Recognition and Retrieval from Document Image Collections

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Engineering

by

Million Meshesha
200299004
million@research.iiit.ac.in

International Institute of Information Technology
Hyderabad 500 032, India
http://www.iiit.ac.in
August 2008

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
Hyderabad, India

CERTIFICATE

This is to certify the thesis entitled "RECOGNITION AND RETRIEVAL FROM DOCUMENT IMAGE COLLECTIONS" submitted by Million Meshesha, to the International Institute of Information Technology Hyderabad, India, for the award of the degree of Doctor of Philosophy in Computer Science and Engineering is a record of bonafide research work carried out by him under my supervision, during the period (2002 - 2007). The results embodied in this thesis have not been submitted for any other degree or diploma.

Date

Dr. C. V. Jawahar

Acknowledgment

I am deeply indebted to my advisor Dr. C. V. Jawahar for his dedication, encouragement, motivation and also for his inspiring guidance and supervision throughout my thesis work. I am also greatly indebted to Dr. P. J. Narayanan (PJN) for his concern, encouragement and advise. My sincere thanks also forwarded to Dr. Anoop M. Namboodiri for his critical comments on my journal articles and conference papers. I would also like to thank document understanding research groups at Centre for Visual Information Technology (CVIT), who had made great contribution by sharing ideas, comments and materials. My dearest thanks goes to MNSSK Pavan Kumar, KS Sesh Kumar, A. Balasubramanian, K. Pramod Sakar, Suryakant Patidar, N. V. Neeba and Anand Kumar for their invaluable suggestions and kindness to help me in any way possible. I give my warmest gratitude to Satyanarayana (CVIT) for his unreserved support related to administrative matters. I am thankful to my friends Kula Kekeba, Ermias Abebe and Sebsibe H/Mariam for sharing their ideas, Ethio news and resources.

My thanks also goes to Addis Ababa University (AAU) for granting me study leave with the necessary benefits, without which I could not have been able to join my Ph. D. study here in IIITH, India. I thank Ministry of Education (MOI) for granting me scholarship, and Ethiopian Embassy, India for actively serving as a bridge in matters between me and the other end (AAU and MOI). I also extend my sincere thanks to faculty members and administrative staffs of Faculty of Informatics (AAU).

My greatest gratitude is extended to my family. I forward special thanks to my brother Bahiru Eyassu and his family who cares, advises and plays a critical role in the other side of my life in Ethiopia. My friends in Ethiopia are the engine during my study, especially Hamere Getachew, Daniel Teklu, Enchalew Yifru, Getaw Shumiye, Yonas Bahiru, Yared Girma, Tesfaye Gebeyehu and many more.

Last but not least, I would like to thank all CVIT and IIITH community (faculties, students and administrative staffs) who have a significant contribution in one way or another during my research work.

Abstract

The present growth of digitization of books and manuscripts demands an immediate solution to access them electronically. This will enable the archived valuable materials to be searchable and usable by users in order to achieve their objectives. This requires research in the area of document image understanding, specifically in the area of document image recognition as well as document image retrieval. In the last three decades significant advancement is made in the recognition of documents written in Latin-based scripts. There are many excellent attempts in building robust document analysis systems in industry, academia and research labs. Intelligent recognition systems are commercially available for use for certain scripts. However, there is only limited research effort for the recognition of indigenous scripts of African and Indian languages. In addition, diversity of archived printed documents poses many challenges to document analysis and understanding. Hence in this work, we explore novel approaches for understanding and accessing the content of document image collections that vary in quality and printing.

In Africa around 2,500 languages are spoken. Some of these languages have their own indigenous scripts in which there is a bulk of printed documents available in the various institutions. Digitization of these documents enables us to harness already available language technologies to local information needs and developments. We present an OCR for converting digitized documents in Amharic language. Amharic is the official language of Ethiopia. Extensive literature survey reveals that this is the first attempt that reports the challenges toward the recognition of indigenous African scripts and a possible solution for Amharic script. Research in the recognition of Amharic script faces major challenges due to (i) the use of large number of characters in the writing and (ii) the existence of large set of visually similar characters. Here we extract a set of optimal discriminant features to train the classifier. Recognition results are presented on real-life degraded documents such as books, magazines and newspapers to demonstrate the performance of the recognizer.

The present OCRs are typically designed to work on a single page at a time. We

argue that the recognition scheme for a collection (like a book) could be considerably different from that designed for isolated pages. The motivation here is therefore to exploit the entire available information (during the recognition process), which is not effectively used earlier for enhancing the performance of the recognizer. To this end, we propose self adaptable OCR framework for the recognition of document image collections. This approach enables the recognizer to learn incrementally and adapt to document image collections for performance improvement. We employ learning procedures to capture the relevant information available online, and feed it back to update the knowledge of the system. Experimental results show the effectiveness of our design for improving the performance of the recognizer on-the-fly, thereby adapting to a specific collection.

For indigenous scripts of African and Indian languages there is no robust OCR available. Designing such a system is also a long-term process for accessing the archived document images. Hence we explore the application of word spotting approach for retrieval of printed document images without explicit recognition. To this end, we propose an effective word image matching scheme that achieves high performance in presence of script variability, printing variations, degradations and word-form variations. A novel partial matching algorithm is designed for morphological matching of word form variants in a language. We employ a feature extraction scheme that extracts local features by scanning vertical strips of the word image. These features are then combined based on their discriminatory potential. We present detailed experimental results of the proposed approach on English, Amharic and Hindi documents.

Searching and retrieval from document image collections is challenging because of the scalability issues and computational time. We design an efficient indexing scheme to speed up searching for relevant document images. We identify the word set by clustering them into different groups based on their similarities. Each of these clusters are equivalent to a variation in printing, morphology, and quality. This is achieved by mapping IR principles (that are widely used in text processing) for relevance ranking. Once we cluster list of index terms that define the content of the document, they are indexed using inverted data structure. This data structure also provides scope for incremental clustering in a dynamic environment. The indexing scheme enables effective search and retrieval in image-domain that is comparable with text search engines. We demonstrate the application of the indexing process with the help of experimental results.

The proliferation of document images at large scale demands a solution that is effective to access document images at collection level. In this work, we investigate machine learning and information retrieval approaches that suits this demand. Ma-

chine learning schemes are effectively used to redesign existing approach to OCR development. The recognizer is enabled to learn from its experience and improve its performance over time on document image collections. Information retrieval (IR) principles are mapped to construct an indexing scheme for efficient content-based search and retrieval from document image collections. Existing matching scheme is redesigned to undertake morphological matching in the image domain. Performance evaluation using datasets from different languages shows the effectiveness of our approaches. Extension works are recommended that need further consideration in the future to further the state-of-the-art in document image recognition and document image retrieval.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

It is becoming increasingly important to have information available in digital format for effective access, reliable storage and long term preservation [131][142][182]. This is catalyzed by the advancement of information technology and the need for automatic data processing. In addition, large digital libraries are emerging for archival of multimedia documents. In particular, with storage becoming cheaper and imaging devices becoming increasingly popular, efforts are on the way to digitize and archive large quantity of multimedia data (text, audio, image and video). At this stage, most of the archived materials are printed books and digital libraries are collection of document images.

The first and significant single collection of free electronic books is the Project Gutenberg [71]. Project Gutenberg is the brainchild of Michael Hart, who in 1971 came up with the idea to make famous and important texts freely available to everyone in the world. Since then a number of international digital library projects are initiated and funded by Digital Libraries Initiatives [52]. As a result of which large digital libraries such as the Universal Library (UL) [110], Digital Library of India (DLI) [7], etc. have come to existence. Recently, Google [149] and Microsoft [126] has also partnered with several major libraries and research institutions for massive digitization projects. The Open Content Alliance (OCA) was also conceived by the Internet Archive [12] and Yahoo [6]. This is a collaborative global effort that helps to build and archive multilingual digitized text and multimedia content. The aim of all these efforts is to digitize all literary, artistic, and scientific works for better access to traditional materials, easier preservation, and make documents freely accessible to the global society. The associated key technological issues are how to make these

large document collections searchable to access relevant information in them.

Research in multimedia information retrieval is also getting closer attention in recent years. This has paved the way for a large number of new techniques and systems for content-based image retrieval (CBIR) [33][48][159], content-based video retrieval (CBVR) [169] and text indexing and retrieval [53][88]. These techniques are used to build many successful prototype systems for indexing and retrieval of multimedia data. Most of the present day digital libraries aim at archiving printed books which are not available on-line. They scan pages from the book and make them accessible with some additional meta data. Success of text image indexing and retrieval schemes mainly depend on the performance of optical character recognition systems (OCRs) [162].

The use and application of OCR systems are well demonstrated for many languages in the world [18][131][175]. Recognition systems with high accuracy rate are commercially available for use for Latin scripts and some of the Oriental languages. However, for those languages in Africa and India with their own scripts, much attention is not given for developing robust OCRs that successfully recognize diverse printed text images. Therefore, alternate approaches are proposed by scholars to access content of digital libraries in these languages [43]. The focus is on recognition free approaches for retrieval of relevant documents from large collections of document images. In this research work we present new strategies and novel approaches that enable us in understanding and accessing the content of document image collections.

Machine learning and information retrieval (IR) techniques have the central role for effective access to document images. IR principles enable us to design a search engine that is used for searching and retrieval of relevant documents from large collections. Most of the digitized printed documents are poor (or mediocre) in quality. Recognition and retrieval in such circumstances is long known to be a challenging task [131]. Machine learning offers one of the most cost effective and practical approaches to the design of pattern classifiers for such documents [18][192]. Feedback mechanisms are used to facilitate adaptation to new situations, and to improve its performance accordingly [191]. Deviating from the conventional OCR design, we explore the prospect of designing recognition systems for a collection of printed document images.

These days, many search engines are available for text-based document collections [3]. The use and application of these retrieval systems is mainly dependent on the availability of effective recognition systems. However, there is limited research effort in this direction for the indigenous scripts of African and Indian languages. Since developing robust recognition system (that successfully recognize printed text images of varying quality, font, size and style) is a long-term process, we aim at designing

an indexing and retrieval scheme in the image domain that is capable of retrieving relevant document images without explicit recognition. Our recognition-free retrieval scheme attempts to mimic the IR technology in the image domain.

1.2 Motivation and Problem Setting

The present growth of digitization of documents demands an immediate solution to enable the archived valuable materials searchable and usable by users. This requires research in the area of document understanding, specifically in the area of document image retrieval as well as optical character recognition.

A direct solution to the problem of accessing document images is the use of character recognition systems to convert document images into text, followed by the use of existing text search engines to make them available for the public at large via the Internet. Significant advancement is made in the recognition of documents written in Latin-based scripts and in some of the Oriental languages [131][140][175]. There are many excellent attempt in building robust document analysis systems in industry, academia and research labs [18][32]. High accuracy OCR systems are reported for Latin scripts with excellent performance on good quality documents with limited variations in fonts [86][177]. On the contrary, there is only limited research effort, that to in uncoordinated manner, for the recognition of the indigenous scripts of African [124] and Indian [140] languages. Scripts of these languages pose many challenges for document understanding. Nuances and variability of the scripts are not well understood or studied. There are large number of characters in the scripts. Symbols are often complex in shape. There are also large number of similar, confusing characters. They often lack a standard representation for the fonts and encoding, in addition to lack of support from operating systems, browsers and keyboard for enabling OCR and other software applications. These issues add to the complexity of the design and implementation of document image recognition systems and document image retrieval systems.

In addition, digitized documents are diverse in nature. These documents vary in quality, language, printing, etc. Such challenges complicate the problem of document analysis and understanding to a great extent even for Latin scripts. Most of the recent research in OCR has been centered around building fully automatic classification systems with very high generalization capability [131]. However, when it comes to the suitability of converting an old book to text and providing a text-like access, the present OCRs are found insufficient [59]. The performance of these recognizers decline as diverse collection of documents (with unseen fonts and poor in quality) are

submitted for recognition. Furthermore, the requirement of different OCR systems to deal with different languages is the drawback of recognition based approaches.

As a result, there is an increasing focus on recognition-free approaches for indexing and retrieval from poor quality document image collections. There have been many approaches proposed for retrieving document images based on image level information. An attempt was made for retrieval of relevant documents from printed [43][96][98][114] [161], and handwritten [23][80][154] documents. However, information retrieval in document image databases has become a growing and challenging problem. This is because of the diversity in quality of the documents that are digitized and available for use. An additional challenge in searching for relevant document images using query words is the existence of word-form variants. The exact word that is present in the document being searched could be a variant of the query word. Diversity of documents (that vary in quality, scripts, fonts, sizes and styles) that are archived in digital libraries also complicate the problem of document analysis and understanding to a great extent. This requires better document image understanding schemes to make the content of these documents accessible to users through indexing and retrieval of relevant documents.

Thus, searching and retrieval for relevant document images can be done with the help of the following approaches: Recognition-based, Recognition-free or a combination of these two approaches. Recognition-free approaches search for retrieval of relevant documents without explicit recognition (in the image domain). Recognition-based approaches, on the other hand, requires an OCR system to convert document images into text and then search in the recognized text. Effectiveness of recognition-based approach greatly depends on the availability of robust character recognition systems, which most non-Latin scripts are lacking at present including indigenous scripts of Indian and African languages. Designing robust recognizers that are applicable for documents varying in quality, fonts, sizes and styles is known to be a long term solution. In contrast, document image retrieval without explicit recognition is a direct approach to access the digitized document image itself with certain level of performance. Specially, for historical printed and handwritten document images it is a promising approach to design an applicable retrieval system.

Hence, in this work we explore the problem of character recognition as a long term solution and document image retrieval as a short term solution. We design novel approaches for accessing the content of printed document image collections that vary in quality, scripts, and printing (fonts, point sizes and type styles).

1.3 Recognition and Retrieval from Document Images

Large collections of paper-based documents and books are being brought to the digital domain, through the efforts of various digitization projects. Digital libraries play a critical role in organizing, preserving and providing access to the various information resources of society [17]. They are advantageous as a means of easily and rapidly accessing books, archives and images of various types. However they should be enabled with the necessary technological means to ease access and use of the archived document images. Optical character recognition (OCR) as well as document image retrieval are the two basic means of access to the document image collection.

An OCR system converts scanned document images into textual representations. The OCR engine accepts a scanned image of a text document, and passes the image through many stages before it is translated into a computer understandable format. The scanned document could be either printed, typewritten or handwritten. The document image is first preprocessed to prepare it for recognition. During preprocessing, the image undergoes some image enhancements such as filtering out noise and increasing the contrast, skew correction, normalization, etc. Then, the image should be segmented to identify lines, words and finally characters in the text document, resulting in a whole bunch of smaller images, each representing a character or connected components. The raw digitized data is then mapped to a higher level using feature extraction schemes that extracts special characteristics and patterns of the character image. The higher level information is then passed to the classifier for identification/classification of the unknown characters given the information learned from the known sample database. Once the OCR converts the document image to text, the result is post-processed to verify and correct recognition errors. The final output is viewed with the help of text processing applications.

Information retrieval (IR) scheme is designed to locate relevant documents in response to a query that describes a user's information need [16]. Modern information retrieval (IR) systems use a range of statistical and linguistic tools to maximize the effectiveness of searching textual documents. One of the main problems that characterize natural language texts is the fact that a single word may occur in some (or many) different forms, as determined by the language's inflectional and derivational morphologies. This is a problem for IR systems even in a language like English that has a relatively simple morphology; it is a much greater problem for languages (such as Amharic, Hindi, etc.) with more complex morphologies that can yield hundreds or even thousands of variants from a single word. Document image retrieval systems

search for relevant documents in the image space, without explicit recognition. The development of indexing and retrieval scheme passes through many steps. Given a document image, it is preprocessed and segmented into word level. Then, features are extracted for representation of the segmented word images. For efficient searching and retrieval from document image collections, document images needs to be indexed. The indexing process in turn involves the application of IR principles so as to identify content bearing words (or index terms). This requires document clustering with the help of effective matching scheme for detecting stopwords, grouping word variants and ranking relevance of documents in each cluster. Clusters are organized using indexing data structure such that, later on search takes place just in the index list for retrieval of relevant documents for users query.

1.4 Objectives of the Study

In this thesis work, we address the following problems.

- Recognition of textual content in document images. To this end, an attempt is made to:
    1. Analyze indigenous African scripts and their language rules to ease document analysis and understanding.
    2. Design an OCR for recognition of printed document images in Amharic language, which has got less emphasis up to now.
    3. Explore machine learning and pattern recognition algorithms for designing feature extraction, classification and feedback mechanism for recognition of document images.
    4. Propose an architecture and learning algorithms of self adaptable OCR (for recognition of document image collections) that is capable of learning and adapting to new pages or documents to improve its performance.
    5. Performance analysis of the OCRs on synthetic and real-life document image collections that are poor in quality and vary in printing.
- Indexing and retrieval of relevant document images without explicit recognition. To this end, we attempt to:
    1. Implement a word spotting approach for content-based retrieval from printed document image collections in Amharic, English and Indian languages.

2. Enable existing word image matching scheme to perform morphological matching so as to take care of word form variations in the language, in addition to degradation and printing variations.
3. Design a feature extraction scheme that is invariant to degradations and printing variations frequently seen in printed document images.
4. Construct an indexing scheme following the IR principles for efficient searching in document image collections for retrieval of relevant ones.
5. Evaluate performance of the retrieval scheme on synthetic and real-life document image collections that vary in degradation, script, printing and word forms.

1.5 Significance of the Work

This research work achieves significance due to the following main facts.

- With massive digitization projects around the globe, large collection of document images are emerging to a great extent. We need to design efficient access techniques to the content of these documents to enable them achieve their intended objective.
- The present advancement in the OCR technology demands for bridging the gap between document image recognition as well as document image retrieval approaches.
- Document images in digital libraries are from diverse languages. Most people in Africa and India look for documents in local/regional language. This raises the need to ease accessibility of the available relevant documents in African and Indian languages.
- Exponential growth of computational resources paves the way for alternate recognition strategies based on machine learning schemes. The new generation OCRs need not recognize an isolated word. It can learn from its recognition experience obtained from the rest of the printed documents.
- An immediate need for effective access to digital libraries initiate document retrieval without explicit recognition. The success of IR capabilities on born-digital textual documents, has raised the expectation on document image retrieval. Document image retrieval is no longer a simple matching. Efficiency and scalability demands the application of IR principles in the image domain.

1.6 Major Contributions

In this research work, we focus on the recognition as well as retrieval from document image collections. Some of our contributions are the following.

1. Study scripts and language rules of African and Indian languages for document analysis and understanding. To the best of our knowledge, this is the first work that reports the challenges toward the recognition of indigenous African scripts and a possible solution to the Amharic script.

2. Design an OCR for recognizing printed documents in Amharic language. The Amharic OCR is extensively tested on real-life document images such as books, magazines and newspapers. The performance results are encouraging to extend it to other African indigenous languages.

3. Propose an architecture and demonstrate the application with a set of prototype algorithms of self adaptable recognizer for document image collections. Our recognizer is data-driven such that it incrementally learns from the feedback propagated during the recognition process for performance improvement.

4. Propose an efficient word image matching scheme to effectively compare printed document images in presence of script variability and word-form variations. Combined feature extraction scheme is also designed that is invariant to printing variations and degradation.

5. Construct an indexing scheme for efficient searching in document images. We map IR principles (that are widely used in text search engine) for relevance ranking in the image domain. With this design we achieve search efficiency comparable with text search engines.

1.7 Scope of the Work

This work attempts to further the state-of-the-art in document image recognition as well as in document image retrieval. Our main focus here is on printed document images that are archived in digital libraries and other applications. The objective of this study is not to design recognition-based (text-based) search engine for printed documents, rather to address issues related to, on the one hand, document image recognition and, on the other hand document image retrieval.

With the help of machine learning and information retrieval approaches we design better document image access schemes. We follow machine learning and pattern recognition approaches in proposing algorithms for feature extraction and classification, as well as in designing feedback mechanism for document images recognition. In addition, schemes for feature extraction, word image matching, clustering and indexing are designed for document image retrieval. The IR principles are mapped from text to image domain for document image indexing and retrieval.

In our work we assume that preprocessing of document images and segmentation of text images into words, that are required for the recognition as well as for the retrieval tasks, are available and, hence they are beyond the scope of the present work. We also not address issues related to system development for designing prototypes.

We report in this work the challenges toward the recognition of indigenous African scripts, but a possible solution is given only for Amharic script. Our present research in document analysis and understanding gives more emphasis to the recognition of Amharic printed documents. The retrieval scheme also considers printed document images in Amharic language along with some of the Indian language documents. The present work does not consider cross-lingual retrieval of documents.

1.8 Organization of the Thesis

This thesis is organized in eight chapters. The first chapter provides an overview of the general background and the problem setting. Motivation behind the present work and the major contributions are also briefly described. In chapter two we present the state of the art in the area of Optical Character Recognition and Document Image Retrieval.

Chapter three introduces African and Indian languages related issues for document image recognition and for document image retrieval tasks. Script analysis and language rules for word formations are presented. We undertake script analysis to identify the underlining unique features of the Amharic script. Chapter four introduces the application of character recognition for Amharic language. We design an OCR for the recognition of Amharic script. Performance analysis is also presented on real-life document images. Chapter five provides an architecture of self adaptable OCR for the recognition of document image collections. The various machine learning approaches incorporated in the learning processes are described. Experimental results are presented to show its applicability for printed document image collections.

Chapter six discusses the proposed matching and feature extraction scheme for content based document image retrieval by comparing word images. Morphological

analysis by partial matching is enabled in the image domain to identify word form variations. Frequently seen degradations in printed document images are modeled for building dataset that are used for evaluation of the performance of the proposed scheme. Chapter seven discuses indexing scheme for organizing a collection of printed document images. The scheme is helpful to develop a scalable system that enhance the efficiency of searching and retrieval in the image domain. The application of IR principles for relevance ranking in the image domain is also presented. Finally, chapter eight provides summary of the work, concluding remarks and scope for future work.

Chapter 2

State of the Art

2.1 Digital Library

Digital Libraries have received wide attention in the recent years allowing access to digital information from anywhere across the world. They have become widely accepted and even preferred information sources in areas of education, science and even with other information needs [17]. The rapid growth of Internet and the increasing interest in development of digital library related technologies and collections [122][125] helped to accelerate the digitization of printed documents in the past few years.

A digital library is a library in which collections are stored in digital formats (as opposed to print, microform, or other media) and accessible using computers [69]. The digital content may be stored locally, or accessed remotely via Internet. Digital libraries play a critical role in organizing, preserving and providing access to the various information resources of society. Digital libraries are advantageous as a means of easily and rapidly accessing books, archives and images of various types. Traditional libraries are limited by storage space; digital libraries have the potential to store much more information, simply because digital information requires very little physical space to contain them. Digital libraries can immediately adopt innovations in technology providing users with improvements in electronic and audio book technology. In summary, the following are some of the advantages of having a digital library [17][189].

- *No physical boundary:* The user of a digital library need not to go to the library physically; people from all over the world can gain access to the same information, as long as an Internet connection is available.

- *Structured approach:* Digital libraries provide access to much richer content in

a more structured manner, i.e., we can easily move from the catalog to the particular book then to a particular chapter and so on.

- *Information retrieval:* The user is able to use any search term (such as word, phrase, title, name, subject) that are supported by the search engine to search the entire collection that satisfy their need.
- *Preservation and conservation:* An exact copy of the original can be made any number of times without any degradation in quality.
- *Resource sharing:* A particular digital library can provide a link to any other resources of other digital libraries very easily; thus a seamlessly integrated resource sharing can be achieved.
- *Multiple accesses:* Any number of users can access the same document at the same time. In addition, digital libraries can be used at any time whenever users need information.

With all these advantages of digitization, Sellen and Harper [168] report that, at present, paper documents remain the medium of choice for reading, even when the most high-tech technologies are to hand. They point four principal reasons for this: (i) paper allows flexible navigation through the content; (ii) paper assists cross-referencing of several documents at one time; (iii) paper invites annotation; and (iv) paper allows the interweaving of reading and writing. It is illuminating to bear these considerations in mind when identifying obstacles to the delivery of document images via digital libraries. Of course, efforts are underway to commercialize electronic document displays offering even more of the affordances of paper including flexibility, low weight, low power, and low cost [20].

2.1.1 Current Status

There are many thousands of digital library projects initiated and funded by Digital Libraries Initiatives in US, Europe, Asian and other part of the world [52]. This is done with the aim of digitizing the important documents, both to preserve them for future generations and to make them easily accessible. These days, large scale digitization projects are underway at Google [149], the Million Book Project [110], MSN [126], Yahoo [6], etc. With continued improvements in book handling and presentation technologies, such as optical character recognition, ebooks, and development of alternative business models, digital libraries are rapidly growing in popularity as demonstrated by many projects around the world. Just as libraries have ventured

into audio and video collections, so have digital libraries such as the Internet Archive. There are many collaborative digitization projects throughout the globe, such as the Collaborative Digitization Project [150], Open Content Alliance [12], etc. These projects help to establish and publish best practices for digitization and work with regional partners to digitize cultural heritage materials.

Among others, the leading projects in the field of digital archive creation and management include [189] project Gutenberg, Google Book Search, Windows Live Search Books, Internet Archive, Cornell University, The Library of Congress World Digital Library, The Digital Library at the University of Michigan, CMU's Universal Library [110], etc.

The Million Book Project (or the Universal Library), led by Carnegie Mellon University achieved the target of digitizing a million books by 2007 [110]. The project has been working with government and research partners in India, China, Egypt, etc. As a result of which the Digital Library of India project [7] was initiated in November, 2002.

2.1.2 Digital Library of India

Since its inception in the year 2002, the digital library of India project currently digitizes and preserves books, though one of the future avenues is to preserve existing digital media of different formats like video, audio, etc. The scanning operations and preservation of digital data takes place at different centers across India, Regional Mega Scanning Center (RMSC) [7]. The RMSCs themselves function as individual organizations with scanning units established at several locations in the region.

The Digital Library of India (DLI) project aims to digitally preserve all the significant literary, artistic and scientific works of people and make it freely available to anyone, anytime, from any corner of the world, for education, research, etc. The project has been successfully digitizing books, which are a dominant store of knowledge and culture. It hosts close to one third of a million books online with about 70 million pages scanned at almost 30 centers across the country [7]. The scanning centers include academic institutions of high repute, religious and government institutions. Summary of the diverse collections of documents archived in Digital Library of India is presented in Table 2.1.

2.1.3 Challenges

The rapid growth of digital libraries worldwide poses many new challenges for document image analysis research and development. Digital libraries promise to offer

Documents	Diversity of Books
Languages	Hindi, Telugu, Urdu, Kannada, Sanskrit, English, Persian, European, others
Typesets	Letter press, Offset printer, Typewriter, Computer Typeset, Handwritten
Fonts	Ranging from 10 to 20 fonts in each language
Styles	Italics, Bold, Sans serifs, Underline, etc.
Sizes	8 to 45 points
Year of Publication	Printed books from 19, 20 and 21st centuries Ancient manuscripts to 2005

Table 2.1: Diversity of document collections in DLI. They vary in languages, fonts, styles, sizes and typesets. Books that are published since 1850 are archived.

more people access to larger document collections, and at far greater speed, than physical libraries can. But digital libraries also tend, for many reasons, to serve poorly, or even to omit entirely, many types of paper-based printed or handwritten documents [20]. These documents, in their original physical (undigitized) form, are readily legible, searchable, and browseable, whereas in the form of document images accessed through digital libraries they often lose many of their original advantages. This is because existing document image understanding techniques can not fully give solution to the challenges of digitized document images. The unusual wide variety of document images found in digital libraries, representing many languages, cultures, and historical periods, tend to pose particularly severe challenges for present day digital image analysis systems. These systems are not robust in the face of multilingual text and non-Latin scripts, unusual typefaces, and poor image quality [21].

Baird [20] noted the immediate need for redesigning document image analysis systems to give solution mainly in the following areas.

The Need for Accurate Transcriptions of Text: The central classical task of digital image analysis research has been, for decades, to extract a full and perfect transcription of the textual content of document images [53]. Although perfect transcriptions have been known to result, no existing OCR technology, whether experimental or commercially available, can guarantee high accuracy across the full range of document images of interest to users. Even worse, it is rarely possible to predict how badly an OCR system will fail on a given document.

Indexing and Retrieval: The indexing and retrieval of document images are critical for the success of digital libraries. Most reported approaches for retrieval of document images first attempt recognition and transcription followed by indexing and search operating on the resulting (often errorfull) encoded text, using standard "bag-of-words" information retrieval methods [53]. Although for some retrieval tasks, error rates typical of commercial OCR machines do not seriously degrade recall or precision of statistical "bag-of-words" methods, some textual analysis tasks (e.g. depending on syntactic analysis), whether modeled statistically or symbolically, can be derailed by even low OCR error rates.

The attempts made on word spotting for retrieval from document images without explicit representation is encouraging. This approach seems to offer the greatest promise of large improvements in effectiveness of document image retrieval (if not in speed). It needs further investigation to reach the level of text information retrieval methods. We review hereunder some of the efforts exerted in similar directions for developing systems for document image recognition and also for document image retrieval in the image domain.

2.2 Recognition of Document Images

Recognition of document images is at the heart of any document image understanding system [131]. Optical character recognition (OCR) systems take scanned images of paper documents as input, and automatically convert them into digital format for computer-aided data processing. The potential of OCR for data entry application is obvious: it offers a faster, more automated, and presumably less expensive alternative to the manual data entry devices, thereby improving the accuracy and speed in transcribing data into the computer system. Consequently, it increases efficiency and effectiveness (by reducing cost, time and labor) in information storage and retrieval.

OCRs have wide range of applications in government and business organizations, as well as individual companies and industries. Some of the major applications of OCR include [130][142]: (i) Library and office automation, (ii) Form and bank check processing, (iii) Document reader systems for the visually impaired, (iv) Postal automation, and (v) Database and corpus development for language modeling, text-mining and information retrieval.

Optical character recognition has a relatively long history. The technology was invented in the early 1800s, when it was patented as reading aids for the blind. In 1870, C. R. Carey patented an image transmission system (the retina scanner) using a mosaic of photocells, and in 1890, P.G. Nipkow invented the sequential scanner,

whereby an image was analyzed line by line [67][120]. This was a major breakthrough for reading machines. However, the practical OCR technology used for reading characters was introduced in the early 1950s as a replacement for keypunching system used in business [131]. A year later, D.H. Shephard developed the first commercial OCR for typewritten data.

It is in 1960 that OCR began to make a noticeable impact on commercial and government data processing installations. The first postal address readers and the social security administration machines to read typewritten documents were installed in 1965. In 1970 there emerged the first OCR system capable of reading a wide variety of forms size, from small documents to large pages. The 1980's saw the emergence of OCR systems intended for use with personal computers. An early successful attempt in implementing character recognition as an aid to the visually handicapped was made by Tyurin in 1990 [67]. Nowadays, it is common to find PC-based OCR systems that are commercially available. However, most of these systems are developed to work with Latin-based scripts [140].

OCR can be done offline or online. OCR converts document images to text that are printed or handwritten. Handwriting recognition are often called optical handwriting recognition (OHR) analogous to optical character recognition (OCR). OCR usually deals with the interpretation of offline data which describes printed or typewritten objects [144]. The goal of OHR is to interpret the contents of the handwritten data and generate a description of that interpretation in the desired format. The OHR task is also referred to as off-line handwriting recognition or online handwriting recognition [39]. On-line handwriting recognition deals with a data stream which is coming from a transducer while the user writes, and off-line handwriting recognition deals with a dataset which has been obtained from a scanned handwritten document. We are concerned in this work on the OCR of printed document images.

2.2.1 Overview of OCR Design

Figure 2.1 shows the framework of OCR. Most of the designs in OCR follow a modification of this architecture. Given a page for recognition, first it is preprocessed. The aim of the preprocessing module is to prepare the image for recognition. Preprocessing involves binarization, skew correction and normalization. It undergoes some image enhancements such as filtering out noise and increasing the contrast. Then, the image is segmented to separate the characters from each other. Segmentation occurs at two levels. On the first level, text, graphics and other parts are separated. On the second level, text lines, words and characters in the image are located. Information

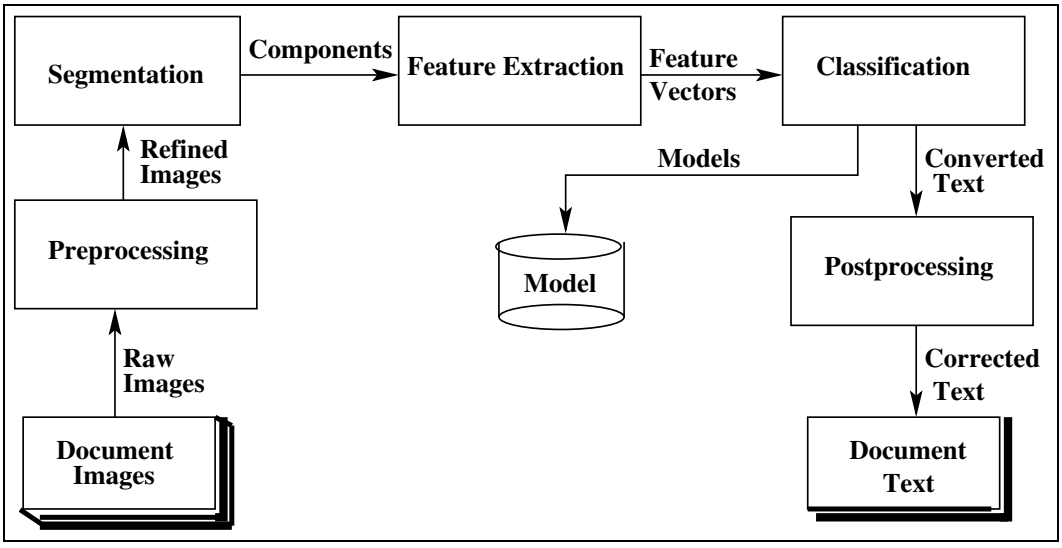


Figure 2.1: Overview of conventional approach to OCR

from connected component analysis and projection analysis can be used to assist text segmentation. Segmentation is followed by feature extraction, which is concerned with the representation of the object.

Feature extraction and classification are the heart of OCR. The character image is mapped to a higher level by extracting special characteristics and patterns of the image in the feature extraction phase. Feature extraction is expected to make the image invariant to rotation, translation, scaling, line-thickness, etc. It could also remove redundant information to compress the data amount and a lot of other things. A number of global and local features have been employed for this, including profiles, moments, structural features, discrete Fourier descriptors, etc. The classifier is then trained with the extracted features for classification task. The classification stage identifies each input character image by considering the detected features. A range of classifiers are in use for these purpose. Classifiers such as Template Matching, Neural Networks, Syntactical Analysis, Hidden Markov Models, Bayesian theory, SVM, etc. have been explored. For improving the recognition result, post-processing module is incorporated. The post-processor is typically intended to improve accuracy by detection and correction of OCR errors. Contextual information such as character n-gram, dictionary, semantic knowledge and domain knowledge are exploited in post-processing.

Below we review some of the notable OCR systems developed in industry, academia and research institutions.

2.2.2 Commercial and Free OCR Systems

Recognition of scanned document images using OCR is now generally considered to be a solved problem for some scripts [74]. This is because high accuracy OCR systems are reported for some languages in the world that use Latin [131] and non-Latin scripts [175].

Many commercial packages are available for Latin scripts. Some of them are the following: *OmniPage Pro from Caere, FineReader from ABBYY, TextBridge from Xerox, Capture from Adobe*, etc. These packages do an impressive job on high quality original documents with accuracy rates in the high nineties. There are also sophisticated OCRs for languages such as Chinese [175], Japanese [195] and Arabic scripts [93]. A commercial OCR software, 'Presto! DanChing' from NewSoft [45] performs OCR on Chinese, Japanese and Roman characters. For Arabic language the pioneer 'Sakhrs' automatic reader [11] is available commercially that supports also English, French and some other languages. Recently, Caere announces also that, in addition to English and thirteen Western European languages, it provides the first Omnifont OCR software for the Arabic language.

There are also commercial products for postal address reading [170] and a reading machine for a blind or visually impaired [102]. The architecture of the postal address reading machine is designed to achieve correct interpretation of text, as well as high speed in performing the interpretation. The reading machine finds and interprets addresses on a stream of postal letters. The addresses can be either machine-printed or handwritten. Here, the primary subtasks correspond to finding the block of text corresponding to the destination address, recognizing characters and words within the address, and interpreting the text using postal directories. An integrated OCR with speech output system for the blind has been marketed by Xerox and others for English language [102]. The reading machine includes a user input device including a user activated button that produces a signal that causes the computer to enter a definition mode to read a definition of current word being spoken by the reading machine.

Unlike commercial OCRs, there are few free and open source OCRs packages available. Some of these are (i) GOCR (or JOCR) [64] from source forge (an open source OCRs program), (ii) Orad from GNU (an open source OCRs program), (iii) Tesseract OCR [66] from Google (a free OCR engine), etc. Microsoft Office application also contains Microsoft Office Document Imaging (MODI) [44] that supports editing documents scanned by Microsoft Office Document Scanning. It was first introduced in Microsoft Office XP and is included in later Office versions including Office 2003 and Office 2007 [189]. Among the many functionality of MODI is it allows users to scan

single or multi-page documents and produce editable text from a scanned document using OCR.

Though OCR systems developed by commercial vendors give high recognition results on good quality pages, evaluation of current OCRs show that the diversity of document collections (language-wise, quality-wise, time-wise, etc.) reduce the performance of OCR systems greatly [127]. The creation of new fonts, the occasional presence of decorative or unusual fonts, and degradations caused by faxed, aged and multiple generation of copies, continues to make existing OCRs a topic of research for better accuracy and reliability [20][170]. This is because most of the present systems has been centered around developing fully automatic systems targeted toward the recognition of a single page at a time. Recent evaluation of commercial software by Lin [111] argue that document recognition research is still in great need for better accuracy and reliability, complementary contents, or downstream information retrieval.

2.2.3 OCRs Development by Research Labs and Academia

A number of OCR systems are also reported by research institutes. Kahan *et al.* [86] described a system that recognizes printed text of various fonts and sizes for the Roman alphabet. The system incorporates different techniques for these purpose. Thinning and feature extraction are performed directly on a graph of the run-length encoding of a binary image. The strokes (stroke endpoints, junctions, etc.) and other shape features are extracted and mapped, using a shape clustering approach (that approximate clusters by rectangular neighborhoods), into binary features. The binary features are then fed into a statistical Bayesian classifier. Confusion classes are then disambiguated through contour analysis using exterior contour tracing algorithm, and characters suspected of being merged are broken and reclassified. Layout context and linguistic context are also applied for getting contextual information.

Baird [18] described the architecture of a versatile page reader that can be readily adapted to multiple-typeface of English text, and single-typeface of Greek, Tibetan, Swedish, chess notation, and typeset mathematics. Baird claims that following an engineering strategy enable an OCR system to easily adapt to various languages through controlling separately and independently the effects of variations in symbol sets, typefaces, sizes of text, page layouts, linguistic contexts, and imaging defects due to a change in language. Algorithms of all language-specific rules are avoided; instead, automatic learning from examples and generalized table-driven methods are adopted. For geometric page layout analysis they used a global-to-local strategy that requires no

prior knowledge of the symbol set. A post-processor is designed that uses dictionaries, punctuation rules and lexical constraints. This is done through data-directed filtering algorithms in a uniform and modular manner. For character recognition, a hybrid of structural shape analysis and Bayesian statistical classification, is implemented. The classifier is trainable by example and usually done offline.

An omni-document recognition system is presented by Bokser [32] which handles documents ranging from office memos and magazine articles to spreadsheets and parts lists. The system has four primary phases: segmentation, image recognition, ambiguity resolution, and document analysis. The recovery mechanism for improperly segmented text is resegmentation during the image recognition phase, which attempts to cleave segments that contain joins, glue segments that contain splits, and cleanse segments that contain noise. The feature vector used is of fixed length. It is obtained by dividing the input image into a fixed number of zones, and assigning to each a value related to the relative density of black pixels in that zone. A robust neural network classifier with back-propagation is used for classification that needs fewer parameters to outline the decision regions. Post-processing module is integrated to resolve ambiguities during recognition based on language-specific knowledge and document properties to pick the best choice. Character trigram probabilities are estimated by combining several group n-gram probabilities.

Xu and Nagy [192] propose a prototype extraction method for document-specific OCR systems and segmentation-free word recognition. Recognition is based on representative character prototypes. The recognizer adapt to new pages or documents by automatically extracting the required prototype bitmaps. Training samples (of the most frequently appearing characters in writing) are automatically generated from a portion of document page to be recognized using the corresponding transcripts. A commercial OCR package is used for generating the text transcript. Bayesian word-shift algorithm is used for automatically extracting prototypes from unsegmented text images using the known transcript of the image. Then a document specific OCR system is trained with a portion of characters sampled from the given document image. Template matching is used for segmentation-free word recognition. The system adapt to more characters (that are not appearing in the training set) by generating new templates and adding them to the existing template set.

A great advancement is observed to bring a versatile OCRs that handle documents varying in printing, layout, etc. There are commercial and free OCRs available that achieve high recognition accuracy on documents with reasonably good quality and standard fonts. Most of the research concentrates for Latin-based script. It is obvious that further research is still needed to enhance OCRs flexibility to handle the challenge

posed by the archived document image collections.

2.2.4 OCR Systems for Oriental Languages

Many scholars have pursued research on Oriental languages. Optical character recognition systems for these languages have been studied extensively in the literature. There are a number of reports available for the recognition of scripts of oriental languages as compared to other non-Latin scripts. Document analysis and understanding in these scripts faces specific challenges because of [175]: (i) the existence of large number of characters in the scripts, (ii) the degree of complexity of character formation, (iii) the existence of various visually similar characters in the scripts. Thus, character recognition becomes a tough task due to the large number of classes and high degree of inter-class similarity.

Research in Japanese OCR began in Japan in the late 1950s. Frequently used classifiers for Japanese OCR are the City-block distance classifier, the linear classifier, the projection distance classifier, quadratic classifier, etc. [94]. Statistical dimensionality reduction techniques (such as discriminant analysis and principal component analysis (PCA) are used to select smaller member of discriminative features. A document recognition system for Kanji/Japanese characters is presented by Yamashita *et al.* [195]. The Japanese character set is a combination of several scripts. Kanji (Chinese characters), Kana (Japanese consonants and syllabary characters), Roman and Arabic numerals [172]. The Japanese OCR system provides a flexible framework for object-oriented management of data and processing modules. The framework allows the user to change the combination of processing modules and to select pipelining (parallel processing) or sequential processing. The system includes processing modules for layout analysis functions such as blob detection, block segmentation, and model matching, and for character recognition functions such as feature extraction and classification, post-processing, and error correction through a user interface. The feature data consists of the distribution of local contour directions and the distance from the frame of the buffer to the nearest black pixel. The set of extracted feature data is matched with every template in a recognition dictionary, and city-block distances are calculated for classification. The results obtained by the OCR are sent to the dictionary-based contextual post-processor to improve the recognition accuracy.

Chinese character recognition is also one of the active research area since 1966 [175]. A large portion of the research work in Chinese character recognition falls into two main groups [175]: the statistical classification and structural analysis. The first group of methods extract feature vectors from training samples, compute the statisti-

cal distributions, and classify an unknown input by extracting its feature vector and estimating the most likely character class to which it belongs. The second group of methods recognize a character by precisely matching the individual strokes between the input character and a number of template characters. Other research studies include neural networks, combination of multiple classifiers, stroke extraction and analysis, recognition of character strings with touching characters, etc. Applications include bank cheque processing, postal address recognition, document data extraction, etc. High-Precision Two-Kernel Chinese Character Recognition in General Document Processing Systems is reported by Zhao and Lee [200]. Two recognition engines are used to recognize the character images. The weights of these kernels and features are calculated from the relative stroke widths of character images. Post-processing module calculates confidence values for different candidates and then select the most confident candidate as the OCR result.

Current research activities for Korean/Hangul character recognition is classified into either holistic or segmentation-based approach [175]. In a holistic method, a character itself is a recognition unit and therefore grapheme's do not have to be segmented for character recognition. Most of the holistic approaches extract a global feature, and adopt a multi-stage strategy to alleviate the difficulty of large number of character classes. In this strategy, a rough classification or pre-classification is performed first, and then a final decision is made by a statistical pattern classifier such as neural networks, stroke matching, hidden Markov models (HMM), etc. In a segmentation-based method, on the other hand, a grapheme is the recognition unit and all the grapheme should be segmented from a character prior to their recognition. Apart from these two major approaches, on-line recognizers have also been tried on off-line data. They convert the character image (off-line) into a temporal data (on-line) by estimating the stroke sequence, and then apply a high-performance on-line classifier upon the converted data. The success of this approach depends on the consistency of stroke sequencing.

A system for recognition of Arabic text is reported by scholars. A survey is provided for offline Arabic characters recognition [8] and offline Arabic handwriting recognition [113]. Recognizing Arabic script is very challenging [93] because of two special reasons. Character orthography is cursive in general and Arabic letter are normally connected on the base line. Different Arabic characters may have exactly the same shape, and are distinguished from each other only by the addition of a complementary character by positioning above, below, or within the confines of the character. Arabic characters are always connected even when typewritten or machine printed. Most of the difficulties lie in the segmentation of words or sub-words into individual characters

for subsequent classification and recognition. It is therefore of a primary importance to tackle the problem of segmentation in any potentially practical Arabic OCR system. Outer contours and structural techniques are used for segmenting words into characters. Moment invariant and Fourier descriptors are used for character representation. For recognition, Karhunen-Loeve transform, the viterbi algorithm, Hough transform methods are implemented. Gillies *et al.* [63] describes a system for the recognition of Arabic text in document images. Various techniques have been implemented for the recognition task. Horizontal and vertical projection profile features, and chain code features based on a chain-like representation of the edges (borders or contours) are extracted. A neural network classifier is used for recognition.

Script	Works	Features	Classifiers
Hindi/ Devanagari	[22][25][26] [27][106]	Density, moments, structural	multiple connectionist, distance-based
Telugu	[103][134] [153]	Connected components	Template matching, fringe distance
Bangla	[40][55][137] [138][139]	Structural; run-based template; Curvature Features	Tree classifier; template matching
Tamil	[10]	Gabor filters	Artificial neural networks
Malayalam	[84]	Entire image as features	Binary decision tree
Oriya	[42][129]	Topological; stroke-based; water overflow from a reservoir	Hybrid tree classifier; matching
Kannada	[13][117]	Structural features; fisher discriminant analysis	Support Vector Machine (SVM)
Gurmukhi	[105][107] [108] [109]	Structural features	Binary tree; Nearest neighbor classifier
Gujarati	[9][50]	Moments	Hamming distance; K-Nearest neighbor
Urdu	[141]	Structural features; run-based template	Tree classifier
Bilingual	[41][81]	Structural features; run-based template	Neural Networks; SVM

Table 2.2: Existing works for the recognition of document images in Indian languages.

