ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

DEPARTMENT OF INFORMATION SCIENCE

AMHARIC DOCUMENT IMAGE RETRIEVAL WITHOUT EXPLICIT RECOGNITION

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

MESFIN WORKU ASFAW

JUNE, 2009

AMHARIC DOCUMENT IMAGE RETRIEVAL WITHOUT EXPLICIT RECOGNITION

BY

MESFIN WORKU ASFAW

Advisor Dr. Million Meshesha

Co-Advisor Dr. Diana Inkpen

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

JUNE, 2009

DEDICATION

TO My Children

Duke and Nana

From The Bottom of My Heart

ACKNOWLEDGMENT

My greatest gratitude is extended to Dr. Million Meshesha and Dr. Diana Inkpen for their dedication on advising me this research, without them I could not reach to the final stage. My special thanks go to Dr. Million Meshesha, who supervised my research. His guidance and supervision helps the research to become successful.

I would like also to thank the Oromia Information Communication and Technology Development Agency (OICTDA) for supporting me for two years by providing me the time and learning resources, without which I could not have even started my masters' education in the Information Science Department, of the Informatics Faculty of the Addis Ababa University.

I also like to thank my wife, W/ro Fikirte Lulseged, which happens to be a language expert, for helping me to understand the Amharic writing system and for doing a manual work in query selection and checking the query word with the documents.

I am deeply indebted to my sister W/rt Azaletch Worku who sacrifices her life to make my life successful. She supports me in everything I need. I owe her a lot and I do not think I am capable of returning her favors.

I sincerely thank my brother Dereje Worku for providing me his laptop for two years. Because of him I comfortably work on my research day and night without bothering about the availability of computer.

I would also like to thank W/ro Ayenalem Tesfaye , W/rt Genet Mezimir, Ato Sisay Adugan and Ato Tsegaw kelela for the support they made during my research work.

Table of Contents

ACKNOWLEDGMENT ..... Page
Table of Contents i
List of Tables ii
List of Figures v
List of Equations vi
List of Algorithms vii
List of Acronyms viii
ABSTRACT ix
CHAPTER ONE x
1. INTRODUCTION 1
1.1. Background of the Study 1
1.2. Statement of the Problem 1
1.3. Objectives of the Study 3
1.3.1. General Objective 5
1.3.2. Specific Objective 5
1.4. Scope and limitation of the Study 5
1.5. Methodology of the Study 6
1.5.1. Nature of the Data 7
1.5.2. Data Collection and Processing 7
1.5.3. Experimentation and Measurements 7
1.6. Application of Results 8
1.7. Organization of the Thesis 8
CHAPTER TWO 9
2. LITERATURE REVIEW 10
2.1. Writing Systems 10
2.1.1. Amharic Writing System 10
2.1.2. Amharic Word formations 11
2.1.3. Problems with the Amharic Writing system 14
2.2. Retrieval Systems 16
2.2.1. Text-Based Document Retrieval 17
2.2.2. Image-Based Document Retrieval 18
2.2.2.1. Recognition-Based Document Image Retrieval (OCR). 20
2.2.2.2. Document image retrieval without explicit recognition 21

2.3. Document Image Segmentation 24
2.3.1. Pixel based segmentation 25
2.3.2. Area-based segmentation 28
2.3.3. Edge-based segmentation 29
2.3.4. Physics-based segmentation 29
2.4. Representation of document image s 30
2.4.1. Word representation 31
2.5. Matching Words in Document Images 32
2.5.1. Similarity measure 33
2.6. Retrieval and Ranking 36
2.6.1. Performance Measures 37
2.7. Amharic Retrieval Systems 40
2.7.1.1. Related works in Document image retrieval 41
CHAPTER THREE: 45
3. AMHARIC DOCUMENT IMAGE RETRIEVAL TECHNIQUES 45
3.1. System Design 46
3.2. Preprocessing 47
3.2.1. Binarization 47
3.3. Segmentation 48
3.3.1. Line Segmentation 48
3.3.2. Word Segmentation 49
3.3.3. Query Segmentation 49
3.4. Word Feature Extraction 50
3.5. Feature Matching 52
3.6. Similarity Measure 54
3.7. Performance Evaluation 54
CHAPTER FOUR 57
4. EXPERIMENTATION 57
4.1. Preprocessing 58
4.2. Segmentation 59
4.2.1. Line Segmentation 59
4.2.2. Word Segmentation 62
4.2.3. Query Segmentation 66

List of Figures

Page

Figure 2 1 Origins of Ethiopic writing system, a family tree model (adapted from Coulmas,1989) 12

Figure 2.2 Logical views of a document adapted from (Baeza-Yates & Neto, 1999) 19

Figure 2.3 The process of retrieving information adapted from (Baeza-Yates & Neto, 1999) 20

Figure 2.4 Block Diagram of OCR adapted from Howard Wing Ho Leung. 22

Figure 2 5 Conceptual Diagram of the Searching Procedure Adapted from (Million and Jawahar, 2008) 43

Figure 3.1 Architecture of the proposed system 46

Figure 4.1 Document Image retrieval Process. 57

Figure 4.2 Horizontal segmented document image 61

Figure 4.3 Figure showing word level segmentation 64

Figure 4.4 Divided regions of a word image 69

Figure 4.5 Area based divided regions of word image 71

List of Equations

Equation 2 1: Euclidean distance measure 33

Equation 2.2 Cosine similarity measure equation 34

Equation 2.3 Minkwski-Form Distance 34

Equation 2.4 Quadratic Form Distance 35

Equation 2.5 Mahalanobis Distance 35

Equation 2.6 Simplified Mahalanobis Distance 35

Equation 2.7 Kullback-Leibler divergence measures 36

Equation 2.8 Jeffrey-divergence measures 36

Equation 2.9 Performance Measure (Recall) 38

Equation 2.10 Performance Measure (Precision) 39

Equation 2.11 Mean average precision 39

Equation 2.12 Dynamic time warping equation 44

Equation 3.1 : Modified Euclidean Distance Measure 53

Equation 3.2 : Performance Evaluation (F-Measure) 55

List of Acronyms

(DIP)	Document Image Processing
ASCII	American Standard for Information Exchange
CBIR	Content Based Image Retrieval
DIR	Document Image Retrieval
Dpi	Dot per inch
EICTDA	Ethiopian Information Communication and Technology Development Agency
FEX	Feature extractor
ICT	Information Communication Technology
IDF	Inverse Document Frequency
IR	Information Retrieval
JD	Jeffrey-Divergence
KL	Kullback-Leibler divergence measures
LIS	Latent Semantic Indexing
MAP	Mean Average Precision
MICR	Magnetic Ink Character Recognition
OCR	Optical Character Recognition
OICTDA	Oromia Information Communication and Technology Development Agency
OMR	Optical Mark Recognition
SNNP	Southern Nations, Nationalities and People
TF	Term Frequency
DTW	Dynamic time warping

ABSTRACT

Retrieval of the stored information is a key issue. Especially image retrieval needs an emphasis, because the nature of the data is complex and difficult to retrieve. There are many problems to be studied in the area of image retrieval. From these, Document Image Retrieval is one of the issues that have to be given attention.

Document retrieval can use either a textual-based retrieval system or an image-based retrieval system. Document image retrieval system can also be done in two ways: recognition-based document image retrieval or document image retrieval without explicit recognition.

Currently, little has been done on the Amharic document retrieval systems. The Amharic text retrieval systems which are covered by the researchers considered limited Amharic documents that are available only in hardcopy format.

The proposed system incorporates document images and user queries. The document image is preprocessed, segmented at word level and the feature of each word is extracted. Then the textual query is rendered to convert into an image query, preprocessed, segmented and the feature is extracted. The technique used for feature extraction considers the word shape analysis. The extracted feature of the image query is matched with the feature of the document images, at word level using Euclidean and cosine similarity measures. Finally relevant document images are retrieved in ranked order in response to the given query.

To verify the validity of the approach proposed, experiment is carried out on 121 scanned Amharic documents that are selected from printed legal documents and news items.

The data retrieval effectiveness is measured using retrieval measures such as precision, recall and F-Score.

The experimental results confirmed the validity of the model for retrieving relevant document images from the collection of scanned document images.

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the Study

Information is a basic ingredient of economic, social and political development for any society; therefore, countries like Ethiopia have to exert the maximum effort to utilize the available information. Information Communication Technology (ICT) allows the acquisition, transmission, storage and manipulation of information. The source of the information can be text documents, images, recorded sounds and videos, and other sources.

Retrieval of the stored information is a key issue. Especially multimedia data retrieval, including image needs an emphasis, because the nature of the data is complex and difficult to retrieve.

According to (Long, zhang, & Feng, 2003) early work on image retrieval can be traced back to the late 1970s. Early techniques were not generally based on visual features but on the textual annotation of images. That means, images were first annotated with text and then searched using a text-based retrieval approach from traditional database management systems.

First-generation Content-Based Image Retrieval (CBIR) systems were based on manual textual annotation to represent image content. However, this technique can only be applied to small data volumes and is limited to very narrow visual domains (Saeed, Pejas, & Long, 2006).

Content-based image retrieval, a technique which uses visual contents to search images from large scale image databases according to users' interests, has been an active and fast advancing research area since the 1990s. During the past decade, remarkable progress has been made in both theoretical research and system development in CBIR (Long, zhang, & Feng, 2003). However, there remain many challenging research problems that continue to attract researchers from multiple disciplines such as medical, aerial photos and satellite imagery analysis, etc.

Content-based image retrieval uses the visual contents of an image such as color, tonnage, shape, texture, spatial layout and other important features to represent and index the image (Long, zhang, & Feng, 2003).

Multimedia document retrieval algorithms can be divided into three interrelated algorithms (Haque, Chowdhury, & Rahman, 2005): image featuring, feature matching and the combination of multiple evidences. Features are extracted, both from queries and multimedia documents, before matching the features of the query with the features of the documents.

Image featuring is the automated process of locating and encoding distinctive characteristics of the image. This is the same as image feature extraction. Feature matching is a mechanism of comparing two images by using the extracted features of images. The combination of multiple evidences means using multiple features of an image that helps the feature matching process to identify similar images easily.

In recent years, there has been much interest in the research area of Document Image Retrieval (DIR) (Tan, Lu, & Lim, 2004).

CHAPTER TWO

2. LITERATURE REVIEW

2.1. Writing Systems

According to the definition of Ayele (1997) Writing Systems are components of knowledge systems which assist in synthesizing ideas, thoughts, and deeds through the use of signs, symbols or other pictorial renderings. Specifically, writing is a means by which people record, objectify, and organize their activities and thoughts through images and graphs. Writing is a means to inscribe meanings that are expressed through sounds. This means that writing facilitates the proper recording and transmissions of events and deeds from one generation to other

It is not possible to determine which language family a language belongs to by looking at the writing system. Writing systems can be deployed for political or religious reasons as well as linguistic ones.

For example, Hindi and Urdu are very similar languages and belong to the same language family (Indo-European). Linguistically they are dialects of the same language. Hindi uses the Devanagari writing system derived from the extinct language Sanskrit; Urdu writing uses the Nastaliq script derived from Arabic. Similarly, Croatian and Serbian use the Latin and Cyrillic alphabets, respectively, even though the two languages are very closely related. Conversely, many unrelated languages may use the same alphabet (krysstal.com, 2002).

2.1.1. Amharic Writing System

According to some literatures, the development of the Ethiopic writing system is dated back far beyond the birth of Christ, but no definite time has been mentioned. Some literatures indicate the probable time and simply stated that it was developed in the same time with that of other Semitic writing systems.

The Ge'ez or Ethiopic script possibly developed from the Sabaean/Minean script. The earliest known inscriptions in the Ge'ez script date to the 5th century BC. (Ager, 2008)

The Ethiopic writing system has its origins in the same ancestral writing systems as those of European alphabets, namely the Semitic scripts that proliferated in the Middle East more than three thousand years ago (Coulmas, 1989). Little is known about the precise timing and location of the emergence of the earliest Semitic phonetic writing system, though speculations abound (Bloor, 1995).

All that seems reasonably certain is that a consonantal script developed among Semitic people on the Eastern shore of the Mediterranean sometime between 1800-1300 BC (Gaur, 1987:88).

Amharic script which is a successor of Ge'ez and dates back to 300 AD (Encarta, 2007) is used for writing in Ethiopia and Eritrea, for languages like Amharic, Tigre and Tigrigna.

A family tree model of the writing systems (Coulmas, 1989) shows two main branches descending from Proto-West Semitic: North Semitic and South Semitic. Among the descendants in the North Semitic branch are Hebrew, Arabic and Greek (and hence Roman and Cyrillic). The South Semitic side is usually held to have produced Ethiopic

via the Sabean system, which is speculatively dated as emerging in the 11th and 10th centuries BC. Figure 2.1 depicts a family tree of Ethiopian script (columas, 1989).

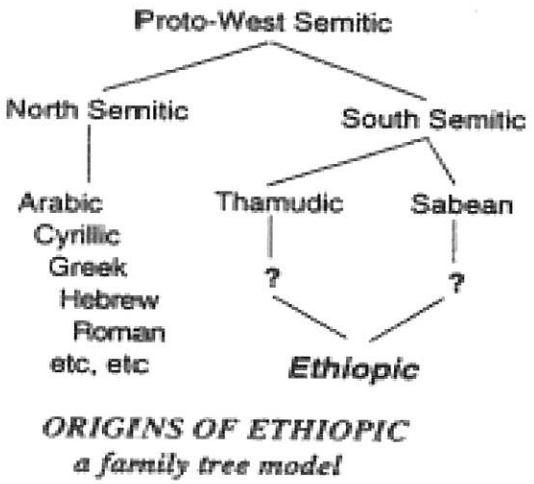


Figure 2 1 Origins of Ethiopic writing system, a family tree model (adapted from Coulmas,1989)

(Bernal, 1990) rejecting the family tree model, dates the origins of Ethiopic script earlier, relating it to Thamudic, an older script.

Amharic is the official language of Ethiopia. It is a Semitic language family that has the largest number of speakers next to Arabic (Hayward, 1999). As per the 1998 statistical census, Amharic is spoken by 17.4 million people as a mother tongue and 5.1 million people as a second language (Raymond G. Gordon, 2005).

There is a growing body of literatures in Amharic in many forms. These literatures include government proclamations and records, educational books, religious material, novels, poetry, proverb collections, technical manuals, medical topics, newspapers, magazines and many others.

The Amharic writing system is considered as a syllabic system rather than alphabetic. In the Amharic syllabic writing system, each character stands for a syllable rather than a single sound (Encyclopedia, 2007).

Haile (1967) stated that the Amharic writing system is a syllabic one in that it allows anyone to write Amharic texts if s/he can speak Amharic and has knowledge of the Amharic alphabet.

Like many Semitic languages, Amharic uses tri-consonantal roots in its verb morphology. The result of this is that a fluent speaker of Amharic can often decipher written text by observing the consonants, with the vowel variants being supplemental detail. (Wikipedia, Amharic, 2009)

The other important feature of Amharic writing system is that it is written from left to right and there is no distinction between upper and lower case letters

The Amharic orthography, as it is represented in the Amharic character set, called ፊደል (Fidel) consists of 312 distinct symbols. These symbols are classified into four groups. In the first category, there are 34 core characters, each of which occurs in seven orders (one basic and six non basic forms) totaling 238, which represent syllable combinations consisting of a consonant and a following vowel. The second category (4*5=20 characters) consists of four labialization symbols, which have five orders. The third categories (5*5=25) are another five labialization symbols that have five orders, the fourth category are numbers which are represented by 20 different symbols and there are 9 punctuation marks. Table 2.1 shows the number of characters in each group (refer the appendices).

No	Type of Amharic Character	Number of Characters
1	Core Characters	238
2	Labialized characters	45
3	Numbers	20
4	Punctuation Marks	9
5	Total	312

Table 2.1 Total Number of Characters in Amharic Alphabet ፊደል (Fiedel)

A set of 38 phones, seven vowels and thirty-one consonants, makes up the complete inventory of sounds for the Amharic language (Yimam, 1986). Amharic consonants are generally classified as stops, fricatives, nasals, liquids, and semi-vowels (LESLAU, 2000). The Amharic vowels are (አ ኡ ኢ ኣ ኤ እ ኦ and ዐ ዑ ዒ ዓ ዕ ዖ).

There are Amharic characters with the same sound. Amharic characters (ኀ ሀ ሐ ኸ) all representing the /h/ sound. (አ ዐ) that represent the /a/ sound. (ሰ ሠ) represent the /s/ sound (ጸ ፀ) represent the /sss'/ sound. These characters are considered as redundant by some scholars who proposed to remove them from Amharic alphabet (Bloor, 1995). To the contrary there are people who object the removal of these characters and they argue that words like ንጉሥ should not be written as ንጉስ or እሳት cannot be written as እሣት.

