ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

FEATURE EXTRACTION AND MATCHING IN AMHARIC DOCUMENT IMAGE COLLECTIONS

ADANE LETTA

JUNE 2011

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

FEATURE EXTRACTION AND MATCHING IN AMHARIC DOCUMENT IMAGE COLLECTIONS

A Thesis Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Science

By

ADANE LETTA

JUNE 2011

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

FEATURE EXTRACTION AND MATCHING IN AMHARIC DOCUMENT IMAGE COLLECTIONS

By

ADANE LETTA

Name and signature of Members of the Examining Board

Name	**Title**	**Signature**	**Date**
_____	Chairperson	_____	_____
_____	Advisor(s),	_____	_____
_____	Examiner,	_____	_____

DEDICATION

In memory of my dad who was a good father and teacher.

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my advisor Million Meshesha (Ph.D) for his commitment, constructive comments and suggestion throughout the thesis work.

I would like to extend my special thanks for my mother who always gives me her heartfelt love and encouragement. My greatest thanks also go to my sincere younger brother Alex who has been my right hand in my postgraduate studies. Besides, I am very grateful to my love Jolie, my sister Mimi and all my brothers.

I am also indebted to University of Gondar for granting me the chance of attending graduate program in AAU. My thanks must go also for my colleagues and the staff of School of Information Science whose effort helped me much all along.

ABSTRACT

The ubiquity of digital computers and the boom of the Internet and World Wide Web resulted in massive information explosion over the entire world. Different types of information are uploaded in the Internet such as text documents, document images and other multimedia files. Document images facilitate office automation by preserving scanned documents in a document image database. However, information retrieving from document image database becomes a difficult task for organizations due to lack of efficient retrieval schemes. To overcome this challenge, recognition based and recognition free retrieval approaches are attempted by researchers. Recognition based retrieval first applies optical character recognition (OCR) to convert document images into text and then performs text retrieval using search engines. On the other hand, recognition free approach attempts to search and retrieve directly from document images relying on image features.

Due to the limitation of OCR systems, recognition based retrieval is not effective. Hence, attempts are made by different researchers to develop a document image retrieval system without explicit recognition. On top of this, attempts are made to develop effective Amharic document image retrieval system. As a continuation, the current study is initiated to explore and design feature extraction and matching schemes that are insensitive to word variants, difference in font types, sizes and styles and degradation.

In doing so, eight feature extraction methods and four matching techniques are tested. Of the four matching schemes dynamic time warping is insensitive to font types, sizes and styles difference. The eight feature extraction techniques are tested for performance, and then each feature is combined systematically following best stepwise feature selection method. The result shows that combined features score better performance than individuals. Using the best performer matching algorithm stemming is performed in image domain to handle word variants. Accordingly, promising experimental results are registered for word variants. The explored matching, feature extraction and stemming techniques are integrated with the previous Amharic document image retrieval system and tested on noisy document images. As the experimentation, the performance of the current system outperforms the previous attempts. Besides, relevant conclusions are drawn and some valid recommendations are forwarded to future investigation.

Table of contents

List of Tables

List of figures

List of Equations

List of Algorithms

List of Acronyms

ASCII:	American Standard Code for Information Interchange
COG:	Centre of Gravity
DFT:	Discrete Fourier Transform
DIR:	Document Image Retrieval
DIRS:	Document Image Retrieval System
DTW:	Dynamic Time Warping
IDF:	Inverse Document Frequency
IR:	Information Retrieval
NCC:	Normalized Cross Correlation
OCR:	Optical Character Recognition
TF:	Term Frequency

CHAPTER ONE

INTRODUCTION

1.1. Background

The ubiquity of digital computers and the boom of the Internet and World Wide Web resulted in massive information explosion over the entire world. Information is often based on data and usually in the form of a document or an audible or visible communication which has identifiable attribute [13].

Different types of information are uploaded in the Internet, such as text document, document images and multimedia data. Document images are used mostly by organizations for preserving scanned documents in a document image database. This facilitates office automation and the move towards paperless office. Hence, these digital documents have become vital asset of organizations and users over the entire world [64], thus the need of visual content management is on the rise.

However, information retrieving from document image database becomes a difficult task for organizations. Information retrieval is finding documents of unstructured nature that satisfies an information need of users from or within large document collections [16]. Many document image collections are now scanned and being made available over the Internet. Effective access to such document image sources is limited by the lack of efficient retrieval schemes [6].

To this end, the concept of Document Image Retrieval (DIR) became the hot issue in information retrieval. DIR aims at identifying relevant documents from the document image database only relying on image features [63]. DIR is mainly focused on retrieving document images from document image database using textual or image query. The existing technology has made it possible to produce, process, store, and transmit document images efficiently. Hence, large quantities of printed documents are digitized and stored as images in databases [80]. However, searching and retrieving relevant documents from image database is not the easiest task. Basically two approaches are attempted by researchers on this area: recognition based and recognition free retrievals.

Under the traditional recognition based retrieval scenario, scanned document images need to be first converted to editable text through optical character recognition (OCR) tool [23]. Nevertheless, using OCR tool is still not perfect and requires human correction that could incur a prohibitive cost for a large document corpus [15]. Similarly, for a huge number of document images archived in digital libraries using OCR for the retrieval purpose is wasteful of resources and has been proven prohibitively expensive, especially the difficult post-OCR correction process [61].

According to Million [45], character recognition from image documents that are printed in Amharic script is a challenging task due to: lack of previous experience, printing variations, large number of characters in the script, visual similarity of Amharic characters in shape, language related issues and degradations of documents. Hence, designing efficient Amharic document image retrieval system without recognition is one basic remedy to overcome limitations of the recognition system (OCR) in the short run.

Retrieval without explicit recognition works by considering image features (foregrounds and backgrounds) without the need of converting the image documents to editable texts [45]. Designing effective document image retrieval system without explicit recognition requires matching and feature extraction schemes that tolerate the basic challenges of real life document images.

1.2. Statement of the Problem and Justification

The invention of digital devices and their use in office and home resulted in the production of voluminous amount of printed documents. Due to this information overload, retrieving the required information manually from printed documents has become tiresome and time consuming. Hence, there should be effective and efficient system which eases information access from bulk of documents.

Many researchers proposed different document image retrieval techniques for digitizing printed and handwritten documents and making retrieval possible. Due to limitations of using OCR system in document image retrieval, an attempt had been made by different researchers for retrieving document images written in different languages such as English[15,36,48,61,65,73], Arabic[3,73], Hindi [6,48] and

Chinese[15]— without explicit recognition. To the best of the researchers' knowledge, three attempts are made in designing document image retrieval system for Amharic document image corpus [4, 44, 48].

Million and Jawahar [48] use image properties for searching relevant document images from document image database of Amharic, English and Hindi text collections. They use word spotting approach for retrieving printed documents images. Image features are extracted using projection profile, word profile and moments. Besides, dynamic time wrapping (DTW) is used for matching and aligning word images with different sizes. DTW-based partial matching technique is proposed to take care of word form variations.

Mesfin [44] proposes an Amharic document image retrieval system which attempts to search for relevant document images as per the user query. He employs pre-processing techniques for binarization of the document images. Image segmentation by thresholding is used to identify words in document images. Then, word shape analysis is used for extracting word features. On this proposed system, image query to word image matching is performed using Euclidian distance which is greatly sensitive to word variants. Besides, the proposed system is designed to handle one font type, style and size.

The work of Abreham [4] further explores the introduction of indexing scheme in order to improve efficiency and effectiveness of Amharic document image retrieval system during searching. To this end, inverted file is used as indexing scheme for organizing content bearing word images. However, this system performs retrieval without considering difference in font types, sizes and styles.

The proposed systems by Mesfin and Abreham perform retrieval without considering different font sizes, styles and types. The document images that are used as test data are font type, style and size specific. Thereby, complexity of extracting and processing image feature is still a bottleneck in designing a better document image retrieval system. Hence, the main concern of the current study is exploring different matching and feature extraction techniques which are insensitive to the artifacts in real life word images, such as noise, word variations and difference due to font sizes, styles and types.

To this end, this research is intended to explore solutions for the following research questions.

- Which matching algorithms and feature extraction schemes are insensitive to word variation and handle difference in font styles, types and sizes, to design the prototype system?
- Is it possible to design effective and efficient stemming algorithm that group together word variants?
- To what degree the performance of the document image retrieval system is improved by introducing effective combined feature extraction and partial matching approaches?
- What flaws are identified and what possible remedies are recommended to come up with an applicable system?

1.3. Objectives of the Study

1.3.1. General objective

The general objective of this research is designing and integrating effective and efficient matching and feature extraction schemes in order to enhance the performance of Amharic document image retrieval system. Exploring further matching and feature extraction techniques has a great impact to control the various artifacts, including word variants, font sizes, styles and type variations and degradation observed in real-life document images.

1.3.2. Specific objectives

On the way of attaining the general objective, the study specifically achieves the following objectives.

- Review previously proposed related works and then identifies different techniques and algorithms that have been used.
- Study different feature extraction and word level matching techniques and designing efficient feature extraction and matching techniques which are insensitive to font style, size and type differences.

- Develop document image retrieval system that take into account word variants and word image size difference.
- Prepare dataset collections which are used to train and then test the performance of the prototype system.
- Evaluate effectiveness of the information retrieval system using evaluation techniques such as recall, precision and F-measure.
- Identify and recommend future research directions for further investigation in the area.

1.4. Methodology of the Study

Methodology is an approach which involves data collection, analysis and interpretation that show how a researcher achieves the objectives and answers the research questions [33]. Hence, in order to achieve the specific and general objectives of the study and answer the research questions, the following methods are used.

1.4.1. Literature review

Several previously proposed related literatures (from books, articles and conference proceedings) are briefly reviewed in order to have detail understanding on the present research. As continuation of two previous attempts [4, 44], different techniques and tools which are relevant for the current research are analyzed, modified and adopted from their works. Since the current research is designing document image retrieval system for Amharic printed documents, the feature of Amharic scripts are studied from printed documents.

1.4.2. Dataset collection

Computer printed Amharic documents with various font styles, sizes and types are collected from book, magazines and news paper and converted to gray scale digital images, digitized, using a flatbed scanner at 300 DPI. Moreover, clean word images with different fonts are generated using Microsoft Office Document Image tool. Besides, to easily convert one Amharic font type into different font sets Abnet Amharic font converter is used. All the prepared data sets are used as an input for testing the performance of the investigated Amharic document image retrieval system.

1.4.3. Implementation tool

Previous attempts made by Mesfin [44] and Abreham [4] had used Java programming language for image pre-processing, segmentation, feature extraction, indexing, document image to query matching and retrieval of relevant documents as per user's query. The reason is that this programming language has a lot of built in methods for image processing. Hence, since the current study is a continuation of the specified researchers, the same programming language is used. In addition, the researcher is familiar with this programming language on previous undergraduate studies.

1.4.4. Testing procedure

To test the performance of the Amharic document image retrieval system, Amharic document images with difference in font styles, font sizes and font types are given as input. Then, the performance of the system is measured using three most widely used performance measuring techniques: Precision, Recall and F-measure [16]. Accordingly, precision is the fraction of retrieved documents that are relevant; recall is the fraction of relevant documents that are retrieved and F-measure is the weighted harmonic mean of precision and recall that is a single measure that trades off precision versus recall.

1.5. Significance of the Study

Different Documents articulated and printed in Amharic scripts are piled high in information centres, libraries, museums and government and private institutes [47]. Manually accessing these printed documents is tiresome and time consuming. To tackle the specified problem, very few studies have been done by [4, 44 and 48] on designing Amharic document image retrieval system. Besides, this study puts its own contribution in developing effective and efficient Amharic document image retrieval system. It can also be implemented in different governmental and non-governmental institutions to move towards paperless office, by electronically preserving Amharic printed document and retrieve the required information timely. In addition, it has an inevitable contribution for library digitization. Furthermore, it can be used as an input for future works which might be aimed to develop a full-fledged Amharic document image retrieval system.

1.6. Scope and Limitation of the Study

This research is the extension of [44] and [4] who made attempts in the area of document image retrieval. Hence, the study adopts and modifies some image processing techniques from them. On top of that the main intent of this research is further investigating feature extraction and matching algorithms that are insensitive to font difference and word variants and thereby enhance effectiveness and efficiency of document image retrieval system. Four matching algorithms are evaluated and the best matching algorithm that is insensitive to font type, size and style variations is selected. Furthermore, to get better performance feature extraction technique, group of features are combined using best stepwise feature selection method. In addition, to handle word variants DTW base suffixes and prefixes removal technique is integrated. The performance of the integrated system is evaluated on noisy and clean document images. However, the performance of the designed system isn't evaluated on large Amharic document image corpus. Besides, infixes detection and removal is not handled.

1.7. Organization of the Study

The paper is organized in five chapters. The first chapter discusses the background of the study and statement of the problem. It also presents general and specific objectives of the study, methodology of the study, scope and limitation of the research and application of the investigated results.

Chapter two presents literature review on the history of writing system and the development of the Amharic writing system. Moreover, a brief review on information retrieval, document image retrieval and related work are discussed.

Chapter three describes the proposed feature extraction and matching algorithms. The evaluation measures that are used for measuring the performance of each feature extraction and matching algorithms are briefly introduced.

Chapter four emphasized experimental results that are used to confirm the validity of the proposed feature extraction and matching techniques for Amharic document image retrieval. Finally, conclusion and recommendation of the research are jotted in chapter five.

CHAPTER TWO

LITERATURE REVIEW

Only recently, the advancement in computing device and office applications grounded with the ability to create, manipulate, store, retrieve and transmit electronic documents. Electronic text document is usually easy to search and retrieve, thus led to the development of many text search engines. However, traditionally, transmission and storage of information have been performed by paper documents. Therefore, there remains huge volume of printed documents whose electronic representations are simply not available [19, 35, 54]. Voluminous amount of documents articulated and printed in Amharic scripts are piled high in information centres, libraries, museums and government and private institutions [47]. Due to this, retrieving the required information from document collections is challenging task. Hence, to overcome the challenge printed documents should be digitized and retrieved using appropriate retrieval techniques.

2.1. Overview of Writing

These days, information can be communicated among the society in different ways. Communication is possible using spoken language and gesticulations in the ways similar to those used by people who lived hundred thousand years ago [2]. However, spoken language as a purely phonetic medium of communication, no longer suffice, where its limitation in space and time are not comparable to the demands of the progressive development of civilization. Therefore, human devises writing as a remedy [27]. Writing is defined as a process of putting words or figures on to papers for communicating information [58]. According to Lawrence [37], writing provides a way of extending human memory by imprinting information into media which is less changeable than the human brain. In addition, writing uses to hand cultural development down to the future generations as a secure possession for further cultivation [27].

Development of writing system on various part of the world may not have linear relationship. As clarified by Bruce [10], contrary to what was once widely believed, writing had multiple origins and developed in a far from unilinear fashion.

Accordingly, the three best-known early writing systems (Sumerian, Shang Chinese, and Maya) appear to have developed completely independent of one another. However, every writing system composes a set of visible or tactile signs that are used to represent units of language in a systematic way [37].

The earliest writings are credited to the Sumerians who occupied Southern Mesopotamia (part of modern day Iraq) from around 3300 to 2000BC. The Sumerians invented glass, became expert in technology of brick manufacturer and building and devised positional significance in mathematics, all of which were significant development [2].

Therefore, it is not surprising if the Sumerians claim the world's first scribes; since it has often been claimed that writing is the essential distinguished feature of urban life. Their alluvial terrain had no wood or stone, to the Sumerians wrote on what they had: clay, which they formed into tablets and on which they made marks with a stylus cut from a reed [25]. The Sumerians writing system used Archaic Sumerian script which is not fully understandable and much harder to translate by modern scholars [2, 25].

On the other hand, the art of writing in Egypt is believed to have begun in its earliest stages almost 7000 years ago, but the hieroglyphic system — among the oldest writing system in the world— seems to be firmly established day back to 3000BC [24, 37]. So, like that of Sumerians, Egyptian writing system is one of the earliest writing systems. The Egyptian invented papyrus, a material made by pounding strip of reed- pitch, which could be made into rolls that may be considered as the first real transportable book. Papyrus was less permanent and more subject to damage than stone and clay tables, but it was more efficient for recording and transporting written work for others [2].

2.2. The Ethiopic Script

At about the beginning of our era South-Arabian Semitic migrated from Habashat on the Arabian coast across to Africa and founded a kingdom there with its capital at Axum. The immigrants called themselves Ge'ez, which means the 'emigrants' [27]. Besides, according to Aleksandar [5], the name Ge'ez comes from the South Arabian

people, the Agazyān, who crossed the Red Sea and settled in North Ethiopia and along the Red Sea coast.

Hans [27] describes that during the 1st century the emigrants used the Sabean script for the representation of their language, the so-called Ge'ez language. It was replaced, however, in about 350 by another form, the old Abyssinian script. This script in its later form transmitted in manuscripts and usually described as the Ethiopic or Ge'ez script.

Two arguments are set on how Ge'ez script was originated. Primarily, the classical Ethiopic language (Ge'ez) belongs to the southern group of the Semitic language within the Hamito-Semitic family [5, 27, 31]. They emphasized that it is mostly closely related to classical Arabic and to the south Arabian epigraphic language which is so far known only from inscriptions. On the other hand, Gabriella [24] argued that Ge'ez syllographs descend fairly directly from Hieroglyphs.

Ethiopic script as a syllabic script, which originated from the south Arabian epigraphic alphabet, has been used in Ethiopia since the 4th century [31]. Hans [27] notes that Ethiopic script is one of the Semitic group of scripts those so called-alphabetical script which were, and to some extent still are, employed by the Semitic peoples. This Ethiopic script, sometimes called neo-syllabic, is read from left to right, doubtless under Greek influence, unlike that of the other Semitic scripts, with the exception of Ugaritic [5].

According to Hans [27], the Semitic alphabetical scripts exhibit special characteristics; their signs serve solely for the expression of consonants, while the vowels are not expressed. Consonantal alphabets are all descendents of the Proto-Sinaitic script. In a "pure" consonantal alphabet, vowels are not written [37]. The structure of Semitic language is governed by the principle that the consonants are the bearer of meaning, while the vowels play a more secondary role.

Hans [27] also emphasizes that the Semitic script from the very beginning of their development fall into two great branches: the North-Semitic and the South-Semitic alphabet. The South Semitic scripts are distinguished as two main groups: North-Arabian Scripts and South-Arabian Scripts.

The South Arabian Scripts consist of the (Mino) Sabean and the (old Abyssinia) Ethiopic script— Ge'ez. The oldest of these to have been found in Ethiopia probably dates from the 4th century B.C and was the dedicatory inscription from the locality of Yeha [5]. The common writing surface of ancient Ge'ez is "Birana", a parchment made from animal skin [24].

Ge'ez represents the outcome of the script reform; twenty-four sings were selected from the Sabean alphabet, which were slightly modified, and two new signs were invented to represent sounds lacking in the Sabean script [27, 42]. It is fairly massive in size, with its 182 syllographs. There are essentially 26 main syllographs, all consonants, in Ge'ez; while the rest are essentially those with additional strokes and modifications added on to the main forms to indicate a vowel sound associated with it [24]. The basic feature that distinguishes the Ethiopic script in a quite exceptional way is the indication of vowels [27].

APPENDIX I shows the syllographs of the Ethiopic script as originally used for the Ge'ez language. Each sign is a syllable (consonant plus vowel), except any sign on the sixth column (ə) represents the consonant plus the middle central vowel /ə/ or no vowel at all. In which case, it is used as a pure consonant in a consonant cluster [24, 37]. Minor modifications are made on the leading character, the addition of little lines, or bending or lengthening of the stroke, to generate the six consonantal characters. Gabriella [24] shows the representation of vowels that are added on the leading symbol, as listed in Table 2.1. These six characters are expressed as: leading character + ū, leading character + ī, leading character + ā, leading character + ē, leading character + ĕ (consonant without vowel), and leading character + ō.

Leading Character → Leading character + vowels

	Gee'z ä	Ka'eb u	Salis Ï	Rab'e a	Hamis é	Sadis i	Sab'e o
h	G	G<	H>	H	H¦	I	J
l	K	K<	K=	L	K?	M	KA

Table 2.1 Representation of leading characters and vowels [24].

As shown in Table 2.2, besides the basic signs, there are series of derived signs to represent labialized velar consonants. For example, these are velar sounds like /k/, /g/, /q/, and /h/ that are pronounced with the lips rounded regardless of the vowel [37].

	a	i:	a:	e:	(ə)		a	i:	a:	e:	(ə)
q^w	ቋ	ቍ	ቋ	ቌ	ቍ	k^w	ኰ	ኲ	ኳ	ኴ	ኵ
h^w	ኈ	ኈ	ኋ	ኌ	ኈ	g^w	ጐ	ጒ	ጓ	ጔ	ጕ

Table 2.2 Signs represent labialized velar consonants

It was in the territory of these Ge'ez -speaking people that the Ethiopian state with its capital at Aksum existed for about a thousand years from 1st to 10th century [5]. Accordingly, the adoption of Christianity by the state, most probably around the middle of the 4th century, showed the advanced use of the language in religious writings. Thus, from at least the 4th to around 11th century Ge'ez was in official spoken and written use by the people of the state of Aksum, but after the state collapsed it was no longer used as a spoken language being replaced from the 13th century onwards, if not earliest, by Amharic.

Gebriella [24] stresses that by the ninth or tenth centuries ancient Ge'ez ceased to exist as a spoken language in Ethiopia, even the spoken Ge'ez also split into many closely related tongues, mainly Tigriña in the north and Amharic in the south. However, written Ge'ez was kept firmly in use purely for sacred and scholarly endeavours, from the thirteenth through the seventeenth centuries which is known as the "classical period" of Ethiopian literature.

Ge'ez remained the official language of Ethiopia until the middle 19th C, when the 1st official documents were issued in Amharic, the chronicles of Emperor Theodros II, 1855-1868. Furthermore, Ge'ez has remained the liturgical language of the Orthodox church of Ethiopia and writings that were primarily religious and historic [5].