2.2.5 OCRs for Indian Scripts

There are also studies conducted on the recognition of Indian languages [39][140]. Summary of the works done are presented in Table 2.2. A comprehensive review of Indian scripts recognition is reported by Pal and Chaudhuri [140]. The review revealed that most works are concerned about the development of OCR for Devanagari and Bangla scripts; the two of the most popular languages in India. Structural and topological features with tree-based and Neural Network classifiers are mainly used for the recognition of Indian scripts. A major break-through in this respect is the Devanagari and Bangla OCR system developed at Indian Statistical Institute [140], which is made commercially available by CDAC (Centre for Development of Advanced Computing).

Printed Devanagari character recognition has been attempted based on K-nearest neighbor (KNN) and Neural Networks classifiers [27][41]. For classification purpose, the basic, modified and compound characters are separated. Modified and basic characters are recognized by a structural features (such as concavities and intersections) based binary tree classifier. A hybrid of structural and run-based template features were used for the recognition of compound characters. These results were also extended to Bangla script [40][41]. A complete OCR system for printed Bangla script is presented by Chaudhuri and Pal [40], where the compound characters are recognized using a tree classifier followed by template-matching approach. Stroke features are used to design the tree classifiers. The character unigram statistics is used to make the the tree classifier efficient. Several heuristics are also used to speed up the template matching approach. A dictionary-based error-correction scheme has been integrated where separate dictionaries are compiled for root word and suffixes that contain morpho-syntactic informations as well. A similar approach was tried for Urdu [141] in which a tree-classifier is employed for the recognition of Urdu script after extracting a combination of topological, contour and water reservoir features. Antanani and Agnihotri [9] reported character recognizer for Gujarathi script that uses minimum Euclidean distance, hamming distance and KNN classifier with regular and Hu invariant moments.

Lehal and Singh reported a complete OCR system for Gurmukhi Script [108]. They use two feature sets: primary features like number of junctions, number of loops, and their positions and secondary features like number of endpoints and their locations, nature of profiles of different directions. A multistage classifications scheme is used by combining binary tree and nearest neighbor classifier. They supplement the recognizer with post-processor for Gurmukhi Script where statistical information of Panjabi language syllable combinations, corpora lookup and certain heuristics have

been considered. An OCR system were also reported on the recognition of Tamil and Kannada script [148]. Recognition of Kannada script using Support Vector machine (SVM) has been proposed [13]. To capture the shapes of the Kannada characters, they extract structural features that characterizes the distribution of foreground pixels in the radial and angular directions.

For the recognition of Telugu script, Negi *et al.* [134] proposed a compositional approach using connected components and fringe distance template matching. An OCR system for Oriya script was reported recently [129]. Structural features (such as vertical line, number and position of holes, horizontal and vertical run code) are extracted for modifiers (matra) and run length code, loop and position of hole for composite characters, and a tree-based classification is developed for recognition. The system has been integrated with spell checker with the help of dictionary and a huge corpus to post-process and improve the accuracy of the OCR. An OCR system for Malayalam language is also available [84]. A two level segmentation scheme, feature extraction method and classification scheme, using binary decision tree, is implemented. The development of the post-processor, a spell checker tuned for the OCR system improved the accuracy of the system.

Though there are various pieces of works reported by many research institutions, the document analysis technology on Indian scripts is not so mature. This is attributed to the existence of large number of characters in the scripts and their complexity in shape [39]. As a result of which a bilingual recognition systems has been reported in recent past [41][81]. An OCR system that can read two Indian language scripts: Bangla and Devanagari (Hindi) is proposed in [41]. In the proposed model, document digitization, skew detection, text line segmentation and zone separation, word and character segmentation, character grouping into basic, modifier and compound character category are done for both scripts by the same set of algorithms. The feature sets and classification tree as well as the lexicon used for error correction differ for Bangla and Devanagari. Jawahar *et al.* [81] presents character recognition experiments from printed Hindi and Telugu text. The bilingual recognizer is based on principal component analysis followed by support vector classification. Attempts that focused on designing a hierarchical classifier with hybrid architecture [101], as well as a hierarchical classifiers for large class problems [36] are also reported in the recent past.

2.2.6 OCRs for African Indigenous Scripts

While the use and application of OCR systems is well developed for most languages in the world that use both Latin and non-Latin scripts, an extensive literature survey reveals that few reports are available on the indigenous scripts of African languages. It is only recently that research reports have been seen for the recognition of Amharic language.

Amharic printed character recognition is discussed in [46], with more emphasis on designing suitable feature extraction scheme for script representation. Two template comparison techniques are presented for identifying Amharic characters. One compares the every pixel of the input character to a set of templates, the other derives a characteristic signature from the character and compares this against a set of signature templates.

Recognition using direction field tensor as a tool for Amharic character segmentation is also reported [196]. Primitive structural features and their spatial relationships are extracted for recognition. A tree structure is used to represent the spatial relationship of primitive structures. Worku and Fuchs [5] presented a handwritten Amharic bank check recognition using Hidden Markov Random Field (HMRF). They attempted for handwriting recognition of the legal amount field of Amharic bank check. For recognition purpose, natural features are extracted and a classification technique by estimating likelihood using pseudo-marginal probability is developed. Then contextual information based on the syntactical structure of Amharic checks is applied for improving recognition accuracy. We demonstrate in this thesis an Amharic OCR that works well on real-life printed document images. This is possibly the first attempt targeted on designing a complete OCR for Amharic language.

Apart from this, there are no similar works being found for other indigenous African scripts [124]. Therefore, there is a need to exert much effort to come up with better and workable OCR technologies for African scripts in order to satisfy the need for digitized information processing in local languages.

2.2.7 Holistic Word Recognition

Most recognition approaches proceed by segmenting words into characters or components which are then recognized separately to determine the correct word level. The recognition result of a word is then the composition of the individually recognized parts. Recently researchers have begun to focus on holistic word recognition approaches [115]. Holistic recognition approaches treat words as a single and inseparable unit and attempt to recognize words as a whole using their representational

features, without any character segmentation.

Madhvanath *et al.* [115] provide a survey of the holistic paradigm in human reading and its applications in handwritten word recognition. They show that holistic word recognition is a viable alternative to the popular analytical (character segmentation-based) approach to handwriting recognition, and point out parallels in human reading studies.

Lavrenco *et al.* [104] present a holistic word recognition approach for handwritten historical documents, which is motivated by the fact that for severely degraded documents, segmentation of words into characters very often produces poor results. The quality of the original documents does not allow to have high accuracy during recognition. The researchers focused on producing transcriptions that will allow successful retrieval of images from historical documents. A document is described using a Hidden Markov Model, where words to be recognized represent hidden states. The state transition probabilities are estimated from word bigram frequencies. Word images are represented by fixed length feature vectors, using features ranging from coarse (e.g. word length) to more detailed descriptions (e.g. word profile). They reported a recognition accuracy of 65%, which exceeds performance of other systems that operate on non-degraded input images (of non historical documents).

2.3 Retrieval from Document Images

With a significant progress in the digitization of libraries, document images are gaining popularity as an information source. Hence, access to the contents of the document image database is an important and a challenging problem in document image indexing and retrieval. Most published methods for retrieval of document images first attempt recognition and transcription followed by indexing and search operating on the resulting (in general, erroneous) encoded text using (e.g., standard bag-of-words) information retrieval (IR) methods [177].

A comprehensive review on the indexing and retrieval methods for document images is presented by Doermann [53]. This review focuses on the research issues when the documents are represented as text with the help of OCRs. Possibly this is more relevant for languages like English. The survey summarized the effect of recognition result on retrieval as follows: At OCR character error rates below 5%, the IR methods suffer little loss of either recall or precision; however, at error rates above 20%, both recall and precision degrade significantly [53].

A crucial open problem is therefore the effectiveness of first OCR, and then IR methods [111]. This is basically a long-term solution for the document understanding

requirement. In addition, for indigenous scripts of India, Ethiopia, etc., there are no robust OCRs that can successfully recognize printed text images. Hence, we need an alternate approach for effective access to relevant documents from these large collections of text images. Document image indexing and retrieval using only image properties, without explicit recognition is also studied by many scholars as an alternate approach to OCR solutions. Increasing focus is on indexing and retrieval mainly from poor quality document images collections. We review below various attempts made on the indexing and retrieval of document images without explicit recognition.

2.3.1 Word Spotting for Handwritten Documents Retrieval

Word spotting is an approach that represents word images using a set of features and match two words by comparing the corresponding feature vectors. This approach is quite effective when compared to recognition-based search and retrieval from poor quality documents.

The pioneer work in this direction is the use of word spotting for matching and retrieval of word images from handwritten manuscripts [118]. The approach segmented pages into words and matched these words, and used the matching scores to retrieve the words relevant to a given query. This is a useful approach for locating similar occurrences to the query word.

At the heart of the word spotting idea, is the word image matching algorithm that determines the similarity between images. This is crucial to its performance. Matching establishes the correspondence between primitives extracted from two or more objects by measuring the resemblance of one pattern with another pattern using some dissimilarity measure [187]. Its accuracy and efficiency determine the quality of the clustering and the size of the collection that can be processed using the word spotting process. Since the number of word image pairs grows as a quadratic polynomial in the size of the collection, the per-pair matching time must be small. Matching is a central problem in computer vision and pattern recognition, which has wider applications in multimedia retrieval, especially in content based image retrieval (CBIR) [187].

The work of Rath and Manmatha [155] discussed the problem of matching handwritten words in historical documents. They explored different word image matching techniques: translation invariant Euclidean Distance Mapping (EDM), affine invariant word transformation based on Scott and Longuet-Higgins algorithm (SLH), dynamic time warping (DTW), etc. EDM compares two images pixel-by-pixel after

Work	Data	Approach	Applications
Rath and Manmatha [155]	Offline, Historic	Word Image Matching	Single writer document collections
Srihari *et al.* [171]	Offline	Writer Matching	Forensic document retrieval
Russell *et al.* [160]	Online	Recognition results	Indexing and retrieval for multi-user and single script collections
Kamel [87]	Online	KLT-based, RTree	Online document retrieval
Jain and Anoop[80]	Online	Ink Matching	Search in single-writer document collections
Rath *et al.* [156]	Offline, Historic	Word Clustering	Search engine for retrieval from George Washington historical manuscripts
Chan *et al.* [37]	Offline	Word Image Matching	Searching printed and handwritten Arabic documents
Balasubramanian [23]	Online	Synthesis and Matching	Search, Index multi-user document collections
Rath and Manmatha [157]	Offline, Historic	Word spotting	Retrieval from single writer historical documents

Table 2.3: Overview of existing works in the area of handwritten document retrieval.

aligning them. It counts pixels difference to determine the matching cost. SLH is a feature-based matching that recovers an affine warping transform between sample points taken from the edge of the template and candidate image. Based on experimental results, they proposed dynamic time warping (DTW) for matching and retrieval of handwritten document images. Jain and Namboodiri [80] reported DTW-based word-spotting algorithm for indexing and retrieval of online documents. They extracted features such as the height of the sample point in the word, as well as the direction and the curvature of the stroke in the word and then compared each word in a document with the keyword to decide dissimilarity between words. A retrieval mechanism for online handwriting is proposed [23], which can handle different writing styles, specifically for Indian languages. The scheme uses handwriting synthesis to do matching in the ink domain as opposed to the use of a recognizer. The approach provides a keyboard-based search interface that enables to search handwritten data from any platform, in addition to pen-based and example-based queries. The frame-

work also allows cross-lingual document retrieval across Indian languages. The main challenge with handwritten image retrieval scheme is the writing variations (between various writers, digitizers and writing conditions), which is not the main issue in the case of printed document image retrieval. Table 2.3 summarizes works done for both online and offline retrieval of handwritten documents.

Work	Data	Approach	Applications
Chaudhury *et al.* [43]	Offline	Word Image Matching	Accessing Indian language document images
Trenkle and Vogt [181]	Offline	Word Image Matching	Information retrieval from document images
Tan [179]	Offline	Dot Product	Retrieval from document images
Balasubramanian *et al.* [24]	Offline	Word Matching	Search in large document image collections
Lu and Tan [114]	Offline	String Matching	Retrieval from document image databases
Marinai *et al.* [161]	Offline, Historic	Word Indexing	Search in modern printed documents
Ataer and Duygulu [15]	Offline, Historic	Word Image Matching	Retrieval of Ottoman documents
Konidaris *et al.* [96]	Offline, Historic	Word Image Matching	Word spotting in historical documents
Kumar *et al.* [98]	Offline	Hashing	Search in document image collections
Pramod and Jawahar [165]	Offline	Reverse Annotation of Word Images	Search over collections of document images in Indian languages

Table 2.4: Summary of works done in the area of printed document image retrieval.

2.3.2 Word Spotting for Printed Documents Retrieval

As shown in Table 2.4, a number of attempts have been made for retrieval of printed document images. For Indian languages, Chaudhury *et al.* [43] proposed a method to query documents at word level by exploiting the structural characteristics of the Indian scripts. They employed geometric feature graphs for representation, and suffix

trees for indexing the printed text. Annotation of documents by hyper-linking was provided for personalized access. An indexing and retrieval scheme for searching in document image databases is presented in [98]. Indexing is done using locality sensitive hashing (LSH) (a technique which computes multiple hashes) using word image features computed at word level. For efficiency and scalability, content-sensitive hashing is implemented through approximate nearest images of Indian poet of antiquity books written in one of Indian language called Telugu.

Trenkle and Vogt [181] presented a preliminary experiment on word-level image matching, where a query word is expanded to include its font variations. For each word image, features are extracted and for the baseline, concavities, line segments, junctions, dots and stroke directions are extracted and a matching technique is employed for measuring the similarity between keyword and candidate words during the retrieval phase. Retrieval from printed document images without the use of OCR is also reported in [179]. After document images are segmented into characters, features, such as the vertical and horizontal traverse densities (VTD) are extracted. Based on these features, an n-gram based document vector is constructed to measure similarity between documents using the dot product technique. Lu and Tan [114] proposed an inexact string matching to handle the problem of heavily touching characters and kerning during document image retrieval. After word images are represented by a primitive string, the proposed technique matches partial word images to estimate how a word image is relevant to the other and decide whether one word is a portion of the other.

The retrieval of documents from historical printed text collections is also attempted in [15][96][161]. Konidaris *et al.* [96] proposed a technique for word spotting in old Greek early Christian manuscripts. The aim is to search for keywords typed by the user in a large collection of digitized historical printed documents in which the retrieval result is optimized by user feedback. A method for the retrieval of Ottoman documents based on word matching is reported by Ataer and Duygulu [15]. A hierarchical matching technique is designed to find the similar instances of the word images. The matching method consecutively tests length similarity and similarity of quantized vertical projection profiles for the entire words, as well as for the ascender and descender part of the words. The work of Marinai *et al.* [161] presented word-level indexing of printed documents in the 17th to 19th centuries. The proposed approach indexes homogeneous document collections by automatically adapting to documents that vary in scripts and font styles. Unsupervised character clustering is done using self organizing maps (SOM). At run time query words are aligned with indexed words to deal with broken and touching characters.

Success of matching techniques depends on the features employed for the similarity computation. In this respect, the extraction of appropriate features for an application has been recognized as an important problem in pattern recognition [182]. Feature extraction helps to find an appropriate representational scheme for objects to be processed. A comprehensive review of the various feature extraction methods for an OCR system is found in Trier *et al.* [182]. Features are the basis of content based image retrieval (CBIR) [159] and object classification [182], and it is a broad research area in the field of computer vision and pattern recognition [65][92].

In an attempt to select the most suitable features for matching handwritten word images, detailed analysis was made on a range of features [154]. Some of the considered features are single valued features (such as profiles and gray-scale variance) for which one scalar value is calculated per column in the original image. Others are multivariate features (such as Gaussian smoothing and Gaussian derivatives) for which multiple values are calculated per image column. Each feature's individual and combined matching performance had been analyzed and, finally, the authors proposed the increased benefit of combining a set of profile features for matching handwritten documents.

Most of these works attempt for matching and retrieval from historical printed and handwritten documents, where the present OCRs fail to achieve reasonable performance because of degradations and unusual fonts. However, in order to advance to the level of text search engine we need to undertake further research so as to design an efficient scheme that answer users expectation from the retrieval system and scalability issue in the image domain.

2.3.3 Indexing Document Images

The basic idea behind indexing approach is two fold [53]. First, it provides the ability to characterize and index documents in a meaningful way. Second, it provides mechanisms to efficiently retrieve, in ranked order, the most relevant documents for a given query.

The traditional approach to indexing document images involves first converting them into textual format using an OCR system and then apply text based indexing scheme [183]. The most common way of characterizing the content of text document during indexing is to consider the full text and perform preprocessing for the extraction of representative terms [53] [68]. This is achieved through applying a series of document processing techniques (such as tokenization, terms normalization and term weighing) that identify word images, extract relationships between terms, preserve

their content, and weigh for measuring relevance [16][95]. The extracted index terms are clustered together based on their similarity measurement and are indexed using the selected indexing structure.

The word spotting approach is used for indexing historical handwritten manuscript in [119]. The approach first segments the page into words and then matches the actual word images against each other to create equivalence classes. Each equivalence class consists of multiple instances of the same word. For each equivalence class, the number of words in it are counted. Those classes with the highest number of elements are stop-words and, hence they are eliminated from further consideration. A list is then made of the remaining classes and ordered according to the number of words contained in each of the classes. The user then assigns for each member of classes its ASCII interpretation. The words in these classes are then indexed and ready for searching and retrieval.

Our present work proposes an indexing scheme for organizing printed document images of books and newspapers. Spotting a word from handwritten images is attempted by a direct matching of a keyword to all words in the document [118]. However proper indexing and retrieval needs to identify similar words and group them based on their similarity, and evaluate the relative importance of each of these words and word clusters. We need to apply IR measures (such as tokenization, normalization, discrimination and term weighing) that identify word images, extract relationships between them, preserve their content, and weigh for measuring relevance of documents. Tokenization is done for identification of all individual words that constitute the text documents [76]. For document images this is equivalent to word detection. Tokenization is followed by normalization. Documents must be normalized so that word variants are detected there by comparison of the user query with the documents is successful and query terms are composed on the concept level and not in the exact character level. Documents must also be discriminative to be able to find a specific document out of the collection.

Normalization: During normalization, suffixes are stripped off using the stemming process and stop words, both functional and/or domain-specific terms are filtered out for more compact representation of the document.

Stemming improves the performance (recall) of retrieval systems. It aims to implicitly group together words with similar meaning into one set. This is done by conflating morphological variants of a given word [97]. Since many terms are mapped to one, stemming reduces the total number of clusters, which further reduces the size and complexity of the index file built. Text search engines use language rules to

stem words. A common stemming algorithm in the text domain is the Porter stemmer [147]. It iteratively removes morphological and inflexional endings (i.e. suffixes) from words in English. However for document image indexing language information is not directly usable and, hence no known stemmers are available.

The words in a document are not equally indicative of its meaning. There are important words as well as a number of words that are irrelevant descriptors for indexing documents in document images. A representation for documents is obtained that extract relationships between terms, and their relative weights. Word weights reflect the importance of each word appearing in a document. There are many variations of word weighting techniques [97][186].

1. Term Discrimination Value. It is a measure of how well the use of a word will help to distinguish documents from each other. The discrimination value measures the difference in similarity when the word is considered from that when it is not considered. This is done by computing weight for each term in each document collection.

2. Signal-to-Noise Ratio. This method measures the value of a term for indexing based on information theory. The basis of information theory is the definition of the information content of a given signal (or sequences) of words. The information content of a term is measured as an inverse function of the probability of occurrence of words in a given document. The higher the probability of occurrence of a word, the less information it contains and vice versa.

3. TF/IDF weighting. TF/IDF technique weighs term frequencies by the inverse document frequencies [97]. This approach is simple to apply and as effective as other methods [143].

Most systems use TF/IDF weighting technique [16][186]. This is based on the realization that words, which are common to all the documents, are not really important for document representation. They are more characteristic of all document collection than of individual documents within the collection. Most common stopwords belong to several word groups, such as articles (*a, an, the*), conjunctions (*and, or, as, than*), propositions (*about, by, at, after*), adverbs (*not, too, so, very*), which are often referred to as function words [16]. There are also words not carrying a significant meaning when used in a specific domain under consideration, which are considered as stopwords or fluffs. Such stopwords are evenly distributed in all the documents. They account for 40 to 50 percent in English text [97] [164]. Text search engines built stop list or negative dictionary of these words. When a text is preprocessed, each word

occurrence is checked against this stop list. If the word is found in the dictionary, it is ignored from further processing during indexing and searching. Word weighting schemes, such as term frequency (TF) or inverse document frequency (IDF), may also be used to identify stopwords, especially subject dependent ones. Stopwords usually have high TF weight, but less IDF weight.

Indexing data structure: The above text processing methods generate content bearing index terms. These index terms are organized using an indexing schemes. A number of data structures are discussed in literature [164] [174]. The main ones are inverted index, signatures and latent semantic indexing. While latent semantic indexing organizes documents based on their semantic content, signature file associates an ordered list of terms (signatures) with each document. Inverted index, the most successful method for indexing large textual archives [164] [174], maps a word to the set of documents that contain it and constructs an index list of words and their location in a document.

Inverted index and signature file are the two major indexing data structure extensively used for text retrieval [121]. The performance of signature file depends linearly on database size, which potentially implies hard scalability problems for huge text collections. Signature files are not competitive data structure in web setting [121]. Recent improvements of inverted index significantly enhanced its performance and hence successfully used for indexing large textual archives [205].

Clustering: The indexing structure is used to organize clusters. Clustering algorithms are required to bring together similar word images into one set. Clustering attempts to find natural groups of components based on some similarity measure [54]. It has a wide applications in the area of data analysis, classification and retrieval [38][186]. Specifically, the application of clustering in indexing and retrieval include [73]: (a) clustering documents in a collection (e.g. digital libraries), (b) constructing a taxonomy of a collection of documents by grouping closely-related documents together, and (c) efficient query processing by focusing only on relevant subsets (clusters) rather than the entire document collection.

Document clustering provides efficient search strategies and effective search results. The efficiency is exhibited by limiting the search only among cluster keywords rather than the entire words in a document collection. Effectiveness of clustering emanates from the cluster hypothesis [186]: closely related documents tend to be relevant to the same query. The most popular clustering algorithms include k-means and hierarchical clustering [54] [72]. While hierarchical clustering groups the raw data without any

knowledge of the number of clusters, k-means algorithm partitions the data into fixed number of disjoint, non-hierarchical k clusters.

Hierarchical clustering is either agglomerative or divisive. Agglomerative clustering creates a series of nested clusters. The basic idea behind this clustering algorithm is that: it initially starts with N singleton clusters and successively merge pairs of clusters until the termination criteria is satisfied. Divisive hierarchical clustering, on the other hand, starts with all of the input points in one cluster and form the sequence by successively splitting clusters [54]. The feature representation and similarity measures between document images yields non-metric pairwise distances. In such cases, the popular choice of clustering is the agglomerative hierarchical clustering.

Both k-means and hierarchical clustering algorithms are designed to work in static environment, where adding new documents require complete re-clustering which is a costly operation. To solve this problem, algorithms that work under dynamic environment are proposed. Such algorithms include suffix tree clustering [198], star clustering [14], and incremental clustering [35] [38] [73]. With these algorithms, cluster maintenance is done easily and efficiently without affecting existing cluster structure. Among these techniques, incremental clustering algorithm is the most widely used technique in the text-domain [35] [73].

Can [35] introduced an algorithm for incremental clustering of textual document databases. The aim of this work is to create an effective and efficient retrieval environment, since periodic updating of clusters is required due to the dynamic nature of document databases. The experimental observations showed that the algorithm is cost effective with respect to re-clustering and can be used for many increments of document databases of various sizes.

Motivated by document and image classification problems in information retrieval, Moses *et al.* [38] proposed incremental clustering. They dealt with the problem of clustering dynamic point sets in a metric space. The goal is to avoid any major modifications in the clustering while processing updates, because any extensive recomputation of the index information will swamp the cost of clustering itself. The authors define the incremental clustering problem as follows: for an update sequence of n points in a metric space m, maintain existing collection of k clusters such that as each input point is presented, either it is assigned to one of the current k clusters, or it starts off a new cluster.

Recently, Hammouda and Kamel [73] also applied incremental clustering for organizing large collections of text documents. They selected this algorithm since managing, searching and browsing large repositories of web content requires efficient organization. The incremental algorithm relies on pair-wise document similarity in-

formation. Clusters are represented using a cluster similarity histogram. Clusters are required to maintain high cohesiveness while new documents are being added.

Annotation: Once documents are clustered they are annotated manually or automatically. Rath *et al.* [156] discussed automatic annotation of word images with a lexicon and probabilities using a relevance-based language model. After documents are segmented into words, they are annotated using a statistical model with the entire lexicon and probabilities. A language model retrieval approach is then used to search the documents. The technique was applied to build retrieval system for historical handwritten manuscripts. Towards enabling search over large collections of images in Indian languages, Pramod and Jawahar presented reverse annotation technique [165]. To be able to rank relevant images, the approach is extended to probabilistic reverse annotation.

2.4 Discussion

Searching and retrieval from document images can be done with the help of the following approaches: Recognition-based, Recognition-free or a combination of them. Table 2.5 gives a brief comparison between recognition-based and recognition-free approaches. Recognition-based approaches requires an OCR system to convert document images into text and then search in the recognized text. Recognition-free approaches, on the other hand, search for retrieval of relevant documents without explicit recognition (in the image domain). Effectiveness of recognition-based approach greatly depends on the availability of robust character recognition schemes. Accordingly, the present approach to OCR design needs further consideration to Come up with suitable recognizer for diverse document image collections. Hence, in this work we explore alternate OCR framework. In addition, we investigate the recognition-free approach for retrieval of document image collections in Indian and African languages.

2.5 Summary and Comments

The ever increasing archival of document images to a large extent brought many challenges to document understanding. Documents that varies in script, quality and printing are archived by many applications in a large scale. The advancement of OCR systems for Latin-based scripts helped a lot the digitization process. However the effect of document quality and unseen fonts reduce its usability for historical

Recognition-based Approaches	**Recognition Free Approaches**
Popular examples are text retrieval search engines.	Popular applications are CBIR and CBVR.
It needs explicit understanding of symbols for retrieval of documents.	It builds appropriate intermediate representations/features to solve the problem.
The problem is challenging, unless a robust recognizer is available.	The problem is relatively straight forward to build with certain level of performance.
The approaches for text retrieval are highly content/language dependent.	The approaches are weekly related to the content/language of document.
Retrieval from recognized text require less offline processing.	Document image retrieval requires high-amount of offline processing.
There are fast & efficient systems available for searching in text documents.	Most of the schemes are slow and inefficient for retrieval in image domain.
It is easy to extract semantics from the textual content.	There is difficulty to build semantics unless the textual representation is available.
Representation is compact and scalable for indexing.	Document image representation is bulky and difficult to index.
Selected works in character recognition area include [18][32] [53][140][195].	Selected works in document image retrieval include [43][114][154][161][171].

Table 2.5: Comparison of recognition-based and recognition-free approaches for retrieval from document image collections.

documents even for Latin-based scripts. For other non-Latin scripts research in OCR is not advancing to the expected level, especially for Indian and African indigenous scripts. This inhibit access to the content of document images in these languages. As a result digitized document images are not playing all the useful roles that encoded documents do, or even many of the roles that their original physical embodiments did. An immediate action needs to be taken to make existing archived document images useful. A promising alternate direction is document image retrieval without explicit recognition. Word level search and retrieval is attempted for printed and handwritten materials. However most of the works are again for Latin-based scripts.

Therefore document image retrieval schemes needs to be designed for Indian and African languages. The efficiency and scalability issues for document image retrieval

should be addressed to fulfill users expectation. Hence, in this work we show the various schemes that have been designed for facilitating searching and retrieval of relevant information from document image collections. Existing recognizers are not designed to work on document image collections. Their performance declines as diverse document image collections are submitted for recognition. We argue that the OCR design for document image collection should be different from that of a single page. The architecture for OCRs need to be redesigned in this direction. In this work we attempt to consider these issues. We start our presentation with script analysis in the next chapter and then our attempt to design an OCR system for Amharic language; an African language that has got less emphasis in the recent past.

Chapter 3

Language Issues Related to Recognition and Retrieval

3.1 Introduction

Writing is a means of communication and preserving historical information for future use. Writing systems were preceded by proto-writing, systems of ideographic and/or early mnemonic symbols which are used as means of communication. The best known examples are: Jiahu Script (6600 BC), Vinca script (4500 BC) and early Indus script (3500 BC) [189]. The first true alphabetic writing appeared around 2000 BC, as a representation of language developed in Egypt.

Writing systems (or scripts) can be categorized into the following [112][136]: (i) Logographic script (a character is used to represent one concept, e.g. Chinese characters), (ii) Syllabic or Abugida script (consonants are written as the main letters, and then special symbols are used to indicate which vowels follow the consonant, e.g. Amharic characters, Indian Devanagari characters), (iii) Alphabetic (the sounds in a language can be represented by an appropriate consonant and vowel alphabet, e.g. Latin alphabet), and (iv) Abjad (a consonantal alphabet in which vowels are not written, e.g. Arabic alphabet).

3.2 African Scripts

Africa is the second largest continent in the world, next to Asia, covering about one-fifth of the total surface area of the Earth. Africa is not one country with a uniform culture. Africa has a very rich diversity on culture, history and languages [57]. In Africa more than 2500 languages (including regional dialects) are spoken which is an estimated one third of the world's total. Its many languages testify to the vast diversity of the African people. However, document analysis research has not yet addressed the indigenous African scripts, as much they deserve.

3.2.1 African Languages

The languages of Africa are grouped into four language families: Afro-Asiatic, Nilo-Saharan, Khoisan, and Niger-Congo [56].

The Afro-Asiatic Family: Almost 400 Afro-Asiatic languages constitute the most important group of languages spoken in northern Africa. The Semitic branch of the family includes languages spoken in Asia as well as in Africa. The many Arabic languages, the leading members of this branch, are the major languages of North Africa (Tunisia, Morocco, etc.) and East Africa (Sudan, Ethiopia, etc.). An Ethiopian language, Amharic is grouped under Afro-Asiatic language. Other Semitic languages spoken in East Africa include Tigre and Tigrigna in Ethiopia and Eritrea, respectively. Languages of the Berber branch of the Afro-Asiatic family are spoken in Morocco, Algeria, and Tunisia. The Cushitic branch, confined to Ethiopia, Eritrea, Somalia, Kenya, Sudan, and Tanzania, includes such languages as Oromo and Somali.

The Nilo-Saharan Family: Around 200 Nilo-Saharan languages are found in a broken chain from the great bend of the Niger River in West Africa to Ethiopia, throughout most of the upper Nile valley, and in parts of Uganda and Kenya. The western most branch of this family is Songhai spoken in Mali and Niger. The Saharan branch of this family includes languages spoken in Nigeria, Chad and Libya. Along the River Nile near Egypt and in south-west are the Nubian languages. The Nubian alphabet was derived from that of the Coptic language. In Sudan, Uganda and Kenya a group of Nilotic languages such as Dinka, Nuer, Shilluk, and Acholi (or Luo) also belongs to this branch of family.

The Khoisan Family: The Khoisan languages comprise the smallest language family in Africa, with only around 200,000 speakers of the 30 languages altogether.

Most of these languages are spoken by the Khoikhoi and San peoples of southern Africa; the largest of them is Nama. In Tanzania there are two other representatives of this family: Sandawe and Hadza. The Khoisan languages are best known for the unusual click consonants characteristic of most of them; in some Khoisan languages nearly every word begins with a click. Some of the Khoisan languages have a system of grammatical gender, which is found elsewhere in Africa only in the Afro-Asiatic family.

The Niger-Congo Family: The largest African languages family comprising over 1,400 languages, this family includes several subfamilies, including Kordofanian, Mande, and Atlantic-Congo, which is further sub-categorized into subfamilies including Benue-Congo, Atlantic, Gur, Kwa, and Ijoid. In the Benue-Congo subfamily, a relationship exists among most of the languages of southern and central Africa. These languages have become widely known as Bantu. Some of Bantu languages are Zulu and Xhosa in South Africa; Makua in Mozambique; Nyanja in Malawi; Shona in Zimbabwe; Bemba in Zambia; Fang and Bulu in Cameroon, Yoruba, Igbo, and Efik in Nigeria. From south-eastern Nigeria to Liberia are found the languages of the Kwa branch. This branch includes such important languages as Ewe in Togo and Ghana; Akan in Ghana; and Anyin in cote d'Ivoire. From Liberia to the desert north of Dakar, are several languages of the Atlantic branch. These include Themne in Sierra Leone, Wolof in Dakar, and Fulani spoken in Guinea, eastern Nigeria (Nigerian Fulfulde), and Senegal (Pulaar). Speakers of languages of the Mande branch inhabit in the West Africa. One Mande language, known as Bambara, is spoken by up to three million people in Senegal, Mali, Guinea and Cote d'Ivoire. Other important Mande languages are Mende in Sierra Leone and Kpelle in Liberia. The Mande languages are believed to be the oldest offshoots of the parent Niger-Congo language spoken more than 5,000 years ago.

3.2.2 African Indigenous Scripts

Out of the total languages spoken in Africa, some are indigenous languages, while others are installed by conquerors of the past. English, French, Portuguese, Spanish and Arabic are official languages of many of the African countries. Most African languages with a writing system use a modification of the Latin and Arabic scripts. However, there are also many languages in Africa with their own indigenous scripts that vary considerably in shapes. Some of these scripts are the following [116]: Egyptian hieroglyphs, Amharic script (Ethiopia), Vai script (West Africa), Bassa script (Liberia),

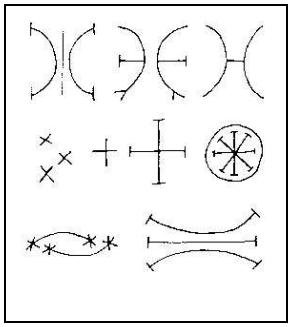


Figure 3.1: Alphabets of Nsibidi Script

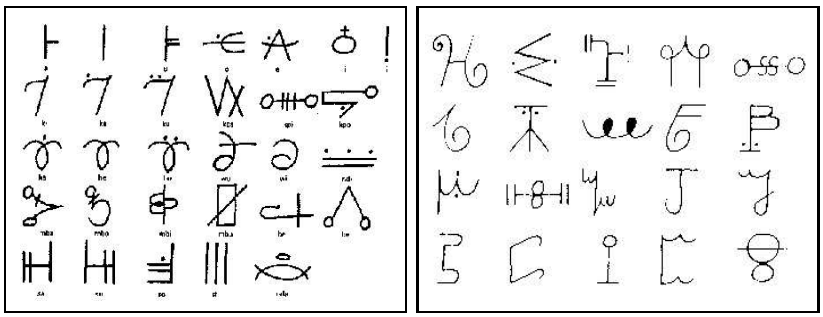
Mende script (Sierra Leone), Nsibidi/Nsibiri script (Nigeria and Cameroon), Shumom script (Cameroon), and Meroitic script (Sudan).

Egyptian hieroglyphs: Egyptian writing system is an ancient pictographic writing now dated to be 3400B.C. It consists of approximately 121 bi-literals, 75 tri-literals, and various determinants and phonetic complements. The bi-literals were individual symbols which expressed two sounds and the tri-literals were individual symbols which express three sounds. Phonetic complements are mono-literals found in front of and/or behind multi-consonantal signs in order to provide clarity and also to complete the meaning of the word. They normally repeat sounds already found in the word, but have no separate sound value.

Nsibidi script: Nsibidi is a writing system of the Ejagham people of Nigeria. It is seen on tombstones, secret society buildings, costumes, ritual fans, headdresses, textiles, and in gestures, body and ground painting. Alphabets of Nsibidi script are shown in Figure 3.1.

Amharic script: It is designed as a meaningful and graphic representation of knowledge. Amharic script is a component of the African knowledge systems and one of the signal contributions made by Africans to the world history and cultures. It is created to holistically symbolize and locate the cultural and historical parameters of the Ethiopian people. The system, in its classic state, has a total of 182 syllographs, which are arranged in seven columns, each column containing 26 syllographs. Through time many characters are added to give it the present shape. Detailed discussion of the writing system is made in Section 3.4.

Meroitic Script: The Meroitic script is very similar to the Egyptian Writing System. It was used by the Meroe people of the Sudan. The system is written from right to left, unlike the Egyptian system which is written both from right to



(a) (b)

Figure 3.2: Alphabets of (a) Mende Script and (b) Vai script.

left, left to right, and vertically. Meroitic is an alphabetic script with 23 signs used in a hieroglyphic form and in a cursive form. The majority of texts are cursive. There is a simple one-to-one correspondence between the two forms of Meroitic, except that in the cursive form, consonants are joined in ligatures to a following vowel.

Mende script: The Mende language is a major language of Sierra Leone, with some speakers in neighboring Liberia. It is spoken both by the Mende people and by other ethnic groups as a regional lingua franca in southern Sierra Leone. Mende script is not only considered a writing system, it is a work of art. Mende is a tonal language. Notable features of Mende writing system are: (i) it Consists of 195 symbols, (ii) it is written from right to left in horizontal lines. Figure 3.2 (a) depicts characters of Mende script.

Vai script: The Vai script is a writing system used by the Vai people of West Africa since the 20th century. It is one of many indigenous secret writing systems in Africa. Vai is a simple syllabic script written from left to right. The writing system is based on the mora, a unit of duration (or weight) such that a short syllable has one mora and a long syllable has two. A syllable is long if it contains a long vowel or ends with a consonant. Vai is a tonal language and has 12 vowels and 31 consonants. Out of the 12 vowels seven are oral vowels and the rest five are nasal vowels. Figure 3.2 (b) presents vai script.

Shumom Writing System: The Shumom people are the people of Cameroon in West Africa. Cameroonians use the Shumom writing systems, perhaps beginning with the hieroglyphics of the Ancient Egyptians writing. The writing system went through seven stages of development. The first stage had over five

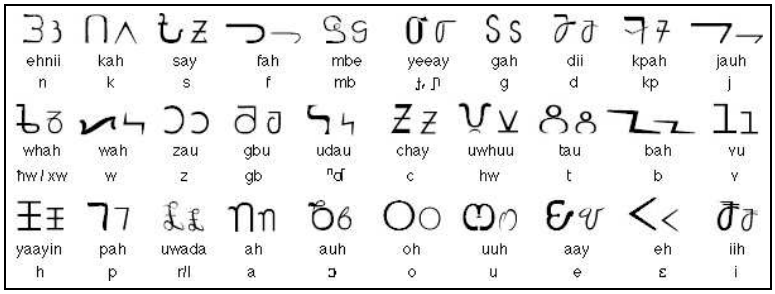


Figure 3.3: Alphabets of Bassa Script

hundred pictographs and the last stage has had only 35 syllographs, graphs designed to represent all the phonetic and tone sounds in the Bamum language of the Shumom people.

Bassa Script: Bassa is the most commonly spoken languages in Liberia which has its own written script. Bassa script is phonemic rather than syllabic. Bassa is a tonal language. Tones are marked using a system of dots and dashes which appear inside the vowel letters. Bassa script set is described in Figure 3.3.

There are also other languages in Africa known to have their own written language. These scripts include the Kpelle, Gola, Lorma, Grebo, and Kissi. Most of these scripts have diminished over time, as a result of abandonment.

Africans have traditionally spoken not only their mother tongue but also a local or regional lingua franca, such as Hausa, Swahili, or Arabic, associated with trade. Basically multilingualism is extensive throughout the continent, and Arabic is a major world language. In the same way, some African languages are important transnational languages which function as lingua francas. Apart from Arabic, which is not confined to Africa, the most widely spoken African tongues are Hausa, Swahili and Amharic, both of which are used over wide areas as lingua francas [189]. Hausa has the largest number of speakers, spoken by 39 million people. It is followed by Swahili which is spoken by 35 million speakers and Amharic which has around 34 million speakers.

Native speakers of Hausa are mostly found in Niger and Nigeria, but the language is widely used as a lingua franca in a much larger swathe of West Africa, including Benin, Burkina Faso, Cameroon, Ghana, Niger, Nigeria, and Togo. Swahili is the mother tongue of the Swahili people in the East African coast from Somalia to Mozambique. Swahili is an official language in Tanzania and Kenya. It is also spoken in Uganda, Rwanda, Burundi, Congo (DRC), Somalia, Comoros Islands (including Mayotte), Mozambique and Malawi.

Amharic language is mainly spoken in Ethiopia and Eritrea. Amharic is used as an official language in Ethiopia since the 14th century [28]. It is also the working language of Ethiopia and the most commonly learned language next to English throughout the country. Amharic script is used for writing in the various languages in Ethiopia and Eritrea, including Amharic, Tigre and Tigrigna.

3.3 Indian Languages

India is the second most populous country in the world, with a population of over one billion, and is the seventh largest country by geographical area. It is a home to some of the most ancient civilizations, and a center of important historic trade routes.

India is rich in languages, boasting not only many indigenous language groups, such as the Dravidian languages, but also the Indo-European, Indo-Aryan languages, and the absorption of Middle-Eastern and European influences as well. Distinct, often ancient, and rich literary traditions are to be found in several languages [189].

In India, there are 22 official languages spoken throughout the country, including Assamese, Bengali, Gujarati, Kannada, Kashmiri, Malayalam, Marathi, Oriya, Punjabi, Sindhi, Tamil, Telugu and Urdu. Among these, Hindi has been adopted along with English as the official language of the central government. The number of people speaking each language varies greatly. Hindi, the most widely spoken language, has more than 337 million speakers, followed by Bengali which is spoken by 70 million and Telugu by 66 million [58]. Relatively few people speak Andamanese.

Languages such as Sanskrit and Tamil are very ancient languages with a rich history of literary development. Sanskrit literature is more than 5,000 years old and Tamil is around 3,000. Telugu is another language with a notably ancient history and body of literature. An official language, and the main language of Hindu liturgy, Sanskrit is also very much a living language. It is being revived as a spoken tongue and also used in rituals and ceremonies or as part of daily prayers in Hinduism.

Generally speaking, in India, there are 24 languages which are spoken by a million or more people, in addition to thousands of smaller languages. Besides the Indo-Aryan and Dravidian languages, there are many Tibeto-Burman and Austro-Asiatic languages spoken in India, among others. The Andamanese languages, spoken on the Andaman Islands, are apparently not related to any other language family.

Figure 3.4: Devanagari script full character set as in [136]

3.3.1 Indigenous Indian Scripts

Indian languages have corresponding distinct alphabets. Today, more than eighteen languages have their own scripts, though some scripts like Devanagari and Bengali may be shared by two or more languages. Urdu and Sindhi are also included in the list. With the exception of Urdu, the alphabets of all these languages are native to India.

A remarkable feature of the alphabets of India is the manner in which they are organized. They are organized according to phonetic principle, unlike the Roman alphabet, which has a random sequence of letters. The languages of India are phonetic in nature and, hence the writing system for any language maps the sounds of the aksharas to specific shapes [176]. The basic set of aksharas for most languages consist of sixteen vowels and about forty consonants [176]. The actual rules for forming consonant-vowel combinations and conjunct characters vary from script to script. In most languages, a consonant-vowel combination is written by adding a vowel extension (matra) which is equivalent to the medial vowel representation. Conjuncts are generally written by concatenating half forms of each consonant as in Devanagari and derived scripts, or one below the other as in South Indian scripts. The aksharas are shown along with the equivalent Roman Diacritic representations.

India has different ways of writing its languages. Most of these written forms, or scripts, come from an ancient Indian script called Brahmi. The scripts run from left to right. There is no equivalent to capital letters. The Devanagari script used for Sanskrit is also used for Hindi, Marathi, and Nepali. The Devanagari script is shown in Figure 3.4.

The Roman script used for European languages has the individual letter as its basic unit. In Indian scripts, however, the basic unit is the whole syllable - a consonant plus a vowel. The numerals in Indian scripts are the origin of the Arabic numerals used in European writing systems.

3.4 Amharic Language

Ethiopia is located in Eastern Africa with a population of more than 75 million [189]. Ethiopia is a mosaic of ethnicities with eighty-four indigenous languages. Some of these are: Amharic, Afar, Hadiya, Harari, Somali, Sidama, Gurage, Oromo, Tigrigna, etc. There are more than 200 different dialects spoken. English is the most widely spoken foreign language and is the medium of instruction in secondary schools and higher institutions. Amharic was the language of primary school instruction, but has

been replaced these days in many regional states by local languages such as Oromifa and Tigrigna.

The Ethiopian languages are divided into four major language groups, such as Cushitic, Omotic, Nilo-Saharan and Semitic. Amharic belongs to the Semitic family of languages. Amharic is the second most spoken Semitic language in the world, after Arabic. It is the official and working language of Ethiopia and thus has official status nationwide. It is also the working language of several of the regional states within the federal system of Ethiopia. It has been the working language of government and non-governmental organizations, as well as most private institutions. These days, Amharic is spoken by roughly 30% of the population as a first language, and an additional 20% as a second language, totaling about half of the population. Outside Ethiopia, Amharic is the language of some 2.7 million people living notably in Eritrea, Egypt, Israel and Sweden. In general, there are more than 34 million speakers of Amharic [57][58].

Amharic language has its own script. Understanding the characteristics and distinct features of this script have immense use for the development of optical character recognition system. Hence, in the following section a detailed analysis of Amharic script is provided.

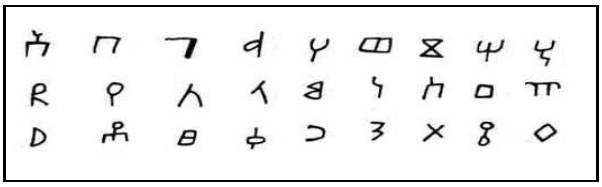


Figure 3.5: List of Sabean characters

3.4.1 Amharic Writing System

Amharic has its own writing system called FIDEL. Character sets are derived from Sabean writing (shown in Figure 3.5), which has had twenty-seven symbols in its unvocalized shape. But later Geez pursued the most original course taken Semitic script in denoting vowels by a variety of changes in the structure of the consonantal symbol. Vowels have thus become an integral part of Amharic writing which now assumed the character of a syllabary.