2.1.2. Amharic Word formations

Written Amharic words are easy to understand for the people who speak the language. There are some words that can be interpreted wrongly just by looking at the words without the context. The meaning of these morphologically similar words depends on

the context. For example the word "አል" can be read as "alä" 'he said' or "allä" 'there is'. We can also take the word "ይመታአል" which can be read yəmätall 'he hits', yəmmättall 'he is hit'. This is called Gemination which has multiple meaning, according to Webster Dictionary (1913).

Haddis Alemayehu, the known Ethiopian novelist, tries to resolve this problem in his novel "Fikir eske Mekaber" ፍቅር እስከ መቃብር by placing a dot above the characters, whose consonants are geminated.

According to (Wikipedia, Amharic, 2009), the free online encyclopedia, in Amharic there is a clear distinction of person, number and gender that plays a role within the grammar of the language. Considering personal pronouns I, she, he, and they in English can be represented as (እኔ əne); (እስዋ əsswa), (እሱ əsu) and (እነሱ ənesu).

Amharic distinguishes eight combinations of person, number, and gender. For first person, there is a two-way distinction between singular ('I') and plural ('we'), whereas for second and third persons, there is a distinction between singular and plural and within the singular a further distinction between masculine and feminine ('you masculine. Singular'.(አንተ), 'you feminine singular' (አንቺ),you plural, 'he', 'she', 'they').

All Amharic verbs agree with their subjects; that is, the person, number, and (2nd and 3rd person singular) gender of the subject of the verb are marked by suffixes or prefixes on the verb (Wikipedia, Amharic, 2009). For example 'መጣህ', for second person singular and 'መጣች' for third person singular.

Amharic has a fairly rich morphology marking. The subject is marked on the verb using subject suffix pronouns, as in ሰበርኩ I broke'; the direct object is optionally marked on

the verb, as in ሰበረኝ 'he broke me'; some prepositional phrase complements are optionally marked on the verb as in እስኪሰብረኝ until he broke me'; functional elements like negation marks, conjunctions and some auxiliary verbs are also bound morphemes and are attached to the verb. For example, in አልሰብርም 'I will not be broken', the negation is marked by "አል" (Solomon, Menzel, & Tafila, 2005).

Usually word boundaries use either a space like in Roman writing system or they are indicated by two vertically-placed dots like a colon,": " A sentence boundary is indicated by four dots "::" the symbol ፣ is used as a comma. The colon and semicolon are represented as ፡ and ፤, respectively. The question mark symbol, three vertically-placed dots, is no more in use and replaced by the Roman Script question mark "?" . The modern Amharic writing system adds to its punctuations marks the following three foreign systems. Quotes are usually in the French style <<...>> and parentheses and exclamation marks are as in the Roman system: (...), !.

2.1.3. Problems with the Amharic Writing system

Despite its easiness of reading Amharic documents, there are some problems that have to be resolved. The first problem is the presence of redundant characters in the language writing system. These characters, (ነ, ሀ, ሐ, ኸ),". (አ, ዐ,) ,(ሰ, ሠ) and (ጸ ፀ), can be used interchangeably in one word. For example to write the name of the famous athlete Hayele Gebreselase one can write using either ሀይሌ ገብረስላሴ, ሐይሌ ገብረስላሴ, ነይሌ ገብረስላሴ, ሃይሌ ገብረሥላሴ, ኃይሌ ገብረሥላሴ, ሐይሌ ገብረሥላሤ, in twenty four different ways. This may create confusion for the reader and, if a search algorithm is implemented based on one of the representation; the others may be missed even if they represent the same thing. Having gemination of consonants is

another problem of Amharic writing system, which leads a reader to understand the word differently than it is intended; for example the word "ይመታል" which can be read yəmätall 'he hits', or yəmmättall 'he is hit' gives different meaning to the reader.

The other problem is the formation of compound words. Compound words are sometimes written as two separate words and other times as a single word. For example the word school can be written as ትምህርትቤት or ትምህርት ቤት or ትምህርት፡ቤት

2.2. Retrieval Systems

Information retrieval is the key technology for knowledge management which guarantees access to large corpora of unstructured data. Currently, it is also the basic technology behind Web search engines and an everyday technology for many Web users (Mandl, 2008)

According to (Baeza-Yates & Neto, 1999) Information Retrieval (IR) deals with representation, storage, organization of and access to information items. The representation and organization of the information items should provide the user with easy access to the information in which he/she is interested.

Mandl (2008) also defines Information Retrieval as a way of dealing with the storage and representation of knowledge and the retrieval of information relevant to a specific user request.

Information retrieval systems respond to the user queries that can be words from a natural language or it can be multimedia information. The similarity between the

content of the document and the query determine the relevance of retrieved information

In a real world there are two document types that need to be retrieved and be used. These are structured text documents and image documents. Document retrieval can use either a textual-based retrieval system or an image-based retrieval system. Document image retrieval system also has two way of retrieving, which is either recognition-based document image retrieval (OCR) or document image retrieval without explicit recognition.

2.2.1. Text-Based Document Retrieval

Text-based retrieval systems retrieve documents from the collection, based on the known character values that can be represented either in ASCII or Unicode representation formats. Text documents are represented by natural language words, mostly without considering syntactic or semantic context.

Text Retrieval Process

Modern computers are making it possible to represent a document by its full set of words; however, even modern computers might have to reduce the set of representative keywords. This can be accomplished through the elimination of stop words and the use of stemming , and the identification of *noun groups* (Baeza-Yates & Neto, 1999).

Different logical representation of documents can be obtained by performing the elimination of stop words, the use of stemming and identification of noun groups as shown in Figure 2.2.

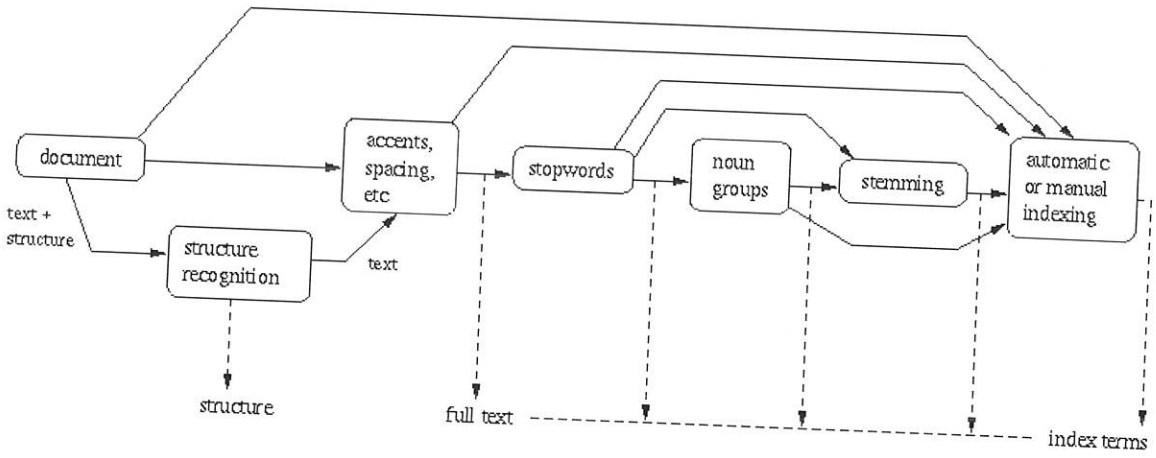


Figure 2.2 Logical views of a document adapted from (Baeza-Yates & Neto, 1999)

As shown in Figure 2.3, when the retrieval process initiated, the user first specifies a user need which is then parsed and transformed by the same text operations applied to the text. Then, *query operations* might be applied before the actual *query*, which provides a system representation for the user need, is generated. The query is then processed to obtain the *retrieved documents* (Baeza-Yates & Neto, 1999).

Before being sent to the user, the retrieved documents are ranked according to a likelihood of relevance. The user then examines the set of ranked documents in the search for useful information. At this point, he/she might pinpoint a subset of the documents seen as definitely of interest and initiate a user feedback cycle. In such a cycle, the system uses the documents selected by the user to change the query formulation.

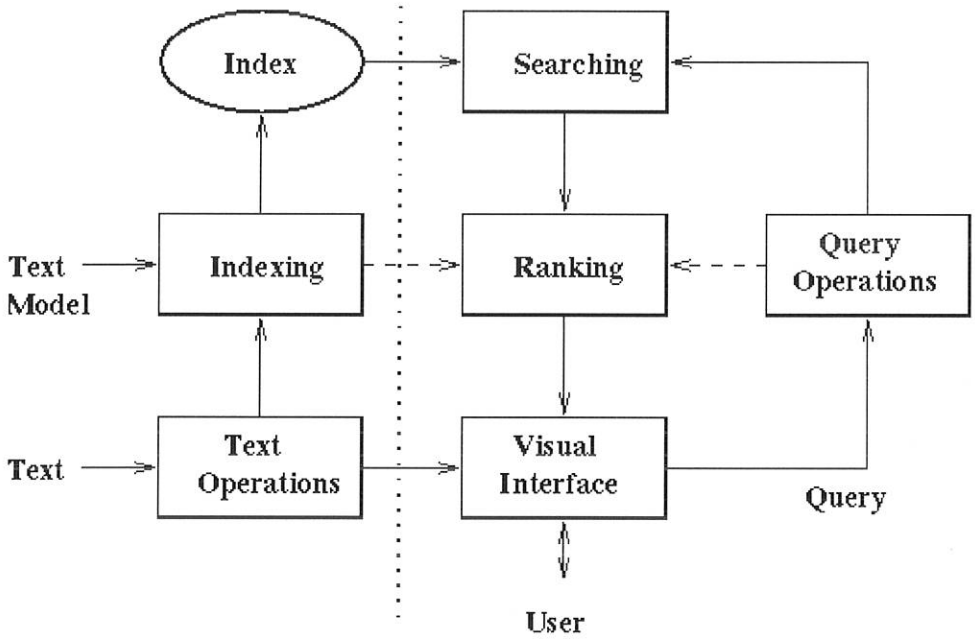


Figure 2.3 The process of retrieving information adapted from (Baeza-Yates & Neto, 1999)

2.2.2. Image-Based Document Retrieval

Modern technology has made it possible to produce, process, store, and transmit document images efficiently. Large quantities of printed documents are digitized and stored as images in databases.

Hull (2004) describes a document image database as that contains digital representations of documents that are captured on scanners or digital photocopiers. Queries about the contents of the database are answered by searching for a match between the query and an image in the database.

Early works on image retrieval were not generally based on visual features, but on the textual annotation of images. In other words, images were first annotated with text and then searched using a text-based approach from traditional database management systems (Long, zhang, & Feng, 2003).

In image document retrieval, image features are used for matching the features of the image document with that of the query. As Haque, Chowdhury, and Rahman (2005) state retrieval algorithms can be divided into three image featuring, feature matching and the combination of multiple evidences. Features are extracted, both from queries and words in image documents, based on which similarity between the query and documents is measured using matching algorithm.

In recent years, there has been much interest in the research area of Document Image Retrieval (DIR). DIR is relevant to document image processing. A Document image processing system needs to analyze different text areas in a page document, understand the relationship among these text areas, and then convert them to a machine-readable version, in which each character object is assigned to a certain class (Tan, 2004).

2.2.2.1. Recognition-Based Document Image Retrieval (OCR).

Optical Character Recognition (OCR) refers to the process of converting printed text documents into software-translated Unicode or ASCII Text. The printed documents available in the form of books, papers, magazines, etc. are scanned using standard scanners which produce an image of the scanned document.

Optical Character Recognition (OCR) is a type of document image analysis where a scanned digital image that contains either machine printed or handwritten script is input into an OCR software engine and translated into an editable machine-readable digital text format, character codes, such as ASCII

XoXus, (2003).and Seethalakshmi (2005) pointed out that OCR deals with machine recognition of characters present in an input image obtained using the scanning

classification methods pattern is recognized and compared to the OCR engine's large dictionary of characters from various fonts and languages. Once a likely match is made, it is recorded and a set of characters in the word block are recognized until all likely characters have been found for the word block. The word is then post processed to check whether it is correctly recognized or not by comparing with the large dictionary of complete words that exist for that language.

OCR is being used by libraries to digitize and preserve their holdings. OCR is also used to process checks and credit card slips and sort the mail. Billions of magazines and letters are sorted every day by OCR machines, considerably speeding up mail delivery.

Although the technology of Document Image Processing (DIP) may be utilized to automatically convert the digital images of these documents to the machine-readable text format using Optical Character Recognition (OCR) technology, it is typically not a cost effective and practical way to process a huge number of paper documents. One reason is that the technique of layout analysis is still immature in handling documents with complicated layouts. Another reason is that OCR technology still suffers an inherent weakness in its recognition ability, especially with document images of poor quality. Interactive manual correction/proofreading of OCR results is usually unavoidable in most DIP systems (Tan, 2004). The OCR must also be tolerant to the range of image distortions that occur in practice.

At present the recognition of Latin-based characters from well-conditioned documents can be considered as a relatively feasible technology. On the other hand, the processing of non-Latin scripts is still a subject of active research. (EICTDA, 2008)

Ethiopic script-based OCR processing is currently among the least developed ICT disciplines in the country. Developments in this area are mainly limited to preliminary research activities undertaken at different institutions of higher education's, such as the former School of Information Science for Africa (SISA). Such efforts are undertaken in an uncoordinated ways. (EICTDA, 2008)

An alternative solution is to perform matching directly on the image data. This bypasses the need for OCR and reduces the associated storage requirement. Any necessary in variances to image distortions are modeled at their source rather than one step removed, as they are if OCR data is utilized. Various solutions have been proposed for matching queries to database entries when both are images (Hull, 2004).

2.2.2.2. Document image retrieval without explicit recognition

Besides recognition based document image retrieval, searching from document is directly done to identify relevant documents from the corpus. This approach also depends on great extent on document image processing. Given a document image, it is preprocessed before detecting the content and segment the image into its constituent parts, mostly towards for retrieval from document image.

2.3. Document Image Segmentation

Segmentation is the process of dividing an image into parts that have a strong correlation with objects or areas of the real world contained in the image. (Supatra & Nualsawat, 2007) In other word, image segmentation is a process of extracting from the image domain one or more connected regions satisfying uniformity (Skarbek, et al, 1994).

There are four popular approaches of image segmentation (Chakravarty & Mitra, 2007): Pixel-based segmentation, region-based segmentation, edge-based segmentation and physics-based segmentation.

2.3.1. Pixel based segmentation

In this segmentation techniques (also called histogram thresholding), images are assumed to be composed of regions with different gray levels which are partitioned into a number of peaks, each corresponding to one region. Skarbek, et al(1994) classifies this segmentation method into three main techniques.

A. Histogram-based techniques: one or more peaks are identified, based on the identified pixels surrounding intervals are utilized in pixel classification process;

B. Segmentation by clustering data: pixel values are collected into groups with one or more representatives which next are used in the pixel classification process;

C. Segmentation by fuzzy clustering: fuzzy membership functions are evaluated for all pixels and for all fuzzy clusters defined; hard clusters of pixels are obtained by a defuzzification process and next subdivided into maximal connected regions.

Image segmentation by thresholding is a simple but powerful approach for images containing solid objects which are distinguishable from the background or other objects in terms of pixel intensity values. The pixel thresholds are normally adjusted interactively and displayed in real time on screen. When the values are defined properly, the boundaries are traced for all pixels within the range in the image. Grayscale thresholding works well when an image has uniform regions and contrasting background (Wu, 1999).

In a color image, a pixel can have a maximum value of 255 which can be expressed in terms of 8 bits. However, using binary codes for segmentation will not yield an optimum result, since two pixels of intensities separated by a value of even 1 (e.g., pixel values of 27 and 28) will be counted as two different segments (Chakravarty & Mitra, 2007). To solve this problem. Chakravarty (2007) proposed XOR-ing the binary bit with its previous (more significant) bit. So, instead of extracting the bit planes in binary from color images, it is recommended that the bit planes be in gray codes.

Techniques which make decisions based on local pixel information are effective when the intensity levels of the objects fall squarely outside the range of levels in the background. Because spatial information is ignored, however, blurred region boundaries can create problems (Drakos, 1996).

The thresholds are adjusted interactively by showing all pixels within the range in one color and all pixels outside the range to a different color, like in black and white. All pixels within the range can be segmented to generate the final boundaries.

Thresholding algorithm; this technique is based upon a simple concept. A parameter Θ called the brightness threshold is chosen and applied to the image a[i, j] as shown in Algorithm 2.1: (Young, Gerbrands, & van Vliet, 2007)

$$If\ a[i,j] > \theta \qquad a[i,j] = Object = 1$$

$$Else \qquad a[i,j] = background = 0$$

Algorithm 2.1 Threshold selection adapted from (Young, Gerbrands, & van Vliet, 2007)

Where i, j are the x and y coordinate of the pixel, Θ is the threshold value set and a[i,j] is the pixel value at the coordinate specified by i and j

The output is the label "object" or "background" which, due to its dichotomous nature, can be represented as a Boolean variable "1" or "0".

