (lower arcs); i.e. the state confidence of the 1-best alignment's fourth state would stay 0.7 as this state is skipped in all other competing alignments, and all other state confidences would sum up to 1.0.

Writer Adaptation

The decoding in the second pass can be carried out using CMLLR transformed features. The segments to be recognized are first clustered using a generalized likelihood ratio clustering with a Bayesian information criterion (BIC)-based stopping condition [8]. The segment clusters act as writer labels required by the unsupervised adaptation techniques. The CMLLR matrices are calculated in pass two for every estimated writer cluster and are used for a writer-dependent recognition system, which uses the models from the writer adaptive training of Sect. 9.3.4.

9.4 RWTH OCR Software Framework for Large Vocabulary OCR

The RWTH ASR software framework⁵ is based on the RWTH Aachen University Open Source Speech Recognition System [69], abbreviated as *RWTH ASR*. RWTH ASR has been designed for the special requirements of research applications. On the one hand it should be very flexible, to allow for rapid integration of new methods, and on the other hand it has to be efficient, so that new methods can be studied on real-life tasks in reasonable time and so that system tuning is feasible. The flexibility is achieved by a modular design, where most components are decoupled from each

⁵<http://www.hltpr.rwth-aachen.de/rwth-ocr/>

other and can be replaced at runtime. The application programming interface (API) is subdivided into several modules and allows for an integration of (high and low level) methods in external applications.

The applicability of the toolkit to real-life speech recognition tasks has been proven by building several large vocabulary systems in recent international research projects, for example TC-STAR [40] (European English and Spanish), GALE [63, 68] (Arabic and Chinese), and Quaero [55] (English, French, German, and Spanish).

A good example of the flexibility of the toolkit is the expeditious development of systems for continuous sign language recognition using video input [12] and for handwriting recognition [14, 15]. Only the feature extraction had to be replaced to adapt the system to these tasks. In the following sections, we will focus on the parts of the framework which are relevant for OCR.

An important aspect for developing a system for a large vocabulary task is the support for grid computing. Nearly all processing steps for training and decoding can be distributed in a cluster computer environment. The parallelization scales very well, because we divide the computations on the segment level, which requires synchronization only at the end of the computation.

The toolkit is published under an open source license, called “RWTH ASR License” and is publicly available.⁶ This RWTH ASR License grants free usage including redistribution and modification for non-commercial use.

9.4.1 Feature Extraction

The feature extraction is implemented in a generic framework for data processing, called *Flow*. The data flow is modeled by links connecting several nodes to a network. Each node performs some type of data manipulation including loading, storing, and caching of data.

The networks are created at runtime based on a network definition in XML documents, which makes it possible to implement or modify data processing tasks without modifying and recompiling the software. The individual nodes can be either instances of a C++ class or a subnetwork of other nodes.

By using cache nodes, data types sent through the network can be written to disk at any point in the network. The stored data can be read afterwards without repeating the computations of the nodes before the cache node.

Flow networks are used to compute feature vectors as well as to generate and process data alignments, i.e. mappings from feature vectors to HMM states. Using the caching nodes, features and alignments can be reused in processing steps requiring multiple iterations.

⁶<http://www.hltpr.rwth-aachen.de/rwth-asr/>

9.4.2 Visual Modeling

A word is modeled by a sequence of glyph models. The writing variant model gives for each word in the vocabulary a list of glyph model sequences together with a probability of the variant's occurrence. The toolkit supports context-dependent modeling of subunits (glyphs for OCR, phones for ASR) using decision trees for HMM state model tying. However, context-dependent modeling has not been used so far for our OCR systems.

The toolkit supports strict left-to-right HMM topologies, each representing a (potentially context-dependent) sub-word unit. All HMMs consist of the same number of states, except for a dedicated white-space (or silence) model. The transition model implements loop, forward, and skip transitions with globally shared transition probabilities.

The emission probability of an HMM state is represented by a Gaussian mixture model (GMM). By default, globally pooled variances are used. However, several other tying schemes, including density-specific diagonal covariance matrices, are supported.

For the unsupervised refinement or re-estimation of model parameters the toolkit supports the generation and processing of confidence-weighted state alignments. Confidence thresholding on the state level is supported for unsupervised training as well as for unsupervised adaptation methods. The toolkit supports different types of state confidence scores; most are described in [23]. The emission model can be re-estimated based on the automatically annotated observations and their assigned confidence weights, as presented in [14, 24].

9.4.3 Model Adaptation

The software framework supports maximum likelihood linear regression (MLLR) and feature space MLLR (fMLLR) (also known as constrained MLLR, CMLLR) for writer adaptive modeling.

The fMLLR consists of normalizing the feature vectors by the use of a maximum likelihood estimated affine transform, as described in [22]. As an extension, the estimation of dimension reducing affine transforms, as described in [41], is supported. fMLLR is implemented in the feature extraction front-end, allowing for use in both recognition and in training, thus supporting writer adaptive training [15].

For MLLR [38] affine transforms are applied to the means of the visual model. A regression class tree approach [39] is used to adjust the number of regression classes to the amount of adaptation data available. As a variation, it is possible to do adaptation using only the offset part (and not the matrix part) of the affine transform.

The adaptation methods can be utilized both for unsupervised and supervised adaptation. The transformation estimation can make use of weighted observations allowing for confidence-based unsupervised adaptation.

9.4.4 Language Modeling

The toolkit does not include tools for the estimation of language models. However, the decoder supports n-gram language models in the ARPA format, produced e.g. by the SRI Language Modeling Toolkit [75]. The order of the language model is not limited by the decoder. Class language models, defined on word classes instead of words, are supported as well. Alternatively, a weighted finite state automaton representing a (weighted) grammar can be used.

9.4.5 Decoder

The decoder included in our toolkit is based on a word conditioned tree search [51]. Word conditioned tree search is a one-pass dynamic programming algorithm which uses a pre-compiled lexical prefix tree as a representation of the writing variants dictionary. When using a tree lexicon, the word identity is not known until a leaf node is reached. Therefore, the language model (LM) probability can only be applied at the word end, although an early incorporation of the LM can be achieved using LM look-ahead. To make the application of the dynamic programming principle possible, the search space has to be structured by introducing separate copies of the lexical tree for each preceding word sequence. The length of this word sequence depends on the order of the language model used; e.g. for a bigram language model only the direct predecessor word is required.

The search space would be too large to be constructed as a whole, so instead only the active portions are constructed dynamically in combination with a beam search. The beam search strategy retains for every time step only the most promising hypotheses. Hypotheses with a too low score compared to the best state hypothesis are eliminated by *state pruning*. The beam width, i.e. the number of surviving hypotheses, is defined by a threshold. *Language model pruning* is applied to the word start hypotheses after applying the language model, which limits the number of active tree copies. In addition, histogram pruning restricts the absolute number of active hypotheses.

The state pruning can be refined by incorporating the language model probabilities as early as possible using a language model look-ahead [57]. The anticipated language model probability for a certain state in the tree is approximated by the best word end reachable. This probability is incorporated in the pruning process by combining it with the probability of the state hypothesis.

The decoder can also generate a word graph (also called a lattice), which is a compact representation of the set of alternative word sequences with corresponding word boundaries [58]. This word graph can be used in later processing steps. Our system produces word graphs as finite state automata with attached word boundaries or alternatively in the HTK standard lattice format.

The computation of emission probabilities can be optionally accelerated by the use of SIMD instructions provided by modern processors [36]. The feature vectors

Table 9.1 Corpus statistics for the *IFN/ENIT-database Arabic handwriting subcorpora*

Subsets	#Observations [k]			
	Writers	Words	Characters	Frames
a	0.1	6.5	85.2	452
b	0.1	6.7	89.9	459
c	0.1	6.5	88.6	452
d	0.1	6.7	88.4	451
e	0.5	6.0	78.1	404
f	n.a.	8.6	64.7	n.a.
s	n.a.	1.5	11.9	n.a.

as well as the means of the Gaussian mixture models are then transformed to integers using a scalar quantization. The following computations on these quantized vectors are performed using MMX or SSE2 instructions.

9.4.6 Documentation

The documentation is divided into two parts: usage documentation and source code documentation. While the source code documentation is helpful for extending the software, the usage documentation is more comprehensive and more relevant for the normal user.

The usage documentation is organized in a wiki and covers all steps of the model training, multi-pass recognition, and the common concepts of the software and the used file formats. Emerging questions can be asked in a support forum.

9.5 Datasets

In the following we describe the corpora we used for closed-vocabulary isolated handwritten word recognition, and our novel Arabic newspaper corpus for open-vocabulary machine-printed text recognition tasks.

9.5.1 IFN/ENIT-database Arabic Handwriting Database

The *IFN/ENIT-database* is divided into four training subsets with an additional fold for testing [48]. The current database version (v2.0p1e) contains a total of 32492 Arabic words handwritten by about 1000 writers, and has a vocabulary size of 937 Tunisian town names. Here, we follow the same evaluation protocol as for

the ICDAR 2005, 2007, 2009, and ICFHR 2010 competitions [46, 47]. The corpus statistics for the different subsets can be found in Table 9.1.

It should be noted that all experiments with this database in the following sections were done without any pruning, and thus the improvement of the system accuracy is due to the proposed refinement methods only.

9.5.2 The RWTH Arabic Machine-Print Newspaper Corpus

In 1995, the DARPA Arabic machine-print (DAMP) corpus was collected by SAIC[9, 49]. It consists of 345 images from newspapers, books, magazines, etc., but is not publicly available.

The synthetic APTI database [74] for Arabic machine-printed documents offers many synthetically rendered fonts but seems unsuitable for large vocabulary and domain-specific OCR tasks.

In [1] a Multi-Modal Arabic Corpus (MMAC)⁷ containing a list of six million Arabic words is presented, which may be used as a lexical lookup table to check the existence of a given word. However, no large amounts of image segments with corresponding ground-truth annotations to be used in OCR experiments are currently provided. Recently, the PATDB [2] has been presented, which will be interesting for future work, but which is not yet available.

The objective of the MADCAT[56] project is to produce a robust, highly accurate transcription engine that ingests documents of multiple types, especially Arabic scripts, and produces English transcriptions of their content. Some parts of the Arabic handwriting data, which was created by the Linguistic Data Consortium (LDC) and used in previous MADCAT evaluations [49], has been recently used for the OpenHaRT 2010 [52] competition. However, no machine-printed documents have been provided so far.

Therefore, we started in 2010 with the generation of the large vocabulary RWTH Arabic Machine-Print Newspaper (RAMP-N) corpus⁸ suitable for OCR research, by collecting more than 85k PDF pages of newspaper articles from the following websites:

- <http://www.addustour.com> (Lebanon)
- <http://www.albayrakonline.com> (Jordan)

In our current collection (see Table 9.2), the newspaper data in the training corpus ranges from April to May 2010 and the development corpus from May 2010, and the evaluation corpora were collected in September 2010.

We automatically generate ground-truth annotations with the freely available PDFlib Text Extraction Toolkit (TET),⁹ which reliably extracts Unicode text, im-

⁷<http://www.ashrafraouf.com/mmac>

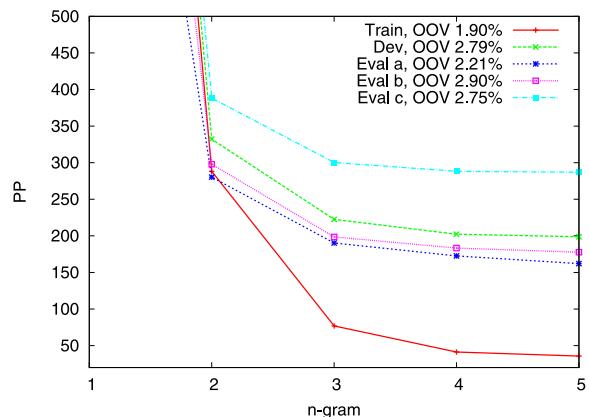
⁸<http://www.hltpr.rwth-aachen.de/~dreuw/arabic.php>

⁹<http://www.pdflib.com/products/tet/>

Table 9.2 RAMP-N corpora statistics

	Train	Dev	Eval a	Eval b	Eval c	LM Training
Running words	1,483,136	7,775	20,042	17,255	15,290	228,492,763
Running characters	5,970,997	30,884	72,358	64,293	62,065	989,494,230
Text lines	222,421	1,155	3,480	2,439	2,224	22,910,187
Pages	409	2	5	4	4	85,316
Fonts	20	5	12	7	6	–
OOV Rate	1.90 %	2.79%	2.21%	2.90%	2.75%	–

Fig. 9.6 Perplexities (PP) for different n-gram contexts using modified Kneser–Ney smoothing and a vocabulary size of 106k words



ages, and metadata from PDF documents. Additionally, detailed glyph and font information as well as the position on the page can be extracted.

In addition to the 28 Arabic base forms, and after filtering out texts with Latin glyphs, the Arabic texts in our current collection include 33 ligatures, 10 Arabic-Indian digits, and 24 punctuation marks. They are modeled by 95 position-independent or by 197 position-dependent glyph HMMs [13, 67]. The position-dependent glyph transcriptions have been created by a rule-based approach based on the six Arabic characters, which have only an isolated or final form [42].

Text Corpora About 228M running words have been collected for domain-specific language model (LM) estimation. As vocabulary we currently use the 106k most frequent words of the 228M LM data corpus, resulting in about 126k writing variants due to ligatures, an average out-of-vocabulary (OOV) rate of 2.5 % (see Table 9.2), and a 0 % out-of-glyph rate. None of the segments in the development or evaluation corpora belong to the LM training data. The resulting perplexities, which are relatively high due to the rich morphology in Arabic, for different n-gram language models using modified Kneser–Ney smoothing are presented in Fig. 9.6.

9.6 Experimental Results

The proposed approach is applied to isolated Arabic handwritten and continuous Arabic machine-printed texts. The experiments for isolated word recognition are conducted on the *IFN/ENIT-database* [61] using a closed lexicon; experiments for continuous line recognition are done on the novel large vocabulary RAMP-N corpus.

9.6.1 First-Pass Decoding

In this section we compare our ML trained baseline systems (see Sect. 9.3.2 for visual model details) to our discriminatively trained systems using the MMI and MPE criteria and their margin-based extensions.

Each of the 120 glyph models in our Arabic handwriting recognition base system is modeled by a three-state left-to-right HMM with three separate GMMs. The position-dependent glyph model of our ML trained baseline system includes 361 mixtures with 36k Gaussian densities with globally pooled diagonal variances.

The discriminative training is initialized with the respective ML trained baseline model and iteratively optimized using the Rprop algorithm (see Sect. 9.3.3). For isolated Arabic word recognition on the *IFN/ENIT-database*, we compare our ML trained baseline system with MMI/M-MMI criteria only.

Discriminative GHMMs

In general, the number of Rprop iterations and the choice of the regularization constant C have to be chosen carefully (see optimization in Sect. 9.3.3), and were empirically optimized in informal experiments to 30 Rprop iterations and $C = 1.0$ (see detailed Rprop iteration analysis and convergence without over-training in Fig. 9.8).

The results in Table 9.3 show that the discriminatively trained models clearly outperform the ML trained baseline models, especially the models trained with the additional margin term. The strong decrease in word error rate (WER) for experiment setup *abd-c* might be due to the training data being separable for the given configurations, whereas the strong improvement for experiment *abcde-e* was expected because of the test set *e* being part of the training data.

In the following experiments, we additionally use glyph-dependent lengths (GDLs) as described in [13, 15], resulting in an ML trained baseline model with 216 glyph models, 646 mixtures, and up to 55k densities (see Sect. 9.3.2). The necessity of this glyph-dependent model length estimation is exemplified by visualizing the state alignment in Fig. 9.7. Different background colors are used for the respective HMM states.

By estimating glyph-dependent model lengths, the overall mean of glyph length changed from 7.89 px (i.e. 2.66 px/state) to 6.18 px (i.e. 2.06 px/state) when downscaling the images to 16 px height while keeping their aspect ratio. Thus every state

Table 9.3 Comparison of ML trained baseline systems and discriminatively trained systems using MMI and M-MMI criteria after 30 Rprop iterations on the *IFN/ENIT-database*

Train	Test	WER [%]		
		ML	MMI	M-MMI
abc	d	10.88	10.59	8.94
abd	c	11.50	10.58	2.66
acd	b	10.97	10.43	8.64
bcd	a	12.19	11.41	9.59
abcd	e	21.86	21.00	19.51
abcde	e	11.14	2.32	2.95

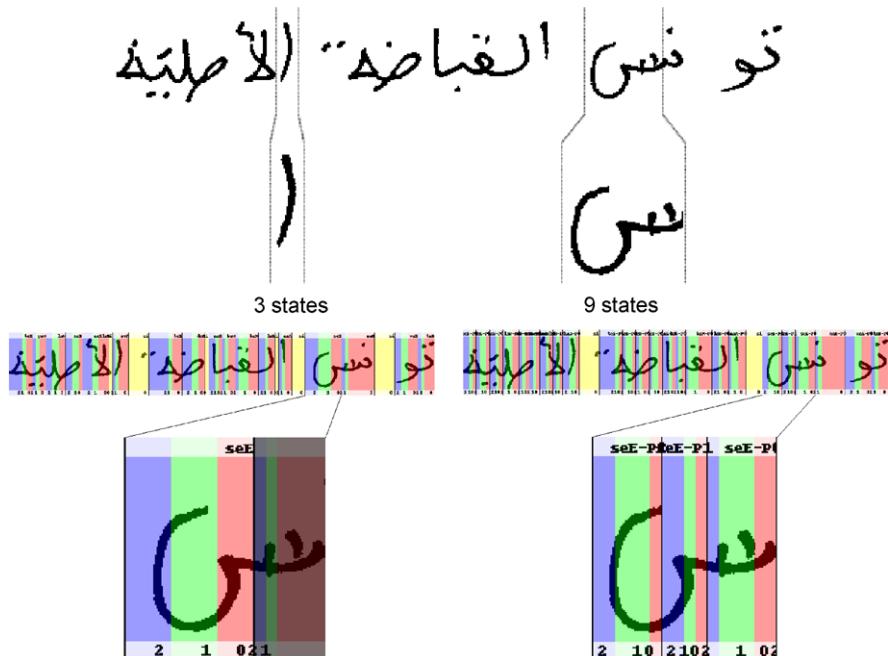


Fig. 9.7 *Top:* More complex characters should be represented by more states. *Bottom:* Using GDL glyph models, frames previously aligned to a wrong neighboring glyph model (*left, black shaded*) are aligned to the correct glyph model (*right*)

of a GDL glyph model has to cover less pixels due to the relative reduction of approximately 20 % pixels.

In Fig. 9.8 detailed WER and character error rate (CER) plots over M-MMI training iterations are shown. It can be observed that both WER and CER are smoothly and almost continuously decreasing with every Rprop iteration, and that about 30 Rprop iterations are optimal for the considered datasets.

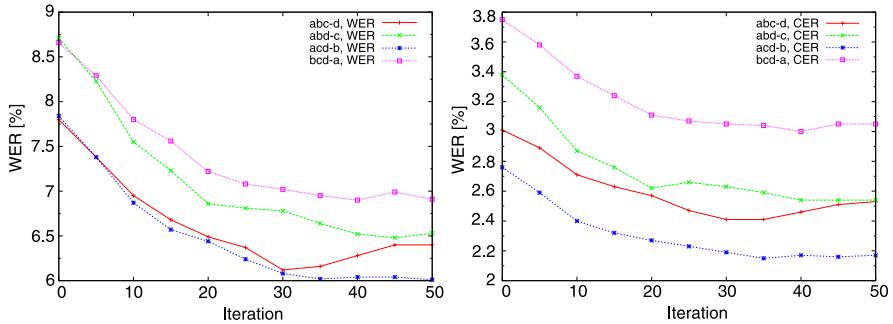


Fig. 9.8 Decreasing word error rates (WER) and character error rates (CER) are shown for all different training subsets of the *IFN/ENIT-database* over M-MMI Rprop iterations (baseline with GDL estimation)

Table 9.4 Results for margin-based M-MMI criterion after 30 Rprop iterations on the *IFN/ENIT-database* using GDLs

Train	Test	WER [%]			
		ML	GDL	+MMI	+M-MMI
abc	d	10.88	7.83	7.4	6.12
abd	c	11.50	8.83	8.2	6.78
acd	b	10.97	7.81	7.6	6.08
bcd	a	12.19	8.70	8.4	7.02
abcd	e	21.86	16.82	16.4	15.35

The final results for discriminative GHMM training with additional GDL estimation are presented in Table 9.4.

Hybrid MLP/HMM vs. Tandem MLP-GHMM

Due to a position and GDL modeling of the 28 base Arabic characters [13], we finally model the Arabic words in the *IFN/ENIT-database* by 216 different glyph models (i.e., 215 glyphs and one white-space model). The system described in [14] (see also M-MMI column in Table 9.7) is used to generate an initial alignment of the features to the 216 labels. Our discriminative GHMM baseline system (see Table 9.5) uses 3 mixtures per glyph label, resulting in up to 646 mixtures with 55k densities. The MLP networks have been trained on raw pixel column features from the sets *a*, *b*, and *c* only.

Raw MLP Features The hierarchical system uses at the first level no windowing of the input features, a single hidden layer with 2000 nodes, and 216 output nodes, which are reduced by a log-PCA transformation to 32 components. The second network concatenates these features in addition to the raw features, and uses a window

Table 9.5 System comparison: MLP-GHMM performs best; both GHMM and MLP-GHMM systems are M-MMI trained

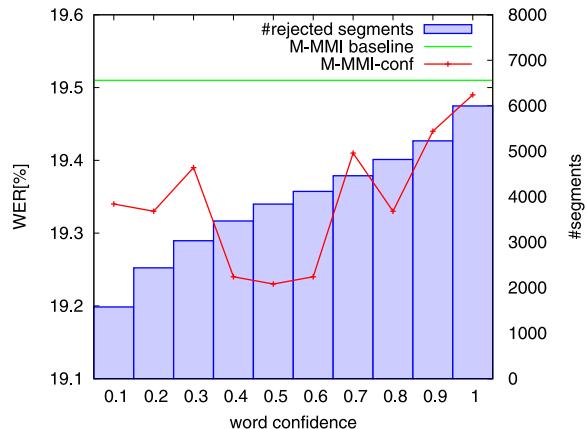
Train	Test	GHMM		MLP/HMM		MLP-GHMM	
		WER [%]	CER [%]	WER [%]	CER [%]	WER [%]	CER [%]
abc	d	6.12	2.41	4.54	1.70	3.47	1.50
abd	c	6.78	2.63	2.64	0.93	1.38	0.75
acd	b	6.08	2.19	2.70	0.87	2.52	0.98
bcd	a	7.02	3.05	3.11	1.32	2.60	1.09
abcd	e	15.35	6.14	11.57	4.54	7.26	3.03

size of 9 consecutive features. The 576-dimensional features (i.e. $32 \times 2 \times 9$ features) are forwarded to a single hidden layer with 3000 nodes, and reduced by a log-PCA transformation to 32 components.

TRAP-DCT MLP Features The system uses a TRAP-DCT [31] pre-processing of the raw pixel input features. The TRAP-DCT pre-processing for sliding window image patches can be interpreted as a modular block-based DCT of the patches at image row level. The hierarchical system uses at the first level a spatio-temporal TRAP-DCT window to augment the 32-dimensional raw pixel input feature vectors to a 256-dimensional vector. Again, the first level hierarchical network uses a single hidden layer with 1500 nodes, and 216 output nodes, which are reduced by a log-LDA transformation to 96 components. The second network concatenates these features in addition to the raw features, and uses a window size of 5 consecutive log-LDA network features, and a window size of 9 consecutive raw input features to account for different spatio-temporal information. The 768-dimensional features (i.e. $96 \times 5 + 32 \times 9$ features) are forwarded to a single hidden layer with 3000 nodes, and finally reduced by a log-LDA transformation to 36 components.

We empirically optimized raw, TRAP-DCT, and feature combinations on the different *IFN/ENIT-database* training subsets, which showed no significant difference. The TRAP-DCT log-posterior features are used in Table 9.5 for the hybrid MLP/HMM approach, which turned out to perform slightly better than the raw features in these informal experiments. Furthermore, we observed that a discriminative MLP-GHMM system is about 25 % relative better than a generatively trained one, especially in combination with the concatenated RAW+TRAP-DCT features. The comparison in Table 9.5 shows a significant advantage of the retrained MLP-GHMM system over the hybrid MLP/HMM and the GHMM baseline. The achieved 7.26 % WER on evaluation set e is about 50 % relatively better than the M-MMI trained baseline system, and to the best of the authors knowledge, outperforms all error rates reported in the literature.

Fig. 9.9 Results for word confidence-based M-MMI-conf training on the evaluation setup *abcd-e* of the *IFN/ENIT-database* using different confidence thresholds and their corresponding number of rejected segments (baseline without GDL estimation)



9.6.2 Second-Pass Decoding and Unsupervised Model Adaptation

In this section we evaluate our discriminative training for unsupervised model or writer adaptation during a second-pass decoding step.

Confidence-Based Discriminative GHMMs

In a first experiment we used the complete first-pass output of the M-MMI system for an unsupervised model adaptation. The results in Table 9.6 show that the M-MMI based unsupervised adaptation without confidences cannot improve the system accuracy. With every Rprop iteration, the system is even more biased by the relatively large amount of wrong transcriptions in the adaptation corpus.

The discriminative M-MMI-conf training is initialized with the respective M-MMI trained model and iteratively optimized using the Rprop algorithm (see Sect. 9.3.3). Using the word confidences for the M-MMI-conf based model adaptation of our first-pass alignment to reject complete word segments (i.e. feature sequences X_1^T) from the unsupervised adaptation corpus, the results in Table 9.6 show a slight improvement only in comparison to the M-MMI trained system. Figure 9.9 shows the resulting WER for different confidence threshold values and the corresponding number of rejected segments. For a confidence threshold of $c = 0.5$, more than 60 % of the 6033 segments of set *e* are rejected from the unsupervised adaptation corpus, resulting in a relatively small amount of adaptation data.

Using the state confidences for the M-MMI-conf-based model adaptation of our first-pass alignment to decrease the contribution of single frames (i.e. features x_t) during the iterative M-MMI-conf optimization process (see optimization in Sect. 9.3.3), the number of features for model adaptation is reduced by approximately 5 % for a confidence threshold of $c_{\text{threshold}} = 0.5$: 375 446 frames of 396 416 frames extracted from the 6033 test segments are considered during the optimization, and only 20 970 frames are rejected based on state-confidence thresholding (see also Fig. 9.5). Note also that the CER is decreased to 6.49 %.

Table 9.6 Results for M-MMI-conf model adaptation on the evaluation setup *abcd-e* of the *IFN/ENIT-database* after 30 Rprop iterations (baseline without GDL estimation)

Training/Adaptation	WER [%]	CER [%]
ML	21.86	8.11
M-MMI	19.51	7.00
+ unsupervised adaptation	20.11	7.34
+ supervised adaptation	2.06	0.77
M-MMI-conf (word confidences)	19.23	7.02
M-MMI-conf (state confidences)	17.75	6.49

Table 9.7 Results for confidence-based M-MMI-conf model adaptation after 15 Rprop iterations on the *IFN/ENIT-database* using GDL, and margin-based M-MMI criterion after 30 Rprop iterations

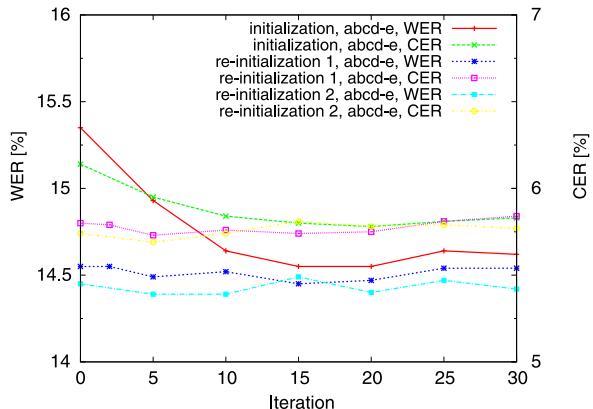
Train	Test	WER [%]					
		1st pass				2nd pass	
		ML	GDL	+MMI	+M-MMI	M-MMI-conf	
abc	d	10.88	7.83	7.4	6.12	5.95	
abd	c	11.50	8.83	8.2	6.78	6.38	
acd	b	10.97	7.81	7.6	6.08	5.84	
bcd	a	12.19	8.70	8.4	7.02	6.79	
abcd	e	21.86	16.82	16.4	15.35	14.55	

Interestingly, the supervised adaptation on test set *e*, where only the correct transcriptions of set *e* are used for an adaptation of the model trained using set *abcd*, can again decrease the WER of the system down to 2.06 %, which is even better than an M-MMI optimization on the full training set *abcde* (see Table 9.3).

Table 9.7 shows the final results of our Arabic handwriting recognition system with additional GDLs as described in [15]. Again, the WER of the GDL-based system can be decreased by our proposed M-MMI training during both decoding passes down to 14.55 %.

In Fig. 9.10 a combined WER/CER plot over M-MMI-conf training iterations on the evaluation setup *abcd-e* (see initialization plots) is shown. It can be observed that both WER and CER are slightly decreasing with every Rprop iteration, and that between 10 and 15 Rprop iterations are optimal for the considered small amount of unsupervised labeled test datasets. Due to the robustness of the confidence- and margin-based M-MMI-conf criterion against outliers, the proposed unsupervised and text-dependent model adaptation can even be applied in an iterative manner by a reinitialization of the text transcriptions. In Fig. 9.10, we reinitialize two times the model adaptation process after 15 Rprop iterations. The results in Fig. 9.10 show the robustness of our approach, leading to a slightly improved WER of 14.39 %.

Fig. 9.10 Evaluation of iterative M-MMI-conf model adaption on the evaluation setup *abcd-e* of the *IFN/ENIT-database*: text transcriptions are updated in an unsupervised manner after 15 Rprop iterations. The performance remains robust even after several reinitializations



Writer Adaptation

The writer adaptive training (WAT) models (see Sect. 9.3.4) can also be used as a first-pass decoding system. The results in Table 9.8 show that the system performance cannot be improved without any writer clustering and adaptation of the features during the decoding step.

To show the advantage of using CMLLR-based writer adapted features in combination with WAT models, we estimate in a first *supervised* experiment the CMLLR matrices directly from the available *writer labels* of the test subsets. The matrices are calculated for all writers in pass two and are used for a writer-dependent recognition system, which uses the WAT models from Sect. 9.3.4. Note that the decoding itself is still unsupervised!

In the unsupervised adaptation case, the unknown writer labels of the segments to be recognized have to be estimated first using BIC clustering. Again, the CMLLR matrices are calculated in pass two for every estimated cluster label and are used for a writer-dependent recognition system, which uses the WAT models from Sect. 9.3.4.

Table 9.8 shows that the system accuracy could be improved by up to 33 % relative to the supervised-CMLLR adaptation case. In the case of unsupervised writer clustering, the system accuracy is improved by onefold only.

If we look at the cluster histograms in Fig. 9.11, it becomes clear that the unsupervised clustering is not adequate. Each node in our clustering process as described in [8] is modeled as a multivariate Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$, where μ_i can be estimated as the sample mean vector and Σ_i can be estimated as the sample covariance matrix. The estimated parameters are used within the criterion as distance measure, but more sophisticated features than the PCA reduced sliding window features seem necessary for a better clustering, which will be interesting for future work.

As opposed to the supervised estimation of 505 CMLLR transformation matrices for the evaluation setup with training sets *abcd* and test set *e* (see Table 9.1), the unsupervised writer clustering could estimate only two clusters being completely

Fig. 9.11 Histograms for unsupervised clustering over the different test subsets and their resulting unbalanced segment assignments

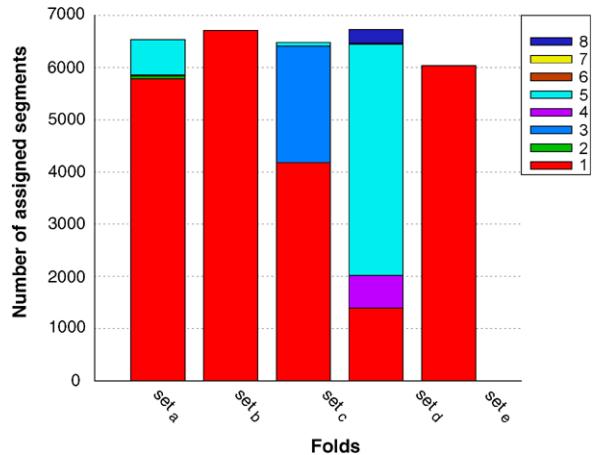


Table 9.8 Comparison of GDL, WAT, and CMLLR based feature adaptation using unsupervised and supervised writer clustering

Train	Test	WER [%]					
		1st pass			2nd pass		
		ML	+GDL	+WAT	WAT+CMLLR	Unsup.	Sup.
abc	d	10.88	7.83	7.54	7.72	5.82	
abd	c	11.50	8.83	9.09	9.05	5.96	
acd	b	10.97	7.81	7.94	7.99	6.04	
bcd	a	12.19	8.70	8.87	8.81	6.49	
abcd	e	21.86	16.82	17.49	17.12	11.22	

unbalanced, which is obviously not enough to represent the different writing styles of 505 writers. Due to the unbalanced clustering and only a small number of clusters, all other cases are similar to the usage of the WAT models only (see Table 9.8).

However, the supervised-CMLLR adaptation results show that a good writer clustering can bring the segments of the same writer together and thus improve the performance of the writer adapted system.

9.6.3 Visual Inspections

The visualizations in Fig. 9.12 show training alignments of Arabic words to their corresponding HMM states. The upper rows show the alignment to the ML trained model, the lower rows to the M-MMI trained models. We use R-G-B background colors for the 0-1-2 HMM states, respectively, from right to left. The position-dependent glyph model names (see Sect. 9.3.2) are written in the upper line, where

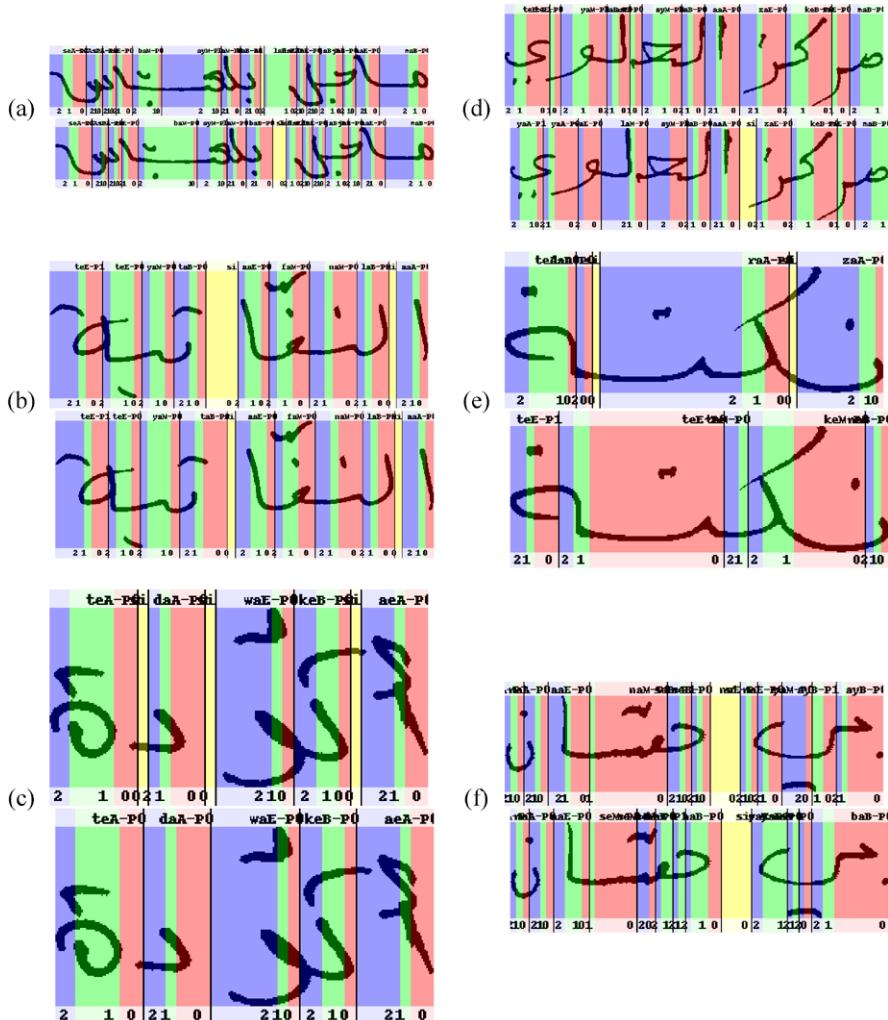


Fig. 9.12 *Left column:* Supervised training alignment comparisons. The *upper* rows show alignments to the maximum-likelihood (ML) trained model, the *lower* rows to the margin-based maximum mutual information (M-MMI) trained models. *Right column:* Unsupervised test alignment comparisons. The *upper* rows show incorrect unsupervised alignments to the ML trained model, the *lower* rows correct unsupervised alignments to the M-MMI trained models

the white-space models are annotated by “si” for “silence”; the state numbers are written in the bottom line. Thus, HMM state loops and state transitions are represented by no-color-changes and color-changes, respectively.

It can be observed in the left column of Fig. 9.12 that especially the white spaces, which can occur between compound words and parts of Arabic words (PAWs) [13], help in discriminating the isolated- (A), beginning- (B), or end-shaped (E) glyphs

of a word with respect to the middle-shaped (M) glyphs, where usually no white spaces occur on the left or right side of the character (cf. [42, 61] for more details about A/B/M/E shaped characters). The frames corresponding to the white-space part of the words are aligned in a more balanced way in Fig. 9.12(a) and 9.12(b) using the M-MMI modeling (lower rows) opposed to ML modeling (upper rows): the proposed M-MMI models learned that white spaces help to discriminate different glyphs. This can even lead to a different writing variant choice without any white-space models [13] (see Fig. 9.12(c)). Note that we cannot know in advance in training if a white space is used or not, and if so, how large it is, as it is not transcribed in the corpora and depends on the writer's handwriting style (e.g. the cursive style used in Fig. 9.12(a)).

In the right column of Fig. 9.12, unsupervised test alignments are compared. The upper rows show incorrectly recognized words by unsupervised alignments to the ML trained model, the lower rows correctly recognized words by unsupervised alignments to the M-MMI trained models. Due to the discriminatively trained glyph models, the alignment in Fig. 9.12(d) to the M-MMI model is clearly improved over the ML model, and the system opts for the correct compound-white-space writing variant [13]. In Fig. 9.12(e), again the alignment is improved by the discriminatively trained white-space and glyph models. Figure 9.12(f) shows a similar alignment to the white-space model, but a clearly improved and correct alignment to the discriminatively trained glyph models.

9.6.4 Comparisons with Other Systems

IFN/ENIT-database Competitions at ICDAR/ICFHR In Table 9.9 we compare our own evaluation results on the ICDAR 2005 [48] setups (without any tuning on test data as explained in Sect. 9.6.2) and ICDAR 2007/2009 and ICFHR 2010 [46, 47] setups. It should be noted that the result for the *abcd-e* condition is the best known error rate in the literature [18].

The ICDAR 2009 test datasets, which are unknown to all participants, were collected for the tests of the ICDAR 2007 competition. The words are from the same lexicon as those of the *IFN/ENIT-database* and written by writers who did not contribute to the datasets before, and are separated into set f and set s. Our results (externally calculated by TU Braunschweig) in Table 9.9 ranked third at the ICDAR 2009 competition and are among the best purely HMM-based systems, as the A2iA and MDLSTM systems are hybrid system combinations or full neural network-based systems, respectively. Also note that our single HMM-based system is better than the independent A2iA systems (cf. [46] for more details). In particular, our proposed M-MMI-conf-based approach for unsupervised model adaptation even generalizes well on the *set s*, which has been collected in the United Arabic Emirates and represents significantly different handwriting styles.

Note the 36 % relative improvement shown in Table 9.9 that we achieved in the recent ICFHR 2010 Arabic handwriting competition [47] with the proposed M-MMI

Table 9.9 Comparison of ICDAR/ICFHR Arabic handwriting recognition competition results on the *IFN/ENIT-database*

Competition	Group	WER [%]			
		abc-d	abcd-e	abcde-f	abcde-s
ICDAR 2005 [48]	UOB	15.00	24.07	—	—
	ARAB-IFN	12.06	25.31	—	—
	ICRA (Microsoft)	11.05	34.26	—	—
ICDAR 2007 [45]	SIEMENS [70]	—	18.11	12.78	26.06
	MIE (DP)	—	—	16.66	31.60
	UOB-ENST (HMM)	—	—	18.07	30.07
ICDAR 2009 [46]	MDLSTM	—	—	6.63	18.94
	A2iA (combined)	—	—	10.58	23.34
	(MLP/HMM)	—	—	14.42	29.56
	(HMM)	—	—	17.79	33.55
	RWTH ASR (this work, M-MMI)	6.12	15.35	14.49	28.67
	RWTH ASR (this work, M-MMI-conf)	5.95	14.55	14.31	27.46
ICFHR 2010 [47]	UPV PRHLT (HMM)	7.50	12.30	7.80	15.38
	RWTH ASR (this work, MLP-GHMM)	3.47	7.26	9.12	18.94
	UPV PRHLT (HMM, w/o vert. norm.)	—	—	12.09	21.55
	CUBS-AMA (HMM)	—	—	19.68	32.10
Other results	BBN [50]	10.51	—	—	—

training framework and an MLP-based feature extraction. Our system ranked second and used again no system combinations. Interesting is the result of the UPV PRHLT group, who significantly improved their relatively simple baseline system due to a vertical centroid normalization of sliding window-based features [47, 60]. Note that our MLP-GHMM does not perform any pre-processing.

9.6.5 Machine-Printed Arabic Text Recognition

In a first set of experiments we optimized the feature extraction parameters and compared position-independent and dependent glyph models, using single-density models only. In both cases we used glyph HMMs with six states with skip transitions and three separate GMMs with a globally pooled covariance matrix. The results for the development set of the RAMP-N database (see Sect. 9.5.2) in Fig. 9.13 show an error rate reduction of about 50 % relative for position-dependent glyph models compared to a position-independent glyph modeling. Note that we empirically optimized the PCA reduction to 30 components, and that the fea-

Fig. 9.13 Comparison of position-independent and position-dependent glyph modeling on the RAMP-N development corpus, using single-density models and PCA reduced appearance-based sliding window features

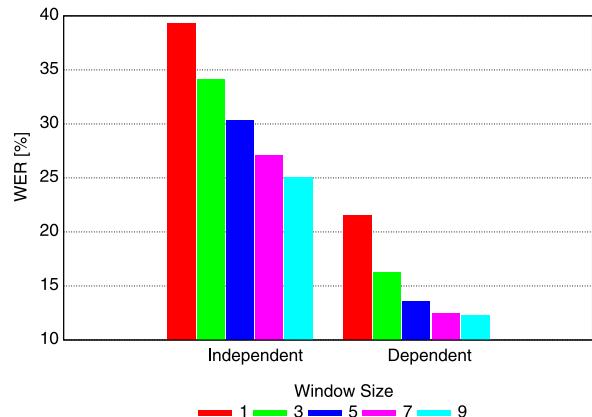


Fig. 9.14 Some examples of various professional newspaper fonts used in the RAMP-N corpora (example images taken from <http://www.layoutltd.com/>)

AXtAlFares	الخط الحسن يزيد الحق وضوها
AXtManalFont	الخط الحسن يزيد الحق وضوها
AXtSHAReQXL	الخط الحسن يزيد الحق وضوها
AXtGIHaneBoldItalic	الخط الحسن يزيد الحق وضوها
AXtKarim	الخط الحسن يزيد الحق وضوها
AXtMarwanBold	الخط الحسن يزيد الحق وضوها
AXtMarwanLight	الخط الحسن يزيد الحق وضوها
AXtSHAReQ	الخط الحسن يزيد الحق وضوها
AXtCalligraph	الخط الحسن يزيد الحق وضوها
AXtHammed	الخط الحسن يزيد الحق وضوها
AXtThuluthMuBassat	الخط الحسن يزيد الحق وضوها

ture extraction parameters are similar to those used in handwritten text recognition.

Some examples of the professional ArabicXT fonts¹⁰ occurring in the RAMP-N corpus, which are widely used by newspapers, magazines, or book publishers, are shown in Fig. 9.14.

Experiments with Gaussian mixture models (GMMs) instead of single densities (see Fig. 9.15) improve the WER/CER as expected, as they implicitly model the up to 20 different font appearances in the corpora. Note that glyph-dependent length (GDL) models as e.g. successfully used for handwriting in [13, 14, 60] (also see Fig. 9.7) lead only to small improvements so far for machine-printed text recognition.

The results in Table 9.10 show detailed results for each font appearing in the RAMP-N subset Eval a. High WER but low CER are due to OOV words, which are

¹⁰<http://www.layoutltd.com/arabicxt.php>

Fig. 9.15 Results for position-dependent GMMs on the RAMP-N subset Eval a

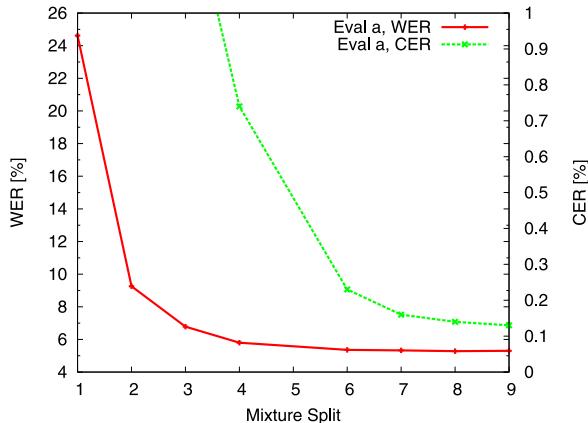


Table 9.10 Font-wise results for ML trained GMMs on the RAMP-N subset Eval a

Font	Lines	Errors	Words	OOV	WER [%]	Errors	Glyphs	CER [%]
AXtAlFares	2	10	2	2	500.00	0	19	0.00
AXtCalligraph	1	0	8	0	0.00	0	21	0.00
AXtGIHaneBoldItalic	15	19	129	4	14.73	12	591	2.03
AXtHammed	3	0	5	0	0.00	0	31	0.00
AXtKaram	9	2	83	0	2.41	4	300	1.33
AXtManal	1	0	2	0	0.00	0	4	0.00
AXtManalBlack	5	5	27	1	18.52	11	112	9.82
AXtMarwanBold	109	46	385	18	11.95	13	2002	0.65
AXtMarwanLight	3261	828	18963	405	4.37	79	83091	0.10
AXtShareQ	5	10	64	0	15.62	7	299	2.34
AXtShareQXL	68	35	371	13	9.43	10	1973	0.51
AXtThuluthMubassat	1	0	3	0	0.00	0	13	0.00
Total (Eval a)	3480	955	20042	443	4.76	136	88456	0.15

often recognized as a sequence of PAWs instead of a single word, resulting in one substitution and many insertion errors, but zero edits at the character level. Simply replacing those word sequences between white-space blocks can further reduce the WER. Interesting subjects for future work will therefore remain larger lexica or character and PAW language models to further reduce the effect of OOVs. Due to unbalanced font frequencies a re-rendering of the training data in other fonts might further reduce the error rates in future works.

The results in Table 9.11 show the difference between rendered and scanned results, where we additionally compared supervised layout and unsupervised layout

Table 9.11 Results for ML trained GMMs using rendered and scanned data of the RAMP-N subset Eval a

Layout Analysis	Rendered		Scanned	
	WER	CER	WER	CER
Supervised	4.76	0.15	5.79	0.64
OCRopus	—	—	17.62	3.79

Fig. 9.16 Results for M-MPE training on RAMP-N corpus Eval a

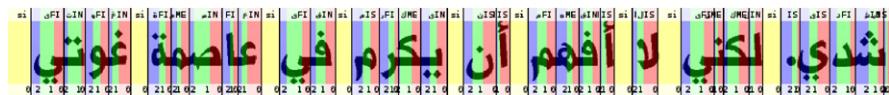
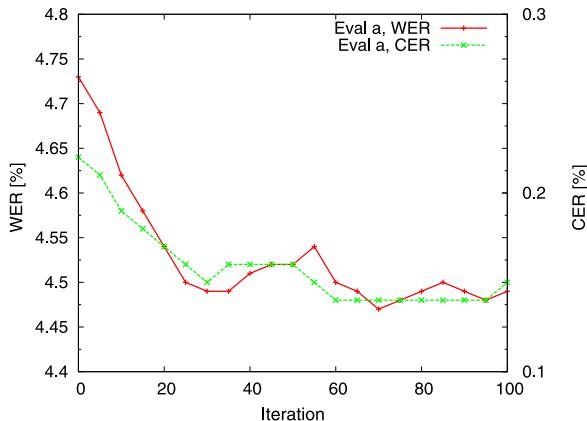


Fig. 9.17 Example of an unsupervised alignment on RAMP-N corpus Eval a

analysis using OCRopus.¹¹ The scans were generated by printing and scanning the PDFs in their original size, i.e. DIN-A2 at 600 dpi. It can be seen that the main performance decrease is due to OCRopus’s layout analysis problems and not the scan quality.

As it is often observed that discriminative GHMM training performs better with fewer Gaussian mixture densities, we use a split-6 ML trained model to initialize our M-MPE training (cf. Λ_0 in Sect. 9.3.3). The results in Fig. 9.16 show again a significant reduction in terms of WER and CER. Note that BBN’s Glyph HMM system PLATO [49] reported similar relative improvements for position-dependent glyph models and discriminative MMI/MPE training.

In Fig. 9.17 an unsupervised alignment example is shown for a line segment of RAMP-N subset Eval a, which seems suitable for post-processing steps such as syntax highlighting or reCAPTCHA-like [77] processes. We used an ML trained GHMM model resulting in zero word/character errors.

¹¹<http://code.google.com/p/ocropus/>

9.7 Conclusions

We presented our hidden Markov model (HMM)-based RWTH ASR system which represents a unique framework for large vocabulary optical character recognition (OCR). The advantages of confidence- and margin-based discriminative training using an MMI/MPE training criterion for model adaptation with an HMM-based multi-pass decoding system were shown for Arabic handwriting on the *IFN/ENIT-database* corpus (isolated word recognition), and preliminary results were shown for Arabic machine-printed text on the RAMP-N corpus (open-vocabulary, continuous line recognition). More details are presented in [11].

We discussed an approach on how to modify existing training criteria for handwriting recognition like, for example, MMI and MPE, to include a margin term. The modified training criterion M-MMI was shown to be closely related to existing large margin classifiers (e.g. SVMs) with the respective loss function. This approach allows for the direct evaluation of the utility of the margin term for handwriting recognition. As expected, the benefit from the additional margin term clearly depends on the training conditions. The proposed discriminative training approach could outperform the ML trained systems on all tasks.

The impact of different writing styles was dealt with by using a novel confidence-based discriminative training for model adaptation, where the use of state confidences during the iterative optimization process based on the modified M-MMI-conf criterion could decrease the word error rate on the *IFN/ENIT-database* by relatively 33 % in comparison to an ML trained system.

Interesting topics for further research remain the hybrid HMM/ANN approaches [20, 25], combining the advantages of large and nonlinear context modeling via neural networks while profiting from the Markovian sequence modeling. This is also supported by the 36 % relative improvement we could achieve in the ICFHR 2010 Arabic handwriting competition [47] by using the proposed discriminative GHMM framework but with an MLP-based feature extraction.

We proposed an approach to automatically generate large corpora for machine-printed text recognition. The preliminary results on the novel RAMP-N database showed that our framework is able to recognize Arabic handwritten *and* machine-printed texts. Future work will focus on using more visual training data, larger lexica, higher order n-gram language models, and character- or PAW-based language models like those successfully used in [49].

Acknowledgements We would like to thank Christian Plahl, Stefan Hahn, Simon Wiesler, Patrick Doetsch, Robert Pyttel, Stephan Jonas, Jens Forster, Christian Gollan, and Thomas Deselaers for their support.

This work was partly realized as part of the Google Research Award “Robust Recognition of Machine Printed Text” and as part of the Quaero Programme, funded by OSEO, the French State agency for innovation.

References

1. AbdelRaouf, A., Higgins, C., Pridmore, T., Khalil, M.: Building a multi-modal Arabic corpus (MMAC). *Int. J. Doc. Anal. Recognit.* **13**, 285–302 (2010). doi:[10.1007/s10032-010-0128-2](https://doi.org/10.1007/s10032-010-0128-2)
2. Al-Hashim, A.G., Mahmoud, S.A.: Printed Arabic text database (PATDB) for research and benchmarking. In: Proceedings of the 9th WSEAS International Conference on Applications of Computer Engineering, ACE'10, pp. 62–68, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point (2010)
3. Anastasakos, T., Balakrishnan, S.V.: The use of confidence measures in unsupervised adaptation of speech recognizers. In: International Conference on Spoken Language Processing (ICSLP), Sydney, Australia (1998)
4. Bazzi, I., Schwartz, R., Makhoul, J.: An omnifont open-vocabulary OCR system for English and Arabic. *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(6), 495–504 (1999)
5. Bertolami, R., Bunke, H.: Hidden Markov model-based ensemble methods for offline handwritten text line recognition. *Pattern Recognit.* **41**(11), 3452–3460 (2008)
6. Biem, A.: Minimum classification error training for online handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(7), 1041–1051 (2006)
7. Bourland, H., Morgan, N.: Connectionist speech recognition: A hybrid approach. In: Series in Engineering and Computer Science, vol. 247. Kluwer Academic, Dordrecht (1994)
8. Chen, S.S., Gopalakrishnan, P.S.: Speaker, environment and channel change detection and clustering via the Bayesian information criterion. In: DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne, Virginia, USA, February 1998, pp. 127–132 (1998)
9. Davidson, R., Hopely, R.: Arabic and Persian OCR training and test data sets. In: Symp. on Document Image Understanding Technology, 30 April–2 May, 1997
10. Do, T.-M.-T., Artières, T.: Maximum margin training of Gaussian HMMs for handwriting recognition. In: International Conference on Document Analysis and Recognition (ICDAR), Barcelona, Spain, July 2009, pp. 976–980 (2009)
11. Drew, P.: Probabilistic sequence models for image sequence processing and recognition. Ph.D. thesis, RWTH Aachen University, Aachen, Germany, December 2011
12. Drew, P., Rybach, D., Deselaers, T., Zahedi, M., Ney, H.: Speech recognition techniques for a sign language recognition system. In: Interspeech, Antwerp, Belgium, August 2007, pp. 2513–2516 (2007)
13. Drew, P., Jonas, S., Ney, H.: White-space models for offline Arabic handwriting recognition. In: International Conference on Pattern Recognition (ICPR), Tampa, Florida, USA, December 2008
14. Drew, P., Heigold, G., Ney, H.: Confidence-based discriminative training for model adaptation in offline Arabic handwriting recognition. In: International Conference on Document Analysis and Recognition (ICDAR), Barcelona, Spain, July 2009, pp. 596–600 (2009)
15. Drew, P., Rybach, D., Gollan, C., Ney, H.: Writer adaptive training and writing variant model refinement for offline Arabic handwriting recognition. In: International Conference on Document Analysis and Recognition (ICDAR), Barcelona, Spain, July 2009
16. Drew, P., Heigold, G., Ney, H.: Confidence and margin-based MMI/MPE discriminative training for offline handwriting recognition. *Int. J. Doc. Anal. Recognit.* **14**(3), 273–288 (2011)
17. Drew, P., Doetsch, P., Plahl, C., Ney, H.: Hierarchical hybrid MLP/HMM or rather MLP features for a discriminatively trained Gaussian HMM: A comparison for offline handwriting recognition. In: IEEE International Conference on Image Processing, Brussels, Belgium, September 2011
18. El Abed, H., Märgner, V.: Improvement of Arabic handwriting recognition systems: combination and/or reject? In: Document Recognition and Retrieval XVI. Proc. SPIE, vol. 7247, San Jose, CA, USA, January 2009
19. El Abed, H., Märgner, V.: ICDAR 2009—Arabic handwriting recognition competition. *Int. J. Doc. Anal. Recognit.* **14**(1), 3–13 (2010)

20. Espana-Boquera, S., Castro-Bleda, M., Gorbe-Moya, J., Zamora-Martinez, F.: Improving off-line handwritten text recognition with hybrid HMM/ANN models. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(4), 767–779 (2011)
21. Fink, G.A., Plötz, T.: Unsupervised estimation of writing style models for improved unconstrained off-line handwriting recognition. In: International Workshop on Frontiers in Handwriting Recognition (IWFHR), La Baule, France, October 2006
22. Gales, M.J.F.: Maximum likelihood linear transformations for HMM-based speech recognition. *Comput. Speech Lang.* **12**(2), 75–98 (1998)
23. Gollan, C., Bacchiani, M.: Confidence scores for acoustic model adaptation. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Las Vegas, NV, USA, April 2008, pp. 4289–4292 (2008)
24. Gollan, C., Ney, H.: Towards automatic learning in LVCSR: rapid development of a Persian broadcast transcription system. In: Interspeech, Brisbane, Australia, September 2008, pp. 1441–1444 (2008)
25. Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(5), 855–868 (2009)
26. Heigold, G.: A log-Linear discriminative modeling framework for speech recognition. Ph.D. thesis, RWTH Aachen University, Aachen, Germany, June 2010
27. Heigold, G., Deselaers, T., Schlüter, R., Ney, H.: Modified MMI/MPE: a direct evaluation of the margin in speech recognition. In: International Conference on Machine Learning (ICML), Helsinki, Finland, July 2008, pp. 384–391 (2008)
28. Heigold, G., Schlüter, R., Ney, H.: Modified MPE/MMI in a transducer-based framework. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Taipei, Taiwan, April 2009, pp. 3749–3752 (2009)
29. Heigold, G., Dreuw, P., Hahn, S., Schlüter, R., Ney, H.: Margin-based discriminative training for string recognition. *IEEE J. Sel. Top. Signal Process.* **4**(6), 917–925 (2010)
30. Heigold, G., Wiesler, S., Nussbaum, M., Lehnert, P., Schlüter, R., Ney, H.: Discriminative HMMs, log-linear models, and CRFs: what is the difference? In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Dallas, Texas, USA, March 2010, pp. 5546–5549 (2010)
31. Hermansky, H., Sharma, S.: Traps—classifiers of temporal patterns. In: International Conference on Spoken Language Processing (ICSLP) (1998)
32. Jacobs, C., Simard, P.Y., Viola, P., Rinker, J.: Text recognition of low-resolution document images. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 695–699 (2005)
33. Jebara, T.: Discriminative, generative, and imitative learning. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA (2002)
34. Juan, A., Toselli, A.H., Domnech, J., Gonzlez, J., Salvador, I., Vidal, E., Casacuberta, F.: Integrated handwriting recognition and interpretation via finite-state models. *Int. J. Pattern Recognit. Artif. Intell.* **2004**, 519–539 (2001)
35. Kae, A., Learned-Miller, E.: Learning on the fly: font-free approaches to difficult OCR problems. In: International Conference on Document Analysis and Recognition (ICDAR), Barcelona, Spain, July 2009, pp. 571–575 (2009)
36. Kanthak, S., Schütz, K., Ney, H.: Using SIMD instructions for fast likelihood calculation in LVCSR. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Istanbul, Turkey, June 2000, pp. 1531–1534 (2000)
37. Kemp, T., Schaaf, T.: Estimating confidence using word lattices. In: European Conference on Speech Communication and Technology, Rhodes, Greece (1997)
38. Leggetter, C.J., Woodland, P.C.: Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Comput. Speech Lang.* **9**(2), 171–185 (1995)
39. Leggetter, C.J., Woodland, P.C.: Flexible speaker adaptation using maximum likelihood linear regression. In: ARPA Spoken Language Technology Workshop, Austin, TX, USA, January 1995, pp. 104–109 (1995)

40. Lööf, J., Gollan, C., Hahn, S., Heigold, G., Hoffmeister, B., Plahl, C., Rybach, D., Schlüter, R., Ney, H.: The RWTH 2007 TC-STAR evaluation system for European English and Spanish. In: Interspeech, Antwerp, Belgium, August 2007, pp. 2145–2148 (2007)
41. Lööf, J., Schlüter, R., Ney, H.: Efficient estimation of speaker-specific projecting feature transforms. In: International Conference on Spoken Language Processing (ICSLP), Antwerp, Belgium, August 2007, pp. 1557–1560 (2007)
42. Lorigo, L.M., Govindaraju, V.: Offline Arabic handwriting recognition: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(85), 712–724 (2006)
43. Lu, Y.: Machine printed character segmentation—an overview. *Pattern Recognit.* **28**(1), 67–80 (1995)
44. Lu, Z.A., Bazzi, I., Kornai, A., Makhoul, J., Natarajan, P.S., Schwartz, R.: A robust language-independent OCR system. In: AIPR Workshop: Advances in Computer-Assisted Recognition. Proc. SPIE, vol. 3584, pp. 96–104 (1998)
45. Märgner, V., El Abed, H.: ICDAR 2007 Arabic handwriting recognition competition. In: International Conference on Document Analysis and Recognition (ICDAR), September 2007, vol. 2, pp. 1274–1278 (2007)
46. Märgner, V., El Abed, H.: ICDAR 2009 Arabic handwriting recognition competition. In: International Conference on Document Analysis and Recognition (ICDAR), Barcelona, Spain, July 2009, pp. 1383–1387 (2009)
47. Märgner, V., El Abed, H.: ICFHR 2010 Arabic handwriting recognition competition. In: International Conference on Frontiers in Handwriting Recognition (ICFHR), Kalkota, India, November 2010
48. Märgner, V., Pechwitz, M., Abed, H.E.: ICDAR 2005 Arabic handwriting recognition competition. In: International Conference on Document Analysis and Recognition (ICDAR), Seoul, Korea, August 2005, vol. 1, pp. 70–74 (2005)
49. Natarajan, P.: Portable language-independent adaptive translation from OCR, final report (phase 1). Technical report, BBN Technologies, June 2009
50. Natarajan, P., Saleem, S., Prasad, R., MacRostie, E., Subramanian, K.: Multi-lingual offline handwriting recognition using hidden Markov models: a script-independent approach. In: Arabic and Chinese Handwriting Recognition. LNCS, vol. 4768, pp. 231–250. Springer, Berlin (2008)
51. Ney, H., Ortmanns, S.: Progress in dynamic programming search for LVCSR. *Proc. IEEE* **88**(8), 1224–1240 (2000)
52. NIST 2010 open handwriting recognition and translation evaluation plan, version 2.8, February 2010
53. Nopsuwanchai, R., Povey, D.: Discriminative training for HMM-based offline handwritten character recognition. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 114–118 (2003)
54. Nopsuwanchai, R., Biem, A., Clocksin, W.F.: Maximization of mutual information for offline Thai handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(8), 1347–1351 (2006)
55. Nußbaum-Thom, M., Wiesler, S., Sundermeyer, M., Plahl, C., Hahn, S., Schlüter, R., Ney, H.: The RWTH 2009 Quaero ASR evaluation system for English and German. In: Interspeech, Makuhari, Japan, September 2010
56. Olive, J.: Multilingual automatic document classification analysis and translation (MADCAT). Proposer Information Pamphlet SOL BAA 07-38, DARPA/IPTO (2007)
57. Ortmanns, S., Ney, H.: Look-ahead techniques for fast beam search. *Comput. Speech Lang.* **14**(1), 15–32 (2000)
58. Ortmanns, S., Ney, H., Aubert, X.: A word graph algorithm for large vocabulary continuous speech recognition. *Comput. Speech Lang.* **11**(1), 43–72 (1997)
59. Padmanabhan, M., Saon, G., Zweig, G.: Lattice-based unsupervised MLLR for speaker adaptation. In: ISCA ITRW Automatic Speech Recognition: Challenges for the Millennium, Paris, France 2000

60. Pastor, A.G., Khoury, I., Juan, A.: Windowed Bernoulli mixture HMMs for Arabic handwritten word recognition. In: International Conference on Frontiers in Handwriting Recognition (ICFHR), Kalkota, India, November 2010
61. Pechwitz, M., Snoussi Maddouri, S., Mägner, V., Ellouze, N., Amiri, H.: IFN/ENIT-database of handwritten Arabic words. In: Colloque International Francophone sur l'Ecrit et le Document (CIFED), Hammamet, Tunisia, October 2002
62. Pitz, M., Wessel, F., Ney, H.: Improved MLLR speaker adaptation using confidence measures for conversational speech recognition. In: International Conference on Spoken Language Processing (ICSLP), Beijing, China (2000)
63. Plahl, C., Hoffmeister, B., Hwang, M.-Y., Lu, D., Heigold, G., Lööf, J., Schlüter, R., Ney, H.: Recent improvements of the RWTH GALE Mandarin LVCSR system. In: Interspeech, Brisbane, Australia, September 2008, pp. 2426–2429 (2008)
64. Povey, D.: Discriminative training for large vocabulary speech recognition. Ph.D. thesis, Cambridge, England (2004)
65. Povey, D., Woodland, P.C.: Minimum phone error and I-smoothing for improved discriminative training. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 1, Orlando, FL, USA (2002)
66. Povey, D., Kanevsky, D., Kingsbury, B., Ramabhadran, B., Saon, G., Visweswarah, K.: Boosted MMI for model and feature-space discriminative training. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Las Vegas, NV, USA, April 2008
67. Prasad, R., Saleem, S., Kamali, M., Meermeier, R., Natarajan, P.: Improvements in hidden Markov model based Arabic OCR. In: International Conference on Pattern Recognition (ICPR), Tampa, FL, USA, December 2008
68. Rybach, D., Hahn, S., Gollan, C., Schlüter, R., Ney, H.: Advances in Arabic broadcast news transcription at RWTH. In: IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Kyoto, Japan, December 2007, pp. 449–454 (2007)
69. Rybach, D., Gollan, C., Heigold, G., Hoffmeister, B., Lööf, J., Schlüter, R., Ney, H.: The RWTH Aachen university open source speech recognition system. In: Interspeech, Brighton, UK, September 2009, pp. 2111–2114 (2009)
70. Schambach, M.-P., Rottland, J., Alary, T.: How to convert a Latin handwriting recognition system to Arabic. In: International Conference on Frontiers in Handwriting Recognition (ICFHR) (2008)
71. Schenck, J., Rigoll, G.: Novel hybrid NN/HMM modelling techniques for on-line handwriting recognition. In: International Workshop on Frontiers in Handwriting Recognition (IWFHR), La Baule, France, October 2006
72. Schlüter, R.: Investigations on discriminative training criteria. Ph.D. thesis, RWTH Aachen University, Aachen, Germany, September 2000
73. Schlüter, R., Müller, B., Wessel, F., Ney, H.: Interdependence of language models and discriminative training. In: IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Keystone, CO, December 1999, vol. 1, pp. 119–122 (1999)
74. Slimane, F., Ingold, R., Kanoun, S., Alimi, M.A., Hennebert, J.: A new Arabic printed text image database and evaluation protocols. In: International Conference on Document Analysis and Recognition (ICDAR), Barcelona, Spain, July 2009, pp. 946–950 (2009)
75. Stolcke, A.: SRILM—an extensible language modeling toolkit. In: International Conference on Spoken Language Processing (ICSLP), Denver, CO, USA, September 2002
76. Valente, F., Vepa, J., Plahl, C., Gollan, C., Hermansky, H., Schlüter, R.: Hierarchical neural networks feature extraction for LVCSR system. In: Interspeech, Antwerp, Belgium, August 2007, pp. 42–45 (2007)
77. von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: reCAPTCHA: Human-based character recognition via web security measures. Science **321**(5895), 1465–1468 (2008)

78. Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X.A., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland, P.: The HTK Book (for HTK Version 3.4). Cambridge University Engineering Department, Cambridge (2006)
79. Zhang, J., Jin, R., Yang, Y., Hauptmann, A.G.: Modified logistic regression: an approximation to SVM and its applications in large-scale text categorization. In: International Conference on Machine Learning (ICML), August 2003

Chapter 10

Arabic Handwriting Recognition Using Bernoulli HMMs

Ihab Alkhoury, Adrià Giménez, and Alfons Juan

Abstract Hidden Markov models (HMMs) are now widely used for off-line handwriting recognition in many languages and, in particular, in Arabic. As in speech recognition, they are usually built from shared, embedded HMMs at the symbol level, in which state-conditional probability density functions are modeled with Gaussian mixtures. In contrast to speech recognition, however, it is unclear which kinds of features should be used and, indeed, very different feature sets are in use today. Among them, we have recently proposed to simply use columns of raw, binary image pixels, which are directly fed into embedded Bernoulli (mixture) HMMs, that is, embedded HMMs in which the emission probabilities are modeled with Bernoulli mixtures. The idea is to bypass feature extraction and ensure that no discriminative information is filtered out during feature extraction, which in some sense is integrated into the recognition model. In this chapter, we review this idea along with some extensions that are currently providing state-of-the-art results on Arabic handwritten word recognition.

10.1 Introduction

Hidden Markov models (HMMs) are now widely used for off-line handwriting recognition in many languages and, in particular, in Arabic [7, 8]. Arabic is spoken by 234 million people and important in the culture of many more [6]. Given a text (line or word) image, it is first transformed into a sequence of fixed-dimension feature vectors, and then fed into an HMM-based decoder to find its most probable transcription. Following the conventional approach in speech recognition [11], HMMs at the global (line or word) level are built from shared, *embedded* HMMs at

I. Alkhoury (✉) · A. Giménez · A. Juan
DSIC, Universitat Politècnica de València, Camí de Vera s/n, 46022 València, Spain
e-mail: ialkhoury@dsic.upv.es

A. Giménez
e-mail: agimenez@dsic.upv.es

A. Juan
e-mail: ajuan@dsic.upv.es

the character (subword) level, which are usually simple in terms of number of states and topology. In the common case of real-valued feature vectors, state-conditional probability (density) functions are modeled as Gaussian mixtures since, as with finite mixture models in general, their complexity can be easily adjusted to the available training data by simply varying the number of components.

After decades of research in speech recognition, the use of certain real-valued speech features and embedded Gaussian (mixture) HMMs is a de facto standard [11]. However, in the case of handwriting recognition, there is no such standard and, indeed, very different sets of features are in use today. In [2], we proposed to bypass feature extraction and to directly feed columns of raw, binary pixels into *embedded Bernoulli (mixture) HMMs (BHMMs)*, that is, embedded HMMs in which the emission probabilities are modeled with Bernoulli mixtures. The basic idea is to ensure that no discriminative information is filtered out during feature extraction, which in some sense is integrated into the recognition model. In this chapter, we review this idea along with some extensions that are currently providing state-of-the-art results on Arabic handwritten word recognition.

In what follows, we briefly review Bernoulli mixtures (Sect. 10.2), Bernoulli HMMs (Sect. 10.3), BHMM-based handwriting recognition (Sect. 10.4), maximum likelihood parameter estimation (Sect. 10.5), and our basic extension to plain BHMMs, which will be referred to as *windowed BHMMs* (Sect. 10.6). Empirical results are then reported in Sect. 10.7. To our knowledge, they are the best results published to date on the well-known IFN/ENIT database of Arabic handwritten Tunisian town names [10]. Our concluding remarks are given in Sect. 10.8.

10.2 Bernoulli Mixture

Let \mathbf{o} be a D -dimensional feature vector. A finite mixture is a probability (density) function of the form:

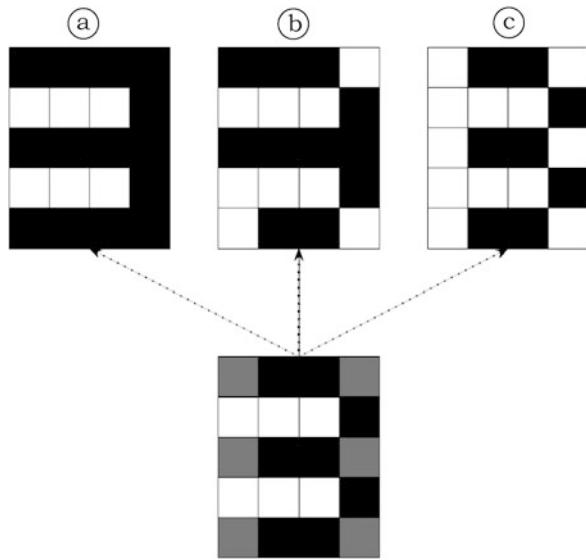
$$P(\mathbf{o} | \boldsymbol{\Theta}) = \sum_{k=1}^K \pi_k P(\mathbf{o} | k, \boldsymbol{\Theta}'), \quad (10.1)$$

where K is the number of mixture components, π_k is the k th component coefficient, and $P(\mathbf{o} | k, \boldsymbol{\Theta}')$ is the k th component-conditional probability (density) function. The mixture is controlled by a parameter vector $\boldsymbol{\Theta}$ comprising the mixture coefficients and a parameter vector for the components, $\boldsymbol{\Theta}'$. It can be seen as a generative model that first selects the k th component with probability π_k and then generates \mathbf{o} in accordance with $P(\mathbf{o} | k, \boldsymbol{\Theta}')$.

A Bernoulli mixture model is a particular case of (10.1) in which each component k has a D -dimensional Bernoulli probability function governed by its own vector of parameters or *prototype* $\mathbf{p}_k = (p_{k1}, \dots, p_{kD})^t \in [0, 1]^D$,

$$P(\mathbf{o} | k, \boldsymbol{\Theta}') = \prod_{d=1}^D p_{kd}^{o_d} (1 - p_{kd})^{1-o_d}, \quad (10.2)$$

Fig. 10.1 Three binary images (**a**, **b** and **c**) are shown as being generated from a Bernoulli prototype depicted as a gray image (black = 1, white = 0, gray = 0.5)



where p_{kd} is the probability for bit d to be 1. Note that this equation is just the product of independent, unidimensional Bernoulli probability functions. Therefore, for a fixed k , it can not capture any kind of dependencies or correlations between individual bits.

Consider the example given in Fig. 10.1. Three binary images (**a**, **b** and **c**) are shown as being generated from a Bernoulli prototype depicted as a gray image (black = 1, white = 0, gray = 0.5). The prototype has been obtained by averaging images **a** and **c**, and it is the best approximate solution to assign a high, equal probability to these images. However, as individual pixel probabilities are not conditioned to other pixel values, there are $2^6 = 64$ different binary images (including **a**, **b** and **c**) into which the whole probability mass is uniformly distributed. It is then not possible, using a single Bernoulli prototype, to assign a probability of 0.5 to **a** and **c**, and null probability to any other image such as **b**. Nevertheless, this limitation can be easily overcome by using a Bernoulli mixture and allowing a different prototype to each different image shape. That is, in our example, a two-component mixture of equal coefficients, and prototypes **a** and **b**, does the job.

10.3 Bernoulli HMM

Let $O = (o_1, \dots, o_T)$ be a sequence of feature vectors. An HMM is a probability (density) function of the form:

$$P(O | \Theta) = \sum_{q_0, \dots, q_{T+1}} \prod_{t=0}^T a_{q_t q_{t+1}} \prod_{t=1}^T b_{q_t}(o_t), \quad (10.3)$$

where the sum is over all possible *paths* (state sequences) q_0, \dots, q_{T+1} , such that $q_0 = I$ (special *initial* or *start* state), $q_{T+1} = F$ (special *final* or *stop* state), and $q_1, \dots, q_T \in \{1, \dots, M\}$, M being the number of regular (non-special) states of the HMM. On the other hand, for any regular states i and j , a_{ij} denotes the *transition* probability from i to j , while b_j is the *observation* probability (density) function at j .

A Bernoulli (mixture) HMM (BHMM) is an HMM in which the probability of observing \mathbf{o}_t , when $q_t = j$, is given by a Bernoulli mixture probability function for the state j :

$$b_j(\mathbf{o}_t) = \sum_{k=1}^K \pi_{jk} \prod_{d=1}^D p_{jkd}^{o_{td}} (1 - p_{jkd})^{1-o_{td}}, \quad (10.4)$$

where π_{jk} and p_{jk} are, respectively, the prior and prototype of the k th mixture component in state j .

Consider the upper part of Fig. 10.2, where a BHMM example for the number 3 is shown, together with a binary image generated from it. It is a three-state model with single prototypes attached to states 1 and 2, and a two-component mixture assigned to state 3. In contrast to the example in Fig. 10.1, prototypes do not account for the whole digit realizations, but only for single columns. This column-by-column emission of feature vectors attempts to better model horizontal distortions at character level and, indeed, it is the usual approach in both speech and handwriting recognition when continuous-density (Gaussian mixture) HMMs are used. The reader can check that, by direct application of Eq. (10.3) and taking into account the existence of two different state sequences, the probability of generating the binary image generated from this BHMM example is 0.063.

As discussed in the introduction, BHMMs at global (line or word) level are built from shared, embedded BHMMs at character level. More precisely, let C be the number of different characters (symbols) from which global BHMMs are built, and assume that each character c is modeled with a different BHMM of parameter vector Θ_c . Let $\Theta = \{\Theta_1, \dots, \Theta_C\}$, and let $O = (\mathbf{o}_1, \dots, \mathbf{o}_T)$ be a sequence of feature vectors generated from a sequence of symbols $S = (s_1, \dots, s_L)$, with $L \leq T$. The probability of O can be calculated, using embedded HMMs for its symbols, as:

$$P(O | S, \Theta) = \sum_{i_1, \dots, i_{L+1}} \prod_{l=1}^L P(\mathbf{o}_{i_l}, \dots, \mathbf{o}_{i_{l+1}-1} | \Theta_{s_l}), \quad (10.5)$$

where the sum is carried out over all possible segmentations of O into L segments, that is, all sequences of indices i_1, \dots, i_{L+1} such that

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

and $P(\mathbf{o}_{i_l}, \dots, \mathbf{o}_{i_{l+1}-1} | \Theta_{s_l})$ refers to the probability (density) of the l th segment, as given by (10.3) using the HMM associated with symbol s_l .

Consider now the lower part of Fig. 10.2. An embedded BHMM for the number 31 is shown, which is the result of concatenating BHMMs for the digit 3, blank

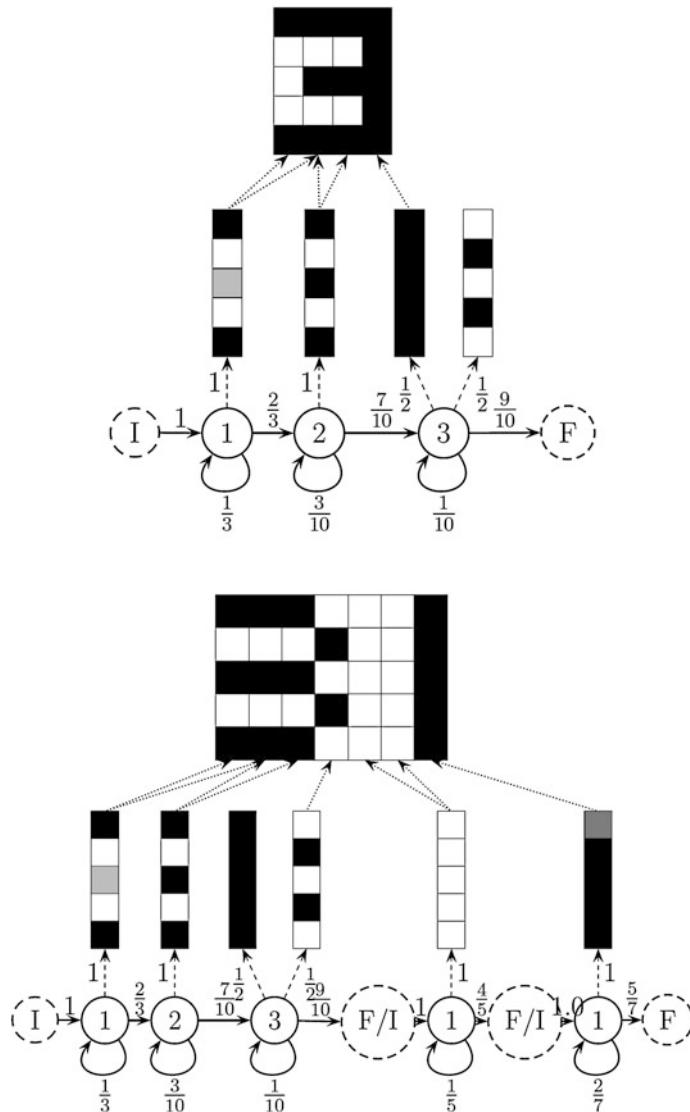


Fig. 10.2 BHMM examples for the numbers 3 (top) and 31 (bottom), together with binary images generated from them. Note that the BHMM example for the number 3 is also embedded into that for the number 31. Bernoulli prototype probabilities are represented using the following color scheme: black = 1, white = 0, gray = 0.5, and light gray = 0.1.

space, and digit 1, in that order. Note that the BHMMs for blank space and digit 1 are simpler than that for digit 3. Also note that the BHMM for digit 3 is shared between the two embedded BHMMs shown in the figure. The binary image of the number 31 shown above can only be generated from two paths, as indicated by

the arrows connecting prototypes to image columns, which only differ in the state generating the second image column (either state 1 or 2 of the BHMM for the first symbol). It is straightforward to check that, according to (10.5), the probability of generating this image is 0.0004.

10.4 BHMM-Based Handwriting Recognition

Given an observation sequence $O = (o_1, \dots, o_T)$, its most probable transcription is obtained by application of the conventional Bayes decision rule:

$$w^* = \arg \max_{w \in W} p(w | O) \quad (10.6)$$

$$= \arg \max_{w \in W} p(w)p(O | w), \quad (10.7)$$

where W is the set of possible transcriptions; $p(w)$ is usually approximated by an *n-gram language model* [4]; and $p(O | w)$ is a *text image model* which, in this work, is modeled as a BHMM (built from shared, embedded BHMMs at character level), as defined in Eq. (10.5). A particularly interesting case arises when the set of possible transcriptions reduces to a (small) finite set of *words (class labels)*. In this case, $p(w)$ is simply the *prior* probability of word w , while $p(O | w)$ is the probability of observing O given that it corresponds to a handwritten version of word w .

10.4.1 The Forward Algorithm

In order to efficiently compute $p(O | w)$ as a BHMM probability of the form given in Eq. (10.5), we use a dynamic programming method known as the *forward algorithm* [11, 12]. For each time t , symbol s_l and state j from the HMM for symbol s_l , we define the *forward probability* $\alpha_{lt}(j)$ as:

$$\alpha_{lt}(j) = P(O_1^t, q_t = (l, j) | S, \Theta), \quad (10.8)$$

that is, the probability of generating O up to its t th element and ending at state j from the HMM for symbol s_l . This definition includes (10.5) as the particular case in which $t = T$, $l = L$ and $j = F_{s_L}$; that is,

$$P(O | S, \Theta) = \alpha_{LT(F_{s_L})}. \quad (10.9)$$

To compute $\alpha_{LT(F_{s_L})}$, we must first take into account that, for each position l in S except for the first, the initial state of the HMM for symbol s_l is joined with the final state of its preceding HMM, i.e.,

$$\alpha_{lt}(I_{s_l}) = \alpha_{l-1t}(F_{s_{l-1}}) \quad \begin{cases} 1 < l \leq L, \\ 1 \leq t \leq T. \end{cases} \quad (10.10)$$

Keeping (10.10) in mind, we can proceed at symbol level as with conventional HMMs. For the final states, we have:

$$\alpha_{lt}(F_{sl}) = \sum_{j=1}^{M_{sl}} \alpha_{lt}(j) a_{slj} F_{sl} \quad \begin{array}{l} 1 \leq l \leq L, \\ 1 \leq t \leq T, \end{array} \quad (10.11)$$

while, for regular states, $1 \leq j \leq M_{sl}$, we have:

$$\alpha_{lt}(j) = \left[\sum_{i \in \{I_{sl}, 1, \dots, M_{sl}\}} \alpha_{lt-1}(i) a_{sli} j \right] b_{sli}(\mathbf{o}_t), \quad (10.12)$$

with $1 \leq l \leq L$ and $1 < t \leq T$. The base case is for $t = 1$:

$$\alpha_{l1}(i) = \begin{cases} a_{s_1 I_{s_1} i} b_{s_1 i}(\mathbf{o}_1), & l = 1, 1 \leq i \leq M_{s_1}, \\ 0 & \text{otherwise.} \end{cases} \quad (10.13)$$

The forward algorithm uses a dynamic programming table for $\alpha_{lt}(\cdot)$ which is computed forward in time to avoid repeated computations.

Figure 10.3 shows an application example of the forward algorithm to the BHMM and observation of Fig. 10.2 (bottom). Non-null (and a few null) entries of the dynamic programming table are represented by graph nodes aligned with states (vertically) and time (horizontally). Node borders are drawn in black or gray, depending on whether they are in valid paths (i.e., those from which the observation sequence can be generated) or not. Also, those associated with special states are drawn with dotted lines. Numbers at the top of each node refer to $\alpha_{lt}(\cdot)$ and thus, for instance, the probability of generating O up to the third image column and ending at state 2 of the BHMM for the first symbol is $\alpha_{13}(2) = \frac{10}{450}$. Computation dependencies between nodes are represented by arrows, which are labeled above by, first, the transition probability, and then the observation probability at the target state (see Eq. (10.4)). For instance, the numbers above the arrow pointing to node $\alpha_{13}(4)$ are: $a_{s_1 23} \cdot b_{s_1 3}(\mathbf{o}_4) = \frac{7}{10} \cdot (\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1^5) = \frac{7}{10} \cdot \frac{1}{2}$.

From Fig. 10.3, we can clearly see that, as indicated at the end of Sect. 10.3, there are only two paths from which the observation can be generated. They share all nodes drawn with black borders except the two nodes aligned with the second observation vector. In accordance with Eq. (10.9), the probability of the observation sequence is $\alpha_{37}(F) = 0.0004$.

10.4.2 The Backward Algorithm

The *backward algorithm* is similar to the forward algorithm but, as its name indicates, it uses a dynamic programming table which is computed backward in time [11, 12]. The basic definition in this case is the *backward probability*:

$$\beta_{lt}(j) = P(O_{t+1}^T | q_t = (l, j), S, \Theta), \quad (10.14)$$

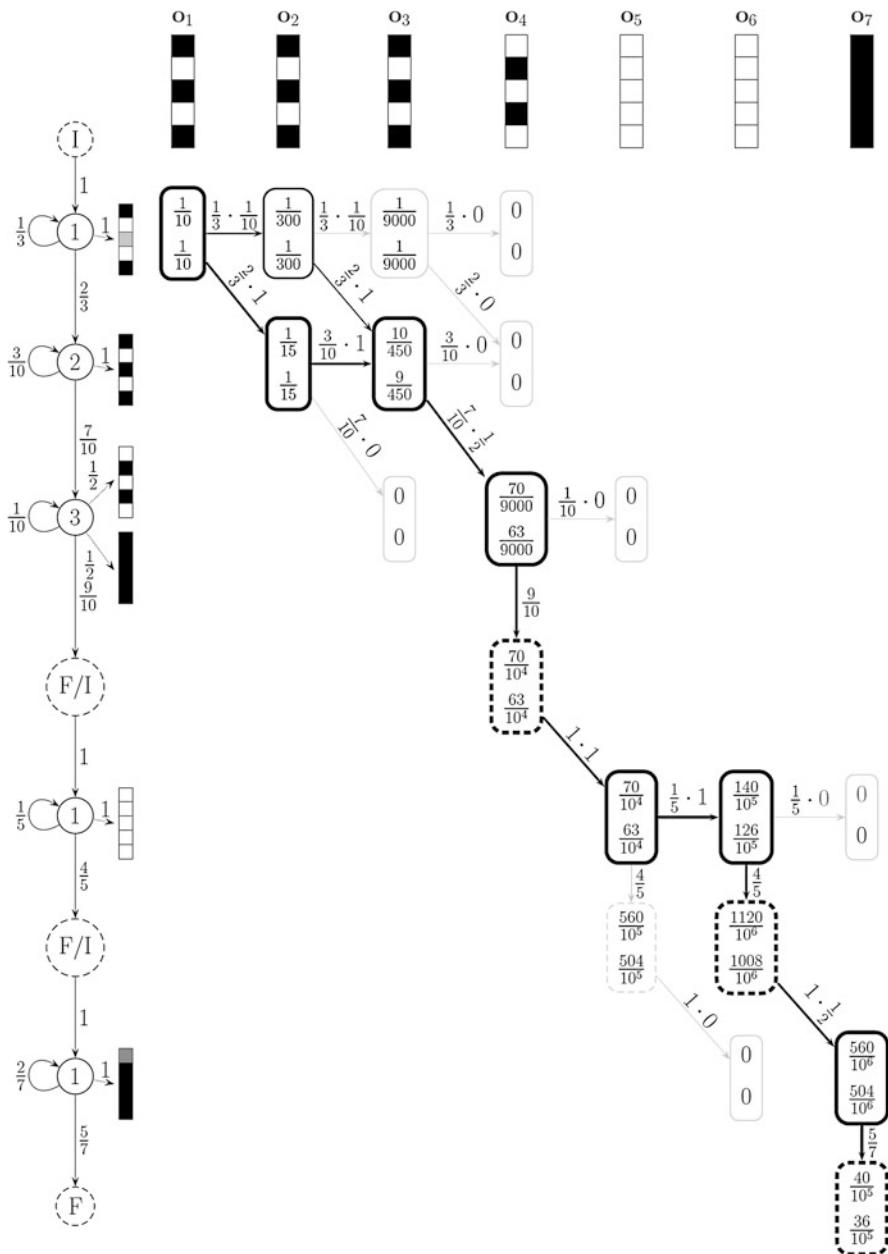


Fig. 10.3 Application example of the forward and Viterbi algorithms to the BHMM and observation of Fig. 10.2 (bottom). Numbers at the *top* of the nodes denote forward probabilities, while those at the *bottom* refer to Viterbi scores

which measures the probability (density) of generating O_{t+1}^T given that the t th vector was generated in state j of the BHMM for the symbol s_l . Using this definition, Eq. (10.5) can be rewritten as:

$$P(O | S, \Theta) = \sum_{j=1}^{M_{s_l}} a_{s_l I_{s_l} j} b_{s_l j}(\mathbf{o}_1) \beta_{11}(j). \quad (10.15)$$

Taking into account that:

$$\beta_{lt}(F_{s_l}) = \beta_{l+1t}(I_{s_{l+1}}) \quad \begin{cases} 1 \leq l < L, \\ 1 \leq t < T, \end{cases} \quad (10.16)$$

the backward probability for the initial and regular states, $i \in \{I_{s_l}, 1, \dots, M_{s_l}\}$, can be efficiently computed as:

$$\beta_{lt}(i) = a_{s_l i} F_{s_l} \beta_{lt}(F_{s_l}) + \sum_{j=1}^{M_{s_l}} a_{s_l i j} b_{s_l j}(\mathbf{o}_{t+1}) \beta_{l+1t}(j) \quad \begin{cases} 1 \leq l \leq L, \\ 1 \leq t < T, \end{cases} \quad (10.17)$$

where the base case is defined for $t = T$ as:

$$\beta_{lT}(i) = \begin{cases} a_{s_L i} F_{s_L} & l = L, 1 \leq i \leq M_{s_L}, \\ 0 & \text{otherwise.} \end{cases} \quad (10.18)$$

10.4.3 The Viterbi Algorithm

Although the forward and backward algorithms efficiently compute the exact value of $P(O | S, \Theta)$, it is common practice to approximate it by the *Viterbi* or *maximum approximation*, in which the sums in Eqs. (10.3) and (10.5) are replaced by the max operator, i.e.,

$$P(O | S, \Theta) \approx \max_{\substack{i_1, \dots, i_L, q_1, \dots, q_T}} \prod_{l=1}^L \hat{P}(\mathbf{o}_{i_l}^{i_{l+1}-1} | \Theta_{s_l}), \quad (10.19)$$

where \hat{P} is defined as:

$$\hat{P}(\mathbf{o}_{i_l}^{i_{l+1}-1} | \Theta_{s_l}) = a_{s_l I_{s_l} q_l} \cdot \prod_{t=i_l}^{i_{l+1}-2} a_{s_l q_t q_{t+1}} \cdot a_{s_l q_{l+1-1} F_{s_l}} \cdot \prod_{t=i_l}^{i_{l+1}-1} b_{s_l q_t}(\mathbf{o}_t). \quad (10.20)$$

In contrast to the exact definition, this approximation allows us to identify a single, best state sequence or *path* associated with the given observation sequence. The well-known *Viterbi algorithm* efficiently computes this approximation, using dynamic programming recurrences similar to those used by the forward algorithm.

Formally, we need to compute the probability $Q(l, t, j)$ of the most likely path up to time t that ends with the state j from the BHMM for symbol s_l . For the special states, it can be computed as:

$$Q(l, t, I_{s_l}) = Q(l - 1, t, F_{s_{l-1}}) \quad \begin{cases} 1 < l \leq L, \\ 1 \leq t \leq T, \end{cases} \quad (10.21)$$

$$Q(l, t, F_{s_l}) = \max_{1 \leq j \leq M_{s_l}} Q(l, t, j) a_{s_l j} F_{s_l} \quad \begin{cases} 1 \leq l \leq L, \\ 1 \leq t \leq T, \end{cases} \quad (10.22)$$

while, for the regular states with $1 \leq l \leq L$ and $1 < t \leq T$, we have:

$$Q(l, t, j) = \left[\max_{i \in \{I_{s_l}, 1, \dots, M_{s_l}\}} Q(l, t - 1, i) a_{s_l i} j \right] b_{s_l j}(\mathbf{o}_t). \quad (10.23)$$

The base case is for $t = 1$:

$$Q(l, 1, i) = \begin{cases} a_{s_1 I_{s_1} i} b_{s_1 i}(\mathbf{o}_1) & l = 1, 1 \leq i \leq M_{s_1}, \\ 0 & \text{otherwise.} \end{cases} \quad (10.24)$$

Clearly, the Viterbi algorithm can be seen as a minor modification of the forward algorithm in which only the most probable is considered in each node computation. Indeed, the application example shown in Fig. 10.3 is used for both the forward and Viterbi algorithms. Now, however, the relevant numbers are those included at the bottom of each node, which denote $Q(l, t, j)$; i.e., at row 2 and column 3, we have $Q(1, 3, 2) = \frac{9}{450}$. Consider the generation of the third observation vector at the second state (for the first symbol). It occurs after the generation of the second observation vector, either at the first or the second state, but we only take into account the most likely case. Formally, the corresponding Viterbi score is computed as:

$$Q(1, 3, 2) = \max \left\{ \frac{1}{15} \cdot \frac{3}{10} \cdot 1, \frac{1}{300} \cdot \frac{2}{3} \cdot 1 \right\} = \max \left\{ \frac{9}{450}, \frac{1}{450} \right\} = \frac{9}{450}.$$

Note that forward probabilities do not differ from Viterbi scores up to $Q(1, 3, 2)$, since it corresponds to the first (and only) node with two incoming paths. The Viterbi approximation to the exact probability of generating the observation sequence is obtained at the final node: $Q(3, 7, F) = 0.00036$. The most likely path, drawn with thick lines, is retrieved by starting at this node and moving backwards in time in accordance with the computation of Viterbi scores. As usual in practice, the final Viterbi score in this example (0.00036) is a tight lower bound of the exact probability (0.00040).

10.5 Maximum Likelihood Parameter Estimation

Maximum likelihood estimation of the parameters governing an embedded BHMM does not differ significantly from the conventional Gaussian case, and it can be carried out using the well-known expectation maximization (EM, Baum–Welch)

re-estimation formulae [11, 12]. Let $(O_1, S_1), \dots, (O_N, S_N)$, be a collection of N training samples in which the n th observation has length T_n , $O_n = (\mathbf{o}_{n1}, \dots, \mathbf{o}_{nT_n})$, and was generated from a sequence of L_n symbols ($L_n \leq T_n$), $S_n = (s_{n1}, \dots, s_{nL_n})$. At iteration r , the E step requires the computation, for each training sample n , of their corresponding forward and backward probabilities (see (10.8) and (10.14)), as well as the expected value for its t th feature vector to be generated from the k th component of the state j in the HMM for symbol s_l ,

$$z_{nltk}^{(r)}(j) = \frac{\pi_{s_{nl}jk}^{(r)} \prod_{d=1}^D p_{s_{nl}jkd}^{(r)} o_{ntd} (1 - p_{s_{nl}jkd}^{(r)})^{1-o_{ntd}}}{b_{s_{nl}j}^{(r)}(\mathbf{o}_{nt})},$$

for each t, k, j and l .

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

$$\mathbf{p}_{cjk}^{(r+1)} = \frac{1}{\gamma_{ck}(j)} \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j) \mathbf{o}_{nt}}{P(O_n | S_n, \Theta^{(r)})}, \quad (10.25)$$

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

$$\gamma_{ck}(j) = \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j)}{P(O_n | S_n, \Theta^{(r)})}, \quad (10.26)$$

and $\xi_{nltk}^{(r)}(j)$ is the probability for the t th feature vector of the n th sample to be generated from the k th component of the state j in the HMM for symbol s_l ,

$$\xi_{nltk}^{(r)}(j) = \alpha_{nlt}^{(r)}(j) z_{nltk}^{(r)}(j) \beta_{nlt}^{(r)}(j). \quad (10.27)$$

Similarly, the k th component coefficient of the state j in the HMM for character c has to be updated as:

$$\pi_{cjk}^{(r+1)} = \frac{1}{\gamma_c(j)} \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j)}{P(O_n | S_n, \Theta^{(r)})}, \quad (10.28)$$

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

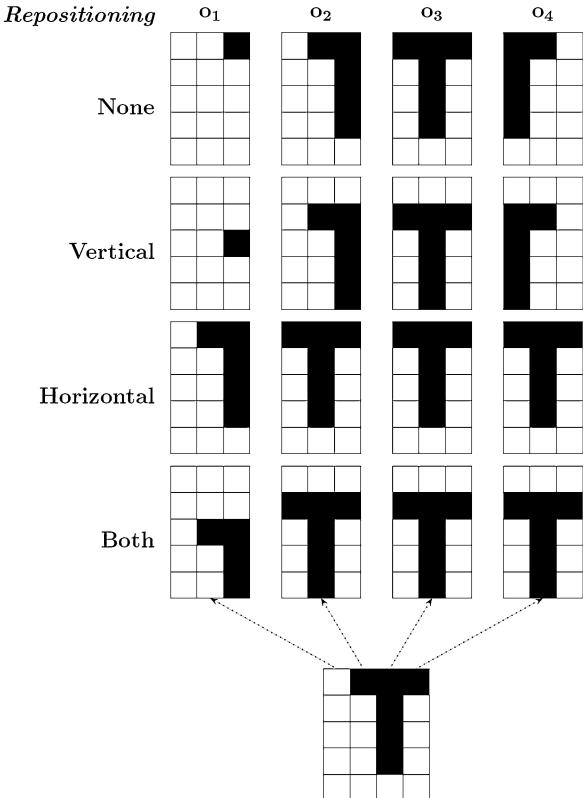
$$\gamma_c(j) = \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \alpha_{nlt}^{(r)}(j) \beta_{nlt}^{(r)}(j)}{P(O_n | S_n, \Theta^{(r)})}. \quad (10.29)$$

To avoid null probabilities in Bernoulli prototypes, they can be smoothed by linear interpolation with a flat (uniform) prototype, **0.5**,

$$\tilde{\mathbf{p}} = (1 - \delta) \mathbf{p} + \delta \mathbf{0.5}, \quad (10.30)$$

where, for instance, $\delta = 10^{-6}$.

Fig. 10.4 Example of transformation of a 4×5 binary image (bottom) into a sequence of four 15-dimensional binary feature vectors $O = (o_1, o_2, o_3, o_4)$ using a window of width 3. The standard method (no repositioning) is compared with the three repositioning methods considered: vertical, horizontal, and both directions



10.6 Windowed BHMMs

Given a binary image normalized in height to H pixels, we may think of a feature vector o_t as its column at position t or, more generally, as a concatenation of columns in a window of W columns in width, centered at position t . This generalization has no effect on the definition of BHMM nor on its maximum likelihood estimation, though it might be very helpful to better capture the image context at each horizontal position of the image. As an example, Fig. 10.4 shows a binary image of 4 columns and 5 rows, which is transformed into a sequence of four 15-dimensional feature vectors (first row) by application of a sliding window of width 3. For clarity, feature vectors are depicted as 3×5 subimages instead of 15-dimensional column vectors. Note that feature vectors at positions 2 and 3 would be indistinguishable if, as in our previous approach, they were extracted with no context ($W = 1$).

Although one-dimensional, “horizontal” HMMs for image modeling can properly capture nonlinear horizontal image distortions, they are somewhat limited when dealing with vertical image distortions, and this limitation might be particularly strong in the case of feature vectors extracted with significant context. To overcome this limitation, we have considered three methods of window *repositioning* after window extraction: *vertical*, *horizontal*, and *both*. The basic idea is to first compute

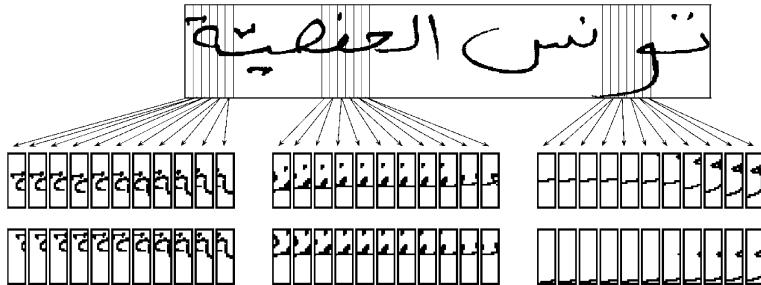


Fig. 10.5 Original sample *pf069_011* from IFN/ENIT database (*top*) and its sequence of feature vectors produced with and without (both) repositioning (*center* and *bottom*, respectively)

the center of mass of the extracted window, which is then repositioned (translated) to align its center to the center of mass. This is done in accordance with the chosen method, that is, horizontally, vertically, or in both directions. Obviously, the feature vector actually extracted is that obtained after repositioning. An example of feature extraction is shown in Fig. 10.4, in which the standard method (no repositioning) is compared with the three repositioning methods considered.

To illustrate the effect of repositioning with real data, Fig. 10.5 shows the sequence of feature vectors extracted from a real sample of the IFN/ENIT database, with and without (both) repositioning. As intended, (vertical or both) repositioning has the effect of normalizing vertical image distortions, especially translations.