Amharic script has also evolved through the centuries and has undergone many transformation in structural shape and an increase number of symbols. This happens on the one hand, by modifying the original Sabean characters (as depicted in

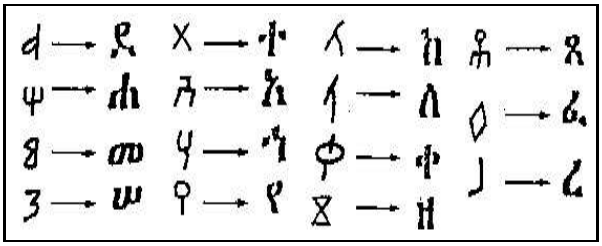


Figure 3.6: Amharic characters that are modified from their original Sabean characters

Figure 3.6) and, on the other hand, by adding other necessary ones in the writing. The need for such transformation through time is due to [184]: (i) the tendency towards round forms, (ii) the changed direction of writing, and (iii) the turn of some characters by ninety degrees.

Amharic script has now 33 core characters each of which occurs in seven different forms or orders (one basic form and six non-basic forms) (as shown in Figure 3.7). The seven orders represent syllable combinations consisting of a consonant and following vowel [28]. This is why the Amharic writing system is often called a syllabary rather than an alphabet. The non-basic forms are derived from the basic forms by more-or-less regular modifications. Other symbols representing labialization, numerals, and punctuation marks are also available. These bring the total number of characters in the script to 310. Table 3.1 shows summary of the number of characters in each group. Existence of such large number of Amharic characters in the writing system is, among others, a great challenge in the development of OCR for the language. Since Amharic writing system does not have a symbol for zero, negative, decimal point, and mathematical operators, the Hindu-Arabic numerals and Latin mathematical operators are used for computational purpose.

Type of Amharic Characters	Total Characters
1. Basic characters	231
2. Labialized characters	89
3. Punctuation marks	9
4. Numerals	20
Total	349

Table 3.1: Total Number of Symbols in Amharic Writing System (FIDEL)

Figure 3.7: The Full Amharic script (FIDEL) set.

3.4.2 Characteristic of the Script

Amharic characters have a number of notable characteristics. This is mainly attributed to character formation in the script. Since one character is formed from the other by adding strokes or marks, most of the alphabets have some common features. Some of these features of Amharic characters are discussed below.

Shape similarities among characters: The shape of many Amharic characters shows similarities with few distinctions among them. The distinction is shown, for instance by adding strokes at the top of the character, for example, 'ደ' and 'ጀ', 'ተ' and 'ቸ', 'ከ' and 'ኸ', etc.

Structural relation: Many basic characters are also clearly related in graphical structure, for instance, 'ነ' and 'ከ', 'ረ' and 'ፈ', etc. There are very similar characters that are difficult even for humans to separate visually, for example 'ለ' and 'ሰ'; 'ደ' and 'ጸ', etc.

Shape differences: There are also remarkable differences in shapes among the basic characters. Consider 'ሀ' and 'በ' (both are open in one side but in opposite direction), 'ሙ' and 'ወ' (both are formed from two loops but differ in the connection of the loops), etc.

Size and width differences: Amharic characters can differ in size both vertically and horizontally. There are very short characters (such as ፈ, ረ, ሙ) and there are very long characters (such as ቸ, ፕ, ጀ). There is also noticeable variance in width, for instance between ነ, ሙ, and ሠ.

Vowel formation: An interesting peculiarity of the Amharic writing system is the way vowels are formed. Vowels are written with small appendages to the consonant letters, with modifications of their shapes. This method of writing vowels is similar to that of Indic alphabets. Specifically speaking, vowels are derived from consonants in two ways. Some vowels (such as the fourth and seventh orders) take a modified shape of the base character by shortening/elongating one of its main strokes. On the other hand, adding strokes (and loops) to the right, left, top or bottom of each base character forms the remaining vowels (like second, third and fifth orders). As shown in Figure 3.8, the second, third, and fifth orders are formed (with few exceptions) according to patterns of great regularity, while the fourth, sixth and seventh orders are highly irregular. For instance, the second order is mostly constructed by adding

a horizontal stroke at the middle of the right side of the base character; where as, the sixth order is formed by adding a stroke, loop or other forms in either side of the base character.

Base character	2nd order	3rd order	4th order	5th order	6th order	7th order
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ፊሉ	ፊቡ	ፊቢ	ፊባ	ፊቤ	ፊብ	ፊቦ
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ

Figure 3.8: Overview of vowel formation from basic characters in Amharic script

Other features: As compared to Latin script, the concepts of upper-case and lower-case letters are absent in Amharic writing system. In addition, a line of Amharic characters lies at the same level, having no ascent and descent features as in the Latin characters. On the other hand, like English, the writing mode is from left to right and top-to-bottom. Words are separated with blank space/Ethiopic two-dots, sentences with Ethiopic four-dots, and paragraphs with recognized horizontal space.

3.5 Word Morphology

Different languages have their own language rules for formation of words from meaningful stems. Based on these rules words with the same meaning may appear in various forms within printed texts. Hindi, Amharic and English are inflected languages which utilizes affixes to form different words that express grammars in the language. Words in Hindi, Amharic and English can be inflected for tense (present, future, past), person (first, second, third), gender (masculine and feminine), number (singular, plural) and/or case (direct, oblique, and vocative). Amharic and Hindi are verb-final languages, in which modifiers usually precede the nouns they modify. Unlike the word order Subject-Verb-Object in English sentences, the typical word order in Amharic and Hindi is Subject-Object-Verb; usually modifiers precede the nouns they modify. On account of being a verb-final language, Hindi and Amharic uses postpositions (as opposed to, preposition in English) to express various case relationships. For example, English phrase 'on the table' is written as 'the table on'.

English verbs are marked in the 3rd person (he, she and it), like 'he sits'. In English, there are four moods (declarative, imperative, conditional, and subjunctive),

two aspects (continuous, and perfect) and three tenses (present, past, and future). Most English verbs express tense through the use of various combinations of the auxiliary verbs 'be' and 'have + main verb'. English has regular verbs that add '-ed' or '-en' to form the past tense, e.g., 'walk - walked', and irregular verbs that undergo internal vowel changes, e.g., 'drink - drank'. English nouns are marked for numbers such as singular and plural. Possession is expressed by '-s', e.g., mother's.

3.5.1 Hindi Word Morphology

Hindi is a highly inflected language which utilizes prefixes and suffixes to form words and to express grammatical relations.

Hindi nouns are marked for number (singular and plural), gender (masculine and feminine), case (direct, oblique, and vocative). The direct case is used to mark subjects of sentences; the oblique case is used with postpositions. There are four declensional paradigms for masculine and four declensional paradigms for feminine nouns. Adjectives agree with the nouns they modify in number, and case. 3rd-person pronouns are the same as proximate and remote demonstratives 'yh (this)' and 'vh (that)'. There is a 2nd-person honorific pronoun 'ap' which is used with both singular/plural and male/female addressees.

Hindi uses postpositions, rather than prepositions to express various case relationships. Postpositions require that the nouns be used in the oblique case.

Hindi verbs are marked for person (1st, 2nd, 2nd honorific, 3rd), number (singular and plural), aspect (imperfective and perfective), mood (indicative, imperative, optative). Hindi verbs occur in the following forms: root (kha 'eat'), imperfect stem (khatA), perfect stem (khayA), and infinitive (khanA). The stems agree with nouns in gender and number. Verbs are also marked for tense (present, past, future). Tense distinction of present vs. past is expressed by the auxiliary verb - 'honA (to be)'. Hindi verbs usually consist of a verb stem followed by an auxiliary verb, which is analogous to English is 'go-ing', except that the order is reversed.

All nouns are assigned gender (masculine or feminine). There are direct and indirect cases in Hindi. Indirect cases are used when the noun is followed by a postpositions. otherwise direct cases are used.

3.5.2 Amharic Word Morphology

Amharic has a very rich morphology [29] like Hindi. In Amharic most of the grammatical information is conveyed through affixes (prefixes, infixes and suffixes) attached

to the roots or stems. For instance,the word 'mesewiya' is modified by 'ye-mesewiya', 'mesewiya-w', 'mesewiya-win', etc. More examples are given in Table 3.2.

Amharic Word	Meaning	Word Variants
addis	new	ke-addis, addis-oc, addis-oc-u, addis-oc-u-n
bet	house	ye-bet, bet-oc, bet-oc-u, bet-oc-u-n, bet-it-u
gaddal	kill	gaddal-ku, gaddal-ku-t, gaddal-ku-la-t, gaddal-ku-ba-t, ta-gaddal
dingay	stone	dingay-un, dingay-oc, dingay-ama, ye-dingay
Itopia	Ethiopia	Itopia-n, Itopia-awi, Itopia-wi-t, ye-Itopia
faras	horse	faras-u, ye-faras, faras-anna, le-faras-u
tmhrt	education	tmhrt-in, ye-tmhrt, le-tmhrt, tmhrt-achin
isport	sport	isport-awi, isport-egna, ye-isport
zefen	song	zefen-sh, zefen-ih, ye-zefen, zefen-in
habt	wealth	habt-am, ye-habt, habt-ena, habt-am-oc

Table 3.2: Sample Amharic words with their variants

An Amharic verb root usually consists of a set of three to five consonants [197]. The verb may be inflected for person, number, gender, mood, causitive, transitive, etc. For example, 'gaddal (kill)' - 'gaddal-ku (I killed)' - 'gaddal-ku-t (I killed him)' - 'gaddal-ku-la-t (I killed something for him)' - 'gaddal-ku-ba-t (I killed something on him)', etc. Depending on the tense and other grammatical features, the consonants may be separated by particular vowels. Sometimes, consonants are geminated (doubled). A verb form normally also has one or more suffixes and possibly one or more prefixes as well, agreeing with the subject and sometimes the direct or indirect object of the verb. There are at least ten different classes of verbs, each modifying its stem in a number of different ways. Verbs are marked for person, number and gender.

Amharic nouns take suffixes indicating possession ('my', 'his', 'her', etc.), plural, and other grammatical functions. Amharic nouns are marked for two genders: masculine and feminine. The feminine gender can also be used to express smallness, e.g. bet-it-u, bet-, 'house' + feminine marker -it- + definite article -u 'small'. It can also be used to express intimacy. There are special words that indicate gender for people

and animals, e.g., wnd hakim, 'male doctor' and set hakim, 'female doctor'; awra doro, 'rooster' and set doro, 'hen.' Amharic nouns have singular and plural numbers. Plural is marked by the suffix '-oc'.

Both Amharic nouns and adjectives are highly inflected for number and gender. For instance, in comparison to the English plural marker 's or -es', there are more than three major and very common plural markers in Amharic nouns (e.g. 'woch', 'occ', etc.). Amharic verbs are also highly inflected for gender, person, number and tenses. Moreover, possessions, cases and article markers are often indicated through affixes in Amharic.

Definiteness is expressed by an article that has a masculine (male) and feminine (female) form in the singular. It follows the noun, as in 'betitu' above. As a result of which words with the same meaning may appear in various forms in written texts.

Since Amharic is morphologically very productive, derivations and word formations in the language involve a number of different linguistic features including affixation, reduplication and compounding (for details see [29][197]). Obviously, these high inflectional forms and extensive derivational features of the language are presenting various challenges for document image matching and retrieval.

3.6 Printing Variations

In addition to English, Indian languages (Hindi, Telugu, Tamil, etc.), and African languages such as Amharic (Ethiopia), Bassa (Liberia) have their own indigenous scripts. Each of these scripts contains a set of characters that greatly vary in number and shapes within language and among languages. For printing in a specific script, there are different fonts, sizes and styles available for use. Some of the commonly used fonts in Hindi, Amharic and English printed documents include the following: Indian languages, like Hindi uses fonts like 'Yogesh', 'Ganesh', 'Natraj' and 'Surakh'; Telugu uses fonts like 'Hemalatha', 'Eenadu', 'Vartha', etc. In the same way, from Africa Amharic uses fonts like 'PowerGeez', 'VisualGeez', 'Alphas' and 'Feedel'. English also uses a number of fonts like 'Ariel', 'Times New Roman', 'Courier' and 'Sans Serif'. Each of these fonts offer several stylistic variants such as normal, **bold**, and *italic*. and font sizes, including 10, 12 and 14. These fonts, styles and sizes produce texts that greatly vary in their appearances (i.e. in size, shape, quality, etc.) in printed documents.

Ethiopic Software	Fonts included	Type
ACIS	Hahu Lite, Hahu Lite Gothic, Hahu Lite Serif, Hahu Lite Times	ttf
Agafari	AGF-Zemen, AGF-Dawit, AGF-Rejim, AGF-Ejji Tsihuf, AGF-Yigezu Bisrat, Agaw, AgawBd	ttf
EthioWalia	AMH3	ttf
Acuwork	ALXethiopian	ttf
Addis Word	AddisWord1, AddisWord2	ttf
Brana	BranaI, BranaII	ttf
Feedel	Geezigna, Geez, Geez II, GeezA, GeezB, GeezNewA, GeezNewB, GeezNet, ZewdituA, GeezSindeA, GeezSindeB, ZewdituB, Geez Unicode	ttf
OmniTech	Amharic Kechin Normal, Amharic Yigezu Bisrat Normal, Amharic Gazetta Ordinay	ttf
Power Ge'ez	Ge'ez-1, Ge'ez-2, Ge'ez-3 , Ge'ez-1 Number	ttf
Visual Ge'ez	VG2-Agazian, VG2-Main, VG2-Title, VGUnicode, VGUnicodeA, VGUnicodeT	ttf
Wazema	A1 Desta, A1 Kidan, A1 Tesfa, A2 Desta, A2 Kidan A1 Qelem, A2 Qelem, A2 Tesfa	ttf
Phonetic	GeezType, GS GeezMahtemUnicode, GS GeezScript (U)	ttf
Ge'ezFont	GeezAddis, geezBasic, geezLong, GeezThin, geezDirib	ttf
Ge'ez Frontiers	GF Zemen Primary & Secondary, ENH Zena he GF Yigezu Bisrat Primary & Secondary, GF Zemenu (U)	ttf
Alpas	ET-Saba, ET-Saba New, ET-Saba Bold	ttf
Amharisch	Amharisch	ttf
C.B. Hale	Amharic-A, Amharic-B	ttf
Ethiopic	Ethiopic, AmharicLSU (U), EthiopicLSU (U)	ttf
Samawerfa	Addis98	ttf
Compose	Geezigna	ttf
SenaMirmir	jiret (U)	ttf
Washra	Ethiopia Primary normal & Slanted, EthiopiaAnsiP, Washra Primary normal & Slanted, Wookianos Secondary EthiopiaAnsiS, YebSe Primary & Secondary, Washrax Secondary normal & Slanted, Wookianos Primary Ethiopia Secondary normal & Slanted	ttf
NCI	ET-Nebar, ET-NCI, ET-Sami	ttf

Table 3.3: List of Amharic Unicode-based and ASCII-based fonts. Font names followed by "A & B", "I & II", or "Primary & Secondary" indicates division of the encoding systems into two sets. Frequently occurring characters are in the set 'A', 'I', or 'Primary'. Those fonts with 'U' are Unicode fonts

3.6.1 Amharic Computer Fonts

Amharic fonts have been encoded on personal computer systems in one of three approaches. The governing design constraint in devising an encoding systems has been the address space limitation for encoding glyphs that many font systems impose [194]. The first approach is *diacritic based systems* whereby the character set is broken down into base glyphs and diacritic symbols. The use of diacritic symbols is a means to encode more characters into available space. Less frequently used characters may be omitted. The second approach is *whole glyph systems* whereby glyphs remain intact, again less frequently used characters may be omitted to meet font space limitations. The other approach is *distributed encoding systems* whereby the first two approaches are taken but characters are spread over multiple physical fonts. With this approach the less frequent characters need not be omitted from the character set.

Since the 1980's many efforts have been exerted to develop fonts for Amharic scripts. Now, there are many fonts available for word processing as listed out in Table 3.3. We can observe that there are more than 80 ASCII-based Amharic computer fonts that are widely used for encoding Amharic text. The effort to develop the various Amharic computer fonts was mainly with the aim of reducing the number of keystrokes required to type a single Amharic character since the number of characters in the Amharic writing system is greater than 256 (which is the maximum limit of the number of characters in an ASCII based font file). It is now possible to type most Amharic characters using two key strokes per character reducing up to four key strokes required previously. All these different Amharic fonts developed by different suppliers do not follow a common standard for representation of the fonts. The glyphs and the combination rules differ from one font set to another.

Given the peculiarities of Amharic fonts and the characteristics of the Amharic language, building an OCR system is a challenging task. However, recently, with the objective of solving such and related problems, Unicode-based Amharic fonts are emerging, exploiting the Unicode platform, which provides a unique number for every character in the script, no matter what the platform, the program, or the language is used [185] [206]. Unicode has started to replace ASCII, ISO 8859 and EUC at all levels. It enables users to handle not only practically any script and language used on this planet, it also supports a comprehensive set of mathematical and technical symbols to simplify scientific information exchange [185].

Unicode contains the characters required to represent practically most known languages. This includes not only the Latin, Greek, Cyrillic, Hebrew, Arabic, Armenian, and Georgian scripts, but also Chinese, Japanese and Korean Han ideographs as well as scripts such as Indian (Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil,

Telugu, Kannada, Malayalam), Thai, Lao, Tibetan, Amharic, Mongolian, and many more [185]. Several scripts are expected to be included in the next revision of Unicode, including African scripts Vai, Meroitic, basic Egyptian Hieroglyphics, etc.

To use Amharic script practically, however, additional character encoding standards for mapping computer fonts need be created to ease recognition (viewing) of Amharic texts written in one font by another (especially for ASCII-based texts).

3.7 Character Unigram Statistics

We apply statistical language modeling and compute frequency of occurrences of Amharic characters in a text. This is important to distinguish characters which are frequently used for writing and to make sure that these characters are properly recognized during document understanding. In a language like Amharic where there are large number of characters, this enables to enhance the performance of the recognizer with the use of unigram based post-processor to resolve any ambiguity during the recognition process.

Common statistical language modeling (SLM) techniques include n-gram model, maximum entropy language model, etc. [190]. N-gram model is the most widely used SLM today. N-gram models use the previous n-1 words/characters to approximate the probability of occurrence of a word/character. The probability of a string s, $p(s)$ can be expressed as:

$$p(s) = \prod_{i}^{l} (p(c_i|c_1, ..., c_{i-1})) \tag{3.1}$$

where c_i is the i^{th} character in the sentence and $c_1, ..., c_{i-1}$ is the history, previous $i-1$ characters. Statistical language models determines the conditional probability of the character, given its history $p(c_i|c_1, ..., c_{i-1})$. But, for large i estimating the probability $p(c_i|c_{i-1})$ becomes difficult. So researcher usually use the cases of n-gram models like $n = 1, 2, 3$, which are referred to as a unigram, bigram, and a trigram models, respectively. While bi-gram predicts the probability of occurrence of a character based on the previous one word, tri-gram involves two characters. We use unigram language model [85] to compute frequency of occurrence of characters in a given text. The unigram posits that each character occurrence in a document is independent of all other character occurrences. It gives the probability of occurrence of a character in a given text. In unigram models, we need to make the approximation on the probability of occurrence of a character; hence we can approximate p(s) as:

Character	Occuerence	
	Frequency	%
ን	132064	0.04804
ት	110928	0.04035
በ	93227	0.03391
የ	89934	0.03271
ው	85659	0.03116
መ	76975	0.02800
ሰ	73097	0.02659
አ	72316	0.02631
ር	71654	0.02607
ል	66412	0.02416
ለ	66342	0.02413
ተ	60783	0.02211
ም	58224	0.02118
ያ	53564	0.01949
አ	46965	0.01708
ና	41547	0.01511
ይ	40818	0.01485
ደ	40590	0.01477
ን	39453	0.01435
ገ	39181	0.01425
.	.	.
.	.	.
.	.	.
ሓ	1	3.64E-07
ጭ	1	3.64E-07
ኩ	1	3.64E-07

Figure 3.9: The most frequently occurring Amharic characters.

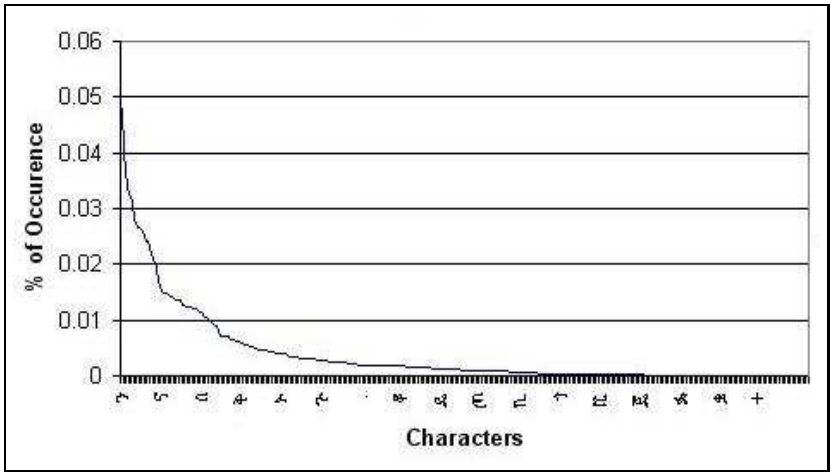


Figure 3.10: Occurrence of Amharic characters in printed text.

$$p(s) = \prod_{i}^{l} p(c_i) \tag{3.2}$$

Since models like unigram needs huge corpus to obtain better and meaningful statistics, we used the Amharic corpus available online [166]. The corpus has more than 2.5 million characters. We run the unigram model on this corpus. The result is shown in Figure 3.9.

As can be presented in Table 3.9, the most frequently occurring character is '**ን**' with probability around 0.05, followed by characters '**ት**' and '**ል**' with probabilities 0.04 and 0.03. The result shows that very few characters are occurring frequently. Thus, out of the 310 characters in Amharic script, twenty characters occur 50% of the time. Further analysis reveals that the first 50 characters in the list occurs 75% of the time. This shows that in Amharic written text just around 25% of the characters have got more importance than others. It also be noted that not all characters in the script occur in the writing. In the present experiment, more than 25% of the total characters do not appear in the text.

Figure 3.10 plots the frequency of occurrence of Amharic characters. As can be seen, the curve sharply drops drastically downward until it reaches the probability of occurrence of 0.005. This shows that frequently occurring characters are very few and most of the characters occur at a probability of less that 0.005. This information is useful to design an efficient recognizer.

3.8 Need for Recognition & Retrieval Scheme

In Africa and India there are a number of indigenous scripts used for writing. Accordingly, there is a bulk of printed documents (such as correspondence letters, books, newspapers, and magazines) available in government and private offices, libraries and museums. Digitization of these documents enables to harness already available information technologies to local information needs and developments, thereby, among others, to facilitate indexing and retrieval, to save storage space, and to preserve historical documents.

All these stress the importance and tremendous need for document understanding systems such as OCR and document image retrieval engines, if at all one is to harness already available information technology to local information needs and developments. The encouraging efforts being made and the advancements registered in the area of application software capable of interfacing with the scripts is an additional motivation for seeking OCR solution.

Those African languages that use modified scripts of Latin and Arabic language can be integrated to the existing Latin and Arabic OCRs with the same additional language processing modules. Commercial packages such as ABBYY FineReader recognizes texts written in some of the African languages such as Afrikaans, Xhosa and Zulu (South Africa), Kongo (Congo), Swahili (East Africa), Swazi (Swaziland), etc. that uses Latin script. Therefore, we need to give more emphasis to indigenous scripts of Africa and India.

3.9 Summary and Comments

In Africa around 2,500 languages are spoken. Some of these languages have their own indigenous scripts, while others are installed by conquerors of the past. English, French, Portuguese, Spanish and Arabic are official languages of many of the African countries. Most African languages with a writing system use a modification of the Latin and Arabic scripts. However, there are also many languages with their own indigenous scripts.

We attempt to introduce some of these languages with their own scripts in this work. The only Language that is used as official language with its own scripts and writing system (called FIDEL) is Amharic. Amharic script is used for various languages in Ethiopia and Eritrea including Amharic, Tigre and Tigrigna.

Amharic script has 33 core characters each of which occurs in seven orders (one basic form and six non-basic forms), which represent syllable combinations consisting of a

consonant and following vowel. Other symbols representing labialization, numerals, and punctuation marks are also available. These bring the total number of Amharic characters to 310. Since Amharic writing system does not have a symbol for zero, negative, decimal point, and mathematical operators, the Arabic numerals and Latin mathematical operators are used for arithmetic computation.

Amharic script has some notable features. The shape of many Amharic characters shows similarities, and many basic characters are also clearly related in structure. An interesting peculiarity of the Amharic writing system is the way vowels are formed. Most vowels are derived from consonants either by appending strokes or loops to the right, left, top or bottom of base character or by modifying the shape of the base character. As a result of which there are many similar characters in the script.

Amharic is an example of a language with a very rich morphology, which means that systems for searching Amharic text databases can be effective in operation only if full account is taken of the many word variants that may occur.

This chapter introduces the various indigenous African scripts, with detailed discussion of the challenges towards the recognition of Amharic language. We highlight features of Amharic characters and problems related to the scripts that have bearings on Amharic OCR development, especially availability of large number of characters in writing, existence of a large set of visually similar symbols and use of a number of distinct fonts in printed documents that are designed in an unstructured manner. In the next chapter we present an OCR adopted for the recognition of Amharic printed documents.

Chapter 4

Recognition of Amharic Script

4.1 Introduction

Nowadays, it is becoming increasingly important to have information available in digital format for increased efficiency in data storage and retrieval. Optical character recognition (OCR) is being recognized as one of valuable input devices in this respect [131]. This is also supplemented by the emergency of digital libraries for digitization of paper documents and the advancement of information technology that supports faster, more powerful processors and high speed output devices (printers and others) to generate more information at a faster rate. The availability of relatively inexpensive document scanners and optical character recognition software has made OCR an attractive data entry methodology [162].

The use and application of OCR systems are well developed for most languages in the world that use both Latin and non-Latin scripts [131][175]. However, there is limited research effort in this direction for the indigenous scripts of African languages. Therefore, there is a need to exert much effort to come up with comprehensive OCR technologies for African and Indian scripts in order to satisfy the need for digital information processing in local languages.

In this work, we focus on the design of an OCR system for one of the widely used indigenous script of African language called Amharic. After a thorough script analysis(in Chapter 3), we design an Amharic OCR system (following the conventional top down approach). The performance of the OCR is tested on real-life printed document images in which we have got encouraging result. Figure 4.1 depicts sample real-life Amharic page used for testing the OCR.

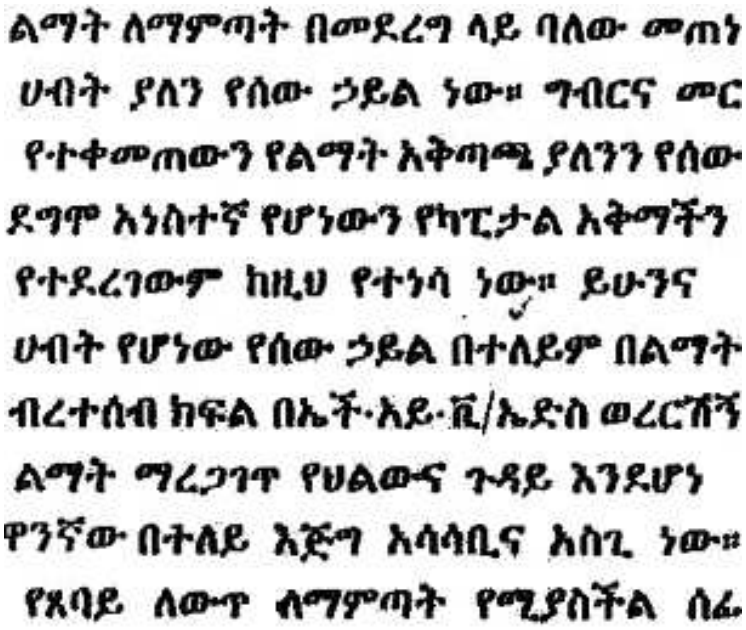


Figure 4.1: Sample Amharic page taken from Newspaper.

4.2 Challenges in Building Amharic OCR

Character recognition from document images that are printed in Amharic script is a challenging task. To develop a successful recognizer for this script, we need to address some of the following issues.

- *Lack of previous experience:* Document analysis and understanding has got few attention for African languages. It is just recently that few works are reported in the area of natural language processing and optical character recognition of Amharic documents. Most African languages with their own indigenous scripts is almost untouched. As a result, to undertake research in document analysis and understanding for African languages one has to start from scratch. One has to study and identify the available scripts, the characteristics of the scripts, preparing datasets for training, digitizing documents for evaluating the designed scheme and so on.

- *Printing variations:* Printed documents vary in fonts, sizes and styles. Building character recognition system is challenging in this situation. Details of the Amharic computer fonts used for printing purpose are discussed in Section 3.6. These fonts produce characters that greatly vary both in shape, width, line thickness, etc. Samples of characters printed using different fonts are shown in

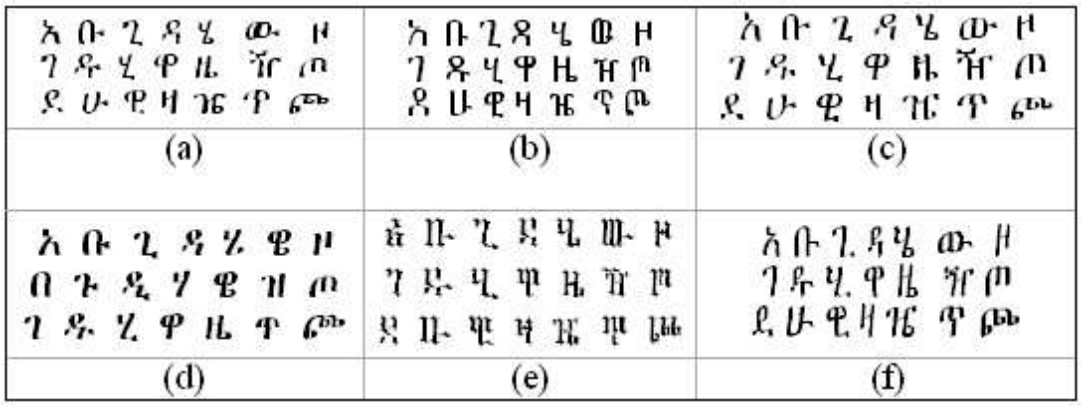


Figure 4.2: Sample characters printed using (a) Agafari-Zemen font, (b) Agafari-Rejim font, (c) Power Geez font, (d) Visual Geez font, (e) Agafari- Yigezu Bisrta font, and (f) Agafari-Ejji Tsihuf font.

Figure 4.2. All characters are written with uniform font size of 14 and normal style. But we can observe many variations in characters setup from font to font. Some fonts produce a character large in shape or a character small in size. Some fonts produce characters with thick line, while others are thin. There are characters vertically higher or lower, and horizontally spaced from each other. These fonts can be used with different styles and sizes during the production of documents as a result of which characters greatly vary in appearances and size within the text. We need to standardize the variation in size by applying normalization techniques and extract suitable features so that the representation is invariant to printing variations.

- *Large number of characters in the script:* The total number of characters in Amharic script is more than three hundred. Existence of such a large number of Amharic characters in the writing system is a great challenge in the development of Amharic character recognizer. Memory and computational requirements are very intensive. We need to design a mechanism to compress the dimension of character representation and select suitable classifier so as to come up with computationally efficient recognizer.

- *Visual similarity of characters in shape:* There are a number of similar characters in Amharic script that are even difficult for humans to identify them easily (examples are presented in Figure 4.3. This is mainly because (i) one character is derived from another character, and (ii) vowels are formed by at-

taching strokes to the base characters. Robust discriminant features needs to be extracted for classification of each of the character into their proper category or class.

Figure 4.3: Sample similar characters in Amharic

- *Degradations of documents:* Document images from printed documents, such as books, magazines, newspapers, etc. are extremely poor in quality (see Figure 4.1). Popular artifacts in printed document images include: (i) Excessive dusty noise, (ii) Large ink-blobs joining disjoint characters or components, (iii) Vertical cuts due to folding of the paper, (iv) Cuts at arbitrary direction due to paper quality or foreign material, (v) Degradation of printed text due to the poor quality of paper and ink, (vi) Floating ink from facing pages etc. This is one of the main challenge where most advanced character recognition systems developed for Latin and non-Latin scripts also fails. We need to carefully design an appropriate representational scheme and classification method so as to accommodate the effect of degradation.

- *Language related issues:* African indigenous languages pose many additional challenges. Some of these are: (i) lack of standard representation for the fonts and encoding, (ii) lack of support from operating systems, browsers and keyboard, and (iii) lack of language processing routines.

These issues add to the complexity of the design and implementation of an optical character recognition system.

4.3 Recognition of Amharic Characters

We design a prototype Amharic OCR on top of an earlier work for Indian languages [81] and reported the result of our works in [123][124]. Figure 4.4 shows the general framework of the Amharic OCR design.

The OCR accepts scanned document image or scans document pages from a flat-bed scanner. Document image is preprocessed before individual components are extracted for recognition. Text lines, words and characters are located in the segmentation

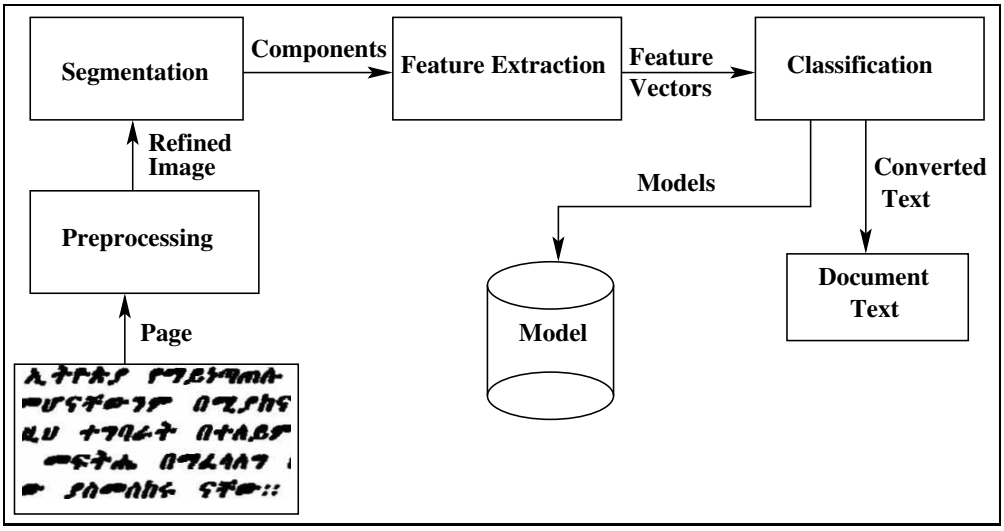


Figure 4.4: Overview of the Amharic OCR design.

phase. Then we extract features that are used for classification of characters. The final output is converted text with corresponding ASCII representation.

Preprocessing is necessary for efficient recovery of the text information from scanned image. The pre-processing algorithms employed on the scanned image depend on paper quality, resolution of the scanned image, the amount of skew in the image, the format and layout of the images and text and soon. In this work we apply binarization, noise removal, normalization and skew correction preprocessing techniques.

Scanned documents often contain noise that mainly arises due to printer, scanner, paper quality and age of the document. We use binarization to convert gray scale image (with 256 possible different shades of gray from black to white pixels) to binary colors (with black and white pixels) to ease image processing. In due course binarization removes some of the noises in images. However, because of the level of degradations in digitized document images of magazines, books and newspaper, there is a need to apply filters so as to reduce the effect of degradation during the recognition process. We use Gaussian filtering [130] that removes noise by convolving the original image with a mask of 3×3.

During the scanning operation using digitizing devices, like scanners, a certain amount of image skew (small tilt) is unavoidable. Document images are often misaligned to the standard axes, due to improper placement of the paper on the scanner bed. We make sure that the page is aligned properly with all its text lines horizontal. In our work, projection profile have been used for skew detection and correction in the range of $\pm 20°$.

Once pages are preprocessed, they are segmented into individual components. Segmentation is an operation that seeks to decompose an image into sub-images of individual symbols. The page segmentation algorithm follows a top-down approach by identifying text blocks in the document pages. A logical subdivision of a text document would be initially into lines, followed by words in each line and subsequently into characters. We use projection profile for this purpose. Lines are detected from text blocks using horizontal projection. A text line can be found between two consecutive boundary lines where the height of projection profile is least. Identified text lines are then segmented into its constituent words.

We employ vertical projection for word segmentation. This approach scans each line, the valleys of which show the boundaries of each word in the line. Next, characters are identified using vertical projection. To identify the boundaries of characters, each word is scanned in the vertical direction. During scanning if there is no black pixel encountered till the base line is reached, then this scan marks a demarcation line for characters.

Once character boundaries have been detected, it is often useful to extract character components. We use connected component analysis to identify the connected segments of a character. The final result is a set of character components with their bounding box that are scaled to a standard size of 20×20. The output of segmentation module, i.e., a set of connected components, are used as input for feature extraction, training and testing the recognition engine.

4.4 Feature extraction

Feature extraction is the problem of identifying relevant information from raw data that characterize the component images distinctly. There are many popular methods to extract features. Selection of a feature extraction method is probably the single most important factor in achieving high recognition performance in character recognition systems. A survey of feature extraction schemes for character recognition is available in [182]. Different feature extraction methods are designed for different representations of the characters. Some of these features consider profiles, structural descriptors and transform domain representations. Alternates one could consider the entire image as the feature. The former methods are highly language specific and become very complex to represent all the characters in the script. The later scheme provides excellent results for printed character recognition.

As a result, we extract features from the entire image by concatenating all the rows to form a single contiguous vector. This feature vector consists of zeros ($0s$) and ones

(1s) representing background and foreground pixels in the image, respectively. With such a representation, memory and computational requirements are very intensive for languages like Amharic that have large number of characters in the writing.

Therefore we need to transform the features to obtain a lower dimensional representation. There are various methods employed in pattern recognition literatures for reducing the dimension of feature vectors. In the present work we use Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

4.4.1 Principal Component Analysis

Principal component analysis (PCA) can help identify new features (in a lower dimensional subspace) that are most useful for representation [54]. This should be done without losing valuable information. Principal components can give superior performance for font-independent OCRs, easy adaptation across languages, and scope for extension to handwritten documents. Various extensions of PCA have also been proposed in computer vision, including Binary PCA [180]. While binary PCA models binary data as Bernoulli distribution, classical PCA models the image with Gaussian assumption. Both methods reduce the dimension of a dataset to reveal its essential characteristics and the subspace captures the main structure of the data. In the present work, we employ the conventional PCA for extracting relevant information from high dimensional datasets on which further transformation to classification sub-space is performed using LDA.

Consider the i^{th} sample represented as an M-dimensional (column) vector $\mathbf{x}_i$, where M depends on the image size. For a given training datasets, $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}$, we compute the covariance matrix (Σ):

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}[\mathbf{x_i} - \mu][\mathbf{x_i} - \mu]^T \tag{4.1}$$

Then we need to identify minimal dimension $M^{'}$ such that

$$\sum_{i=1}^{K}\lambda_i / Trace(\Sigma) \geq \alpha \tag{4.2}$$

where λ_i is the i^{th} largest eigen value of the covariance matrix Σ and α is a limiting value. Eigenvectors corresponding to the largest $M^{'}$ eigenvalues are the direction of greatest variance. The $M^{'}th$ eigenvector is the direction of greatest variation perpendicular to the first through $(M^{'}-1)st$ eigenvectors. The eigenvectors are

arranged as rows in matrix $\mathbf{A}$ and this transformation matrix is used to compute the new feature vector by projecting as, $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$. With this we get the best one dimensional representation of the component images with reduced feature size.

Principal component analysis (PCA) yields projection directions that maximize the total scatter across all classes. In choosing the projection which maximizes total scatter, PCA retains not only between-class scatter that is useful for classification, but also within-class scatter that is unwanted information for classification purposes. Much of the variation seen among document images is due to printing variations and degradations. If PCA is applied on such images, the transformation matrix will contain principal components that retain these variations in the projected feature space. Consequently, the points in the projected space will not be well separated and the classes may be smeared together.

Thus, while the PCA projections are optimal for representation in a low dimensional space, they may not be optimal from a discrimination standpoint for classification. Hence, in our work, we further apply linear discriminant analysis to extract useful features from the reduced dimensional space using PCA.

4.4.2 Linear Discriminant Analysis

The use of linear transformations for dimensionality reduction is attractive because they are simple to compute and analytically tractable; at the same time selects features based entirely upon their discriminatory potential. Linear discriminant analysis (LDA) projects the high dimensional data onto a lower dimensional space. Let the projection be $\mathbf{y} = \mathbf{D}\mathbf{x}$, where $\mathbf{x}$ is the input image and $\mathbf{D}$ is the transformation matrix. The objective of LDA is to find a projection that maximizes the ratio of the between-class scatter and the within-class scatter. There are different ways to perform discriminant analysis. Some of these include Fisher discriminants [54], Optimal Discriminant Vectors of Foley-Sammon [60], etc.

We apply the algorithm proposed by Foley and Sammon for linear discriminant analysis. The algorithm extracts a set of optimal discriminant vectors (ODV) for a two-class problem which suits the Support Vector Machine (SVM) classifier we used for classification task.

Consider the i^{th} image sample represented as an $M^{'}$ dimensional (column) vector $\mathbf{x_i}$, where $M^{'}$ is the reduced dimension using PCA. From the given sets of training components, $\mathbf{x_1}, \cdots, \mathbf{x^{'}_M}$, we compute with-in class scatter (W) matrix and between-

class difference (δ) as:

$$W_i = \sum_{j=1}^{M_i^{'}} (\mathbf{x_j^i} - \mu_i)(\mathbf{x_j^i} - \mu_i)^T \tag{4.3}$$

$$\delta = \mu_1 - \mu_2 \tag{4.4}$$

where $\mathbf{x_{ij}}$ is the j^{th} sample in the i^{th} class.

Sum of the within-class scatter is also determined by:

$$A = cW_1 + (1-c)W_2 \tag{4.5}$$

where $0 \leq c \leq 1$ and the scatter space using

$$S_{ij} = d_i A^{-1} d_j \tag{4.6}$$

Algorithm 1 Optimal discriminant features

1: $d_1 = \alpha_1 A^{-1}\Delta$
2: $S_1^{-1} = 1/S_{11}$
3: **for** $n = 2$ to K **do**
4: $\quad d_n = \alpha_n A^{-1}\{\Delta - [d_1 ... d_{n-1}] S_{n-1}^{-1}[1/\alpha_1 0 ... 0]\}$
5: $\quad \omega_n = (d_n^t \Delta)^2 / d_n^t A d_n$
6: $\quad S_n^{-1} = \frac{1}{c_n}[\frac{c_n S_{n-1}^{-1} + S_{n-1}^{-1} y_n y_n^t S_{n-1}^{-1}}{-y_n^t S_{n-1}^{-1}} | \frac{-S_{n-1}^{-1} y_n}{1}]$
7: **end for**
8: **Output:** discriminant values $\omega_i, 1 \leq i \leq M^{''}$

Algorithm 1 is called recursively for extracting an optimal set of discriminant vectors d_n that corresponds to the first $M^{''}$ highest discriminant values such that $\omega_1 \geq \omega_2 \geq \cdots \geq \omega_M^{''} \geq 0$. Here, K is the number of iterations for computing discriminant vectors as well as discriminant values, and α_n is chosen such that $d_n^t d_n = 1$. c_n and y_n are determined in the following manner:

$$c_n = s_{nn} - y_n^t S_{n-1}^{-1} y_n \tag{4.7}$$

$$y_n = [S_{in} \cdots S_{(n-1)(n)}] \tag{4.8}$$

It should be noted that the first $M^{''}$ optimal discriminant features are enough for lower dimensional representation without affecting the performance of modular classifier.

To investigate the performance of PCA and LDA as compared to the original features, we conduct experimental analysis on the various dataset taken from the

UCI machine learning repository [135]. These datasets are used by the machine learning community for the empirical analysis of machine learning algorithms. We use five of the datasets for the experiment: Vowel, Satimage, Pendigits, Letter and Optdigits. These datasets have 11, 5, 10, 26 and 10 classes, respectively, and each of them contains more than 1000 samples.

Dataset	Original		PCA		LDA	
	Feature dimension	Accuracy rate	Reduced dimension	Accuracy rate	$d *^N C_2$	Accuracy rate
Vowel	10	55.31	9	53.19	5*55	53.19
Satimage	19	88.03	17	87.58	5*10	90.12
Pendigits	16	95.77	13	95.46	5*50	95.99
Letter	16	72.80	15	75.16	5*325	81.20
Optdigits	64	97.90	41	97.90	5*50	98.63

Table 4.1: Comparison of the performance of PCA and LDA with the original features.

Table 4.1 shows the test results obtained. In almost all experiments LDA outperforms the other two representations. This is because of its ability to extract optimal discriminant features that suits SVM-based DDAG classifiers. Hence, the use of linear discriminant features for each pair of classes in each component classification is an attractive solution to design better classifiers.

4.4.3 Combining PCA and LDA

Since the principal purpose of classification is discrimination, the discriminant vector subspace offers considerable potential as feature extraction transformation. However, this approach suffers from large storage space requirement since independent low-dimensional representations need to be computed for each pair of component classifier and is computationally expensive for large class problems requiring stack space for storing $N(N-1)/2$ transformation matrices. For instance, for dataset with 200 classes LDA generates around 20 thousand transformation matrices for each pair and is very expensive for large class problems.

Discriminant analysis of principal components has wider application for face recognition [201][202] and for facial image retrieval [173], in which good generalization of the system was demonstrated by experiments that carried out testing on new classes/individuals without retraining the PCA bases, and sometimes the LDA bases. This is because PCA and LDA learned from any subset can yield more efficient rep-

resentation for any specific variation. With this approach, a linear projection using PCA transform maps the input image x in to the character or connected components-subspace y. Then, linear discriminating transform in PCA feature space maps the connected components-subspace in to the classification subspace z. After this composite linear projection, the classification is performed in the classification space based on some distance measure criterion.

In this work, we propose a two-stage feature extraction scheme: PCA followed by LDA for optimal discriminant feature extraction that reduces storage and computational complexity, while enhancing the performance of SVM-based DDAG classifiers. We first reduce the feature dimension using PCA and then run LDA on the reduced lower dimensional space to extract the most discriminant feature vector. The discriminant vector is used for training and testing SVM-based decision directed acyclic graph (DDAG) classifier.

4.5 Classification

Training and testing are the two basic phases of any pattern classification problem. During training phase, the classifier learns the association between samples and their labels from labeled samples. The testing phase involves analysis of errors in the classification of unlabeled samples in order to evaluate classifier's performance. In general it is desirable to have a classifier with minimal test error.