The central question in thresholding is: how to choose the threshold Θ? Correct threshold selection is crucial for successful threshold segmentation. Threshold selection can be interactive or can be the result of some threshold detection method.

- **Fixed threshold** - One alternative is to use a threshold that is chosen independently of the image data. If it is known that one is dealing with very high-contrast images where the objects are very dark and the background is homogeneous and very light, then a constant threshold of 128 on a scale of 0 to 255 might be sufficiently accurate (Young, Gerbrands, & van Vliet, 2007).

- **Histogram-derived thresholds** - In most cases the threshold is chosen from the brightness histogram (color value of the pixel) of the region or image that we wish to segment (Young, Gerbrands, & van Vliet, 2007).

- **Isodata algorithm** - The histogram is initially segmented into two parts using a starting threshold value such as $\Theta_0 = 2B-1$, half the maximum dynamic range. The sample mean (mf,0) of the gray values associated with the foreground pixels and the sample mean (mb,0) of the gray values associated with the background pixels are computed. A new threshold value Θ_1 is now computed as the average of these two sample means. The process is repeated, based upon the new threshold, until the threshold value does not change any more (Young, Gerbrands, & van Vliet, 2007).

- **Background-symmetry algorithm** - This technique assumes a distinct and dominant peak for the background that is symmetric about its maximum (Young, Gerbrands, & van Vliet, 2007).

- **Triangle algorithm** - A line is constructed between the maximum of the histogram at brightness b_{max} and the lowest value b_{min} = (p=0)% in the image. The distance d between the line and the histogram h[b] is computed for all values of b from b = b_{min} to b = b_{max}. The brightness value b_o where the distance between h[b_o] and the line is maximal is the threshold value, that is, $\theta = b_o$. This technique is particularly effective when the object pixels produce a weak peak in the histogram (Young, Gerbrands, & van Vliet, 2007).

The thresholding produces a segmentation that yields all the pixels that, in principle, belong to the object or objects of interest in an image (Young, Gerbrands, & van Vliet, 2007)

2.3.2. Area-based segmentation

This segmentation algorithms use uniformity criteria calculated in regions of image domain. There are two types of techniques in this type of segmentation. (Skarbek, et al, 1994)

A. Region growing: a number of basic uniform regions (seeds) are given and different strategies are applied to join surrounding neighborhoods; to distinguish this group from the next one it is important that seeds here are not resulting from splitting or subdivision processes of non-uniform regions;

B. Split and merge: start from non-uniform regions, subdivide them until uniform ones are obtained, and then apply some merging heuristics to fit them to maximal possible uniform areas

The image is partitioned into connected regions by grouping neighboring pixels of similar intensity levels. Adjacent regions are then merged under some criterion involving perhaps homogeneity or sharpness of region boundaries (Drakos, 1996).

2.3.3. Edge-based segmentation

This type of segmentation is broadly classified as local or global (Skarbek et al, 1994).. The local technique to determine an edge point needs only information of the neighborhood point. The global technique, on the contrary, makes sort of global optimization, and therefore the given edge point could be identified after many optimization steps involving changes in large areas

Edge-based methods center around contour detection: their weakness in connecting together broken contour lines make them, too, prone to failure in the presence of blurring (Drakos, 1996).

2.3.4. Physics-based segmentation

This segmentation technique employs physical models to partition an image into regions that correspond to surfaces or objects in the scene. The objective of these techniques is to segment a multi-spectral image at object boundaries and not at the edges of highlights and shadows in an image (Skarbek et al, 1994).

Physics-based segmentation techniques allow the segmentation of real images based on physical models for image formation (Skarbek et al, 1994).

In the analysis of the document images it is essential to distinguish between the foreground that is the objects of interest and background. The techniques that are used to find the objects of interest usually referred to as segmentation techniques. It is important to understand that: there is no universally applicable segmentation technique that will work for all image documents, and no segmentation technique is perfect. (Young, Gerbrands, & van Vliet, 2007)

2.4. Representation of document image s

Given a document image, it is preprocessed offline to threshold, skew-correct, remove noise and thereafter to segment into words. Then the features are extracted for individual words. They are also normalized so that the word representations become insensitive to variations in size, font and various degradations popularly present.

Feature extraction follows the segmentation phase in image document retrieval. The segmented individual image is considered and features are extracted. Features include visual browsing, color similarity measures, and text. Primitives are used for extracting parts of an image.

Feature extraction consists of identifying the position and properties of features in an image such as points, lines or regions. Each feature has an attached vector giving its position and associated properties. For instance a point might be characterized by its difference from its neighborhood, a line may have the property of length, angle, curvature and width, and a region might have values of texture, area, circumference and orientation (Hamlyn, 1994).

Shape features of objects or regions have been used in many content-based image retrieval systems. Compared with color and texture features, shape features are

usually described after images have been segmented into regions or objects. Since robust and accurate image segmentation is difficult to achieve, the use of shape features for image retrieval has been limited to special applications where objects or regions are readily available (Long, zhang, & Feng, 2003).

The state-of-art methods for shape description can be categorized into either boundary-based (rectilinear shapes, polygonal approximation, finite element models, and Fourier-based shape descriptors) or region-based methods (Long, zhang, & Feng, 2003).

2.4.1. Word representation

It is difficult to extract word images, particularly from newspapers and old books, because their quality is poor most of the time. According to Million and Jawahar (2008), the difficulty rose from the following factors of the documents:

A. Excessive dusty noise,
B. Large ink-blobs joining disjoint characters or components,
C. Vertical cuts due to folding of the paper,
D. Cuts of arbitrary direction due to paper quality or foreign material,
E. Degradation of printed text due to the poor quality of paper and ink,
F. Floating ink from facing pages, etc.

Million and Jawahar (2008) also found at least three categories of features that are effective to address word representation:

Word Profiles: Profiles of the word provide a coarse way of representing a word image for matching. Profiles like upper word, lower word, projection, density and transition profiles are used for word representation. Some of them (e.g., upper and

on simple functions of those feature values, e.g., Euclidean distance. Unfortunately, these measures are often frail and difficult to interpret (Stauffer, 2004).

For proper searching and retrieval, one needs to identify the similar words and group them, and evaluate the relative importance of each of these words and word clusters.

2.5.1. Similarity measure

Measuring the similarity between documents has practical applications in document image retrieval. The word, rather than the character, is the basic unit of meaning in document image retrieval (Tan, 2004).

Feature vectors from query and target documents are matched to assess similarity, which is sorted to decide the presentation order of the documents in the list. Conceptually, similarity is an opposite concept of difference and hence differences are computed from the feature vectors of the target and query objects. Several different distance functions are used to compute similarity such as Euclidean, Canberra, and Manhattan distance functions.

Euclidean Similarity

Euclidean distance is the most simple to conceive and it is given by the following formula, where t_i is the vector value of the image word at i's position and q_i is the vector value of the query word at i's position (Haque, Chowdhury, & Rahman, 2005):

$$d_{Euclidean} = \sqrt{\sum_{i=0}^{M-1} (t_i - q_i)^2}$$

Equation 2 1: Euclidean distance measure

Cosine Similarity

The similarity score between two document vectors is defined as their scalar product divided by their lengths. A scalar product is calculated by summing up the products of the corresponding elements. This is equivalent to the cosine of the angle between two document vectors seen from the origin. So, the similarity between the query document image Q and the archived document image Y is (Tan, 2004):

$$S(\bar{Q}.\bar{Y}) = \frac{\sum_{k=1}^{k} q_k \cdot y_k}{\sqrt{\sum_{k=1}^{k} q_k^2 \sum_{k=1}^{k} y_k^2}}$$

Equation 2.2 Cosine similarity measure equation

Where, $\bar{Q}$ is the document vector of the image Q, and $\bar{Y}$ is the document vector of the image Y. K is the dimension of the query and document vector (Tan, 2004) .

Minkowski-Form Distance

If each dimension of an image feature vector is independent of others and is of equal importance, the Minkowski-form distance L_p is appropriate for calculating the distance between two images. This distance is defined as (Long, zhang, & Feng, 2003)

$$D(K_1, K_2) = \left(\sum_{n} |K_1(n: T_1) - K_2(n: T_2)|^p \right)^{1/p}$$

Equation 2.3 Minkwski-Form Distance

Where K_1 (n:T_1) and K_2(n:T_2) are the two distributions and =1≤p ≤∞. L_1 computes the Manhattan distance, L_2 is the Euclidean distance and finally L_∞ measures the maximum distance (Dr. Fugui Long, 2003).

Quadratic Form (QF) Distance

The Minkowski distance treats all bins of the feature histogram entirely independently and does not account for the fact that certain pairs of bins correspond to features which are perceptually more similar than other pairs. To solve this problem, quadratic form distance is introduced (Long, zhang, & Feng, 2003).

$$D(I,J) = \sqrt{(F_I - F_J)^T A (F_I - F_J)}$$

Equation 2.4 Quadratic Form Distance

Where $A=[a_{ij}]$ is a similarity matrix, and a_{ij} denotes the similarity between bin i and j. F_I and F_J are vectors that list all the entries in $f_i(I)$ and $f_i(J)$ (Long, zhang, & Feng, 2003).

Mahalanobis Distance

The Mahalanobis distance metric is appropriate when each dimension of image feature vector is dependent on each other and is of different importance. It is defined as (Long, zhang, & Feng, 2003)

$$D(I,J) = \sqrt{(F_I - F_J)^T C^{-1} (F_I - F_J)}$$

Equation 2.5 Mahalanobis Distance

Where C is the covariance matrix of the feature vectors. The Mahalanobis distance can be simplified if feature dimensions are independent. In this case, only a variance of each feature component, c_i, is needed (Long, zhang, & Feng, 2003).

$$D(I,J) = \sum_{i=1}^{N} (F_I - F_J)^{2/c_i}$$

Equation 2.6 Simplified Mahalanobis Distance

Kullback-Leibler (KL) Divergence and Jeffrey-Divergence (JD)

The Kullback-Leibler (KL) divergence measures how compact one feature distribution can be coded using the other one as the codebook. The KL divergence between two images I and J is defined as (Long, zhang, & Feng, 2003):

$$D(I,J) = \sum_i f_i(I) log \frac{f_i(I)}{f_i(J)}$$

Equation 2.7 Kullback-Leibler divergence measures

The Jeffrey-divergence (JD) is defined by:

$$D(I,J) = \sum_i f_i(I) log \frac{f_i(I)}{f_i} + f_i(J) log \frac{f_i(J)}{f_i}$$

Equation 2.8 Jeffrey-divergence measures

Where $fi = \frac{[f_i(I)+f_i(j)]}{2}$ In contrast to KL-divergence, JD is symmetric and numerically more stable when comparing two empirical distributions (Long, zhang, & Feng, 2003)

2.6. Retrieval and Ranking

Using the similarity measure and retrieving a document by itself is not enough for a retrieval system. Ranking the retrieved documents is necessary so that users will get a highly relevant document at the top of the search result.

In information retrieval, a ranking function is a function used by search engines to rank matching documents according to their relevance to a given search query. (Wikipedia, 2009)

Some very simple ranking functions include: (Wikipedia, 2009)

- The constant ranking function assigning the same score to all documents.
- The term frequency ranking function counting the number of times that each query term occurs in the document, then summing these.
- The tf-idf ranking function computing the product of the term frequency and inverse document frequency for each query term, then multiplying these.

More sophisticated ranking functions include:

- Okapi BM25: a variant of tf-idf. As of 2008, represents the state-of-the-art and is used in many practical applications.

Term frequency: The term count in the given document is simply the number of times a given term appears in that document.

The tf–idf weight (term frequency–inverse document frequency) is a statistical measure used to evaluate how important a word is to a document in a collection or corpus.

The inverse document frequency is a measure of the general importance of the term (obtained by dividing the number of all documents by the number of documents containing the term, and then taking the logarithm of that quotient).

2.6.1. Performance Measures

The number of known relevant documents is usually used to calculate the performance of the system. Both recall and precision measures are set oriented. However, most current systems present ranked results. In this case, a recall and precision value pair

can be obtained for each position on the ranked list taking into account all documents from the top of the list down to that position. Plotting these values leads to the recall-precision graph. The average of precision values at certain levels of recall is calculated as the mean average precision (MAP), which expresses the quality of a system in one number (Mandl, 2008).

As a consequence, the empirical evaluation of performance is a central concern in information retrieval research (Baeza-Yates & Neto, 1999).

The most important and commonly used measurements are recall and precision. Recall indicates the ability of a system to find relevant documents, whereas precision measures show how good a system is in finding only relevant documents (Mandl, 2008).

Recall is calculated as the fraction of relevant documents found among all relevant documents, whereas precision is the fraction of relevant documents in the result set. The recall requires knowledge of all the relevant documents in a collection that could never be put together in any real world collection (Mandl, 2008).

Recall is defined as the ratio between the number of relevant items retrieved and the total number of relevant items in the collection (Haque, Chowdhury, & Rahman, 2005):

$$Recall = \frac{Number\ of\ relevant\ documents\ that\ are\ retrievd}{Total\ number\ of\ relevant\ documents\ in\ the\ collection}$$

Equation 2.9 Performance Measure (Recall)

Precision measures the retrieval accuracy and is defined as the ratio between the number of relevant items retrieved and the number of total items retrieved for a given

level of retrieval Hence, precision provides the level of purity of retrieval at a given recall level (Haque, Chowdhury, & Rahman, 2005):

$$Precision = \frac{Number\ of\ relevant\ documents\ that\ are\ retrieved}{Number\ of\ documents\ being\ retrieved}$$

Equation 2.10 Performance Measure (Precision)

In fact precision and recall are not enough for evaluating IR systems. Modified measures that combine precision and recall and consider the order of the retrieved documents are needed. The most widely-used measure is the mean average precision (MAP score). It computes precision at each point in the ranking where a relevant document was found, then averages over these values and then over all queries (Manning, Raghavan, & Schütze, 2008)

Mean Average Precision: an information retrieval performance measure that combines precision and recall and rewards relevant documents ranked higher in the list of retrieved documents computed as the average of the precision values for each relevant document in the ranked results. (Inkpen, 2008)

The precision and recall are based on the retrieved documents of the system. Precision at r is the average of the precision after each relevant document is retrieved and it is denoted by Equation 2.11

$$AveP_r = \frac{\sum_{r=1}^{N}\left(P(r)\ \mathrm{X}\ rel(r)\right)}{number\ of\ relevant\ documents}$$

Equation 2.11 Mean average precision

Where *r* is the rank, *N* the number retrieved, *rel()* a binary function on the relevance of a given rank, and *P()* precision at a given cut-off rank:

When no relevant document is retrieved, the precision value is taken to be 0. When a relevant document is not retrieved at all, the precision value in Equation 2.11 is taken to be 0.

Usually precision is more important than recall in IR systems, if the user is looking for an answer to a query, not for all the possible answers. Recall can be important when a user needs to know all the relevant information on a topic. (Inkpen, 2008)

Among evaluation measures, MAP has been shown to have especially good discrimination and stability. For a single information need, Average Precision is the average of the precision value obtained for the set of top K documents existing after each relevant document is retrieved, and this value is then averaged over information needs (Manning, Raghavan, & Schütze, 2008).

2.7. Amharic Retrieval Systems

Currently, the number of research works done on Amharic document retrieval system includes: (Yoseph, 2005), (Yalemisew, 2005) (Eyasu, 2005), (Tewodros, 2003), (Bizuneh, 2003), (Mulegeta, 2002), (Ethiopia, 2002) and (Saba, 2001); all of them are based on text retrieval techniques.

(Saba, 2001) By designing a database system that stores Amharic web page data, tries to retrieve Amharic documents on the web.

(Tewodros, 2003) Conducted a research using The Latent Semantic Indexing (LIS) technique to retrieve Amharic documents

Using Web Based Self Organization Map (WEBSOM) that applies a neural network's self organizing algorithm (Bizuneh, 2003) explored the possibility of retrieving texts written in Amharic language.

(Yalemisew, 2005) Uses document clustering system for Amharic documents. He uses frequent item set hierarchical algorithm. A top down cluster search mechanism is used to find out the best matching cluster for a query.

Atelach Alemu and Lars Asker (2007) have made an attempt to develop Amharic retrieval system using fuzzy matching for Amharic-English cross lingual information retrieval.

For document image retrieval, the attempt made so far is developing an OCR engine for Amharic. Some of the research in this respect include (Wondewosen, 2004), (Million, 2000), Dereje (1999) and Worku (1997). They attempted to develop Optical Character Recognition (OCR) systems for representing Amharic documents in textual format, so that other can use the result to design search engines for Amharic text. However, the performance of these OCR engines is not yet ready for application.

2.7.1.1. Related works in Document image retrieval

In recent years, a number of attempts have been made by researchers to avoid the use of character recognition for various document image retrieval applications. Tan (2004) described a method for automatically selecting sentences and key phrases to create a summary from an image document without any need for recognition of the characters in each word.