2.3. Amharic Script

Amharic, the national language of Ethiopia, has about 27 million speakers [52]. Accordingly, it is spoken mainly in North and Central Ethiopia. Besides, there are Amharic speakers in a number of other countries, particularly in Egypt, Israel and

Sweden. Amharic has been a written language for roughly 600 years and has a rich legacy of both typeset and calligraphic literature [18]. Moreover, Amharic script is still the normal medium for newspapers, magazines, novels, poetry, primary school texts books (for some regional States, such as Amhara, Addis Ababa and others), official and legal documents and other printed materials as well as for private correspondence [66].

Amharic language belongs to the Semitic branches of the Afro-Asiatic family of language which is originated from Ge'ez, similar to Tigrē and Tigreñ languages [31]. The Amharic script adopted Ethiopic script (Ge'ez) for its written. All the twenty six Ge'ez scripts together with sings represent labialized velar consonants are adopted. However, Amharic script assimilated some new (Hamitic) letters to denote sounds that are not found in Ge'ez [27, 37]. In about the middle of the fourteenth century seven more new signs were introduced, all these signs, as depicted in Table 2.3 formed through minor modification from phonetically similar ones that already exist in Ethiopic script.

Ethiopic	Phonetic value	Amharic	Phonetic value
ሰ	sa'	ሸ	ša
ተ	ta'	ቸ	ča
ነ	na'	ኘ	ña
ከ	ka'	ኸ	ĥa
ዘ	za'	ዠ	ža
ደ	da'	ጀ	ǧa
ጠ	ţa	ፏ	ča

Table2.3 additional Amharic letters [27].

The Amharic writing system consists of a core thirty three characters (ÜdL, fidel); each of which occurs in a basic forms and in six other forms known as orders. The seven orders represent syllable combination consisting of a consonant and following vowels. This is why the Ethiopian system is often called a syllabary rather than an alphabet. In addition to 231 characters, there are nearly 51 others which contain a special features usually representing labialization [42]. Besides characters and

labializations, numerals and punctuation marks are also available. These all bring the total number of Amharic scripts to 310 [46].

2.3.1. Amharic numeration system

Bender et al. [42], as shown in Table 2.4, points out that the Amharic numeration system consists of basic single characters for one to ten, for multiple of ten (twenty to ninety), hundreds and thousands.

These numerals are derived from the Greek numerals with some modifications; each has a horizontal stroke below and above. In the system, there is no symbol for representing zero value and it is not a place value system, thus arithmetic computation using this system is very difficult. Consequently, in most printed document Hindu-Arabic numerals are used [42].

Ethiopic	**Arabic**	**Ethiopic**	**Arabic**	**Ethiopic**	**Arabic**
í	1	ô	8	û	60
î	2	õ	9	ü	70
ï	3	ö	10	ý	80
ð	4	÷	20	þ	90
ñ	5	ø	30	ÿ	100
ò	6	ù	40	—	1000
ó	7	ú	50		

Table 2.4 Arabic Vs Ethiopic Numerals

2.3.2. Amharic punctuation marks

Amharic punctuation marks consist of different symbolic representations. Mostly used punctuation marks are: the basic word divider, **ሁለት ነጥብ**-hulet netib, which has two dots arranged like a colon (፡) and a sentence ending is represented using, **ዐራት ነጥብ**-arat netib, four square dots arranged in a square pattern (፨). Some others equivalent to comma represented as, **ነጠላ ሰረዝ**--netela serez,(፣) and semicolon represented, **ድርብ ሰረዝ**-derib serez,(፤). Question mark, quotation mark and exclamation mark are

represented as, ጥያቄ ምልክት-tiyake milikit, (?),ትምህርተ ጥቅስ-- timihrite tiks, (« ») and ትምህርተ አንክሮ-- timihrite ankro (!), respectively [42].

2.3.3. Amharic script features

Million [46] discusses specific features of Amharic writing system:

- **Shape similarities among characters:** the shapes of many Amharic characters show similarities with few distinctions, for instance, ደ and ጀ, ተ and ቸ, and ኸ and ከ.
- **Structural relation**: some basic characters of the writing system are also clearly related in graphical structure such as ካ and ከ, ረ and ፈ.
- **Shape differences**: There are remarkable differences in shape among the basic characters. For example consider ሀ and በ, these two letters are open in one side but in opposite direction. On the other hand, መ and ወ are formed from two loops but vary in loops connection, ሠ and ጠ have three legs which end in opposite directions.
- **Size and width differences:** Amharic characters can differ in size either vertically or horizontally. Some characters are very short like (ፈ, ሠ, መ) and some others are very long such as (ቸ, ኘሮ, ኽ). Furthermore, there is also noticeable difference in width, for example between ካ, ማ and ጨ.

2.3.4. Challenges of Amharic writing system

Bender et al. [42], Thomas [66] and Bethlehem [82], point out problems of the Amharic writing system as follows.

2.3.4.1. Redundancy symbols

The use of different symbols in writing for representing the same concept result a considerable systemic redundancy, this creates a challenge in Amharic writing system. For instance four different symbols represent the same sound/h/ + vowel: ሀ, ኀ, ኃ and ሐ. There are additional symbols, i.e. /s/: ሰ , ሠ and /s'/ : ፀ , ጸ which can be used interchangeably as one sound. Such redundancy may result an ambiguity on learners and users of the writing system [42].

For instance, using three of these symbols-- **ሀ**, **ጎ**, **ሐ** -- to represent the name ‘härägäwäyini’ in Amharic resulted, at least, six different Amharic wordings, **ሀረገወይን**, **ሃረገወይን**, **ሐረገወይን**, **ሓረገወይን**, **ኀረገወይን**, and **ኃረገወይን**. Consequently, designing retrieval system, especially document image retrieval system for Amharic printed documents is somehow complex due to such redundancies.

2.3.4.2. Feature similarity among characters

Similarity among features of different characters resulted confusion specially in designing retrieval systems, for example, a system may group the word **ደመረ** and **ጀመረ** in the same cluster. Even if the first letter of both word have similarities with few distinctions, the remaining two characters are similar. Thus, such feature similarities among different characters greatly affect the performance of retrieval system [66].

2.3.4.3. Font variations

Different Amharic fonts have been produced over the last two decades, such as, Alpas, Brana I/II, Agafari, Powergeez, Geez, ALXethiopian, Visual Geez Unicode, Ethiopia Jiret, Nyala, Addis98, and others. However, due to lack of standardization all Amharic fonts use different set of symbols, thus Amharic text written in one font type cannot read by other font types. Bethlehem [82], citing Zelalem (2001) notes that font variation is also a problem in designing retrieval system for printed Amharic documents. Besides, font type and size differences have their own challenges. Using four different font types: Ethiopia Jiret, Nyala, GF Zemen Unicode, ALXethiopian, font size: 10,12 and 14 and font style: normal, *italic* and **bold** , the word ’midirini’ can be represented as shown in Table 2.5.

Font types	Font size 10	Font size 12	Font size 14	Font style Normal	Font style **Bold**	Font style *Italic*
Ethiopia Jiret	ምድርን	ምድርን	ምድርን	ምድርን	**ምድርን**	*ምድርን*
Nyala	ምድርን	ምድርን	ምድርን	ምድርን	**ምድርን**	*ምድርን*
GF Zemen Unicode	ምድርን	ምድርን	ምድርን	ምድርን	**ምድርን**	*ምድርን*
ALXethiopian	M D R N	M D R N	M D R N	M D R N	**M D R N**	*M D R N*

Table 2.5 Font type, size and style variations

As it is observed, difference in font types, sizes and styles are resulted different word lengths. For instance, the word written in font type Ethiopia Jiret, font size 12 and font style Normal is the shortest in length, but the word written in ALXethiopian with similar font size and style is the longest of all the others. Moreover, the word that is written in Ethiopia Jiret, is more condensed than others. The first three letters-- ምድር- -are leaning on each other. On the other hand, the last word which is written in font type of ALXethiopian, size 12 and style normal seems bold. Thus, variation in font types, styles and sizes have a profound impact on designing retrieval system for Amharic printed documents.

2.3.4.4. Formation of compound words

The Amharic writing system uses multitudes of ways to denote compound words and there is no agreed upon spelling standard for constructing these words [20]. According to Bender et al. [42], compound words are sometimes written as a single word and other times as two separate words. For example, the word 'megneta bet' which means "bed room" can possibly be written as '**መኝታ·ቤት·**' or '**መኝታ· ቤት·**' and also the word 'bet mekides' which means "temple" can be written as '**ቤተመቅደስ**' or '**ቤተ መቅደስ**' [83]. This inflation nature of the writing system highly affects the performance of Amharic document image retrieval systems.

2.4. Information Retrieval

The invention of printer by Gutenberg in 15th century resulted in massive printed documents in Europe, then over the entire world [2]. To date with the advancement of the Internet services and the increase in number and importance of sophisticated telecommunications systems, microcomputers, and computer terminals in offices, information overload is an increasingly common occurrence [56]. Similarly, since Amharic is the official and working language of Ethiopia and the most commonly learnt language, there is a bulk of information available in printed document formats. Furthermore, large and distributed collections of scientific, artistic, and commercial data comprising images, text, audio and video abound in our information-based society [81]. As a result, retrieving the required information from these collections is becoming a challenging task.

The specified problem can be tackled by developing effective and precise Information Retrieval (IR) system for users to search, browse and interact with the bulk of document corpus and do so in a timely manner [81]. According to Meadow et al. [43], IR has been defined as finding relevant information in a store of information or a database. IR system can be developed to retrieve the required information from textual documents or document images. The focus of this work is designing matching and feature extraction scheme for retrieving relevant information from Amharic document images.

2.5. Document Image Retrieval

Due to the use of scanners, digital photocopiers and digital cameras, wide varieties of information which have been conventionally stored on paper, are now being converted into electronic form for better storage and intelligent processing. Therefore, the world has experienced a phenomenal growth on the size of multimedia data and document images [34, 36, 39]. However, extracting information from the document images is difficult as it compared with digital texts [39]. On that account, in order to satisfactorily exploit information from these collections of document images, it is necessary to develop effective and efficient document image analysis techniques.

The objective of document image analysis is to recognize the text and graphics components in document images, and to extract the intended information as per human needs [54]. Two techniques for document image analysis are developed. The first is an optical character recognition (OCR) system that reads the document image and converts it into an appropriate electronic format, and then applies both indexing and retrieval with this format. This kind of retrieval is called recognition based retrieval. The second approach is recognition free retrieval which is based on keyword spotting, where the document image is first segmented into words, and the user keywords are located in the image by a word-to-word matching[11, 62].

2.5.1 Recognition based retrieval

Recognition based retrieval is the usual way of information retrieval from scanned document images. This recognition technique digitize document images and then perform optical character recognition to transform characters, which were contained

in the digitized printed document into a machine-editable text (character-coded text) such as in the American Standard Code for Information Interchange (ASCII) or Unicode format [15, 35, 36]. OCR systems are more widely used to store, search, and excerpt information from paper-based document images [54].

Million [45] and Darwish [35] emphasize that the OCR engine accepts a scanned image (printed, typewritten or handwritten text document), and then passes the image through many stages before it is translated into a computer understandable format. The first stage is preprocessing where the document image undergo some image enhancements such as noise filtering, skew correction and normalization. After that, image segmentation is performed to identify lines, words and then characters— bunch of smaller images— using image analysis heuristics. On the resulted raw digitized data, feature extraction techniques are applied to extract special patterns of the character image. Then, automatic classifiers are used to determine the character code that corresponds to each character image. It also exploits sequential context in order to convert character image into textual format. Once document images are converted to texts, traditional information retrieval strategies are applied—searching the result using a text search engine.

For more than a decade attempts have been made to develop a robust OCR for printed and handwritten Amharic documents by different researchers [21, 32, 47, 75, 76, 78,]. However, developing robust OCR for Amharic script is still challenging.

Although robust OCR systems for Latin characters have been available [32, 80], to date there are no robust OCRs for Ethiopic script that can successfully recognize printed document images of varying quality, size, style and font [48].

Million [45] points out that character recognition from image documents which are printed in Amharic scripts is a challenging task due to: lack of previous experience, printing variations, large number of characters in the script, visual similarity of characters in shape, language related issues and degradations of documents. Hence, designing an efficient Amharic document image retrieval system that is free of recognition looks a short term solution to overcome limitations of OCR systems.

2.5.2. Recognition free retrieval

To overcome the problem of OCR a promising direction is to search for relevant documents using only image properties, without explicit recognition [48]. Consequently, document image retrieval without explicit recognition becomes a very attractive field of research with the continuous growth of interest and requirements for the development of the modern society [40].

According to Bhardwaj et al. [7], in recognition-free retrieval of document image with word spotting (word level image to query matching), the document image is preprocessed and word segmentation are performed, then feature vectors are extracted from word images and stored in a database. When a user provides a textual query word, the query word is first rendered, the similarity between the query and the word image in the database is computed and document images are returned according to descending order of their similarity. Generally, to retrieve information from document images a number of techniques are involved: prepocessing, segmentation, feature extraction, indexing, matching and displaying relevant document images in ranked order.

2.6. Image Document Preprocessing

To perform preprocessing the printed document should be digitized by optical scanning or by digital camera that yield a file of picture elements or pixels. This digitized image is the raw input to document analysis. So, it is important to understand that the image of the document contains only raw data that must be further processed to extract the relevant information. Such processing includes: thresholding and noise reduction [54].

2.6.1. Thresholding (Bianrization)

In any image analysis or enhancement problem, it is very essential to identify the objects of interest from the rest. The group of pixels representing objects of interest are called foreground pixels and the rest are called background pixels [39]. Therefore, the objective of binarization (thresholding) is to automatically choose a threshold that separates the foreground region with a single intensity (ON) and background region

with a different intensity (OFF). After that, it reduces a gray-scale or color image to binary image, i.e, 1(ON) and 0(OFF) [54].

According to Young et al. [79] the concept of bianrization can be applied using Algorithm 2.1.

```
If   a[m,n] ≤  θ
        a[m,n]= Object=1
Else
        a[m,n]=background=0
```

Algorithm 2.1. Binarization

Where θ (the brightness threshold) is chosen and applied to the image $a[m,n]$

They also describe that there is no universal rule for selecting threshold value. Nevertheless, techniques such as fixed threshold, histogram derived thresholds and background symmetry algorithm can be used. Fixed threshold uses threshold that is chosen independently of the image data. Histogram derived threshold chooses threshold from the brightness histogram of the image region. Background symmetry algorithm assumes a distinct and dominant peak of the background that is symmetric about the background to select the threshold.

Furthermore, Wu and Manmatha [69] classify thresholding techniques into global and adaptive. Global technique binarizes the entire image using a single threshold. This single threshold is selected using the value at the valley of the intensity histogram of the image, by assuming two peaks one corresponding to the foreground and the other for the background. In contrast, adaptive algorithm computes a threshold for each pixel based on information extracted from its neighborhood. In this case, different threshold value is used based on difference in intensity of image regions.

2.6.2. Reduction of noise

Document images often suffer from various types of document degradations such as impulse, noise and low contrast. In document image noises get introduced due to many reasons: aging, photocopying, or during data capturing. Therefore, document

images need to be preprocessed so as to improve the effectiveness of retrieval by reducing extraneous data, noise. Image and signal processing methods are applied to reduce noise. Methods such as mean filter and median filter are the most popularly applied techniques for denoising. These methods generally tend to reduce the effect of noise on the performance of the document image retrieval algorithms [39, 54, 61].

2.7. Image Segmentation

Segmentation can be considered as the key issue in document image processing and image understanding. Image segmentation leads to more compact image representations by partitioning an image into a set of disjoint segments to represent image structures [59]. Similarly Cufi et al. [77] describe that the aim of image segmentation is the domain-independent partition of the image into a set of regions, which are visually distinct and uniform with respect to some property such as grey level, texture or colour.

In matching and retrieving information from document images using word spotting, line segmentation and word level segmentation are performed. The word segmentation module separates the text region into the unit of lines and then finally to words. It uses a horizontal projection profile cut method to segment a text block into lines, and utilizes gaps and special symbols as delimiters between words in order to separate a text line into word units [11]. Cufi et al. [77] emphasize that segmentation methods are based on two basic properties of the pixels in relation to their local neighbourhood, these are: discontinuity (boundary-based methods) and similarity (region-based methods). The boundary approach calculates the gradient vector and takes its value as threshold value, pixels with a gradient magnitude above the threshold are considered. Similarity method tries to identify areas of images that are homogenous. Hampton [12] notes that image segmentation is possible through three approaches. The first is region approach where each pixel is assigned to a particular object or region. The other is a boundary approach where the locations of boundaries existing between regions are desired. The final perspective is the edge approach, where identification of the edge pixels is followed by a linking process of these edges which will form the required boundaries.

Skarbek and Koschan [74] forward four document image segmentation techniques: pixel based, area based, edge based and physics based segmentations.

According to Bukhari et al. [65], pixel based segmentation approach attempts to classify document images into predefined classes on the base of individual pixels. Techniques for pixel based segmentation can be categorized as: histogram based techniques, segmentation by clustering data in colour space and segmentation by fuzzy clustering [74].

Area based or region based segmentation approach tries to isolate areas of document images that are homogeneous according to a given set of characteristics. Candidate areas may be grown, shrunk, merged, split, created or destroyed during the segmentation process. There are two typical region-based segmentation algorithms: region-growing, and split-and-merge [77].

Edge based segmentation is one of the most frequently used segmentation techniques in digital image processing. The boundaries of object surfaces in an image often lead to oriented localized changes in intensity of an image, called edges. Edge detection refers to the process of identifying and locating sharp discontinuities in an image [49]. Skarbek and Koschan [74] classify edge detection as local and global classification techniques. Local technique is used to determine an edge point using information in the neighbourhood of that point, but global technique makes a sort of global optimization, thus the given edge point could be identified after many optimisation steps are involved.

Physics based segmentation technique partition real images into regions that correspond to surfaces or objects in the scene based on physical models for image formation [74].

2.8. Feature Extraction

Feature extraction involves extracting the meaningful information from the document images, which is most relevant for a given application; so that feature extraction reduces the storage requirement. Consequently, the system becomes faster and effective in retrieving information [40, 45]. Furthermore, Pavan and Jawahar [39]

describe that feature is a compact representation of the information content in multimedia data; such as image, audio and video.

Many different features have been employed in image processing and pattern recognition for representation of document images [45]. Generally, these features can be classified as: General features and Domain specific features [57]. General features are application independent features, and according to the abstraction level they can be classified as: pixel-level features, local features and global features. In pixel-level, features are calculated at the position of each pixel. Local features calculated over the results of subdivision of the image band on image segmentation. Global features are calculated over the entire image. On the other hand, domain specific features are application dependant features.

Based on the problem domain various feature representation techniques are forwarded by different researchers [1, 36, 48, 68].

Zagoris et al. [36] propose six features representation techniques that are extracted from every word which are capable of capturing the word similarities and discarding the small differences due to noise or font styles and sizes. These are:

- **Width to height ratio**: The width and height of the word form important information concerning the word shape.
- **Word area density**: This feature represents the percentage of the black pixels included in the word bounding box. Accordingly, word area density can be calculated by dividing the total black pixels in the word bounding box with the product of the width and height of the word bounding box.
- **Centre of gravity**: It represents the Euclidean distance from the word's centre of gravity (COG) to the upper left corner of the bounding box. To calculate its value, first, the horizontal and vertical centre of gravity should be determined, then normalized geometrical moments should be computed, and finally, as depicted in Equation 2.1, the centre of the gravity feature is equal to the Euclidean distance from the upper left corner of the image**.**

$$COG=\sqrt{C_x^{\,2} + C_y^{\,2}} \quad \text{..... (2.1)}$$

Where Cx is the horizontal centre and Cy the vertical centre of gravity.

- **Vertical projection**: This feature consists of a vector extracted from the smoothed and normalized vertical projection of the word image. Accordingly, the smoothed and normalized vertical projection has common width and height properties for all the word images and the following steps are used in calculating vertical projection feature values:
    1. The vertical projection VPorig[i] is defined as a number of black pixels that are reside in each width of the image.
    2. A new projection VP [i] is produced with width l and maximum height h, the value of l and h is considered as the mean of the width and height of the image. It is computed as shown in Equation 2.2.

$$VP[i] = \frac{h}{h_{max}} VPorig\left[i \frac{l_{orig}}{l}\right] \quad \text{..... (2.2)}$$

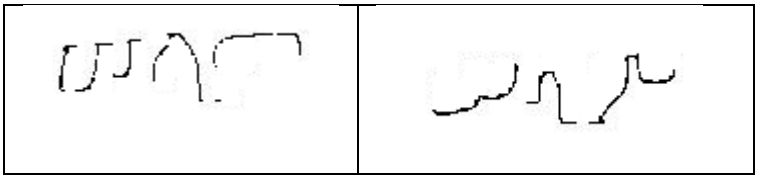
    Where l_{orig} is original width of the original projection VPorig and h_{max} is height of word bounding box.
- **Top–bottom shape projection**: the top–bottom shape projection can be considered as signatures of the word shape. In order to compute the top shape projection, the word image is scanned from top to bottom. If a black pixel is found for the first time, all the following pixels of the same column are converted to black. Similarly, the bottom shape projections are calculated, if the word image is scanned from bottom to top and all the pixels are converted to black until a black pixel is found.
- **Upper grid features**: The upper grid features (UGF) is a vector with binary values that are extracted from the upper part of each word image. In doing so, first the image's horizontal projection is extracted, and from it, the upper part of the word is determined by the following steps:
    1. Smooth the horizontal projection with a mean 5×1 mask.
    2. Starting from the top find the position i in the horizontal projection histogram V(i) where $V(i) \geq 3/2H$ and H is maximum height of the horizontal projection. If the position at i is below half of the word then the word has no upper part.

Rath and Manmatha [68], Million and Jawahar [48] and Balasubramanian et al. [1] considered two single-valued features representation techniques for extracting information from document images, i.e., one scalar value is calculated per column in

the original image. These single-valued features are: word profile and structural features.