We use Support Vector Machine (SVM) for classification task [145][188]. SVMs are pairwise discriminating classifiers with the ability to identify the decision boundary with maximal margin. Maximal margin result in better generalization and a global solution for the problem, which is a highly desirable property for a classifier to perform well on a novel dataset. Support vector machines are less complex (smaller VC dimension) and perform better (lower actual error) with limited training data. Classifiers like KNN (K-nearest neighbor) provide excellent results (very low empirical error) on a dataset on which they are trained, while the capability to generalize on a new dataset depends on the labeled samples used. Computational complexity of KNN increases with more and more labeled samples. Thus, the SVM classifier is more suitable for OCR problems with large number of classes and high dimensional input data due to its effective training and testing algorithms and natural extension to the kernel methods. Since the design of SVMs allows the number of support vectors to be small compared to the total number of training examples, this scheme provides a compact representation of the dataset as a result of which computation and storage requirements to implement SVM classifier on large datasets is reduced.

SVMs have a wider application for isolated handwritten digit recognition, object recognition, speaker identification, face detection in images, and text categorization [34].

4.5.1 Support Vector Machine

The SVM classifier is a two-class classifier based on the use of discriminant functions. A discriminant function represents a surface which separates the patterns so that the patterns from the two classes lie on the opposite sides of the surface. The SVM is essentially a separate surface.

Consider labeled training dataset $x_i, y_i, i = 1, ..., l, y_i \in -1, 1, x_i \in \Re^d$ where d is the dimensionality of the dataset. Suppose we have a hyperplane that separates the positive from the negative examples. The points x which lie on the hyperplane satisfy $wx + b = 0$, where w is normal to the hyperplane and b is the offset. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin; the point where the training data satisfy the following constraints.

$$w \cdot x_i + b \geq 1 \; for \; y_i = +1 \tag{4.9}$$

$$w \cdot x_i + b \leq -1 \; for \; y_i = -1 \tag{4.10}$$

Identification of the optimal hyperplane for separation involves maximization of an appropriate objective function, i.e, solving the following quadratic optimization problem during training an SVM.

Maximize:

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x) \tag{4.11}$$

subject to the constraints $\alpha_i \geq 0, i = 1, \cdots, l$ and

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{4.12}$$

where α_i are the Lagrangian multipliers corresponding to each of the training data points x_i. The result of the training phase is the identification of a set of labeled support vectors x_i and a set of coefficients α_i. Support vectors are the samples near the decision boundary and most difficult to classify. They have class labels y_i ranging ± 1. The decision is made from:

$$f(x) = sgn(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x)) \tag{4.13}$$

where the function K is the kernel function. It is defined as $K(x, y) = \phi(x).\phi(y)$, where $\phi : \Re^d \longrightarrow H$ maps the data points in d dimensions to a higher dimensional (possibly infinite dimensional) space H. For a linear SVM, $K(x, y) = x.y$. We do not need to know the values of the images of the data points in H to solve the problem in H. By finding specific cases of Kernel functions, we can arrive at Neural Networks or Radial Basis Functions. More detail discussion and a tutorial on SVM is available in [34].

Binary classifiers like SVM are basically designed for two class classification problems. However, because of the existence of a number of characters in any script, optical character recognition problem is inherently multiclass in nature. The field of binary classification is mature, and provides a variety of approaches to solve the problem of multiclass classification. Most of the existing multiclass algorithms address the problem by first dividing it into smaller sets of a pair of classes ($^NC_2$) and then combine the results of binary classifiers using a suitable voting methods such as majority or weighted majority approaches.

4.5.2 Multiclass Classification

Multiclass SVMs are usually implemented as combinations of two-class solution. The original technique for multiclass classification using SVMs is known as the one-versus-rest technique [54]. More recently an improved technique, one-versus-one [75], has been developed. In the case of one-versus-rest, the ith class is trained with all of the examples in the ith class against all other examples, resulting in an $N - 1$ classifiers (where N is the number of classes). The class label of the test result is decided by combining the N classifier outputs using, say winner-take-all method. Where as, one-versus-one consists of a combination of binary classifiers, only two classes are used to train each classifier. In an N-classification problem, one-versus-one technique results in $N(N - 1)$ classifiers. The decision is made by combining these $N(N - 1)$ classifier outputs using some voting strategy (e.g. majority vote, max wins, etc.).

Graphs and trees are effective means for integrating multiple classifiers [145][151]. A tree is a simple graph such that there is a unique simple non-directed path between each pair of vertices of a graph. Decision trees are classical examples of a tree-based classifier combination scheme. In a decision tree classifier, decision is taken at each node based on some of the attributes and samples traverse down the tree along the selected path. Another possible tree-classifier is by designing the tree such that each node partitions the classes into mutually exclusive sets. A decision directed acyclic graph (DDAG) can be designed to solve the same problem in a better manner

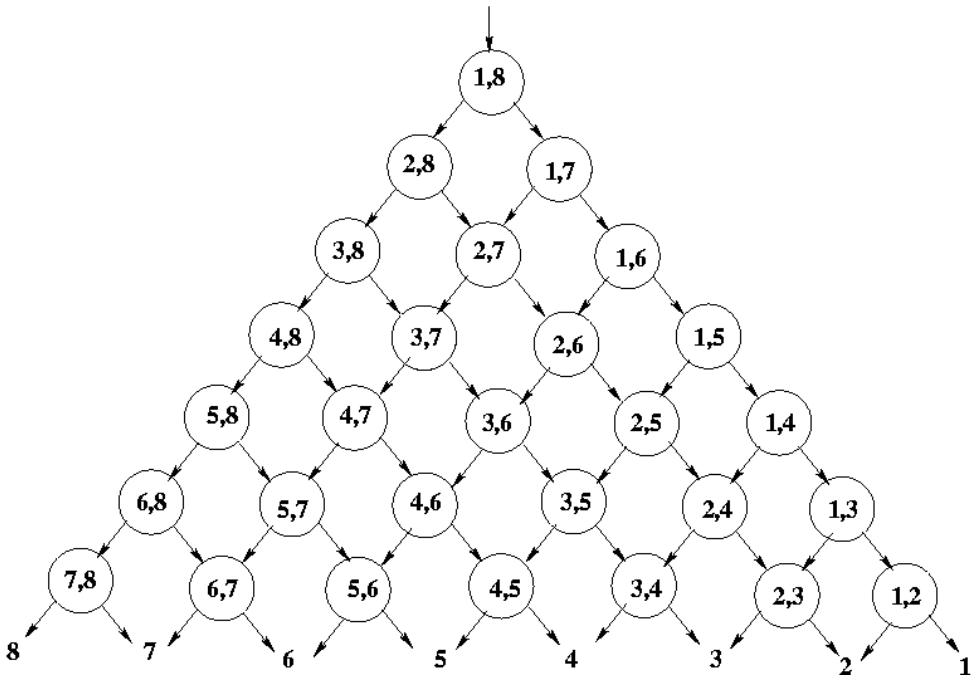


Figure 4.5: A rooted binary DDAG multiclass classifier for 8 class problem

without partitioning into mutually exclusive sets of classes. This can reduce the risk in making a mistake by giving too much importance to the root node, or the nodes that are encountered initially. It has been shown that the integration of pairwise classifiers using DDAG results in better performance as compared to other popular techniques such as decision tree, Bayesian, etc. [145].

DDAG is defined as follows [145]. Given a space X and a set of boolean functions $\mathcal{F} = \{f : X \rightarrow \{0, 1\}\}$, the class $DDAG(\mathcal{F})$ of Decision DAGs, on N classes over $\mathcal{F}$ are functions which can be implemented using a rooted binary DAG with N leaves labeled by the classes where each of the $K = N(N-1)/2$ internal nodes is labeled with an element of $\mathcal{F}$. The nodes are arranged in a triangle with the single root node at the top, two nodes in the second layer and so on until the final layer of N leaves. The ith node in jth layer, $j < N$, is connected to the ith and $i+1$-st node in the $j+1$-st layer.

We construct a directed acyclic graph (DAG) using the one-vs-one method, where each node in the graph corresponds to a two-class classifier for a pair of classes [145]. The multiclass classifier built using DDAG for an 8-class classification problem is depicted in Figure 4.5. It can be observed that the number of binary classifiers built for a N class classification problem is $N(N-1)/2$. The input vector is presented at the

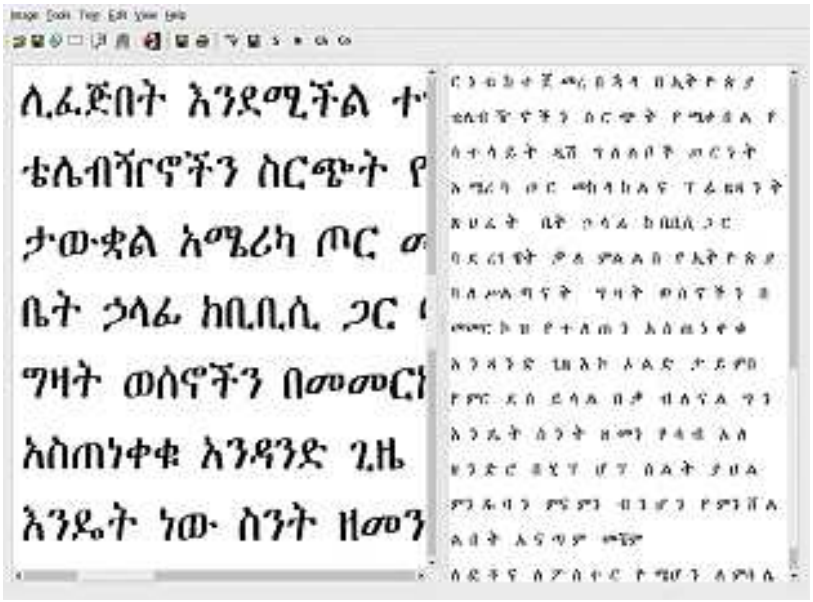


Figure 4.6: Screenshot of the user interface. The left side displays scanned image, while the right side shows its equivalent textual form after recognition.

root node of the DAG and moves through the DAG until it reaches the leaf node where the output (class label) is obtained. At each node a decision is made concerning to which class the input belongs. For example, at the root node of Figure 4.5 a decision is made whether the input belong to class 1 or class 8. Samples from other classes get classified into left or right subtree. Practically each node in the evaluation sequence removes one possible class. If it is not belong to class 1, it moves to the right child. In the same way, if it is not belong to class 8, it moves to the left child. This process is repeated at each of the remaining nodes till it reaches the leaf node where the decision is reached to classify the input as one of the eight classes.

DDAG-based SVM classifier is trained pair-wise with the discriminant features selected from the given two-class characters using the two-stage feature extraction scheme (PCA followed by LDA) as presented in Algorithm 2. The classifier then creates two-class models that are used for classification and recognition purpose.

4.6 Results and Discussions

We have a prototype system for the recognition of a given Amharic document images into equivalent textual format. A screenshot of an interface is shown in Figure 4.6. The system accepts scanned document image. Scanned pages are preprocessed, that

Algorithm 2 PCA-LDA directed DDAG-based SVM classifier

1: Given dataset $x_1, x_2, ..., x_N$ with M dimension and K classes
2: Use PCA to reduce the dimension to $M^{'}$
3: **for** $i = 1$ to K **do**
4: **for** $j = K$ down to $i + 1$ **do**
5: extract ODV with dimension $M^{''}$ for class (i, j)
6: use SVM to classify classes (i, j) based on their ODV
7: **end for**
8: **end for**

is, binarized, skew corrected, scaled and noise removed before individual components are extracted. Preprocessed pages are then segmented into character components for feature extraction and classification. Following Amharic letters shape formation, we can first decompose lines in a text page into words and then words into appropriate character components for recognition and then recompose the recognized components to determine the order of characters, words and lines in a text page. To improve recognition result post-processor can be used. The character level recognition results can be combined to get the word level output. At this stage, one could use a post-processor to improve the recognition accuracy. Statistical information collected (about the frequency of occurrence of characters in a text) using unigram language model provides apriori information to construct the post-processor.

Once character components are identified, optimal discriminant features (in a lower dimensionality space) are extracted for classification. We use a two stage dimensionality reduction scheme based on 99 percent principal component analysis which is followed by 15 percent linear discriminant analysis. The original dimension of feature vector of each character image after normalization is 400 (20×20). Principal component analysis reduces the dimensionality from 400 to 295 and linear discriminant analysis further to 50. We are dealing with such reduced optimal discriminant feature vectors. Both methods perform well in feature dimensionality reductions. The use of a two-stage feature extraction scheme further solves the problem of confusion of similar characters (say, between ሀ and ሐ) when PCA only is used for representation. This is because the new scheme extracts optimal features that discriminate between a pair of characters during classification using support vector machine. It is observed that, for example, given characters ሀ and ሐ, the use of PCA alone might discover the gross features of characters ሀ and ሐ, but ignore the upper stroke direction that distinguishes ሀ and ሐ.

Res	Fonts				Point Size				Style		
	PG	VG	AG	AL	10	12	14	16	Normal	Bold	Italic
DS	7850	7850	7850	7850	7680	7680	7680	7680	7680	7680	7680
%	99.08	96.24	95.53	95.16	98.64	99.08	98.06	98.21	99.08	98.21	89.67

Table 4.2: Recognition rate on the various fonts, sizes and styles. N.B. Res = Result; DS = Dataset; % = Accuracy rate.

We conduct extensive experiments to evaluate the performance of the recognition process on the various datasets of Amharic scripts. The experiments are organized in a systematic manner considering the various situations encountered in real-life printed documents. Our datasets are of two types: One set of datasets considers printing variations (such as fonts, styles and sizes). The other contains degraded documents taken from newspapers, magazines and books. We report the performance of the Amharic OCR in all these datasets and the result obtained is promising to extend it for other indigenous African scripts.

4.6.1 Recognition Results on Printing Variations

Performance evaluation is done in a step-wise manner as follows. In the first experiment, we consider the printing variation. We test on the most popular fonts such as PowerGeez (PG), VisualGeez (VG), Agafari (AG) and Alpas (AL) that are used for typing and printing purposes, four point sizes (10, 12, 14 and 16) and font styles (such as normal, **bold** and *italics*). We consider an average of more than 7500 samples for the experiment. Performance results are shown in Table 4.2.

High recognition rates were obtained for multiple point size. The results are almost uniform through out all font sizes as we scale them to a standard size of 20×20 before feature extraction and dimensionality reduction. This makes the system invariant to point size variations.

The recognition rate for fonts is more than 95 percent. It works well for 'Power Geez' font as the system is trained on datasets prepared using this font. There is around three percent accuracy difference as compared to other fonts. Recognition accuracy is comparable for other fonts. Mis-classifications mainly occurred because of two reasons. The first problem is related to the similarity in vowel formation between third and fifth orders of the same base characters. The degree of complexity of characters shape formation by individual font is also another factor for the reduction in the accuracy rate.

The system works well for normal and bold styles with more than 98 percent accu-

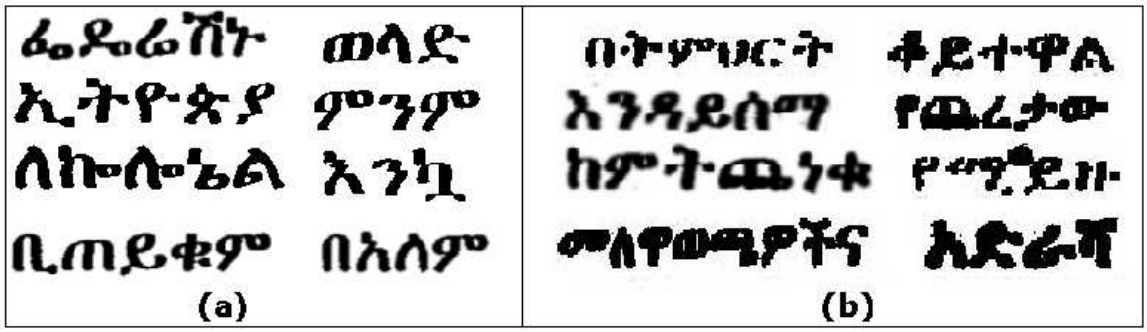


Figure 4.7: (a) A set of correctly recognized sample words, (b) A set of mis-recognized sample words

racy. Since we trained the OCR deliberately with normal font style, the recognition rate for italics is reduced. This is because the shape of characters written using italics style is very complex unless skew is corrected. It is obvious that, as can be done else where, better result can be obtained by retraining the classifier with added new training samples of italics style.

4.6.2 Recognition Results on Real-life Documents

In general, better performance has been registered on the above synthetic datasets; on the average 96.95% accuracy is obtained. This is because paper and printing qualities were reasonably good. Figure 4.7 (a) shows sample word images of this type that are correctly recognized. The challenge with these datasets is printing variations that happened due to fonts, sizes and styles used for document production.

In real-life situations, however we also encounter the problem of degradations in document images. To see the effect of this artifacts, we evaluate the performance of the OCR on Amharic documents digitized from books, magazines and newspapers. We apply Gaussian filtering algorithm to reduce the effect of degradations in the images during the recognition process. Recognition results are shown in Table 4.3.

Document	Test Data size	Accuracy (%)
Books	6240	91.45
Newspaper	5430	88.23
Magazine	5560	90.37

Table 4.3: Performance of the system on degraded real-life documents

On the average around 90 percent recognition is obtained. As depicted in Figure 4.7 (b) characters are mis-recognized because of artifacts such as cuts, merges, ink blobs, etc. that are commonly observed in printed document images scanned from books, magazines and newspapers. Degradations due to cuts break character components into two or more, and ink-blobs and merges join disjoint characters as one connected components. Due to these changes, for instance, a cut at the middle of the character ሀ results in visually similar characters with ሃ and ገ, and also a cut below ዐ or above ዐ results in similar characters with ባ or ሀ, respectively. Further, a blob below the horizontal line change character ሟ to character ወ, a blob that fills the hole of character ፉ looks the same as ረ, etc. The same thing also happens with merge; for instance, a merge of characters ባ and ገ results in ጠ. This requires degradation modeling in order to simulate the effect of noise and accordingly design advanced noise removal and image enhancement algorithms before applying segmentation, feature extraction and classification techniques.

In general, the major challenges in designing an OCR for Amharic language are as follows. One, there are large number of character categories (classes). Two, the existence of characters with similar shape and structure. Three, the existence of wide variety of character shapes due to different type faces and image quality in printed documents.

As the complexity and diversity of document images increases, the performance of the OCR is greatly affected since it is sensitive to degradation and unseen fonts. This is also true for those OCRs designed for Latin-based scripts. As a solution, most commercial systems suggest retraining the OCR engine again with the unseen samples for performance improvement. This is however inefficient and time consuming. Hence, there is a need to investigate alternate approaches.

4.7 Summary and Comments

We present an OCR system for the recognition of printed document in Amharic language. We employ a two-stage feature extraction procedure, PCA followed by LDA. Our feature extraction scheme selects optimal discriminant vectors for representation. DDAG-based SVM classifier is implemented for classification task. The system is now being extensively tested on both printing variations and degraded documents. The use of SVM classifier is advantageous because of their generalization capability.

Performance analysis of the system shows that our OCR works well on reasonably good quality printed documents. However, several external issues affect its performance for attaining consistent recognition rate across documents that vary in fonts,

styles and quality. We observe that the accuracy of the system declines with a change in the quality of documents and font styles such as italics. This is because of the inherent problem of top-down design approach. It allows only recognizing one page at a time. In this set up, there is no mechanism to feedback information obtained during the recognition process through which an OCR learns and improves its performance over time. This necessitates redesigning the architecture of the OCR to work on document image collections (say, a book).

In general, the following issues need further consideration for improving the performance of an OCR system designed following the top-down approach.

- *Quality:* The quality of the original document greatly affects the performance of the OCR. This is because (i) the original document might be old, and has suffered physical degradation, (ii) it might also be a low quality document, with variations in toner density where ink blobs may occur, and (iii) many characters are broken/cut which presumably are faint areas on the original which is not picked up by the scanning process.

- *New fonts and styles:* Books and newspapers are written in distinct fonts and various styles. When the system is presented with such unseen fonts and styles, its performance drastically reduces. These are additional challenges for the OCRs design.

To improve the above problems the OCR system requires retraining with the unseen pages. Since the system is static in nature (with no feedback mechanism), retraining involves discarding the previous model and training the system again offline. This is time consuming and not efficient to work in a large document image collections in an interactive and intelligent way. We need to follow a different approach to the OCR design that can enable it learn from its mistakes online and improve its performance over time. In the next chapter we present an architecture for self adaptable OCR that targeted to work on document image collections.

Chapter 5

Self Adaptable Recognizer for Document Image Collections

5.1 Introduction

The success of access to document image in the newly emerging digital libraries considerably depends on the availability of robust OCRs that can take care of the diversity in the document image collections [118]. We need a recognition system that is capable of intelligently adapting to the characteristics of documents of interest.

Most of the recent research in OCR has been centered around building fully automatic, high performing intelligent classification systems with very high generalization capability [131]. Intelligent OCRs with excellent performance on a given page is reported for Latin scripts [86][178] and some of the Oriental languages [175]. However, when it comes to the suitability of converting an old book to text and providing a text-like access, even the present day OCRs are found to be insufficient [118]. The performance of these recognizers decline as diverse collection of documents (with unseen fonts and poor quality documents) are submitted for recognition. We argue that an adaptive recognition system, which can learn from its mistakes, is better suited for such document collections. An intelligent OCR that learns from its experience and improves its performance over time, can adapt to varying document image collections in printing (font, point size and type style), scripts and quality. It leads to better results. This has become necessary for the recognition of poor (or mediocre) quality documents with unseen fonts from a large collection of digitized books and manuscripts.

In this work, we argue that:

1. The technique for recognizing a book (a reasonably large collection of documents

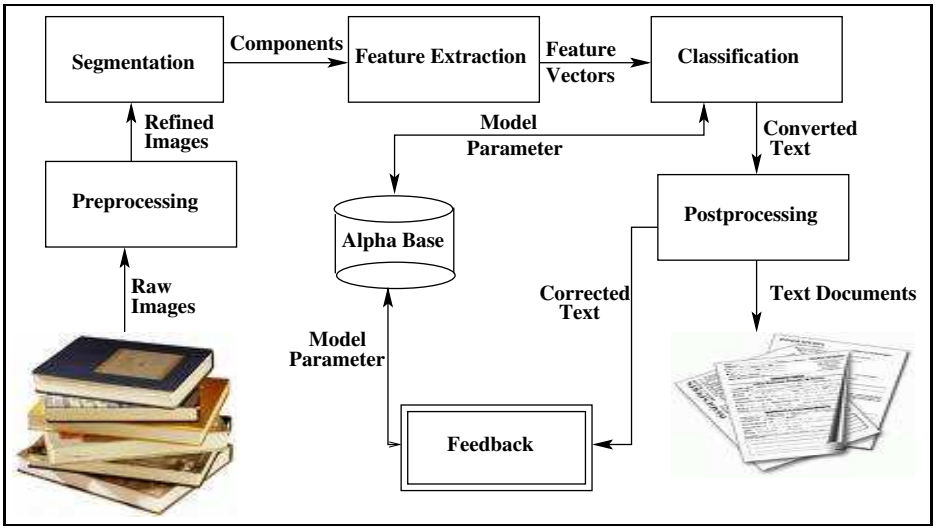


Figure 5.1: An architecture of OCR that closes the conventional open-loop system of classifier followed by postprocessor. This happens by introducing a feedback mechanism during the recognition process.

in single font and consistent formatting) can be different from that of OCRs that are designed to be part of scanner drivers or for isolated pages.

2. The notion of generalization from a small set of training samples, which is critical to the pattern classification problem, need not be the only performance goal. One can wish of 'overfitting' to a book, as long as the recognition engine can provide better results on the specific collection. In general, the multimodality of the data distribution need not be completely modeled; but can be accepted as a reality.

In this work, deviating from the conventional OCR design, we present the prospect of formulating an intelligent and self adaptable recognizer for a collection of document images. We explore the idea of obtaining better performance at the cost of additional computation. An approach for designing a semi-automatic adaptive OCR for document images in digital libraries is presented in [158]. The OCR is designed to support an interactive retraining (based on users feedback) for performance improvement. During the recognition process users evaluate the performance of the OCR and feedback additional new samples for further retraining.

By integrating machine learning algorithms for validating, labeling, sampling and incremental learning in a recognition cycle, we design self adaptable OCR for document image collections [124], as shown in Figure 5.1. Recently Neeba and Jawahar

also reported extension work [132]. Postprocessor based feedback mechanism is enabled to provide additional knowledge to the recognizer. The postprocessor accepts words generated by the recognizer and produces error-corrected ones. The postprocessor significantly improves the accuracy of the OCR by suggesting, often, a best-fit alternative to replace the mis-spellings. Mis-recognized samples that are identified by the postprocessor, are validated, labeled and fedback to the OCR such that it builds its knowledge through learning in an incremental fashion. This scheme can repeatedly work on the collection and get adapted to the book (or a collection). As a byproduct of the adaptation, the system also builds parameters for reuse at a later stage.

5.2 Overview of the Design

Machine learning offers one of the most cost effective and practical approaches to the design of pattern classifiers for a broad range of pattern recognition applications like character recognition. Learning algorithm could be supervised, unsupervised or reinforcement [128]. In supervised learning, a teacher provides labels for each pattern in a training set and seeks to minimize the error in prediction. Getting such labeled data is, however costly since it requires expert annotation to create, but is literally exponentially more useful than unlabeled data. However, considerable research has shown that unlabeled data can frequently be used to improve the accuracy of learning algorithms. Learning from unlabeled data is referred to as unsupervised learning. Unsupervised learning tries to form clusters or natural grouping of the input patterns without an explicit teacher. The distinction between supervised and unsupervised learning is the use of labeled vs. unlabeled data. Semi-supervised learning employs minimal supervision and extrapolate the knowledge to unlabeled data. It attempts to combine supervised and unsupervised learning. In semi-supervised procedure, the classifier can be crudely designed using small set of labeled samples, and then tuned up by allowing it to run without supervision on a large, unlabeled set.

Reinforcement learning employs a critic to provide a binary criticism (either accept or reject). Using reinforcement learning, the classifier is trained in the following manner: present training samples to the classifier, compute its tentative category label, and use the known target category label to improve the performance of the classifier. Supervised techniques have been successfully demonstrated for character recognition application as offline training in OCR systems. However applicability of semi-supervised and reinforcement learning is not yet explored in its full potential. To come up with robust recognizers we need to exploit the potential of such learning

paradigms.

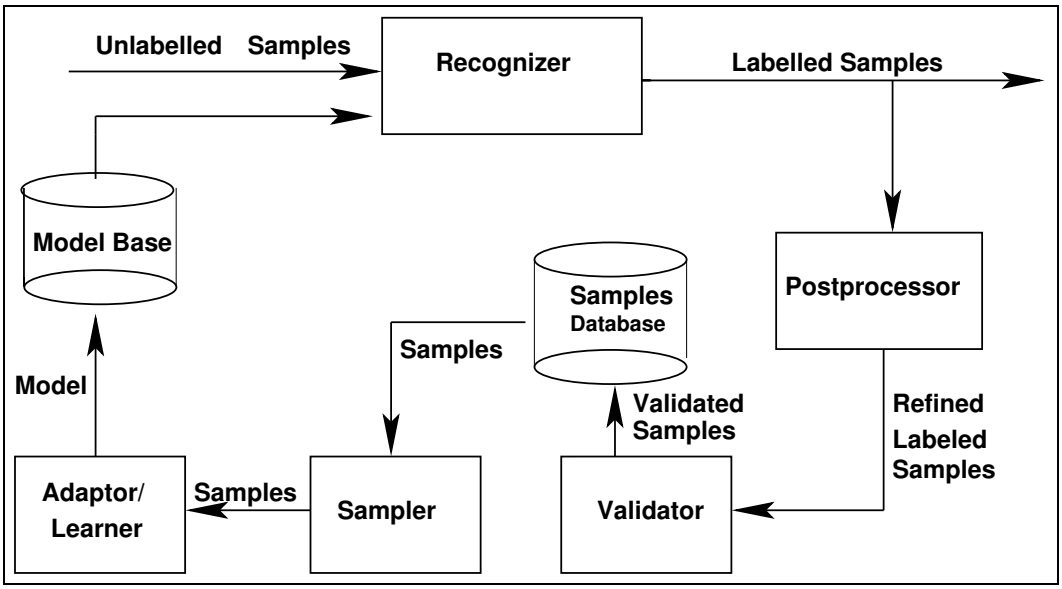


Figure 5.2: The learning framework of the OCR in the recognition of document image collections

Architecture of the proposed learning recognition scheme in a given document image collections is shown in Figure 5.2. The figure is basically an expanded view of the feedback block presented in Figure 5.1. We follow Algorithm 3 for the recognition of text in document image collections. Given a word image for recognition, the Basic Recognizer Unit (BRU) converts it into text. A postprocessor is then used to rectify/verify the recognized word. In our present implementation, we use a simple dictionary-based postprocessor which is designed following a reverse dictionary approach with the help of a trie data structure. In this approach, two dictionaries are created. One is filled with words in the normal manner and the other with the same words reversed. This method narrows down the possible set of choices by using both dictionaries.

Our Basic Recognition Unit is an SVM-based classifier (which is discussed in Section 4.5) trained offline on few *apriori* available synthetic training examples (that are prepared in single font, style and size) by extracting vector representation of the entire image. In this work, the conventional open-loop system of classifier followed by postprocessor is closed by automatically generating training data from the "test" image and retraining (in fact incrementally modifying) the basic recognition unit. Before passing mis-recognized samples as new training datasets we run a validator to detect outliers that deviate from the original sample. Samples that are similar to

Algorithm 3 Recognition of text in document image collections

1: Document images are preprocessed and words are extracted for recognition.
2: BRU outputs the recognized text.
3: Postprocessor checks the validity of the word based on some language model and outputs the corrected word, if found to be invalid.
4: Validator visually validates the result of post-processing. By matching the rendered text and word image, visual similarity is carefully computed. Errors introduced by the postprocessor results in outliers.
5: Those with visually similar appearance are considered as new training samples. Semi-supervised learning is used in labeling the new samples.
6: Bagging is used to create a sample set and classifier is trained (preferably incrementally) to obtain the new set of parameters and stored in α-base.
7: Base-classifier incrementally learns for knowledge updation. The newly created models are used for subsequent recognition.
8: Repeat steps 2-7 until the required accuracy is achieved or the performance improvement is less than a threshold set.

the original ones are labeled using labeler and transferred to the sample database. This scheme, thus, has a framework for the creation, validation, labeling and use of mis-recognized samples as new training examples for performance improvement over time.

Our design of the OCR is highly data-driven. It employs feedback for performance improvement. Initial recognition errors are immediately detected and corrected. This technique is also useful when sufficient training samples or *apriori* knowledge is lacking (for example recognizing in a new font/style). The implementation of this approach could be computationally intensive. However, today's OCR environment, especially the ones available for digitizing books and large document image collections can afford to do so.

5.3 Learning Schemes

There are possibly two ways in which the continuous performance improvements can take place in recognizers for document collections in digital libraries.

1. Learn from the direct or indirect user feedback (as in the case of relevance feedback models, search engines etc.) captured during the search.

2. Improve the performance of recognizers by adapting to a specific collection by absorbing the input from language models (e.g.. Dictionaries).

In our case, we do not model the user interactions, and follow the second approach, where knowledge from the postprocessor is used for developing new labeled examples, and thereby improving the recognition accuracy.

In the conventional OCRs also, postprocessor that employs dictionaries or language models are integrated for improving the performance of the base-classifier. However, they are not designed to learn from their experience. If a word which is misclassified once is given to the base classifier, it repeats the same mistake again. However, we are interested in a framework, which will make the classifier intelligent and correct the mistake in the future.

We integrated a number of modules that suits the learning framework:

- Sampler to make sure that each time new datasets are selected and fed for training the classifier.
- Incremental Classifier to incrementally learn to update the model used for recognition during normal operation.
- Labeler to label unlabeled new examples generated through the feedback mechanism designed during the learning process.
- Validator to control the quality of the new training samples (passed through feedback) in the image-space.

5.3.1 Automatic Preparation of Labeled Data

The learning framework we devise is a data-driven one, where large quantities of new samples flow in the framework. One of the basic problem of the learning process in such a dynamic environment is getting high quality new training samples. Once the Basic Recognition Unit (BRU) recognizes a word and outputs a textual representation, postprocessor verifies the validity of the word with the help of a dictionary (or a language model) and either accepts the word as valid or corrects with an alternative. There are five possible situations.

1. BRU correctly recognizes a word and postprocessor accepts it as valid word.
2. BRU correctly recognizes, but postprocessor fails to accept it as valid and suggests an alternative which is a wrong word.

3. BRU makes a mistake, and postprocessor corrects to the right word.
4. BRU makes a mistake, and postprocessor corrects/modifies to a wrong word.
5. Postprocessor fails to suggest an acceptable alternative to a text provided by BRU.

We employ the knowledge from the postprocessor in creating a new set of labeled examples. Postprocessor corrects wrongly recognized words with the best match. If recognized words match one-to-one with the knowledge of the postprocessor, then the postprocessor accepts it as valid word (situation 1). If some of the recognized constituent characters of the word matches with the knowledge of the postprocessor, even though the BRU makes the mistake the postprocessor corrects to the right word (situation 3). There is, however a possibility of suggesting unrelated word (with the original one) for replacement. This happens because of two situations. One is when words are not correctly recognized by BRU (situation 4 and 5). The other is when the recognized words are not in the dictionary (situation 2). By adding the word to the knowledge base, the post processor corrects the mistake in the future.

Thus, situations 2, 4 and 5 results in outlier words. Consider a scenario (of situation 4) where the word "century" is recognized as "ccutnry". When the word "ccutnry" is presented to the postprocessor, it suggests "country" as the most probable word for replacement, which is unrelated with the original word. We attempt to detect and remove such outliers from feeding back as a new training samples in the learning process. To detect such outliers we validate the result of the post-processing, by matching in the image space. *This is more of verification rather than recognition.*

Given the text, it is rendered into an image by setting font and point size, and then matched with the original word image. Let $p_1, p_2, \ldots p_m$ be a set of feature vectors extracted by scanning the vertical strips of the word image and $q_1, q_2, \ldots q_n$ be the sequence of feature vectors corresponding to the vertical strips of the rendered word image, where m and n are width of the given words. They are aligned using dynamic time warping (DTW) for similarity measure. If all the symbols match (one-to-one), then we have achieved the labeling of all the connected components in the document image. When only some of the symbols match, they are labeled and the rest of the symbols are treated as unlabeled. In short, at the end of the first phase of validation using a DTW-based algorithm, most of the components from a given document image gets labeled automatically and added to the sample database. The rest of the samples are treated as unlabeled. Note that, at this stage, our interest is limited to improving the performance of BRU, rather than addressing the problem of merges and splits in the document images.

Algorithm 4 EM algorithm

1: Initialize parameter $\theta = \mu_j^0$, Σ_j^0, P_j^0 based on the class size C.
2: [E-step:] Compute the probability P_{ij}^{t+1} that x_i belongs to class j given parameter θ^t using multivariate Gaussian distribution (where t is iteration counter. Initially it is 0).
3: [M-step:] Re-compute the parameter values:

$$P_j^{t+1} = \frac{1}{n}\sum_i P_{ij}^{t+1} \tag{5.1}$$

$$\mu_j^{t+1} = P_j^{t+1}\frac{\sum_i P_{ij}^{t+1} x_i}{\sum_i P_{ij}^{t+1}} \tag{5.2}$$

$$\Sigma_j^{t+1} = \frac{\sum_i P_{ij}^{t+1}(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_i P_{ij}^{t+1}} \tag{5.3}$$

4: Terminate if converged, else go to E-step
5: **Output:** Optimal parameter values μ_j^i, Σ_j^i, P_j^i

For each unlabeled data x_i we attach probabilities p_{ij} of belonging to class c_j, i.e. $p_{ij} = prob(x_i \in c_j)$. This is an initialization. These probabilities are then iteratively improved by an Expectation Maximization (EM) [193] based formulation. EM is often used as a clustering algorithm with the objective of maximizing the likelihood of class belongingness of the unlabeled samples (x_i) given parameter θ:

$$\theta' = argmax \prod_{i=1}^{n} p(x_i|\theta) \tag{5.4}$$

The basic algorithm of EM involves two steps (see Algorithm 4). The E-Step uses the current parameter estimates to find the best probabilistic labels for class membership using a multivariate normal (Gaussian) distribution. The M-Step then refines the parameters to maximize the total likelihood. The steps are iterated until the change in parameter values falls below some predefined threshold. In a way, it allows to put the unlabeled examples into a cluster with most of the samples already labeled. When most of the samples are labeled, it converges in one step, and assigns label to the unlabeled samples.

5.3.2 Sampling Training Examples

Pattern classifiers learn from the training examples provided. The ability to actively select the most useful training samples is an important aspect of building an efficient classifier. Boosting and bagging are being increasingly used for sampling representative datasets for training classifiers [79][128]. They generate representative samples for training classifiers such that the learning algorithm has some control over the subset of the input space used in the training.

Boosting uses all instances of the datasets at each iteration, but associates a weight for each sample that reflects its relative importance for correctly predicting the outcome. A classifier is then trained with the weighted sample set. For those samples where there is mis-recognition, the weights are increased, and for others the weights are decreased. Then the classifier is trained again using the new weighted sample set, leading to different classifiers. This process repeats until saturation point is reached. Bagging, on the other hand, takes the available training samples and generates a new sample set by selecting them randomly and with replacement.

In our implementation, training examples are generated throughout the recognition process. We need to use these samples as new training datasets in subsequent iterations; as a result of which we use bagging for sampling. By generating a new set in order to train classifiers, the recognizer can be made more adaptable to a given situation at each of the succeeding iterations. Bagging is useful especially when the learning machine is unstable; in the sense that a small change in the training set yields large variations in the classification. For each session $k = 1, 2, \ldots, n$, new training set of size N is sampled from the database.

Algorithm 5 Bagging Algorithm

1: **Input:** Training sample database (D)

2: Define the probability of the j^{th} sample in the training set:

$$P(j) = \frac{1}{m} \tag{5.5}$$

3: Sample N times based on the distribution $P(j)$ from D with replacement.

4: Repeat until the learning process saturate at η

5: **Output:** New training samples D^k for the kth iteration

Bagging works as shown in Algorithm 5. Consider a training dataset of $D =$

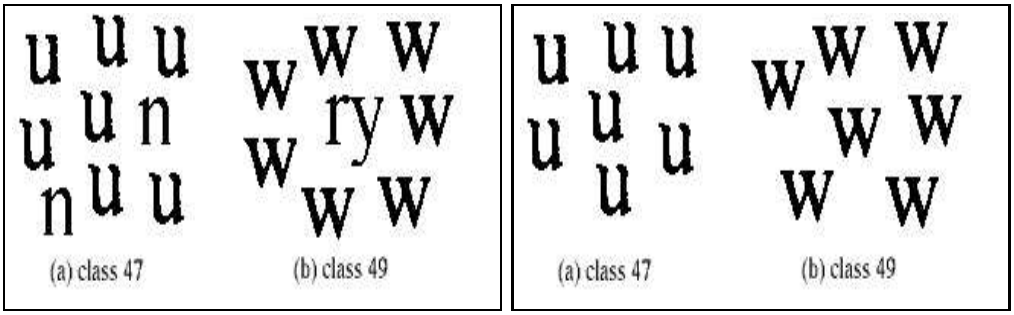
$d_1^{c_i}, d_2^{c_i}, \ldots, d_m^{c_i}$, where m is the total samples available in class c_i. Bagging selects a subset of representative samples randomly from the available training sample collections that are accumulated through feedback.

In each session k, a re-sampled training set D^k is built for constructing/training a classifier C^k. Classifier C^k has better accumulated knowledge than the previous C^{k-1} increasing its applicability to the given document collection.

Algorithm 6 Decremental clustering Algorithm

Input: Training sample, $s_1, s_2, ..., s_M$ and their respective classes $c_1, c_2, ..., c_N$
for $i = 1$ to N **do**
  for $j = 1$ to M **do**
    extract profile and moment features of samples
    define centroid of the cluster, say s_k^i
  end for
  for $j = 1$ to M **do**
    Compute similarity measure between sample images ($dis(s_k^i, S_j^i)$
    if ($dis(s_k^i, S_j^i) >$ threshold τ
      move S_j^i from the list
  end for
end for

It should be noted that before we sample examples for further training, we validate (in image space) whether each samples are in their respective classes or not. This is done because of factors such as mislabeling, segmentation error or during data transfer, there is a possibility that samples are misplaced from their proper classes. Our problem here is given training sample images, $s_1, s_2, ..., s_M$ and their respective classes $c_1, c_2, ..., c_N$, validate the samples in each class so that $s_j^i \in c_i$. This problem is addressed following decremental clustering [199] procedures (see Algorithm 6) that computes the similarity of all characters in a class and reject if there is a character that is outlier. This enables us to make sure that all member samples represent the given class. Figure 5.3 (a) shows list of samples in two of the classes before validation. The classes contains outlier samples that needs to be rectified. After we validate samples in each class, the remaining members represent the given class as shown in Figure 5.3 (b). It is observed that validation has a paramount importance to improve the quality of training samples generated in the learning process.



(a) (b)

Figure 5.3: Samples (a) before validation. (b) after validation.

5.3.3 Incremental Updation of the Classifier

The performance of the recognizers is improved when classifier model is updated with the new datasets generated during the learning process. A typical approach involves discarding the existing model and retraining the classifier with the entire training datasets (i.e. both the previous and new samples) that have been accumulated thus far. While a large number of training data can help for reducing the generalization error, the learning process itself can get computationally intractable. Hence, it is necessary to update an existing classifier in an incremental fashion so as to accommodate the new training samples (available through feedback) without compromising classification performance on old dataset.

We use SVM classifier [128] as the basic recognition unit, and employ an incremental learning approach [51] to train the Support Vector Machine (SVM) employed for the classification task. By identifying the decision boundary with maximal margin, SVM results in better generalization during classification (details of SVM is available in Section 4.5). Margin maximization leads to an optimization problem the solution of which is expressed uniquely in terms of support vectors s_i. An input pattern x is classified into class $y \in \{-1, +1\}$ according to the decision function:

$$y = sgn(\sum_i \alpha_i y_i K(s_i, x)) \tag{5.6}$$

which takes the form of a linear combination of kernels $K(s_i, x)$ weighted by training labels y_i and coefficients α_i. The coefficients α_i are nonzero only for training data that are support vectors, so that $\sum_i \alpha_i y_i K(s_i, x)$ is sparse and the support vectors capture the relevant information present in the training data. Therefore, SVMs suit well to be trained according to an incremental learning procedure.

With incremental learning techniques only the new data is to be considered at each

iteration during the learning process. The use of incremental learning is advantageous in terms of both computational time and space complexity, achieving performance results similar to the retraining approach.

Algorithm 7 Incremental algorithm

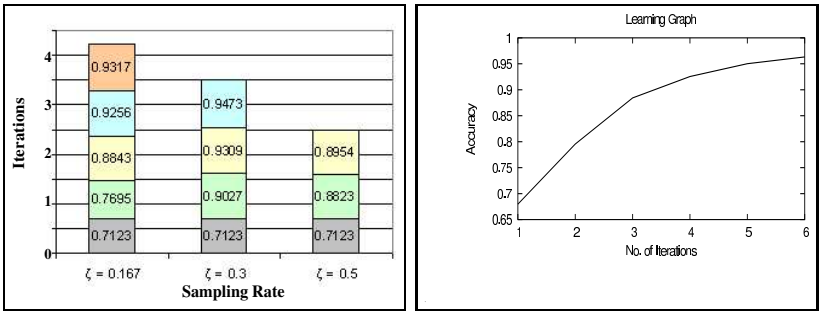
1: Remember Support Vectors (SV) from the previous training.
2: Incremental step:
3: Extract features of the new training datasets.
4: Combine SVs with the new incoming datasets
5: Train the classifier with the combined samples.
6: **Output:** New classifier model

Incremental SVM works as follows. The representation of the data seen so far for each class is given by the set of support vectors describing the learned decision boundary. These support vectors are combined with the new incoming datasets to provide the training data for the incremental step. The incremental step then updates the solution for addition of a single training sample x_i by incrementing the coefficient i and simultaneously adjusting previously assigned coefficients $\alpha_j (j < i)$ on the present and all previous training samples. The detail is given in Algorithm 7.

5.4 Implementation, Results and Discussions

Our implementation in c/c++ is tested on document image collections. The design of the recognizer is based on a multi-core approach. Keeping the futuristic large scale computational applications, it is implemented as layers with plugin interfaces for modules to ease replacing one module with another. SVM based classifier could be modified as a cascaded classifier combination with out major changes in other parts of the code. Each module internally can decide on multiple algorithm implementations of the same functionality that may be interchanged at run-time. This helps in selection and use of an appropriate algorithm or a set of parameters for a document collection or script. It allows transparent run-time addition and selection of modules thereby enabling the decoupling of the application and the plug-ins. Detail is discussed in Appendix A.

The learning scenario involves postprocessor based feedback mechanism to update



(a) (b)

Figure 5.4: (a) A change in accuracy and learning rate at different sampling rate (ζ). It shows a decrease in the number of iterations as the sampling rate increase with better accuracy at $\zeta = 0.3$ (b) An improvement in the performance of the recognizer through learning from poor quality documents.

the knowledge of the recognizer. Machine learning procedures are integrated for labeling, sampling and validating the new training samples collected online. Scanned images are segmented into words and submitted for recognition in a batch. Recognition results are post-processed to resolve ambiguity among the candidate words. Mis-recognized words are validated and fedback as new training datasets for learning incrementally. The labeler is used to split these words into components and assign their membership category (class). The sampler selects subset of these datasets and pass them for learning. Sampling speeds up the learning rate η if proper sampling rate ζ is used. As shown in Figure 5.4(a) the learning rate and improvements in the recognition accuracy are related with the sampling rate used. Initially the accuracy jumps from 71.23% to 76.95%, 90.27% and 88.23% for sampling rate 0.167, 0.3. 0.5 respectivelly. We can observe that the number of iterations decrease as the sampling rate increase, however, better accuracy of 94.73% is obtained at sampling rate 0.3 relative to others. This shows that sampling at 30% of the original training datasets speeds the learning rate with better improvements in recognition results. This is only an empirical observation. We use this sampling rate for our experimentation.

Once new training samples are selected, incremental SVM classifier is used for updating the knowledge of the recognizer. Table 5.1 presents performance evaluation of incremental learning vis-a-vis retraining. The result shows that the two approaches have comparable performance. However, the use of incremental learning has further advantage in terms of both computational time and space complexity. In a way, it eases the implementation of SVM classifier on large datasets.