Liu and Jain (1998) presented an image-based form of document retrieval. They proposed a similarity measure for forms that is insensitive to translation, scaling,

moderate skew, and image quality fluctuations, and developed a prototype retrieval system. Niyogi (1997) described an approach to retrieve information from document images stored in a digital library by means of knowledge-based layout analysis and logical structure derivation techniques, in which significant sections of documents, such as the title, are utilized. Tang (1994) proposed methods for automatic knowledge acquisition in document images by analyzing the geometric structure and logical structure of the images.

In the domain of Chinese document image retrieval, He (1999) proposed an index and retrieval method based on the stroke density of Chinese characters. Spitz (1997) described character shape codes for duplicate document detection, information retrieval, word recognition, and document reconstruction, without resorting to character recognition.

Tan (2004) proposed a method for text retrieval from document images without the use of OCR. In this method, documents are segmented into character objects, whose image features are utilized to generate document vectors. Then, the text similarity between documents is measured by calculating the dot product of the document vectors.

A segmentation-free word image matching approach treats each word object as a single, indivisible entity, and attempts to recognize it using features of the word as a whole. For example, Ho (1992) proposed a word recognition method based on word shape analysis without character segmentation and recognition.

Searching in scanned documents is an important problem in Digital Libraries. If OCR systems are not available, the scanned images are inaccessible. To resolve this problem, a searching procedure without an intermediate textual representation should

be designed and used. Word profiles, structural features and transform domain representations can be employed for characterizing the word images (Million & Jawahar, 2008).

Million and Jawahar, (2004) proposed an image document retrieval system and they represent the conceptual diagram of the searching procedure as shown in the Figure 2.5.

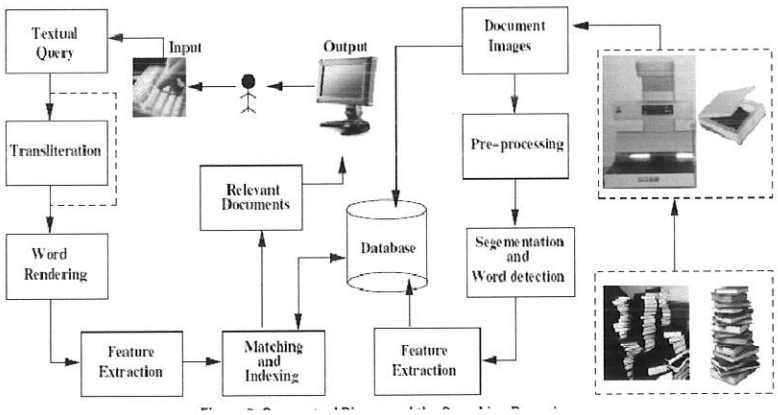


Figure 2 5 Conceptual Diagram of the Searching Procedure Adapted from (Million and Jawahar, 2008)

The effort made by (Million & Jawahar, 2008) use three categories of features: word profiles, moments and transform domain representations.

Word profiles provide a coarse way of representing word images for matching. Projection, transition, upper and lower profiles are features considered for word image representation. While projection and transition profiles capture the distribution of ink along one of the two dimensions in a word image, upper and lower word profiles capture part of the outlining shape of a word.

They also use a dynamic time warping (DTW) algorithm that have a great advantage in aligning and comparing word images for a word image matching.

CHAPTER THREE:

3. AMHARIC DOCUMENT IMAGE RETRIEVAL TECHNIQUES

With the development of Information Communication Technology and the advancement of computers in the day to day life of the society, the need for accessing documents in digitized form is growing fast. The availability of documents in digitized form helps users to access, modify, and use documents easily.

However, there are still a lot of documents that are found only in hard copy format and need to be digitized. OCR is one way of digitizing hardcopy documents by reading the characters of printed and handwritten documents into the computer. Another system that can be considered is digitizing documents in image format and storing them in the computer for future use without any character recognition.

This research is concerned with the retrieval of document images without any character recognition system. For this purpose, the proposed retrieval system is designed in such a way that it incorporates the following main image processing techniques. Documents are scanned and processed offline. To facilitate searching, scanned document images are segmented into word level. Then, the features of the word images are extracted and are used for matching word images.

The system accepts textual queries from users. A textual query is first converted into an image by rendering. Features are extracted from these images and then search is carried out for retrieval of relevant document images. Results of the search are document images containing the queried word.

The performance of the system will be evaluated using precision and recall measures.

3.1. System Design

The proposed system incorporates: document images and user queries. The document image is preprocessed, segmented and the feature of each word is extracted. The system also accepts user query in to the system in text form. The text form query is then converted to image query, the converted image query is preprocessed, segmented and the feature is extracted. The extracted feature of image query will be matched with that of document image and document images that have a word which is similar to the query word are retrieved. The design is shown in figure 3.1

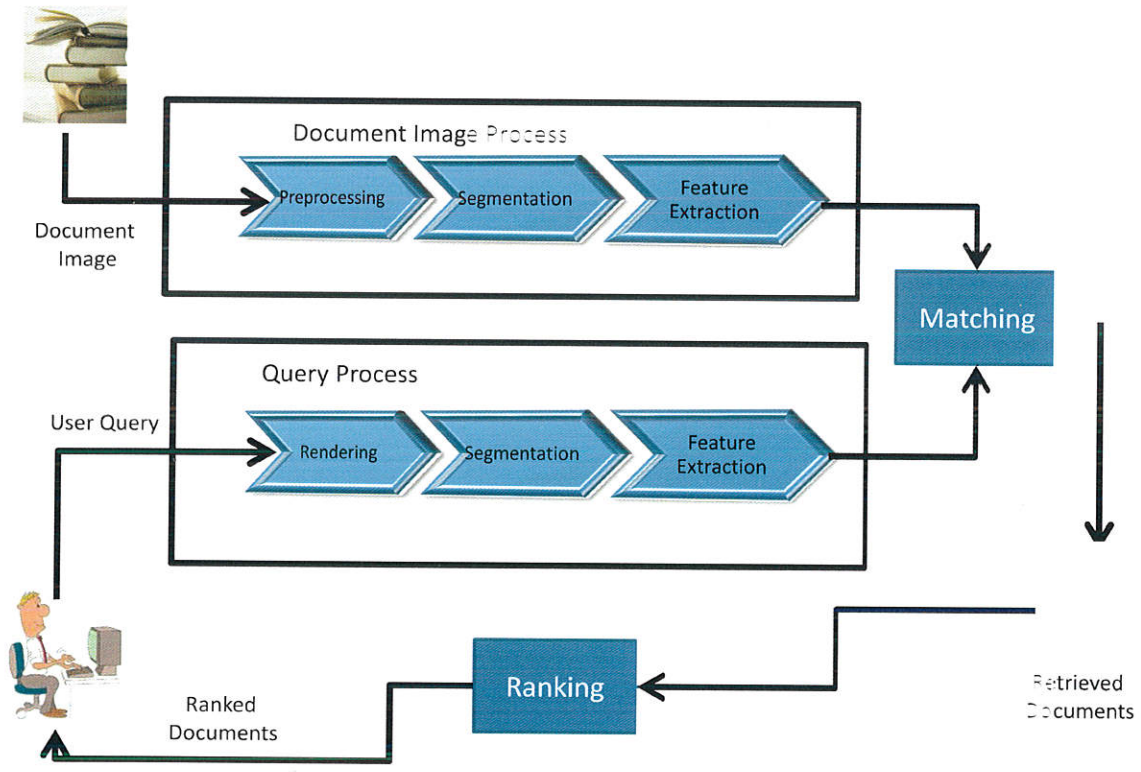


Figure 3.1 Architecture of the proposed system

3.2. Preprocessing

The document image needs to be preprocessed before it is submitted for image segmentation. The first task to be performed in image preprocessing is to access each pixel of the image and extract the color value of the image. In this research work, it the document image is scanned in grayscale, with 300dpi intensity. In grayscale image, color values from 0-255 have been used for processing. When the pixel value is 0, it means that the pixel is a complete black and when the value is 255 it is a complete white. In-between the color is gray and it is darker when it reaches 0 and lighter when it reaches 255.

3.2.1. Binarization

Binerization is the process of representing the document images in the form of binary number. Based on the fixed threshold method, algorithm 3.1 is used to determine the pixel values as part of a background or an object (foreground). This technique is based upon a simple concept. A brightness threshold parameter Θ is chosen and applied to the image DocImage[i, j] as shown in Algorithm 3.1:

```
For i=1 to width of Image do
    For j=1 to height of image do
        If DocImage[i, j]≥Θ     DocImage[i, j]=0 // part of the background
        Else                    DocImage[i, j]=1 //part of the forground/object
```

Algorithm 3.1 Threshold Selection

Where DocImage[i ,j] is the pixel value at the x and y coordinate specified by i^{th} and j^{th} point.

3.3. Segmentation

Image segmentation is the process of pixel detection in order to identify the attributes of pixels and defines the boundaries for pixels that belong to same group (Wu, 1999).

In this research the image segmentation by thresholding technique is implemented. Image segmentation by thresholding is a simple but powerful approach for images containing solid objects which are distinguishable from the background or other objects in terms of pixel intensity values (Wu, 1999). The pixel thresholds are normally adjusted interactively and displayed in real-time on screen. When the values are defined properly, the boundaries are traced for all pixels within the range in the image. Gray scale thresholding works well when an image has uniform regions and contrasting background.

In this research work line and word segmentation are considered.

3.3.1. Line Segmentation

Line segmentation is a process of identifying the lines that hold text and lines that are blanks from the document image. By using the value of the pixels in the line each line is segmented using the fixed threshold method. This technique is used on the bases of the Θ value and applied to the DocImage[i, j] as shown in Algorithm 3.2.

Lines that contain text and lines that does not contain text are identified by

```
For i=0 to height of Image do
        For j=0 to width of image do
        {
                If DocImage[i, j]≤Θ  =Text Line //part of the forground/object
                Break
        }
                If DocImage[i, j]≥Θ =Nontext Line //part of the background
```

Algorithm 3.2. Determining text line and non text line

Where DocImage[i ,j] is the pixel value at the x and y coordinate specified by i^{th} and j^{th} point.

3.3.2. Word Segmentation

Since the lines that hold text data and non text data are identified, the next step is to identify the top and bottom border of each text area and the identification of each word image boundaries.

Identification of word boundaries requires the task of distinguishing words from word spaces. Although it seems simple, the presence of spaces that precede or succeed a character makes it difficult to identify a word separator from a character separator. Hence, it complicates the job of word border identification as one of the challenges in word level segmentation (Le, Thoma, & Wechsler, 1996).

According to Le, Thoma, & Wechsler (1996) a word separator space is greater than one third of the character height in English text documents. Since this research uses computer printout Amharic documents, the researcher accepts the proposal and the word spacing used in this process is chosen to be one-third of a character height.

By identifying the word borders each text line in the document image is further segmented into word images.

3.3.3. Query Segmentation

The process of Query Segmentation is the same as that of image segmentation as long as the query is submitted in image form. However, the system is designed to accept the query in text form. Hence, it is necessary to convert the submitted query text

into an image. For this purpose, the proposed system incorporates an algorithm that changes the input text to image. This process is called text rendering.

Rendering is the process of generating an image from a model, by means of computer programs. The model is a description of three-dimensional objects in a strictly defined language or data structure. It would contain geometry, viewpoint, texture, lighting, and shading information (Wikipedia, 2009).

Text rendering is the process of preparing characters that are stored in memory for display as glyphs (Apple Computer, 1996).

To roughly evaluate the similarity between the word object extracted from the document image and for the user query word, the query has been resized to fit the document image size.

3.4. Word Feature Extraction

After word images are detected from documents the next step is extracting the features of the word image. The features extracted must describe certain characteristics of the word. The choice of features is often based on heuristics and it depends on the data. Here, the technique used for feature extraction of the word image considers the word shape analysis. The word shape analyses that are considered in this research are two types. The first selects an image feature called the vertical bar pattern in order to construct document vectors. The second also extracts an image feature based on the pixel values in the predefined area. The idea is to calculate a vector value for the image and represent each word image in one vector. This will help to represent word images as a sequence of vectors. For example a word A can be represented as

In typical retrieval systems, recall tends to increase as the number of retrieved items increases; while at the same time the precision is likely to decrease. Further, when the number of relevant images is greater than the number of the retrieved images, recall is meaningless. As a result, precision and recall are only rough descriptions of the performance of the retrieval system (Long, zhang, & Feng, 2003).

By combining the precision and recall it is possible to obtain a single objective measure. This is called F-measure. It is the weighted harmonic mean of precision and recall, this can be defined as:

$$F = \frac{2(Precision * Recall)}{Precision + Recall}$$

Equation 3.2 : Performance Evaluation (F-Measure)

The F-measure is derived from recall and precision. It is a strict measurement because it does not only reflect the absolute value of the recall and precision, but also reflects the degree of balance between the two (Lu & Tan, 2007).

Precision, Recall and F-measure are set-based measures where order of documents not taken into account. If the first k retrieved documents are consider, and the precision and recall values are computed it is possible to plot the relation between precision and recall for each value of k. If the $(k+1)^{st}$ is not relevant recall is the same, but precision decreases. If the $(k+1)^{st}$ is relevant recall and precision increase.

Since precision and recall by themselves do not give us a good evaluation measure and users are mostly interested on the first k documents of the retrieval result, an approach that measures the retrieval performance of the first k documents tells more

about the system performance. In this research the R-precision is calculated by using Equation 2.11 for the first 10 retrieved documents.

For an information need, the average precision is the arithmetic mean of the precisions for the set of top k documents retrieved after each relevant document is retrieved.

Precision at k is interested in the proportion of good results among the k first answers. Precision at a fixed level has an advantage because it does not need an estimate of the set of relevant document and a disadvantage because it is unstable measure, which means it does not average well

CHAPTER FOUR

4. EXPERIMENTATION

The present system uses image documents that are stored in a computer. The Image documents are preprocessed and binerized to give a file that can be accessed by the system. The Image documents are then segmented line by line and word images are identified. Each word image feature is extracted and document vectors are generated.

The user gives query text to the system and the query is rendered to generate a query image. This query image is preprocessed and word images are identified. The segmented word feature is extracted and query vector is generated for the word.

Both vectors, are matched using a similarity measure and with a selected threshold similarity score image documents that have a threshold value are retrieved The whole process is shown on figure 4.1.

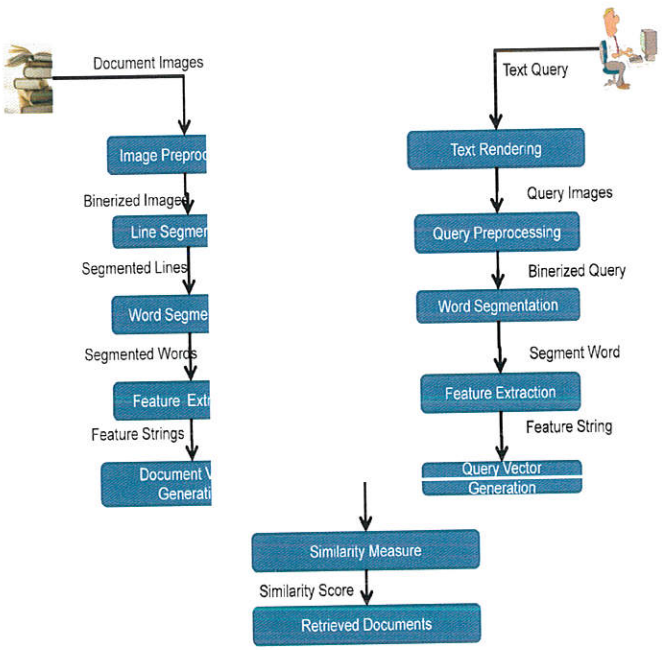


Figure 4.1 Document Image retrieval Process.

4.1. Preprocessing

After the system receives the document images it preprocesses them and the first preprocessing task is determining the color values of each pixel. In this research work the following Java code is used by implementing Java Advanced Imaging (JAI) for determining the color value of each pixel.

The Java implementation code for the determining color value of the pixel

```
     File image = new File(args[0]);
//Open the image (using the name passed as a command line
parameter)
       PlanarImage pi = JAI.create("fileload", args[0]);
//get the width and height of the image
  int width=pi.getWidth();  int height = pi.getHeight();
//get the sampleModel of the image
     SampleModel sm=pi.getSampleModel();
//Get the number of color bands
      int nbands =sm.getNumBands();
//define arrays of pixel
     int [] pixel = new int[nbands];
// Get an iterator for the image.
     RandomIter iterator = RandomIterFactory.create(pi, null);
// Scan the image pixels in each row and column.
     for (int h=0; h<height; h++)
        for (int w=0; w<width; w++)
          {
// Get the array of values for the pixel on the x and y
coordinate.
            iterator.getPixel(w,h,pixel);
            int  pixeltotal=0;
            int pixelaverage = 0;
             for (int band = 0; band < nbands; band++) {
                pixeltotal = pixel[band] + pixeltotal;;
             }

            pixelaverage=pixeltotal / nbands }
```

Once the color value of each pixel is known the next step needed is changing each pixels to the value of 0 and 1, this is called bineraization.