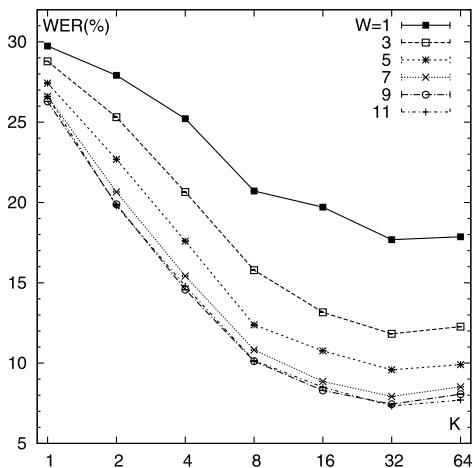
10.7 Experiments

Experiments were carried out on the very popular IFN/ENIT database of Arabic handwritten Tunisian town names [10]. More precisely, we used the IFN/ENIT database in version 2.0, patch level 1e (v2.0p1e), which is exactly the version used as training data in the Arabic handwriting recognition competition held at ICDAR (International Conference on Document Analysis and Recognition) in 2007 [7]. It comprises 32492 Arabic word images written by more than 1000 different writers, from a lexicon of 937 Tunisian town/village names. For the experiments reported below, each image was first rescaled in height to $D = 30$ rows, while keeping the original aspect ratio, and then binarized using Otsu binarization. The resulting set of binary images was partitioned into five folds labeled as a, b, c, d and e, as defined in [7].

10.7.1 Effect of the Window Width

In [3], we have recently found that the sliding window width has a very positive effect on the accuracy of our BHMM-based word recognizer, though, as usual, it has

Fig. 10.6 WER (%) as a function of the number of mixture components (K) for varying sliding window widths (W)

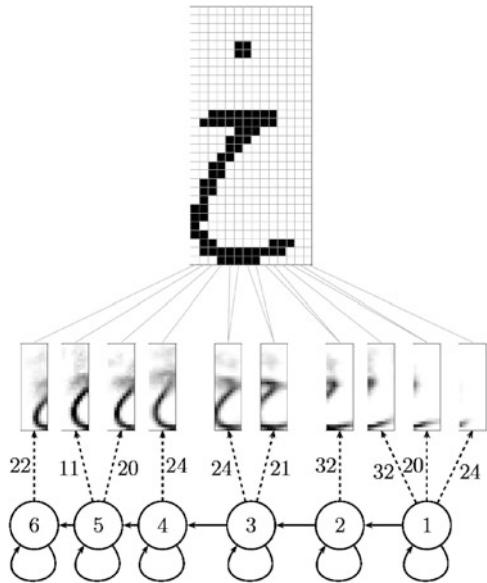


to be combined with an adequate number of components for the state-conditional finite mixture models. This is clearly shown in Fig. 10.6, where the word error rate (WER %) of our BHMM-based recognizer is plotted as a function of the number of mixture components (K), for varying sliding window widths (W). Each WER estimate (plotted point) was obtained by cross-validation with the first four standard folds (abcd), using BHMMs of 6 states. For $K = 1$, BHMMs were trained by first segmenting the training set with a “neutral” model, and then using the resulting segments to perform a Viterbi initialization followed by four EM iterations. For $K > 1$, they were trained by first splitting the components of the models trained with $K/2$ components and then, as before, applying four EM iterations. The conventional Viterbi algorithm was used to compute the most probable word for each test word image.

From Fig. 10.6 it becomes clear that the use of a sliding window improves the results to a large extent. In particular, the best result, 7.4 %, is obtained for $W = 9$ and $I = 32$, though very similar results are also obtained for $W = 7$ and $W = 11$. It is worth noting that the best result achieved with no sliding windows ($W = 1$) is 17.7 %.

To get some insight into the behavior of our BHMMs, the model for character $\dot{\chi}$, trained from folds abc with $W = 9$ and $K = 32$, is (partially) shown in Fig. 10.7 (bottom) together with its Viterbi alignment with a real image of the character $\dot{\chi}$, extracted from sample *de05_007* (top). As in Fig. 10.2 (bottom), Bernoulli prototypes are represented as gray images, where the gray level of each pixel measures the probability of its corresponding pixel to be black (white = 0 and black = 1). From these prototypes, it can be seen that the model works as expected; i.e., each state from right to left accounts for a different local part of $\dot{\chi}$, as if the sliding window was moving smoothly from right to left. Also, note that the main stroke of

Fig. 10.7 BHMM for character $\dot{\zeta}$, trained from folds abc with $W = 9$ and $K = 32$ (bottom), together with its Viterbi alignment with a real image of the character $\dot{\zeta}$, extracted from sample *de05_007* (top)



the character $\dot{\zeta}$ appears almost neatly drawn in most prototypes, whereas its upper dot appears blurred, probably due to a comparatively higher variability in window position.

10.7.2 Effect of the Number of States

In accordance with the empirical results reported in [5], we have only tried BHMMs of 6 states in the experiment described above. However, as discussed in [1], letters in Arabic script differ significantly in length, and thus it might not be appropriate to model all of them using BHMMs of identical numbers of states. With this idea in mind, a new experiment was carried out, similar to that described above, but with a fixed sliding window of $W = 9$ and a variable number of states per character. To decide the number of states for each character, we first Viterbi-segmented all training data using BHMMs of 4 states, and then computed the average length of the segments associated with each character. Given an average segment length for character c , \bar{T}_c , its number of states was set to $F \cdot \bar{T}_c$, where F is a factor measuring the average number of states that is required to emit a feature vector. Thus, its inverse, $\frac{1}{F}$, can be interpreted as a *state load*, that is, the average number of feature vectors that are emitted in each state. For instance, $F = 0.2$ means that only a fraction of 0.2 state is required to emit a feature vector or, alternatively, that $\frac{1}{0.2} = 5$ feature vectors are emitted on average in each state. Figure 10.8 shows the WER obtained as a function of F , $F \in \{0.2, 0.3, 0.4, 0.5\}$, for varying values of the number of mixture components.

Fig. 10.8 WER (%) as a function of the factor F for varying values of the number of mixture components (K)

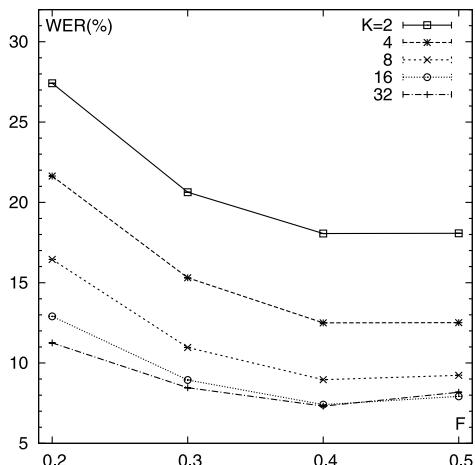
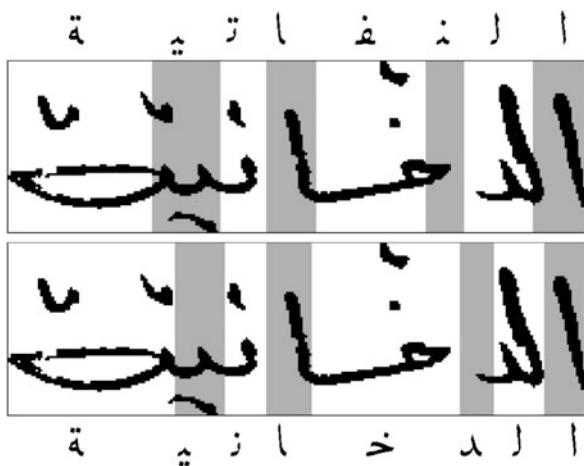


Fig. 10.9 Sample *dm33_037* incorrectly recognized with BHMMs of 6 states (*top*), but correctly recognized with BHMMs of variable number of states (*bottom*). In both cases, the recognized word has been Viterbi-aligned at the character level (background color) and state level (bottom and upper ticks)



The best result plotted in Fig. 10.8 is a WER of 7.3 %, using $F = 0.4$ and $K = 32$. This result is slightly better than the 7.4 % obtained with 6 states per character.

In Fig. 10.9, the sample *dm33_037* has been recognized using BHMMs with $W = 9$, $K = 32$, and both 6 states (top) and a variable number of states, with $F = 0.4$ (bottom). In both cases, the recognized word has been Viterbi-aligned at the character level (background color) and state level (bottom and upper ticks). Although the BHMMs of 6 states produce a recognition error, the BHMMs of the variable number of states are able to recognize the correct word, "النفّاثة" (bottom). Note that there are two letters, "ن" and "ذ", that are written at the same vertical position or, more specifically, at a specific column, and thus it is very difficult for our BHMMs to recognize them as two different letters. On the other hand, the

Table 10.1 Word error rate (WER %) of the BHMM-based recognizer (with $W = 9$, $K = 32$, and $F = 0.4$) in different training-test combinations of the abcde folds, for four repositioning methods: none, vertical, horizontal, and both directions

WER %					
Training	Test	None	Vertical	Horizontal	Both
abc	d	7.5	4.7	8.4	4.8
abd	c	6.9	3.6	7.7	3.8
acd	b	7.7	4.5	8.1	4.4
bcd	a	7.6	4.4	8.2	4.6
abcd	e	12.3	6.1	12.4	6.1
abcde	e	4.0	2.2	3.9	2.0

incorrectly recognized word (top) is not very different in shape from the correct one; e.g. the characters “ج” and “ج” are very similar (type B [9]).

10.7.3 Effect of Repositioning and Final Results

In the experiments described above, we have not tried window repositioning after window extraction but, as discussed in Sect. 10.6, many recognition errors of our BHMM-based classifier might be due to its limited capability to properly model vertical image distortions. In order to study the effect of repositioning on the classification accuracy, the standard method (no repositioning) was compared with the three repositioning methods described in Sect. 10.6: vertical, horizontal, and both directions. This was done with $W = 9$, $K = 32$, and $F = 0.4$, for the four partitions considered in the previous experiments (abc-d, abd-c, acd-b, and bcd-a), and also for the partitions abcd-e and abcde-e, which are commonly used to compare classifiers in the IFN/ENIT task, especially abcd-e. The results are included in Table 10.1.

As expected, from the results in Table 10.1 it becomes clear that vertical (or both) window repositioning greatly improves the results obtained with the standard method or horizontal repositioning alone. To our knowledge, the result obtained for the abcd-e partition with vertical (or both) repositioning, 6.1 %, is the best result reported on this partition to date. Indeed, it represents a 50 % relative error reduction with respect to the 12.3 % of WER obtained without repositioning which, to our knowledge, was the best result published until now [3].

10.8 Concluding Remarks

Embedded Bernoulli HMMs (BHMMs) have been described and tested for Arabic handwriting recognition on the well-known IFN/ENIT database of handwritten Tunisian town names. Apart from our previous basic approach, in which narrow, one-column slices of binary pixels are fed into BHMMs, we have used a sliding

window of adequate width to better capture the image context at each horizontal position of the word image. Also, we have considered three methods of window repositioning after window extraction to help our BHMM-based recognizer in dealing with vertical image distortions. The experiments reported have carefully studied the effects of the window width, the number of states, and repositioning. As expected, the best results have been obtained with an adequate adjustment of the window width, number of states, number of mixture components, and—which seems even more important—(vertical) window repositioning after window extraction. A WER of 6.1 % has been achieved on the standard abcd-e partition, which, to our knowledge, outperforms the best result known to date.

References

1. Dreuw, P., Heigold, G., Ney, H.: Confidence-based discriminative training for model adaptation in offline Arabic handwriting recognition. In: ICDAR, pp. 596–600 (2009)
2. Giménez, A., Juan, A.: Embedded Bernoulli mixture HMMs for handwritten word recognition. In: Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR 2009), Barcelona, Spain, pp. 896–900 (2009)
3. Giménez, A., Khoury, I., Juan, A.: Windowed Bernoulli mixture HMMs for Arabic handwritten word recognition. In: Proc. of the Int. Conf. on Frontiers in Handwriting Recognition (ICFHR 2010), Kolkata, India (2010)
4. Goodman, J.T.: A bit of progress in language modeling. Tech. rep. (2001)
5. Khoury, I., Giménez, A., Juan, A.: Arabic handwritten word recognition using Bernoulli mixture HMMs. In: Proc. of the 3rd Palestinian Int. Conf. on Computer and Information Technology (PICCIT 2010), Hebron, Palestine (2010)
6. Lorigo, L., Govindaraju, V.: Offline Arabic handwriting recognition: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **28**(5), 712–724 (2006). doi:[10.1109/TPAMI.2006.102](https://doi.org/10.1109/TPAMI.2006.102)
7. Märgner, V., Abed, H.E.: ICDAR 2007—Arabic handwriting recognition competition. In: Proc. of the 9th Int. Conf. on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, vol. 2, pp. 1274–1278 (2007)
8. Märgner, V., Abed, H.E.: ICDAR 2009 Arabic handwriting recognition competition. In: Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR 2009), Barcelona, Spain, pp. 1383–1387 (2009)
9. Märgner, V., Pechwitz, M., Abed, H.E.: ICDAR 2005 Arabic handwriting recognition competition. In: Proc. of the 8th Int. Conf. on Document Analysis and Recognition (ICDAR 2005), Seoul, Korea, vol. 1, pp. 70–74 (2005)
10. Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT—database of handwritten Arabic words. In: 7th Colloque International Francophone sur l’Ecrit et le Document, CIFED, Hammamet, Tunis, pp. 21–23 (2002)
11. Rabiner, L., Juang, B.H.: Fundamentals of Speech Recognition. Prentice Hall, Englewood Cliffs (1993)
12. Young, S., et al.: The HTK Book. Cambridge University Engineering Department, Cambridge (1995)

Chapter 11

Handwritten Farsi Word Recognition Using Hidden Markov Models

Puntis Jifroodian Haghghi and Ching Y. Suen

Abstract One of the most important script groups, which is based on Arabic alphabet, is the Persian/Farsi script. This script is the basis of different languages used in Middle East and Central Asian regions. For the development of Farsi handwritten word recognition systems, the CENPARMI group designed and collected a database. Based on statistical features, a Hidden Markov Model based recognizer is developed. First evaluation of the performance of this recognizer shows promising results.

11.1 An Overview of the Indo-Iranian Languages

The Indo-Iranian languages are a branch of the Indo-European languages. This language family is widely used in central and southern Asia, in Iran, Afghanistan, Iraq, Pakistan, Turkey, India, and Bangladesh. This language family is divided into two subfamilies, known as Indic and Iranian. Table 11.1 shows the Iranian and Indic subfamilies of Indo-Iranian languages and some instances of their member languages [1].

In this section, we will describe more about the Farsi language and the characteristics of its scripts.

11.1.1 Farsi Language

Farsi (Persian) is widely used in Iran, Afghanistan, Tajikistan, Uzbekistan, Bahrain, and the surrounding areas, as shown in Fig. 11.1. Farsi has been a medium for literary and scientific contributions to the Islamic world for five centuries. Prior to

P.J. Haghghi (✉) · C.Y. Suen

CENPARMI (Center for Pattern Recognition and Machine Intelligence), Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec, Canada
e-mail: p_jifroo@cenparmi.concordia.ca

C.Y. Suen

e-mail: suen@cenparmi.concordia.ca

Table 11.1 Iranian and Indic subfamilies of Indo-Iranian languages and some instances of their member languages

INDIC	IRANIAN		
Sanskrit	Avestan	Old Persian	Scythian
Prakrit	Pashto	Persian (Farsi)	
Pali		Arabic	
Gujarati		Kurdish	
Marathi		Ossetic	
Hindustani		Baluchi	
Hindi			
Urdu			
Benagali			
Bihari			
Sindhi			
Bhili			
Rajasthani			
Punjabi			
Pahari			

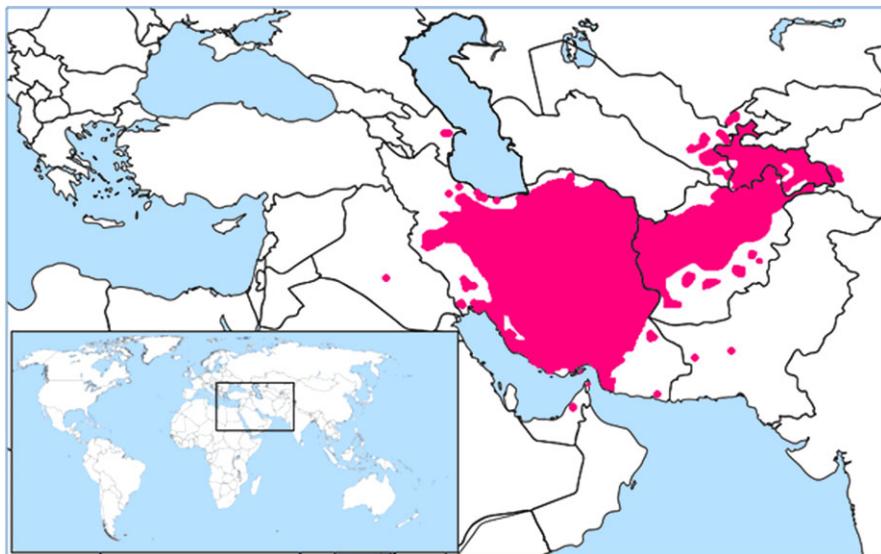


Fig. 11.1 Areas shown in red are Farsi-speaking areas in Asia

British colonization, Farsi was widely used as a second language in the southwestern region of the Asian continent. It took prominence as the language of culture and education in several Muslim courts in southern Asia and became the “official language” under the Mughal emperors [2].

Fig. 11.2 Four variants of the word Payment in Farsi



Fig. 11.3 Four variants of the word Money Order which is equivalent to حوالات (Havaleh) in Farsi

In Farsi, the same words can be written in different shapes. Some variants are depicted in Fig. 11.2 and Fig. 11.3. These variants have been written by different writers and have been collected in our dataset.

Concentrating on these shape differences can help us to design a more accurate recognition system with better results.

11.2 Design Cycle

To design a handwriting recognition system with a high performance, we have to train it with various handwriting styles. Therefore, as the first step, we gathered a Farsi dataset which could be used as a reference point to evaluate the performance of not only a handwriting recognition system, but also of a word spotting system [3].

The next step was to design and implement the handwriting recognition system. The design cycle is shown in Fig. 11.4.

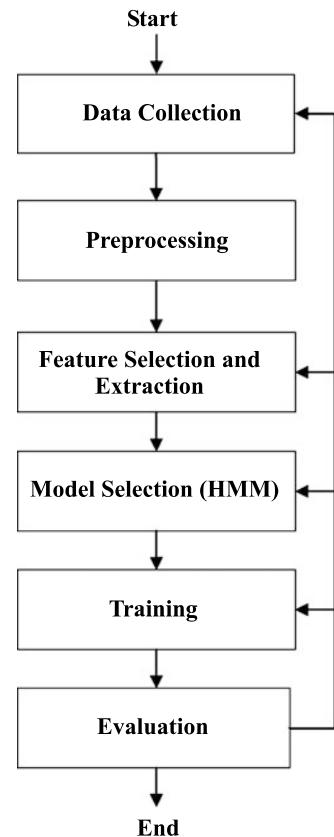
11.3 Dataset

To evaluate our recognition system, we collected a Farsi dataset which includes all types of Farsi scripts such as isolated characters, digits, numeral strings, special symbols, words, and texts [3]. The main purpose has been to design a dataset which not only facilitates the development and evaluation of the Farsi recognition systems but can also be used to compare the performance of different recognition systems. Figure 11.5 provides an overview of the whole dataset.

The Farsi words dataset consists of about 70 word classes which are officially used for measurement and counting purposes. These words include the measurement units of distance, volume, weight, currency and words which are usually used in documents and daily business activities. The categorized Farsi words are shown in Tables 11.2 and 11.3.

Each word class consists of approximately 516 images. In the verification and post-processing stages some images were deleted because of remaining noise that could mislead the classifier. Some Farsi handwritten words and their equivalents are shown in Table 11.4.

Fig. 11.4 Block diagram of the recognition system



11.4 Pre-processing

After data collection, we had to process the images to make them ready for the recognition stage. Therefore, this stage was called the word pre-processing stage. If the pre-processing stage was not completed successfully, then the raw data would not have a good quality. It could then mislead the classifiers, and the recognition system would fail. The other reason to consider this stage as one of the most important steps in the word recognition procedure is that it forms the foundation in designing real applications for the real world. To train our recognition system, we collected specific kinds of data to fulfill this purpose, in a controlled environment. In other words, the data was not gathered from real-world texts such as bank checks, envelopes, etc. Real-world texts are not as clean and neat as artificial ones. They may be written on folded and dirty pieces of paper, scripts can be broken, people may be careless in writing the scripts, etc. Therefore, a good recognition result comes from a good pre-processing which delivers clean data. Prior to the pre-processing stage we save the smallest rectangle containing the word image and remove the extra pixels around the box.

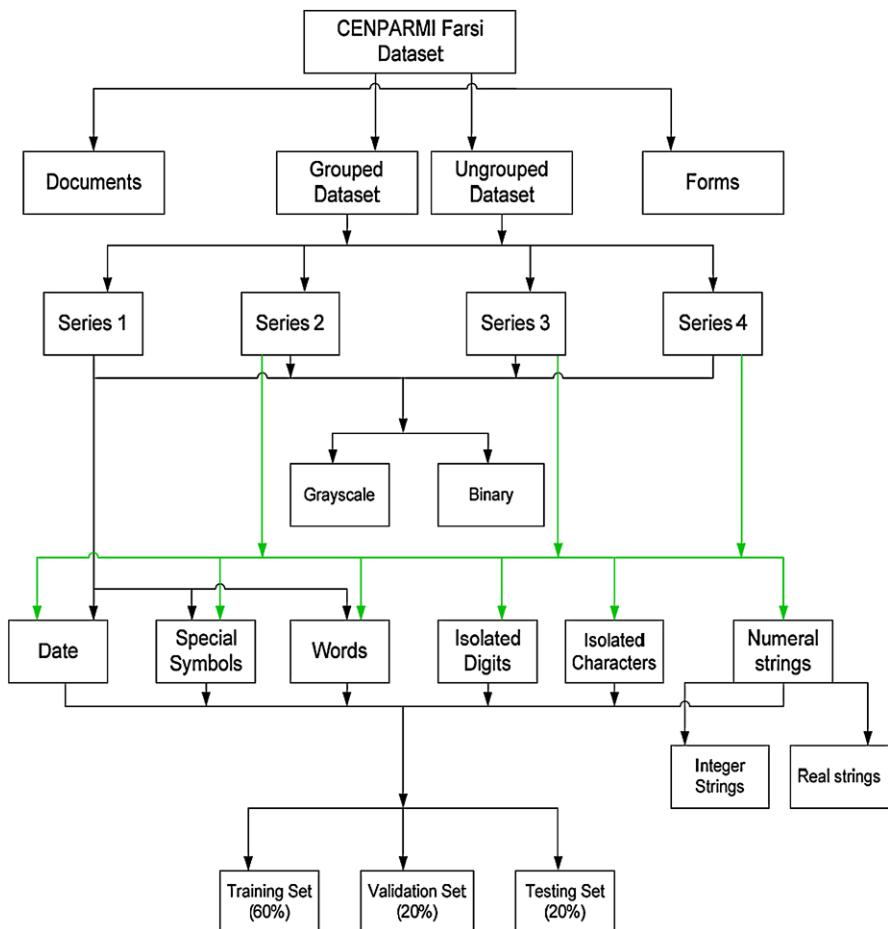


Fig. 11.5 CENPARMI Farsi dataset structure. Grouped data include training, validation, and testing subsets

This following sections will describe the pre-processing steps taken through our recognition path, i.e., image binarization, skeletonization, and dilation.

11.4.1 Image Binarization

Image binarization converts an image of 256 gray levels to a binary level, black and white image. The reasons we use a binary image instead of a gray image for word recognition processes are as follows:

1. Image binarization segments an image into foreground and background pixels.

Table 11.2 Categorized, selected Farsi words (Part 1)

Selected Words	Farsi Equivalent
Distance	
Centimeter	سانتی متر
Inches	اینچ
Meter	متر
Volume	
Milliliter	میلی لیتر
Liter	لیتر
Weight	
Milligram	میلی گرم
Gram	گرم
Kilogram	کیلو گرم
Counting	
One	پک
Two	دو
Three	سه
Four	چهار
Five	پنج
Six	شش
Seven	هفت
Eight	هشت
Nine	نه
Ten	ده
Other words	
Balance	مانده
Amount	مبلغ
Cash	نقد
Expense	هزینه
Credit	اعتبار
Delivery	تحویل
Due	سررسید
Custom	گمرک
Expire	انقضا
Interest	سود
Inventory	انبار
Issue	مورد
Period	مدت
Plus	اضافه

2. Each pixel value in a binary image is saved in a single bit instead of 8 bits for 256 gray levels, so the image will have a smaller size. The smaller size of the image will lead to less usage of the memory and processor.
3. To preprocess our image by removing the noise, and skeletonizing and dilating the image, it is much easier to deal with two values instead of 256.

Table 11.3 Categorized, selected Farsi words (Part 2)

Selected Words	Farsi Equivalent
Other words	
Price	قیمت
Loan	وام
Rent	اجاره
Stock	سهام
Tax	مالیات
Total	مجموع
Money Order	حواله
Volume	حجم
Document	سند
Weight	وزن
Width	عرض
Duty	عوارض
Carton	کارتون
Debit	بدھی
Number	شماره
Bill	صورتحساب
Account	حساب

Table 11.4 Selected Farsi handwritten words and their equivalents

Word in English	Farsi Printed word	Equivalent Extracted Handwritten Image
Amount	مبلغ	
Liter	لیتر	
Milligram	گرم	
Tons	تن	
Delivery	تحویل	

4. The computer is a binary system; therefore, designing a system based on this fact makes the system more compatible with the computer.

The bit value of zero is interpreted as black, while the bit value of one is interpreted as white.

To binarize an image, we consider either a single parameter known as the intensity threshold or multiple thresholds known as a band of intensity values. Then, each pixel in the image is compared with the threshold. If the pixel's intensity is higher than the threshold, the pixel is set to one; otherwise it is set to zero. To define the intensity threshold we have used Otsu's algorithm [4].

After we have binarized the images, we can start making them ready for the recognition process.

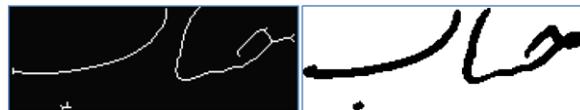


Fig. 11.6 Black background image at the *left* shows the skeleton of the white background image at the *right*. They both show the word Account in Farsi

11.4.2 Skeletonization

The main purpose of skeletonization is to extract a region-based shape feature of the general form of an object without having to change the structure of the object. To process the word images we used the skeleton of the words to avoid dealing with the unequal stroke widths.

The Zhang–Suen thinning algorithm was used to extract the skeleton of the images [5]. Figure 11.6 shows a word image and its skeleton.

11.4.3 Dilation

Dilation is one of the basic operations used in mathematical morphology. Suppose that X is the set of Euclidian coordinates corresponding to an input binary image, and that K is the set of coordinates for the structuring elements. Let K_x denote the translation of K so that its origin is at x . Then, the dilation of X by K is simply the set of all points x such that the intersection of K_x with X is not empty [7]. The dilation operation usually uses a structuring element for probing and expanding the boundary pixels of an image [6]. We dilate the skeletonized word image to make our system independent of stroke width. The dilation operation takes two pieces of data as inputs. The first is the image which is to be dilated. The second is a set of coordinate points known as a structuring element (kernel). This kernel determines the precise effect of the dilation of the input image. The following K matrix shows the structuring element, which is a 4×4 square:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (11.1)$$

Figure 11.7 shows a written word image from the Farsi dataset and its dilated image.

After gathering the dataset and pre-processing the data to enhance their qualities and make them ready for the recognition processes, we can start the recognition procedure. In the next section, we will describe feature extraction, which is the first step in the recognition procedure.



Fig. 11.7 Black background image at *left* shows the dilated image of the white background image at *right*. They both show the word Account in Farsi

11.5 Feature Selection and Extraction

For feature extraction, we first need to define the features. In the real world, we recognize everything by extracting its identifying features, and these can be different from one object to the other. In pattern recognition, choosing the features depends on the classifier which will be used to recognize them and the characteristics of the pattern. Selected features should be able to model the characteristics of the pattern.

Features can be extracted globally or locally from an image. In extracting the global features, we usually consider the word as a whole image, which is called the holistic approach. In the holistic approach, word segmentation is not necessary, whereas in analytical approaches, the word is segmented into smaller units. Therefore, the features are extracted locally from each small unit. Examples of local features include: percentage of foreground pixels within a window, foreground-background transition statistics, percentage of the foreground pixels in the core, and regions of ascenders and descenders. Global features can be structural or statistical features. Coefficients of the Fourier transform and invariant moments are considered as global statistical features [8].

As described earlier, Farsi has a cursive script (connected sequence of characters in a word), making the whole word a complex stroke, and it has numerous varieties in shape. Therefore, structural features identify the script characteristics better than statistical features. Statistical features (such as number of connected components, holes, ascenders and descenders) cannot tolerate a large degree of variability.

11.5.1 Feature Selection

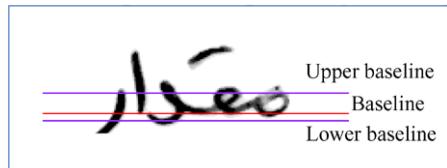
The classifier's accuracy strongly depends on the type of features we choose to recognize a word. To select the features, we had to consider the hidden Markov model (HMM) capabilities and the Farsi script characteristics. Experience has shown that HMM can make more accurate recognitions using the structural features.

In our recognition system, we used baseline dependent features to identify the words. Baseline dependent features emphasize the existence of descenders and ascenders. We used a sliding window to extract the image features locally. The region which is restricted in the sliding window is called the frame. The frame's height is equal to the word's height, and the width is twice the stroke width of the word.



Fig. 11.8 The word “Account” is shown in the figure. Red and green windows are consecutive and the 50 % overlap is shown

Fig. 11.9 Baselines are shown for the Amount word image



Before feature extraction, one of the fundamental morphological operations, dilation, is applied to the skeleton of the word image to make the word’s stroke widths equally four pixels wide to ensure proper contour generation. Therefore, the sliding window’s width was considered to be eight pixels to get a clearer shape of the image. In other words, each two consecutive windows have 50 % of overlap, as shown in Fig. 11.8.

The more identifying features we give to the classifier, the better recognition result we can obtain, unless we mislead the classifier by overtraining it. Therefore, to extract more features, we divided each window horizontally into 15 equal blocks. Finally, the local features were extracted.

Our selected features were language independent, and some of them were calculated with reference to the baselines (main baseline, upper and lower baselines) of the word. These baselines will be described in the next section.

11.5.2 Baseline Detection

Finding the baselines is a necessary step prior to doing some script processing tasks, such as skew corrections, segmentation, and feature extractions. We avoided skew correction in our pre-processing module because there was no or little skew detected in our word images. The correction caused distortion in our images, so we considered the skew as the variations in handwriting.

For each word, the baseline, upper baseline, and lower baseline were detected. The upper and lower baselines divided the word image into three zones. The restricted zone between the upper and lower baselines was called the core zone. The baselines for an image are shown in Fig. 11.9.

To extract the baselines we first scan the image from top to bottom, considering the image as a matrix of zeros and ones. The row with the most number of black

pixels is considered as the baseline. Scanning upward from the baseline to the top of the image, the row with the most number of black pixels is considered as the upper baseline. Scanning downward from the baseline, the row with the most number of black pixels is called the lower baseline.

The features we used to identify our words were baseline dependent, and they could be divided into distribution features and concavity features, which are described in the following sections. Twenty-five features were extracted for each frame.

11.5.3 Distribution Features

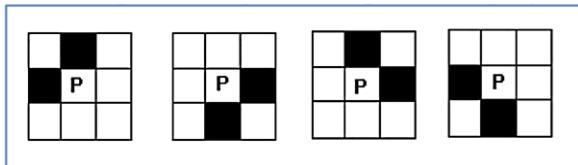
In our system, distribution features are calculated based on the foreground (white) pixel densities. It is easier to detect the distribution features than topological features, but they are less resistant to noise and local distortion. Since we removed noises in the pre-processing stage and we avoided some pre-processing which may have caused distortion such as skew correction, we mostly used the distribution features to recognize the word images. Our approach is based on the algorithm described in [9] with few alterations.

Our feature vector includes 17 distribution features per frame, which are described below:

- F1 Density level of the block. Let b be the density level of the block. Then $b = 0$ if the number of foreground pixels in the current cell is zero, else $b = 1$.
- F2 Density of foreground pixels, in other words, the sum of foreground pixels for each row of the block.
- F3 Number of transitions between two consecutive blocks of different density levels.
- F4 Derivative feature between the current frame and the previous one which shows the difference between the y position of gravity centers of the current frame and the previous one.
- F5–F12 Eight features that represent the number of white pixels in each column of the current frame.
- F13 Normalized position of the center of gravity of the foreground pixels in the current frame with respect to the lower baseline. F13 is calculated as follows:

$$F13 = \frac{g-1}{h}$$
- F14 Density of foreground pixels over the baseline in the current frame.
- F15 Density of foreground pixels under the baseline in the current frame.
- F16 Number of transitions between two consecutive blocks of different density levels above the lower baseline.
- F17 This feature represents the zone that includes the center of gravity. If the center of gravity is above the upper baseline $F17 = 1$, if it is between the upper and lower baselines $F17 = 2$, and if it is under the lower baseline $F17 = 3$.

Fig. 11.10 Four templates to show the concavity of a background pixel



11.5.4 Concavity Features

To gather some information about the local concavity and to identify the stroke direction in each frame, we calculated the local concavity features. They were extracted by using a 3×3 window. Our approach is based on the algorithm described in [9]. We considered each 3×3 subset of pixels by centering each subset with a background pixel (which is a black pixel in our images), and tried to match the subset with one of the four templates shown in Fig. 11.10. These four templates are the four possible types of concavity configurations for a background pixel.

Then we kept track of the number of background pixels which matched one of the above templates. Since we had different heights for each image, we normalized the heights by dividing them by the height of the image.

Therefore, F18–F21 were calculated as follows: N_{lu} stands for the number of background pixels which are surrounded by a white pixel at the left and up positions. N_{ur} stands for the number of background pixels which are surrounded by a white pixel at the up and right positions, N_{rd} stands for the number of background pixels which are surrounded by a white pixel at the right and down positions, and N_{dl} stands for the number of background pixels which are surrounded by a white pixel at the down and left positions.

$$F18 = \frac{N_{lu}}{H}; \quad F19 = \frac{N_{ur}}{H}; \quad F20 = \frac{N_{rd}}{H}; \quad F21 = \frac{N_{dl}}{H} \quad (11.2)$$

We also calculated these features for the core zone of the image. To normalize these values, we divided them by the height d of the core zone as follows:

$$d = \text{UpperBaseline_YCoordinate} - \text{LowerBaseline_YCoordinate} \quad (11.3)$$

$$F22 = \frac{CN_{dl}}{d}; \quad F23 = \frac{CN_{rd}}{d}; \quad F24 = \frac{CN_{lu}}{d}; \quad F25 = \frac{CN_{ur}}{d} \quad (11.4)$$

Because of the connected nature of the Farsi script, HMM is an acceptable classifier to recognize the Farsi words. HMM systems can stochastically model sequences of variable lengths which occur very frequently in Farsi handwritten words. HMM can also cope with nonlinear distortions along one direction [8]. We chose a discrete HMM to limit the number of observation symbols. Therefore, the features should be quantized to a codebook vector. To quantize the feature vector to a codebook vector, we used the K-means clustering algorithm, which is described in the following section.

11.6 Clustering

Clustering is used to group data which have similar characteristics. The purpose is to retrieve the relevant information more quickly. The difference between clustering and classification is that, in classification, we assign data to predefined classes, while in clustering, clusters are created during the assignment [10]. In this study the K-means algorithm is used.

11.6.1 K-means Clustering Algorithm

The K-means clustering algorithm can cluster n observations into k mutually exclusive clusters. Each cluster is represented by a vector which is the mean of the existing data in the cluster. Therefore, the data will finally be assigned to the cluster with the nearest mean. The purpose of the K-means algorithm is to minimize the squared Euclidean distances between the data in each cluster [11]. The K-means is calculated as follows:

$$\sum_{i=1}^k \sum_{d_j \in S_i} \|d_j - \mu_i\|^2 \quad (11.5)$$

where $\{d_1, d_2, \dots, d_n\}$ is the set of data to be clustered, k is the number of clusters, and μ_i is the mean of data in each set S_i .

The K-means algorithm is a heuristic algorithm. Therefore, the result depends strongly on the initial clusters, and there is no guarantee of achieving the global optimum. The algorithm can be simply described as follows:

1. k initial means are randomly selected from the data.
2. The data with the nearest mean to the initial k randomly selected data will be assigned to the k -th cluster.
3. The centroid of each cluster becomes the new mean.
4. Steps 2 and 3 are repeated until the sum of distances from each object to its cluster centroid cannot be decreased further [13].

11.6.2 Optimum Number of Clusters (k)

In the K-means algorithm, k is the number of clusters that is predefined to the algorithm and is considered as an input argument. Choosing an inappropriate value for k may lead to a bad recognition result. The proper choice of k is difficult and depends on the shape and scale of the distribution of points in a dataset and the desired clustering resolution [14].

Choosing k Using Silhouette Plot

As described, a way to find the proper value of k is to analyze the existing clusters. You can use a silhouette plot function to see how well the data is separated into different clusters. There is a measure which ranges from -1 to $+1$ and indicates if a data point is very distant from the cluster that it does not belong to or is very close to it. Positive 1 shows a data point that is very close to another cluster and is probably wrongly clustered. The larger the quantity of data, the more time consuming and complex the calculation will be when plotting the K-means silhouette. Therefore, the silhouette plot is not a practical way of optimizing the number of clusters.

Rule of Thumb to Find k

Another way to calculate the optimum number of clusters is through the following formula [15]:

$$k \approx \left(\frac{n}{2}\right)^{\frac{1}{2}} \quad (11.6)$$

where n is the number of data points. For instance, our feature vector's size is $716,596 \times 25$ (the total number of blocks times the total number of features). Therefore, the number of data points is 1,791,490,012 and k is approximately 2993.

The recognition process starts with model selection, which will be described in the following sections.

11.7 Model Selection and Hidden Markov Models

The model selection depends on the characteristics of the problem. A model is selected to predict output from input, which can be parametric or nonparametric. There are different types of models which can perform the recognition, such as linear models, classification and regression trees, neural networks, kernels, and hybrid methods. The selection of the optimal model is difficult. The selected method should perform best on unseen (test) data.

Among all of the classification models, hidden Markov models (HMMs) were chosen because of the connected nature of the Farsi script. HMM systems stochastically model sequences of variable lengths and cope with nonlinear distortions along one direction. As described previously, the discrete HMM was selected to limit the number of observation symbols [8]. The HMM has particular advantages when compared to the other models, such as embedded training, i.e., automatic training of character models on non-segmented words [16].

In this section, we will describe HMM concepts, as well as our HMM and its initial estimation. We will also discuss the practical issues related to the implementation and optimization of HMMs.

11.7.1 Markov Systems

A Markov system is a chain that has different states with stochastic identities. In this chain, we have states at time t that are influenced by the states at time $t - 1$. HMMs are the best solution to these kinds of problems in speech recognition, handwriting recognition, gesture recognition, bioinformatics, and financial analysis, etc. In a Markov model, all the states are visible; therefore, the state transition probabilities are the only parameters [17].

11.7.2 Hidden States

In HMMs, the states are hidden to the observer, but the outputs are clear. We call this kind of Markov model a hidden one because the sequence which leads to a specific output is hidden, even if the parameters are all precisely identified.

11.7.3 HMM Notation

An HMM has three parameters, which are shown as follows:

$$\lambda = (\pi, A, B) \quad (11.7)$$

where λ is an HMM which is defined by π , A and B . π is the initial distribution of the states. A is the state transition matrix, and B is the confusion matrix.

We consider N as the number of states in a model and M as the number of distinct observation symbols per state, which in a discrete HMM is the number of clusters. In other words, M is the number of alphabets. We also have to determine the topology of our model.

We calculate the state transition matrix as follows:

$$A = \{a_{ij}\} \quad (11.8)$$

$$a_{ij} = P[q(t+1) = S_j | q_t = S_i]; \quad 1 \leq i, j \leq N \quad (11.9)$$

We denote $S = \{S_1, S_2, S_3, \dots, S_n\}$ which represents the states, and the state at time t is q_t . The topology of the HMM shows the states which can be reached through each state. For instance, in our model we considered that each state has a self-transition or a transition to the next state or the next two states, which is shown in Fig. 11.11. If any state can reach all the other states, then we have $a_{ij} > 0$ for all states [17].

The observation for a symbol probability distribution is called the confusion matrix, which is shown by B . It is calculated as follows:

$$B = \{b_j(k)\} \quad (11.10)$$

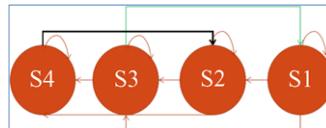


Fig. 11.11 A hidden Markov model with four states. Each state has a self-transition, a transition to the next state, and another transition to the next two states

$$b_j(k) = P[V_k \text{ at } t \mid q_t = S_j]; \quad 1 \leq j \leq N \text{ and } 1 \leq k \leq N \quad (11.11)$$

The confusion matrix shows the probability of emission for symbol k at state j . Another parameter which should be defined to complete the model is the vector of the initial state probabilities. It is calculated as follows:

$$\pi = \{\pi_i\} \quad (11.12)$$

$$\pi_i = P[q_1 = S_i]; \quad 1 \leq i \leq N \quad (11.13)$$

where π is a vector which shows the probability of being in each state at time $t = 1$.

11.7.4 Discrete or Continuous HMM

In speech recognition, a continuous HMM is more acceptable, while in handwriting recognition, it has always been a challenge to select between the continuous and discrete HMMs. In 1996, a study was conducted to show a comparison between the continuous and discrete HMM for cursive handwriting recognition [18]. The research showed that the discrete HMM leads to a better result in handwriting recognition.

In a discrete hidden Markov model (DHMM), the output of the process is observed as a sequence of observations which belong to a finite alphabet. These observations represent the indices of a codebook. The codebook is calculated by a vector quantization method as per our description in the previous section. While calculating the quantized vector, some data will be lost due to the quantization error; this is called distortion. In other words, DHMM quantizes the data to a limited alphabet, which causes a loss of information.

By using a continuous HMM (CHMM) we can eliminate the distortion problem, but CHMM has more parameters and requires more memory [8].

Therefore, we chose DHMM to design and implement our handwriting word recognition system.

11.7.5 Three Main Problems in HMM: Evaluation, Decoding, and Learning

For the HMM to be useful in real applications, three main problems should be solved.

Evaluation Suppose we have different HMMs, each with a set of triple $\lambda = (\pi, A, B)$ and data as a sequence of observations. The problem is to find the best model which can generate the data. The forward algorithm can solve this problem by calculating the probability of an observation sequence given an HMM model. This problem usually needs to be solved in script recognition or speech recognition processes, when we have different models and we want to match a testing script or spoken word with the existing models.

Decoding The problem is to find the hidden states that lead to the sequence of observations. The Viterbi algorithm is usually used to solve this problem. There is no correct sequence to be decoded. Therefore, we use the optimal criteria to solve the problem [17]. This problem needs to be solved widely in natural language processing (NLP), where we need to tag words with their syntactic classes as nouns, verbs, etc. Thus we consider the words in a sentence as observations and the syntactic classes as hidden states. The purpose is to find the best syntactic class for a word, given the context [19].

Learning In this problem, we have the observation sequence, we know the hidden states that have led to the observations, and we try to find the best (most probable) HMM (π, A, B) that describes the observed sequence. The forward-backward algorithm is usually used to solve this problem.

In script recognition, we use the solution to problem 3 (learning) to model the script, the solution to problem 2 (decoding) to improve the model, and the solution to problem 1 (evaluation) to find the best matched script for the given test data.

11.7.6 Our Hidden Markov Model and Initial Estimation

It is very important to keep the original size of the image when we model and design our recognition system with a hidden Markov chain, because the number of sliding windows which should cover the whole image is an identifying feature which can be used to differentiate some of the words.

One of the most important parameters which should be defined for an HMM is the number of states. As described in [20], to calculate the number of states for our model, we first calculated the least number of sliding windows per class. Then, we chose the smallest value as the number of states for the classifier.

For the topology of our model, we considered a right-to-left HMM (RTL HMM) to fit the Farsi scripts characteristic, and each state could have a self-transition, or a

transition to the next or two next states, as was shown in Fig. 11.11 for a four-state model.

To start working with the HMM, we also had to define some initial values for A , B and π .

We initially considered equal transition probabilities for the states. The following image shows a transition matrix for the model that is illustrated in Fig. 11.11.

	S_1	S_2	S_3	S_4
S_1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0
S_2	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
S_3	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$
S_4	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$

This transition matrix is an $(N \times N)$ size matrix, where N is the number of states.

To calculate the initial values for the confusion matrix, we considered an equal emission probability for all symbols. Therefore, if M shows the number of symbols, the probability to emit each symbol at each state is $\frac{1}{M}$. For instance, for an HMM model with four states and three symbols we get the following confusion matrix:

	O_1	O_2	O_3
S_1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
S_2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
S_3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
S_4	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

where O shows the observation or symbol.

Finally, to calculate π , the random values are chosen for the probability of the initial states. Then, we normalize the values to make the entries of the array add up to 1.

11.7.7 Training the HMM Word Recognition System

There are different ways to train an HMM. A detailed description can be found in [21]. We chose the maximum likelihood (ML) criterion to train the models. In this criterion, during the training stage the HMM parameters are first initialized and then iteratively re-estimated such that the likelihood of the model produced by the training sequences increases. In our recognition system, we first defined the initial estimation as per our description in Sect. 11.7.6. The training process stops when the likelihood reaches a maximum value. Maximum mutual information (MMI) and minimum discrimination information (MDI) are alternative HMM training criteria [21].

We used the Baum–Welch (BW) algorithm to train word class models. The first step is to calculate $P(O|\lambda)$, which is the probability of the observation sequence O , given the model λ .

We used the forward algorithm to calculate $P(O|\lambda)$. The forward variable, $\alpha_t(i)$, is defined as follows:

$$\alpha_t(i) = P(O_1, O_2, O_3, \dots, O_t, q_t = S_i | \lambda) \quad (11.14)$$

which is the probability of the partial observation sequence $O_1, O_2, O_3, \dots, O_t$, and state S_i at time t , given the model λ . The algorithm for inducting $\alpha_t(i)$ is described below.

Initialization:

$$\alpha_t(i) = \pi_i b_i(O_t), \quad 1 \leq i \leq N \quad (11.15)$$

Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \leq t \leq T-1, \quad 1 \leq j \leq N \quad (11.16)$$

Termination:

$$P(O|\lambda) = \sum_{i=1}^N \alpha_T(i) \quad (11.17)$$

The purpose of the BW algorithm is to adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$. This is the most difficult problem in the HMM domain. The BW is an iterative algorithm based on the forward and backward algorithms. The backward variable $\beta_t(i)$ is defined as:

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, O_{t+3}, \dots, O_T, q_t = S_i, \lambda) \quad (11.18)$$

which is the probability of the observation sequence from $t+1$ to the end, at state S_i at time t given the model λ . The algorithm for inducting $\alpha_t(i)$ is described below.

Initialization:

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \quad (11.19)$$

Induction:

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad t = T-1, T-2, \dots, 1, \quad 1 \leq i \leq N \quad (11.20)$$

Now we can define the probability of being in state S_i at time t , and state S_j at time $t+1$, given the model and the observation sequence as follows:

$$\xi(i, j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \quad (11.21)$$

$$\xi(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \quad (11.22)$$

$\gamma_t(i)$, the probability of being in state S_i at time t given the observation sequence O and model λ , is calculated as follows.

$$\gamma_t(i) = P(q_t = S_i | O, \lambda) \quad (11.23)$$

or it can be calculated, using the forward and backward variables, as:

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{P(O|\lambda)} \quad (11.24)$$

The expected number of transitions from S_i is denoted by

$$\sum_{t=1}^{T-1} \gamma_t(i) \quad (11.25)$$

and the expected number of transitions from S_i to S_j is denoted by

$$\sum_{t=1}^{T-1} \xi_t(i, j) \quad (11.26)$$

To re-estimate the parameters π , A and B of an HMM, we can use the following formulas:

1. Expected frequency in state S_i at time $t = 1$.

$$\bar{\pi}_i = \gamma_1(i)$$

2. Transition coefficient = expected number of transitions from state S_i to S_j , divided by the expected number of transitions from state S_i .

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

3. Observation symbol probability = expected number of times in state j , while observing symbol v_k , divided by the expected number of times in state j .

$$\bar{b}_j(k) = \frac{\sum_{t=1, S.t.O_t=v_k}^T \gamma_t(i)}{\sum_{t=1}^T \gamma_t(j)}$$

11.7.8 Testing the HMM Word Recognition System

In the testing stage our problem is to find the best model that can generate the data. Viterbi's algorithm is able to match a single model to an observed sequence of symbols.

Consider the following variables:

- $\delta_t(i)$: Scores the likelihood of the observation sequence $O_1, O_2, O_3, \dots, O_t$ having been produced by the most likely sequence of model states, which ends at state i at time t .
- $\psi_t(i)$: The array used to trace the ML path which keeps a record of the states which maximized the likelihood from time 1 to t .

Viterbi's algorithm is described as follows [21].

Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \quad (11.27)$$

$$\psi_1(i) = 0 \quad (11.28)$$

Recursion:

$$\delta_t(i) = \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ij}] b_j(O_t), \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \quad (11.29)$$

$$\psi_t(i) = \arg \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ij}], \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \quad (11.30)$$

Termination:

$$P^* = \max_{1 < i < N} [\delta_T(i)] \quad (11.31)$$

$$q_T^* = \arg \max_{1 < i < N} [\delta_T(i)] \quad (11.32)$$

Backtracking for state sequence:

$$q_T^* = \psi_{T+1}(q_{T+1}^*), \quad t = T - 1, T - 2, \dots, 1 \quad (11.33)$$

Here P^* is the probability of the sequence being produced by each model. The model that has the greatest likelihood of producing this observation sequence defines the word class.

11.8 Results

As we have described in detail, our experiments were conducted on the CENPARMI Farsi dataset with a lexicon size of about 70 frequently used words in Farsi documents. A holistic approach was chosen to model each word as a hidden Markov model (HMM). In the literature, words in small size lexicons were modeled separately, while for the large size lexicons the path discriminant method could be used to reduce the memory usage and process time. A right-to-left HMM was designed to consider the nature of the Farsi script. The numbers of states were chosen based on the least number of sliding windows per class.

The number of iterations to train each model was ten, and the training results improved steadily using the Baum–Welch algorithm. In our research we tried to show the importance of the baseline-related features. Successful experiments with high recognition rates showed that the distribution and concavity features could improve the system performance.

Finally, encouraging recognition rates of 98.76 % and 96.02 % have been obtained for the training and testing sets, respectively. Our designed system is reliable, robust to noise, and reasonably fast.

References

1. Nicholas, S.W.: Indo-Iranian Languages and Peoples. Oxford University Press, Oxford (2002)
2. Clawson, P.: Eternal Iran. Palgrave Macmillan, New York (2004)
3. Haghighi, P.J., Nobile, N., He, C.L., Suen, C.Y.: A new large-scale multi-purpose handwritten Farsi database. In: Proc. International Conference on Image Analysis and Recognition, Halifax, NS, Canada, pp. 278–286 (2009)
4. Otsu, N.: A threshold selection method from gray-scaled histogram. IEEE Trans. Syst. Man Cybern. **8**, 62–66 (1978)
5. Zhang, T.Y., Suen, C.Y.: A fast parallel algorithm for thinning digital patterns. In: Communications of the ACM, vol. 2, pp. 236–239 (1984)
6. Dougherty, E.R.: An Introduction to Morphological Image Processing. SPIE Optical Engineering Press, Washington (1992)
7. Gonzales, R., Woods, R.: Digital Image Processing. Prentice Hall, Upper Saddle River (2008)
8. Haji, M.: Farsi handwritten word recognition using continuous hidden Markov models and structural features. M.S. thesis, Shiraz University, Iran (2005)
9. El-Hajj, R.: Arabic handwriting recognition using baseline dependant features and hidden Markov models. In: Proc. 8th International Conference on Document Analysis and Recognition (ICDAR), Seoul, Korea, pp. 893–897 (2005)
10. Ward, J.H.: Hierarchical grouping to optimize an objective function. J. Am. Stat. Assoc. **58**, 236–244 (1963)
11. MacKay, D.: Information Theory, Inference and Learning Algorithms. Cambridge University Press, New York (2003)
12. Blumenstein, M., Cheng, C.K., Liu, X.Y.: New pre-processing techniques for handwritten word recognition. In: Proc. 2nd IASTED Conference on Visualization, Imaging and Image Processing, Marbella, Spain, pp. 480–484 (2002)
13. Matlab Image Processing Toolbox. http://www.mathworks.com/access/helpdesk/help/pdf_doc/images/images_tb.pdf
14. Ravichandra, R.: Data mining and clustering techniques. In: Proc. Documentation Research and Training Centre (DRTC) Workshop on Semantic Web. DRTC, Bangalore (2003)
15. Ketchen, D.J., Shook, C.L.: The application of cluster analysis in strategic management research: an analysis and critique. Strateg. Manag. J. **17**, 441–458 (1996)
16. Kessentini, Y., Paquet, T., Benhamadou, A.: A multi-stream HMM-based approach for offline multi-script handwritten word recognition. In: Proc. 11th International Conference on Frontiers in Handwriting Recognition (ICFHR 11), Montreal, Canada, pp. 147–152 (2008)
17. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**, 257–286 (1989)
18. Rigoll, G., Kosmala, A., Rottland, J., Neukirchen, C.: A comparison between continuous and discrete density hidden Markov models for cursive handwriting recognition. In: Proc. 13th International Conference on Pattern Recognition (ICPR'96), Vienna, Austria, pp. 39–48 (1996)

19. Boyle, R.: Hidden Markov models. http://www.comp.leeds.ac.uk/HiddenMarkovModels/html_dev/main.html
20. Murphy, K.: How to use HMM Toolbox (1998). http://people.cs.ubc.ca/~murphyk/Software/HMM/hmm_usage.html
21. Britto, A. Jr.: A two-stage HMM-based method for recognizing handwritten numeral strings. Ph.D. dissertation, Pontifical Catholic University of Paraná, Curitiba, Brazil (2001)

Chapter 12

Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks

Alex Graves

Abstract Offline handwriting recognition requires a combination of computer vision and sequence learning. In most systems the two elements are handled separately, with sophisticated pre-processing techniques used to extract the image features and sequential models such as HMMs used to provide the transcriptions. This chapter considers an alternative system, based on multidimensional recurrent neural networks, that learns directly from pixel data, and describes its winning entry to a major Arabic offline handwriting recognition competition.

12.1 Introduction

Offline handwriting recognition is usually performed by first extracting a sequence of features from the image, then using either a hidden Markov model (HMM) [9] or an HMM/neural network hybrid [10] to transcribe the features.

However, a system trained directly on pixel data has several potential advantages. One is that defining input features suitable for an HMM requires considerable time and expertise. Furthermore, the features must be redesigned for every different alphabet. In contrast, a system trained on raw images can be applied with equal ease to, for example, Arabic and English. Another potential benefit is that using raw data allows the visual and sequential aspects of handwriting recognition to be learned together, rather than treated as two separate problems. This kind of ‘end-to-end’ training is often beneficial for machine learning algorithms, since it allows them more freedom to adapt to the task [13].

Furthermore, recent results suggest that recurrent neural networks (RNNs) may be preferable to HMMs for sequence labelling tasks such as speech [5] and online handwriting recognition [6]. One possible reason for this is that RNNs are trained discriminatively, whereas HMMs are generative. Although generative approaches offer more insight into the data, discriminative methods tend to perform better at tasks such as classification and labelling, at least when large amounts of data are

A. Graves (✉)
Technical University of Munich, Munich, Germany
e-mail: graves@in.tum.de

available [15]. Indeed, much work has been done in recent years to introduce discriminative training to HMMs [11]. Another important difference is that RNNs, unlike HMMs, do not assume successive data points to be conditionally independent given some discrete internal state, which is often unrealistic for cursive handwriting.

This chapter will describe an offline handwriting recognition system based on recurrent neural networks. The system is trained directly on raw images, with no manual feature extraction. It won several prizes at the 2009 International Conference on Document Analysis and Recognition, including first place in the offline Arabic handwriting recognition competition [14].

The system was an extended version of a method used for online handwriting recognition from raw pen trajectories [6]. The *long short-term memory* (LSTM) network architecture [3, 8] was chosen for its ability to access long-range context, and the *connectionist temporal classification* [5] output layer allowed the network to transcribe the data with no prior segmentation.

Applying RNNs to offline handwriting is more challenging, since the input is no longer one dimensional. A naive approach would be to present the images to the network one vertical line at a time, thereby transforming them into 1D sequences. However, such a system would be unable to handle distortions along the vertical axis; for example, the same image shifted up by one pixel would appear completely different. A more robust method is offered by *multidimensional recurrent neural networks* (MDRNNs) [7]. MDRNNs, which are a special case of directed acyclic graph networks [1], generalise standard RNNs by providing recurrent connections along all spatio-temporal dimensions present in the data. These connections make MDRNNs robust to local distortions along any combination of input dimensions (e.g. image rotations and shears, which mix vertical and horizontal displacements) and allow them to model multidimensional context in a flexible way. We use multidimensional LSTM [7] because it is able to access long-range context along both input directions.

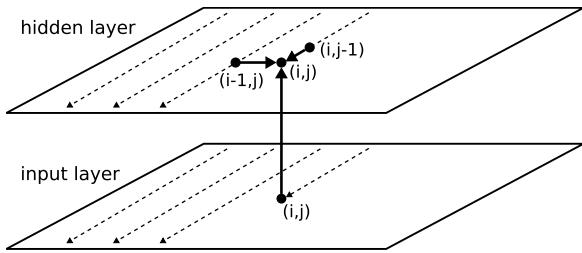
The problem remains, however, of how to transform 2D images into 1D label sequences. The solution presented here is to pass the data through a hierarchy of MDRNN layers, with subsampling windows applied after each level. The heights of the windows are chosen to incrementally collapse the 2D images onto 1D sequences, which can then be labelled by the output layer. Hierarchical structures are common in computer vision [17], because they allow complex features to be built up in stages. In particular our multilayered structure is similar to that used by convolutional networks [12], although it should be noted that because convolutional networks are not recurrent, they are difficult to apply to unsegmented cursive handwriting recognition.

The system is described in Sect. 12.2, experimental results are given in Sect. 12.3, and conclusions and directions for future work are given in Sect. 12.4.

12.2 Method

The three components of the recognition system are: (1) multidimensional recurrent neural networks, and multidimensional LSTM in particular; (2) the connectionist

Fig. 12.1 Two-dimensional RNN. The thick lines show connections to the current point (i, j) . The connections within the hidden layer plane are recurrent. The dashed lines show the scanning strips along which previous points were visited, starting at the top left corner



temporal classification output layer; and (3) the hierarchical structure. In what follows we describe each component in turn, then show how they fit together to form a complete system. For a more detailed description of (1) and (2) we refer the reader to [4].

12.2.1 Multidimensional Recurrent Neural Networks

The basic idea of multidimensional recurrent neural networks (MDRNNs) [7] is to replace the single recurrent connection found in standard recurrent networks with as many connections as there are spatio-temporal dimensions in the data. These connections allow the network to create a flexible internal representation of surrounding context, which is robust to localised distortions.

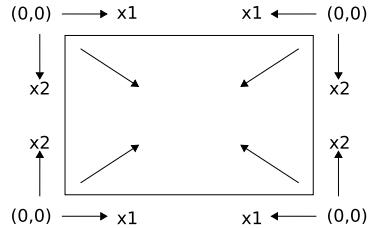
An MDRNN hidden layer scans through the input in 1D strips, storing its activations in a buffer. The strips are ordered in such a way that at every point the layer has already visited the points one step back along every dimension. The hidden activations at these previous points are fed to the current point through recurrent connections, along with the input. The 2D case is illustrated in Fig. 12.1.

One such layer is sufficient to give the network access to all context against the direction of scanning from the current point (e.g. to the top and left of (i, j) in Fig. 12.1). However we usually want surrounding context in all directions. The same problem exists in 1D networks, where it is often useful to have information about the future as well as the past. The canonical 1D solution is to use *bidirectional recurrent neural networks* [18], where two separate hidden layers scan through the input forwards and backwards. The generalisation of bidirectional networks to n dimensions (*multidirectional networks*) requires 2^n hidden layers, starting in every corner of the n -dimensional hypercube and scanning in opposite directions. The 2D case is shown in Fig. 12.2. All the hidden layers are connected to a single output layer, which therefore receives context information from all directions.

The error gradient of an MDRNN can be calculated with an n -dimensional extension of backpropagation through time. As in the 1D case, the data is processed in the reverse order of the forward pass, with each hidden layer receiving both the output derivatives and its own n ‘future’ derivatives at every timestep.

Let $a_j^{\mathbf{p}}$ and $b_j^{\mathbf{p}}$ be respectively the input and activation of unit j at point $\mathbf{p} = (p_1, \dots, p_n)$ in an n -dimensional input sequence \mathbf{x} with dimensions (D_1, \dots, D_n) .

Fig. 12.2 Axes used by the four hidden layers in a multidirectional 2D RNN. The arrows inside the rectangle indicate the direction of propagation during the forward pass



Let $\mathbf{p}_d^- = (p_1, \dots, p_d - 1, \dots, p_n)$ and $\mathbf{p}_d^+ = (p_1, \dots, p_d + 1, \dots, p_n)$. Let w_{ij} and w_{ij}^d be respectively the weight of the feedforward connection from unit i to unit j and the recurrent connection from i to j along dimension d . Let θ_h be the activation function of hidden unit h , and for some unit j and some differentiable objective function O let $\delta_j^{\mathbf{p}} = \frac{\partial O}{\partial a_j^{\mathbf{p}}}$. Then the forward and backward equations for an n -dimensional MDRNN with I input units, K output units, and H hidden summation units are as follows:

Forward Pass

$$a_h^{\mathbf{p}} = \sum_{i=1}^I x_i^{\mathbf{p}} w_{ih} + \sum_{\substack{d=1: \\ p_d > 0}}^n \sum_{\hat{h}=1}^H b_{\hat{h}}^{\mathbf{p}^-} w_{\hat{h}h}^d$$

$$b_h^{\mathbf{p}} = \theta_h(a_h^{\mathbf{p}})$$

Backward Pass

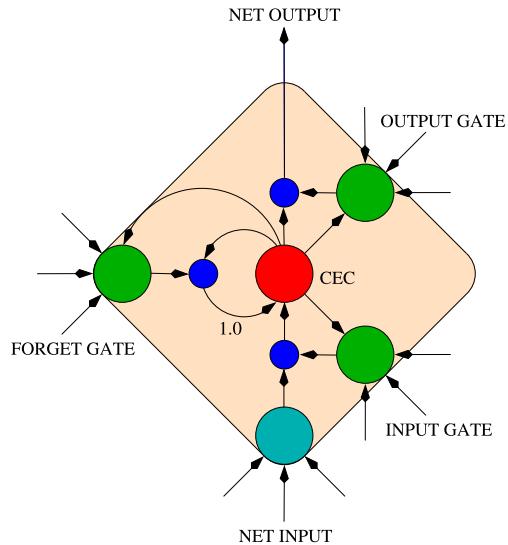
$$\delta_h^{\mathbf{p}} = \theta'_h(a_h^{\mathbf{p}}) \left(\sum_{k=1}^K \delta_k^{\mathbf{p}} w_{hk} + \sum_{\substack{d=1: \\ p_d < D_d - 1}}^n \sum_{\hat{h}=1}^H \delta_{\hat{h}}^{\mathbf{p}^+} w_{\hat{h}h}^d \right)$$

Multidimensional LSTM

Long short-term memory (LSTM) [3, 8] is an RNN architecture designed for data with long-range interdependencies. An LSTM layer consists of recurrently connected ‘memory cells’, whose activations are controlled by three multiplicative gate units: the input gate, forget gate and output gate. The gates allows the cells to store and retrieve information over time, giving them access to long-range context. An illustration of an LSTM memory cell is shown in Fig. 12.3.

The standard formulation of LSTM is explicitly one dimensional, since each cell contains a single recurrent connection, whose activation is controlled by a single forget gate. However, we can extend this to n dimensions by using instead n recurrent connections (one for each of the cell’s previous states along every dimension) with n forget gates.

Fig. 12.3 LSTM memory cell. The internal state of the cell is maintained with a recurrent connection of fixed weight 1.0. The three gates collect activations from inside and outside the block, and control the cell via multiplicative units (small circles). The input and output gates scale the input and output of the cell, while the forget gate scales the internal state



Consider a multidimensional LSTM (MDLSTM) memory cell in a hidden layer of H cells, connected to I input units and K output units. The subscripts c , ι , ϕ and ω refer to the cell, input gate, forget gate and output gate respectively. b_h^p is the output of cell h in the hidden layer at point p in the input sequence, and s_c^p is the state of cell c at p . f_1 is the activation function of the gates, and f_2 and f_3 are respectively the cell input and output activation functions. The suffix ϕ, d denotes the forget gate corresponding to recurrent connection d . The input gate ι is connected to previous cell c along all dimensions with the same weight (w_{ci}), whereas the forget gates are connected to cell c with a separate weight $w_{c(\phi,d)}$ for each dimension d . Then the forward and backward pass are as follows:

Forward Pass

Input Gate:

$$b_i^p = f_1 \left(\sum_{i=1}^I x_i^p w_{ii} + \sum_{\substack{d=1: \\ p_d > 0}}^n \left(w_{ci} s_c^{p_d^-} + \sum_{h=1}^H b_h^{p_d^-} w_{hi}^d \right) \right)$$

Forget Gate:

$$b_{\phi,d}^p = f_1 \left(\sum_{i=1}^I x_i^p w_{i(\phi,d)} + \sum_{\substack{d'=1: \\ p_{d'} > 0}}^n \sum_{h=1}^H b_h^{p_{d'}^-} w_{h(\phi,d)}^{d'} + \begin{cases} w_{c(\phi,d)} s_c^{p_d^-} & \text{if } p_d > 0 \\ 0 & \text{otherwise} \end{cases} \right)$$

Cell:

$$a_c^{\mathbf{p}} = \sum_{i=1}^I x_i^{\mathbf{p}} w_{ic} + \sum_{\substack{d=1: \\ p_d > 0}}^n \sum_{h=1}^H b_h^{\mathbf{p}_d^-} w_{hc}^d$$

State:

$$s_c^{\mathbf{p}} = b_t^{\mathbf{p}} f_2(a_c^{\mathbf{p}}) + \sum_{\substack{d=1: \\ p_d > 0}}^n s_c^{\mathbf{p}_d^-} b_{\phi,d}^{\mathbf{p}}$$

Output Gate:

$$b_{\omega}^{\mathbf{p}} = f_1 \left(\sum_{i=1}^I x_i^{\mathbf{p}} w_{i\omega} + \sum_{\substack{d=1: \\ p_d > 0}}^n \sum_{h=1}^H b_h^{\mathbf{p}_d^-} w_{h\omega}^d + w_{c\omega} s_c^{\mathbf{p}} \right)$$

Cell Output:

$$b_c^{\mathbf{p}} = b_{\omega}^{\mathbf{p}} f_3(s_c^{\mathbf{p}})$$

Backward Pass

Cell Output:

$$\varepsilon_c^{\mathbf{p}} \stackrel{\text{def}}{=} \frac{\partial O}{\partial b_c^{\mathbf{p}}} = \sum_{k=1}^K \delta_k^{\mathbf{p}} w_{ck} + \sum_{\substack{d=1: \\ p_d < D_d - 1}}^n \sum_{h=1}^H \delta_h^{\mathbf{p}_d^+} w_{ch}^d$$

Output Gate:

$$\delta_{\omega}^{\mathbf{p}} = f'_1(a_{\omega}^{\mathbf{p}}) \varepsilon_c^{\mathbf{p}} f_3(s_c^{\mathbf{p}})$$

State:

$$\varepsilon_s^{\mathbf{p}} \stackrel{\text{def}}{=} \frac{\partial O}{\partial s_c^{\mathbf{p}}} = b_{\omega}^{\mathbf{p}} f'_1(s_c^{\mathbf{p}}) \varepsilon_c^{\mathbf{p}} + \delta_{\omega}^{\mathbf{p}} w_{c\omega} + \sum_{\substack{d=1: \\ p_d < D_d - 1}}^n (\varepsilon_s^{\mathbf{p}_d^+} b_{\phi,d}^{\mathbf{p}_d^+} + \delta_t^{\mathbf{p}_d^+} w_{ct} + \delta_{\phi,d}^{\mathbf{p}_d^+} w_{c(\phi,d)})$$

Cell:

$$\delta_c^{\mathbf{p}} = b_t^{\mathbf{p}} f'_2(a_c^{\mathbf{p}}) \varepsilon_s^{\mathbf{p}}$$

Forget Gate:

$$\delta_{\phi,d}^{\mathbf{p}} = \begin{cases} f'_1(a_{\phi,d}^{\mathbf{p}}) s_c^{\mathbf{p}_d^-} \varepsilon_s^{\mathbf{p}} & \text{if } p_d > 0 \\ 0 & \text{otherwise} \end{cases}$$

Input Gate:

$$\delta_l^p = f'_1(a_l^p) f_2(a_c^p) \varepsilon_s^p$$

12.2.2 Connectionist Temporal Classification

Connectionist temporal classification (CTC) [5] is an output layer designed for sequence labelling with RNNs. It does not require pre-segmented training data, or post-processing to transform its outputs into transcriptions. It trains the network to predict a conditional probability distribution over all possible output label sequences, or *labellings*, given the complete input sequence.

A CTC output layer contains one more unit than there are elements in the alphabet L of labels for the task. The output activations are normalised at each timestep with the softmax activation function [2]. The first $|L|$ outputs estimate the probabilities of observing the corresponding labels at that time, and the extra output estimates the probability of observing a ‘blank’, or no label. For a length T input sequence \mathbf{x} , the complete sequence of CTC outputs therefore defines a probability distribution over the set L'^T of length T sequences over the alphabet $L' = L \cup \{\text{blank}\}$. We refer to the elements of L'^T as *paths*. Since the probabilities of the labels at each timestep T are conditionally independent given \mathbf{x} , the conditional probability of a path $\pi \in L'^T$ is given by

$$p(\pi | \mathbf{x}) = \prod_{t=1}^T y_{\pi(t)}^t, \quad (12.1)$$

where y_k^t is the activation of output unit k at time t .

Paths are mapped onto labellings $\mathbf{l} \in \mathbf{L}^{\leq T}$ by an operator \mathcal{B} that removes first the repeated labels, then the blanks. So for example, both $\mathcal{B}(a, -, a, b, -)$ and $\mathcal{B}(-, a, a, -, -, a, b, b)$ yield the labelling (a, a, b) . Since the paths are mutually exclusive, the conditional probability of some labelling $\mathbf{l} \in \mathbf{L}^{\leq T}$ is the sum of the probabilities of all paths corresponding to it:

$$p(\mathbf{l} | \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi | \mathbf{x}). \quad (12.2)$$

This ‘collapsing together’ of different paths onto the same labelling is what allows CTC to use unsegmented data, because it means that the network only has to learn the order of the labels, and not their alignment with the input sequence.

Although a naive calculation of Eq. (12.2) is unfeasible, it can be efficiently evaluated with a dynamic programming algorithm, similar to the forward-backward algorithm for HMMs.

To allow for blanks in the output paths, for each labelling $\mathbf{l} \in \mathbf{L}^{\leq T}$ consider a modified labelling $\mathbf{l}' \in \mathbf{L}'^{\leq T}$, with blanks added to the beginning and the end and inserted between every pair of labels. The length $|\mathbf{l}'|$ of \mathbf{l}' is therefore $2|\mathbf{l}| + 1$.

For a labelling \mathbf{l} , define the *forward variable* $\alpha(s, t)$ as the summed probability of all path beginnings reaching index s of \mathbf{l}' at time t , and the *backward variables* $\beta(s, t)$ as the summed probability of all path endings that would complete the labelling \mathbf{l} if the path beginning had reached s at time t . Both the forward and backward variables are calculated recursively [5]. The label sequence probability is given by the sum of the products of the forward and backward variables at any timestep, i.e.

$$p(\mathbf{l}|\mathbf{x}) = \sum_{s=1}^{|\mathbf{l}'|} \alpha(s, t) \beta(s, t). \quad (12.3)$$

Objective Function

Let S be a training set, consisting of pairs of input and target sequences (\mathbf{x}, \mathbf{z}) , where $|\mathbf{z}| \leq |\mathbf{x}|$. Then the objective function \mathcal{O} for CTC is the negative log probability of the network correctly labelling all of S :

$$\mathcal{O} = - \sum_{(\mathbf{x}, \mathbf{z}) \in S} \ln p(\mathbf{z}|\mathbf{x}). \quad (12.4)$$

The network can be trained with gradient descent by first differentiating \mathcal{O} with respect to the outputs, then using backpropagation through time to find the derivatives with respect to the weights.

Note that the same label (or blank) may be repeated several times for a single labelling \mathbf{l} . We define the set of positions where label k occurs as

$$lab(\mathbf{l}, k) = \{s : \mathbf{l}'_s = k\}, \quad (12.5)$$

which may be empty. Setting $\mathbf{l} = \mathbf{z}$ and differentiating \mathcal{O} with respect to the network outputs for a particular element (\mathbf{x}, \mathbf{z}) in the training set, we obtain:

$$\frac{\partial \mathcal{O}}{\partial a_k^t} = -\frac{\partial \ln p(\mathbf{z}|\mathbf{x})}{\partial a_k^t} = y_k^t - \frac{1}{p(\mathbf{z}|\mathbf{x})} \sum_{s \in lab(\mathbf{z}, k)} \alpha(s, t) \beta(s, t), \quad (12.6)$$

where a_k^t and y_k^t are respectively the input and output of CTC unit k at time t for some $(\mathbf{x}, \mathbf{z}) \in S$.

Decoding

Once the network is trained, we can label some unknown input sequence \mathbf{x} by choosing the labelling \mathbf{l}^* with the highest conditional probability, i.e.

$$\mathbf{l}^* = \arg \max_{\mathbf{l}} p(\mathbf{l}|\mathbf{x}). \quad (12.7)$$

In cases where a dictionary is used, the labelling can be constrained to yield only sequences of complete words using the CTC token passing algorithm [6]. For the experiments in this paper, the labellings were further constrained to give single word sequences only, and the n most probable words were recorded. For words with variant spellings, the summed probability of all variants was used as the probability.

Let D be a dictionary of words. All words in a subset U of D are unique, and all other words in D are variants of some word in U . For each word $u \in U$, define $v(u)$ as the set of variants of u , which includes u itself. For each word w , define the modified word w' as w with blanks added at the beginning and end and between each pair of labels. Therefore $|w'| = 2|w| + 1$. For segment s of word w' at timestep t in the output sequence, the value of $\text{tok}(w, s, t)$ is defined as the probability of the most probable partial output path $\pi(1 : t)$ such that $\pi(t) = w'(s)$ and $B(\pi(1 : t)) = w(1 : s/2)$, where $A(b : c)$ denotes the subsequence of sequence A from index b to index c .

At every timestep t of the length T output sequence, each segment s of each modified word w' holds a single token $\text{tok}(w, s, t)$. This is the highest token reaching that segment at that time. The *output token* $\text{tok}(w, -1, t)$ is the highest token leaving word w at time t .

Pseudocode is provided in Algorithm 1. Note that in cases where decoding speed is important, the algorithm could be optimised by storing the words in a tree structure.

12.2.3 Network Hierarchy

Many computer vision systems use a hierarchical approach to feature extraction, with the features at each level used as input to the next level [17]. This allows complex visual properties to be built up in stages. Typically, such systems use subsampling, with the feature resolution decreased at each stage. They also generally have more features at the higher levels. The basic idea is to progress from a small number of simple local features to a large number of complex global features.

We created a hierarchical structure by repeatedly composing MDLSTM layers with feedforward layers. The basic procedure is as follows. (1) The image is divided into pixel windows, each of which is presented as a single input to the first set of MDLSTM layers (e.g. a 4×3 window is collapsed to a length 12 vector). If the image does not divide exactly into windows, it is padded with zeros. (2) The four MDLSTM layers scan through the window vectors in all directions. (3) The activations of the MDLSTM layers are collected into windows. (4) These windows are given as input to a feedforward layer. Note that all the layers have a 2D array of activations: e.g. a 10 unit feedforward layer with input from a 5×5 array of MDLSTM windows has a total of 250 activations.

The above process is repeated as many times as required, with the activations of the feedforward layer taking the place of the original image. The purpose of the windows is twofold: to collect local contextual information, and to reduce the area

```

1: Initialisation:
2: for all words  $w \in D$  do
3:    $tok(w, 1, 1) = \ln y_b^1$ 
4:    $tok(w, 2, 1) = \ln y_{w_1}^1$ 
5:   if  $|w| = 1$  then
6:      $tok(w, -1, 1) = tok(w, 2, 1)$ 
7:   else
8:      $tok(w, -1, 1) = -\infty$ 
9:   end if
10:   $tok(w, s, 1) = -\infty$  for all other  $s$ 
11: end for
12:
13: Algorithm:
14: for  $t = 2$  to  $T$  do
15:   sort output tokens  $tok(w, -1, t - 1)$  by ascending value
16:   for all words  $w \in D$  do
17:     for segment  $s = 1$  to  $|w'|$  do
18:        $P = \{tok(w, s, t - 1), tok(w, s - 1, t - 1)\}$ 
19:       if  $w'(s) \neq \text{blank}$  and  $s > 2$  and  $w'(s - 2) \neq w'(s)$  then
20:         add  $tok(w, s - 2, t - 1)$  to  $P$ 
21:       end if
22:        $tok(w, s, t) = \max(P) + \ln y_{w'(s)}^t$ 
23:     end for
24:      $tok(w, -1, t) = \max(tok(w, |w'|, t), tok(w, |w'| - 1, t))$ 
25:   end for
26: end for
27:
28: Termination:
29: for all unique words  $u \in U$  do
30:    $tok(u, -1, T) = \sum_{w \in v(u)} tok(w, -1, T)$ 
31: end for
32: output  $n$  best  $tok(u, -1, T)$ 

```

Algorithm 1: CTC token passing algorithm for single words

of the activation arrays. In particular, we want to reduce the vertical dimension, since the CTC output layer requires a 1D sequence as input. Note that the windows themselves do not reduce the overall amount of data; that is done by the layers that process them, which are therefore analogous to the subsampling steps in other approaches (although with trainable weights rather than a fixed subsampling function).

For most tasks we find that a hierarchy of three MDLSTM/feedforward stages gives the best results. We use the standard ‘inverted pyramid’ structure, with small layers at the bottom and large layers at the top. As well as allowing for more fea-

tures at higher levels, this leads to efficient networks, since most of the weights are concentrated in the upper layers, which have a smaller input area.

Unless we know that the input images are of fixed height, it is difficult to choose window heights that ensure that the final feature map will always be one dimensional, as required by CTC. A simple solution is to collapse the final array by summing over all the inputs in each vertical line; i.e. the input at time t to CTC unit k is given by

$$a_k^t = \sum_x a_k^{(x,t)} \quad (12.8)$$

where $a_k^{(x,y)}$ is the uncollapsed input to unit k at point (x, y) in the final array.

Furthermore, the widths of the windows must be chosen to prevent the final feature map from being shorter (horizontally) than the number of labels for a particular sequence, since CTC assumes that the input sequence is at least as long as the label sequence. If the trained system is to be applied to images of unknown dimensions, it is therefore a good idea to ensure that the final feature map is considerably longer than the target label sequence for every element of the training set.

12.2.4 Combined System

Figure 12.4 shows how MDLSTM, CTC, and the layer hierarchy combine to form a complete recognizer.

12.3 Experiments

Variants of the above system won several competitions at the 2009 International Conference on Document Analysis and Recognition (ICDAR 2009). In this section we describe the winning entry to the offline Arabic handwriting recognition competition.

12.3.1 Data

The competition was based on the publicly available IFN/ENIT database of handwritten Arabic words [16]. The data consists of 32,492 images of individual handwritten Tunisian town and village names, of which we used 30,000 for training, and 2,492 for validation. The images were extracted from artificial forms filled in by over 400 Tunisian people. The forms were designed to simulate writing on a letter, and contained no lines or boxes to constrain the writing style.

Each image was supplied with a ground truth transcription for the individual characters, and the postcode of the corresponding town. There were 120 distinct

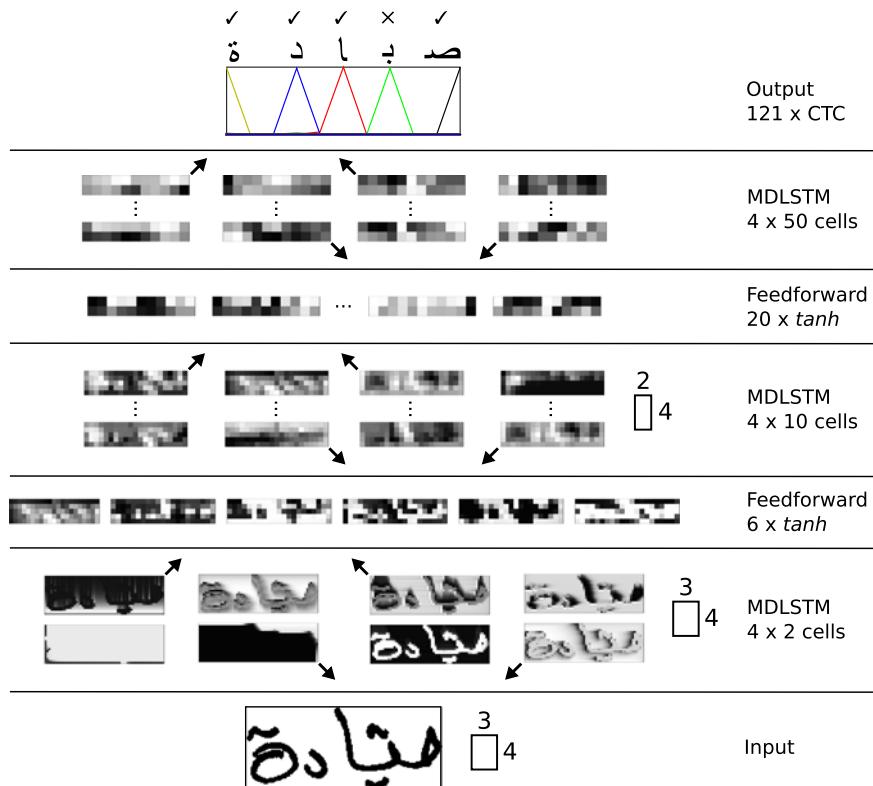


Fig. 12.4 A complete handwriting recognition system. First the input image is collected into windows 3 pixels wide and 4 pixels high which are then scanned by four MDLSTM layers. The activations of the cells in each layer are displayed separately, and the arrows in the corners indicate the scanning direction. Next the MDLSTM activations are gathered into 4×3 windows and fed to a feedforward layer of tanh summation units. Again the activations are displayed separately. This process is repeated two more times, until the final MDLSTM activations are collapsed to a 1D sequence and transcribed by the CTC layer. In this case all characters are correctly labelled except the second to last one

characters in total, including variant forms for initial, medial, final and isolated characters. The goal of the competition was to identify the postcode, from a list of 937 town names and corresponding postcodes. Many of the town names had transcription variants, giving a total of 1,518 entries in the complete dictionary.

The test data (which is not published) was divided into sets ‘f’ and ‘s’. The main competition results were based on set ‘f’. Set ‘s’ contains data collected in the United Arab Emirates using the same forms; its purpose was to test the robustness of the recognizers to regional writing variations. The systems were allowed to choose up to 10 postcodes for each image, in order of preference. The test set performance using the top 1, top 5, and top 10 answers was recorded by the organisers.

12.3.2 Network Parameters

Three versions of the MDLSTM handwriting recognition system were entered for the competition, with slightly different parameters. Within the competition, they were given the collective group ID ‘MDLSTM’. For the first two networks (assigned system IDs 9 and 10 in the competition) the topology shown in Fig. 12.4 was used, with each layer fully connected to the next layer in the hierarchy, all MDLSTM layers connected to themselves, and all units connected to a bias weight. These networks had 159,369 weights in total. The third network (system ID 11) had twice as many units in each of the hidden layers. That is, the four MDLSTM layers in the first level had four cells each, the first feedforward layer had 12 units, the MDLSTM layers in the second level had 20 cells each, the second feedforward layer had 40 units and the MDLSTM layers in the third level had 100 cells each. This gave a total of 583,289 weights.

For all networks the activation function used for the LSTM gates was the logistic sigmoid $f_1(x) = 1/(1 + e^{-x})$, while tanh was used for f_2 and f_3 (cf. Sect. 12.2.1).

The networks were trained with online gradient descent, using a learning rate of 10^{-4} and a momentum of 0.9. Both the CTC objective function \mathcal{O} (Sect. 12.2.2) and the character error rate (total number of insertions, deletions and substitutions needed to transform the network outputs into the target sequences, divided by the total length of the target sequences) were evaluated on the validation set after every pass through the training set. For networks 9 and 11 the error measure was the character error rate, while for network 10 the error measure was the CTC objective function. Networks 9 and 10 were created during the same training run, with the two different error measures used as a stopping criterion. For all networks training was stopped after 30 evaluations with no reduction in the error measure on the validation set. The weights giving the lowest error on the validation set were passed to the competition organisers for assessment on the test sets.

Figure 12.5 shows the error curves for networks 9 and 10 during training. Note that, by the time the character error is minimised, the CTC error is already well past its minimum and has risen substantially. This is typical for networks trained with CTC output layers.

Network ID 9 took 86 passes through the training set to complete training, network ID 10 took 49 passes, and network ID 11 took 153 passes. The time per pass, which grows with the number of network weights, was around 62 minutes for networks one and two, and around 180 minutes for network three. The fact that network three required more training passes than network two is untypical, since usually the more weights a network has the fewer passes it takes to minimise a particular error measure. However, the same network minimised the CTC error on the validation set after only 22 passes, and the decrease in validation character error rate between 22 and 152 passes was only 0.2.

Table 12.1 summarises the differences between the three networks.

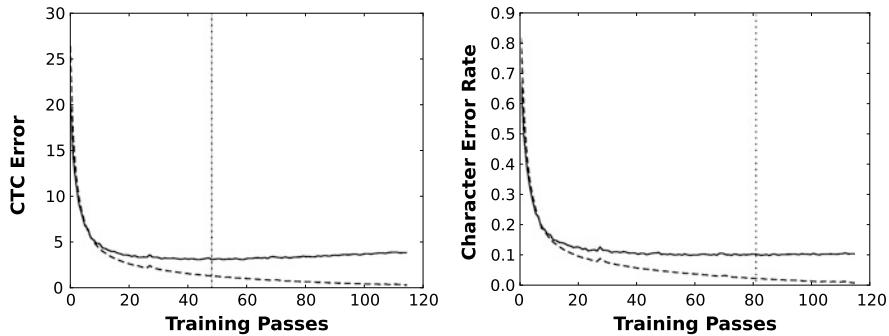


Fig. 12.5 Error curves during training of networks 9 and 10. The CTC error is shown on the left, and the character error is shown on the right. In both plots the solid line shows the error on the validation set, the dashed line shows the error on the training set, and the vertical dotted line indicates the point of lowest error on the validation set

Table 12.1 The three MDLSTM networks entered for the Arabic handwriting competition

ID	Weights	Error measure	Passes	Approx. pass time (mins)
9	159,369	Character	86	62
10	159,369	CTC	49	62
11	583,289	Character	153	180

12.3.3 Results

Table 12.2 [14] shows that all three MDLSTM networks (group ID MDLSTM, system ID 9–11) outperformed all other entries in the 2009 International Conference on Document Analysis and Recognition (ICDAR 2009), in terms of both recognition rate and speed. The recognition rates were also better than any of the entries in the ICDAR 2007 competition, which used the same training and test data, although the Siemens and MIE systems were faster.

The overall difference in performance between networks 9 and 10 is negligible, suggesting that it isn't that important which error measure is used for early stopping. This is significant, since, as discussed above, using the CTC error for early stopping can lead to much shorter training times. Of particular interest is that the performance on set s (with handwriting from the United Arab Emirates) is about the same for both error measures. One hypothesis was that, because using the CTC error leads to fewer training passes, network 10 would overfit less on the training data and therefore generalise better to test data drawn from a different distribution.

Network 11 gave about a 2 % improvement over networks 9 and 10 in word recognition for both test sets, if only the best word was used. Although significant, this improvement comes at a cost of a more than threefold increase in word recognition time. For applications where time must be traded against accuracy, the number of units in the network layers (and hence the number of network weights) should be tuned accordingly.

Table 12.2 ICDAR 2009 Arabic offline handwriting recognition competition results. Results are % of correctly recognised images on reference datasets d and e , new datasets f and s , subsets f_a, f_f , and f_g . The average recognition time in ms per image on subsets t and t_1 is shown in the last two columns. (G-ID: Group ID, S-ID: System ID)

G-ID	S-ID	Set d	Set e	Set f_a	Set f_f	Set f_g	Set f	Set s			Time (ms)			
								Top 1	Top 1	Top 1	Top 5	Top 10	Set t	Set t_1
UOB-ENST	1	92.52	85.38	83.57	84.77	85.09	82.07	89.74	91.22	69.99	81.44	84.68	812.69	841.25
	2	89.06	81.85	79.49	80.90	81.11	78.16	89.06	91.88	65.61	81.44	85.95	2365.48	2755.01
	3	89.84	83.52	80.89	82.15	82.17	79.55	90.60	92.16	67.83	83.47	86.65	2236.58	2754.08
	4	92.59	86.28	85.42	86.96	87.21	83.98	91.85	93.00	72.28	85.19	87.92	2154.48	2651.57
REGIM	5	79.52	63.53	58.81	59.27	60.42	57.93	73.43	78.10	49.33	65.10	71.14	1564.75	1712.15
	6	93.90	87.25	86.73	88.54	89.36	85.58	92.57	94.12	70.44	82.01	84.87	1056.98	956.82
A12A	7	94.92	82.21	83.53	84.86	84.67	82.21	91.24	92.47	66.45	80.52	83.13	519.61	1616.82
	8	97.02	91.68	90.66	91.92	92.31	89.42	95.33	95.94	76.66	88.01	90.28	2583.64	1585.49
MDLSTM	9	99.72	98.64	92.59	93.79	94.22	91.43	96.11	96.61	78.83	87.98	90.40	115.24	122.97
	10	99.60	97.60	92.58	94.03	94.40	91.37	96.24	96.61	78.89	88.49	90.27	114.61	122.05
RWTH-OCR	11	99.94	99.44	94.68	95.65	96.02	93.37	96.46	96.77	81.06	88.94	90.72	371.85	467.07
	12	99.91	98.71	86.97	88.08	87.98	85.51	93.32	94.61	71.33	83.66	86.52		
LITIS-MIRACL	13	99.79	98.29	87.17	88.63	88.68	85.69	93.36	94.72	72.54	83.47	86.78	17845.12	18641.93
	14	99.79	98.29	87.17	88.63	88.68	85.69	93.36	94.72	72.54	83.47	86.78		
LSTS	15	96.72	91.25	86.97	88.08	87.98	83.90	—	—	65.99	—	—	542.12	560.44
	16	93.04	85.46	83.29	84.51	84.35	82.09	90.27	92.37	74.51	86.14	88.87	143269.81	145157.23
Results of the 3 best systems at ICDAR 2007														
Siemens	08	94.58	87.77	88.41	89.26	89.72	94.05	95.42	73.94	85.44	88.18	109.406	125.31	
MIE	06	93.63	86.67	84.38	85.21	85.56	83.34	91.67	93.48	68.40	80.93	83.73	188.439	210.55
UOB-ENST	11	92.38	83.92	83.39	84.93	85.18	81.93	91.20	92.76	69.93	84.11	87.03	2172.55	2425.47

12.4 Conclusion

This chapter introduced a general offline handwriting recognition system based on MDLSTM recurrent neural networks. The system works directly on raw pixel data, and therefore requires minimal changes to be used for languages with different alphabets. It won several competitions at the ICDAR 2009 conference, including the Arabic offline handwriting recognition competition.

Various extensions to the system are currently being explored, including more efficient decoding, complete page transcription, and weight pruning for increased speed.

References

1. Baldi, P., Pollastri, G.: The principled design of large-scale recursive neural network architectures-DAG-RNNs and the protein structure prediction problem. *J. Mach. Learn. Res.* **4**, 575–602 (2003)
2. Bridle, J.S.: Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: Fogelman-Soulie, F., Herault, J. (eds.) *Neurocomputing: Algorithms, Architectures and Applications*, pp. 227–236. Springer, Berlin (1990)
3. Gers, F., Schraudolph, N., Schmidhuber, J.: Learning precise timing with LSTM recurrent networks. *J. Mach. Learn. Res.* **3**, 115–143 (2002)
4. Graves, A.: Supervised sequence labelling with recurrent neural networks. Ph.D. in Informatik, Fakultät für Informatik—Technische Universität München (2008)
5. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: *Proceedings of the International Conference on Machine Learning, ICML 2006*, Pittsburgh, PA, USA (2006)
6. Graves, A., Fernández, S., Liwicki, M., Bunke, H., Schmidhuber, J.: Unconstrained online handwriting recognition with recurrent neural networks. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) *Advances in Neural Information Processing Systems*, vol. 20. MIT Press, Cambridge (2008)
7. Graves, A., Fernández, S., Schmidhuber, J.: Multidimensional recurrent neural networks. In: *Proceedings of the 2007 International Conference on Artificial Neural Networks*, Porto, Portugal, September 2007
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
9. Hu, J., Lim, S.G., Brown, M.K.: Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognit.* **33**, 133–147 (2000)
10. Jaeger, S., Manke, S., Reichert, J., Waibel, A.: On-line handwriting recognition: the NPen++ recognizer. *Int. J. Doc. Anal. Recognit.* **3**, 169–180 (2001)
11. Jiang, H.: Discriminative training of HMMs for automatic speech recognition: A survey. *Comput. Speech Lang.* **24**(4), 589–608 (2010)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998)
13. LeCun, Y., Muller, U., Ben, J., Cosatto, E., Flepp, B.: Off-road obstacle avoidance through end-to-end learning. In: *Advances in Neural Information Processing Systems (NIPS 2005)*. MIT Press, Cambridge (2005)
14. Märgner, V., El Abed, H.: In: *ICDAR 2009 Arabic Handwriting Recognition Competition*, Jul. 2009, pp. 1383–1387 (2009)
15. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In: *NIPS*, pp. 841–848 (2001)

16. Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT-database of handwritten Arabic words. In: 7th Colloque International Francophone sur l'Ecrit et le Document (CIFED 2002), Hammamet, Tunis (2002)
17. Reisenhuber, M., Poggio, T.: Hierarchical models of object recognition in cortex. *Nat. Neurosci.* **2**(11), 1019–1025 (1999)
18. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **45**, 2673–2681 (1997)

Chapter 13

Application of Fractal Theory in Farsi/Arabic Document Analysis

Saeed Mozaffari

Abstract In the past, fractal theory has been used mostly in computer-generated graphics, and image compression fields. But recent researches show that fractal theory has been successfully adopted in document analysis realm, such as: online and offline character recognition, font identification, and watermarking in document images. After a short review on fractal dimension, fractal coding and decoding, this chapter will present the results of fractal theory in Farsi/Arabic document analysis.

13.1 Introduction

The term “fractal” was first proposed by Mandelbrot in 1975 [10]. It originated from the French word *fractus*, which means to break a stone in an amorphous way. The word properly indicates both the chaotic and fragmentary properties of the fractal theory. Although a comprehensive definition of fractals seems to be elusive, the general consensus defines a set as fractals if it has the following properties [5]. (1) The set must have details at every scale. (2) It should be self-similar. (3) There must be a simple algorithmic approach to describe it. Figure 13.1 shows some fractal images.

With the advent of powerful computers, fractal theory has been used frequently in different fields, such as computer-generated graphics, image compression, and pattern description and recognition.

In the graphic modeling area, fractals are used to ease the difficulty of creating complex scenes and objects such as mountains, trees, and coastlines with fairly small pieces of code.

Fractal image compression methods are based on the fact that our natural environment generally shows self-similarity on different scales. Therefore, a considerable amount of redundancy is implied in the images according to this self-similarity property. By means of the iterated function system (IFS) [5], there would be a contractive transformation for each image that has a fixed point resembling the original

S. Mozaffari (✉)

Electrical and Computer Department, Semnan University, Semnan, Iran
e-mail: mozaffari@semnan.ac.ir

Fig. 13.1 Some examples of fractal images



image. In other words, applying that transform iteratively on an arbitrary starting image, the result will converge to the original image [7].

Fractal theory was originally used for texture description and segmentation. It is called the Lindenmayer system (L-system) in texture analysis and is mostly based on a recursive, context-free, deterministic grammar [17]. At each iteration, all applicable rules are applied simultaneously, and the expansion is stopped after a pre-determined number of iterations. In addition to texture, several fractal features and fractal-based recognition algorithms have been proposed to classify other patterns like faces [1, 3].

Fractal theory has been used frequently in the field of document analysis. To detect a document's skew, it is segmented into blocks by a fractal approach [22]. Wang et al. have done some mathematical description and verification to a cluster of IP addresses and a computer directory/file tree based on fractal theory [21]. Tao and Tang presented a new approach based on modified fractal signatures (MFSs) and modified fractal features (MFFs) for the discrimination of Oriental and Euromerican scripts [20]. Bangla, English, and Devnagari scripts were separated with fractal-based features in a trilingual script postal automation system [15]. Eiterer et al. proposed an address block segmentation approach based on fractal dimension [4]. Fractal image encoding is exploited to obtain image indexing systems that are able to deal with the images in compressed form, which makes them suitable for use with large databases [2]. Fractal descriptors can categorize similar documents based on font matching [8]. The fractal dimension of a text document is utilized to achieve a better diversification of the extracted sentences for summarizing structured documents [16]. The recognition accuracy of an optical character recognition (OCR) system was improved by the use of a pre-clustering of the writings accord-

ing to fractal analysis of the writing styles [6]. A writer identification technique was proposed in [24] by combining Gabor wavelet and mesh fractal dimensions.

The aim of this chapter is to address some recent applications of fractal theory in Farsi/Arabic document analysis and recognition. In the following sections, after a short review of basic concepts in fractal theory, on-line and off-line character recognition, font recognition, and watermarking applications will be discussed in more detail.

13.2 Fractal Dimension

Fractal dimension is a basic concept in fractal theory which is used in some applications like font recognition. According to the definitions presented in [5], the topological dimension of a totally disconnected set is always zero. The topological dimension of a set F is n if arbitrary small neighborhoods of every point of F have a boundary with topological dimension of $n - 1$. The topological dimension is always an integer. For example, an interval has topological dimension 1 because at each point we can find a neighborhood, which is also an interval, whose boundary is a disconnected set and hence has topological dimension zero.

There are many definitions for non-integral dimensions. The most famous one is the box dimension, which is defined as follows.

For $F \in R^n$ let $N_\varepsilon(F)$ denote the smallest number of sets with diameter no larger than ε that can cover F . The box dimension of F is:

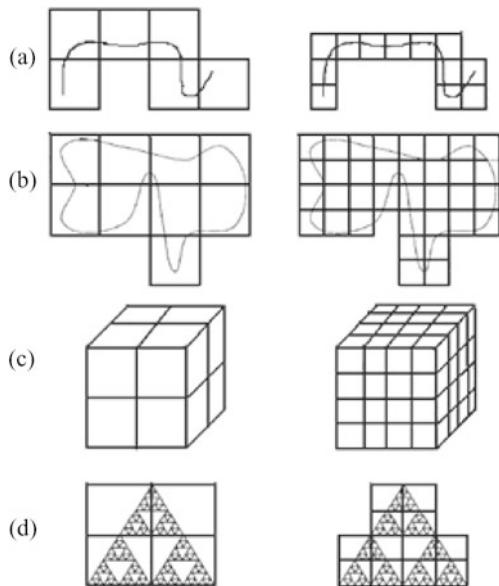
$$\lim_{\varepsilon \rightarrow 0} \frac{\log N_\varepsilon(F)}{-\log \varepsilon} \quad (13.1)$$

The fractal dimension can be considered as a scaling relationship. Figure 13.2 shows four examples of sets and their scaling relationship which is determined by the number of boxes it takes to cover the set. For each example, the scaling relationship is described as follows:

- a. A curve of length l can be covered by $\frac{l}{\varepsilon}$ boxes of size ε and $2\frac{l}{\varepsilon}$ boxes of size $\frac{\varepsilon}{2}$.
- b. A region of area A can be covered by $\frac{A}{\varepsilon^2}$ boxes of size ε and $2^2\frac{A}{\varepsilon^2}$ boxes of size $\frac{\varepsilon}{2}$.
- c. A set with volume V can be covered by $\frac{V}{\varepsilon^3}$ boxes of size ε and $2^3\frac{V}{\varepsilon^3}$ boxes of size $\frac{\varepsilon}{2}$.
- d. If the Sierpinski triangle is covered with N boxes of size ε , then it takes $3N$ boxes of size $\frac{\varepsilon}{2}$ to cover it. This is shown for ε equal to half the width of the set in the figure.

Figures 13.2(a), (b), and (c) whose dimensions are 1, 2, and 3, have box sizes corresponding to increasing the number of boxes required to cover the set by a factor of 2^1 , 2^2 , and 2^3 . However, for the Sierpinski triangle the number of boxes increases by 2^d where $d = \frac{\log(3)}{\log(2)}$. So in this case, the fractal dimension is a number between 1 and 2. An avid reader may refer to [5] for more information.