We also measure computational cost of the learning scenario. We use a computer

Iterations	Retraining	Incremental
1	0.652475	0.652475
2	0.867845	0.882446
3	0.898620	0.907383
4	0.901226	0.910813
5	0.929970	0.939294
6	0.941537	0.948294
7	0.943962	0.959026
8	0.952096	0.959026
9	0.952096	0.959026
10	0.952096	0.959026

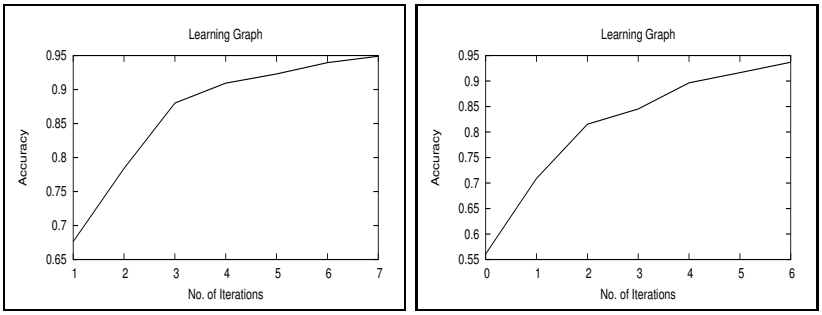
Table 5.1: Performance of incremental learning vs. retraining of the classifier during the learning process.

Iterations	Time	Accuracy
1	6.52 minutes	0.716013
2	37.25 minuets	0.893196
3	72.44 minuets	0.919695
4	110.27 minuets	0.944539
5	159.01 minuets	0.950921
6	201.39 minuets	0.961627

Table 5.2: Computational time during the learning process.

with AMD Athlon processor and 2 GHz CPU. The statistics is summarized in Table 5.2. The result shows that one session takes on the average 39.40 minutes for recognizing 17 pages, preparing and feeding back new training samples, and learning/training the classifier in an incremental fashion. It is obvious that the scheme is computationally intensive, but in today's technological advancement we can tolerate this with the aim of gaining better performance as well as saving users time in preparing labeled training samples.

We validate the performance of the learning framework on real-life printed document images (of English) that: (i) are poor in quality (with various artifacts like cuts, breaks, blobs, etc.), and (ii) varies in fonts and styles. Performance results of the learning process are depicted in Figures 5.4(b) and 5.5, respectively. The result demonstrate the advantage of our learning strategy for performance improvement.



(a) (b)

Figure 5.5: Recognition accuracy of the OCR through learning in presence of (a) font variations and (b) style variations.

Because of the quality and printing variations in document images, a very low recognition result is obtained at the initial stage, which amounts to less than 70%. Within few iterations of learning, however, the recognition accuracy improved, on the average, to 95%. Most of the recognition errors happen since characters do not always look the way they should because of degradation or printing variations. A character's hole is filled, e.g. 'o', 'u' and 'n' are recognized as 'dot', some part of the character (e.g. dot of 'i') is missing and recognized as '1' or 'l', etc. Different fonts produce a character with different shape that may visually similar with other character.

It has been observed that the recognizer is able to adopt to the given document images based on the additional samples of confusing components that are fedback for learning. In this way, the learning framework can enable the OCR to learn from its experience and adapt to varying document image collections in printing and quality.

5.5 Summary and Comments

We have presented a novel approach for learning during the recognition of document image collections. The architecture integrates advanced learning procedures (for labeling, validating, sampling and incremental learning) that automatically interacts and pass feedback for further learning. This enables the OCR to easily accumulate knowledge for performance improvement. This strategy is promising for the recognition of large digitized document images in applications, like digital libraries. Performance analysis shows that the learning framework is effective for the OCR to adapt and learn to document image collections. This is to say that the architecture for self adaptable OCR enables it to cope with the challenge from the diverse digitized printed document images that are created and archived by many applications.

We believe that our approach will open a new research direction to come up with a generic OCR systems that are applicable for digitized books and manuscripts in a large scale.

Further work is needed for (i) Extending this framework for many of the complex Indian and African scripts; (ii) Addressing the segmentation errors at various stages. The present design assumes that segmentation (of pages into words as well as words into recognizable symbols) is available. This assumption needs to be relaxed in the future.

Developing an applicable OCR system is known to be a long-term process. With the present digitization of document images at large scale, there is a pressing need for document understanding tools and methods. In response, an alternate approaches to OCR is suggested by scholars that performs document image retrieval without explicit recognition. In the next chapter we investigate the word spotting approach for content-based retrieval from document images in Indian and African languages.

Chapter 6

Searching by Word Spotting from Document Images

6.1 Introduction

Success of text image indexing and retrieval systems in digital libraries of books mainly depends on the performance of optical character recognition systems (OCRs), which convert the document images into text. For books from oriental languages of Asia (India, Thailand, Tibet, etc.), Africa (Ethiopia, Sudan, Liberia, etc.) and many other countries, there are no robust OCRs that successfully recognize printed text images of varying quality, size, style and font. Thus, the present day OCR technology does not allow adapting the solution to an easier parallel problem of textual search for many of the digital libraries content in Africa and Indian languages. Hence, we need an alternate approach for effective access to relevant documents from these large collections of text images [43]. A promising direction is to search for relevant documents using only image properties without explicit recognition; that is, to design a scheme that enables to access the document image itself without converting it to text.

We reported a preliminary study conducted in this direction [82], with some prior results on searching word images using the word spotting approach. An approach for searching and retrieval of relevant documents from document image collections is also presented [24]. It explains the framework for efficient indexing and retrieval schemes in building document image retrieval system. A conceptual block diagram of our approach is shown in Figure 6.1. The system accepts textual query from users. The textual query is first converted to image by rendering. To render a given text word we use facilities provided by popular graphics libraries, like ImageMagick by

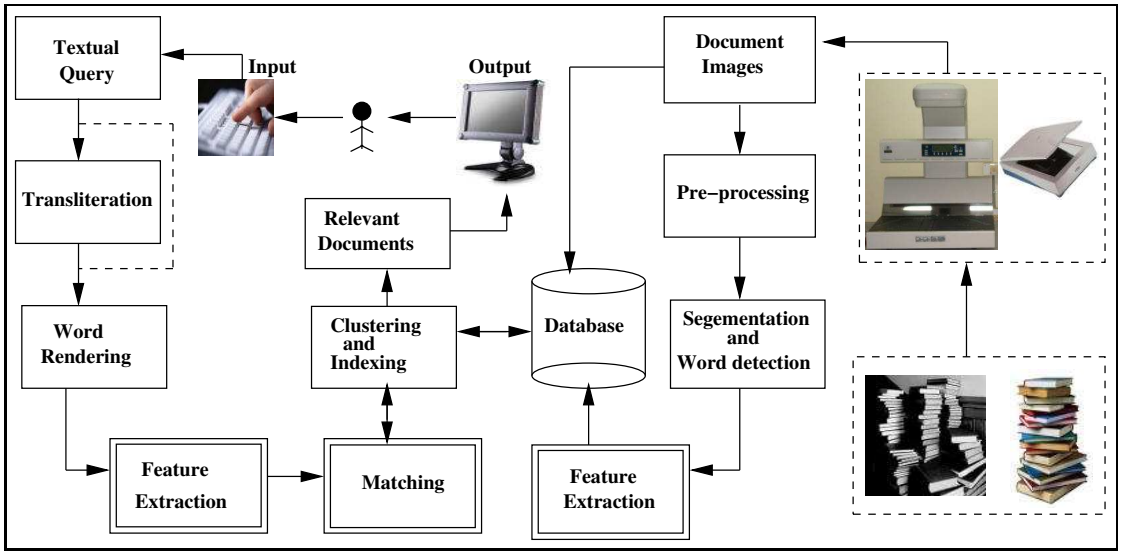


Figure 6.1: Conceptual diagram of the searching procedure. In this chapter we deal with those in double boxes.

setting font, style and point size.

Features are then extracted from this image, and finally search is carried out for retrieval of relevant documents. Results of the search are document images containing the queried word sorted in accordance to their relevance to the query word. We argue that an effective word image matching scheme is crucial for a successful search using the word spotting approach.

Hence we address here the problem of word image matching scheme, along with feature extraction scheme, for efficient content-based retrieval from printed document images. The major contributions of the chapter are the following:

- Designing an innovative matching scheme for grouping together morphological variants of a word in a given language without any explicit textual representation. We demonstrate effective use of the matching algorithm for word images, that is flexible enough to compensate for morphological variants of a word. We present the matching algorithm in Section 6.4.

- Proposing a scheme for extracting features invariant to fonts, styles, sizes and degradations. We combine a set of local features based on their discriminatory potential. The feature extraction scheme is effective for representation of word images as discussed in Section 6.5.3. We argue that good performance can be obtained with the help of the combined local features.

We undertake extensive experiments to evaluate the effectiveness of the proposed approach. Retrieval effectiveness measures such as recall, precision and F-score are used to analyze the performance of the feature extraction and matching schemes. Results are demonstrated on English, Hindi and Amharic languages. These languages are selected because of their wide usage for printing documents. In addition to English, Hindi and Amharic are the official languages of India and Ethiopia, respectively. They are the main working languages in most institutions that are spoken by more than 75% of the people in India and Ethiopia. Accordingly, there is a bulk of historical printed documents available that needs to be digitized and accessible via the Internet.

6.2 Matching and Retrieval from Document Collections

With the emergence of many digital libraries, document images are gaining popularity as an information source. Hence, access to the contents of the document image database is an important and a challenging problem in document image processing. There is an increasing focus on indexing and retrieval from poor quality printed document image collections. Word level matching and retrieval has been attempted for printed [43][114][161][181] and handwritten [154] [80] document images. These approaches are useful for locating similar occurrences to the query word. This can be done by exact matching of the query word against words in the database.

In an attempt to develop retrieval system in the image domain, much attention is given to historical printed documents, where the OCR engine fails because of degradation and unseen fonts. However, we argue that developing a search engine is more involved than considering only the effect of degradation and printing variations. There are various challenges in matching word images from printed documents. We divide them into two major groups.

- *Degradation and printing variations*: Printed documents are often poor in quality. Fig. 6.2 shows sample real-life document images, in which degradations like salt and pepper, cuts, blobs and erosion, could be observed. We need to simulate them so as to design and validate the effectiveness of the matching scheme for efficient retrieval from such document collections. Printed documents are also written using various fonts, styles and sizes. As a result of which the shape and appearance of words varies substantially. A good matching scheme will have to also take care of these variations.

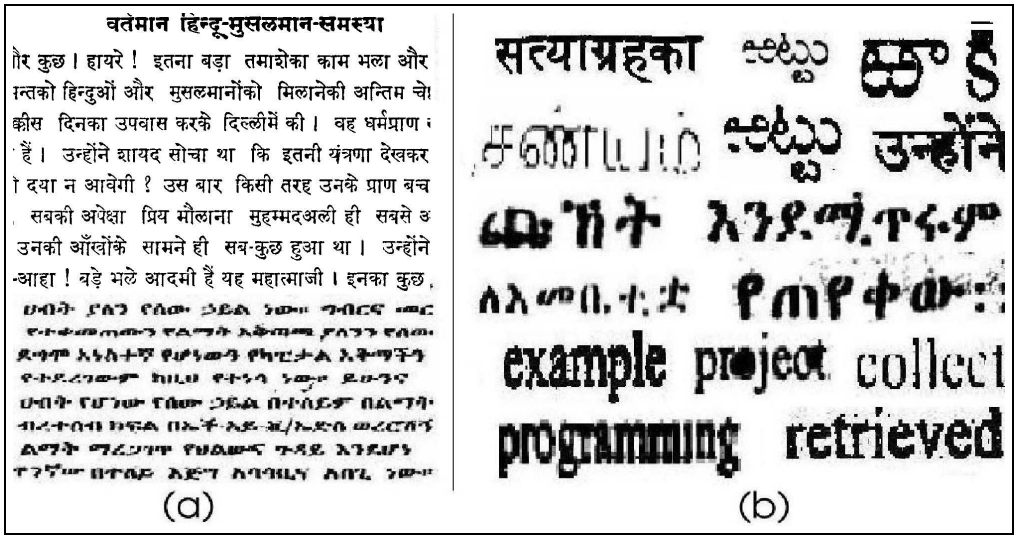


Figure 6.2: (a) Samples of real-life printed documents, (b) Some of the degraded words

- *Word morphology*: Each language has its own language rules, depending on which different morphological variants of a word are generated (see Fig. 6.3). To address this problem we need to enable existing matching algorithms perform morphological analysis in the image domain.

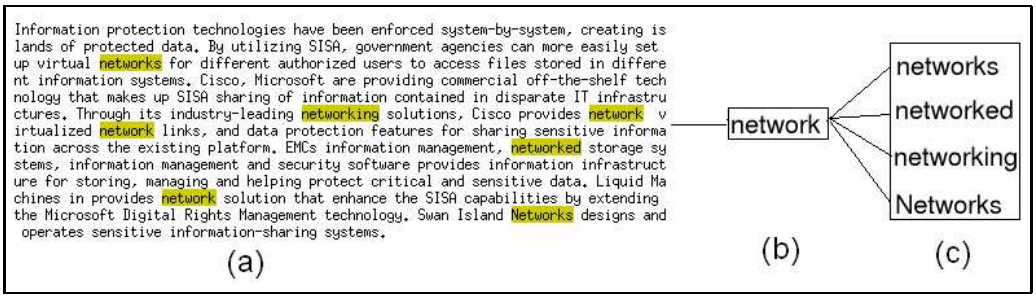


Figure 6.3: (a) Sample real-life printed document image, (b) A particular root word, and (c) Some of the word form variants in the document.

Word form variation is one of an important and challenging issue in information retrieval. The existence of word form variations in a document results in the appearance of various forms of the same word with the same meaning or conceptual representation, thus being equal from the standpoint of users' requests. As shown in Fig. 6.3, a word 'network' has many variants. When users query for a document using this word they should retrieve documents that contain 'networks', 'networked',

'networking', etc. Users expect document image retrieval system to provide the same convenience as it has been done effectively by text search engines. Most existing matching schemes fail to perform well in this situation. We argue that the ability to deal with these issues is crucial for effective matching scheme. We design a matching scheme that is flexible enough to control morphological variants of a word image.

6.3 Challenges for Retrieval from Printed Document Images

Most printed books, manuscripts, newspapers, etc. that are archived in digital libraries are of poor in quality, vary in printing and have word form variants. Indexing and retrieval of image documents is a great challenge in this situation.

6.3.1 Modeling Degradation in Printed Document

There are a number of artifacts that affects the quality of printed document images, including blobs, cuts, merges, etc. These artifacts can be observed from Figure 6.2. Degradation in printed text mostly occur due to the poor quality of paper and ink drops caused by foreign material like printers, photocopy or fax machines. The existence of large ink-blobs in the text joins disjoint characters or components. There may be cuts due to folding of the paper, paper quality or foreign material. The occurrence of cuts discontinue character continuity such that one character may be split into two or more characters.

Modeling document degradations enables us to closely examine the performance of matching scheme. We consider four categories of degradations that are commonly observed in printed documents. In this work we model salt and pepper, cuts and blobs, in addition to erosion of boundary pixels simulated by Zheng and Kanungo [203].

1. Salt and pepper noise is distributed all over the image flipping white pixels to black if it is a background and black pixels to white if it is a foreground. It is modeled with the help of parameters such as $P^s(b)$ (the probability of occurrence of black pixel in the image) and $P^s(w)$ (the probability of occurrence of white pixel in the image).

2. Cuts noise corrupt some part of an image by flipping black pixels into white. The occurrence of cuts in a document image breaks continuity of the shape of characters (or components). It happens due to print font quality, folding of

Figure 6.4: Sample Hindi, Amharic, Telugu and English word images degraded with Cuts, Blobs, Salt and Pepper, and Erosion of edge pixels (as shown from top to bottom row-wise).

paper, etc. To model this noise we compute the probability of occurrence of white pixels $P^c(w)$ in a specified size s.

3. Blobs affect some part of the image by flipping pixel values to black. Blobs occur due to large ink drops within the document image during printing, faxing and photocopying. The existence of these noises merge separate characters or components. In modeling blob noise, we use parameters, $P^b(b)$, probability of flipping white pixels to black pixels for the specified size s.

4. Erosion of pixels happens mostly at the boundary of the image due to the imperfections in scanning. This degradation is modeled with the fact that black and white pixels are swapped according to some probabilities directly related to the distance from the boundary pixel. Thus, the level of erosion of pixels depends on the probability of flipping foreground pixel and background pixel.

More detailed discussion of these degradation models are available in Appendix B.

Using these degradation models we generate datasets of word images in Amharic, Hindi and English languages. Sample words are shown in Fig. 6.4. By varying parameter values of the degradation models, we generate datasets that mimic various levels of degradations in real-life, scanned document images. In Section 6.5, we use these datasets to systematically analyze the performance of feature extraction and word image matching schemes.

Printing Variations: For printing in a specific script, there are different fonts, sizes and styles available for use. Hindi, Amharic and English languages use a number

of fonts, each of them offering several stylistic variants and font sizes (see Section 3.6 for the detail).

These fonts, styles and sizes produce texts that greatly vary in their appearances (i.e. in size, shape, quality, etc.) in printed documents. We need to select suitable representational scheme that are invariant to printing variations.

6.3.2 Word Morphology

As discussed in Section 3.5, each language has its own language rules for formation of words from meaningful stems. Based on these rules words with the same meaning can be constructed in different ways. This is one of the main challenges in searching for relevant document using query words since the exact word that is present in the document being searched is a variant of the query word.

As can be observed from language rules, there are two possible variants of a word: (i) word form variation (word variants that are formed, for instance by adding prefix and/or suffix to the root word), and (ii) synonymous words (words with the same meaning). For instance, for the query word 'direct', word-form variations include 'directs', 'directing', 'redirected', etc., where as, synonymous words include 'lead', 'organize', 'guide', etc. depending on the context. Addressing the second type of variation needs knowledge of the contextual meaning of the word and is beyond the scope of the present work. Here we address the first problem, namely word-form variations.

In the text domain, most of the variations of word forms obey the language rules. A range of computational techniques (like word counting, automatic hyphenation, etc.) are used for analyzing and representing written text in a language at one or more levels of linguistic analysis (such as morphological, syntactic, semantic, etc.). Syntactic analysis is designed to group words based on the grammatical rules that govern the structure of the sentence. Semantic analysis, on the other hand, is the study of meaning of the sentence that adds contextual knowledge to the purely syntactic process.

Morphological analysis deals with the processing of individual word forms to identify the recognizable portion of words (i.e. word stem) by removing word suffixes and prefixes. It identifies word-stem from a full word-form with the assumption that word variants have similar semantic interpretations and can be considered as equivalent for the purpose of information retrieval. In the text-domain, a number of stemming algorithms have been developed based on language information to reduce a word to its stem or root form. Text-based search engines use existing stemming algorithms to

identify word form variants. However, for document-image retrieval, language information is not directly usable. and, hence there are no known stemming algorithms exist for identifying variants of word images. We need to design a matching scheme that mimics textual morphological analysis.

6.4 Morphological Matching in the Image Domain

Appearance of word variants in the document are unavoidable as they are constructed based on language rules. We need to map morphological analysis used in the text domain in order to perform morphological matching of word images. There are a number of factors to be considered during matching. Some of these are: what are the features used for matching, how the similarity between different word images are measured, and how is the optimal match computed.

6.4.1 Feature Extraction Scheme

Feature extraction is the problem of gathering information from raw data, which is most relevant for a given application. Many different features have been employed in image processing and pattern recognition for representation of document images. Most of them were proposed for the recognition task. A comprehensive review of the various feature extraction methods for an OCR system is found in Trier *et al.* [182]. The required characteristics for the selection of features are invariant to transformations and degradation effects [65][182]. Features should also be distinct with respect to their neighborhood, stable with respect to noise, and complementary with respect to other features [61].

We experiment here with three categories of features. These features include word profiles, moments and transform domain representations. Profiles of the words provide a coarse way of representing word images for matching. Projection, transition, upper and lower profiles are features used here for the representation of word images. While projection (F_1) and transition (F_2) profiles capture the distribution of ink along one of the two dimensions in a word image, upper (F_3) and lower (F_4) word profiles capture part of the outlining shape of a word. These features were effectively employed for matching handwritten words [155] . To measure the distance between the upper and lower boundary of the word image, vertical distance (F_5) is used. This is done by recording the appearance of the first ink pixel (on top boundary) and last ink pixel (on bottom boundary), and by finding their relative difference.

Moment-based features are computed to analyze the shape of a word image, where

each order of moment has different information for the same image. Here we employ statistical moments (such as mean (F_6), standard deviation (F_7) and skew (F_8)) and region-based moments (such as the zeroth-order M_{00} (F_9) and first-order M_{01} (F_{10}) moments). Statistical moments measure the central tendency and distribution of ink pixels in word images. Region-based moments represent the entire shape region by describing the considered region using its internal characteristics, i.e., the pixels contained in that region. The knowledge of these moments can be used to calculate central moments, like cm_{02} (F_{11}) that computes squared distance from the y component of the mean.

A compact representation of a series of observations (such as profiles) is possible in a transform domain using fewer coefficients. In this work we use Discrete Fourier descriptors (F_{12}) for word shape representation. We compute N Fourier descriptors following the vertical strips of the word image, in a one dimensional space, $G(i), i = 0, 1, \ldots, N-1$, where N is width of the word.

A detailed discussion of these features is given in Appendix C. We will analyze in subsequent sections their tolerance/insensitivity to variations in scripts, fonts, styles and sizes as well as to degradations that are seen in printed document images.

6.4.2 Matching by DTW

Good matching performance can be achieved by a technique that aligns and finds the best match between pairs of queried and referenced word images. We test here cross correlation and dynamic time warping matching algorithms. These matching algorithms have a great advantage in aligning and comparing word images with different sizes. Rather than comparing sequences of feature vectors one-to-one that might not correspond well (as might be done by Euclidean distance measure, for instance), they search and recover the optimal space of mappings between two sequences of signals. As a result, they suit to implement feature vectors extracted following vertical strips of word images for similarity measure.

Cross Correlation

Cross correlation is a statistical measure of how closely two signals are related. It has a wider applications in area-based matching that often employ correlation scores as the similarity measure between pixels or regions of the image. The correlation function compares sequences of feature vectors of two word images by aligning them against each other, and at each position the correlation coefficient between the corresponding pairs of feature vectors of the two word images is computed. Optimal matching score

of the two aligned images are obtained from their alignment vector. This gives us a dissimilarity measure of the match. Thus, for decision purpose, the maximum of the resulting correlation among a set of referenced and template word images defines the best match.

Dynamic Time Warping

Dynamic time warping (DTW) is a dynamic programming based procedure used to align sets of sequences of feature vectors and compute matching score for finding the dissimilarity of words. Dynamic time warping is a popular matching procedure in speech analysis and recognition [163], handwritten documents retrieval [80][155], data mining [30][91], etc. It is often used in speech recognition, for instance to determine if two waveforms represent the same spoken phrase. In a speech waveform, the duration of each spoken sound and the interval between sounds are permitted to vary, but the overall speech waveforms must be similar.

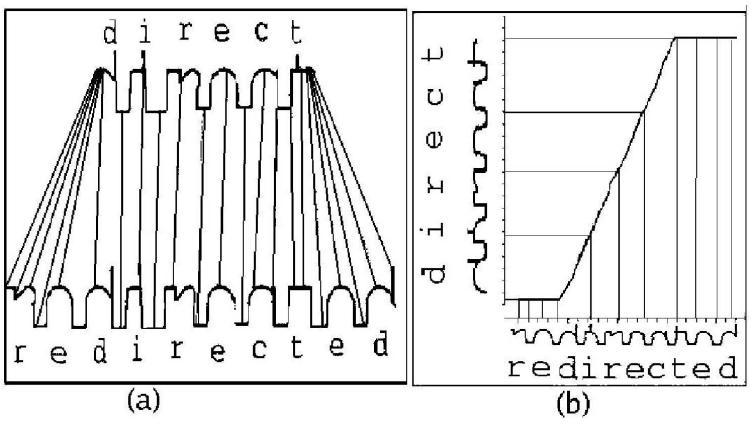


Figure 6.5: Plots demonstrating the dynamic time warping procedure for matching words using DTW; (a) Alignment of upper profiles of two English words, (b) Profile of the two words and the optimal warping path.

The cleverness of both cross correlation and dynamic time warping lies in the computation of the distance between the two sequences of signals. They are powerful in aligning word size variations. The alignment is also optimal in the sense that DTW minimizes (while cross correlation maximizes) a cumulative distance measure consisting of local distances between aligned sequence of points. This is mostly a non-linear mapping where input sequences can be of different lengths; for example, we may find that time $\mathbf{t_1}$ in the input signal corresponds to the time $\mathbf{t_{1+4}}$ in the matched signal as shown in Figures 6.5 (a), 6.7 (a), and 6.6 (a).

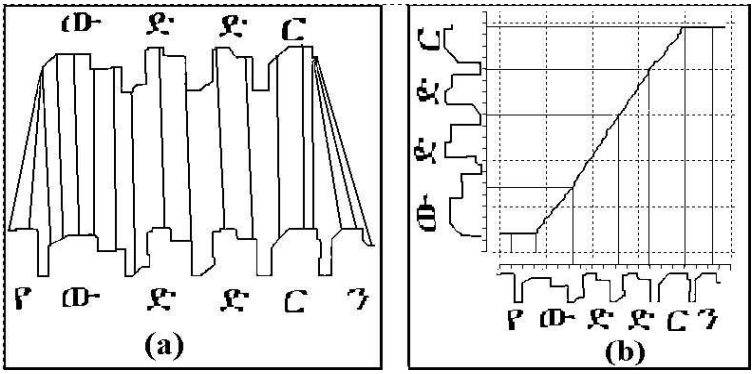


Figure 6.6: Plots demonstrating the dynamic time warping procedure for matching Amharic words using DTW; (a) Alignment of upper profiles of two words, (b) The optimal warping path.

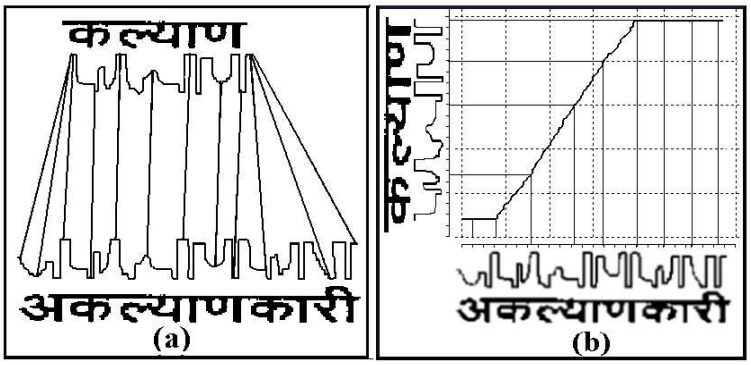


Figure 6.7: Plots demonstrating the dynamic time warping procedure for matching words using DTW; (a) Alignment of lower profiles of two Hindi words, (b) Profiles of the two words and the optimal matching path.

Let two word images (say their moment) be represented as a sequence of features $\mathcal{G} = \mathbf{G_1}, \mathbf{G_2}, \ldots, \mathbf{G_N}$ and $\mathcal{H} = \mathbf{H_1}, \mathbf{H_2}, \ldots \mathbf{H_M}$. The DTW-cost between these two sequences is $D(M, N)$, where M and N are the lengths of the two sequences, which is calculated using the recurrence equation:

$$D(i,j) = \min \begin{cases} D(i-1, j-1) \\ D(i, j-1) \\ D(i-1.j) \end{cases} + d(i,j) \tag{6.1}$$

where, $d(i, j)$ is the cost in aligning the ith element of $\mathcal{G}$ with the jth element of $\mathcal{H}$. Squared Euclidean distance is used to compute $d(i, j)$.

$$d(i,j) = \sum (G_i - H_j)^2 \tag{6.2}$$

The use of $D(i, j-1)$, $D(i-1, j)$ and $D(i-1, j-1)$ in the calculation of $D(i, j)$ realizes a local continuity constraint, which ensures no samples left out in time warping so that it can handle local distortions in word images and is not restricted to a single global transform.

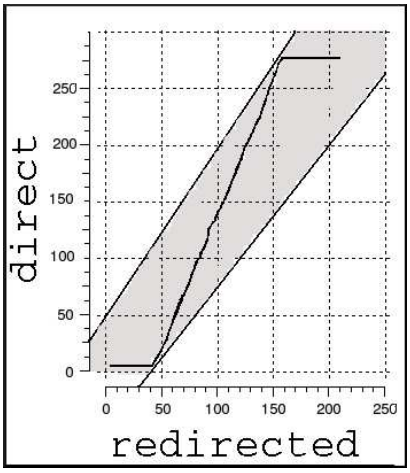


Figure 6.8: DTW matching using Sakoe - Chiba band.

We also imposed global constraint using Sakoe - Chiba band [163] (see Figure 6.8) so as to ensure the maximum steepness or flatness of the DTW path. $D(M, N)$ is taken as the final matching score to decide the similarity between the two sequences, $\mathcal{G}$ and $\mathcal{H}$. The optimal warping path (OWP) is the cheapest one with minimum distance, among all available paths in the DTW matching space, that runs from $D(0, 0)$ to $D(M, N)$ with length L. It is defined as:

$$OWP(G, H) = \arg\min_{i} \ cost(W_i) \tag{6.3}$$

where W_i is a specific warping path of length L in the matching space. The matching cost, $D(M, N)$ is normalized by the length of the optimal warping path to reduce the effect of word size variations during alignment.

6.4.3 Morphological Matching for Detecting Word Forms

Most existing matching algorithms might be efficient for searching an exact match to a query word. However the retrieval process for a good search engine is more involved than the simple word-level matching. Due to language variability, a word, with similar meanings, usually appears in various forms. This requires a method to conflate such different morphological variants of a word to a common stem/root. In

Algorithm 8 Morphological Matching

1: Partition the warping path into three regions of length k ($L/3$) each: beginning, middle and end.
2: **for** $i = 1$ to k **do**
3: if there is matching cost concentration at the beginning.
4: reduce extra cost from the total matching score.
5: else break.
6: **end for**
7: **for** $i = L$ down to $2k$ **do**
8: if there is matching cost concentration in the end.
9: reduce extra cost from the total matching score.
10: else break.
11: **end for**
12: Normalize the matching score by the length of the optimal warping path.

the text domain, variation of word forms may obey the language rules. Text search engines use this information to identify and group word form variants. However, for document-image retrieval, this information is not directly usable.

We propose DTW-based partial matching technique that takes care of word form variations in the beginning and at the end. We follow Algorithm 8 for grouping word form variations. A close observation of the OWP shows that for word variants the DTW path deviates from the diagonal either in the horizontal or vertical direction from the beginning or end of the path. Note that, as the DTW path deviates from the diagonal line, the matching cost increases tremendously. For example, Fig. 6.5 (b) presents the matching scenario of the root word ('direct') with its variant ('redirected'), in which the optimal path deviates from the diagonal line at the two extreme ends. This is because of the extra characters 're' and 'ed' in the word 'redirected'. Profiles of these extra characters have no or minimal contribution for measuring similarity of the two words and, hence their cost needs to be cut from the total matching score.

To this end, we analyze whether the dissimilarity between words is concentrated at the end, in the beginning or both. This involves partitioning the optimal warping path into three regions (beginning, middle and end) and checking the matching score at the two extremes (beginning and end). In Fig. 6.5 (b) it is observed that characters 're' in the beginning and 'ed' at the end of the word 'redirected' are get matched with characters 'd' and 't' of the word 'direct'. Due to this, there is an alignment concentration at the beginning and end (see Fig. 6.5 (a)), such that local matching

cost is very high relative to other points. This increases the total matching score of the two words. We ignore the additional cost at the two extremes, reducing the total matching score. For instance, for a query word "direct", the matching scores of the referenced words "indirect" and "redirected" are only the matching of the 6 characters, 'd-i-r-e-c-t', of both words.

Once an optimal sub-path is identified, a normalized cost corresponding to this segment is considered as the matching score for the pair of words. With the reduction of dissimilarity cost of a variant word (against its root word), we found that a large set of words get grouped together. This increases recall of the system during searching and retrieval.

The time complexity of the algorithm is $\theta(n)$, where n is the length of OWP of two words. This happens when variants of a given word are encountered, which occurs rarely at a rate less than 0.01% of words in the collection. In more than 99% of the case no variants of a word exist and, hence the algorithm runs in constant time, which makes it practically applicable for searching in document images.

6.5 Results and Discussion

In this section we present an extensive experimental results to evaluate the performance of the proposed matching scheme.

6.5.1 Datasets and Performance Measure

To undertake extensive experiments, we prepare ground-truthed word-level datasets in English, Hindi and Amharic languages.

The datasets used for the experiment contains a set of words that are: (i) composed of basic words and their morphological variants, (ii) printed using the various fonts, styles and sizes, and (iii) generated using degradation models for salt and pepper, cuts, blobs and erosion of pixels. The datasets are organized language-wise. From each language more than 4000 basic word images are collected each of which have on the average five variants. These words are a comprehensive mix of the various words used for writing, with detailed inclusion of their word forms (that are formed by adding suffix and prefix). These words are degraded using degradation models and also duplicated using the specified fonts, sizes and styles in the language. With this we have a total of more than $800,000$ word images for the three languages. We use these datasets to evaluate the performance of feature extraction and matching schemes proposed in the present work.

Performance is measured in terms of precision (P) and recall (R). While recall is the percentage of relevant words that are matched and retrieved from the collection, precision is the percentage of matched and retrieved words that are relevant. They are computed as follows:

$$R = \frac{rm}{rd} \tag{6.4}$$

$$P = \frac{rm}{tm} \tag{6.5}$$

where rm is relevant words matched and retrieved, rd is relevant words in the database, and tm is total matched and retrieved words.

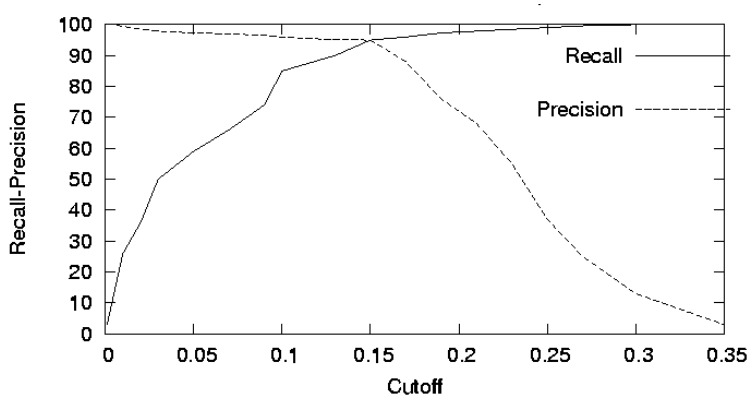


Figure 6.9: Effect of cutoff point on recall and precision. Our interest is to arrive at a cutoff point τ where recall and precision cross.

Fig. 6.9 shows the effect of cutoff τ on recall and precision. The best possible cutoff point where recall and precision cross is set automatically by linear interpolation from recall and precision curves,

$$\begin{cases} y_r = ax_r + b \\ y_p = ax_p + b \end{cases} \tag{6.6}$$

where a and b are slope and y-intercepts, respectively.

Recall and precision are basically inversely related; an increase in recall is at the expense of precision and vice versa. In order to select features that balance recall and precision, we compute a single score using F-measure (F). Computing F-score provides the best possible compromise between recall and precision. Determination of the maximum value for F means finding the best possible compromise between recall and precision. F-measure (F) is a weighted harmonic mean of recall (R) and precision (P), which is expressed as:

$$F = \frac{(2PR)}{(P+R)} \tag{6.7}$$

This measure rewards schemes that keep recall and precision close together.

Local Vs. Global Features: There are a number of features available for representation of a given object. These features are either local or global features. Local features represent a distinguishable small area of a region. Curvature, boundary segments and corners are some of the commonly used local features in which a sequence of features are extracted by dividing the object into sub parts. Where as, global features consider all points to describe some characteristics of regions of the object such as area (size), perimeters, etc.; as a result, one value is computed for representation.

We evaluate both global and local features extracted from word images using the feature extraction methods discussed in section 6.4.1. We extract local features following the vertical strips of word images that forms a feature vector, $\mathcal{G} = G_1, G_2, \ldots, G_N$, of length N (i.e. width of a word). To compare the performance of global and local features matching score is computed with the help of DTW algorithm. Results show that the average score obtained using global features is 53.32% recall, 50.24% precision and 51.73% F-score. In comparison, local features register 82.92% recall, 80.53% precision and 81.71% F-score. From their performance one can observe that local features have better representational capability than global features.

If only global features are used to represent a word image, the local features of the word are mixed together. This results in mis-grouping of different word images, which share similar global features, while having different local features and contextual information. All the analysis given in the succeeding sections are limited to the local features.

Matching Schemes: Using local features of word images, we evaluate cross correlation and DTW matching algorithms. The average performance of the two algorithms shows that DTW registers 89.58% recall, 90.81% precision and 90.19% F-score, where as cross correlation achieves 76.43% recall, 78.83% precision and 77.61% F-score. The F-score reflects that DTW out-performs cross correlation, which is also seen in both recall and precision. This is because cross correlation is relatively sensitive to small changes in the keyword being matched with referenced words though they are the same. DTW, on the other hand, compensates for most of the variations observed in printed word images during the alignment process. We therefore propose a modified DTW matching technique for computing similarity of word images in printed documents.

6.5.2 Feature Selection for Matching Word Images

Feature selection enables to explore the usefulness of selecting subsets of features that together have good predictive power. An individual feature may not give us higher performance. However, it can provide a significant improvement of performance when combined systematically with other features. This requires a method to combine them and evaluate their performance on a given datasets. Feature selection is therefore, a strategy employed to identify best features (from a set of features) for an application that improve retrieval effectiveness. Because of its computational efficiency, we use forward feature selection approach for combining well-performing features. This approach begins with the evaluation of all features with exactly one feature at a time, and selects the one with the best performance. It then evaluates again by combining the selected feature with the remaining features and again selects combined features with the best performance. The selection process repeats until no improvement is obtained from extending the combined run. Detail discussions of feature selection procedure is presented in Appendix E.

We undertake experiment to assess the performance of individual feature's as well as their combined one. We use local features to represent word images. Given n features, a modified DTW algorithm is used to combine these features and compute their matching score, as follows.

$$D(i,j) = \min \begin{cases} D(i-1,j-1) \\ D(i,j-1) \\ D(i-1.j) \end{cases} + \sum_{k=1}^{n} d^k(i,j) \tag{6.8}$$

where $d^k(i,j)$ is the cost in aligning the i^{th} and j^{th} elements of feature k extracted from two words using squared euclidean distance.

$$d(i,j) = \sum_{k} \left(G_i^k - H_j^k\right)^2 \tag{6.9}$$

The DTW matching score is used to decide the similarity between words, based on which recall and precision is computed.

Table 6.1 shows performance of individual features, in which, on the average, vertical distance registers the highest F-score of 79.76%. Projection profile and mean comes next with 79.09% each. Their relative good performance is attributed to the discriminatory information contained by them.

Among the combined runs, highest F-score is obtained by the following features (see Appendix E for detail result). On English datasets, transition profile, lower profile, moment M_{00} and standard deviation register 90.05% F-score. On Hindi, transition

Features	Recall	Precision	F-Score
F_1	82.58	79.25	79.09
F_2	74.10	51.97	55.97
F_3	62.16	61.94	60.31
F_4	74.20	74.63	72.65
F_5	82.08	80.86	79.76
F_6	82.57	79.24	79.09
F_7	56.82	54.90	53.11
F_8	39.77	50.77	43.07
F_9	79.68	70.47	72.40
F_{10}	81.36	62.13	66.71
F_{11}	44.01	20.75	27.05
F_{12}	61.19	68.70	63.68

Table 6.1: Average performance of individual features for English, Hindi and Amharic languages. DTW is used for matching local feature vectors extracted from vertical strips of word images.

profile, lower profile, moment M_{00} and vertical distance result in 91.79% F-score, and, on Amharic transition profile, lower profile, moment M_{00} and upper profile register 92.24% F-score. For all languages combining transition profile, moment M_{00} and lower profile performs well. Each of them when combined provides new information required to identify a given word uniquely. The difference lies on the fourth combined feature. For English standard deviation provides new data to control spatial distribution and spread of ascenders and descenders of the word. In Hindi words there is a headline *shirorekha*. By measuring the distance from this line to lower boundary, vertical distance supplies new information for better representation. Combining upper profile with lower profile is important to capture the outlining shape of the word both from the top and bottom so as to separate similar looking Amharic characters in a word.

Thus, in general, more than 90% F-score is obtained by combined features, which shows their high representational power than individual features. In the next section we evaluate appropriateness of the combined features for degraded word images and printing variations.

Information Content of Features: Features should distinctly identify one object from another in a feature space. We analyze how features represent words in a feature space using Kullback Leibler distance (KL-distance). KL-distance is a standard mea-

Feature 1	Feature 2	KL-score
F_1	F_2	0.524786
F_1	F_3	0.778462
F_1	F_4	0.578985
F_1	F_5	0.792096
F_1	F_6	0.678430
F_1	F_7	0.678430
F_1	F_8	0.898420
F_1	F_9	0.863610
F_1	F_{10}	0.884166
F_1	F_{11}	0.666631
F_1	F_{12}	0.655436

Table 6.2: Sample comparison of information content of features using KL distance. The result shows the KL-score of projection profile (feature 1) against all the remaining features (Feature 2).

sure of the information content of each feature [133]. Given two features extracted from the same word image W, $f_1(w)$ and $f_2(w)$, KL-distance (or relative entropy) is a measure of the distance between these features [54]. KL-distance (D_{KL}) is defined as follows:

$$D_{KL}(f_1(w), f_2(w)) = \sum_w f_1(w) ln \frac{f_1(w)}{f_2(w)} \tag{6.10}$$

This provides information as to how these features distinctly identify a given word image. Thus, although it is not strictly a distance metric, it is a non-negative, continuous function that is zero if and only if the two features are identical.

$$D_{KL}(f_1(w), f_2(w)) = \begin{cases} 0 & if\ f_1(w) = f_2(w) \\ > 0 & Otherwise \end{cases} \tag{6.11}$$

For instance, from a word image 'abacus', we extract its features using the above feature extraction methods and compare these features with each other. The same is done for twenty word samples taken from the datasets for computing average KL-score. We present in Table 6.2 sample KL-scores, that compares projection profile against the remaining features. For all cases the KL-score is high. According to this method, those features with minimum distance (scores approaching 0) are similar. As show in Table 6.2, the minimum score is 0.524786. This means that the features we use are distinct in the feature space.

Degradation	English			Hindi			Amharic		
	Recall	Prec.	F-score	Recall	Prec	F-score	Recall	Prec	F-score
Cuts	93.76	88.15	90.87	92.34	92.41	92.37	93.72	94.93	94.32
Salt & pepper	96.56	96.02	96.29	93.28	93.17	93.20	96.88	97.11	96.99
Blobs	89.79	86.43	88.08	85.95	92.33	89.03	89.46	93.48	91.43
Erosion	92.38	93.29	92.83	92.77	92.58	92.67	94.91	95.72	95.31

Table 6.3: Test result on degraded word images of the three languages: English, Hindi and Amharic. N.B. Prec. = Precision.

6.5.3 Relevance of Combined Features

Once we collect datasets of word images using our degradation models, they are represented using the combined local features proposed in the previous section for experimentation. Table 6.3 presents performance of the feature extraction scheme on degraded text images of the three languages (English, Hindi and Amharic). Test result shows that performance varies depending on the degradation type. On the average 92.52%, 95.49%, 89.51% and 93.61% F-scores are obtained on cuts, salt and pepper, blobs and erosion, respectively. Minimum result is registered for blobs; because word images degraded by blobs are relatively confusing to identify as this noise adds more ink pixels that changes the visual meaning of a word image. The result registered language-wise indicates that, on the average, 92.02%, 91.82% and 94.51% F-scores are obtained for English, Hindi and Amharic, respectively. From the result we can observe the following. First, high performance is registered on degraded word images. Second, results are comparable across the three languages. This means that the proposed feature extraction scheme is insensitive to degradations, even with a change in script.

Font, size and style difference are common in printed documents. We also evaluate the performance of combined features from this perspective. We select the most commonly used fonts, sizes and styles in printing documents in the three languages. Word images are then generated using these printing variations. These words are used to test the performance of the combined features suggested by the feature extraction scheme proposed in this study.

As presented in Table 6.4, average performance of the proposed scheme on the various fonts, styles and sizes is 93.06%, 96.50% and 85.31% F-scores, respectively. The result for styles is reduced because of the complexities of words produced by italics typeface. Since characters generated by italics are skewed, there is a need to introduce skew detection and correction modules before matching words written in italic. By applying image processing better accuracy can be obtained for italics too.

Item	English				Amharic				Hindi			
	Ty	Rec	Pre	Fsc	Ty	Rec	Pre	Fsc	Ty	Rec	Pre	Fsc
Font	Ari	96.43	97.81	97.12	PG	97.03	97.28	97.15	Yo	95.92	97.46	96.68
	Tim	96.87	97.63	97.25	VG	92.19	90.31	91.24	Ga	81.09	85.27	83.13
	Cou	93.15	95.11	94.12	AL	93.97	96.76	95.34	Na	92.84	95.39	94.10
	San	81.45	91.07	85.99	FE	88.91	92.18	90.52	Su	93.89	94.21	94.05
Style	Nor	96.94	98.13	97.53	Nor	96.17	98.01	97.08	Nor	96.41	96.79	96.60
	Bld	96.42	94.85	95.63	Bld	95.72	97.52	96.61	Bld	94.24	94.78	94.51
	Ita	56.37	61.08	58.63	Ita	73.91	72.68	73.29	Ita	57.03	58.76	57.88
Size	10	96.31	96.98	96.64	10	96.52	97.04	96.78	10	93.29	96.26	94.75
	12	96.83	98.26	97.54	12	96.97	98.11	97.54	12	96.18	96.84	96.51
	14	96.54	97.16	96.85	14	95.24	97.67	96.44	14	94.42	96.58	95.49

Table 6.4: Test results on datasets of the various fonts, sizes and styles of English, Amharic and Hindi languages. Ty = Type, Rec = Recall, Pre = Precision, Fsc = F-Score, Ari = Ariel, Cou = Courier, Tim = Times, San = SanSerif, Bld = Bold, Ita = Italics, Yo = Yogesh, Ga = Ganesh, Na = Natraj, Su = Surakh.

In general, on the average 91.62% F-score is obtained. This shows that the combined local features suggested by the feature extraction scheme are invariant against printing variations (fonts, styles and sizes).