Through experiment, a threshold parameter Θ of 128 has been used for bineraization using a fixed threshold selection method.. Hence, pixels which have a value between 0-127, inclusive, are considered as black, and pixels with a value between 128 to 255 have been considered as white, so that the background and the foreground of the image can be easily identified.

The preprocessing step of this research do not include character point size, word spacing and character leading. However, all documents used have a font size of 12, a font type of Power geez Unicod1 and a plain style.

4.2. Segmentation

4.2.1. Line Segmentation

Areas of non-textual borders can be roughly located by identifying non-textual rows; at the same time, areas of textual borders can be located by identifying textual rows.

Since leading characters are not considered in this paper, a textual row and a non textual row are identified based on the presences or absence of a pixel which has its average color value less than 128. Using this criterion, the image is segmented into a series of rows for identifying a line that has a text or a line for which each pixel contains a white space.

The output of the horizontal image segmentation coordinates are put in a file with a prefix of "D" and "B" to identify the lines that has a text and white space, respectively. The partial view of the output file is shown on Table 4.1

B 0	B 1	B 2	B 3	B 4	B 5	B 6	B 7	B 8	B 9
B 10	B 11	B 12	B 13	B 14	B 15	B 16	B 17	B 18	B 19
B 20	B 21	B 22	B 23	B 24	B 25	B 26	B 27	B 28	B 29
B 30	B 31	B 32	B 33	B 34	B 35	B 36	B 37	B 38	B 39
B 40	B 41	B 42	B 43	B 44	B 45	B 46	B 47	B 48	B 49
B 50	B 51	B 52	B 53	B 54	B 55	B 56	B 57	B 58	B 59
B 60	B 61	B 62	B 63	B 64	B 65	B 66	D 67	D 68	D 69
D 70	D 71	D 72	D 73	D 74	D 75	D 76	D 77	D 78	D 79
D 80	D 81	D 82	D 83	D 84	D 85	D 86	D 87	D 88	D 89
D 90	D 91	D 92	D 93	D 94	D 95	D 96	D 97	D 98	D 99
D 100	D 101	D 102	D 103	D 104	D 105	D 106	D 107	D 108	D 109
D 110	D 111	B 112	B 113	B 114	B 115	B 116	B 117	B 118	B 119
B 120	B 121	B 122	B 123	B 124	B 125	B 126	B 127	B 128	B 129

Table 4.1 The partial view of the output file that holds the line numbers of text area and non text area.

The "B" and "D" indicates if the line is blank or data space, respectively, and the numbers indicate the vertical pixel position. From the above table one can easily conclude that the document has no text data values from pixel 0 to pixel 66 and has text data from pixel 67 to pixel 111.

For the purpose of identifying lines that contain text area and lines that contain non textual area the following Java code is implemented.

The Java implementation code for identifying lines that contain text and lines that does not contain text

```
FileWriter fwhb = new FileWriter("border.txt");
PrintWriter pwhb = new PrintWriter(fwhb);
for (h = 0; h < height; h++) {
    for (w = 0; w < width; w++) {
        if (pixelaverage / nbands < 128) {
```

```
                pwhb.println("D " + h);
                break;
            }
        if (pixelaverage / nbands > 128) {
            pwhb.println("B " + h);
        }
    }
    pwhb.flush();
    pwhb.close();
```

The above Java code segments the document horizontally by identifying a horizontal text area and a horizontal non text area as shown in Fig 4.2

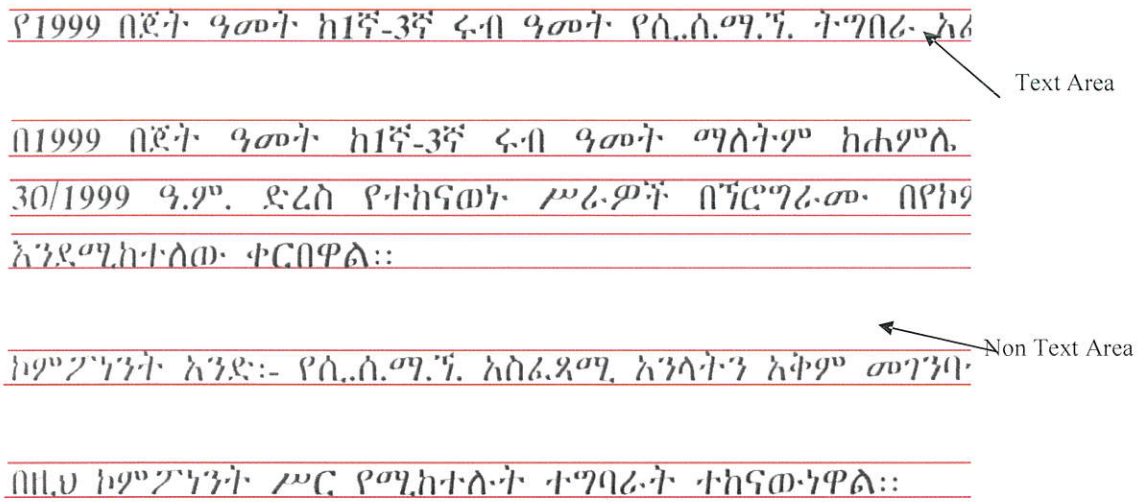


Figure 4.2 Horizontal segmented document image

By using the above Java code the lines that contain data and the lines that contain white spaces are identified and the output is stored to the text file. Using the text file created, the horizontal boundaries of the text areas are identified, and these boundaries in turn are used for identifying word borders.

The Java code below is used for identifying horizontal borders of the text area.

The Java implementation code for identifying horizontal borders of the text area

```
FileWriter fwvb = new FileWriter ("HorizontalBord.txt");
pwvb = new PrintWriter(fwvb);
BufferedReader input = new BufferedReader (new FileReader
("border.txt"));
```

```
Vector<String> line2 = new Vector<String>(2, 2);
while ((line = input.readLine()) != null) {
line2.addElement(line);                    }
for (int i = 0; i < (line2.size() - 1); i++) {
          if       (line2.elementAt(i).charAt(0)       ==
          line2.elementAt (i + 1).charAt(0)) {
          continue;
          }
          while (line2.elementAt(i).contains("B")) {
          String[] data = line2.elementAt(i).split(" ");
              Lnumber = Integer.parseInt(data[1]);
              border = Lnumber;
              i++                    }
          dataStart = (border + 1);
          while (line2.elementAt(i).contains("D")) {
          String[] data = line2.elementAt(i).split(" ");
              Lnumber = Integer.parseInt(data[1]);
              containt = Lnumber;
              i++;
          }
          dataEnd = containt;
      }
      input.close();
      pwvb.flush();
      pwvb.close();
```

4.2.2. Word Segmentation

As discussed in chapter Three Identification of word boundaries requires the task of distinguishing words from word spaces. For this purpose the following Java code is used to identify spaces between characters as a part of a word image.

The Java implementation code for distinguishing words from word spaces

```
Int  count=0;
 for (w = 0; w < width; w++) {
       for (h = dataStart; h < dataEnd; h++) {
          if (pixelaverage / nbands < 128) {
          count = 0;
          break;          }
       }
       if (pixelaverage / nbands > 128) {
       count++;
       }
       if (count >= (dataEnd - dataStart) / 3) {
         pwvb.println("B," + wend + "," + dataStart + "," +
         (wstart) + "," + dataEnd);}
```

Since the main purpose of the segmentation stage is to segment the document image into the word level, identifying the word border is an essential task. By reading the data start position and the data end position from the file, we can get the top and bottom border of the word, whereas to get the left and right border of the word the following Java code is implemented:

The Java implementation code for obtaining the top and bottom of the word

```
FileWriter fwwb = new FileWriter("wordborder.txt");
pwwb = new PrintWriter(fwwb);
BufferedReader inputv = new BufferedReader(new
FileReader("HorizontalBord.txt"));
String linev = null;
Vector<String> line2 = new Vector<String>(2, 2);
while ((linev = inputv.readLine()) != null) {
line2.addElement(linev);
}
 for (int i = 0; i < (line2.size() - 1); i++) {
  if (line2.elementAt(i).charAt(0) == line2.elementAt
      (i + 1).charAt(0)) {
      continue;
   }
   while (line2.elementAt(i).contains("B")) {
      String[] data1 = line2.elementAt(i).split(",");
      wborderstart1 = Integer.parseInt(data1[1]);
      vborderstart1 = Integer.parseInt(data1[2]);
      wborderend1 = Integer.parseInt(data1[3]);
      vborderend1 = Integer.parseInt(data1[4]);
       i++;
       }
   while (line2.elementAt(i).contains("D")) {
    String[] data1 = line2.elementAt (i).split(",");
    wdatastart1 = Integer.parseInt(data1[1]);
    vdatastart1 = Integer.parseInt(data1[2]);
    wdataend1 = Integer.parseInt(data1[3]);
    vdataend1 = Integer.parseInt(data1[4]);
    i++;
    }
    if (wdataend1 - wborderend1 > 20 || vdataend1-
    vdatastart1<10) {
    pwwb.println("C," + (wborderend1 + 1) + "," +
     (vdatastart1) + "," + wdataend1 + "," +
    vdataend1);
    }
```

```
    }
inputv.close();
        pwwb.flush();
        pwwb.close();
```

By knowing the borders of each word the above Java code segments the document image into word level as shown in Figure 4.3.

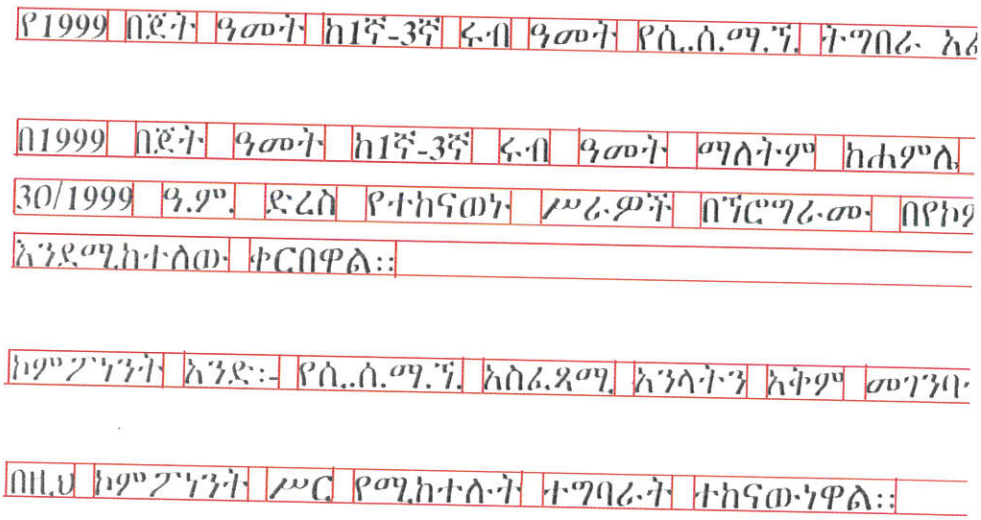


Figure 4.3 Figure showing word level segmentation

Since there is a chance of getting characters or words that have less importance for searching Amharic words, for example like the word "፣ ና ኑ ፤ ፡ ፡ አለ ሂድ " and so on, it is necessary to remove such characters from the word collection. For the purpose of removing less important words, Algorithm 4.1 that checks the length of the word has been introduced. This algorithm checks if the length of the word bounded by the word border algorithm is greater than certain threshold, if it is greater than that, it will consider the word as a proper Amharic word, otherwise it will ignore the word and it will not write its borders to the word border file. Here, the minimum length of the word is taken as 20 pixels and the minimum height of the word is taken as 10 pixels. These values are considered by using the two consecutive Amharic characters of "ነ ነ " for measuring the width and "ሙ" for measuring the height. The researcher has observed

that "ነ " is the character that has a smallest width and "ው" is a character that has the smallest height in Amharic alphabet. Most of the words formed by combining two characters in Amharic are considered has no importance in retrieving the document. Since the width of Amharic characters varies, it is safe to assume a word that is formed by two characters of the smallest width is not important for retrieval.

The algorithm for determining the minimum length of a word is:

```
If WordLength(i,j)>MinimumLength
      If WordHeight>MinimumHeight
            Write to file
```

Algorithm 4.1 Determine the minimum length and height of the word image

The Java implementation of Algorithm 4.1 is shown below:

The Java implementation code for determining the minimum length and height of the word image

```
if (wdataend1 - wborderend1> 20 && vdataend1-vdatastart1>10) {
    pwwb.println("C," + (wborderend1 + 1) + "," +(vdatastart1)
    + "," + wdataend1 + "," + vdataend1);
}
```

The outputs of the word image segmentation coordinates are put in a file with a prefix of "C" to identify the coordinates of the word image. The partial view of the output file is shown on Table 4.2.

C,84,67,165,111	C,193,67,249,111	C,277,67,444,111	C,473,67,532,111
C,562,67,761,111	C,789,67,950,111	C,85,181,254,229	C,282,181,482,229
C,512,181,595,229	C,625,181,681,229	C,709,181,876,229	C,905,181,964,229
C,993,181,1177,229	C,1207,181,1316,229	C,1345,181,1528,229	C,1557,181,1828,229
C,84,243,199,281	C,229,243,482,281	C,512,243,625,281	C,84,474,335,519
C,365,474,495,519	C,525,474,738,519	C,768,474,851,519	C,880,474,936,519
C,965,474,1131,519	C,1160,474,1312,519	C,1342,474,1390,519	C,1420,474,1476,519
C,1509,474,1609,519	C,1638,474,1722,519	C,1751,474,1920,519	C,83,536,228,574
C,257,536,470,574	C,82,590,310,634	C,340,590,529,634	C,83,823,220,868

Table 4.2 The partial view of the output file that holds the word boundaries.

The "C" is simply used to indicate the file content is a coordinate value. The numbers indicate the left, top, right and bottom coordinate values of the image word. From the above table we can easily see that the document has a word image bounded by the coordinates 84,67,165,111.

4.2.3. Query Segmentation

The system accepts the query from the users in text form and it converts the text query into an image query by the process called rendering. The image query is resized to fit the document image size. From the experiment made during the system development the researcher determines that a resizing factor 4.22 gives a good result. So, the determined factor is used for resizing the query text. However, a mechanism should be developed to determine a dynamic resizing factor.

The extracted features are represented in a vector that has many values which are either a result of the vertical bar pattern or the predefined area methods.

During Word feature extraction of individual word images the need for normalization arises so that the word representations become insensitive to variations in size, font and various degradations popularly present in document images.

The normalization process for parallel bar vertical line feature extraction uses the following techniques: First the word height is divided into four regions, which are the top, middle, bottom and lower regions. These regions are generated by dividing the height of the word by three and get the gap. Assign the region from top border up to the top border + gap position as a top region, the middle region is the area between the top border + gap and top border + 2 (gap), the bottom region is a region from the lower border of the middle region by adding the gap, the remaining region is considered as a lower region. Then each value that have a data at the top level is counted and multiplied by one to get the total top region pixel data value of the top region. With the same logic the middle, the bottom and the lower pixel values are calculated except that the multiplication factor is 2, 3 and 4, respectively.

Then, each pixel at the top level will be counted and multiplied by one, to get the total top region pixel value of the top region. With the same logic the middle, the bottom and the lower pixel values is calculated except that the multiplication factor is 2, 3 and 4, respectively.

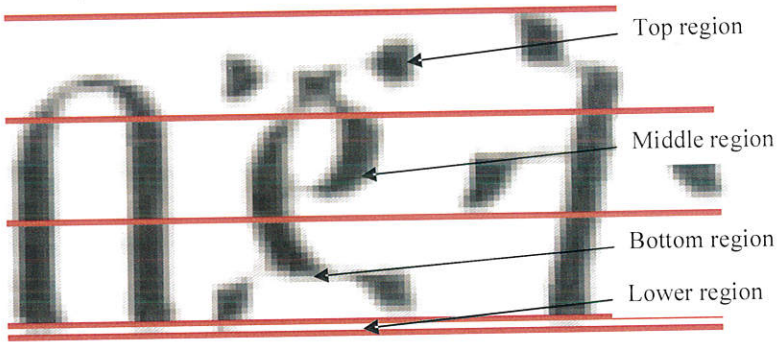


Figure 4.4 Divided regions of a word image

Figure 4.4 indicates the boundary lines detected. In this way a word can be represented in to different regions where each region will have different pixel value.

Finally, the sum of the data pixels from top, middle, bottom and lower will be divided by the sum of pixel values of the top, middle, bottom and lower values. This always gives a result between 0 and 1.

The Java Code that is used to normalize and extract the document image features is represented below. These features are then used for feature matching which will be discussed letter in this chapter.