Word profiles: Profiles of the word provide a common way of representing a word image for matching. Word profiles such as upper word, lower word, projection and transition profiles are used for word representation. Upper and lower word profiles extract part of the outlining shape of a word, while projection and transition profiles extract the distribution of black pixels along one of the two dimensions in a word image.

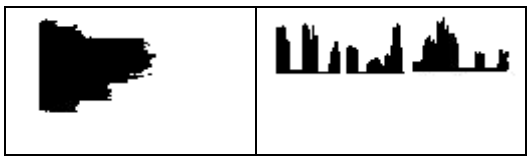
- **Upper/Lower word profile:** features are extracted by computing, for each image column, the distance from the upper/lower boundary of the word image to the closest "ink" pixel. Upper and lower word profile extract features of the word image **ሠላም** as shown in Figure 2.1.



(a) (b)

Figure 2.1 Features extracted in Upper (a) and Lower (b) word profiles

- **Projection profile**: each feature vector value is computed by summing up black pixel values in the corresponding image column. For instance, features extracted from the word image **ሠላም** in horizontal and vertical projection profiles are depicted in Figure 2.2.



(a) (b)

Figure 2.2 Extracted features in Horizontal (a) and Vertical (b) projection profiles

- **Partial projection profile:** computes the projection profiles for three horizontal strips in the original image: above, between and below the specified baselines. This can be shown in Figure 2.3.

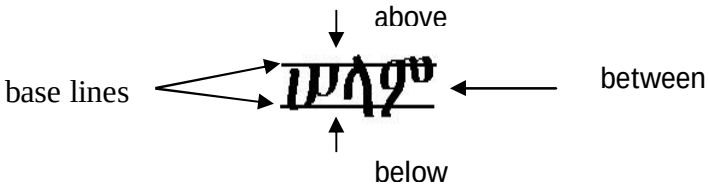


Figure 2.3 The three Horizontal strips of a word image

A single normalization factor is used for all the profiles, because the top and bottom strip can exhibit low variation for words with no ascenders or descenders. If all profiles would be individually normalized, slight errors in the baseline position estimation could seriously affect the result.

- **Background to ink transitions:** computes the number of transitions from the background to an "ink" pixel. This is determined by appropriate thresholding value.

Structural features: Structural features of the word images are used to match two word images based on some image similarities. Statistical moments, such as mean and standard deviation and region-based moments such as zeroth and first-order moments are employed for describing the shape of word images. Statistical moments measure the central tendency and distribution of ink pixels within word images. Region-based moments represent the entire shape region of the word image by using the pixels contained in that region. Generally, structural features are robust for document images with salt and pepper noisy artifacts.

According to Shutler [29], to characterize a grey level image so as to extract structural features an image can be considered as a two-dimensional continuous function f(x, y). With this assumption, as shown in Equation 2.3, the general form of a moment of order (p + q), evaluating over the complete image plane ξ is:

$$Mpq = \iint_{\xi} \psi_{pq}(x, y) f(x, y) dxdy \quad \text{..... (2.3)}$$

Where p,q = 0,1,2, ..., ∞

2.8.1. Feature normalization

Image features can be extracted using any of the specified techniques. However, to have image features insensitive to font types, sizes and styles, features should be normalized. Aksoy and Haralick [60] describe that feature normalization is a procedure which represents each feature components in a range of [0, 1]. It is required to approximately equalize ranges of the features and makes them to have approximately the same effect in the computation of similarity. Accordingly, the commonly used normalization techniques are: linear scaling to unit range and linear scaling to unit variance. Given the lower bound (l) and the upper bound (u) values of the feature component x, linear scaling to unit range normalizes the result $\bar{x}$ being in [0,1] as shown in Equation 2.4.

$$\bar{x} = \frac{x-l}{u-l} \quad \text{..... (2.4)}$$

The other normalization technique is linear scaling to unit variance. As depicted in Equation 2.5, it transforms the feature component x to random variable. If each feature is normally distributed, the value of $\bar{x}$ will be in range of [-1, 1].

$$\bar{x} = \frac{x-\mu}{\sigma} \quad \text{..... (2.5)}$$

Where μ and σ are sample mean and sample variance of that feature, respectively.

2.9. Matching Word Images

Image matching is a key component in almost any image analysis process. Matching in document images can identify the word images of the documents that are more similar to the query word through the extracted feature vectors [36]. According to Abbadeni [51], similarity can be defined as a mapping function between the feature vectors that represent the content of images and a positive real value which is chosen to quantify the degree of resemblance between the compared images. In another way, Kokare and Shirdhonkar [40] note that the task of matching algorithm is to compare query features with the indexed features of the word images that present in the database of documents.

Matching in document images is not easy; there are a number of factors to be considered [45]. Some of these are:

- What are the features used for matching?
- How the similarities between different word images are measured?
- How the optimal match is computed?

Optimal match can be computed using different matching algorithms: Cross correlation, Normalized cross correlation, Dynamic time warping, Euclidean distance, Manhattan distance and Cosine similarity [9, 17, 28, 30, 45, 67].

2.9.1. Cross correlation

According to Million [45], cross correlation is one basic technique for effective matching of words images with variable length. Accordingly, cross correlation is a statistical measure of how closely two different signals are related. The correlation function compares sequences of extracted feature vectors of two word images by aligning them against each other, and at each feature position the correlation coefficient between the corresponding pairs of feature vectors is computed.

Given sequence of feature vectors of two word images as: $F = f_1, f_2,...,f_M$ and $G = g_1,g_2, ... ,g_N$. The comparison between these two sequences of vectors produces an alignment vector W of length $h = m + n – 1$, this can be shown using Equation 2.6.

$$\begin{cases} \sum_{j=0}^{j} \left(D(F_j, G_k)\right) \text{ if } i = 0 \ldots m-1 \\ \sum_{j=0}^{j} \left(D(F_j, G_k)\right) \text{ if } i = m \ldots n-1, m \neq n \\ \sum_{j=0}^{(l-1)-i} \left(D(F_t, G_j)\right) \text{ if } i = n \ldots h-1 \end{cases} \quad \text{..... (2.6)}$$

Where k is (n - h)- i + j and t is (n - h) - (h - 1 - i) + 1 and D (i, j) is defined as:

$$D(i, j) = \begin{cases} 1 \; if \; i = j \\ 0 \; if \; i \neq j \end{cases}$$

For decision, the maximum of the resulting correlation values is taken as the best match among the various word images that are compared with the given user query.

2.9.2. Normalized cross correlation

The cross correlation between two signals is a standard approach to feature detection [28]. According to Tsai and Lin [17], normalized cross correlation (NCC) has been commonly used as a metric to evaluate the degree of similarity (or dissimilarity) between two compared images.

The main advantage of the normalized cross correlation over the cross correlation is that NCC is less sensitive to linear changes in the amplitude of illumination in the two compared images. Similarly, MacLean and Tsotsos [72] describe that the concept of normalized correlation was developed to combat the effect of different illumination levels on image. The NCC value is confined in the range between –1 and 1.

As depicted in Equation 2.7, normalized cross correlation overcomes the problem of correlation by normalizing (calculating the correlation coefficient) the image and query vectors to unit length, yielding a cosine like correlation coefficient [28].

$$\Upsilon(u, v) = \frac{\sum_{x,y}[f(x,y)-\bar{f}_{u,v}][t(x-u,y-v)-\bar{t}]}{\left\{\sum_{x,y}\{[f(x,y)-\bar{f}_{u,v}]^2[t(x-u,y-v)-\bar{t}]^2\right\}^{0.5}} \quad \text{..... (2.7)}$$

Where $\bar{t}$ is the mean of the template and $\bar{f}_{u,v}$ is the mean of f(x, y) in the region under the template.

2.9.3. Dynamic time warping (DTW)

DTW has been widely used in the speech processing, bio-informatics and also by the online handwriting communities to match single dimensional signals [67]. Million [45] describes that dynamic time warping (DTW) is a dynamic programming based matching procedure that aligns sequences of feature vectors and computes matching score for finding the dissimilarity among words.

DTW is a powerful matching technique to handle word variations by aligning and comparing word images with different sizes. In addition, Rath and Manmatha [67] note that DTW offers a more flexible way to compensate skew and slant angle variations.

According to Million [45], let two word images be represented as a sequence of features G = g_1, g_2,..., g_N and H = h_1,h_2,..., h_M, the DTW-cost between the two sequences is D (M,N), where M and N are the lengths of the two sequences. It is calculated using the recurrence equation as given in Equation 2.8.

$$D(i, j) = \min \begin{cases} D(i-1, j-1) \\ D(i, j-1) + d(i, j) \\ D(i-1, j) \end{cases} \quad \text{..... (2.8)}$$

Where, d (i, j)-- squared Euclidean distance-- is the cost in aligning the i^{th} element of G with the j^{th} element of H. It is computed using Equation 2.9.

$$d(i, j) = \sum (G_i - H_j)^2 \quad \text{..... (2.9)}$$

Nguyen et al. [50] describe that in DTW the words images are similar if the cost approaches to Zero or dissimilar if the cost is near to 1.

2.9.4. Cosine similarity

According to Goker and Davies [9], in cosine similarity, the similarity measure is usually the cosine of the angle that separates the two document vectors $\overrightarrow{D}$ and $\overrightarrow{Q}$. Similarly, Yue and Chew [80] describe that the similarity score between two compared document vectors is their scalar product (the product of the corresponding elements) divided by their lengths. This is equivalent to the cosine of the angle that separates two document vectors. The cosine formula is given in Equation 2.10.

$$\text{Score } (\overrightarrow{d}, \overrightarrow{q}) = \frac{\sum_{k=1}^{m} d_k . q_k}{\sqrt{\sum_{k=1}^{m} (d_k)2} \sqrt{\sum_{k=1}^{m} (q_k)2}} \quad \text{..... (2.10)}$$

Where $\overrightarrow{d}$ = (d_1,d_2,d_3,....,d_m) is a document index representation of each component of d_k ($1 \leq k \leq m$) and $\overrightarrow{q}$ = (q_1,q_2,q_3,...., q_m) is the query vector [9]. Accordingly, the cosine angle is 0 if the vectors are orthogonal in multidimensional space and 1 if the angle is 0 degree.

2.9.5. Euclidean distance

Euclidean distance is, also called Cartesian distance, the standard distance in Euclidean space [30]. Accordingly, it is calculated as the square root of the sum of the squares of the arithmetical differences of the corresponding coordinates of the two

points. The Euclidean distances between the images in the database and the queries are calculated and used for ranking [41]. Wang et al. [38] present that among all the image metrics, Euclidean distance is the most commonly used due to its simplicity.

Accordingly, let X and Y be two *M* by *N* images, X=(x^1,x^2, ... ,x^{MN}), Y=(y^1,y^2, ..., y^{MN}), where x^{kN+l}, y^{kN+l} are the gray levels at location (*k, l*). The Euclidean distance (x , y) ∈ *d x y* is given in Equation 2.11.

$$d_E^2(x, y) = \sum_{K=1}^{MN}(x^k - y^k)^2 \quad \text{..... (2.11)}$$

The query image will be more similar to the database of word images, if the computed Euclidean distance approaches to 0 [41].

2.10. Document Ranking

IR systems perform searching of documents based on user's query. Query is what the user conveys to the computer in an attempt to communicate the information need [16]. The search result of IR systems is depending on query-document similarity. This is accomplished by ordering documents relevant to user's query, on average, much higher in ranking than irrelevant documents that might be retrieved [71]. In other word, documents which are identified as possible relevant to user's query are ordered from most likely to least likely; this process is called ranking [26]. The main aim of ranking is to sort the retrieved documents according to their degree of relevance to the query provided by the users [4]. Consequently, ranking algorithm is at the core of IR systems and operates according to basic premises of document relevance in which different set of premises yield distinct document retrieval models [55].

In retrieving information from document images using a single word query, ranking the retrieved documents is possible based on their term frequency and inverse document frequency (TF*IDF) weights [4]. On the other hand, Punitha et al. [53] discuss that computing dissimilarity between the retrieved document images and the query image is used to rank documents according to their relevance. The dissimilarity values do not represent the degree of dissimilarities of the retrieved images with respect to the query; rather provide a means to rank the retrieved images.

2.11. Performance Evaluation of Information Retrieval Systems

Performance evaluation of IR system is important because it plays a significant role in judging the efficiency and effectiveness of the retrieval process. Moreover, it helps to know whether the designed and developed system performs as intended [9, 70]. According to Vihay and Gwang [70], the most comprehensive and objective performance evaluation approach should take many aspects of the retrieval process into consideration, such as:

- The resources used by the system to perform a retrieval operation.
- The amount of time and effort spent by a user to obtain the required information.
- The ability of the system to retrieve useful items.

Nevertheless, it is a common practice in research investigations to concentrate mainly on measures pertaining to the quality of the retrieval output, effectiveness, rather than efficiency. The reason is that, in retrieving information from a collection of documents the user query request is inherently vague; so, the retrieved documents are not exact answers and have to be ranked according to their degree of relevance to the query. Hence, information retrieval systems require the evaluation of how precise the answer set is [8].

According to Manning et al. [16], the standard approach to text retrieval system evaluation revolves around the notion of relevant and non-relevant documents. On this base, there are a number of evaluation metrics which are used for evaluating the performance of IR systems. The most frequently used once are Recall, Precision and F-measure [8, 16, 40]. Recall pertains to the percentage of all the relevant documents in the document collection that are retrieved. Precision refers the amount of retrieved document images that are relevant to the user query and F-measure is a single measure that computes the trade off between precision and recall.

Anurag et al. [8] note that it is misleading to use either only recall or only precision; both measures should be used together since system performance can vary widely depending on the number of documents retrieved. Accordingly, the performance

evaluation of document image retrieval systems has been mostly addressed by comparing Precision-Recall plots [62].

On the other hand, Vihay and Gwang [70] present that recall and precision are often conflicting goals; the increase in recall greatly affects precision and the reverse. Therefore, to tackle this trade off another effectiveness evaluation metric, F-measure, is forwarded.

2.12. Related Works

Due to limitations of optical character recognition systems a number of attempts are made by researchers to avoid the use of character recognition for various document image retrieval applications. Consequently, the research direction towards retrieving information without recognition from document images considers printed or handwritten documents which are encoded in different languages. In this thesis, we attempt to review researches that are done on Amharic, English and other languages.

2.12.1 Document image retrieval in English document images

Rath and Manmatha [68] are made an attempt on retrieving information from historical hand written manuscripts. They use word level matching—word spotting—for retrieving similar document images. Single-valued features, such as--projection profile, partial projection profile, upper/lower word profile, background to ink transitions and grayscale variance— are extracted for each word images. The extracted feature consists of one feature value per column of the word image. These features are slant/skew normalized. On this attempt, two feature sets are tested, Gaussian smoothing and Gaussian derivatives, for better performance. Then Dynamic Time Warping (DTW) algorithm is used to align and compare sets of features which have been extracted for word image retrieval. According to the authors, among the single-valued features, the upper word profile feature clearly performs best (64% of average precision), and the two next best features are projection profile and upper projection profile with performance of 50%. From the two feature sets Gaussian Smoothing performs best (62.78%). After testing the performance of each feature extraction techniques, they show that combined features (best 3 features) yield

performance superior to each feature (72.56%). Further investigation on feature extraction and matching scheme is recommended to increase the matching precision.

Zagoris et al. [36] describe an approach that performs the word matching directly in the document images bypassing character recognition and using word images as queries. The presented Document Image Retrieval System (DIRS) works based on word spotting. This system consists of offline (document image preprocessing, feature extraction) and online (query preprocessing , feature extraction and comparing and matching) procedures. In order to extract powerful features for the description of the word and query images seven features extraction techniques are used. These are: width to height ratio, word area density, centre of gravity, vertical projection, top–bottom shape projections, upper and lower grid features. Consequently, based on descriptors the Minkowski distance is calculated in order to find similar words through a matching procedure. The precision and the recall metrics are used to evaluate the performance of the proposed system. The overall mean precision and the mean recall values for the proposed system are 87.8% and 99.26%, respectively. The proposed system is designed to retrieve English printed document images, so further investigation on other languages is open for research.

An information retrieval system from document image corpus is designed by Yue and Chew [80]. The Left-to-Right Primitive String (LRPS) is used to represent feature of word images. Primitive features are described using Line-or-Traversal Attribute (LTA) and the Ascender-and-Descender Attribute (ADA). Once each word image is described by a primitive string, partial matching is performed to evaluate the similarity between the feature strings of the word image with the subsequence feature strings of the query. Then, an inexact string matching technique is utilized to measure the similarity between the two primitive strings generated from the two word images. Their similarities are computed by dynamic programming with recurrences.

The proposed system achieves a precision ranging from 89.06% to 99.36 % and a recall ranging from 85.67% to 99.19%, depending on the threshold values used. A lower threshold value results in a higher recall, but at the expense of lower precision. The reverse is true for a higher threshold value. Accordingly, if the goal is to retrieve as many relevant words as possible, then a lower threshold value should be set. On the other hand, for low tolerance to precision the threshold value should be raised. They

recommend future research on investigating robust matching technique that considers the setting of matching scores between different primitive codes.

2.12.2. Information retrieval from Ottoman, Hindi and Chinese document images

A method for searching historical Ottoman documents based on word level matching is presented by Ataer and Duygulu [22]. They propose a system which composes of segmentation and matching phases. Segmentation identifies lines and words from the document image using projection profile. The horizontal projection profiles and the vertical projection profiles with some threshold are used to identify lines and word images, respectively. The hierarchical matching technique is used to find the similar instances of the query to the word images. This technique composed of four feature sets: length of the word, quantized vertical projection profile, quantized vertical projection profile of ascenders and quantized vertical projection profile of descenders. These four features are used in four consecutive tests and each test discards the dissimilar word images for the next test. Euclidean distance between the vertical projection profile of the query image and the remaining word documents are computed to get the threshold value.

The performance of lines and word images segmentation result is 100% and 82%, respectively. The average mean precision value is used to evaluate the performance of the proposed matching scheme. An average mean precision value of 85.24% is recorded for all the words in the data set. To improve the retrieving performance of the proposed system, they recommend further investigation on better feature extraction techniques.

Anand et al. [6] propose an efficient indexing and retrieval scheme for searching in document image databases. Words images are automatically segmented; features are computed from words using a combination of scalar (ascenders, descenders and the aspect ratio), profile, structural and transform domain feature extraction methods. Profile and structural feature extraction methods compose projection profiles, background to ink transitions and upper and lower word profiles features. Fixed length descriptions of the extracted features are obtained by computing lower order coefficients of a Discrete Fourier Transform (DFT). DFT helps to make the

representation more robust by discard noisy high order coefficients. Similarly, features are extracted from the query by first converting it into an image through rendering. Then indexing is done using locality sensitive hashing (LSH) based on word image features computed at word level. After that, content similarity search is done by efficient nearest neighbor searching with content-sensitive hashing algorithm on a collection of Kalidasa's writings (Hindi documents). Precision, recall and F-score values are used to measure the performance of the proposed system. Accordingly, 96.17% precision, 92.95% recall and 94.36% F-score are recorded. Future improvements could include better feature selection such as machine learning techniques to include multiple fonts and styles.

A method for multilingual (for English and Chinese languages) text retrieval from document images without the use of OCR is proposed by [15]. In preprocessing noise, deskew and layout analysis are involved to remove headlines and pictures from the document image. Then connected component analysis is performed to identify character objects. Image features specifically the Vertical Traverse Density (VTD) and Horizontal Traverse Density (HTD) are used to store object features. Based on the extracted character features a hash table is created to assign a unique number to each distinctive n-gram and to keep track of the frequency of occurrences of all distinctive n-grams and this becomes the document vector. Then the n-gram vector is used to measure the similarity between two document vectors using the dot product of the document vectors.

To measure the performance of the system using precision and recall, three thresholds values (at 0.1, 0.15 and 0.2) are used. At threshold 0.2, the average precision is 100 percent but the recall is 44 percent for English documents and 39.6 percent for Chinese documents. On the other hand, using a lower threshold at 0.1, lower the precision percentage is recorded for both English (73.9) and Chinese (69.4) documents but the recall rate have been raised for documents printed in the two languages. Future work includes improving the feature extraction and matching methods robustness toward different fonts and noisy documents.

2. 12.3. Retrieving information within Amharic document image corpus

The pioneer for designing document image retrieval scheme for Amharic printed texts together with Hindi and English printed text documents are Million and Jawahar [48]. They present word image matching scheme that is crucial for content-based document image retrieval from printed text collection. To closely examine the performance of features and matching scheme, they prepare a model for document degradations by considering four categories of degradation: salt and pepper, cuts, blobs and erosion. Features in word images are represented in three categories: word profiles (Projection, transition, upper and lower profiles), moments (statistical moments, region-based moments) and transform domain (discrete Fourier descriptors). In these features category local features has a better representational capability than global features. Matching algorithms such as cross correlation and dynamic time warping (DTW) are investigated. These techniques have a great advantage in aligning and comparing word images with different sizes. The correlation function compares sequences of feature vectors of two word images by aligning them against each other and computes the correlation coefficient between two word images. DTW is a dynamic programming based approach which is used to align sets of sequences of feature vectors and compute a similarity measure. DTW algorithm based morphological analysis is used for matching a given word against its variants in the image domain. All extracted features are tested for performance; but combined features outperform individual features. There from, using combined local features DTW is more precise than cross-correlation. DTW based partial matching scheme is also tested to control morphological variants of a word for performance improvement.

Using local features of word images, they evaluate both cross correlation and DTW matching algorithms. The average performance of the DTW registered 89.58% recall, 90.81% precision and 90.19% F score and cross correlation achieves 76.43% recall, 78.83% precision and 77.61% F score. The result reflects that DTW out-performs cross correlation in all the three tested performance measures. The reason is that, cross correlation is relatively sensitive to small changes in the keyword being matched, while DTW compensates for most of the variations observed in printed word images during the alignment process. Nevertheless, further investigation on feature extraction and matching technique is proposed to come up with better

performance. After an attempt is made by Million and Jawahar [48] two further attempts are made by Mesfin [44] and Abreham [4] on retrieving information from Amharic document images.