6.5.4 Effectiveness of Morphological Matching

As a solution to the problem of language variability we propose a partial matching scheme. We conduct experiment in two steps to evaluate its performance. First, the matching process is done skipping the partial matching module. We compute recall, precision and F-score for this. Then, the test is repeated incorporating the partial matching procedure. Table 6.5 presents comparison of the test result before and after applying the partial matching scheme. Figure 6.5 shows some of the word variants that are grouped into one set for a given word in English, Amharic and Hindi. The result shows that an average improvement of 9.87% F-score is registered. This indicates that the proposed partial matching algorithm is able to greatly enhance the performance of the system. As can be observed from Table 6.5, the algorithm registers in the nineties for most of the cases, except for italic font style. This is because of the complexity of the word shape created by this style for representation.

Precision vs. recall graph is shown in Fig. 6.10. The effectiveness of our scheme is observed from the graph as it increases both precision and recall by moving the entire curve up and out to the right. This shows a significant contribution of the

Item	Type	Before			After		
		Recall	Precision	F-score	Recall	Precision	F-score
Font	Arial	90.76	93.14	91.93	97.29	99.62	98.44
	Courier	90.38	93.39	91.86	96.17	99.01	97.57
	SansSerif	62.17	89.54	73.39	90.56	96.28	93.33
	Times	91.43	91.06	91.24	96.93	99.36	98.13
Size	10	89.46	91.79	90.61	97.57	99.09	.98.32
	12	90.76	93.14	91.93	97.29	99.62	98.44
	14	81.22	88.93	84.90	95.65	98.46	97.03
Style	Normal	90.76	93.14	91.93	97.29	99.62	98.44
	Bold	81.35	87.93	84.51	96.81	99.58	98.18
	Italic	56.23	57.05	56.64	75.16	83.78	79.24
Degradation	Cuts	83.64	86.69	85.14	91.27	96.06	93.60
	Blobs	74.18	83.36	78.50	88.94	95.76	92.22
	Salt & Pepper	93.67	92.73	93.20	92.40	99.38	95.76
	Erosion	90.53	90.82	90.67	94.33	96.84	95.57

Table 6.5: Performance of partial matching technique. Test results are shown both before and after incorporating the partial matching module.

morphological matching procedure in grouping together (see Figure 6.11) the various variants of a word which have a great impact on the overall performance of the retrieval system.

The proposed word image matching scheme works well on diverse datasets. It registers more than 90 percent F-score in most cases. This guarantees the advantage of the morphological matcher to realize an effective search and retrieval system for printed document images that have many functionality.

The success of this work greatly depends on the performance of the page segmentation algorithm and efficiency of the indexing scheme we are going to design. The matching scheme also needs to be more general so that it also addresses synonymous word variants that are seen in real life documents.

6.6 Summary and Comments

Developing an efficient searching system in the image domain requires designing an effective feature extraction and word image matching schemes. We achieved this by designing a novel DTW-based matching algorithm for appropriate feature selection and morphological matching in the image domain. By organizing datasets that encompass the various scripts, fonts, styles, sizes, degradations and word forms, we

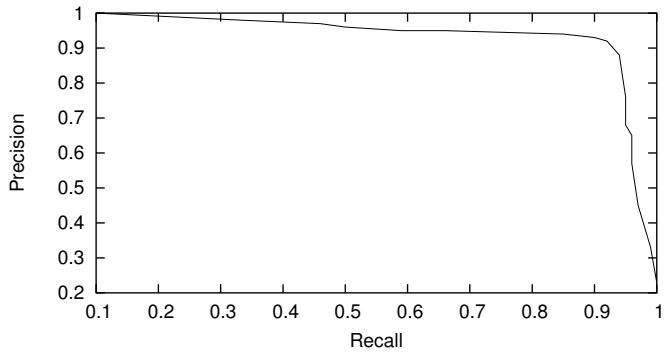


Figure 6.10: Precision-recall graph for the proposed matching scheme

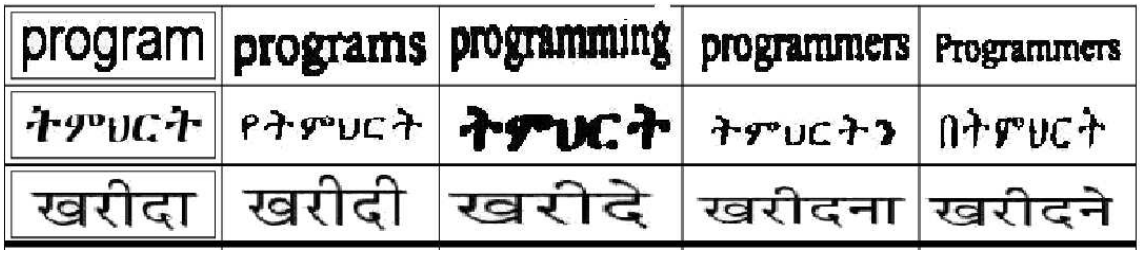


Figure 6.11: Sample grouped word images from English, Amharic and Hindu languages (from top to bottom corresponding to the word in double box)

experiment the efficacy of the proposed approach. We employ degradation models to simulate salt and pepper, cuts, blobs and erosion of pixels in real-life documents. We use this as a basis to select appropriate representational scheme and matching algorithm for searching and retrieval of relevant documents in the image domain.

We investigate a number of features composed of profiles, moments and transform domain representations of a word image. The effect of global and local features is considered in similarity measure. Local features are extracted by scanning vertical strips of a word image. They represent word images better than global features. Local features are more informative of the distinctive characteristic of the word such as inter and intra-characters (that form words) and inter-character spacing. This information greatly helped us to see how features behave to degradation effects and printing variations. Features are analyzed individually and in combination. Performance analysis reveals that the use of combined local features is effective to manage the various artifacts and printing variations seen in printed documents. We propose combined features of moments and profiles that are invariant to fonts, styles, sizes

and degradations.

In addition, the introduction of DTW-based partial matching scheme enables to control morphological variants of a word for performance improvement. We undertake a comprehensive experiments on both feature extraction and matching schemes using datasets prepared in English, Amharic and Hindi scripts. The result shows the effectiveness of our feature extraction and word image matching schemes to undertake efficient content-based retrieval of document images.

Designing effective word image matching scheme helps us further to develop a sophisticated data structure for indexing and clustering word images. This is a major step to solve scalability issues in searching for relevant documents from large collections in the image domain. In the next chapter we present the proposed indexing scheme for speeding up searching from document image collection.

Chapter 7

Indexing Document Image Collections

7.1 Introduction

Information retrieval (IR) system is designed to locate the whereabouts of documents that can satisfy information needs of users from the available huge collection of documents. Its basic operation is to measure the similarity between a document and a query. Documents with high similarity are presented to the user as the result of retrieval. Developing a retrieval scheme in the image domain pass through multiple of steps. Some of these are image processing, feature extraction, matching and retrieval of relevant documents. With the massive increase in image collections, each of these steps could be computationally expensive. As a matter fact, content-based image retrieval (CBIR) systems have thus far focused on enabling search and retrieval over relatively small image collections. The scalability and performance issues need to be further addressed to come up with an applicable search and retrieval procedure for document image collections.

In a typical book, there could be around 90, 000 words and processing all of them online is practically impossible. We need to introduce the indexing step so that all computationally intensive operations are done during the indexing process offline and enable minimal operations during online retrieval. The system also needs to be scaled up to the users requirement. Better indexing scheme is required to organize documents and keep them up-to-date. Queries must be handled quickly, at a rate comparable to text search engines. Hence, developing a successful search engine in image domain demands to address such issues.

A conceptual block diagram of the search and retrieval procedure is shown in Fig-

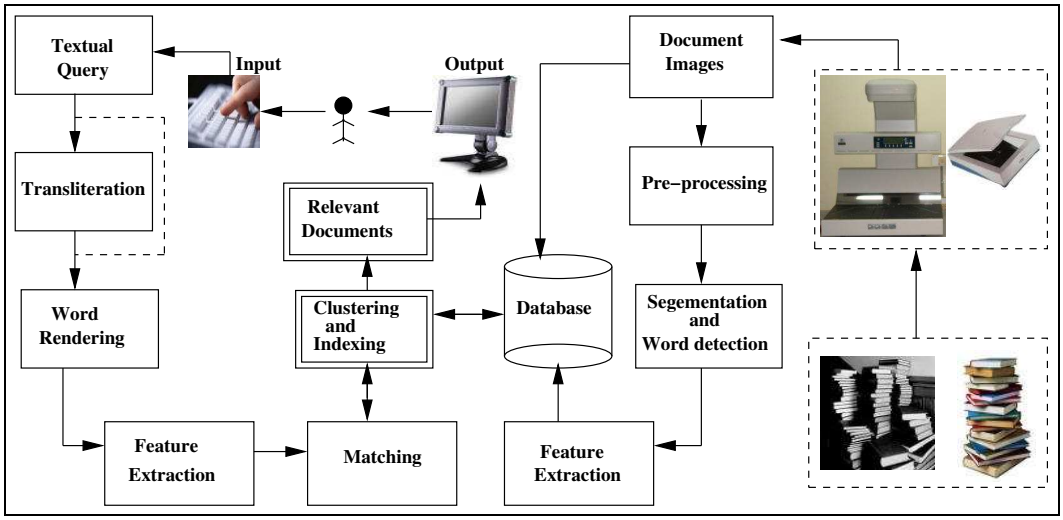


Figure 7.1: Conceptual diagram of the searching procedure. This chapter attempts those in double boxes.

ure 7.1. We address the feature extraction and word image matching problems in document images, as presented in Chapter 6. Here we propose an effective indexing scheme that enhances searching and retrieval of relevant documents from a collection of printed document images. The major contributions in this chapter are the following:

- Indexing documents require identification of their representative index terms. To this end, we mimic basic document processing techniques (such as morphological analysis, stopword detection and relevance measurement) in the image-domain. Section 7.3 summarizes the proposed document processing techniques for extraction of relevant index terms for clustering word images.

- In a rapidly evolving corpora, dynamic update of clusters is a basic requirement. Hence we propose incremental clustering to suit the dynamism of document images archival in digital libraries. Our indexing structure also allows dynamic update of clusters. Section 7.4 presents clustering algorithms for grouping together similar word images, as well as for updating clusters in a dynamic environment.

- Designing an effective indexing scheme in the image domain that speed-up searching and retrieval of relevant documents. Section 7.5 discusses the proposed indexing data structure.

Modules	Purpose	Algorithm(s) Used	
		Text search engine	Current work
Stemming	Group morphological variant words	Language modeling (e.g. Porter algorithm)	Morphological matching using Dynamic time warping
Stopword detection	Remove common words	Stop word list	Inverse document frequency (IDF)
Relevance measurement	Rank documents	Term frequency (TF)	Modified TF/IDF
Clustering	Group index terms	—	Improved hierarchical clustering
Indexing data structure	Organize index lists	Inverted index and signature file	Inverted index

Table 7.1: Mapping text processing techniques for effective indexing in image domain. A comparative review of their application in text search engines vs. the present work

7.2 Overview of the Indexing Process

Indexing is the process of converting a collection of data into a list suitable for easy search and retrieval [97]. The basic task of indexing is to process a given document collection, identify relevant terms that represent it and organize these terms into a list that serves as access points to actual documents. Indexing document images has many advantages [53]. It provides the ability to characterize and organize documents in a meaningful way. Indexing also provides mechanisms to retrieve, in ranked order, the most relevant documents for a given query.

Designing an effective indexing scheme requires organizing representative index terms (i.e. word images) using suitable indexing data structure. This necessitates identification of relevant index terms from document images. Determining index terms from large collection is one of a challenging task during indexing. Because all words are not equally significant to represent a document content. In addition, there are word variants that needs to be grouped together. As a result of which, documents need to pass through a serious of preprocessing steps.

In a text retrieval system, documents are indexed with the words present in them. The most common way of characterizing the content of text document during indexing is to perform document processing for the extraction of representative terms [68].

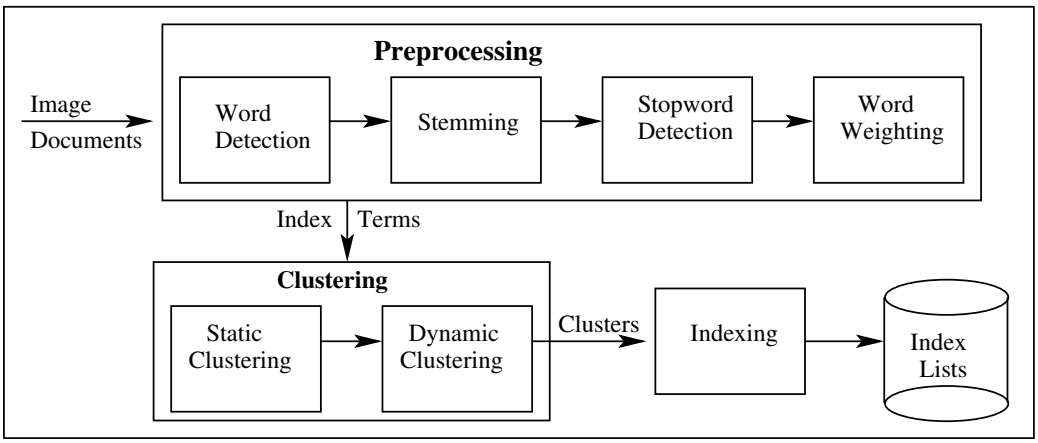


Figure 7.2: Design steps of an indexing scheme for document image collections

This is achieved through applying a series of document processing techniques that identify words, extract relationships between words, and measure their relevance. The extracted index terms are grouped together based on their similarity measurement and are indexed using the selected indexing structure. Once the indexing scheme is built, given a query, the relevant documents are immediately located out of the index.

By observing the steps followed for indexing textual documents, we find that developing an effective indexing scheme for document images require an integration of IR principles for stemming, stopword detection and relevance measurement. The application of IR measures in image space has helped us immensely to generate index terms that are used for clustering and indexing document images. Table 7.1 shows a comparative review of how IR principles are effectively used for indexing documents in image domain.

Once we identify index terms, they are organized using indexing scheme. Based on the indexing structure, documents containing similar word images are clustered together and keywords are annotated to generate index vectors that are used for searching and retrieval. To facilitate searching and retrieval from rapidly evolving corpora in digital libraries, we also enable dynamic update of clusters, without involving complete re-clustering.

An overview of the indexing process we follow for constructing index lists from document images is given in Figure 7.2. The system accepts a collection of document images as an input. They are processed to generate content-bearing list of index terms (or word images). These terms are then clustered based on their similarity measures. A set of clusters are finally organized in an indexing data structure using cluster keywords (centroids). We annotate keywords to ease searching from large

collection of document images. The indexing process finally constructs index lists that are used during the searching process. This speeds up searching in document image collections.

7.2.1 Datasets

To demonstrate the effectiveness of our indexing strategy we undertake experiments at various levels using documents written in English, Hindi and Amharic languages. We select these languages because, in addition to English, Hindi and Amharic are the official languages of India and Ethiopia, respectively. More than half of the population speak these languages in their respective countries. As a result, there are a number of printed documents available in the form of books, magazines, newspapers, etc. for information processing. Application of better indexing scheme for organizing digitized document images written in these languages will facilitate users search and retrieval for relevant documents.

Given document images, they are preprocessed offline to threshold, skew-correct, remove noise and thereafter detect word images. Tokenization (or segmentation) decomposes a document and return an array of word images (or tokens). Once word images are detected, then we extract a combined one-dimensional feature vectors (consisting of word profiles and moments) $\mathcal{F} = \mathbf{F_1}, \mathbf{F_2}, \ldots, \mathbf{F_N}$ for representation. The feature vector is normalized such that the word representations become insensitive to variations in word size. We use these datasets for evaluating the performance of each of the algorithms implemented during the construction of the indexing scheme.

7.3 Extracting Index Terms from Document Images

Our indexing scheme automatically stems word variants, detects language related stop-words and cluster similar words together and finally index content-bearing words based on their relevance measurement. This is done without considering any language specific rules. We demonstrate this using datasets of the various languages.

7.3.1 Stemming Word Variants

The indexing process for a good search engine is more involved than direct word-to-word matching. Because, a word, with similar meanings, usually appears in various

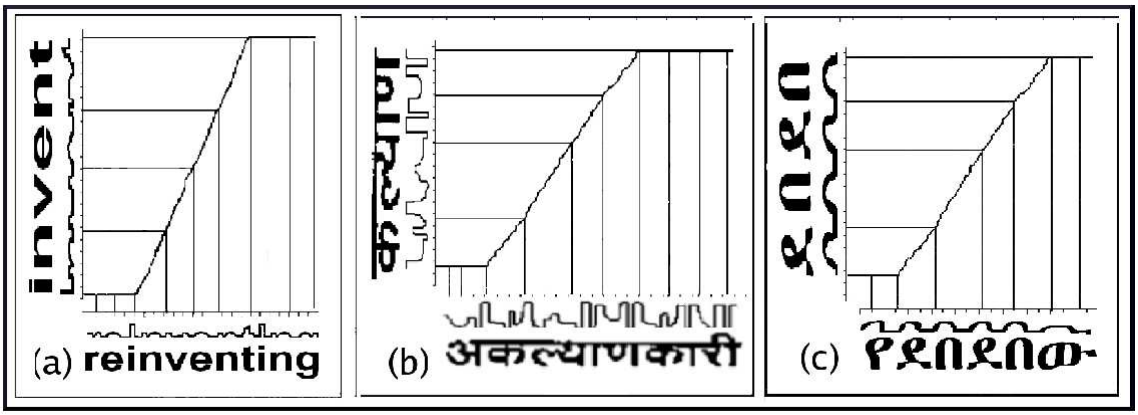


Figure 7.3: DTW alignment between a pair of words in English (a), Hindi (b) and Amharic (c) languages, respectively. The middle line in the graph shows DTW warping path that is recovered by aligning two word images.

forms. This requires a stemmer to logically conflate the various morphological variants of a word into a common stem/root. For instance, when users enter the query word 'invent', they most likely want to retrieve documents containing the terms 'invents', 'invented', 'reinventing' and other variants as well. Thus, the use of stemming improves recall, i.e., the number of documents retrieved in response to a given query. Since many terms are mapped to one, stemming also reduces the total number of clusters which further decreases the size and complexity of the index files in the retrieval system.

In the text domain, since variation of word forms usually obey the language rules, text search engines use this information for stemming. For document image indexing language-related information is not directly usable. We design dynamic time warping (DTW)-based partial matching scheme for morphological matching in the image domain, as discussed in Section 6.4.

By applying our morphological matching scheme, we take care of word form variations taking place at the beginning and the end. Figure 7.3 (a) depicts plots for matching profiles of words 'invent' and 'reinventing' along with matching profiles of other words in Hindi (b) and Amharic (c). It can be observed (from Figure 7.3) that for word variants the DTW path deviates from the diagonal line at the two extreme ends. This happens, for instance, during matching the root word ('invent') with its variant ('reinventing'). The word 'reinventing' differ from 'invent' both at the beginning and at the end because of the addition of 're' and 'ing'. Profiles of these extra characters have minimal contribution to the matching score, and, hence the added

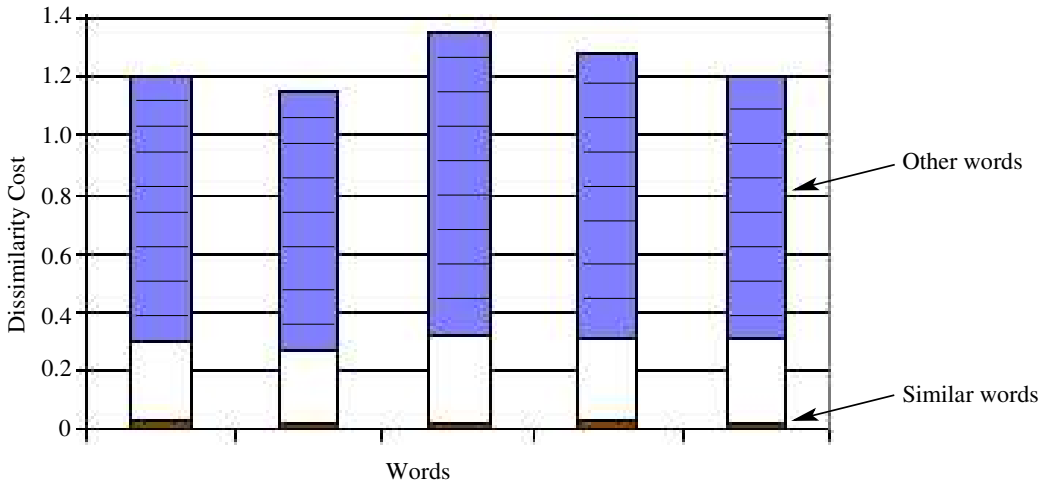


Figure 7.4: Dissimilarity cost of a set of example words. When we match with a given word, dissimilarity cost of variant words is very low while those of any other randomly selected words is big. Based on this information we can easily identify and group variants of a given word

costs are subtracted in order to compute the net-cost of the partial match.

With the reduction in the dissimilarity score of a variant word (against its root word), we find that a large set of words get grouped together. Figure 7.4 shows dissimilarity cost of a set of example words with a given reference words. Note that similar words have very low dissimilarity (with DTW score less than 0.06), while any other word have high dissimilarity (with DTW score of more than 0.25). This shows that with our partial matching technique, variant words have smaller dissimilarity than any other arbitrary words. Hence the use of the net cost of the partial matching algorithm as a distance measure is helpful to group together similar words during clustering. This makes it easier to determine relevance of words for representing documents during indexing.

7.3.2 Detection of Stop Words

Stopwords have little power to discriminate relevant documents from non-relevant once. We ignore stopwords, because they tend to slow down searching without improving the result. We propose inverse document frequency (IDF) for identifying stopwords from a collection of candidate index terms. IDF weight of each word is defined by:

$$IDF = logN - logd_j \tag{7.1}$$

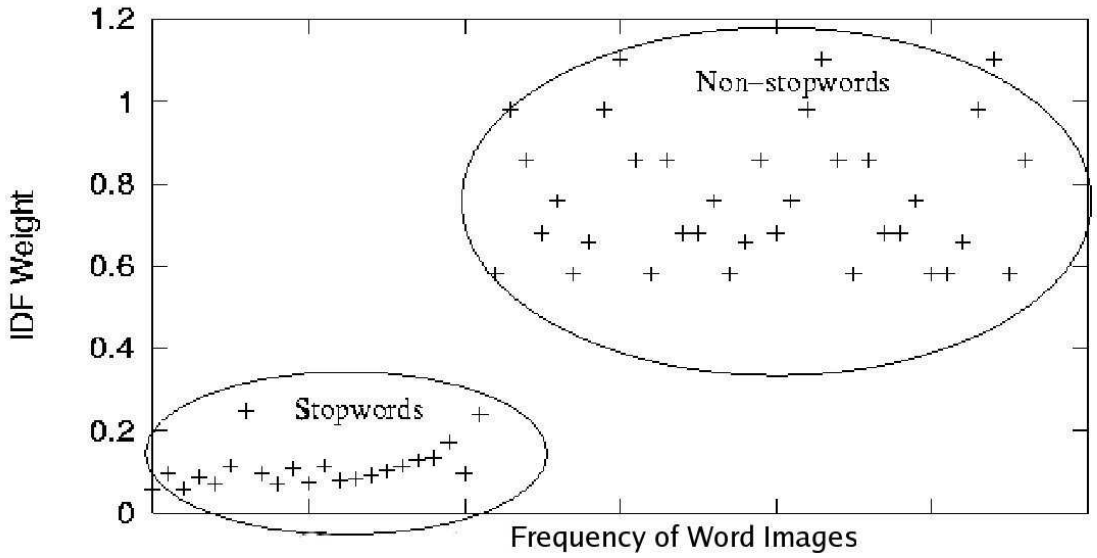


Figure 7.5: Using IDF weight stopwords are identified from other non-stopwords. These stopwords are detected automatically during the Indexing Process.

where, N is the total number of documents in the collection, and d_j is the number of documents containing the word j.

We compute IDF weight of each word image automatically during the indexing process. The proposed IDF technique explores the uniformity of the presence of similar words across the documents. To ease analysis of relevance of documents for a given term, similar words are first grouped together into one cluster. Then we use IDF to measure number of documents containing a given word. The result shows the extent of importance of words in document representation. Since stop words usually have minimum IDF weight, we observe that the importance of a word decreases with the increase in the number of documents containing it. Figure 7.5 shows how IDF weight categorizes stopwords from other relevant words. Since these stopwords appear in most of the documents, they have near zero IDF weight. This means that these words are less meaningful to characterize any of the document. We exclude all stopwords with less IDF weight from the list of index terms. English words such as 'and', 'am', 'this', 'have', 'of, 'over', etc. have an IDF weight less than 0.30. In a given English book with more than 200 pages, around 40 percent of the total word images found as stopwords. This reduces the size of index terms used for indexing document images.

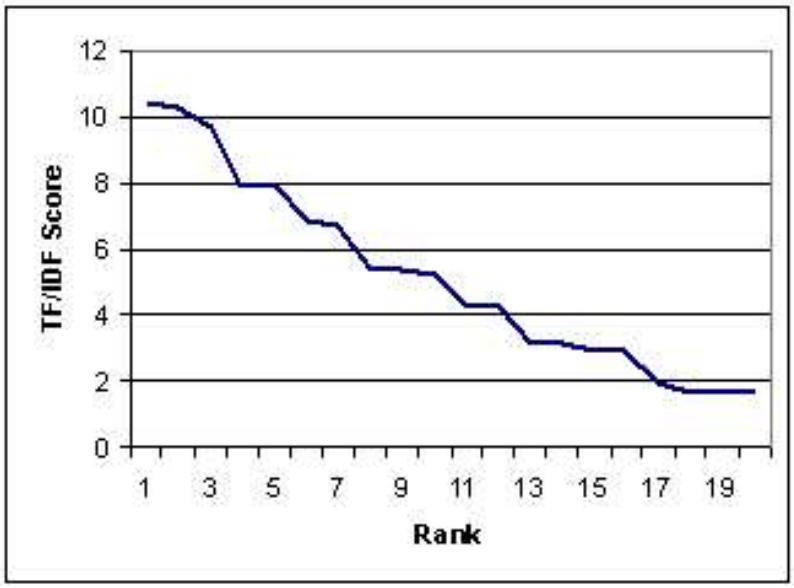


Figure 7.6: Documents are ranked in descending order based on their TF/IDF weight. Those document with high TF/IDF score are more relevant than other documents.

7.3.3 Word Weighting

Knowing relevance of documents for a given term requires analysis of words occurring in a document collection. Word analysis relies largely on frequency-based weighting methods [97]. Word weights reflect the importance of each term (or word) appearing in a document. In our present work, we propose IR measure called TF/IDF weighting technique to measure relevance of documents in image-domain. The use of this technique ensures that words appearing in more documents are often more valuable than those occurring in few documents. TF/IDF weighting is defined as:

$$W_{ij} = f_{ij} * (logN - logd_j) \tag{7.2}$$

where, W_{ij} is weight of term j in document i, f_{ij} is frequency of term j in document i, N is the number of documents in the collection, and d_j is the number of documents containing the term j. Thus the weight W_{ij} quantifies the extent of relevance of the document for a given word. Relevant documents usually have high TF/IDF weights relative to other documents.

During indexing, after stopwords are screened out, documents that contain relevant terms are ranked based on their TF/IDF weights. Those documents with highest occurrence of similar words are more relevant and, hence appear at the top of the list. Figure 7.6 shows rank of documents retrieved for the query word "English" using

TF/IDF score. Documents are ranked in descending order based on their TF/IDF score. Documents with the highest TF/IDF score are given more priority, since these documents contain relatively more of the queried word (and its variants) than others. This ensures that users have got relevant documents in response to their query, increasing their satisfaction in the retrieval system.

7.4 Clustering Document images

To enhance searching in large collection of document images, there is a need to cluster documents based on the words they contain. Given a document images, we segment them into n words. For each word, a one-dimensional feature vector $\mathcal{F} = \mathbf{F_1}, \mathbf{F_2}, \ldots \mathbf{F_M}$ is extracted (where M is width of the word). Then they are pre-processed to extract index terms. We cluster these terms based on their dissimilarity measure. A cutoff point λ is set on the optimal clustering diameter such that similar words have a matching cost less than λ.

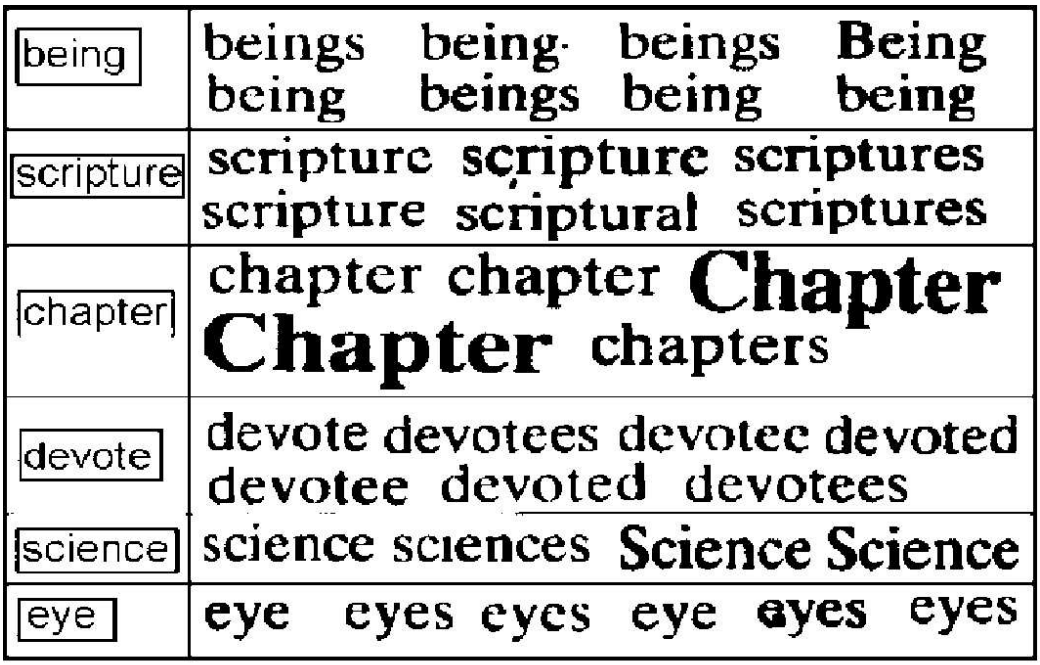


Figure 7.7: Sample clustered words for the various index terms generated from English document images. Index terms are shown in special boxes

The feature extraction and matching schemes proposed in Chapter 6 are used for word image representation and similarity measure. The distance between two

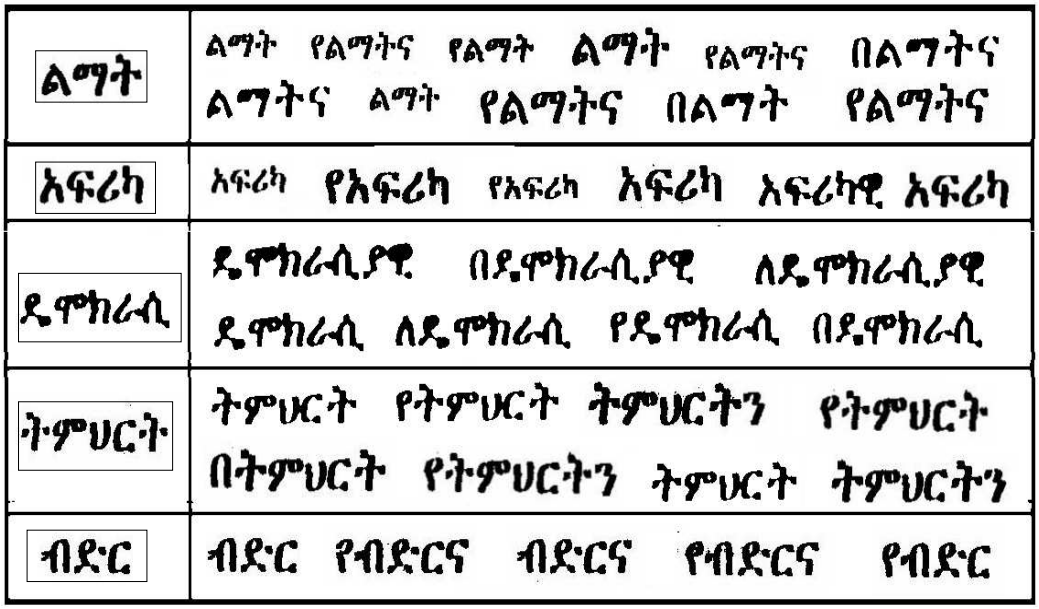


Figure 7.8: Sample clustered words for the various index terms generated from Amharic document images. Index terms are shown in special boxes

sequence of feature vectors, $dis(\mathcal{F}, \mathcal{G})$ is taken as a common metric to decide the similarity between word images. We use DTW-based partial matching algorithm proposed in Section 6.4 for the similarity computation. The result shows the effectiveness of the algorithm in cutting matching cost of all similar words that vary in morphology (word forms), fonts, styles, sizes and document quality. Sample clustered words from English and Amharic documents are shown in Figure 7.7 and 7.8, respectively. The result shows that, for the word image in special boxes, similar words are clustered together into one set. Each of the clusters are therefore just a variation of the single-base word. Such variants of a word can generally be grouped into three:

- Morphological variants of a word. For instance words under 'devote' includes words such as 'devotee', 'devotees', 'devoted'.

- Printing variations of a word. This include changes in fonts, styles and sizes. For instance, consider words clustered under 'chapter'. They vary in size and style.

- Degraded words because of cuts (e.g. devotee), blobs (e.g. eye), merge (e.g. scripture), etc.

Clustering similar words together is advantageous during the retrieval process. Because it increases efficiency in searching and retrieval of relevant document images. In addition, users are not penalized by using a specific word (like 'science') during query formulation from retrieving documents that contain similar words such as 'Science' and 'sciences'.

In our present work we use an improved hierarchical agglomerative clustering algorithm [54] because of the following reasons. (i) It works based-on pair-wise distance. (ii) The thresholds can be easily set and quickly evaluated. (iii) The procedure is inherently incremental, thus allowing for new images to be added to existing clusters and (iv) It is efficient in computation - $O(N^2)$, N being the number of word images. Using this approach, we attempt to group together similar words into one cluster. For each cluster, we then define cluster representative which is used later during cluster manipulations. A cluster centroid is usually constructed as the average vector of all members in a given cluster. This requires feature vectors with equal dimensions. Since we are dealing with features that differ in size, assignment of cluster centroid is not that simple and straight-forward. We decide centroid γ of each cluster using:

$$\gamma = \arg\min_{i} dis(w_i^k, w_{i+1}^k) \tag{7.3}$$

where $dis(w_i^k, w_{i+1}^k)$ computes the distance between members w_i^k and w_{i+1}^k of the cluster k. Then a word that exists at minimal distance from all others is used as cluster centroid. This gives us a word that has the highest similarity to all other words in the cluster. With our approach, it is observed that basic/root words are mostly considered as centroid words for most clusters. We use the centroids to annotate clusters to enhance searching in index table.

During clustering, we assume the following invariant parameters to keep uniform.

- a cutoff point of λ on the optimal clustering diameter,
- for each cluster c_i, the radius (r) or intra-cluster distance $max_{r \in c_i} dis(c_i, r)$ is at most λ, and
- for each cluster c_i and c_j, the inter-cluster distance $dis(c_i, c_j)$ is at least λ.

With this assumption, the modified agglomerative clustering scheme works as given in Algorithm 9.

As pointed above, the algorithm starts with n singleton clusters and compares these clusters with each other to group those similar k ones together. Though the worst case time complexity, $O(n^2)$ seems the same as the original agglomerative hierarchical

Algorithm 9 Agglomerative clustering algorithm.

```
for i = 1 to n do
    for j = 1 to n do
        if (i ≠ j) and (dis(c_j, c_i) < λ)
            S_k = S_k ∪ c_j
    end for
    Compute IDF score of S_i
    if (IDF < α)
        Reject the cluster as stopword
        Update the stopword list
    else
        Identify cluster centroid γ of S_i
        Compute TF/IDF score of documents in cluster S_i
        Sort documents in S_i in descending order of their TF/IDF score
end for
Return cluster list S.
```

clustering algorithm, the modified algorithm is much faster during clustering as it requires only $(n-k)$ comparisons, rather than $(n-1)$) comparisons. At every iteration we cluster only those points that were not previously clustered. With our technique the pairwise distances need to be computed for only those word images that have not yet been clustered. The novelty of this clustering algorithm is that it can be extended to support incremental clustering algorithm for easier update of existing clusters to incorporate new document images.

7.4.1 Incremental Clustering of New Documents

Digital libraries are archiving an ever-increasing volume of data, which is posing a growing challenge for information retrieval systems to effectively store and retrieve up-to-date information. In this respect, a major issue that needs to be answered is how to update existing clusters of document images. This is a costly operation to manage for most existing clustering algorithms, since they are designed for static environment.

We need a clustering algorithm that is suitable for maintaining clusters in such dynamic environment, without frequently performing complete re-clustering. For this, we extend incremental clustering approach. Incremental clustering algorithm works by assigning new documents to their respective existing clusters. The requirement

Algorithm 10 Incremental clustering algorithm.

```
for j = 0 to n do
    for i = 0 to m do
        if(dis(w_j, c_i) ≤ min) //min is initially +inf
            min = dis(w_j, c_i)
            pos = i;
    end for
    if (min ≤ α)
        Insert w_j to the pos^th cluster.
        Sort the cluster based on TF/IDF score.
    else
        Add new cluster c_{m+1}
end for
Return updated cluster list C.
```

with this algorithm is that at all times it should preserve all the desirable properties of the previous clustering structure while providing an efficient extension to the dynamic case. The invariant parameters (for inter-cluster and intra-cluster distance) that are assumed above for clustering existing documents are also assumed here during clusters update so that the clusters have the same behavior.

We implement the incremental clustering algorithm to solve the following problem: for an update of n input words, maintain a collection of m existing clusters, $c_1, c_2, ...c_m$, such that each new input is either assigned to one of the current $\mathcal{C}$ clusters, or started off a new cluster. The procedure we follow for cluster update is given in Algorithm 10. Given a set of new word images the algorithm searches for cluster that contains similar word images using cluster centroid.

The algorithm is implemented to work incrementally such that for each new word received it calculates dissimilarity between the word and each cluster. Let a set of new word images $w_1, w_2, ..., w_k$ are ready for clustering, the algorithm follows the following steps:

1. Check whether the new word image w_j is a stopword or relevant word. If it is a stopword, then consider the next word w_{j+1}. Otherwise, proceed to the next step.

2. Analyze the similarity of cluster c_i with the new word image w_j. The decision depends on similarity metric computed between the new word and cluster centroid. If similar cluster c_i is found, the word w_j is added this cluster. Otherwise,

a new cluster c_{m+1} is added to the existing clusters.

3. If the word is added to one of the existing cluster, recompute relevance of documents in a cluster and sort them accordingly.

4. Perform steps 1 to 3 again for the remaining $k-1$ new word images.

Clustering algorithm	Time complexity
Incremental Clustering	$O(k * m)$
Re-clustering	$O(n+k)^2$

Table 7.2: Comparison of time complexity of incremental clustering algorithm and re-clustering.

Incremental clustering algorithms are more powerful than re-clustering. We can see this fact from their computational complexity. If we apply the re-clustering procedure the algorithm runs for a total of $(n+k)$ by $(n+k)$ iterations, that is, for the sum of n size of existing collection of word images and k size of new word images. Where as, the incremental algorithm runs k new word images against m number of clusters. Table 7.2 summarizes time complexity of incremental algorithm and re-clustering. Thus, incremental clustering algorithm has dual advantage: saving clustering time without affecting existing indexing structure.

Annotation After word image clustering process has been completed, we have a set of similar words in each cluster that are annotated by the cluster representative. Given textual query from users, the query is rendered to convert it to image. Features are extracted from these images and then search is carried out by matching with cluster centroids. Documents from the matched cluster are then retrieved. Since matching word images online is computationally expensive, we would like to build a text index for each document using the word image clusters. Manual annotation of clusters is prohibitive and OCRing the representative words in a cluster is not accurate. The solution we propose is what we call Reverse Annotation.

We obtain a collection of unique words from a text corpus of the language, and render them to obtain word images. Features are extracted and DTW matching algorithm (see Section 6.4 for detail) is used to compare the keyword image against the representative words of each cluster. The cluster that is closest to a keyword image is annotated by the keyword. We thus obtain a textual equivalent for each word cluster and thus a text representation for each word image. We have essentially

used feature based techniques to annotate images, so that a text-based search could be enabled. Sample clusters are shown in Figure 7.9. The clusters consist of example clusters from English, Hindi and Amharic with the textual word they are annotated with. The cluster centroid is marked in blue.

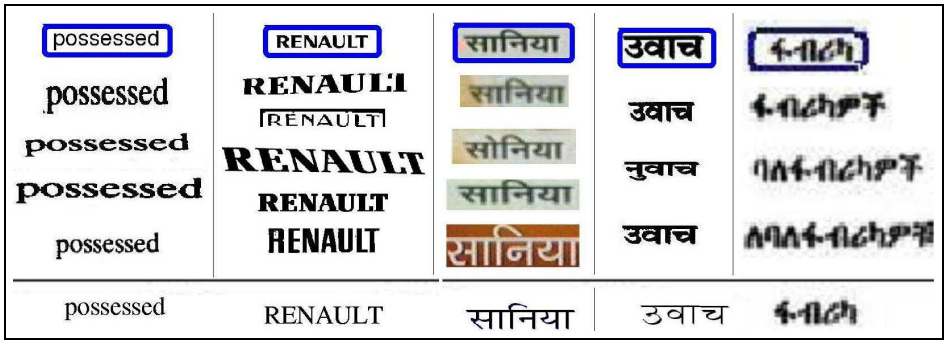


Figure 7.9: Sample annotated word images from English, Hindi and Amharic languages. The representative image is highlighted, and the textual keyword is given underneath each cluster.

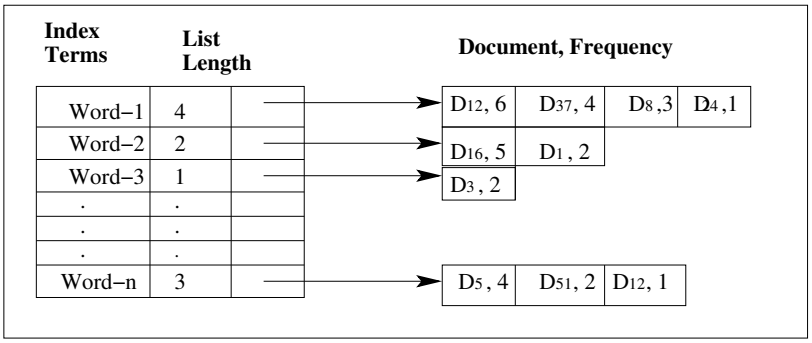


Figure 7.10: Inverted indexing data structure is used to construct index lists. Each index terms are then linked to the actual document images that contain them.

7.5 Indexing Structure for Organizing Document Images

All clusters with words that are deemed important are indexed using the indexing data structure. Indexing provides the ability to organize clustered documents in a structured way such that subsequent searching and retrieval of documents as per users query is enhanced.

DocID	Frequency	Word Coordinates
Page-20	3	(835 871 680 781);(1240 1277 214 324);(1282 1319 797 884)
Page-63	3	(346 384 404 501);(677 713 29 130);(1092 1128 787 874)
Page-91	2	(646 683 749 856);(425 461 816 859)
Page-1	1	(525 561 304 390)
Page-8	1	(869 904 486 585)
Page-12	1	(104 142 345 448)
Page-21	1	(700 736 271 367)
Page-215	1	(144 184 387 488)
Page-30	1	(948 985 196 294)
Page-47	1	(17 52 396 484)
Page-148	1	(561 601 270 357)

Table 7.3: Sample entries for the keyword 'devote' in the inverted index. The first page (Page-20), for instance, contains three words of 'devote' and its variants shown in Figure 7.7. To view the word we use the four coordinates.

We use inverted indexing data structure to organize document images around the words they contain as depicted in Figure 7.10. We implement this data structure since building and maintaining it is a relatively low cost task. Inverted index enhances search where queries are formulated in words. For document retrieval this means that given a word we can immediately locate the addresses of all the documents containing that word.

In building an inverted index, we start with the extraction of postings (i.e. list of keywords), and then maintain sorted postings as a collection of inverted lists. An inverted index over a collection of image databases, therefore, consists of a set of inverted lists of documents, one for each index term (or keyword). The inverted list for a word is a sorted list of locations where the term appears in the collection and frequency of occurrence of a word in a document. Table 7.3 shows the location of a word 'devote' in pages it appeared in as shown in the inverted index file. A location consists of a document identifier (docID) and the four position indicator coordinates (top, bottom, left and right) of the word image within the page. Given an index term , w, and a corresponding location l and frequency of occurrence f, we refer to the three-tuple (w, l, f) as a posting for the word w.

In simplest form, the information kept in each index entry for a word w has the structure:

$$< f_w; d_1, d_2, ...d_{f_w}; c_t, c_b, c_l, c_r >$$

where f_w is the number of documents that contain a specific word w, $1 \leq d_1 < d_2 < ... < d_{f_w} \leq N$, list of documents from the total collection that contain w , and c_t, c_b, c_l and c_r are top, bottom, left and right coordinates of the word image in a document.

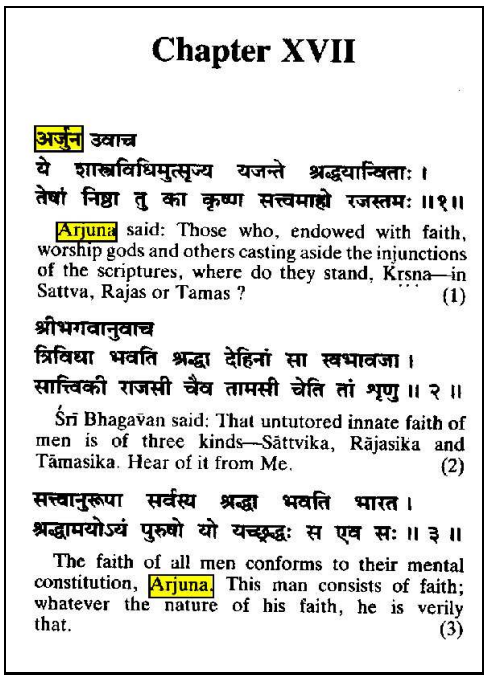
7.6 Retrieval from Document Image Collection[1]

Once the indexing of document images (as words) done, efficient retrieval can be achieved. Balasubramanian, in his Masters Thesis [23] focused on the problems related to annotation and retrieval in documents (primarily online). Major contributions of this work include: (i) Annotation tool for online handwriting data [31][99], (ii) A synthesis scheme for online handwriting in Indian languages [47] and its application for retrieval from handwritten document collections. (iii) Application of the recognition-free retrieval scheme in the graphics stream of PDF [2], to demonstrate the application in an open source software XPDF. In addition, a prototype web-based system was also described for searching in document images [24], which is an extension of the present work. The system has many basic features as discussed below.