Fig. 13.2 Four sets and the number of ε boxes required to cover them



13.3 Iterated Function System

Fractal image compression is based on the concepts and mathematical results of an iterated function system (IFS) [7]. The fundamental principle of fractal coding consists of the representation of any image I by a contractive transformation T in which the fixed point is too close to the original image. In other words, when we apply that transform iteratively on an arbitrary starting image, the result will converge to the original image:

$$I_{n+1} = T(I) \quad (13.2)$$

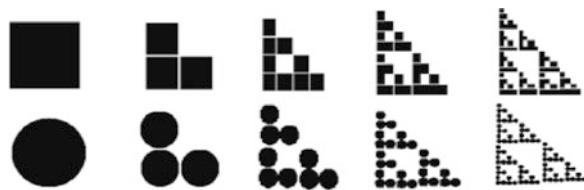
$$I = \lim_{n \rightarrow \infty} T(I) \quad (13.3)$$

An IFS is a set of geometrical elementary linear or affine contractive transformers that allows us to generate a fractal image. These n transforms make possible the definition of a function T defined by:

$$T(I) = \bigcup_{i=1}^n T_i(I) \quad (13.4)$$

Banach's fixed point theorem guarantees that, within a complete metric space, the fixed point of such a transformation may be recovered by an iterated application to an arbitrary initial element of that space. The fixed point (image) that is obtained

Fig. 13.3 Generation of a fractal image by IFS



has some specific properties. In particular, it is made of copies of itself but modified by the elementary transforms. In Fig. 13.3, the IFS is made of three transforms, a reduction, followed by a translation, and repositioning in a triangle shape. The fixed point is the Sierpinski triangle, which is independent of the initial image (circle or square).

Fractal compression became a practical reality with the introduction of the partitioned IFS (PIFS) by Jacquin [7]. It differs from the IFS in the way that each of the individual mappings operates on a subset of the image, rather than on the entire image. A PIFS defines a transform T that is the union of affine contractive transforms defined on domains included in the image:

$$T(I) = T_1(I_1) \cup T_2(I_2) \cup \dots \cup T_n(I_n) \quad (13.5)$$

The set of all images obtained from all the transformations of sub-images I_i enables us to partition the spatial domain of I . So, if the right PIFS is built, the initial image would be the attractor of the IFS and could be derived from any image.

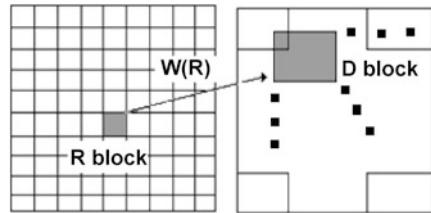
13.4 Two-Dimensional Fractal Coding and Decoding

Two-dimensional (2D) fractal coding and decoding algorithms are used for fractal image compression and coding. They are fundamental issues needed in the following sections, so they are described in more detail. Other related subjects like fractal coding speed enhancement and image quality augmentation are not addressed in this chapter.

13.4.1 Fractal Image Coding

An image to be encoded is partitioned into non-overlapping range blocks, R , with size $N \times N$ and overlapping domain blocks, D , with size $2N \times 2N$ as depicted in Fig. 13.4.

Fig. 13.4 One of the block mappings in PIFS representation



Suppose we are dealing with an $M \times M$ grayscale image, $I(x, y)$, in which each pixel can have one of 256 levels (ranging from black to white). In this case, the number of R blocks would be $n_r = [\frac{M}{N}] \times [\frac{M}{N}]$.

$$I(x, y) = \begin{bmatrix} (x_0, y_0) & (x_0, y_0) & \dots & \dots & (x_0, y_{M-1}) \\ (x_1, y_0) & (x_1, y_1) & \dots & \dots & (x_1, y_{M-1}) \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ (x_{M-1}, y_0) & (x_{M-1}, y_1) & \dots & \dots & (x_{M-1}, y_{M-1}) \end{bmatrix} \quad (13.6)$$

The input image $I(x, y)$ is further grouped into n range blocks $I(x, y) = r_1 + r_2 + \dots + r_{n_r}$.

Each range block is characterized by the number of its pixels, $N \times N$, and its starting point, rs , which always points to the top left pixel in the corresponding block.

$$r_k = \left\{ \begin{bmatrix} (x_i, y_j) & (x_{i+1}, y_j) & \dots & \dots & (x_{i+N-1}, y_j) \\ (x_i, y_{j+1}) & (x_{i+1}, y_{j+1}) & \dots & \dots & (x_{i+N-1}, y_{j+1}) \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ (x_i, y_{j+N-1}) & (x_{i+1}, y_{j+N-1}) & \dots & \dots & (x_{i+N-1}, y_{j+N-1}) \end{bmatrix} \mid i = rsx_k, j = rsy_k \right\} \quad (13.7)$$

Since the size of D blocks is assumed to be $2N \times 2N$, the collection D contains $n_d = [M - 2N + 1] \times [M - 2N + 1]$ overlapped squares. Similar to the range blocks, each of the domain blocks is characterized by the number of its pixels, $2N \times 2N$,

and its starting point, ds .

$$d_k = \left\{ \begin{bmatrix} (x_i, y_j) & (x_{i+1}, y_j) & \dots & \dots & (x_{i+2N-1}, y_j) \\ (x_i, y_{j+1}) & (x_{i+1}, y_{j+1}) & \dots & \dots & (x_{i+2N-1}, y_{j+1}) \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ (x_i, y_{j+2N-1}) & (x_{i+1}, y_{j+2N-1}) & \dots & \dots & (x_{i+2N-1}, y_{j+2N-1}) \end{bmatrix} \mid i = dsx_k, j = dsy_k \right\} \quad (13.8)$$

The task of a fractal coder is to find a D block in the same image for each R block such that transformation of this domain block, $W(D)$, minimizes the collage error in Eq. (13.9):

$$\text{CollageError} = \min \|R - W(D)\|^2 \quad (13.9)$$

The transformation W , in Eq. (13.9), which maps each D block into its corresponding R block is assumed to be an affine transformation. An affine transformation preserves colinearity and ratios of distances. It does not necessarily preserve angles or lengths. In other words, an affine transformation can transform a rectangle into a parallelogram. Usually the affine transformation set is limited to scaling, stretching, skewing, and rotating.

Since range-to-domain block matching under several transformations is very time consuming, it is usually desirable to restrict the transformation set into isometric affine transformations. In this manner, one can speed up the encoding process at the expense of image quality reduction, which is not very crucial in pattern recognition applications. A transformation f is called isometric if it keeps the distance function, d , invariant:

$$d(x, y) = d(f(x), f(y)) \quad (13.10)$$

The only isometric affine transformation is the *rotation*, possibly composed with the *flip*. Among all rotations, four preserve the orientation of a square, namely, the identity, the 90° rotation, the 180° rotation, and the 270° rotation. Composing them with the flip operator, eight deformation matrices are obtained (Table 13.1).

As mentioned before, a D block has four times as many pixels as an R block. So, we must average the 2×2 sub-squares corresponding to each pixel of the R block when minimizing Eq. (13.9), and this averaging process must be performed repeatedly. The simplest way to reduce this computational burden is with the use of a look-up table containing intensity means.

The transformation W in Eq. (13.9) is a combination of geometrical and luminance transformations. According to Eq. (13.11) a point at coordinate (x, y) with

Table 13.1 Eight isometric affine transforms

Index	Isometry	Matrix
1	identity	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
2	x flip	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
3	y flip	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
4	180° rotation	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
5	$x = y$ flip	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
6	270° rotation	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
7	90° rotation	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
8	$x = -y$ flip	$\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$

gray level z is rotated and scaled by the geometrical parameters a , b , c , and d and is offset by parameters e and f . Its gray level is scaled by S (contrast operator) and offset by O (brightness operator).

Minimizing Eq. (13.9) means two things. First, it means finding a good choice for D_i , and second, it means finding a good contrast and brightness setting for W_i in Eq. (13.11). A choice of D_i , along with a corresponding S_i and O_i , determines a map W_i .

$$W_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \quad (13.11)$$

In some applications, like optical character recognition (OCR), the shape and style of lines and patterns are more important than the gray level distribution of pixels. The gray level distribution represents pressure and writing speed; therefore, considering the geometrical relationship between range and domain blocks rather than the distribution of the pixel gray levels is more practical. Instead of finding the scaling and offset parameters (S and O), usually the average of gray levels in the R block is utilized.

The fractal code is defined as the set of all n_r range-to-domain affine transformations. Each transformation $f(k)$ consists of six real numbers:

- Starting point of the R block, $rs_k = (rsx_k, rsy_k)$.
- Starting point of the corresponding D block, $ds_k = (dsx_k, dsy_k)$.
- The index of the d_k to R_k transformation, T_k . The index is a number between 1 and 8 (Table 13.1).
- The average intensity of the range block, which is a number between 0 and 255.

The encoding algorithm can be summarized as follows:

1. Input the original gray level image.
2. Partition the input image into R blocks.

3. Create a list of D blocks.
4. Scan the image from top to bottom and from left to right.
5. Search for a fractal match. Given a region, loop over all possible D blocks to find the best match using a given metric (Eq. (13.9)).
6. After finding the best match, set the fractal codes.

13.4.2 Fractal Image Decoding

The reverse process of generating an image from a fractal model is called decoding. The decoding process starts with an arbitrary $M \times M$ initial image. For each fractal transformation $f(k)$, the $2N \times 2N$ domain block d_k is constructed from the initial image, given its start point ds_k stored in the fractal code. Then its corresponding stored affine transformation T_k is applied on constructed D block d_k . After down-sampling according to the averaging transformation in the encoding process, the $N \times N$ obtained block is translated to the corresponding R block at rsk . This completes one iteration.

The decoding algorithm is iterated about 6 to 16 times until the fixed point image is obtained. The final image is assumed to be created when the difference between two successive images of the sequence is small enough. To measure the quality of the fractal compression of an image, besides the compression ratio, the peak signal-to-noise ratio is generally used.

The simplified 2D fractal coding/decoding method mainly concerns the geometrical characteristics of grayscale images for pattern recognition rather than image compression. Therefore, high image quality is not expected to be obtained. Figure 13.5 shows the results of the decoding algorithm with different numbers of iterations. It is obvious that the fixed point image is obtained approximately after 5 iterations.

13.5 One-Dimensional Fractal Coding and Decoding

In the previous section, 2D fractal image coding and decoding algorithms are explained for off-line character recognition. However, these algorithms can be simplified into a one-dimensional (1D) version which is suitable for on-line character recognition.

In an on-line character recognition system, the most common writing tool is a digitizing tablet and a special pen that records the coordinates of the plotted points at a constant frequency. In on-line recognition, the writing order is available and the writing line has no width. Moreover, temporal information, like writing velocity and pen lifts, are available and can be used for the recognition process.

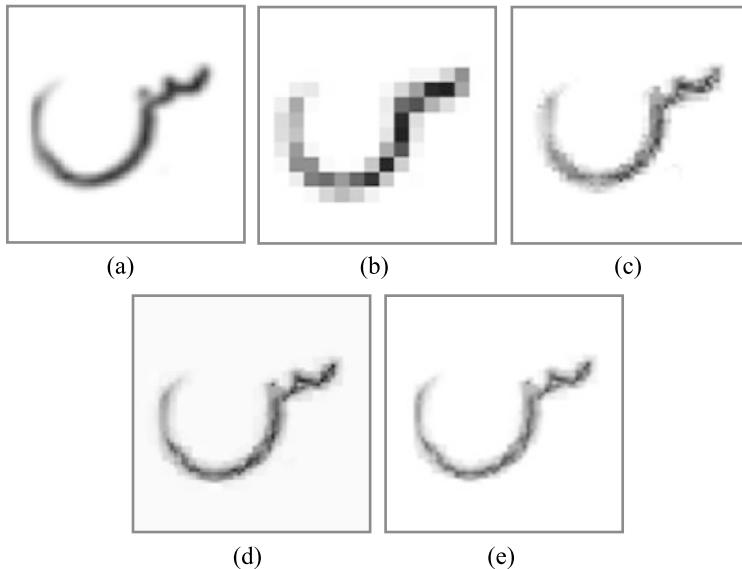


Fig. 13.5 Results of decoding algorithm. (a) Original image. (b) Decoded image after 1st iteration. (c) Decoded image after 2nd iteration. (d) Decoded image after 5th iteration. (e) Decoded image after 10th iteration

13.5.1 One-Dimensional Fractal Coder

In on-line recognition, the process is performed on 1D data rather than 2D images as in the case of off-line recognition. In this case, we are dealing with a time ordered sequence of points based on the pen positions. Therefore, gray level, contrast, and luminosity information are meaningless in on-line data. Since on-line data is modeled as a set of (x, y) coordinates, we can simplify Eq. (13.11) by omitting the third row which includes parameters of luminance transformation to obtain Eq. (13.12):

$$v_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} \quad (13.12)$$

Then v_i determines how the partitioned ranges of a signature are mapped to their domains. The 1D encoding process is similar to the 2D fractal coding presented before. The digit locus is divided into non-overlapping range segments with the length of N . For each range segment, a two times larger corresponding domain segment with the length of $2N$ is searched within the digit locus, such that under appropriate affine transformations (Table 13.1), the identified domain segment can best approximate the range segment. Assume that, after pre-processing, we are dealing with a uniform resampled input locus with M sample points $I(x, y) = \{(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)\}$. The input locus can be grouped

into $n_r = [\frac{M}{N}]$ range segments and $n_d = [M - 2N + 1]$ domain segments.

$$\begin{aligned} r_k &= \{(x_i, y_i), \dots, (x_{i+N-1}, y_{i+N-1}) | i = rs_k\} \\ d_k &= \{(x_j, y_j), \dots, (x_{j+2N-1}, y_{j+2N-1}) | j = ds_k\} \end{aligned} \quad (13.13)$$

Each of the R and D segments is characterized by the number of their points (N and $2N$) and their centroid and according to Eq. (13.14):

$$\begin{aligned} rc_k &= (rcx_k, rcy_k) = \frac{1}{N} \left(\sum_{t=rs_k}^{rs_k+N-1} x_t, \sum_{t=rs_k}^{rs_k+N-1} y_t \right) \\ dc_k &= (dcx_k, dcy_k) = \frac{1}{2N} \left(\sum_{t=ds_k}^{ds_k+2N-1} x_t, \sum_{t=ds_k}^{ds_k+2N-1} y_t \right) \end{aligned} \quad (13.14)$$

Similar to fractal image coding, the range-to-domain segment transformations are limited to isometric affine transformations according to Table 13.1. During the search for the best D segment k , each of (x_i, y_i) points in the candidate domain segment d_k is downsampled as $\frac{N}{2N}(x_i, y_i)$.

Each fractal code $f(k)$ consists of five real numbers that indicate an affine transformation:

- The range segment centroid $rc_k = (rcx_k, rcy_k)$.
- The domain segment centroid $dc_k = (dcx_k, dcy_k)$.
- The index of d_k to R_k transformation, T_k .

13.5.2 One-Dimensional Fractal Decoder

For the decoding process, first an arbitrary initial locus S with M points is created. For each fractal code $f(k)$, the D segment is constructed, given the domain segment centroid dc_k stored in the fractal code and the number of its points $2N$. Then the stored affine transformation T_k is performed on it. After downsampling to N points, its centroid is shifted to the stored R segment centroid rc_k . This procedure will be repeated for all $n_r R$ segment-to- D segment transformations.

13.6 Applications

In the previous sections, some major issues in fractal theory were presented. In the following, several applications of fractal theory, related to Farsi/Arabic document analysis, are described. On-line and off-line character recognition, font recognition, and watermarking applications will be addressed in this section.

13.6.1 On-Line and Off-Line Character Recognition

The recognition of handwritten alphanumeric characters is a challenging problem in pattern recognition. This is due to the large diversity of writing styles and the low image quality, especially in practical applications. Previous efforts to use fractals for Farsi/Arabic character recognition have mainly focused on feature extraction methods [12, 13].

The fractal codes extracted directly from the fractal coder can be used as a feature vector. Since the input image is scanned from top to bottom and from left to right, the starting point coordinate of the R block/segment (rsx_k, rsy_k) can be deduced by this systematic tracing procedure. Omitting the starting point, a feature vector with the length of $4 \times n$, is obtained for each input pattern [12]. Some of the Farsi/Arabic numerals and characters are only different in small regions. Quad-tree partitioning is a method of diving an image block according to its complexity. This partitioning method outperforms simple partitioning approaches in the field of Farsi/Arabic character recognition [13].

The Mapping Vector Accumulator (MVA) feature introduced by Linnell and De-ravi [9] records the angle and magnitude of the domain-range mapping vector. The matrix itself is an accumulator, where the angle and magnitude are first quantized and then the appropriate element of the accumulator is incremented.

The Domain-Range Co-Location Matrix (DRCLM) [9] is another fractal-based feature which measures levels of self-similarity in different parts of the image. It encapsulates information from the relative location of the domain block and its corresponding range block. In this method the image is divided into four equal-sized non-overlapping segments. When a mapping occurs from one segment to the other, then the entry at the corresponding cell in the matrix will be incremented. This is then repeated for all range blocks in the image.

After the fractal features are extracted as described above, they can be fed into traditional classifiers like neural networks or support vector machines. Mozaffari et al. explored the use of described fractal features for Farsi digits recognition [11]. Although fractal theory is used for feature extraction, it can also be utilized as the classifier. The inherent property of fractal theory based on the fixed point theorem of IFSs has also been exploited by some researchers; this is called fractal transformation. In this approach, the distortion between an input pattern and the pattern after one decoding iteration was used for classification [19]. According to the comparison made by Tan and Yan, this classifier outperformed others (HMM, PDBNN) in terms of error rate and training time for face recognition [18]. One drawback of this method is that its complexity is linear to the size of the database, which is not as much the case for neural networks.

This classifier also obviates the need for retraining of the whole database when addition or removal of a sample from the database occurs. The distortion between the input and decoded images after one iteration is highly affected by the size of the range blocks. Figure 13.6 shows the effect of range block size (N) on the fractal transformation. According to Fig. 13.6, when an input image with a size of 64×64 is coded with $N = 4$ or $N = 8$, the decoded images are almost the same for all

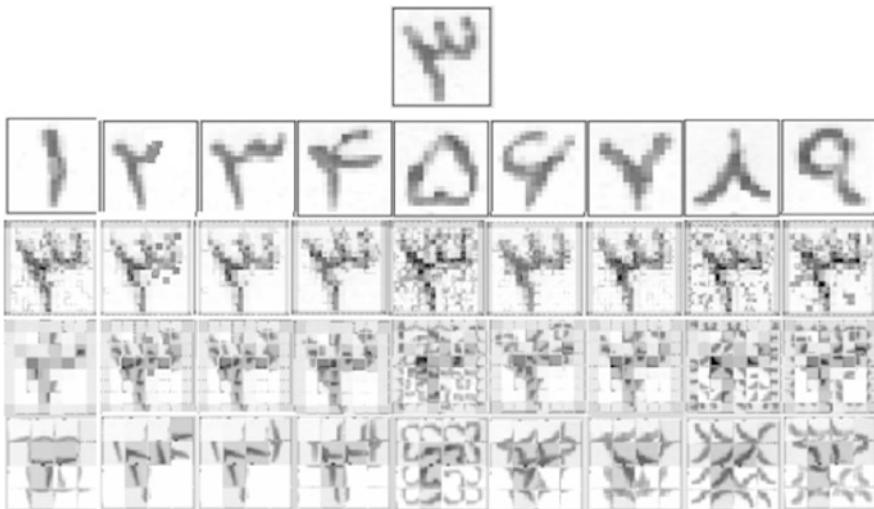


Fig. 13.6 Results of fractal transformation classifier for digit recognition. *First row:* Input image. *Second row:* Nine reference samples. *Third row:* Results of applying fractal codes of input image with $N = 4$ on second row images after one iteration. *Fourth row:* Results of applying fractal codes of input image with $N = 8$ on second row images after one iteration. *Fifth row:* Results of applying fractal codes of input image with $N = 16$ on second row images after one iteration

reference images after one decoding iteration. So the best choice for range block size is $N = 16$ using the fractal transformation classifier.

It is worth mentioning that the above methods for fractal feature extraction and classification can be easily utilized for on-line character recognition.

Comparing fractal-based features and classifiers with well-established character recognition algorithms is a difficult task. Drawing a comprehensive conclusion on benchmarking these methods requires large datasets and specific performance evaluation strategies, and these topics are beyond the scope of this chapter.

13.6.2 Font Recognition

A font is defined as a set of alphabets in the same family whose topological characteristics are standard in the printing industry. Font recognition is one of the associated topics in optical character recognition (OCR) and document image retrieval (DIR) systems. The performance of an OCR system degrades in the case of multi-font documents. Font identification can be used as a criterion for document filtering in DIR systems.

Ben Moussa et al. proposed a system based on fractal geometry features for Arabic font recognition [23]. They considered a document as a texture and regarded font recognition as a texture identification problem. They combined two fractal dimension features (box counting dimension (BCD) and dilation counting dimension

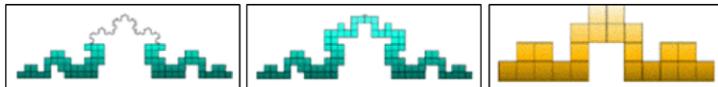


Fig. 13.7 General aspect of curve in BCD process [23]. (left) Recovery process of von Koch curve. (middle) Von Koch curve covered with r size. (right) Von Koch curve covered with $n \times r$ size

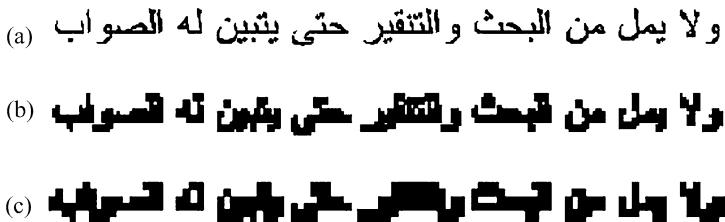


Fig. 13.8 Text image recovery by boxes [23]. (a) Recovery process of Von Koch curve. (b) Text image covered with r size. (c) Text image covered with $n \times r$ size

(DCD)) to obtain the main feature. With the use of this feature and a radial basis function (RBF) classifier, an average recognition rate of 98 % was reported.

The values of fractal dimension depend on the number of $N(r)$ intervals of length r for covering an object E , which is defined as:

$$FD(E) = \frac{\log(N(r))}{\log(\frac{1}{r})} \quad (13.15)$$

The BCD feature is calculated as follows:

$$BCD(r) = \lim_{r \rightarrow 0} \frac{\log(N(r))}{\log(\frac{1}{r})} \quad (13.16)$$

Figure 13.7 shows the general aspect of curve in the BCD process. Figure 13.8 shows how the BCD process is applied on documents. The DCD feature is defined as:

$$DCD(d) = \lim_{d \rightarrow 0} \left(n - \frac{\log(V(r))}{\log(d)} \right) \quad (13.17)$$

where d is the maximum dilation radius, n is the dimension of the space, and V is the dilation body. Figure 13.9 shows the general aspect of curve in the DCD process. Figure 13.10 shows a document under the DCD process.

13.6.3 Watermarking

With the advent of the Internet, multimedia data like text, image, video, and audio files are distributed on a large scale over the net. Digital watermarking, the process

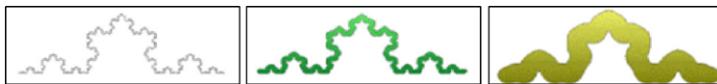


Fig. 13.9 General aspect of curve in DCD process [23]. (*left*) Original image. (*middle*) Von Koch curve after dilation with r level. (*right*) Von Koch curve after dilation with $2 \times r$ level

- (a) ولا يمل من البحث والتغیر حتى يتبيّن له الصواب
- (b) ولا يمل من البحث والتغیر حتى يتبيّن له الصواب
- (c) ولا يمل من البحث والتغیر حتى يتبيّن له الصواب

Fig. 13.10 Various dilation levels of text image [23]. (**a**) Original text image. (**b**) Text image after dilation with r level. (**c**) Text image after dilation with $2 \times r$ level

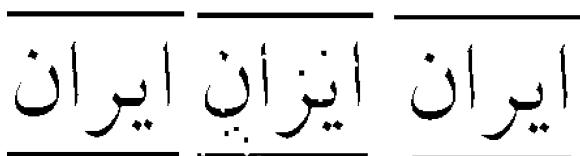


Fig. 13.11 Effect of disturbance on document. (*left*) Original text image. (*middle*) Imposing of some disturbance to the original image. (*right*) Imposing of the same amount of disturbance as (*middle*)

of hiding a watermark in a multimedia object without perceptual degradation, is a technique for copyright and ownership purposes.

Among the different multimedia data, documents need serious attention for digital watermarking. Due to both the low capacity and high sensitivity of the human visual system to small disturbances, document watermarking has become a challenging topic in document analysis. Figure 13.11 shows the sensitivity of documents to disturbances. Figures 13.11 (middle) and 13.11 (right) have the same number of pixels changed from black to white or vice versa. However, the disturbance is much more noticeable in Fig. 13.11 (middle). (In Fig. 13.11 (right), the upper horizontal line was enlarged while the lower horizontal line was shrunk.)

Pi et al. modified the classical fractal coding method in which the fractal affine transform is determined by the range block mean and contrast scaling [14]. They proposed a fractal watermarking approach in which the watermark is embedded in the range block means. Since the new fractal coding approach is mean-invariant, the range block mean is a suitable and secure place to hide a watermark. Figure 13.12 shows the fractal watermarking results.

To show the robustness of the fractal watermarking approach, some attacks have been imposed to the host image. Figures 13.13, 13.14, 13.15, 13.16 show the results.

Fig. 13.12 Results of fractal watermarking. (a) Host document image. (b) Watermark. (c) Document containing the watermark. (d) Decoded watermark

دانشگاه سمنان در سال ۱۳۵۳ خورسندی با نام مدرسه عالی دکنیولوژی سمنان آغاز به کار کرد. اکنون که حدود ۳۷ سال از عمر آن می‌گذرد با بیس از ۲۰۰ عضو هناب علمی، ۱۳ دانستکده و ۵۵ هزار دانشجو در مقاطع کارشناسی، کارشناسی ارشد و دکتری به نکی از هواکو مهند علمی ایران بدل سده است. دانشگاه سمنان در سالهای اخیر رسد حسنگری در اسناد اردهای علمی و نووهی داشته است.



(a) (b)

دانشگاه سمنان در سال ۱۳۵۳ خورسندی با نام مدرسه عالی دکنیولوژی سمنان آغاز به کار کرد. اکنون که حدود ۳۷ سال از عمر آن می‌گذرد با بیس از ۲۰۰ عضو هناب علمی، ۱۳ دانستکده و ۵۵ هزار دانشجو در مقاطع کارشناسی، کارشناسی ارشد و دکتری به نکی از هواکو مهند علمی ایران بدل سده است. دانشگاه سمنان در سالهای اخیر رسد حسنگری در اسناد اردهای علمی و نووهی داشته است.



(c) (d)

Fig. 13.13 Results of fractal watermarking. (a) Document containing the watermark. (b) Decoded watermark

دانشگاه سمنان در سال ۱۳۵۳ خورسندی با نام مدرسه عالی دکنیولوژی سمنان آغاز به کار کرد. اکنون که حدود ۳۷ سال از عمر آن می‌گذرد با بیس از ۲۰۰ عضو هناب علمی، ۱۳ دانستکده و ۵۵ هزار دانشجو در مقاطع کارشناسی، کارشناسی ارشد و دکتری به نکی از هواکو مهند علمی ایران بدل سده است. دانشگاه سمنان در سالهای اخیر رسد حسنگری در اسناد اردهای علمی و نووهی داشته است.



(a) (b)

Fig. 13.14 Effect of noise on the fractal watermarking. (a) Document with pepper and salt noise with $\sigma = 0.01$. (b) Decoded watermark. (c) Document with pepper and salt noise with $\sigma = 0.1$. (d) Decoded watermark

دانشگاه سمنان در سال ۱۳۵۳ خورسندی با نام مدرسه عالی دکنیولوژی سمنان آغاز به کار کرد. اکنون که حدود ۳۷ سال از عمر آن می‌گذرد با بیس از ۲۰۰ عضو هناب علمی، ۱۳ دانستکده و ۵۵ هزار دانشجو در مقاطع کارشناسی، کارشناسی ارشد و دکتری به نکی از هواکو مهند علمی ایران بدل سده است. دانشگاه سمنان در سالهای اخیر رسد حسنگری در اسناد اردهای علمی و نووهی داشته است.



(a) (b)

دانشگاه سمنان در سال ۱۳۵۳ خورسندی با نام مدرسه عالی دکنیولوژی سمنان آغاز به کار کرد. اکنون که حدود ۳۷ سال از عمر آن می‌گذرد با بیس از ۲۰۰ عضو هناب علمی، ۱۳ دانستکده و ۵۵ هزار دانشجو در مقاطع کارشناسی، کارشناسی ارشد و دکتری به نکی از هواکو مهند علمی ایران بدل سده است. دانشگاه سمنان در سالهای اخیر رسد حسنگری در اسناد اردهای علمی و نووهی داشته است.

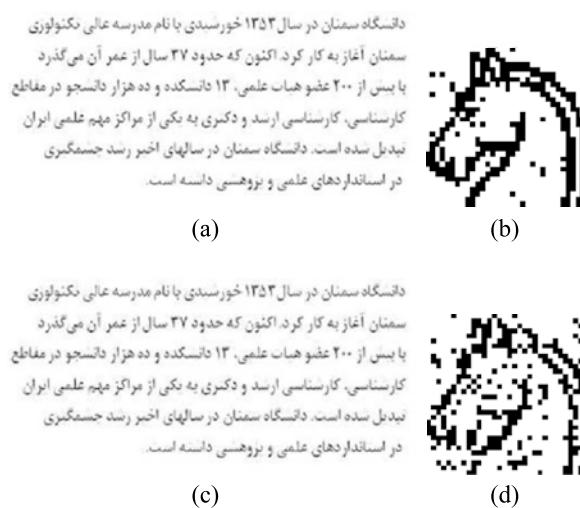


(c) (d)

Fig. 13.15 Effect of occlusion on the fractal watermarking. (a) Document with 25 % occlusion. (b) Decoded watermark. (c) Document with 50 % occlusion. (d) Decoded watermark



Fig. 13.16 Effect of JPG compression on the fractal watermarking.
(a) Compressed document with quality factor $Q = 70$.
(b) Decoded watermark.
(c) Compressed document with quality factor $Q = 40$.
(d) Decoded watermark



Most of the previous efforts on data hiding were focused on gray level document images, in which the host document is regarded as an ordinary gray level image and conventional watermarking algorithms are utilized. Although this scheme has a high data hiding capacity, it suffers from a lengthy fractal coding process.

As an alternative point of view, a document can be regarded as a binary image, which is the most common type in the archives. In this manner, the number of fractal parameters that must be computed and stored can be reduced. Furthermore, since the numbers of possibilities for the pixel values in range/domain blocks are restricted, a lookup table containing predefined parameters can reduce the computational burden. As a result, the fractal coding time, especially for large documents, will be

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mtaer in waht oredr the lteers in a wrd are, the olny iprmoeent tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a tottl mses and you can stil rae it wouthit porbelm. Tths is bcuseae the huamn mnid deos not raed ervey Iteter by istlef, but the wrd as a whole.

(a)

چنانچه به طور رومزره به زبان فارسی صبحت می کنند، خواهد توافنت این نوشه را بخواهند. در داشنگاه کمپریج انگلستان تحقیقی روی روش خواندن شدن کلمات در مغز اجتنم شده است که مخصوص می کند که مغز انسان تهناحروف ایدها و انتهای کلمات را پذراش کرده و کلمه را می خواند. به همین دلیل است که با وجود به هم ریتختگی این نوشه شما تواستید آنرا بخواهید

(b)

Fig. 13.17 Texts including incorrect words in which only the first and the last letters are in the right place. But humans can still read them without a problem. (a) English text. (b) Farsi text

considerably diminished. Note that the proposed binary image fractal coding methods for non-text images cannot be employed directly for document images. Data hiding in documents seems to be a paradox. As Fig. 13.11 shows, binary document images are very delicate and sensitive to modifications. On the other hand, humans, especially experienced readers, do not read words letter by letter. The MRC Cognition and Brain Sciences Unit at Cambridge University presented some interesting issues relating to the human mind. Figure 13.17 shows these results.

The preceding challenges on methods of watermarking using document images are some of the fascinating topics on which our research group in the Electrical and Computer Department at Semnan University are working. The results will be published in the future.

13.7 Conclusions

In this chapter some basic issues of fractal theory such as fractal dimension and fractal coding and decoding were presented. Then, several applications of fractal theory in Farsi/Arabic character recognition, font recognition, and digital watermarking were studied.

References

1. Athale, S.S., Mitra, S.K., Banerjee, A.: A fractal based approach for face recognition. In: Proc. IEEE Conf. Intelligent Sensing and Information Processing, pp. 170–174 (2005)
2. Distasi, R., Nappi, M., Tucci, M.: Using fractal encoding for image indexing. In: Proceedings of International Conference on Image Analysis and Processing, pp. 975–980 (1999)
3. Ebrahimpour-Komleh, H., Chandran, V., Sridharan, S.: Face recognition using fractal codes. In: Proceedings of the International Conference on Image Processing (ICIP), Thessaloniki, Greece, October 2001, vol. 3, pp. 58–61 (2001)
4. Eiterer, L.F., Facon, J., Menoti, D.: Postal envelope address block location by fractal-based approach. In: Proc. of the XVII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI/SIACG), vol. 1, pp. 90–97 (2004)
5. Fisher, Y.: Fractal Image Compression: Theory and Applications. Springer, Berlin (1995)

6. Freche, T., Vincent, N.: Fractal analysis of a handwritten line by profile approximation. In: Proc. of the IEE Third European Workshop on Handwriting Analysis and Recognition, pp. 20/1–20/6 (1998)
7. Jacquin, A.E.: Image coding based on a fractal theory of iterated contractive image transform. *IEEE Trans. Image Process.* **8**, 18–30 (1992)
8. Khelifi, B., Zaghdén, N., Alimi, M.A., Mullot, R.: Unsupervised categorization of heterogeneous text images based on fractals. In: Proceedings of the 19th International Conference on Pattern Recognition (ICPR) (2008)
9. Linnell, T.A., Deravi, F.: Novel fractal domain features for image classification. In: Proc. of the International Conference on Visual Information Engineering, July 2003, pp. 33–36 (2003)
10. Mandelbrot, B.B.: *The Fractal Geometry of Nature*. Freeman, New York (1983)
11. Mozaffari, S., Faez, K., Kanan, H.R.: Feature comparison between fractal codes and wavelet transform in handwritten alphanumeric recognition using SVM classifier. In: Proceedings of the 17th International Conference on Pattern Recognition (ICPR), pp. 331–334 (2004)
12. Mozaffari, S., Faez, K., Kanan, H.R.: Recognition of isolated handwritten Farsi/Arabic alphanumeric using fractal codes. In: Proc. of the 6th IEEE Southwest Symposium on Image Analysis and Interpretation, pp. 104–108 (2004)
13. Mozaffari, S., Faez, K., Ziaratban, M.: Character representation and recognition using quad tree-based fractal encoding scheme. In: Proceedings. Eighth International Conference on Document Analysis and Recognition (ICDAR), pp. 819–823 (2005)
14. Pi, M.H., Li, C.H., Li, H.: A novel fractal image watermarking. *IEEE Trans. Multimed.* **4**, 488–499 (2006)
15. Roy, K., Majumder, K.: Trilingual script separation of handwritten postal document. In: Proceedings of the Sixth Indian Conference on Computer Vision, Graphics & Image Processing (ICVGIP), pp. 693–700 (2008)
16. Ruiz, M.D., Bailón, A.B.: Summarizing structured documents through a fractal technique. In: Lecture Notes in Business Information Processing, vol. 12, pp. 328–340 (2009)
17. Svoboda, T., Kybic, J., Hlavac, V.: *Image Processing, Analysis & Machine Vision—A MATLAB Companion*. Thomson Learning, Toronto (2008)
18. Tan, T., Yan, H.: Face recognition by fractal transformations. In: Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (1999)
19. Tan, T., Yan, H.: Signature verification using fractal transformation. In: International Conference on Pattern Recognition (ICPR), pp. 851–854 (2000)
20. Tao, Y., Tang, Y.Y.: Discrimination of oriental and Euramerican scripts using fractal feature. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR), Seattle, WA, USA, September 2001, pp. 1115–1119 (2001)
21. Wang, L., Ma, J., Wang, X.: Research on address tactic of Internet documents' semantization based on fractal theory. In: Proceedings of the 1st International Conference on Information Science and Engineering (ICISE), pp. 938–941 (2009)
22. Yu, C.L., Tang, Y.Y., Suen, C.Y.: Document skew detection based on the fractal and least squares method. In: Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR), pp. 1149–1152 (1995)
23. Zahour, A., Ben Moussa, S., Benabdellahid, A., Alimi, A.M.: New features using fractal multi-dimensions for generalized Arabic font recognition. *Pattern Recognit. Lett.* **31**, 361–371 (2010)
24. Zhang, J., He, Z., Cheung, Y.-M., You, X.: Writer identification using a hybrid method combining Gabor wavelet and mesh fractal dimension. In: Lecture Notes in Computer Science, vol. 5788, pp. 535–542 (2009)

Chapter 14

Multi-stream Markov Models for Arabic Handwriting Recognition

Youssi Kessentini, Thierry Paquet, and AbdelMajid Ben Hamadou

Abstract In this chapter, we describe an off-line unconstrained Arabic handwritten word recognition system based on a multi-stream segmentation-free HMM. The proposed system proceeds without explicit segmentation of handwriting into graphemes and makes use of low level feature sets. Features are combined according to the multi-stream paradigm, providing a convenient formalism to asynchronously combine several information sources, using cooperative Markov models. A two-level decoding algorithm is proposed to reduce the complexity. The system has been tested on the IFN/ENIT database, and the results show significant improvement for the multi-stream approach compared to the performances reported recently on the same database.

14.1 Introduction

Arabic is the second most widely used alphabet around the world, after the Latin script. However, Arabic handwriting recognition systems have not been studied in the same proportion by the research community. During recent years, Arabic handwriting recognition has received much more attention due to the development of new databases [16] and the organization of international competitions [13, 14] allowing one to compare the performances of the systems and to promote research development. Arabic script is based on an alphabet and rules different from those of Latin. In fact, Arabic script is written from right to left, and includes 28 basic

Y. Kessentini (✉) · T. Paquet

Laboratoire LITIS EA 4108, Université de Rouen France, Site du Madrillet, 76800 Saint-Etienne du Rouvray, France

e-mail: yousri.kessentini@univ-rouen.fr

T. Paquet

e-mail: thierry.paquet@univ-rouen.fr

A. Ben Hamadou

Laboratoire MIRACL, Université de Sfax Tunisie, Route de Tunis km10, B.P. n 242, 3021 Sfax, Tunisia

e-mail: abdelmajid.benhamadou@isimsf.rnu.tn

letters. There is no difference between upper and lower case characters. The character shape is context sensitive; i.e., it depends on its position within a word. The shapes are dependent on the four positions: beginning of a (sub)word, middle of a (sub)word, end of a (sub)word, and in isolation. Additional small markings called “diacritics” (e.g., dots, Hamza, and chadda) compose the Arabic alphabet. Some characters may have exactly the same main shape, and are distinguished from each other only by the presence or the absence of these diacritics, their number, or their position with respect to the main shape. Some combinations of two or three letters have special shapes called “ligatures,” which work exactly like Latin ligatures such as œ and æ. These multiple configurations give a total of 170 different character shapes, as compared to the 52 different character shapes of Latin script. Moreover, Arabic words are segmented into parts of Arabic words (PAWs), each consisting of a group of letters. A word is composed of one or more PAWs. A more detailed description of the characteristics of Arabic script is given in [12, 25].

Despite the differences between Arabic and Latin scripts, they are both alphabetic. This main property allows one to build recognition systems based on character models that are further concatenated to build word models. Stochastic models, especially hidden Markov models (HMMs) [18], have been widely applied to Arabic handwriting recognition in recent years [1, 9, 21]. In fact, HMMs have a huge capacity to integrate contextual information and to absorb the variability. A particular approach known as multi-stream HMMs has been proposed in our previous works [7, 8]; it consists of the asynchronous combination of several information sources, using cooperative Markov models. In this paper, we describe the architecture of our multi-stream recognition system and we present its optimized implementation using a two-level decoding scheme.

This paper is organized as follows. Section 14.2 presents the different information combination strategies. Section 14.3 defines the multi-stream framework and the new decoding algorithm. It also includes a discussion about its complexity. Section 14.4 presents the pre-processing and the feature extraction steps. Finally, Sect. 14.5 provides experimental results, followed by a conclusion.

14.2 Information Combination Strategies

Information combination is a widely used technique in handwriting recognition to improve the classification rates, or to preserve them when dealing with more difficult vocabularies or scripts. This is why various combination strategies have been proposed in the literature [5, 11, 24]. They can be grouped into two broad categories: feature fusion methods and decision fusion techniques.

The first category, commonly known as early integration [15], consists in the concatenation of the input feature sets (or feature streams) into a unique large feature space, and subsequently uses a traditional HMM classifier to model the combined observations in the unique feature space (see Fig. 14.1).

In contrast, decision fusion, known as late integration [17], consists in combining multiple classifier outputs (decisions). Different feature representations obtained

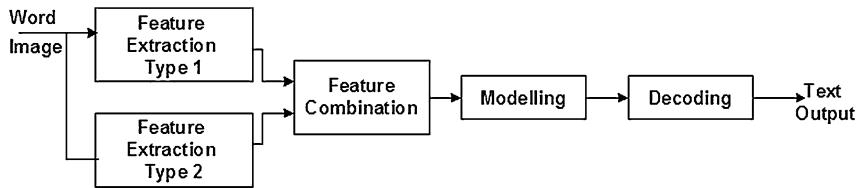


Fig. 14.1 Feature combination approach

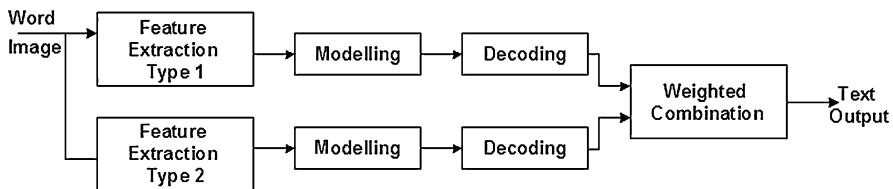


Fig. 14.2 Decision combination approach

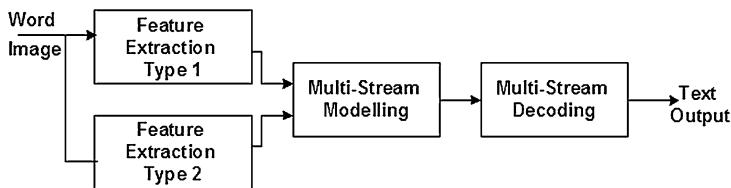


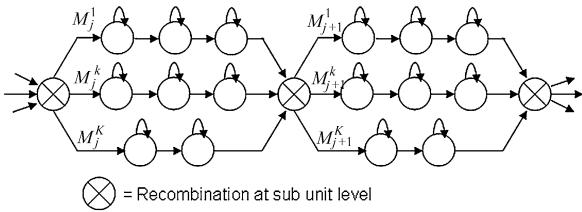
Fig. 14.3 Multi-stream combination approach

from the word image are modeled and decoded separately by individual HMM classifiers. The decoded outputs are then combined to get the final text output (see Fig. 14.2).

In [2], the authors compare these two combination methods in the case of off-line handwritten text line recognition and show that both combination methods improve recognition performances compared to any recognizers built from the individual feature streams. Furthermore, in their case, the early integration approach outperforms the decision level combination.

A particular method within the decision fusion framework of sequence models falls into the multi-stream hidden Markov model paradigm (see Fig. 14.3). This approach has been particularly studied in the domain of automatic speech recognition (ASR) [3, 23]. It offers a means to merge different independent sources of information at subunit levels, allowing an asynchronous modeling of streams.

Fig. 14.4 General form of K-stream model with anchor points between subunit models



14.3 Multi-stream Formalism

The multi-stream formalism is an adaptive method to combine several individual feature streams using cooperative Markov models. This problem can be formulated as follows. Assume an observation sequence X composed of K input streams X^k ($k = 1, \dots, K$) representing the utterance to be recognized, and assume that the hypothesized model M for an utterance is composed of J subunit models M_j ($j = 1, \dots, J$) associated with the subunit level at which we want to perform the recombination of the input streams (e.g., characters). To process each stream independently of each other up to the defined subunit level, each subunit model M_j is composed of K models M_j^k (possibly with different topologies). Recombination of the K stream models M_j^k is forced at some temporal anchor states (\otimes in Fig. 14.4). The resulting statistical model is illustrated in Fig. 14.4. A detailed discussion of the mathematical formalism is given in our previous work [8].

The recognition problem can be formulated as one of finding the word model M^* that maximizes the posterior probability given a sequence of observations X :

$$M^* = \operatorname{argmax}_{M \in \Theta} P(M|X) \quad (14.1)$$

where Θ is the set of all possible word hypotheses.

The Bayes formula gives:

$$M^* = \operatorname{argmax}_{M \in \Theta} \frac{P(X|M)P(M)}{P(X)} \quad (14.2)$$

$P(X)$ being independent of the model M , it can therefore be ignored for the computation of M^* . When we assume equal prior probabilities $P(M)$ for all possible word hypotheses, then the recognition problem consists in determining the model M^* that maximizes the likelihood $P(X|M)$.

Using subunit decomposition gives:

$$P(X|M) = \prod_{j=1}^J P(X_j|M_j) \quad (14.3)$$

Assuming that each stream is independent, each submodel likelihood can be computed as a combination of the K-stream likelihood using a combination function f and a set of weighting parameters W , as depicted in Eq. (14.4):

$$P(X|M) = \prod_{j=1}^J f(W, P(X_j^k|M_j^k)) \quad (14.4)$$

Most of the approaches use a linear weighted combination function of log likelihood as follows:

$$\log P(X|M) = \sum_{j=1}^J \sum_{k=1}^K W_j^k P(X_j^k|M_j^k) \quad (14.5)$$

In practice, different combination rules have been proposed for multi-stream systems, including linear (sum, product), nonlinear (MLP), or other (maximum, minimum, median...) rules. More details are presented in [6]. A linear combination of rules requires the estimation of the stream weights according to their relative reliability. Many weighting strategies are proposed in the literature, including fixed weights, which have to be trained prior to application, and adaptive weights, which are estimated during recognition [6].

Having described the general multi-stream formalism, in the following sections we present how to decode and train these models.

14.3.1 Multi-stream Decoding

Decoding multi-stream models requires a more sophisticated procedure than the Viterbi search. Two different algorithms have been proposed to solve the problem of decoding.

HMM-Recombination Algorithm

Here, a composite (or product) HMM is built by merging an n-tuple of states from the n-stream HMMs [3]. The topology of this composite model is defined so as to represent all possible state paths given the initial HMM topologies. Figure 14.5 shows a multi-stream HMM with 2 streams and its corresponding product HMM. The product HMM parameters are determined as follows. The transition probabilities of the product HMM are derived from the transition probabilities of the 2 single-stream HMMs assuming independence of the models between two recombination states. For example:

$$P(a - B|a - A) = P(a|a) \times P(B|A)$$

The conditional observation likelihood of the composite HMM is obtained using a combination of the observation likelihoods of the single-stream components, for example:

$$P(X^1(t), X^2(t)|a - A) = P(X^1(t)|a)^W P(X^2(t)|A)^{1-W}$$

where $X^1(t)$ (similarly, $X^2(t)$) is the observation vector corresponding to stream 1 (similarly, stream 2) and W is the reliability of stream 1 ($0 \leq W \leq 1$). Decoding under such a model requires computing a single best path using the well-known Viterbi decoding algorithm. We have demonstrated in [8] that this algorithm has a large complexity, especially when dealing with a large number of streams. An alternative consists in using the two-level decoding algorithm.

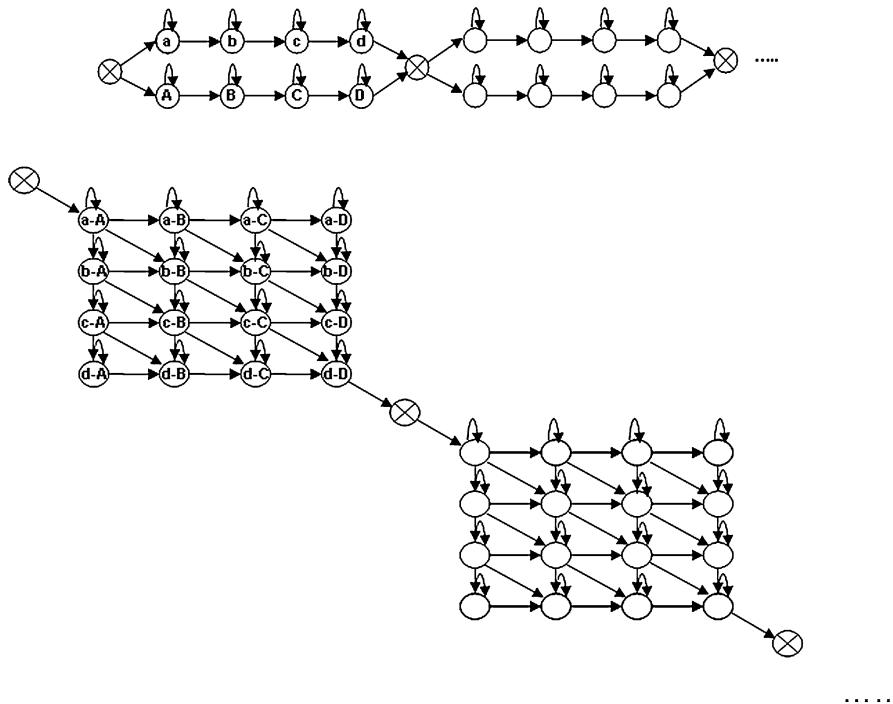


Fig. 14.5 Example of a multi-stream HMM with 2 streams and its corresponding product (composite) HMM

Two-Level Dynamic Programming

This decoding strategy was initially proposed in [20] to optimize the Viterbi algorithm in the case of large vocabulary recognition problems. The decoding takes place in two steps. First, a dynamic programming process is applied at the subunit level (phoneme or character models), and each submodel is scored on arbitrary portions of the frame data. Second, submodels are merged together in order to find the best overall score, during a second dynamic programming stage at word level.

In the case of multi-stream HMMs, the first level of this algorithm is slightly modified, and each stream HMM is independently decoded for each possible portion of the frame data; the individual stream scores are then combined for further use at the second level. In the second level, only the character boundaries are decoded without the necessity of going through the HMM states. The details of the first level are presented in Algorithm 1.

The first decoding level can be viewed as a standard Viterbi algorithm which is used to decode the best character alignment for all possible positions of the character within the observation stream. The last step of the first level consists in combining the different stream scores using a combination function as a weighted sum of log-likelihoods. Once the scores are computed for all character stream models, they can

- 1: For each stream S_k and each observation stream $X_t^k (k = 1, \dots, K)$
 - 2: For each character model $M_j^k (j = 1, \dots, C)$
 - 3: For each beginning frame $b, (b = 1, \dots, T - 1)$
 - 4: For each end frame $e, (e = b + 1, \dots, T)$ compute the best score corresponding to the best state sequence.
 - 5: Store the obtained likelihood in $\Psi(M_j^k, b, e)$
 - 6: Combine the stream likelihoods using a combination function f ,
- $$\Psi(M_j, b, e) = f(\Psi(M_j^1, b, e), \dots, \Psi(M_j^K, b, e))$$

Algorithm 1: Two-level dynamic programming: First level

be reused to decode any lexicon; hence we avoid repeating character decoding for each hypothesized occurrence of the character at the same position while decoding each word of the lexicon. Given these scores, the second level of the computation pieces together the individual character scores to maximize the overall accumulated score over the entire word. This can be accomplished using dynamic programming as follows:

$$L(c_1, \dots, c_l, e) = \max_{1 \leq b \leq e} [L(c_1, \dots, c_{l-1}, b-1) \times \Psi(c_l, b, e)]$$

where $L(c_1, \dots, c_l, e)$ is the score of the best path ending at frame e using the character sequence c_1, c_2, \dots, c_l . The best path ending at frame e using exactly l character models is the one with maximum score over all possible beginning frames, b , of the concatenation of the best path ending at frame $b-1$ using exactly $l-1$ character models fold the best score of the character model c_l from frame b to frame e (calculated at the first level). Therefore, the second level consists of the procedure in Algorithm 2.

- 1: For each word in the lexicon
- 2: For each character composing the word
- 3: For each end frame $e (2 \leq e \leq T)$
- 4: Compute the score of the best concatenation of character until frame e :

$$L(c_1, c_2, \dots, c_l, e) = \max_{1 \leq b \leq e} [L(c_1, c_2, \dots, c_{l-1}, b-1) \times \Psi(c_l, b, e)]$$

Algorithm 2: Two-level dynamic programming: Second level

14.3.2 Computational Complexity

We compare the computational complexity of the two decoding algorithms. In the case of the two-level decoding algorithm, the first level is independent of the lexicon size, and its computational complexity is given by $\mathcal{O}(T^2 N^2 KC)$, where T is the length of a sequence of observations, N the number of states per character model,

Table 14.1 Variation of the operations number ($\times 10^4$) with respect to the number of streams K , $T = 100$, $L = 5$, $N = 4$, $V = 100$, $C = 26$

	Two-level	HMM-recombination
$K = 2$	1332	6400
$K = 3$	1748	102400
$K = 4$	2164	1638400
$K = 5$	2580	26214400

Table 14.2 Variation of the operations number ($\times 10^4$) with respect to the lexicon size V , $T = 100$, $L = 5$, $N = 4$, $K = 3$, $C = 26$

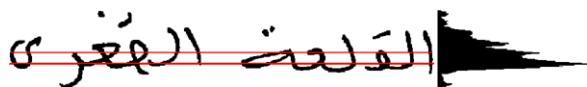
	Two-level	HMM-recombination
$V = 100$	1748	102400
$V = 1000$	6248	1024000
$V = 2000$	11248	2048000
$V = 5000$	26248	5120000

C the number of character models, and K the number of streams. In the second level, the computation depends on the word in the lexicon, and the complexity is $\mathcal{O}(T^2 LV)$, where L is the average length of the words in the lexicon and V is the number of words in the lexicon. The approximate computational complexity for the two-level decoding algorithm is then $\mathcal{O}(T^2 N^2 KC + T^2 LV)$.

In the case of the HMM-recombination algorithm, the Viterbi algorithm is applied to decode the product HMM. The complexity of the Viterbi algorithm is $\mathcal{O}(T(LN)^2 V)$. In the product HMM, the number of states increases to LN^K . Therefore, the complexity of the HMM-recombination algorithm becomes $\mathcal{O}(T(LN^K)^2 V) = \mathcal{O}(TL^2 N^{2K} V)$.

Note that by just looking at the complexity expressions of the two algorithms, it is hard to see which strategy is better. To get a feeling for the computational complexity of each decoding strategy, typical values of $T = 100$, $L = 5$, $N = 4$, $K = 4$, $V = 100$, $C = 26$ result in $\mathcal{O}(16384 \times 10^7)$ for the HMM-recombination algorithm, and $\mathcal{O}(2.164 \times 10^7)$ for two-level algorithm. Tables 14.1 and 14.2 present, respectively, the variation of the approximate computational complexity of the two algorithms with respect to the number of streams K and the lexicon size V . By analyzing the values in these tables, it is possible to have a better idea of the computational complexity of each decoding strategy. It is clear that the two-level algorithm is more advantageous when dealing with a large lexicon size and a great number of streams.

Note that we can reduce the complexity of the two-level algorithm. In fact, T^2 can be reduced to $T(T - D)$, where D is an estimation of the duration of the character models, without loss of accuracy. The complexity of the first level can also be reduced assuming that the same HMM topology is used for all streams. In this case, all stream character HMMs are simultaneously decoded, and the complexity is reduced to $T^2 N^2 C$. The overall complexity then becomes $\mathcal{O}(T(T - D)(N^2 C + LV))$.

Fig. 14.6 Baseline detection

14.3.3 Multi-stream Training

Training the multi-stream HMM consists of two tasks: the first task is the estimation of its HMM stream component parameters (mixture weights, means, variances, and state transition probabilities), and the second task is the estimation of appropriate stream exponents. Maximum likelihood parameter estimation by means of the expectation maximization (EM) algorithm can be used in a straightforward manner to train the first set of parameters. This can be done in two ways: either by training each stream component parameter set separately, based on single-stream observations, and subsequently combine the resulting single-stream HMMs, or by training the entire parameter set (excluding the exponents) at once using the multi-modal observations. The second training step concerns the optimization of the stream weights. Two different strategies have been investigated in [8], and it was shown that both of the weighting strategies perform similarly.

14.4 Recognition System Architecture

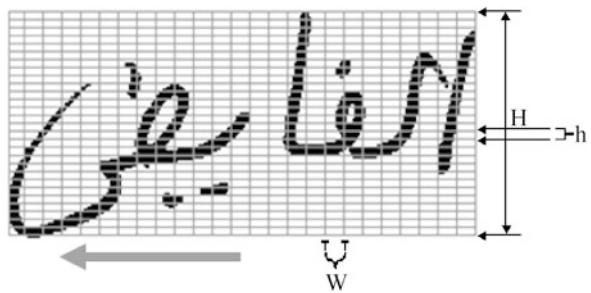
The proposed system is based on multi-stream models for the recognition of Arabic handwritten words. In the following subsection, we describe the different stages of our approach. In the first step, pre-processing is applied to the word image. Two types of features are considered in this work: (i) contour-based features and (ii) density-based features. Contour-based features are extracted from the lower and the upper contours, and density-based features are computed on two different sliding windows with different widths.

14.4.1 Pre-processing

Pre-processing is applied to word images in order to eliminate noise and to simplify the procedure of feature extraction.

- Normalization: In an ideal model of handwriting, a word is supposed to be written horizontally, with ascenders and descenders aligned along the vertical direction. In real data, such conditions are rarely respected. We use slant and slope correction so as to normalize the word image [10].
- Contour smoothing: Smoothing eliminates small blobs on the contour.
- Baseline detection: Our approach uses the algorithm described in [22] based on the horizontal projection curve that is computed with respect to the horizontal pixel density (see Fig. 14.6). The baseline position is used to extract baseline-dependent features that emphasize the presence of descenders and ascenders.

Fig. 14.7 Word image divided into vertical frames and cells



14.4.2 Feature Extraction

An important task in multi-stream combination is to identify features that carry complementary information. In order to build the feature vector sequence, the image is divided into vertical overlapping windows or frames. The sliding window is shifted along the word image from right to left, and a feature vector is computed for each frame.

Two feature sets are proposed in this work. The first one is based on directional density features. These kinds of features, initially proposed for Latin script [10], have proved to be discriminative for Arabic script [8]. The second feature set is based on foreground (black) pixel densities [4].

14.4.3 Density Features

Here we recall the definition proposed in [4]. From each frame 26 features are extracted for a window of 8-pixel width (and 32 features for a window of 14-pixel width). There are two types of features: features based on foreground (black) pixel densities, and features based on concavity. In order to compute some of these features (for example, f_2 and f_{15} as described next) the window is divided into cells where the cell height is fixed (4 pixels in our experiments) as presented in Fig. 14.7. For each frame t , the features are the following:

- f_1 : density of foreground (black) pixels.
- f_2 : number of transitions between two consecutive cells of different density levels.
- f_3 : difference in y position of gravity centers of foreground pixels in the current frame and in the previous one.
- f_4-f_{11} : densities of black pixels for each vertical column of pixels in each frame (note that the frames here are of 8-pixel width).

The next features depend on the baseline position:

- f_{12} : vertical position of the center of gravity of the foreground pixels in the whole frame with respect to the lower baseline.

Fig. 14.8 Five types of concavity configurations for a background pixel P

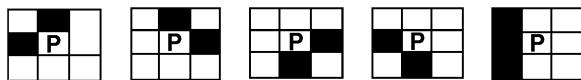
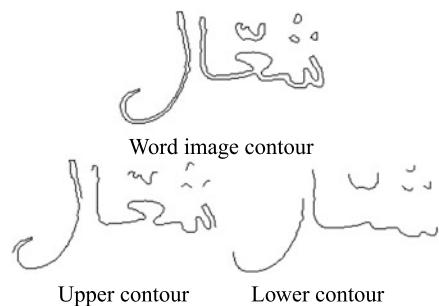


Fig. 14.9 Word image contours



- $f_{13}-f_{14}$: density of foreground pixels over and under the lower baselines for each frame.
- f_{15} : number of transitions between two consecutive cells of different density levels above the lower baseline.
- f_{16} : zone to which the gravity center of black pixels belongs with respect to the upper and lower baselines (above upper baseline, a middle zone, and below lower baseline).
- $f_{17}-f_{26}$: five concavity features in each frame and another five concavity features in the core zone of a word, that is, the zone bounded by the upper and lower baselines. They are extracted by using a 3×3 grid as shown in Fig. 14.8.

14.4.4 Contour Features

These features are extracted from the word contour representation. Each word image is represented by its lower and upper contours (see Fig. 14.9). A sliding window is shifted along the word image. Two parameters characterize a window: window width (8 pixels) and window overlap between two successive positions (5 pixels). For each position of a window, we extract the upper contour points (similarly, the lower contour points). For every point in this window, we determine the corresponding Freeman direction, and the direction points are accumulated in the directional histogram (8 features).

In addition to the directional density features, a second feature set is computed at every point of the upper contour (similarly, it is done for every point on the lower contour). The last (black) point (say, p^*) in the vertical black run started at an upper contour point (say, p) is considered and, depending on the location of p^* , one of four situations may arise. The point (p^*) can belong to a:

- Lower contour (see corresponding p points marked red in Fig. 14.10).
- Interior contour on closure (see blue points in Fig. 14.10).

Fig. 14.10 Contour feature extraction



- Upper contour (see yellow points in Fig. 14.10).
- No point found (see green points in Fig. 14.10).

The black points in Fig. 14.10 represent the lower contour.

The histogram of the four kinds of points is computed in each window. This second feature set provides additional information about the structure of the contour like the loops, the turning points, the simple lines, and the end points on the word image (altogether, four different features).

The third feature set indicates the position of the upper contour (similarly, lower contour) points in the window. For this purpose, we localize the core zone of the word image. More precisely, we extract the lower and upper baselines of word images. These baselines divide the image into three zones: (1) a middle zone, (2) the lower zone, (3) the upper zone. This feature set (3 features) provides additional information about the ascending and the descending characters, which are salient characteristics for recognition of Arabic script. Hence, in each window we generate a 15-dimensional (8 features from chain code, 4 features from the structure of the contour, and 3 features from the position of the contour) contour (for upper or lower contour) based feature vector.

14.4.5 Character Models

In order to model the Arabic characters, we built up to 159 character HMMs. An Arabic character may actually have different shapes according to its position within the word (beginning, middle, end word position). Other models are specified with additional marks such as “shadda.” Each character HMM is composed of four emitting states. The observation probabilities are modeled with Gaussian mixtures (three per state). Embedded training [19] is used where all character models are trained in parallel using the Baum–Welch algorithm applied on word examples. The system builds a word HMM by concatenation of the character HMM corresponding to the word transcription of the training sample.

14.5 Experiments and Results

To evaluate the performance of our recognition system, experiments have been conducted on the IFN/ENIT benchmark database [16]. This database contains a total of 32492 Arabic handwritten words of 937 Tunisian town/villages names written by more than 1000 writers. Some town/village names occur in the database

Table 14.3 Recognition performance using single-stream features

Models	Top 1	Top 2	Top 5	Top 10
(1) Upper contour	70.5	78.6	86.3	90.4
(2) Lower contour	63.5	73.1	82.6	86.4
(3) Density1	65.1	73	80.6	83.2
(4) Density2	68.7	78.1	83.3	86.9

Table 14.4 2-stream HMM recognition performances

Models	Top 1	Top 2	Top 5	Top 10
2-stream 1-2	75.3	83	89.1	92
2-stream 1-3	76.3	84.3	90.5	93.3
2-stream 1-4	79.6	85.7	91.6	94.5
2-stream 2-3	75.5	82.9	89	92
2-stream 2-4	77.2	84.8	90.8	93.6
2-stream 3-4	73.8	81.9	88.3	91.1

with slightly different writing styles according to, e.g., the presence or absence of “shadda.” It follows that our lexicon is made of about 2100 valid entries. Four different sets (a, b, c, d) are predefined in the database for training and one set (e) for testing.

Table 14.3 shows the experimental results of the performance of our recognition system using four different single streams (upper contour, lower contour, and density with two windows varying in their widths; Density1 and Density2 correspond to the windows of 8-pixel and 14-pixel widths, respectively) as a function of the size of the list of word hypothesis. The best recognition rate is 70.5 % obtained using the upper contour feature. From these results it appears that the upper contour is significantly better than the three other feature streams for the recognition of Arabic script.

To improve the performance given in Table 14.3, we try to combine the 4 single streams according to the multi-stream formalism. Six possible pairs of streams can be formed. Here the multi-stream two-level decoding algorithm is used. The recognition results of the 2-stream HMM are presented in Table 14.4. We obtain exactly the same performances presented in [8] when using the HMM-recombination algorithm. The main advantage of the two-level algorithm is to reduce the computational complexity, as explained in Sect. 14.3.2.

In all these experiments, we notice that the multi-stream approach improves the performance obtained with any of the single-stream HMMs. The best 2-stream recognition rate is 79.6 % in Top 1 and is obtained by combining upper contour and Density2 features. The gain is 9.1 % compared to the best single-stream recognition rate. Also combining density and contour feature streams gives a better performance than combining two contour streams or two density streams.

To compare the multi-stream approach to the standard combination strategies, namely fusion of features and fusion of decisions, we report the best 2-stream result

Table 14.5 Multi-stream results vs. decision and feature fusion approaches

Models	Top 1	Top 2	Top 5	Top 10
2-stream	79.6	85.7	91.6	94.5
Decision fusion	75.4	83.2	89.5	92.2
Feature fusion	74.1	82.6	88.4	90.8

obtained by combining the features corresponding to the upper contour and Density2. As shown in Table 14.5, the multi-stream approach performs better than the two other standard combination strategies of each individual stream model.

In order to compare our results to the most recent works presented in the literature, we have participated in the international competition in Arabic handwriting recognition systems at ICDAR 2009 [14]. 17 different Arabic handwriting recognition systems have been tested using datasets f and s, which are unknown to all participants. During this competition, our recognition system achieved a recognition rate of 82.09 % on set f and 74.51 % on set s. These performances on set s correspond to the third best system among 17 participating systems. The most important results of this competition are given in [14]. Note that set s was collected in the United Arab Emirates while all training data comes from Tunisia, which shows that the generalization ability of our recognition system is interesting.

14.5.1 Recognition Time

The recognition time is defined as the time in seconds required to recognize one word. It is measured in CPU-seconds, which is the time for which the recognition process has exclusive use of the central processing unit of a computer with a multitasking operating system. In this part, the recognition time covers only the recognition process, excluding the pre-processing, segmentation, and feature extraction steps. The machine used for these tests is an Intel E7340, 2.4 GHz processor, with 2 GB of RAM memory. Table 14.6 compares the word recognition time of the two decoding strategies with respect to the number of streams. The performance of the HMM-recombination algorithm is completely flawed on large vocabulary tasks, especially when dealing with a large number of streams. In contrast, the two-level algorithm presents a significant improvement in recognition time and appears less sensitive to the variation in the number of streams and the lexicon size. Despite these improvements, the recognition time of the two-level decoding algorithm is still high, and many investigations are still needed to improve its performance. This can be performed by using a lexicon tree instead of a flat lexicon, or by introducing an estimation of the character duration on the decoding step. The simultaneous decoding of all stream character HMMs in the first step of the two-level algorithm can also reduce the computational cost.

Table 14.6 Word recognition time (in seconds) with respect to the number of streams K and the lexicon size V

$V =$	Two-level			HMM-recombination		
	100	1000	1600	100	1000	1600
$K = 2$	4.2	5	7.5	1.2	13.2	23.8
$K = 3$	9.4	10.1	13.4	15.2	162.8	298.6
$K = 4$	10.9	12	17.5	182.4	1836	3283.2

14.6 Conclusion

This paper presents a multi-stream recognition system for off-line Arabic handwritten words. The proposed approach combines low level feature streams, namely, density-based features extracted from two different sliding windows with different widths, and contour-based features extracted from upper and lower contours. The multi-stream paradigm provides an interesting framework for the integration of multiple sources of information. A two-level decoding algorithm is proposed to reduce the complexity of the decoding step and significantly speed up the recognition process while maintaining the recognition accuracy. The system has been tested on the IFN/ENIT database, and the results show significant improvement for the multi-stream approach compared to the performances reported recently on the same database.

References

1. Al-Hajj, R., Likforman-Sulem, L., Mokbel, C.: Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition. *Trans. Pattern Anal. Mach. Intell.* **31** (2009)
2. Bertolami, R., Bunke, H.: Early feature stream integration versus decision level combination in a multiple classifier system for text line recognition. In: ICPR, vol. 2, pp. 845–848 (2006)
3. Bourlard, H., Dupont, S.: Sub-band-based speech recognition. In: IEEE Int. Conf. on Acoust., Speech, and Signal Processing, pp. 1251–1254 (1997)
4. El-Hajj, R., Likforman-Sulem, L., Mokbel, C.: Arabic handwriting recognition using baseline dependent features and hidden Markov modeling. In: ICDAR’05, Seoul, South Korea, vol. 2, pp. 893–897 (2005)
5. Gnter, S., Bunke, H.: Ensembles of classifiers for handwritten word recognition. *Int. J. Doc. Anal. Recognit.* **5**(4), 224–232 (2003)
6. Hagen, A.: Robust speech recognition based on multi-stream processing. Ph.D. thesis, Ecole Polytechnique Federale de Lausanne (2001)
7. Kessentini, Y., Paquet, T., Benhamadou, A.: A multi-stream HMM-based approach for off-line multi-script handwritten word recognition. In: ICFHR’08, vol. 1, pp. 147–152 (2008)
8. Kessentini, Y., Paquet, T., Hamadou, A.B.: Off-line handwritten word recognition using multi-stream hidden Markov models. *Pattern Recognit. Lett.* **30**(1), 60–70 (2010)
9. Khorsheed, M.S.: Recognising handwritten Arabic manuscripts using a single hidden Markov model. *Pattern Recognit. Lett.* **24**, 2235–2242 (2003)
10. Kimura, F., Tsuruoka, S., Miyake, Y., Shridhar, M.: A lexicon directed algorithm for recognition of un-constrained handwritten words. *IEICE Trans. Inf. Syst.* **E77-D**(7) (1994)
11. Kuncheva, L.I.: *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, New York (2004)

12. Lorigo, L.M., Govindaraju, V.: Offline Arabic handwriting recognition: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(5) (2006)
13. Märgner, V., El Abed, H.: In: Arabic Handwriting Recognition Competition, ICDAR'07, vol. 2, pp. 1274–1278 (2007)
14. Märgner, V., El Abed, H.: In: Arabic Handwriting Recognition Competition, ICDAR'09, pp. 1383–1387 (2009)
15. Okawa, S., Bocchieri, E., Potamianos, A.: Multi-band speech recognition in noisy environments. In: Proceedings of IEEE International Conference on Acoustic, Speech, and Signal Processing, Seattle, Washington, pp. 641–644 (1998)
16. Pechwitz, M., Maddouri, S., Märgner, V., Ellouze, N.: IFN/ENIT-DataBase for handwritten Arabic words. In: CIFED'02, pp. 129–136 (2002)
17. Prevost, L., Michel-Sendis, C., Moises, A., Oudot, L., Milgram, M.: Combining model-based and discriminative classifiers: application to handwritten character recognition. In: 7th IC-DAR'03, vol. 1, pp. 31–35 (2003)
18. Rabiner, L.R.: A tutorial on hidden Markov model and selected applications in speech recognition. In: IEEE Proceedings, vol. 77, pp. 257–286 (1989)
19. Rabiner, L.R., Bergh, A., Wilpon, J.R.: An embedded word training procedure for connected digit recognition. In: Conference Record 1982 International Conference on Acoustics, Speech, and Signal Processing, pp. 1621–1624 (1982)
20. Sakoe, H.: Two-level DP matching—a dynamic programming-based pattern matching algorithm for connected word recognition. *IEEE Trans. Acoust. Speech Signal Process.* **27**, 588–595 (1979)
21. Schambach, M., Rottland, J., Alary, T.: How to convert a Latin handwriting recognition system to Arabic. In: ICFHR'08, pp. 265–270 (2008)
22. Vinciarelli, A., Bengio, S., Bunke, H.: Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(6), 709–720 (2004)
23. Wellekens, C.J., Kangasharju, J., Milesi, C.: The use of meta-HMM in multistream HMM training for automatic speech recognition. In: Proc. of Intl. Conference on Spoken Language Processing, Sydney, pp. 2991–2994 (1998)
24. Xu, L., Krzyzak, A., Suen, C.: Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans. Syst. Man Cybern.* **3**, 418–435 (1992)
25. Zheng, L., Hassin, A.H., Tang, X.: A new algorithm for machine printed Arabic character segmentation. *Pattern Recognit. Lett.* **25**(15), 1723–1729 (2004)

Chapter 15

Toward Distributed Cursive Writing OCR Systems Based on a Combination of Complementary Approaches

Maher Khemakhem and Abdelfettah Belghith

Abstract Large amounts of cursive writing documents are still waiting to be computerized for several and different purposes. These documents are in general of medium to low quality; hence they require a sophisticated recognition algorithm capable of properly extracting the correct text from low quality cursive documents. The Dynamic Time Warp (DTW) algorithm is among the most effective algorithms for cursive writing optical character recognition (OCR). However, the DTW is a rather complex task requiring extensive computational capabilities, which hinders its commercial deployment on nonspecialized stand alone machines. Volunteer grids, such as XtremWeb and BOINC, provide viable infrastructures to speed up the DTW execution time. Recent experiments conducted on the Scientific Research Tunisian Grid (SRTG), an XtremWeb volunteer grid, confirmed this claim and showed a very tangible speedup along very high recognition rates. Such infrastructures present several practical advantages, such as the possibility of noncondemnation of the involved computers and the possibility of their simultaneous use by different users and/or applications. Unfortunately, volunteer grid infrastructures are inherently unable to guarantee the continuous availability of the stored data and, more importantly, the engaged processing capacities. Any involved computer may renegade and depart from the system at will, which consequently affects the application performance. Agent technology can be exploited here to solve the problem. In this chapter, we propose a service-oriented grid architecture (SOGA) based on the integration of both grid and agent technologies. An analytical study is conducted to ascertain and evaluate the key performance parameters of our proposed SOGA. The results confirm that our proposal provides a solid and viable solution for the large scale recognition of printed cursive writing based on the DTW algorithm.

M. Khemakhem (✉)

MIRACL Lab, The Higher Institute of Management University of Sousse, PO BOX 763,

4000 Sousse, Tunisia

e-mail: maher.khemakhem@fsegs.rnu.tn

A. Belghith

HANA Research Group, ENSI, University of Manouba, Manouba, Tunisia

e-mail: abdelfattah.belghith@ensi.rnu.tn

15.1 Introduction

Large amounts of printed cursive writing documents in several different languages, such as the Arabic language, are in urgent need of computerization. These documents are usually of poor quality, necessitating adequate and efficient optical character recognition (OCR) systems. The existing OCR systems are, however, very sensitive to the input document quality. The medium or low quality of any input document decreases dramatically the corresponding recognition rates. Conducted experiments and evaluations on several cursive writing OCR approaches show and confirm that the Dynamic Time Warp (DTW) algorithm is among the best techniques for such a mission [38, 39].

The DTW algorithm is the result of the adaptation of dynamic programming to the field of pattern recognition [17, 20, 21, 37, 40, 41, 46] and provides very interesting recognition rates for cursive printed texts such as Eastern and Middle Eastern languages (e.g., Arabic) and even Latin connected characters [37, 38, 40, 41]. The ability of this algorithm to properly recognize any medium or low quality printed cursive writing without prior character segmentation from within a reference library composed of isolated characters makes it very attractive. However, the required enormous amount of computing constitutes DTW's main pitfall and hence restricts its widespread use.

Many works and approaches have been proposed to solve this problem [5, 16, 20, 21, 38, 43, 46]. In a previous work [39], we showed that grid computing, and more specifically volunteer grids, can instantiate an adequate solution based on the large amount of computing and storage capacities they can provide. Volunteer grid computing presents several advantages, yet it also yields some problems. Avoiding the sole condemnation of participating computers to distributed missions or applications may be considered among the most important advantages. This advantage will lead surely to flexible grids where participating computers can be used simultaneously and seamlessly by their owners and by distributed missions and applications. Moreover, any participating computer in a volunteer grid may choose to depart or disassociate at any time. However, this very great flexibility causes serious problems of both data availability and processing loss [25], and may even disrupt the proper functioning of running applications unless some defined measures are undertaken.

Agent technology, especially the use of intelligent agents, can play a vital leverage to achieve the grid vision [28, 35, 48]. This technology provides autonomy, intelligence, and mobility which can be exploited to solve both of these problems. We propose in this chapter a service-oriented grid architecture (SOGA) integrating volunteer grid computing and agent technology in an attempt to provide an effective viable solution for a large scale cursive writing OCR based on the DTW algorithm.

The chapter is organized as follows. Section 15.2 gives an overview on distributed OCR systems. Section 15.3 provides a formulation of the DTW basics, and Sect. 15.4 describes the DTW algorithm for printed cursive writing recognition. Section 15.5 provides a brief overview of grid computing and agent technology. Our proposed SOGA and the correspondent performance evaluation are presented in Sect. 15.6. Finally, we conclude and present some perspectives and further investigations.

15.2 Existing Distributed OCR Systems

A few solutions for large scale OCR are provided by computer scientists. Representative examples including The Australian Newspaper Digitization Project [7], OCRGrid [8], Kirtas [9], and OCropus [10].

The national library of Australia has used OCR software to establish a large scale Historic Digitization Project. The Australian Newspaper Digitization Program (ANDP) claimed that an “acceptable” OCR still has some distance to go, and needs further improvements. Besides, the poor quality of the original source documents urged the National Library of Australia to go ahead and work with what was available but using further indirect refinements. In order to improve the quality of OCR accuracy, the committee of the Library adopted from the thirteen methods they came out with only five new ones, which are going to be tested and investigated. These methods are used mostly to compare image optimization software, to experiment using grayscale files, to use Australian dictionaries, to clean/correct OCR text manually, and to use confusion matrix and language modeling post/during OCR processing. They looked for improving OCR accuracy by using both a combination of methods and also manual methods of humans correcting the mistakes of machines. The Australian Library was considered the first worldwide party to involve public users in the correction of texts instead of the contractor. Such a solution was considered labor intensive for the Library before the emergence of web 2.0 technologies. Although the public was not informed that they could introduce corrections to the texts, they embarked on this correction immediately. There were measures undertaken to check the accuracy of the OCR-corrected text by counting the number of corrected lines and the number of different corrected articles. However, the intervention of public users may badly affect the content of the articles, so they have to make sure that no data has been added to the original text.

OCRGrid is a platform for distributed and cooperative OCR systems. The main idea of OCRGrid is to deploy a lot of OCR servers on a network to allow end users to search for and use an adequate server. As servers can cooperate with each other, clients can benefit from a distributed parallel environment and consequently accelerate OCR tasks. Applications searching for enhancing accuracy can also benefit from OCRGrid due to the use of a majority logic technique which requires the running of many OCR engines. A multilingual processing environment can also be realized by combining many community-supported OCR servers for various languages with localized dictionaries.

The Kirtas technology is an automatic book scanner which can do batch OCR for large volumes of books and other documents. By using innovative “automatic page-turning scanner” technology and a high-resolution Canon digital camera, Kirtas ensures image processing, quality control, OCR, and metadata. It can handle 15 left-to-right languages including English and French, 5 right-to-left languages including Arabic, and 3 bilingual languages including Arabic/English. Kirtas OCR processing rates are very fast (about one page per second). Many public and university libraries decided to exploit the Kirtas technology to digitize their old books. However, the downside of using such a technology lies in its very high price, which consequently limit its adoption on a large scale.

OCRopus is an open source OCR system sponsored by Google. It targets the research community by improving the state of the art of optical character recognition and has also sought to serve on large scale commercial document conversions. The system applications include a modern digital library and the recognition of classical literature. Its main perspective consists of familiarizing the system with more languages so as to become omni-lingual and omni-script by contributing in an open source community. Although the system has been evolving, it has not yet incorporated the Arabic language into its framework.

15.3 Preliminaries

An OCR system is generally decomposed into four stages. The first one concerns the acquisition of the text scanned image to be provided in the form of raw bitmaps of pixels (or binary data). The second stage deals with the pre-processing of this raw data and mainly concerns filtering the scanned images, and framing, positioning, and segmenting the text into connected components (i.e., groups of connected characters). The third stage concerns the description and the feature extraction, and consequently the determination of the characteristic fragments (i.e., the feature vectors) of the group of connected (cursive) characters to be recognized. As such, a certain combination of characteristic fragments can be assigned with adequate confidence by the decision process to a recognized class. The final stage forms the culminating point of the recognition process: the decision on the correct classification of the unknown.

What makes DTW an attractive and efficient procedure to use in the recognition process is its ability to eliminate time differences between the characters to be recognized. Based on dynamic programming path finding, DTW presents a computationally efficient procedure to find the optimal time alignment between two occurrences of the same character and, more generally, between any two given forms.

Consider two symbols A and B , each represented by a sequence of feature vectors as provided by stage three of the recognition system, namely:

$$A = a_1, a_2, \dots, a_i, \dots, a_m \quad (15.1)$$

$$B = b_1, b_2, \dots, b_j, \dots, b_n \quad (15.2)$$

where m and n represent the number of feature vectors composing respectively A and B . As a measure of the difference between two feature vectors a_i in A and b_j in B , a distance $D(i, j)$ is employed. Consequently, a matrix D of distances is formed. The optimal time alignment between the two symbols is represented by the optimal path lying between point $(1, 1)$ and point (m, n) in the D matrix and warping through the matrix in such a way that at any point (i, j) , the cumulative distance $S(i, j)$ is minimum. A key operation for DTW is the computation of the

summation distance S representing the cumulative distance $S(m, n)$ at end point (m, n) and defined by:

$$S = \min \sum_{i=1}^m D(i, W(i)) \quad (15.3)$$

where $W(i)$ represents the warping function that governs the way to find a significant optimal path in matrix D with $W(1) = 1$ and $W(m) = n$. The warping function is in fact a model of time axis (i.e., the axis of the symbol A to be recognized) fluctuation in a character pattern. By using the continuity conditions defined in [37, 40, 41] by

$W(i + 1) - W(i) = 0, 1, 2$, for each $1 \leq i \leq m$ we obtain the following functional equations on the cumulative distances:

$$S(i, j) = D(i, j) + \min_{\substack{1 \leq l \leq m \\ 1 \leq j \leq n}} \left\{ \begin{array}{l} S(i - 1, j), \\ S(i - 1, j - 1), \\ S(i - 1, j - 2) \end{array} \right\} \quad (15.4)$$

with the initial condition

$$S(1, 1) = D(1, 1) \quad (15.5)$$

We then have $S = S(m, n)$. The warping function and the optimal alignment can be found by a backtracking technique.

15.4 DTW for Cursive Writing Character Recognition

15.4.1 Isolated Character Recognition

We consider a reference library of R trained characters forming the alphabet in some given fonts for a given language, and denoted by C_i , $i = 1, 2, \dots, R$. The technique consists in using the DTW pattern method to match an input character against the reference library. The input character is thus recognized as the reference character that provides the best time alignment S among all R characters; namely, character A is recognized to be C_k if the time alignment it provides is, e.g.,

$$S_k = \min_{1 \leq r \leq R} \{S_r\} \quad (15.6)$$

That is, S_k is the summation distance corresponding to the matching of A with the reference character C_k .

15.4.2 Recognition of Connected or Cursive Characters

Words in cursive writing (such as Arabic writing) are inherently written in blocks of connected characters. Many researchers have considered the segmentation of these

blocks into separated characters before performing the recognition phase based on isolated characters [4, 6, 23, 36, 49]. We may, therefore, simply employ the isolated character recognition procedure described in the preceding section. The viability and robustness of the DTW technique, however, is its ability and efficiency to perform the recognition without the need for a prior segmentation into isolated characters.

Let T constitute a given connected sequence (a block) of characters to be recognized. T is then composed of a sequence of N feature vectors that actually represent the concatenation of some subsequences of feature vectors, each representing an unknown character to be recognized. As portrayed on Fig. 15.1, the text T lies on the time axis (the X -axis) in such a manner that feature vector t_i is at time i on this axis. The reference library is portrayed on the Y -axis, where reference character C_r is of length l_r , $1 \leq r \leq R$ (that is, composed of l_r feature vectors).

By extension of our earlier notation, we use $S(i, j, r)$ to represent the cumulative distance at point (i, j) relative to reference character C_r . The objective is to detect simultaneously and dynamically the number of characters composing T and to recognize these characters. There surely exist a number k and indices (m_1, m_2, \dots, m_k) such that $C_{m_1} \oplus C_{m_2} \oplus \dots \oplus C_{m_k}$ represents the optimal alignment to text T where \oplus denotes the concatenation operation. The path warping from point $(1, 1, m_1)$ to point (N, l_{m_k}, m_k) and representing the optimal alignment is therefore of minimum cumulative distance, that is:

$$S(N, l_{m_k}, m_k) = \min_{1 \leq r \leq R} \{S(N, l_r, r)\} \quad (15.7)$$

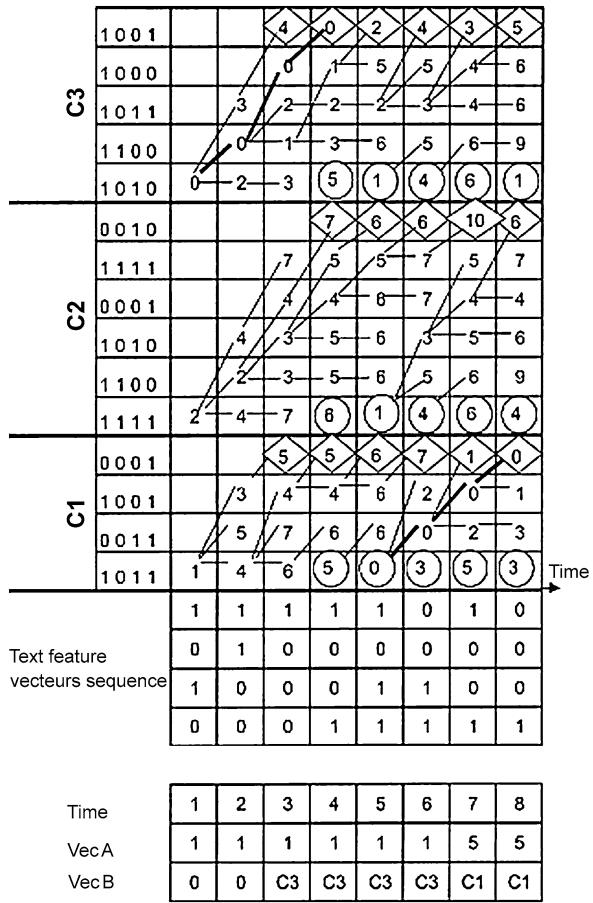
This path, however, is not continuous, since it spans many different characters in the distance matrix. We therefore must allow at any time the transition from the end of one reference character to the beginning of a new character. The end of reference character C_r is first reached at time $i = \lceil \frac{l_r+1}{2} \rceil$, and the warping function reaches point (i, l_r, r) . As we can see on Fig. 15.1, the end of reference characters C_1, C_2, C_3 are first reached respectively at times 3, 4, 3. The end points of reference characters are shown on Fig. 15.1 inside diamonds, and points at which transitions occur are within circles. From these times on and up until time N , the warping function always reaches the ends of the reference characters. At each time i , we allow the start of the warping function at the beginning of each reference character along with the addition of the smallest cumulative distance of the end points found at time $(i - 1)$. The resulting functional equations are then:

$$S(i, j, r) = D(i, j, r) + \min_{\substack{1 \leq i \leq N \\ 1 \leq j \leq l_r \\ 1 \leq r \leq R}} \left\{ \begin{array}{l} S(i - 1, j, r), \\ S(i - 1, j - 1, r), \\ S(i - 1, j - 2, r) \end{array} \right\} \quad (15.8)$$

with the boundary conditions

$$S(i, 1, r) = D(i, 1, r) + \min_{\substack{\{1+\beta \leq i \leq N\} \\ 1 \leq k \leq R \\ 1 \leq r \leq R}} S(i - 1, l_k, k) \quad (15.9)$$

Fig. 15.1 The DTW mechanism



where $\beta = \lceil \frac{1 + \min_{1 \leq r \leq R} \{l_r\}}{2} \rceil$.

To trace back the warping function and the optimal alignment path, we have to maintain the transition time (to be memorized) from one reference character to the others. This can easily be accomplished by the following backtracking procedure:

$$b(i, j, r) = \text{trace} \min_{\substack{1 \leq i \leq N \\ 1 \leq j \leq l_r \\ 1 \leq r \leq R}} \left\{ \begin{array}{l} b(i - 1, j, r), \\ b(i - 1, j - 1, r), \\ b(i - 1, j - 2, r) \end{array} \right\} \quad (15.10)$$

where the *trace min* function returns the element corresponding to the term that minimizes the functional equations. The functioning of this procedure is portrayed on Fig. 15.1 through the two vectors *VecA* and *VecB*, where *VecB*(*i*) represents the reference character yielding the least cumulative distance at time *i*, and *VecA*(*i*) provides the link to the start of this reference character in the text *T*. The heavy marked path through the distance matrix represents the optimal alignment of text *T* to the reference library. We observe that text *T* is recognized as *C*₁ \oplus *C*₃.

The amount of computing to be achieved is enormous, which makes the DTW algorithm very slow if adopted for any large scale document recognition. Many researchers attempted to solve this problem. The proposed solutions can be classified into the following two categories:

- The first one is mainly based on the design of dedicated architectures and processors (VLSI) that can speed up the DTW algorithm [22, 46]. Unfortunately, these proposed solutions did not find the expected success because of the high cost of the architectures or processors.
- The second one is based on the distribution of the DTW algorithm itself [5, 16, 42, 43]. All these proposed solutions take advantage of the DTW algorithm's inherent parallelism, yet they differ in the accomplishment of this task. Different proposals of tightly coupled and loosely coupled architectures were investigated [42, 43] and showed that important speedups with varying efficiency factors are attainable.

In this chapter, we attempt to take advantage of some emerging technologies to speed up the DTW algorithm. We propose an approach integrating both grid computing and agent technology. The essence of our approach consists in appropriately splitting the bitmap image (binary image) of the entire text to be recognized into sub-bitmap images (binary matrices) and then distributing them, dynamically and according to the current state of the grid and the multi-agent system, among several OCR service agents. Thus, every binary matrix will be processed by a given OCR service agent on a given participating machine within the volunteer grid.

15.5 Grid Computing and Agent Paradigm: A Brief Overview

15.5.1 *Grid Computing*

A grid is a collection of machines, sometimes referred to as “nodes,” “resources,” “members,” “donors,” “clients,” “hosts,” “engines,” and many other such terms. These nodes contribute to the grid by providing access to and use of their resources. Some resources may be used by all users of the grid; others may have specific restrictions [19, 32, 33].

In most organizations, there are large amounts of underutilized computing resources. Most desktop machines, for example, are busy less than 5 percent of the time. In some organizations, even server machines can often be relatively idle. Grid computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing the efficiency of resource usage [33]. Furthermore, machines may have enormous unused disk drive capacity. Grid computing, and more specifically “data grids,” can be used to aggregate this unused storage into a much larger virtual data store, possibly configured to achieve improved performance and reliability over that provided by any single machine [33].

A grid computing environment is an infrastructure that allows many institutions, regardless of their geographical locations, to pool a large collection of their heterogeneous networked computers and systems to share a set of software and hardware resources, services, and licenses, etc. [18, 27, 33]. This remarkable ability of resource sharing in various combinations yields many advantages, such as:

- Increasing the efficiency of resource usage.
- Facilitating remote collaboration between institutions, researchers, etc.
- Providing users with a huge computing power.
- Providing users with a huge storage capacity.

The design and implementation of grid infrastructures and distributed grid computing applications has been a very active subject for the last few years [14, 18, 19, 27, 33, 48]. Two main categories of grids can be distinguished based on whether participating nodes are solely and completely advocated or voluntarily connected. A dedicated grid is the result of the federation of a number, usually very large, of geographically dispersed computers completely dedicated to the grid mission [19]. As such, these gridded computers can be used only by an authorized community for some well-defined specific purposes. A volunteer grid, however, is composed of many volunteer federated computers which can be used by several communities and hence are not akin to specific missions [1–3]. In this type of grid computing environment, any computer can be voluntarily and dynamically connected or disconnected by its own owner. In the remainder of this paper, we shall focus solely on this type of volunteer grid, such as XtremWeb [11, 12] and BOINC [13, 15] (Berkeley Open Infrastructure for Network Computing).

XtremWeb [11, 12] is a P2P system developed by the University of Paris-Sud, France [12] for intensive calculation and computing purposes. BOINC provides a special large scale computing environment as a volunteer grid, though occasionally it may be dedicated to specific users for specific applications [13].

Both of these grid infrastructures, being volunteer grids, cannot guarantee both the continuous availability of stored data and the continuous seizure of computing resources, as any participating computer may at will and at any time choose to depart from the system. Some approaches have been proposed to deal with these problems [25]. It is our aim in this chapter to propose a viable framework to solve both of these issues based on agent technology.

15.5.2 Agent Paradigm

The agent paradigm is very attractive, for it mimics human and animal societies or communities in terms of interaction and coordination to jointly achieve a global goal or to solve a complex problem [28].

An agent can be a physical or a logical entity that owns certain characteristics, skills, and goals. An agent is able to achieve some tasks or goals without a cen-

tral control and sometimes has its own prior defined planning. It can interact and cooperate with others to achieve complex tasks. It is able sometimes to dynamically adapt its processes and skills to meet certain situations. An agent can be intelligent (cognitive) or reactive (not intelligent), mobile or stationary (not mobile) [24, 26, 30, 31, 34, 44, 45, 47, 48, 50–54].

Cognitive Agent

An intelligent agent, also called a cognitive agent, acts autonomously. It is able to achieve complex tasks without any exogenous help. It owns an internal knowledge base that can be used to manage all of its processes including its skills and its interactions with other agents. It owns specific goals and it is capable of making local decisions such as optimizing the achievement of some tasks. A cognitive agent usually performs very specific and selected tasks. The great advantage of this type of agent is the simplicity with which it can be reused in many applications and within different societies because of its inherent modularity. However, it is usually very difficult to design, implement, and integrate into agent societies.

Reactive Agent

A reactive agent is not intelligent and may act only upon receiving a stimulus from within its environment. It is considered as an integral part of its own community. Unlike a cognitive agent, it does not own a specific goal and is not able to make internal decisions. It cannot solve tasks or achieve goals without the help of and the coordination with other agents. The great advantage of this type of agent is the relatively low complexity of its design, implementation, and integration within an agent society or community.

An agent can be stationary (usually located at the same location, for example, on the same networked PC) or mobile (capable of moving from one node to another within the system). A multi-agent system is simply a society or a community of agents that can interact to solve complex problems in a distributed manner.

A multi-agent system can have several advantages [44]: speedup due to concurrent processing, less communication bandwidth requirements because processing is located nearer the source of information, more reliability because of the lack of a single point of failure, improved responsiveness due to the great amount of available processing, and an easier system development based on the modularity inherent to and provided by the decomposition into agents. Agent interaction and cooperation are considered among the major issues in multi-agent system design and development. Interaction among agents is what makes a population or a society of agents capable of solving a complex problem.

15.6 The Proposed Architectural Model

Our proposed service-oriented grid architecture (SOGA) is based on a volunteer grid computing that interconnects several volunteer institutions' local area networks (LANs), hence providing a myriad of participating desktop heterogeneous computers. We integrate into every computer system an intelligent stationary agent that can participate to any given OCR process. This agent, called an OCR service agent, is able to provide the OCR service to the users.

Within each LAN a specific cognitive agent, called the home forecast agent, is responsible for tracking the local agents' activities, and for negotiating and cooperating with the other distant forecast agents in other LANs (called the foreign forecast agents) in order to decide which stationary agents (OCR service agents) can participate in the proposed work. The forecast state of the entire grid represents the set of current nodes that can provide a volunteer service along with their loads, specifications, and capabilities. We assume that the multi-agent system (MAS) forecast state is updated periodically to track node departures and joins and to ascertain the load characteristics, communication latencies and requirements, and current capabilities of the different nodes in the grid. It is interesting to note that these specific cognitive agents may either be stationary or mobile or both. If they are stationary, it is up to every volunteer LAN to decide on the location of its local forecast agent, for example, within one of its continuously running proxies. A LAN participates in the grid and the MAS whenever its forecast agent is up and running. On the other hand, a forecast agent may be mobile. In such a case, we assume that the mobile agent has a prior knowledge of all LAN members of the grid along with the structure of the overlay network connecting them. In the rest of the paper, we consider that all specific forecast agents are stationary.

Every intelligent stationary service agent has, in addition to communication capabilities, the following minimal set of skills:

1. A method to interface the user to start up, control, and follow its own launched cursive writing recognition process.
2. A method to interact with its home (local) forecast agent. This method is in particular responsible for notifying the home forecast agent of the node's joins and departures; to acquire the forecast state just before launching an OCR application; and to acquire from the home forecast agent updates on any departure of currently participating nodes during the execution and, more importantly, on the identity of the nodes appointed (by foreign forecast agents as described next) to continue and finish the required task.
3. A method to make decisions about the number and selection of the target OCR service agents of the grid that will participate in the work. Such decisions can be adequately made based on the size of the text to be recognized and the detailed forecast state of the grid obtained from its home forecast agent.
4. A method for pre-processing the text binary data: noise reduction, segmentation of the text into connected components, and feature extraction.
5. A method to appropriately fragment the binary image of the text to be recognized into parts (binary matrices). The fragmentation process may take into account

diverse criteria including but not limited to the forecast state of the grid, the characteristics and capabilities of each participating node, the heterogeneity of participating nodes, and the targeted speedup or character recognition rate.

6. A method to implement the DTW algorithm.
7. A method to launch and monitor the execution on the different involved nodes. This method is in particular responsible for gathering execution results and providing them in an appropriate manner to the local (initiator) user.

Every forecast agent in our SOGA includes, in addition to its communication capabilities, the following minimal set of skills:

1. A method to monitor local nodes' memberships (involvement to running distributed applications), their characteristics, and their current capabilities.
2. A method to accommodate newcomers within its local LAN based on a sniffing technique.
3. A method to track node renegation and departure throughout the grid, and to notify accordingly local stationary nodes that have launched the current OCR applications.
4. A method to cooperate with other foreign forecast agents to build a complete global forecast state of SOGA.
5. A method to recover an unfinished process from a locally participating computer that prepares to renegade or depart from the system. This method is in particular responsible for designating another member to finish the assigned work and for notifying the local or distant stationary agent that launched the application. The distant agent is notified through its home forecast agent. As such, it is this method that solves the issue of processing power loss, and maintains whenever possible the same service level.

Thus, if a user would like to launch a cursive writing OCR process, he just needs to use any one of the local idle stationary agents. Consequently, this chosen agent will be considered as the main (the launcher or the initiator) agent. The remaining intelligent stationary agents that will participate in the work will be considered as collaborators. We note that the main agent will be responsible only for the user interface and the management of the distributed OCR process. It will not participate in this recognition process, though it may participate in other applications. Consequently, our proposed architectural model gives the possibility of launching several cursive writing OCR applications at the same time and throughout the grid.

One of the advantages of our model compared to the one proposed by Zhongzhi et al. [48] is the complete lack of the interoperability issue, as the data exchanged between agents is either binary data bytes or ASCII code bytes.

15.6.1 The Analytical Model

To conduct a performance evaluation of the proposed SOGA, we need to define the different system parameters. We denote by:

P : the total number of connected volunteer computers of the grid that will participate in a cursive writing OCR process. P denotes then the number of participating intelligent stationary agents, which is dynamically varying as nodes join and depart from the system.

U_i : the CPU utilization percentage of participating computer i , $1 \leq i \leq P$.

X_i : the total number of machine instructions needed to compute and recognize the part of the input text assigned to agent i . Given P , X_i is obtained by a pseudo-optimal sequential assignment algorithm through the concatenation of some contiguous binary matrices. Therefore, $\sum_{i=1}^{P-1} X_i$ represents the total number of machine instructions needed to compute and recognize the entire cursive writing text at hand. We will neglect the time needed for the pretreatment process, as it is only done once and represents a very small fraction of the total time achieved by the DTW algorithm.

σ_i : denotes the computing power of stationary agent i where $i = 1, 2, \dots, P$. This value is expressed in millions of machine instructions per second (Mips).

C_{mi} : the total time needed to communicate one data packet from the launcher (main) stationary agent to collaborator agent i . This communication time is supposed to include all required details of the communication process such as the propagation delay, the queueing delays, the network capacity and current load, etc.

C_{im} : the total time needed to communicate one data packet from collaborator agent i to the main agent. We shall assume that $C_{mi} = C_{im}$.

$NTPack_i$: denotes the total number of data packets to be transmitted from the main stationary agent to collaborator agent i . We consider here that an IP packet contains 536 bytes of payload.

$NRPack_i$: denotes the total number of packets received at the main stationary agent from collaborator agent i . $NRPack_i$ represents the recognition results achieved by collaborator agent i .

$CT(P)$: The completion time, it expresses the total time needed by the P participating computers (agents) to compute and recognize the whole text at hand.

We will suppose next that:

- The time needed to get the detailed forecast state of the grid from the home forecast agent is constant and negligible.
- The time needed to select the target collaborator agents and then to split pseudo-optimally the binary image of the entire text to be recognized is also constant and negligible.

Otherwise these two constants should be added to the expression of the $CT(P)$, which is given here by the following equation:

$$CT(P) = \max_{1 \leq i \leq P-1} \left[(C_{im} \times (NTPack_i + NRPack_i)) + \left(\frac{X_i}{\sigma_i \times (1 - U_i)} \right) \right] \quad (15.11)$$

$CT(1)$ denotes the total time required by a sequential architecture (or a single computer) to compute and recognize the same text at hand. Its expression is given by the following equation:

$$CT(1) = \frac{\sum_{i=1}^{P-1} X_i}{\sigma} \quad (15.12)$$

where σ denotes the computing power of the computer used in the sequential OCR process.