Retrieving Amharic document image without explicit recognition is proposed by Mesfin [44]. First document images segmentation by thresholding is performed to identify lines and then words. On the other hand, textual query is rendered and converted to query word image. Accordingly, using word shape analysis (Vertical bar patter and area based), the feature of word images and query word are extracted. Each feature of the word image is normalized so that the word representations become insensitive to variation in size, font and various degradations popularly present in the text document. After that, normalized Euclidean distance is computed to get the similarity score between features of the word with features of the query. As a result, those document images containing the word image with a high score similarity values are retrieved.

Four threshold values are chosen for testing the matching scheme (0.042, 0.043, 0.044 and 0.045). The average registered recall and precision are 84.5% and 29.4% for threshold 0.045, 78.78% and 32.56% for threshold 0.044, 70.59% and 33.29% for threshold 0.043, and 59.14% and 35.0% for threshold 0.042. As a threshold value decreases the precision increase while the recall decreases. Future work is recommended to design efficient feature extraction and matching scheme that considers word variants and font types, sizes and styles difference.

Abreham [4] presents an algorithm for indexing document images in order to access information from Amharic document images corpus. At first, the inverse document frequency and the cutoff(threshold) values are calculated to remove the stopword images. If the cutoff value is greater than the inverse document frequency then the word image is considered as stopword, otherwise indexed. Then, the inverted index structure is used for indexing word images. Cosine similarity measurement is used to compare the degree of similarity between two word images on normalized feature vector of the two word images. He proposes prefix and suffix detection algorithm to reduce a word to its stem or root. The proposed algorithm uses the length of the root word image for identification. Finally, the retrieved documents are ranked based on the TF*IDF (term frequency and inverse document frequency) weighting scheme.

To measure the performance of the system four threshold levels (0.9, 0.91, 0.92 and 0.93) are used. Accordingly, best performance is registered at threshold value of 0.91 with precision (72.92%), recall (39.14%) and F-measure (47.82%). Better precision is not registered; since the proposed system clusters dissimilar words to the same group. Hence, further study on a better feature extraction and matching techniques are recommended to handle different format of the word images like, font sizes, types and styles and word form variations.

Document image retrieval schemes, for different languages, are proposed by different researchers. Different feature extraction, indexing, and matching techniques are forwarded to enhance the efficiency and effectiveness of document image retrieval system. However, some algorithms are limited for retrieving information from document images which are printed in one language. Besides, some proposed feature extraction and matching techniques are incapable to handle difference in font types, styles and sizes, and word form variations. Thereby, complexity of extracting and processing image feature is still a bottleneck in designing a better document image retrieval system. So, future investigation on feature extraction and matching schemes has a paramount significant in identifying robust techniques. Therefore, the main concern of the current research is exploring different matching and feature extraction techniques and designing a system which is insensitive to word variations, difference in font sizes, styles and types, and noises in document images. Consequently, the system can have a great contribution for searching information in real life document images.

CHAPTER THREE

FEATURE EXTRACTION AND MATCHING TECHNIQUES

Document image retrieval without recognition searches for relevant document, as per user request, by using only image features [48]. To retrieve information from document images a number of techniques are involved: prepocessing, segmentation, feature extraction, indexing, matching and ranking. Among these, implementing feature extraction and matching techniques are challenging tasks. This is because, the performance of document image retrieval system (DIR) greatly depends on selecting and implementing feature extraction and matching schemes that are insensitive to font variations and controls visual similarity between characters. Thus, as an extension to the previous works of Mesfin[44] and Abreham[4], the present study attempts to integrate feature extraction and partial matching schemes on to Amharic document image retrieval system.

3.1. Design of the System

Designing an Amharic document image retrieval system passes through a series of steps. The architectural view of the system is depicted in Figure 3.1. The proposed system consists of two parts, i.e., indexing and searching.

Indexing is an offline process that enables to organize document images using extracted word features. Indexing involves image processing, segmentation, feature extraction and term indexing. Given Amharic document images, the system first performs image processing that converts document images into digital format and then the digitized document images are binarized into digital and manageable representations. After that, line and word segmentation are performed, respectively. Then, features are extracted at word level. Finally, relevant word images are indexed and stored in index file.

The second part of the proposed system is online searching for relevant document images. It consists of text rendering, query processing, feature extraction, matching query to terms in index file and document ranking. Text rendering converts user's query to equivalent word image. Then, the query is binarized and its feature is extracted based on which the similarity measure finds similar index terms in index

file. Accordingly, if the computed similarity values fulfill the minimum threshold α, document images with index terms are retrieved and ranked according to their relevance for the given query.

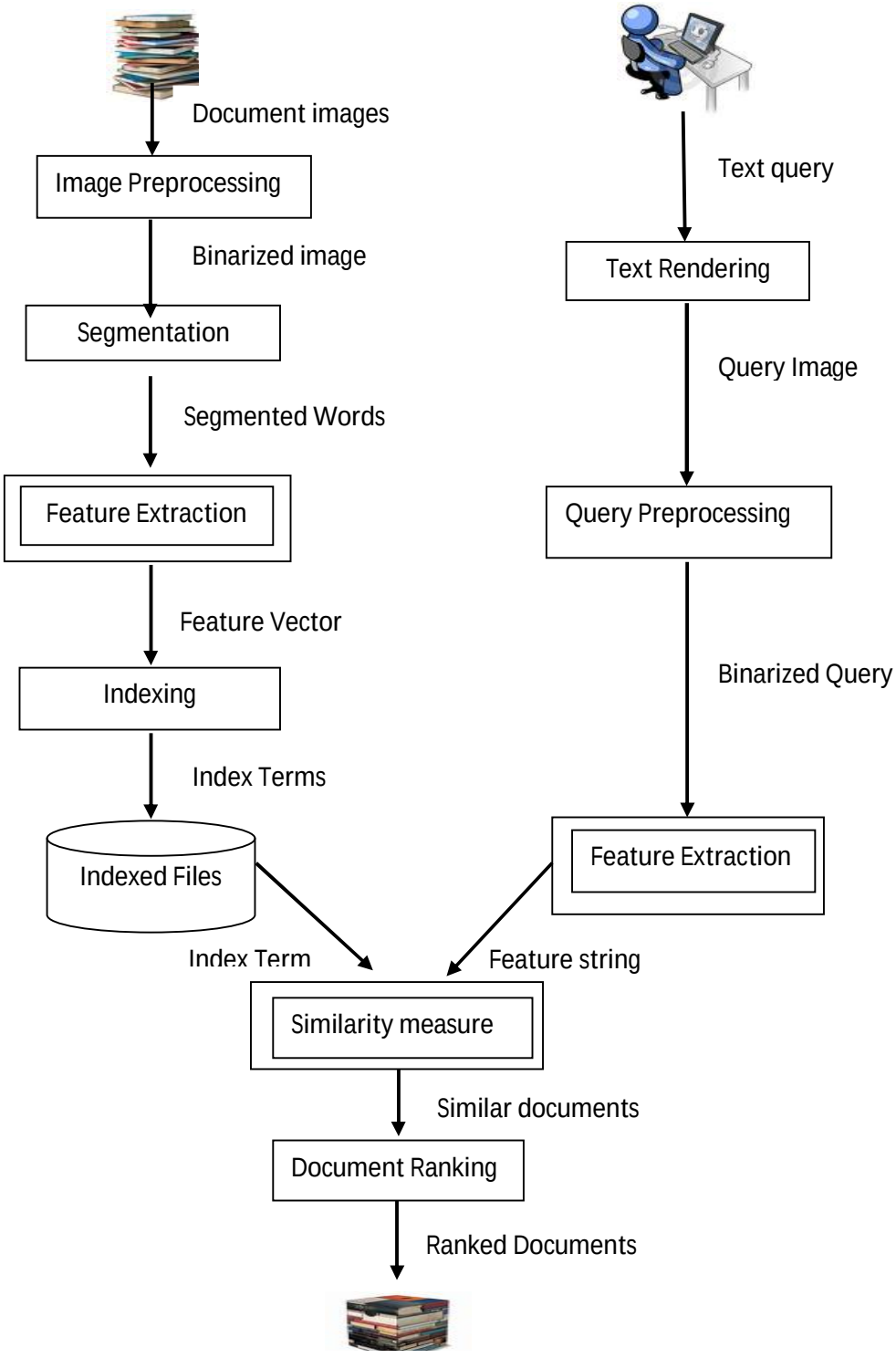


Figure 3.1. Architecture of the proposed Amharic document image retrieval system. Rectangles in double line represent the focus of the present work.

3.2. Feature Extraction Techniques

Since document image retrieval without recognition is based on image features, the following feature extraction techniques are used in the current study. These are projection profile (vertical and horizontal projections), shape projection (top and bottom shape projections), word profiles (upper and lower word profiles), vertical distance and top-bottom projection. All these techniques extract image features based on fixed threshold pixel intensity value, δ.

Projection profile extracts the distribution of black pixels along one of the two dimensions of word image. The extracted features of word image are represented in one dimensional vector. Similarly, all the remaining feature extraction techniques: word shape projection, word profile, vertical distance and top-bottom shape projection represent features in one dimensional vector.

3.2.1 Word projection

Word projection composes horizontal and vertical projections which extract word image features based on density of black pixels on horizontal and vertical base, respectively.

Horizontal projection refers to the total number of black pixels which are residing in each height of the image. This technique extracts features from document images which are scanned with the greyscale mode using 300dpi intensity. The pixel value of grey level images is represented in x and y coordinates with x and y taking integer values.

Given the top right corner (x_1, y_1) and left bottom corner (x_2, y_2) coordinates of the word image, horizontal projection works as shown in Algorithm 3.1. The algorithm first reads the integral value of right top corner and left bottom corner of the word image form a file. Then, for each value in the range of 1 to height and 1 to width, it counts the number of black pixels that reside in each coordinate (width, height) of the word image. After that, each value of the vector is normalized by total pixel value column-wise such that the computed normalized result ranges between 0 and 1.

1. *Read coordinate values from a file*
2. *For j = 1 to maximum height of the word image do*
3. *For i = 1 to maximum width of the word image do*

    If pixel intensity of Image (i, j) < = δ

    sum = sum+ 1

    Else

    count = count + 1

4. *End for*
5. *Normalize result of the sum with total count of pixel column-wise*
6. *End for*

Algorithm 3.1 Extracting features of word images by Horizontal projection

On the other hand, vertical projection refers to the total number of black pixels which are residing in each width of the image. It extracts word image features almost similar to horizontal projection. The only difference is that, in this case, instead of counting black pixels horizontally, pixels with intensity less than or equal to δ are counted in vertical base. For representing Vertical projection algorithmically, interchange step 1 and step 2 of the algorithm of Horizontal projection and change normalization at step 5 by width-wise, rather than column-wise. Similar to horizontal projection, this algorithm starts by reading the value of right top corner and left bottom corner of the word image form a file. Then, for each value in width range and in height range, it counts the number of black pixels that reside in each pixel (width, height) of the word image. After that, the counted black pixels are normalized by the sum of white and black pixels in each width of the word image.

3.2.2. Word profile

Word profile contains Upper word profile and Lower word profile. Both feature extraction techniques extract features by computing the distance from the upper or lower boundary of the word image to the closest black pixel intensity which is less or equal to δ.

As depicted in Algorithm 3.2, the upper word profile first reads the left top and right bottom coordinate values from a file. Then, starts to count the number of white pixels from the top to bottom until it come across pixel of intensity less than or equal to δ, i.e, black pixel. The top height value subtracts form the height at the first black pixel. After that, the subtracted pixel value is normalized by the total pixel value to form a normalized vector.

1. *Get the value of tc and bc*
2. *Read coordinate values from a file*
3. *For i =1 to maximum width of the word image do*
4. *For j = 1 to maximum height of the word image do*

if pixel intensity of Image (i, j) > δ

continue

else

fresult = j - tc

5. *End for*
6. *Normalized the value of fresult with bc - tc*
7. *End for*

*Where **tc** and **bc** refers to top value and bottom value of the black pixel.*

Algorithm 3.2 Upper word profile

Opposite to Upper word profile, the lower word profile starts to count white pixels from bottom to top until it reaches the closest black pixel. Its algorithmic representation differs from Upper word profile at step 3(in this case the for loop iterates from maximum height to 1) and fresult is computed by subtracting j from bc (bottom height value of the black pixel). As that of upper word profile, values are normalized to compress in the range [0, 1].

3.2.3 Word shape projection

In word shape projection feature extraction algorithm Top and Bottom shape projections are explored. Top and Bottom shape projections extract feature of word image by scanning the word image from top to bottom and bottom to top, respectively. Algorithm 3.3 represents the explored top shape projection technique.

1. *Read coordinate values from a file*
2. *For i =1 to maximum width of the word image do*
3. *For j = tc to bc do*

   if pixel intensity of Image (i, j) <= δ

   count = bc - j

   else

   continue

4. *End for*
5. *Normalized the value of count with height of the word image*
6. *End for*

*Where **tc** refers to top corner and **bc** refers to bottom corner value of the word image height*

Algorithm 3.3 Top shape projection

The Top shape projection can be computed by scanning from top to bottom. If a black pixel is found for the first time, all pixels down to the black pixel are converted to black. On the other hand, bottom shape projection is computed by scanning from bottom to top. Algorithm 3.3 is modified at step 3(insert for j=bc to tc) and count is computed by subtracting tc from j, rather than j from bc. In this case, if a black pixel is found for the first time, all pixels up on the black pixel are converted to black.

3.2.4 Vertical distance

This feature extraction technique is based on the features value of Upper and Lower word profiles. The values of Upper word profile and Lower word profile are subtracted to get the vertical distance between the upper shape and lower shape of the word images. Accordingly, the computed result is used to represent word images. Algorithmic representation of this feature extraction technique is shown on Algorithm 3.4.

1. *For i=0 to length of vector size generated in Algorithm 3.2*
2. *val1= vector element at (i) of vector generated in algorithm 3.2*
3. *val2= vector element at (i) of vector generated in algorithm 3.2*
4. *value 3= Math.abs(value1-value2)*
5. *Normalized value3 with top - bottom of the word image*
6. *End for*

Algorithm 3.4 Vertical distance

3.2.5. Top-Bottom shape projection

Top-Bottom shape projection is the extension of Top shape projection and Bottom shape projection. This technique captures the holistic shape of each character of the given word image. As represented in Algorithm 3.5, to get feature of top-bottom shape projection, the extracted features from top shape projection and bottom shape projection are summed up and normalized using linear scaling to unit range normalization technique. The normalized result represents feature value of Top-Bottom shape projection.

1. *For i=0 to length of vector size generated in Algorithm 3.3*
2. *val1= vector element at (i) of vector generated in algorithm 3.3*
3. *val2= vector element at (i) of vector generated in algorithm 3.3*
4. *value 3= value1+ value2*
5. *Add value3 on to vector Q*
6. *End for*
7. *for i=0 to size of Q*
8. *val4= (elem. at(i)- Min elem. of Q)/(Max elem. of Q- Min elem of Q)*
9. *add val4 on to a vector*
10. *end for*

Algorithm 3.5 Top-Bottom shape projection

3.2.6 Feature Combination

The appropriate feature extraction techniques are combined based on best step wise feature selection method. In best stepwise feature selection, subsequent features are selected based on combined improvement of feature extraction performance [84]. If, for example, there are three individual features A, B, and C and feature A is selected

as the first best feature, then the combination of A, B and A, C are evaluated. This continues until the performance of the combined feature converges.

3.3. Matching Techniques

Once word image features are extracted using the specified feature extraction techniques, matching should be performed to retrieve the required relevant documents from document image collections. In the present study Euclidean distance measure, Cosine similarity, Normalized cross correlation (NCC) and Dynamic time warping (DTW) matching algorithms are explored to compute the degree of similarity or dissimilarity between two word images. The Euclidean distance and Cosine similarity are adopted from Mesfin's [44] and Abreham's [4] previous attempts. These similarity measures are implemented by considering word lengths of the matched word images. Then, compare sequences of feature vectors one-to-one even though they might not be comparable. Consequently, they fail to handle word form variation and font size, type and style differences which affect the length of word image features.

Hence, Million [48] proposes DTW to handle the specified challenges in matching Amharic word images. Besides, NCC is investigated in this research to tackle the problem of word variants and font size, type and style differences. DTW and NCC algorithms align and find the best match between the query and word images. Therefore, they are basic techniques for effective matching of word images with variable feature lengths.

As depicted in Algorithm 3.6, in all the explored similarity measuring techniques, the similarity value is computed for the two comparable word image features. If the computed value is greater than the threshold set, the query is similar to the word image; otherwise the two word images are dissimilar.

1. *compute similarity of I_1 and I_2 using any of the four similarity measures*
2. *if similarity(I_1,I_2) > β*

   return "similar"

3. *else*

   return "dissimilar"

Where I_1 and I_2 are compared word features and β is the minimum threshold value

Algorithm 3.6. Similarity comparison

3.3.1. Normalized cross correlation (NCC)

NCC computes the similarity of two word images by aligning them against each other and then normalizing the result by standard deviation of their features. The formula of the proposed NCC technique is showed in Equation 3.1 [72].

$$Sim(I_1, I_2) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n}\left(x_{i\_} meanI_1\right) * \left(x_{j\_} meanI_2\right)}{\delta I_1 * \delta I_2} \quad \text{..... (3.1)}$$

Where $meanI_1$ and $meanI_2$ refer to the mean and δI_1 and δI_2 refer to the standard deviation of the compared word images. Given features vector of word images: H= h_1,h_2,.... h_M and I=i_1,i_2,..., i_N, where M and N are maximum feature lengths. Algorithm 3.7 presents how the NCC matching technique works.

1. *for i=0 to M*
2. *for j=0 to N*
   if h_i - I_j < γ
   compute the NCC by equation 3.1
   break
   else
   continue
3. *end for*
4. *end for*

Algorithm 3.7 Normalized cross correlation similarity measure

The NCC algorithm computes the similarity between two word images by aligning one word against the other. If the difference between each features of the word image is greater than the threshold γ, then compute the NCC, continue otherwise.

3.3.2. Dynamic time warping (DTW)

Similar to Normalized Cross Correlation, DTW matching algorithm is used to effectively compare word form variants and match word with font style, size and type variations. However, DTW measures the dissimilarity between word images.

DTW works by calculating the optimal cost of matching as shown in Algorithm 3.8. Then, the resulted optimal cost is used to get the final similarity value [48].

Accordingly, let the word images with length M and N are represented as a sequence of features F = f_1, f_2, ..., f_M and G = g_1,g_2,...., g_N. The optimal score of matching among F and G would be at D [M, N], where M and N represent the length of vector F and G, respectively.

1. $d(i,j)=\sum_k^M \left(F_i^k - G_j^k\right)^2$
2. *for i = 1 to N do*
   D(i, 0) = D(i – 1, 0) + d(i, 0)
3. *end for*
4. *for j = 1 to M do*
   D(0,j) = D(0, j - 1) + d(0, j)
5. *end for*
6. *for i = 1 to N do*
   for j = 1 to M do
   vertical(D(i – 1, j) + d(i, j)
   diagonal(D(i – 1, j - 1) + d(i, j)
   horizontal(D(i, j -1) + d(i, j)
   min(vertical, diagonal, horizontal)
7. *end for*
8. *end for*

Where d(i,j) compute the Euclidean distance between the vector element of two compared word images.

Algorithm 3.8 Dynamic time warping

3.4. Datasets and Performance Evaluation

To perform intensive experimentation on integrated Amharic document image retrieval system, 5016 word images which are printed in Amharic scripts are collected from real life noisy document images. In addition, a total of 534 clean word images are created which represent:

- Four Amharic font types (Power Geez, Nyala, ALEXethiopan and visual Geez Unicode)
- Three font styles (**bold**, *italic* and Normal)
- Three font sizes (10, 12 and 14) and tested to select the best matching schemes.

To perform further investigation on word form variants, 450 clean word images are collected to represent word variants and typographical errors of five Amharic words. These word images have similar font types, sizes and styles with the above 534 clean word images. To test insensitivity of feature extraction and matching schemes, 1243 word images are collected from noisy computer printed books, magazines and newspapers written in Amharic scripts. A total of 7243 word images are used to test the performance of the system. The summary is presented in Table 3.1.

Document Images	**Number of Word images extracted**
Books	3717
Newspapers	552
Magazines	1994
Clean documents	1180
Total	**7443**

Table 3.1. Types of extracted word images

To measure the performance of the explored system three evaluation metrics are used: precision, recall and F-measure. These performance evaluation techniques are universally accepted to measure the effectiveness of document image retrieval systems [16].

Meadow et al. [14] describe that precision and recall are calculated from relevance which is, in turn, typically measured by asking the user, intermediary, or third-party judge to evaluate the relevance of each of the information retrieved after a search.

Precision: refers to the percentage of retrieved document images that are relevant to the query [16]. It is defined in Equation 3.2.

$$\text{Precision (N)} = \frac{R}{R+I} \quad \text{..... (3.2)}$$

Where R and I are number of relevant and irrelevant matches, respectively.

Recall : pertains to the percentage of all the relevant document images in the search database which are retrieved [16] and represented in Equation 3.3.

$$\text{Recall (N)} = \frac{R}{TR} \quad \text{..... (3.3)}$$

Where TR is the total number of relevant documents available in the corpus, that may be retrieved or not retrieved by the IR system.

Vihay and Gwang [70] emphasize that recall and precision are often conflicting goals in the sense that if one wants to see more relevant items (i.e., to increase recall level), usually more non-relevant ones are also retrieved (i.e., precision decreases). Therefore to balance this trade off, another effectiveness evaluation parameter called F-measure is forwarded.

According to Manning et al. [16], F-measure is a single measure that computes where recall and precision are maximized. It is also called the weighted harmonic mean of precision and recall. Equation 3.4 represents the formula that is used to calculate the F-measure.

$$\text{F} = \frac{2PR}{P+R} \quad \text{..... (3.4)}$$

Where P and R are precision and recall, respectively.