Web-based GUI: The web interface allows the user to type in Roman text and simultaneously view the text in one of the Indian languages of his choice. Users can also have the option to search with cross lingual retrieval and can specify the kind of retrieval they want to use. Many retrieval combinations are also provided in the advanced search options such as case insensitivity, boolean searching using !, &, | *and parenthesis*, displaying up to 50 search results per page, and various others. There is on the fly character transliteration available. The user can first choose a particular language (such as Hindi, Telugu, etc.) and then see the text in the corresponding language as he keeps typing the query in Roman (OM-Trans).

Delivery of Images: In order to facilitate users access to the retrieved document images, there is a need to control image size and quality. When a book is typically scanned at a resolution of 600 dpi, the original scanned size of a single page is around 12MB as a PNG file. Viewing such page is too slow and needs network resources. It is wise to make these images available in a compressed form. We compress the above image to a size ranging between 30 to 40 KB in TIFF format, by reducing the size of the image. This makes sure that the delivery of images are faster over the Internet.

[1]This section overlaps with the Masters Thesis of Mr. A. Balasubramanian [23]



Chapter XVII

अर्जुन उवाच
ये शास्त्रविधिमुत्सृज्य यजन्ते श्रद्धयान्विताः ।
तेषां निष्ठा तु का कृष्ण सत्त्वमाहो रजस्तमः ॥१॥

Arjuna said: Those who, endowed with faith, worship gods and others casting aside the injunctions of the scriptures, where do they stand, Krsna—in Sattva, Rajas or Tamas ? (1)

श्रीभगवानुवाच
त्रिविधा भवति श्रद्धा देहिनां सा स्वभावजा ।
सात्त्विकी राजसी चैव तामसी चेति तां शृणु ॥ २ ॥

Śrī Bhagavān said: That untutored innate faith of men is of three kinds—Sāttvika, Rājasika and Tāmasika. Hear of it from Me. (2)

सत्त्वानुरूपा सर्वस्य श्रद्धा भवति भारत ।
श्रद्धामयोऽयं पुरुषो यो यच्छ्रद्धः स एव सः ॥ ३ ॥

The faith of all men conforms to their mental constitution, Arjuna. This man consists of faith; whatever the nature of his faith, he is verily that. (3)

Figure 7.11: Search result with Dynamic Coloring for query word 'Arjuna' seen both in English and Devanagari

TIFF image format helps us in general for achieving the trade-off between image size and quality. It keeps the quality of the image during the compression process over JPG and BMP formats.

Speculative Downloading: The system also supports speculative downloading, where some related pages with the currently retrieved page are prefetched for quick viewing during searching and retrieval as per users query. This mechanism is helpful especially when the user is viewing a collection, page by page, with the assumption that he might view the next page also. Speculative downloading is a background process.

Dynamic Coloring: When a user searches for relevant pages to a given query, our system searches and displays the result with dynamic coloring of all the words in the page that are similar to the queried word. This helps users to easily evaluate relevance of the retrieved page to their need. We adopt false coloring mechanism such that each word in a query carries a unique color in a document image. All this coloring happens at runtime (Figure 7.11) at image level.

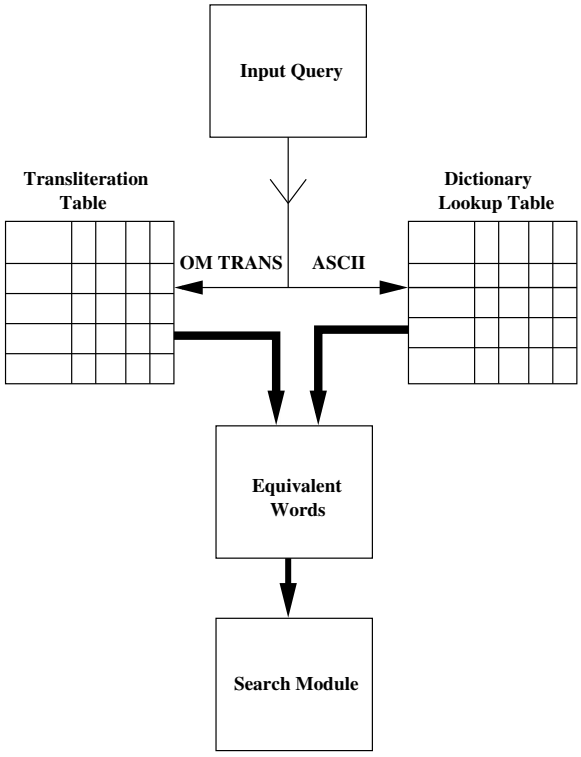


Figure 7.12: A conceptual diagram that shows document searching in multiple languages using transliteration and dictionary based approach

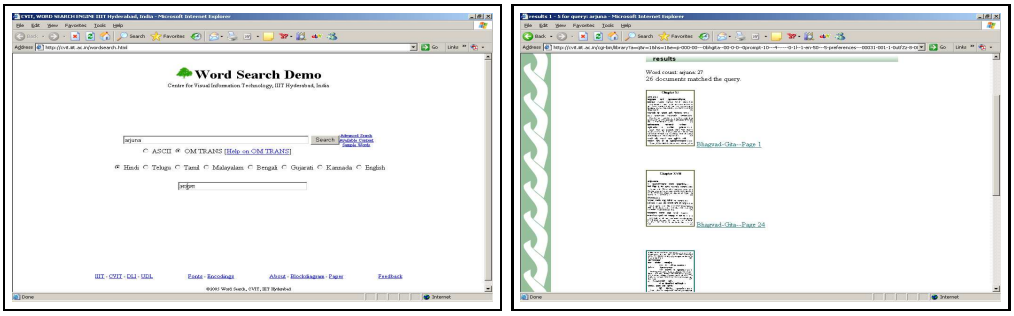
Cross-lingual Search The system can also search for cross-lingual documents for a given query. As shown in Figure 7.12, we achieve this in two ways: transliteration and dictionary-based approaches. The figure is basically an expanded view of the cross-lingual block presented in Figure 7.1.

Since Indian scripts share a common alphabet, we can transliterate the words across languages. This helps us to search in multiple languages at the same time. We use OM-Transliteration scheme [77]. In OM-Transliteration scheme, there is a Roman equivalent for all the basic Indian language characters. Figure 7.13 shows a sample transliteration map built for this purpose. For Example, "Bhim" can be typed in its Roman equivalent using OM-trans as "Bhima". Then the transliteration table is looked up for searching in Hindi (Devanagari script) and Telugu pages. Their cumulative result is finally displayed back to the user. A screenshot showing implementation result is presented in Figure 7.14.

We also have a dictionary-based translation for cross-lingual retrieval. In this approach, every English word has an equivalent word in the corresponding Indian

Alphabet	a	aa	i	ii	u	uu	....
Hindi	अ	आ	इ	ई	उ	ऊ	....
Telugu	అ	ఆ	ఇ	ఈ	ఉ	ఊ	....
:	:	:	:	:	:	:	
:	:	:	:	:	:	:	
:	:	:	:	:	:	:	

Figure 7.13: Sample entries of the transliteration map built for cross-lingual retrieval in English, Devanagari and Telugu



(a) (b)

Figure 7.14: Screenshots of implementation results for cross-lingual search. (a) An interface where users enter their query, (b) View of result of the search as thumbnails

and other oriental languages. If a user queries for the word 'India', the dictionary lookup points to 'Baarta' in Hindi for searching relevant documents across languages. This table is extended also for other Indian languages. The result of the search are documents that contain the query word 'India' in all the languages.

We tried searching in scanned documents from 'Bhagavat-Gita'. Pages from this book contain English and Devanagari text. These pages are of poor quality. We tried searching for the occurrences 'arjuna'. It fetched pages which contain 'Arjuna' in both English and Devanagari. Sample results are shown in Figure 7.15 (b). The system also retrieved words containing variants of the word 'arjuna'.

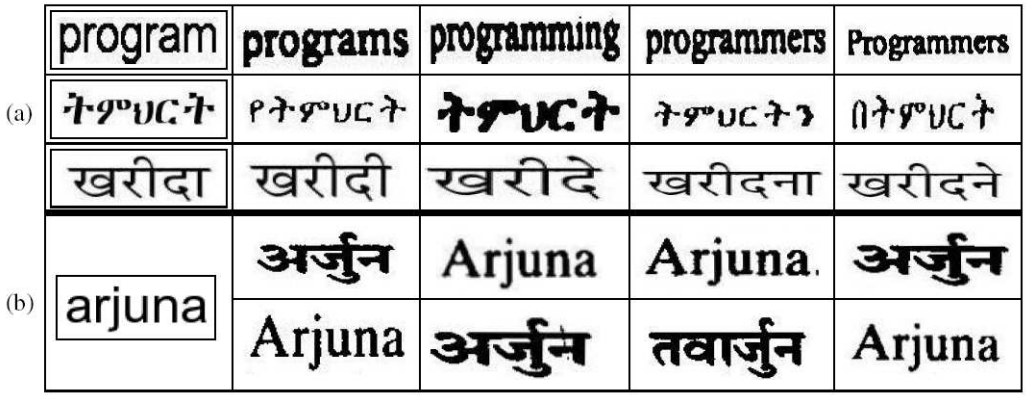


Figure 7.15: Results: (a) Some sample word images retrieved for the queries given in special boxes. Examples are from English, Amharic and Hindi Languages. The proposed approach takes care of variations in word-form, size, font and style successfully. (b) Example result for crosslingual search from Bhagavat Gita pages. Similar words are retrieved both in Devanagari and English.

7.7 Discussion

In this work, we present an indexing scheme for effective search and retrieval of relevant documents from a collection of printed document images. We identify the word set by clustering them into different groups after processing the page to be indexed for detection of relevant words in it. We map IR measures for relevance ranking and stop-word detection in image space. After collecting those similar words into one cluster, we analyze relevance of a cluster. A cluster with stopwords is detected by computing the uniformity of the presence of similar words across the documents. Note that, all the computations are done without an explicit textual representation. Once we generate list of index terms that define the content of the document, they are indexed using inverted index data structure.

Once the index structure is built, it is ready for searching relevant documents from the database. For searching and retrieval we need only to refer the inverted lists. During searching, the keyword of each cluster is used for matching words with the query. This offers fast searching and retrieval. Then a link is made to the clustered DocIDs in the posting lists to generate relevant documents for users query along with the whereabouts of their queried word in the actual document image as shown in Table 7.3. Since DocIDs are stored in sorted order of their relevance to the keyword during clustering, there is an assurance that users will retrieve relevant documents in ranked order as per their query. This kind of organization allows users

to undertake comprehensive search for all possible occurrence of the word in the collection of document image database.

Search engine	Search time	Transfer time
Greenstone	0.13 seconds	0.31 seconds
Our system	0.16 seconds	0.34 seconds

Table 7.4: Comparison of search efficiency of our system with Greenstone text search engine.

Using our prototype system for retrieval of document images, we measure the speed of the system in comparison to text retrieval system like Greenstone. Table 7.4 presents summary of the result. Greenstone is a suite of software for building and distributing digital library collections [70]. Our system takes 0.16 seconds to search and retrieve relevant documents from image databases and 0.34 seconds to transfer that page for viewing by users over the intranet. Where as, Greenstone text search takes 0.13 seconds to search and retrieve relevant documents from databases and 0.31 seconds to transfer that page for viewing by users over the intranet. The speed of our system is almost comparable with the Greenstone text search. This shows the effectiveness of the system. The strategy we followed is to perform text processing and indexing offline. The online search then takes place on the index lists. Compressing the image (to a size of few KB) has also a significant impact during the transfer of the document image for viewing.

We demonstrate the performance of the retrieval scheme on sample document image databases taken from three languages: English, Amharic and Hindi. Around 15 query words are used for testing. Results of the search are document images containing queried word sorted based on their relevance to the query. During selection of query

Language	Dataset	Test*	Precision	Recall
English	2507	15	95.89	97.69
Hindi	3354	14	92.67	93.71
Amharic	2547	14	94.51	96.63

[2]*Number of words used for testing

Table 7.5: Performance of the proposed approach on three datasets in English, Hindi and Amharic. Percentages of precision and recall are reported for some test words

words, priority is given to words with many variants. For these words precision and recall are tabulated in Table 7.5. Percentage of relevant words which are retrieved from the entire collection is represented as recall, where as, percentage of retrieved words which are relevant is represented as precision. It is found that a high precision and recall (that is close to 95%) is registered for all the languages. This is because of the effectiveness of our retrieval scheme. The effectiveness emanates from the proposed feature extraction, word image matching and indexing procedures designed in this work.

7.8 Summary and Comments

These days huge collection of document images have been archived by many applications. The retrieval system is expected to aid in searching the huge document image collection and thus the scalability issues come to forth. Indexing provides the ability to characterize the document image corpus in a meaningful way such that subsequent searching and retrieval of documents as per users query is enhanced.

Indexing the large collection takes immense time. For us indexing is an offline activity. Searching and retrieval is the only online process. That is why the system manages to run fast in the above sample database. Because it only checks keywords of the index to search for similar words with the query.

To facilitate the indexing process, we propose proper document representation. We accomplish this task through applying IR measures for identifying word variants, detecting stop-words and measuring relevance of documents. Such document processing techniques help us to extract index terms that are significant to represent the content of a document. This is done automatically during clustering document images by the words they contain.

We cluster documents using an improved agglomerative hierarchical clustering algorithm. We implement this algorithm to ease cluster updates with the help of incremental clustering algorithm. With this, we achieve efficient cluster update without a need for complete re-clustering every time. This saves much time and creating new indexes will be a smooth process. The result of clustering shows that, for the given word images, similar words are clustered together. Each of the clusters are therefore just a variation of the single base word in printing, degradations and morphology. We achieve this with the help of the feature extraction and word image matching scheme proposed in the previous chapter.

Once clusters are identified, they are organized in an inverted index data structure. Implementation of this indexing scheme enhances searching that is comparable with

the text search engine. We claim that our indexing scheme is independent of any specific language rules. Hence it can easily be extended for indexing document images that are available in any languages. Evaluation of the performance of the indexing and retrieval scheme on document image databases of different languages shows the effectiveness of the scheme.

Chapter 8

Conclusions and Future Work

8.1 Comments and Conclusions

As diverse and large quantity of documents are getting archived by many applications including the digital libraries, access to the content of such collections becomes a challenging task. We attempt to design schemes for document image recognition as well as document image retrieval so as to facilitate access to digitized document collections. In Africa and India there are a number of languages with their own indigenous scripts. Accordingly, there is a bulk of printed materials available in libraries, information centers, museums and other institutions. But, the document analysis and understanding area has been given less emphasis for these languages in the recent past.

In this work, we explore different approaches for addressing the document analysis and understanding problems with emphasis on African and Indian languages. After extensive script analysis, we have designed an OCR for the recognition of Amharic printed document images. We have conducted experiments to evaluate its performance, in which we have got good results on reasonably diverse quality documents. However, the performance of the OCR varies with the diversity of document images under consideration. This behavior is also true for those OCR systems built following the conventional design approach. In this approach, the recognizer is trained offline and used online for recognition. There is no mechanism for the system to correct its mistakes during the recognition process and optimize its performance for specific needs. We need a recognition system that is capable of intelligently adapting to the characteristics of documents of interest. Machine learning offers one of the most cost effective and practical approaches to the design of pattern classifiers for character recognition problem.

We have presented data-driven self-adaptive OCR system that learns from its experience and improves its performance over time. We achieve this by closing the conventional open-loop system of classifier followed by post-processor by automatically generating training data from the test image and incrementally retraining and modifying the basic recognition unit. To this end, we integrate advanced learning procedures to the recognition system that automatically interacts and pass feedback for further learning. This enables the OCR to easily accumulate knowledge for performance improvement. Experimental results show the effectiveness of our design for the recognition of document image collections. This strategy is promising for the recognition of large digitized document images in many applications, including digital libraries.

Developing an applicable OCR system is a long term process, as compared to the immediate need to access the large amount of archived document images for intended use. In this work, an attempt has been made to address the problem of content-based retrieval from printed document image collections. This has a paramount importance for Indian and African languages that have no robust OCRs at present. We have proposed feature extraction and word image matching schemes for effective search in document images. The performance of these schemes is analyzed on diverse datasets that varies in script, printing, quality and word form variations. Experimental results show the effectiveness of our scheme for word image matching.

Searching in large collection of document images is computationally expensive. The scalability and performance issues need to be further addressed to come up with an applicable search and retrieval procedure for document image collections. In response, we have presented an efficient clustering and indexing scheme. IR principles are mapped from text to image domain for relevance ranking and stop words detection. The clustering scheme group together words that varies in printing, quality and morphology. Performance evaluation demonstrates that our matching and indexing scheme guarantees an effective search and retrieval in image-domain that is comparable with text search engines.

The availability of large document image collections is challenging for the existing algorithms used for document image recognition and document image retrieval. The present approaches needs to be redesigned or new approaches needs to be proposed for the digitized documents to be useful for the intended purpose.

8.2 Scope for Future Work

In this thesis work, we propose schemes to advance the state of the art in the area of optical character recognition as well as in recognition-free document image retrieval.

We recommend the following extension works in the future.

In the present work an attempt is made to design an OCR for Amharic documents. The performance of the system is promising to come up with an applicable system. For improving the recognition result on diverse documents post-processor (based on language models) needs to be developed and integrated to the OCR scheme.

Since printed document images archived by many applications are more of historical and poor in quality, there is a need to apply advanced image pre-processing techniques for document analysis. Document image processing algorithms for document image collections are still missing. Schemes that learns from document image collections itself for better performance are needed.

We attempt to redesign existing approach to OCR. Evaluation of the framework on collection of document images taken from English languages proof its effectiveness. Further work is needed for extending this framework for many of the complex Indian and African scripts, and also for realizing multilingual OCR system.

Conventionally, the design of post-processors is language specific. But our problem is diverse in nature because of the document images archived by digital libraries. We recommend further work to develop a post-processor that is independent of language specific cases. This is necessary to ease the development of multilingual OCRs that work on the various scripts of Indian and African languages.

The present design assumes that segmentation (of pages into words as well as words into recognizable symbols) is available. This assumption needs to be relaxed in the future to address the segmentation errors at various stages.

Recognition from poor quality documents results in a number of recognition errors. Retrieval of documents in this situation requires more functionalities. There is a need to design effective schemes for retrieval in presence of OCR errors.

This work investigate various techniques for proposing an OCR as well as document image retrieval schemes that suits document image collection. We recommend

further work to combine the two approaches so as to exploit their strength and design better scheme to access document image collections.

Our retrieval and recognition work is more concentrated on scanned printed documents. For better visualization of the large image collections at hand, we recommend to further the research work for the recognition and retrieval from complex documents (such as camera-based, handwritten, etc.).

In this work we enable existing matching scheme mimic morphological analysis to group together word form variants. The scheme considers suffixes and prefixes of word images. By considering the different levels of linguistic analysis (such as morphological, syntactic, semantic, etc.), more general matching scheme needs to be designed that also addresses other word variants (e.g. synonymous).

The matching and indexing schemes (designed in this work) needs engineering effort to make them applicable for real-life situations. To this end, we recommend an extension work to develop an online system for searching hundreds and thousands of books over the Web.

With increase in digital resources, document image collections and number of users, it is time to design self-improving document image retrieval systems with direct and indirect feedback for performance enhancement. Schemes needs to be formulated for selection of appropriate features and matching schemes based on relevance feedback from end-users and/or query expansion.

Appendix A

Architecture and Implementation of OCR[1]

A.1 Design of the System

A prototype interactive multilingual OCR (IMOCR) system is designed with the objective of enabling the recognizer with capabilities for learning and adaptation [158]. The interactive multilingual OCR (IMOCR) system has a framework for the creation, adaptation and use of structured document analysis applications. IMOCR is a self-learning system that improves its recognition accuracy through knowledge derived in the course of feedback, adaptation and learning. There are various notable goals towards the design and implementation of IMOCR.

- To setup a system that learns through a recurrent process involving frequent feedback and backtracking.
- To manage diverse collection of documents that vary in scripts, fonts, sizes, styles and document quality.
- To have flexible application framework that allows researchers and developers to create independent modules and algorithms rather than to attempt to create an all encompassing monolithic application.
- To allow end users the flexibility to customize the application by providing the opportunity to configure the various modules used, their specific algorithm and parameters.

[1]This is the work of Sachin Rawat as part of his final year B. Tech project

By doing so the system will be able to interactively learn and adapt for a particular document type. In essence, it provides a framework for an interactive $test \Rightarrow feedback \Rightarrow learn \Rightarrow adapt \Rightarrow automate$ cycle.

A.1.1 Data Corpus

Our OCR system has been designed under the assumption that vast amount of training samples are not available for every category of document. Rather, it builds a huge collection (or corpus) over time based on the feedback mechanism. The corpus encompass all types of data that are involved in printed documents. The collection contains synthetic data (that varies in scripts, fonts, sizes and styles) and real-life documents (such as books, magazines and newspapers). Having such wealth of data enables the system to learn and accumulate knowledge out of it.

A.1.2 Design Considerations

From the implementation point of view, the following design considerations have been made:

- The system supports multiple languages in the same framework. Languages currently supported include English, Hindi, Tamil, Telugu and Malayalam.
- Each aspect, right from the GUI to the font-selection are platform independent.
- Dual mode running of the system is enabled by interactive interface as well as batch mode support.
- The application is designed to allow the user to choose the run-time configuration of the system right from the modules, the inner-lying algorithms, the input-output format, the parameters etc. via support for user specified configuration files. The GUI is designed for multi-level access to configuration parameters that scale up for users with various levels of skill.
- A multi-core approach is taken so as to ease development and integration of modules in the future.

A.2 Architecture of the IMOCR

We design the general architecture of an interactive multilingual OCR (IMOCR) system that is open for learning and adaptation. An overview of the architecture

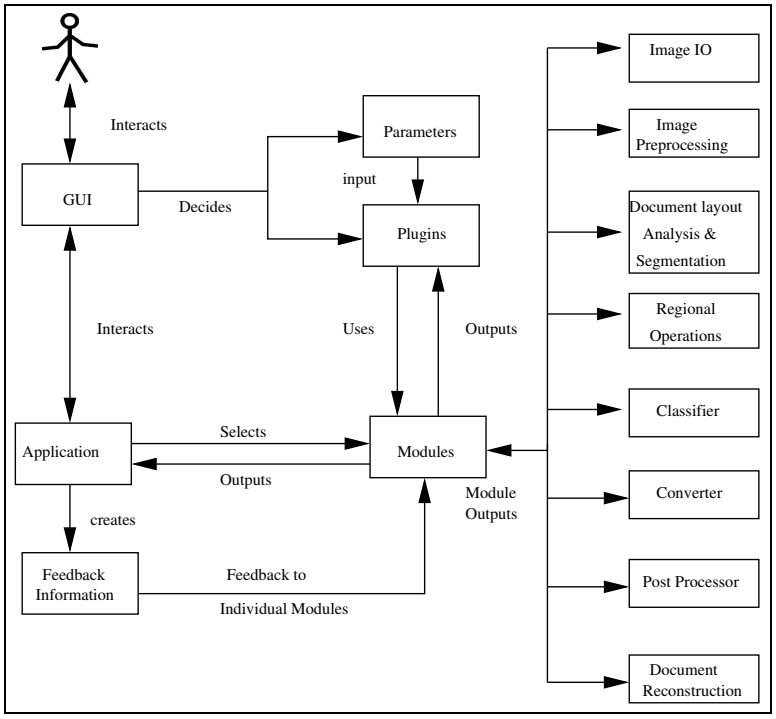


Figure A.1: An architecture of the prototype OCR system

of the system is shown in Figure A.1. The IMOCR design is based on a multi-core approach. At the heart of the same is an application tier, which acts as the interface between the Graphical User Interface (GUI) and the OCR modules. This application layer identifies the user-made choices, initializes data and document structures and invokes relevant modules with suitable parameters. The GUI layer provides the user with the tools to configure the data-flow, select the plug-ins and supply initialization parameters. The system provides appropriate performance metrics in the form of graphs and tables for better visualization of the results of each step and module during the recognition process.

The last layer is the module/algorithm layer where the actual OCR operations are done. This layer is segmented based on clearly identified functionality. Each module implements a standard interface to be invoked via the application. Each module internally can decide on multiple algorithm implementations of the same functionality that may be interchanged at run-time. This helps in selection and use of an appropriate algorithm or a set of parameters for a book, collection or script. This layer is designed on the principle of plug-ins. The system allows transparent runtime addition and selection of modules (as shared objects/dynamic libraries) thereby enabling the decoupling of the application and the plug-ins. Feature addition and deployment is

as simple as copying the plug-in to the appropriate directory. The other advantages of this approach are lower size of application binary, lower runtime memory footprint and effective memory management (through dynamic loading and unloading, caching etc.).

A.3 Recognition Module

The recognition module has three main parts: the segmenter, the labeler and the resolver. Scanned images are initially preprocessed to improve the performance of the recognition process [65]. They are binarized and skew corrected. We implement Isodata for binarization and Yin's algorithms [204] for skew detection and correction. The segmentation module is designed to analyze the overall structure of the document and identify sub-regions such as text, tables, images/graphics etc.

Text blocks are further segmented into lines and words. There are several layout analysis algorithms available in literature. The process of segmentation of a page is carried out sequentially in two major steps. The first is separation of text and image regions. Then, the segmentation of text into columns, blocks, text lines and words. The performance of the page layout analysis module depends on the type of document image. Every algorithm has a set of freely tunable parameters, which can be set to give apt results for the document image. However, these parameters are either set by interacting with the human or automatically by using a learning algorithm [100]. In some cases, the output of the layout analysis may not give apt results for any parameter value. We provide a mechanism for the human to correct the segmented output for further analysis of the document. These human corrections can in-turn be used to improve the segmentation process over iterations.

The segmenter comes up with a list of words that are input to the labeler for feature extraction and training. The labeler detects characters and their constituent components using connected component analysis and extracts features of connected components using principal component analysis (PCA). There are a large number of classes available in the Indian Languages. Therefore there is a large amount of training overhead in computation and storage. This training overhead is reduced by performing dimensionality reduction using principal component analysis (PCA).

The features in the Eigenspace are used for classification using a Support Vector Machine (SVM). Support Vector Machines have good generalization capability and perform well over Indian language datasets. The labeler produces a vector of class labels that is used to produce the final recognition results. Details of the feature extraction algorithm and the classifier can be found in Jawahar et el. [81].

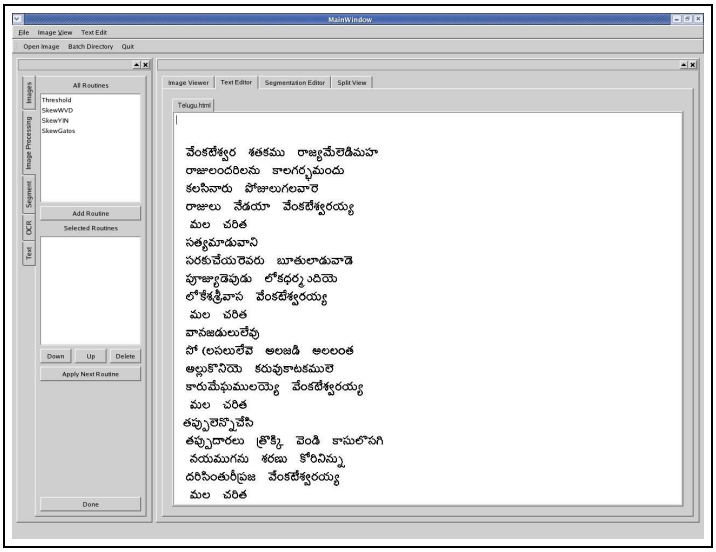


Figure A.2: A screenshot of interactive and adaptive OCR system. The user interacts for selection of algorithms and parameters, feedback and error correction, adaptation and learning.

The resolver module is designed to resolve confusing characters during the recognition process. It uses language specific information to resolve the confusion. Once the confusions are resolved the class labels of each word are converted into UNICODE using a converter. The post-processor is based on contextual information and language-based reverse dictionary approach to correct any mis-recognized words. The post processor can significantly improve the performance of the system. We use a bootstrapping process to retrain the system using the errors corrected by the resolver. The system continuously learns through semi-automatic adaptation from feedback; each time gaining knowledge that improves its recognition rate. The feedback is initiated from the result of the resolver.

The output of the post-processor is a UNICODE string, which can be converted into any language specific font using the converter available. The recognized text is finally reconstructed in various formats including PDF, DOC, RTF, HTML, and Plain text. The output is structurally and visually similar to the input document image as shown in Figure A.2.

Appendix B

Degradation Modeling

B.1 Overview

Most printed document images (of books, manuscripts and newspapers) are of poor in quality. They are almost inevitably degraded in the course of printing, photocopying, faxing, scanning, etc. Degradation models are developed to simulate the various artifacts seen in real-life documents. To improve the performance of the recognizers and also to analyze their performance quantitatively, degradation models were proposed [19][89]. A survey of document degradation models is found in Baird [19].

We develop degradation models to simulate the various artifacts seen in real life documents. Modeling document degradations enables us to closely examine the performance of feature extraction and matching schemes. We model four categories of degradations that are commonly observed in printed documents: salt and pepper, cuts, blobs and erosion of boundary pixels. We generate word images in the various languages by applying our degradation models; samples are shown in Figure B.1, B.2 and B.3. By varying the parameters, our degradation models generate datasets that are similar to the real, scanned images.

B.2 Salt and Pepper Noise

This type of noise is caused by errors in data transmission and scanning. It is distributed all over the image flipping white pixels to black (i.e. pepper) if it is a background and black pixels to white (i.e. salt) if it is a foreground. A useful probability

Figure B.1: Sample Hindi word images degraded with Cuts, Blob, Salt and Pepper and Erosion of Pixels. The following parameters are used to model each of them: for Cuts ($p(b) = 0.02$, $s = 7$), Salt and Pepper ($p(a) = 0.025$, $p(b) = 0.025$), Blobs ($p(a) = 0.001$, $s = 10$), and Erosion ($\alpha = 2, \beta = 3$)

density function for modeling salt and pepper noise is:

$$P(z) = \begin{cases} P^s(b) & \text{for } z = w \\ P^s(w) & \text{for } z = b \\ 0 & \text{otherwise} \end{cases} \tag{B.1}$$

where $P^s(b)$ is the probability of occurrence of black pixel in the image and $P^s(w)$ is the probability of occurrence of white pixel in the image. We use these parameters, $P^s(b)$ and $P^s(w)$ in modeling salt and pepper noise. Thus, by changing values of $P^s(b)$ and $P^s(w)$, we can control the extent of degradation effect of salt and pepper noise in the generative process. An increase in the probability of occurrence of $P^s(b)$ and $P^s(w)$ results in a severe degradation of both foreground and background of a document image and vice versa.

B.3 Cuts and Breaks

The occurrence of cuts in a document image breaks continuity of the shape of characters (or components) by changing pixel values into background (white). Such degradation occurs due to paper quality, folding of paper and print font quality. These

Figure B.2: Sample Amharic word images corrupted with Cuts, Blob, Salt and Pepper and Erosion of Pixels. The following parameters are used to model each of them: for Cuts ($p(b) = 0.02$, $s = 7$), Salt and Pepper ($p(a) = 0.025$, $p(b) = 0.025$), Blobs ($p(a) = 0.001$, $s = 10$), and Erosion ($\alpha = 2, \beta = 3$)

noises corrupt a size of $n \times m$ pixels at a time by flipping black pixels into white. We model cuts using the following probability density function.

$$P(z) = \begin{cases} P^c(w) & \text{for } z = b \\ 0 & \text{otherwise} \end{cases} \tag{B.2}$$

where, $P^c(w)$ is the probability of occurrence of white pixels in a specified size s. The extent of effect of cut depends on value of parameters $P^c(w)$ and s. With the increase in $P^c(w)$ frequent cuts appear in the document. The size of cut, s determines the area covered by cut noises at a time. An increase in s means more area of the image is covered by cut and vice versa.

B.4 Blobs

Blobs occur due to large ink drops within the document image during printing, faxing and photocopying. The existence of these noises merge separate characters or components, thereby distorting character readability. The effect of blobs is $n \times m$ pixels at a time by flipping their values to black. We use parameters, s (size of blobs) and $P^b(b)$ (probability of flipping white pixels to black pixels for the specified s) in modeling blob noise, as defined below.

Figure B.3: Sample Telugu word images corrupted with Cuts, Blob, Salt and Pepper and Erosion of Pixels. The following parameters are used to model each of them: for Cuts ($p(b) = 0.02$, $s = 7$), Salt and Pepper ($p(a) = 0.025$, $p(b) = 0.025$), Blobs ($p(a) = 0.001$, $s = 10$), and Erosion ($\alpha = 2, \beta = 3$)

$$P(z) = \begin{cases} P^b(b) & \text{for } z = w \\ 0 & \text{otherwise} \end{cases} \tag{B.3}$$

The degradation effect of blobs varies with parameter values. With an increase in $P^b(b)$, more blobs appear in the document. Also, as s increases, the area covered by ink drop increases and vice versa.

B.5 Erosion of Boundary Pixels

Degradation caused by erosion of boundary pixels affects the image by changing either black pixels to white or vice versa. It happens mostly at the boundary of the image due to the imperfections in scanning. We use the degradation model proposed by Zheng and Kanungo [203], which states that black and white pixels are swapped according to some probabilities directly related to the distance from the boundary pixel. The probability of flipping foreground pixel and background pixel is therefore:

$$p(0 \mid 1, d, \alpha) = \alpha_0 e^{-\alpha d^2} + \eta \tag{B.4}$$

$$p(1 \mid 0, d, \beta) = \beta_0 e^{-\beta d^2} + \eta, \tag{B.5}$$

respectively, where, η is constant, α and β are used for controlling the flipping probabilities of foreground and background. Degradation of the image varies with different

model parameters. If $\alpha < \beta$, more foreground pixels change to background. If $\alpha > \beta$, more background pixels change to foreground.

Appendix C

Feature Extraction

C.1 Overview

Feature extraction helps to find an appropriate representational scheme for objects to be processed. The extraction of appropriate features for an application has been recognized as an important problem in pattern recognition [182]. Features are the basis of content based image retrieval (CBIR) [159] and object classification [182], and it is a broad research area in the field of computer vision and pattern recognition [65][92].

An image region can be described by either boundary descriptors or region descriptors [159]. The shape of boundary segments can be described quantitatively by using Fourier descriptors. Other features that counts the number of pixels with values above and below the mean and finds perimeter and area of a region (such as moments) are region descriptors.

We present here three categories of features: sequential representations, moments and transform domain representations.

C.2 Word Profiles

A useful region-based representation is the profile. Profile-based features are useful as shape signatures. Profiles of the word provide a coarse way of representing a word image for matching. We consider here upper, lower, projection and transition profiles. Detailed description of profile features are given below:

Projection Profile: Projection profile (F_1) measures the distribution of black pixels in the word image. It is calculated by summing the black pixel values in the image

in either of the three directions: horizontal, vertical and/or diagonal. Horizontal projection is found by summing ink pixels along horizontal rows of the image, where as diagonal projection is by summing ink pixels along the diagonal. We use vertical projection profile which computes the sum of ink pixels along the vertical columns of the image. For a word image, $w(x, y)$, vertical projection profile is defined as:

$$F_1 = \frac{1}{N} \sum_{x=0}^{N-1} w(x, y) \tag{C.1}$$

where $w(x, y) = 1$.

Background-to-Ink Transition: This feature (F_2) records the number of transitions from an ink pixel to the background one. Such feature is important for representing internal structural shape of word images. It is given by:

$$F_2 = \frac{1}{\omega} \sum_{x=0}^{N-1} tran \tag{C.2}$$

where ω is the maximum possible transitions in writing a character in the script (For instance, for English script $\omega = 10$ is enough as there is no character that transits more than this number.) and $tran$ is defined as:

$$tran = \begin{cases} 1 & if\ w(x-1, y) = 0\ \&\ w(x, y) = 1 \\ 0 & otherwise \end{cases} \tag{C.3}$$

Upper and Lower Profiles: Profiles capture the outlining shape of a word. Upper profile (F_3) is computed by recording the distance from the top boundary of the word image to the closest ink pixel in that row. Where as, lower profile (F_4) is computed by recording the distance from the bottom boundary of the word image to the closest ink pixel in that row. The upper and lower profiles correspond to two series of vectors representing the top and the bottom of the word image. They are defined as follows.

$$F_3 = (x - upbound)/N \tag{C.4}$$

$$F_4 = (x - lowbound)/N \tag{C.5}$$

where x is the first ink pixel in a row when seen from top and bottom; *upbound* and *lowbound* represents the upper boundary and the lower boundary of a word image, respectively.

Vertical Distance: This feature (F_5) computes the distance between the two extreme black pixels with reference to upper and lower boundary points in a strip of the word image. This is done by recording the appearance of the first ink pixel (on top boundary) and last ink pixel (on bottom boundary), and by finding their relative difference. Vertical distance is defined as follows:

$$F_5 = (upbound - lowbound)/N \tag{C.6}$$

where *upbound* and *lowbound* represents the upper boundary and lower boundary of a word image, respectively.

C.3 Moments

The evaluation of moments represent a systematic method of shape analysis, where each moment order has different information for the same image [65]. We consider here statistical moments and region-based moments.

Statistical Moments: We consider here such moments as mean, standard deviation and skew.

Mean (F_6) is used to compute the average number of foreground pixels, out of the total pixels per column in a given word image. It is the first order statistical moment that measures the central tendency of ink pixels in word images. Mean is defined as:

$$F_6 = \frac{1}{N}\sum_{x=0}^{N-1} w(x,y) \tag{C.7}$$

Standard deviation (F_7), on the other hand, is the sum of squared deviation of pixels from the mean. It is the second order statistical moment about the mean which measures the spread of ink pixels in a strip of a word image. Standard deviation is given by:

$$F_7 = \frac{1}{N}\sum_{x=0}^{N-1} (w(x,y) - \mu)^2 \tag{C.8}$$

Skew (F_8) is the sum of cubed deviation of pixels from the mean. It is the third order statistical moment about the mean. Skewness measures the degree of symmetry of pixel distributions in the given word image. It is defined as follows.

$$F_8 = \frac{1}{N\sigma_y^3}\sum_{x=0}^{N-1} (w(x,y) - \mu)^3 \tag{C.9}$$

Region-based Moments: The most commonly used region attributes are calculated from the following three region-based moments: m_{00}, m_{01} and m_{10}. Region-based moment representations use the entire shape region by describing the considered region using its internal characteristics, i.e. the pixels contained in that region. For an image $w(x, y)$, region based moment is defines as follows.

$$m_{i,j} = \sum_{x=0}^{N-1} \sum_{y=1}^{M} x^i y^j w(x, y) \tag{C.10}$$

We are interested here with the zeroth-order moment, m_{00}, (F_9) and first order moment, m_{01}, (F_{10}). The zeroth-order moment, m_{00} counts the number of black pixels in a vertical strip of a word image, out of the total foreground pixels. It gives a measure of the area of the shape. It is defined as:

$$F_9 = \sum_{x=0}^{N-1} w(x, y) / \sum_{x=0}^{N-1} x \tag{C.11}$$

The first-order moment with respect to y axis, m_{01}, is used to locate centroid of the image. It is given by:

$$F_{10} = \sum_{x=0}^{N-1} y * w(x, y) / \sum_{x=0}^{N-1} x \tag{C.12}$$

The knowledge of these moments can be used, among others, to calculate central moments as shown below.

Central Moments: Central moments are moments around the mean of word images. We can use these moments to provide useful descriptors of shape. Central moments are calculated by shifting the origin to the center of the image (μ_x, μ_y) as defined below:

$$cm_{i,j} = \sum_{x=0}^{N-1} (x^i - \mu_x)(y^j - \mu_y) w(x, y) \tag{C.13}$$

with the mean $\mu_x = m_{10}/m_{00}$ and $\mu_y = m_{01}/m_{00}$. The normalized central moments are then computed by dividing the result with $\sum_{x=0}^{N-1} w(x, y)$. We experiment with central moment cm_{02} (F_{11}) that computes squared distance from the y component of the mean μ_y.

$$F_{11} = \sum_{x=0}^{N-1} (y^2 - \mu_y) w(x, y) \tag{C.14}$$

C.4 Transform Domain Representation

A compact representation of a series of observations (such as profiles) is possible in a transform domain. Fewer set of coefficients are enough to represent robustly in a transformed domain, and these coefficients are matched at a coarse level for recognition and retrieval.

A number of transform domain representations are reported in literature [65], such as Fourier transform (FT), wavelet transform (WT), etc. Wavelet transform separates data into different frequency components, and then studies each component with resolution matched to its scale. The Fourier transform converts the image information from spatial domain to frequency domain to be represented in a more compact form. The Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT) are the two main types of the Fourier transform often used in applications [146]. DCT is a special case of Fourier transform in which the sine components have been eliminated leaving only the cosine terms.

We use Discrete Fourier descriptors (F_{12}) for word representation. We compute Fourier descriptors for each vertical strips of the word image in a one dimensional space, $w(x), x = 0, 1, \ldots, N-1$. They are attractive for use with boundary matching. The main idea of Fourier Descriptors is to use the Fourier transformed boundary as the shape feature [65]. It describes the shape of an input word image in the frequency domain. For a given shape signature, the coefficients are defined as follows:

$$F_{12} = \frac{1}{N} \sum_{x=0}^{N-1} e^{-j\pi xi/N} w(x, y) \tag{C.15}$$

Appendix D

Matching Word Images

D.1 Overview

Matching can be defined as the establishment of the correspondence between objects (i.e. queried one against those in the datasets). Good matching performance can be achieved by a technique that aligns and finds the best match between pairs of queried and referenced word images with respect to each other [155].

Defining a similarity measure for feature based matching is more complicated [61]. The definition must be based on the attributes of the features. In most feature based matching algorithms heuristics and thresholds are used in order to compute the similarity measure, called a cost function or benefit function. While a cost function is to be minimized, a benefit function must be maximized in order to achieve a good matching score. In our work, we test two matching algorithms, such as cross correlation and dynamic time warping. These matching algorithms have a great advantage in aligning and comparing word images having size variations.

D.2 Matching using Dynamic Time Warping

Time series are a ubiquitous form of data occurring in virtually every scientific discipline [163]. A common task with time series data is comparing one sequence with another. In some domains a very simple distance measure, such as Euclidean distance will suffice. However, it is often the case that two sequences (e.g. identical word images) have the approximately the same overall component shapes, but these shapes do not line up in X-axis. This is caused by factors like degradations, printing variations, etc. as shown in Figure D.1. In order to find the similarity between such sequences, it is necessary to match sequences with tolerance of small local misalignments and

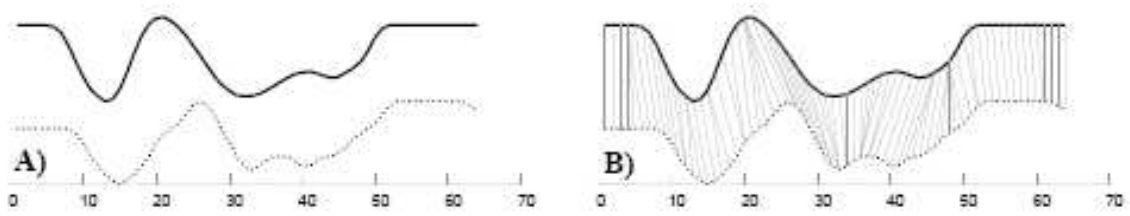


Figure D.1: An example of how two sequences (extracted from two word images) are aligned using dynamic time warping. (A) Sequences extracted using word profile. Note that, while the sequences have an overall similar shape, they are not aligned in the time axis. (B) DTW finds an alignment between the two sequences.

warp the time axis of one (or both) sequences to achieve a better alignment. Dynamic time warping (DTW), is a technique for efficiently achieving this warping [155]. For example, in speech comparison, small fluctuation of the tempo of the speakers should be allowed in order to identify similar contents.

The use of dynamic programming based DTW for dissimilarity measure has the following advantages [62][90]. (i) It preserves the ordering information by enforcing a local continuity constraint. (ii) It minimizes a cumulative distance consisting of available local distances. (iii) It can easily handle features of different lengths. (iv) Considerable speed-up can be achieved by constraining the search space. (v) Covariance and other statistical measures are not required over the entire collection, (vi) It is quickly computable - O(m, n), where m, n are feature lengths.

The last few years have seen an increasing understanding that Dynamic Time Warping (DTW), a technique that allows local flexibility in aligning time series, is superior to the ubiquitous Euclidean Distance for time series classification, clustering, and indexing [62]. Because of the robustness of DTW for similarity measurement, it has been widely used in the area of speech processing [152], robotics [167], bioinformatics [1], handwritten documents retrieval [80][155], data mining [91], etc.

Let the word images (say their sequential representation) with length M and N are represented as a sequence of features $\mathcal{F} = \mathbf{F_1}, \mathbf{F_2}, \ldots \mathbf{F_M}$ and $\mathcal{G} = \mathbf{G_1}, \mathbf{G_2}, \ldots, \mathbf{G_N}$. The DTW-cost between these two sequences is calculated using the recurrence equation of dynamic programming given in Section 6.4.2 The DTW algorithm for computing the optimal cost of matching is given in Algorithm 11.

Once the optimal cost is obtained, the optimal warping path between the two words, F and G is generated by backtracking the DTW minimal score in the matching space. The algorithm for backtracking the optimal warping path is given in

Algorithm 12. The optimal warping path with length L has coordinates in the range $((i_0, j_0), (i_1, j_1), \ldots, (i_k, j_k))$ that runs from $D(0,0)$ to $D(M,N)$. This path is the cheapest one with minimum distance among exponentially available many warping paths in the search space.

Score for matching the two sequences $\mathcal{F}$ and $\mathcal{G}$ is considered as $D(M,N)$, where M and N are the lengths of the two sequences.