Let $S(P)$ denote the recognition process speedup factor. Speedup here is in fact a measure of how much faster a computation finishes under the SOGA proposed approach than under a sequential mode. Its expression is given by the following equation:

$$S(P) = \frac{CT(1)}{CT(P)} \quad (15.13)$$

15.6.2 Numerical Results

In this section, we shall concentrate on Arabic writing as an instance of cursive writing. To further proceed with the numerical study of our proposed analytical model, we need to evaluate some statistical characteristic of cursive Arabic writing. As such, we considered an Arabic legal text corpus composed of 30 pages selected from Volume I, Part I of the United Nations Publications (ISSN 1014-5559). This text corpus (samples of which are portrayed on Fig. 15.2 and Fig. 15.3) was scanned by an HP scanner with a resolution of 300 dpi (dots per inch). The corpus selection of this legal text presents, generally, a very rich terminology, yet it corresponds to point sizes and fonts widely used in virtually all legal documents potentially subject to large scale OCR operations.

The feature extraction procedure used is based on Hadamard's transformation of dimension 30 [29]. Consequently, all feature vectors are of dimension 30. The conducted statistical study on the processed 30 pages of Arabic text emphasizes the following quantities:

- The average number of characters per page is approximately 900 characters.
- the total number of obtained connected (cursive) components (fragments or binary matrices) within the entire text corpus is 11917.
- The total number of pixel vectors composing the 11917 fragments is 427102.
- The average length of a component is approximately 36 pixel vectors, and
- The average width of an Arabic character of the studied corpus is approximately 15 pixel vectors.

Figure 15.4 portrays the number of components (dot matrices representing connected sub-words) as a function of the number of pixel vectors they contain (i.e., the matrix width in pixel vectors). We observe that connected components tend rather to

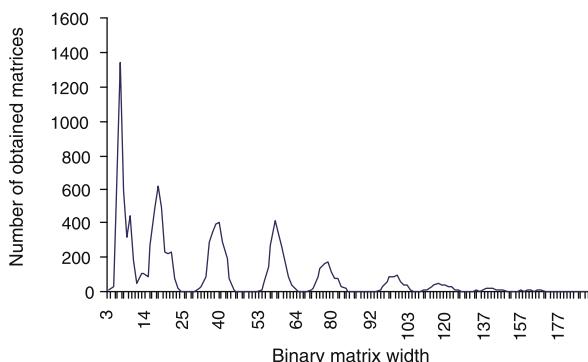
٨١ - ينفي استخدام موظفين مؤهلين ، وأن يكون بينهم عدد كاف من المتخصصين مثل المربين ، والمبجهين المهنيين ، والمستشارين ، والخواصيين الاجتماعيين ، وطبياء وخصائص العلاج النفسي . وينفي أن يعين هؤلاء وغيرهم من المتخصصين ، عادة ، على

Fig. 15.2 A corpus text sample



Fig. 15.3 The biggest sub-word

Fig. 15.4 Text corpus statistics



have a small to moderate number of pixel vectors, indeed 36 vectors on the average representing around 2.4 connected characters on the average per sub-word.

However, for the sake of our experiments of the DTW OCR and the proposed SOGA system on huge documents, we require a significantly larger amount of input text. As such, we shall rather consider a text input of 30000 pages of similar texts obtained by replicating the 30 considered pages 1000 times. Consequently, this amounts to 11917000 connected fragments or components to be assigned among the $P - 1$ agents, that is, 427102000 pixel vectors in total.

The assignment process of these connected components among the $P - 1$ collaborator agents is achieved by the following algorithm:

1. Fix P the number of agents that will participate in the work, a main agent and $P - 1$ collaborator agents.
2. Divide the total number of pixel vectors by $P - 1$, under the used hypothesis that agents are homogeneous in computing power and that the communication latency to any one of these agents is approximately the same. Let a_{opt} be the result of this division; it represents the ultimate optimal number of pixel vectors to be assigned to each agent.
3. The sequence of fragments composing the entire text is now assigned to collaborator agents in a sequential consecutive manner. The first collaborator agent will

be assigned a number of consecutive components such that the sum of their composing pixel vectors is the least value equal to or larger than a_{opt} . Let Fr_1 be such a value. Collaborator agent i , $i = 1, \dots, P - 2$ will then be assigned a number of consecutive components Fr_i such that $(\sum_{j=1}^i Fr_j)$ is the least value equal to or larger than ia_{opt} . The remaining components are assigned to collaborator agent $P - 1$.

Consequently, the expression of $X_i \forall i = 1, 2, \dots, P - 1$, representing the total number of machine instructions to be accomplished by collaborator agent i , is given by

$$X_i = \alpha \sum_{j=1}^{Fr_i} \left\{ \sum_{r=1}^R \left\{ \sum_{t=1}^{N_j} L_r(t) \right\} \right\} \quad (15.14)$$

where:

α denotes the total number of machine instructions needed to execute the cumulative distance at any given warping point (see Eqs. (15.8) and (15.9) and Fig. 15.1). Conducted experiments revealed that α can be approximated by 250; recall that each feature vector is of dimension 30.

R is the total number of reference characters composing the reference library V . Recall that Arabic characters have some specific characteristics where some characters may have four different shapes according to their positions within the word [6]. Consequently, the total number of Arabic characters (including shape variation) is around 100. Furthermore, we suppose that each reference character within library V is represented by three different occurrences; hence the value of R will be around 300.

t denotes the time axis as adopted on Fig. 15.1.

N_j is the total number of feature vectors composing component j within the sequence of components Fr_i assigned to collaborator agent i . Recall that feature vectors are of dimension 30 and that the feature extraction procedure based on Hadamard's transformation conserves the size of the image [37, 40, 41, 43].

$L_r(t)$ denotes the total number of warping points to be executed at time t within reference character r . It can be expressed as follows:

$$L_r(t) = \sum_{i=1}^{2t-1} \delta(l_r - i) \quad (15.15)$$

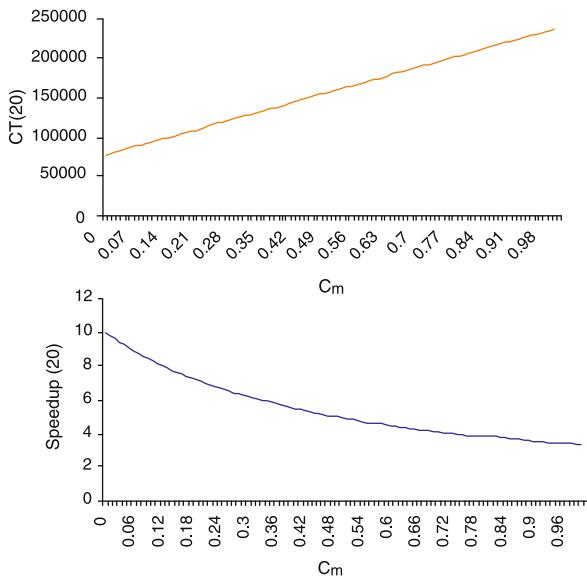
where $\delta(i)$ denotes the integer function that equals one for $i \geq 0$ and equals 0 otherwise.

l_r denotes the width or the number of feature vectors composing the reference character $C_r \forall r = 1, \dots, R$ which is approximated to 15 (we recall that this value represents the average width of an Arabic character of the studied corpus).

We will suppose next that:

- The communication delay of a data packet is the same from any collaborator agent to the main agent and vice versa. While this keeps our numerical analysis simple and clearer, it needs undoubtedly a further refinement that is beyond the

Fig. 15.5 Case where the number of OCR service agents = 20



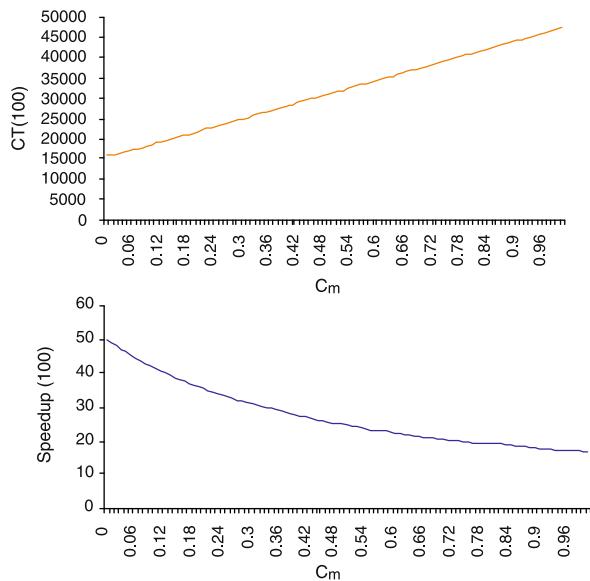
scope of the current paper. Consequently, we are assuming that $C_{mi} = C_{im} = C_m \forall i = 1, 2, \dots, P - 1$. Indeed, for an extremely fast underlying network, that is, for a zero communication delay, we have $C_m = 0$ second. For a point-to-point communication capacity of 4.6 kbps, we get $C_m = 1$ second as the packet size is 576 bytes including 40 bytes of TCP-IP header (a total of 4608 bits). We shall then let C_m vary between 0 and 1 second to accommodate an array of communication latencies and study their impacts on the completion time and speedup.

- All participating computers are homogeneous in execution power and capabilities. Consequently, we shall assume that $\sigma_i \forall i = 1, 2, \dots, P - 1$ and σ are constant and equal to 1000 Mips. This value corresponds approximately to the average of the computing power of our lab computers.
- The assumed value for U_i is 50 % for all $i = 1, \dots, P$.

Figure 15.5 portrays the completion time and the speedup as a function of the parameter C_m when 20 service agents are used; that is, $P = 21$. When an extremely fast network is used, that is, for $C_m = 0$, CT equals to 77845 seconds (around 21.6 hours). Recall that we are treating a rather huge corpus of 30000 pages. We observe that CT increases as the underlying network gets slower. For $C_m = 1$, that is, in the case of point-to-point communication capacities of 4.6 kbps, CT amounts to 236862 seconds (just over 65 hours). The speedup equals 10 for $C_m = 0$, as U_i is 50 % for all $i = 1, \dots, P$. The speedup becomes lower as the underlying network becomes slower; for $C_m = 1$ we get a speedup of just 3.286.

Figure 15.6 portrays the completion time and the speedup as a function of C_m but for 100 service agents. As expected, we get the exact same behavior as for Fig. 15.5; however, more efficiency is obtained. When an extremely fast network is used, that is, for $C_m = 0$, CT is only equal to 15569 seconds (around 4.32 hours); however, for

Fig. 15.6 Case where the number of OCR service agents = 100



$C_m = 1$ CT amounts to 47374 seconds (around 13.2 hours). The speedup equals 50 for $C_m = 0$ and equals 16.432 for $C_m = 1$.

Figure 15.7 portrays the completion time and the speedup when 1000 service agents are used. Here, we get large speedups of 500 for $C_m = 0$ and 164.234 for $C_m = 1$. These large speedups allow us to reach very fast completion times of 1556.916 seconds (around 0.43 hour) for $C_m = 0$ and 4739.916 seconds (around 1.3 hours) for $C_m = 1$.

These obtained results confirm that our proposed model yields a very interesting framework for large scale cursive writing OCR under the DTW algorithm. In the quest of an improved recognition rate, our proposed framework combined complementary approaches, yet it permits us easily to integrate others. Each of these complementary approaches and techniques would be implemented as a new service agent.

15.7 Conclusion and Perspective

In this chapter, we proposed a service-oriented grid architecture (SOGA) based on the integration of agent technology within a volunteer grid in an attempt to solve the problem of large scale printed cursive document OCR using the DTW algorithm.

The performance evaluation analysis and numerical experiments showed that our proposed SOGA architecture provides a viable framework to speed up the recognition of very large printed cursive writing documents based on the DTW algorithm. Moreover, this SOGA eases the way to improve, at will, the recognition rate of the DTW algorithm by making it possible to combine it with some other complementary

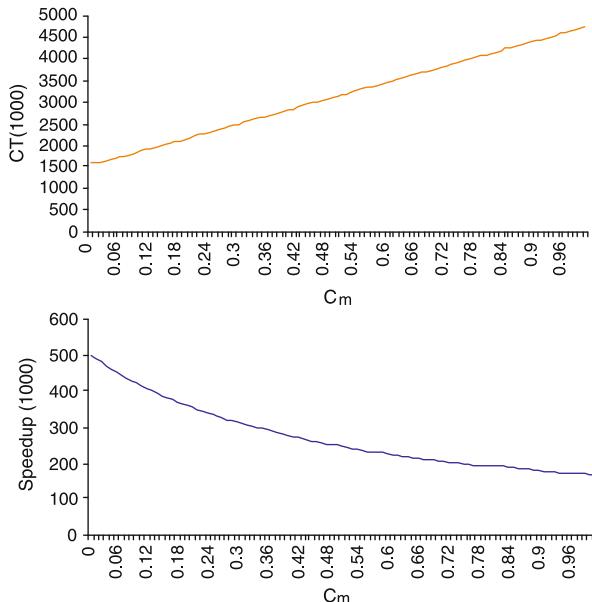


Fig. 15.7 Case where the number of OCR service agents = 1000

approaches or techniques. The number of service agents to be involved to reach an adequate completion time depends on both the underlying network and the current utilization of each of these agents.

References

1. Abdennadher, N.: Vers un outil Peer-To-Peer orienté calcul intensif. In: Flash Informatique, EPFL, August 2005
2. Abdennadher, N.: Using the volunteer computing platform XtremWeb-CH: lessons and perspectives. In: Workshop on Grid Computing: e-Infrastructure, Applications and Research, ES-STT, UTIC, Tunisia (2007)
3. Abdennadher, N.: XtremWeb-CH: une plateforme global computing pour les applications de haute performance. Internal report, August 2004, HES-SO/EIG
4. Al-Badr, A., Haralick, R.: A segmentation-free approach to text recognition with application to Arabic text. Int. J. Doc. Anal. Recognit. **1**(3), 147–166 (1998)
5. Alves, C.E.R., et al.: Parallel dynamic programming for solving the string editing problem on CGM/BSP. In: Proc. SPAA'02, Winnipeg, Manitoba, Canada, August 10–13, 2002
6. Amin, A.: Off-line Arabic character recognition: the state of the art. Pattern Recognit. **31**(5), 517–530 (1998)
7. Available at <http://www.nla.gov.au/>
8. Available at <http://www.ocrgrid.org/>
9. Available at <http://www.kirtas.com/>
10. Available at <http://code.google.com/p/ocropus/>
11. Available at <http://www.xtremwebch.net>
12. Available at <http://www.xtremweb.net>

13. Available at <http://boinc.berkeley.edu>
14. Bahi, J.M., et al.: Synchronous and asynchronous solution of a 3D transport model in a grid computing environment. *Appl. Math. Model.* **30**, 616–628 (2006). Available on www.sciencedirect.com
15. Bertis, V., Bolze, R., Desprez, F., Reed, K.: Large scale execution of a bioinformatic application on a volunteer grid. In: *IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008*, 14–18 April 2008, pp. 1–8 (2008)
16. Bradford, P.G.: Efficient parallel dynamic programming. In: Proc. 30th Annual Allerton Conference on Communication, Control and Computing, University of Illinois, Urbana, IL, USA, pp. 185–194 (1992)
17. Bridle, J.S., et al.: An algorithm for connected word recognition. In: Proc. IEEE, ICASSP, pp. 899–902, May 1982
18. Buyya, R., et al.: A gentle introduction to grid computing and technologies. In: Proc. CSI, India, May 7–19, 2005
19. Capello, F.: The evolution of GRID5000. In: *Workshop on Grid Computing: e-infrastructure, Applications and Research, ESSTT, UTIC Tunisia* (2007)
20. Cheng, H.D., et al.: VLSI architecture for pattern matching using space-time domain expansion approach. In: Proc. IEEE Int. Conf. Computer Design VLSI Comput, NY, Oct 7–10, 1985
21. Cheng, H.D., et al.: VLSI architecture for dynamic time-warp recognition of handwritten symbols. In: *IEEE ASSP*, vol. 34, Jan 1986
22. Cheng, H.-D., et al.: A VLSI architecture for dynamic time-warp recognition of handwritten symbols. *IEEE Trans. Acoust. Speech Signal Process.* **34**(3), 603–613 (1986)
23. Cheung, A., et al.: An Arabic optical character recognition system using recognition based segmentation. *Pattern Recognit.* **34** (2001)
24. Coté, M., et al.: Une architecture Multi-agents réutilisables pour les environnements riches en informations. In: *Série Scientifique, Montréal*, July 2002
25. Fedak, G., He, H., Cappello, F.: BitDew: a programmable environment for large-scale data, management and distribution. Technical report N 6427, INRIA, January 2008
26. Ferber, J.: *Les Systèmes Multi-agents. Vers une Intelligence Collective*. InterEditions, Paris (1995)
27. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid. *Int. J. Supercomput. Appl.* (2002)
28. Foster, I., Jennings, N.R., Kesselman, C.: Brain meets brawn: why grid and agents need each other. In: Proc. AAMAS'04, Leeds, UK, March 2004
29. Gonzalez, R.C., Wintz, P.: *BitDew: Digital Image Processing*. Addison-Wesley, Reading (1988)
30. Hasegawa, W.: Research on Mobile Agent Security (510080). School of Information Science, Japan Advanced Institute of Science and Technology, February 2007
31. Hayes, C.C.: Agents in a nutshell—a very brief introduction. *IEEE Trans. Knowl. Data Eng.* **11**(1), 127–132 (1999)
32. <http://www.globus.org>
33. IBM: Introduction to grid computing with Globus. IBM RedBook. SG24-6895-01. ISBN 0738427969, September 2003
34. Jarras, I., Chaib-draa, B.: Aperçu sur les systèmes Multi-agents. In: *Série Scientifique, Montréal*, July 2002
35. Jonquet, C.: Dynamic service generation: agent interactions for service exchange on the grid. Ph.D. dissertation, University of Montpellier II, November 2006
36. Kanoun, S., et al.: Reconnaissance d'images de textes Arabes par approche Affixale. In: Proc. MCSEAI'04, Sousse, Tunisia, May 9–12, 2004
37. Khemakhem, M.: Reconnaissance globale de caractères imprimés Arabes et Latins par comparaison dynamique. In: Proc. Regional Conference on Computer Science and Arabization, Tunis, Tunisia, March 1988

38. Khemakhem, M., Belghith, A.: A multipurpose multi-agent system based on a loosely coupled architecture to speedup the DTW algorithm for Arabic printed cursive OCR. In: Proc. AICCSA-2005, Cairo, Egypt, January 2005
39. Khemakhem, M., Belghith, A.: A P2P grid architecture for distributed Arabic OCR based on the DTW algorithm. *Int. J. Comput. Appl.* **31**(1) (2009)
40. Khemakhem, M., et al.: Reconnaissance de caractères imprimés par comparaison dynamique. In: Proc. AFCET, Antibes, Sept. 1987
41. Khemakhem, M., et al.: Arabic type written character recognition using dynamic comparison. In: Proc. 1st Computer Conference, Kuwait, March 1989
42. Khemakhem, M., Belghith, A., Ben Ahmed, M.: Etude et evaluation de deux méthodes de distribution de l'algorithme de comparaison dynamique pour la reconnaissance de caractères Arabes. In: Proc. First Maghrebin Symposium on Programming and Systems, Algeria, October 1991
43. Khemakhem, M., Belghith, A., Ben Ahmed, M.: Modélisation architecturale de la comparaison dynamique distribuée. In: Proc. Second International Congress on Arabic and Advanced Computer Technology, Casablanca, Morocco, December 1993
44. Lesser, V.R.: Cooperative multi-agent systems: a personal view of the state of the art. *IEEE Trans. Knowl. Data Eng.* **11**(1), 133–142 (1999)
45. Mella, P.: Complex systems vs. Simplex systems: the behaviour of collectivities following the combinatory system view. In: Proc. 6th International Conference on Complex Systems (CS02), Complexity with Agent Based Modeling, Chuo University, Tokyo, Japan, September 9–11, 2002
46. Quénot, G.R., et al.: A dynamic programming processor for speech recognition. *IEEE J. Solid-State Circuits* **24**(F9), 20 (1989)
47. Russel, S., et al.: Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs (1995)
48. Shi, Z., et al.: Agent based grid computing. *Appl. Math. Model.* **30**, 629–640 (2006). Available on www.sciencedirect.com
49. Vinciarelli, A.: A survey on off-line cursive word recognition. *Pattern Recognit.* **35**, 1433–1446 (2002)
50. Weis, G.: Multi-agent Systems, a Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge (1999)
51. WolSKI, R.: Dynamically forecasting network performance using the network weather service. Ucsd Technical Report tr-cs96-494, University of California, San Diego, La Jolla, CA, January 7, 1998
52. Wooldridge, M., et al.: Intelligent agents: theory and practice. *Knowl. Eng. Rev.* **10**(2), 115–152 (1995)
53. Wooldridge, M., et al.: A methodologie for agent oriented analysis and design. In: Proc. Third International Conference on Autonomous Agents (Agents'99), pp. 69–76. ACM, Seattle (1999)
54. Worldlanguage products. Available at <http://www.worldlanguage.com/ProductsArabic/OCR/Page1.htm>

Part III

Evaluation

Chapter 16

Data Collection and Annotation for Arabic Document Analysis

Ilya Zavorin and Eugene Borovikov

Abstract The creation of good quality document corpora is not a trivial task, but such corpora are essential for advancing OCR technology. Documents in Arabic certainly present their own challenges to this process, and here we describe our data creation and annotation efforts for Arabic document analysis. The resulting corpora include both on-line and off-line handwritten data as well as logos, signatures, and mixed-script machine-printed text. All these are described in detail, and some typical examples of documents are given.

16.1 Introduction

Access to good quality corpora is fundamental for estimating and improving the accuracy of document analysis algorithms. A quality document corpus typically consists of a set of document images, each accompanied by its *ground truth*. The ground truth typically includes the document's source text (preferably line-aligned with the image) and may also include information about fonts, scripts, languages being used, text and non-text block boundaries, document reading order, etc. Corpora with a rich ground truth are difficult to find or collect, but their benefits for the development of document analysis technology are hard to overestimate. Fortunately, sophisticated corpus generation [16] and creation [5] tools are becoming more accessible, making the creation of an elaborate ground truth much easier and more effective.

One of the main purposes for creating publicly available document corpora is uniform benchmarking and comparison of various document analysis systems. When such corpora are developed for a particular document analysis task to be evaluated (such as off-line handwriting recognition or document layout analysis), there is often a bias as to what type of data and/or annotation is actually generated. The reasons for the bias are, *first*, limited access to raw data and, *second*, the difficulty of

I. Zavorin (✉) · E. Borovikov

Knowledge and Information Management Division, CACI, 4831 Walden Lane, Lanham,
MD 20706, USA

e-mail: izavorin@CACI.com

E. Borovikov

e-mail: yborovikov@CACI.com

defining what a “typical” test document of a particular type is in terms of its textual content, layout, etc.

In addition, since these corpora are usually used to evaluate technologies that are still maturing, the test data often must be simplified to provide a reasonable level of complexity for systems to be evaluated. As a result, such corpora may not represent well the type of documents that a specific potential user of these document analysis technologies might actually encounter.

Compared to Latin-script languages such as English and Spanish, there are relatively few large-scale publicly available test corpora available for Arabic document analysis. One such corpus is the University of Maryland Arabic database, which includes, among document sets in several European, Asian, and Middle Eastern languages, 166071 Arabic handwritten business documents with signatures [9]. A collection of 350 documents signed by 70 different persons with both Persian or Arabic cursive signatures was used to evaluate a methodology for signature extraction and verification [3]. The images included mixed alphanumeric content in Arabic, Persian, and English using different fonts and sizes as well as logos and lines. Two other corpora designed to evaluate bank check processing are briefly described by Lorigo et al. [10].

In the context of Arabic handwriting recognition (HWR), one of the important publicly available corpora is the IFN/ENIT database of handwritten Arabic words [6, 15], which includes 26459 segmented words containing Tunisian town/village names (with ground truth) written with a great variations in handwriting styles by 411 writers. This database is free for academic researchers and can be licensed for corporate research as well. The existence of this data set allowed its creators to conduct several Arabic HWR competitions, where their HWR system competed with others [1, 11–14]. They also showed [4] how to optimally combine several Arabic HWR systems to attain impressive recognition accuracy rates beyond 95 %.

In what follows, we present results of data creation and annotation efforts for Arabic document analysis that we undertook driven by the needs of specific government clients. The resulting corpora can be separated into the following groups based on the document analysis tasks they were developed for: optical character recognition (OCR, Arabic News and Anfal corpora), and intelligent character recognition (ICR, AMA and OnAR corpora).

16.2 OCR Data

Here we describe our OCR corpus collection and annotation efforts. For each corpus, we present a motivation for creating it, discuss why it is unique, and reveal some details of our collection and annotation process.



Fig. 16.1 A sample Arabic News corpus document

16.2.1 Arabic News Corpus

The *Arabic News* corpus was created primarily to develop and test the capabilities of pre-OCR image processing, including adaptive noise removal, page de-skewing, border detection, and reading order determination. It presents a good testing platform for Arabic OCR robustness to variations in fonts and text layout.

The corpus was collected by scanning newspaper articles and saving the results in both grayscale and bitonal single page Tagged Image File Format (TIFF) images—one article per image. The rest of the newspaper page was masked by blank paper. The ground truth was prepared and line-aligned with the source images by a native Arabic speaker and stored into plain Unicode text files—one text file per source image.

The News corpus contains a total of 100 pages with various news articles. A sample of the Arabic News document is shown in Fig. 16.1. As we can observe, the

image captures a nonrectangular region of the news article, exhibiting a variety of fonts and various kinds of noise. It was successfully utilized in our pre-OCR adaptive image enhancement system called ImageRefiner [2].

16.2.2 The Anfal Corpus Collection

We developed this corpus as part of a larger project for a government client. The ultimate goal of the project was to assess how document image processing could be improved to yield more relevant content within a large document exploitation system that we had previously developed for the client. To this end, a thorough evaluation had to be performed of a number of commercial and academic software tools developed for the following document image processing tasks:

- Logo recognition
- Signature recognition
- Writing system (script) and language identification
- Edge and border detection/recognition

While experiments involved data in various languages, Arabic documents were the primary focus of the effort. For each of the image processing tasks listed above, a corpus of realistic data had to be created that reflected as much as possible the type of data that actually occurred within the exploitation system. The approach we took was to start with a single set of document images and create multiple derived corpora by adding ground truth metadata corresponding to each image processing task. As the name of the corpus suggests, the initial data set, which was provided to us by the client, consisted of various documents related to the Al-Anfal Campaign [7]. These documents had been initially indexed with Zyindex [8], so the data set included at least some limited ground truth information.

Logo Recognition

We assembled a set of 1058 document images composed of bitonal TIFF images selected from the initial Anfal collection. When making selections, we attempted to create a heterogeneous set of test images with varying

- Logo shapes and sizes
- Logo locations in a document
- Degrees of overlap with other structures, such as handwritten notes
- Logo counts in a single document, i.e. documents with no logos, documents with a single logo, documents with multiple logos

Among the documents chosen, approximately 18 percent contained one or more logos on the page. We also note that some documents with logos also contained other graphics such as photos (see Fig. 16.2) or borders (see Fig. 16.3). Approximately

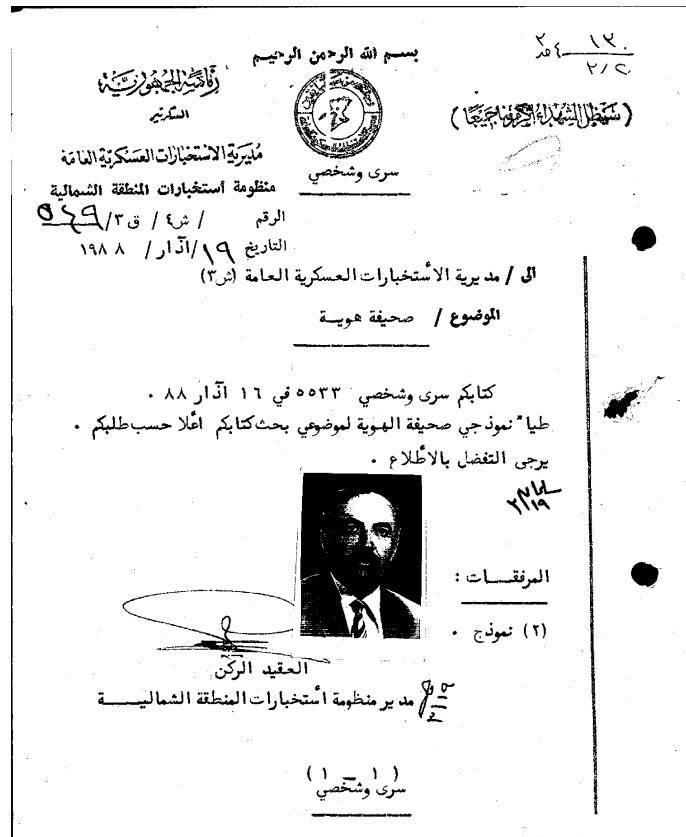


Fig. 16.2 A sample Anfal image with a logo also containing a photo

17 percent of the documents with logos featured an official Iraqi government agency stamp (see Fig. 16.4). Information about the presence or absence of a logo in a given image was simply recorded in a spreadsheet; no location information or any other additional metadata was recorded about the image. In addition, for a more detailed logo matching evaluation, we extracted two logos: the Iraqi Republic logo and the Iraqi logo featuring the head of a bird inside a ring (see Fig. 16.5). These logos were chosen primarily for their prevalence throughout the collection, the republic logo appearing 106 times, and the “bird circle” logo appearing 123 times.

Signature Recognition

We collected 399 Arabic images with signatures from the Anfal collection, each of which contained between one and seven signatures. In general, it should be noted

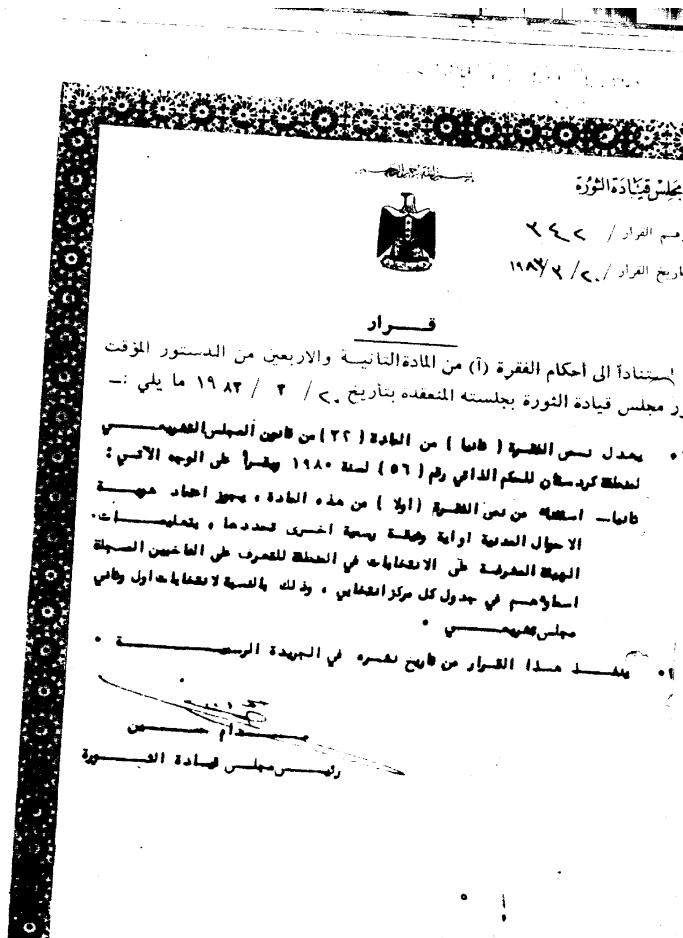


Fig. 16.3 A sample Anfal image with a logo also containing a border

that collection of a corpus containing signature samples is a very laborious task. The decision of whether or not a particular signature is present in an image is sometimes subjective due to high variability in handwriting, poor image quality, signatures often being obstructed by stamps and/or noise, and a lack of other identification clues, such as the printed name of the signer (see examples in Figs. 16.6 and 16.7). This often forces extraction of signature image data from document images that are relatively clean and noise-free.

Script and Language Identification

In addition to 276 Arabic script mixed-language documents picked from the Anfal collection, we collected two additional sets of Arabic script documents. One was

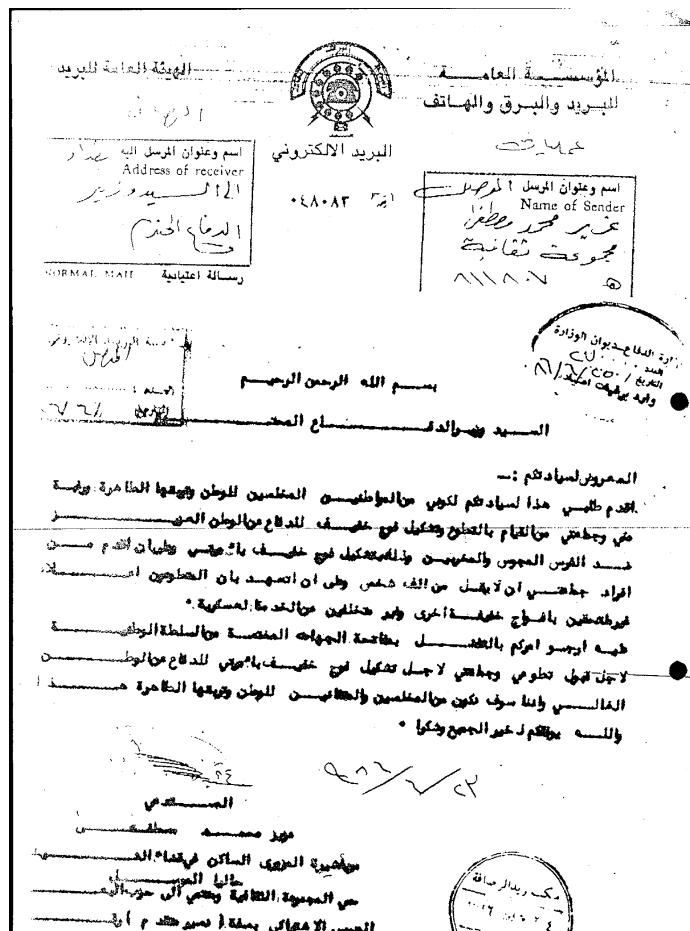


Fig. 16.4 A sample Anfal image with a logo also containing a stamp

very small, containing only five documents with mixed Arabic and English text (see Fig. 16.8), while the other one contained 396 documents written in Urdu with various levels of noise and distortion in the images (see Figs. 16.9 and 16.10). The specific script and language of each document were recorded in a single spreadsheet.

Edge and Border Detection/Recognition

The assembled experimental corpora included two sets of 200 images each from the Anfal collection and 8 miscellaneous images from various sources. The images of

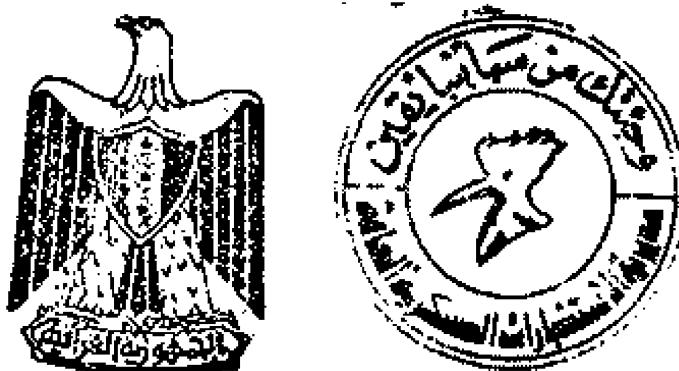


Fig. 16.5 Sample Republic of Iraq (*left*) and “bird in circle” (*right*) logos

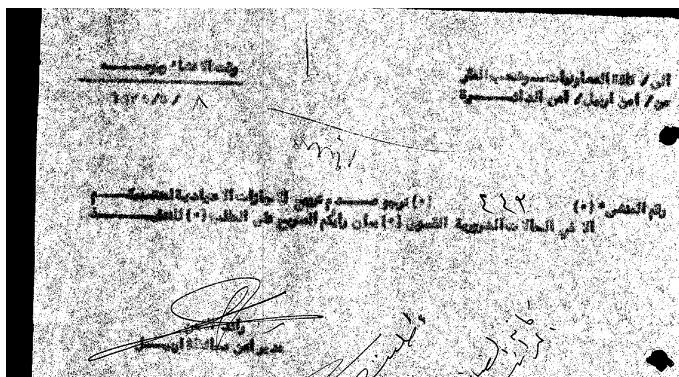


Fig. 16.6 A sample Anfal signature image containing significant noise

the Anfal collection are mixed, including clean backgrounds and noisy (speckled or salt-and-pepper) backgrounds (see Figs. 16.11 and 16.12). The borders of the images are solid black. Each image contains one to four borders. The 8 extra images contain different types of borders and were used to test the reliability of border removal.

16.3 Arabic Handwritten Data

Our Arabic handwritten data collection effort was limited to the scope of the Arabic handwriting recognition project. Hence the resulting corpora are by no means comprehensive, but they may provide a good test platform for various off- and on-line Arabic recognition endeavors.

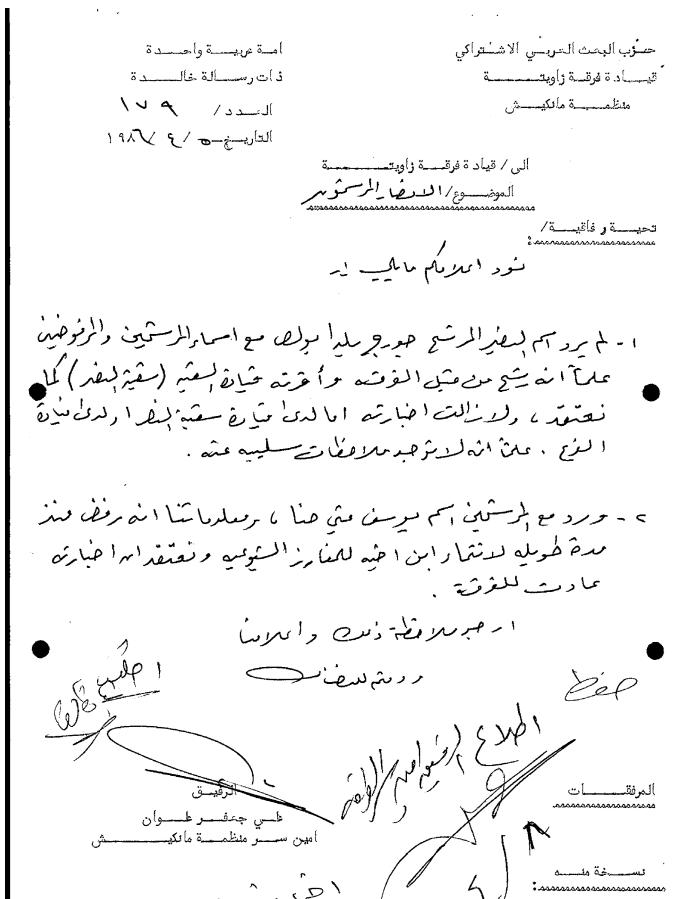


Fig. 16.7 A sample Anfal signature image containing a significant amount of other handwritten text

16.3.1 AMA Data Set

Many Arabic handwriting recognition and evaluation efforts use the popular IFN/ENIT corpus, available freely in the academia. Although this corpus is relatively large and displays a fair amount of variability, thus serving as a great test bench for many projects, it still has the disadvantage of not reflecting accurately the types of writing likely to be encountered in real-world applications other than postal processing.

To support more general Arabic handwriting recognition research, we created an annotated test corpus to exhibit a large variety of documents (including notes, personal letters, business correspondence, item lists, etc.) typically encountered by our clients. Due to the sensitivity of client data, we could not use samples of real documents to populate the corpus or even as a starting point for synthetic data gen-

أمثلة ومسائل
الدرس الرابع والعشرين

- | | | |
|----|--|-----------------------------|
| ١٠ | سَيَسْتَقِيلُ فِي السَّاعَةِ الثَّانِيَةِ | <i>at (signifying time)</i> |
| ١١ | نَحْنُ فِي الصَّفَّ | <i>in (place)</i> |
| ١٢ | رُوَيْتُ فِي يَوْمِ الْعِيدِ | <i>on (time)</i> |

Prepositions covered thus far fall into two categories: those prefixed to the noun they modify and those that stand independent of it. They include the following, starting with attached prepositions, each listed with an example. Remember that a noun modified by a preposition is in the genitive case: (مسجل) (مُسجَّل).

١٣	اَكْتَبَ بِالْقَلْمَنْ	<i>write with</i>
١٤	اَسْكَنْ بِبَيْرُوْتِ	<i>live in (place)</i>
١٥	اَشْتَرَى بِالْكِتَابِ بِخَمْسِينْ دُولَارِ	<i>for \$50.</i>
١٦	اَشْتَرَى بِالْكِتَابِ بِخَمْسِينْ دُولَارِ	<i>by plane</i>
١٧	وَصَلَّى بِالْمَطَافِرِ	<i>in Arabic</i>
١٨	تَكَلَّمُ بِالْلُّغَةِ الْعَرَبِيَّةِ	<i>Quoted the book in Arabic.</i>
١٩	كَانَتْ اِنْتَ لِيْ	<i>as you know.</i>
٢٠	كَانَتْ اِنْتَ لِيْ	<i>the teacher, like me, to the student.</i>
٢١	لِسَامِيَّةُ اُرْبَعَةُ اُولَادٍ	<i>Samia has four children.</i>
٢٢	كَتَبَتْ لِجَمِيعِ اَصْدِقَائِيْ	<i>to all my friends.</i>
٢٣	هِيَ مِنْ مَدِيْنَةِ بَغْدَادِ	<i>from the city of Baghdad.</i>
٢٤	هِيَ مِنْ مَدِيْنَةِ بَغْدَادِ	<i>in her room (see examples 9-12).</i>
٢٥	دَعَيْتُ إِلَى الْكِتَبِ	<i>till two.</i>
٢٦	دَعَيْتُ إِلَى الْكِتَبِ	<i>till two.</i>
٢٧	دَعَيْتُ إِلَى الْكِتَبِ	<i>حتَّى</i>
٢٨	دَعَيْتُ إِلَى الْكِتَبِ	<i>about his family.</i>
٢٩	دَعَيْتُ إِلَى الْكِتَبِ	<i>عن أُسْرَتِه.</i>
٣٠	دَعَيْتُ إِلَى الْكِتَبِ	<i>on the table.</i>
٣١	دَعَيْتُ إِلَى الْكِتَبِ	<i>على</i>
٣٢	هُوَ فِي عُرْقَيْنِ	<i>He is in Urquay.</i>
٣٣	دَرَسْتُ حَتَّى السَّاعَةِ الثَّانِيَةِ	<i>درست حتى الساعة الثانية.</i>
٣٤	وَاللَّهِ اَسْأَلُ	<i>by God, I swear.</i>
٣٥	نَفَعَلُ لِلْيَوْمِ عَذَا يَوْمِ الْجَمْعَةِ	<i>except Friday.</i>

Fig. 16.8 A sample Anfal image for script/language identification: a mixed Arabic–English document

eration. Therefore, we adopted the following approach. We reviewed small samples of real-world data to identify sources of variability in terms of visual characteristics likely to affect handwriting recognition: writing implements, the presence of lines and borders on the paper, styles of writing, etc. To the extent possible, we also considered writer demographics, such as origin, age, and gender.

We then developed a target distribution of synthetically generated data for the resulting corpus that would reflect the variety of the samples and pose appropriately realistic challenges for the Arabic off-line handwriting recognition systems that would be evaluated on the data. Clearly, since the amount of real data reviewed was very small, no claims could be made regarding the extent to which the test corpus would be representative of the entire collection of client data. However, given the circumstances of the project, our approach serves as a valid second-best methodology of generating verisimilitudinous test data.

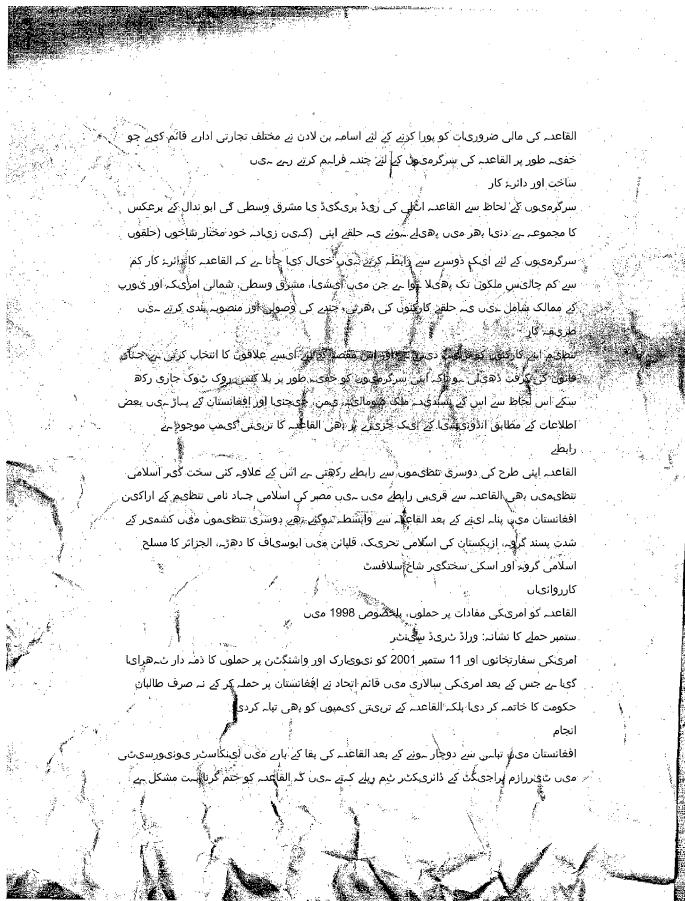


Fig. 16.9 A sample Anfal image for script/language identification: a noisy Urdu document

Table 16.1 contains target specifications for synthetic documents to be created,¹ while Table 16.2 shows the target distributions used when recruiting native Arabic writers to create these documents.

As a result, we have collected a corpus of 5000 distinct page images of handwritten notes, lists, and tables. The ground truth was provided in an XML format and specified the writer and document characteristics applicable to the image, as well as the location (bounding box) and contents of each word and PAW (part of Arabic word). The relevant elements are listed below. Table 16.3 describes the significant

¹The “multipage” designator in Table 16.1 indicates that a page participates in a multi-page logical document, not that we represent the multiple pages as a single unit. Within this collection, each page image is treated separately.

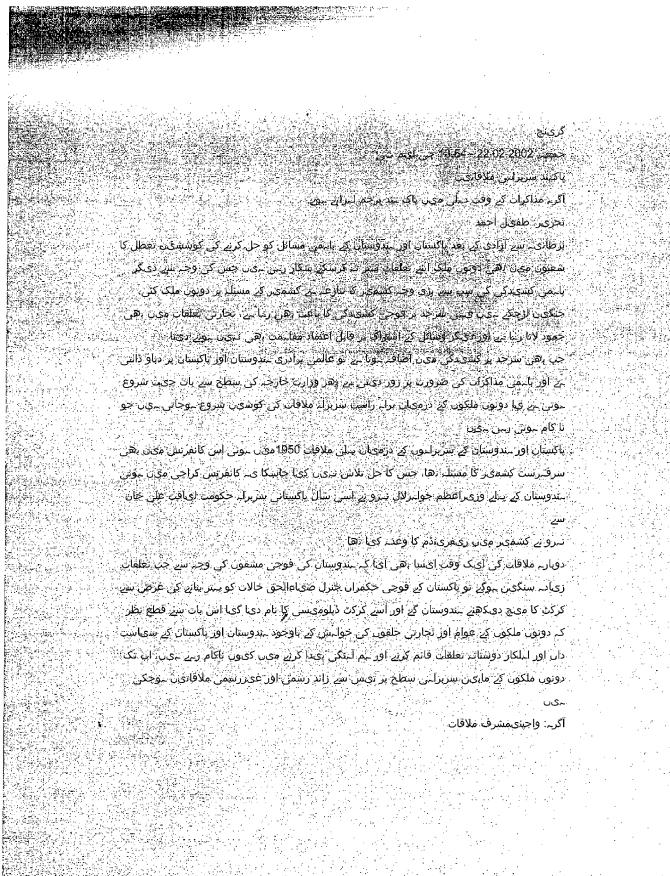


Fig. 16.10 Another sample Anfal image for script/language identification: a noisy Urdu document

attributes of these elements, and Fig. 16.13 shows a sample fragment of a ground truth file.

DL_DOCUMENT occurs once per document.

DL_PAGE occurs once per page image, which in this collection means once per document. It is contained within **DL_DOCUMENT**, and it includes attributes specifying writer information.

DL_ZONE occurs once per word. These elements are contained within the **DL_PAGE** element. Each **DL_ZONE** element specifies the bounding box of its word, the contents of that word, and the positions that separate the word into PAWs, as well as implicitly specifying the contents of the PAWs.

To aid in distinguishing a system's success and failure on particular writing styles from success and failure on particular content, the collection includes multiple writers' variants of the same content. The corpus comprises 200 distinct source docu-

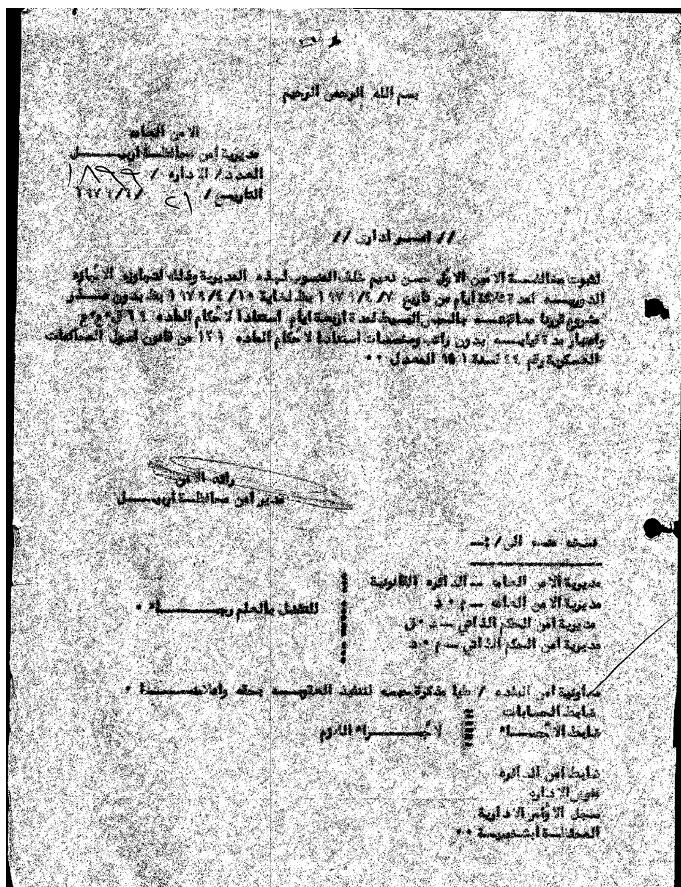


Fig. 16.11 A sample Anfal border image

ments, each written separately by 25 different writers. Each written document produces a single page image. For this purpose “document” and “page” are interchangeable, although the content of a page may be written as though it logically forms part of a longer, multi-page document.

Each writer received a packet including the text to produce and instructions for how to produce it, as a result of a random assignment of document characteristics, and each writer also filled out a log reflecting the creation of the documents and the characteristics used. The ordering of the documents varied in the writers' packets, to account for the effects of fatigue. All paper originals were scanned at 300 dpi.

The XML-formatted ground truth was produced by human interaction with a tool developed by Applied Media Analysis, Inc. (AMA); see Fig. 16.14. Provided with the image and the source text, with the same line breaks in each, the tool allows the user to draw boxes to indicate the image regions that correspond to provided lines in the text, and then to draw boxes around PAW elements, identify the ones

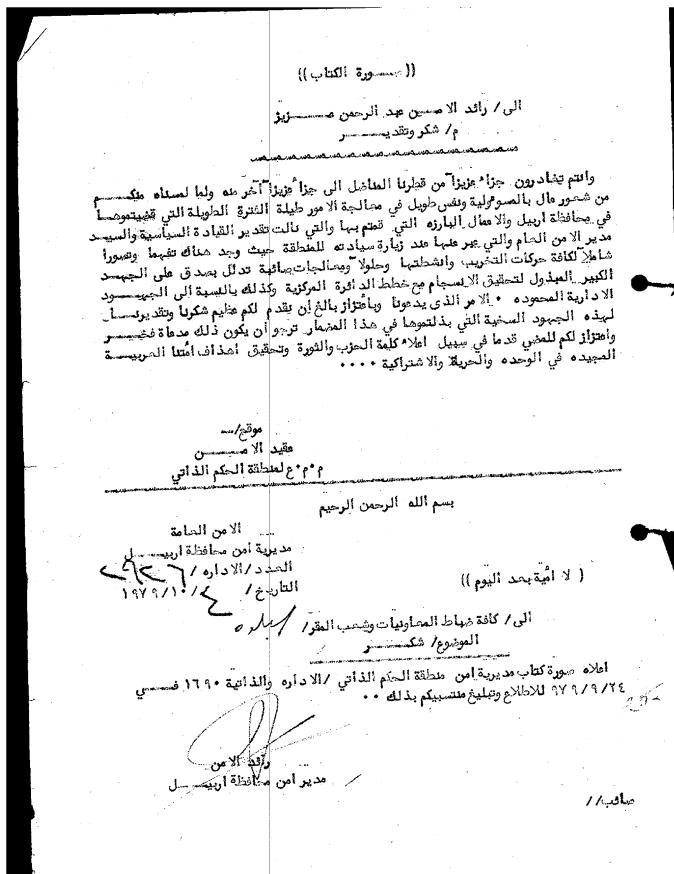


Fig. 16.12 Another sample Anfal border image

they correspond to, and group them into words. The system can also suggest these correspondences, which the user can accept or correct. The tool requires the user to check correctness before releasing a document.

16.3.2 OnAR Data Set for On-Line Recognition

Although the main objective of our research project at the time was to develop a robust recognizer for scanned, i.e., off-line, Arabic handwriting, we still saw some value in gathering a small collection of on-line Arabic handwriting samples that would help us to improve our *glyph tracer* module by observing how natural Arabic handwriting is done. We captured a small corpus of on-line data using the following methods:

Table 16.1 Off-line document characteristics

Characteristic	Value	Percentage
Text Quantity	Half Page	≥20 %
	Multipage	≥10 %
Paper Type	Lined Paper	≥60 %
	White Paper	≥30 %
Background	Logos/Borders	≥5 %
	Normal	≈50 %
Writing Style	Hurried	≈40 %
	Neat	≈10 %
	Normal	≈50 %
Device	Pencil	≥10 %
Format	Marker	≥5 %
	Formal memo	≥20 %
Format	Informal memo	≥20 %
	Lists: Names	≥20 %
Format	Lists: Numbers	≥15 %
	Forms	≥5 %
Format	Poems	≥3 %
	Diagrams	≥2 %

Table 16.2 Off-line writer characteristics

Characteristic	Value	Percentage
Gender	Female	≥20 %
	Male	≥20 %
Origin	North Africa	≈15 %
	Egypt/Levant	≈25 %
Origin	Iraq	≈15 %
	Gulf	≈15 %
Age	Under 35	≥20 %
	Over 50	≥20 %
Education	Advanced	≥4 %

Toshiba a Tablet PC with Windows XP Tablet PC Edition provided a convenient way of capturing natural handwriting from motions of the electronic pen on the tablet PC screen. Advantages of this approach include: quality of captured digital ink (600 dpi spatial resolution and pressure info), rich support for ink handling and rendering, and not having to upload the captured ink to a PC for processing. However, writing on a tablet PC is far from being natural and requires some training.

Pegasus NoteTaker is a small device that captures handwriting that occurs on a sheet of regular paper. The device consists of two pieces: the base (handwrit-

Table 16.3 Ground truth XML elements and attributes

Element	Attribute	Meaning
DL_DOCUMENT	src	Corresponding image file name
DL_PAGE	src	Corresponding image file name
	Origin	Writer regional origin, drawn from the list in Table 16.2, or “Other”
	Age	Writer age
	Gender	Writer gender
	Education	Writer level of education
	Training	Distinguishes native speakers from non-native speakers with native-like abilities. A total of four writers belong to the latter category
DL_ZONE	col	X-coordinate of upper left corner of bounding box of word
	row	Y-coordinate of upper left corner of bounding box of word
	width	Width of bounding box of word
	height	Height of bounding box of word
	offsets	Offsets of boundaries between PAWs within the word, relative to the edge of the word
	contents	Text of the word, with PAWs separated by spaces. To retrieve the content of the word’s PAWs, split this content on spaces; to retrieve the content of the original word, concatenate those elements, or simply remove the spaces

```

<DL_DOCUMENT src="001-020-00020.TIF" docTag="xml" NrOfPages="1">
  <DL_PAGE gedi_type="DL_PAGE" src="001-020-00020.TIF" pageID="1"
    width="2552" height="3298" Origin="Levant" Age="18"
    Gender="F" Education="High School Grad"
    Training="Native">
    <DL_ZONE gedi_type="DL_TEXTLINEGT" id="None"
      col="1866" row="140" width="212" height="74"
      contents="ه خو ۲ ۱" offsets="35,120,195"
      segmentation="word" />
    <DL_ZONE gedi_type="DL_TEXTLINEGT" id="None"
      col="1620" row="145" width="179" height="85"
      contents="ع ز ا ۲ ۱" offsets="27,50,114,164"
      segmentation="word" />
    ...
  
```

Fig. 16.13 Sample fragment of AMA ground truth file

ing collector) and the digital pen (handwriting transmitter). To capture the user’s handwriting or drawing, the base is turned on and positioned in front of the paper sheet. The user takes the digital pen and starts writing or drawing on the piece

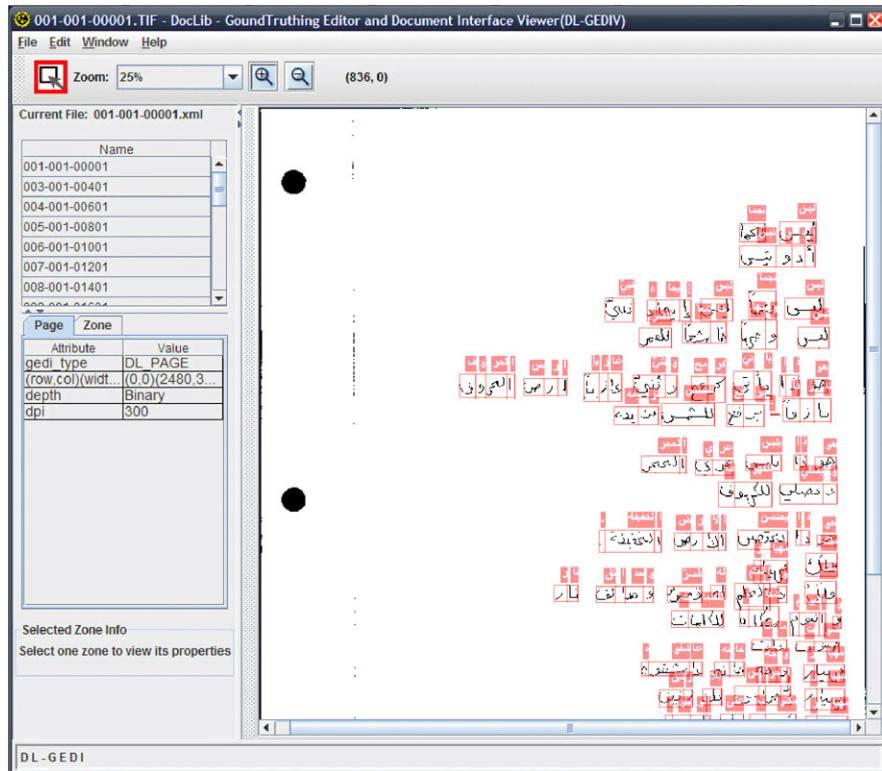


Fig. 16.14 GUI for Arabic handwriting ground truth tool by AMA

of paper. The digital pen acts to the user as a normal pen, yet it transmits all the strokes the user makes to the base. When one page is done, the user touches a special button on the base, the note gets saved, and the base is ready to take the next one. Upon the completion of the data entry session, the base can be connected to a PC to upload all notes it captured for further processing. The major advantage of this device is its natural and noninvasive handwriting capture. There are also some disadvantages, such as fairly low quality of the captured ink (e.g., 100 dpi, no pressure information) and the need for some extra steps, such as uploading ink to a PC, and transforming it to some standard digital ink format.

We have observed natural Arabic handwriting and used our observations to make our off-line glyph tracer more consistent with human writing behavior. As a helping tool, we have implemented a graphical user interface (GUI) on-line handwriting capture and recognition application for a tablet PC; its main window is shown in Fig. 16.15.

The user is shown a printed Arabic word or a phrase that needs to be handwritten, and is prompted to use the tablet PC stylus to write a sample or two of that template phrase. The results containing motion and pressure information are saved into an



Fig. 16.15 GUI for Arabic handwriting on-line capturing

XML file. We have used IFN set A as templates for our Arabic writer and collected about 600 on-line handwritten samples by a Tunisian-native writer.

16.4 Conclusion

Developing all-purpose OCR/ICR corpora is a challenging task, especially when rich ground truth is desired. The rewards of creating such corpora should be obvious, but surprisingly few data sets are publicly available. Here, we presented several data sets that we helped create and annotate during our Arabic document analysis efforts. Most of them were driven by the needs of specific government clients:

OCR Arabic printed text

News Collection of Arabic newspaper articles with a variety of fonts, image noise, and page skews, described in Sect. 16.2.1.

Anfal Predominantly machine-printed documents used for analysis of auxiliary to OCR tasks including language/script identification, border detection, and signature and logo detection as described in Sect. 16.2.2.

ICR Arabic handwriting

AMA Handwritten documents for off-line handwriting recognition, with ground truth granularity of PAW, described in Sect. 16.3.1.

OnAR Handwritten phrases (replicating IFN-A data set) with the movement and pressure information for on-line handwriting recognition as described in Sect. 16.3.2.

The News corpus presents a line of challenges to an Arabic OCR engine and its pre- and post-processing stages, e.g., adaptive image cleanup, segmentation, reading order detection, and font adaptation. The text layout and contents also vary quite significantly.

The Anfal corpus is rich in content and can serve various purposes besides Arabic OCR, e.g., signature, stamp and logo detection, handwriting recognition, and document image noise removal. However, that kind of ground truth requires the development of specialized annotation tools.

In the case of the AMA corpus, we faced the problem of restricted access to the real target documents, and had to resort to detailed descriptions of handwriting in those documents, types of paper, and special marks in order to mimic real-life data. While such data might exist in abundance for a particular document analysis task, access to this data by those who would be in charge of developing the test corpora is often severely restricted due to its personal, proprietary, classified, or other sensitive nature. We described our approach to overcoming this problem and producing a good quality corpus with rich ground truth.

The on-line Arabic recognition set was relatively easy to create with access to a native speaker (capable of operating a tablet PC) and the IFN data set (accompanied by the ground truth). The obvious benefits include access to the motion and pressure information to study handwriting dynamics. Having it based on a well-known data set allows for yet another benchmark test.

All of the described data sets can be available for research purposes via their respective U.S. government agencies, but some restrictions to the foreign parties may certainly apply.

Acknowledgements The authors would like to express their thanks to David Doermann (University of Maryland) for providing an Arabic handwriting ground creation tool, Volker Märgner (TU Braunschweig) for providing access to the IFN database, Leila Saidi and Anna Borovikov (both CACI) for helping with Arabic documents ground truth creation and annotation, and Kristen Summers (CACI) for providing technical expertise and leadership in the area of document understanding in general and in Arabic OCR in particular. The authors also express their thanks to Luis Hernandez of the U.S. Army Research Laboratory for his support of various efforts described here. Part of the research reported in this document was supported by the U.S. Army Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as presenting the official policies or position, whether expressed or implied, of the U.S. Army Research Laboratory or the U.S. Government unless so designated by other authorized documents. Citation of manufacturer or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. Abed, H.E., Märgner, V.: Arabic handwriting recognition competition. *Int. J. Doc. Anal. Recognit.* (2010). Special Issue ICDAR 2009 Competitions
2. Borovikov, E., Zavorin, I.: A multi-stage approach to Arabic document analysis. In: Märgner, V., El Abed, H. (eds.) *Guide to OCR for Arabic Scripts*. Springer, Berlin (2012)

3. Chalechale, A., Naghdly, G., Premaratne, P., Mertins, A.: Cursive signature extraction and verification. In: Second Int. Workshop on Information Technology and Its Disciplines (WITID 2004), Kish Island, Iran, July 2004, pp. 109–113 (2004)
4. El Abed, H., Märgner, V.: Comparison of combination methods of Arabic handwritten word recognizers. In: 5th International Multi-Conference on Systems, Signals and Devices, pp. 1–6 (2008)
5. Fischer, A., Indermühle, E., Bunke, H., Viehhauser, G., Stolz, M.: Ground truth creation for handwriting recognition in historical documents. In: International Workshop on Document Analysis Systems (2010)
6. <http://www.ifnenit.com>
7. http://en.wikipedia.org/wiki/Al-Anfal_Campaign
8. <http://www.searchtools.com/tools/zindex.html>
9. Li, Y., Zheng, Y., Doermann, D., Jaeger, S.: Script-independent text line segmentation in freestyle handwritten documents. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(8), 1313–1329 (2008)
10. Lorigo, L.M., Govindaraju, V.: Off-line Arabic handwriting recognition: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(5), 712–724 (2006)
11. Märgner, V., El Abed, H.: ICDAR 2007—Arabic handwriting recognition competition. In: Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR), vol. 2, pp. 1274–1278 (2007)
12. Märgner, V., El Abed, H.: ICDAR 2009—Arabic handwriting recognition competition. In: Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR), July 2009, vol. 3, pp. 1383–1387 (2009)
13. Märgner, V., El Abed, H.: ICFHR 2010—Arabic handwriting recognition competition. In: Proceedings of the 12th International Conference on Frontiers in Handwriting, November, pp. 709–714 (2010)
14. Märgner, V., Pechwitz, M., El Abed, H.: ICDAR 2005—Arabic handwriting recognition competition. In: Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 70–74 (2005)
15. Pechwitz, M., Snoussi Maddouri, S., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT-database of handwritten Arabic words. In: Proceedings of CIFED, pp. 129–136 (2002)
16. Zi, G., Doermann, D.: Document image ground truth generation from electronic text. *Proc. Int. Conf. Pattern Recognit.* **2**, 663–666 (2004)

Chapter 17

Arabic Handwriting Recognition Competitions

Volker Märgner and Haikal El Abed

Abstract Competitions are a general practice to support the development of new systems. Due to the availability of a database of handwritten words Arabic handwriting recognition systems made a considerable improvement. At first this chapter presents the IFN/ENIT-database which is a standard for Arabic handwritten word recognizer development. The second part includes a presentation of the participating systems and the results achieved at five international competitions from the first one at the International Conference on Document Analysis and Recognition ICDAR 2005 to the competition at the ICDAR 2011. The competitions show a remarkable progress of Arabic handwriting recognition system quality during the seven years. Even though most systems used Hidden Markov Model (HMM) based methods and most times HMM based systems were the winners the overall best result was reached by a Neural Net based technique.

17.1 Introduction

The development of new methods is considerably affected by the measuring techniques used to assess the system quality. Pattern recognition systems in general and Arabic handwriting recognition systems in particular need adapted assessment methods. The two essential parts of the assessment of recognition systems are the used data and the evaluation metrics. Usually commercially initiated system development or system comparisons are the starting point for data collections and system assessment. This was the case for the annual tests of optical character recognition (OCR) accuracy from 1992 to 1996 organized by the Information Science Research Institute (ISRI) at the University of Nevada, Las Vegas, funded by the U.S. Department of Energy with the mission to foster the improvement of automated technologies for understanding machine printed documents. To pursue this

V. Märgner (✉) · H. El Abed

Institute for Communications Technology (IfN), Technische Universität Braunschweig,
Schleinitzstrasse 22, 38092 Braunschweig, Germany
e-mail: v.maergner@tu-bs.de

H. El Abed
e-mail: elabed@tu-bs.de

goal, ISRI conducted research on developing new metrics of recognition performance, the combined use of recognition and retrieval technologies, and an annual OCR technology assessment program where an independent comparison of performance characteristics of all available technologies for character recognition from machine printed documents was performed. This project brought together research groups and companies from all over the world and showed that data, metrics, and technology assessment is a basic requirement to accelerate the development of OCR technology, which made a great leap forward for printed Latin characters.

The situation for Arabic character recognition and especially Arabic handwriting recognition is quite different. While in recent years an increasing interest in Arabic handwriting recognition can be noticed, there is only a minor commercial interest. Commercial system development is funded by governments using a great amount of confidential handwritten documents not available for non-commercial research. For many years researchers at universities used for their experiments only a small amount of data, and the results were not comparable with the results of other groups.

A fundamental change occurred with the publication of the IFN/ENIT-database of handwritten Tunisian town names in 2002 [45]. From that time on, more and more research groups used this freely available data for their research, and as a logical consequence three years later a first competition for Arabic handwritten word recognizer was organized during the International Conference on Document Analysis and Recognition (ICDAR) in 2005 [40]. Since this first competition up until 2011 five competitions were held, and a considerable improvement of the system performance could be observed.

This chapter is organized as follows. In Sect. 17.2 the database, the training, and the test sets are presented in detail. Section 17.3 presents the participating groups and gives a short description of the submitted systems. Section 17.4 describes the tests and the results achieved by the different systems. Finally the chapter ends with concluding remarks and an outlook.

17.2 The IFN/ENIT-Database

The development of recognition systems requires a large amount of data to train and test a system. Real-world data, especially from a bank or a post area, often are confidential and inaccessible for non-commercial research. An alternative is to use artificial data instead of scarce real-world data. For the IFN/ENIT-database a form was developed which allowed a writer to write a city name and a postcode without constraints, which means in a similar quality as town names of an address are written on a letter. This also made the usually time-consuming generation of ground truth (GT) as fast and easy as possible. The names of 937 Tunisian town/villages were written together with the postcode. An example of a filled form is shown in Fig. 17.1. Town names and numbers were extracted automatically, and GT and baseline information were added automatically as well. Finally, the GT and baseline information were verified manually several times by different persons.

Fig. 17.1 An example of a filled form

The form pages are scanned with 300 dpi and converted to black and white (binary) images. The cropped images of the names and the postcode are extracted,

Table 17.1 A dataset entry of the IFN/ENIT-database. The symbols M, B, A, E represent the used character shapes (middle, begin, alone, end position in a word)

and for each name a label is assigned. To support the training and testing process of a recognition system, optimally the label of the name consists of the postcode, the name in Arabic code set ISO 8859-6, and additionally a code that describes the Arabic character shapes. This code uses Latin characters as indexes. “B” stands for beginning, “M” for middle, “E” for end, and “A” for alone/isolated character shapes. An “L” marks the “Chadda”. These descriptors are linked to the Arabic character code with the “_”. The codes for each character are separated by “|”. Additionally, a straight line as baseline estimation is given by left and right y-coordinates where the line cuts the boundary of the name image box. Table 17.1 shows an example of a dataset entry of the IFN/ENIT-database.

The database in version 2.0 patch level 1e (v2.0p1e) consists of 32492 Arabic words handwritten by more than 1000 writers. The words written are 937 Tunisian town/village names [40]. Each writer filled out one to five forms.

17.2.1 The Training Sets

The whole database is divided into different sets *a* to *e*. These sets can be used for the development of recognition systems. Table 17.2 shows some statistical details of the training sets of the IFN/ENIT-database.

17.2.2 The Test Sets

The test dataset for the first competition in 2005 was set *e*; at that time this set was unknown to the competition participants. For the following competitions new datasets, again unknown to all participants, were collected. The words are from the same lexicon as those of IFN/ENIT-database and written by writers who did not

Table 17.2 Number of names, characters, and PAWs appearing in the IFN/ENIT-database v2.0p1e

Set	Names	Characters	PAWs
<i>a</i>	6537	51984	28298
<i>b</i>	6710	53862	29220
<i>c</i>	6477	52155	28391
<i>d</i>	6735	54166	29511
<i>e</i>	6033	45169	22640
Total	32492	257336	138060

Table 17.3 Features of datasets *f*, *s*, *t* and *t*₁

Set	Names	Characters	PAWs
<i>f</i>	8671	64781	32918
<i>s</i>	1573	11922	6109
<i>t</i>	1000	7921	4252
<i>t</i> ₁	100	821	412

Table 17.4 Frequency of number of PAWs

PAWs	Frequency in %		PAWs	Frequency in %	
	Set <i>f</i>	Set <i>s</i>		Set <i>f</i>	Set <i>s</i>
1	4.69	4.32	6	9.11	8.96
2	16.58	15.13	7	3.16	3.50
3	25.82	25.30	8	2.24	2.67
4	23.11	23.67	>8	0.21	0.38
5	15.11	15.77			

contribute to the datasets before. For the test purposes, these data are separated into datasets *f*, *s*, *t*, and *t*₁. Table 17.3 shows some statistics of these sets.

Set *f* was collected in Tunisia, while set *s* was collected in the United Arab Emirates (UAE) at the University of Sharjah. Table 17.4 shows the frequency of PAWs (parts of Arabic words) within each name of the new datasets *f* and *s*. The sets *t* and *t*₁ are subsets of sets *a* to *f* used to measure the processing time of the systems in the competition environment.

17.3 Participating Systems

The following sections give a brief description of the systems submitted to the competitions organized from 2005 to 2011. Each system description was provided by the system's authors and edited (summarized) by the competition organizers. The descriptions vary in length according to the level of detail in the provided source

information. The participants of each competition are presented separately. Some of the groups participated with more or less similar systems in more than one competition; the participating systems are described for each competition separately.

17.3.1 ICDAR 2005

The results of the first competition of Arabic handwriting based on the IFN/ENIT-database were presented at the International Conference on Document Analysis and Recognition ICDAR 2005.

ICRA

The system with the name ICRA (Intelligent Character Recognition for Arabic), which also means read in Arabic, was sent by Ahmad Abdul Kader, employed as a software architect by Microsoft within the handwriting recognition group. His participation in this competition is as a freelancer and not as a Microsoft employee. A short description of the system follows:

- The system uses a novel idea that is inspired by the nature of Arabic writing. It is based on the concept of what is known as PAW (part of Arabic word).
- ICRA is a two-tier recognizer.
- The first tier is a neural net-based PAW recognizer that is aided by a PAW lexicon. The PAW lexicon is extracted from the master village names lexicon.
- The second tier is a neural net-based word recognizer that is aided by another lexicon. The literals (alphabet) of this lexicon are actually the PAWs and not the characters.
- ICRA was trained on sets a, b, and c and tested on set d of the IFN/ENIT-database.

Details of the system are published in [3].

SHOCRAN

The system with the name SHOCRAN (System for Handwritten Optical Character Recognition for Arabic Names) comes from a group of researchers in Egypt. It is declared as a confidential project. The system is trained on all four datasets of the IFN/ENIT-database.

TH-OCR

The system with the name TH-OCR was developed at the State Key Laboratory of Intelligent Technology and Systems, Department of Electronic Engineering, Tsinghua University, Beijing, China, by Pingping Xiu, Hua Wang, Jianming Jin, Yan Jiang, Liangrui Peng, and Xiaoqing Ding.

Based on previous research work on a multilingual document recognition system for Chinese, Japanese, Korean, English, Tibetan, and Uyghur languages, this research work was extended to Arabic OCR. A first step was the development of a printed Arabic document recognition system in the year 2004 [25]. The system structure of the handwritten Arabic OCR system is similar to that of the printed Arabic OCR system, but the key technologies of handwritten character segmentation and recognition are more complex and sophisticated. The system consists of text line, word, and character segmentation, character recognition, and post-processing based on a language model.

As the ICDAR 2005 Arabic Handwriting Recognition Competition is running on a closed dictionary set, recognition results are compared with candidate address items to find the best match. A candidate address is seen as a template, and all characters are checked to find their possible correspondences in the recognition results. Two types of cost are taken into account; one is recognition cost and the other is matching cost.

The research work on handwritten Arabic OCR aims at developing a practical system to digitize handwritten Arabic documents.

UOB

The system with the name UOB was developed at the University of Balamand, Lebanon, by Chafic Mokbel.

The UOB system is a pure hidden Markov model (HMM) system developed for speech recognition at the origin. It uses a complete toolkit like HTK,¹ called HCM. HCM permits the development of large HMM networks and it integrates language modeling. The properties of HCM are published in papers in the speech recognition area, e.g., [43].

The work on handwritten word recognition was started using HCM with a Ph.D. project by Mr. Ramy El-Hajj, who developed the feature extraction module. All the work on handwritten word recognition is being done in tight collaboration with Laurence Likforman-Sulem from ENST-Paris. A paper describing the feature extraction module was presented at ICDAR 2005 [4].

For the UOB system all four datasets are used for training. No confidence measure is implemented.

REAM

The system with the name REAM (Reconnaissance de l’Ecriture Arabe Manuscrite) comes from a group at the Laboratoire des Systèmes et de Traitement du Signal—ENIT, Tunisia. The authors of the system are Sameh Masmoudi Touj, Najoua Es-soukri Ben Amara, Hamid Amiri, and Noureddine Ellouze.

¹<http://htk.eng.cam.ac.uk/>

The system uses a hybrid planar Markov model to adapt to horizontal and vertical variations of the handwritten word. The approach is presented in a journal paper [49].

The principal idea of this approach is the partitioning of handwritten words into five logical horizontal bands which correspond to typical Arabic parts of words like upper and lower diacritics, ascenders, descenders, and median zone. This segmentation is done in a sophisticated way using knowledge about the typical Arabic writing style. Additionally, based on features of the median zone, vertical segmentation points are detected. In the next step for each type of segment a specific technique of feature extraction is adopted. Finally the recognition is realized using the concept of a planar HMM (PHMM). In the paper [49] the first results of the system on parts of the IFN/ENIT-database are reported.

17.3.2 ICDAR 2007

The results of the second competition of Arabic handwriting based on the IFN/ENIT-database were presented at the International Conference on Document Analysis and Recognition ICDAR 2007.

MITRE

Tom Hines and Amlan Kundu from MITRE Corporation, USA, presented a system which implements automatic recognition of off-line unconstrained handwritten Arabic words using over-segmentation of characters and variable duration HMM (VDHMM). First, a segmentation algorithm based on morphology and linguistic information is used to translate the 2D image into a 1D sequence of subcharacter symbols. This sequence of symbols is modeled by one single contextual VDHMM. The output of the VDHMM module is a string of likely characters which are processed by a post-processing module. This module maps the string of characters to a set of hypothesized words from the given lexicon. For hypothesis generation, the Levenshtein string-distance function is used along with a custom substitution cost table computed during the training procedure to reflect character confusion probability.

UOB-ENST

This system was submitted by Chafic Mokbel and Ramy Al-Hajj of the University of Balamand (UOB), Lebanon, and Laurence Likforman-Sulem of École Nationale Supérieure des Télécommunications ENST-Paris, France. The realized handwritten word recognition system is an HMM-based system without pre-segmentation.

The system, called UOB-ENST, was a participant of the ICDAR 2005 competition on Arabic Handwriting Recognition. This year three variants of the UOB-ENST system were presented: a basic variant very similar to what was presented in ICDAR 2005 [4] and two advanced systems that better handle the slanted handwriting [5].

Mie University

Fumitaka Kimura, A. Al-Marakeby, W. Ohyama, and T. Wakabayashi from Mie University, Japan, developed a system with essentially the same recognition algorithm as the lexicon directed word recognition algorithm for English postal words described in [30, 31]. This algorithm was recently applied to Bangla city name recognition [44]. Several modifications for Arabic word recognition were made, such as:

1. Connected components of pre-segmentation result are sorted from right to left in decreasing order of x -coordinates.
2. Parameters for word length estimation are optimized for handwritten Arabic words.
3. Character classifier was retrained using character samples extracted from Arabic words in sets a to e .

ICRA

The ICRA system, developed by Ahmad Abdul Kader, also participated in the ICDAR 2005 competition. It is an Arabic handwriting recognition engine that is inspired by properties specific to the Arabic writing script. The approach is primarily motivated by the Arabic letters' conditional joining rules. A lexicon of Arabic words can be expressed in terms of a new alphabet of PAWs. PAWs can be expressed in terms of letters. The recognition problem is decomposed into two problems that are solved simultaneously. To find the best matching word for an input image, a two-tier beam search is performed. In tier one the search is constrained by a letter-to-PAW lexicon. In tier two, the search is constrained by a PAW-to-word lexicon. The searches are driven by a PAW recognizer [3].

CACI

This recognition system was developed by Ilya Zavorin, Eugene Borovikov, Ericson Davis, and Anna Borovikov of CACI, Knowledge and Information Management Division, Lanham, USA. It is designed to be highly configurable and to allow the user to combine several classifiers making recognition a multi-tier process, in which lexicon reduction is performed at initial stages of the process in order to boost both the accuracy and efficiency of the later stages.

The HMM-based recognizer consists of two major components: a *tracer*, which in many ways simulates natural handwriting, and a *classifier*. The tracer is applied to a word image and performs feature extraction in several steps.

For the classifier a word lexicon was built that consists of discrete HMMs corresponding to unique words that appear in training data. A word HMM model has a left-to-right topology and is a connection of individual character models. Each character model also has a left-to-right topology, and the number of its states depends on the complexity of the corresponding character.

Three different configured systems were prepared for the competition: one system with good generalization, one system with improved top 10 results, and one system with further improved performance.

CEDAR

At the University at Buffalo, Center of Excellence for Document Analysis and Recognition, CEDAR, USA, Sargur N. Srihari, Gregory R. Ball, Harish Srinivasan, and Chen Huang developed the system CEDARABIC. It views each word as a stream of information. Words are encoded and segmented based on ligature and separation points. An HMM engine finds the best match for the image against the lexicon by assigning the (oversegmented) segments to the characters. Each character for the word is scored, and the algorithm minimizes the distance each character is from its ideal, minimizing the distance of the word overall. Character combinations are used for characters not easily horizontally segmented. The dots are stripped off and viewed as an error correction mechanism for the overall word.

Paris V

The system, submitted by Fares Menasri, Nicole Vincent, Emmanuel Augustin, and Mohamed Cheriet, was developed at SIP/CRIP5 University of Paris 5, in collaboration with A2iA SA, France. The system is a hybrid HMM/NN system with grapheme segmentation adapted for Arabic [41].

The main idea is the introduction of a heuristically defined 34 shape alphabet which is intended to take advantage of the shape redundancy of letters in Arabic writing. In this approach, one shape in our alphabet can be shared by several letters in the Arabic alphabet, and the same Arabic letter can be modeled by multiple letters in our shape alphabet, depending on the importance of the variation of the Arabic letter with regard to its position in the word.

Instead of recognizing a sequence of letters, we recognize a sequence of shapes. A dictionary allows us to map a sequence of shapes with its corresponding sequence of letters. This shape alphabet avoids the training of different HMMs to model the same information, and it also allows each model to be trained on more data (as a result of the same model being trained on similar shapes coming from different letters).

Siemens

The Hidden-Markov Recognizer for Arabic script (HMR-A) was submitted by Theophile Alary, Jörg Rottland, and Marc-Peter Schambach from Siemens AG Industrial Solutions and Services, Konstanz, Germany.

The script word recognizer (HMR-A) is the result of experiments with the standard HMM-based script word recognizer (HMR) for Latin script that is widely in use within Siemens AG for postal automation projects.

The system is based on techniques developed in 1993 presented, e.g., in [8, 26]. A feature vector sequence is created by a sliding window, followed by an HMM Viterbi decoding. The letter HMMs are multiple left-to-right models for different writing variants. A series of improvements has been applied to the system, e.g. [46]. During development of the system, the focus has been placed on computational efficiency to make it applicable to large postal applications (up to 40000 mail pieces per hour). Two systems are realized for this competition. The first system is configured as it would run in postal applications; the second system uses a combination of three different feature extraction algorithms.

17.3.3 ICDAR 2009

The results of the third competition of Arabic handwriting based on the IFN/ENIT-database were presented at the International Conference on Document Analysis and Recognition ICDAR 2009.

UOB-ENST

This system was submitted by Chafic Mokbel and Ramy Al-Hajj from the University of Balamand (UOB), Lebanon, and Laurence Likforman-Sulem from Telecom ParisTech, France. The realization of the handwritten word recognition system is a HMM-based system without pre-segmentation.

This system participated as well in ICDAR 2005 and 2007 competitions. In this year four variants of the UOB-ENST system were presented: a basic variant similar to that presented at ICDAR 2005 [4] and two advanced systems developed for better handling of slanted handwriting [5]. The system is an HMM-based system, of analytic type without pre-segmentation. It uses the general-purpose HMM toolkit called HCM [43]. The development of the handwriting systems was carried out within the Ph.D. thesis of Ramy El-Hajj in tight collaboration with ENST-Paris. The advanced version was developed to reduce the recognition errors resulting from slanted handwriting and the erroneous positions of diacritical points and marks. The proposed system comprises two stages: the first stage is for recognition and classification based on the technique of slanted windows (with different angles) to extract the features, and the second stage comprises a combined post-processing step. Different combination methods were used and examined, such as majority vote rules

and Borda count combination operator. In addition, a combination method based on an artificial neural network (ANN) with a multi-layer perceptron was used [6].

REGIM

The Research Group on Intelligent Machines (REGIM) at Ecole Nationale d'Ingénieurs de Sfax (ENIS), University of Sfax, Tunisia, participated with one system, submitted by Abdelkarim ElBaati, Monji Kherallah, Houcine Boubaker, Mahdi Hamdani, Adel M. Alimi, and Abdellatif Ennaji from LITIS, University of Rouen, France. This system is based on the restoration of the temporal order of the off-line trajectory of a word [15]. To benefit from dynamic information, a sampling operation by the consideration of trajectory curvatures is calculated. Studies showed that there is a correlation between the angular velocity $V_\sigma(t)$ and the curve $C(t)$. The curvilinear velocity signal uses beta-elliptical modeling, which was developed for on-line systems [28] to calculate features, for feature extraction. For recognition an HMM-based system using HTK is used [21].

MDLSTM

These systems were submitted by Alex Graves from Techische Universität München, München, Germany. This multilingual handwriting recognition system is based on a hierarchy of multidimensional recurrent neural networks [19]. It can accept either on-line or off-line handwriting data, and in both cases works directly on the raw input without any pre-processing or feature extraction. It uses the multidimensional long short-term memory network architecture [19], an extension of long short-term memory to data with more than one spatio-temporal dimension. The basic structure of the system, including the hidden layer architecture and the hierarchical subsampling method is described in [20].

LSTS

This system was submitted by Samia Snoussi-Maddouri from LSTS group at the Ecole Nationale d'Ingénieurs de Tunis (ENIT), Tunis, Tunisia. This system is called Transparent Neural Network (TNN), combining global and local vision modeling (GVM-LVM) of words [35]. In the forward propagation movement, the GVM proposes a list of words containing structural features characterizing the presence of some letters in the word. Then, in the backpropagation movement, these letters are confirmed or not according to their proximity to corresponding printed letters. The correspondence between the letter shapes and the corresponding printed letters is performed by LVM using the correspondence of their normalized Fourier descriptors [34]. The particularities of the TNN are that it does not use any training steps. It can be used for different languages or different lexicons by a simple change of the content of each layer.

A2iA

The A2iA Arab-Reader system was submitted by Fares Menasri and Christopher Kermorvant (A2iA SA, France), Anne-Laure Bianne (A2iA SA and Telecom Paris-Tech, France), and Laurence Likforman-Sulem (Telecom ParisTech, France). This system is a combination of two different word recognizers, both based on HMM. The first one is a hybrid HMM/NN with grapheme segmentation [32]. It is mainly based on the standard A2iA word recognizer for Latin script, with several adaptations for Arabic script [42]. The second one is a Gaussian mixture HMM based on HTK, with sliding windows (no explicit pre-segmentation). The computation of features was greatly inspired by Al-Hajj's work on geometric features for Arabic recognition [6]. The results of the two word recognition systems are combined to compute the final answer [6].

LITIS-MIRACL

This system was submitted by Yousri Kessentini (LITIS and MIRACL), Thierry Paquet (LITIS, University of Rouen, France), and AbdelMajid Benhamadou (MIRACL, University of Sfax, Tunisia). This word recognition system is based on a multi-stream segmentation-free HMM. Two feature vector sequences are created using a sliding window, and they are simultaneously decoded according to the multi-stream formalism. One stream is composed of density features, while the other is made of contour features [27].

RWTH-OCR

These systems were submitted by Philippe Dreuw, Stephan Jonas, Georg Heigold, David Rybach, and Hermann Ney from RWTH Aachen University, Human Language Technology and Pattern Recognition, Aachen, Germany. Without any pre-processing of the input images, simple appearance-based image slice features X_t at every time step $t = 1, \dots, T$ which are augmented by their spatial derivatives in horizontal direction $\Delta = X_t - X_{t-1}$, are extracted. In order to incorporate temporal and spatial context into the features, 7 consecutive features in a sliding window, which are later reduced by a PCA transformation matrix, are concatenated. System-1 is a multi-pass system. The first-pass system is built using a modified maximum mutual information training criterion. The second-pass is automatically built using a novel unsupervised confidence-based discriminative training criterion on the output of the first-pass system to automatically adapt the model to the unknown testing data [11]. System-2 is an HMM-based handwriting recognition system, in which Viterbi is trained using the maximum-likelihood training criterion. A lexicon with multiple writing variants, where the white spaces between the pieces of Arabic words are explicitly modeled as proposed in [10], is used.

17.3.4 ICFHR 2010

The results of the fourth competition of Arabic handwriting based on the IFN/ENIT-database were presented at the International Conference on Frontiers in Handwriting Recognition ICFHR 2010.

UPV-PRHLT

These systems were submitted by Adrià Giménez Pastor, Ihab Khoury, and Alfons Juan Císcar, from the Universitat Politècnica de València (UPV), València, Spain. They are based on Bernoulli HMMs (BHMMs), that is, HMMs in which conventional Gaussian mixture density functions are replaced with Bernoulli mixture probability functions [17]. Also, in contrast to the basic approach followed in [17], in which narrow, one-column slices of binary pixels are fed into BHMMs, the UPV-BHMM systems are based on a sliding window of adequate width to better capture the image context at each horizontal position of the word image. This new, windowed version of the basic approach is described in [18]. The UPV-BHMM systems were trained from input images scaled in height to 30 pixels (while keeping the aspect ratio), and then binarized by means of the Otsu algorithm. A sliding window of width 9 was applied, and thus the resulting input (binary) feature vectors for the BHMMs had 270 bits [29].

Two systems were submitted: UPV-BHMM (S-ID 1) and UPV-BHMM2 (S-ID 2). They only differ in the way the sliding window is applied. In the UPV-BHMM system, the sliding window is applied at each column of the input image. In the UPV-BHMM2 system, however, the sliding window is repositioned after each application, so as to align its center to the image mass center within the window.

REGIM

The Research Group on Intelligent Machines (REGIM) at the Ecole Nationale d'Ingénieurs de Sfax (ENIS), University of Sfax, Tunisia, participated with a system submitted by Mahdi Hamdani and Adel M. Alimi.

This system is based on HMMs [22], and it is an improved version of the work presented in [21]. The improvement consists of the optimization of the HMM architectures (number of states) using particle swarm optimization (PSO). The maximum likelihood is used as a fitness function by the PSO. The features used are based on the transformation of the pixel values extracted from normalized images using the Karhunen–Loéve transform. More details about the features are presented in [12]. The results of single PSO-HMMs are improved using the combination methods described in [13].

CUBS-AMA

The CUBS-AMA system was submitted by Safwan Weshah and Venu Govindaraju from the Center for Unified Biometrics and Sensors (CUBS), University at Buffalo, The State University of New York, USA, and Huiping Li and David Doermann from Applied Media Analysis, Inc. (AMA), Maryland, USA.

This system is a Gaussian mixture HMM based on the Hidden Markov Model Toolkit (HTK), gradient, structure, and concavity [16]. The feature vector sequences are created using a sliding window. The system achieves better results after feature reduction and applying different states for each model.

RWTH-OCR

Philippe Dreuw, Christian Plahl, Patrick Doetsch, and Hermann Ney from the RWTH Aachen University, Human Language Technology and Pattern Recognition, Aachen, Germany, have submitted the RWTH-OCR systems.

The submitted System-1 is a discriminatively trained system using a modified maximum mutual information training criterion.

First, the hidden Markov model (HMM)-based handwriting recognition system is Viterbi trained using the maximum likelihood training criterion. For Gaussian mixture training in our base system, the authors perform supervised model training by iteratively re-estimating the emission model parameters and splitting of the mixtures. For the discriminatively trained model, the authors use 7 splits with up to 128 densities per mixture and 3 mixtures per character label, resulting in 55,807 densities.

This model is retrained using a discriminative training approach based on the modified maximum mutual information (M-MMI) criterion as presented in [11]. In [23] it is shown that the objective function used in M-MMI training is a smooth approximation to an SVM with hinge loss function which can be iteratively optimized with standard gradient-based optimization techniques like Rprop. In this work, the approximation level and the margin are chosen beforehand and then kept fixed during the complete optimization using the Rprop algorithm.

The authors use a lexicon with multiple writing variants where the white spaces between the pieces of Arabic words are explicitly modeled as writing variants as proposed in [10]. System-1 performs a single recognition pass using the discriminatively trained model without any pruning.

System-2 is an HMM-based handwriting recognition system which is Viterbi trained using the maximum likelihood training criterion as described in [10]. The authors use a lexicon with multiple writing variants where the white spaces between the pieces of Arabic words are explicitly modeled as proposed in [10].

For Gaussian mixture training in our base system, the authors perform supervised model training by iteratively re-estimating the emission model parameters and splitting of the mixtures. For the generatively trained model, the authors use 8 splits with up to 256 densities per mixture and 3 mixtures per character label, resulting in

81,779 densities. System-2 performs a single recognition pass using the maximum likelihood trained model without any pruning.

17.3.5 ICDAR 2011

The results of the fifth competition of Arabic handwriting based on the IFN/ENIT-database were presented at the International Conference on Document Analysis and Recognition ICDAR 2011.

JU-OCR

The JU-OCR system is submitted by Gheith Abandah and Fuad Jamour, from the University of Jordan, Jordan.

JU-OCR is a recognition system for handwritten Arabic text. This system is intended to recognize unlimited vocabulary and is based on explicit grapheme segmentation. It segments a cursive word into the set of graphemes that forms it, then it recognizes each of the graphemes, and it maps a recognized grapheme into the letters that form a word.

JU-OCR uses the segmentation algorithm described in [1]. This algorithm is a rule-based algorithm that utilizes features extracted from the skeleton of Arabic subwords to segment them into a set of graphemes. Most graphemes are forms of the Arabic letters. However, some graphemes are parts of letters (over-segmentation), and some graphemes are vertical ligatures (under-segmentation). Each grapheme has a main body, and some graphemes have secondary bodies. A body is a connected component that forms a blob recognizable by humans. Statistical and morphological features [2] are extracted from each body of the graphemes and passed to a Random Forest (RF) classifier [7] to recognize the body. We use the OpenCV implementation of the RF classifier. After each body is recognized, the bodies are combined to form graphemes. This combination is carried out through rules for what bodies combine to form graphemes. Finally, another set of rules is used to map graphemes into letters. These rules map some graphemes to one letter each, multiple graphemes to one letter, or one grapheme to multiple letters.

As this competition uses a limited vocabulary of 937 words, the authors have developed a string matching algorithm that finds the closest word match of a recognized grapheme sequence without mapping the graphemes to letters. The matching algorithm finds a weighted edit distance between the predicted grapheme sequence and the most probable grapheme sequence of each word in the dictionary. The word that has the minimum distance is the recognition result.

CENPARMI

The CENPARMI-OCR system is submitted from Muna Khayyat, Louisa Lam, and Ching Y. Suen, from the Computer Science and Software Engineering Department,

Concordia University, Center for Pattern Recognition and Machine Intelligence (CENPARMI), Montreal, Quebec, Canada.

The CENPARMI-OCR uses three sets of features appropriate for Arabic handwriting, with each set of features passed to one classifier. The confidence levels and classification results of the classifiers were used for the final classification result. The three sets of features are: gradient features [48], Gabor features [9] and Fourier features [33].

The authors used three different SVMs [47], each of which was trained on a different feature set. The three classifiers are divided into two groups: primary and secondary. The former group consists of the classifier on which the gradient features were trained, while the latter group consists of the two classifiers on which Gabor and Fourier features were trained.

The testing samples are tested on the three classifiers. The system verifies the result of the primary classifier. If the confidence value (posterior probability) of the primary classifier for a sample is below a predetermined threshold, the system verifies the classification results of the two secondary classifiers. If they agree on the class, then the sample would be assigned to this (common) class together with the higher confidence value from these two secondary classifiers.

RWTH-OCR

The RWTH-OCR Arabic Handwriting Recognition System for ICDAR 2011 competition is submitted by Patrick Doetsch, Philippe Dreuw, Mahdi Hamdani, Christian Plahl, and Hermann Ney from RWTH Aachen University, Human Language Technology and Pattern Recognition, Aachen, Germany.

Without any pre-processing of the input images, the authors extract simple appearance-based image slice features X_t at every time step $t = 1, \dots, T$ which are augmented by their spatial derivatives in the horizontal direction $\Delta = X_t - X_{t-1}$.

Due to a character and position dependent length modeling of the 28 base Arabic characters [10], the authors finally model the Arabic words by 121 different character labels. The system described in [11] is used to generate an alignment of the features to the 121 labels.

The raw slice features X_t together with their corresponding state alignments are then processed by a hierarchical MLP framework originally described in [50].

A TRAP-DCT MLP network is based on the MLP framework originally described in [50]. The system uses a TRAP-DCT [24] pre-processing of the raw pixel input features. In order to incorporate temporal and spatial context into the features, the authors concatenate consecutive features in a sliding window, where the MLP outputs are later reduced by a linear discriminant analysis (LDA) transformation.

The hierarchical system uses at the first level a spatio-temporal TRAP-DCT window to augment the 30-dimensional raw pixel input feature vectors to a 240-dimensional vector. The first level hierarchical network uses a single hidden layer with 750 nodes, and 121 output nodes, which are reduced by a log-LDA transformation to 96 components. The second network concatenates these features in addition

to the raw features, and uses a window size of 5 consecutive log-LDA network features, and a window size of 9 consecutive raw input features to account for different spatio-temporal information. The 750-dimensional features (i.e., $96 \times 5 + 30 \times 9$ features) are forwarded to a single hidden layer with 1500 nodes, and finally reduced again by a log-LDA transformation to 36 components.

The HMM-based handwriting recognition system is Viterbi trained using the maximum likelihood training criterion. For Gaussian mixture training in our base system, the authors perform supervised model training by iteratively re-estimating the emission model parameters and splitting of the mixtures. For the discriminatively trained model, the authors use 7 splits with up to 128 densities per mixture and 3 mixtures per character label, resulting in 70745 densities.

REGIM

The Research Group on Intelligent Machines (REGIM) at the Ecole Nationale d'Ingénieurs de Sfax (ENIS), University of Sfax, Tunisia, participated with a system submitted by Mahdi Hamdani, Tarek M. Hamdani, and Adel M. Alimi.

This system is based on HMMs [22], and it is an improved version of the work presented in [21]. The improvement consists of the optimization of the HMM architectures (number of states) using PSO. The maximum likelihood is used as a fitness function by the PSO. The used features are based on the transformation of the pixel values extracted from normalized images using the Karhunen–Loéve transform. More details about the features are presented in [12]. The results of single PSO-HMMs are improved using the combination methods described in [13].

17.4 Tests and Results

Before the results of the competitions are discussed, some organizational details and statistics should be mentioned. All five competitions were organized as closed competitions. Running versions of the recognizers were sent to the competition organizers and tested on a collection of new datasets unknown to the participants. The first competition from 2005 differs slightly from the others, as only datasets *a* to *d* were used for training and *e* for testing. For the following competitions dataset *e* was also made available to the participants, and the new datasets *f* and *s* were collected and from then on used for system performance testing. To test the speed performance of the systems the two subsets *t* and *t*₁ were used. Before we study the competition results in detail, it is interesting to take a look at the number of groups using the IFN/ENIT database. Figure 17.2 shows the number of registered users from the beginning in 2002 until 2011. A steady rise of users can be seen with about ten new users each year. This increasing interest in the IFN/ENIT database is a good basis for a system performance comparison. In Fig. 17.3 the participating groups and the number of systems sent to the competitions are shown. Starting with 6 groups and

Fig. 17.2 Number of registered IFN/ENIT users from the beginning in 2002 until 2011

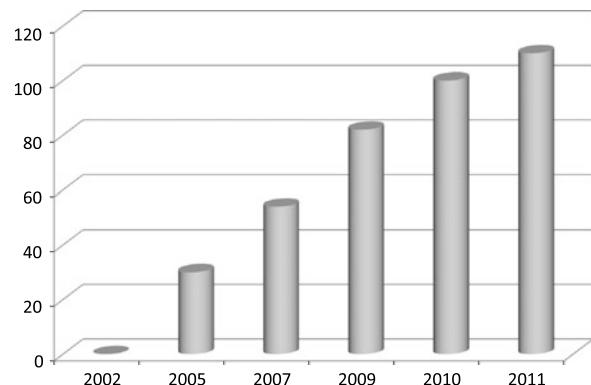


Fig. 17.3 Participating groups and the number of systems sent to the competitions

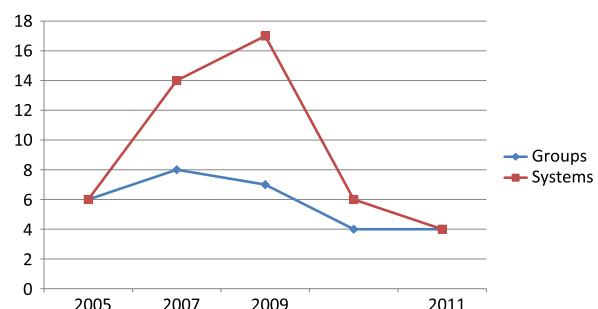


Table 17.5 ICDAR 2005: Recognition results in % of correct recognized images on reference dataset *d* and new dataset *e* and *s*. (G-ID: Group ID, S-ID: System ID)

G-ID	S-ID	Set <i>d</i>			Set <i>e</i>		
		Top 1	Top 5	Top 10	Top 1	Top 5	Top 10
ICRA	1	88.95	94.22	95.01	65.74	83.95	87.75
SHOCRAN	2	100	100	100	35.70	51.62	51.62
TH-OCR	3	30.13	41.95	46.59	29.62	43.96	50.14
UOB	4	85.00	91.88	93.56	75.93	87.99	90.88
REAM*	5	89.06	99.15	99.62	15.36	18.52	19.86
ARAB-IFN	6	87.94	91.42	95.62	74.69	87.07	89.77

6 systems in 2005 a maximum of 17 systems from 7 groups were tested in 2009. In 2011 finally 4 systems from 4 groups were tested. Most participating groups came from universities or independent research teams, and some of them participated in three competitions. But the participation of four groups from companies also shows that there is a commercial interest in Arabic handwriting recognition.