The Java Implementation of normalizing and extracting parallel bar vertical line image word features :

```
FileWriter fwwv = new FileWriter("C:/ImageVector/"+
Vector.getName().replaceAll(".jpg", ".txt"),true);
    pwwv = new PrintWriter(fwwv);
    Vector VlinepixVal = new Vector();
    String line3=null;
    Vector<String> line4 = new Vector<String>(2, 2);
    while ((line3 = inputB.readLine()) != null) {
      line4.addElement(line3);
    }
    for (int m = 0; m < (line4.size()); m++) {
     if (line4.elementAt(m).isEmpty()) {
      m = m + 1;
     }
```

```
int boundgap = (bottomCorner - topCorner) / 3;
int topbound = topCorner + boundgap;
int midbound = topbound + boundgap;
int bottombound = midbound + boundgap;
int countTotal = 0;
int countT, countM, countB, countL = 0;
 for (w = leftCorner; w < rightCorner; w++) {
  int pixelValT = 0,  int pixelValM = 0;
  int pixelValB = 0, int pixelValL = 0;
   for (h = topCorner; h < topbound; h++) {
    for (int band = 0; band < nbands; band++) {
      pixelaverage = pixel[band] + pixelaverage;
    }
    if (pixelaverage / nbands < 128) {
     pixelValT++;
    }
   countT++;
   }
   for (h = topbound; h < midbound; h++) {
     for (int band = 0; band < nbands; band++) {
       pixelaverage = pixel[band] + pixelaverage;
     }
     if (pixelaverage / nbands < 128) {
      pixelValM++;
     }
    countM++;
    }
    for (h = midbound; h < bottombound; h++) {
     for (int band = 0; band < nbands; band++) {
     pixelaverage = pixel[band] + pixelaverage;
    }
    if (pixelaverage / nbands < 128) {
    pixelValB++;
    }
    countB++;
  }
  for (h = bottombound; h < bottomCorner; h++) {
    for (int band = 0; band < nbands; band++) {
      pixelaverage = pixel[band] + pixelaverage;
    }
    if (pixelaverage / nbands < 128) {
    pixelValL++;
    }
    countL++;
    }
countTotal = countT +(countM * 2)+(countB * 3) + (countL * 4);
float  VlineVal=(float)(pixelValL*4+pixelValB*3+pixelValM*2  +
        pixelValT) / countTotal;
VlinepixVal.addElement(VlineVal);
}
for (int k = 0; k < VlinepixVal.size(); k++) {
```

```
    pwwv.print(VlinepixVal.elementAt(k) + ",");
   }
  pwwv.println();
pwwv.flush();
pwwv.close();
}
```

To roughly extract the feature of a word image using area based feature extraction a word is divided into 144 distinct areas by segmenting the word into 9 vertical and 16 horizontal parts.

Then each pixel that has a data within the area is counted. At the same time each pixel is counted to get the total number of pixels.

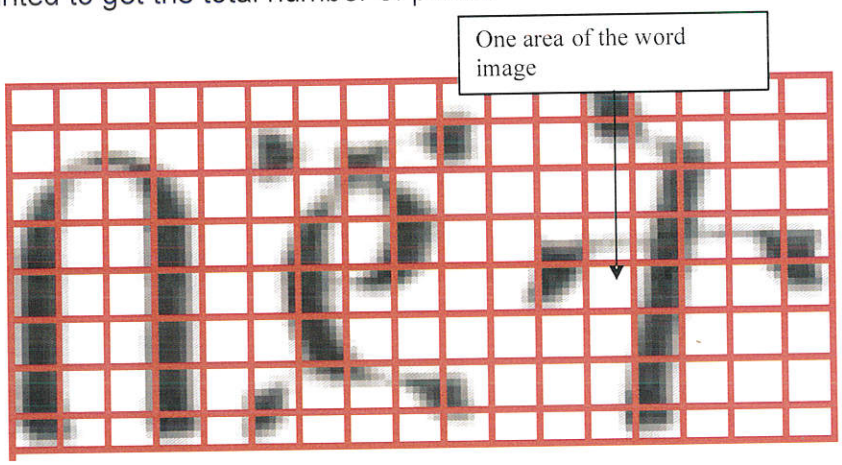


Figure 4.5 Area based divided regions of word image

Figure 4.5 indicates the areas selected. In this way a word can be represented in to different areas where each area will have different pixel value.

Finally, the sum of the data pixels in the area is divided by the sum of pixel values of the area. This always gives a result between 0 and 1.

The Java code that is used to normalize and extract the document image features is represented in the following Java code. These features are then used for feature matching.

The Java Implementation of normalizing and extracting Area based image word features :

```
 FileWriter fwwv = new FileWriter("C:/ImageVector1/"+
 Vector.getName().replaceAll(".jpg", ".txt"),true)
 pwwv = new PrintWriter(fwwv);
Vector VlinepixVal = new Vector();
int leftcord=0;
int rightcord=0;
int topcord=0;
int bottomcord=0;
int boundgapV = (rightCorner - leftCorner) / 15;
int boundgapH = (bottomCorner - topCorner) / 7;
int nbands = sm.getNumBands();
int[] pixel = new int[nbands];
int d = 0;
int wt;
int z;
Vector<String> cell = new Vector<String>(2, 2);
for (int x = leftCorner; x < rightCorner; x += boundgapV){
 if (x+boundgapV>rightCorner)
   z=rightCorner;
 else
  z=x+boundgapV;
  for (int y = topCorner; y < bottomCorner; y+= boundgapH){
   if (y+boundgapH>bottomCorner)
    wt=bottomCorner;
   else
     wt=y+boundgapH;
     cell.addElement("Cell " + d+" "+x+" "+z+" "+y+"
     "+wt);//Integer.toString(d));
    d++;
   }
 }
 for (int i = 0; i < cell.size(); i++) {
  String[] data = cell.elementAt(i).split(" ");
  leftcord = Integer.parseInt(data[2]);
  rightcord=Integer.parseInt(data[3]);
  topcord=Integer.parseInt(data[4]);
  bottomcord=Integer.parseInt(data[5]);
  int pixelValTVH = 0;
  int countVH=0;
  RandomIter iterator = RandomIterFactory.create(pi, null);
  for (int wQ = leftcord; wQ <rightcord; wQ++) {
     for ( int hQ = topcord; hQ <bottomcord; hQ++) {
      iterator.getPixel(wQ, hQ, pixel);//getPixel(wQ, hQ,
      pixelQ);
      pixelaverage = 0;
        for (int band = 0; band < nbands; band++) {
         pixelaverage = pixel[band] + pixelaverage;
       }
```

```
            if (pixelaverage / nbands < 128) {
              pixelValTVH++;
            }
          countVH++;
          }
        }
        float VlineValV = (float) (pixelValTVH) / countVH;
        VlinepixVal.addElement(VlineValV);
        pixelValTVH = 0;
        countVH=0;
      }
      for (int k = 0; k < VlinepixVal.size() - 1; k++) {
       pwwv.print(VlinepixVal.elementAt(k) + ",");
      }
  pwwv.println();
  pwwv.flush();
  pwwv.close();
```

A file is used to store the vector values of each word image, each file can be treated as a vector, so called document vector

4.4. Similarity Measure

The Similarity measures used in this research are cosine similarity measure and Euclidean similarity measure. For the purpose getting the similarity results between 0 and 1, the Euclidean similarity is normalized as shown in Equation 3.1.

Based on Equation 3.1 and Equation 2.1 the following Java codes are implemented to measure similarities between image word and query word.

The Java implementation code for the Euclidean similarity measure is

```
BufferedReader inputvQ = new BufferedReader(new FileReader
  ("wordvalueQ.txt"));
BufferedReader inputvI = new BufferedReader(new FileReader
  (VectorValue));
String linevQ = null, linevI = null;
double SIres=0,  double SQres=0,  double difre=0;
String[] dataQ = null;
String[] dataI = null;
Vector<String> lineQ5 = new Vector<String>(2, 2);
```

```
Vector<String> line2 = new Vector<String>(2, 2);
while ((linevQ = inputvQ.readLine()) != null) {
  lineQ5.addElement(linevQ);
}
for (int k = 0; k < (lineQ5.size()); ++k) {
  dataQ = lineQ5.elementAt(k).split(",");//elementAt(k);
}
while ((linevI = inputvI.readLine()) != null) {
  line2.addElement(linevI);
}
for (int i = 0; i < (line2.size()); i++) {
 dataI = line2.elementAt(i).split(",");
  if (dataQ.length <= dataI.length) {
    for (int h = 0; h < dataQ.length; ++h) {
     Qres = Double.parseDouble(dataQ[h]);
     SQres=SQres+Qres;
     Ires = Double.parseDouble(dataI[h]);
     difre=(Math.pow((Qres-Ires),2))+difre;
      SIres=SIres+Ires;
     }
   } if (dataQ.length > dataI.length) {
   for (int h = 0; h < dataI.length; ++h) {
    Ires = Double.parseDouble(dataI[h]);
    Qres = Double.parseDouble(dataQ[h]);
ci
    SQres=SQres+Qres;
    difre=(Math.pow((Qres-Ires),2))+difre;
   }
 }
 double similarityE=0;
 similarityE=(Math.sqrt(difre))/(SQres+SIres);
                  }
```

The Java implementation code for the Cosine similarity measure is

```
BufferedReader inputvQ = new BufferedReader (new FileReader
("wordvalueQ.txt"));
BufferedReader inputvI = new BufferedReader (new FileReader
 (VectorValue));
String linevQ = null,   String linevI = null;
Vector<String> lineQ5 = new Vector<String>(2, 2);
Vector<String> line2 = new Vector<String>(2, 2);
while ((linevQ = inputvQ.readLine()) != null) {
  lineQ5.addElement(linevQ);
}
for (int k = 0; k < (lineQ5.size()); ++k) {
 dataQ = lineQ5.elementAt(k).split(",");
}
while ((linevI = inputvI.readLine()) != null) {
  line2.addElement(linevI);
}
```

```
for (int i = 0; i < (line2.size()); i++) {
  dataI = line2.elementAt(i).split(",");
  if (dataQ.length <= dataI.length) {
    for (int h = 0; h < dataQ.length; ++h) {
     Qres = Double.parseDouble(dataQ[h]);
     Ires = Double.parseDouble(dataI[h]);
     dotproduct = (Qres * Ires) + dotproduct;
     sqrData1 = (Qres * Qres) + sqrData1;
     sqrData2 = (Ires * Ires) + sqrData2;
    }
   }
  if (dataQ.length > dataI.length) {
    for (int h = 0; h < dataI.length; ++h) {
     Ires = Double.parseDouble(dataI[h]);
     Qres = Double.parseDouble(dataQ[h]);
     dotproduct = (Qres * Ires) + dotproduct;
     sqrData1 = (Qres * Qres) + sqrData1;
     sqrData2 = (Ires * Ires) + sqrData2;
  }
 }
 double similarityC = 0;
 similarityC = dotproduct / Math.sqrt(sqrData1 * sqrData2);
```

4.5. Performance Evaluation

To verify the validity of the approach proposed in this paper for searching the user specified Amharic word in Amharic Document images, the experiments have been carried out on 121 scanned documents that are selected from recent Amharic documents encoded in the computer. The documents collected fall into three categories that are proclamations, news items and guidelines. The main font used in the document is Power geez Unicode 1. The total number of pages and number of Amharic words considered from the 121 documents are 483 and 109, 238 respectively. For testing the system 28 word queries were selected by a language expert from the collection of the 121 documents.

For this research a program is written in Java for image segmentation, feature extraction and feature matching. The efficiency of each algorithm is tested separately

and then the segmentation, feature extraction and feature matching algorithms are integrated and tested on the sample data to measure the performance of the system.

The computer system used for implementing the system is x64-based PC that has Intel(R) Pentium(R) Dual CPU T3200 @ 2.00GHz, 2000 Mhz, 2 Core(s), 2 Logical Processor(s), Installed Physical Memory (RAM) 4.00 GB and Microsoft® Windows Vista™ Home Premium Operating System. For the purpose of storing a scanned document and the vector representation of the document large hard disk space is needed. For a half a page document that holds 49 words 119kb is needed to store the scanned document and 68kb is needed to store the vector representation. By considering the minimum one can get a hard disk space needed by multiplying the number of pages scanned by the sum of the scanned and vector file size. For the purpose of using large collection of document images the system requires a huge storing space.

4.6. Testing Cases

Three different cases of testing are made by combining the feature extraction methods and the similarity measures used. As described in chapter three two feature extraction methods and two similarity measure algorithms are implemented. At first the test is done only by using a parallel bar vertical line feature extraction and the Euclidean similarity measure. Then the researcher wants to improve the performance of the system and use the same feature extraction method together with the cosine similarity measure. However the performance of the system does not improve as expected. Then the researcher again tries to increase the performance by using area based feature extraction. The three cases of testing are:

1. parallel bar vertical line feature extraction method with Euclidean similarity measure
2. parallel bar vertical line feature extraction method with Cosine similarity measure
3. Area Based feature extraction method with Euclidean similarity measure

From the testing made on the document image retrieval system, four thresholds are selected and evaluation measure is made based on the selected thresholds for all cases. The thresholds chosen for the Euclidean measure of parallel bar vertical line feature extraction are 0.024, 0.027, 0.030 and 0.033. The thresholds chosen for the Cosine similarity measure of parallel bar vertical line feature extraction are 0.92, 0.94, 0.96 and 0.98. The thresholds chosen for the Euclidean measure of Area Based feature extraction are 0.042, 0.043, 0.044 and 0.045.

Based on these thresholds the evaluation measure is made whether the word image is similar to the query provided. The retrieved documents for Euclidean measure of parallel bar vertical line feature extraction based on the query provided are displayed in Table 4.3

Query Text	Documents that hold the Query Text	Threshold=0.033			Threshold=0.030			Threshold=0.027			Threshold=0.024		
		Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved
አዶላ	1	33	1	32	17	1	16	5	1	4	2	1	1
ዓመት	28	37	22	15	21	17	4	11	10	1	7	7	0
አዋጅ	72	65	56	9	52	51	1	50	50	0	33	33	0
የኦሮሚያ	74	80	63	17	45	45	0	34	34	0	14	14	0
መንግሥት	68	103	65	38	74	58	16	52	49	3	31	30	1
አባዱላ	51	18	12	6	8	4	4	2	1	1	1	1	0
ፊንፊኔ	57	39	28	11	19	17	2	9	9	0	5	5	0
ከተማ	20	15	10	5	10	8	2	9	7	2	4	4	0
ሲሚንቶ	1	20	1	19	3	1	2	3	1	2	2	1	1
መመሪያ	44	121	44	77	102	44	58	62	35	27	30	23	7
አዲተር	10	48	6	42	25	3	22	11	1	10	2	0	2
ባለሙያዎች	10	106	10	96	72	7	65	22	6	16	16	4	12
አንቀጽ	65	39	35	4	26	25	1	16	16	0	8	8	0
ዋልታ	4	11	4	7	4	3	1	3	2	1	2	2	0
ገጠር	15	4	4	0	4	4	0	4	4	0	4	4	0
አገር	7	19	2	17	7	2	5	1	1	0	1	1	0
ኢትዮጵያ	13	48	10	38	15	6	9	6	5	1	5	4	1
መስተዳድር	44	78	41	37	42	28	14	26	18	8	15	12	3
ሚኒስቴር	5	21	1	20	6	1	5	2	1	1	1	1	0
ኢኮኖሚ	25	47	20	27	17	9	8	7	6	1	3	3	0
ማሻሻያ	38	32	19	13	19	14	5	10	8	2	9	8	1
ትምህርት	8	24	7	17	13	6	7	10	6	4	5	5	0
ሠራተኞች	15	42	12	30	18	7	11	4	2	2	3	2	1
በአዲስ	5	34	4	30	15	4	11	4	1	3	1	1	0
ዓላማ	23	15	10	5	10	9	1	6	6	0	4	4	0
ውስጥ	72	26	26	0	13	13	0	6	6	0	4	4	0
መሣሪያ	15	97	15	82	70	15	55	36	13	23	17	7	10
ክፍል	28	20	13	7	10	10	0	7	7	0	6	6	0

Table 4.3: Table showing retrieved documents based on Euclidean Similarity measure of parallel bar vertical line feature extraction

The retrieved documents for Cosine measure of parallel bar vertical line feature extraction based on the query provided are displayed in Table 4.4

Query Text	Documents that hold the Query Text	Threshold=0.92			Threshold=0.94			Threshold=0.96			Threshold=0.98		
		Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved
አዶላ	1	42	1	41	32	1	31	7	0	7	1	0	1
ዓመት	28	23	20	3	13	10	3	3	3	0	1	1	0
አዋጅ	72	67	62	5	58	54	4	40	38	2	6	6	0
የኦሮሚያ	74	12	12	0	10	10	0	6	6	0	1	1	0
መንግሥት	68	103	65	38	74	58	16	52	49	3	31	30	1
አባዱላ	52	40	27	13	32	24	8	7	6	1	1	1	0
ባንክ	2	10	1	9	6	0	6	2	0	2	1	0	1
ፊንፊኔ	57	78	54	24	66	48	18	43	30	13	4	2	2
ከተማ	20	45	17	28	29	16	13	17	10	7	3	2	1
ሲሚንቶ	1	22	1	21	14	1	13	4	1	3	2	0	2
መመሪያ	44	80	43	37	70	42	28	61	35	26	11	10	1
አዲተር	10	50	9	41	40	6	34	28	2	26	5	1	4
ባለሙያዎች	10	39	9	30	28	8	20	15	5	10	6	2	4
አንቀጽ	65	53	48	5	42	37	5	21	19	2	2	2	0
ገጠር	15	15	12	3	10	10	0	7	7	0	3	3	0
አገር	7	46	6	40	31	4	27	7	2	5	2	1	1
ኢትዮጵያ	13	38	9	29	29	8	21	13	4	9	1	0	1
መስተዳድር	44	80	42	38	68	40	28	59	34	25	9	6	3
ሚኒስቴር	5	73	3	70	63	2	61	52	2	50	24	0	0
ኢኮኖሚ	25	49	19	30	35	16	19	11	6	5	1	0	1
ማሻሻያ	38	70	32	38	63	30	33	51	24	27	23	11	12
ትምህርት	8	18	7	11	13	4	9	9	4	5	2	2	0
ሠራተኞች	15	56	14	42	33	10	23	9	4	5	1	0	1
በአዲስ	5	56	4	52	31	2	29	9	2	7	2	0	2
ዓለማ	23	14	9	5	10	7	3	8	5	3	6	5	1
ውስጥ	72	68	54	14	32	27	5	15	13	2	3	3	0
መሣሪያ	15	80	15	65	69	15	54	59	14	45	11	4	7
ክፍል	28	56	23	33	34	18	16	14	10	4	1	1	0

Table 4.4: Table showing retrieved documents based on Cosine Similarity of Parallel bar vertical line feature extraction.