Algorithm 11 Algorithm for computing the Optimal cost of matching

1: $d(i,j) = \sum_k \left(F_i^k - G_j^k\right)^2$;
2: $D(i,j) = d(i,j)$
3: **for** $i = 1$ to N **do**
4: $\quad D(i,0) = D(i-1,0) + d(i,0)$
5: **end for**
6: **for** $j = 1$ to M **do**
7: $\quad D(0,j) = D(0,j-1) + d(0,j)$
8: **end for**
9: **for** $i = 1$ to N **do**
10: $\quad$ **for** $j = 1$ to M **do**
11: $\quad\quad$ vertical$(D(i-1,j) + d(i,j)$
12: $\quad\quad$ diagonal$(D(i-1,j-1) + d(i,j)$
13: $\quad\quad$ horizontal$(D(i,j-1) + d(i,j)$
14: $\quad\quad$ min(vertical, diagonal, horizontal)
15: $\quad$ **end for**
16: **end for**
17: **Output:** Optimal Matching Cost

Although DTW has been successfully used in many domains, it is expensive to calculate as the time complexity is $O(M,N)$. In addition it can produce pathological results. The algorithm may try to explain variability in the Y-axis by warping the X-axis. This can lead to unintuitive alignments where a single point on one time series maps onto a large subsection of another time As a result there is a need to introduce techniques to speed up similarity search.

The use of global constraint speed up the calculations of optimal matching cost by preventing pathological warping. Two commonly used global constraints exist: *Sakoe - Chiba band* [163] and *Itakura Parallelogram* [78]. The Sakoe-Chiba Band limits the warping path to a band enclosed by two straight lines that are parallel to the

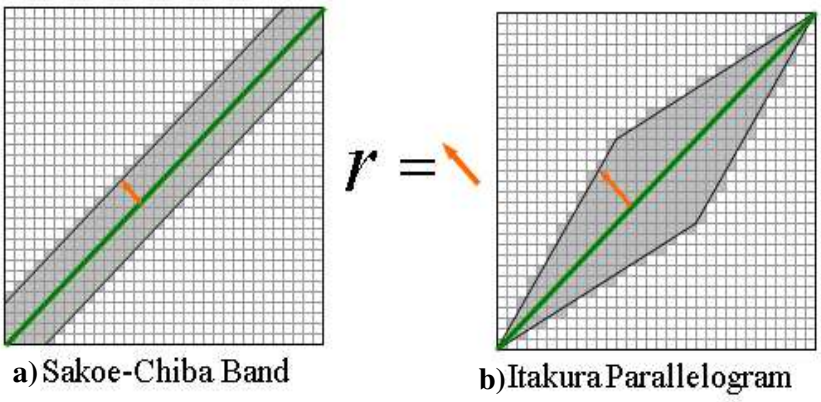


Figure D.2: The two global constraints: Sakoe - Chiba band and Itakura Parallelogram

diagonal of the warping matrix. The Itakura Parallelogram limits the warping path to be within a parallelogram whose major diagonal is the diagonal of the warping matrix. This is shown in Figure D.2.

Algorithm 12 Backtracking the Optimum DTW Warping Path

1: **while** i and $j >= 0$ **do**
2: $backtrack(D(i-1,j), D(i-1,j-1), D(i,j-1)$
3: $case1$:
4: i is unmatched; i = i - 1;
5: $case2$:
6: i and j matched i = i - 1; j = j - 1;
7: $case3$:
8: j is unmatched; j = j - 1;
9: end case
10: **end while**
11: **Output:** Optimal warping path

Keogh [90] views a global constraint as a constraints on the indices of the warping path entry $W_k = (i,j)_k$ and gives a general framework of global constraints in terms of inequalities on the indices to the elements in the warping matrix such that:

$$j - r \leq i \leq j + r \tag{D.1}$$

where r is the allowed range of warping for a given point in a sequence. While r is

constant for the Sakoe-Chiba Band, it is a function of i for the Itakura Parallelogram. For our case the use of *Sakoe - Chiba band* performs well than *Itakura Parallelogram* since it gives equal range r of warping in all the given $M \times N$ matching space. So we use *Sakoe - Chiba band* to constrain the maximum steepness or flatness of the DTW path. By incorporating the Sakoe-Chiba band constraint into the definition of dynamic time warping distance, every non-diagonal steps in a X or Y direction is constrained by some value (say 3) and diagonal steps in the diagonal direction is set to, say 2. With this constraints used, the above algorithm (11) can be modified by the Algorithm 13.

Algorithm 13 Algorithm for computing the Optimal cost of matching using the selected DTW satisfying constraints within a Sakoe-Chiba band

$i = j = 0$;
$d(i,j) = \sum_k \left(F_i^k - G_j^k\right)^2$;
$D(i,j) = 2d(i,j)$
for $i = 1$ to N **do**
  $D(i,0) = D(i-1,0) + 3d(i,0)$
end for
for $j = 1$ to M **do**
  $D(0,j) = D(0,j-1) + 3d(0,j)$
end for
for $i = 1$ to N **do**
  for $j = 1$ to M **do**
    vertical($D(i-2,j-1) + 3d(i,j)$)
    diagonal($D(i-1,j-1) + 2d(i,j)$)
    horizontal($D(i,j-2) + 3d(i,j)$)
    min(vertical, diagonal, horizontal)
  end for
end for
Output: Optimal Matching Cost

D.3 Cross Correlation

Cross correlation is a statistical measure of how closely two signals are related [49]. It has a wider applications in area-based matching algorithms [83] that often employ

correlation scores as the matching measure between pixels or regions of the image. The correlation function compares sequences of feature vectors of two word images by aligning them against each other, and at each position the correlation coefficient between the corresponding pairs of feature vectors of the two images is computed. Consider feature vectors $\mathcal{F} = \mathbf{F_1}, \mathbf{F_2}, \ldots, \mathbf{F_M}$ and $\mathcal{G} = \mathbf{G_1}, \mathbf{G_2}, \ldots, \mathbf{G_N}$. The comparison between these two sequences of vectors produces an alignment vector W of length $l = m + n - 1$ as defined below:

$$W_i = \begin{cases} \sum_{j=0}^{i} (D(F_j, G_k))\ if\ i = 0...m-1 \\ \sum_{j=0}^{n-1} (D(F_j, G_k))\ if\ i = m...n-1, m \neq n \\ \sum_{j=0}^{(l-1)-i} (D(F_t, G_j))\ if\ i = n...l-1 \end{cases} \tag{D.2}$$

where, k is $(n-1) - i + j$ and t is $(n-1) - (l-1-i) + 1$.

$D(i, j)$ is defined as:

$$D(i,j) = \begin{cases} 1\ if\ i = j \\ 0\ if\ i \neq j \end{cases}$$

Optimal matching score of the two aligned word images are obtained from their alignment vector. This gives us a dissimilarity measure of the match. Thus, for decision purpose the maximum of the resulting correlation values are taken as the best match. Correlation is therefore a similarity measure that selects a match that maximizes benefits.

Appendix E

Feature Selection

E.1 Feature Selection Procedure

Feature selection enables us to explore the usefulness of selecting subsets of features that together have good predictive power [72]. We use feature selection for combining features that improve the performance of word image matching for document image retrieval. Among the available approaches, sequential feature selection algorithms receive great attention [4]. There are forward and backward sequential feature selection algorithms. Backward feature selection begins with all features and repeatedly removes a feature whose removal yields the best performance improvement. Whereas, forward feature selection begins with the evaluation of all features with exactly one feature at a time, and selects the one with the best performance. It then evaluates again by combining the best performer feature with the remaining features and selects combined features with the best performance. This cycle repeats until no improvement is obtained from extending the combined run. The later method expresses how much the next feature brings new information to the preceding collection of features. We use forward feature selection approach in evaluating the above features because of its computational efficiency [72].

We follow a step-wise procedure in evaluating the performance of features based on forward feature selection approach. First, we test matching performance of individual features, and then, performance of combined run of best feature with the remaining ones , as described below.

Step 1: Each feature is tested individually. This step comes-up with one best performer feature with the highest F-score. This feature is combined with the other features in the next round.

Step 2: In second step combined features of two are tested. This is done by combining the best performer in the first step with each of the other features. This step result in a feature that perform best when combined with other feature.

Step 3: This step further test combined run of the best two features (selected in the previous two step) with each of the remaining ones. In this way the performance of features is tested by combining one best performer feature at a time.

Step 4: Combining and testing procedure stops when adding new feature to other combined features result in no performance improvement.

A modified DTW algorithm is implemented for measuring the performance of the combined features at each step. For the experiment we use datasets prepared in English, Hindi and Amharic languages.

Features	English			Hindi			Amharic		
	Rec	Pre	Fsc	Rec	Pre	Fsc	Rec	Pre	Fsc
F_1	88.91	77.98	79.87	74.14	76.64	75.04	84.68	83.12	82.36
F_2	84.92	84.53	81.22	56.72	39.05	43.43	80.65	32.34	43.25
F_3	84.04	73.88	75.33	32.01	33.58	32.78	70.43	78.36	72.81
F_4	50.55	66.88	56.71	88.18	67.15	74.79	83.87	89.85	86.45
F_5	83.37	76.22	78.68	87.60	82.53	84.14	75.27	83.82	76.46
F_6	88.91	77.98	79.87	74.14	76.64	75.04	84.68	83.12	82.36
F_7	75.83	80.00	77.33	38.72	23.02	27.73	55.91	61.69	54.29
F_8	57.21	68.80	61.11	15.62	19.86	16.82	46.50	63.67	51.28
F_9	90.69	76.05	79.06	75.78	76.44	75.61	72.58	58.94	62.53
F_{10}	89.14	56.23	63.65	80.23	84.17	81.51	74.73	46.00	54.99
F_{11}	54.37	25.68	31.85	44.37	24.44	31.52	33.28	12.13	17.78
F_{13}	65.41	78.80	70.55	51.49	53.24	51.80	66.67	74.07	68.71

Table E.1: Performance of local features for English, Hindi and Amharic languages. DTW is used for matching sequences of feature vectors extracted from vertical strips of word images. N. B. Rec = Recall, Pre = Precision & Fsc = F-Score

Experimental Result: We undertake extensive experiment to select the most promising features for representation of word images. Performance of individual features for the three languages (English, Hindi and Amharic) is shown in Table E.1.

Among individual features, on the average, vertical distance registers the highest F-score of 79.76%. Projection profile and mean comes next with 79.09% each. Their relative good level of performance is attributed to the relative high level of information contained by them. In general performance of individual features is very poor. We then combined the best performing feature with others to see if there is performance improvement. The cycle repeats until there is not further performance improvement.

English				Hindi				Amharic			
F_e	Rec	Pre	Fsc	F_h	Rec	Pre	Fsc	F_a	Rec	Pre	Fsc
F_1	87.58	88.87	86.48	F_1	87.56	91.16	89.12	F_1	89.63	91.01	90.15
F_3	87.58	91.80	88.56	F_3	86.81	93.05	89.67	F_2	91.01	93.61	92.24
F_5	88.47	91.96	88.75	F_4	90.79	92.83	91.79	F_5	85.02	92.63	88.19
F_6	87.58	88.87	86.48	F_6	87.49	91.10	89.06	F_6	90.81	90.73	90.67
F_7	89.58	90.81	90.05	F_7	89.49	91.87	90.61	F_7	79.20	83.65	80.21
F_8	68.52	85.24	74.61	F_8	91.88	55.47	56.31	F_8	83.33	91.59	86.50
F_{10}	89.58	81.85	82.36	F_{10}	87.51	91.48	89.18	F_{10}	87.46	87.42	87.23
F_{11}	86.92	91.99	88.23	F_{11}	82.93	91.71	86.57	F_{11}	89.78	93.74	91.52
F_{12}	83.59	92.29	86.88	F_{12}	76.89	72.63	73.39	F_{12}	75.21	83.45	78.16

Table E.2: Performance results of combined local features for English, Hindi and Amharic languages. DTW is used for matching sequences of feature vectors extracted from vertical strips of word images. N. B. Rec = Recall, Pre = Precision & Fsc = F-Score, F_e - combined Features of F_2, F_4, F_9 and the stated feature column-wise, F_h - combined Features of F_2, F_9, F_5 and the stated feature column-wise, F_a - combined Features of F_3, F_4, F_9 and the stated feature column-wise

Table E.1 shows the final performance results of combined local features for the three languages. Among the combined runs, highest F-score is obtained by the following features. On English datasets, background to ink transition, moment M_{00}, lower profile and standard deviation register 90.05% F-score. On Hindi, background to ink transition, lower profile, moment M_{00} and vertical distance result in 91.79% F-score, and, on Amharic background to ink transition, lower profile, moment M_{00} and upper profile register 92.24% F-score. Thus, more than 90% F-score is obtained by combining features as compared to individual features whose best result is not more than 80%. From this we can conclude that combining features result in better representation of a word image.

Appendix F

Publications Related to This Work

1. **Million Meshesha** and C. V. Jawahar, "Matching Word Images for Content-based Retrieval from Printed Document Images", Int. Journal of Document Analysis and Recognition (IJDAR), 2008.

2. C. V. Jawahar, A. Balasubrahmanian, **Million Meshesha** and Anoop Namboodiri, "Retrieval of Online Handwriting by Synthesis and Matching", Pattern Recognition, 2008.

3. **Million Meshesha** and C. V. Jawahar, "Optical Character Recognition of Amharic Documents", African Journal of Information and Communication Technology", Vol 3, No 2, pp. 53 - 66, June 2007.

4. **Million Meshesha** and C. V. Jawahar, "Indexing Word Images for Recognition-free Retrieval from Printed Document Databases", Information Sciences (revised and submitted).

5. **Million Meshesha** and C. V. Jawahar, "Indigenous Scripts of African Languages", African Journal of Indigenous Knowledge Systems, Vol 6, No 2, pp. 132 - 142, 2007.

6. **Million Meshesha** and C. V. Jawahar, "Self-Adaptable Recognizer for Document Image Collections", In Proc. of Int. Conf. on Pattern Recognition and Machine Intelligence (LNCS), pp. 560-567, 2007.

7. A. Balasubrahmanian, **Million Meshesha**, C. V. Jawahar, "Retrieval from Document Image Collections", In Proc. of the 7th IAPR Workshop on Document Analysis Systems (DAS) (LNCS), pp 1-12, 2006.

8. Sachin Rawat, K. S. Sesh Kumar, **Million Meshesha**, Indiraneel Deb Sikdar, A. Balasubramanian and C. V. Jawahar, "Semi-automatic Adaptive OCR for Digital Libraries", In Proc. of the 7th IAPR Workshop on Document Analysis Systems (DAS) (LNCS), pp 13-24, 2006.

9. K. Pramod Sankar, **Million Meshesha**, C. V. Jawahar, "Annotation of Images and Videos based on Textual Content without OCR", In Workshop on Computation Intensive Methods for Computer Vision, 9th European Conf. on Computer Vision (ECCV), 2006.

10. **Million Meshesha** and C. V. Jawahar, "Recognition of Printed Amharic Documents", In Proc. of the 8th IEEE Int. Conf. of Document Analysis and Recognition (ICDAR), pp 784-788, 2005.

11. C. V. Jawahar, **Million Meshesha** and A. Balasubrahmanian, "Searching in Document Images", In Proc. of the 4th Indian Conf. on Computer Vision, Graphics and Image Processing (ICVGIP), pp. 622-627, 2004.

Bibliography

[1] J. Aach and G. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.

[2] A.Balasubramanian and C.V.Jawahar. Textual search in graphics stream of pdf. In *Int. Conf. on Digital Libraries (ICDL)*, 2006.

[3] M. Abrams. *World Wide Web - Beyond the Basics*. Prentice Hall, Inc, New Jersey, 1998.

[4] D. W. Aha and R. L. Bankert. A comparative evaluation of sequential feature selection algorithms. In *Proc. of the 5th Int. Workshop on Artificial Intelligence and Statistics*, pages 1–7, 1995.

[5] Worku Alemu and S. Fuchs. Handwritten Amharic bank check recognition using hidden markov random field. In *Document Image Analysis and Retrieval Workshop (DIAR)*, page 28, 2003.

[6] The Open Content Alliance. at `http://www.opencontentalliance.org`, 2005.

[7] V. Ambati, N. Balakrishnan, R. Reddy, L. Hari, and C.V. Jawahar. The digital library of india project: Process, policies and architecture. In *Int. Conf. on Digital Libraries (ICDL)*, 2006.

[8] A. Amin. Off-line Arabic characters recognition: The state of art. *Pattern Recognition*, 31(5):517–530, 1998.

[9] S. Antanani and L. Agnihotri. Gujarati character recognition. *Proc. of the 5th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 418–421, 1999.

[10] K. H. Aparna and V. S. Chakravarthy. A complete OCR system development of Tamil magazine documents. In *Tamil Internet, Chennai, Tamilnadu, India*, 2003.

[11] AramediA. at `http://aramedia.com/`, 2007.

[12] Internet Archive. at `http://www.archive.org`.

[13] T. V. Ashwin and P. S. Sastry. A font and size independent OCR system for printed Kannada documents using support vector machines. *Special Issue of Sadhana*, 27:35–58, 2002.

[14] A. J. Aslam, E. Pelekhov, and D. Rus. The star clustering algorithm for static and dynamic information organization. *Journal of Graph Algorithms and Applications*, 8(1):95–129, 2004.

[15] E. Ataer and P. Duygulu. Retrieval of Ottoman documents. In *Proc. of the 8th ACM Int. workshop on Multimedia information retrieval*, pages 155 – 162, 2006.

[16] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, MA, 1999.

[17] D. Bainbridge, J. Thompson, and H. I. Witten. Assembling and enriching digital library collections. In *Proc. of Joint Conf. on Digital libraries*, pages 323–334, 2003.

[18] H. S. Baird. Anatomy of a versatile page reader. *Proc. of the IEEE*, 80(7):1059–1065, 1992.

[19] H. S. Baird. State of the art of document image degradation modeling. In *Proc. of the 4th IAPR Workshop Document Analysis Systems (DAS)*, pages 1–16, 2000.

[20] H. S. Baird. Digital libraries and document image analysis. In *Proc. of the 7th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 2–14, 2003.

[21] H. S. Baird, V. Govindaraju, and D. Lopresti. Document analysis systems architectures for digital libraries. In *Proc. of Int. Assoc. for Pattern Recognition Workshop on Document Analysis Systems*, 2004.

[22] R. Bajaj, L. Dey, and S. Chaudhury. Devanagari numeral recognition by combining decision of multiple connectionist classifier. *Special Issue of Sadhana*, 27:59–72, 2002.

[23] A. Balasubramanian. *Document Annotation and Retrieval System*. M. Sc. Thesis, Int. Institute of Information Technology, Hyderabad, India, 2006.

[24] A. Balasubramanian, Million Meshesha, and C. V. Jawahar. Retrieval from document image collections. In *Proc. of the 7th Int. Assoc. for Pattern Recognition Workshop on Document Analysis Systems (DAS)*, pages 1–12, 2006.

[25] V. Bansal and R. M. K. Sinha. A Devanagari OCR and a brief overview of OCR research for Indian scripts. In *Proc. of the Symposium on Translation Support Systems*, 2000.

[26] V. Bansal and R. M. K. Sinha. Integrating knowledge sources in Devanagari text recognition system. *IEEE Trans. on Systems, Man, & Cybernetics*, 30(4):500–505, 2000.

[27] V. Bansal and R. M. K. Sinha. A complete OCR for printed Hindi text in Devanagari script. In *Proc. of the 6th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 800–804, 2001.

[28] M.L. Bender. *Language in Ethiopia*. Oxford University Press, London, 1976.

[29] Girmay Berhane. Word formation in Amharic. *Journal of Ethiopian Languages and Literature*, 2(507-2), 1992.

[30] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Workshop on Knowledge Discovery in Databases (KDD)*, pages 359–370, 1994.

[31] A. Bhaskarbhatla, S. Madhavanath, M. Pavan Kumar, A. Balasubramanian, and C. V. Jawahar. Representation and annotation of online handwritten data. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition(IWFHR)*, pages 136–141, 2004.

[32] M. Bokser. Omnidocument technologies. *Proc. of the IEEE*, 80(7):1066–1078, 1992.

[33] C. Breiteneder and H. Eidenberger. Content-based image retrieval in digital libraries. In *Proc. of the Int. Conf. on Digital Libraries: Research and Practice*, pages 67–74, 2000.

[34] J. C. Burges. A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery*, 1998.

[35] F. Can. Incremental clustering for dynamic information processing. *ACM Trans. on Information Systems*, 10(2):143–164, 1993.

[36] Tejo Krishna Chalasani, Anoop M. Namboodiri, and C. V. Jawahar. Support vector machine based hierachical classifiers for large class problems. In *Proc. of the 6th Int. Conf. on Advances in Pattern Recognition (ICAPR)*, 2007.

[37] J. Chan, C. Ziftci, and D.A. Forsyth. Searching off-line Arabic documents. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1455–1462, 2006.

[38] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proc. of the 29th Annual ACM Symposium on Theory of Computing*, pages 626–635, 1997.

[39] B. B. Chaudhuri. On OCR of a printed Indian script. In *Digital Document Processing: Major Directions and Recent Advances, B. B. Chaudhuri (ed.), Springer-Verlag London Ltd*, pages 99–119, 2007.

[40] B. B. Chaudhuri and U. Pal. A complete printed Bangla OCR system. *Pattern Recognition*, 31(5):531–549, 1997.

[41] B. B. Chaudhuri and U Pal. An OCR system to read two Indian language scripts: Bangla and Devanagari (Hindi). *Proc. of the 4th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 1011–1015, 1997.

[42] B. B. Chaudhuri, U. Pal, and M. Mitra. Automatic recognition of printed Oriya script. In *Proc. of the 6th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 795–799, 2001.

[43] S. Chaudhury, G. Sethi, A. Vyas, and G. Harit. Devising interactive access techniques for Indian language document images. In *Proc. of the 7th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 885–889, 2003.

[44] Microsoft Corporation. at `http://msdn2.microsoft.com`, 2007.

[45] NewSoft Technology Corporation. at `http://www.newsoftinc.com/`, 2007.

[46] J. Cowell and F. Hussain. Amharic character recognition using a fast signature based algorithm. In *Proc. of the 7th Int. Conf. on Information Visualization*, page 384, 2003.

[47] C.V.Jawahar and A. Balasubramanian. Synthesis of online handwriting in Indian languages. In *Int. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 2006.

[48] R. Datta, J. Li, and J. Z. Wang. Content-based image retrieval - Approaches and trends of the new age. In *Proc. of the 7th Int. Workshop on Multimedia Information Retrieval*, pages 253–262, 2005.

[49] N. Devillard. Infrared jitter imaging data reduction algorithms. In *The Astronomical Society of the Pacific (ASP) Conf. Series*, pages 172–333, 1999.

[50] J. Dholakia, Atul Negi, and S. Rama Mohan. Zone identification in the printed Gujarati text. In *Proc. of the 8th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 272– 276, 2005.

[51] C. Diehl and G. Cauwenberghs. SVM incremental learning, adaptation and optimization. In *Prioc. IEEE Int. Joint Conf. Neural Networks (IJCNN)*, pages 2685–2690, 2003.

[52] Digital Library Initiatives Project. at `http://www. dli2.nsf.gov`, 2003.

[53] D. Doermann. The indexing and retrieval of document images: A survey. *Computer Vision and Image Understanding (CVIU)*, 70(3):287–298, 1998.

[54] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Willey & Sons, New York, 2001.

[55] A. Dutta and S. Chaudhury. Bengali alpha-numeric character-recognition using curvature features. *Pattern Recognition*, 26(12):1757–1770, 1993.

[56] Microsoft Encarta Online Encyclopedia. African languages. at `http://uk. encarta.msn.com`, 2006.

[57] The Columbia Encyclopedia. Africa. 2001. 6th ed., at `http://www.bartleby. com/65/af/ Africa.html`.

[58] Ethnologue. Languages of the World, 14th ed. at `http://www.ethnologue. com/`, 2006.

[59] S. Feng and R. Manmatha. A hierarchical, HMM-based automatic evaluation of OCR accuracy for a digital library of books. In *Joint Conf. on Digital Libraries (JCDL)*, pages 109–118, 2006.

[60] D. H. Foley and J. W. Sammon. An optimal set of discriminant vectors. *IEEE Trans. on Computing*, 24:271–278, 1975.

[61] W. Forstner. A feature based correspondence algorithm for image matching. *Int. Archives of Photogrammetry and Remote Sensing*, 26(3/3):150–166, 1986.

[62] A. W. Fu, E. Keogh, L. Y. H. Lau, and C. A. Ratanamahatana. Scaling and time warping in time series querying. In *Proc. of the 31st Very Large Data Base (VLDB) Conf.*, 2005.

[63] A. M. Gillies, E. J. Erlandson, J. M Trenkle, and S. G. Schlosser. Arabic text recognition system. In *Proc. of the Symposium on Document Image Understanding*, 1999.

[64] GOCR. at `http://jocr.sf.net/`, 2007.

[65] W. Gonzalez. *Digital Image Processing*. Addison Wesley, Massachusetts, 1992.

[66] Google. at `http://code.google.com/p/tesseract-ocr/`, 2007.

[67] P. J. Gray. *Automatic Data Processing Handbook*. McGraw-Hill Book Company, Newyork, 1977.

[68] Ed Greengrass. Information retrieval: A survey. at `"http://www.csee.umbc.edu/cadip/ readings`, 2000.

[69] Daniel I. Greenstein and Suzanne Elizabeth Thorin. *The Digital Library: A Biography*. Digital Library Federation, 2002.

[70] Greenstone. Greenstone digital library software. at `http://www.greenstone.org/`, 2007.

[71] Project Gutenberg. at `http://www.guten berg.org`, 2003.

[72] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(7-8):1157–1182, 2003.

[73] K. M. Hammouda and M. S. Kamel. Incremental document clustering using cluster similarity histograms. In *The IEEE/WIC Int. Conf. on Web Intelligence (WI)*, pages 597–601, 2003.

[74] R. T. Hartley and K. Crumpton. Quality of OCR for degraded text images. In *Proc. of the 4th ACM Conf. on Digital libraries*, pages 228–229, 1999.

[75] T. Hastie and R. Tibshirani. Classification by pairwise coupling. In *Proc. of the Conf. on Advances in Neural Information Processing Systems*, volume 10, pages 507–513, 1998.

[76] P. B. Heidorn. Natural language processing of visual language for image storage and retrieval. *Doctoral Dissertation, University of Pittsburgh*, 1997.

[77] Indian Language Transliteration. at: `http://www.cs.cmu.edu/~madhavi/Om/`.

[78] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 23(1):67–72, 1975.

[79] V. S. Iyengar, C. Apte, and T. Zhang. Active learning using adaptive resampling. In *Proc. of the 6th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 92–98, 2000.

[80] A. K. Jain and A. M. Namboodiri. Indexing and retrieval of on-line handwritten documents. In *Proc. of the 7th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 655–659, 2003.

[81] C. V. Jawahar, M. N. S. S. K. Pavan Kumar, and S. S. Ravi Kiran. A bilingual OCR for Hindi-Telugu documents and its applications. In *Proc. of the Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 408–412, 2003.

[82] C. V. Jawahar, Million Meshesha, and A. Balasubramanian. Searching in document images. *4th Indian Conf. on Computer Vision, Graphics and Image Processing (ICVGIP)*, pages 622–627, 2004.

[83] C. V. Jawahar and P. J. Narayanan. Generalised correlation for multi-feature correspondence. *Pattern Recognition Letters*, 35:1303–1313, 2002.

[84] K. Jithesh, K. G. Sulochana, and R. R. Kumar. Optical character recognition (OCR) system for Malayalam language. In *National Workshop on Application of Language Technology in Indian Languages*, 2003.

[85] K. S. Jones. Language modeling's generative model: Is it rational? at `http://www.cl.cam.ac.uk/ ksj21/langmodnote 4.pdf`, 2004.

[86] S. Kahan, T. Pavlidis, and H. S. Baird. On the recognition of printed characters of any font and size. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 9(2):274–288, 1987.

[87] I. Kamel. Fast retrieval of cursive handwriting. In *Proc. of the 5th Int. Conf. on Information and Knowledge Management*, pages 91–98, 1996.

[88] J. Kamps, C. Monz, M. Rijke, and B. Sigurbjrnsson. Language-dependent and language-independent approaches to cross-lingual text retrieval. In *Proc. of the 4th Workshop of the Cross-Language Evaluation Forum (CLEF)*, pages 152–165, 2003.

[89] T. Kanungo, R. M. Haralick, and I. T. Phillips. Global and local document degradation models. In *Proc of 2nd Int. Conf. on Document Analysis and Recognition*, pages 730–734, 1993.

[90] E. Keogh. Exact indexing of dynamic time warping. In *Proc. of 28th Int. Conf. on Very Large Data Bases*, pages 406–417, 2002.

[91] E. Keogh and N. Pazzani. Scaling up dynamic time warping for data mining applications. In *Proc. of the 6th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 285–289, 2000.

[92] L. Keyes and A.C. Winstanley. Fourier descriptors as a general classification tool for topographic shapes. In *Proc. of the Irish Machine Vision and Image Processing Conf.*, pages 193–203, 1999.

[93] M. S. Khorsheed. Off-line Arabic character recognition - A review. *Pattern Analysis & Applications*, 5(1):31–45, 2002.

[94] F. Kimura. OCR technologies for machine printed and hand printed Japanase text. In *Digital Document Processing: Major Directions and Recent Advances, B. B. Chaudhuri (ed.), Springer-Verlag London Ltd*, pages 49–71, 2007.

[95] S. Klick, K. Kise, A. Dengel, M. Junker, and S. Agne. Document information retrieval. In *Digital Document Processing: Major Directions and Recent Advances, B. B. Chaudhuri (ed.), Springer-Verlag London Ltd*, pages 351–378, 2007.

[96] T. Konidaris, B. Gatos, K. Ntzios, I. Pratikakis, S. Theodoridis, and S. J. Perantonis. Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *Int. Journal on Document Analysis and Recognition*, 9(2):167 – 177, April 2007.

[97] R. R. Korfhage. *Information Storage and Retrieval*. John Wiley, New York, 1997.

[98] A. Kumar, C. V. Jawahar, and R. Manmatha. Efficient search in document image collections. In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, 2007.

[99] Anand Kumar, A. Balasubramanian, Anoop M. Namboodiri, and C.V. Jawahar. Model-based annotation of online handwritten datasets. In *Int. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 2006.

[100] K. S. Sesh Kumar, A. M. Namboodiri, and C. V. Jawahar. Learning to segment document images. In *Int. Conf. on Pattern Recognition and Machine Intelligence*, 2005.

[101] M. N. S. S. K. Pavan Kumar and C. V. Jawahar. Design of hierarchical classifier with hybrid architectures. In *Proc. of 1st Int. Conf. on Pattern Recognition and Machine Intelligence (PReMI)*, pages 276–279, 2005.

[102] R. C. Kurzweil. *Kurzweil reading machine for the blind (users manual).* Kurzweil computers products, Cambridge, MA, 1990.

[103] C. Vasantha Lakshmi and C. Patvardhan. A complete multi-font OCR systems for printed Telugu text. In *Language Engineering Conf.*, 2002.

[104] V. Lavrenko, T. M. Rath, and R. Manmath. Holistic word recognition for handwritten historical documents. In *Proc. of 1st Int. Workshop on Document Image Analysis for Libraries*, pages 278– 287, 2004.

[105] G. S. Lehal and C. Singh. Printed Devnagari script OCR system. *Vivek*, 10(2):12–24, 1997.

[106] G. S. Lehal and C. Singh. Feature extraction and classification for OCR of Gurmukhi script. *Vivek*, 12(2):2–12, 1999.

[107] G. S. Lehal and C. Singh. A Gurmukhi script recognition system. In *Proc. of the 15th Int. Conf. on Pattern Recognition (ICPR)*, pages 557–560, 2000.

[108] G. S. Lehal and C. Singh. A complete OCR system for Gurmukhi script. In *Structural, Syntactic and Statistical Pattern Recognition, T. Caelli, A. Amin, R.P.W. Duin, M. Kamel and D. de Ridder (Eds.), Lecture Notes in Computer Science*, pages 344–352, 2002.

[109] G. S. Lehal and C. Singh. A post-processor for Gurmukhi OCR. *Special Issue of Sadhana*, 27:99–111, 2002.

[110] The Universal Library. at `http://ulib.org`.

[111] X. Lin. DRR research beyond COTS OCR software: A survey. In *SPIE Conf. on Document Recognition and Retrieval XII*, pages 16–20, 2005.

[112] Lawrnce Lo. at `http://www.ancientscripts.com/`, 2006.

[113] L. M. Lorigo and Venu Govindaraju. Offline Arabic handwriting recognition: A survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(5):712–724, 2006.

[114] Y. Lu and C. L. Tan. Information retrieval in document image databases. *IEEE Trans. on Knowledge and Data Engineering*, 16(11):1398–1410, 2004.

[115] S. Madhvanath and V. Govindaraju. The role of holistic paradigms in handwritten word recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 23(2):149–164, 2001.

[116] S. Mafundikwa. African alphabets, Harare, Zimbabwe. November 2000, at `http://www.ziva.org.zw/afrikan.htm`.

[117] V. N. Manjunath, P. S. Aradhyal, G. H. Kumar, and S. Noushathl. Fisher linear discriminant analysis based technique useful for efficient character recognition. In *Proc. of the 4th Int. Conf. on Intelligent Sensing and Information Processing*, pages 49–52, 2006.

[118] R. Manmatha and W. B. Croft. Word spotting: Indexing handwritten archives. In *Intelligent Multi-media Information Retrieval, Mark T. Maybury, Ed. MIT Press, Cambridge, MA*, pages 43–64, 1997.

[119] R. Manmatha, C. Han, E. M. Riseman, and W. B. Croft. Indexing handwriting using word matching. In *Proc. of the 1st ACM Internal Conf. on Digital Libraries*, pages 151–159, 1996.

[120] J. Mantas. An overview of character recognition methodologies. *Pattern Recognition*, 19(6):425–430, 1986.

[121] M. Martynov and B. Novikov. An indexing algorithm for text retrieval. In *Proc. of the Int. Workshop on Advances in Databases and Information Systems (ADBIS)*, page 171176, 1996.

[122] A. T. McCray and M. E. Gallagher. Principles for digital library development. *Communications of the ACM*, 44(5):48–54, 2001.

[123] Million Meshesha and C. V. Jawahar. Recognition of printed Amharic documents. In *Proc. of the 8th Int. Conf. on Document Analysis and Recognition ICDAR*, pages 784–788, 2005.

[124] Million Meshesha and C. V. Jawahar. Optical character recognition of Amharic documents. *African Journal of Information and Communication Technology*, 3(2):53–66, 2007.

[125] Lesk Michael. *Practical digital libraries: Books, bytes & bucks*. Morgan Kaufmann, San Francisco, CA, 1997.

[126] Microsoft. Live search. at `http://search.live.com`, 2007.

[127] Mike Noel and Wei Wei. Survey of the state of the art in human language technology. at `http://cslu.cse.ogi.edu/HLTsurvey/ indextop.html`.

[128] T. M. Mitchell. *Machine Learning*. New York: McGraw Hill, 1997.

[129] S. Mohanty and H. K. Behera. A complete OCR development system for Oriya script. In *Proc. of symposium on Indian Morphology, phonology and Language Engineering*, 2004.

[130] S. Mori, H. Nishida, and H. Yamada. *Optical Character Recognition*. John Wiley & Sons, Inc., New York, 1999.

[131] G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000.

[132] N. V. Neeba and C.V. Jawahar. Recognition of books by verication and retraining. In *The 9th Int. Conf. on Pattern Recognition (ICPR)*, 2008.

[133] H. Neemuchwala, A. Hero, and P. Carson. Image matching using alpha-entropy measures and entropic graphs. *Signal Processing*, 85(2):277–296, 2005.

[134] A. Negi, C. Bhagvathi, and B. Krishna. An OCR system for Telugu. In *Proc. of the 6th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 1110–1114, 2001.

[135] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases. at "http://www.ics.uci.edu/ mlearn/ MLRepository.html, 1998.

[136] Omniglot. Writing systems and languages of the world. at http://www.omniglot.com/writing/, 2007.

[137] U. Pal and B. B. Chaudhuri. OCR in Bangla : An Indo-Bangladesh language. In *Proc. of the 12th Int. Conf. on Pattern Recognition*, pages 269–274, 1994.

[138] U. Pal and B. B. Chaudhuri. Automatic recognition of unconstrained off-line Bangla handwritten numerals. In *Advances in Multimodal Interfaces, Springer Verlag Lecture Notes on Computer Science (LNCS-1948), Eds. T. Tan, Y. Shi and W. Gao*, pages 371–378, 2000.

[139] U. Pal and B.B. Chaudhuri. Computer recognition of printed Bangla script. *Int. J. Systems Science*, 26(11):2107–2123, 1995.

[140] U. Pal and BB Chaudhuri. Indian script character recognition: A survey. *Pattern Recognition*, 37(9):1887–1899, 2004.

[141] U. Pal and Anirban Sarkar. Recognition of printed Urdu script. In *Proc. of the 7th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 1183–1187, 2003.

[142] T. Pavlidis and S. Mori. Optical character recognition. *Proc. of the IEEE*, 80(7):1026–1028, 1992.

[143] C. Peters and P. Sheridan. Multilingual information access. *Lecture Notes in Computer Science*, 1980:51–59, 2001.

[144] R. Plamondon and S.N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[145] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.

[146] I. Popivanov and R. J. Miller. Similarity search over time series data using wavelets. In *Proc. of the 18th Int. Conf. on Data Engineering*, pages 212–221, 2002.

[147] M. F. Porter. An algorithm for suffix stripping. In *Program*, volume 14, pages 130–137, 1980.

[148] G. Prathap. Indian language document analysis and understanding. *Special Issue of Sadhana*, 27, 2002.

[149] Google Print. at `http://print.google.com`, 2005.

[150] The Collaborative Digitization Project. at `http://www.cdpheritage.org/`, 2007.

[151] J. R. Quinlan. Learning decision tree classifiers. In *ACM Computing Surveys (CSUR)*, volume 28, pages 71–72, 1996.

[152] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. N.J, Prentice Hall, Englewood Cliffs, 1993.

[153] P.V.S. Rao and T.M. Ajitha. Telugu script recognition - A feature based approach. In *Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 323–326, 1995.

[154] T. Rath and R. Manmatha. Features for word spotting in historical manuscripts. In *Proc. of the 7th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 218–222, 2003.

[155] T. Rath and R. Manmatha. Word image matching using dynamic time warping. *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2:521–527, 2003.

[156] T. Rath, R. Manmatha, and V. Lavrenko. A search engine for historical manuscript images. In *Proc. of the 27th Annual Int. Research and Development in Information Retrieval (SIGIR)*, pages 369–376, 2004.

[157] T.M. Rath and R. Manmatha. Word spotting for historical documents. *Int. Journal of Document Analysis and Recognition (IJDAR)*, 9(2):139–152, 2007.

[158] Sachin Rawat, K. S. Sesh Kumar, Million Meshesha, Indraneel Deb Sikdar, A. Balasubramanian, and C. V. Jawahar. A semi-automatic adaptive OCR for digital libraries. In *Proc. of the 7th Int. Assoc. for Pattern Recognition Workshop on Document Analysis Systems (DAS)*, pages 13–24, 2006.

[159] Y. Rui, T. S. Huang, and S. F. Chang. Image retrieval: Past, present, and future. *Journal of Visual Communication and Image Representation*, 10:1–23, 1999.

[160] G. Russell, M. P. Perrone, Y. M. Chee, and A. Ziq. Handwritten document retrieval. In *Proc. of the Int. Workshop on Frontiers in Handwriting Recognition*, pages 233–238, Korea, August 2002.

[161] G. Soda S. Marinai, E. Marino. Font adaptive word indexing of modern printed documents. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 28(8):1187–1199, 2006.

[162] W. Saffady. *Introduction to Automation of Libraries*. American Library Assoc., Chicago, 1994.

[163] H. Sakoe and S. Chiba. Dynamic programming optimization for spoken word recognition. *IEEE Transition on Acoustics, Speech and Signal Processing*, 26(1):623–625, 1978.

[164] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[165] P. K. Sankar and C. V. Jawahar. Probabilistic reverse annotation for large scale image retrieval. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[166] SAY Project Language Resources. at `http://crl.nmsu.edu/say/`, 2006.

[167] M. Schmill, T. Oates, and P. Cohen. Learned models for continuous planning. In *7th Int. Workshop on Artificial Intelligence and Statistics*, 1999.

[168] A. J. Sellen and R. H. R. Harper. *The Myth of the Paperless Office*. The MIT Press, Cambridge, MA, 2002.

[169] J. Sivic, M. Everingham, and A. Zisserman. Person spotting: video shot retrieval for face sets. In *Int. Conf. on Image and Video Retrieval (CIVR)*, pages 226–236, 2005.

[170] S. N. Srihari. High-performance reading machines. *Proc. of the IEEE*, 80(7):1120–1132, 1992.

[171] S. N. Srihari, C. Huang, H. Srinivasan, and C. Bhole. Spotting words in handwritten Arabic documents. In *Proc. of Document Recognition and Retrieval XIII (SPIE)*, volume 606702, pages 1–12, 2006.

[172] S. N. Srihari, G. Srikantan, T. Hong, and S. W. Lam. *Research in Japanese OCR, Handbook of Character Recognition and Document Image Analysis, H. Bunke and P.S.P. Wang (Eds.).* World Scientific Publishing Co., Singapore, 1997.

[173] H. Su, D. D. Feng, R. Zhao, and X. Wang. Face recognition method using mutual information and hybrid feature. In *Proc. of the 5th Int. Conf. on Computational Intelligence and Multimedia Applications (ICCIMA)*, page 436, 2003.

[174] V. S. Subrahmanian. *Principles of Multimedia Database Systems.* Morgan Kaufmann Publishers, Inc., USA, 1998.

[175] C. Y. Suen, S. Mori, S. H. Kim, and C. H. Leung. Analysis and recognition of Asian scripts - The state of the art. In *Proc. of the 6th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 866–878, 2003.

[176] Multilingual Systems. at `http://acharya.iitm.ac.in/cgi-bin/script_disp.pl`, 2000.

[177] K. Taghva, J. Borsack, and A. Condit. Effects of OCR errors on ranking and feedback using the vector space model. *Information Processing and Management*, 32(3):317–327, 1996.

[178] K. Taghva, J. Borsack, and A. Condit. Evaluation of model-based retrieval effectiveness with OCR text. In *ACM Trans. on Information Systems*, volume 14, pages 64–93, 1996.

[179] C. L. Tan, W. Huang, Z. Yu, and Y. Xu. Imaged document text retrieval without OCR. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 24(6):838–844, 2002.

[180] F. Tang, R. Crabb, and H. Tao. Representing images using nonorthogonal haar-like bases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2120–2134, 2007.

[181] J. M. Trenkle and R. C. Vogt. Word recognition for information retrieval in the image domain. In *Proc. of Document Analysis and Information Retrieval*, pages 105–122, 1993.

[182] O. D. Trier, A. K. Jain, and T. Taxt. Feature extraction methods for character recognition: A survey. *Pattern Recognition*, 29(4):641–662, 1996.

[183] H. R. Turtle and W. B. Croft. A comparison of text retrieval models. *Computer Journal*, 35(3):279–290, 1992.

[184] E. Ullendorff. *The Ethiopians: An Introduction to the Country and People, 3rd ed.* Oxford University Press, London, 1973.

[185] Unicode. at `http://www.unicode.org`, 2006.

[186] van Rijsbergen. *Information Retrieval, 2nd edition.* Butterworths, Boston, 1979.

[187] R. C. Veltkamp and M. Hagedoorn. State of the art in shape matching. In *Principles of Visual Information Retrieval*, pages 87–119, 1999.

[188] M. Vidyasagar. *A Theory of Learning and Generalization.* Springer-Verlag, New York, 1997.

[189] Wikipedia. at `http://en.wikipedia.org/wiki`, 2007.

[190] P. Wong and C. Chan. Postprocessing statistical language models for a handwritten Chinese character recognizer. *IEEE Trans. on Systems, Man, and Cybernetics–part B: Cybernetics*, 29(2):286 – 222, 1999.

[191] P. Wu and B. S. Manjunath. Adaptive nearest neighbor search for relevance feedback in large image databases. In *Proc. of the ninth ACM Int. Conf. on Multimedia*, volume 9, pages 89–97, 2001.

[192] Y. Xu and G. Nagy. Prototype extraction and adaptive OCR. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(12):1280–1296, 1999.

[193] B. Blumberg Y. Ivanov and A. Pentland. Expectation maximization for weakly labeled data. In *Proc. of the 18th Int. Conf. on Machine Learning*, 2001.

[194] Daniel Yacob. The live ge'ez remote processing protocol. at `http://libeth.sourceforge.net/LiveGeez.html`, 1999.

[195] A. Yamashita, T. Amano, Y. Hirayama, N. ltoh, S. Katoh, T. Mano, and K. Toyo kawa. A document recognition system and its applications. *IBM J. Res, Develop*, 40(3), 1996.

[196] A. Yaregal and J. Bigun. Ethiopic character recognition using direction field tensor. In *Proc. of the 18th Int. Conf. on Pattern Recognition (ICPR)*, pages 284–287, 2006.

[197] Baye Yimam. The interaction of tense, aspect, and agreement in Amharic syntax. In *Proc. of the 35th Annual Conf. on African Linguistics: African Languages and Linguistics in Broad Perspectives, John Mugane, John P. Hutchison, and Dee A. Worman (eds.)*, pages 193–202, 1996.

[198] O. Zamir and O. Etzioni. Web documeent clustering: A feasibility demonstration. In *Proc of the 21st Annual Int. ACM SIGIR Conf.*, pages 46–54, 1998.

[199] S. Zhang, X. Wu, J. Zhang, and C. Zhang. A decremental algorithm for maintaining frequent itemsets in dynamic databases. In *Proc. of the 7th Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK)*, pages 305–314, 2005.

[200] S. Zhao and H. Lee. High-precision two-kernel Chinese character recognition in general document processing systems. In *Proc. of the 6th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 617–621, 2001.

[201] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys (CSUR)*, 35(4):399–458, 2003.

[202] W. Zhao, A. Krishnaswamy, R. Chellappa, D. L. Swets, and J. Weng. Discriminant analysis of principal components for face recognition. In *Proc. of the 3rd Int. Conf. on Automatic Face and Gesture Recognition*, pages 336–341, 1998.

[203] Q. Zheng and T. Kanungo. Morphological degradation models and their use in document image restoration. In *Int. Conf. on Image Processing*, pages 193–196, 2001.

[204] X. Zhu and X. Yin. A new textual/non-textual classifier for document skew correction. In *Proc. of the 16th Int. Conf. Pattern Recognition (ICPR)*, volume 1, pages 480–482, 2002.

[205] J. Zobel, A. Moffat, and R. Sacks-Davis. Efficient indexing technique for full-text database systems. In *Proc. of the 18th Int. Conf. on Very Large Data Bases*, pages 352–362, 1992.

[206] G. Zsigri. Links to fonts and keyboards. at `http://zsigri.tripod.com/fontboard/ links.html`, 2003.