The results of the competitions in detail are shown in Tables 17.5, 17.6, 17.7, 17.8, and 17.9. In the conference proceedings [14, 36–40] the results are also dis-

Table 17.6 ICDAR 2007: Recognition results in % of correct recognized images on reference datasets d and e , new datasets f and s , subsets f_a , f_f , and f_g . The average recognition time per image on subsets t and t_1 is shown in the last two columns. ($ID\ 01$: MITRE; $ID\ 02$ – 04 : CACI; $ID\ 05$: CEDAR; $ID\ 06$: Mie; $ID\ 07$ – 08 : Siemens; $ID\ 09$ – 12 : UOB-ENST; $ID\ 13$: ICRA; $ID\ 14$: Paris V)

ID	Set d	Set e	Set f_a	Set f_f	Set f_g	Set f			Set s			Time (ms)		
						Set f			Set s			Time (ms)		
						Top 1	Top 5	Top 10	Top 1	Top 5	Top 10	Set t	Set t_1	
01	66.34	64.89	62.61	63.79	63.90	61.70	81.61	85.69	49.91	70.50	76.48	10886.4	15815.0	
02	40.45	37.73	12.21	12.71	13.26	11.95	26.44	34.51	8.01	17.17	23.78	852.140	1171.25	
03	70.62	68.62	15.95	17.62	17.92	15.79	21.34	22.33	14.24	19.39	20.53	882.625	1040.46	
04	48.68	44.04	14.64	15.42	16.09	14.28	29.88	37.91	10.68	21.74	30.20	888.171	1099.53	
05	68.07	57.37	59.84	60.67	60.86	59.01	78.76	83.70	41.32	61.98	69.87	34.171	41.71	
06	93.63	86.67	84.38	85.21	85.56	83.34	91.67	93.48	68.40	80.93	83.73	188.439	210.55	
07	91.23	84.27	83.90	84.84	84.97	82.77	92.37	93.92	68.09	81.70	85.19	39.218	61.87	
08	94.58	87.77	88.41	89.26	89.72	87.22	94.05	95.42	73.94	85.44	88.18	109.406	125.31	
09	90.02	81.80	80.58	82.43	82.67	79.10	87.69	90.21	64.97	78.39	82.20	680.781	754.06	
10	92.12	83.52	83.08	84.51	84.56	81.65	90.81	92.35	69.61	83.79	85.89	2174.97	2447.5	
11	92.38	83.92	83.39	84.93	85.18	81.93	91.20	92.76	69.93	84.11	87.03	2172.55	2425.47	
12	93.32	85.13	83.23	84.79	85.29	81.81	88.71	90.40	70.57	79.85	83.34	2191.23	2430.78	
13	88.33	83.87	82.74	83.68	84.14	81.47	90.07	92.15	72.22	82.84	86.27	359.687	402.34	
14	89.80	80.24	81.36	83.45	83.82	80.18	91.09	92.98	64.38	78.12	82.13	383.078	472.18	

Table 17.7 ICDAR 2009: Recognition results in % of correct recognized images on reference datasets d and e , new datasets f and s . The average recognition time in ms per image on subsets t and t_1 is shown in the last two columns. (G-ID: Group ID, S-ID: System ID)

G-ID	S-ID	Set d	Set e			Set f			Set s			Time (ms)		
			Top 1	Top 5	Top 10	Top 1	Top 5	Top 10	Top 1	Top 5	Top 10	Set t	Set t_1	
REGIM	1	92.52	85.38	82.07	89.74	91.22	69.99	81.44	84.68	812.69	841.25			
	2	89.06	81.85	78.16	89.06	91.88	65.61	81.44	85.95	2365.48	2755.01			
	3	89.84	83.52	79.55	90.60	92.16	67.83	83.47	86.65	2236.58	2754.08			
	4	92.59	86.28	83.98	91.85	93.00	72.28	85.19	87.92	2154.48	2651.57			
	5	79.52	63.53	57.93	73.43	78.10	49.33	65.10	71.14	1564.75	1712.15			
A12A	6	93.90	87.25	85.58	92.57	94.12	70.44	82.01	84.87	1056.98	956.82			
	7	94.92	82.21	82.21	91.24	92.47	66.45	80.52	83.13	519.61	1616.82			
	8	97.02	91.68	89.42	95.33	95.94	76.66	88.01	90.28	2583.64	1585.49			
	9	99.72	98.64	91.43	96.11	96.61	78.83	87.98	90.40	115.24	122.97			
	10	99.60	97.60	91.37	96.24	96.61	78.89	88.49	90.27	114.61	122.05			
RWTH-OCR	11	99.94	99.44	93.37	96.46	96.77	81.06	88.94	90.72	371.85	467.07			
	12	99.91	98.71	85.51	93.32	94.61	71.33	83.66	86.52					
	13	99.79	—	85.69	93.36	94.72	72.54	83.47	86.78	17845.12	18641.93			
	14	99.79	—	85.69	93.36	94.72	72.54	83.47	86.78					
	15	96.72	91.25	83.90	—	—	65.99	—	—	542.12	560.44			
LSTS	16	93.04	85.46	82.09	90.27	92.37	74.51	86.14	88.87	143269.81	145157.23			
	17	18.58	14.75	15.05	29.58	35.76	11.76	23.33	29.62	612.56	685.42			

Table 17.8 ICFHR 2010: Recognition results in % of correct recognized images on reference datasets d and e , new datasets f and s . (G-ID: Group ID, S-ID: System ID)

G-ID	S-ID	Approach	Set d	Set e	Set f			Set s		
					Top 1	Top 1	Top 1	Top 5	Top 10	Top 1
UPV PRHLT	1	HMM	98.90	96.04	87.91	93.71	94.72	78.45	87.35	90.40
					99.38	98.03	92.20	95.72	96.29	84.62
REGIM	3	HMM	94.12	86.62	79.03	89.35	91.34	68.44	81.99	84.98
CUBS-AMA	4	HMM	89.97	80.80	80.32	88.26	88.96	67.90	78.58	79.87
RWTH-OCR	5	HMM	99.99	99.77	90.88	95.35	96.04	81.06	90.08	92.63
					99.66	98.84	90.94	95.31	96.00	80.29
										91.80

Table 17.9 ICDAR 2011: Recognition results in % of correct recognized images on reference datasets d and e , new datasets f and s . (G-ID: Group ID, S-ID: System ID)

G-ID	S-ID	Approach	Set d	Set e	Set f			Set s		
					Top 1	Top 1	Top 1	Top 5	Top 10	Top 1
JU-OCR	1	RF	75.49	63.75	63.86	80.18	84.65	49.75	66.86	72.46
CENPARMI-OCR	2	SVM	99.90	99.91	40.00	69.33	74.00	35.52	54.56	63.84
RWTH-OCR	3	HMM	99.67	98.61	92.20	95.73	96.15	84.55	91.99	93.52
REGIM	4	HMM	94.12	86.62	79.03	89.35	91.34	68.44	81.99	84.98

cussed in detail. In the following sections some general aspects of the results are discussed.

17.4.1 The First Competition 2005

The first competition on Arabic handwritten word recognition was presented at ICDAR 2005 [40]. Five systems participated in this competition, and the organizers added their system results for comparison. It is interesting to see that four systems work with comparable good results on the dataset d from training data (Table 17.5). Two systems are different: one system seems to be overtrained, showing 100 % correct recognition, and a second system shows only 30 % correctly recognized names. On the unknown dataset e (see Table 17.5) only two systems show comparable good results with more than 70 % correctly recognized names. The third best result is 65 % followed by 35 %. The results vary very strongly, and even the best are not very good. An analysis of the testset e showed that this set differs in its statistical attributes from the training sets. Hence, for the following competitions set e was added to the training data, and further data were collected for testing.

Fig. 17.4 Tests with dataset *d* at the competitions 2007–2011

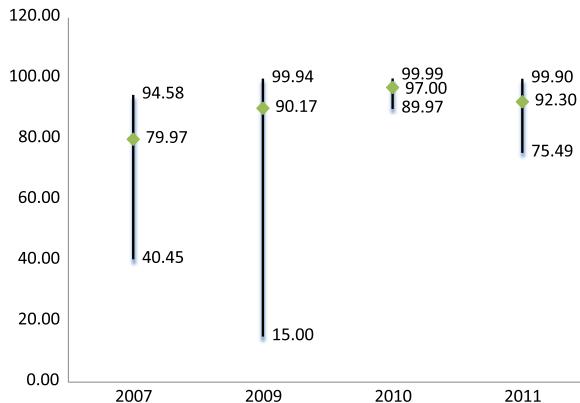
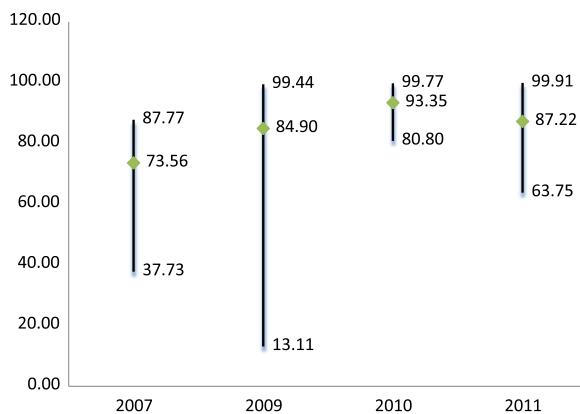


Fig. 17.5 Tests with dataset *e* at the competitions 2007–2011



17.4.2 Tests with Known Data at the Competitions 2007–2011

The comparison of the system performance based on the recognition results of sets *d* and *e*, which are part of the training sets, shows high recognition rates with better results over the years (refer to Tables 17.6, 17.7, 17.8, and 17.9). The results on set *d* are better than on set *e*, but in the competitions 2010 and 2011 the results are very similar and close to a 100 % recognition rate. Figures 17.4 and 17.5 show the maximum, minimum, and mean results of each competition.

17.4.3 Main Tests with Set *f*

The most important test to compare the performance of different systems is, of course, the test using the unknown set *f*. The features of this set are similar to those of sets *a* to *e*. Set *f* was also collected in Tunisia, the same country where the training sets were collected. The statistical recognizer works best if the data used for

Fig. 17.6 Tests with dataset f at the competitions 2007–2011

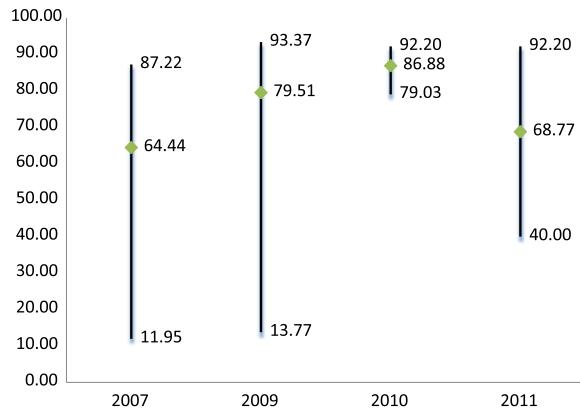
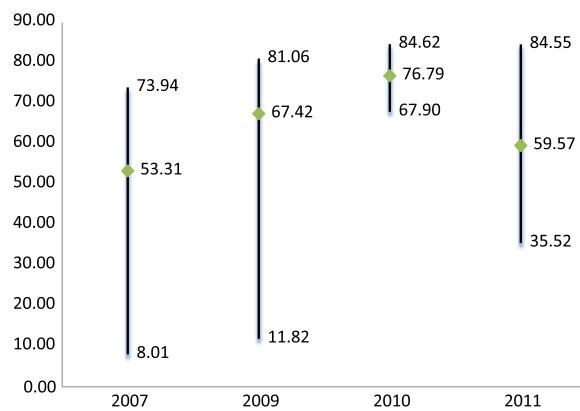


Fig. 17.7 Tests with dataset s at the competitions 2007–2011

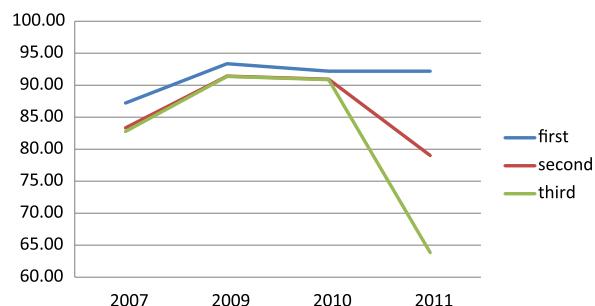


the training of the system parameters are statistically the same as the data used for the tests. The graphic in Fig. 17.6 shows again the maximum, mean, and minimum results of the systems. A constant increase of the mean value from the 2007 to 2010 competitions can be seen, but the best system result was in 2009, and this result was not reached by the winning systems in 2010 and 2011. It is worth mentioning that the three winning systems from 2009 to 2011 all are from different groups.

17.4.4 Robustness Tests with Set s

The test with data collected in the UAE is very interesting. Although all training data were collected in Tunisia, the recognition rate on this set is still high. Figure 17.7 shows a graphic of the system results from 2007 to 2011. The loss of about 12 % of the best system in 2009 compared to the recognition rate on set f is higher than the loss of the two winning systems 2010 and 2011, which is less than 8 %. This shows a better generalization ability of these systems compared to that system.

Fig. 17.8 The three best participating systems at the competitions of 2007–2011



17.4.5 Speed Tests with Sets t and t_1

The processing speed was measured at two competitions using the two test sets t (1000 images) and t_1 (100 images) respectively. The average processing time per name image is shown in the last two columns of Tables 17.6 and 17.7. A substantial difference in speed can be observed. The slowest system is more than 1000 times slower than the fastest one. An average processing time of 34 ms per image is the best result, which is unfortunately combined with a low recognition rate. But the second fastest system with 39 ms per image is combined with a good recognition result on set f . The winning system of the 2009 competition is about ten times slower, but the second best system is only three times slower. As most systems are not speed optimized, these results show a very good performance of the recognition systems and the closeness to commercial systems.

17.5 Conclusions

The competition results show that Arabic handwriting recognition systems have made remarkable progress during the last eight years. Most of the systems participating in the competitions show a very high increasing accuracy, some also in combination with a very high processing speed. Systems are now ready for practical application under the conditions known lexicon of small or medium size and training data with the same statistical appearance as the test data.

Looking at the recognizers, it is notable that most systems used hidden Markov model (HMM)-based methods. These methods are widely used in speech and handwriting recognition. But it is very interesting that the winning system in 2009, with a result which was not reached by the systems in the 2010 and 2011 competitions, used a neural net technology.

Obviously, Fig. 17.8 shows that since 2009 no further improvement of the recognition quality could be achieved. The results of 2011 show new groups trying to get better results with new approaches, but generally a minor interest of the leading groups is seen.

The next steps are now to test the systems with real-world data, with an open lexicon, and with pages of text. But that is another story.

References

1. Abandah, G., Jamour, F.: Recognizing handwritten Arabic script through efficient skeleton-based grapheme segmentation algorithm. In: Proceedings of the 10th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 977–982 (2010)
2. Abandah, G., Malas, T.: Feature selection for recognizing handwritten Arabic letters. *Dirasat Eng. Sci.* **37**(2), 242–256 (2010)
3. Abdulkadr, A.: Two-tier approach for Arabic offline handwriting recognition. In: 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR), pp. 161–166 (2006)
4. Al-Hajj, R., Likforman-Sulem, L., Mokbel, C.: Arabic handwriting recognition using baseline dependant features and hidden Markov modeling. In: 8th International Conference on Document Analysis and Recognition (ICDAR), vol. 2, pp. 893–897 (2005)
5. Al-Hajj, R., Mokbel, C., Likforman-Sulem, L.: Reconnaissance de l’écriture arabe cursive: Combinaison de classificateurs MMCs à fenêtres orientées. In: Colloque International Franco-phone sur l’Ecrit et le Document (CIFED), pp. 271–276 (2006)
6. Al-Hajj, R., Likforman-Sulem, L., Mokbel, C.: Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(7), 1165–1177 (2009)
7. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
8. Caesar, T., Gloger, J.M., Mandler, E.: Pre-processing and feature extraction for a handwriting recognition system. In: 2nd International Conference on Document Analysis and Recognition (ICDAR), pp. 408–411 (1993)
9. Chen, J., Cao, H., Prasad, R., Bhardwaj, A., Natarajan, P.: Gabor features for offline Arabic handwriting recognition. In: Proceedings of the 9th International Workshop on Document Analysis Systems (DAS), pp. 53–58 (2010)
10. Dreuw, P., Jonas, S., Ney, H.: White-space models for offline Arabic handwriting recognition. In: Proceedings of the International Conference on Pattern Recognition (ICPR), pp. 1–4 (2008)
11. Dreuw, P., Heigold, G., Ney, H.: Confidence-based discriminative training for model adaptation in offline Arabic handwriting recognition. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), pp. 596–600 (2009)
12. El Abed, H., Märgner, V.: Comparison of different pre-processing methods for offline recognition of handwritten Arabic words. In: Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR), vol. 2, pp. 974–978 (2007)
13. El Abed, H., Märgner, V.: Improvement of Arabic handwriting recognition systems—combination and/or reject? In: Proceedings of the Document Recognition and Retrieval XVI, vol. 7247–10 (2009)
14. El Abed, H., Märgner, V.: ICDAR 2009—Arabic handwriting recognition competition. *Int. J. Doc. Anal. Recognit.* **14**(1), 3–13 (2011). Special Issue on Performance Evaluation
15. Elbaati, A., Kherallah, M., El Abed, H., Ennaji, A., Alimi, A.M.: Arabic handwriting recognition using restored stroke chronology. In: International Conference on Document Analysis and Recognition (ICDAR) (2009)
16. Favata, J., Srikanthan, G., Srihari, S.: Handprinted character/digit recognition using a multiple feature/resolution philosophy. In: Proceedings of the International Workshop on Frontiers in Handwriting Recognition (IWFHR), pp. 57–66 (1994)
17. Giménez, A., Juan, A.: Embedded Bernoulli mixture HMMs for handwritten word recognition. In: Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR), pp. 896–900 (2009)
18. Giménez, A., Khouri, I., Juan, A.: Windowed Bernoulli mixture HMMs for Arabic handwritten word recognition. In: Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR), Kolkata, India, November 2010
19. Graves, A.: Supervised sequence labelling with recurrent neural networks. Ph.D. thesis, Fakultät für Informatik, Technische Universität München (2007)

20. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in Neural Information Processing Systems 21 (2009)
21. Hamdani, M., El Abed, H., Kherallah, M., Alimi, A.M.: Combining multiple HMMs using on-line and off-line features for off-line Arabic handwriting recognition. In: Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR), pp. 201–205 (2009)
22. Hamdani, M., El Abed, H., Hamdani, T.M., Märgner, V., Alimi, A.M.: Improving a HMM-based off-line handwriting recognition system using MME-PSO optimization. In: Proceedings of the Document Recognition and Retrieval XVIII (2011)
23. Heigold, G., Deselaers, T., Schlüter, R., Ney, H.: Modified MMI/MPE: A direct evaluation of the margin in speech recognition. In: Proceedings of the International Conference on Machine Learning, pp. 384–391 (2008)
24. Hermansky, H., Sharma, S.: TRAPs—classifiers of temporal patterns. In: Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP), pp. 1003–1006 (1998)
25. Jin, J., Wang, H., Ding, X., Peng, L.: Printed Arabic document recognition system. In: Proc. of SPIE-IS&T Electronic Imaging, vol. 5676, pp. 48–55 (2005)
26. Kaltenmeier, A., Caesar, T., Glover, J.M., Mandler, E.: Sophisticated topology of hidden Markov models for cursive script recognition. In: 2nd International Conference on Document Analysis and Recognition (ICDAR), pp. 139–142 (1993)
27. Kessentini, Y., Paquet, T., Benhamadou, A.: A multi-stream HMM-based approach for off-line multi-script handwritten word recognition. In: Proc. 10th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 147–152 (2008)
28. Kherallah, M., Haddad, L., Alimi, A.M., Mitiche, A.: Towards the design of handwriting recognition system by neuro-fuzzy and beta-elliptical approaches. In: Proc. of the Artificial Intelligence Applications & Innovations (AIAI), pp. 187–196 (2004)
29. Khoury, I., Giménez, A., Juan, A.: Arabic handwritten word recognition using Bernoulli mixture HMMs. In: Proceedings of the 3rd Palestinian International Conference on Computer and Information Technology (PICCIT), February 2010
30. Kimura, F., Tsuruoka, S., Chen, Z., Shridhar, M.: Context directed handwritten word recognition for postal service applications. In: 5th Conference on Advanced Technology, pp. 199–213 (1992)
31. Kimura, F., Shridhar, M., Chen, Z.: Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words. In: 2nd International Conference on Document Analysis and Recognition (ICDAR), pp. 18–22 (1993)
32. Knerr, S., Augustin, E.: A neural network-hidden Markov model hybrid for cursive word recognition. In: Proc. Fourteenth International Conference on Pattern Recognition, vol. 2, pp. 1518–1520 (1998)
33. Lavrenko, V., Rath, T.M., Manmatha, R.: Holistic word recognition for handwritten historical documents. In: Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL), January 2004, pp. 278–287,
34. Maddouri, S.S., Amiri, H.: Une Méthode de reconnaissance de mots manuscrits Arabes par réseaux de neurones transparent. In: Conf. Inter. en Informatique (1999)
35. Maddouri, S.S., Amiri, H.: Combination of local and global vision modelling for Arabic handwritten words recognition. In: International Workshop on Frontiers in Handwriting Recognition (IWFHR), August 2002, pp. 128–135,
36. Märgner, V., El Abed, H.: ICDAR 2007—Arabic handwriting recognition competition. In: Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR), vol. 2, pp. 1274–1278 (2007)
37. Märgner, V., El Abed, H.: ICDAR 2009—Arabic handwriting recognition competition. In: Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR), July 2009, vol. 3, pp. 1383–1387,

38. Märgner, V., El Abed, H.: ICFHR 2010—Arabic handwriting recognition competition. In: Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR), November 2010, pp. 709–714,
39. Märgner, V., El Abed, H.: ICDAR 2011—Arabic handwriting recognition competition. In: Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR) (2011)
40. Märgner, V., Pechwitz, M., El Abed, H.: ICDAR 2005 Arabic handwriting recognition competition. In: 8th International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 70–74 (2005)
41. Menasri, F., Vincent, N., Augustin, E., Cheriet, M.: Shape-based alphabet for off-line Arabic handwriting recognition. In: Proc. International Conference on Document Analysis and Recognition (ICDAR), vol. 2, pp. 969–973 (2007)
42. Menasri, F., Vincent, N., Augustin, E., Cheriet, M.: Un système de reconnaissance de mots Arabes manuscrits hors-ligne sans signes diacritiques. In: Proc. Colloque International Francophone sur l’Ecrit et le Document (CIFED), pp. 121–126 (2008)
43. Mokbel, C., Abi Akl, H., Greige, H.: Automatic speech recognition of Arabic digits over Telephone network. In: International Conference Research Trends in Science and Technology (RTST) (2002)
44. Pal, U., Roy, K., Kimura, F.: A lexicon driven method for unconstrained Bangla handwritten word recognition. In: 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR), pp. 601–606 (2006)
45. Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT—database of handwritten Arabic words. In: Colloque International Francophone sur l’Ecrit et le Document (CIFED), pp. 127–136 (2002)
46. Schambach, M.-P.: Model length adaptation of an HMM based cursive word recognition system. In: 7th International Conference on Document Analysis and Recognition (ICDAR), 3–6 August, 2003, vol. 1, pp. 109–113 (2003)
47. Shawe-Taylor, J., Cristianini, N.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge (2000)
48. Shi, M., Fujisawa, Y., Wakabayashi, T., Kimura, F.: Handwritten numeral recognition using gradient and curvature of gray scale image. *Pattern Recognit.* **35**(10), 2051–2059 (2002)
49. Touj, S., Ben Amara, N., Amiri, H.: Arabic handwritten words recognition based on a planar hidden Markov model. *Int. Arab J. Inf. Technol.* **2**(4), 318–325 (2005)
50. Valente, F., Vepa, J., Plahl, C., Gollan, C., Hermansky, H., Schlüter, R.: Hierarchical neural networks feature extraction for LVCSR system. In: Proceedings of the Interspeech, pp. 42–45 (2007)

Chapter 18

Benchmarking Strategy for Arabic Screen-Rendered Word Recognition

Fouad Slimane, Slim Kanoun, Jean Hennebert, Rolf Ingold,
and Adel M. Alimi

Abstract This chapter presents a new benchmarking strategy for Arabic screen-based word recognition. Firstly, we report on the creation of the new APTI (Arabic Printed Text Image) database. This database is a large-scale benchmarking of open-vocabulary, multi-font, multi-size and multi-style word recognition systems in Arabic. Such systems take as input a text image and compute as output a character string corresponding to the text included in the image. The challenges that are addressed by the database are in the variability of the sizes, fonts and styles used to generate the images. A focus is also given on low resolution images where anti-aliasing is generating noise on the characters being recognized. The database contains 45,313,600 single word images totalling more than 250 million characters. Ground truth annotation is provided for each image from an XML file. The annotation includes the number of characters, the number of pieces of Arabic words (PAWs), the sequence of characters, the size, the style, the font used to generate each image, etc. Secondly, we describe the Arabic Recognition Competition: Multi-Font Multi-Size Digitally Represented Text held in the context of the 11th International Conference on Document Analysis and Recognition (ICDAR'2011), during September 18–21, 2011, Beijing, China. This first

F. Slimane (✉) · J. Hennebert · R. Ingold
DIVA Group, Department of Informatics, University of Fribourg, Bd. de Perolles 90,
1700 Fribourg, Switzerland
e-mail: fouad.slimane@unifr.ch

J. Hennebert
e-mail: jean.hennebert@unifr.ch

R. Ingold
e-mail: rolf.ingold@unifr.ch

S. Kanoun
National School of Engineers (ENIS), University of Sfax, BP 1173, Sfax 3038, Tunisia
e-mail: slim.kanoun@ieee.org

A.M. Alimi
REGIM Group, National School of Engineers (ENIS), University of Sfax, BP 1173, Sfax 3038,
Tunisia
e-mail: Adel.Alimi@ieee.org

edition of the competition used the freely available APTI database. Two groups with three systems participated in the competition. The systems were compared using the recognition rates at the character and word levels. The systems were tested on one test dataset which is unknown to all participants (set 6 of APTI database). The systems were compared on the ground of the most important characteristic of classification systems: the recognition rate. A short description of the participating groups, their systems, the experimental setup and the observed results are presented. Thirdly, we present our DIVA-REGIM system (out of competition at ICDAR'2011) with all results of the Arabic recognition competition protocols.

18.1 Introduction

It is universally acknowledged that more than 500 million people around the world speak and use Arabic as their liturgical language. Arabic is important in the culture of many people. In the last twenty years, most of the efforts in Arabic text recognition have been toward the recognition of scanned printed documents [4, 17, 25]. Most of these works have been evaluated on private databases; therefore, the comparison of systems is rather difficult. To our knowledge, there are currently few large-scale image databases of Arabic printed text available for the scientific community. One of the only references we have found is on the ERIM [24] database containing 750 scanned pages collected from Arabic books and magazines. However, it seems difficult to have access to this database. In the Arabic handwriting recognition field, public databases exist such as the freely available IFN/ENIT-database [22]. Open competitions are even regularly organized using this database [19–21].

On the other hand, a corpus is a large structured set of text, electronically stored and processed. A text corpus or lexical database in Arabic is available from different associations or institutes [1, 2, 12]. However, such text corpora are not directly usable for the benchmarking of recognition systems that take images as input. Access to a corpus of both language and images is essential during optical character recognition (OCR) development, particularly while training and testing a recognition application [3]. Excellent corpora have been developed for Latin-based languages, but only a few relate to the Arabic language. This limits the penetration of both corpus linguistics and OCR in Arabic-speaking countries. In [3], the authors describe the construction and provide a comprehensive study and analysis of a multi-modal Arabic corpus (MMAC) that is suitable for use in both OCR development and linguistics. MMAC contains six million Arabic words and includes connected segments as well as naked pieces of Arabic words (NPAWs) and naked words (NWords); a ground truth annotation is offered for each image. MMAC is publicly and freely available.

To bring into account the above information, we initiated the development of a large database of images of printed Arabic words in 2009. This database is used

for our own research and is made available for the scientific community to evaluate their recognition systems. The database has been named Arabic Printed Text Image (APTI).

The purpose of the APTI database is a large-scale benchmarking of open-vocabulary, multi-font, multi-size and multi-style text recognition systems in Arabic. The images in the database are synthetically generated from a large corpus using automated procedures. The challenges that are addressed by the database are in the variability of the sizes, fonts and styles used to generate the images. Special attention is also paid to low resolution images where anti-aliasing is generating noise on the characters to recognize. Naturally, APTI is well suited for the evaluation of screen-based OCR systems that take as input images extracted from screen captures or pdf documents. Performances of classical scanned-based OCR or camera-based OCR systems could also be measured using APTI. However, such evaluations should take into account the absence of typical artefacts present in scanned or camera documents.

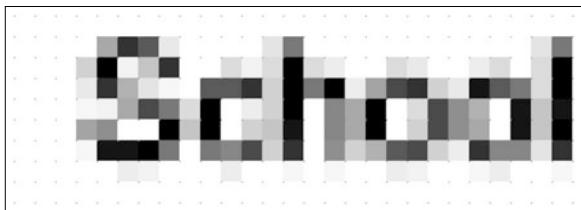
Being synthetically generated, the challenges of the database remain multiple:

- Large-scale evaluation with a realistic sampling of most of the Arabic character shapes and their accompanying variations due to ligatures and overlaps.
- Availability of multiple fonts, styles and sizes that must nowadays be treated by recognition systems.
- Emphasis on the low resolution images that are nowadays frequently present on computer screens.
- Isolated word images where inter-word language models cannot be used.
- Semi-blind evaluation protocols with decoupled development/evaluation sets.

Research work on Arabic optical text recognition has increased considerably since the 1980s. Scanner-based OCR has made considerable advances over the two past decades, thanks to the combined progress of the acquisition devices, recognition algorithms and computer capacities. OCR is nowadays practically considered as a solved problem in the case of Latin-based character inputs acquired in high resolution from flat bed scanners. First Arabic OCR systems were made available in the market in the 1990s. Currently, a few commercial systems are available, but the only independent system of comparison was made 15 years ago. Compared to the high quality and widespread usage of OCR systems for Latin characters, Arabic OCR still has to be developed, especially for the case of low resolution printed words.

More challenging tasks are now appearing where the conditions are more adverse, showing a significant drop of the performance in comparison with more classical applications. This is the case for screen-based OCR where inputs are typically at a lower resolution, showing multiple fonts and sizes and including potentially single words with very short sequences of characters. Recognition of low resolution text is quite interesting due to the wide range of applications and occurrence of low resolution text in screen shots, images and videos.

Fig. 18.1 An image of word ‘School’ at ultra low resolution and anti-aliased



This work is in the field of screen-rendered text OCR applied to the Arabic language. Recognition of screen-rendered text can be used to:

- Recognize low resolution text in videos.
- Provide meanings or translation of text from screen-shots of documents.
- Correct web page errors due to bad background and foreground combination [26].
- Enable web indexing tools to capture semantic important information from web images.
- Develop tools which read screen text for blind or visually impaired people.

The remainder of this chapter is organized as follows. In Sect. 18.2, we illustrate some of the major challenges in the recognition of low resolution text images. The first free APTI database on low resolution will be presented in Sect. 18.3. The first edition of the ICDAR’2011 Arabic recognition competition will be described in Sect. 18.4. In Sect. 18.5, we present the DIVA-REGIM system with results of the competition protocols followed immediately by some conclusions.

18.2 OCR Challenges for Low Resolution Text Images

Recognizing a low resolution text by OCR is a challenging task and involves several difficulties. Screen-rendered text can be on ultra low resolution (see Fig. 18.1) and is generally smoothed to make it look better to the human eye. Smoothed characters are difficult to segment because they are too close to other characters. This, for example, makes contour- and projection-based segmentation inapplicable. Also, the same character of the same logical description (font, size, etc.) is often rendered differently within the same document depending on its position. Furthermore, screen text can be displayed at any screen position. This means that the text can occur at an inhomogeneous background or that single words can be rendered isolated. Baseline detection for isolated words is difficult since commonly used horizontal projections of single words are not sufficient. Generally, the appearance of screen-rendered text depends on the font type, magnification, size, background, position, operating system or application, context and used smoothing algorithm [31].

18.3 APTI Database

Available since July 2009, APTI has been freely distributed to the scientific community for benchmarking purposes.¹ At the time of this writing, more than 20 research groups from universities, research centres, and industries worldwide are working with the APTI database. The APTI database is synthetic, and images are generated using automated procedures. In this section, we present the specificities of this database.

18.3.1 Corpus

APTI is created using a mix of non-decomposable and decomposable Arabic words [5, 7, 9, 13, 16]. Non-decomposable words are formed by country/town/village names, Arabic proper names, general names, Arabic prepositions, etc., whereas decomposable words are generated from root Arabic verbs using Arabic schemes [15]. To generate the lexicon, different Arabic books such as Al-bukhala of Gahiz² and *The Muqaddimah—An introduction to the history of Ibn Khaldun*³ were parsed. A collection of Arabic newspaper articles were also taken from the Internet as well as a large lexicon file produced by [15]. This parsing procedure totalled 113,284 single different Arabic words, leading to a pretty good coverage of the Arabic words from different disciplines, e.g. literature, culture, art, medicine, and law.

18.3.2 Fonts, Styles and Sizes

Taking as input the Arabic words, the APTI images are generated using 10 different sizes (6, 7, 8, 9, 10, 12, 14, 16, 18 and 24 points) and 10 different fonts as presented in Fig. 18.2. These fonts have been selected to cover different complexities of Arabic printed character shapes, going from simple fonts with no or few overlaps and ligatures like Andalus to more complex fonts rich in overlaps, ligatures and flourishes like Diwani Letter. All word images are generated also using four different styles: plain, italic, bold and a combination of italic and bold. These sizes, fonts and styles are widely used on computer screens, Arabic newspapers and many other

¹<http://diuf.unifr.ch/diva/APTI/>

²Al-Jahiz (born in Basra, c. 781–December 868 or January 869) was a famous Arab scholar, believed to have been an Afro-Arab of East African descent (<http://en.wikipedia.org/wiki/Al-Jahiz>).

³Ibn Khaldoun (May 27, 1332–March 19, 1406) was a famous historian, scholar, theologian, and statesman born in North Africa in present-day Tunisia (http://en.wikipedia.org/wiki/Ibn_Khaldoun).

- A : نقدم في هذا البحث قاعدة بيانات لكلمات عربية
- B : نقدم في هذا البحث قاعدة بيانات لكلمات عربية
- C : نقدم في هذا البحث قاعدة بيانات لكلمات عربية**
- D : نقترح في هذا البحث قاعدة بيانات لكلمات عربية
- E : نقترح في هذا البحث قاعدة بيانات لكلمات عربية
- F : نقدم في هذا البحث قاعدة بيانات لكلمات عربية
- G : نقدم في هذا البحث قاعدة بيانات لكلمات عربية
- H : نقدم في هذا البحث قاعدة بيانات لكلمات عربية
- I : نقدم في هذا البحث قاعدة بيانات لكلمات عربية
- J : نقترح في هذا البحث قاعدة بيانات لكلمات عربية

Fig. 18.2 Fonts used to generate the APTI database: (A) Andalus, (B) Arabic Transparent, (C) AdvertisingBold, (D) Diwani Letter, (E) DecoType Thuluth, (F) Simplified Arabic, (G) Tahoma, (H) Traditional Arabic, (I) DecoType Naskh, (J) M Unicode Sara

documents. The combination of fonts, styles and sizes guarantees a wide variability in APTI images.

18.3.3 Procedure for Creating Images

The word images are generated using a developed program. As a consequence, the artefacts or noise usually present on scanned or camera-based documents are not present in the images. Such degradations could actually be artificially added, if needed [6], but it is currently out of the scope of APTI. The text image generation, for example on a screen, can be done in many different ways. They all usually lead to slight variations of the target image. We have opted for a rendering procedure that allows us to include effects of down-sampling and anti-aliasing. These effects are interesting in terms of the variability of the images, especially at low resolution. The procedure involves the down-sampling of a high resolution source image into a low resolution image using anti-aliasing filtering. We also use different grid alignments to introduce variability in the application of the anti-aliasing filter. The details of the procedure are as follows:

1. A grey-scale source image is generated in high resolution (360 pixels/inch) from the current word in the lexicon, using the selected font, size and style.
2. Columns and rows of white pixels are added to the right-hand side and to the top of the image. The number of columns and rows is chosen to have a height and width multiple of the down-sampling factor [28]. This effect allows us to

```

<?xml version="1.0" encoding="UTF-8" ?>
- <wordImage id="78">
  - <content transcription="الله">
    <paw id="1" nbChars="1">Alif_I</paw>
    <paw id="2" nbChars="2">Laam_B TildAboveAlif_E</paw>
    <paw id="3" nbChars="2">Laam_B Alif_E</paw>
    <paw id="4" nbChars="1">Faa_I</paw>
  </content>
  <font name="Arabic Transparent" fontStyle="Plain" size="24" />
  <specs encoding="png" width="96" height="36" effect="none" />
  <generation type="downsampling5" renderer="java" filtering="antialiasing" />
</wordImage>

```

Fig. 18.3 Example of XML file including ground truth information about a given word image

have the same deformation in all images and artificially move the down-sampling grid.

3. Down-sampling and anti-aliasing filtering are applied to obtain the target image in low resolution (72 pixels/inch). The target image is in grey level.

18.3.4 Sources of Variability

APTI presents many sources of variability related to the generation procedure of images. The following list describes some of them:

1. 10 different fonts; 10 different sizes and 4 different styles.
2. Very large vocabulary that allows to test systems on unseen data.
3. Various artefacts of the down-sampling and anti-aliasing filters due to the insertion of columns of white pixels at the beginning and the top of word images.
4. Various forms of ligatures and overlaps of characters due to the large combination of characters in the lexicon and due to the used fonts.
5. Variability of the height of each word image. The height of each word image is related to the sequence of characters appearing in the word.

18.3.5 Ground Truth

In document image analysis and pattern recognition, ground truth refers to the various attributes associated with the text on the image such as the size of tokens, characters, used font, size, etc. Ground truth data is crucial for the training and testing of document image analysis applications [18]. However, each token word image in the APTI database contains ground truth information. Figure 18.3 shows the ground truth XML file containing information about the sequence of characters as well as the generation procedure.

The XML file includes the four following attributes:

Table 18.1 Quantity of words, PAWs and characters in APTI

	Words	PAWs	Characters
	113,284	274,833	648,280
* 10 Fonts * 10 Font Sizes * 4 Font Styles			
Total	45,313,600	109,933,200	259,312,000

1. *Content*: this contains the transcription of Arabic words, the number of pieces of Arabic words (nPaws) and sub-elements for each PAW with the sequence of characters. In our representation, characters are identified using plain English labels as described below.
2. *Font*: this contains the font name, font style and size used to generate the word image.
3. *Specs*: this presents the encoding of image, width, height and eventual additional effect.
4. *Generation*: this indicates the type of generation, the tool used for generation and the used filters in the generation procedure. In the current version of APTI, this element is constant as the same generation procedure has been applied. The type ‘downsampling5’ is used, indicating that the generation procedure corresponds to a down-sampling, using factor 5, from high resolution source images.

The different character labels can be observed in Table 18.5 showing their statistics through the sets of APTI. As the Arabic character shapes vary according to their position in a word, the character labels also include a suffix to specify the position of the character in the word: B standing for beginning, M for middle, E for end and I for isolated. The character ‘Hamza’ is always isolated, so we don’t use the suffix position for this character. We also artificially inserted characters labels such as ‘NuunChadda ن’ or ‘YaaChadda ي’ to represent the character shape issued from the combination of ‘Nuun ن’ and ‘Chadda’ or ‘Yaa ي’ and ‘Chadda’.

18.3.6 Database Statistics

The APTI database includes 113,284 different single words. Table 18.1 shows the total quantity of word images, PAWs and characters in APTI.

APTI is divided into six equilibrated sets to allow for flexibility in the composition of development and evaluation partitions. Five sets are available for the scientific community, and the sixth one is kept internal for potential evaluation of systems in blind mode (the set 6 was used at the ICDAR’2011 Arabic Recognition Competition presented in Sect. 18.4). The words in each set are different, but the distribution of all used letters is nearly the same in the various sets (see Table 18.5).

Table 18.2 Distribution of letters in set 1 (left) and set 2 (right)

Letter label	Nb. Occ	Isolate	Begin	Middle	End	Letter label	Nb. Occ	Isolate	Begin	Middle	End
Alif	15078	١ 5823		٤ 9255		Alif	14925	١ 5777		٤ 9148	
Baa	4513	ب 128	ب 1978	ب 2226	ب 181	Baa	4763	ب 150	ب 2039	ب 2344	ب 230
Taaa	9926	ت 587	ت 3626	ت 5332	ت 381	Taaa	9884	ت 642	ت 3551	ت 5347	ت 344
Thaa	634	ٿ 12	ٿ 261	ٿ 341	ٿ 20	Thaa	633	ٿ 19	ٿ 230	ٿ 349	ٿ 35
Jiim	1893	ج 60	ج 781	ج 1016	ج 36	Jiim	1897	ج 54	ج 756	ج 1034	ج 53
Haaa	2953	ح 69	ح 1135	ح 1648	ح 101	Haaa	2963	ح 93	ح 1159	ح 1619	ح 92
Xaa	1407	خ 16	خ 587	خ 782	خ 22	Xaa	1435	خ 18	خ 622	خ 777	خ 18
Daal	3187	د 988		د 2199		Daal	3033	د 963		د 2070	
Thaal	514	ڌ 167		ڌ 347		Thaal	520	ڌ 166		ڌ 354	
Raa	6304	ر 1813		ر 4491		Raa	6243	ر 1823		ر 4420	
Zaay	1064	ڙ 389		ڙ 675		Zaay	1054	ڙ 379		ڙ 675	
Siiin	3674	س 68	س 1434	س 2083	س 89	Siiin	3556	س 77	س 1338	س 2041	س 100
Shiin	1457	ش 18	ش 580	ش 831	ش 28	Shiin	1446	ش 22	ش 558	ش 838	ش 28
Saad	1374	ص 14	ص 439	ص 882	ص 39	Saad	1377	ص 22	ص 420	ص 906	ص 29
Daad	922	ض 41	ض 358	ض 497	ض 26	Daad	943	ض 42	ض 374	ض 492	ض 35
Thaaa	1419	ٿ 42	ٿ 392	ٿ 920	ٿ 65	Thaaa	1426	ٿ 38	ٿ 401	ٿ 925	ٿ 62
Taa	242	ٿ 6	ٿ 58	ٿ 163	ٿ 15	Taa	238	ٿ 7	ٿ 66	ٿ 149	ٿ 16
Ayn	2764	ع 67	ع 1003	ع 1575	ع 119	Ayn	2823	ع 85	ع 1074	ع 1543	ع 121
Ghayn	981	غ 12	غ 413	غ 543	غ 13	Ghayn	970	غ 15	غ 444	غ 495	غ 16
Faa	2305	ڦ 87	ڦ 1213	ڦ 923	ڦ 82	Faa	2256	ڦ 62	ڦ 1184	ڦ 937	ڦ 73
Gaaf	2784	ڧ 97	ڧ 937	ڧ 1614	ڧ 136	Gaaf	2734	ڧ 104	ڧ 872	ڧ 1632	ڧ 126
Kaaf	2101	ڪ 69	ڪ 914	ڪ 985	ڪ 130	Kaaf	2090	ڪ 63	ڪ 891	ڪ 1002	ڪ 134
Laam	6745	ڸ 175	ڸ 3546	ڸ 2206	ڸ 818	Laam	6926	ڸ 193	ڸ 3513	ڸ 2334	ڸ 886
Miim	7871	ڸ 177	ڸ 4043	ڸ 2844	ڸ 807	Miim	7836	ڸ 162	ڸ 4152	ڸ 2704	ڸ 818
Nuun	7484	ڽ 2437	ڽ 1264	ڽ 1905	ڽ 1878	Nuun	7433	ڽ 2391	ڽ 1262	ڽ 1848	ڽ 1932
NuunChadda	225	ڽ 0	ڽ 0	ڽ 225	ڽ 0	NuunChadda	224	ڽ 0	ڽ 0	ڽ 224	ڽ 0
Haa	2670	ه 223	ه 704	ه 1196	ه 548	Haa	2687	ه 224	ه 705	ه 1201	ه 559
Waaw	4421	و 1621		و 2800		Waaw	4313	و 1480		و 2833	
Yaa	6641	ي 317	ي 2516	ي 2640	ي 1168	Yaa	6630	ي 317	ي 2432	ي 2701	ي 1183
YaaChadda	725	ي 0	ي 192	ي 533	ي 0	YaaChadda	727	ي 0	ي 210	ي 517	ي 0
Hamza	192	ء 192		ء 192		Hamza	187		ء 187		
HamzaAboveAlif	1437	ء 1102		ء 335		HamzaAboveAlif	1483	ء 1156		ء 327	
TaaaClosed	1417	ت 441			ت 976	TaaaClosed	1407	ت 429			ت 978
HamzaUnderAlif	253	ء 182		ء 71		HamzaUnderAlif	250	ء 160		ء 90	
AlifBroken	162	ى 53			ى 109	AlifBroken	161	ى 47			ى 114
TildAboveAlif	84	ـ 32	ـ 132		ـ 52	TildAboveAlif	84	ـ 40			ـ 44
HamzaAboveAlifBroken	210	ى 3	ى 167	ى 39	ى 1	HamzaAboveAlifBroken	208	ى 0	ى 166	ى 34	ى 8
HamzaAboveWaaw	89	ڳ 30		ڳ 59		HamzaAboveWaaw	90	ڳ 32		ڳ 58	

18.3.7 Division into Sets

The algorithm for the distribution of words in the different sets has been designed to have similar allocations of letters and words in all sets. This procedure is simply stressing a fair distribution of words that include characters with few occurrences. This type of distribution is important to avoid under-representation of a given character in a given set and therefore to avoid potential problems while training or testing time. Tables 18.2, 18.3, 18.4 present the distribution of each shape of Arabic characters in their respective six sets.

Table 18.3 Distribution of letters in sets 3 and 4 respectively

Letter label	Nb. Occ	Isolate	Begin	Middle	End	Letter label	Nb. Occ	Isolate	Begin	Middle	End
Alif	15165	15988		19177		Alif	15120	15866		19254	
Baa	4692	456	1955	2343	238	Baa	4704	432	1979	2362	231
Taaa	9897	5617	3546	5380	354	Taaa	9797	5633	3625	5208	331
Thaa	631	516	245	335	35	Thaa	634	529	219	360	26
Jiim	1887	553	784	998	52	Jiim	1924	561	808	1016	39
Haaa	3017	563	1194	1659	101	Haaa	2933	568	1205	1552	108
Xaa	1439	511	643	765	20	Xaa	1401	516	615	749	21
Daal	3075	5947		2128		Daal	2990	5909		2081	
Thaal	528	5185		343		Thaal	504	5144		360	
Raa	6169	51746		4423		Raa	6335	51833		4502	
Zaay	1054	5362		692		Zaay	1066	5400		666	
Siiin	3674	575	555	555	103	Siiin	3512	563	555	555	94
Shiiin	1418	518	545	527	28	Shiiin	1434	517	5596	5796	25
Saad	1388	517	5390	5948	33	Saad	1411	519	5422	5937	33
Daad	936	550	5346	5511	29	Daad	906	534	5381	5457	34
Thaaa	1431	539	5393	5937	62	Thaaa	1426	534	5399	5929	64
Taa	240	51	546	5176	17	Taa	238	50	564	5159	15
Ayn	2769	564	5105	51560	130	Ayn	2718	572	51016	51518	112
Ghayn	983	512	5423	5530	18	Ghayn	984	512	5399	5566	7
Faa	2221	554	51178	5910	79	Faa	2313	573	51264	5894	82
Gaaf	2853	5107	5984	51640	122	Gaaf	2883	5106	5999	51639	139
Kaaf	2099	576	5904	5996	123	Kaaf	2145	586	5935	5978	146
Laam	6972	5183	53606	52259	924	Laam	7002	5207	53656	52247	892
Miim	7957	5190	54066	52899	802	Miim	7806	5157	53963	52848	838
Nuun	7289	52319	51293	51811	1866	Nuun	7316	52341	51239	51860	1876
NuunChadda	224	50	50	5224	0	NuunChadda	223	50	50	5223	0
Haa	2590	5192	5631	51222	546	Haa	2718	5201	5681	51252	585
Waaw	4325	51507		52818		Waaw	4333	59	51494		52839
Yaa	6876	5318	52527	52764	5270	Yaa	6685	5322	52443	52699	51221
YaaChadda	709	50	5198	5511	0	YaaChadda	719	50	5215	5504	50
Hamza	190		5190			Hamza	193		5193		
HamzaAboveAlif	1455	51113		5322		HamzaAboveAlif	1512	511164		5348	
TaaaClosed	1394	5435			5959	TaaaClosed	1364	5398			5966
HamzaUnderAlif	256	5169			587	HamzaUnderAlif	247	5171			576
AlifBroken	164	558			5106	AlifBroken	163	542			5121
TildAboveAlif	83		5139		544	TildAboveAlif	83		5138		545
HamzaAboveAlifBroken	208	54	5170	5127	57	HamzaAboveAlifBroken	209	55	5161	535	58
HamzaAboveWaaw	89	521		568		HamzaAboveWaaw	91	524		567	

18.3.8 APTI Evaluation Protocols

In this section, we propose the definition of a set of robust benchmarking protocols on top of the APTI database. Preliminary experiments with a baseline recognition system have helped in calibrating and validating these protocols. From the obtained results, we believe that the large amount of data available in APTI and the different sources of variability (cf. Sect. 18.3.4) make it well suited for significant and challenging system evaluation.

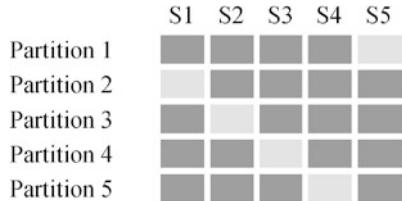
Table 18.4 Distribution of letters in sets 5 and 6 respectively

Letter label	Nb. Occ	Isolate	Begin	Middle	End	Letter label	Nb. Occ	Isolate	Begin	Middle	End
Alif	15046	15689		19357		Alif	15019	15797		19222	
Baa	4730	ب 161	ب 1991	ب 2341	ب 237	Baa	4717	ب 146	ب 1998	ب 2354	ب 219
Taaa	9942	ت 580	ت 3629	ت 5389	ت 344	Taaa	9897	ت 641	ت 3612	ت 5304	ت 340
Thaa	643	ث 26	ث 242	ث 347	ث 28	Thaa	628	ث 22	ث 227	ث 353	ث 26
Jiim	1915	ج 60	ج 809	ج 990	ج 56	Jiim	1939	ج 49	ج 803	ج 1048	ج 39
Haaa	3000	ح 83	ح 1134	ح 1680	ح 103	Haaa	3000	ح 83		ح 1655	ح 82
Xaa	1403	خ 15	خ 611	خ 754	خ 23	Xaa	1407	خ 7	خ 618	خ 751	خ 31
Daal	3028	د 901		د 2127		Daal	3086	د 939		د 2147	
Thaal	516	ذ 159		ذ 357		Thaal	518	ذ 164		ذ 354	
Raa	6253	ر 1824		ر 4429		Raa	6267	ر 1864		ر 4403	
Zaay	1042	ز 386		ز 656		Zaay	1045	ز 377		ز 668	
Siiin	3629	س 59	س 2091	س 78		Siiin	3603	س 73	س 1359	س 109	
Shiin	1455	ش 25	ش 566	ش 838	ش 26	Shiin	1458	ش 26	ش 582	ش 817	ش 33
Saad	1371	ص 14	ص 413	ص 896	ص 48	Saad	1389	ص 21	ص 415	ص 921	ص 32
Daad	921	ض 41	ض 369	ض 470	ض 41	Daad	920	ض 43	ض 335	ض 503	ض 39
Thaaa	1446	ط 33	ط 412	ط 934	ط 67	Thaaa	1462	ط 24	ط 428	ط 937	ط 73
Taa	239	ت 5	ت 52	ت 169	ت 13	Taa	241	ت 4	ت 65	ت 158	ت 14
Ayn	2755	ع 68	ع 1017	ع 1552	ع 118	Ayn	2723	ع 80	ع 1007	ع 1510	ع 126
Ghayn	990	غ 15	غ 422	غ 534	غ 19	Ghayn	1004	غ 15	غ 425	غ 540	غ 24
Faa	2339	ف 73	ف 1257	ف 920	ف 89	Faa	2315	ف 62	ف 1226	ف 928	ف 99
Gaaf	2762	ق 103	ق 959	ق 1574	ق 126	Gaaf	2803	ق 99	ق 974	ق 1584	ق 146
Kaaf	2136	ك 84	ك 914	ك 980	ك 158	Kaaf	2140	ك 85	ك 913	ك 1004	ك 138
Laam	6790	ل 188	ل 3433	ل 2288	ل 881	Laam	6724	ل 174	ل 3466	ل 2203	ل 881
Miim	7797	م 175	م 4067	م 2732	م 823	Miim	7817	م 166	م 4038	م 2779	م 834
Nuun	7400	ن 2435	ن 1273	ن 1825	ن 1867	Nuun	7264	ن 2411	ن 1231	ن 1835	ن 1787
NuunChadda	224	ن 0	ن 0	ن 224	ن 0	NuunChadda	223	ن 0	ن 0	ن 223	ن 0
Haa	2705	ه 178	ه 699	ه 1297	ه 531	Haa	2724	ه 230	ه 695	ه 1236	ه 565
Waaw	4264	و 1466		و 2798		Waaw	4352	و 1514		و 2838	
Yaa	6648	ي 327	ي 2507	ي 2656	ي 1160	Yaa	6735	ي 301	ي 2535	ي 2652	ي 1250
YaaChadda	735	ي 0	ي 168	ي 567	ي 0	YaaChadda	733	ي 199	ي 534	ي 2	
Hamza	192		ء 192			Hamza	188		ء 188		
HamzaAboveAlif	1456	ء 1158		ء 298		HamzaAboveAlif	1427	ء 1113		ء 314	
TaaaClosed	1409	ة 433			ة 976	TaaaClosed	1385	ة 430			ة 955
HamzaUnderAlif	248	ء 171			ء 77	HamzaUnderAlif	247	ء 179			ء 68
AlifBroken	161	ى 55			ى 106	AlifBroken	161	ى 43			ى 118
TildAboveAlif	83	ـ 146			ـ 37	TildAboveAlif	83	ـ 137			ـ 46
HamzaAboveAlifBroken	208	ى 2	ى 167	ى 32	ى 7	HamzaAboveAlifBroken	210	ى 6	ى 164	ى 39	ى 1
HamzaAboveWaaw	89	ء 28			ء 61	HamzaAboveWaaw	90	ء 23			ء 67

Error Estimation

The objective of any benchmarking of recognition systems is to estimate, as reliably as possible, the classification error rate \hat{P}_e . It is important to remember that, whatever the task and data used, \hat{P}_e is a function of the split of the data into training and test sets. Different splits will result in different error estimates. APTI is composed of quite large sets of data, which helps in reaching stable estimates of \hat{P}_e . Our objective is then to obtain a reliable estimate of \hat{P}_e while keeping the computation load tractable. Therefore, we have opted for a *rotation method*, as described in [14, Sect. 7]. The idea is to reach a trade-off between the *holdout method*, which leads to pessimistic and biased values of the error rate, and the *leave-one-out method*, which

Fig. 18.4 Illustration of the rotation method. For a given partition, the training sets are depicted in *dark grey* and the testing sets in *light grey*



gives a better estimate but at the cost of larger computational requirements. The rotation method we are proposing is illustrated in Fig. 18.4. The procedure is to perform independent runs on five different partitions between training and testing data.

The final error estimate is taken as the average of the error rates obtained on the different partitions:

$$\hat{P}_e = \frac{1}{5} \sum_{i=1}^5 \hat{P}_{e,i} \quad (18.1)$$

In the previous equation, $\hat{P}_{e,i}$ is the error rate obtained independently on a trained and tested system using the sets defined in partition i . The procedure actually corresponds to computing the average performance of five independent systems.

Training and Testing Conditions

Using the procedure described in Sect. 18.3.8, we can define different combinations of training and testing conditions. The objectives are to measure the impact of some of the variability of the data. We therefore propose 20 protocols as summarized in [28].

18.4 ICDAR'2011 Arabic Recognition Competition

This competition was organized by the Document, Image and Voice Analysis (DIVA) research group from the University of Fribourg, Switzerland in collaboration with the REsearch Group on Intelligent Machines (REGIM) group at Ecole Nationale d'Ingénieurs de Sfax (ENIS), from the University of Sfax, Tunisia and the group at the Institute of Communications Technology (IFN) of the Technical University of Braunschweig, Germany. The competition was organized in a ‘blind’ manner. The testing data for the evaluation is composed of an unpublished set (*set 6* of APTI) which is kept secret for evaluation purposes. The participants were asked to send an executable version of their recognizer to the organizers who, in turn, arrange to run the systems against an unseen set of data. We invited groups working on Arabic word recognition to adapt their system to the APTI database and send us executables of their systems. The scientific objectives of this first edition are to measure the impact of font size on the recognition performances. This is evaluated

Table 18.5 Distribution of characters in the different sets

Char label (Char)	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
Alif (ا)	15,078	14,925	15,165	15,120	15,046	15,019
Baa (ب)	4513	4763	4692	4704	4730	4717
Taaa (ت)	9926	9884	9897	9797	9942	9897
Thaa (ث)	634	633	631	634	643	628
Jiim (ج)	1893	1897	1887	1924	1915	1939
Haaa (ح)	2953	2963	3017	2933	3000	3000
Xaa (خ)	1407	1435	1439	1401	1403	1407
Daal (د)	3187	3033	3075	2990	3028	3086
Thaal (ذ)	514	520	528	504	516	518
Raa (ر)	6304	6243	6169	6335	6253	6267
Zaay (ز)	1064	1054	1054	1066	1042	1045
Siin (س)	3674	3556	3674	3512	3629	3603
Shiin (ش)	1457	1446	1418	1434	1455	1458
Saad (ص)	1374	1377	1388	1411	1371	1389
Daad (ض)	922	943	936	906	921	920
Thaaa (ط)	1419	1426	1431	1426	1446	1462
Taa (ظ)	242	238	240	238	239	241
Ayn (ع)	2764	2823	2769	2718	2755	2723
Ghayn (غ)	981	970	983	984	990	1004
Faa (ف)	2305	2256	2221	2313	2339	2315
Gaaf (ق)	2784	2734	2853	2883	2762	2803
Kaaf (ك)	2101	2090	2099	2145	2136	2140
Laam (ل)	6745	6926	6972	7002	6790	6724
Miim (م)	7871	7836	7957	7806	7797	7817
Nuun (ن)	7484	7433	7289	7316	7400	7264
Haa (ه)	2670	2687	2590	2718	2705	2724
Waaw (و)	4421	4313	4325	4333	4264	4352
Yaa (ي)	6641	6630	6876	6685	6648	6735
NuunChadda (ڻ)	225	224	224	223	224	223
YaaChadda (ڦ)	725	727	709	719	735	733
Hamza (ء)	192	187	190	193	192	188
HamzaAboveAlif (ڦ)	1437	1483	1455	1512	1456	1427
HamzaUnderAlif (ڦ)	253	250	256	247	248	247

Table 18.5 (Continued)

Char label (Char)	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
TildAboveAlif (ܲ)	84	84	83	83	83	83
TaaaClosed (ܽ)	1417	1407	1394	1364	1409	1385
AlifBroken (ܱ)	162	161	164	163	161	161
HamzaAboveAlifBroken (ܱܲ)	210	208	208	209	208	210
HamzaAboveWaaw (ܾܲ)	89	90	89	91	89	90
Quantity of characters	108,122	107,855	108,347	108,042	107,970	107,944
Quantity of PAWs	45,982	45,740	45,792	45,884	45,630	45,805
Quantity of words	18,897	18,892	18,886	18,875	18,868	18,866

in mono-font and multi-font contexts. The protocols are defined to evaluate the capacity of the recognition systems to handle different sizes and fonts using digitally low resolution images in order find a robust approach to screen-based OCR.

The evaluation was reported as word and character recognition rates. In this first edition of the competition, we have proposed two protocols, as described below.

18.4.1 Mono-font Competition Protocol—First APTI Protocol for Competition: APTIPC1

In this protocol, we test Arabic mono-font and multi-size systems trained on the Arabic Transparent font and sizes from 6 to 24.

- *Tested Fonts:* Arabic Transparent.
- *Tested Style:* Plain.
- *Tested Sizes:* 6, 8, 10, 12, 18, 24.
- *Set 6 word images:* 18,866 for each size/font.
- *Number of tests in APTIPC1:* 6.

18.4.2 Multi-font Competition Protocol—Second APTI Protocol for Competition: APTIPC2

In this protocol, we test Arabic multi-font and multi-size systems trained on five fonts and sizes from 6 to 24.

- *Tested Fonts:* Diwani Letter, Andalus, Arabic Transparent, Simplified Arabic and Traditional Arabic.
- *Tested Style:* Plain.

- *Tested Sizes:* 6, 8, 10, 12, 18, 24.
- *Set 6 word images:* 18,866 for each size/font.
- *Number of tests in APTIPC2:* 30.

18.4.3 Participating Systems

The following section gives a short description of the systems submitted to the ICDAR'2011 Arabic Recognition Competition: Multi-Font Multi-Size Digitally Represented Text. The system descriptions vary in length according to the level of detail provided by the participants.

IPSAR System

The IPSAR system was submitted by Samir Ouis, Mohammad S. Khorsheed and Khalid Alfaifi, members of the Image Processing and Signal Analysis & Recognition (IPSAR) Group. This group is part of the Computer Research Institute (CRI) at King Abdulaziz City for Science & Technology (KACST) from the kingdom of Saudi Arabia.

IPSARec is a cursive Arabic script recognition system where ligatures, overlaps and style variation pose challenges to the recognition system. It is based on the Hidden Markov Model Toolkit (HTK), a portable toolkit for speech recognition systems which is customized here to recognize characters. IPSARec is an omnifont, unlimited vocabulary recognition system. It does not require segmentation. The proposed system proceeds with three main stages: extracting a set of features from the input images, clustering the feature set according to a pre-defined codebook and finally, recognizing the characters.

Each word/line image is transferred into a sequence of feature vectors. Those features are extracted from overlapping vertical windows, divided into cells where each cell includes a predefined number of pixels, along the word/line image, then clustered into discrete symbols.

Stage two is performed within HTK. It couples the feature vectors with the corresponding ground truth to estimate the character model parameters. The final output of this stage is a lexicon-free system to recognize cursive Arabic text. During recognition, an input pattern of discrete symbols representing the word/line image is injected to the global model which outputs a stream of characters matching the text line.

For more details about this system, we refer to [17].

UPV-BHMM Systems

These systems were submitted by Ihab Alkhoury, Adria Gimenez and Alfons Juan, from the Universitat Politècnica de Valencia (UPV), Spain. They are based

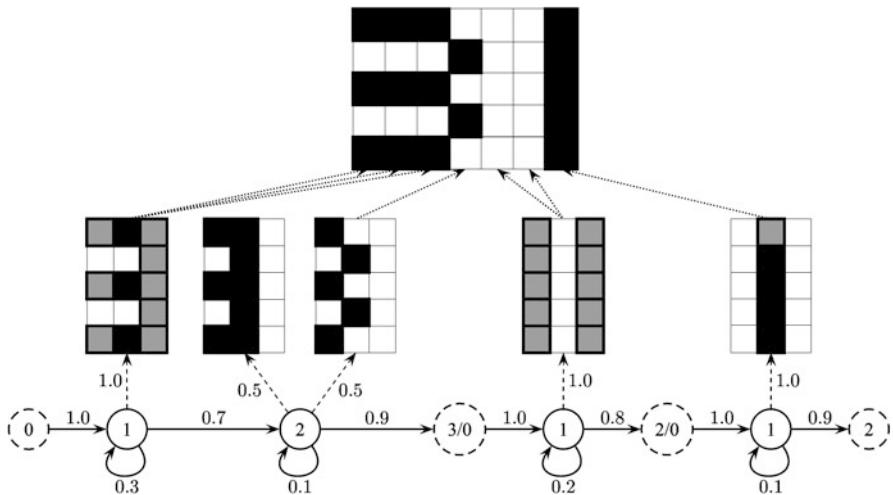


Fig. 18.5 Generation of a 7×5 word image of the number 31 from a sequence of 3 windowed ($W = 3$) BHMMs for the characters 3, 'space' and 1

on Bernoulli HMMs (BHMMs), that is, HMMs in which conventional Gaussian mixture density functions are replaced with Bernoulli mixture probability functions [10]. Also, in contrast to the basic approach followed in [10], in which narrow, one-column slices of binary pixels are fed into BHMMs, the UPV-BHMM systems are based on a sliding window of adequate width to better capture image context at each horizontal position of the word image. This new, windowed version of the basic approach is described in [11]. As an example, Fig. 18.5 shows the generation of a 7×5 word image of the number 31 from a sequence of 3 windowed ($W = 3$) BHMMs for the characters 3, 'space' and 1.

The UPV-PRHLT systems were trained from input images scaled in height to 40 pixels (while keeping the aspect ratio) after adding a certain number of white pixel rows to both top and bottom sides of each image, and then binarized with the Otsu algorithm. A sliding window of width 9 was applied, and thus the resulting input (binary) feature vectors for the BHMMs had 360 bits. The number of states per character was adjusted to 5 states for images with font size of 6, and 6 states for other font sizes. Similarly, the number of mixture components per state was empirically adjusted to 64. The estimation and recognition parameters were carried out using the expectation maximization (EM) algorithm.

Two systems were submitted: *UPV-PRHLT-REC1* and *UPV-PRHLT-REC2*. They are used for both tasks/protocols. In the first task (one style), there are no differences between systems; one model for each font size is trained and used later to recognize the test corpus. For the second task, in the first system, for each font size, a different model for each font style is trained. The test corpus is recognized on all models, and the recognized text word of the highest probability is selected. For the second task in the other system, a different character is considered for each style. A model for all styles together is trained and used to recognize the test corpus.

Table 18.6 APTIPC1—ICDAR’2011 competition results for participant systems

System		Size						Mean RR
		6	8	10	12	18	24	
IPSAR System	WRR	5.7	73.3	75.0	83.1	77.1	77.5	65.3
	CRR	59.4	94.2	95.1	96.9	95.7	96.8	89.7
UPV-PRHLT-REC1	WRR	94.5	97.4	96.7	92.5	84.6	84.4	91.7
	CRR	99.0	99.6	99.4	98.7	96.9	96.0	98.3
UPV-PRHLT-REC2	WRR	94.5	97.4	96.7	92.5	84.6	84.4	91.7
	CRR	99.0	99.6	99.4	98.7	96.9	96.0	98.3

18.4.4 Competition Results

All systems have been tested using the *set 6* (18,866 single word images) of the APTI database in different sizes and fonts. All participants sent us a running version of their recognition systems. The systems can be classified into two classes depending on the operating system: two systems are developed under Linux (UPV-PRHLT-REC1 and UPV-PRHLT-REC2) and one system under the Microsoft Windows environment.

Table 18.6 presents all system results of the first APTI protocol (APTIpc1). For each test the best result is marked in bold.

This first test is mono-font and mono-size. The test images presented to the systems are those using the font Arabic Transparent, plain and sizes 6, 8, 10, 12, 18 and 24. For most of the systems, we observed good results in character recognition and slightly worse results for word recognition. Both UPV-BHMM systems have the same behaviour and show the best results with an average of 91.7 % for the word recognition rate and 98.3 % for the character recognition rate. Compared to other competition systems, the IPSAR system has the best character recognition rate on size 24.

Tables 18.7, 18.8 and 18.9 present system results of the second APTI protocol (APTIpc2) for competition. This second test is multi-font and mono-size. The test images presented to the systems are those using the fonts (Arabic Transparent, Andalus, Simplified Arabic, Traditional Arabic and Diwani Letter), plain and sizes 6, 8, 10, 12, 18 and 24.

In APTIpc2, the recognition rate is not as good as in APTIpc1 for the Arabic Transparent font. The best system is UPV-PRHLT-REC1 with an average of 83.4 % for the word recognition rate and 96.4 % for the character recognition rate.

The UPV-PRHLT-REC1 system shares good results for most fonts and sizes in this APTIpc2. The IPSAR system gives good results for the Traditional Arabic and Diwani Letter fonts in font size 10, 12 and 24.

Table 18.7 APTIPC2—IPSAR system results

Font		Size						Mean RR
		6	8	10	12	18	24	
Andalus	WRR	13.9	35.7	65.6	73.8	69.5	64.5	53.8
	CRR	67.4	82.4	92.4	94.4	93.0	92.5	87.0
Arabic Transparent	WRR	29.9	40.0	73.2	74.9	65.9	69.1	58.8
	CRR	78.2	84.4	94.1	95.1	93.9	95.5	90.2
Simplified Arabic	WRR	30.8	39.8	73.2	75.5	66.2	68.6	59.0
	CRR	77.6	84.3	94.2	94.9	93.1	94.4	89.8
Traditional Arabic	WRR	4.6	3.4	46.7	55.1	52.9	50.4	35.5
	CRR	49.8	49.2	85.9	88.5	87.5	88.3	74.9
Diwani Letter	WRR	9.7	3.3	39.9	55.8	49.5	64.0	37.0
	CRR	60.1	48.3	83.4	89.1	91.7	92.6	77.5
Mean RR of the system							WRR	48.8
							CRR	83.9

Table 18.8 APTIPC2—UPV-PRHLT-REC1 system results

Font		Size						Mean RR
		6	8	10	12	18	24	
Andalus	WRR	94.1	75.5	81.1	83.6	83.9	85.0	83.8
	CRR	98.9	94.8	96.1	96.7	96.7	97.0	96.7
Arabic Transparent	WRR	94.7	78.2	78.9	81.8	83.1	83.8	83.4
	CRR	99.0	95.2	95.5	96.1	96.2	96.1	96.4
Simplified Arabic	WRR	95.8	82.4	84.2	85.3	85.6	88.0	86.9
	CRR	99.2	96.2	96.7	96.9	97.0	97.4	97.2
Traditional Arabic	WRR	57.6	38.3	43.6	43.5	42.9	46.2	45.4
	CRR	89.3	81.9	84.3	83.6	83.5	85.0	84.6
Diwani Letter	WRR	61.7	27.7	30.9	31.6	76.4	35.1	43.9
	CRR	90.9	75.8	77.8	78.1	94.9	79.6	82.8
Mean RR of the system							WRR	68.7
							CRR	91.5

18.5 DIVA-REGIM System

The DIVA-REGIM system is part of a joint collaboration between the DIVA (Document, Image and Voice Analysis) group from the University of Fribourg, Switzerland and the REGIM (REsearch Group on Intelligent Machines) group from the Uni-

Table 18.9 APTIPC2—UPV-PRHLT-REC2 system results

Font		Size						Mean RR
		6	8	10	12	18	24	
Andalus	WRR	83.1	73.6	79.5	77.7	71.1	71.7	76.1
	CRR	96.0	94.1	95.1	94.9	93.6	93.5	94.5
Arabic Transparent	WRR	86.1	84.3	84.1	81.1	75.5	75.6	81.1
	CRR	97.1	96.5	96.6	96.1	94.9	94.8	96.0
Simplified Arabic	WRR	87.6	82.6	83.5	81.2	74.2	76.2	80.9
	CRR	97.4	96.1	96.5	96.1	94.7	95.0	96.0
Traditional Arabic	WRR	43.7	36.9	42.3	40.9	37.6	40.2	40.2
	CRR	83.6	80.5	83.2	82.1	80.8	82.2	82.1
Diwani Letter	WRR	41.9	26.4	29.7	29.2	68.4	29.9	37.6
	CRR	83.2	74.5	76.8	76.5	93.4	76.7	80.2
Mean RR of the system							WRR	63.2
							CRR	89.7

versity of Sfax, Tunisia. This system is a cascading system working in three steps: feature extraction, font recognition and word recognition using font-dependent models.

18.5.1 Pre-processing

The pre-processing phase aims at the reduction of the variability between character shapes due to misalignment on the Y -axis. Classically, this pre-processing phase normalizes all inputs by shifting the images so that the characters of a word or sequence of words are aligned vertically according to a common baseline.

A data-driven baseline detection system is proposed in this work. The idea is to detect a probable baseline region using data-driven methods trained on local character features. Once a probable baseline region is recognized by the Gaussian mixture models (GMMs)-based system, the final position of the baseline is fine-tuned using the classical horizontal projection histogram, but limited to this region. The baseline recognition system is actually similar to the system presented in [30] for Arabic font recognition. Each word image is normalized in grey level into a rectangle with fixed height and then transformed into a sequence of feature vectors computed from a narrow analysis window, sliding from right to left on the word image. Again, the features used here for the baseline detection are actually the same as for the font recognition system and are presented in [30]. In our settings, the analysis window is shifted by 1 pixel for each feature vector. We performed several tests to determine the optimal size of the window and we converged to a 4 pixel width and 30 pixel height.

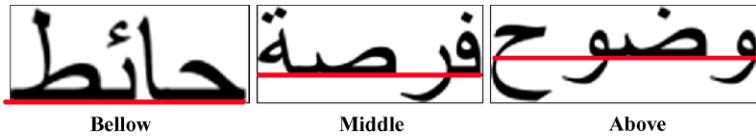


Fig. 18.6 Example of three baseline positions considering the bounding box of single word images

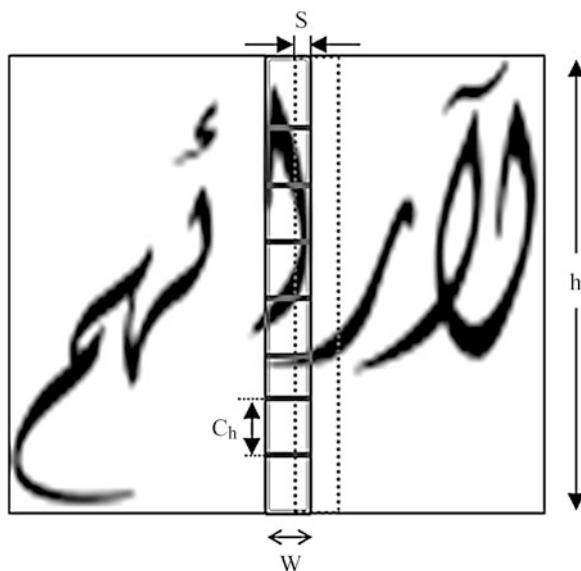
GMMs are used to estimate the likelihoods of three baseline positions (called here *below*, *middle* and *above* positions) as illustrated in Fig. 18.6. Each position is represented by a single GMM, which can actually be seen as a single-state HMM. Assuming the independence of the feature vectors, the GMMs are able to compute a global likelihood of a baseline position simply by multiplying the local likelihoods of each feature vectors computed separately. Each model is trained using an expectation maximization procedure by pooling a large quantity of feature vectors from words in known baseline positions [8]. Being state-less, the obtained models are currently independent of any character but become conditioned to the three baseline positions. Thanks to the large quantity of data, models can typically scale up to a large number of Gaussians (in our settings 8192 Gaussians). At testing time, the GMM showing the largest likelihood is selected, indicating a probable baseline region. Finally, the baseline position is fine-tuned by computing horizontal projection histograms limited to the recognized region.

18.5.2 Feature Extraction

The proposed feature extraction works on binary and grey level images. It depends on horizontal sliding window and vertical frames for a word with specific Arabic fonts (each horizontal window divided into vertical frames without overlap). The width of the horizontal sliding window could be w pixels, where w is an integer number that is determined empirically depending on the developed system for each Arabic font. The height of this window is equal to h pixels, where h represents a fixed integer number. The narrow analysis window slides horizontally from right to left on the word image with a shift of s pixels, where s is an integer window equal to 1. This allows us to take enough samples to be able to reliably estimate character models. For complex Arabic fonts, each horizontal frame is divided into cells where the cell height (C_h) is fixed. This yields a fixed number of cells in each frame according to the normalized word image height. Figure 18.7 illustrates the basic definitions used above.

In our case, the analysis window has a uniform size and moves one pixel from right to left. We conducted several tests to determine the optimal size of the sliding window according to the Arabic font used. As a result, no segmentation into letters is made, and the word image is transformed into a matrix of values where the number of lines corresponds to the number of analysis windows, and the number of columns is equal to the number of coefficients in each feature vector. The feature extraction is divided into two parts. The first part extracts, for each window:

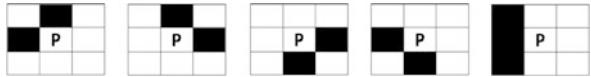
Fig. 18.7 Basic definition used in sliding window feature extraction



- Number N_1 of black connected components.
 - Number N_2 of white connected components.
 - Ratio N_1/N_2 .
 - Position of the smallest black connected component divided by the height of the window.
 - Sum of the perimeter P of all components in window/perimeter of window P_w .
 - Compactness $(4\pi A)p^2$ where P is the shape perimeter in window and A is the area.
 - Gravity centre of the window, of the right and left half and of the first third, the second and the last part of the window: $\sum_{i=1}^n \frac{x_i}{nW}$; $\sum_{i=1}^n \frac{y_i}{nH}$ where W is the width and H is the height of the window.
 - Position of baseline/height image.
 - Number of extremum in vertical projection.
 - Number of extremum in horizontal projection.
 - Size of the smallest connected component.
 - Density of black pixels in the window.
 - Density of black pixels below the low baseline.
 - Density of black pixels above the low baseline.
 - Densities of black pixels in each column of the window. As the width of the window is w pixels, it has w columns in each window.
- When $n(i)$ is the number of black pixels in the cell i , and $b(i)$ is the intensity of the cell i :

$$\begin{cases} b(i) = 0 & \text{if } n(i) = 0 \\ b(i) = 1 & \text{else} \end{cases}$$

Fig. 18.8 Five types of concavity configurations for a background pixel P



- Number of black/white transitions between cells: $f = \sum_{i=2}^{n_c} |b(i) - b(i - 1)|$ where n_c is the number of cells in the window.
- Number of black/white transitions between cells located above the low baseline.
- Position difference between the gravity centres g of writing pixels in two consecutive windows: $f = g(t) - g(t - 1)$.
- Area belonging to the text gravity centre in the window (up area $f = 1$, medium area $f = 2$, below area $f = 3$).
- Number of white pixels that belong to one of the five configurations shown in Fig. 18.8. The number of pixels in each configuration is then normalized by the number of pixels in the window.
- Number of background pixels in the five configurations mentioned above but only for pixels located in the middle area of writing, between the two baselines (see Fig. 18.9).
- Number of background pixels in the five configurations mentioned above but only for pixels located in the lower area of writing, below the lower baseline.
- Number of background pixels in the five configurations mentioned above but only for pixels located in the upper area of writing, over the upper baseline.
- The moment invariants (7 moments).
- The affine moment invariants (6 moments).
- The Zernike moments (12 moments).
- The Fourier descriptors (9 descriptors).
- The histogram of the Freeman directions (8 directions).
- The sum of the gradient norms.

The different recognition systems that depend on the font do not use the same feature. For all systems, however, each feature vector x_n has M components including $M/2$ basis features concatenated with $M/2$ delta coefficients computed as a linear difference of the basis features in adjacent windows. The deltas are computed in a similar way as in speech recognition, to include larger contextual information in an analysis window using the following formula:

$$\begin{cases} \Delta x_n^j = x_{n+1}^j - x_{n-1}^j, & \forall 1 < j < M/2 \\ \Delta x_n^j = x_n & \text{where } n = 0 \text{ or } n = N \end{cases}$$

18.5.3 Character Models Training

First, starting from all Arabic character shapes (more than 120), we grouped similar character shapes into 65 models according to the following rules: (1) beginning and middle shapes share the same model; (2) end and isolated shapes share the same

Fig. 18.9 Upper and lower baselines on sample data

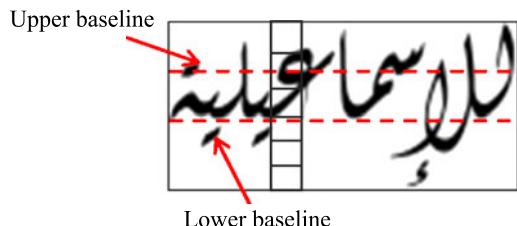
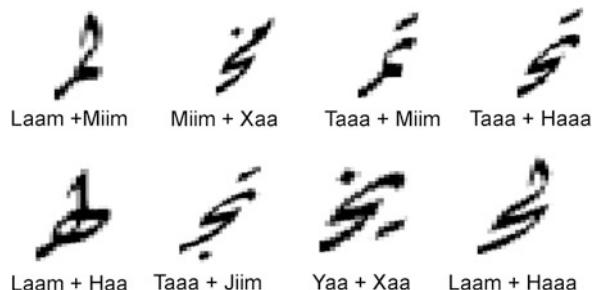


Fig. 18.10 Additional sub-model examples for *Diwani Letter* font



model. These rules apply for all characters with the exception of the characters *Ayn* ‘݂’ and *ghayn* ‘݄’ where the beginning, middle, end and isolated shapes are very different. This strategy of grouping is natural as beginning-middle and end-isolated character shapes are visually similar. The selection procedure of the different sub-models has been driven by grouping shapes of letters presenting few variations. The grouping strategy is explained in more detail in [27, 29]. Our hypothesis here is that the emission probability estimators based on Gaussian mixtures will offer enough flexibility to model the common parts and the variations within each letter category. Using the terminology introduced for speech recognition [23], our models are said to be context independent; i.e., each sub-model is considered independent from the next.

Second, for the used fonts presenting many ligatures between letters, we have added a new character sub-model: a selected set of their corresponding variations. Figure 18.10 presents some examples.

18.5.4 Ergodic Topology

In this topology every sub-model can be reached from every other sub-model. All transitions from one sub-model to another are allowed. Using ergodic topology offers the advantage of relatively lightweight memory and CPU footprint, when compared to more heavyweight approaches based on finite-state or stochastic grammars.

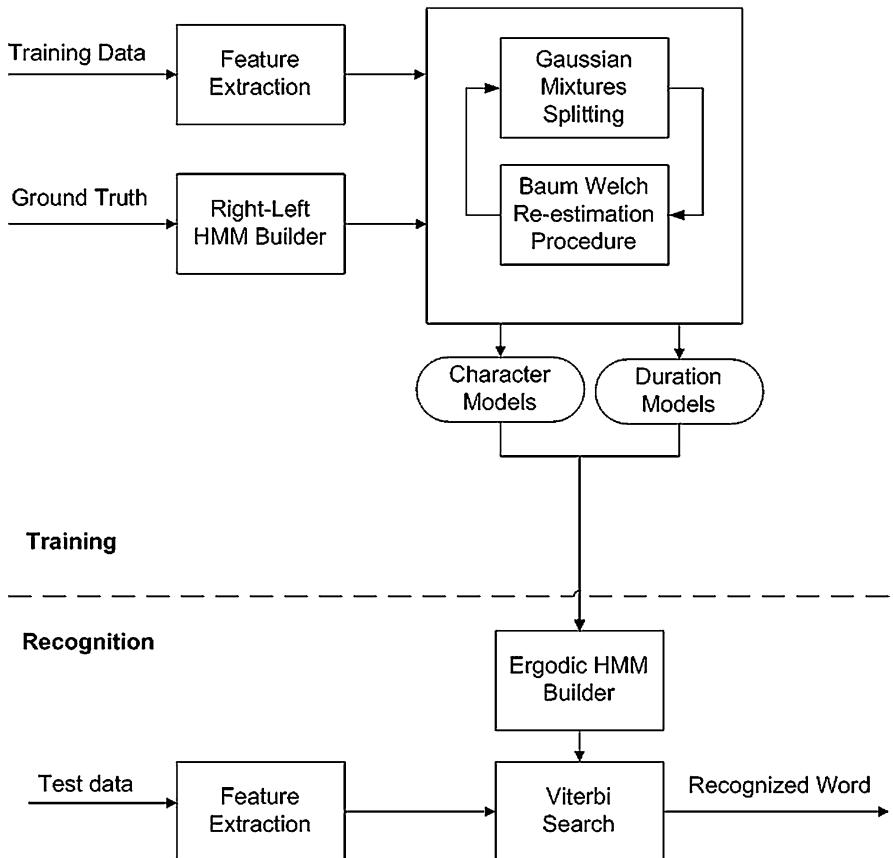


Fig. 18.11 HMM-based word recognition system

18.5.5 Training and Recognition

Our word recognition system is based on hidden Markov models (HMMs). The DIVA-REGIM system has a similar architecture to the one presented in [27]. One of its main characteristics is that it is open vocabulary, i.e. able to recognize any Arabic printed word on ultra low resolution. The training-testing system architecture is illustrated in Fig. 18.11. Note that the baseline detection system shares a similar training-testing architecture; the only difference is the fact that HMMs are here used instead of GMMs.

We used the Hidden Markov Model Toolkit (HTK) to realize our evaluation [32]. HTK was originally developed at the Speech Vision and Robotics Group of the Cambridge University Engineering Department (CUED). This toolbox has been built to experiment with HMMs and has been extensively used in speech recognition research. HTK is a set of command line executables used for initializing, modifying,

Table 18.10 APTIPC1—DIVA-REGIM system results

System		Size						Mean RR
		6	8	10	12	18	24	
DIVA-REGIM	WRR	97.47	98.67	99.02	99.32	99.44	99.76	98.95
	CRR	99.74	99.82	99.86	99.92	99.94	99.97	99.88

ing, training and testing HMMs. The use of HTK typically goes through four phases: preparation of data, training, recognition and recognition performance evaluation.

In the learning phase, all training files are first used for the initialization of HMM models for each letter, using HTK HCompV. For each training word image, the corresponding sub-models are connected to form a right-left HMM. An embedded training using the Baum–Welch iterative estimation procedure is used with the HTK tool HERest. Using a training set, all the observation sequences are used to estimate the emission probability functions of each sub-model. The training procedure actually involves two steps that are iteratively applied to increase the number of Gaussian mixtures to a given M value. In the first step, a binary split procedure, along the iteration process, is applied to the Gaussians to increase their number. In the second step, the Baum–Welch re-estimation procedure is launched to estimate the parameters of the Gaussians. However, the expectation maximization (EM) algorithm is used to iteratively refine the component weights, means and variances to monotonically increase the likelihood of the training feature vectors.

At recognition time, an ergodic HMM is formed using all sub-models. The recognition is done by selecting the best state sequence in the HMM using a Viterbi procedure implemented with the HTK tool HVite. Performances are evaluated in terms of word recognition rates using an unseen set of word images. The evaluation is obtained using the HTK tool HResult.

18.5.6 Experimental Results

Table 18.10 presents the DIVA-REGIM system results for the first APTI protocol (APTIpc1) for competition. The results are good for the majority of font sizes with an average of 98.95 % for word recognition rate and 99.88 % for character recognition rate. These results are better than those of the other participating systems in the ICDAR’2011 competition.

Table 18.11 presents the DIVA-REGIM system results for the second APTI protocol (APTIpc2) for competition. In term of results, DIVA-REGIM seems to be the best system compared to the other participating systems in the ICDAR’2011 competition with an average of 91.92 % and 97.72 % respectively for word recognition rate and character recognition rate.

Table 18.11 APTIPC2—DIVA-REGIM system results

Font		Size						Mean RR
		6	8	10	12	18	24	
Andalus	WRR	94.34	97.61	97.58	99.27	98.50	99.47	97.80
	CRR	97.94	99.26	99.45	99.70	99.49	99.82	99.28
Arabic Transparent	WRR	86.52	95.67	96.65	96.45	97.49	97.78	95.09
	CRR	93.87	98.51	99.13	99.10	99.40	99.25	98.21
Simplified Arabic	WRR	83.29	92.73	96.82	96.43	96.50	96.97	93.79
	CRR	92.16	97.37	98.99	98.82	99.13	98.71	97.53
Traditional Arabic	WRR	77.56	92.72	94.56	95.44	94.55	95.11	91.66
	CRR	96.03	98.87	98.92	98.94	99.00	98.79	98.43
Diwani Letter	WRR	57.47	80.50	84.18	89.88	90.08	85.46	81.26
	CRR	89.33	95.06	96.04	97.16	96.99	96.25	95.14
Mean RR of the system							WRR	91.92
							CRR	97.72

18.6 Conclusion

APTI is challenging, especially when we consider the recognition rate at the word level. APTI aims at a large-scale benchmarking of open-vocabulary text recognition systems. While it can be used for the evaluation of any OCR system, APTI is naturally well suited for the evaluation of screen-based OCR systems. The challenges addressed by the database are the variability of the sizes, fonts and styles, and the protocols that are defined are efficient enough to evidence the impact of such variability. The objective of the first competition, organized at the 11th International Conference on Document Analysis and Recognition (ICDAR’2011), in September 18–21, 2011, Beijing, China, for the recognition of multi-font and multi-size Arabic text, was to evaluate and compare different systems and approaches. We have presented in this chapter the results of four different systems on the ICDAR’2011 competition protocols with benchmarking strategy for Arabic low resolution word recognition.

Acknowledgements The authors would like to thank all ICDAR’2011—Arabic Recognition Competition: Multi-Font Multi-Size Digitally Represented Text participants and the anonymous reviewers for their comments and suggestions, which have much improved the presentation of this work.

References

1. Abbès, R., Dichy, J., Hassoun, M.: The architecture of a standard Arabic lexical database: some figures, ratios and categories from the DIINAR.1 source program. In: Proceedings of

- the Workshop on Computational Approaches to Arabic Script-Based Languages, Semitic'04, pp. 15–22. Association for Computational Linguistics, Stroudsburg (2004)
- 2. Abdelraouf, A., Higgins, C.A., Khalil, M.: A database for Arabic printed character recognition. In: Proceedings of the 5th International Conference on Image Analysis and Recognition, ICIAR'08, pp. 567–578. Springer, Berlin (2008)
 - 3. AbdelRaouf, A., Higgins, C., Pridmore, T., Khalil, M.: Building a multi-modal Arabic corpus (MMAC). *Int. J. Doc. Anal. Recognit.* **13**, 285–302 (2010)
 - 4. Al-Muhtaseb, H.A., Mahmoud, S.A., Qahwaji, R.S.: Recognition of off-line printed Arabic text using hidden Markov models. *Signal Process.* **88**, 2902–2912 (2008)
 - 5. Al-Sughaiyer, I.A., Al-Kharashi, I.A.: Arabic morphological analysis techniques: a comprehensive survey. *J. Am. Soc. Inf. Sci. Technol.* **55**, 189–213 (2004)
 - 6. Baird, H.: The state of the art of document image degradation modelling. In: Chaudhuri, B.B. (ed.) *Digital Document Processing, Advances in Pattern Recognition*, pp. 261–279. Springer, London (2007)
 - 7. Ben Hamadou, A.: A compression technique for Arabic dictionaries: the affix analysis. In: COLING'86, pp. 286–288 (1986)
 - 8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. B* **39**(1), 1–38 (1977)
 - 9. Dichi, J., Hassoun, M.: The DIINAR.1—Arabic lexical resource, an outline of contents and methodology. *ELRA Newslett.* **10**(2), 5–10 (2005)
 - 10. Gimenez, A., Juan, A.: Embedded Bernoulli mixture HMMs for handwritten word recognition. In: Proc. of the 10th Int. Conf. on Doc. Analysis and Recognition (ICDAR), pp. 896–900 (2009)
 - 11. Gimenez, A., Khouri, I., Juan, A.: Windowed Bernoulli mixture HMMs for Arabic handwritten word recognition. In: 2010 International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 533–538 (2010)
 - 12. Graff, D., Chen, K., Kong, J., Maeda, K.: Arabic Gigaword, 2nd edn. Linguistic Data Consortium, Philadelphia (2006)
 - 13. Hilal, Y.: Tahil sarfi lil arabia. In: Proc. Comput. Process. Arabic Language, Kuwait (1985)
 - 14. Jain, A.K., Duin, R.P.W., Mao, J.: Statistical pattern recognition: a review. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 4–37 (2000)
 - 15. Kanoun, S., Slimane, F., Guesmi, H., Ingold, R., Alimi, A.M., Hennebert, J.: Affixal approach versus analytical approach for off-line Arabic decomposable vocabulary recognition. In: ICDAR, pp. 661–665 (2009)
 - 16. Kanoun, S., Alimi, A.M., Lecourtier, Y.: Natural language morphology integration in off-line Arabic optical text recognition. *IEEE Trans. Syst. Man Cybern., Part B, Cybern.* **41**(2), 579–590 (2011)
 - 17. Khorsheed, M.S.: Offline recognition of omnifont Arabic text using the HMM toolkit (HTK). *Pattern Recognit. Lett.* **28**, 1563–1571 (2007)
 - 18. Lee, C.H., Kanungo, T.: The architecture of TRUEVIZ: a groundtruth/metadata editing and visualizing toolkit. *Pattern Recognit.* **36**(3), 811–825 (2003)
 - 19. Märgner, V., El Abed, H.: ICDAR 2009—Arabic handwriting recognition competition. In: ICDAR, pp. 1383–1387 (2009)
 - 20. Märgner, V., El Abed, H.: ICFHR 2010—Arabic handwriting recognition competition. In: ICFHR, pp. 709–714 (2010)
 - 21. Märgner, V., El Abed, H.: ICDAR 2011—Arabic handwriting recognition competition. In: 2011 International Conference on Document Analysis and Recognition (ICDAR), pp. 1444–1448 (2011)
 - 22. Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT—database of handwritten Arabic words. In: Proc. of CIFED 2002, pp. 129–136 (2002)
 - 23. Rabiner, L., Juang, B.-H.: *Fundamentals of Speech Recognition*. Prentice Hall, Upper Saddle River (1993)
 - 24. Schlosser, S.: ERIM Arabic database. Document Processing Research Program, Information and Materials Applications Laboratory, Environmental Research Institute of Michigan (1995)

25. Shaaban, Z.: A new recognition scheme for machine-printed Arabic texts based on neural networks. In: Proceedings of World Academy of Science, Engineering and Technology, vol. 31, July 2008
26. Shafait, F., Rashid, S.F., Breuel, T.M.: An evaluation of HMM-based techniques for the recognition of screen rendered text. In: International Conference on Document Analysis and Recognition, September 2011, pp. 1260–1264 (2011)
27. Slimane, F., Ingold, R., Alimi, A.M., Hennebert, J.: Duration models for Arabic text recognition using hidden Markov models. In: CIMCA, pp. 838–843 (2008)
28. Slimane, F., Ingold, R., Kanoun, S., Alimi, A.M., Hennebert, J.: Database and evaluation protocols for Arabic printed text recognition. In: DIUF—University of Fribourg, Switzerland (2009)
29. Slimane, F., Ingold, R., Kanoun, S., Alimi, A.M., Hennebert, J.: Impact of character models choice on Arabic text recognition performance. In: International Conference on Frontiers in Handwriting Recognition (ICFHR), Novemberf 2010, pp. 670–675 (2010)
30. Slimane, F., Kanoun, S., Alimi, A.M., Ingold, R., Hennebert, J.: Gaussian mixture models for Arabic font recognition. In: ICPR, pp. 2174–2177 (2010)
31. Wachenfeld, S., Klein, H.-U., Jiang, X.: Recognition of screen-rendered text. In: Proceedings of the 18th International Conference on Pattern Recognition—Vol. 02, ICPR'06, pp. 1086–1089. IEEE Comput. Soc., Washington (2006)
32. Young, S.J., Evermann, G., Gales, M.J.F., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland, P.C.: The HTK Book, Version 3.4. Cambridge University Engineering Department, Cambridge (2006)

Part IV

Applications

Chapter 19

A Robust Word Spotting System for Historical Arabic Manuscripts

Mohamed Cheriet and Reza Farrahi Moghaddam

Abstract A novel system for word spotting in old Arabic manuscripts is developed. The system has a complete chain of operations and consists of three major steps: pre-processing, data preparation, and word spotting. In the pre-processing step, using multi-level classifiers, clean binarization is obtained from the input degraded document images. In the second step, the smallest units of data, i.e., the connected components, are processed and clustered in a robust way in a library, based on features which have been extracted from their skeletons. The preprocessed data are ready to be used in the final and third step, in which occurrences of queries are located within the manuscript. Various techniques are used to improve the performance and to cope with possible inaccuracies in data and representation. The latter techniques have been developed in an integrated collaboration with scholars, especially for relaxing the system to absorb various scripts. The system is tested on an old manuscript with promising results.

19.1 Introduction

Word spotting (WS) is one of the basic tools used in the retrieval and understanding of historical documents [33]. This fast approach has shown a good performance where the conventional recognition-based approaches fail. This is of great importance, considering the large number of national and international programs dedicated to preserving and understanding vast numbers of old and degraded historical manuscripts [2, 11, 16, 26]. However, there are some barriers to the application of WS, as historical documents have usually suffered different types of degradation; for example, faded ink, the presence of interfering patterns (bleed-through, etc.), and deterioration of the cellulose structure [19]. Moreover, the writing styles in historical documents are very complex, as the lines are very close together and words

M. Cheriet (✉) · R.F. Moghaddam

Synchromedia Laboratory for Multimedia Communication in Telepresence, École de Technologie Supérieure, Montréal, QC H3C 1K3, Canada
e-mail: mohamed.cheriet@etsmtl.ca

R.F. Moghaddam

e-mail: reza.farrahi@synchromedia.ca

may overlap [1]. Therefore, WS techniques for historical documents must be robust and adaptive.

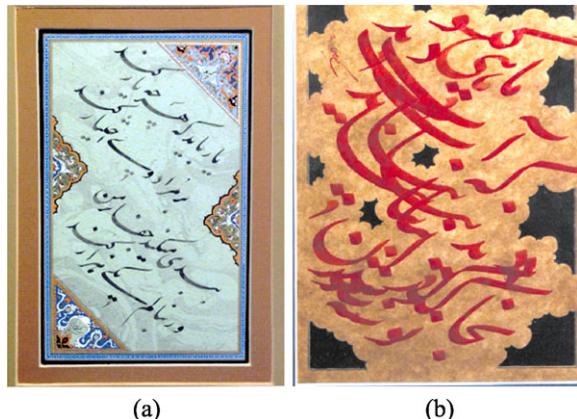
Pattern recognition and document understanding [4, 35] are actually equivalent to a process in which humans (authors, writers, and copiers of documents and manuscripts) are modeled as a kind of machine. These people usually differ in the writing they produce mainly because of their philological and psychological differences and limitations. In other words, some aspects of these variations are rooted in the inability of such a “machine” to reproduce a well-standardized set of patterns. Performance [5] is a term used to measure the level of this imperfection, and is a concept that has been used widely in many fields, from linguistics to engineering. In contrast, a major part of the difference between the writing styles of two manuscripts arises from differences in the thinking of the writers, rather than their inability to replicate a standard style. This brings us to *competence* [5], a concept which has been somewhat ignored in document analysis and understanding. In contrast to performance, competence actually depends on the knowledge of the writer. We believe that a successful approach to the analysis and understanding of historical-manuscript images should be based on accepting differences in competence and allowing enough room for variations. A discussion on performance and competence is provided in the Appendix A. In Fig. 19.1, an Arabic verse is shown in three different scripts. The high level of variation is evident from the figure. Worse still, even within a single script, there can be various styles [7]. For example, in Fig. 19.2, two samples in Persian Nastaliq script are shown. From a visual point of view, there is no similarity between these two patterns. However, from a paleographical point of view, there are common rules that are hidden or are inaccessible directly from the images. The key factor in understanding this is a strong and intense collaboration between pattern recognition researchers and scholars, who can infer the fundamental way of thinking of the writer behind the manuscript. In this work, our objective is to create an integrated collaboration with scholars. The next step is to meld the data from several manuscripts from the same era in order to move beyond standard word spotting. In this way, we are able to build a database of prototypes of queries. Also, close collaboration with paleographers enables us to understand and generate prototypes automatically. This also helps in adapting the model and the method effectively to the specific script of each manuscript. Our long-term goal is to build a basis for understanding historical documents capable of eventually providing transliterations.

WS is usually based on matching a graphical query to the possible candidates from the target document. The direct matching approaches [14], which are based on a two-dimensional (2D) comparison of the regions of interest (ROIs) of the query and the candidate, have been improved by using several mapping techniques, such as dynamic time warping [18] and the Euclidean distance transform [12]. The main drawback of 2D full-comparison approaches is their high computational cost [25]. Moreover, variations in both stroke width and style can reduce the performance of this type of WS. In [15], a WS method has been presented which is based on zones of interest, in order to reduce the computational cost associated with 2D methods.



Fig. 19.1 An Arabic verse produced in several different typefaces: (a) Nastaliq, (b) Sols, (c) Urdu

Fig. 19.2 Two samples of Nastaliq script: (a) Chalipa, (b) Siah-mashq



The matching was performed using gradient-based features, and is limited to possible starting points of words estimated by some guiding measures. The robustness of the method has been improved by generating different possible writing forms of a query word. However, as the method relies on the guides to catch a word, it may be very sensitive to spatial shifts. Also, its dependency on the gradient may result in errors because of the degradation on old manuscripts that leads to difficulties in finding a single threshold value for a whole document page. In [37], a complete word recognition method has been proposed which is inspired from biological problems; it identifies rarely occurring words as possible candidates for special names (people, cities, etc.) and then asks for their labels using an appropriate interface. The corresponding feature vector is very large (around 6,000 features), and the method requires word segmentation data.