The retrieved documents for Euclidean measure of Area Based feature extraction based on the query provided are displayed in Table 4.5

Quarry Text	Documents that hold the Quarry Text	Threshold=0.045			Threshold=0.044			Threshold=0.043			Threshold=0.042		
		Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Documents Retrieved	Relevant Documents Retrieved	Non Relevant Documents Retrieved
አዶላ	1	64		64	46		46	23	0	23	11		11
ዓመት፡	28	5	2	3	2	1	1			0			0
አዋጅ	72	102	71	31	89	69	20	73	63	10	56	48	8
የኦሮሚያ	74	111	73	38	107	71	36	90	64	26	80	62	18
መንግሥት፡	68	57	42	15	34	33	1	19	18	1	9	8	1
አባዱላ	51			0			0			0	102	49	53
ባንክ	4	11	4	7	4	3	1	3	2	1	2	2	0
ፊንፊኔ	57	112	56	56	105	53	52	92	53	39	78	44	34
ከተማ	20	85	20	65	73	15	58	54	14	40	39	10	29
ሲሚንቶ	1	104	1	103	90	1	89	74	1	73	59	0	59
መመሪያ	44	19	10	9	8	5	3	7	4	3	2	1	1
አዲተር	10	113	10	103	100	10	90	91	10	81	73	8	65
ባለሙያዎች	10	16	3	13	11	3	8	4	1	3	2	1	1
አንቀጽ	65	119	65	54	116	65	51	110	65	45	97	61	36
ገጠር	15	120	15	105	115	15	100	111	14	97	105	14	91
አገር	7	108	7	101	95	5	90	81	5	76	63	5	58
ኢትዮጵያ	13	117	13	104	112	13	99	99	13	86	81	11	70
መስተዳድር	44	121	44	77	119	44	75	117	44	73	112	42	70
ሚኒስቴር	5	121	5	116	120	4	116	118	4	114	110	4	106
ኢኮኖሚ	25	112	25	87	103	25	78	90	24	66	69	21	48
ማሻሻያ	38	104	38	66	85	35	50	64	27	37	47	25	22
ትምህርት	8	102	8	94	84	8	76	65	8	57	48	8	40
ሠራተኞች	15	93	15	78	82	15	67	67	14	53	51	13	38
በአዲስ	5	121	5	116	115	4	111	107	4	103	96	3	93
ዓላማ	23	68	21	47	60	20	40	42	15	27	21	11	10
ውስጥ	72	93	64	29	83	60	23	67	48	19	49	38	11
መሣሪያ	15	60	13	47	48	12	36	34	9	25	19	4	15
ክፍል	28	114	28	86	105	28	77	97	27	70	81	24	57

Table 4.5: Table showing retrieved documents based on Euclidean Similarity of Area based feature extraction.

To further evaluate the performance of the proposed model, the precision and recall of the Parallel Bar vertical line feature extraction for the Euclidean similarity measure testing results are measured and presented on Table 4.6

Quarry Text	Documents that hold the Quarry Text	Threshold=0.033				Threshold=0.030				Threshold=0.027				Threshold=0.024			
		Relevant Documents Retrieved	Non Relevant Documents Retrieved	Recall %	Precision %	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Recall %	Precision %	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Recall %	Precision %	Relevant Documents Retrieved	Non Relevant Documents Retrieved	Recall %	Precision %
አዶላ	1	1	32	100.00	3.03	1	16	100.00	5.88	1	4	100.00	20.00	1	1	100.00	50.00
ዓመት	28	22	15	78.57	59.46	17	4	60.71	80.95	10	1	35.71	90.91	7	0	25.00	100.00
አዋጅ	72	56	9	77.78	86.15	51	1	70.83	98.08	50	0	69.44	100.00	33	0	45.83	100.00
የኦሮሚያ	74	63	17	85.14	78.75	45	0	60.81	100.00	34	0	45.95	100.00	14	0	18.92	100.00
መንግሥት	68	65	38	95.59	63.11	58	16	85.29	78.38	49	3	72.06	94.23	30	1	44.12	96.77
አባዱላ	51	12	6	23.53	66.67	4	4	7.84	50.00	1	1	1.96	50.00	1	0	1.96	100.00
ባንክ	4	4	7	100.00	36.36	3	1	75.00	75.00	2	1	50.00	66.67	2	0	50.00	100.00
ፊንፊኔ	57	28	11	49.12	71.79	17	2	29.82	89.47	9	0	15.79	100.00	5	0	8.77	100.00
ከተማ	20	10	5	50.00	66.67	8	2	40.00	80.00	7	2	35.00	77.78	4	0	20.00	100.00
ሲሚንቶ	1	1	19	100.00	5.00	1	2	100.00	33.33	1	2	100.00	33.33	1	1	100.00	50.00
መመሪያ	44	44	77	100.00	36.36	44	58	100.00	43.14	35	27	79.55	56.45	23	7	52.27	76.67
አዲተር	10	6	42	60.00	12.50	3	22	30.00	12.00	1	10	10.00	9.09	0	2	0.00	0.00
ባለመ.ያዎች	10	10	96	100.00	9.43	7	65	70.00	9.72	6	16	60.00	27.27	4	12	40.00	25.00
አንቀጽ	65	35	4	53.85	89.74	25	1	38.46	96.15	16	0	24.62	100.00	8	0	12.31	100.00
ገጠር	15	4	0	26.67	100.00	4	0	26.67	100.00	4	0	26.67	100.00	4	0	26.67	100.00
አገር	7	2	17	28.57	10.53	2	5	28.57	28.57	1	0	14.29	100.00	1	0	14.29	100.00
ኢትዮጵያ	13	10	38	76.92	20.83	6	9	46.15	40.00	5	1	38.46	83.33	4	1	30.77	80.00
መስተዳድር	44	41	37	93.18	52.56	28	14	63.64	66.67	18	8	40.91	69.23	12	3	27.27	80.00
ሚኒስቴር	5	1	20	20.00	4.76	1	5	20.00	16.67	1	1	20.00	50.00	1	0	20.00	100.00
ኢኮኖሚ	25	20	27	80.00	42.55	9	8	36.00	52.94	6	1	24.00	85.71	3	0	12.00	100.00
ማሻሻያ	38	19	13	50.00	59.38	14	5	36.84	73.68	8	2	21.05	80.00	8	1	21.05	88.89
ትምህርት	8	7	17	87.50	29.17	6	7	75.00	46.15	6	4	75.00	60.00	5	0	62.50	100.00
ሠራተኞች	15	12	30	80.00	28.57	7	11	46.67	38.89	2	2	13.33	50.00	2	1	13.33	66.67
በአዲስ	5	4	30	80.00	11.76	4	11	80.00	26.67	1	3	20.00	25.00	1	0	20.00	100.00
ዓለማ	23	10	5	43.48	66.67	9	1	39.13	90.00	6	0	26.09	100.00	4	0	17.39	100.00
ውስጥ	72	26	0	36.11	100.00	13	0	18.06	100.00	6	0	8.33	100.00	4	0	5.56	100.00
መሣሪያ	15	15	82	100.00	15.46	15	55	100.00	21.43	13	23	86.67	36.11	7	10	46.67	41.18
ክፍል	28	13	7	46.43	65.00	10	0	35.71	100.00	7	0	25.00	100.00	6	0	21.43	100.00
Average				68.66	46.15			54.33	59.06			40.71	70.18			30.65	84.11

Table 4.6 : Performance evaluation of Euclidean distance similarity for Parallel bar vertical line feature extraction.

The composite F-measure of recall and precision are presented in Table 4.7 for the Euclidean similarity measure of parallel bar vertical line feature extraction.

Quarry Text	Threshold=0.033			Threshold=0.030			Threshold=0.027			Threshold=0.024		
	Recall %	Precision %	F %	Recall %	Precision %	F %	Recall %	Precision %	F%	Recall %	Precision %	F%
አዶላ	100.00	3.03	5.88	100.00	5.88	11.11	100.00	20.00	33.33	100.00	50.00	66.67
ዓመት	78.57	59.46	67.69	60.71	80.95	69.39	35.71	90.91	51.28	25.00	100.00	40.00
አዋጅ	77.78	86.15	81.75	70.83	98.08	82.26	69.44	100.00	81.97	45.83	100.00	62.86
የኦሮሚያ	85.14	78.75	81.82	60.81	100.00	75.63	45.95	100.00	62.96	18.92	100.00	31.82
መንግሥት	95.59	63.11	76.02	85.29	78.38	81.69	72.06	94.23	81.67	44.12	96.77	60.61
አባዱላ	23.53	66.67	34.78	7.84	50.00	13.56	1.96	50.00	3.77	1.96	100.00	3.85
ፊንፊኔ	49.12	71.79	58.33	29.82	89.47	44.74	15.79	100.00	27.27	8.77	100.00	16.13
ከተማ	50.00	66.67	57.14	40.00	80.00	53.33	35.00	77.78	48.28	20.00	100.00	33.33
ሲሚንቶ	100.00	5.00	9.52	100.00	33.33	50.00	100.00	33.33	50.00	100.00	50.00	66.67
መመሪያ	100.00	36.36	53.33	100.00	43.14	60.27	79.55	56.45	66.04	52.27	76.67	62.16
አዲተር	60.00	12.50	20.69	30.00	12.00	17.14	10.00	9.09	9.52	0.00	0.00	
ባለሙያዎች	100.00	9.43	17.24	70.00	9.72	17.07	60.00	27.27	37.50	40.00	25.00	30.77
አንቀጽ	53.85	89.74	67.31	38.46	96.15	54.95	24.62	100.00	39.51	12.31	100.00	21.92
ዋልታ	100.00	36.36	53.33	75.00	75.00	75.00	50.00	66.67	57.14	50.00	100.00	66.67
ገጠር	26.67	100.00	42.11	26.67	100.00	42.11	26.67	100.00	42.11	26.67	100.00	42.11
አገር	28.57	10.53	15.38	28.57	28.57	28.57	14.29	100.00	25.00	14.29	100.00	25.00
ኢትዮጵያ	76.92	20.83	32.79	46.15	40.00	42.86	38.46	83.33	52.63	30.77	80.00	44.44
መስተዳድር	93.18	52.56	67.21	63.64	66.67	65.12	40.91	69.23	51.43	27.27	80.00	40.68
ሚኒስቴር	20.00	4.76	7.69	20.00	16.67	18.18	20.00	50.00	28.57	20.00	100.00	33.33
ኢኮኖሚ	80.00	42.55	55.56	36.00	52.94	42.86	24.00	85.71	37.50	12.00	100.00	21.43
ማሻሻያ	50.00	59.38	54.29	36.84	73.68	49.12	21.05	80.00	33.33	21.05	88.89	34.04
ትምህርት	87.50	29.17	43.75	75.00	46.15	57.14	75.00	60.00	66.67	62.50	100.00	76.92
ሠራተኞች	80.00	28.57	42.11	46.67	38.89	42.42	13.33	50.00	21.05	13.33	66.67	22.22
በአዲስ	80.00	11.76	20.51	80.00	26.67	40.00	20.00	25.00	22.22	20.00	100.00	33.33
ዓላማ	43.48	66.67	52.63	39.13	90.00	54.55	26.09	100.00	41.38	17.39	100.00	29.63
ውስጥ	36.11	100.00	53.06	18.06	100.00	30.59	8.33	100.00	15.38	5.56	100.00	10.53
መሣሪያ	100.00	15.46	26.79	100.00	21.43	35.29	86.67	36.11	50.98	46.67	41.18	43.75
ክፍል	46.43	65.00	54.17	35.71	100.00	52.63	25.00	100.00	40.00	21.43	100.00	35.29
Average	68.66	46.15	55.20	54.33	59.06	56.60	40.71	70.18	51.53	30.65	84.11	44.92

Table 4.7: F-Measure Euclidean Similarity for parallel bar vertical line feature extraction.

The composite F-measure of recall and precision are presented in table 4.9 for Cosine similarity measure of parallel bar vertical line feature extraction.

Query Text	Threshold=0.92			Threshold=0.94			Threshold=0.96			Threshold=0.98		
	Recall %	Precision %	F%	Recall %	Precision %	F%	Recall %	Precision %	F%	Recall%	Precision %	F%
አዶላ	100.00	2.38	4.65	100.00	3.13	6.06	0.00	0.00		0.00	0.00	
ዓመት	71.43	86.96	78.43	35.71	76.92	48.78	10.71	100.00	19.35	3.57	100.00	6.90
አዋጅ	86.11	92.54	89.21	75.00	93.10	83.08	52.78	95.00	67.86	8.33	100.00	15.38
የኦሮሚያ	16.22	100.00	27.91	13.51	100.00	23.81	8.11	100.00	15.00	1.35	100.00	2.67
መንግሥት	95.59	63.11	76.02	85.29	78.38	81.69	72.06	94.23	81.67	44.12	96.77	60.61
አባዱላ	51.92	67.50	58.70	46.15	75.00	57.14	11.54	85.71	20.34	1.92	100.00	3.77
ባንክ	50.00	10.00	16.67	0.00	0.00		0.00	0.00		0.00	0.00	
ሬንፊኔ	94.74	69.23	80.00	84.21	72.73	78.05	52.63	69.77	60.00	3.51	50.00	6.56
ከተማ	85.00	37.78	52.31	80.00	55.17	65.31	50.00	58.82	54.05	10.00	66.67	17.39
ሲሚንቶ	100.00	4.55	8.70	100.00	7.14	13.33	100.00	25.00	40.00	0.00	0.00	
መመሪያ	97.73	53.75	69.35	95.45	60.00	73.68	79.55	57.38	66.67	22.73	90.91	36.36
ኦዲተር	90.00	18.00	30.00	60.00	15.00	24.00	20.00	7.14	10.53	10.00	20.00	13.33
ባለሙያዎች	90.00	23.08	36.73	80.00	28.57	42.11	50.00	33.33	40.00	20.00	33.33	25.00
አንቀጽ	73.85	90.57	81.36	56.92	88.10	69.16	29.23	90.48	44.19	3.08	100.00	5.97
ገጠር	80.00	80.00	80.00	66.67	100.00	80.00	46.67	100.00	63.64	20.00	100.00	33.33
አገር	85.71	13.04	22.64	57.14	12.90	21.05	28.57	28.57	28.57	14.29	50.00	22.22
ኢትዮጵያ	69.23	23.68	35.29	61.54	27.59	38.10	30.77	30.77	30.77	0.00	0.00	
መስተዳድር	95.45	52.50	67.74	90.91	58.82	71.43	77.27	57.63	66.02	13.64	66.67	22.64
ሚኒስቴር	60.00	4.11	7.69	40.00	3.17	5.88	40.00	3.85	7.02	0.00	0.00	
ኢኮኖሚ	76.00	38.78	51.35	64.00	45.71	53.33	24.00	54.55	33.33	0.00	0.00	
ማሻሻያ	84.21	45.71	59.26	78.95	47.62	59.41	63.16	47.06	53.93	28.95	47.83	36.07
ትምህርት	87.50	38.89	53.85	50.00	30.77	38.10	50.00	44.44	47.06	25.00	100.00	40.00
ሠራተኞች	93.33	25.00	39.44	66.67	30.30	41.67	26.67	44.44	33.33	0.00	0.00	
በአዲስ	80.00	7.14	13.11	40.00	6.45	11.11	40.00	22.22	28.57	0.00	0.00	
ዓላማ	39.13	64.29	48.65	30.43	70.00	42.42	21.74	62.50	32.26	21.74	83.33	34.48
ውስጥ	75.00	79.41	77.14	37.50	84.38	51.92	18.06	86.67	29.89	4.17	100.00	8.00
መሣሪያ	100.00	18.75	31.58	100.00	21.74	35.71	93.33	23.73	37.84	26.67	36.36	30.77
ክፍል	82.14	41.07	54.76	64.29	52.94	58.06	35.71	71.43	47.62	3.57	100.00	6.90
Average	78.94	44.71	57.08	62.87	48.06	54.48	40.45	53.38	46.02	10.24	55.07	17.26

Table 4.9: F-Measure Cosine similarity for parallel bar vertical line feature extraction.