Baeza-Yates and Ribeiro-Neto [55] note that the harmonic mean function assumes values in range of [0, 1]. It is 0 when no relevant documents have been retrieved and

is 1 when all retrieved documents are relevant. In addition, the harmonic mean assumes a high value when both recall and precision are high.

In the next chapter experimental results are presented to show the effectiveness of the various matching and feature extraction techniques. To select the best techniques F-measure is used as it identifies better schemes that optimize both recall and precision.

CHAPTER FOUR

EXPERMENTATION

The present study investigates the possibility of selecting and integrating better feature extraction and matching techniques that handle word form variations and font types, styles and size differences as well as noisy real life document images.

Given Amharic document images, the proposed system first performs image preprocessing that converts document images into digital format and then binarizes the document images into manageable representations. After that, line and word segmentation are performed and features are extracted at word level. Then, relevant words are indexed and stored in index file. When a user provides a textual query word, the query word is first rendered, the similarity between the query and the word image in the index file is computed and document images are returned according to their order of relevance. Image preprocesing, binarization, line and word segementation are adopted from Mesfin [44], but indexing and document ranking are adopted form Abreham [4]. Hence, in this study detailed experimentation is presented in order to select feature extraction and matching algorithms that have better performances.

4.1 Experimental Setup

The entire experimentation is done on a laptop computer with processor of Intel(R) Celeron(R) CPU 2.20 GHz, 2 GB RAM and Microsoft Window Vista operating system. To develop the prototype system Java NetBeans IDE 6.1 is used. This programming language has built in methods which are used to perform image processing. Five detail experiments are made to evaluate the insensitivity of feature extraction and matching techniques to various artifacts.

Experiment 1

The first test is performed to select best matching algorithm from the explored four matching algorithms. In doing so, all the matching algorithms are integrated with the eight feature extraction schemes. After that, each matching algorithm is tested for

performance on clean word images. To perform the experimentation 534 clean word images are used. These word images are printed in

- Three different font sizes: 10 pt., 12 pt. and 14 pt.
- Three different font styles: **Bold**, *Italic* and Normal and
- Four different Amharic font types: Visual Geez Unicode, Power Gee'z, Nyala and ALEXethiopan

Five query words are selected (a^TÇ?, mNGST, kF t¼, Që‰S and ì UbR) to evaluate the performance of matching techniques. The performance of each matching algorithm is measured in terms of precision (P), recall (R) and F-measure (F). The three performance measures are also used on the remaining experiments.

Experiment 2

The best performer matching algorithms is also tested on 96 word images containing typographical errors and redundant Amharic characters. For instance the word images a^T‡?¶and a^Ty„:¶ are created to represent typographical errors of the root word a^TÇ?¶. Besides, to evaluate the performance of the system for words that use different Amharic writing styles, words image mNGCT is created for the root word mNGST and ìHbR to the root word ì UbR. All the 96 word images are printed in four different font types, three different font sizes and styles. In the experimentation five query words are used to measure the performance of the matching scheme.

Experiment 3

This experiment is used to choose the best feature extraction algorithm using the best performer matching algorithm. First, to perform the experimentation feature extraction and matching algorithms are integrated and tested on 100 clean word images for five selected words (›=ƒÄåÁ, S"Óeƒ, Ÿõ}—, pÆe and TIu`) which are printed in four different font types, three different font sizes and three different font styles. Then, features are tested on 1247 word images which are extracted from noisy real life document images: one book and two magazines. After that, based on best stepwise feature selection method, features are combined and

evaluated for performance. To evaluate the performance of individual and combined features five query words are used.

Experiment 4

On the fourth experiment best matching and feature extraction algorithms are tested on clean word images for partial matching of word variants. To perform partial word image matching, word images are generated for five systematically selected words: aʌTÇ?, mNGST, kF t¼, Që‰S and ì UbR. For each selected root word eight word variants and the root word itself are created in four different font types, three different font sizes and three different font styles. Total of 450 clean word images and five query words are used in the experimentation.

Experiment 5

In the final experiment the selected matching, feature extraction and stemming schemes are integrated with the previous attempt and tested on noisy real life document images which are collected from books, magazines and newspapers. Document images with low, medium and high level noises are collected and total of 5016 word images are extracted. These words images are printed with different writing styles and font sizes.

4.2 Amharic Word Images Representational Schemes

In designing document image retrieval system, selecting good feature extraction algorithms have a paramount significance on improving the effectiveness of the retrieval system. For this reason, eight different feature extraction techniques are tested on the current work to investigate their representational capability of Amharic word images. These techniques are: Word projection (Horizontal and Vertical projections), Word profile (Upper and Lower word profiles), Shape projection (Top and Bottom shape projections), Vertical distance and Top-Bottom shape projection.

4.2.1 Word projection

This projection technique consists of Horizontal and Vertical projections. Horizontal projection technique captures the total number of black pixels which reside in each height of the word image. The Java sub code for this feature extraction technique is presented in Figure 4.1.

```
for (h = topCorner; h < bottomCorner; h++) {
        for (w = leftCorner; w < rightCorner; w++) {
             iterator.getPixel(w, h, pixel);
             pixelaverage = 0;
             for (int band = 0; band < nbands; band++)
                pixelaverage = pixel[band] + pixelaverage
             if (pixelaverage / nbands <= 128)
                pixelValLV++;
              countWV++;       }
           countTotalV = countWV + pixelValLV;
          float VlineValV = (float) (pixelValLV)/ countTotalV;
          VlinepixValh.addElement(VlineValV); countWV=0;pixelValLV=0;
}
```

Figure 4.1 Extracting features using Horizontal projection

Given the top left (height = topCorner and width = leftCorner) and bottom right (height = bottomCorner and width = rightCorner) coordinates of the word image, the Horizontal projection iterates in each height and counts the number of black pixels at each width of the word image. The total counted black pixels are then normalized by the sum of black and white pixels that reside at each height of the word image. Normalization helps to have representation of the word image that is less sensitive to variation in sizes, fonts and types that exist in real document images. The value of each normalized feature is between 0 and 1. This value adds on to a vector to get all useful feature representation of the word image.

Similar to Horizontal projection, features extraction in Vertical projection captures the total number of black pixels. To represent Vertical projection in Java sub code, one has to interchange the top two for loops in Figure 4.1. This feature extraction technique works exactly the same as horizontal projection but black pixels are captured on each width of the word image rather than its height. On each iteration extracted feature is normalized and adds on to a vector until all features of the word image are generated.

Figure 4.2 represents the application of Horizontal and Vertical projection on the word image mNGST (font type-Power Geez, font size-12 and font style-normal).

(a) **(b)**

Figure4.2 Representation of the word mNGST *in Horizontal (a) and Vertical (b) projections*

Horizontal projection captures the content of the black pixel on the given word image without considering character shapes. Similarly, the vertical projection captures the top projection of the word image based on black pixels but generates the shape of each character in the image without inside loops and exact stroke representations.

4.2.2 Word profile

Upper word profile and Lower word profile are the two word profile feature extraction techniques. Upper word profile extracts feature of the given word image based on the upper shape representation of the word image. The Java sub code for this technique is depicted in Figure 4.3.

Accordingly, having the top left and bottom right coordinates of the word image, iteration is performed for each width of the image until it gets a black pixel on the height of the word image. When the black pixel is found on the way, the distance of this pixel from the top most black pixel is measured and recorded as a feature value. This feature value is then normalized by the distance between top most and the bottom most black pixels of the word image. Finally, the normalized feature value is added on to a vector until all useful features of the word image are extracted.

```
for (w = leftCorner; w < rightCorner; w++) {
        for (h = topCorner; h < bottomCorner; h++) {
          iterator.getPixel(w, h, pixel);  pixelaverage = 0;
          for (int band = 0; band < nbands; band++) {
             pixelaverage = pixel[band] + pixelaverage; }
          if (pixelaverage / nbands <= 128) {
              pixelValtb = h-top;
             break;   }
          else if(h==bottomCorner-1)
              pixelValtb=lower-top; break;
          else
             ;
    }
     float VlineVaTb = (float) (pixelValtb)/(lower-top);
     VlinepixValtb.addElement(VlineVaTb);
}
```

Figure 4.3 Feature extraction using Upper word profile

On the other hand, Lower word profile extracts features of the word image based on the lower shape representation of the image; this is the only difference with upper word profile. Its Java sub code is similar to Upper word profile but the first and the second bold lines are replaced with

for(h=bottomCorner-1; h >= topCorner; h--) and *pixelValtb= lower-h,* respectively. Hence, in Lower word profile, iteration starts from the lower height of the word image and continues until a black pixel is found at first time on the way. If a black pixel exists, its difference with the bottom most black pixel is computed as feature value of that specific pixel element. Then, normalization is performed in similar fashion to Upper word profile.

The pictorial representation of the extracted feature of the word image mNGST using Upper and Lower word profile are shown in Figure 4.4. This word image is printed in font type – Power Gee'z, style-normal and size-12.

(a) **(b)**

Figure4.4 Representation of the word mNGST *in Upper (a) and Lower (b) word profiles*

Upper and Lower word profiles represent word images by capturing the top and bottom boundary of the word image, but not its precise shape.

4.2.3 Word shape projection

In word shape projection Top shape projection and Bottom shape projection are investigated. Top shape projection extracts feature of the word image by scanning the word image from top to bottom. The Java sub code for this feature extraction technique is depicted in Figure 4.5.

```
for (w = leftCorner; w < rightCorner; w++) {
    for (h = topCorner; h < bottomCorner; h++) {
        iterator.getPixel(w, h, pixel); pixelaverage = 0;
        for (int band = 0; band < nbands; band++)
            pixelaverage = pixel[band] + pixelaverage;
        if (pixelaverage / nbands <= 128) {
            pixelValttb=lower-h; break;  }
        else
            ;
    }
     float VlineVaTb = (float) (pixelValttb)/(lower - top);
    VlinepixValttb.addElement(VlineVaTb);   }
```

Figure 4.5 Top shape projection feature extraction

At the time of scanning, if there is a black pixel on the way, it captures the height value of that image at that specific pixel location. Then, it subtracts this value from height value of the bottom most black pixel of the word image. The subtracted value represents the feature of the word image at each specific location. In other word, if a black pixel found for the first time on the way down to the height, all pixels below that specific location are converted to black. After that, the extracted feature value is normalized by the vertical distance of top most and bottom most black pixels of the word image and adds on to a vector until the end of the iteration.

The second word shape projection is Bottom shape projection. Opposite to top shape projection, bottom shape projection extracts feature of the word image by scanning the image from bottom to top. For representing Bottom shape projection, the Java sub code the first and the second bold lines in Figure 4.5 are replaced by: f*or(h=bottomCorner-1;h>=topCorner;h--) and Pixelvaltb= h-top;*, respectively. In this case, if there is a black pixel at the time of scanning, it captures the height value at that pixel location. After that, the extracted value subtracted from height of the top most black pixel of the word image to get the word feature at that specific location. Then, the extracted feature value is normalized by the highest vertical distance of the word image and adds on to a vector until the end of the iteration.

The extracted feature of the word image mNGST which is printed in font size of 12, font type of Power Gee'z and font style of normal using Top shape and Bottom shape projection is represented in Figure 4.6. As depicted on the figure the Top shape projection representation is similar at the top with Upper word profile but on this case once a black pixel is found pixels below it are converted to black. Bottom shape projection looks Lower word profile at the bottom, but in Bottom shape once balk pixel is found, the pixels above it are converted to black.

(a) **(b)**

Figure 4.6 Representation of the word image mNGST *in Top (a) and Bottom (b) shape projections*

Top shape projection and Bottom shape projection extract the holistic top and bottom shape representation of the word image; however, they are incapable to represent the exact shape and structure of the word image.

4.2.4 Vertical distance

This feature extraction technique computes the vertical distance between the top and bottom profile of the word image by subtracting features that are extracted using Upper and Lower word profile. Given the feature vectors of Upper word profile and Lower word profile, the Vertical distance is computed using Java as shown in Figure 4.7. Accordingly, the computed vertical distance values are used as feature representation for the word image.

```
for (int k = 0; k < VlinepixValtb.size(); k++)
    {
      double val1 = (VlinepixValbst.elementAt(k));
      double val2 = (VlinepixValtb.elementAt(k));
      float vdistance= (float)Math.abs(val1 - val2);
      pwwv.print(vdistance + ",");
    }
```

Figure 4.7 Extracting features using Vertical distance

The Graphical representation of this feature extraction technique on the image word mNGST with font size of 12, font type Power Gee'z and font style normal is shown in Figure 4.8.

The bold lines show the vertical distance between upper and lower profiles

Figure 4.8 Graphical representation of the word mNGST *in Vertical distance*

Vertical distance is the combination of Upper and Lower word profiles; hence, it captures the upper and lower boundary of the word image simultaneously. Nevertheless, the exact shape and structural representation of the given word image is not captured.

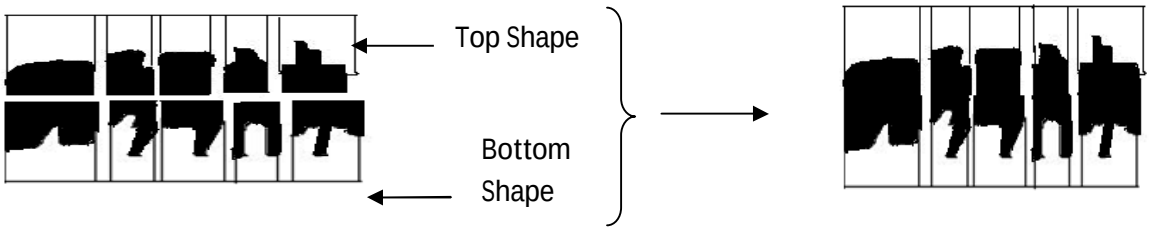
4.2.5. Top-Bottom shape projection

This feature extraction technique is the combined features of Top shape projection and Bottom shape projection. Given the feature vectors of Top shape projection and Bottom shape projection, feature vectors are summed up and normalized with linear scaling with unit range. The normalized results are then add on to a vector as Top-Bottom shape representation of the given word image. The Java code which shows the normalization of the summed result is attached on APPENDIX 4. Figure 4.9 shows the Java sub code for summing up the feature vectors of Top shape projection and Bottom shape projection.

```
for (int k = 0; k < VlinepixValttb.size(); k++)    {
        double val1 = (VlinepixValttb.elementAt(k));
        double val2 = (VlinepixValbt.elementAt(k));
         float sum= (float)(val1 + val2);
        MinMax.addElement(sum);       }
```

Figure 4.9 Extracting features using Top-Bottom shape projection

For the word image mNGST (font size of 12, font type Power Gee'z and font style normal), Figure 4.10 shows its features that are extracted using Top-Bottom shape projection.



(a) (b)

Figure 4.10 Top shape (a) and Bottom shape (b) projections of the word image mNGST

Top-Bottom shape projection is the combined feature of Top and Bottom shape projections. This shape projection tries to capture the approximate shape of the word image, but the extracted feature contains enlarged shape without internal loops.

4.3 Similarity Measures

Once word features of the document image are extracted, the appropriate matching technique should be used to design an effective Amharic document image retrieval system. On the present research, four different matching techniques are tested. These are Euclidean distance, Cosine similarity, Normalized Cross Correlation (NCC) and Dynamic Time Warping (DTW). NCC and DTW align and find the best match between query and word images. On the other hand, Cosine similarity and Euclidean distance measure the similarity between the query and the word image without aligning their feature against each other. On the next section, the highest registered precision, recall and F-measure at the selected best threshold value are discussed for each matching algorithms.

4.3.1 Euclidean distance

Mesfin [44] uses Euclidean distance to measure the dissimilarity between query and word images. This matching technique measures the dissimilarity between word images without aligning their features against each other. The Euclidean distance between two compared word images is calculated as represented in Figure 4.11.

```
if(dataQ.length <= addressflag) {
        for(int h=0; h < dataQ.length; h++)  {
             querval = Double.parseDouble(dataQ[h]);
             sumofQ =  querval +  sumofQ;
            dataval = Double.parseDouble(dataI[h]);
           diff = (Math.pow((dataval-querval),2)) + diff;
          sumofD = dataval + sumofD; }
    similarity= (Math.sqrt(diff))/(sumofD + sumofQ);}
```

Figure 4.11 Euclidean distance measure

This matching algorithm is evaluated and its performance is represented in Table 4.1. Using this matching algorithm, the highest mean F-measure value of 66.17% is scored with horizontal projection followed by Vertical distance and Top shape projection with F-measure of 53.46% and 47.18%, respectively.

The Euclidean distance matching algorithm measures the dissimilarity between two compared word images without aligning their feature values against each other. Thus, it favours for word images with identical feature values. The performance measure for the remaining font types, sizes and styles are not satisfactory. On that account, using this technique for proper match of word variants and handling font size, type and style differences are challenging.

4.3.2 Cosine similarity

Cosine similarity measures the similarity between word images without aligning their feature vector values. This feature matching technique computes the cosine angle between the compared word images features as shown in Figure 4.12.

```
if(dataQ.length > addressflag) {
        for(int h=0; h < addressflag; h++)  {
          dataqr = Double.parseDouble(dataQ[h]);
              dataimg = Double.parseDouble(dataI[h]);
             dotproduct = (dataqr * dataimg) + dotproduct;
             sqrdata = (dataqr * dataqr) + sqrdata;
           sqrdatacompare = (dataimg * dataimg) + sqrdatacompare;
         }
        similarity = dotproduct / Math.sqrt(sqrdata * sqrdatacompare);
}
```

Figure 4.12 Cosine similarity measure

Query word Image	Lower word profile			Upper word profile			Top shape projection			Bottom shape projection			Vertical projection			Horizontal projection			Vertical distance			Top-Bottom shape pro.		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
a^TÇ?¶	37.5	37.5	37.5	62.5	38.41	42.61	25	40	30.76	50	50	50	25	28.57	26.66	62.5	83.33	71.42	50	100	66.66	37.5	27.27	31.57
mNGST	62.5	45.45	52’63	50	44.44	47.05	75	28.57	41.37	37.5	75	50	37.7	60	46.15	62.5	62.5	62.5	87.5	70	77.77	50	36.36	42.1
kF t¼	12.5	33.33	18.18	12.5	50	20	25	100	40	25	66.66	36.36	12.5	100	22.22	62.5	38.46	47.61	12.5	100	22.22	50	66.66	57.14
Që¿S	25	100	40	62.5	71.42	66.66	50	100	66.66	37.5	100	54.54	37.5	100	54.54	87.5	41.17	55.99	37.5	60	46.15	50	66.66	57.14
ì UbR	25	100	40	12.5	100	22.22	50	66.66	57.14	25	100	40	37.5	100	54.54	87.5	100	93.33	37.5	100	54.54	62.5	20.83	31.25
Average	**32.5**	**63.25**	**37.66**	**40**	**60.85**	**40.7**	**45**	**67.04**	**47.18**	**35**	**78.33**	**46.18**	**30**	**77.77**	**40.81**	**72.5**	**65.09**	**66.17**	**45**	**86**	**53.46**	**35**	**43.55**	**43.83**

Table 4.1 Performance of Euclidean distance

Query word Image	Lower word profile			Upper word profile			Top shape projection			Bottom shape projection			Vertical projection			Horizontal projection			Vertical distance			Top-Bottom shape pro.		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
a^TÇ?¶	25	66.66	36.36	37.5	27.27	31.57	25	28.57	26.66	25	11.76	15.99	25	100	40	50	57.14	53.33	25	66.66	36.36	25	25	25
mNGST	25	50	33.33	37.5	20	26.08	62.5	14.28	23.25	25	12.5	16.66	25	25	25	62.5	62.5	62.5	25	50	33.33	25	11.76	15.99
kF t¼	25	100	40	25	66.66	36.36	25	40	30.76	25	100	40	12.5	100	22.22	50	100	66.66	12.5	100	22.22	25	33.33	28.57
Që¿S	25	100	40	62.5	33.33	43.47	50	80	61.53	62.5	26.31	37.03	37.5	75	50	62.5	41.66	50	25	100	40	50	33.33	40
ì UbR	37.5	50	42.85	37.5	100	54.54	50	12.9	20.51	12.5	14.28	13.33	37.5	100	54.54	75	100	85.71	37.5	25	30	50	15.38	23.57
Average	**27.5**	**73.33**	**38.5**	**40**	**49.45**	**38.4**	**42.5**	**35.12**	**32.54**	**30**	**32.97**	**24.61**	**27.5**	**80**	**38.37**	**60**	**72.26**	**63.64**	**45**	**86.66**	**32.38**	**35**	**23.76**	**26.62**

Table 4.2 Performance of Cosine similarity

The test result of Cosine similarity is listed in Table 4.2. The highest F-measure value of 63.64% is registered with horizontal projection. Lower word profile and Upper word profile come next with F-measure of 38.50% and 38.4%, respectively.

Cosine similarity matches word images by computing the similarity angle between their feature values. Therefore, this matching scheme favours for word images which are printed in the same font types or whose feature values are approximately equivalent. Thus, in most of the test performed by each feature extraction algorithms word images printed in font type ALXethiopian and Power Gee'z, font style Italic and Normal, and font size 12 are retrieved for the query word image which is printed in font type Power Gee'z, style Normal and size 12. Other font types, sizes and styles that have significance difference with the query feature values registered poor performance measure, even if, it refers similar word as the query.

4.3.3. Normalized cross correlation (NCC)

This feature matching technique matches two word images by aligning their features against each other. The Java sub code for implementing this matching scheme is depicted in Figure 4.13.

```
if(dataQ.length > addressflag) {
   for(int im =0; im<addressflag;im++) {
        for(int h=count; h < dataQ.length; h++) {

            dataq = Double.parseDouble(dataQ[h]);

            datawim = Double.parseDouble(dataI[im]);

            if(Math.abs(datawim - dataq) < 0.12) {

                diffq = (dataq - meanQ);  diffim= (datawim - meanI);

                 mulrslt = (diffq * diffim) + mulrslt; count=h; break;   }
            else
                    count=h+1
            }
       }
       similarity = mulrslt / (stdevI * stdevQ);
}
```

Figure 4.13 NCC Matching technique

NCC computes the similarity between two word images if the length difference of the aligned features is less than the fixed threshold, i.e, < 0.12. Otherwise, it continues until the aligned features fulfil the specified condition. Here, the fixed threshold value is the feature length difference of the compared word images at each alignment. This value is selected based on the intensive experimentation made. The best precision and recall value on different font types, sizes and styles is recorded at this threshold value.