In other work, several feature-based approaches have been developed [25, 28]. For example, in a corner-points-based method [28], the shapes have been described and compared by their corresponding corner points. In [25], words are described based on several profiles, such as an upper word profile. The latter is a high-performance method, basically because of its robustness with respect to writing variations as a result of using dynamic time warping [17]. Another method based

on dynamic time warping is presented in [27], in which the semicontinuous hidden Markov model was also used. A set of samples for each word has been selected as the prototype for each query, in order to cope with variations in writing styles. Features are extracted on a sliding patch, which may, however, lead to high computational cost.

Using connected components (CCs) or subwords as the lowest unit has been used previously in the literature [29]. However, in that work, deletion of symbols has been used, which breaks the assumption that CCs are the lowest unit, and brings that method to the “character” level.

In [40], a character-level recognition system is introduced in which various features such as spatial features, histograms, and skeleton-based graphs are used. However, as mentioned, the system is based on character segmentation and is completely different from our approach.

Good segmentation of lines and words is required in many of these methods. The availability of the baseline and the requirement of skew-free word images are some other limitations of these approaches.

The skeleton has also been used in document understanding [24, 34, 39, 41]. However, because of the poor quality of the input images, the skeleton-based methods have not been considered extensively for use with old and historical documents. With new and robust enhancement methods [20], however, weak connections are preserved, and smoothness of the strokes on the historical documents is ensured. This is our main motivation for developing a skeleton-based WS technique.

Although skeletons are actually 2D images, they can be described using features. In this way, the complexity of the skeleton-based approaches is reduced significantly. Usually, topological features have been used in skeleton-based recognition techniques [42]. While topological features are very valuable sources of information, other types of information, such as geometrical data, are also needed, in order to arrive at a complete description of skeletons in the document processing applications.

In this work, we develop a WS system based on comparison of the topological and geometrical skeleton descriptors. The skeletons are obtained from the enhanced document images after the pre-processing step. Also, in order to make the method insensitive to variations in writing style, the lowest-level units, the connected components (CCs, also called subwords), are used for matching. Like the document text, the user query is decomposed into its CCs, and then that sequence of CCs is compared to the document CCs. To reduce the number of comparison operations, the document CCs are first compared to one another, and a library of main or *basis CCs* (BCCs) is created.

The chapter is organized as follows. The problem statement is presented in Sect. 19.2. In Sects. 19.3 and 19.4, our proposed approach is introduced, and many steps such as the process of extraction of descriptors using variable-space transformation are discussed. Then, in Sect. 19.5, the actual word spotting process is described. The experimental results and discussions are provided in Sect. 19.6. Finally, our conclusions and some prospects for future work are presented.

19.2 Problem Statement

The document images of a handwritten Arabic manuscript are available. We assume the document has just one writer. Also, because of the writing style, it is assumed that line and word segmentations are difficult if not impossible. The goal is to search and index a set of few words in the document images without transliterating its text. We propose a novel word spotting technique to solve this problem, of which a brief description is presented in the following section.

19.3 A Skeleton-Based Segmentation-Free Word Spotting Approach

To solve the problem, a shape-based analysis is performed on the set of subwords or CCs of the input document image. Then, the query words will be spotted just by comparing their shapes to those of the document CCs in a fast and real-time way. These two steps are discussed in detail in the following two sections. The first step, which we call data preparation, uses a complexity-based distance on variable-dimension feature spaces to identify similar shapes in the document. The second step, the spotting process, uses this information to retrieve candidates on the document image for the end-user query words.

19.4 Data Preparation

19.4.1 Pre-processing and a Priori Information

Enhancement of historical documents is a key step in their analysis and understanding. Following [20], multi-level classifiers are used to remove the background, restore the interfering patterns, and preserve weak strokes and connections. Depending on the degree of degradation of the input documents, different scenarios can be designed for the pre-processing stage [18].

In this work, a simple pre-processing step is considered, which is shown in Fig. 19.3. As the first step, several parameters are estimated from some selected pages of the input manuscript. These parameters are considered a priori information. The most important parameters are the average stroke width w_s and the average line height h_l (to be discussed in the next subsection). Based on these parameters, a rough binarization of the input images is created. We use a fast grid-based implementation of Sauvola's method [21], the selected scale of which is h_l . Then, using a new kernel-based implementation, based on w_s , the stroke map (SM) [20] of the input images is computed. To improve computation performance, integral image representation [3, 30] is used. Although the SM provides clear binarized views of the images, a few corrections are also considered. It is well known that loops play a

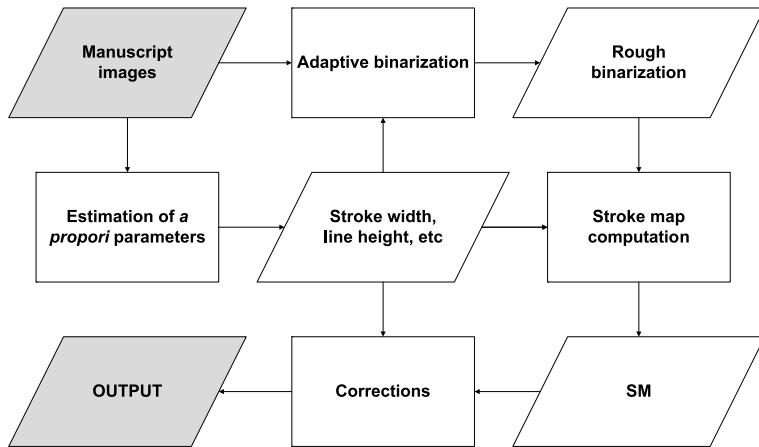


Fig. 19.3 Schematic representation of the pre-processing stage

critical role in recognition and understanding [23]. Usually, some loops are missed because of degradation, or even because of the casual writing style of the manuscript writer. Thanks to w_s and the use of morphological operators, many of the missed loops are recovered. The erosion operator is applied [$2w_s/3$] times considering the BW10 representation [21]. Then, the remaining regions, which are candidate loops, are analyzed based on their size and added to the shape. Another correction involves the removal of small spots, again based on w_s . All spots smaller than w_s are deleted. The final output images are now clean and corrected, and ready to be used in the subsequent stages. In this way, pixels of interest (POIs), which are actually the binarized regions of the text, have been recognized. After pre-processing, the strokes obtained on the document images are smooth and continuous, and accurate and correct skeletonization can be performed. Descriptions of a priori information are presented in the Appendix B.

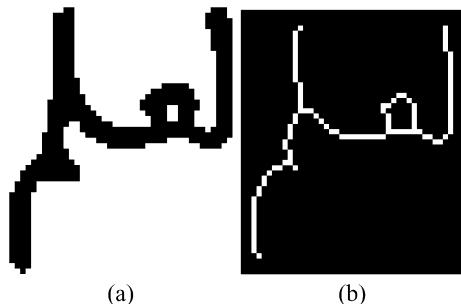
Figure 19.4 shows a sample page of the input document and its preprocessed and clean output image. Having the clean images at hand, the next step is to prepare and extract shape data from them. As mentioned before, the CCs are the lowest unit in our approach. In the preparation stage, a library of CCs is created in which the CCs are clustered. This approach is effective in reducing the computational time of the spotting step. The substeps of the preparation stage are: (i) extraction of CCs, (ii) clustering of CCs into metaclasses, (iii) generating skeletons of CCs and extracting their features, (iv) selecting BCCs, and (v) creating a database of pseudo-words. A sample CC and its skeleton image are shown in Fig. 19.5. There are many approaches for extracting the skeleton of a shape: distance transform-based methods [32], the Voronoi diagram [31], and thinning [10, 42]. On document images, strokes and CCs are actually networks of lines with almost constant width, which means that a thinning method should provide better results in document image processing. We follow the thinning method described in [10].

In the sections below, the details of these steps are discussed.



Fig. 19.4 (a) A sample page from one of the manuscripts. (b) The output of the pre-processing stage

Fig. 19.5 An example of a CC (a) and its skeleton image (b)



19.4.2 Extraction of CCs

We use the standard procedure available via the `bwlabeln` function of Matlab [9, 36] to isolate the connected components. The main problem after extraction of the CCs is the detection of dots and diacritics. Based on w_s , all CCs smaller than a factor of w_s are declared to be dots. Each dot is assigned to a neighboring CC. In order to find the corresponding CC, the minimum weighted vertical distance is used. The *weighted vertical distance* between a candidate dot CC_i and another CC_j is defined as follows:

$$d_{wv}(CC_i, CC_j) = \min_{k \in CC_j} (\alpha |x_{CM_i} - x_k| + \|r_{CM_i} - r_k\|)$$

where $r_{CM_i} = (x_{CM_i}, y_{CM_i})$ is the center of mass of CC_i and $r_k = (x_k, y_k)$ is the position of a black pixel k on CC_j . The parameter α is a large number which weights the distances vertically. We use $\alpha = 10$. The dot CC_i is attached to CC_j with minimum d_{wv} :

$$\bar{j} = \arg \min_j d_{wv}(CC_i, CC_j)$$

Any dot which is further from the corresponding CC than $h_l/2$ is discarded.

In subsequent steps, we need to generate sequences of CCs. To achieve this, the left and right CCs of any CC are determined locally. Because only one CC is searched at this step, this process is very local. Therefore, the accuracy is very high, and the process is robust with respect to baseline variations and the vertical displacement of CCs. The average line height h_l is used as a measure to find the isolated CCs.

19.4.3 Clustering into Metaclasses

Although shape matching could be performed on the whole set of CCs, the presence of a large number of CCs makes this direct matching very time consuming and costly. On average, there are around 500 CCs on each page of the input document. Therefore, the total number of document CCs is very high. Moreover, matching all the shapes against one another may reduce the performance and accuracy of the spotting technique. Therefore, before starting the matching process (which will be discussed in the following sections), the CCs are clustered into a few metaclasses. In this classic solution, distribution of the CCs into a few metaclasses (about 10) is considered. In this way, not only is the accuracy of the method increased, but the number of required comparisons of the user input query to the document CCs is reduced. The exact number of metaclasses depends on the input document and will be determined automatically.

The main matching technique used in this work is *skeleton-based*, and will be discussed in the next section. In order to enhance the performance of the system, an attempt is made to use *skeleton-independent* features at the metaclass clustering stage. We use a selected page of the input manuscript instead of all pages, because these metaclasses are just intended to represent roughly the complexity of the CCs. On this page, using a set of features, and by applying the self-organizing map (SOM) technique [13], an unsupervised clustering map for CCs is obtained. This map will be used for all other pages to partition the CCs based on their complexity. The features used at this stage mainly represent the complexity of the CC shape from various points of view. In this work, eight features are used at the metaclass clustering stage. Below, these eight features used for SOM clustering are listed:

1. *Vertical center of mass (CM)*. It is well known that the position of a word with respect to the baseline is an important feature in understanding and recognition processes. However, as was discussed in the introduction, a smooth baseline has not been followed by the writers of historical documents, and usually baselines suffer from very sharp changes in both direction and position. Even within a single word, subwords can appear far from the average baseline of the word. Therefore, computed and estimated baselines are not very accurate and may lead to major error. This is one of the chief reasons why we adapt a segmentation-free WS technique in this work. Nonetheless, to incorporate and utilize the baseline information in an implicit and robust way, we use a feature which is computed

based on the vertical position of the center of mass (CM) of each CC. In order to reduce the effect of off-the-baseline pixels, a power-law transform is used in the computations. First, a normalized vertical distribution of the CC pixels, H_v , is generated:

$$H_v = \sum_x u_{CC}(x, y) / \max_y \left(\sum_x u_{CC}(x, y) \right)$$

where u_{CC} represents the image of a CC. Then, H_v is modified using the following power-law transform:

$$H'_v = H_v^n, \quad n > 1$$

This power-law transform adjusts H_v toward its maximum point. We use $n = 5$. Finally, the CC's CM is recalculated after applying the power-law transform using the new H'_v as a vertical weight: $u'_{CC}(x, y) = u_{CC}(x, y)H'_v(y)$. To arrive at a common range of variation for all features, the vertical CM is renormalized to between -3 and 3 .

2. *Aspect ratio.* The aspect ratio of the bounding box of a CC is another rich feature which helps to differentiate among special shapes (such as the Aleph in Arabic scripts), and is also helpful for identifying very complex shapes which are suspected to be combinations of two or more touching shapes. The bounding box of a CC includes the CC's shape in such a way that the shape does not touch the boundaries of the bounding box.
3. *Height ratio.* The next feature is the height ratio, which is based on one of the a priori parameters, the average vertical extent of text line h_e . The height ratio of a CC is defined as the ratio of the vertical size of its bounding box to h_e . This feature provides a global measure of the size of a CC.

The remaining four features mostly represent the topological complexity of a CC.

4. *Number of branch points.* The number of branch points and the number of end points (feature number 5) are actually related to the skeleton of the CC. Branch points and end points are defined in the next section.
5. *Number of end points.* (See 4.)
6. *Loop feature.* The next feature, the loop feature, is a logical flag which is set to 1 if there is at least one loop (hole) in the CC's shape.
7. *Horizontal frequency.* The horizontal frequency is used as a discriminative measure of the complexity of the shape. For computing horizontal frequency, the number of horizontal maxima of the smoothed horizontal distribution, H_h , of the CC's shape is used:

$$H_h = \sum_y u_{CC}(x, y) / \max_x \left(\sum_y u_{CC}(x, y) \right)$$

8. *Dot feature.* The dot feature is set to one if the CC has at least 1 attached dot and is 0 otherwise. This feature is as effective as the loop feature.

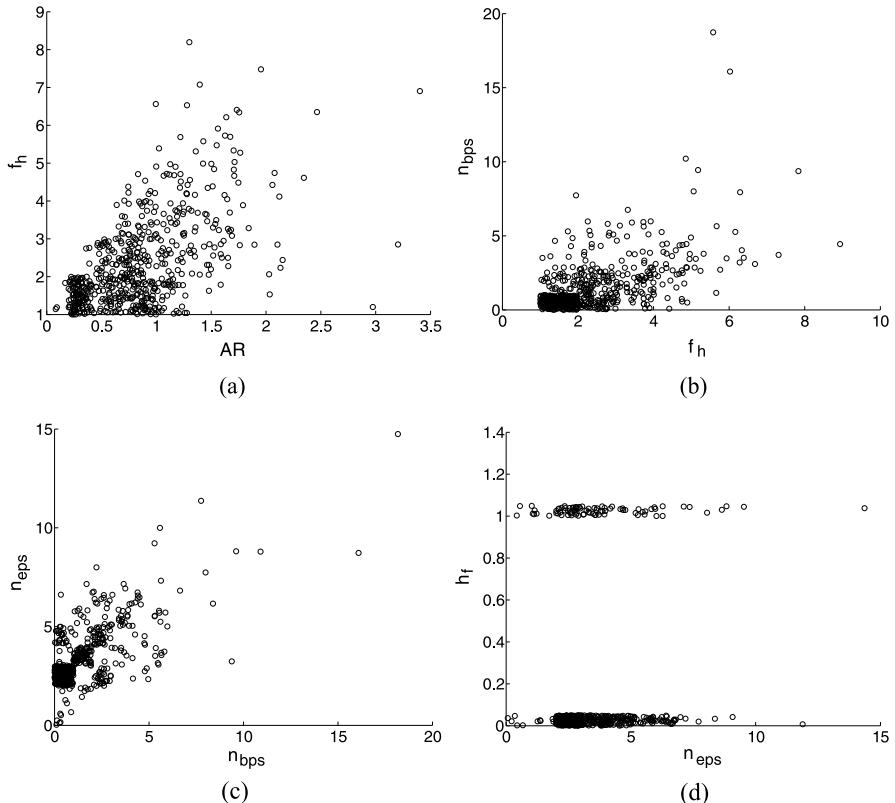
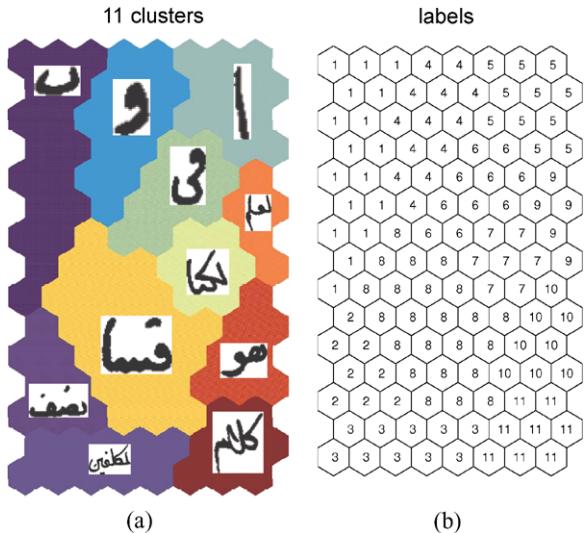


Fig. 19.6 An example of cross distribution of different features used for metaclass clustering. The independence of features can be seen from the distributions. The data are obtained from one page of one of the manuscripts

Figure 19.6 shows a sample of cross distribution of the features against one another computed on a training page. As can be seen from the figure, the features are highly independent. Although some of these features may be correlated lightly to each other, the diversity of the CCs avoids us from selecting a fewer number of features from this small set of features. It is worth noting that these features are not used in calculating the distances in the matching stage (Sect. 19.4.5). It is worth noting that, for better visualization, the data have been *shaken* in the figures by adding small random values.

Figure 19.7 shows an actual SOM of one of the manuscripts. The labels of the metaclasses are selected automatically, and do not have any influence on the succeeding steps. It is obvious that the complexity increases from top to bottom on the map. Because of the special nature of the Aleph letter, one of the metaclasses is dedicated to it. Although this map is obtained just from one page of the manuscript, it will be used for all other pages after calculating the features of their CCs and mapping them on the SOM. It is worth noting that, although metaclasses are very

Fig. 19.7 (a) The SOM of one of the manuscripts obtained from its training page. Sample CCs from the metaclass are shown on the map. The complexity of the CCs increases from *top* to *bottom*. (b) The same SOM with labels of the metaclasses shown on the cells



discriminative, some similar CCs can be placed in two different metaclasses. Therefore, later at the spotting stage, more than one metaclass will be considered.

19.4.4 Generating Skeletons of CCs, and Feature Extraction

The main matching technique in this work is based on comparison of skeletons of the CCs. Here, the process of compilation of the skeleton images into sets of one-dimensional features is described. In this work, a descriptor-based representation is used for each skeleton. The descriptors can be divided into two major categories: topological and geometrical. The details of these two categories in the proposed method are provided in the following sections. Before that, however, we define some skeleton concepts:

- A *singularity point (SP)*: A connection point on the skeleton whose maximum distance to its connected points is less than the minimum dot size.
- An *end point (EP)*: A pixel on the skeleton which has only one connection on the skeleton and is not connected to an SP.
- A *branch point (BP)*: A pixel on the skeleton which has three connections on the skeleton.
- A *branch to an EP*: All pixels on the skeleton between that EP and another EP/BP.

Based on the definition of EP and BP, the possible EPs and BPs of each skeleton are determined and added to the skeleton information.

Before using the skeleton information of a CC, some corrections should be made. The most important of these is recovery of coincident BPs. By definition, a BP has three connections. However, it is possible that the thinning process will lead to a

BP with four connections. These BPs are actually equivalent to two correct, adjunct BPs. The error is corrected by shifting two of the connections and creating two BPs. Also, some of the faded BPs are recovered by comparing the length of the branch to any EP. If the branch is longer by more than a factor of h_e , then a BP is inserted on that branch.

Once the skeletons are available, the next step is to extract their features. As discussed before, we consider two types of features: topological and geometrical.

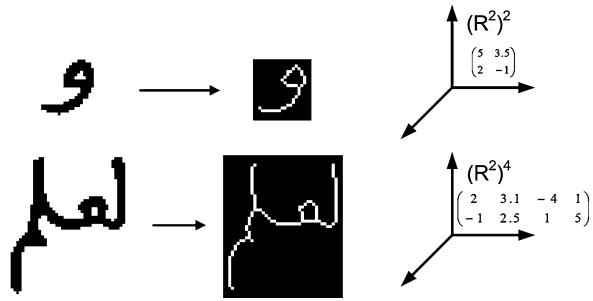
19.4.5 Feature Extraction: A Transformation from the Skeleton Domain to the Feature Space

Although the skeleton image of a CC contains all the information to represent it (ignoring possible errors in the skeletonization step), its complexity is still high. The skeleton is still an image by itself, which requires time-consuming 2D matching methods. Moreover, the matching methods should be robust with respect to the possible variations in the skeletons' form. Ignoring the 2D scale transforms [18], the other solution is to transform the useful skeleton information to other spaces. For this purpose, transformations, such as the topological graph matrix [34] among others, can be considered. However, as discussed before, a CC's information consists of more than just topological data. Here, as a general approach, we consider several transformations on the skeleton image to several spaces representing different aspects of the shape under study. The details of each transformation will be provided below in their corresponding subsections. Let us consider u_{skel} to be the skeleton image of a typical CC, where $u_{\text{skel}} : \Omega_{\text{skel}} \rightarrow \{0, 1\}$, and $\Omega_{\text{skel}} = \Omega_u \subset \Omega \subset \mathbb{R}^2$. The domain Ω is the domain of the whole page that hosts the CC under study. Let us call T the set of transformations that maps u_{skel} to the proper spaces: $T = \{T_i | T_i : \Omega_{\text{skel}} \rightarrow (\mathbb{R}^{m_i})^{n_{i,u_{\text{skel}}}} i = 1, \dots, n_T\}$, where n_T is the number of transforms, and, for each transform T_i the target space has m_i $n_{i,u_{\text{skel}}}$ dimensions. m_i depends only on the transformation T_i , while $n_{i,u_{\text{skel}}}$ also depends on the complexity of u_{skel} . It is assumed that each transform produces features that are expandable depending on the complexity of the input skeleton image. This expandability is formulated as the variable $n_{i,u_{\text{skel}}}$ parameter. In other words, the transformation T is a variable transform for each CC. In particular, the dimension of the target spaces is variable. This characteristic is very important in defining distance, which will be discussed in the next subsection. In summary, T provides a 1D representation for the skeleton images:

$$u_{\text{skel}} \xrightarrow{T} v = \{f_i\}_{i=1}^{n_T}, \quad (\mathbb{R}^{m_i})^{n_{i,u_{\text{skel}}}} \ni f_i = T_i(u_{\text{skel}}) \quad (19.1)$$

Figure 19.8 shows a typical example of how the target space of a transformation depends on the complexity of the input skeleton image. The second shape is more complex, and therefore its feature space is bigger. Details of the transformations are provided in the following section.

Fig. 19.8 A typical example of how a transformation target space depends on the complexity of the input skeleton image



In summary, after applying T , each CC will be represented by a corresponding set of features, such as $\{f_i\}_{i=1}^{n_T}$. This approach keeps open the possibility of extending the description of the CC if needed. At the same time, the descriptor is very concentrated, and therefore computational time is short compared to that of full 2D matching techniques. It is worth noting that the computational cost of the spotting step is much less, because the preparation step is separated and is carried out offline. In the next subsection, the distance suitable for this representation is defined.

Distance in the Feature Space

Let us consider a set of CCs represented by $\{v_\psi\}_{\psi=1}^{n_{CC}}$ features, where n_{CC} is the number of CCs and $v_\psi = \{f_{i,\psi}\}_{i=1}^{n_T}$, $f_{i,\psi} \in (\mathbb{R}^{m_i})^{n_{i,\psi}}$. Here, $n_{i,\psi}$ is equivalent to $n_{i,u_{skel}}$ in (19.1), where u_{skel} is replaced with the skeleton image of CC_ψ . The goal is to define a distance which provides a robust and effective discrimination between CCs, despite their variable representation. We call the distance between two CCs, say CC_ψ and CC'_ψ , $d_{\psi,\psi'}$:

$$d_{\psi,\psi'} = \sum_{i=1}^{n_T} w_i d_{i,\psi,\psi'} / \sum_{i=1}^{n_T} w_i \quad (19.2)$$

where $d_{i,\psi,\psi'}$ is the distance between $f_{i,\psi}$ and $f_{i,\psi'}$, and w_i is the associated weight. Their assigned weights correspond to the complexity of the $f_{i,\psi}$ and $f_{i,\psi'}$ spaces, and act as normalization factors.

Because of the difference between the dimensions of the $f_{i,\psi}$ and $f_{i,\psi'}$ spaces, d_i is defined as follows:

$$d_{i,\psi,\psi'} = \eta_{i,\psi,\psi'} (d_{i,c} + d_{i,s}) \quad (19.3)$$

where $d_{i,c}$ is the minimum distance between $f_{i,\psi}$ and $f_{i,\psi'}$ on a *common intersection* subspace of their spaces, $d_{i,s}$ stands for the difference between the space dimensions, and $\eta_{i,\psi,\psi'}$ is a normalization factor that takes into account the complexity of the shapes and will be discussed later. $d_{i,c}$ searches for the best overlapping of $V^{i,\psi}$

and $V^{i,\psi'}$. Let us assume, without loss of generality, that $n_{i,\psi} \geq n_{i,\psi'}$. Then, in mathematical notation, we have

$$d_{i,c} = \min_{\hat{V}^{i,\psi}} d_i(\hat{f}_{i,\psi}, f_{i,\psi'}) \quad (19.4)$$

where $\hat{f}_{i,\psi} \in \hat{V}^{i,\psi} = (\mathbb{R}^{m_i})^{n_{i,\psi}} \subset (\mathbb{R}^{m_i})^{n_{i,\psi}} = V^{i,\psi}$ is a subselection of $f_{i,\psi}$ on $\hat{V}^{i,\psi}$ as a subset of $V^{i,\psi}$. For the definitions of m_i and $n_{i,\psi}$, see Sect. 19.4.5. In Eq. (19.4), $d_i(\hat{f}_{i,\psi}, f_{i,\psi'})$ is fixed and does not search for the best dimension. Here, the details of $d_i(\hat{f}_{i,\psi}, f_{i,\psi'})$ are presented. Assuming that $f_{i,\psi} = \{\phi_{i,\psi,1}, \dots, \phi_{i,\psi,n_{i,\psi}}\}$ where $\phi_{i,\psi,k} = (\phi_{i,\psi,k,l})_{l=1}^{m_i} \in \mathbb{R}^{m_i}$, we then define

$$d_i(\hat{f}_{i,\psi}, f_{i,\psi'}) = \sum_{k=1}^{n_{i,\psi}} \|\phi_{i,\psi,k} - \phi_{i,\psi',k}\|_1$$

where $\|\cdot\|_1$ is the L_1 norm. This norm is selected because of the heterogeneous nature of the $\phi_{i,\psi}$ elements.

The second term in definition (19.3), $d_{i,s}$, represents the distance resulting from the difference in complexity of the two shapes:

$$d_{i,s} = \frac{1}{n_{i,\psi}} (n_{i,\psi} - n_{i,\psi'}) \left(1 - \frac{1}{2^{m_i}}\right)$$

The first factor, $n_{i,\psi} - n_{i,\psi'}$, is the distance between the dimensions of two spaces $V^{i,\psi}$ and $V^{i,\psi'}$, while the second factor, $1 - 1/2^{m_i}$, stands for the correction of the randomly generated extra dimensions. If $CC_{\psi'}$ has extra complexity because of computational error (i.e., if $n_{i,\psi'}$ is, for example, its true value plus one), then that extra dimension results in a change in $d_{i,c}$. If we assume that the element values of that extra dimension are independent of the other dimensions and are random, considering binary values for the elements, the amount of change in the distance is, on average, $1 - (1/2)^{m_i}$. Therefore, we use the same amount for the unmatched dimensions in $d_{i,s}$.

In definition (19.3), $\eta_{i,\psi,\psi'}$ stands for a transformation of the distances based on the complexity of the CCs. Although in partitioning the CCs into metaclasses, CCs with different complexities are placed in proper metaclasses, as will be discussed in the section on word spotting, the distance between CCs from different metaclasses is needed to capture misclassified CCs which may have been mapped to other metaclasses. Also, in order to make the clustering process coherent over all metaclasses, the distance d_i should be normalized in some way with respect to complexity. Although d_i in (19.3) is sensitive to the complexity of its inputs, the effect of complexity on the distance in metaclasses with shapes of low complexity is different, and can be explained as follows. In shapes of low complexity, the resolution of the description is very rough. In other words, very fine discrimination between shapes is impossible. This means that many features between two shapes of low complexity may be the same, while the shapes themselves are completely different. This effect

scales down the distance value in these metaclasses. In order to compensate for this effect, the distance in metaclasses of low complexity should be corrected according to their complexity. This can be performed by adjusting η . For shapes of low complexity, η is shifted to a minimum average value, which represents the average complexity of all shapes:

$$\eta_{i,\psi,\psi'} = \max(\eta_{i,0}, \hat{\eta}_{i,\psi,\psi'})$$

where $\eta_{i,0}$ is the minimum average value, and $\hat{\eta}_{i,\psi,\psi'} = \max(\hat{\eta}_{i,\psi}, \hat{\eta}_{i,\psi'})$ where $\hat{\eta}_{i,\psi}$ stands for the complexity of the shape CC_ψ with respect to T_i . In this work, $\hat{\eta}_{i,\psi}$ is set to $n_{i,\psi}$.

In the next subsections, based on the collaboration with scholars, we select a number of topological and geometrical features for relaxing the method to absorb various scripts.

Topological Features

In this and the following subsections, a set of transformations compatible with the aforementioned transformations formulation, introduced in Sect. 19.4.5, is presented. $n_T = 6$, in our case. The topological features of skeletons have been used extensively in object recognition. In this work, we use a set of topological features adapted to document images. The first transformation, T_1 , assigns a set of features to each BP of a CC:

1. $\phi_{1,\psi,k,1}$: Is BP connected to a loop?
2. $\phi_{1,\psi,k,2}$: Is BP connected to an EP?
3. $\phi_{1,\psi,k,3}$: Is BP connected to another BP?

where ψ is the CC index, and k counts on all BPs of CC_ψ . Therefore, $m_1 = 3$, and $n_{1,\psi}$ is equal to the number of BPs of CC_ψ . The second transformation, T_2 , generates topological features associated with EPs:

1. $\phi_{2,\psi,k,1}$: Is EP connected to a BP?
2. $\phi_{2,\psi,k,2}$: Is EP connected to another EP?
3. $\phi_{2,\psi,k,3}$: Vertical state of EP with respect to vertical CM.

Now, we have $m_2 = 3$, and $n_{2,\psi}$ is equal to the number of EPs of CC_ψ .

Also, the states of the EPs and BPs with respect to the dots (SPs) are converted into two additional feature sets. For BPs the state takes into account the vertical location of the dot (whether it is above or below the BP) and is compiled as T_3 . It has two features:

1. $\phi_{3,\psi,k,1}$: Is there any dot above BP?
2. $\phi_{3,\psi,k,2}$: Is there any dot below BP?

Therefore, $m_3 = 2$, and $n_{2,\psi}$ is equal to the number of BPs of CC_ψ . The next transformation, T_4 , does the same job for EPs. However, because of the high degree of variation in the position of dots with respect to EPs, we just consider the existence of a dot near the EPs ($m_4 = 1$):

1. $\phi_{4,\psi,k,1}$: Is there any dot near EP?

In both T_3 and T_4 , the nearest state is calculated implicitly, and each dot is being assigned to the *two* closest BPs (and EPs); to tolerance the writing variations, a dot is assigned to its two nearest BPs (and EPs). The last topological transformation, T_5 , describes the state of dots:

1. $\phi_{5,\psi,k,1}$: Is the dot above the vertical center of mass?

Therefore, $m_5 = 1$, and $n_{5,\psi}$ is equal to the number of dots of CC_ψ . In summary, there are 5 topological descriptor sequences for each skeleton. It is worth noting that, although we call T_i , $i = 1, \dots, 5$ topological transformations, geometrical features are presented implicitly within them. For example, $\phi_{2,\psi,k,3}$ is essentially geometrical. However, as the transformation in the following subsection considers the actual shape of branches between EPs and BPs, we prefer to call it solely a geometrical transformation.

Geometrical Features

Although topological descriptors contain a large amount of skeleton information, they are not sufficient for complete description of a stroke skeleton. Here, an additional transformation, T_6 , is introduced and is assigned to each EP based on the geometrical attributes of the skeleton branches:

1. $\phi_{6,\psi,k,1}$: Is the branch associated with the EP clockwise?
2. $\phi_{6,\psi,k,2}$: Is the branch S-shaped?
3. $\phi_{6,\psi,k,3}$: Vertical location of EP with respect to its corresponding BP.

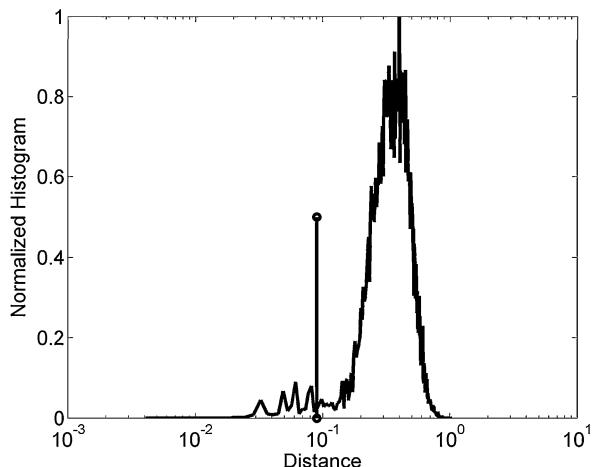
Therefore, $m_6 = 3$, and $n_{2,\psi}$ is equal to the number of EPs of CC_ψ . The S-shaped state and direction of a branch are determined using numerical fitting of a Bézier curve on the branch [6].

It is worth noting that the variable-dimension descriptors introduced here could be converted to fixed length feature vectors if needed. For an example, see [22].

19.4.6 Creating the Library

Here, a review of the process of building the library is presented. In order to have a line-segmentation- and word-segmentation-free system, the CCs of each page of the document are extracted directly. The position (the center of mass) and the bounding box of each CC are generated and added to the page information. Based on the minimum and maximum dot sizes, as well as the average line height, small CCs are discarded or attached to other CCs (as dots). Then, the skeleton information of the CCs is generated, as discussed in the previous section. Based on this information, the CCs can be compared to one another or to possible user queries.

Fig. 19.9 Distribution of distances between the CCs of a training page. The threshold distance obtained using Otsu's method is also shown on the distribution



As discussed above, in order to reduce the number of comparisons, the document CCs are first grouped into a library, which consists of a set of metaclasses. Each cluster is a set of similar BCCs. In each metaclass, the CCs are clustered together. The prototype or representer of each cluster is called a basis CC (BCC). All new CCs, are compared against the existing BCCs. Based on the distance of a CC to the BCCs in a metaclass, that CC is assigned to one of the BCCs or appears as a new BCC. This process is performed in two rounds. First, all the CCs of a new page are compared to the existing BCCs in the library in the corresponding metaclasses. Then, CCs which should appear as new BCCs are added to the metaclasses. In the second round, the remaining CCs are compared again to the updated metaclasses and are distributed on the clusters.

Using the six descriptor sequences, consisting of five topological descriptors and one geometrical descriptor, the features of each CC are available. According to the complexity of the CC, the length of the sequences will be variable. The distance between two skeletons is computed based on definition (19.2).

A threshold distance is used as a measure in clustering. Figure 19.9 shows a sample distribution of the distances between all CCs obtained from the training page. As can be seen, the distribution consists of two distributions. The distribution on the left corresponds to similar CCs. Using Otsu's method, the threshold distance, d_{thr} , between two distributions is obtained. In our experiments, the value has been between 0.08 and 0.1 for different manuscripts. This can also be interpreted as the uncertainty in the skeleton descriptors. For example, a missing BP in the descriptors of a skeleton with 6 BPs will introduce a distance of around 0.08.

As the process of extracting skeleton information and generating descriptors is independent of the comparison process, it is performed before that process. This makes the computational complexity of the system very low.

19.4.7 Generating Pseudo-words

After generating the BCC library of a document, the user queries can simply be compared to the BCCs in the relevant clusters, and corresponding BCCs can be obtained. Also, all the CCs that are clustered around each BCC are known. Therefore, possible CCs of the user query are easily available. However, the user query is a sequence of CCs (a word). Therefore, a correct combination of the CCs spotted must be found. To enhance the performance of the system, an extra step is added before the user is allowed to search the document. In this step, a database of all possible CC sequences in the document, which we call *pseudo-words*, is created. A schematic diagram of this process is shown in Fig. 19.10. Each pseudo-word, which is a set of a few CCs which are close together in the horizontal direction, is labeled by its BCC sequence and assigned to the first CC in the corresponding CC sequence. To generate the pseudo-words, the neighboring CC information that was created in the CC extraction step (see Sect. 19.4.2) is used. As an example, in Fig. 19.11, the BCC labels of sequences started from a typical CC, CC_i , are shown. The sequences are sent to the appropriate databases within the pseudo-word database. When the sequence of BCCs of the user query is obtained, the possible pseudo-words are extracted from the database, and the position of the spotted sequence is determined using their corresponding first CC. To speed up the searching process in the spotting process, the pseudo-word database is sorted. As the BCC labels are integer, sorting is fast and effective. It is worth noting that the third sequence, $BCC_nBCC_mBCC_l$, is not actually a word, but is considered to be a possible sequence in the database.

19.4.8 Application of Markov Clustering (MCL) in Correcting the Library and BCCs

Because of the large number of CCs and the incremental processing they require, the library is built up gradually and new BCCs are generated when needed. One drawback of this progressive approach is that the BCCs generated, which are usually from the beginning of the document, may not best represent the clusters in the library, and, at the same time, break down clusters among false clusters centered on the outliers of the true clusters are possible. In order to address this problem, while trying to maintain the progressive nature of the library construction, correction of BCCs, which represent the kernel of clusters, is considered. This correction is performed in a periodic way in alignment with the main library growth process. The details of the correction procedure are presented below. In brief, in each cluster in the library, the CCs are compared to one another to select a new representer for that cluster, as its new BCC. In order to increase the robustness of the selection process, the Markov clustering (MCL) [38] technique is used. This technique has been proved to perform well without any need for extra effort in terms of parameter adjustment. In other words, although parameter adjustment can be performed in MCL to improve its execution time performance, its clustering performance is insensitive

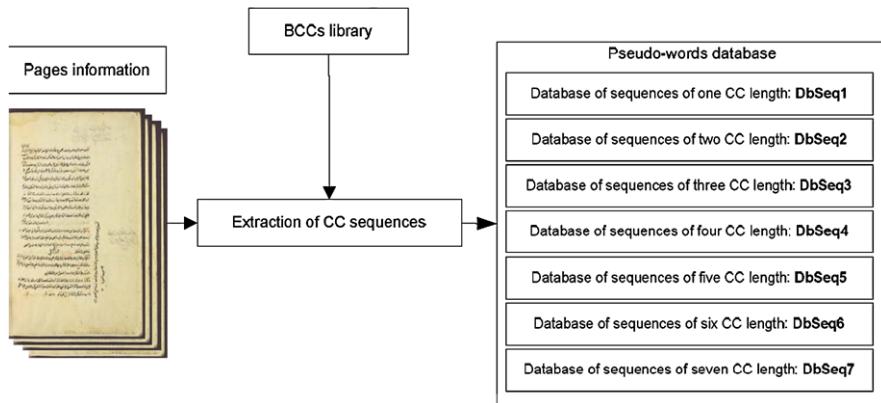


Fig. 19.10 A schematic view of the process of generating pseudo-words

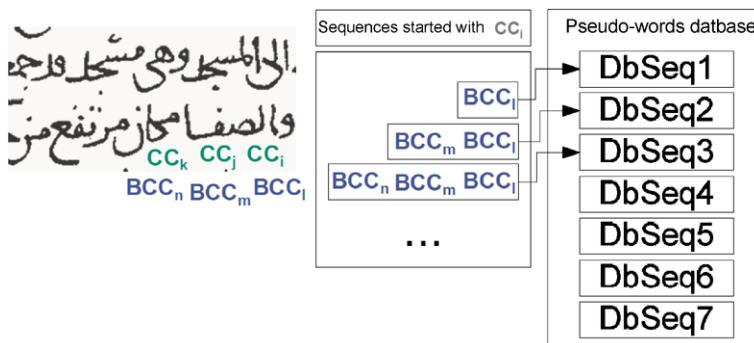


Fig. 19.11 An example of how pseudo-word sequences are generated

to its parameters on a wide range of values. Algorithm 1 provides the steps for correcting a cluster. Consider, for example, a cluster which corresponds to BCC_k . Let us call this cluster CL_{BCC_k} . First, the distance matrix, D , is calculated:

$$D_{ij} = d_{CC_i, CC_j}, \quad CC_i, CC_j \in CL_{BCC_k}$$

This matrix is symmetrical, and so only the distances between the CCs inside the cluster under process are required. This feature helps keep the computational cost low. Then, the similarity matrix, W , is computed:

$$W_{ij} = \exp(-d_{ij}^2 / h_{MCL}^2) \quad (19.5)$$

where h_{MCL} is set equal to the distance threshold, d_{thr} . It is obvious that the diagonal elements of W are all 1. Therefore, W can be directly used as the stochastic matrix in MCL [38]. The MCL technique, which is described briefly in the Appendix C, provides a binary matrix in which similar objects have a value of 1, and other values

are zero. Therefore, an object with the highest number of 1s on its corresponding row in the output matrix of MCL is a good choice for the new representer or BCC of the cluster. In other words, the new BCC is $CC_{\tilde{j}}$, where

$$\tilde{j} = \arg \max_j \sum_i W_{ij} \quad (19.6)$$

In the correction step, the technique described above is applied to all clusters in the library. In this work, the correction step is applied at the end of the processing of each page. This is reasonable, as the probability of there being similar shapes on a page is high. In Fig. 19.12, the process of selecting a new BCC for a typical cluster is shown. Figure 19.12(a) shows a graph of a cluster, with the edges weighted based on W_{ij} . An initial BCC of the cluster is also shown. As can be seen from the figure, that BCC is not the best representer of the cluster. In Fig. 19.12(b), the state after application of MCL is shown. A subcluster survives this process, and the center of this subcluster is chosen as the new BCC and the representer of the cluster under study.

Set parameters p and r : $p = \dots$ and $r = \dots$;

Get the similarity matrix, W , according to Eq. (19.5);

Apply the MCL algorithm as described in Appendix C;

Find the row with maximum summation in W ;

Select the corresponding CC of that row as new BCC of the cluster;

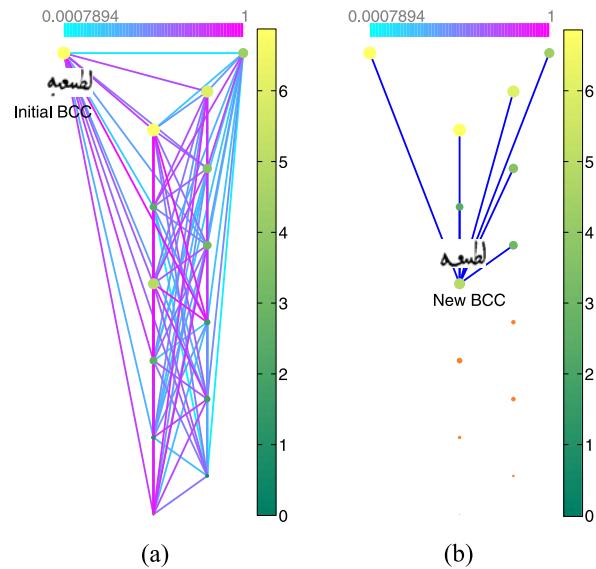
Algorithm 1: Application of MCL to select a new BCC for a cluster

19.5 Spotting Process

After the preparation stage, a library of BCCs and the database of pseudo-words of the manuscript under study are available. In the library, the CCs are distributed among several metaclasses. In each metaclass, the CCs are clustered around BCCs. In the pseudo-word database, all observed sequences of CCs of different lengths (from 1 to 7) are collected. Each sequence is labeled by the sequences of the corresponding BCCs of the CC sequence. The address of the first CC of the sequence is also stored in the database. With these data, the spotting stage can be performed anytime. The details of the spotting process are presented below.

In the spotting process, a search is made for a graphical query from the user in the manuscript. It is assumed that the user provides the graphical query, which is the image of the word in question. Also, it is assumed that the query is written in a very similar style to the writing style of the manuscript. First, the query image is analyzed in a similar way to the document images. The steps of the spotting stage are listed in Algorithm 2. The first step is to process the query. This is done

Fig. 19.12 Application of MCL to correct BCCs.
(a) A typical cluster graph.
(b) The graph after applying MCL



in the same way as the manuscript images themselves are processed: the query is preprocessed, and CC extraction and corrections are performed. In this step, the a priori information of the manuscript is used. With clean and processed CCs and their skeletons, each CC from the query is mapped to a meta-class. Although meta-classes provide a good distribution of CCs, it is possible that a CC from the query will be mapped to the wrong meta-class. This is more likely to occur on the borders of the meta-classes. Therefore, in order to enhance the performance of the system, for cells on the borders of the meta-classes, the neighboring meta-classes are also considered as candidate meta-classes. For example, in Fig. 19.13(a), the border cells of meta-class 8 of a manuscript SOM are shown in yellow. The interior cells are in white for that meta-class. If a query CC is mapped in an interior cell of that meta-class, just one meta-class is considered. In contrast, if a CC is mapped in a border cell (yellow), the neighboring meta-classes are also considered for searching. For example, in Fig. 19.13(b), the searchable meta-classes are shown in gray for one of the border cells from meta-class 8. Another example for a border cell of meta-class 10 is shown in Fig. 19.13(c). Having the corresponding meta-classes of each query CC, the closest BCCs from each meta-class are determined. More than one BCC from each meta-class is considered in order to compensate for the possible variations in writing style and touching strokes, and also errors in feature extraction. Although this can also introduce incorrect BCCs, as will be discussed below, the impact of these wrong BCCs is small, because of the limited number of possible combinations of the database sequences. We consider at most 4 candidate BCCs from each meta-class. For example, if a CC is mapped in a cell which has two neighboring meta-classes, 12 candidate BCCs will be assigned to the CC. Using these data, all possible sequences of BCCs are created. For a query of length 3 CCs, 12^3 sequences could be obtained. The final step is to search for these sequences in the pseudo-word database. Be-

cause the labels of BCCs are integers and the database of pseudo-words is sorted, the candidate sequences can be easily found in the database. The main challenge in the spotting step is generating the graphical queries or prototypes, which will be discussed in the next subsection.

- Get the user graphical query;
- Process the query to extract its CCs;
- Map the CCs on the metaclasses;
- Find the corresponding BCCs of each CC from metaclasses;
- Generate candidate sequence of BCCs;
- Search the candidate sequence in the pseudo-word database;
- Present the matched sequence to the user as the spotting result;

Algorithm 2: Spotting process

19.5.1 Prototype Preparation

The main challenge for the WS system is the selection of an appropriate set of graphical queries to be spotted. Because of drastic variations in writing style, a single query cannot be used on all manuscripts. One approach is to ask the user to write a graphical query in a style that is as close as possible to the style of the document under study [18]. In this work, another approach is used to reduce the effort of the end-user, which is to ask that individual to manually spot a few occurrences of the query on the document images. Then, the selected words are used as the graphical queries. The benefit of this method is that the prototypes obtained in this way are exactly in the style of the writer of the manuscript (who may have written many other manuscripts). Moreover, many features of the personal style of the writer, such as overlapping and touching of specific parts of words, are reflected in the prototypes. This enhances the performance of the system. The long-term idea behind this approach is to build a collection of prototypes for each keyword. The collection is partitioned into different sets, each representing a writer, a writing style of a certain period, or a script in general.

As an enhancement to the system, each approved spotted occurrence of a query is used as a new prototype to be searched. We call this enhancement the *extended prototype set*.

19.6 Experiments and Results

The performance of the spotting method has been evaluated against manuscripts provided by the Institute of Islamic Studies (IIS), at McGill University, Montreal.

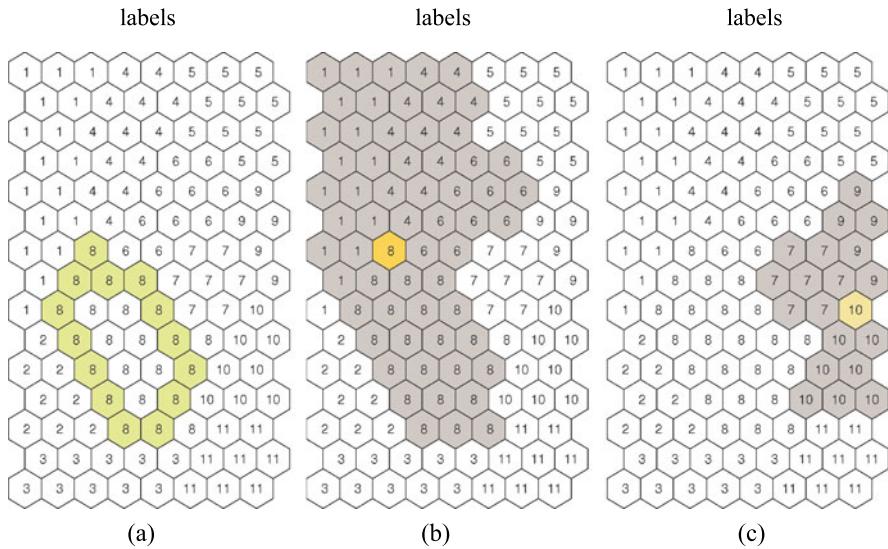


Fig. 19.13 (a) Border cells and interior cells of a metaclass. (b) The participating metaclasses and cells of the cell shown in yellow if a CC is mapped in that cell. (c) Another example for a border cell from another metaclass

The IIS is home to the *Rational Sciences in Islam*¹ (RaSI) project, which is in the process of creating a large-scale, unique database of Islamic philosophical and scientific manuscripts, most written in Arabic, but some in Persian and Turkish. One of RaSI's component initiatives is the *Post-classical Islamic Philosophical Database Initiative* (PIPDI), directed by Dr. Robert Wisnovsky. The PIPDI is responsible for the indexing of metadata pertaining to, and data contained in, approximately 600 works of philosophy produced during the post-classical period of Islamic intellectual history (1050–1850). Its dataset is stored in 30 terabytes of storage space.

The first experiment is performed on a work by *Abu al-Hasan Ali ibn Abi Ali ibn Muhammad al-Amidi* (1243 A.D. or 1233 A.D.). The title of the manuscript is *Kitab Kashf al-tamwihat fi sharh al-Tanbihat* (commentary on *Ibn Sina's al-Isharat wa-al-tanbihat*). Among the works of Avicenna, this one received the attention of later scholars more than the others. It was particularly well received in the period between the late twelfth century and the first half of the fourteenth century, when more than a dozen comprehensive commentaries were written on the work. These commentaries seem to have been one of the main avenues to understanding and developing Avicenna's philosophy, and therefore any study of the post-Avicenna philosophy needs to pay specific attention to these commentaries. *Kashf al-tamwihat fi sharh al-Tanbihat*, one of the early commentaries written on *al-Isharat wa-al-tanbihat*, is

¹<http://islamsci.mcgill.ca/RASI/>; <http://islamsci.mcgill.ca/RASI/pipdi.html>

an as-yet-unpublished commentary, and awaits the attention of scholars. The document images have been obtained using camera imaging (21 megapixels, full-frame CMOS sensor, 1/4 sec shutter speed) at 300 DPI resolution. Each full color image is around 800 KB in size. The dataset consists of 51 folios, corresponding to 24,883 CCs (almost 500 CC on each folio). Folio 4a is used as the training page (shown in Fig. 19.4(a)). The following parameters were extracted from the training page: $w_s = 7$, $h_l = 130$, and $h_s = 70$. The value of the distance threshold is obtained from the histogram of distances on the training page: $d_{\text{thr}} = 0.0823$. All shapes smaller than a minimum dot size (which is selected as a fraction of w_s , 5 pixels) are discarded. The maximum dot size is assumed to be 12, and two different factors for the horizontal and vertical directions considered for the dots. In the horizontal direction, the dot size is multiplied by 3, but for the vertical direction the factor is 2. Also, to increase the speed, if the difference between space dimensions is more than a maximum value, a distance of 1 is assigned to the corresponding d_i . We used 6 as that maximum value. Also, because of the irregular behavior of dots, a lower weight is assigned to dot-based features (T_3 to T_5). For this purpose, a down factor of 0.2 is used. In this way, while the method benefits from the presence of dots to provide more accurate differentiation, it has a high tolerance for dot position (see Sect. 19.4.5) and missing or extra dots. Also, SOM clustering, which consists of 11 metaclasses, is learned from the training page (shown in Fig. 19.7). With the clean document images, the BCC library is created. The library contains 3420 BCCs. The library is prepared using two approaches, one without correction of BCCs, and another with correction (Sect. 19.4.8). In order to limit the computational load of the correction step, it is only applied to clusters with a few CCs (between 3 and 15). The computational overhead for the correction of the library is 10 %. It is worth noting that the pre-processing and preparation stages are completely independent of the spotting stage. Therefore, they can be scheduled to be performed beforehand. When the library and pseudo-word database are ready, the user can be exposed to the manuscript. The computational time needed for pre-processing a single image is approximately 40 seconds. Also, the preparation steps (extraction of CCs and generation of their skeleton features) take 460 seconds for each document image. This preparation time is hidden from the end-user. The end-user faces only the computational time of the spotting step, which is less than one second for the current database. The next step, generation of the pseudo-words, is performed, and seven databases of pseudo-words of up to seven CCs in length are built. The total number of pseudo-words is 138,688.

Once all the data have been prepared, the next step is the actual spotting of any keyword in the manuscript, which can be done anytime the end-user wants to do it in real time. The main challenge is providing the graphical queries. In this work, samples from the manuscript are used. The end-user is asked to select a few samples from the document images. More than one sample is used, in order to compensate for possible topological variations in the writing style such as the touching of strokes. The spotting step is performed on a set of 20 keywords provided by the scholars from the IIS. The graphical queries are processed in the same way as the manuscript images. Once the CCs of a query are available, they are mapped against

Table 19.1 Performance of the system in terms of precision, recall, and F-measure. The best performance is obtained when both library correction and expansion of query set are used. In this case, the average hit rate per query is also higher

	Without expansion of queries			With expansion of queries			Avg hit rate	Max hit rate
	Precision	Recall	F-measure (%)	Precision	Recall	F-measure (%)		
Without library correction	0.6528	0.4429	51.21	0.7266	0.5502	61.87	0.3017	0.7144
With library correction	0.7126	0.5429	60.85	0.7721	0.6989	73.20	0.4513	0.8094

the library to find their possible BCC matches. To compensate for misclassified CCs, more than one metaclass is used to search for possible matches when a query CC is on a border cell of a metaclass. We use all the neighboring metaclasses in this case. Also, more than one BCC is sought in each metaclass in order to cover small variations and to capture mistakes in skeletonization. In the end, for each query CC, at most 12 candidate BCCs are selected. Using these matches, all possible pseudo-words are constructed and searched in the pseudo-word database, resulting in words spotted on the manuscript. The first column of Table 19.1 shows the performance of the system with and without library correction. As seen from the measures, a high boost is obtained after the correction. The last column of the table, which provides the hit rates, also proves the effectiveness of the correction step. The hit rate is defined as the ratio of the number of spotted words, starting with a single query, to the total number of occurrences of that word. The table provides the average hit rate of all 20 keywords and the maximum hit rate obtained within these keywords. Figure 19.14 shows a set of spotted words. The variations in the topology and style can be seen in the figure.

To reduce the labor load on the end-user for the selection of the prototypes of the queries, as an expansion, the words spotted in the first run obtained starting from the selected graphical queries, are used as a new set of queries in a second run. The second column of Table 19.1 shows the performance after this extension. The main limits on the performance of the system are mistakes in skeletonization (such as missed loops and holes, cuts in strokes, and undetected small branches), errors in description of skeletons (especially in geometrical features that describe the behavior of a branch), and sensitivity of the distance function to the large variations in the complexity of CCs. Also, there are many challenges from the perspective of writing style: a word can have various forms, which are totally different in topology and geometry, even within a single manuscript, including touching strokes and CCs. Our approach of having more than one query for each word extracted from the manuscript seems to be effective, and we hope it will lead to labeled databases of graphical queries organized based on the script and writer's century. In this way, the end-user will simply select the proper set of queries by entering a query string.



Fig. 19.14 A graphical query is shown (in a box), along with a few of its spotted instances. The ability of the system to tolerate writing variations and the presence of dots can be seen from the spotting results

The computational complexity of the system is very low. The descriptors are generated before the spotting process, which reduces the complexity of the method. The computational cost of the system is one order of magnitude lower than 2D matching methods (Euclidean distance transform, for example).

19.7 Conclusions and Future Prospects

A complete system for word spotting in historical documents is developed. The system is based on the skeletons of the connected components. The skeletons are described using six descriptors based on topological and geometrical features which can be seen as a set of transformations to variable-dimension feature spaces. The features are associated with the branch points and end points on the skeletons. The distance between shapes is computed using a robust and nonsensitive distance, which considers the possible difference in the space dimensions. The matching technique, which is built using this distance, does not require line or word segmentation, and therefore is very robust and insensitive to writing style. The performance of spotting is promising, because of the continuous and smooth strokes that are provided by the pre-processing step. This step is performed automatically using multi-level classifiers. First, the manuscript data are prepared in a library of CCs, which contains all the CC clusters, and the pseudo-word database. The system has been tested on a manuscript from the Institute of Islamic Studies (IIS) with promising

results. Note that the system can be easily adapted to a dotless strategy, simply by dropping the SP-based descriptors from the distance definition (19.2).

Integration of the WS techniques with the PIPDI will be our main goal in the future. This will provide more complete indexing and also easier access for scholars to the content of the dataset.

A few ideas for the future are as follows. A possible improvement to the system is the recovery of faded BPs. We are working on recovering the possible BPs based on the local geometrical variations on branches. Also, as the number of topological descriptors is very much higher than the geometrical ones, using different weights for topological and geometrical distances may be considered. In fact, one of our future directions will be to increase the number of geometrical features and descriptors for better representation of CCs, without destroying the size-independent nature of the skeletons. As discussed, the casual writing style or unique style of some scripts can result in CCs touching. These can be detected by a pre-processing step, based on the CCs' complexity and bounding box.

In another direction, although, because of the very local nature of the features used, the method is robust with respect to mild skew on the input document images (as seen in Fig. 19.4), skew correction will be added to the pre-processing step in the future.

Acknowledgements The authors would like to thank the NSERC of Canada for their financial support. Also, we would like to acknowledge Dr. Robert Wisnovsky and his team, from IIS, McGill University, for their collaboration and fruitful comments and discussions.

Appendix A: Competence vs. Performance

Pattern recognition and document understanding [4, 35] are actually equivalent to a process in which humans (authors, writers, and copiers of documents and manuscripts) are modeled as a kind of machine. These people usually differ in the writing they produce mainly because of their philological and psychological differences and limitations. In other words, some aspects of these variations are rooted in the inability of such a “machine” to reproduce a well-standardized set of patterns. This inability can be related to motor limitations of hand, mind, or memory, for example. In modern societies, and for short time scales, where a high level of education based on a single “school” can be assumed, this is close to the reality. Performance [5] is a term used to measure the level of this imperfection, and is a concept that has been used widely in many fields, from linguistics to engineering. Performance is a measure of the output of a writing process. It not only reflects the personality of the writer, but also, and mainly, the errors and inabilities of the writer. Most approaches to document understanding, which usually originated from the engineering point of view, overestimate performance. This tradition has resulted in attempts to filter out, “average,” or ignore variations. In earlier times, there was more diversity still, although even in modern societies there are variations in the way individuals think. Every author, writer, and copier has a unique and well-developed

ability to think for himself. Perhaps it would be difficult to gauge the value of the large number of manuscripts produced over a period of a thousand years or so, but, if we look at them on a scale of a few years, they appear as individual gems. The deep philosophical thinking of their authors is reflected in their lives through many stories told about them. It is obvious that their thinking would be reflected in their writing style as well. Moreover, their writing styles should also be as unique as the philosophical schools to which they belong. In other words, a major part of the difference between the writing styles of two manuscripts arises from differences in the thinking of the writers, rather than their inability to replicate a standard style. This brings us to *competence* [5], a concept which has been somewhat ignored in document analysis and understanding. In contrast to performance, competence actually depends on the knowledge of the writer. We believe that a successful approach to analysis and understanding of historical-manuscript images should be based on accepting differences in competence and allowing enough room for variations. This is why our approach is adaptable to any manuscript under study, as it “learns” and organizes the information in the manuscript, and can therefore capture a large proportion of the possible variations. This may, of course, lead to a sharp rise in the resources needed to understand the documents, as well as in the complexity of the models. But, at the same time, if the approach attempts to understand the underlying philosophy behind the writing style, it can easily isolate and locate sources of variations in competence, and can therefore control the complexity of the model.

Appendix B: A Priori Information

In order to use the document properties directly in processing, some parameters, considered as a priori information, are defined and included in the models. The first and most important parameter is the average stroke width, which depends on the writing style and the acquisition setup. The average line height and the minimum and maximum dot sizes are a few other parameters.

As discussed in the previous sections, the key concepts in the proposed framework are characteristic lengths, which are defined based on the range of interactions. These parameters can be extracted using various tools, such as wavelet transform or kernel-based analysis. Although these parameters may vary drastically, even on a single image from one site (paragraph) to another, their behavior is usually very robust and almost constant over a whole dataset. Therefore, many learning and data mining methods can be used to obtain robust values for the characteristic lengths. In this work, we assume that the values of these parameters are known a priori and are constant for each manuscript. Below, a few characteristic lengths are defined.

1. Stroke width

The most important characteristic length on a document image is the *stroke width*. In this work, we use the *average stroke width*, w_s , as a priori information in the form of a constant parameter. It is estimated using a kernel-based algorithm (see Appendix C).

2. Line height

The second most important characteristic length on document images is the shortest distance between text sites, which are usually text lines. We call this parameter the *line height*. By definition, line height is the distance between two adjoining baselines. Again, in this work, only the *average line height*, h_l , is used.

3. Vertical extent of text line

The average vertical extent of text line h_e is defined as the average distance to which the text pixels extend from a text line. It is different from the line height h_l , which is the average distance between two successive baselines.

Appendix C: Markov Clustering

Robust and parameterless clustering of objects is an interesting and at the same time difficult problem. Usually, the similarity measure between objects is not normalized, and therefore threshold-based methods, or the methods that assume that the number of clusters is known *a priori*, are very sensitive to parameters. There are many approaches to parameterless clustering techniques, such as Markov clustering [38] and improved incremental growing neural gas [8]. Markov clustering (MCL) is a robust technique in which the similarity values between like objects is increased, gradually and through a few interactions, while the value for nonsimilar objects decreases. This process eventually leads to zero plus one values in the similarity matrix. Although there are two parameters in this technique, which will be discussed later, their effects on the performance of the clustering are mainly limited to the number of iterations, and the convergence of the process is usually independent of them. It is worth noting that the translation of distances to similarities (for example, relation (19.5)), which is independent of MCL and a common step in all clustering techniques, depends strongly on the nature of the problem under study. This is why the parameters h_{MCL} are selected equal to d_{thr} .

Algorithm 3 provides the details of the MCL process. In each MCL iteration, there are two main operations. The parameter p specifies the intensity of the expansion operation, which is represented by the power operation in the algorithm. The parameter r controls the inflation operation implemented as entry-wise power operations and column renormalization. In this work, $p = 2$ and $r = 1.2$ are used. If we imagine the similarity value between different objects as the capacity of imaginary pipes that connect the objects, the expansion operation actually increases the capacity of high-flow pipes, and reduces the capacity of pipes with little flow, in a gradual and smooth way. The inflation operator preserves the stochastic nature of the process. In the end, there will only be two types of pipes: open pipes with the highest capacity ($W_{ij} = 1$) and blocked pipes ($W_{ij} = 0$). The objects that are connected with open pipes represent a cluster, and the objects with the highest number of connections can be considered as the representer of that set of objects.

Set parameters p and r ;

Get the stochastic Markov matrix, W ;

repeat

 Apply expansion operation: $W = W^P$;

 Apply inflation operation: (a) $W_{ij} = W_{ij}^r$, (b) $W_{ij} = W_{ij} / W_{jj}$;

until steady state is obtained;

Algorithm 3: Operation cycle of MCL technique

References

1. Al-Khatib, W.G., Shahab, S.A., Mahmoud, S.A.: Digital library framework for Arabic manuscripts. In: Shahab, S.A. (ed.) AICCSA'07, Amman, Jordan, May 13–16, 2007, pp. 458–465 (2007)
2. Antonacopoulos, A., Downton, A.: Special issue on the analysis of historical documents. *Int. J. Doc. Anal. Recognit.* **9**(2), 75–77 (2007)
3. Bradley, D., Roth, G.: Adaptive thresholding using the integral image. *J. Graph. Tools* **12**(2), 13–21 (2007)
4. Chang, H.-H., Yan, H.: Analysis of stroke structures of handwritten Chinese characters. *IEEE Trans. Syst. Man Cybern., Part B, Cybern.* **29**(1), 47–61 (1999)
5. Chomsky, N.: Aspects of the Theory of Syntax, 1st edn. MIT Press, Cambridge (1969). ISBN-10: 0262530074
6. Farin, G.: Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, 5th edn. Academic Press, San Diego (2001)
7. Gacek, A.: Arabic manuscripts: a vademecum for readers. In: Handbook of Oriental Studies. Section 1 The Near and Middle East, vol. 98. Brill, Leiden/Boston (2009). ISBN-10: 90 04 17036 7
8. Hamza, H., Belaid, Y., Belaid, A., Chaudhuri, B.B.: Incremental classification of invoice documents. In: ICPR'08, Tampa, FL, USA, December 8–11, 2008, pp. 1–4 (2008)
9. Haralick, R.M., Shapiro, L.G.: Computer and Robot Vision. Addison-Wesley/Longman, Reading (1992)
10. Huang, L., Wan, G., Liu, C.: An improved parallel thinning algorithm. In: ICDAR'03, pp. 780–783 (2003)
11. Joosten, I.: Applications of microanalysis in the cultural heritage field. *Mikrochim. Acta* **161**(3), 295–299 (2008)
12. Kane, S., Lehman, A., Partridge, E.: Indexing George Washington's Handwritten Manuscripts: A study of word matching techniques. CIIR technical report, University of Massachusetts, Amherst (2001)
13. Kohonen, T.: Self Organizing Maps, 3rd edn. Springer, Berlin (2001)
14. Leydier, Y., Le Bourgeois, F., Emptoz, H.: Omnilingual segmentation-free word spotting for ancient manuscripts indexation. In: Le Bourgeois, F. (ed.) ICDAR'05, vol. 1, pp. 533–537 (2005)
15. Leydier, Y., Ouji, A., LeBourgeois, F., Emptoz, H.: Towards an omnilingual word retrieval system for ancient manuscripts. *Pattern Recognit.* **42**, 2089–2105 (2009)
16. Manso, M., Carvalho, M.L.: Application of spectroscopic techniques for the study of paper documents: A survey. *Spectrochim. Acta, Part B, At. Spectrosc.* **64**(6), 482–490 (2009)
17. Matuschek, M., Schlüter, T., Conrad, S.: Measuring text similarity with dynamic time warping. In: Proceedings of the 2008 International Symposium on Database Engineering and Applications, Coimbra, Portugal, pp. 263–267. ACM, New York (2008)

18. Moghaddam, R.F., Cheriet, M.: Application of multi-level classifiers and clustering for automatic word-spotting in historical document images. In: ICDAR'09, Barcelona, Spain, July 26–29, 2009, pp. 511–515 (2009)
19. Moghaddam, R.F., Cheriet, M.: Low quality document image modeling and enhancement. *Int. J. Doc. Anal. Recognit.* **11**(4), 183–201 (2009)
20. Moghaddam, R.F., Cheriet, R.M.: Restoration of single-sided low-quality document images. *Pattern Recognit.* **42**, 3355–3364 (2009)
21. Moghaddam, R.F., Cheriet, M.: A multi-scale framework for adaptive binarization of degraded document images. *Pattern Recognit.* **43**(6), 2186–2198 (2010)
22. Moghaddam, R.F., Cheriet, M., Adankon, M.M., Filonenko, K., Wisnovsky, R.: IBN SINA: a database for research on processing and understanding of Arabic manuscripts images. In: DAS'10, Boston, Massachusetts, pp. 11–18. ACM, New York (2010)
23. Nagy, G.: Twenty years of document image analysis in PAMI. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(1), 38–62 (2000)
24. Nakayama, K., Hasegawa, H., Hernandez, C.A.: Handwritten alphabet and digit character recognition using skeleton pattern mapping with structural constraints. In: Proc. ICANN'93, Amsterdam, September 1993, pp. 941 (1993)
25. Rath, T., Manmatha, R.: Word spotting for historical documents. *Int. J. Doc. Anal. Recognit.* **9**(2), 139–152 (2007)
26. Barni, M., Bernaldin, J.-A., Lahanier, C., Piva, A.: Signal processing in visual cultural heritage, special issue. In: IEEE Signal Processing Magazine, vol. 25(4) (2008)
27. Rodriguez-Serrano, J.A., Perronnin, F., Llados, J., Sanchez, G.: A similarity measure between vector sequences with application to handwritten word image retrieval. In: CVPR'09 (2009)
28. Rothfeder, J.L., Feng, S., Rath, T.M.: Using corner feature correspondences to rank word images by similarity. In: Workshop on Document Image Analysis and Retrieval, Madison, June 20, 2003
29. Saykol, E., Sinop, A.K., Gudukbay, U., Ulusoy, O., Cetin, A.E.: Content-based retrieval of historical ottoman documents stored as textual images. *IEEE Trans. Image Process.* **13**(3), 314–325 (2004)
30. Shafait, F., Keysers, D., Breuel, T.M.: Efficient implementation of local adaptive thresholding techniques using integral images. In: Document Recognition and Retrieval XV, San Jose, CA, January 2008
31. Sharma, O., Mioc, D., Anton, F.: Voronoi diagram based automated skeleton extraction from colour scanned maps. In: ISVD'06, pp. 186–195 (2006)
32. Shih, F.Y., Pu, C.C.: A skeletonization algorithm by maxima tracking on Euclidean distance transform. *Pattern Recognit.* **28**(3), 331–341 (1995)
33. Spitz, A.L.: Using character shape codes for word spotting in document images. In: Dori, D., Bruckstein, A. (eds.) *Shape, Structure and Pattern Recognition*, pp. 382–389. World Scientific, Singapore (1995)
34. Steinherz, T., Intrator, N., Rivlin, E.: A special skeletonization algorithm for cursive words. In: IWFHR'00, pp. 529–534 (2000)
35. Tang, Y.Y., Suen, C.Y., De Yan, C., Cheriet, M.: Financial document processing based on staff line and description language. *IEEE Trans. Syst. Man Cybern.* **25**(5), 738–754 (1995)
36. The Mathworks Inc., Natick, MA. MATLAB Version 7.5.0
37. van der Zant, T., Schomaker, L., Haak, K.: Handwritten-word spotting using biologically inspired features. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**, 1945–1957 (2008)
38. van Dongen, S.: Graph clustering by flow simulation. Ph.D. thesis, Univ. Utrecht, May 2000
39. Vijaya Kumar, V., Srikrishna, A., Ali Shaik, S., Trinath, S.: A new skeletonization method based on connected component approach. *Int. J. Comput. Sci. Netw. Secur.* **8**, 133–137 (2008)

40. Yalniz, I.Z., Altingovde, I.S., Güdükbay, U., Ulusoy, O.: Ottoman archives explorer: A retrieval system for digital Ottoman archives. *J. Comput. Cult. Herit.* **2**(3), 1–20 (2009)
41. Zeng, J., Liu, Z.-Q.: Stroke segmentation of Chinese characters using Markov random fields. In: ICPR'06, vol. 1, pp. 868–871 (2006)
42. Zhu, X.: Shape recognition based on skeleton and support vector machines. In: Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques, vol. 2, pp. 1035–1043 (2007)

Chapter 20

Arabic Text Recognition Using a Script-Independent Methodology: A Unified HMM-Based Approach for Machine-Printed and Handwritten Text

**Premkumar Natarajan, Rohit Prasad, Huagu Cao, Krishna Subramanian,
Shirin Saleem, David Belanger, Shiv Vitaladevuni, Matin Kamali,
and Ehry MacRostie**

Abstract We describe BBN’s script-independent methodology for multilingual machine-print OCR and offline handwriting recognition (HWR) based on the use of hidden Markov models (HMM). The feature extraction, training, and recognition components of the system are all designed to be script-independent. The HMM training and recognition components are based on BBN’s Byblos hidden Markov modeling software. The HMM parameters are estimated automatically from the training data, without the need for laborious manually created rules. The system does not require any pre-segmentation of the data, either at the word level or at the character level. Thus, the system can handle languages with cursive handwritten scripts in a straightforward manner. The script independence of the system is demonstrated with experimental results in three scripts that exhibit significant differences in glyph characteristics: Arabic, Chinese, and English. Experimental results demonstrating the viability of the proposed methodology are presented. Offline HWR of free-flowing Arabic text is a challenging task due to the plethora of factors that contribute to the variability in the data. In light of this book’s focus on Arabic scripts, we address some of these sources of variability, and present experimental results on a large corpus of handwritten documents. Experimental results are provided for specific techniques such as the application of context-dependent HMMs for the cursive Arabic script and unsupervised adaptation to account for the stylistic variations across scribes/writers. We also present an innovative integration of structural features in the HMM framework which results in a 10 % relative improvement in performance. We conclude with a new technique for dealing with noise related to the dots that are an integral yet disconnected part of many Arabic characters.

P. Natarajan (✉) · R. Prasad · H. Cao · K. Subramanian · S. Saleem · D. Belanger ·
S. Vitaladevuni · M. Kamali · E. MacRostie
BBN, Cambridge, MA, USA
e-mail: pnataraj@bbn.com

20.1 Introduction

Commercial off-the-shelf (COTS) optical character recognition (OCR) software can accurately recognize clean machine-printed text with simple layouts. However, the recognition of handwritten text continues to be a challenging research problem due to the various sources of variability in handwritten data. Furthermore, most OCR and offline handwriting recognition (HWR) systems are designed for a particular script or language. Here, we describe a unified approach to OCR and offline HWR that, in principle and by design, is script-independent and can be used for the majority of the world's scripts and languages. The basic modeling paradigm we employ is that of hidden Markov models (HMMs) [1]. The core feature extraction, training, and recognition components are the same for all languages; only the data-specific components, such as the glyph/character set, the optional word lexicon, and the language model, depend on the specific language. Except for the pre-processing and feature extraction components, which are specific to OCR and offline HWR, the training and recognition components are taken without significant modification from our Byblos HMM system [2, 3] which has been applied to the speech recognition task for more than two decades. Hence, we call our text recognition system the BBN Byblos OCR (offline HWR) system.

HMMs are capable of modeling the variability of a feature vector as a function of one independent variable. In speech [2], there is one natural independent variable: time. In text recognition, there are two independent variables since text images are two-dimensional (2D), so 1D HMMs cannot be used directly. We structure the text recognition problem as a combination of two 1D pattern recognition tasks. The first task, also called line finding, is to locate the individual lines of text on a page, and the second task is to recognize the text content of each line.

Even the offline HWR problem at the level of a single line is in truth 2D as well; however, we make it into a 1D problem by extracting a feature vector that is a function of only one dimension (usually horizontal position). The feature vector is extracted from narrow vertical strips along each line of text [4]. The fact that the feature vector we extract does not depend on the script being recognized is one reason that our approach is script-independent. The other reason is that the HMM modeling approach itself does not change with the script being recognized. In particular, the fact that there is no separate character segmentation component, neither in training nor in recognition, allows the same system to recognize scripts where the characters are separate or connected. To demonstrate the script independence of our approach, we present offline HWR results in three different scripts: Arabic, Chinese, and English. Arabic and English present the challenge of dealing with a cursive script, and Chinese presents the challenge of dealing with a large number of characters. There have been a number of research efforts that use HMMs in offline printed and handwriting recognition [5–29]. In all these efforts, the recognition of only a single language or script is attempted. From a methodological perspective, our approach differs from other approaches principally in the emphasis we place on script independence.

In the following, we first briefly review the theoretical framework of the HMM paradigm. Next, we provide a description of the BBN Byblos system. The description here closely follows the related sections in [4] and is summarized here for the convenience of the reader. The reader is referred to [4] for more details. We then present experimental recognition results for Arabic on the IFN/ENIT and the DARPA MADCAT corpus, followed by a description of two enhancements that are targeted at improving the performance of the system on Arabic. We conclude the chapter by highlighting two real-world applications of the text recognition technology.

20.2 Theoretical Framework

20.2.1 Problem Formulation

We represent a line of text within a scanned image as a sequence of feature vectors, X . The aim is to find the character sequence that maximizes $P(C|X)$, the probability of a sequence of characters C given the feature vector sequence X . Using the Bayes rule, $P(C|X)$ may be written as:

$$P(C|X) = P(X|C)P(C)/P(X) \quad (20.1)$$

We call $P(X|C)$ the feature model and $P(C)$ the language model (or grammar). $P(X|C)$ is a model of the feature vector sequence X , given a sequence of characters C , and is approximated as the product of the component probabilities, $P(X_i|c_i)$, where X_i is the sequence of feature vectors that corresponds to character c_i . The feature model for each character is given by a specific HMM.

$P(C)$, the language model, is the prior probability of a sequence of characters, C , and it provides a soft constraint on allowable character sequences. The language model used in the Byblos OCR system is an n -gram Markov model which computes $P(C)$ by multiplying the probabilities of consecutive groups of n characters or words.

$P(X)$ in (20.1) is the a priori probability of the data and does not depend on C ; therefore, we can maximize $P(C|X)$ by maximizing the product $P(X|C)P(C)$.

20.2.2 BBN Byblos System

As mentioned earlier, the Byblos text recognition system is a statistical, HMM-based recognition system that uses the Byblos HMM engine, which was originally developed for speech recognition at BBN. Figure 20.1 shows a block diagram of the system. As indicated in the diagram, the OCR system can be subdivided into two basic functional components: training and recognition. Both training and recognition share a common pre-processing and feature extraction stage. In the rest of this section we describe the training and recognition components in detail, starting with the pre-processing step.

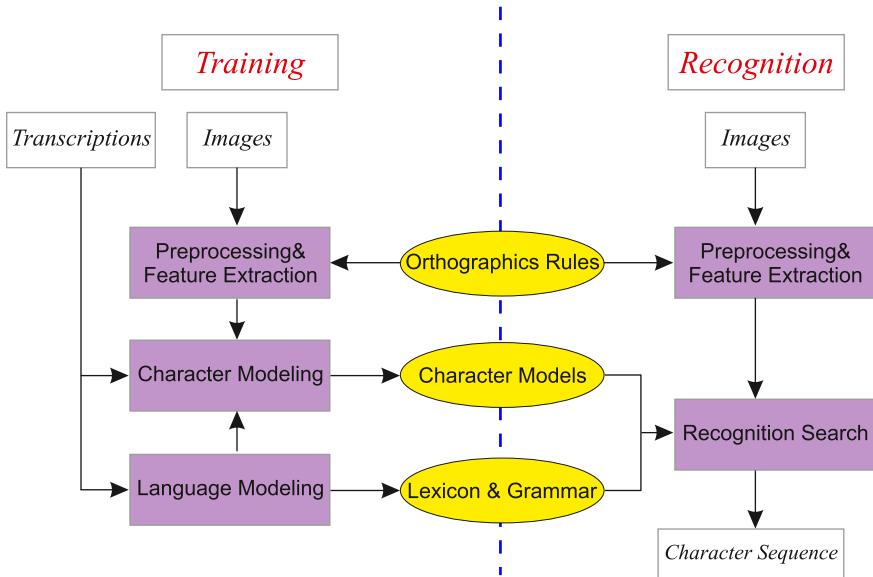


Fig. 20.1 Block diagram of BBN Byblos text recognition system

Pre-processing

Document images typically contain noninformation-bearing variability (often referred to as noise) that is introduced by one or more physical processes, such as scanning, faxing, writing style, presence or absence of ruled lines, crumpling, folding, etc. The goal of pre-processing is to minimize noise before the image is further processed. We note that any particular type of variability could be undesirable in one context, whereas that same variability might provide useful information in another. For example, variations in writing style (slant, overhanging strokes, flourishes, etc.) are undesirable from the perspective of recognizing the content of handwritten pages, whereas those very variations are the target of feature extraction when the task is that of identifying the writer of a particular document.

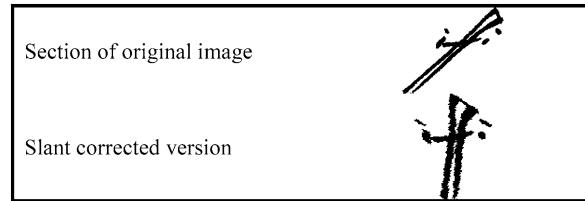
The pre-processing stage in the Byblos OCR system is designed to minimize variations that are undesirable from the perspective of recognizing the text. To that end, we first apply a set of simple filters, such as median filters, that minimize artifacts such as salt-and-pepper noise and stains. Next, we deskew the image using the approach described in [4]. We then apply techniques that detect and remove ruled lines while minimizing any distortions in the shapes of the character glyphs [30, 31].

The final pre-processing operation that we apply is a slant normalization technique to make the vertical strokes within character glyphs perpendicular to the baseline. To normalize the slant, we estimate and then apply a nonlinear, 2D transform to each connected component (CC) within an input (black-white) text image. The estimation of the nonlinear transform is based upon the approach presented in [32]. We apply the slant correction procedure iteratively until the estimated slant is below

Table 20.1 Slant correction example (English)

Original Image	
Iteration 1	
Iteration 2	
Iteration 3	
Iteration 4	

Table 20.2 Jagged perimeter caused by slant correction step



a certain threshold. Slant correction is only applied to handwritten text. Table 20.1 shows an English handwritten word that is slanted to the right and the slant-corrected versions of the word for each of four successive iterations of the correction procedure. While repeated application of the transform progressively reduces the slant of text, a closer inspection of the image indicates that the transform makes the perimeter of the text more jagged. A section of the original word image and the corresponding slant-corrected section after four applications of the nonlinear transform are shown in Table 20.2. Clearly, one area of future work is to improve the slant correction procedure to reduce the manifestation of such jagged perimeters.

Line Finding

After pre-processing, the image is segmented into lines of text. For machine-printed text, the Byblos OCR system uses an HMM-based line finding technique that takes advantage of certain regularities that are characteristic of machine-printed text lines. The HMM-based technique is not well suited to handle the irregular nature of unconstrained handwritten text. More generally, as can be seen from a review of the proceedings of the recently concluded International Conference on Document Analysis and Recognition (ICDAR) 2011, finding text lines in handwritten documents remains a challenging task that continues to attract significant research attention. Under the DARPA MADCAT program, members of the BBN research team have developed several different line finding techniques [33, 34]. Furthermore, in [35], BBN presented a novel graph clustering based ensemble framework for combining the output of several different line finding techniques to significantly improve the accuracy of finding handwritten text lines. The ensemble combination approach uses a weighted undirected graph in which nodes correspond to connected components



Fig. 20.2 Example of line finding output using the ensemble framework

and an edge between two nodes denotes that one of the line finding algorithms in the ensemble put the associated pair of connected components within a single text line. Text line segmentation is then posed as the problem of minimum cost partitioning of the nodes in the graph such that each cluster corresponds to a unique line in the document image. Experimental results on a challenging Arabic dataset (see Fig. 20.2) using the ensemble method shows a relative gain of 18 % in the F1 score over the best individual method within the ensemble.

Both the individual techniques and the combination framework are used in the Byblos offline HWR system. Figure 20.2 shows an example of the text lines produced by the ensemble algorithm. All the text components within a single line are colored with the same color (red, blue, etc.). Despite the presence of some errors (the blue-colored component in the first line should have clearly been colored red), the ensemble technique clearly works well, even on challenging data. While system combination approaches have been used for a long time in speech recognition and in OCR, the work in [35] represents, to the best of our knowledge, the first published attempt at combining the output of multiple line finding algorithms with the goal of producing a single best line finding result.

Script-Independent Feature Extraction

For each line of text, the features are computed from a sequence of overlapped windows. For each window, also called a frame, several features are computed. Figure 20.3 illustrates the process of extracting frames from the sequence of overlapping windows.

The most important features are the percentile features [4] computed from the binary pixels in the window. Blackness is integrated from top to bottom. After dividing by overall blackness, we get a normalized function that rises monotonically from 0 at the top of the window to 1.0 at the bottom. For example, if the feature value corresponding to the 30th percentile is 0.22, it means that 30 % of the blackness in the window occurs at 0.22 of the height of the window, measuring from the top and going down. The percentile features tend to be relatively insensitive to various types of noise.

Once the percentile features are computed, two additional sets of features are computed from the percentile values: the vertical derivatives of percentile features

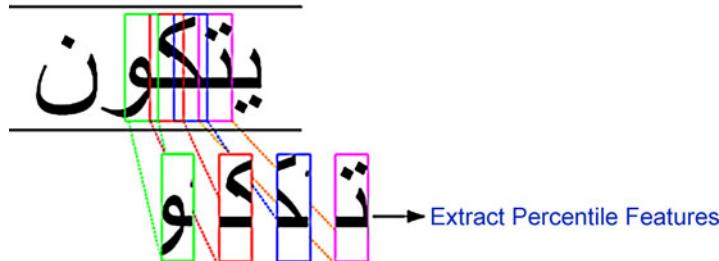


Fig. 20.3 Extracted frames corresponding to overlapping windows within a text line

of a single frame and the horizontal derivatives of percentile features of adjacent frames.

Finally, angle and correlation features are computed from the pixel values. The frame is divided from top to bottom into overlapping blocks, and each block is made square by including material that falls outside the window on both sides. Within a square block, angle and correlation values are computed from a scatter plot of the text pixels, using standard linear regression to find the best linear fit that minimizes the mean-squared error. Basically, the angle feature measures the local slope of the text stroke within the square block, while the correlation feature measures how the text pixels are distributed around the best linear fit.

The feature extraction module thus computes a total of 80 features per frame: 20 percentiles, 20 vertical derivatives, 20 horizontal derivatives, 10 angle features, 10 correlation features, and a single energy feature which is simply the percentage of pixels within the frame that are black. Linear discriminant analysis (LDA) [36] is then used to reduce the number of features per frame. The exact number of LDA features to be used is best selected empirically after running a set of experiments and choosing the number of features that result in the minimum character error rate. The resulting vector of LDA features is a compact numerical representation of the data in the frame, and is the feature vector used in our recognition experiments.

Training

The Byblos system models each character with a multi-state, left-to-right HMM; the model for a word is the concatenation of the models for the characters in the word. Each state has an associated output probability distribution over the features. Each output probability distribution is modeled as a weighted sum of Gaussians, or what is called a Gaussian mixture. A Gaussian mixture is completely parameterized by the means and variances of the component Gaussians, along with the weight of each Gaussian in the mixture. The number of states and the allowable transitions are system parameters that can be set. For our experiments we have used 14-state, left-to-right HMMs.

Training the process of estimating the parameters (transition probabilities and feature probability distributions) of each of the character HMMs is performed using what has been known alternately as the Baum–Welch [37], forward-backward,

or expectation-maximization (EM) algorithm [38, 39], which iteratively aligns the feature vectors with the character models to obtain maximum likelihood estimates of HMM parameters. The algorithm is guaranteed to converge to a local maximum of the likelihood function. The feature probability distributions in our system are characterized by the means, variances, and weights of the Gaussian mixtures.

Depending on the amount of available training data, it may not be possible to get robust estimates of all the HMM parameters for all the characters. That is one reason why we use LDA to reduce the size of our feature vector. Another method that is used to reduce the total number of parameters in the system is to share some of the Gaussians across different character models. We have used three such methods in our Byblos system: the tied mixture (TM) mode [40, 41], the character tied mixture (CTM) mode [42], and the state tied mixture (STM) mode (described later in this chapter). In the TM mode we train only one set of Gaussians (referred to as a codebook of Gaussians or just a codebook) and the codebook is shared between all the states of all character models; in the TM configuration. In the TM configuration, the individuality of each state output probability distribution is characterized solely by the specific component mixture weights. In the CTM mode we train one codebook of Gaussians for each character model; the Gaussians in a character codebook are thus shared among the states of the model for that character, but there is no sharing across characters. The CTM mode offers a greater number of free parameters than the TM mode and with it the possibility of better performance, subject to the availability of sufficient data for appropriately training all the parameters. Our discussion thus far has implicitly assumed that the glyph shape associated with each character is modeled by a single HMM. But in the case of cursive scripts such as Arabic as well as in the case of handwritten text in general, the shape of the glyph associated with a particular character can often change as a function of the context in which it appears. In such cases, it is often desirable to use a separate HMM to model the glyph shape associated with each salient context, where context is typically defined by the characters that precede and follow the character of interest. The STM configuration uses a shared set of Gaussians for each numbered state of all the context-dependent HMMs associated with a particular character.

The training process is performed as follows. We assume that for each line of text we are given the corresponding ground truth, which is simply the sequence of characters on that line. No information is given about the location of each character on the line; that is, no pre-segmentation is necessary. The training algorithm automatically and iteratively aligns the sequence of feature vectors along the line of text with the sequence of character models.

Recognition

After pre-processing a line of text and performing feature extraction, as described above, the recognition process consists in a search for the sequence of character models that has the highest probability of having generated the observed sequence of feature vectors, given the trained character models, a possible word lexicon, and

a statistical language model (typically an n -gram language model) of the possible character or word sequences. The recognition search is a two-pass [43, 44] (a forward pass and a backward pass) beam search for the most likely sequence of characters. The width of the search beam is a system parameter that can be set. Typically, lowering the beam width increases the speed but degrades the accuracy of recognition. The forward pass is an approximate but efficient procedure for generating a small list of character sequences that are possible candidates for being the most likely sequence. The Byblos HMM system uses an approximate bigram language model, i.e., an n -gram language model, where $n = 2$, in the forward pass.

The backward pass is a more detailed search for the most likely character sequence within this small list. Even though we typically use the same set of character HMMs in the forward and backward passes, the search program itself does not impose any such constraint; i.e., if needed we can use two different sets of character HMMs, one in the forward and the other in the backward pass. When required, larger, more computationally complex HMMs are used in the backward pass for greater discriminative power. The backward pass can use either a bigram or a trigram ($n = 3$) language model, but typically a trigram language model is employed.

The use of a word lexicon (vocabulary) during recognition is optional; its use generally results in a lower error rate when the out-of-vocabulary (OOV) rate is low or close to zero. The term out-of-vocabulary refers to words that are present in the test set but are not included in the lexicon. The lexicon itself is estimated from a suitably large text corpus, and the language model, which provides the probability of any character or word sequence, is also estimated from the same corpus. Note that the text corpus used for estimating the language model (and, indeed, the lexicon) need not be limited to the manual transcriptions associated with the images in the training set, because only the sequence of words in the text is needed to estimate the language model probabilities. The only caveat is that the transcriptions associated with the images in the test set not be used for estimating the lexicon or language model.

In the next section we present the results of some experiments that exercise the Byblos offline HWR system on English and Arabic handwritten text datasets.

20.3 Experimental Results

To demonstrate the script independence of our recognition methodology, we report two sets of experimental results, one for English and one for Arabic. The English experiments were all performed during 2006–2007 using the IAM database [45]. Arabic HWR experiments were performed using the IFN/ENIT database [46] as well as the DARPA MADCAT dataset. While the DARPA MADCAT dataset is the corpus used in ongoing active research at BBN, we present our previous results on the IFN/ENIT in order to provide the reader with performance results on an openly available dataset.

20.3.1 English HWR

As mentioned above, we performed our English HWR experiments using data from the IAM database. A total of 1539 images from 657 different writers, scanned at a resolution of 300 dpi, was used in our experiments. We split the IAM corpus into three sets: a training set, a development set, and a held-out test set. In dividing the corpus into the three sets, we ensured that no writer appears both in test and in training (i.e., a writer-independent test condition). Further, in order to ensure that the language models are fair, all handwritten samples of a form are either assigned entirely to a single set (training, dev, or test). In other words, a particular passage of text is either in training or in (dev) test but never in both; a particular writer is also either in training or in (dev) test but never in both.

Each individual English character is modeled by a 14-state HMM. We experimented with context-dependent as well as context-independent models. For any given character, context is fully defined by the identities of the characters to its left and right. For example, in the word cat, the character a is said to be in the context of c and t, whereas in the word halibut, the same character a is in the context of an h and an l.

We experimented with both character and word trigram language models (LMs) which were trained on transcriptions from the IAM training dataset. For our experiments using word-based LMs, we used a 10K word vocabulary derived from the IAM training transcriptions alone. The OOV rate on the test set for the 10K IAM vocabulary was 10.01 %.

We ran five separate English HWR experiments. The results of the five experiments are summarized in Table 20.3. In the first experiment we trained a set of context-independent character HMMs which were tested against the IAM handwritten test data and yielded a word error rate (WER) of 68.50 %, with an associated character error rate (CER) of 40.10 %. We then performed a second experiment in which we replaced the character trigram LM with a word trigram LM. With a word LM, the WER dropped to 52.80 % and the associated CER dropped to 33.80 %.

Often, in handwritten texts, the appearance context of a character modifies the shape of an associated glyph. Such modifications can also vary from one writer to another. Therefore, we ran a third experiment in which we trained context-dependent character HMMs wherein a separate HMM is used to model each contextually distinct instance of a character. Contexts that occur infrequently in the training data are clustered together with other similar contexts. Using context-dependent HMMs and a word trigram LM, we obtained a WER of 49.30 % and a CER of 31.00 %.

In our fourth experiment, we used a single set of Gaussians for each numbered state of all the context-dependent HMMs associated with a particular character. For example, we estimated a set of 128 Gaussians for the first state of all the contextual variants associated with the character a, and a separate set of 128 Gaussians for the second state of all HMMs for a, and so forth. This configuration, referred to as the state tied mixture (STM) configuration, provides a better model for the structural evolution of a character glyph in the direction of writing. With the STM model, we obtained a WER of 46.1 % and a CER of 28.1 %.

Table 20.3 Summary of English offline HWR performance on IAM database

Glyph HMM Configuration	Language Model Configuration	Word Error Rate (WER)
Context-independent, CTM	Character trigram	68.5 %
Context-independent, CTM	Word trigram	52.8 %
Context-dependent, CTM	Word trigram	49.3 %
Context-dependent, STM	Word trigram	46.1 %
Context-dependent, STM, Slant Correction	Word trigram	40.1 %

After exercising the various modeling capabilities of the Byblos system, we ran a final experiment in which we repeated the fourth experiment using slant-corrected versions of the training and test images instead of the raw versions from the database. Testing on the slant-corrected test images using context-dependent HMMs in a 128 Gaussian STM configuration, we obtained a WER of 40.1 % and a CER of 23.3 %.

In 2004, Vinciarelli [24] reported a WER of approximately 57 % on this database using an HMM-based approach with statistical LMs. The 40.1 % WER reported here was obtained with the Byblos system as it existed in 2006–2007 using a 10K word lexicon and standard maximum likelihood models. More recently, Dreuw [47] reported a WER of 38.9 % using a 50K word lexicon with maximum likelihood models and a WER of 29.2 % with discriminatively trained models. It is not possible to directly compare our results with those in [24] or [47] because of unknown differences in the training and test set. While several advances have been incorporated into the Byblos text recognition system since the English experiments described here were performed, we have included these early results here in order to demonstrate the script independence of the described methodology as well as the robustness of the basic Byblos text recognition system.

20.3.2 Arabic HWR Experimental Results on IFN/ENIT Dataset

Our initial set of Arabic HWR experiments was performed on the IFN/ENIT dataset [46] from 2006. We ran multiple recognition experiments on the held-out (fair) development test set to determine the best configuration with which to decode the test set, set d.

The first recognition experiment used 14-state context-independent character HMMs and a character trigram LM, both trained on the IFN/ENIT corpus. For training the character HMMs, we used a CTM configuration with a separate codebook of 256 Gaussians for each character in the training set. The recognition accuracy on the city names in the development set was 40.7 % with this configuration.

In our second experiment we replaced the character trigram LM with a compound-word LM by stringing the words constituting a city name into a single,

distinct token. With the compound word LM, we obtained a recognition accuracy of 88.2 %, almost a factor of 2 better than that obtained using a character trigram LM.

Next, we performed a third experiment in which we trained context-dependent character HMMs. Using context-dependent HMMs and the compound-word LM, we obtained a recognition accuracy of 88.6 %, a 0.4 % absolute improvement over using context-independent character HMMs. The improvement in recognition accuracy with context-dependent HMMs over context-independent HMMs is significantly smaller than the improvement obtained on English data. We believe this is due to the fact that for Arabic both our OCR and HWR systems use the contextual form of characters to train character HMMs. The contextual form in Arabic encodes canonical changes in the shape of a character glyph due to the neighboring characters; therefore, it is not surprising that context-dependent HMMs yield only a small improvement.

In our fourth Arabic HWR experiment, we applied slant correction to the training data and retrained the context-dependent character HMMs. Testing on the slant-corrected test images using context-dependent HMMs and the compound-word LM resulted in a recognition accuracy of 89.0 % on city names in the development set.

We then trained context-dependent character HMMs with an STM configuration. We used a separate set of 128 Gaussians to model the output distribution at each state for all contexts of a particular character. The recognition accuracy on city names in the development set improved to 89.4 %.

In our sixth and final experiment, we used the recognition results from the previous experiment to perform unsupervised adaptation of the character HMMs for each writer. Maximum likelihood linear regression (MLLR) [48] with a maximum of 8 transformations was applied to adapt only the means of the Gaussians associated with each character HMM. The adapted models resulted in a recognition accuracy of 89.8 %, a 0.4 % absolute improvement over un-adapted recognition.

Having exercised the various modeling capabilities of our system on the development set, we tested the best models against the images in test set, set d. The recognition accuracy on set d with unsupervised writer-adapted context-dependent STM character HMMs trained on slant-corrected images and compound-word LM was 89.4 %.

Once again, the Arabic offline HWR experiments reported here were performed in 2006 using the then current IFN/ENIT dataset. Therefore, we compare the performance of the Byblos HMM system with the performance of other systems in the ICDAR 2005 evaluation. On set d, the Byblos system outperformed the best reported result in the ICDAR 2005 Arabic handwriting competition [28, 29]. To assess the performance of the different model configurations on set d, we repeated all the recognition experiments on set d. The recognition accuracy with all six configurations on development and test sets is summarized in Table 20.4. While our standard practice is to report WERs, we have reported accuracies on the IFN/ENIT corpus because that is the metric used in the ICDAR evaluations with this corpus.

Table 20.4 Accuracy rates on the Arabic IFN/ENIT corpus

Glyph HMM Configuration	Language Model	Accuracy—%	
		Dev Set	Set d
Context-independent, CTM	Character	40.7	41.1
Context-independent, CTM	Compound word	88.2	87.1
Context-dependent, CTM	Compound word	88.6	88.2
Context-dependent, CTM, Slant correction	Compound word	89.0	88.8
Context-dependent, STM, Slant correction	Compound word	89.4	89.0
Context-dependent, STM, Slant correction, unsupervised writer adaptation	Compound word	89.8	89.4

Table 20.5 Division of MADCAT corpus into training, development, and test sets

Set	Number of documents
Training	37,608
Development	868
Test	885
Written by writers in training	430
Written by writers not in training	455

20.3.3 Arabic HWR: Experiments with DARPA MADCAT Dataset

The DARPA MADCAT corpus consists of scanned images of handwritten versions of Arabic newswire articles, web log posts, and newsgroup posts. The scribes/writers were chosen from different demographic backgrounds. Varied writing conditions such as the type of writing instrument (pen or pencil), type of paper (ruled or unruled), and writing speed (careful, normal, and fast) were introduced into the collection process. The handwritten documents were scanned at a resolution of 600 dpi using a high quality scanner.

The dataset used in our experiments comprises 39.5K Arabic handwritten documents with an average of 20 lines and 100 words per document. The associated ground truth annotations include the coordinates of text lines and the corresponding tokenized transcriptions. We divided the dataset into several subsets for training, tuning, and testing purposes as described in Table 20.5. The development set is used to optimize the weights for combining the scores from the glyph models and LM. Consistent with standard practice in the speech recognition community, the optimized weights are applied unchanged to the fair test set.

Each character glyph is modeled by a 14-state state-tied HMM. Several features, including the gradient, concavity, percentile, angle, and correlation features, are computed and transformed using the LDA transformation technique. The LDA matrix is itself estimated during the training process. The output probability distribution of each state is modeled using a Gaussian mixture with 1500 Gaussians.

Table 20.6 Word error rates on the DARPA MADCAT dataset

HMM Configuration	Word Error Rate (WER)
WI Models	27.1 %
WA Models	25.9 %
WA Models + Unsupervised Adaptation	25.2 %

We use a trigram LM trained on an Arabic text collection of over 217 million words based on a vocabulary of 120 thousand words.

Using the configuration described above, we performed writer-independent (WI) modeling, writer-adapted (WA) modeling, and unsupervised adaptation combined with writer-clustered modeling. WI modeling implies that a single set of HMMs are trained on all available training data and applied to all writers in the test data. In WA modeling, the parameters of the WI HMMs are adapted (i.e., re-estimated) via the MAP adaptation technique using available supervised writer annotations. WA modeling produces several sets of HMMs, each one adapted using data from a single writer in the training set. A modification of the approach would be to cluster several writers into a single equivalence class. Such a clustering approach is appropriate when the amount of data from a single writer is inadequate to robustly adapt the WI model parameters. At runtime, writer classification techniques are applied to select the best set of WA HMMs to be applied to each document in the test set. The WERs on the fair test set for the three experiments are shown in Table 20.6.

As shown in Table 20.6, the WA models deliver an improvement of 1.2 % absolute (4.4 % relative) over the WI models. Unsupervised adaptation provides an additional 0.7 % absolute (2.7 % relative) reduction in the WER.