Then, the value differences with their mean are computed for the query and the image features. After that, the results are multiplied and summed up. Finally, the total sum is normalized by the multiplied value of standard deviation of both the query and the word image features.

The test result of NCC is listed in Table 4.3, the highest mean F-measure of 66.31% is recorded with Lower word profile. The second highest mean F-measure of 65.42% is scored with Bottom shape projection. The third highest F-measure of 60.14% is registered with Top-Bottom shape projection.

The main problem with this matching scheme is that, the selected threshold is fixed and sometimes failed to handle font sizes, types and style differences. So, when NCC is integrated and tested on different fonts, it favours for features whose compared feature value difference is less than the fixed threshold, whatever their similarities look like.

4.3.4 Dynamic time warping (DTW)

DTW is a powerful matching technique to take care of word variations and font differences by aligning and comparing word images features.

Given feature sequences of two word images, F = f_1,f_2..., f_M and G = g_1,g_2...., g_N. DTW first generates a two dimensional matrix using the given feature sequences. The generated matrix is length of D[M][N], where M and N are feature lengths of the two word images. The DTW minimal cost value is calculated using dynamic programming as depicted in Figure 4.14. The value at D[M][N] is the optimal cost of matching between the two word images.

```
for(int i=1;i<addressflag;i++)  {
    for(int j=1;j<dataQ.length;j++) {
        double ver = D[i-1][j] + d[i][j];
        double dia = D[i-1][j-1] + d[i][j];
        double hri = D[i][j-1] + d[i][j];
       double temp= Math.min(dia, hri);
      D[i][j]= Math.min(temp, ver);
    }
 }
```

Figure 4.14 DTW matching

According to the experimentation made and depicted in Table 4.4, the highest average F-measure of 100% for DTW is registered with Upper word profile. Vertical projection and Top-Bottom shape projection comes next with F-measure of 97.33% and 95.08%, respectively. This matching technique properly handles font type, size and style differences; with a better feature extraction scheme it enables to design a good IR system that can achieve a better performance.

Query word Image	Lower word profile			Upper word profile			Top shape projection			Bottom shape projection			Vertical projection			Horizontal projection			Vertical distance			Top-Bottom shape pro.		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
aʰTÇ?¶	62.5	83.33	71.42	50	80	61.53	75	85.71	80	75	85.71	80	75	50	60	75	6.74	12.37	62.5	71.42	66.66	62.5	100	76.92
mNGST	75	100	85.71	75	100	85.71	37.5	100	54.54	62.5	100	76.92	100	26.66	42.1	75	16.21	26.66	50	57.14	53.33	75	100	85.71
kF t¼	75	50	60	50	57.14	53.33	50	50	50	62.5	62.5	62.5	62.5	8.19	14.49	25	2.77	4.16	50	80	61.53	37.5	30	33.33
QëiS	50	40	44.44	62.5	50	55.55	0	0	0	50	22.22	30.76	12.5	0.72	1.37	12.5	2.77	4.16	75	18.75	30	50	30.76	38.09
ì UbR	87.5	58.33	70	62.5	31.25	41.66	50	21.05	29.62	62.5	100	76.92	75	18.75	30	0	0	0	62.5	83.33	71.42	50	100	66.66
Average	**70**	**66.33**	**66.31**	**60**	**63.67**	**59.55**	**42.5**	**51.35**	**42.83**	**62.5**	**74.08**	**65.42**	**65**	**20.86**	**37.59**	**37.5**	**5.58**	**9.54**	**60**	**62.12**	**56.58**	**55**	**72.14**	**60.14**

Table 4.3 Performance of Normalized cross correlation (NCC)

Query word Image	Lower word profile			Upper word profile			Top shape projection			Bottom shape projection			Vertical projection			Horizontal projection			Vertical distance			Top-Bottom shape pro.		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
aʰTÇ?¶	87.5	100	93.33	100	100	100	100	100	100	100	100	100	100	100	100	62.5	71.42	66.66	87.5	100	93.33	100	100	100
mNGST	100	100	100	100	100	100	87.5	87.5	87.5	100	100	100	100	100	100	75	100	85.71	87.5	100	93.33	100	100	100
kF t¼	87.5	87.5	87.5	100	100	100	62.5	100	76.92	87.5	87.5	87.5	87.5	100	93.33	75	100	85.71	50	100	66.66	87.5	100	93.33
QëiS	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	87.5	77.77	82.35	87.5	87.5	87.5	100	100	100
ì UbR	75	100	85.71	100	100	100	62.5	62.5	62.5	50	100	66.66	87.5	100	93.33	62.5	100	76.92	50	100	66.66	75	100	85.71
Average	**90**	**97.5**	**93.3**	**100**	**100**	**100**	**82.5**	**90**	**85.38**	**87.5**	**97.5**	**90.38**	**95**	**100**	**97.33**	**72.5**	**89.83**	**79.45**	**72.5**	**92.5**	**81.49**	**92.5**	**100**	**95.08**

Table 4.4 Performance of Dynamic time warping (DTW)

4.3.5. Selecting matching algorithm

Four different matching algorithms are explored in the present research. Hence, to select best performing matching algorithm intensive experimentations have been made as shown above in Tables 4.1, 4.2, 4.3 and 4.4. Using the eight features, the performances of the four matching algorithms are evaluated.

DTW registered the best performance of 100% mean F-measure on Upper word profile, since it matches word images by aligning one word against the other. This matching algorithm is insensitive to changes in font types, sizes and styles. NCC comes next with highest performance measure of 66.31% with Lower word profile. This matching algorithm also aligns feature values; however, it uses fixed threshold value which sometimes fails to handle font types, sizes and styles difference. On the other hand, Euclidean distance and Cosine similarity measures are unable to align and match features of word images, so the F-measure values of 66.17% and 63.64% are registered on Horizontal projection for both schemes, respectively.

In general, the results reflect that DTW outperforms the remaining three matching algorithms in matching word images which are printed in different font types, styles and sizes of Amharic script. So, it is used for further experimentation to select optimal feature extraction scheme and partial matching technique.

4.3.6 DTW on typographical errors and word writing styles variations

The performance of DTW is tested on word images which represent typographical errors and difference in word writing styles. As the experimentation shown in Table 4.5, the best average F-measure of 74.58% is registered. DTW performs better for handling typographical errors. Nevertheless, writing styles variations on Amharic characters are not properly handled. So, future investigation has to be made on the system to enhance its insensitivity on redundant Amharic characters.

Query words	R	P	F
a^TÇ?¶	100	100	100
mNGST	62.5	100	76.92
kF t¼	43.27	77.77	56
Që‰S	100	100	100
ì UbR	25	100	40
Average	**66.15**	**95.55**	**74.58**

Table 4.5 DTW on typographical errors and word writing style variations

4.4. Selecting Feature Extraction Algorithm

Once the best performing matching algorithm (i.e, DTW) is selected, the performances of feature extraction schemes are evaluated individually and in combination. First, all the eight feature extraction algorithms are evaluated individually on clean word images. According to the experimentation made and shown in APPENDIX 3, the highest average F-measure of 100% for DTW achieved with four feature extraction techniques: Lower word profile, Upper word profile, Bottom shape projection and Top-Bottom shape projection. The second highest performance is registered on Vertical projection with 98.66% F-measure. Top shape projection comes next with F-measure of 96.31%. Then, F-measure 95.38% is recorded for Vertical distance. Finally, the least performance is recorded on Horizontal projection with 88.27% of F-measure. The results show that most of the feature extraction algorithms are insensitive to difference due to font types, sizes and styles.

Consequently, to select the best feature extraction technique, all the eight feature extraction algorithms are tested on low and medium level noisy real life document images. The test result is listed in Table 4.6. Among the eight features, the best performance of 93.9% average F-measure is achieved by Vertical distance. Bottom shape comes next with mean F-measure of 90.66%. Top-Bottom and Vertical projection are third and fourth with mean F-measure of 89.33% and 82.66%, respectively.

After that, the best performer, Vertical distance, feature extraction algorithm is combined with other six feature extraction algorithms, other than Horizontal projection, using best stepwise feature selection method. Horizontal projection cannot combine with the other seven feature extraction algorithms since the feature vector size of this algorithm is not compatible with others. Therefore, feature value transformation should be done to combine Horizontal projection with the remaining algorithms. However, this is not tackled in this research.

On the second step the combination of Vertical distance with Vertical projection achieved F-measure of 95.02%. This best combination is again selected and recombined with the remaining five feature extraction techniques. At the third step the best performance is registered when Vertical distance and Vertical projection are combined with Lower word profile with mean F-measure of 95.52%. After that, the combination of Vertical distance, Vertical projection and Lower word profile with the remaining four uncombined feature extraction schemes result in less performance, i.e F-measure of 92.83%. Since the result starts diminishing best stepwise feature selection stops feature combination here.

The average F-measure scores of the best individual and combined feature extraction techniques are shown in Table 4.7. Accordingly, the best performance of 95.52% is registered by the combination of Vertical distance, Vertical projection and Lower word profile. This result shows that, useful features of word images which are not extracted using individual features are extracted by combined features. Thus, the combined features show better performance than individuals. Besides, DTW is achieved good matching performance on low and medium level noisy document images. However, testing is performed on word images that are extracted from noisy real life document images without employing noise removal technique. Therefore, integrating noise removal scheme is crucial for improving the performance of these combined features.

Query word Image	Lower word profile			Upper word profile			Top shape projection			Bottom shape projection			Vertical projection			Horizontal projection			Vertical distance			Top-Bottom shape projection		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
a^TÇ?¶	42.85	100	59.99	85.71	85.71	85.71	100	87.5	93.33	100	87.5	93.33	100	100	100	100	8.53	15.71	100	87.5	93.33	100	87.5	93.33
mNGST	50	50	50	0	0	0	0	0	0	100	100	100	100	100	100	100	28.57	44.39	100	100	100	100	100	100
Që‰S	40	40	40	60	60	60	100	83.33	90.9	80	80	80	80	80	80	40	25	30.76	100	71.42	82.84	80	80	80
a§J	100	100	100	70	70	70	100	100	100	100	100	100	100	87.5	93.33	0	0	0	100	87.5	93.33	100	87.5	93.33
ugR	80	66.66	73.72	80	80	80	100	26.31	41.62	80	80	80	40	40	40	0	0	0	100	100	100	80	80	80
Average	**62.57**	**71.33**	**64.72**	**59.14**	**59.14**	**59.14**	**80**	**59.42**	**65.17**	**92**	**89.5**	**90.66**	**84**	**81.5**	**82.66**	**16.42**	**12.45**	**18.17**	**100**	**89.28**	**93.9**	**92**	**87**	**89.33**

Table 4.6 Test result of each feature extraction algorithms on noisy real life documents

Query word Image	Vertical distance			Vertical distance & Vertical projection			Vertical distance & Vertical projection & Lower W. P.			Vertical distance & Vertical projection & Lower Word Profile & Bottom shape projection		
	R	P	F	R	P	F	R	P	F	R	P	F
a^TÇ?¶	100	87.5	93.33	100	100	100	100	100	100	100	100	100
mNGST	100	100	100	100	100	100	100	100	100	100	100	100
Që‰S	100	71.42	82.84	100	83.33	90.9	80	100	88.88	100	83.33	90.9
a§J	100	87.5	93.33	100	87.5	93.33	100	100	100	100	70	82.35
ugR	100	100	100	100	83.33	90.9	80	100	88.88	100	83.33	90.9
Average	**100**	**89.28**	**93.9**	**100**	**90.83**	**95.02**	**92**	**100**	**95.52**	**100**	**87.33**	**92.83**

Table 4.7 Test results of the best performing individual and combined features on noisy real life documents

4.5 Stemming Word Images Using DTW

To handle word variants in document images a modified DTW stemming technique is proposed. The stemming technique handles prefixes and suffixes in word images. In doing so, the optimal alignment line between the two word images is constructed by DTW matching algorithm. After that, backtracking is done to identify and remove suffixes and prefixes.

If the two word images are similar, the optimal line lies on the diagonal; otherwise the optimal line moves in either vertical or horizontal directions. Accordingly, when consecutive equal diagonal elements exist, they are counted as diagonal moves. This helps to get the total diagonal moves by discarding suffixes and prefixes. After that, the counted diagonal move divides the non-diagonal moves and the resulted value is used to perform partial matching. Hence, as the result approaches to zero the two word images are similar. For example, to match two word images mNGST and ymNGSTnT, the prefix and suffixes of the later word image should be stemmed. The backtracking result for the two word images are presented in Figure 4.15.

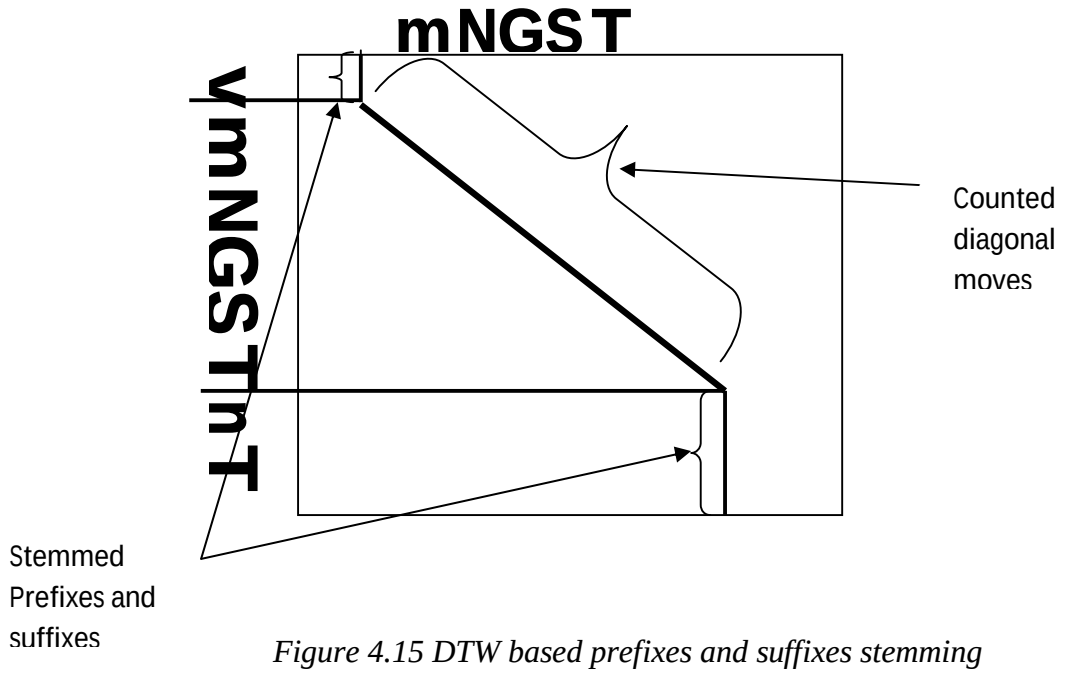


Figure 4.15 DTW based prefixes and suffixes stemming

DTW based stemming approach and the best performer combined features are used to perform the experimentation. Before that, to easily observe the effect of the

investigated stemming technique, experiment is made without stemming on clean word images that contain word variants. Words and their variants are listed in APPENDIX II. The result is listed on Table 4.8. Accordingly, the best performance of 52.27% average F-measure is registered.

On the other hand, when we test the same word images using DTW based stemming technique, as depicted in Table 4.8, the best performance of 72.48 % F-measure is recorded. This shows that matching word image with its word variants by DTW based stemming exceeds matching without stemming by 20.21%. Thus, the proposed DTW based stemming technique handles word form variants with more than 70% of retrieval performance. However, diagonal counter sometimes count diagonal moves that are not on the exact diagonal alignment of the DTW, so the performance of the system is affected. Therefore, identifying better backtracking technique which improves the retrieving performance needs further investigation.

Query words	Before stemming			After stemming		
	R	P	F	R	P	F
aˆTÇ?¶	69.31	95.31	80.26	86.36	71.69	78.35
mNGST	50.57	80	61.97	81.60	73.19	77.17
kF t¼	31.25	43.85	36.49	48.75	79.59	60.46
Që¿S	52.27	95.83	67.64	56.81	94.33	70.92
ì UbR	8.04	100	14.89	62.06	96.42	75.52
Average	**42.28**	**82.99**	**52.27**	**67.11**	**83.04**	**72.48**

Table 4.8 Test result of DTW before and after stemming

4.6. Performance of the System

The selected matching, feature extraction and stemming techniques are integrated to the previous Amharic document image retrieval system and tested on noisy real life document images that are gathered from books, magazines and newspapers. The integrated Amharic document image retrieval system is evaluated using query words which are representative of low, medium and high level noisy document images. The experimental result of the system is shown in Table 4.9.

Noise levels	R	P	F
Low	100	68.88	80.46
Medium	100	50	66.66
High	83.33	44.04	55.82

Table 4.9 System performance on low, medium and high level noisy document images

Accordingly, the maximum F-measure of 80.46% is achieved on low level noisy document images. On the other hand, 66.66% and 55.82 % F-measures are registered on medium and high level noisy document images, respectively. This implies that, the existence of noise on document images highly affect the performance of the integrated system. On that account, integrating noise removal techniques with the current attempt enhances its performance. Besides, further investigation on more precise stemming technique helps to improve the effectiveness of the system.

4.7 Findings and challenges

Designing effective Amharic document image retrieval system that are insensitive to word variants, font types, sizes and styles difference are based on integrating effective feature extraction and matching techniques. Thus, this study investigates four matching schemes. DTW matching algorithm outperforms all the remaining three algorithms in handling font type, size and style difference. In addition, eight feature extraction algorithms are analyzed individually for performance on noisy word images using DTW matching algorithm. After that, based on best stepwise feature selection technique features are combined. The result shows that combined features of Vertical distance, Vertical projection and Lower word profile register better perform than individual features. To handle word form variations DTW based stemming technique is designed. The experimental result shows that the DTW based partial word matching controls word image variants.

Furthermore, the selected matching, feature extraction and partial matching schemes are integrated with the previous Amharic document image retrieval system and tested on low, medium and high level noisy document images. Without the application of noise removal method, a promising result is registered.

The performance of the current system outperforms the two previous attempts of Mesfin[44] and Abreham[4]. The best performance of the system that is designed by [44] registered 82% mean average precision with direct query to word image match. His system is tested on low level noise document images that are printed in specific font type, style and size. Even if the present study take into account four font types, three font sizes and three font styles, mean average precision of 100% is registered for query to word matching on low and medium level noisy document images. In addition, the performance of the DTW stemming technique outperforms the Cosine based stemming technique which is proposed by [4]. Effectiveness of the cosine based stemming technique shows F-measure value of 41.59% on low level noise document images and with single font type, size and style. Nevertheless, the current DTW based stemming technique registers 80.46% F-measure on low noisy document images and with four font types, three font sizes and three font styles. This shows that current investigated system has a paramount significance for designing applicable Amharic document image retrieval system.

However, the performance of the system is significantly affected by the existence of noise and different levels of degradations in real life document images. Hence, integrating noise reduction technique improves the performance of the document image retrieval system. Besides, investigating high performance stemming technique that controls word variations in situation where there are printing variations and document quality differences is open for research. Furthermore, intensive investigation on segmenting and document ranking should be made on the future to handle noisy document images.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

Information is retrieved from document images by designing recognition based or recognition free IR systems. Nowadays researchers attempt to design recognition free retrieval system for different languages such as English, Chinese, Hindi, etc. The main intent of this research is also to design Amharic document image retrieval system without recognition. On the present work an attempt is made to explore feature extraction and matching techniques that are crucial to enhance the performance of the document image retrieval system.

5.1 Conclusion

This study aims to design a system which handles word variants and difference in font types, sizes and styles by proposing effective feature extraction and matching algorithms.

To this end, eight feature extraction algorithms are explored, namely; Horizontal projection, Vertical projection, Upper word profile, Lower word profile, Top shape projection, Bottom shape projection, Vertical distance and Top-Bottom shape projection. Using these feature extraction algorithms, one dimensional feature vector is constructed. Then, features are tested individually and in combination following best stepwise feature selection technique.

Also, four matching algorithms: Normalized Cross Correlation (NCC), Dynamic Time Warping (DTW), Cosine similarity and Euclidean distance are evaluated. These matching algorithms are tested on clean word images which are printed in four different font types, three different font styles and three different font sizes. From the four matching algorithms; DTW outperforms the remaining with 100% F-measure on clean word images. This shows that DTW is insensitive to font types, sizes and styles difference. This matching technique is further used to identify the best feature extraction algorithm. Thus, for individual features, DTW matching is made on 1267 noisy word images. The best performance of 93.9% F-measure is registered for individual feature Vertical distance. However, best performance of 95.52% F-measure

is registered by combining Vertical distance, Vertical projection and Lower word profile. This implies useful features of word images which are not extracted using individual features are extracted by combined features.

To handle word variants DTW based stemming algorithm is designed. Both prefixes and suffixes are removed using backtracking technique. This stemming technique improves the performance of the retrieval system by F-measure of 20.21%, when we compare before and after stemming performance results.

The selected feature extraction, matching and stemming techniques are integrated with the previous Amharic document image retrieval system and evaluation is made on low, medium and high level noise document images. These noisy Amharic document images are representative of different font types, sizes and styles. Based on the experimentation, the maximum F-measure of 80.46% is achieved on low level noisy document images. On the other hand, 66.66% and 55.82 % F-measures are registered on medium and high level noise document images, respectively. However, on the investigated system noise removal technique is not integrated, thus the performance of the system is deteriorated. Integrating noise removal scheme improves the effectiveness of the system. In addition, to handle multiple words with the same meaning semantic based word image retrieval needs to be investigated.

5.2 Recommendation

Even if this research has a paramount significance in retrieving the required information from Amharic printed document images, there are some issues that need to be further investigated to develop efficient and effective system.

- Pre-processing techniques like noise removal helps to increase the effectiveness and efficiency of the system. Introducing these techniques improve the performance of Amharic document image retrieval system.
- Document images are retrieved given a single query word. Thus, retrieving document images using multiple word queries by incorporating logical statements are still open for research.

- Infix detection and removal are not tackled on the current work. However, the performance of the system can be improved by introducing better partial matching scheme that is insensitive to infixes.
- The present work tries to handle word variants, font sizes, styles and type differences. However, there are synonymous and multiple words that need to be identified to enhance the retrieval performance. Thus, further studies have to be conducted to handle synonymous and multiple words in the document images.
- The explored feature extraction and matching algorithms are tested on table, line and picture free Amharic document images. Hence, future works should focus on handling all these artifacts.
- Further investigation should be done on integrating skew detection technique that enhances effectiveness of the Amharic document image retrieval system.
- Better segmentation and document ranking techniques should be considered in the future.

REFERENCES

1. Balasubramanian A., Million Meshesha and Jawahar C.V. (2006). *Retrieval from Document Image Collections.* In Assoc. for Pattern Recognition Workshop on Document Analysis Systems (DAS), pp. 1–12.
2. Pollitte A.S. (1989) *Information Storage and Retrieval Systems*. Elis Horwood limited, England.
3. Abdelsapor, A., Adly N., Darwish K., Emam O. and Nagi M. (2006) *Building a Heterogeneous Information Retrieval Test Collection of Arabic Document Images.* The Fifth International Conference on Language Resources and Evaluation (LREC), Genoa Italy.
4. Abreham Gebretsadik (2010) *Searching in Amharic Document Image Corpus.* M.Sc Thesis, Addis Ababa University, Addis Ababa, Ethiopia.
5. Aleksandar Ferenc (1985) *Writing and Literature in Classical Ethiopic (Ge'ez).* Literatures in African Languages, Wiedza Powszechna State Publishing House, Warsaw, Poland.
6. Anand Kumar, Jawahar C.V. and Manmatha R. (2007) *Efficient Search in Document Image Collections.* In Proc. of the Asian Conf. on Computer Vision (ACCV), Part I, LNCS 4843, pp. 586–595.
7. Anurag Bhardwaj, Srirangaraj Setlur and Venu Govindaraju (2009) *Keyword Spotting Techniques for Sanskrit Documents.* Lecture Notes in Computer Science, Vol.5402, pp. 403-416.
8. Anurag Seetha, Sujoy Das and M.Kumar (2005) *Information Retrieval Performance Evaluation.* ADIT Journal of Engineering 2(1): 60-65.
9. Ayse Goker and John Davies (eds.) (2009) *Information Retrieval: Searching in the 21st Century*. John Wiley and Sons Ltd, United Kingdom.
10. Bruce G. Trigger (1998) *Writing System*: *A Case Study in Culture Evolution.* Norwegian Archaeological Review, 31(1): 39-62.
11. Jeong C.B. and Kim S.H. (2004) *A Document Image Preprocessing System for Key Word Spotting*. Lecture Notes in Computer Science, Vol. 3334, pp. 440-443.
12. Carnell Hampton, Time Persons, Chris Wyatt and Yoachun Zhang (1998) *Survey of Image segmentation.* Available from: http://www.citeseer.nj.nec.com/hampton98survery.html [Accessed 10/12/2010].

13. Charles Oppenheim, Joan Stenson and Richard M. S. Wilson (2003) *Studies on Information as an Asset*. Journal of Information Science, 29(3): 159-166.
14. Charles T.Meadow, Bert R.Boyce and Donald H.kraft (2000) *Text Information Retrieval System*. 2nd ed. Academic Press: California, USA.
15. Chew Lim Tan, Weihua Huang, Zhaohui Yu and Yi Xu (2002) *Imaged Document Text Retrieval Without OCR*. IEEE Transaction on Pattern Analysis and Machine Intelligence, 24(6): 838-844.
16. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schutze (2008) *Introduction to Information Retrieval.* Cambridge University Press, New York, USA.
17. Tsai D. M. and Lin C. T. (2003) *Fast Normalized Cross Correlation for Defect Detection*. Pattern Recognition Letters, 24 (15):2625–2631.
18. Daniel Yacob (2005) *Developments towards an Electronic Amharic Corpus.* In Proceedings of the 12th Conference on Traitement Automatique des Langues Naturelles (TALN), Dourdan, France.
19. David Doermann (1998) *The Indexing and Retrieval of Document Images*: *A Survey*. Computer Vision and Image understanding. 70(3): 287-298.
20. Seid H. and Gamback B. (2005) *A Speaker Independent Continuous Speech Recognizer for Amharic. I*n Proc. of Interspeech, Lisboan, Portugal.
21. Ermias Abebe (1998) *Recognition of Formatted Amharic Text Using OCR Techniques*. M.Sc Thesis, Addis Ababa University, Addis Ababa, Ethiopia.
22. Esra Ataer and Pinar Duygulu (2006) *Retrieval of Ottoman Documents*. In 8th ACM SIGMM International Workshop on Multimedia Information Retrieval. pp. 155-162.
23. Salton G. and McGill (1983) *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company.
24. Gabriella F. Scelta (2001) *The Comparative Origin and Usage of the Ge'ez Writing System of Ethiopia.* Available from: www.thisisgabes.com/images/stories/docs/paper_gabriella.pdf [Accessed 4 /12/2010].
25. Geoffery Sampson (1985) *Writing Systems*. Stanford University press, California, USA.
26. Gerald J. Kowalski and Mark Maybury (2000*) Information Storage and Retrieval Systems*. 2nd ed. Kluwer Academic Publisher, Norwell, USA.

27. Hans Jensen (1969) *Sign, Symbol and Script: An Account of Man's Efforts to Write.* Translated from Germany by Goerge Unwin. 3rd ed. George Allen and Unwin Ltd., London, England.
28. Lewis J. P. (1995) *Fast Normalized Cross correlation.* Vision Interface, pp. 120–123. Available from: http://citeseer.ist.psu.edu/lewis95fast.html [Accessed 24/12/2010].
29. Jamie Shutler (2002) *Statistical Moments-An Introduction*. Available from: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/SHUTLER3/node1.html [Accessed 26/01/ 2011].
30. Jianfeng Wang and Xiaorong Zhao (2010) *A New Approach for Image Retrieval with Integrated Euclidean Distance and Rotational Correlation.* IEEE International Conference on Digital Object Identifier. pp. 391-395.
31. Joanna Mantel-Niecko (1985) *Ethiopian Literature in Amharic, Literatures in African Languages*. Wiedza Powszechna State Publishing House, Warsaw, Poland.
32. John Cowell and Fiaz Hussan (2003) *Amharic Character Recognition using a Fast Signature Based Algorithm*. Proceedings of the IEEE conference on Image Visualisation, London, UK, pp. 384–389.
33. John W.Creswell (2009) *Research Design*, Third Edition, SAGE Publication Inc., USA.
34. Jonathan J. Hull (1994) *Document Image Matching and Retrieval with Multiple Distortion-Invariant Descriptors.* International Workshop on Document Analysis Systems, pp. 383-400.
35. Kareem Darwish and Ossama Emam (2006) *Retrieving Arabic Printed Document*. A Survey IBM Technology Development Centre, Cairo, Egypt.
36. Konstantinos Zagoris, Kavallieratou Ergina and Nikos Papamarkos (2010) *A Document Image Retrieval Syste*m. Engineering Applications of Artificial Intelligence, Vol. 23, Issue 6, pp. 872-879.
37. Lawrence Lo (n.d.) *Ancient Scripts.* Available from: http://www.ancientscripts.com [Accessed 16 /11/2010].
38. Liwei Wang, Yan Zhang and Jufu Feng (2005) *On the Euclidean Distance of Images*. IEEE Transaction, Pattern Analysis and Machine Intelligence, 27(8): 1,334–1,339.

39. Pavan Kumar M. N. S. S. K. and Jawahar C. V. (2004) *Information Processing from Document Images*. Information Technology: Principles and Applications (ed) Ray A. K. and T. Acharya, Prentice Hall of India, New Delhi. pp. 522—547.
40. Manesh B. Kokare and Shirdhonkar M. S. (2010) *Document Image Retrieval*: *An Overview.* International Journal of Computer Application. 1(7):114-119.
41. Manesh Kok'are, Chatterji B.N. and Biswas P.K. (2003) *Comparison of Similarity Metrics for Texture Image Retrieval*. In Conference on Convergent Technologies for Asia-Pacific Region. 2. pp. 571 – 575.
42. Marvin L.Bender, Sydney W.Head and Roger Cowley (1976) *Language in Ethiopia: The Ethiopian Writing System*. Oxford University Press.
43. Meadow, Boyce C. T., B. R. and Kraft D. H. (1999) *Text Information Retrieval Systems*. 2nd ed. San Diego: Academic Press, United Kingdom.
44. Mesfin Worku (2009) *Amharic Document Image Retrieval without Explicit Recognition*. M.Sc Thesis, Addis Ababa University, Addis Ababa, Ethiopia.
45. Million Meshesha (2008b) *Recognition and Retrieval from Document Image Collections*. Ph.D Dissertation, International Institute of Information Technology, Hyderabad, India.
46. Million Meshesha and Jawahar C. V. (2007a) *Indigenous Scripts of African Languages*. African Journal of Indigenous Knowledge Systems, 6(2): 132-142.
47. Million Meshesha and Jawahar C. V. (2007b) *Optical Character Recognition of Amharic Documents.* African Journal of Information and Communication Technology, 3(2): 53-66 10.
48. Million Meshesha and Jawahar V.C (2008a) *Matching Word Images for Content-Based Retrieval from Printed Document Images*. International Journal on Document Analysis and Recognition. 11(1): 29-38.
49. Senthilkumaran N. and Rajesh R. (2009) *Edge Detection Techniques for Image Segmentation*: *A Survey of Soft Computing Approach.* International Journal of Recent Trends in Engineering, 1(2): 250-254.
50. Nhu Van Nguyen, Jean-Marc Ogier, Salvatore Tabbone and Alain Boucher (2009) *Text Retrieval Relevance Feedback Techniques for Bag of Words Model in CBIR.* World Academy of Science, Engineering and Technology.
51. Noureddine Abbadeni (2003) *A New Similarity Matching Measure Application to Texture-Based Image Retrieval*. The 3rd international Workshop on texture analysis and synthesis. pp. 1-6.

52. Omniglot (n.d.) *Writing Systems and Languages of the World*. Available from: http://www.omniglot.com/writing/ethiopic.htm [Accessed 09/11/2010].
53. Punitha P., Naveen and Guru D.S. (2006) *Indexing and Retrieval of Document Images by Spatial Reasoning*. In Proceedings of the Third International Conference on Distributed Computing and Internet Technology (ICDCIT'2006), Bhubaneswar, India. LNCS 4317, pp. 457 – 464.
54. Rangachar Kasturi, Lawrence O'gorman and Venu Govindaraju (2002) *Document Image Analysis*: *A primer*. Sadhana, 27(1): 3–22.
55. Ricardo Baeza-Yates and Berthier Ribeiro-Neto (1999) *Modern Information Retrieval.* ACM Press: New York, USA.
56. Robert M. Losee (1989) *Minimizing Information Overload*: *The Ranking of Electronic Messages.* Journal of Information Science, 15(3): 179-189.
57. Ryszard S. Chora´s (2007) *Image Feature Extraction Techniques and Their Applications for CBIR and Biometrics Systems.* International Journal of Biology and Biomedical Engineering. Issue 1, Vol. 1, pp. 6-16.
58. Collin S.M.H (1994) *Dictionary of computing.* 2nd ed. Peter Collin Publishing ltd. UK.
59. Raja S.V.Kasmir, Abdul Khadir A.Shaik and Riazahamed S.S. (2009) *Moving toward Region-Based Image Segmentation Techniques: A study*. Journal of Theoretical and Applied Information Technology, 5(1): 81-87.
60. Selim Aksoy and Robert M. Haralick (2000) *Effects of Feature Normalization on Image Retrieval*. IAPR International Conference on Pattern Recognition.
61. Shijian Lu, Linlin Li and Chew Lim Tan (2008) *Document Image Retrieval through Word Shape Coding*. IEEE Transaction on Pattern Analysis and Machine Intelligence 30(11): 1913 – 1918.
62. Simone Marinai (2006) *A Survey of Document Image Retrieval in Digital Libraries.* 9th Colloque International Francophone Sur l'Ecritet le Document (CIFED), pp. 193–198.
63. Simone Marinai, Emanuele Marino and Giovanni Soda (2007) *Exploring Digital Libraries with Document Image Retrieval.* In ECDL, pp. 368-379.
64. Suresh Pabboju and A.Venu Gopal Reddy (2009) *A Novel Approach for Content-Based Image Indexing and Retrieval System using Global and Region Features.* IJCSNS International Journal of Computer Science and Network Security, 9(2): 119-130.

65. Syed Saqib Bukhari, Mayce Ibrahim Ali Al and Faisal Shafait (2010) *Document Image Segmentation using Discriminative Learning over Connected Components.* DAS '10, Boston, MA, USA.
66. Thomas Bloor (1995) *The Ethiopic Writing System: A Profile*. Journal of the Simplified Spelling Society, J19: pp.30-36.
67. Toni M. Rath and Manmatha R. (2002) *Word Image Matching Using Dynamic Time Warping*. Technical Report MM-38. Centre for Intelligent Information Retrieval, University of Massachusetts Amherst.
68. Toni M. Rath and Manmatha R. (2003) *Features for Word Spotting in Historical Manuscripts.* Proceeding of the 7th International Conference on Document Analysis and Recognition, pp. 512–527.
69. Wu V. and Manmatha R. (1998) *Document Image Clean-Up and Binarization.* In Proceedings of SPIE conference one Document Recognition V, San Jose, California.
70. Vihay V.Raghavan and Gwang S.Jung (1989) *A Critical Investigation of Recall and Precision as Measures of Retrieval System Performance.* ACM Transactions on Information Systems, 7(3): 205-229.
71. Croft Bruce W. (2000) *Advances in Information Retrieval*. Kluwer Academic Publisher, Norwell, USA.
72. James MacLean W. and John K. Tsotsos (2008) *Fast Pattern Recognition using Normalized Grey-Scale Correlation in a Pyramid Image Representation.* Machine Vision & Applications manuscript: Machine Vision and Application, 19(3): 163–179.
73. Walid Magdy, Kareem Darwish and Motaz El-saban (2009) *Efficient Language-Independent Retrieval of Printed Documents without OCR.* Lecture Notes in Computer Science, Vol. 5721, SPIR, pp. 334-343.
74. Wladyslaw Skarbek and Andreas Koschan (1994) *Colour Image Segmentation*: *A Survey*. Technische Universitat, Berlin, Germany.
75. Wondwossen Mulugeta (2004) *Optical Character Recognition for Special Type of Handwriting Amharic Text ("Yekum Tsifet"): Nueral Network Approach.* M.Sc Thesis, Addis Ababa University, Addis Ababa, Ethiopia.
76. Worku Alemu (1997) *Application of Optical Character Recognition to the Amharic Scripts.* M.Sc Thesis, Addis Ababa University, Addis Ababa, Ethiopia.

77. Cufi X., Munoz X., Freixenet J. and Marti J. (2002) *A Review on Image Segmentation Techniques Integrating Region and Boundary Information*. Advances in Imaging and Electron Physics, Vol.120, pp. 1–39.
78. Yaregal Assabie and Josef Bigun (2006) *Ethiopic Character Recognition Using Direction Field Tensor*. ICPR, pp 284-287, Hong Kong.
79. Young I., Gerbrands J. and Van Vliet L. J. (2007) *Fundamentals of Image Processing*. Qunatitative Image Group, Delft University of Technology, Netherland.
80. Yue Lu and Chew Lim Tan (2004) *Information Retrieval in Document Image Databases*. IEEE Transaction on Knowledge and Data Engineering, 16(11): 42-49.
81. Zoran Pe˘cenovi´c, Minh Do, Serge Ayer and Martin Vetterli (1998) *New Methods for Image Retrieval.* ICPS - Congress on Exploring New Tracks in Imaging. Antwerp, Belgium.
82. Bethlehem Mengistu Hilemariam (2002) *N-gram-Based Automatic Indexing for Amharic Text*. M.Sc Thesis, Addis Ababa University, Addis Ababa, Ethiopia.
83. Tewodros Hailemeskel (2003) *Amharic Text Retrieval: An Experiment using Latent Semantic Indexing (LSI) with Singular Value Decomposition (SVD)*. M.Sc Thesis, Addis Ababa University, Addis Ababa, Ethiopia.
84. Milan Sonka and Fitzpatric J.Michael (eds.) (2004) *Handbook of Medical Imaging: Medical Image Processing and Analysis.* Vol 2. SPIE- The International Society for Optical Engineering, Washington, United States of America.

APPENDIX I: The Amharic character and their Roman counter parts

	Ge'ez ä	*Ka'eb* u	*Salis* ï	*Rab'e* a	*Hamis* é	*Sadis* i	*Sab'e* o
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ḥ	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
h	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
a	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ạ	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ṭ	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
p	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
ts	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
ts	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

APPENDIX II: R*oot words and their variants and word writing style variations*

Selected root words	a^TÇ?¶	mNGST	kF t¼	Që¿S	ì UbR
Word variants	ya^TÇ?¶	ymNGST	lkF t¼	kQë¿S	ì UbR³
	ba^TÇ?¶	ymNG C T	kF t¼^N	wdQë¿S'ƒ	b ì UbR
	a^TÇ?¶§#	mNGST M	kF t¼nT	yA¼Që¿S	ì UbR t¼
	ya^TÇ?¶§#	NGST	bkF t¼~'ƒ	Që¿S³	y ì UbR"
	a^TÇ?¶§#Ŧ	b_t mNGST	wdkF t¼nT	Që¿SnT	ì UbR t In T
	a^TÇ?¶M	ANdmNGST	kF [¼	Që¿SNM	ì HbR¿{
	ba^TÇ?¶U	mNGST³	ANdkF t¼¿[>	QdS	ì UuR
	ANda^TÇ?¶M	mNGSTnT	`F t¼	Q;¿S	wd ì UbR
	a^TÇ:¶	mNGbT	kF t¼^	bQë¿SnT	mUbR
	a^T‡?¶	EMRmNGST nT	kkF t¼¿ê{N	ANdQë¿S	y ì Ub™§#

APPENDIX III: Performance of Dynamic Time Warping (DTW)

Query word Image	Lower word profile			Upper word profile			Top shape projection			Bottom shape projection			Vertical projection			Horizontal projection			Vertical distance			Top-Bottom shape pro.		
	Threshold < 2.1			<0.45			<1.9			<2.3			<0.32			<0.012			<2.1			<2.1		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
›=ƒÄåÁ	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	87.5	100	93.33	100	100	100	100	100	100
S”Óeƒ	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Ÿõ}—	100	100	100	100	100	100	87.5	100	93.33	100	100	100	100	100	100	87.5	100	93.33	100	100	100	100	100	100
pÆe	100	100	100	100	100	100	100	88.88	94.11	100	100	100	100	100	100	87.5	100	77.77	100	100	100	100	100	100
Tlu`	100	100	100	100	100	100	100	88.88	94.11	100	100	100	87.5	100	93.33	62.5	100	76.92	62.5	100	76.92	100	100	100
Average	**100**	**100**	**100**	**100**	**100**	**100**	**97.5**	**95.55**	**96.31**	**100**	**100**	**100**	**97.5**	**100**	**98.66**	**85**	**100**	**88.27**	**92.5**	**100**	**95.3**	**100**	**100**	**100**