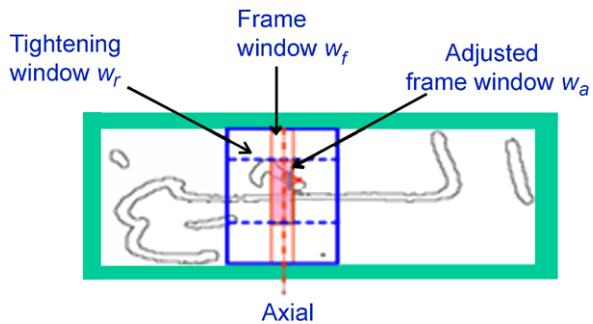
20.4 Improved Glyph Modeling for Arabic

We now describe two modifications of our Byblos text recognition system that were designed and incorporated with the goal of improving recognition performance on Arabic. The first modification extends the feature set by including structural features that include the Gradient-Structure-Concavity (GSC) features [49, 50]. The use of these features allows the system to more effectively capture long-term structural context, thereby improving the discriminative power of the features. The second modification is focused on mitigating the impact of the dots that are an integral but disconnected part of Arabic characters.

20.4.1 Structural Features

In Arabic script, many characters share common primary shapes and differ only in the number and position of the dots and strokes. Therefore, features

Fig. 20.4 Frame tightening for robust feature computation



that capture structural properties such as loops, branch points, endpoints, and dots can be effective in modeling these variations. One such family of features is the GSC set of features. GSC features are symbolic, multi-resolution features that combine three different attributes of the shape of a character: the gradient representing the local orientation of strokes; structural features, which extend the gradient to longer distances and provide information about stroke trajectories; and concavity, which captures stroke relationships at long distances.

For computing the GSC features, first a gradient map is constructed from the normalized image by estimating gradient value and direction at each pixel. Next, gradient features are obtained by counting the pixels which have a similar gradient. The structure features enumerate complex patterns of the contour. To compute the concavity features, pixels which lie in certain special regions such as holes and strokes are detected. The image is then divided into bins, and the number of such pixels in each bin is counted.

The width of the sliding window used to compute the GSC features is wider than that used for the percentile features. Furthermore, since the baseline of a word image may fluctuate within portions of the same word, we algorithmically tighten the upper and lower boundaries of the sliding window. Figure 20.4 illustrates the frame tightening procedure.

We set the width of the frame, f to a desired value, w_f . A tightened window w_t is obtained by first expanding frame f to both the left and right sides (i.e., increasing w_f) and then adjusting the upper and lower boundaries of the frame, f , by considering the spread of black pixels within the expanded frame. The GSC features are computed from the adjusted frame window. The tightened window is divided evenly into overlapping vertical bins, and four sets of GSC features are computed for each bin. The GSC features computed for each frame are then projected to a smaller number of dimensions using LDA in a manner similar to the way it is applied to the percentile features. Measured on the DARPA MADCAT dataset, the GSC features result in approximately 10 % relative reduction in WER over the standard percentile features.

Table 20.7 Effect of the presence and absence of dots on the meaning of a word

Words with Characters Differing in Dots	Meaning of the Word
سم / شم	smell/poison
علی / غلی	boil/on

20.4.2 Dotless Glyph Modeling

For several Arabic characters, the only discriminating information, when in initial or medial position, is the location and number of dots. These dots are an integral part of the characters and are NOT considered diacritics. Diacritics, which are often visually similar to dots, are marks above or below the characters, which are used to indicate short vowels or consonant doubling, for the most part. Diacritics are rarely used, whether in print or in handwriting. When they are used, it is usually for disambiguation. The problem of figuring out where the dots are and how many there are, is a serious one for Arabic script. For example, there are five Arabic letters (b, n, y, t, and unvoiced th) where the only difference between them, when in the initial or medial position, is the location and number of dots! Table 20.7 illustrates the difference in meaning of the words, when the characters differ only in terms of dots.

The LM used in our HMM-based recognition system plays a significant role in disambiguating these cases.

However, disambiguating characters that differ only in position or number of dots becomes extremely challenging in real-world handwritten data because writers are often inconsistent in how and where they write dot(s). These inconsistencies not only lead to poor accuracy in recognizing the characters, but also affect the accuracy in recognizing the primary shape. Therefore, to improve the accuracy in recognizing the primary shape of the characters, we trained glyph HMMs with small connected components removed from the image. Glyphs are mapped to equivalence classes that differ only in dots, thereby reducing the total number of glyphs. Figure 20.5 compares the dot-stripped image with the original image.

In performing a preliminary assessment of the dotless modeling approach, we used the glyph HMMs trained with dots removed to compute a line-level score for each hypothesis in the N-best list that is generated using glyph models trained on the original images (which include the dots). The additional score from the dotless models is combined with other scores such as the HMM scores from the original images and LM scores to re-rank the N-best list. On the DARPA MADCAT Arabic corpus, N-best rescoring with this additional knowledge source results in a 5 % relative improvement in WER.

Fig. 20.5 Comparison of a text line image with and without dots

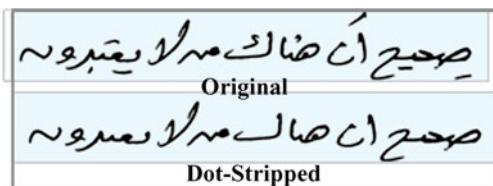


Fig. 20.6 Screenshot of BBN MDATS system for Chinese

20.5 Applications

The BBN Byblos text recognition system has been integrated in several real-world applications. Here, we describe two such applications that demonstrate the versatility of the technology.

BBN Multilingual Document Analysis and Translation System (MDATS): The MDATS system [51] integrates BBN's text recognition technology along with its named-entity extraction technology and commercially available machine translation software in order to transcribe and index large archives of scanned document images in several different languages. MDATS incorporates a browser-based user interface; Fig. 20.6 shows a screenshot of the MDATS home page. As shown in Fig. 20.6, the system links the physical location of the text with its electronic transcription. By clicking on the English button, the user can view the English transla-

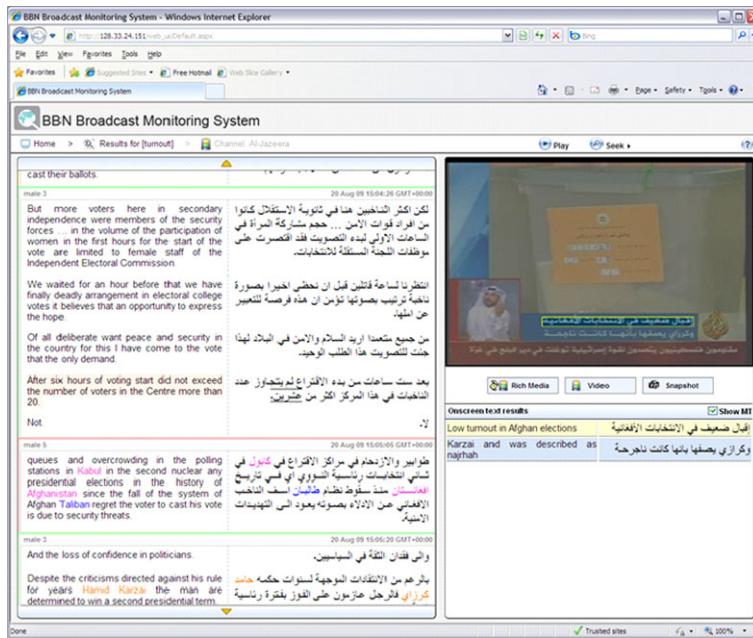


Fig. 20.7 BBN Broadcast Monitoring System with Arabic videotext recognition capability

tion of the source language transcriptions. The system currently supports 10 different languages, including Arabic script languages.

Real-time Multimedia Broadcast Monitoring System: BBN's Broadcast Monitoring System (BMS) is a turnkey system that provides real-time transcription, translation, and indexing of live, streaming broadcast videos using BBN's automatic speech recognition (ASR) technology to recognize the spoken content in the videos. The BMS also integrates BBN's text recognition system in order to recognize any onscreen text in the videos.

Figure 20.7 shows a screenshot of the live BMS system with videotext recognition capability. The left pane of the applications screen shows the real-time transcription of the audio into source language text and its subsequent translation into English. Highlighted words indicate named entities that were automatically detected using BBN's named-entity extraction technology. On the top right portion of the applications screen, the live streaming video is displayed and detected text boxes are overlaid on the video itself. On the lower right-hand side, the output of the text recognition is displayed along with a machine-produced translation into English.

20.6 Summary

BBN's Byblos text recognition system is a state-of-the-art, HMM-based, script-independent text recognition methodology which has been applied to text image

data from several different languages and domains. Ongoing research work at BBN is focused on improving each step in the processing pipeline, from pre-processing to glyph modeling to language modeling and recognition.

While the core technology continues to be advanced through ongoing research efforts, the current system is used in several important applications. BBN has coupled the text recognition technology described here with named-entity extraction and statistical machine translation technology to develop the BBN Multilingual Document Analysis and Translation System (BBN MDATS) [51], a turnkey capability for indexing and searching large archives of document images containing text in several different languages. The Byblos text recognition system has also been successfully applied to the task of recognizing text in videos, a task that is referred to as videotext OCR. BBN's videotext OCR capability is used in combination with its automatic speech recognition engine to index and search multimedia archives.

References

1. Rabiner, L.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **77**, 257–286 (1989)
2. Nguyen, L., Anastasakos, T., Kubala, F., LaPre, C., Makhoul, J., Schwartz, R., Yuan, N., Zavaliagkos, G., Zhao, Y.: The 1994 BBN/BYBLOS speech recognition system. In: *Proc. ARPA Spoken Language Systems Technology Workshop*, Austin, TX, pp. 77–81. Morgan Kaufmann, San Mateo (1995)
3. Makhoul, J., Schwartz, R., LaPre, C., Bazzi, I.: A script-independent methodology for optical character recognition. *Pattern Recognit.* **31**(9), 1285–1294 (1998)
4. Natarajan, P., Lu, Z., Bazzi, I., Schwartz, R., Makhoul, J.: Multilingual machine printed OCR. *Int. J. Pattern Recognit. Artif. Intell.* **15**(1), 43–63 (2001)
5. Kundu, A., Bahl, P.: Recognition of handwritten script: a hidden Markov model based approach. *Pattern Recognit.* **22**, 283–297 (1989)
6. Levin, E., Pieraccini, R.: Dynamic planar warping for optical character recognition. In: *IEEE Int. Conf. Acoustics, Speech, Signal Processing*, San Francisco, CA, vol. III, pp. 149–152 (1992)
7. Vlontzos, J.A., Kung, S.Y.: Hidden Markov models for character recognition. *IEEE Trans. Image Process.* **1**, 539–543 (1992)
8. Agazzi, O.E., Kuo, S.: Hidden Markov model based optical character recognition in the presence of deterministic transformations. *Pattern Recognit.* **26**, 1813–1826 (1993)
9. Chen, M.-Y., Kundu, A., Srihari, S.N.: Handwritten word recognition using continuous density variable duration hidden Markov model. In: *Int. Conf. Acoustics, Speech, Signal Processing*, Minneapolis, MN, vol. 5, pp. 105–108 (1993)
10. Bose, C.B., Kuo, S.-S.: Connected and degraded text recognition using hidden Markov model. *Pattern Recognit.* **27**, 1345–1363 (1994)
11. Rocha, J., Pavlidis, T.: Character recognition without segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **17**, 903–909 (1995)
12. Bunke, H., Roth, M., Schukat-Talamazzini, E.G.: Off-line cursive handwriting recognition using hidden Markov models. *Pattern Recognit.* **28**, 1399–1413 (1995)
13. Oh, C., Kim, W.S.: Off-line recognition of handwritten Korean and alphanumeric characters using hidden Markov models. In: *Proc. Int. Conf. Document Analysis and Recognition*, Montreal, Canada, vol. 2, pp. 815–818 (1995)
14. Anigbogu, J.C., Belaid, A.: Hidden Markov models in text recognition. *Int. J. Pattern Recognit. Artif. Intell.* **9**, 925–958 (1995)

15. Casey, R.G., Lecolinet, E.: A survey of methods and strategies in character segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **18**, 690–706 (1996)
16. Allam, M.: Segmentation versus segmentation-free for recognizing Arabic text. *Proc. SPIE* **2422**, 228–235 (1995)
17. Ben Amara, N., Belaid, A.: Printed PAW recognition based on planar hidden Markov models. In: 13th Int. Conf. Pattern Recognition, Vienna, Austria, vol. II, pp. 220–224 (1996)
18. Kaltenmeier, A., Caesar, T., Gloger, J.M., Mandler, E.: Sophisticated topology of hidden Markov models for cursive script recognition. In: Proc. Int. Conf. Document Analysis and Recognition, Tsukuba City, Japan, pp. 139–142 (1993)
19. Mohamed, M., Gader, P.: Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. *IEEE Trans. Pattern Anal. Mach. Intell.* **18**, 548–554 (1996)
20. Kornai, A.: Experimental HMM-based postal OCR system. In: Proc. Int. Conf. Acoustics, Speech, Signal Processing, Munich, Germany, vol. 4, pp. 3177–3180 (1997)
21. Al-Badr, B., Mahmoud, S.: Survey and bibliography of Arabic optical text recognition. *Signal Process.* **41**, 49–77 (1995)
22. Starner, T., Makhoul, J., Schwartz, R., Chou, G.: On-line cursive handwriting recognition using speech recognition methods In: IEEE Int. Conf. Acoustics, Speech, Signal Processing, Adelaide, Australia, vol. V, pp. 125–128 (1994)
23. Park, H., Lee, S.: A truly 2-D hidden Markov model. *Pattern Recognit.* **31**(12), 1849–1864 (1998)
24. Vinciarelli, A., Bengio, S., Bunke, H.: Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(6), 709–720 (2004)
25. Liu, C.-L., Marukawa, K.: Global shape normalization for handwritten Chinese character recognition: a new method. In: Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9) (2004)
26. Wu, T., Ma, S.: Feature extraction by hierarchical overlapped elastic meshing for handwritten Chinese character recognition. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR) (2003)
27. Tang, Y.Y., Tu, L., Liu, J., Lee, S., Lin, W., Shyu, I.: Offline recognition of Chinese handwriting by multifeature and multilevel classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **20**(5) (1998)
28. Märgner, V., Pechwitz, M., ElAbed, H.: ICDAR 2005 Arabic handwriting recognition competition. In: Proc. Int'l. Conf. Document Analysis and Recognition, pp. 1274–1278 (2007)
29. Märgner, V., Pechwitz, M., ElAbed, H.: Arabic handwriting recognition competition. In: Proc. Int'l. Conf. Document Analysis and Recognition, pp. 70–74 (2005)
30. Cao, H., Prasad, R., Natarajan, P.: A stroke regeneration method for cleaning rule-lines in handwritten document images. In: Proceedings of the International Workshop on Multilingual OCR (MOCR) (2009)
31. Cao, H., Prasad, R., Natarajan, P., Govindaraju, V.: Nested state indexing in pairwise Markov networks for fast handwritten document image rule-line removal. In: Proceedings of the 16th IEEE International Conference on Image Processing (ICIP) (2009)
32. Tay, Y.H., Lallican, P.M., Khalid, M., Viard-Gaudin, C., Knerr, S.: An offline cursive handwritten word recognition system. In: Proceedings of IEEE Region 10 Conference (2001)
33. Peng, X., Setlur, S., Govindaraju, V., Ramachandrula, S.: Markov random field based text identification from annotated machine printed documents. In: International Conference on Document Analysis and Recognition, Barcelona, Spain, pp. 431–435 (2009)
34. Kumar, J., Kang, L., Doermann, D., Abd-Almageed, W.: Segmentation of handwritten textlines in presence of touching components. In: Intl. Conf. on Document Analysis and Recognition (ICDAR 11), pp. 109–113 (2011)
35. Manohar, V., Vitaladevuni, S.N., Cao, H., Prasad, R., Natarajan, P.: Graph clustering-based ensemble method for handwritten text line segmentation. In: Proc. Int'l. Conf. Document Analysis and Recognition, pp. 574–578 (2011)

36. Fukunaga, K.: *Introduction to Statistical Pattern Recognition*, 2nd edn. Academic Press, New York (1990). Chapter 10
37. Baum, L.E.: An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities* **3**, 1–8 (1972)
38. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum-likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc., Ser. B, Stat. Methodol.* **39**, 1–38 (1977)
39. Redner, R.A., Walker, H.F.: Mixture densities, maximum likelihood and the EM algorithm. *SIAM Rev.* **26**, 195–239 (1984)
40. Huang, X.D., Jack, M.A.: Semi-continuous hidden Markov models for speech recognition. *Comput. Speech Lang.* **3** (1989)
41. Bellegarda, J., Nahamoo, D.: Tied mixture continuous parameter models for large vocabulary isolated speech recognition. In: *IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Glasgow, Scotland, vol. 1, pp. 13–16 (1989)
42. Lu, Z., Schwartz, R., Natarajan, P., Bazzi, I., Makhoul, J.: Advances in the BBN BYBLOS OCR system. In: *Proc. of ICDAR*, Bangalore, India, pp. 337–340 (1999)
43. Austin, S., Schwartz, R., Placeway, P.: The forward–backward search algorithm. In: *IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Toronto, Canada, vol. V, pp. 697–700 (1991)
44. Schwartz, R., Nguyen, L., Makhoul, J.: Multiple-pass search strategies. In: Lee, C.-H., Soong, F.K., Paliwal, K.K. (eds.) *Automatic Speech and Speaker Recognition: Advanced Topics*, pp. 429–456. Kluwer Academic, Dordrecht (1996)
45. Marti, U.V., Bunke, H.: The IAM database: an English sentence database for offline handwriting recognition. *Int. J. Doc. Anal. Recognit.* **5**(1), 39–46 (2002)
46. Pechwitz, M., Snoussi, S.M., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT-database of handwritten Arabic words. In: *7th Colloque International Francophone sur l'Ecrit et le Document*, CIFED, Hammamet, Tunis, Oct. 21–23, 2002
47. Drewu, P., Heigold, G., Ney, H.: Confidence and margin-based MMI/MPE discriminative training for offline handwriting recognition. *Int. J. Doc. Anal. Recognit.* **14**(3), 273–288 (2011)
48. Legetter, C.J., Woodland, P.C.: Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. In: *Computer Speech and Language*, vol. 9, pp. 171–185 (1995)
49. Favata, J.T., Srikanth, G.: A multiple feature/resolution approach to handprinted digit and character recognition. *Int. J. Imaging Syst. Technol.* (1996)
50. Tulyakov, S., Govindaraju, V.: Probabilistic model for segmentation based word recognition with lexicon. In: *International Conference on Document Analysis and Recognition* (2001)
51. MacRostie, E., Prasad, R., Rawls, S., Kamali, M., Cao, H., Subramanian, K., Natarajan, P.: The BBN document analysis service: a platform for multilingual document translation. In: *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems* (2010)

Chapter 21

Arabic Handwriting Recognition Using VDHMM and Over-segmentation

Amlan Kundu and Tom Hines

Abstract This chapter describes a complete system for the recognition of unconstrained handwritten Arabic words using over-segmentation of characters and a variable duration hidden Markov model (VDHMM). First, a segmentation algorithm based on morphology and linguistic information is used to translate the 2D image into a 1D sequence of subcharacter symbols. This sequence of symbols is modeled by one single contextual VDHMM. Generally, there are two information sources associated with the written text: shape information and linguistic information. Forty-five features are selected to represent the shape information of character and subcharacter symbols in the feature space. The shape information of each character symbol, i.e., a feature vector, is modeled as an independently distributed multivariate discrete distribution or a joint continuous distribution. Linguistic knowledge about character transition is modeled as a Markov chain, where each character in the alphabet is a state and bigram probabilities are the state transition probabilities. In this context, the variable duration state is used to take care of the segmentation ambiguity among the consecutive characters. We outline the substantial effort that has been expended to create a corpus of handwritten Arabic words and characters extracted from these handwritten words. Using this corpus and the IFN dataset 2003, detailed experimental results are described to demonstrate the success of the proposed scheme.

21.1 Introduction

The Arabic script is used not only by Arabic-speaking people, but also by speakers of other languages: Farsi (Iran), Dari, Pushtu (Afghanistan), Kurdish (Turkey and Iraq) and Urdu (Pakistan). The Arabic language is written from right to left. It has 28 characters, but each character can take up to four forms: isolated, initial, middle,

A. Kundu (✉) · T. Hines
MITRE, 7515 Colshire Drive, McLean, VA 22102, USA
e-mail: akundu@mitre.org

T. Hines
e-mail: tomhines@mitre.org

and final. There are short vowels, but they are often omitted in the printed form. Arabic script presents challenges because all orthography is cursive and letter shape is context sensitive. Summarizing,

1. Arabic text (printed or handwritten) is cursive and, in general, from right to left. Arabic letters are normally connected to each other on the baseline.
2. Arabic writing uses letters (which consist of 28 basic letters), ten numerals, punctuation marks, as well as spaces and special symbols.

Arabic is often called an affixal language [14]. A word can often be broken into four parts: (1) prefix, (2) suffix, (3) root, and (4) infix. One root can give rise to 80 words on average by proper combination of prefix, suffix, and infix, and there are many rules for prefix, suffix, and infix combination [14]. This linguistic information is yet to be incorporated into Arabic optical character recognition (OCR) [2, 5, 6, 8], but the importance is gradually becoming recognized.

Before describing the handwriting recognition (HWR) activities related to Arabic, a brief description of the Arabic alphabet and the HWR problem are in order. Figure 21.1 presents the letters of the Arabic alphabet.

From Fig. 21.2, one can see that it is not difficult to spot the four forms of each letter. The detached form and the final form look very similar. The initial and medial ones have the final portion of a letter left out. As an example, to make a hypothetical word beginning with “sheen” and ending with “noon”, we take the initial form of “sheen” and final form of “noon” to make “shn” as shown in Fig. 21.3.

The segmentation problem is very difficult for handwritten Arabic text [5]. The main additional difficulty associated with handwritten text vis-à-vis printed text is the tremendous increase in image variability. Earlier systems dealing with off-line handwritten Arabic text are described in [1, 4, 7, 20]. It is worth noting that most of these works assume perfect or near-perfect segmentation of handwritten Arabic words into separate characters before recognition. Such assumptions are not plausible. The difficulty associated with segmentation has led many researchers to bypass segmentation and character recognition in favor of direct word recognition [16]. Such approaches, however, are limited to available training words. The most general handwriting recognition techniques call for handling character segmentation ambiguity by means of over-segmentation and sophisticated recognition algorithms.

The techniques proposed by Khorsheed and Clocksin [15] have used hidden Markov models (HMMs) to deal with Arabic handwriting. These systems, however, have not made any connection of modeling over-segmentation of characters to variable duration states of HMMs. Instead, they have followed the traditional path of modeling each character by an HMM, and then concatenating character HMMs into a word. The system proposed in this chapter provides enormous simplification in terms of training as described in Sect. 21.4. It is applicable to general Arabic handwriting recognition, and is scalable. We note that a variable duration hidden Markov model (VDHMM)-based system was originally proposed for English handwriting in [10, 11]. However, the handwriting recognition (HWR) system developed in [10] cannot be applied to Arabic handwriting in a straightforward manner for three reasons: (1) the segmentation problem in Arabic is different and more challenging,

**Fig. 21.1** Arabic alphabet

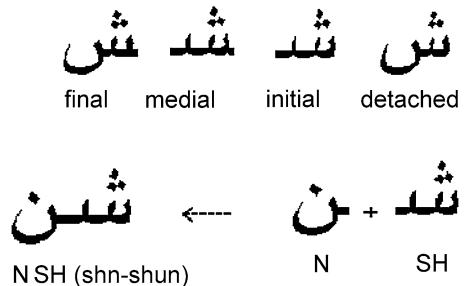
(2) representative features are different, and (3) there are up to four forms of the same letter in Arabic as described above.

The HMM modeling used for HWR by a number of authors follows a more traditional technique used by continuous speech recognition systems [9]. This type of

Fig. 21.2 Four forms of two Arabic letters



Fig. 21.3 Two letters could join cursively even in printed Arabic



modeling uses implicit segmentation in which a small vertical overlapping segment scans the word image in a raster scan fashion. This vertical segment is equivalent to a frame of speech, and is represented by features. From then on, techniques used by continuous speech recognition have been adapted [3, 9, 13, 19, 21, 22] for word recognition. Here, the system is trained with “truthed” words, and no explicit segmentation is needed. Although such techniques have gained currency, there is one theoretical and practical problem. Handwriting, in general, has overlapping segments. Thus, preceding and succeeding characters can have strokes belonging to other characters. Hence, such methods cannot train with proper character morphology when handwriting includes a substantial amount of overlapping strokes among characters. Unfortunately, this is the case with Arabic script. The VDHMM and stroke over-segmentation system does not suffer from such limitations, although it needs segmented and truthed characters for training.

This chapter is organized as follows. In the following sections, we provide detailed descriptions of our off-line Arabic handwriting recognition system based on VDHMM. A description of the overall system is described in Sect. 21.2. Section 21.3 describes the segmentation algorithm and relabeling of segments. A brief introduction to VDHMM and VDHMM statistics computation are given in Sect. 21.4. The details of our implementation, modified Viterbi algorithm, feature extraction, and word hypothesis generation are presented in Sects. 21.5 through 21.8. The experimental results and data collection efforts are discussed in Sects. 21.9 through 21.12, and the concluding remarks are made in Sect. 21.13.

21.2 System Overview

HWR systems involve a training phase in addition to a recognition phase. The training phase provides the system with word recognition capability for handwritten word images in a particular language. All handwritten words are modeled by one

hidden Markov model (HMM) where each distinct character is a state. By “character”, we mean a letter or symbol of an alphabet for a written language, and a “distinct character”, as used herein, refers to each distinct form of the characters of the alphabet. Thus, for example, the English alphabet has 26 distinct characters, or 52 distinct characters if upper and lower cases are considered separately. Although Arabic has 28 different characters, in our implementation, it has a total of 123 distinct forms of these characters (up to four forms, corresponding to the isolated, initial, middle, and final forms of each character, plus a number of ligatures). Therefore, the HMM as applied to Arabic has 123 states. Each state has a variable duration so that it can model a distinct character as made of a number of segments. In this context, the variable duration state is used to take care of the segmentation ambiguity among the consecutive characters.

During the recognition phase, a segmentation algorithm segments each word image so that each character is composed of one or several consecutive elementary units called segments, which are divided by segmentation points. An over-segmentation-relabeling algorithm (described below) determines a sequence of these segments, and the resulting sequence of segments is modeled using a variable duration hidden Markov model (VDHMM). In the VDHMM, various combinations of consecutive segments are observations. Thus, a series of segments leads to an observation sequence, and different combinations lead to different sequences. Each observation is then scored against trained models of distinct characters developed from “truthed” characters, or characters extracted from imaged words during the training phase, in which the identity of each character is known, and each observation is assigned a probability of belonging to each state. A recognition algorithm adapted to the VDHMM uses these probabilities in addition to linguistic knowledge reflected in later-described state probabilities to output an optimal string of one or more characters as composing the word image.

In the training phase, a corpus of training images of handwritten Arabic word samples are processed, in which each image T_0 is captured and subjected to pre-processing and segmentation using an over-segmentation-relabeling algorithm. Moreover, the true segmentation points between characters are identified so that the system is trained on images of truthed characters. Pre-processing includes modifying the captured image to remove noise and correct skewing or slanting of characters. Next, the feature information of individual characters (i.e., the truthed characters) of the imaged word T_0 is extracted. The selected features extracted are those considered to have descriptive characteristics of each distinct character and may be specifically tailored for a particular written language. Thus, based on the knowledge of the characteristics of the particular written language, a set of features may be selected or derived as being appropriate for the language. A more detailed description of the feature selection and representative features of Arabic words will be provided later. Each feature value is scaled in the range from 0–1, and the scaled data string of all features extracted for a given character image is compactly represented by a feature vector. For example, where a given feature is strongly present in a character, the feature is assigned a “1”, and where a given feature is absent, the feature is assigned a “0” in the feature vector for that character image.

The estimation of symbol probability distribution parameters, which is needed to compute the symbol probability during recognition, includes calculating one or more representative feature vectors for each state (i.e., each distinct character) so as to provide one or more mean feature vectors for that state. For a given state, each of these representative feature vectors corresponds to a dominant writing style for the letter assigned as that state. For each state, these representative feature vectors are calculated based on the collection of individual feature vectors that were extracted from the corpus of training images. A statistical distribution is derived for each dominant writing style of a distinct character, thereby providing parameters for a symbol probability distribution. Symbol probabilities give a statistical measure that a given feature vector (such as that extracted from test image X_0) is indicative of a distinct character. Thus, during the training phase, mathematical model parameters are constructed for each of the distinct characters of the chosen language. This allows estimation of the symbol probabilities in the recognition phase in which a given feature vector of an observation (composed of one or more combined consecutive segments) is statistically compared against all feature vectors extracted during the training phase and, for each distinct character, a probability is estimated that the observation is an image of that distinct character. Thus, as we have said, symbol probabilities give a statistical measure that a given feature vector (such as that extracted from test image X_0) is indicative of a distinct character. The symbol is what is observed. It is matched against all the character models created during training. The matching likelihood, a probability, is the symbol probability. *In essence, the symbol probability distribution is a distribution modeling of characters represented by feature vectors.* Furthermore, the symbol probability may be modeled using a continuous, continuous-discrete hybrid, or a discrete distribution, as described later.

For each distinct character (i.e., state), a state duration probability is also estimated. As a result of the segmentation, each individual image of a like distinct character from the corpus of training images may be divided into one or more segments. Based on the collection of the segmentation results gathered from these individual images, the likelihood that an image of a distinct character will be segmented into a certain number of segments (i.e., segmented to have a certain “duration”) may be determined. For example, the ratio of the number of individual images of a character that were divided into two segments against the total number of images of that character appearing in the corpus of training images provides a probability that another image of that distinct character will likewise be segmented into two segments.

In the recognition phase, test images X_0 , or images to be recognized using the system, are processed similarly to the way that training images T_0 are processed. That is, an image X_0 (consisting of one or more characters) is preprocessed and segmented using an over-segmentation-relabeling algorithm, and feature information is extracted for one segment or a combination of several consecutive segments. More specifically, a feature vector is derived for each observation, i.e., for each image built of one or several consecutive segments merged together. For each particular image X_0 , multiple observations are possible, and one observation for each possible combination of consecutive segments is evaluated. An upper limit of total combined segments for an observation may also be defined. The upper limit of combined segments is in a range of 2 to 7 segments. In our experiment, an observation

consists of at most 4 segments. In general, state probabilities include initial, transition, and last-state probabilities computed from a given dictionary, reflecting the likelihood that the imaged word begins with one distinct character, transitions to a second distinct character, and ends with the second distinct character, respectively. Thus, a state transition is given by bigram probabilities. Trigram probabilities can also be used, but the cost of using them is much increased complexity both during training and recognition. The duration, symbol, and state probabilities are processed in a recognition algorithm which outputs optimal character strings based on these probabilities. This determination of optimal strings is not guaranteed to be a true word, and therefore, a post-processing step is used to further match these character strings to lexicon words for final word recognition. A post-processing step, in which a given dictionary is used to provide hypotheses of words based on the output strings from the recognition algorithm, is also shown in Fig. 21.4, which gives a graphical overview of our system.

21.3 Segmentation

For our segmentation algorithm, we propose the following criteria:

- Each complete character can be segmented into at most four parts.
- All touching characters must be split.
- No “null” state is allowed [10].

These criteria are quite realistic, and are realized using mathematical morphology as described next.

21.3.1 Generic Segmentation Algorithm

The mathematical foundation of this algorithm is described in [10], and is not repeated here. After a series of pre-processing operations, an individual word image becomes a collection of black pixels against a white background. The highly dense areas of black pixels are the main character bodies; areas with very little density of black pixels are considered for segmentation points. Usually, some heuristics are also applied at this stage. For instance, if two segmentation points are very close, one of them is spurious and the segmentation points are merged.

The next step is to locate the first segmentation point. Usually, it is the leftmost segmentation point for English and the rightmost segmentation point for Arabic. Next, the sequence of segmentation points is determined. That is, given all the segmentation points, what is the best order to follow? The final step is the verification of the last segmentation point. This step is usually required to reject any residual stroke or ligature that may appear at the end. Therefore, a generic word segmentation algorithm can be written as follows:

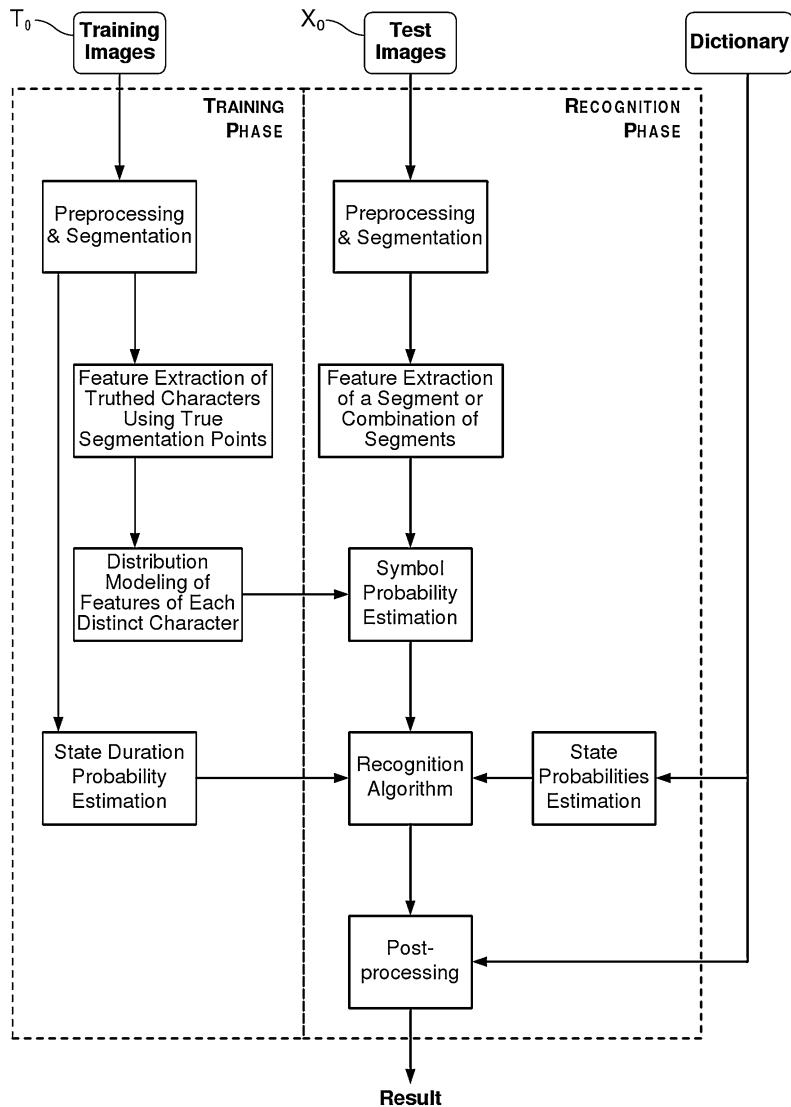


Fig. 21.4 VDHMM-based Arabic HWR system overview

1. Locate the segmentation points.
2. Verify the segmentation points using heuristics.
3. Locate the first segmentation point.
4. Find the correct sequence of segmentation points.
5. Verify the last segmentation point.

The generic algorithm does extremely well in segmenting words into segments according to the segmentation criteria listed above. But the main problem with this

generic algorithm, as far as Arabic is concerned, is that it fails to assign dots in diacritics as consecutive segments of its associated characters. This limitation is overcome by a segment relabeling algorithm as described next.

21.3.2 Segment Relabeling for Arabic Handwritten Words

This generic segmentation algorithm, as described in [10], locates segmentation points, but sequences these segmentation points such that diacritics are placed at the beginning or end of cursively connected characters in their segment sequencing. For example, starting with an Arabic word image X_0 as shown in Fig. 21.5, the generic algorithm segments X_0 as shown in Fig. 21.6, illustrating each segment “S” in a box. The generic algorithm provides the sequence of segments “S” in the order of 1 to 13, labeled respectively as S1 to S13, from right to left for Arabic. Thus, segments S7 and S8 are placed in the segment sequence so as to be at the end of cursively connected segments S2 to S6. However, S5 and S8 in fact make up the Arabic letter, in its medial form, that corresponds to the digraph “noon”, and S4 and S7 in fact make up the Arabic letter, in its medial form, that corresponds to the digraph “ba”. Because S8 is not placed in the segment sequence as a segment consecutive to the main character body (S5) of which it is a part, no observation (combination of segments) will form the correct character. That is, no observation will consist of S5 + S8, and likewise, no observation will consist of S4 + S7, although each of these segment combinations makes up the character. Therefore, it becomes unlikely that a VDHMM-based HWR system relying on the generic segmentation algorithm would recognize the correct character, because no combination of segments would result in the correct boundaries between characters, or the genuine segmentation points.

Figure 21.7 provides another example of an Arabic word image (shown in Fig. 21.8) being processed using the generic segmentation algorithm that results in diacritics and small segments being removed from the segment(s) of their associated main character bodies. As shown in Fig. 21.7, the generic segmentation algorithm places segment S14 so as to be after cursively connected segments S10 to S13 in the segment sequence. However, S14 is in fact associated with S11 (which form the Arabic letter, in its medial form, that corresponds to the digraph “ya”). Furthermore, S15 is in fact associated with S13 and S12 (which together form the Arabic letter, in its final form, that corresponds to the digraph “teh marbuta”), but the displacement of S14 prevents S15 from following segments S12 and S13 in the segment sequence. A similar displacement occurs with S4 being placed at the beginning of the cursively connected segments S5 to S9 and not consecutively to S8 (where S4 and S8 together also form “ya” medial). As a result, no observation evaluated in a VDHMM-based HWR system using the segment sequence illustrated in Fig. 21.7 will have a segment combination which results in the correct characters. That is, no observation will consist of S11 + S14, consist of S12 + S13 + S15, or consist of S4 + S8 which respectively make up the correct characters of the imaged word X_0 of Fig. 21.8.

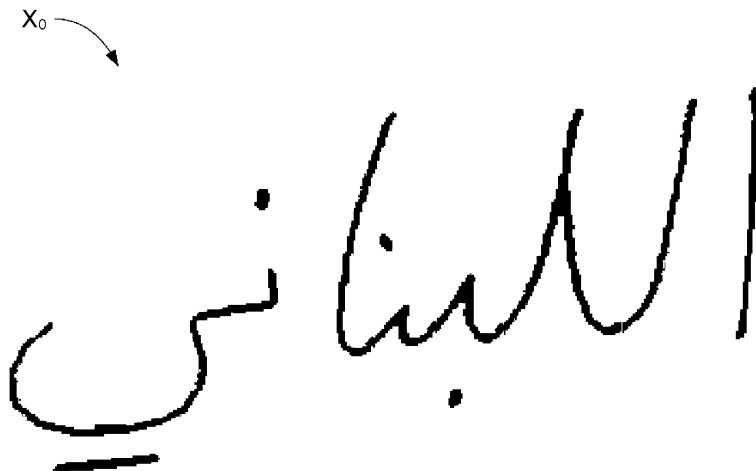


Fig. 21.5 Original Arabic word image (al-lubnani)

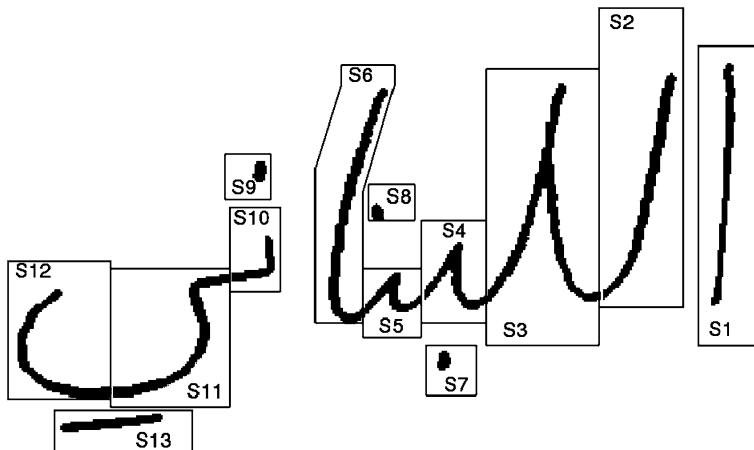


Fig. 21.6 Output of generic segmentation algorithm

The over-segmentation-relabeling (OSR) algorithm will now be described with reference to Figs. 21.9, 21.10, and 21.11 and the example imaged words illustrated in Figs. 21.12, 21.13, 21.14, and 21.15. In general, the OSR algorithm relabels the segments displaced by the generic segmentation algorithm so that these segments immediately precede or follow a segment of the associated main character body. The resulting segment sequence determined by the OSR algorithm may then be used in the system of Fig. 21.4 to recognize an imaged word X_0 (or T_0) as having certain character(s). In the OSR algorithm presented in Figs. 21.9, 21.10, and 21.11, the algorithm finds the number of segments in the original image X_0 (or T_0). This step further includes finding the first and last segments of the segmented word image.

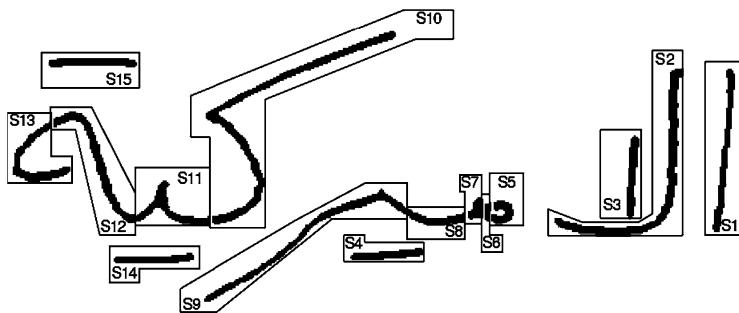


Fig. 21.7 Output of generic segmentation algorithm

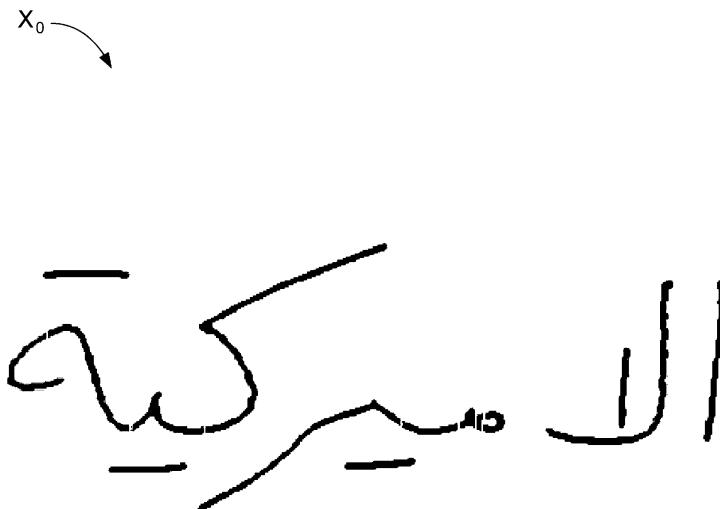
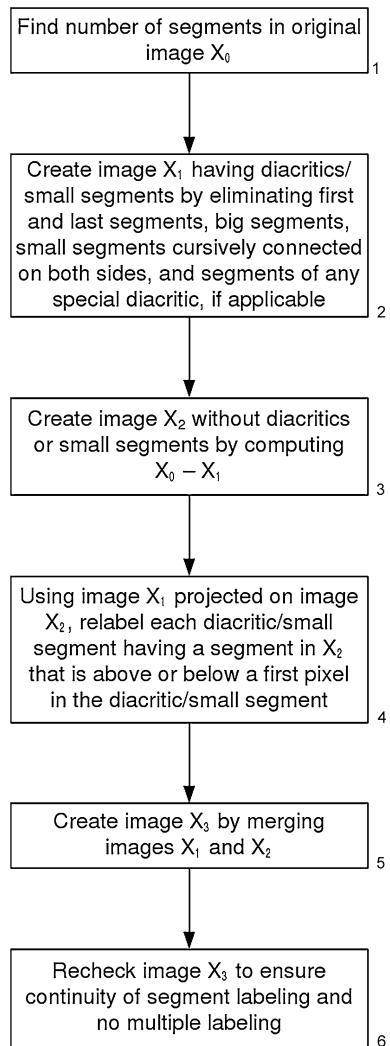


Fig. 21.8 Original Arabic word image (al-amirkiyya)

Next, an image X_1 is created to include only certain diacritics/small segments (referred to herein as “unsituated segments”) by taking image X_0 and eliminating the first and last segments, big segments, and segments of any special diacritic. The big segments are segments having an X-coordinate or Y-coordinate coverage that exceeds a threshold value. The threshold value may vary depending on the language. For Arabic for example, a threshold value may be between 20 to 60 pixels, and a typical value of 45 pixels is found to be quite optimal for all images. Moreover, segments that are not considered “big” in terms of their pixel extent, but are flanked on both their left and right by segmentation points dividing a main body of cursive writing (i.e., “small” segments that are cursively connected to a segment on each side) are also eliminated. These eliminated segments are collectively referred to herein as “situated segments,” since these segments are not candidates for relabeling. As noted, image X_1 is created to include small diacritics. A special diacritic may be, for

Fig. 21.9 Segment relabeling scheme



example, a shadda in Arabic. A shadda may be specially treated as three consecutive small segments. Other special diacritics may be designated as desired based on the particular language applied to the system of Fig. 21.4.

Next, an image X_2 is created to include only the situated segments. Image X_2 is created, for example, by removing the segments of X_1 from original image X_0 (i.e., computing $X_0 - X_1$). In the next two steps of the OSR algorithm, the sequence of segments is determined. In the first of these two steps, in particular, each unsituated segment having a situated segment above or below is relabeled so as to either immediately precede or follow the situated segment in the sequence of segments. In the OSR algorithm presented in Fig. 21.9, the relabeling of these unsituated segments is performed on the image of X_1 based on a projection of image X_1 on image X_2 .

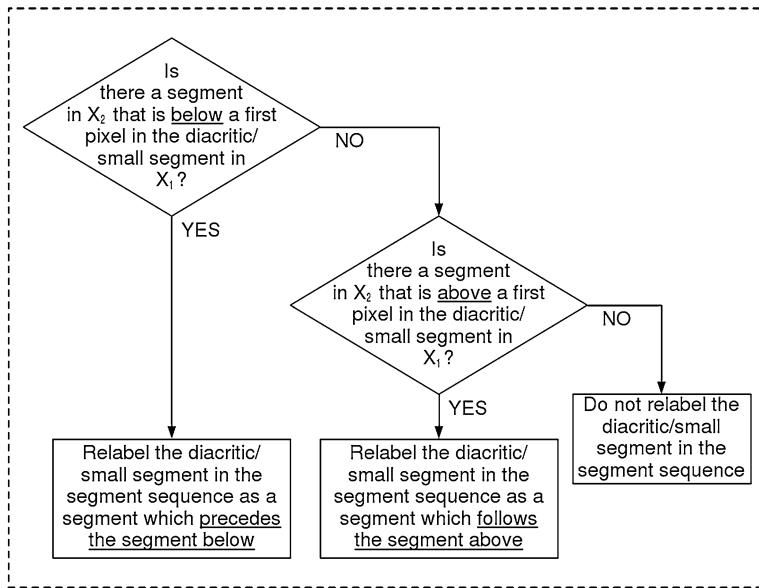


Fig. 21.10 One of two alternate relabeling schemes of Step 4 of Fig. 21.9 where a small segment precedes the main segment below

X_1 projecting on X_2 is illustrated in Fig. 21.13 and Fig. 21.15 for each word image X_0 shown in Fig. 21.5 and Fig. 21.8, respectively. In Fig. 21.13 and Fig. 21.15, for illustration purposes only, unsituated segments of image X_1 are shown as hatched-filled segments, while the situated segments of image X_2 are shown as solid-filled segments. The relabeling procedures illustrated in Fig. 21.13 and Fig. 21.15 correspond to two alternative scenarios of Step 4 of Fig. 21.9, and are described clearly in Fig. 21.10 and Fig. 21.11, respectively.

21.4 VDHMM Statistics

The recognition algorithm uses the VDHMM statistics described below and classifies the imaged word as having a string of one or more characters. A hidden Markov model (HMM) classifier and, in particular, a modified Viterbi algorithm (MVA) adapted to a VDHMM are used to recover from the whole sequence of observations the optimal, or most likely, letter sequence (i.e., the “hidden” state sequence), and thus the set of correct segmentation points from a superset of over-segmentation points. The recognition algorithm relies on the segment sequence determined by the OSR algorithm to recognize certain segmentation points as being the most likely boundaries between characters. For example, from the segment sequence of example image X_3 illustrated in Fig. 21.12, the true segmentation points are between S1 and S2, S2 and S3, S3 and S4, S5 and S6, S7 and S8, S8 and S9, and S10 and S11

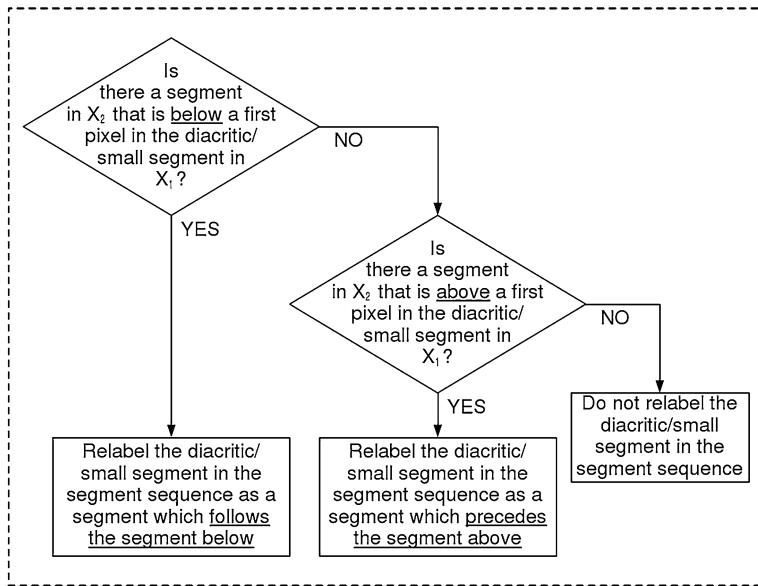


Fig. 21.11 One of two alternate relabeling schemes of Step 4 of Fig. 21.9 where a small segment follows the main segment below

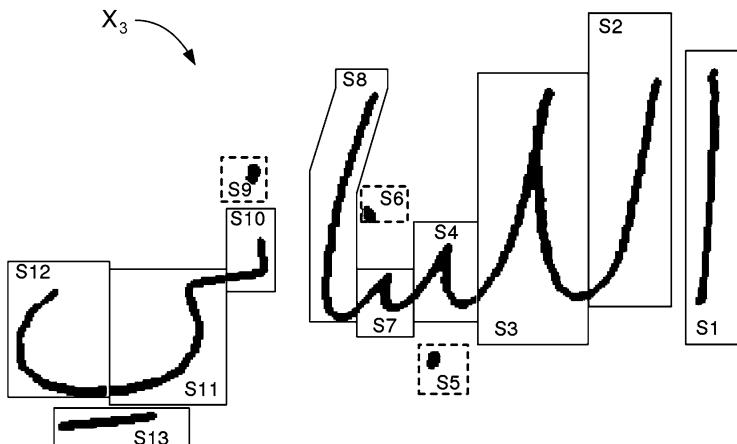


Fig. 21.12 Output of segmentation and relabeling algorithm

(the last three segments from S11 to S13 together forming the Arabic letter, in its final form, that corresponds to the digraph “ya”). In the image X_3 of Fig. 21.14, the true segmentation points are between S1 and S2, S3 and S4, S6 and S7, S8 and S9, S9 and S10, S10 and S11, and S12 and S13 (the last three segments from S13 to S15 together forming the Arabic letter, in its final form, that corresponds to the

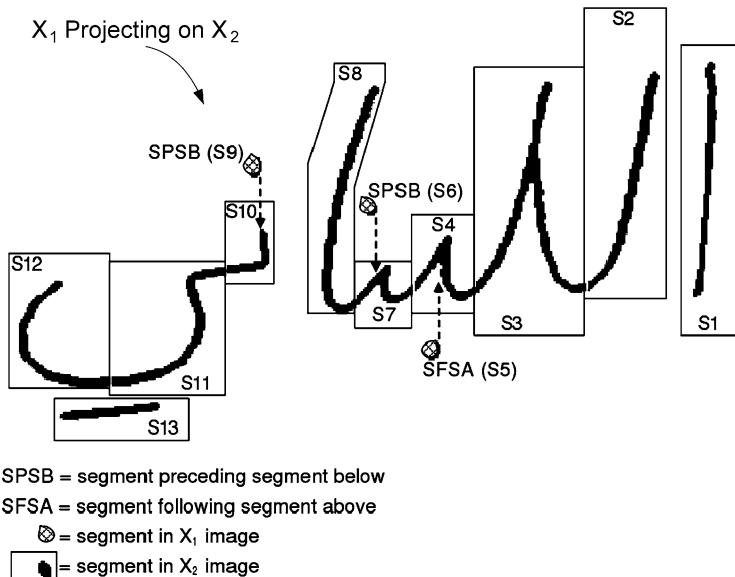


Fig. 21.13 Projection of small segments to be relabeled on the main body of the word image

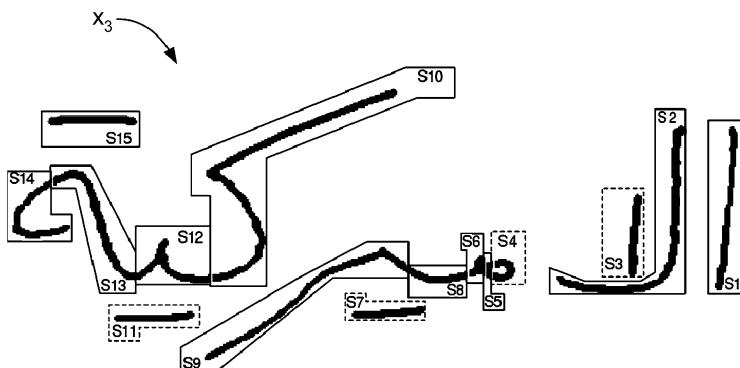
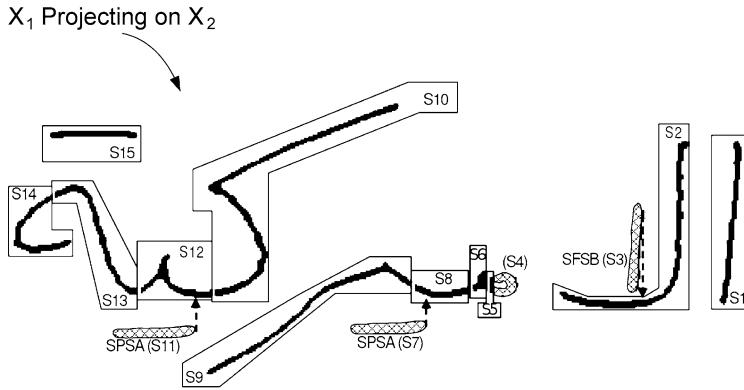


Fig. 21.14 Output of segmentation and relabeling algorithm

digraph “teh marbuta”). The recognition algorithm outputs multiple strings of characters as possible classifications for the imaged word. The post-processing step is then used to hypothesize a set of one or more words from the given dictionary which are suggested by the optimal string(s).

The discrete state duration probability $P(d_j|q_i)$ is estimated from the training samples T_0 with $d = 1; 2; 3; 4$ and $i = 1; 2; \dots; 123$, because the segmentation algorithm segments each handwritten character into at most 4 segments, and there are 123 distinct characters in the Arabic language. In another implementation, more or fewer segments per character and more or fewer states can be considered. The HMM



SFSB = segment following segment below

SPSA = segment preceding segment above

= segment in X_1 image

= segment in X_2 image

Fig. 21.15 Projection of small segments to be relabeled on the main body of the word image

may be denoted by a compact notation $\lambda = (\Pi, A, \Gamma, B, D)$. Here, Π stands for a set of initial probabilities, A stands for a set of transition probabilities, B stands for a set of symbol probabilities, and D stands for a set of duration probabilities. The last state probability Γ is included in this definition. One could interpret this probability as a transition probability to an imaginary “final” or “absorbing” state.

Π , A , Γ , B , and D are defined as follows:

$$\begin{aligned}
 \Pi &= \{\pi_i\}; \quad \pi_i = \Pr\{i_1 = q_i\} \\
 A &= \{a_{ij}\}; \quad a_{ij} = \Pr\{q_j \text{ at } t+1 \mid q_i \text{ at } t\} \\
 \Gamma &= \{\gamma_i\}; \quad \gamma_i = \Pr\{i_T = q_i\} \\
 B &= \{b_j(O_t^{t+d})\}; \quad O_t^{t+d} = (o_t o_{t+1} \cdots o_{t+d}) \\
 D &= \{P(d \mid q_i)\}
 \end{aligned} \tag{21.1}$$

where $O = (o_1 o_2 \cdots o_T)$ is the observation sequence.

In the training phase, the VDHMM model parameter $\lambda = (\Pi, A, \Gamma, B, D)$ is estimated. These statistics, which are defined in Eq. (21.1), are computed from two sources: training images and a given dictionary. As described before with reference to Fig. 21.4, after the segmentation algorithm is executed on the training images, the state duration probability $P(d|q_i)$ is computed by counting the number of segments for each character. Extracting the features from each complete character, which is composed of one or more segments, we are able to estimate the parameters of the symbol probability distribution. The initial state, last state, and the state transition

probabilities are estimated by examining the occurrences of the first character, the last character, and every pair of characters (first-order transitions) for every word in the given dictionary. The preferred dictionary contains not only the list of legal words for the specified application domain, but also their associated frequencies, i.e., the a priori probability for each word. The precision of these estimation procedures depends on the accuracy and the relevance of the available knowledge sources. If the given dictionary truly reflects the reality of word occurrences in the application domain, the language-based probabilities obtained from it are considered to be optimal. If the training images cover all the basic writing styles, one can get a good estimate of the state duration and the symbol probabilities based on these images. In the real world, however, the information gathering mechanism for both these sources is not perfect. In a VDHMM-based HWR system, on the other hand, the dependence on one particular source of information is balanced against the other, since the decision is made after combining these two sources in a sense of adaptive balance. When the word is written ambiguously, the recognition scheme may take most of the advantage from dictionary information. When the system is familiar with the particular writing style, it would like to make the decision relying more on the shape information.

21.4.1 State Probabilities

The 123 letters (i.e., distinct characters) of the Arabic alphabet are defined as the states of this exemplary VDHMM. It is therefore straightforward to compute the initial π_i , transition α_{ij} , and last-state λ_j probabilities as:

$$\pi_i = \frac{\text{no. of words beginning with } l(q_i)}{\text{total no. of words in the dictionary}} \quad (21.2)$$

$$\alpha_{ij} = \frac{\text{no. of transitions from } l(q_i) \text{ to } l(q_j)}{\text{no. of transitions from } l(q_i)} \quad (21.3)$$

$$\gamma_j = \frac{\text{no. of words ended with } l(q_j)}{\text{total no. of words in the dictionary}} \quad (21.4)$$

where “l” stands for “letter” and the function $l(\cdot)$ transforms the state to its representative member of the alphabet. All of these probabilities are derived from the dictionary (shown as dictionary in Fig. 21.4). If the dictionary is changed, these probabilities can be easily recomputed. This capability to adapt to any dictionary makes the system highly portable and scalable.

To calculate the state duration probability, the segmentation procedure must be performed over all training images. Inspecting the segmentation results, the state duration probability $P(d_j | q_i)$ is estimated as

$$P(d | q_i) = \frac{\text{no. of times that letter } (q_i) \text{ is segmented into } d \text{ segments}}{\text{no. of times that letter } (q_i) \text{ appears}} \quad (21.5)$$

Because the segmentation algorithm ensures that the maximum duration for each of the 123 states is 4, there are 492 discrete probabilities ($= 123 \times 4$) that must be estimated for state duration. It is possible to estimate these probabilities by inspection and counting of segmented training images. This is the discrete form of modeling the state durations, and it avoids any a priori assumption about duration distribution.

21.4.2 Symbol Probability Distribution

As noted above, symbol probability may be modeled using a discrete or continuous distribution, wherein the shape information of each character symbol, i.e., feature vector, is modeled either as an independently distributed multivariate discrete distribution or as a Gaussian mixture distribution. Continuous and discrete distributions are described below.

21.4.3 Modeling Characters by Continuous Symbol Probability Distribution

As stated before, symbol probabilities give a statistical measure that a given feature vector (such as that extracted from test image X_0) is indicative of a distinct character. The symbol is what is observed. It is matched against all the character models created during training. The matching likelihood, a probability, is the symbol probability. In essence, a symbol probability distribution is a distribution modeling of characters represented by feature vectors. The most general representation of the probability density function (PDF) is a finite mixture of the form:

$$b_j(x) = \sum_{m=1}^{M_j} c_{jm} \cdot \mathcal{N}[x, \mu_{jm}, U_{jm}], \quad 1 \leq j \leq N \quad (21.6)$$

where N represents a Gaussian distribution with mean vector μ_{jm} and covariance matrix U_{jm} for the m^{th} mixture component in state j , x is the vector being modeled, M_j is the number of Gaussian component N in state j , and c_{jm} is the mixture coefficient for the m^{th} Gaussian component in state j [23]. The mixture gains satisfy the stochastic constraint

$$\sum_{m=1}^{M_j} c_{jm} = 1, \quad 1 \leq j \leq N \quad (21.7)$$

$$c_{jm} \geq 0, \quad 1 \leq j \leq N, i \leq m \leq M_j$$

so that the PDF is properly normalized, i.e.,

$$\int_{-\infty}^{\infty} b_j(x) dx = 1, \quad i \leq j \leq N \quad (21.8)$$

Here, each Gaussian distribution in the feature space is expected to represent one among many different writing styles of the characters. So, the first problem is how to estimate the number of Gaussian distributions for each state, i.e., M_j in Eq. (21.6). For each state, the K-means clustering algorithm with a fixed signal-to-noise ratio (SNR) is used to partition the training samples into several groups, and M_j is equated with the number of groups. The mixture coefficient c_{jm} is then estimated as

$$c_{jm} = \frac{\text{no. of training samples in } \mathcal{H}_{jm}}{\text{total no. of training samples for state } q_j} \quad (21.9)$$

where \mathcal{H}_{jm} is the set of group m of state q_j . Please note that the c_{jm} in Eq. (21.9) satisfies Eq. (21.7) and can be interpreted as the a priori probability of the m^{th} particular writing style for distinct character letter (q_j). For each group in state q_j , the associated parameters for Gaussian distribution are estimated as

$$\mu_{jm} = \sum_{x \in \mathcal{H}_{jm}} \frac{1}{N_{jm}} x \quad (21.10)$$

$$U_{jm} = \sum_{x \in \mathcal{H}_{jm}} \frac{1}{N_{jm}} (x - \mu_{jm})(x - \mu_{jm})^T \quad (21.11)$$

where x is the feature vector of the training sample, N_{jm} is the number of samples in H_{jm} , and T denotes the matrix transposition. In this implementation, the covariance matrix U_{jm} is assumed to be diagonal; that is, the features are assumed to be independent of each other. Further details of parameter computations can be found in [10, 11]. In the recognition phase, $b_j(O)$, the symbol probability density for observation O can be computed from Eq. (21.6) by substituting x by O . It is relevant to mention here that the observation O in VDHMM is composed of one or several consecutive segments. From this viewpoint, the symbol probability is modified as

$$b_j(o_1 o_2 \cdots o_d) = b_j(O_1^d)^d \quad (21.12)$$

where O_1^d is the image built by merging segment images $o_1; o_2; \dots; o_d$ together. The power of d in Eq. (21.12) is used to balance the symbol probability for different numbers of segments. This is a necessary normalization procedure when every node in the Viterbi net is used to represent a segment.

21.4.4 Modeling Characters by Discrete Symbol Probability Distribution

First, all N features are assumed to be independent of each other, and each one is modeled as a discrete distribution [26]. The symbol probability density for observation O can be computed as

$$b_j(O) = \prod_{i=1}^N P(s_i) \quad (21.13)$$

Here, s_i is the i^{th} feature of the observation O . Once again, Eq. (21.12) is used, where O_1^d is the image built by merging segment images $o_1; o_2; \dots; o_d$ together.

For instance, let us take the occurrence of number of loops which is one of the features in the feature vector. In the letter “alif” there is no loop. Since we model (or accept) 0 to 3 loops for loop feature, let us assume that the discrete distribution of loop feature in “alif” is modeled as 1 (no loop), 0 (1 loop), 0 (2 loops), and 0 (3 or more loops). In practical implementation, we assign 0 to a small value (0.001). Similarly, we can model the probability of each feature for any character, i.e., state. During recognition, as a feature vector is computed from a segment combination (combination of 1 to 4 segments), the probability of the feature for each state is evaluated using the discrete probability distribution model. Then, for each state, the probabilities of all the features are multiplied as shown in Eq. (21.13) above.

21.4.5 Modeling Characters by Continuous-Discrete Symbol Probability Distribution

Here, N features are distributed into two groups, N_1 and N_2 . All features belonging to N_1 are distributed using a continuous model given by Eq. (21.6), and all features belonging to N_2 are distributed using a discrete model given by Eq. (21.13). The two probabilities are multiplied and then normalized by Eq. (21.12) to compute the symbol probability. Thus,

$$N = N_1 + N_2$$

All features belonging to N_1 are distributed using a continuous multivariate Gaussian model. Then, Eq. (21.6) can be used to compute the probability (or likelihood) of these observed N_1 features. Let this probability be $b_j(O_{N_1})$.

N_2 features are modeled as independent discrete random variables. Then, Eq. (21.13) can be used to compute the probability (or likelihood) of these observed N_2 features. Let this probability be $b_j(O_{N_2})$.

Now, the combined symbol probability is computed as

$$b_j(O) = b_j(O_{N_1}) \times b_j(O_{N_2})$$

This scheme has not been implemented in our system because of complexity and time constraints.

21.5 Recognition Using Modified Viterbi Algorithm

Given the aforementioned VDHMM statistics, the objective of the recognition phase is to find the optimal state sequence I^* given a sequence of observations O and model parameter λ , i.e.,

$$I^* = \arg \max_I [\Pr(I | O, \lambda)] \quad (21.14)$$

where

$$\max_I \Pr(I | O, \lambda) = \max_I \frac{\Pr(O, I | \lambda)}{\Pr(O)} = \max_{1 \leq i \leq N} \frac{\delta_T(i) \times \gamma(i)}{\Pr(O)} \quad (21.15)$$

and the probability

$$\delta_t(j) = \max_{1 \leq i \leq N} \left\{ \max_{1 \leq d \leq D} \{ \delta_{t-d}(i) a_{ij} P(d | q_j) b_j (O_{t-d+1}^t)^d \} \right\} \quad (21.16)$$

Equations (21.14), (21.15), and (21.16) suggest the Viterbi algorithm for finding the best path. Two modified Viterbi algorithms (MVAs), which provide an ordered list of the best L state sequences and are described in [11], are used in our system. The first MVA is a parallel version which simply extends the Viterbi net to three-dimensional storages, where the third dimension represents the choice. On the other hand, the serial version MVA, which searches the $(l + 1)^{\text{th}}$ globally best path based on the previous l best paths, can be more efficiently programmed. These two MVAs are adapted to VDHMMs by incorporating the duration probability. The modified Viterbi algorithm for the serial version is described in detail in [10, 11].

21.6 Feature Selection

The segment images in handwriting are two-dimensional binary signals. To select good features from these signals, the following criteria are considered useful. (1) Features should be preferably independent of translation and size. To a limited extent, the features should be independent of rotation. (2) Features should be easily computable. (3) Features should be chosen so that they do not replicate each other. These criteria ensure efficient utilization of information content of the feature vector. A comprehensive, cross-lingual feature typology may be used as a starting point, and a set of features for the particular language may be selected from there. Experimental results may be used as the selection criteria to determine whether a selected set of features yields accurate results in the classification stage of the HWR system. In our implementation, a set of 45 features make up a feature vector. The given image segment from which features are to be extracted is first transformed into a binary image with pixels on the object defined as “black” (1) and the background as “white” (0).

Out of 45 features used in our system, 26 features are described in detail in [11]. These include three moment features that capture the global shape information (i.e., “geometrical moments”), as well as 11 geometrical and topological features such as loops, X-joint feature, horizontal and vertical zero crossing features, T-joint feature, number of end points in upper, middle, and lower zones of the character image, and number of segments in upper, middle, and lower zones. Further details regarding these features are described in [11]. These features have been widely used in one form or another (and also used under different names) because they are helpful in capturing both the global and local shape information, and are particularly useful in handwriting since they are robust with respect to writing style variation. Since a binary character image can be described by the spatial distribution of its black pixels, 12 pixel distribution features are computed by counting the pixels in every neighboring zone, excluding the cross neighbors. To compute the pixel distribution features, the image segment is first covered by the minimum bounding rectangle. Then the rectangle is non-uniformly divided into 3×3 zones based on the density of the image segment and the center of gravity. The number of pixels in each coupled zone is counted, and then scaled by the maximum among them [11].

The set of 45 features also includes 19 new features useful for describing Arabic handwriting, and is presented below. It should be understood that more or fewer features may be used to describe Arabic, and that other written languages may be defined by other feature sets, which may include features and weightings particularly suitable for that language.

Two aspect ratio features, f_{hv} and f_{vh} , are computed by finding the maximum vertical extent (vd) and maximum horizontal extent (hd) of the image segment. Feature f_{hv} is based on horizontal-to-vertical aspect ratio, and feature f_{vh} is based on vertical-to-horizontal aspect ratio. We require both features because their maximum values are set to unity.

Four features, f_{du} , f_{dm} , f_{dl} and f_{da} , relating to the number of diacritics or dots in each zone are computed. Each zone contains diacritics (or dots) that are part of the characters. The number of disconnected dots in each zone is computed. Feature f_{du} is based on dots in the upper zone, and is defined in the following manner:

$$f_{du} = \begin{cases} 0.0 & \text{no dots} \\ 0.50 & \text{one dot} \\ 0.75 & \text{two dots} \\ 1.0 & \text{three or more dots} \end{cases} \quad (21.17)$$

Features f_{dm} and f_{dl} may be similarly defined for dots in the middle and lower zones. If any of f_{du} , f_{dm} or f_{dl} is nonzero, f_{da} is set to 1.0; otherwise it is 0.0.

Eight reference line features relating to the number of diacritics or dots with respect to the baseline of the word (global baseline) and the local baseline of the segment (or segment combination) are computed. The baseline is defined as the horizontal line on which the character sits. One can define f_{dub} , which stands for

“dot feature above baseline,” as:

$$f_{dub} = \begin{cases} 0.0 & \text{no dots above baseline} \\ 0.50 & \text{one dot above baseline} \\ 0.75 & \text{two dots above baseline} \\ 1.0 & \text{three or more dots above baseline} \end{cases} \quad (21.18)$$

Similarly, one can define f_{dib} , standing for “dot feature below baseline,” as:

$$f_{dib} = \begin{cases} 0.0 & \text{no dots below baseline} \\ 0.50 & \text{one dot below baseline} \\ 0.75 & \text{two dots below baseline} \\ 1.0 & \text{three or more dots below baseline} \end{cases} \quad (21.19)$$

Similarly, small segments-based features f_{sub} and f_{sib} with respect to baseline are defined as follows:

$$f_{sub} = \begin{cases} 0.0 & \text{no small segments above baseline} \\ 0.50 & \text{one small segment above baseline} \\ 0.75 & \text{two small segments above baseline} \\ 1.0 & \text{three or more small segments above baseline} \end{cases} \quad (21.20)$$

$$f_{sib} = \begin{cases} 0.0 & \text{no small segments below baseline} \\ 0.50 & \text{one small segment below baseline} \\ 0.75 & \text{two small segments below baseline} \\ 1.0 & \text{three or more small segments below baseline} \end{cases} \quad (21.21)$$

The features computed in Eqs. (21.18)–(21.21) use the global baseline of the word; that is, the baseline of the entire word is used as the baseline of the segment(s) used to compute the feature. One can replace this global baseline by the local baseline of the segment(s) used to compute the feature. Thus, four more features f_{duib} , f_{diib} , f_{suib} , and f_{siib} are computed using Eqs. (21.18)–(21.21) but replacing global baseline with local baseline. Other features based on reference lines may be employed.

Two stroke connectedness features, f_{cr} and f_{ci} , are defined as follows:

$$f_{cr} = \begin{cases} 1.0 & \text{if image segment naturally connected to segment on right} \\ 0.0 & \text{otherwise} \end{cases} \quad (21.22)$$

$$f_{ci} = \begin{cases} 1.0 & \text{if image segment naturally connected to segment on left} \\ 0.0 & \text{otherwise} \end{cases} \quad (21.23)$$

Two more zero crossing features known as maximum horizontal zero crossing feature, f_{mzh} , and maximum vertical zero crossing feature, f_{mzv} , are computed.

Maximum horizontal zero crossing feature, f_{mzh} , is defined as:

$$f_{mzh} = \begin{cases} 0.0 & \text{maximum horizontal zero crossing none} \\ 0.25 & \text{maximum horizontal zero crossing one} \\ 0.50 & \text{maximum horizontal zero crossing two} \\ 0.75 & \text{maximum horizontal zero crossing three} \\ 1.0 & \text{otherwise} \end{cases} \quad (21.24)$$

In a similar manner, maximum vertical zero crossing feature, f_{mzv} , is computed by counting the maximum number of vertical zero crossings.

For the given character image, three 8-directional chain code-based features, f_{ch} , f_{rough} , and f_{con} , are computed. At every bifurcation point, a new chain is initiated. All chains with a length greater than a threshold are considered good chains. Feature f_{ch} is based on chain code, and is defined as:

$$f_{ch} = \begin{cases} 0.0 & \text{no good chain} \\ 0.25 & \text{one good chain} \\ 0.50 & \text{two good chains} \\ 0.75 & \text{three good chains} \\ 1.0 & \text{4 or more} \end{cases} \quad (21.25)$$

For the given character image and its 8-directional chain codes, differential chain codes are computed. In these chain codes, the number of nonzero codes is counted, and the ratio of the number of nonzero codes to the number of codes is computed. This ratio is multiplied by a factor (default = 1.25) to give feature f_{rough} , chain code-based roughness measure of the character.

Furthermore, from differential chain codes, the number of entries that represents a sharp turn (90° or more) is computed, and f_{con} , a feature based on chain code sharp turn, is defined similarly as f_{ch} described above.

Observe that all the features are scaled in the range from 0 to 1. The moment features, by virtue of their definition, are also scaled in the same range. Such scaling ensures that no feature gets more or less weight unless otherwise intended.

21.7 Post-processing

The output of the Viterbi algorithm is not guaranteed to be a legal word from a given dictionary, especially if the corpus of training images is not extensive. Accordingly, the HWR system is supplemented with a post-processing module, whose objective is to output hypotheses based on the weighted edit distances of MVA output strings to all the words of the given dictionary, i.e.,

$$W_j^* = \arg \max_{1 \leq j \leq J} \left\{ \sum_{l=1}^L \Pr(W_j | I^{l^{\text{th}}}) \right\} \quad (21.26)$$

assuming a J -word dictionary (W_1, W_2, \dots, W_J) and L character strings or best state sequences ($I^{1^{\text{st}}}, I^{2^{\text{nd}}}, \dots, I^{L^{\text{th}}}$) of a word image as given by the modified Viterbi algorithm [11]. As the VDHMM gives a rough estimation of the word length, a word-length filter may be used with $\pm 30\%$ of the estimated word length as the filter range to trim the dictionary size. For example, if the estimated word length is less than 6 characters, it may be desirable to have a filter range of ± 2 characters of the estimated length. To calculate W_j^* , the error probabilities of insertion, deletion, and substitution for a certain letter, or a pair of letters for conditional error probabilities, are estimated in advance. We can simplify Eq. (21.26) in the following manner. For example, $\Pr(W_j | I^{l^{\text{th}}})$ in Eq. (21.26) is replaced by

$$\Pr(W_j | I^{l^{\text{th}}}) = w^{l^{\text{th}}} \cdot \text{min_edit_distance}(W_j, I^{l^{\text{th}}}) \quad (21.27)$$

where $w^{l^{\text{th}}}$ is the weight factor for the l^{th} output of the modified Viterbi algorithm. The normalized path probability associated with state sequence $I^{l^{\text{th}}}$ may be used as the weight factor $w^{l^{\text{th}}}$. Alternatively, $w^{l^{\text{th}}}$ may be determined from linguistic reasoning and overall character confusion, since once a string is identified in MVA, its probability need not be considered going further, as this string is already unmasked or given. Such mutual character confusion information is derived empirically by analyzing the output of the HWR system.

The minimum edit distance ($\text{min_edit_distance}()$) between W_j and $I^{l^{\text{th}}}$ may be found using the Levenshtein algorithm[25]. The Levenshtein edit distance function may be used with a custom cost table to calculate the edit distance. The cost table gives the costs for insertion or deletion of a character, or substitution of one character for another. It may be derived from a confusion matrix generated from the symbol probabilities computed from training, or by running the system in recognition mode up through the MVA module using character images as input.

It should be noted here that any other edit distance can be used. For example, the programming approach described in [25] may be used to find the minimum edit distance as well. If the state sequence $I^{l^{\text{th}}}$ exactly matches W_j in the given dictionary, that is, $\text{min_edit_distance} = 0$, this word is said to be directly recognized as W_j . Otherwise, the hypotheses based on the weighted edit distances to all the dictionary words are generated. The simple edit-distance metric could be replaced by a more complex analysis based on linguistic knowledge, especially when the observed word involves ambiguous segmentation.

21.8 Data Creation

A training corpus was assembled consisting of 494,901 Arabic character images for training symbol probabilities and 44,592 Arabic word images for computing segmentation statistics. The training corpus was created using multiple handwriting samples of 62 Arabic words in isolation, specifically selected to cover a majority

of Arabic letter glyphs. These 62 words were selected so as to have a high enough frequency to be familiar to most native speakers, increasing the likelihood that the respondents would write naturally. Our selection criteria required that each word occur at least 500 times in the Arabic Gigaword corpus [12]. Respondents wrote each word six times on a data collection form so that multiple variations from the same writer could be captured.

Writing samples were collected from 95 writers of varying education level, gender, and handedness. These word images were further segmented into character images and associated metadata using JCaptcha, an annotation tool developed at MITRE.

Data from the IFN/ENIT database, used with permission, provided 26,459 Arabic word images of Tunisian village names handwritten by 411 writers. From these word images, 192,420 character images were extracted. A further 302,481 character images were extracted from the word images collected by MITRE, yielding a total of 494,901 character images.

From the character images, the parameters of symbol probabilities and state duration probabilities are estimated. A dictionary is used to extract the initial state, last state, and state transition probabilities. After all these probabilities are obtained, the VDHMM is trained.

For test data, six writers wrote the 250 most frequent words of Arabic twice to create the test set. These 250 word types represent 31 % of the Arabic Gigaword corpus tokens. Thus, we have the basic lexicon of 250 words.

21.9 Experiments

During recognition, an optimal state sequence is obtained for each iteration of the modified Viterbi algorithm described above. If its corresponding letter sequence exists in the given dictionary, this word is said to be the result of direct recognition. After four iterations of the modified Viterbi algorithm, a hypothesis generation scheme will be applied, as described before. Each optimal state sequence is compared to each dictionary word using the Levenshtein edit-distance function previously mentioned. The top 1, 5, and 10 matching dictionary words are proposed as hypotheses.

Both discrete probability modeling and continuous probability modeling for symbol probability were evaluated using a 250-word lexicon and 45 feature vectors. For Arabic HWR where feature vectors are modeled with continuous symbol probability distribution, recognition accuracy is found to be consistently below that with discrete probability distribution. For this reason, only results with discrete symbol probability are described below.

Two VDHMM-based systems were built to test against two distinct sets of data: one is privately developed and the other is the well-known IFN/ENIT data. Henceforth, these two systems are called ML system and IFN system, respectively. The

Table 21.1 Recognition results in percent correct on MITRE datasets

Writer	Set	Lexicon Size	Top 1	Top 5	Top 10
A	1	250	65	80	86
		12,000	44	56	61
	2	250	65	78	84
		12,000	46	58	60
B	1	250	55	72	79
		12,000	32	45	51
	2	250	60	73	82
		12,000	39	52	56
C	1	250	49	66	73
		12,000	29	42	48
	2	250	49	66	75
		12,000	26	38	44

Table 21.2 Recognition results in percent correct on IFN datasets

	Lexicon Size	Top 1	Top 5	Top 10
Set d	937	66	85	90
Set e	937	60	75	81

systems differ with respect to the training data used to build the character models, the cost table used in post-processing, and the lexicons used. The ML system is built from approximately 163,000 character training images extracted from 24,912 word images (of 62 unique words that provide maximal character variation) in the MITRE corpus. This ML system is tested against sets of word images from three writers not seen in the training data. These sets consist of the 250 most common words in the Arabic Gigaword corpus, written twice by all writers. Two lexicons of size 250 and 12,000 are used. Instead of providing an average performance score, detailed results from three sets of increasing difficulty are provided in Table 21.1. Writer C is judged as having handwriting that is the least legible. The IFN system is trained using 90,000 characters extracted from IFN datasets A, B, and C. Word recognition experiments are performed on sets D and E (Table 21.2).

In addition to these experiments, the system was also submitted to the ICDAR 2007 Arabic handwriting recognition competition. Along with sets d and e from the IFN database, two new sets of data unseen by the participants were tested. Set f was collected from new writers from the same region as d and e (Tunisia), and set s was collected from writers from the United Arab Emirates (UAE). The results are shown in Table 21.3. As expected, we achieved similar results on sets d and e as in our own experiments. Likewise, it is no surprise that we achieved a similar result on set f. We lost about 12 % on set s, but that was about the average loss by all systems.

Table 21.3 Recognition results in percent correct at ICDAR 2007 competition

System	Set d	Set e	Set f			Set s		
	Top 1	Top 1	Top 1	Top 5	Top 10	Top 1	Top 5	Top 10
VDHMM	63.34	64.89	61.70	81.61	85.69	49.91	70.50	76.48
Best	94.58	87.77	87.22	94.05	95.42	73.94	85.44	88.18

Table 21.4 Recognition accuracy with and without segmentation errors

	With Errors		Without Errors		Total	
Right	227	66 %	434	84 %	661	77 %
Wrong	117	34 %	80	16 %	197	23 %
Total	344	100 %	514	100 %	858	100 %

Clearly, our system did not perform as well as the best system presented at the ICDAR 2007 competition. Since then, we have analyzed our system, and our analysis is given below.

21.10 Analysis of Segmentation Errors

To gauge the effect of segmentation errors on recognition rate, we analyzed 858 images from 104 writers. We defined six types of errors:

- 0 none
- 1 order
- 2 bad segmentation point
- 3 over-segmentation
- 4 under-segmentation
- 5 writer error

Table 21.4 shows a summary of the segmentation error rates for images with and without errors. It shows that segmentation errors are indeed significant. For images without errors, we have a recognition rate of 84 %. For those with errors, the recognition rate drops to 66 %.

Table 21.5 shows a breakdown of error rates by error type. The most common type of error is type 1. We have a 67 % recognition rate on images having at least one segment out of order. The worst case is for type 5 errors. It is understandable that we would only have a 50 % recognition rate when the writer makes a mistake.

Table 21.6 shows the breakdown by number of errors. Most of the images having errors had only one, but we see that even a single segmentation error drops the recognition rate down to 68 %.

We also tabulated error rates by writer and error type. We found that there was not a significant set of writers on which we performed poorly. We conclude that writer quality or style did not contribute to segmentation error rate.

Table 21.5 Recognition accuracy versus segmentation error type

Error Type	Right		Wrong		Total	
0	434	84 %	80	16 %	514	52 %
1	180	67 %	89	33 %	269	27 %
2	9	53 %	8	47 %	17	2 %
3	29	58 %	21	42 %	50	5 %
4	71	59 %	50	41 %	121	12 %
5	5	50 %	5	50 %	10	1 %
Total	728	74 %	253	26 %	981	100 %

Table 21.6 Recognition accuracy versus segmentation error count

Error Count	Right		Wrong		Total	
0	434	84 %	80	16 %	514	60 %
1	173	68 %	81	32 %	254	30 %
2	43	67 %	21	33 %	64	7 %
3	9	43 %	12	57 %	21	2 %
4	2	67 %	1	33 %	3	1 %
5	0	0 %	2	100 %	2	0 %
Total	661	77 %	197	23 %	858	100 %

Of course, there are other sources of error besides segmentation. But these numbers show a strong correlation between segmentation error and recognition failure. The most common type of error is segment ordering, and the most common error count is one. Thus, there is room for improvement in our segmentation algorithm that could possibly fix a majority of these types of errors and significantly improve our recognition rate.

21.11 Feature Vector Analysis

While it appears that some features of this empirically selected feature set are correlated, it has been shown [17, 24] that feature reduction techniques such as principal component analysis (PCA) result in a degradation in performance. Our experiments [17] and analysis showed that we could reduce our feature set from 35 to 28 but at a cost of a 5 % reduction in performance. Since feature computation is very fast, we elected to stay with our set of 45 features. The result is not counter-intuitive. PCA formalism assumes exponential type distributions such as Gaussian distributions. However, in our case the features are mostly discrete; thus the assumption of exponential type distribution is not satisfied. Hence, PCA is not used for feature reduction. Our features are ad hoc in nature. Similar features are used in handwriting

Fig. 21.16 Scanning a word image using an overlapping vertical segment in a raster scan fashion



analysis and recognition due to lack of a coherent and compact model for handwriting creation.

21.12 Character-Based Versus Word-Based Training of HWR Systems

We implemented a system based on implicit segmentation using the Hidden Markov Model Toolkit (HTK), now freely available, in order to replicate the best system reported in the 2007 ICDAR competition. The implicit segmentation is shown in Fig. 21.16; the vertical segment features scan the image in a raster scan fashion with some overlap between two successive segments or frames. For this system, features are computed using the feature computation outlined in [9]. For each character, the Gaussian sum model is used with the number of Gaussian components set to 7 based on experimental results. For such systems, no explicit segmentation is needed, and word images with their “truthed” identities are used to train the system. These systems can be called word-based HWR systems. We will denote this system as the HTK system from now on. This system is able to replicate the best results as reported in the 2007 ICDAR competition using the IFN/ENIT training and test words (see Table 21.3).

We also used the same HTK system trained with the 44,592 images of the MITRE dataset to recognize the test images of Table 21.1. The results and comparison with our VDHMM-based system are shown in Table 21.7.

It is clear that the VDHMM system performs better in this case. We provide the following explanation based on the nature of segmentation and the length of a word.

21.12.1 Implicit Versus Explicit Segmentation

While it is true that implicit segmentation is easier to implement and can perform better for IFN-like data, the inherent weakness of this segmentation scheme should also be addressed. In Fig. 21.17, a word image is shown. Here, the stroke of a character spans over much of the next character. An implicit segmentation-based system will not be able to segment the proper character boundary, and training will

Table 21.7 Comparison of character-based versus word-based training of HWR systems

Writer	Set	Lexicon Size	VDHMM			HTK			Number of Images
			Top 1	Top 5	Top 10	Top 1	Top 5	Top 10	
A	1	250	65 %	80 %	86 %	46 %	68 %	75 %	250
		12,000	44 %	56 %	61 %	14 %	30 %	40 %	250
	2	250	65 %	78 %	84 %	46 %	64 %	72 %	250
		12,000	46 %	58 %	60 %	15 %	34 %	43 %	250
B	1	250	55 %	72 %	79 %	47 %	64 %	69 %	195
		12,000	32 %	45 %	51 %	10 %	30 %	37 %	195
	2	250	60 %	73 %	82 %	38 %	61 %	67 %	195
		12,000	39 %	52 %	56 %	12 %	26 %	32 %	195
C	1	250	49 %	66 %	73 %	28 %	50 %	62 %	250
		12,000	29 %	42 %	48 %	2 %	16 %	26 %	250
	2	250	49 %	66 %	75 %	26 %	53 %	59 %	250
		12,000	26 %	38 %	44 %	6 %	16 %	23 %	250

Fig. 21.17 Example of overlapping strokes. Part of “waw” spans over the next character “sheen” and corrupts character training of implicit segmentation-based HMM schemes



train with segments that contain parts of the stroke from previous and/or succeeding characters. Explicit character segmentation and character-based training can overcome this issue. The word images of the 250 most frequent words collected by MITRE contain many examples of stroke overflow, which is detrimental for the HTK system. Additionally, for short words, the explicit segmentation algorithm described here tends to have fewer segmentation errors. This explains why VDHMM and explicit segmentation can outperform the HTK system on some datasets while underperforming for IFN data.

21.12.2 Long Versus Short Words

The VDHMM system has no built-in normalization against the number of characters in a word image. The system is based on single contextual HMM (SCHMM), where only one dictionary is used [18] during recognition for any observation sequence. The state transition probability is computed from one dictionary. Thus, these probabilities are all less than 1. When many of these probabilities are multiplied, as is the case for long words, this probability tends to be smaller. However, when a number

of segments are combined to create one character, there is a built-in normalization incorporated in the VDHMM system given by Eq. (21.12). The net result is a bias toward short words in the lexicon. Since the 250 most frequent words are usually short, the VDHMM system performs better in this situation.

It should be noted here that the VDHMM-based system can also be implemented using one model per word, as is the case with the HTK system. In this case, the VDHMM-based system is scored against each word, treating that word as the only word in the lexicon. Then scores against all the words in the dictionary can be sorted to find the best score. This is possible because states are explicitly identified with the underlying characters. A little analysis would reveal that such a system does not suffer from score reduction based on many multiplications of a number less than one. Not surprisingly, for IFN data, the one-model-per-word VDHMM system performed better than the single contextual VDHMM system based on one big lexicon, roughly increasing the recognition accuracy by 5 %. Still, for long words, segmentation errors are more likely, and that explains some of the loss in recognition accuracy.

21.13 Conclusions

In this chapter, we have presented a complete system for off-line Arabic handwritten word recognition based on VDHMM and explicit segmentation. A simple training scheme is proposed to train the VDHMM-based HWR system using handwritten characters. The evaluation of our system is carried out with numerous real-world examples. The reason why simple discrete probability modeling for symbol probability works better than more complex continuous probability modeling is that most features are discrete in nature. A continuous-discrete hybrid model might be a better option that we would like to explore in the future.

While it is true that implicit segmentation is easier to implement and can perform better for IFN-like data, the inherent weakness of such segmentation schemes, especially for overlapping strokes of different characters, is also discussed. In this context, the loss of performance by an explicit segmentation scheme due to segmentation errors, especially for long words, is also discussed. Both paradigms, explicit segmentation and character-based training vis-à-vis implicit segmentation and word-based training, have their relative strengths and weaknesses which we have outlined here. We would also like to explore this issue more in the future.

Acknowledgements We would like to thank Linda Van Guilder, Ben Huyck, and Jon Phillips for their work on the VDHMM system, and Jon again for his work on the HTK system.

References

1. Abuhaiba, I.S.I., Holt, M.J.J., Datta, S.: Processing of off-line handwritten text polygonal approximation and enforcement of temporal information. *CVGIP Graph. Model Image Process.* **56**(4), 324–335 (1994)

2. Al-Badr, B., Mahmoud, S.: Survey and bibliography of Arabic optical text recognition. *Signal Process.* **41**, 49–77 (1995)
3. Alma'aadeed, C.H.S., Elliman, D.: Recognition of offline handwritten Arabic words using hidden Markov model approach. In: Proceedings of 16th International Conference on Pattern Recognition, vol. 3, pp. 481–484 (2002)
4. Almuallim, H., Yamaguchi, S.: A method of recognition of Arabic cursive handwriting. *IEEE Trans. Pattern Anal. Mach. Intell.* **9**(5), 715–722 (1987)
5. Amin, A.: Arabic character recognition. In: Bunke, H., Wang, P.S.P. (eds.) *Handbook of Character Recognition and Document Image Analysis*, pp. 397–420. World Scientific, Singapore (1997)
6. Amin, A.: Off-line Arabic character recognition: the state of the art. *Pattern Recognit.* **31**(5), 517–530 (1998)
7. Amin, A., Al-Sadoun, H., Fischer, S.: Hand printed character recognition system using artificial network. *Pattern Recogn.* **29**(4) (1996)
8. Amin, A., Mari, J.: Machine recognition and correction of printed Arabic text. *IEEE Trans. Syst. Man Cybern.* **19**(5), 1300–1306 (1989)
9. Bazzi, I., Schwartz, R., Makhoul, J.: An omni-font open vocabulary system for English and Arabic. *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(46), 482–494 (1999)
10. Chen, M.Y., Kundu, A., Srihari, S.N.: Variable duration HMM and morphological segmentation for handwritten word recognition. *IEEE Trans. Image Process.* **4**(12), 1675–1688 (1995)
11. Chen, M.Y., Kundu, A., Zhou, J.: Off-line handwritten word recognition using a hidden Markov model type stochastic network. *IEEE Trans. Pattern Anal. Mach. Intell.* **16**(5), 481–496 (1994)
12. Graff, D.: Arabic Gigaword LDC2003T12. CD-ROM. Linguistic Data Consortium, Philadelphia (2003)
13. Hamdani, M., Abed, H.E., Hamdani, T.M., Märgner, V., Alimi, A.M.: Improving a HMM based offline handwriting recognition system using MME-PSO optimization. In: Proceedings of SPIE Document Recognition and Retrieval, XV111, Conference, vol. 7874, p. 787408 (2011)
14. Kanoun, S., Ennaji, A., Lecourtier, Y., Alimi, A.M.: Linguistic integration information in the AABTAS Arabic text analysis system. In: Proc. of International Workshop on Frontiers in Handwriting Recognition (IWFHR), Aug. 2002, vol. 8, pp. 389–394 (2002)
15. Khorsheed, M.S., Clocksin, W.F.: Off-line Arabic word recognition using a hidden Markov model. In: Statistical Methods for Image Processing—A Satellite Conference of the 52nd ISI Session, Uppsala (1999)
16. Khorsheed, M.S., Clocksin, W.F.: Multi-font Arabic word recognition using spectral features. In: Proc. of International on Pattern Recognition, vol. 4, pp. 543–546 (2000)
17. Kundu, A., Phillips, J., Hines, T., Huyck, B., Van Guilder, L.C.: Arabic handwriting recognition using variable duration HMM. In: Proceedings of International Conference of Document Analysis and Recognition (ICDAR), pp. 644–648, Brazil, Sep. 2007
18. Llolje, A., Levinson, S.E.: Development of an acoustic-phonetic hidden Markov model for continuous speech recognition. *IEEE Trans. Signal Process.* **39**(1), 29–39 (1991)
19. Lorigo, L.M., Govindaraju, V.: Offline Arabic handwriting recognition: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(5) (2006)
20. Obaid, A.M.: Arabic handwritten character recognition by neural nets. *J. Commun.* **45** (1994)
21. Pechwitz, M., Märgner, V.: Baseline estimation for Arabic handwritten words. In: Proc. of International Workshop on Frontiers in Handwriting Recognition (IWFHR), Aug. 2002, vol. 8, pp. 479–484 (2002)
22. Pechwitz, et al.: HMM based approach for handwritten Arabic word recognition using the IFN/ENIT-database. In: Proc. of 7th International Conference on Document Analysis and Recognition (ICDAR) (2003)
23. Rabiner, L.: A tutorial on HMM and selected applications in speech recognition. *Proc. IEEE* **77**(2), 257–286 (1989)

24. Shlens, J.: A tutorial on principal component analysis. <http://www.cs.cmu.edu/elaw/papers/pca.pdf> pp. 1–13 (2007)
25. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. *J. ACM* **21**, 168–173 (1974)
26. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)

Chapter 22

Online Arabic Databases and Applications

**Houcine Boubaker, Abdelkarim Elbaati, Najiba Tagougui, Haikal El Abed,
Monji Kherallah, and Adel M. Alimi**

Abstract Large databases were developed for handwriting recognition in Latin script. In contrast, very few databases have been developed for Arabic script, and fewer have become publicly available. This paper describes a pilot study in which we present the nature of the Arabic handwritten language and the basic concepts behind the recognition process. An overview of online Arabic databases and applications presented in the literature is discussed in detail. We also present some related works using these databases.

22.1 Introduction

In the last few years, handwriting analysis and recognition has become a paramount subject of researchers' interest. The validation of the work done in this area was successfully established, thank to the databases used. Two sorts of databases are considered. One type is of interest to online studies (e.g., UNIPEN), and the other is of interest to offline studies (CEDAR, IRONOFF, NIST, IFN/ENIT, etc.). All these

H. Boubaker (✉) · A. Elbaati · N. Tagougui · M. Kherallah · A.M. Alimi
Research Group on Intelligent Machines (REGIM), National School of Engineers ENIS,
University of Sfax, BP 1173, Sfax 3038, Tunisia
e-mail: houcine-boubaker@ieee.org

A. Elbaati
e-mail: abdelkarim.elbaati@ieee.org

N. Tagougui
e-mail: najiba.tagougui@ieee.org

M. Kherallah
e-mail: monji.kherallah@ieee.org

A.M. Alimi
e-mail: adel.alimi@ieee.org

H. El Abed
Institute for Communications Technology (IfN), Technische Universität Braunschweig,
Schleinitzstrasse 22, 38116 Braunschweig, Germany
e-mail: elabed@tu-bs.de

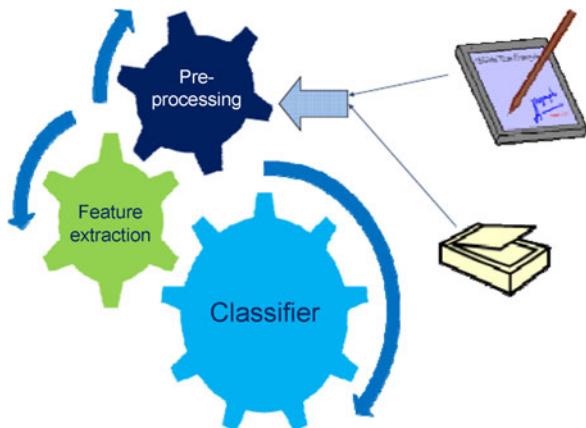
databases are important for the research community in order to test new ideas and algorithms and to perform benchmarks and thereby measure progress and general tendencies.

Since 1997, Märgner et al. have discussed the general concept of benchmarking the output of each module of interest. Their method describes how to build a database with specific ground truth for document analysis systems (DASs) where they focus on the definition and generation of ground truth, especially for the image processing modules of a DAS. They also presented a method to build a database for benchmarking more generally with the use of synthetic data [18]. There exist no freely or commercially available (or Internet accessible) databases of Arabic characters, digits and words in either offline topics or in online topics. An interesting state of the art of Arabic language was established by [7], indicating the difference between the handwritten classical Arabic script as the courant and the modern standard one which represents the majority of use in Arabian countries. The author also gives a summary of techniques concerning Arabic handwriting recognition research. In online studies of the handwriting research topic, Alimi [2] presented a review of online Arabic handwriting recognition systems. He developed an evolutionary neuro-fuzzy approach to recognize Arabic handwritten characters. His experiment consists in testing the performance of the system to recognize Arabic handwritten characters segmented from cursive script. From this task the same writer was asked to write a text extracted from a newspaper containing about 1000 words. These words contain more than 3000 characters (almost all the possible combinations of the 117 Arabic letters were used). The written words were segmented manually into characters, and only the principal component of each character was kept [1, 2]. Mezghani et al. [19] have elaborated a set of 17 basic Arabic isolated letters with 432 samples of each character written by 18 writers. Their experience deals with an online recognition system carried out by a Kohonen neural network trained using an empirical distribution of features such as tangents and tangent differences at regularly spaced points along the character signal. El-Sana presented a recent work which deals with an on-line Arabic handwriting recognition field of the disclosed technique [8]. The method of recognition incorporates delayed strokes and uses a discrete hidden Markov model (HMM) to represent each of the letter shapes in the Arabic alphabet [3]. The dataset used contains between 30,000 and 40,000 Arabic words written without dots.

As a result, until now, there has been no robust standard comprehensive database online or offline for Arabic handwriting script recognition. However, some attempts have been realized, and one of the first databases that was publicly available and became the first standard databases for Arabic is the IFN/ENIT [7] which is an off-line database for Arabic words including 937 Tunisian town/villages names and postal codes written by 411 people. A Persian version of the IFN/ENIT was recently released, including city names handwritten in Farsi. The Persian version consists of 7271 binary images of 1080 Iranian province/city names, collected from 600 writers. For each image in the database, the ground truth information includes its zip code, and a sequence of characters and numbers.

The need to advance Arabic online handwriting recognition systems drives the research community to create and collect online Arabic databases. The validation of

Fig. 22.1 Online/offline handwriting recognition process



the work done in this area cannot be successfully established without common international databases. The objective of this paper is to present standard online Arabic databases which will be important for the research community working in Arabic in order to test new ideas and algorithms and to perform benchmarks and thereby measure progress and general tendencies. This chapter is organized as follows. In Sect. 22.2, the state of the art of Arabic handwriting recognition will be presented. Section 22.3 briefly presents the Arabic script. Section 22.4 presents in detail our LMCA (Lettres, Mots et Chiffres Arabe) database formulation, and to prove the validity of LMCA's structure, some related works will be presented in subsections. In the same way, Sect. 22.5 will be devoted to the ADAB database formulation and to some related works proving its validity. In Sect. 22.6, a conclusion will be presented.

22.2 State of the Art of Arabic Handwriting Recognition

Two axes of research are available in handwriting recognition; the first one is called online, and the second offline. According to Fig. 22.1, using a digital tablet and a special pen offers an interactive dynamic information as a sequence of point coordinates. Using the scanner offers static information as pixels.

The recognition concerns handwritten characters or handwritten words. Three phases are needed for recognition system approval: pre-processing, feature extraction, and classification phases. The advantage of the IRONOFF database is that an offline image and an online trajectory are available. One interest concerns the evaluation of skeleton algorithms. Here, the online data could provide a way to compare the skeleton points (offline image) to an objective trajectory (online coordinates). One could also study the correlation that could exist between the speed of the pen and the gray level distribution or the width of the corresponding strokes. If the online data is jointly accessible with the offline images, it can be used to recover the temporal order of strokes from the offline images and thereby guide and train the segmentation to provide a relevant frame description.

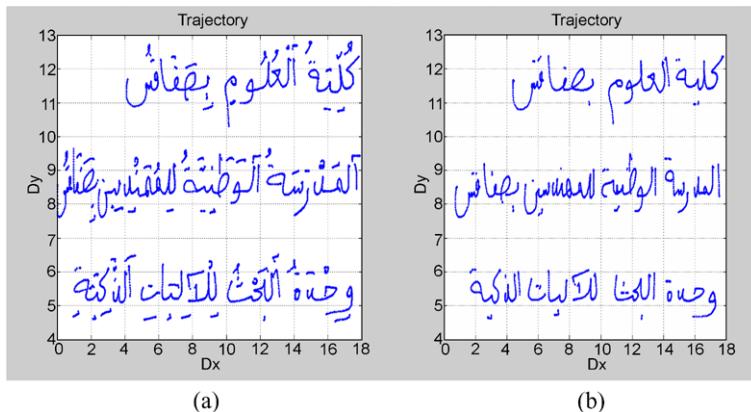


Fig. 22.2 Arabic handwriting

In that sense, these approaches bridge the gap between online and offline character recognition methods [11, 16], which is a very attractive concept since it has been shown that online handwriting exhibits superior results compared to offline recognition [20].

22.3 Arabic Script Description

Arabic script has been adopted for use in a wide variety of languages other than Arabic, including Persian, Kurdish, Malay, and Urdu. Arabic handwriting is a consonantal and cursive writing. This property is exhibited in two forms: printed or handwritten documents. There are no distinct upper and lower case letter forms. Some Arabic handwritten documents are written with some diacritics (see Fig. 22.2(a)), whereas in the majority of cases only points are considered in Arabic handwriting (see Fig. 22.2(b)). The Arabic alphabet is composed of 28 main characters (with diacritics and in isolated form) and is written from right to left. Most characters have four different shapes. The difference between these letters lies in their positions in the word, the number and the position of the diacritic dots, and the presence of the "Hamza" and vowels (see Table 22.1). In fact, the majority of letters change slightly in shape according to their position in the word (initial, medium, or final). This change occurs when letters are either joined to one another or isolated. There is also a big similarity between some letters [3]. In our work, we reduce the number of letters to 57 by eliminating all diacritics as points and vowels. If we do not consider the first letter of Table 22.1, "Hamza," the number will be reduced to 56 letters.

Table 22.1 represents the 28 letters of the Arabic script in their four different forms. Among them, six letters exist only in the isolated form and in the end form. They are marked with empty columns.

Table 22.1 The Arabic alphabet

Character	alone	end	middle	begin	Character	alone	end	middle	begin
Alif	ا	ا			Dhad	ض	ض	ض	ض
Ba	ب	ب	ب	ب	Taa	ط	ط	ط	ط
Ta	ت	ت	ت	ت	Dha	ظ	ظ	ظ	ظ
Tha	ث	ث	ث	ث	Ayn	ع	ع	ع	ع
Jim	ج	ج	ج	ج	Ghayn	غ	غ	غ	غ
Ha	ح	ح	ح	ح	Fa	ف	ف	ف	ف
Kha	خ	خ	خ	خ	Qaf	ق	ق	ق	ق
Dal	د	د			Kaf	ك	ك	ك	ك
The	ذ	ذ			Lam	ل	ل	ل	ل
Ra	ر	ر			Mim	م	م	م	م
Zai	ز	ز			Nun	ن	ن	ن	ن
Sin	س	س	س	س	He	ه	ه	ه	ه
Chin	ش	ش	ش	ش	Waw	و	و	و	و
Sad	ص	ص	ص	ص	Ya	ي	ي	ي	ي

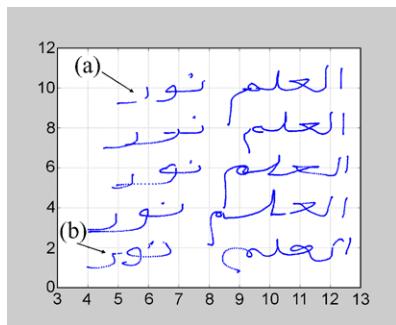
22.3.1 Diacritic Symbols Influence

Some Arabic letters have the same form. However, they are distinguished from each other by the addition of dots in different positions relative to the main stroke. Some Arabic characters use special marks to modify the character accent. When diacritical symbols (dots, special marks) are used, they appear above or below the characters and they are drawn as isolated entities as shown in Fig. 22.2. Diacritical symbols are positioned at a certain distance from the character, which makes some difficulties in separating the border of a text line. Indeed, diacritical symbols can generate some redundant separate lines [3]. We count 15 among the 28 letters of the alphabet which contain dots. Some letters present a zigzag shape called “Hamza.” It takes the same shape of the letter “Ayn” (see Table 22.1), but it is located above the letter “Alif.” The letter “Hamza” is considered as an accent “vowel” in the Arabic alphabet [17].

22.3.2 Pre-processing Step

Pre-processing is primarily related to word processing operations such as normalization to remove handwriting irregularities. Most of the current Arabic word

Fig. 22.3 Errors and multivariability existing between writers. (a) Stiction of pen-down switchers (bad contact). (b) Disconcerted and redundant points



recognition systems do not allow noisy data input. Therefore, current research must deal with the matters of multi-cultural handwriting styles and the adaptation method, which varies from one user to another. In a project on within-writer and between-writer variability, it was found that the number of stroke-shape interpretations in cursive script kept increasing with each new writer in a training system, and the existence of an asymptote was not apparent. The major problems caused by the multivariability are: The input may consist of discrete noise events, like dots or short lines resulting from inadvertently dropping the pen or tapping the pen on the writing surface unwillingly (see Figs. 22.3(a) and 22.3(b)). The input may consist of badly formed shapes, illegible to both humans and machines. Two successive points can be confused by a small segment (see Fig. 22.2(b)). Consequently, the handwriting input may contain device-generated errors: random noise, stiction of pen-down switches, unresponsiveness of switches, pen tilt errors, etc. These problems are shown in Fig. 22.3. In most cases, Arabic writing does not use vowels. The sense of the word is often determined by the context of the sentence. Thus vowels are not considered in our work. We corrected the trajectory by eliminating the majority of noise (diacritics as points and short segments). For this task we developed a simple filter based on distance measurement between successive points of the trajectory. The elimination of isolated points and small segments composed of a number of points is based on threshold optimization. In Figs. 22.4(a) and 22.4(b) we demonstrate the filter effectiveness.

22.4 LMCA Database and Related Works

22.4.1 Database Formulation

A database for character recognition algorithms is of fundamental interest for the training of recognition methods. We developed our own database which contains 30,000 digits, 100,000 Arabic letters, and 500 Arabic words. This database was developed in our laboratory, the REsearch Group on Intelligent Machines (REGIM).

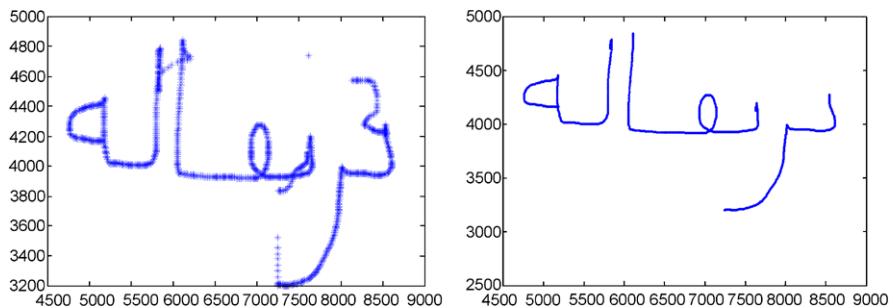


Fig. 22.4 The word “bortoukalaton” (orange) before (a) and after (b) smoothing



Fig. 22.5 (a) Examples of scanned written words. (b) Examples of scanned written digits

Both online/offline handwritten characters and words are considered. The online procedure is based on a collection of coordinates (x, y) of the handwritten trajectory, whereas the offline procedure is based on a collection of images of the handwritten trajectory. These two types of information should be available within the same coordinate system, with the same origin and the same resolution and orientation.

Fifty-five participants were invited to contribute to the development of the handwritten LMCA. The dataset of words of each participant is stored in one data file. When producing the data file, each participant was asked to write some Arabic words. We collected 500 words written by different writers. The data for each participant are stored in one data file. For the digits dataset construction, a participant was asked to write a set of all digits (1000 to 1500 samples of digits). We imposed that the writer should just write the same digit ten times, from 0 to 9, on the same page. One page contains 100 digits. The writer was asked to prepare only one page per day. We have collected 30,000 digits in total. More than half of them are regularly written. The remaining ones are those that have noise in the data, are poorly written, or are deliberately written in strange and unusual ways.

Figures 22.5(a) and 22.5(b) present an example of scanned words and digits. They are presented as an image in JPG format. The same procedure was applied to prepare 100,000 Arabic letters. About two-thirds of the writers were male, about 90 percent were right handed, the youngest writer was 8 years old, and the oldest was

Fig. 22.6 The graphical user interface “Handwriter”



Fig. 22.7 The x and y coordinates from the digital tablet

X	Y	Z
9009	6395	0
9012	6400	0
9014	6405	0
9017	6409	0
9019	6414	0
9020	6418	1
9020	6418	1
9020	6422	1
9020	6422	1
9020	6422	1

66. In the online domain, the forms were sampled with a spatial resolution of 200 dpi and a sampling rate of 100 points/s (Wacom UltraPad A4) and were stored using the UNIPEN format. To collect data, a graphical user interface called “Handwriter” has been developed on a PC/NT window environment (see Fig. 22.6). The online information of the handwriting is kept in a text file. The pen position up and down is detected, respectively, by 0 and 1 values. The trajectory of the handwritten script is collected as coordinates of x and y from the digital tablet (see Fig. 22.7).

22.4.2 Related Works

Online Digit Handwriting Recognition System Based on Trajectory and Velocity Modeling

Digit recognition was studied ten years ago, and it was found that the fuzzy approach enhanced the classification performance. In this study, the feature extraction system was based on the “beta-elliptical” representation [14]. One of the main classification problems is the variability of the feature vector size depending on each digit number of strokes. The recognition process is divided into pre-processing steps and a subsequent classification. To face the complex problems of handwriting recognition, the use of multiple, hybrid and an association of classifier systems has attracted increasing interest during recent years. Based on their complementarities, an association of

Fig. 22.8 Multiple classifier system

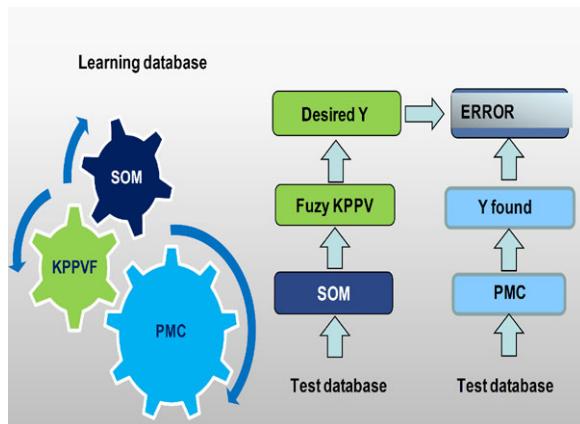


Table 22.2 Comparative study between UNIPEN digit dataset and LMCA digit dataset

Classifier	Modeling system	Dataset	Recognition rate
MLP	Beta-elliptical	30,000 digits LMCA	94.14 %
SVM	Beta-elliptical	Digit set of UNIPEN	94.78 %

classifiers increases the performance of the recognition system while limiting the error bound to the use of a unique classifier. The use of the multiple classifier systems benefits from the strong points of every classifier. In [14], the recognition system is based on the use of neural networks developed in a fuzzy concept. The desired outputs of a multilayer perceptron neural network (MLPNN) are formed using a self-organizing map (SOM) and a fast Kohonen neural network (FKNN) algorithm (see Fig. 22.8). Therefore, this system involves neuro-fuzzy networks based on a SOM and FKNN algorithm association used in the learning process [14]. The global recognition rate obtained is about 95.08 %. When testing our system, the global average squared error obtained is about 0.065. In this study, our aim is to validate the use of a digit set extracted from the LMCA dataset. It is known that the MLP and support vector machine (SVM) techniques give the same performance. The first experience was based on the use of the LMCA digit dataset, whereas the second one was based on the use of the UNIPEN digit dataset. The results obtained were similar, which proves that the developed LMCA digit dataset has a correct format benchmark (see Table 22.2).

Recognition of a Handwritten Arabic Word Based on Visual Encoding and GA

A handwritten word is represented by a continuation of visual codes of Arabic letters. In this case the order of these letters is considered. We attribute N the number of basic letters extracted from a cursive word [15]. Therefore, every gene of the

Table 22.3 Fitness value calculation

Visual indices	Va "1"	Lo "2"	Po "3"	Al "4"	Des "5"	As "6"	Roc "7"	Loc "8"	Loa "9"	Roa "10"	Ain "11"	Sad "12"	# "13"
Va "1"	0	1	1	1	1	1	1	1	1	1	1	1	1
Lo "2"	1	0	1	1	1	1	0.5	0.5	1	1	0.5	0.5	1
Po "3"	1	1	0	1	0.5	1	1	1	1	1	1	1	1
Al "4"	1	1	1	0	1	0.5	1	1	0.5	0.5	1	1	1
Des "5"	1	1	0.5	1	0	1	1	1	1	1	1	1	1
As "6"	1	1	1	0.5	1	0	1	1	0.5	0.5	1	1	1
Roc "7"	1	0.5	1	1	1	1	0	1	1	1	0.5	0.5	1
Loc "8"	1	0.5	1	1	1	1	1	0	1	1	0.5	0.5	1
Loa "9"	1	1	1	0.5	1	0.5	1	1	0	0.5	1	1	1
Roa "10"	1	1	1	0.5	1	0.5	1	1	0.5	0	1	1	1
Ain "11"	1	0.5	1	1	1	1	0.5	0.5	1	1	0	0.5	1
Sad "12"	1	0.5	1	1	1	1	0.5	0.5	1	1	0.5	0	1
# "13"	1	1	1	1	1	1	1	1	1	1	1	1	0

population has N chromosomes and every chromosome has one of the 58 possible values (1 to 57 for the basic Arabian characters and the value 0 for characters with more than one visual indication) numbered from the right to the left. The extraction rate obtained is about 72 %. However, in the second stage, which consists in correcting the weaknesses of the previous method, we developed a genetic algorithm (GA) in order to select the best combination of visual codes extracted from a word by the heuristic method [13]. The GA approach permits the recognition of cursive handwriting without the limitation of a lexical dictionary [12]. Therefore, the convergence of the GA is ensured by the technique given in the fitness function which consists in the use of the visual codes of Arabic words and the comparison method established between the visual indices strings according to Table 22.3. The number of generations (500) and the fitness value (0.5) were fixed as a convergence condition criterion. If the population size was fixed to 100 individuals, the recognition rate was about 99.85 %. These results are encouraging. In this experiment we used the 500 words and the 57 Arabic letters extracted from the LMCA dataset. 200 words were used as data prototypes for the selection of the initial population of the GA, and the others were used for testing our system.

Order Temporal Reconstruction from Arabic Image Word

The word image captured in gray level with a resolution of 300 dpi will be pre-processed in four stages: binarization, filtering, extraction of the skeleton, and elimination of the diacritical signs (see Fig. 22.9(a)). A suitable algorithm segments the skeleton in three types of segments: segments of connection, occlusion, and segments of end of stroke. The starting segment is localized by sweeping the image of the skeleton from the right to the left, and more tests are applied. Another algorithm makes it possible to order these segments based on heuristic rules. These rules count on the fact that Arabic script is written from right to left, and they take into account the natural order of stroke generation [10]. To validate this approach we tested it on a whole of the words extracted from the LMCA dataset. The temporal order signal which is reconstructed will be compared with its original online trajectory signal (see Fig. 22.9(b)) [9].

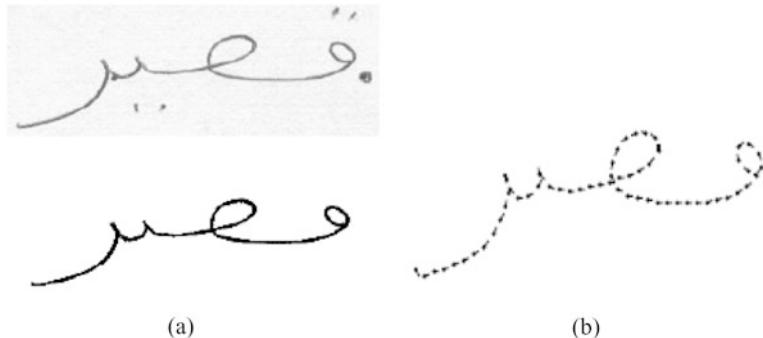


Fig. 22.9 (a) The Arabic word “kassiron” before and after pre-processing. (b) Restoration of the temporal order of the offline word “kassiron”

22.4.3 Conclusion

The different research works and their results prove that our LMCA database can be used in both modeling and recognition systems of Arabic handwriting. The related works presented prove also that LMCA is a standard database and it has the same format of the common UNIPEN or IRONOFF database. Our perspective is to increase the number of writers of LMCA to help make it perform for any techniques of modeling and classification of handwritten Arabic script.

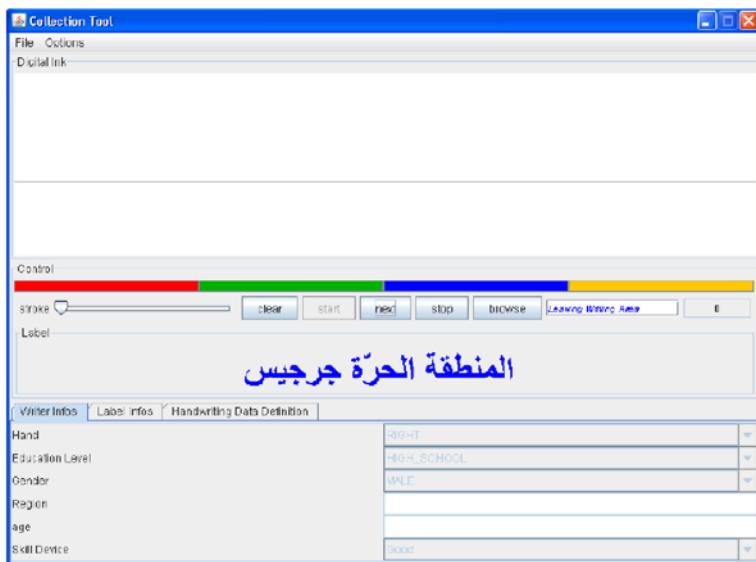
22.5 ADAB Formulation and Related Works

22.5.1 ADAB Formulation

The Arabic DAtaBase (ADAB) was developed to advance the research and development of Arabic online handwritten text recognition systems. This database is developed as a cooperative effort between the Institut fuer Nachrichtentechnik (Ifn) and the National School of Engineers of Sfax (ENIS), Research Group on Intelligent Machines (REGIM) [7]. The database consists of 19,575 Arabic words handwritten by more than 150 different writers, most of them selected from the narrower range of ENIS. The text that is written is from 937 Tunisian town/village names. We plan to extend this database with other Arabic writing styles. For this reason, we have developed special tools for the collection of the data and verification of the ground truth, which will be available for other groups for the collection of their own data in the same form of the ADAB. These tools allow one to record the online written data, to save some writer information, to select the lexicon for the collection, and to rewrite and correct wrong written text. Ground truth was added to the text information automatically from the selected lexicon and verified manually. The ADAB is freely available for noncommercial research

Table 22.4 ADAB sets

Set	Files	Words	Characters	Writers
1	5037	7670	40,500	56
2	5090	7891	41,515	37
3	5031	7730	40,544	39
4	4417	6786	35,832	25
Sum	19,575	30,077	158,420	157

**Fig. 22.10** The ADAB's collection tool

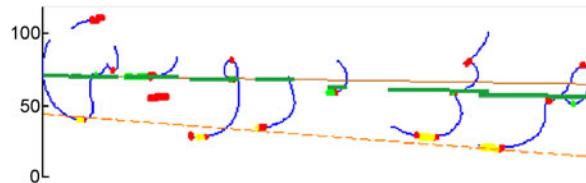
(www.regim.org) [7]. Our aim was to collect a database of handwritten town names written in a quality similar to that on a mobile phone with a digital input device. The collection process starts when the writer clicks on start bottom. The collection tool generates a town name randomly from 937 Tunisian town/village names, and the writer must write the displayed word (see Fig. 22.10). A pre-label will be automatically assigned to each file. It consists of the postcode in a sequence of numeric character references which will be stored in the UPX file format. An InkML file including trajectory information and a plot image of the word trajectory is also generated. Additional information about the writer can also be provided (see Fig. 22.11).

The ADAB is divided into four sets. Details about the number of files, words, characters, and writers for each set 1 to 4 are shown in Table 22.4.

Nom	Taille	Type	Nom	Taille	Type	Nom	Taille
1267267556453	2 Ko	Fichier UPX	1267267556453.inkml	3 Ko	Fichier INKML	1267267556453	2 Ko
1267267583812	2 Ko	Fichier UPX	1267267583812.inkml	5 Ko	Fichier INKML	1267267583812	6 Ko
1267268596625	2 Ko	Fichier UPX	1267268596625.inkml	4 Ko	Fichier INKML	1267268596625	4 Ko
1267268790859	2 Ko	Fichier UPX	1267268790859.inkml	5 Ko	Fichier INKML	1267268790859	3 Ko
1267268813984	2 Ko	Fichier UPX	1267268813984.inkml	5 Ko	Fichier INKML	1267268813984	4 Ko
1267268877562	2 Ko	Fichier UPX	1267268877562.inkml	6 Ko	Fichier INKML	1267268877562	4 Ko
1267268895250	2 Ko	Fichier UPX	1267268895250.inkml	5 Ko	Fichier INKML	1267268895250	6 Ko
1267268926328	2 Ko	Fichier UPX	1267268926328.inkml	6 Ko	Fichier INKML	1267268926328	6 Ko

Fig. 22.11 Samples of UPX files and their corresponding InkML and image files

Fig. 22.12 Example of detected baseline correction (green) obtained from the consideration of topological conditions



22.5.2 Related Works

Online Arabic Handwriting Modeling System Based on Grapheme Segmentation

An online Arabic handwriting modeling system based on grapheme segmentation is presented. The system consists of three modules: detection of the baseline, grapheme segmentation, and feature extraction. The method developed in the first module is distinguished by the consideration of geometrical and topological features for the baseline detection and correction. In the second module, we use the detected baseline to check particular points: the back of the valleys and the angular points for the segmentation of the cursive handwriting trajectory in graphemes. The third module extracts parameters to model the position, the shape, and the fuzzy affectation rate of diacritics associated to each segmented grapheme. Figure 22.12 shows an example of a correct result of baseline detection obtained from the consideration of the topologic conditions compared with the results of the basic stage [5].

In the evaluation phase, the system is applied on the online database ADAB of Tunisian town names using the HMM Toolkit (HTK) as the classification module. The following recognition results for three ameliorated versions of the system were obtained (Table 22.5):

- Version 1: without diacritics detection (ICDAR 2009 competition) [4].
- Version 2: after adjusting the filters and without diacritics detection.
- Version 3: after adjusting the filters and with the extraction and fuzzy affectation of diacritics.

Table 22.5 Recognition rate obtained on ADAB sets 1 and 2

System version	ADAB set 1		ADAB set 2	
	Top 1	Top 5	Top 1	Top 5
Version 1	57.87	72.89	54.26	66.38
Version 2	86.38	96.43	83.55	94.68
Version 3	82.33	93.47	80.61	91.53

ICDAR Competition

The first international online Arabic handwriting recognition competition was held in 2009 [7]. The International Conference on Document Analysis and Recognition (ICDAR) is an international scientific conference in the field of document processing and image analysis. Occurring every two years, it brings together researchers from universities and businesses from all around the world. To compare the performance of the participants' systems, the ADAB was used. As part of the competition, seven systems have been benchmarked by independent leading domain experts. Among the tested recognition systems, several systems have been developed by specialized university laboratories such as the REGIM laboratory. Among the industry players, the Vision Objects MyScript® system proposes natural handwriting recognition technology for Arabic. The systems were tested on known data (sets 1 to 3) and on one test dataset which was unknown to all participants (set 4). The accuracy rate and the recognition speed were measured. With 99 % accuracy rates, the MyScript® system has the highest recognition rates, almost 4 % higher than the second best system participating in the competition. Regarding the recognition speed, Vision Objects won over its competitors hands down with an average processing time of 69 ms per word: more than 25 times faster than the second fastest system in the competition.

The competition results, presented in Table 22.6, show that Arabic handwritten word recognition systems have further made remarkable progress within recent years. Most of the participating systems show a very high accuracy, and some also perform at very high speed [6].

22.5.3 Conclusion

Online recognition of cursive Arabic handwritten words aims to contribute in the evolution of online Arabic handwriting recognition research. Since 2009 the freely available database ADAB is used by some research groups all over the world to develop online Arabic handwriting recognition systems. This database was the basis for the competition ICDAR 2009 for systems that are specialized in online recognition of cursive Arabic handwritten words, which confirms that it is a database with reliable matter.

Table 22.6 Recognition results in % of correct recognized images on reference datasets 1, 2, and 3 and on a subset of dataset 4

System	set 1			set 2			set 3			set 4*		
	top 1	top 5	top 10	top 1	top 5	top 10	top 1	top 5	top 10	top 1	top 5	top 10
MDLSTM-1	99.36	99.94	99.96	99.42	99.96	100.00	99.52	99.94	99.94	95.70	98.93	100
MDLSTM-2	98.55	99.60	99.66	98.77	99.88	99.92	98.89	99.64	99.70	95.70	98.93	100
VisionObjects-1	99.46	99.70	99.70	99.82	99.94	99.96	99.58	99.76	99.76	98.99	100	100
VisionObjects-2	99.29	99.60	99.60	99.51	99.74	99.74	99.26	99.56	99.56	98.99	100	100
REGIM-HTK	57.87	72.89	77.03	54.26	66.38	71.06	53.75	72.31	76.22	52.67	63.44	64.52
REGIM-CV	100	100	100	94.39	96.06	96.06	96.28	97.14	97.52	13.99	31.18	37.63
REGIM-CV-HTK	28.85	51.92	55.77	35.75	58.30	64.26	30.60	52.80	62.80	38.71	59.07	69.89

22.6 Discussion and Future Work

This work was about creating a good quality database with support for online handwritten Arabic script recognition. This type of database has many uses, including training and testing a recognition system. Two databases were created: the first one is the LMCA database which contains 30,000 digits, 100,000 Arabic letters and 500 Arabic words, and the second one is the ADAB, which contains 19,575 samples of 937 Tunisian town/village names. These databases were created by collecting writing contributions from more than 200 Arab persons of different age and sex. The input information can be digits, characters, or words. Many of the contributors expressed that it was unnatural to write on a Wacom or a Genius tablet because during writing the pen's trace isn't visible directly on the tablet as it is on a piece of paper. This could have affected the quality of the collected material and thus the final database. But it is not a real major limit, since the information collected will be used by recognition systems devoted to the recognition of handwritten script on small mobile devices where it is impossible to treat the writing style. To confirm the efficiency of these databases, many works were evaluated, and we have reported the results. The use of the ADAB to test and compare the participating systems in the ICDAR competition proves that it is really a standard database with consistent content. We plan to extend this database with other Arabic writing styles. This database will be important for the research community in order to test new ideas and algorithms and to perform benchmarks and thereby measure progress and general tendencies. We plan to expand these databases so that they became a reference in the field. Our perspective is also to try to expand this data by considering other writers of the Eastern countries.

Acknowledgements The authors thank all participants' contributions to the LMCA and ADAB databases formulation. In addition, they acknowledge the financial support of this work by grants from the General Direction of Scientific Research and Technological Renovation (DGRST), Tunisia, under the ARUB program 01/UR/11/02.

References

1. Alimi, A.M.: A neuro-fuzzy approach to recognize on-line Arabic handwriting. In: Proc. of Int. Conf. on Neural Networks, vol. 3, pp. 1397–1400, August 1997
2. Alimi, A.M.: An evolutionary neuro-fuzzy approach to recognize on-line Arabic handwriting. In: Proc. of the International Conference on Document Analysis and Recognition (ICDAR), pp. 382–386 (1997)
3. Biadsy, F., El-Sana, J., Habash, N.: Online Arabic handwriting recognition using hidden Markov models. In: Proc. of the Tenth International Workshop on Frontiers in Handwriting Recognition (2006)
4. Boubaker, H., Kherallah, M., Alimi, A.M.: New algorithm of straight or curved baseline detection for short Arabic handwritten writing. In: Proc. of the 10th International Conference on Document Analysis and Recognition (ICDAR) (2009)
5. Boubaker, H., El Baati, A., Kherallah, M., El Abed, H., Alimi, A.M.: Online Arabic handwriting modeling system based on the grapheme segmentation. In: Proceedings of the 20th International Conference on Pattern Recognition (ICPR), pp. 2061–2063 (2010)
6. El Abed, H., Kherallah, M., Märgner, V., Alimi, A.M.: ICDAR 2009—Arabic online handwriting recognition competition. In: Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR), vol. 3, pp. 1388–1392, July 2009
7. El Abed, H., Kherallah, M., Märgner, V., Alimi, A.M.: On-line Arabic handwriting recognition competition—ADAB database and participating systems. *Int. J. Doc. Anal. Recognit.* **14**(1), 15–23 (2011)
8. El-Sana, J., Biadsy, F.: On-line Arabic handwriting recognition field of the disclosed technique (2010)
9. Elbaati, A., Boubaker, H., Kherallah, M., El Abed, H., Ennaji, A., Alimi, A.M.: Arabic handwriting recognition using restored stroke chronology. In: Proc. of the International Conference on Document Analysis and Recognition (ICDAR) (2009)
10. Elbaati, A., Kherallah, M., Alimi, A.M., Ennaji, A.: De l'hors-ligne vers un système de reconnaissance en-ligne: application à la modélisation de l'écriture Arabe manuscrite ancienne. In: Proc of the Semaine du Document Numerique, SDN (ANAGRAM) (2006)
11. Jäger, S.: Recovery dynamic information from static, handwritten word images. Ph.D. thesis, Daimler-Benz AG Research and Tech., Verlag Dietmar Fölbach (1998)
12. Jouini, B., Kherallah, M., Alimi, A.M.: A new approach for on-line visual encoding and recognition of handwriting script by using neural network system. In: Proc. of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA), Roanne, France, pp. 161–167 (2003)
13. Kherallah, M., Bouri, F., Alimi, A.M.: Toward an online handwriting recognition system based on visual coding and genetic algorithm. In: Proc. of the International Conference on Adaptive and Natural Computing Algorithms, Coimbra, Portugal, pp. 502–505 (2005)
14. Kherallah, M., Hadded, L., Mitiche, A., Alimi, A.M.: On-line recognition of handwritten digits based on trajectory and velocity modeling. *Pattern Recognit. Lett.* **29**, 580–594 (2007)
15. Kherallah, M., Bouri, F., Alimi, A.M.: On-line Arabic handwriting recognition system based on visual encoding and genetic algorithm. *Eng. Appl. Artif. Intell.* **22**(1), 153–170 (2009)
16. Lallican, P.M., Viard-Gaudin, C.: Off-line handwriting modeling as a trajectory tracking problem. In: Proc. of the 6th International Workshop on Frontiers in Handwriting Recognition (IWFHR), Taejon, Korea, August 1998, pp. 347–356 (1998)
17. Lorigo, L.M., Govindaraju, V.: Offline Arabic handwriting recognition: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(5), 712–724 (2006)
18. Märgner, V., Karcher, P., Pawłowski, A.-K.: On benchmarking of document analysis systems. In: Proc. of the 4th International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 331–336 (1997)

19. Mezghani, N., Mitiche, A., Cheriet, M.: On-line recognition of handwritten Arabic characters using a Kohonen neural network. In: Proc. of the 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR), Niagara-on the-Lake, Ontario, Canada, pp. 490–495. (2002)
20. Seiler, R., Schenkel, M., Eggimann, F.: Off-line cursive handwriting recognition compared with on-line recognition. In: Proc. International Conference on Pattern Recognition (ICPR), Vienna, pp. 505–509 (1996)

Chapter 23

On-line Arabic Handwritten Word Recognition Based on HMM and Combination of On-line and Off-line Features

Sherif Abdelazeem, Hesham M. Eraqi, and Hany Ahmed

Abstract Unconstrained handwritten text recognition is one of the most difficult problems in the field of pattern recognition. This paper presents a new on-line Arabic handwriting recognition system based on hidden Markov models (HMMs). Besides the common use of the off-line features for the HMM-based Arabic recognition systems, we add the use of on-line features and combination of the two approaches. The delta and acceleration features are used as approximations to the derivatives of the observation vectors with respect to time, and they have proved to be very effective in improving the system's performance. Delayed strokes are a well-known problem in on-line handwriting recognition due to its varying writing order among different writers. We solved this problem by removing those delayed strokes by using a new delayed strokes detection approach that makes use of the baseline information and the shape of the strokes. The baseline detection method used in our system is based on horizontal projection. Removing delayed strokes has also led to the ability to combine some of the Arabic characters that share the same primary stroke and are only distinguishable by their delayed strokes into one class (HMM model), which increased the recognition rate. A new algorithm for lexicon reduction based on the detection of the delayed strokes has been developed. The lexicon reduction algorithm is used to improve the system's performance in terms of speed and recognition rate. The on-line database ADAB of Tunisian town names is used for system training and evaluation. We achieved recognition rates up to 97.5 %, which is very promising compared to the highest recognition rate achieved on this database and is significantly higher than the recognition rates achieved by the other HMM-based Arabic handwriting recognition systems.

S. Abdelazeem (✉) · H.M. Eraqi · H. Ahmed

Electronics Engineering Dept., The American University in Cairo (AUC), Cairo, Egypt
e-mail: shazeem@aucegypt.edu

H.M. Eraqi
e-mail: hesham.eraqi@aucegypt.edu

H. Ahmed
e-mail: hanyahmed@aucegypt.edu

Table 23.1 Off-line/on-line approach problems

Approach/Case	Illegal Writing Order (Painting)	Distorted Writing
On-line	×	✓
Off-line	✓	×

23.1 Introduction

In recent years, on-line handwriting recognition systems started to have several applications; mainly due to the increasing popularity of personal digital assistants (PDAs), tablet PCs, and smart phones that use a pen as a convenient and portable input method. There have been significant advancements in the area of handwriting recognition for Latin-based languages. However, Arabic handwriting recognition has received less attention from the research community. The cursive nature of Arabic, characters' overlap "ligatures," and delayed strokes are some of the key problems that make Arabic recognition more difficult than other languages such as Latin or Chinese [10]. More details about Arabic writing characteristics can be found in [10].

Automatic character or text recognition of handwriting can be classified into two approaches: off-line and on-line. In the first approach, off-line recognition, the written word (or letter) is expressed in terms of pixels. In other words, it is treated as an image of the word after it has been acquired. The second approach, on-line recognition, uses the trace of a pen for the classification and recognition of the input information. The difference between the two modes is that the on-line mode provides us with temporal features that are used to infer the dynamics of the writing [1, 4]. The on-line data can still be converted into the off-line form (image) by interpolating the point of every stroke and then applying a Gaussian mask that dilates the writing contour [12].

Each approach (off-line and on-line) has its weak and strong points, whereas the integration between the two approaches has been proven to give a higher performance. This integration tries to make use of the strong points of each approach. The strong points of each approach have been discussed in [18] and are summarized in Table 23.1. The table shows three of the most common optical character recognition (OCR) problems which can be solved by either the off-line or on-line approach while being a big problem for the other approach. The "✓" mark in the table indicates that the approach can tolerate the corresponding problem.

Simultaneous segmentation and recognition has proved to be the key factor in solving the OCR problem for cursively written languages like Arabic [5]. Although using hidden Markov models (HMMs) has proven to be a good simultaneous segmentation and recognition method that solves the segmentation difficulties of the Arabic language, little research progress has been achieved with on-line HMM-based recognition compared to what has been done with off-line HMM-based recognition. One of the main reasons is that, for on-line recognition, the temporal information which is needed by the HMM classifier may be ruined due to the varying

writing order among different writers, which is mainly caused by delayed strokes. Hence, delayed strokes detection is considered a crucial step for proper on-line feature extraction.

The detection of the delayed strokes can be used for lexicon reduction as well. Lexicon reduction is the process of initially eliminating lexicon entries unlikely to match the given test word. This process, lexicon reduction, has desirable effects not only on the recognition time, but also on the recognition accuracy [7]. The lexicon reduction in our system is based on a new method for modeling the variability in the delayed strokes as a feature for lexicon pruning.

Our system is trained and evaluated using the Arabic DAtabase (ADAB) on-line database of Tunisian town names, which is divided into three sets containing 23,251 words (122,559 characters) [4]. A high recognition rate of up to 97.5 % is achieved, which is very promising compared to the highest recognition rate achieved in this database and is significantly higher than the recognition rates achieved by the on-line/off-line HMM-based Arabic handwriting recognition systems [4]. Our system achieves a recognition rate of 95.97 % on the main test performed in the on-line Arabic handwriting recognition competition 2009 (ICDAR 2009), which comes second after the recognition rate achieved by the commercially available OCR product VisionObjects.

This paper is organized as follows. Section 23.2 describes the system architecture and what happens to the test word data from the moment it enters the system until the output word is resulted from the system. Section 23.3 details the description of the pre-processing stage and the delayed strokes detection method. Section 23.4 describes the lexicon reduction algorithm. Section 23.5 is concerned with the off-line and on-line feature extraction methods. Section 23.6 discusses HMM and the training procedure, and Sect. 23.7 describes the post-processing stage. Section 23.8 presents the experiments performed to evaluate our system, and we conclude our work in Sect. 23.9.

23.2 System Architecture

Figure 23.1 shows the architecture of our system and describes how the integration between the on-line and off-line feature extraction methods is done. The different blocks of the architecture are explained in detail in the following sections of this paper. In this section, we present the whole system in a general way and describe what happens to the test word from the moment it enters the system until the output word is resulted from the system.

In our system, first the test word goes through a smoothing stage that removes the jaggedness of the contour resulting from the handwriting irregularity and the imperfection caused by the acquisition device, and then a resampling stage that redistributes data points (originally sampled in equal time intervals) to enforce even spacing (resampling distance) between them. Then these data points go into two main directions that prepare them for both the off-line and on-line feature extraction stages.

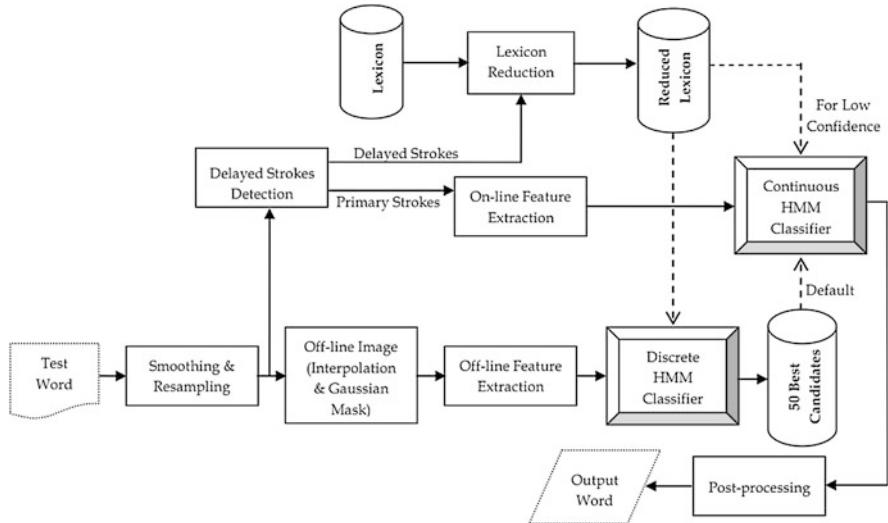


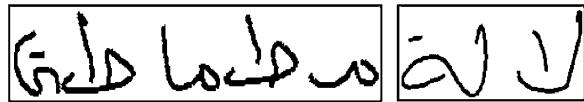
Fig. 23.1 System architecture (recognition module)

In the first direction (off-line), the data points are converted into a bitmap image (as explained in Sect. 23.5.2). Then the off-line feature sequence represented in frames of length of 63 is extracted from the bitmap image by using a right-to-left sliding window technique that is based on the gradient components of the image in 8 directions. This off-line feature vector (frame) will be classified by the discrete HMM, which selects the best 50 candidates out of the reduced lexicon. This small lexicon that is formed with the output best 50 candidates will be used later by the continuous HMM (on-line mode).

In the second direction (on-line) the data points go through a delayed strokes detection stage that detects and removes the delayed strokes from the on-line sequence. These detected delayed strokes are passed to the lexicon reduction stage, which uses them for lexicon pruning by removing the lexicon entries that are not likely to match this delayed strokes sequence. The primary strokes are used for on-line feature extraction, and a feature vector of length 9 (local direction, aspect, delta, and acceleration features) is used for classification by a continuous HMM. The continuous HMM works within the small lexicon of the 50 candidates that is generated from the off-line mode. But in the cases when the confidence of the continuous HMM decision is low, the continuous HMM operation is restarted to work within the reduced lexicon which, in most cases, is bigger than the 50-candidates lexicon.

In order to demonstrate why we restart the operation when the on-line mode HMM confidence is low, let's consider our system's behavior with the test word shown in Fig. 23.2 for the words “الآن” and “مقطاطة”. Both the test words have not been selected to be one of the off-line mode HMM's best 50 candidates, which means that the on-line mode HMMs will definitely give a wrong result, as it works on that small lexicon of 50 candidates. The off-line mode classifies the test words

Fig. 23.2 Example test files from the database ADAB



to be “رميلة” and “طلاوين” respectively, but with a low confidence for both of them. So the on-line mode HMM restarts classification to work on the bigger lexicon that resulted from the lexicon reduction stage, and gives the correct results “اللّا” and “مطماطة” respectively. This demonstrates that even though the off-line mode works as a preliminary stage for the on-line one; off-line errors will not necessarily result in wrong results on the final on-line mode.

Finally, there is a post-processing stage that is responsible for making use of the detected delayed strokes sequence and solving the confusion of the continuous HMM that may result from words that look similar without their delayed strokes (like the Arabic words “تبّار” and “تبيّار”).

23.3 Pre-processing

23.3.1 Smoothing and Resampling

Smoothing is important to remove the jaggedness of the contour resulting from the handwriting irregularity and the imperfection caused by the acquisition device. Every point $P_t(x(t), y(t))$ in the trajectory is replaced according to (23.1).

$$P_t = \sum_{k=-n}^n \alpha_k P_{t+k}, \quad \sum_{k=-n}^n \alpha_k = 1 \quad (23.1)$$

And for each point’s coordinates to be the mean value of itself and its $(2n)$ neighbors, according to (23.2),

$$\alpha_k = (2n + 1)^{-1} \quad (23.2)$$

Besides, a writing speed normalization (resampling) algorithm, based on the trace segmentation method explained in [14], is used to redistribute the data points (originally sampled in equal time intervals) to enforce even spacing (resampling distance) between them. Experiments with our system have showed that the resampling distance should be inversely proportional to the number of states of every character HHM and the number of points in each HMM frame (for on-line feature extraction).

Figure 23.3 shows the effect of smoothing and resampling on the data. After resampling, points of every stroke are equidistant, regardless of how many points are produced in each stroke. Also, experiments have showed that smoothing has increased the final recognition rate of the system.

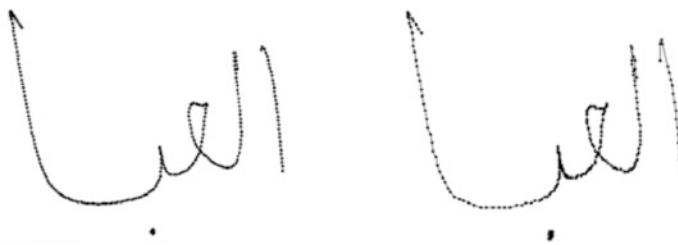


Fig. 23.3 Effect of smoothing and resampling

23.3.2 Baseline Detection

The following stage in pre-processing (delayed strokes detection) depends on the global baseline of the test word. The diacritics represented in delayed strokes, word slope, and words that are constructed from more than one piece of Arabic word (PAW) are the main problems of detecting the Arabic handwriting baseline [2].

Baseline detection in our system is based on the horizontal projection method, which is commonly used by the OCR researchers to detect Arabic baselines [15]. The following steps summarize our algorithm:

1. Construct an off-line bounded image by interpolating every stroke of the word.
2. Remove some of the clear delayed strokes that are easy to detect by their small area and constant writing direction.
3. Search the histogram for a value higher than 80 % of the maximum projection value within the narrow area under the arbitrary baseline (20 % under it).
4. IF it exists, this vertical position is selected to be the arbitrary baseline instead. This solves the problem arising from some Arabic letters that have an upper horizontal segment that may make a peak in the histogram (Fig. 23.5(1)), like the letters “ؑ” and “ؒ”.
5. Increase the thickness of every pixel vertically to be 5 pixels high, as shown in Fig. 23.4. This increases the chance that PAWs baselines meet at the same vertical position.
6. An arbitrary baseline is selected according to the horizontal projection histogram maximum value as shown in Fig. 23.5.
7. IF the arbitrary baseline is within the image upper part (upper 20 % of the image),
8. THEN search the other part of the image for a projection value higher than 60 % of the current baseline projection value. If it exists, this vertical position is selected to be the baseline.
9. This problem happens with the Arabic letter “ؑ” (Fig. 23.5(2)).
10. ELSE IF the arbitrary baseline is within the image lower part (lowest 40 % of the image),
11. THEN Search the other part of the image for a projection value higher than 60 % of the current baseline projection value. If it exists, this vertical position is selected to be the baseline.

Fig. 23.4 Increasing writing thickness vertically (zoomed)

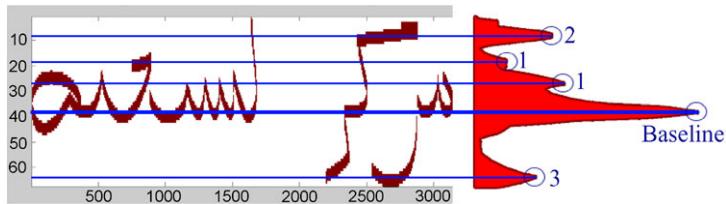
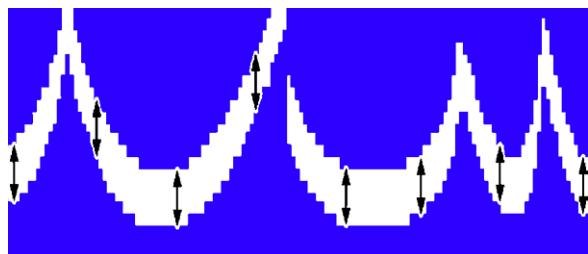


Fig. 23.5 Baseline detection problems using horizontal projection histogram for the Arabic word
مِرْكَزُ الشَّحْيَة

12. This problem happens with some of the Arabic letters like “ن” and “ر” (Fig. 23.5(3)).
13. IF step 6 does not select a new baseline,
14. THEN the arbitrary baseline is selected to be the baseline.

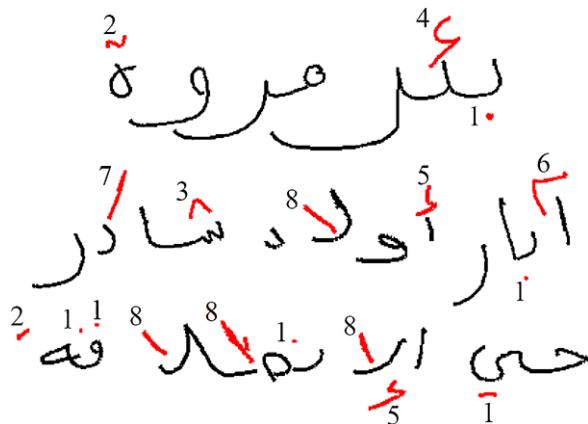
23.3.3 Delayed Strokes Detection

Many of the Arabic characters share the same primary part and are distinguished from each other by the secondary parts, which we call in this paper “delayed strokes” (shown in Fig. 23.6, in red). Delayed strokes are a well-known problem in on-line handwriting recognition. These strokes introduce additional temporal variation to the on-line sequence, because the writing order of delayed strokes is not fixed and varies among different writers. Hence, detecting and removing delayed strokes is an important key step in our system to allow meaningful on-line feature extraction.

A careful study of delayed strokes characteristics shows that all the delayed strokes of handwritten Arabic can be divided into eight categories. All of these categories share a set of common global properties, like the stroke’s dimensions, direction, number of points, trace duration, and the vertical distance from the baseline. Figure 23.6 shows some examples of all of these categories where each delayed stroke is labeled with the number of the category to which it belongs.

The main differences between the delayed strokes categories can be summarized in the following description of each category:

Fig. 23.6 Delayed strokes categories



1. Single dot or connected two-dots stroke that is fully contained above or below a primary stroke and is characterized by its small area, short trace duration, and not being within the range of the baseline.
2. Single dot or connected two-dots stroke that shares the main characteristics of (1), but is slightly drifted from its primary stroke.
3. Connected three-dot stroke that is fully contained like (1), but has a larger area and different aspect ratio (around 1).
4. The “Hamza” stroke, which has similar characteristics as (3), but also has an explicit zigzag shape that is clearly detected.
5. The “Hamza” stroke associated with the Arabic characters “ؑ” and “ؒ”, which is located above or under a vertically straight primary stroke.
6. The “Maad” stroke associated with the Arabic character “ؔ”, which comes above a vertically straight primary stroke and does not have the zigzag shape.
7. The slant straight stroke associated with the Arabic character “ؙ”. It is the only delayed stroke that may be written before its primary stroke.
8. The vertically straight stroke associated with the Arabic characters “ؚ”, “؜”, and the ligature “؝”.

Once a stroke is detected to belong to any of the delayed strokes categories, it is considered a delayed stroke that should be removed from the on-line sequence. These delayed strokes are rearranged to be used for the lexicon reduction stage described in the following section.

23.4 Lexicon Reduction

Lexicon reduction is the process of initially eliminating lexicon entries unlikely to match the given test word. This process, lexicon reduction, has desirable effects not only on the recognition time, but also on the recognition accuracy [7]. In our system,

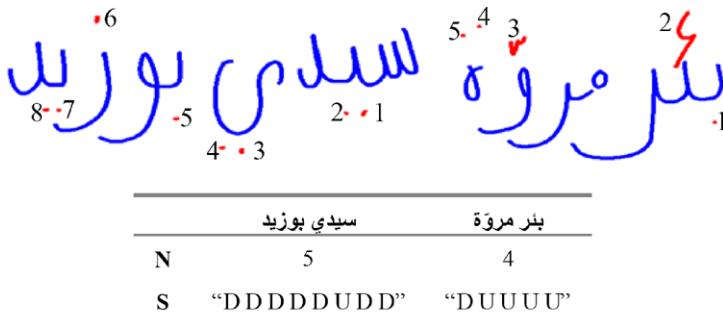


Fig. 23.7 Delayed strokes rearrangement and test word information

lexicon reduction is based on the number of primary strokes and the writing order of the delayed strokes. The lexicon reduction method used in our system is described in the following steps.

23.4.1 Test Word Information

First of all, the delayed strokes of the test word that are detected using the method discussed in Sect. 23.3.3 are used to obtain the following two pieces of information:

1. The number of primary strokes (N). This is obtained by counting the number of strokes that are not detected to be delayed strokes, i.e. primary strokes.
2. The delayed strokes sequence string (S). First, the writing order of the delayed strokes is rearranged so that the delayed strokes to the right come first, as shown in Fig. 23.6. The vertical position of each delayed stroke (up or down) is determined by making use of the baseline information and the relative position of the delayed stroke with respect to its primary stroke. The delayed strokes sequence is a string that consists of downs (D) and ups (U) which represents the vertical positions of the rearranged delayed strokes, as shown in Fig. 23.7.

Figure 23.7 shows two samples from the ADAB of the Arabic names “بئر مروة” and “سیدی بوزید”, where the detected delayed strokes and primary strokes are shown in red and blue, respectively. The numbers on Fig. 23.7 describe the modified writing order of the delayed strokes.

23.4.2 Lexicon Entries Information

In order to know which lexicon entries are unlikely to match the given test word to be eliminated from the lexicon, we obtain two pieces of information for each lexicon

entry. The first is the minimum number of strokes (N_{\min}). The minimum number of strokes in a word is equal to the number of PAWs in it. For any Arabic word (single word without spaces) the last character of every PAW (except for the last PAW of the word) is guaranteed to be one of the following characters:

ا إ آ ء ع د ذ ر ز و ف ي

These characters, end-of-the-PAW letters, are the only Arabic letters that do not come in the middle or at the beginning of a PAW. All the other Arabic letters do not come at the end of a PAW unless they are already the last letter of the word. Thus, for calculating the minimum number of non-diacritic components from a lexicon entry ground truth, first, the number spaces that is not preceded by an end-of-the-PAW letter is calculated “ s ”:

$$\begin{aligned} N_{\min} &= \text{Number of PAWs} \\ &= \text{Number of End-of-the-PAW Characters of the word} + s \end{aligned} \quad (23.3)$$

The second piece of information is the delayed strokes sequence regular expression (S^*). The number of delayed strokes (DS) associated with some Arabic characters is not fixed, and varies among different writers within fixed lower and upper bounds. For example, the Arabic character “ش” delayed strokes may be written in one triangle stroke, two strokes, or three dot strokes as shown in Fig. 23.8.

For the lexicon entry, a regular expression (S^*) that describes all the possible strings that describe the delayed strokes sequence is obtained. Table 23.2 shows all the Arabic characters that have delayed strokes and their lower and upper bounds as well as the regular expression part for each character, which is used to construct the word delayed strokes sequences regular expression (S^*).

By iterating all the characters of the word, the regular expression (S^*) which describes all the possible delayed strokes sequence strings is obtained, where the vertical bars denote alternatives and the square brackets denote optional delayed strokes.

These two pieces of information (N_{\min} and S^*) are obtained for all the lexicon entries and stored off-line to increase the speed of the system.

23.4.3 Eliminating Lexicon Entries

For the test word, N and S are obtained, according to the methods explained before, as well as the stored values of N_{\min} and S^* for all the lexicon entries. For each lexicon entry, if one of the following two conditions is satisfied, this lexicon entry is eliminated from the lexicon:

Fig. 23.8 Delayed strokes in the Arabic character “ش”

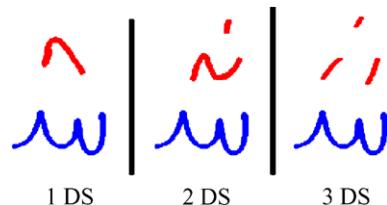


Table 23.2 Arabic characters delayed strokes

Arabic Characters	Delayed Strokes	Regular Expression (S^*)
ك (In beginning or middle of a PAW)	Either 1 up or no delayed strokes	[U]
ج (In isolated form)	Either 1 up or 1 down delayed stroke	U D
أ و ئ خ ذ ز ض ط غ ف ن	The ligature: ة	1 up delayed stroke
Character with “Shaddah” diacritic		U
ق ة ت	Either 1 or 2 up delayed strokes	U [U]
ش ث	Either 1, 2, or 3 up delayed strokes	U [U] [U]
ظ	2 up delayed strokes	U U
The ligatures: ظ ظ		
ب !	1 down delayed strokes	D
ج (In beginning or middle of a PAW)	1 up delayed strokes	
غ	Either 1 or 2 down delayed strokes	D [D]
The ligature: ة	1 up then 1 down delayed strokes	UD

1. $N < N_{\min}$.
2. S does not match S^* .

In fact, delayed strokes are very sensitive to writer and writing style. A mistake in the delayed stroke detection process that over-detects some primary stroke to be a delayed stroke may result in the first condition ($N < N_{\min}$) to be satisfied by mistake and the correct entry to be removed from the reduced output lexicon. Hence, it is better to modify this condition to be $N < N_{\min} - 2$ (a looser condition).

Besides the delayed strokes detection errors, in many cases writers forget to write some of the delayed strokes or write the same delayed stroke more than one time (overwriting). In those cases, the second condition (S does not match S^*) may cause

Table 23.3 Lexicon reduction example

Lexicon Entry	S*	N _{min-2}	سیدي بوزيد						بنر مروءة					
			N=5, S: "D D D D D U D D"						N=4, S: "D U U U U"					
			MED	C1	C2	Elimi-	MED	C1	C2	Elimi-				
الأقصاب	U U U [U] D	4-2=2	6	x	✓	✓	2	x	x	x				
بوجر	D D	2-2=0	6	x	✓	✓	4	x	✓	✓				
بنر مروءة	D U U U [U]	4-2=2	6	x	✓	✓	0	x	x	X				
راس الملاحة	U U [U]	7-2=5	7	x	✓	✓	2	✓	x	✓				
سیدي احمد زروق	D [D] D [D] D U U U [U]	8-2=6	2	✓	x	✓	2	✓	x	✓				
سیدي بوزيد	D [D] D [D] D U D [D]	5-2=3	0	x	x	x	3	x	✓	✓				
المصيدة	D [D] U [U]	3-2=1	5	x	✓	✓	2	x	x	X				
المهدية الزرقاء	D [D] U [U] U U	8-2=6	5	✓	✓	✓	0	✓	x	✓				

the correct entry to be removed from the reduced output lexicon. Hence, this condition is modified so that if (S) does not match (S^*) the condition is still not verified and considers if there is a possible slight edit on (S) that makes it match (S^*) or not. This comparison is performed using a dynamic programming technique called “minimum edit distance” [19], which is an algorithm that is used to measure the amount of difference between two sequences (i.e., the edit distance). From this discussion, the lexicon entries are eliminated from the output lexicon if either of the following two conditions is verified for the test word:

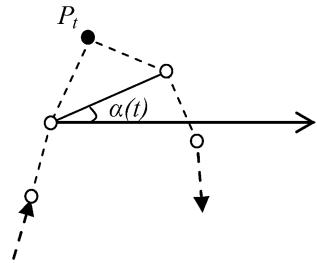
1. $N < N_{\min} - 2$.
2. The minimum edit distance between S and $S^* \geq 2$.

Table 23.3 shows some examples of the eliminated and kept lexicon entries for the two test names of Fig. 23.7 “بنر مروءة” and “سیدي بوزيد” (”).

23.5 Feature Extraction

Feature extraction is a very important step in every recognition system. In this phase of the system, the input data represented in the captured sequence of points $P_t(x(t), y(t))$ are prepared to be used by the HMM classifier that needs temporal information of the input data. In our system two types of on-line features are used, besides a sliding-window-based off-line feature that makes use of the gradient components of the image in different directions.

Fig. 23.9 Writing direction features



23.5.1 On-line Features

Writing Direction

This feature describes the local writing direction using the cosine and sin of $\alpha(t)$, where $\alpha(t)$ for every point P_t in the sequence is the angle between the line connecting P_{t-1} and P_{t+1} and the positive direction of the x -axis [8] as shown in Fig. 23.9.

Vicinity Aspect Feature

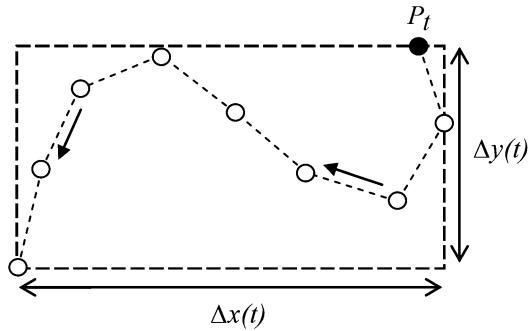
Figure 23.10 shows the vicinity of a point P_t , containing P_t and the group of the succeeding points, where the actual number of points of the vicinity (N) in our system is equal to nine points. This feature characterizes the height-to-width ratio of the bounding box of the vicinity of P_t (P_t and its next N points) as shown in Fig. 23.10. It is represented with $A(t)$ [17], where:

$$A(t) = \frac{\Delta y(t) - \Delta x(t)}{\Delta y(t) + \Delta x(t)} \quad (23.4)$$

Experiments with the on-line recognition system discussed in this paper showed that the size of the vicinity of each point (N) depends on the resampling distance; i.e., the shorter the resampling distance is, the more points are needed in the vicinity. We have taken $N = 9$, and the on-line HMM feature vector (frame) is chosen to contain the features of each consecutive 3 points.

A way to describe the dynamics of the signal, which has been adopted from speech recognition, is the use of delta and acceleration features [3] which are approximations to the derivatives of the observation vector with respect to time. Let O_t denote a feature vector of the current frame t . There is a delta feature vector ΔO_t that describes time difference and an acceleration feature $\Delta^2 O_t$ that describes the second time derivative, that is, approximations to the derivatives of the observation vector with respect to time. The delta and acceleration coefficients are estimated from equations described in [16]. The combination of O_t , ΔO_t , and $\Delta^2 O_t$ yields a new enlarged feature vector O , so that the on-line feature vector (frame) length is 9 (3(sin $\alpha(t)$, cos $\alpha(t)$, $A(t)$) + 3(delta features) + 3(acceleration features)). The

Fig. 23.10 Aspect feature (with $N = 9$)



combination of the feature vector with the delta and the acceleration features yields a new enlarged on-line feature vector (frame) that has proved to increase the system final recognition rate.

23.5.2 Off-line Features

The input data represented in the captured sequence of points $P_t(x(t), y(t))$ are interpolated such that every on-line stroke represents a continuous writing contour of black pixels. Then the data points are converted into a bitmap image f . Then we apply a Gaussian mask that dilates the writing contour [12]. The image is then resized to keep a fixed height (H) while preserving the aspect ratio of the test word. For feature extraction, the gradient operator [12] is applied to the bitmap image to give two gradient components: strength $|g(x, y)|$ and direction $\angle g(x, y)$ for all the points (x, y) of the image f . This is done by applying the Sobel operator [6] on the image to extract the vertical and horizontal gradient components:

$$\begin{aligned} g_x(x, y) &= f(x + 1, y - 1) + 2f(x + 1, y) + f(x + 1, y + 1) - f(x - 1, y - 1) \\ &\quad - 2f(x - 1, y) - f(x - 1, y + 1) \end{aligned} \quad (23.5)$$

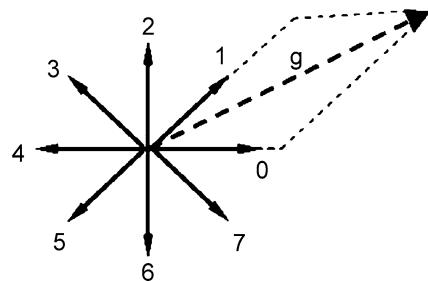
$$\begin{aligned} g_y(x, y) &= f(x - 1, y + 1) + 2f(x, y + 1) + f(x + 1, y + 1) - f(x - 1, y - 1) \\ &\quad - 2f(x, y - 1) - f(x + 1, y - 1) \end{aligned} \quad (23.6)$$

Here $g_x(x, y)$ and $g_y(x, y)$ are the x -direction and y -direction gradient components at the location (x, y) , respectively. Then the gradient strength and direction are converted using (23.7) and (23.8).

$$|g(x, y)| = \sqrt{(g_x^2(x, y) + g_y^2(x, y))} \quad (23.7)$$

$$\angle g(x, y) = \tan^{-1} \left(\frac{(g_y(x, y))}{(g_x(x, y))} \right) \quad (23.8)$$

Fig. 23.11 Projecting the gradient vector g into the two nearest Freeman directions



The gradient vector $g(x, y)$ (expressed as strength $|g(x, y)|$ and direction $\angle g(x, y)$) at each point (x, y) of the image is then decomposed into the eight Freeman directions [6] shown in Fig. 23.11. The gradient vector is decomposed into the eight Freeman directions by projecting the vector into the nearest two Freeman directions as shown in Fig. 23.11.

After applying the gradient operator to the bitmap image and getting the projection of the gradient vectors of the image, eight images are obtained (each corresponding to one of the Freeman directions), where each image is the projection of the gradient vectors of the image into the corresponding Freeman direction. Then a Gaussian mask $h(x, y)$ is applied to each image of the eight, where:

$$h(x, y) = \frac{1}{(2\pi\sigma^2)} \exp\left(-\frac{(x^2 + y^2)}{(2\sigma^2)}\right) \quad (23.9)$$

Our sliding window features are going to be extracted from each image from the obtained eight images (each corresponding to one of the Freeman directions), besides the original image. Figure 23.12 shows the original image of an example file from the on-line database ADAB after it has been converted to the off-line form and its eight projection images.

Now, for each image of the nine (one original image plus eight projection images), a rectangular sliding window of a fixed width and without overlap is used for feature extraction. A window that is too narrow may result in the content to be improperly analyzed, while a window that is too wide may cause incorrect character segmentation. The window is divided into seven cells. It is shifted from right to left (in accordance with the Arabic writing direction) across the normalized gray level script image to generate a feature vector (frame) at each shift position for all the nine images as shown in Fig. 23.13.

In the figure H is the fixed height of the images (150 pixels), h is the fixed height of all the cells ($h = H/7$), and w is the width of a frame, chosen to be 2 pixels.

Thus a feature vector (frame) of 63 features (9 images * 7 cells per frame) that represents the temporal input data for the offline HMM is obtained by calculating F_i for each cell. F_i is the number of black pixels relative to the total area of the cell:

$$F_i = \frac{n_i}{(w * h)} \quad (23.10)$$

where n_i is the number of black pixels in the cell i .

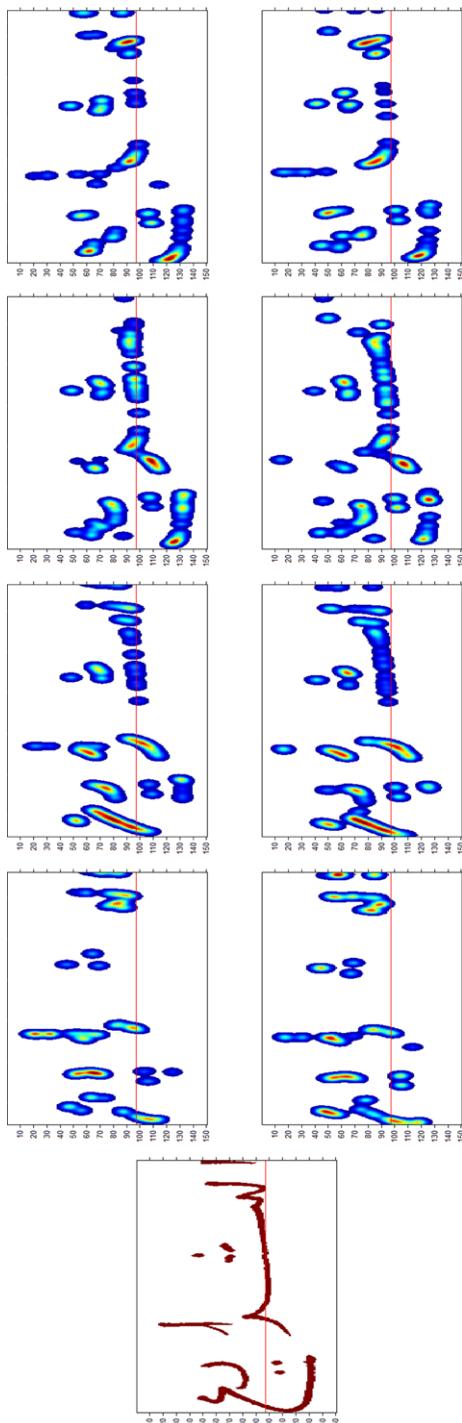
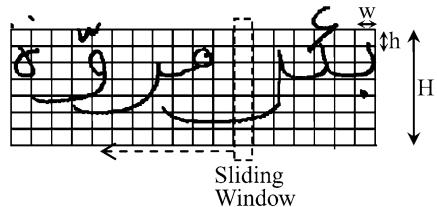


Fig. 23.12 A sample file from the ADAB that shows the original image after being converted to the off-line form with its eight directional subimages after applying the Gaussian mask to them

Fig. 23.13 Sliding window for one image



23.6 HMM and Training

23.6.1 HMM

The hidden Markov model (HMM) is a double stochastic process which can efficiently model the generation of sequential data. HMMs have been successfully used in speech and handwriting recognition. In our system, the same density HMM classifier without modification as implemented in the HTK Speech Recognition Toolkit [9] is used for segmentation and recognition. However, we implement our own parameters of the HMM. HTK models the feature vector with a mixture of Gaussian distributions and uses the Viterbi algorithm in the recognition phase, which searches for the most likely sequence of characters given the input feature vector. HTK supports multiple steps in the recognition process: data preparation, training, recognition and post-processing. The data preparation process supports only the speech data, so we do not use HTK for this step which also includes lexicon reduction (implemented using the task grammar), the dictionary, and the feature extraction process.

There are basically two classes of HMM depending on the type of observation sequence (feature vector) that HMM is trained to recognize: the discrete HMM, where the observation is a discrete sequence, and the continuous HMM, where the observation sequence can take continuous values. We use both types in different stages of the system. The choice of using a discrete HMM in our system besides the continuous HMM was to make use of its faster recognition speed.

Figure 23.14 shows the case of a 20-state left-to-right HMM which we chose for all the continuous models in our system based on some experiments. Transitions to the current and the next states only have been allowed. The same number of states is adopted for all models and the same number of Gaussians as well, while each HMM model represents at least one Arabic character as explained later in the training section (Sect. 23.6.2).

Eighteen states of these are emitting states and have output probability distributions associated with them. HTK is principally concerned with continuous density models in which each observation probability distribution is represented by a mixture Gaussian density. In this case, for state j the probability $b_i(o_t)$ of generating observation o_t is given by (23.11):

$$b_i(o_t) = \prod_{s=1}^S \left[\sum_{m=1}^{M_s} c_{jsm} \Psi(o_{st}; \mu_{jsm}, \Sigma_{jsm}) \right]^{\gamma_s} \quad (23.11)$$

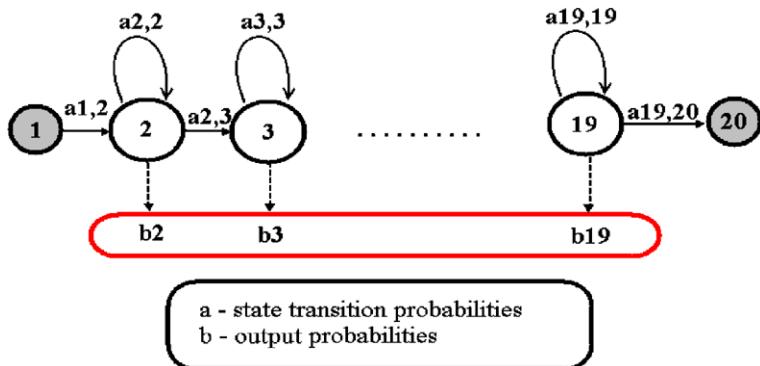


Fig. 23.14 A 20-state left-to-right HMM

where M_s is the number of mixture components in state j for stream s , the exponent γ_s is a stream weight and its default value is one, c_{jsm} is the weight of the m th component, and $\Psi(o; \mu, \Sigma)$ is a multivariate Gaussian with mean vector μ and covariance matrix Σ , that is,

$$\Psi(o; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-\mu)^T \Sigma^{-1}(o-\mu)} \quad (23.12)$$

where n is the dimensionality of o .

Sixteen mixtures have been chosen after experiments to give a robust model for all the characters and increase the performance of the continuous HMM. The transition matrix for this model has 20 rows and 20 columns. Each row will sum to one except for the final row which is always all zero, since no transitions are allowed out of the final state.

For the discrete HMM, after feature extraction, each feature vector is quantized to the nearest codeword in the codebook. The codebook is generated from millions of feature vectors extracted from the training data using the LBG algorithm [11]. Experiments have been conducted to choose the size of the codebook, and a codebook of 256 codewords was found to be suitable for a good recognition rate while maintaining a reasonable recognition time. A ten-state left-to-right HMM is adopted for all discrete models.

23.6.2 Training

In our system, we use the two approaches of feature extraction: off-line and on-line. In fact, experiments showed that our on-line feature extraction mode has an obvious higher performance than the off-line one in terms of recognition rate, which is the common case in the literature. The highest recognition rate achieved in the ICDAR 2009 off-line Arabic handwriting competition main test is 93.37 % and is achieved

Table 23.4 Arabic classes based on the primary part of the characters

Letter and ligatures	Possible Shapes			
	Isolated	End	Middle	Start
Aleph	ا	أ	أ	أ
Ba'a, Ta'a, Tha'a, Nun, Ya'a	ت ث ب	ب ت ث ب	ب ت ث ب	ب ت ث ب
Jeem, Ha'a, Kha'a	ج ح خ	ج ح خ	ج ح خ	ج ح خ
Dal, Thal	د ذ	ذ	ذ	ذ
Raa, Zai	ر ز	ر ز	ر ز	ر ز
Seen, Sheen	س ش	س ش	س ش	س ش
Sad, Dad	ص ض	ص ض	ص ض	ص ض
TTa, ThTha	ط ظ	ط ظ	ط ظ	ط ظ
Ein, Gein	ع غ	ع غ	ع غ	ع غ
Faa, Qaf	ف	ف	ف	ف
Qaf	ق	ق	ق	ق
Kaf	ك	ك	ك	ك
Lam	ل	ل	ل	ل
Meem	م	م	م	م
Nun	ن	ن	ن	ن
Hah, Ta'a	ه ه	ه ه	ه ه	ه ه
Waw	و و	و و	و و	و و
Ya'a	ي ي	ي ي	ي ي	ي ي
Hamza	ء			
LamAleph	ل ل ل ل	ل ل ل ل	ل ل ل ل	ل ل ل ل

by the MDLSTM system [13], while the highest recognition rate achieved in the ICDAR 2009 on-line Arabic handwriting competition main test is 98.99 % and is achieved by the commercially available OCR product VisionObjects [4]. Hence, we selected the off-line feature extraction mode to work as a preliminary stage that selects a large number of candidates (best 50 candidates) from the reduced lexicon to represent a small lexicon for the on-line mode HMM to work within. Having a high number of output candidates (50 candidates), the HMM decoding of the off-line stage takes a significant amount of time. This is why we used a discrete HMM model for the off-line mode so that it takes a reasonable decoding time, and a continuous HMM model for the on-line mode in order to make it accurate enough by avoiding any quantization problems.

The main difference between the on-line HMM (continuous) and the off-line HMM (discrete) in our system is that the on-line HMM is trained to recognize the primary part of the words (without the delayed strokes), while the off-line HMM is trained to recognize each character as a separate class.

For the on-line HMM, the Arabic characters which have a common primary part and are only different in their delayed strokes, like the Arabic characters “س” and

Word Entry	Corresponding Classes				
بَثْر	ر	ـ	ـ	ـ	ـ
مَرْوَة	ـ	ـ	ـ	ـ	ـ
فُوسَانَة	ـ	ـ	ـ	ـ	ـ
فُوشَانَة	ـ	ـ	ـ	ـ	ـ
حِي	ـ	ـ	ـ	ـ	ـ
السَّرُور	ـ	ـ	ـ	ـ	ـ
السَّرُور	ـ	ـ	ـ	ـ	ـ
ـ	ـ	ـ	ـ	ـ	ـ
ـ	ـ	ـ	ـ	ـ	ـ
ـ	ـ	ـ	ـ	ـ	ـ

Word Entry	Corresponding Classes				
بَثْر	ر	ـ	ـ	ـ	ـ
مَرْوَة	ـ	ـ	ـ	ـ	ـ
فُوسَانَة	ـ	ـ	ـ	ـ	ـ
فُوشَانَة	ـ	ـ	ـ	ـ	ـ
حِي	ـ	ـ	ـ	ـ	ـ
السَّرُور	ـ	ـ	ـ	ـ	ـ
السَّرُور	ـ	ـ	ـ	ـ	ـ
ـ	ـ	ـ	ـ	ـ	ـ
ـ	ـ	ـ	ـ	ـ	ـ
ـ	ـ	ـ	ـ	ـ	ـ

(a)

(b)

Fig. 23.15 (a) Example from the on-line training dictionary (without delayed strokes). (b) Example from the off-line training dictionary (with delayed strokes)

“ش”, should belong to the same class and are given the same label during the HMM training, thus reducing the number of classes (HMM models) significantly. The on-line HMM classes are shown in Table 23.4, plus some other symbols encountered in the ADAB database like digits or the Latin letter “V”.

The word-based training dictionary of the on-line HMM can be generated by iterating on the full lexicon words as follows, for all the lexicon words:

Convert the word’s characters into the corresponding classes without the delayed strokes.

IF the produced characters sequence has not been written before in the dictionary, THEN write it.

On the other hand, for the off-line HMMs, we have a separate class for each Arabic character. Moreover, there is a special separate classes for each character if it has a “Shaddah” diacritic; i.e., the Arabic character “en” when it comes with a “haddah” (“ــ”) is a class apart from the other classes of “ــ” and “ــ”. The total number of characters classes is 150, which corresponds to the number of Arabic letters multiplied by the number of different shapes for each letter (initial, middle, end, and isolated), plus some other symbols encountered in the ADAB database like digits or the Latin letter “V”. The word-based training dictionary of the off-line HMM can be generated by iterating the full lexicon words as follows, for all the lexicon words:

Convert the word’s characters into the corresponding classes.

IF the produced characters sequence has not been written before in the dictionary, THEN write it.

Examples from the on-line training dictionary (without delayed strokes) are shown in Fig. 23.15(a) and from the off-line training dictionary (with delayed strokes) in Fig. 23.15(b).

Table 23.5 ADAB entries that share the same primary part

	Classification Result	Possibilities
تَبَار		تَبَار – تَبَار
فُوسَانَة		فُوسَانَة – فُوسَانَة
مَثْلِين		مَثْلِين – مَثْلِين
فَرِيَانَة		فَرِيَانَة – فَرِيَانَة
حَي السَّرُور	حَي السَّرُور	حَي السَّرُور – حَي السَّرُور

In Fig. 23.15(a), the on-line training dictionary, the characters that share the same primary part (only different in delayed strokes) are given the same class label in the dictionary. Hence, we can see in the dictionary that there are some different characters that correspond to the same class, like the two classes marked in red in the figure. They are of the same class (Beginning Nabra) although they are for different characters (“Ba’ā” and “Nun”). This leads to the fact that there are some different words that correspond to the same sequence of classes (like the words “فُوسَانَة” and “فُوشَانَة”, for example).

This also means that for the small dictionary that is composed of the best 50 candidates that have resulted from the off-line mode HMM, the entries that share the same primary part will be grouped together, so that the resulting dictionary size may be a bit less than 50 candidates.

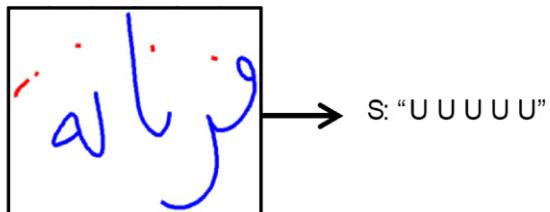
23.7 Post-processing

As discussed in Sect. 23.6, the HMM models of the on-line feature extraction mode are based on removing the delayed strokes. Hence, the output of the on-line HMM classifier will be the same for all the lexicon entries which look similar without their delayed strokes, i.e., the entries which share the same primary part. We have searched the database ADAB to find all the entries (Tunisian town names) that share the same primary part but with different delayed strokes. Those town names are shown in Table 23.5.

In this stage of the system, the result of on-line mode HMM is either one of two cases. The first case is when the classification result of the on-line mode HMMs is unique (corresponds to a candidate with a unique primary part), so there is no longer a need to use the delayed strokes information to tell which candidate of the lexicon corresponds to the result, as the primary part is valid for only one candidate.

The second case is when the classification result of the on-line mode HMM is any of the entries shown in Table 23.5. The delayed strokes information is used to decide which candidate of the candidates that share the same primary part is most probable to be the test word. This is done by making use of the delayed strokes se-

Fig. 23.16 On-line mode HMMs and the problem of lexicon entries that share the same primary part



System	S*	MED
فريانة	U D [D] U U [U]	1
فرنانة	U U U U [U]	0

Table 23.6 System performance on ADAB

Test Set	Training Sets	Recognition Rate
Set 1	Set 2,3	96.64
Set 2	Set 1,3	91.47
Set 3	Set 1,2	97.56

quence minimum edit distance (MED) technique, which is discussed in the lexicon reduction section (Sect. 23.4).

For example, let's consider the test word in Fig. 23.16 “الجنة”. If the on-line mode HMM classified it correctly, it will give the output entry. This is not a decisive result, as the entry “الجنة” is one of the Table 23.5 entries that needs a decision that tells if the correct result is “الجنة” or “الجنة”. According to our method, the entry with a delayed strokes regular expression that is more probable to match the test word delayed strokes sequence is selected to be the system result. In our example, the entry “الجنة” is found to have the minimum delayed strokes sequence MED, so it is selected to be the output candidate of the system, which is the right decision.

23.8 Results

We applied our system on the on-line database ADAB of Tunisian town names [4]. Table 23.6 shows the test results in each set of the ADAB, where, for each test, the data of the other two sets are used for training.

Table 23.7 reports the results of the other HMM-based systems that participated in the on-line Arabic handwriting recognition competition 2009 (ICDAR 2009) [4]. It shows that the recognition rates obtained by our proposed system clearly outperform those obtained by the other HMM-based systems in the competition.

Table 23.8 reports the recognition rates achieved by the participant systems of the on-line Arabic handwriting recognition competition 2009 (ICDAR 2009) [4],

Table 23.7 Results of the HMM-based systems in ICDAR 2009 on-line Arabic handwriting recognition contest

Test Set	Training Sets	REGIM-HTK	REGIM-CV-HTK	Our System
Set 1	Set 1 to 3	57.87	28.85	97.54
Set 2	Set 1 to 3	54.26	35.75	94.43
Set 3	Set 1 to 3	53.75	30.60	98.37

Table 23.8 Results of the ICDAR 2009 on-line Arabic handwriting recognition contest

System	Recognition Rate
VisionObjects-1	98.99
VisionObjects-2	98.99
Proposed system	95.97
MDLSTM-1	95.70
MDLSTM-2	95.70
REGIM-HTK	52.67
REGIM-CV-HTK	38.71
REGIM-CV	13.99

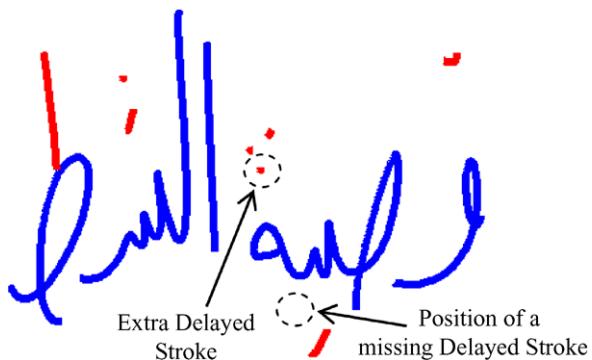
besides our system. The systems are ordered by the top 1 results on test set 4, where all the systems (including ours) are trained with sets 1 to 3 of the ADAB. The table shows that the recognition rate obtained by our proposed system clearly outperforms those obtained by the other HMM-based systems, ordered by the top 1 results on test set 4. Also, our system comes at the top of the table only after the commercially available OCR product VisionObjects.

There are several reasons why our system gives a better performance than most of the other on-line Arabic handwriting recognition systems in the literature. The most important reason is the detection and then the removal of the delayed strokes from the test word before classification by the main (on-line) HMM classifier. The removal of the delayed strokes improves the performance of our system because the delayed strokes are very sensitive to the writing styles of the different writers; thus, their removal reduces the temporal variation of the writing order. Moreover, the removal of the delayed strokes has significantly reduced the number of the input classes to the on-line HMM classifier, leading to better recognition performance in terms of both recognition accuracy and recognition time. Problems with delayed strokes detection such as confusing a primary stroke with a delayed stroke or failing to detect a delayed stroke may lead to a wrong classification result. The solution to those problems is the use of the off-line HMM classifier to classify the input word along with its delayed strokes as an off-line image. Thus, the top 50 candidates produced by the off-line HMM classifier are free from the delayed strokes misdetection problems.

Fig. 23.17 Sample test file from the ADAB



Fig. 23.18 Example test file from the ADAB database, where the detected delayed strokes are in red

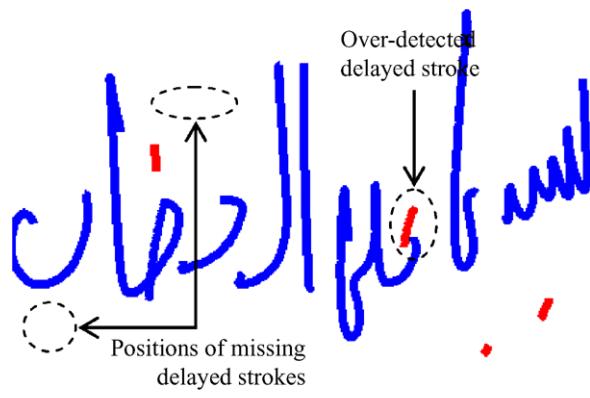


The second main reason for the proposed system's good performance is the lexicon reduction stage. Lexicon reduction improves the system's performance because it eliminates lexicon entries that could have been wrongly selected by the classifier. For example, if we consider the test word “السعد” shown in Fig. 23.17 to be tested in our system while working without a lexicon reduction stage, the correct candidate “السعد” is not chosen to be one of the best 50 candidates of the off-line mode HMM classifier, leading the on-line mode HMM to give a wrong classification result “النصر”. The use of lexicon reduction eliminates many similar and confusing words from the lexicon, which causes the word “السعد” to be selected among the off-line mode best 50 candidates, and then to be the final result of the system. Moreover, lexicon reduction simplifies the classification phase, by reducing the number of possible candidates so that it decreases the decoding time of the HMM and, consequently, the whole system. Experiments have showed that the lexicon reduction stage has reduced the system recognition time by around 40.

The performance of the lexicon reduction stage is further enhanced by relaxing the conditions of elimination from the lexicon (as explained in Sect. 23.4.3) such that they tolerate delayed strokes detection and writing errors. For example, although the test word “قصبة الشط” shown in Fig. 23.18 has one extra delayed stroke that is written by mistake and one missing delayed stroke, the lexicon reduction does not eliminate the correct entry “قصبة الشط” from the lexicon, and the final result of the system is correct.

The third reason for the success of the proposed system is the careful choice of both the off-line and the on-line features. Good feature extraction mechanisms are crucial for the success of any recognition system. The use of gradient features with the off-line HMM classifier and local direction features with the on-line HMM

Fig. 23.19 Sample test file from the ADAB



classifier has been a fundamental reason behind the success of the proposed system. The delta and acceleration features proved to be very effective in improving the system's performance.

Unfortunately, the dependence of the proposed system on the delayed strokes detection results in errors in some cases. For example, the test word “سیدی على الحطاب” shown in Fig. 23.19 has some missing delayed strokes caused by mistakes of the writer and one primary stroke detected as a delayed stroke. This results in a delayed strokes sequence $S = D D U D U$, while the delayed strokes regular expression of the lexicon entry “سیدی على الحطاب” is $S^* = D [D] D [D] D [D] U U D$, which makes a MED with an S of three. So the entry “سیدی على الحطاب” is initially eliminated from the lexicon, and the system fails to classify the test word correctly.

23.9 Conclusions

We presented a new HMM-based recognition system for Arabic handwriting that uses a combination of powerful on-line and off-line feature extraction methods with continuous and discrete HMMs. The use of gradient features with the off-line HMM classifier and local direction features with the on-line HMM classifier has been a fundamental reason behind the success of the proposed system. The delta and acceleration features are used as approximations to the derivatives of the observation vectors with respect to time, and they have proved to be very effective in improving the system's performance. On the other hand, we solved the well-known problem in on-line handwriting recognition of the varying writing order among different writers by introducing a new delayed strokes detection and removal mechanism. Delayed strokes are also used for our newly developed algorithm for lexicon reduction, which depends on the formation of the delayed strokes sequence of the

test word and compares it with the delayed strokes sequence of each lexicon entry. Removing the delayed strokes has not only increased the on-line mode HMM performance by reducing the temporal variation, but it has also increased the performance of the on-line mode HMM by significantly reducing the number of classes by grouping the characters of the same primary part into one class. The lexicon reduction has increased the performance of the system in terms of recognition speed by reducing the size of the HMM lexicon, and in terms of recognition rate by initially eliminating the wrong lexicon entries that would have been chosen by the classifier by mistake. We solved the problem of the delayed strokes being very sensitive to writer and writing style by relaxing the lexicon reduction pruning conditions to tolerate the detection and writing errors of the delayed strokes. The off-line mode HMM is chosen to be a preliminary stage for the on-line mode HMM and to recognize the words with their delayed strokes, so that its results will not be affected by the delayed strokes detection errors. Also, the careful choice of both the number of states as well as the number of Gaussians of the off-line and on-line HMM models is an important step that directly affects the system's recognition rate. Our system is evaluated using the database ADAB, and very high recognition rates are achieved that clearly outperform those obtained by the other Arabic HMM-based systems, and come second after the highest recognition rate achieved on this database in the 2009 on-line Arabic handwriting recognition competition.

References

1. Al-Habian, G., Assaleh, K.: Online Arabic handwriting recognition using continuous Gaussian mixture HMMs. In: Conference on International Intelligent and Advanced Systems (ICIAS), pp. 1183–1186 (2007)
2. Al-Shatnawi, A., Omar, K.: Methods of Arabic language baseline detection—the state of art. *Int. J. Comput. Sci. Netw. Secur.* **8**(10), 137–143 (2008)
3. Dolfling, J.G.A., Haeb-Umbach, R.: Signal representation for hidden Markov model based on-line handwriting recognition. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 4, pp. 3385–3388 (1997)
4. El Abed, H., Kherallah, M., Märgner, V., Alimi, A.M.: Arabic online handwriting recognition competition—ADAB database and participating systems. *Int. J. Doc. Anal. Recognit.* **14**(1), 15–23 (2011). Special Issue on Performance Evaluation
5. Elanwar, R.I., Rashwan, M.A., Mashali, S.A.: Simultaneous segmentation and recognition of Arabic characters in an unconstrained on-line cursive handwritten document. In: World Academy of Science, Engineering and Technology, vol. 29 (2007)
6. Gonzales, R.C., Woods, R.E.: Digital Image Processing, 2nd edn. Addison-Wesley, Reading (2002)
7. Guillevic, D., Nishiwaki, D., Yamada, K.: Word lexicon reduction by character spotting. In: Proc. 7th International Workshop on Frontiers in Handwriting Recognition, pp. 373–382 (2000)
8. Guyon, I., Albrecht, P., Le Cun, Y., Denker, J., Hubbard, W.: Design of a neural network character recognizer for a touch terminal. *Pattern Recognit.* **24**(2), 105–119 (1991)
9. HTK Speech Recognition Toolkit. <http://htk.eng.cam.ac.uk/>
10. Khorsheed, M.S.: Off-line Arabic character recognition—a review. *PAA Pattern Anal. Appl.* **5**, 31–45 (2002)

11. Linde, Y., Buzo, A., Gray, R.M.: An algorithm for vector quantization design. *IEEE Trans. Commun.* **28**, 48–95 (1980)
12. Liu, C., Nakashima, K., Sako, H., Fujisawa, H.: Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognit.* **36**, 2271–2285 (2003)
13. Märgner, V., El Abed, H.: ICDAR 2009 Arabic handwriting recognition competition. In: 10th International Conference on Document Analysis and Recognition (2009)
14. Pastor, M., Toselli, A., Vidal, E.: Writing speed normalization for on-line handwritten text recognition. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR) (2005)
15. Pechwitz, M., Märgner, V.: Baseline estimation for Arabic handwritten words. In: 8th Inter. Workshop on Frontiers in Handwriting Recognition (IWFHR'02), pp. 479–484 (2002)
16. Rabiner, L., Juang, B.H.: Fundamentals of Speech Recognition. Prentice Hall, Englewood Cliffs (1993)
17. Schenkel, M.E.: Handwriting recognition using neural networks and hidden Markov models. In: Series in Microelectronics, vol. 45 (1995)
18. Tanaka, H., Nakajima, K., Ishigaki, K., Akiyama, K., Nakagawa, M.: Hybrid pen-input character recognition system based on integration of on-line-offline recognition. In: Proceedings of the Fifth International Conference on Document Analysis and Recognition (ICDAR), pp. 209–212 (1999)
19. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. *J. ACM* **21**(1), 168–173 (1974)

Index

A

- Agent paradigm, 359
cognitive agent, 360
reactive agent, 360
Arabic script, vii, 4, 7, 36, 40, 148, 170, 269, 335, 336, 500, 544
Arabic alphabet, 4, 148, 171, 509, 545
Arabic characters, viii, 14, 63, 148, 150, 171, 193, 237, 346, 364, 366, 398, 431, 444, 500, 569
ligatures, ix, 13, 36, 64, 124, 149, 150, 171, 336

B

- Benchmarking, 423, *see* Competitions
BHMM, 260
backward algorithm, 261
Bernoulli HMM, 255, 257
Bernoulli mixture, 256
embedded Bernoulli (mixture) HMMs (BHMMs), 256
forward algorithm, 260
Viterbi algorithm, 263
windowed BHMMs, 266

C

- Clustering, 285, 460
K-means clustering algorithm, 285
Markov clustering (MCL), 470–472, 481, 482
optimum number of clusters, 285
self-organizing map (SOM), 460, 462, 549
Combination, 351, 559
combined system, 307
decision combination approach, 337
feature combination approach, 337
hierarchical MLP, 220

- hybrid MLP/HMM, 221, 237
information combination, 336
MLP-GHMM, 221, 237
multi-engine, 73
multi-stream combination approach, 337
on-line and off-line features, 559, 571, 572

Competitions

- ICDAR competition, 554
ICDAR 2005, 400
ICDAR 2007, 402
ICDAR 2009, 405
ICFHR 2010, 408
ICDAR 2011, 410
mono-font competition, 436
multi-font competition, 436
Connected component (CC), 456–460, 463, 469–472
basis connected component (BCC), 456, 458, 469, 470–472
extraction of CCs, 459

D

- Databases, 26, 138, 150, 151, 541
AMA data set, 383
Anfal corpus collection, 378
APTI database, 425, 426, 427, 429, 430, 432
Arabic DAtaBase (ADAB), 551, 552, 578
Arabic Gigaword corpus, 532
Arabic News corpus, 377
corpus collection, 27
DARPA MADCAT dataset, 493, 497
data creation, 531
database conception, 172
document representation, 29
Farsi dataset, 275, 277
IAM, 138, 494

- Databases (*cont.*)
 IFN/ENIT-database, 172, 232, 396, 398
 LMCA database, 543, 546
 multi-modal Arabic corpus (MMAC), 233, 424
 OnAR data set, 388
 RIMES, 138
 RWTH Arabic machine-print newspaper corpus, 233
 synthetic data, 151, 383
- Document processing
 document search, 26
 filter-based, 73
 fractal dimension, 317
 fractal image coding, 319, 323
 fractal image decoding, 319, 323
 fractal theory, 315
 historical document, 79, 80–82, 453, 454, 457
 image meshing, 109
 iterated function system (IFS), 318
 one-dimensional fractal, 323
 orientation estimation, 109
 time-frequency distributions, 110
 two-dimensional fractal, 319
 versatile search, 26
 word spotting, 16, 17, 26, 31, 275, 453, 457
- DTW, 352, 354, 355, 357, 365
- Dynamic programming, 21, 202, 231, 260, 261, 263, 303, 352, 354, 570
- two-level, 340, 341
- E**
- Evaluation, 85, 86, 90, 100, 185–187, 196, 216, 219–221, 232–234, 237, 238, 240, 241, 244, 275, 289, 379, 383, 395, 425, 432, 436, 446–448, 496, 538, 543
 baseline error measurement, 185
 character error rate (CER), 236, 309, 491, 494
 competence, 479
 error estimation, 433
 F-measure, 477
 ground truth, 66, 67, 72, 86, 137, 151, 169, 177, 178, 233, 307, 375–378, 385, 387, 396, 423, 424, 429, 437, 492, 497, 542, 551, 568
 performance evaluation, 362, 368, 479
 word error rate (WER), 235, 268, 494
- F**
- Farsi script
 Farsi, 273
 Farsi/Arabic, 315
- G**
- Farsi language, 273
 Iranian, 274
 Feature extraction/selection, 157, 197, 198, 219, 229, 281, 344, 442, 464, 527, 570
 angle and correlation features, 491
 aspect ratio, 461, 528
 center of mass, 460
 concavity features, 127, 284
 context-dependent, 133
 context-independent, 132
 contour features, 345
 density features, 344
 distance in the feature space, 465
 distribution features, 126, 283
 dot feature, 461
 dynamic Features, 128
 feature vector analysis, 535
 features for grapheme, 130
 Freeman direction, 573
 geometrical features, 468, 528
 global shape information, 528
 gradient-structure-concavity (GSC), 498, 499
 height ratio, 461
 HMM-based features, 159
 horizontal frequency, 461
 loop feature, 461
 minimum edit distance, 531, 570, 580
 moment features, 528
 number of branch points, 461
 number of diacritics/dots, 528
 number of end points, 461
 on-line features, 571
 percentile features, 129, 490
 pixel distribution, 528
 reference line, 528
 sliding-window, 124, 129, 443, 573
 stroke connectedness, 529
 structural features, 498
 topological features, 467, 528
 vicinity aspect feature, 571
 word shape features, 16
 writing direction, 571
 word shape matching, 22
 zero crossing, 529, 530
- Grid computing, 358
 service-oriented grid architecture (SOGA), 362, 365

H

Hidden Markov model (HMM), 286, 289, 446, 511, 575, 576
Bakis model, 202
Bernoulli HMM, 255, 257
continuous, 288, 562
decoding, 235, 239, 304
discrete, 67, 288, 562
discriminative model adaptation, 227
discriminative training, 222
GHMM, 221, 235, 239
grapheme-based HMM, 136
Hidden Markov Model Toolkit (HTK), 575
hybrid MLP/HMM, 221, 237
initial estimation, 289
maximum likelihood, 264
maximum mutual information, 223
modified Viterbi algorithm, 527
multi-stream, 335, 338
multi-stream decoding, 339
multi-stream training, 343
SCHMM, 205
semi-continuous, 205
state confidences, 227
state probabilities, 523
symbol probability, 524
testing, 292
topology, 201
training, 290, 536, 576
unified HMM, 485
variable duration hidden Markov model (VDHMM), 507, 511, 514, 515, 519, 522
Viterbi algorithm, 263
word confidences, 227
writer adaptation, 228, 241
writer adaptive training, 226

L

Layout analysis, 35
LSTM, 298, 300
multidimensional LSTM (MDLSTM), 300, 301, 305, 307, 309, 577

M

Models
character models, 346, 444, 524
character modeling, 524, 526
dotless glyph modeling, 500
glyph modeling, 498
language modeling, 231
language models, 494
model adaptation, 230
pseudo-words, 470

trajectory and velocity modeling, 548
unsupervised model adaptation, 239
visual modeling, 220, 230

N

Neural networks, 20, 286, 549
ANN, 20, 24
connectionist temporal classification (CTC), 303–307
MLP, 136, 137, 216, 339, 411, 549
multidimensional recurrent neural networks (MDRNN), 297–299, 406
recurrent neural networks (RNN), 297, 298

O

OCR systems
Arabic HWR, 495, 497, 514
BBN Byblos system, 487, 501
BMS, 502
English HWR, 494
large vocabulary OCR, 228
MDATS, 501
OCRGrid, 353
OCropus, 353
RWTH OCR, 215, 228
Off-line, 65, 326
On-line, 68, 324, 326, 388, 541, 559

P

Post-processing, 530, 579
accuracy boosting, 68, 72, 73
evidence type, 70
lexicon reduction, 566
Pre-OCR processing, 56
ImageRefiner, 57
Pre-processing, ix, 8, 9, 20, 57, 79, 81, 87, 152, 172, 174, 197, 238, 276, 343, 441, 457, 458, 488, 545, 563, 564
background normalization, 82
background separation, 83
baseline detection, 282, 564
baseline estimation, 182, 183, 186, 191
baseline extraction, 125
binarization, 6, 9, 37, 81, 82, 85, 86, 91, 92, 98, 267, 277, 457
binary document images, 87
bitonal images, 59
CC, *see* Connected component
clutter noise, 88
connected component (CC), 456–460, 463, 469–472
delayed strokes/detection, 564, 565, 566, 567, 569
dilation, 280

Pre-processing (*cont.*)

- grayscale images, 61, 80–82, 323
- holes, 92
- horizontal projection, 11, 36, 37, 42, 112, 156, 157, 182, 184, 186, 187, 190, 197, 343, 426, 441, 442, 564
- linear approximation, 83
- nonlinear approximation, 84
- normalization, 172, 174, 197, 198
- pepper noise, 89
- resampling, 563
- rule-lines, 91
- salt noise, 92
- skeleton, 463, 464
- skeletonization, 280
- skeleton-based, 191
- skew estimation/correction, 153
- smoothing, 153, 563
- text line detection, 41
- text line extraction, 100, 113
- text line finding, 489
- topline estimation, 182, 195

R

Recognition, 204

- accuracy boosting, 68, 72, 73
- Bayes decision rule, 218
- edge and border detection/recognition, 381
- evidence type, 70
- font recognition, 327
- genetic algorithm (GA), 549, 550
- HMM, 23, 24, 200, 289, 290, 292, 446
- isolated character recognition, 355
- lexicon-based approaches, 20
- logo recognition, 378
- multi-engine, 73
- post-recognition, 68
- recognition of connected or cursive characters, 355
- script and language identification, 380
- signature recognition, 379

similarity measure, 22, 159, 481

visual encoding, 549

S

- Segmentation, 11, 12, 59, 103, 156, 513
 - ALCM using steerable filter, 97
 - character segmentation, 13
 - connected line separation, 114
 - explicit segmentation, 536
 - generic segmentation, 513
 - grapheme segmentation, 553
 - Hough space, 187
 - implicit segmentation, 536
 - line segmentation, 11, 43, 44, 49, 107, 156
 - location of text lines, 98
 - over-segmentation, 507, 516
 - reading order, 49, 50
 - ridge-based text line detection, 44
 - segment relabeling, 515
 - segmentation errors, 534
 - segmentation-free, 14
 - segmentation into graphemes, 130
 - text and non-text segmentation, 38, 40, 41
 - text line detection, 41
 - text line extraction, 100, 113
 - text line separation, 94
 - under-segmentation, 410, 534
 - word segmentation, 11, 513
 - $x-y$ cut text line detection, 42

V

- Variable duration hidden Markov model (VDHMM), 507, 511, 514, 515, 519, 522

W

- Watermarking, 328
- Word spotting
 - prototype preparation, 474
 - skeleton-based segmentation-free, 457
 - spotting process, 472