APPENDIX IV: Feature extraction and matching algorithms Java sub code

```
// Feature extraction algorithms

public void FeatureExtraction(PlanarImage pi, int w, int h, int topCorner, int bottomCorner,
int leftCorner,  int rightCorner, SampleModel sm, int pixelaverage, PrintWriter pwwv, String
Imagename, int zz, File image, File Vector) {
 try {
         FileWriter fwwv = new FileWriter("E:/ado/ImageVector/"+
ector.getName().replaceAll(".jpg",
                                    ".txt"),true);
          pwwv = new PrintWriter(fwwv);
          Vector <Float>  VlinepixValbst = new Vector();
          Vector <Float>  VlinepixValttb = new Vector();
          Vector <Float>  VlinepixValtb = new Vector();
          Vector <Float>  VlinepixValbt = new Vector();
          Vector <Float>  VlinepixValv = new Vector();
          Vector <Float>  VlinepixValh = new Vector();
          Vector <Float>  MinMax= new Vector();
          int pixelValbst = 0;  int pixelValbt = 0;
          int pixelValLV=0;   int pixelValtb = 0;
          int pixelValttb = 0;   int countTotalV=0;
          int pixelValBt = 0;  int countWH = 0;
          int countWV = 0;    double min=0;
          double temp1=0;   double temp2=0;
          double max=0;      double tempn=0;
          double tempm=0;
          int top=0,lower=0,outt=0,outl=0;
          int nbands = sm.getNumBands();
          int[] pixel = new int[nbands];
          RandomIter iterator = RandomIterFactory.create(pi, null);
          for (h = topCorner; h < bottomCorner; h++) {
                for (w = leftCorner; w < rightCorner; w++) {
                 iterator.getPixel(w, h, pixel);
                 pixelaverage = 0;
                 for (int band = 0; band < nbands; band++) {
                    pixelaverage = pixel[band] + pixelaverage;
                 }
                 if (pixelaverage / nbands <= 128) {
                    top=h;
                    outt++;
                    break;
                 }
                 else {
                    ;
                 }
              }
              if(outt>0) {
                 break;
              }
           }
           for (int hh = bottomCorner-1; hh >= topCorner; hh--) {
             for (int ww = leftCorner; ww < rightCorner; ww++) {
```

```
            iterator.getPixel(ww, hh, pixel);
            pixelaverage = 0;
            for (int band = 0; band < nbands; band++) {
               pixelaverage = pixel[band] + pixelaverage;
            }
            if (pixelaverage / nbands <= 128) {
               lower=hh;  outl++;
               break;
            }
            else{
               ;
            }
          }
          if(outl>0) {
             break;
          }
         }

// Horizontal Projection
    for (h = topCorner; h < bottomCorner; h++) {
         for (w = leftCorner; w < rightCorner; w++) {
            iterator.getPixel(w, h, pixel);
            pixelaverage = 0;
            for (int band = 0; band < nbands; band++) {
               pixelaverage = pixel[band] + pixelaverage;
            }
            if (pixelaverage / nbands <= 128) {
               pixelValLV++;
            }
            countWV++;
          }
         countTotalV = countWV + pixelValLV;
         float VlineValV = (float) (pixelValLV)/ countTotalV;
         VlinepixValh.addElement(VlineValV);
         countWV=0;pixelValLV=0;
     }

// Vertical Projection
   for (w = leftCorner; w < rightCorner; w++) {
         for (h = topCorner; h < bottomCorner; h++) {
            iterator.getPixel(w, h, pixel);
            pixelaverage = 0;
            for (int band = 0; band < nbands; band++) {
               pixelaverage = pixel[band] + pixelaverage;
            }
            if (pixelaverage / nbands <= 128) {
               pixelValLV++;
            }
            countWV++;
         }
         countTotalV = countWV + pixelValLV;
         float VlineValV = (float) (pixelValLV)/ countTotalV;
         VlinepixValv.addElement(VlineValV);
         countWV=0;pixelValLV=0;
     }
```

```
 //Bottom Shape Projection
 for (int ww = leftCorner; ww < rightCorner; ww++){
    for (int hh = bottomCorner-1; hh >=topCorner; hh-- ) {
       iterator.getPixel(ww, hh, pixel);
       pixelaverage = 0;
       for (int band = 0; band < nbands; band++) {
          pixelaverage = pixel[band] + pixelaverage;
       }
       if (pixelaverage / nbands <= 128) {
          pixelValbt= hh-top;
          break;
       }
       else {
          ;
       }
    }
    float VlineValVBt = (float) (pixelValbt)/(lower - top);
    VlinepixValbt.addElement(VlineValVBt);
 }
 // Upper Word Profile
 for (w = leftCorner; w < rightCorner; w++) {
    for (h = topCorner; h < bottomCorner; h++) {
       iterator.getPixel(w, h, pixel);
       pixelaverage = 0;
       for (int band = 0; band < nbands; band++) {
          pixelaverage = pixel[band] + pixelaverage;
       }
       if (pixelaverage / nbands <= 128) {
          pixelValtb = h-top;
          break;
       }
       else if(h==bottomCorner-1)
       {
         pixelValtb=lower-top;
         break;
       }
       else{
          ;
       }
    }
    float VlineVaTb = (float) (pixelValtb)/(lower-top);
    VlinepixValtb.addElement(VlineVaTb);
 }
// Lower Word Profile
for (int ww = leftCorner; ww < rightCorner; ww++){
    for (int hh = bottomCorner-1; hh >=topCorner; hh-- ) {
       iterator.getPixel(ww, hh, pixel);
       pixelaverage = 0;
       for (int band = 0; band < nbands; band++) {
          pixelaverage = pixel[band] + pixelaverage;
       }
       if (pixelaverage / nbands <= 128) {
          pixelValbst= lower-hh;
          break;
```

```
        }
        else if(hh==topCorner){
          pixelValbst= lower-top;
          break;
        }
        else{
           ;
        }
     }
     float VlineValVBtt = (float) (pixelValbst)/(lower-top);
     VlinepixValbst.addElement(VlineValVBtt);
     pixelValbst = 0;
  }

  // Top shape Projection
for (w = leftCorner; w < rightCorner; w++) {
     for (h = topCorner; h < bottomCorner; h++) {
        iterator.getPixel(w, h, pixel);
        pixelaverage = 0;
        for (int band = 0; band < nbands; band++) {
           pixelaverage = pixel[band] + pixelaverage;
        }
        if (pixelaverage / nbands <= 128) {
           pixelValttb=lower-h;
           break;
        }
        else{
           ;
        }
     }
     float VlineVaTb = (float) (pixelValttb)/(lower - top);
     VlinepixValttb.addElement(VlineVaTb);
  }
  //Top-Bottom Shape projection
for (int k = 0; k < VlinepixValttb.size(); k++) {
    double val1 = (VlinepixValttb.elementAt(k));
    double val2 = (VlinepixValbt.elementAt(k));
     float countTotal= (float)(val1 + val2);
   MinMax.addElement(countTotal);
  }
for (int k = 0; k <  MinMax.size(); k++){
    min=0;
    for(int kk=k+1;kk<MinMax.size(); kk++ ){
      min=Math.min(MinMax.elementAt(k),MinMax.elementAt(kk));
      if(k==0) {
       tempn=min;
      }
      else{
         if(min<=tempn){
           tempn= min;
         }
         else {
            ;
         }
      }
```

```
            }
          }
         for (int k = 0; k <  MinMax.size(); k++){
             max=0;
             for(int kk= k+1;kk<MinMax.size(); kk++ ){
               max=Math.max(MinMax.elementAt(k),MinMax.elementAt(kk));
               if(k==0) {
                tempm=max;
               }
               else {
                 if( max>=tempm){
                    tempm=max;
                 }
                 else{
                    ;
                 }
               }
             }
          }
          for (int k = 0; k < VlinepixValttb.size(); k++) {
             double val1 = (MinMax.elementAt(k));
             double val2 = (VlinepixValttb.elementAt(k));
             float countTotal1= (float)((val1-tempn)/(tempm-tempn));
             float countTotal2= (float)(val2);
             float countTotal = (float)(countTotal1 + countTotal2 )/2;
             pwwv.print(countTotal + ",");
          }
           pwwv.flush();
           pwwv.close();
        } catch (Exception e) {
           System.out.println(e.getMessage());
        }
     }
}

// matching algorithms
// Dynamic time warping
public void DynamicTimeWarping(int wdatastart1Q, int vdatastart1Q, int wdataend1Q,
          int vdataend1Q, int wborderstart1Q, int vborderstart1Q, int wborderend1Q, int
vborderend1Q,
           PrintWriter pwwbQ) {
       try {
            //System.out.println("You are in CosineSimilarity");
            VectorValue = new File("E:/ado/GeneralIndexFile/GeneralIndexFileWeighted.txt");
            if (VectorValue.isFile()) {
               BufferedReader inputvQ = new BufferedReader(new
FileReader("Qwordvalue.txt"));
               BufferedReader inputvI = new BufferedReader(new FileReader(VectorValue));
               String linevQ = null;
               String linevI = null;
               int flagofretrieved = 0;
               double similarity = 0; double querval=0;
               double quervall=0;  double sumofQ=0;
               double sumofQq=0; double sumofD=0;
               double sumofDd=0; double dataval=0;
```

```
double datavall=0;   double diff=0;
double difff=0;         double dotproduct = 0;
double sqrdata = 0;   double sqrdatacompare = 0;
double suffixsimilarity = 0; double datatempsearch;
double datacomparetempsearch;
double dotproductsearch = 0;
double sqrdatasearch = 0;
double sqrdatacomparesearch = 0;
double preffixsimilarity = 0;
double NumberDocument = 0;
int featurelengthsearch;
int flag = 0;
String[] dataQ = null;
String[] dataI = null;
Vector<String> lineQ5 = new Vector<String>(2, 2);
Vector<String> line2 = new Vector<String>(2, 2);
Vector<String> retrieveddocument = new Vector<String>(2, 2);
Vector <Double>  similaritycomp = new Vector();
File InnerImage = new File("E:/ado/Final/");
File Innerdocument[] = InnerImage.listFiles();
for(int in = 0; in < Innerdocument.length; in++)
{
   File Innerpage[] = Innerdocument[in].listFiles();
   NumberDocument = Innerpage.length + NumberDocument;
}
while ((linevQ = inputvQ.readLine()) != null)  {
   lineQ5.addElement(linevQ);
}

for (int k = 0; k < (lineQ5.size()); ++k) {
   dataQ = lineQ5.elementAt(k).split(",");
}
double [] adataQ = new double[dataQ.length];
for(int a=0;a<adataQ.length;a++) {
   adataQ[a]= Double.parseDouble(dataQ[a]);

}
  while ((linevI = inputvI.readLine()) != null) {
   line2.addElement(linevI);
}
//each word of a document image
int addressflag = 0;
for (int m = 0; m < (line2.size()); m++) {
   if(line2.elementAt(m).equals(null)) {
      break;
   }
   addressflag = 0;
   dataI = line2.elementAt(m).split(",");
   //each word for query
   for (int u=0; u<dataI.length; u++) {
      if(Double.parseDouble(dataI[u])==1.5) {
         addressflag = u;
         break;
      }
   }
```

```
//DTW for each feature vector of the word matching
 double[][] d = new double[addressflag][dataQ.length];
 double[][] D = new double[addressflag][dataQ.length];
 Vector <Double>  B = new Vector();
 for(int ii=0;ii<addressflag; ii++)  {
    for(int jj=0;jj<dataQ.length;jj++)  {
            d[ii][jj]= (Math.pow(Double.parseDouble(dataI[ii])-
            Double.parseDouble(dataQ[jj]),2));
       }
 }
 D[0][0]=0;
 for(int a=1;a<addressflag;a++) {
    D[a][0] = D[a-1][0] + d[a][0];
 }
 for(int b=1;b<dataQ.length;b++){
     D[0][b] = D[0][b-1] + d[0][b];
  }
for(int i=1;i<addressflag;i++) {
    for(int j=1;j<dataQ.length;j++) {
       double ver = D[i-1][j] + d[i][j];
       double dia = D[i-1][j-1] + d[i][j];
       double hri = D[i][j-1] + d[i][j];
       double temp= Math.min(dia, hri);
       double mint= Math.min(temp, ver);
       D[i][j]= mint;
    }
 }
  // Back Tracking
 double count=0;
 for(int i = addressflag-1, j = dataQ.length-1;i>0 &&j>0; ) {
      double vr= D[i-1][j];
      double hr= D[i][j-1];
      double dg= D[i-1][j-1];
      double tm= Math.min(vr,hr);
      double fl= Math.min(dg,tm);
      if(vr==hr && vr==dg){
         B.addElement(dg);
         i--;  j--;
         count++;
        }
      else if( vr==fl) {
           B.addElement(fl);
           i--;
         }
      else if(dg==fl){

              B.addElement(fl);
              i--;  j--;
              count++;
        }
       else if(hr==fl) {
          B.addElement(fl);
          j--;

          }
```

```
                    else {
                      ;
                    }
                }
                similarity = D[addressflag-1][dataQ.length-1]/count;
                 System.out.println(similarity);
                 if (similarity <0.03){
                        retrieveddocument.addElement(dataI[addressflag+2]);
                   }
                querval = 0;  sumofQ =  0; dataval = 0; diff = 0; sumofD = 0;
                quervall = 0; sumofQq =  0; datavall = 0; difff = 0;  sumofDd = 0;
                similarity = 0;  preffixsimilarity=0;  suffixsimilarity=0;
              }
              for(int p = 0; p < retrieveddocument.size();p++) {

                System.out.println(retrieveddocument.elementAt(p));
              }
              System.out.println("Total number of pages retrieved =
"+retrieveddocument.size());
          }
        }
        catch (Exception e) {
          System.out.println(e.getMessage());
        }
      }
}

    // Normalized cross correlation
public void NormalixedCrossCorrelation(int wdatastart1Q, int vdatastart1Q, int wdataend1Q,
        int vdataend1Q, int wborderstart1Q, int vborderstart1Q, int wborderend1Q, int
        vborderend1Q, PrintWriter pwwbQ){
       try {
            //System.out.println("You are in CosineSimilarity");
            VectorValue = new File("E:/ado/GeneralIndexFile/GeneralIndexFileWeighted.txt");
            if (VectorValue.isFile())  {
              BufferedReader inputvQ = new BufferedReader(new
FileReader("Qwordvalue.txt"));
              BufferedReader inputvI = new BufferedReader(new FileReader(VectorValue));
              String linevQ = null;
              String linevI = null;
              int eee = 0;
              int flagofretrieved = 0;
              double similarity = 0;
              double norsimilarity=0;
              double datatemp;
              double datacomparetemp;
              double dotproduct = 0;
              double sqrdata = 0;
              double sqrdatacompare = 0;
              double suffixsimilarity = 0;
              double datatempsearch;
              double datacomparetempsearch;
              double dotproductsearch = 0;
              double sqrdatasearch = 0;
              double sqrdatacomparesearch = 0;
```

```
double preffixsimilarity = 0;
double NumberDocument = 0;
double meanvQ=0;   double meanQ=0;
double stdevvQ=0;   double stdvQ=0;
double stdevQ=0;    double lofadataQ=0;
double meanvI=0;   double meanI=0;
double stdevvI=0;   double stdvI=0;
double stdevI=0;      double lofadataI=0;
double min=0;         double temp=0;
double max=0;         double temp1=0;
int count=0;
int flag = 0;
String[] dataQ = null;
String[] dataI = null;
Vector<String> lineQ5 = new Vector<String>(2, 2);
Vector<String> line2 = new Vector<String>(2, 2);
Vector<String> retrieveddocument = new Vector<String>(2, 2);
Vector <Double>  similaritycomp = new Vector();
File InnerImage = new File("E:/ado/Final/");
File Innerdocument[] = InnerImage.listFiles();
for(int in = 0; in < Innerdocument.length; in++)  {
   File Innerpage[] = Innerdocument[in].listFiles();
   NumberDocument = Innerpage.length + NumberDocument;
}
while ((linevQ = inputvQ.readLine()) != null) {
   lineQ5.addElement(linevQ); }

for (int k = 0; k < (lineQ5.size()); ++k) {
   dataQ = lineQ5.elementAt(k).split(",");
}
// mean and standard devation of the query

double [] adataQ = new double[dataQ.length];
for(int a=0;a<adataQ.length-1;a++) {
   adataQ[a]= Double.parseDouble(dataQ[a]);

}
for(int mn=0; mn<adataQ.length-1;mn++){
   meanvQ = adataQ[mn] + meanvQ;

}
meanQ= meanvQ/adataQ.length;
for(int sn=0; sn<adataQ.length-1;sn++) {
   stdevvQ = (adataQ[sn] - meanQ);
   stdvQ=(stdevvQ*stdevvQ) + stdvQ;

}
stdevQ = (Math.sqrt((stdvQ)/(adataQ.length-1)));
while ((linevI = inputvI.readLine()) != null) {
   line2.addElement(linevI);
}
//each word for document
int addressflag = 0;
System.out.println(line2.size());
for (int i = 0;i<line2.size(); i++) {
```

```
if(line2.elementAt(i).equals(null)){
   continue;
}
addressflag = 0;
dataI = line2.elementAt(i).split(",");
//each word for query
for (int u=0; u<dataI.length; u++){
   if(Double.parseDouble(dataI[u])==1.5) {
      addressflag = u;
      //System.out.println("Check val of addressfla");
      break;
     }
}
double [] adataI = new double[addressflag];
for (int im=0; im < adataI.length; im++) {
   adataI[im]=Double.parseDouble(dataI[im]);
  }
for(int mn=0; mn<adataI.length;mn++) {
      meanvI = adataI[mn] + meanvI;
  }
meanI= meanvI/adataI.length;
 for(int sn=0; sn<adataI.length;sn++)
 {
     stdevvI = (adataI[sn] - meanI);
     stdvI=(stdevvI*stdevvI) + stdvI;
 }
stdevI = (Math.sqrt((stdvI)/(adataI.length-1)));
//for each feature vector of the word matching
if(dataQ.length > addressflag)  {
   for(int im =0; im<addressflag;im++)  {
       for(int h=count; h < dataQ.length; h++)   {
          datatemp = Double.parseDouble(dataQ[h]);
          datacomparetemp = Double.parseDouble(dataI[im]);
          if(Math.abs(datacomparetemp - datatemp) < 0.12) {
           dotproduct = (datatemp - meanQ);
           sqrdata = (datacomparetemp - meanI);
           sqrdatacompare = (dotproduct * sqrdata) + sqrdatacompare;
           count=h;
           break;
          }
         else {
          count=h+1;
         }
      }
    }
   similarity = sqrdatacompare/ (stdevI * stdevQ);
}

if(dataQ.length <= addressflag) {
   for(int iq=0;iq<dataQ.length;iq++) {
    for(int hh=count; hh < addressflag; hh++) {
       datatemp = Double.parseDouble(dataQ[iq]);
       datacomparetemp = Double.parseDouble(dataI[hh]);
      if(Math.abs(datacomparetemp - datatemp) <  0.12)
       {
```

```
                        dotproduct = (datatemp - meanQ);
                        sqrdata = (datacomparetemp - meanI);
                        sqrdatacompare = (dotproduct * sqrdata) + sqrdatacompare;
                        count=hh;
                        break;
                       }
                        else{
                         count=hh+1;
                        }
                      }
                   }
                  similarity = sqrdatacompare/(stdevI * stdevQ);
                 }
               dotproduct = 0; sqrdata = 0; sqrdatacompare = 0;  sqrdatasearch= 0;
               sqrdatacomparesearch = 0; dotproductsearch = 0; suffixsimilarity = 0;
               preffixsimilarity = 0; similarity =0;  count=0;  meanvI=0; meanI=0;
               stdevvI=0; stdvI=0;stdevI=0; lofadataI=0;
            }
            for(int j=0;j<similaritycomp.size(); j++) {
                 min=0;
               for(int jj=j+1;jj<similaritycomp.size(); jj++) {
                    min=
Math.min(similaritycomp.elementAt(j),similaritycomp.elementAt(jj));
                    if(j==0) {
                       temp=min;
                    }
                    else  {
                       if(min<=temp) {
                          temp= min;
                       }
                       else{
                       ;
                       }
                    }
               }
             }
            System.out.println("vlaue of min :" + temp);
            for(int j=0;j<similaritycomp.size(); j++){
                 max=0;
               for(int jj=j+1;jj<similaritycomp.size(); jj++){
                    max=
Math.max(similaritycomp.elementAt(j),similaritycomp.elementAt(jj));
                    if(j==0) {
                       temp1=max;
                    }
                    else {
                       if(max >temp1)  {
                          temp1=max;

                       }
                       else
                       {
                       ;
                       }
                    }
```

```
      }
    }
for (int i = 0;i<line2.size(); i++)  {

  if(line2.elementAt(i).equals(null)){
     continue;
  }
  addressflag = 0;
  dataI = line2.elementAt(i).split(",");
  //each word for query
  for (int u=0; u<dataI.length; u++){

     if(Double.parseDouble(dataI[u])==1.5){
        addressflag = u;
        //System.out.println("Check val of addressfla");
        break;
     }
  }
  double [] adataI = new double[addressflag];
  for (int im=0; im < adataI.length; im++){
     adataI[im]=Double.parseDouble(dataI[im]);
  }
  for(int mn=0; mn<adataI.length;mn++) {
       meanvI = adataI[mn] + meanvI;
    }
  meanI= meanvI/adataI.length;
  for(int sn=0; sn<adataI.length;sn++){
       stdevvI = (adataI[sn] - meanI);
       stdvI=(stdevvI*stdevvI) + stdvI;
   }
  stdevI = (Math.sqrt((stdvI)/(adataI.length-1)));
  //for each feature vector of the word matching
  if(dataQ.length > addressflag) {
     for(int im =0; im<addressflag;im++) {
        for(int h=count; h < dataQ.length; h++) {
           datatemp = Double.parseDouble(dataQ[h]);
           datacomparetemp = Double.parseDouble(dataI[im]);
           if(Math.abs(datacomparetemp - datatemp) <0.05) {
             dotproduct = (datatemp - meanQ);
             sqrdata = (datacomparetemp - meanI);
             sqrdatacompare = (dotproduct * sqrdata) + sqrdatacompare;
             count=h;
             break;
            }
          else {
            count= h+1;
           }
         }

      }
     similarity = sqrdatacompare/ (stdevI * stdevQ);
  }

  if(dataQ.length <= addressflag)  {
     for(int iq=0;iq<dataQ.length;iq++) {
```

```
                    for(int hh=count; hh < addressflag; hh++){
                      datatemp = Double.parseDouble(dataQ[iq]);
                      datacomparetemp = Double.parseDouble(dataI[hh]);
                      if(Math.abs(datacomparetemp - datatemp) < 0.12) {
                         dotproduct = (datatemp - meanQ);
                         sqrdata = (datacomparetemp - meanI);
                         sqrdatacompare = (dotproduct * sqrdata) + sqrdatacompare;
                         count=hh;
                         break;
                       }
                        else{
                          count= hh+1;
                        }
                      }
                    }
                     similarity = sqrdatacompare/(stdevI * stdevQ);

                    }
                   norsimilarity= (similarity - temp)/(temp1-temp);
                   System.out.println(norsimilarity);
                  if (norsimilarity >=0.5){
                    retrieveddocument.addElement(dataI[addressflag+2]);
                   }
                  dotproduct = 0;sqrdata = 0; sqrdatacompare = 0;  sqrdatasearch= 0;
                  sqrdatacomparesearch = 0; dotproductsearch = 0;  suffixsimilarity = 0;
                  preffixsimilarity = 0; similarity =0; norsimilarity=0; count=0;
                  meanvI=0; meanI=0; stdevvI=0; stdvI=0; stdevI=0; lofadataI=0;
                }
               for(int p = 0; p < retrieveddocument.size();p++) {

                  System.out.println(retrieveddocument.elementAt(p));
                }
                double Men=0,Eth=0,Kds=0,Kft=0,Mah=0;
                System.out.println("Total number of pages retrieved =
"+retrieveddocument.size());}
        }
        catch (Exception e)
        {
          System.out.println(e.getMessage());
        }
      }
}
```

Declaration

I declare that the thesis is my original work and has not been presented for a degree in any other university.

_____

Date: 11/7/2011

This thesis has been submitted for examination with my approval as university advisor.

_____

Million Meshesha (Ph.D)