From the testing it can be seen that a threshold is set to decide whether word image is similar to the query word. The threshold is set at 0.024, 0.027, 0.030 and 0.033 for Euclidian similarity of parallel bar vertical line feature extraction; 0.92, 0.94, 0.96 and 0.98 for cosine similarity of parallel bar vertical line feature extraction. 0.042, 0.043, 0.044, and 0.045 are set for Euclidean similarity of Area based feature extraction. The comparison of the recalls and precisions is summarized in Tables 4.4, 4.6 and 4.8 respectively.

From the experiment, result obtained by assuming that Microsoft Vista text search retrieves documents that has the same word as the query, it can be seen that the average similarity measure obtained for each threshold is low.

The average recall and precision for Euclidean similarity of the parallel bar vertical line feature extraction are 68.66% and 46.15% for threshold 0.033, 54.33% and 59.06% for threshold 0.030, 40.71% and 70.18% for threshold 0.027 and 30.65% and 84.11% for threshold 0.024.

For the cosine similarity measure of the parallel bar vertical line feature extraction, the average recall and precision values are 78.94% and 44.71% for threshold 0.92, 62.87% and 48.06% for threshold 0.94, 40.45% and 53.38% for threshold 0.96 and 10.24% and 55.07% for threshold 0.98.

Whereas for the Euclidean measure of area based feature extraction the average recall and precisions are 84.59% and 29.40% 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 35.00% for threshold 0.042.

As a threshold value decreases for Euclidean measure the precision increase while the recall decreases. In case of cosine similarity measure the precision increases when the threshold value increases while recall decreases.

Here it is easy to notice that, the results for some query words are very good while the results for some query words are poor, this may be the result of some noise and the assumption made, that the Microsoft Vista search is perfect. In general the proposed method can provide a good recall and precision for retrieving document images.

Besides the precision and recall, the composite F measure is used to evaluate the performance of the proposed system. The results are shown in table 4.7, 4.9 and 4.11.

It can be seen that the average F-measure is 55.20% for a threshold of 0.033, 56.60% for a threshold of 0.030, 51.53% for a threshold of 0.027 and 44.92% for a threshold of 0.024 for Euclidean measure of parallel bar vertical line feature extraction.

The average F-measure is 57.08% for a threshold of 0.92, 54.48% for a threshold of 0.94, 46.02% for a threshold of 0.96 and 17.26% for threshold of 0.98 for cosine measure of parallel bar vertical line feature extraction.

For area based Euclidean similarity measure the average F-measures are 43.64% for 0.045 threshold, 46.07% for 0.044 threshold, 45.24% for 0.043 threshold and 43.98% for 0.042 threshold.

If the purpose is retrieving only relevant document images, then the threshold 0.024 should be used for Euclidean similarity measure of parallel bar vertical line feature extraction, 0.98 threshold should be used for cosine similarity measure of the parallel bar vertical line feature extraction. Whereas for Area based feature extraction of the Euclidean measure 0.042 should be used. On the contrary, if the goal is to retrieve as

many documents as possible then threshold value 0.033 for Euclidean measure of the parallel bar vertical line feature extraction, 0.92 for cosine measure of parallel bar vertical line feature extraction, and for Euclidean measure of area based feature extraction 0.045 can be used.

It can be seen on the Tables 4.7, 4.9 and 4.11 that the highest average F value is 56.60%, 57.08% and 46.07% for Euclidean measures of parallel bar vertical line feature extraction, cosine measures of parallel bar vertical line feature extraction and Euclidean measure of area based feature extraction for thresholds of 0.030, 0.92 and 0.044 respectively This indicates that the threshold 0.030 of the Euclidean measure of the parallel bar vertical line feature extraction, 0.94 of the cosine measure parallel bar vertical line feature extraction and 0.044 of the Euclidean measure of area based feature extraction are better threshold values that can give a good recall and precision.

Comparing the two feature extraction methods that are parallel bar vertical line feature extraction and area based feature extraction, it can be easily concluded that the parallel bar feature extraction method gives a better recall, precision and F-values. With the same analogy of the experiment made, the Euclidean measure gives a better result than that of the cosine similarity measure.

Since the Euclidean similarity measure of the parallel bar vertical line feature extraction gives a better result and threshold 0.30 is chosen as the good threshold value, the Mean Average Precision is calculated for the threshold value of 0.30 of the Euclidean similarity measure of the parallel bar vertical line extraction method.

According to iProspect (2008) 68% of search engine users click a search result with in the first page of the result and most web search engines display 10 results per page, here the number of k retrieved documents that the users are interested are believed to

be 10 and the Precision@ 10 calculated for the system assumes only on the first 10 results of the retrieved documents.

The result for the precision in the 10 first documents (for each query and the average over all the queries) is shown in Table 4.12

Query Text	Mean Average Precision
አዶላ	17%
መመሪያ	98%
ባንክ	100%
ፊንፊኔ	79%
አባዱላ	77%
መንግሥት	100%
የኦሮሚያ	100%
አዋጅ	100%
ዓመት	95%
ገጠር	100%
አገር	23%
መስተዳድር	79%
አንቀፅ	100%
ባለሙያዎች	20%
አዲተር	77%
ከተማ	97%
ሲሚንቶ	100%
ኢትዮጵያ	81%
ሚኒስቴር	100%
ኢኮኖሚ	67%
ማሻሻያ	90%
ትምህርት	93%
ሠራተኞች	69%
በአዲስ	24%
ዓላማ	100%
ውስጥ	100%
መሣሪያ	100%
ክፍል	100%
Average	82%

Table 4.12 Performance Measure Mean Average Precision

As it can be seen in Table 4.12, from the 28 queries 12 of them have a 100 percent R-Precision result and 4 queries give a R-Precision result of less than 30 percent, with a

minimum R-Precision of 17 percent for the query word "አዶላ" which only occurs in one document image with a frequency of 1. The average R-Precision result calculated from the 28 queries is 82 percent. Hence, it can be concluded that the system is effective in retrieving relevant documents.

During experimentation the average processing time for preparing a one-page document image into vector form took 8 seconds for parallel bar vertical line feature extraction and 10 seconds for area based feature extraction. While for searching word images from the prepared vector took an average time of less than a second for a page. For comparison purpose the processing time for two collections of document images is shown in table 4.13.

Feature Extraction	Number of Documents	Number of Pages	Number of words	Processing Time	
				Document Preparation	Searching
Vertical Bar Based	74	246	54198	28 minutes 32 seconds	16 seconds
	121	483	109238	59 minutes 53 seconds	22 seconds
Area Based	74	246	54198	49 minutes 12 seconds	18 seconds
	121	483	109238	68 minutes 4 seconds	27 seconds

Table 4.13The processing time

CHAPTER FIVE

5. CONCLUSION AND RECOMMENDATION

5.1. Conclusion

In this research we propose an approach to search the user specified words from Amharic document images, without the requirement of character recognition. After segmenting document images to a word level, features, vertical bar patterns and area based features are extracted from word images and form the document vector. Document similarity is calculated by finding the minimum distance for the case of Euclidian similarity measure and a scalar product of the two vectors for Cosine similarity. Experiment using 121 documents that have 483 number of pages and 109, 238 number of words is conducted. The experimental results confirmed the validity of the model for retrieving relevant document images from the collection of document images. The proposed system is suitable for retrieving documents in language with no OCR, including Amharic.

The objective of this research is to design a system for the retrieval of Amharic document images in different organizations that have a huge collection of hard copy documents.

The Strength of this research

- During segmentation this system implements a way of considering character spaces and word space to differentiate one word with the other.
- The removal of punctuation marks and stop words from the system is implemented to some extent by considering the length and height of the word.

- The proposed system incorporates a suffix removal mechanism at the time of feature matching by considering the length of the query and image words. This is implemented in such a way that, the system first considers the words length and make a comparison on the bases of the shortest word. For example if the query text is "መመሪያዎች" and the word image found in the document is "መመሪያ" the system first compare the two word lengths and by taking the word in the image document it will match the features. The word in the document image matches the first four characters of the query word and the system reports that it found the similar word by not considering the remaining characters of the query word.

- This system is less susceptible to spelling errors; compared to Microsoft Vista search this is because it calculates the similarity based on the vector values and these values give a good matching result for similar words because they do not consider characters. For example if the query word is " መስተዳደር" and the word found in the document image is " መስተዳድር" the system will retrieve the document image based on the supplied query. Whereas, in the case of Microsoft Vista search, the document will not be retrieved.

- The system ranks retrieved documents using the frequency of the word image in a document image.

- This research utilize, recall, precision, F-measure and R-precision techniques to evaluate the proposed system.

The weak points of this research

- The Binerization mechanism that has been implemented chooses a fixed threshold selection method and the threshold is set to 128 in the scale of 0-255. This threshold may not work with document images that are damaged or have blurred words.

- If the image document segmented has a full justification, that alien's text to both the right and left margins by adding extra space, the Amharic characters in a word can be printed out with extra space like "መ ስ ተ ዳ ደ ር" in between each character. The system considers each character that has an extra space in between as a word.

- Even if the system removes to some extent punctuation marks and stop words, a lot should be done on considering the removal of these items from the document image.

- The suffix removal in the system is in its infant stage and the system does not incorporate infix and prefix removal from the image words.

- The system does not incorporate a spell checking procedure, despite its capability of finding documents with slight spelling errors; the documents should be checked for spelling errors to increase the retrieval performance.

- Same Amharic words like መንግስት and መንግሥት have the same meaning but this research do not include a mechanism to consider these words as a one word.

- The ranking done does not consider the volume of the document.

Based on the experimental results and the techniques used in the system, the following recommendations are suggested to show the areas that need further research.

5.2. Recommendations

- On the present research an experiment is conducted on limited collection of document images. Future research needs to investigate on different documents taken from news papers, magazines, poor quality and hand written documents.
- Affix, infix and suffix removal should be considered for future work.
- Matching variant words in Amharic is necessary as they have the same meaning with the stem. Hence a matching procedure should be designed that considers these words as a one word
- A technique should be incorporated to detect the format of the word images like Bold, underlined, italic, bulleted, numbered, words in tables, etc
- Pre- processing techniques like noise removal, font resizing, font conversion helps to increase the effectiveness and efficiency of the system. Incorporating these systems will increase the performance of the proposed retrieval system.
- This research focuses on document image retrieval given a simple word query. The researcher recommends continuing this research so that the retrieval system uses multiple word queries by incorporating logical statements.

- To increase the performance of the system by decreasing the search time it is good if the word images are clustered and indexed using an efficient indexing structure.

- For future research it is a good idea to incorporate document ranking by considering the term frequency over inverse document frequency tf/idf.

In general the proposed system is suitable for Amharic document image retrieval and by conducting more research in the area of interest it is possible to develop a real system that can work. The development of such system will enhance the availability of documents especially document images in digital form and alleviate the current problems of document image retrieval.

Bibliography

Ager, S. (2008). *Writting System and language of the system.* Retrieved June 10, 2008, from Omniglot: http://www.omniglot.com/writing/ethiopic.htm

Apple Computer, I. (1996, July 6). *Glossary.* Retrieved January 10, 2009, from Developer Connection: http://developer.apple.com/DOCUMENTATION/mac/Text/Text-593.html

Atelach, A. A., & Asker, L. (2007). *Evaluation of Multilingual and Multi-modal Information Retrieval* (Vol. Volume 4730/2007). Springer Berlin / Heidelberg.

Ayele, B. (1997). *Ethiopic, African Writing Systems.* THE RED SEA PRESS.

Baeza-Yates, R., & Neto, R. (1999). *Modern Information Retrieval.* Boston, USA: Addison Wesley.

Bender, M. B. (1976). *Language in Ethiopia.* London: Exford University press.

Bernal, M. (1990). *Black Athena: the Afroasiatic roots of classical civilization.* Rutgers University Press.

Bizuneh, M. (2003). The Application Of WEBSOM Method To Amharic Text Retrieval. *Master's Thesis* .

Bloor, T. (1995). The Ethiopic Writing System: a Profile. *journal of the Simplified Spelling Society , 2*, 30-36.

Chakravarty, I., & Mitra, D. A. (2007). ROBUST ALGORITHMS FOR SEGMENTATION OF IMAGES. *International Conference on Computing.*

Coulmas, F. (1989). *Writing systems of the world.* Oxford, England: Basil Blackwell.

D. Niyogi, S. S. (1997). The use of a document structure analysis to reterive informationfrom documents in digital liabraries. *Proc. SPIE, Document Recognition IV* , 207-218.

Drakos, N. (1996). *Application Challenges to Computational Geometry.* School of Education, University of Leeds., Computer Based Learning Unit. Princeton University, April 1996.

EICTDA. (2008). *Development of Ethiopic Keyboard Layout and Typeface Standards.* Addis Ababa: Ethiopian Information Communication Technology Development Agency.

Encarta, M. (2007). *Encyclopedia*. Retrieved June 12, 2008, from African_Languages: http://encarta.msn.com/encyclopedia_761565449/African_Languages.html#s9

Ethiopia, T. (2002). Application Of Case Based Reasoning For Amharic Legal Precedent Retrieval A Case Study With The Ethiopian Lab .Law. *Master's Thesis* .

Eyassu, S., & Bjorn, G. (2005). Classifying Amharic News Text Using Self-Organizing Maps. *43rd Annual Meeting of The Association For Computational Linguistics; Workshop on Computational Approaches to Semitic Languages.* Michigan: University of Michigan.

Eyasu, S. (2005). Amharic Text Retrieval Using Neural Networks (Nn) A Comparative Study Using News Items. *Master's Thesis* .

Gaur, A. (1987). *A History of Writing.* London.

Gerald Salton, a. M. (1983). *Introduction to Modern Information Retrieva.* McGraw-Hill.

Getachew, A., & Derib, A. (2006). Language Policy in Ethiopia: History and Current Trends. *Ethiopian Journal of Education and Science , 2* (1), 37-62.

Getachew, H. (1967). The Problems of the Amharic Writing System. *interdisciplinary seminar of the Faculty of Arts and Education. HSIU.* Addis Ababa.

Hamlyn, J. K. (1994). *Modular implementation of feature extraction and matching algorithm for photogrammetric stereo imagery.* AUSTRALIA Electronics research laboratory.

Haque, N., Chowdhury, M., & Rahman, S. M. (2005). Impact of image organizations on multimedia document retrieval. *Fourth Annual ACIS International Conference*, (pp. 340 - 343). Melbourne.

Hayward, K. a. (1999). *Amharic. In Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet.* Cambrige University press.

Hull, J. J. (2004). Document image similarity and equivalence detection. *International Journal on Document Analysis and Recognition* , 37-42.

Inkpen, D. (2008). Information Retrieval on the Internet. Ottawa, ON, Canada, Ottawa, ON, Canada, Ottawa, ON, Canada.

Julien Bayou, C. G. (2006). *Background Paper prepared for the Education for All Global Monitoring Report.* United Nations Educational, Scientific and Cultural Organization.

krysstal.com. (2002). *Writting, The history, development and evolution of the world's writing systems.* Retrieved June 9, 2008, from Krysstal web site: http://www.krysstal.com/writing.html

Landis, S. (1995). *Sean Landis' CS718 Project.* Retrieved September 17, 2008, from Sean Landis: Sean Landis' CS718 Project, Fall 1995

Le, D., Thoma, G., & Wechsler, H. (1996). Automated borders detection and adaptive segmentation for binarydocument images. *The 13th International Conference on Pattern Recognition*, (pp. 737 - 741).

LESLAU, W. (2000). *Introductory Grammar of Amharic.* Introductory Grammar of Amharic. By Porta Linguarum Orientalium.

Liu, J., & Jain, A. K. (1998). Image-Based Form Document Retrieval. *International Conference on Pattern Recognition (ICPR'98), Volume 1*, p. 626. Brisbane, Australia.

Long, F., zhang, H., & Feng, D. D. (2003). *Fundamental of Content-Based Image Reterieval.* Springer.

Lu, S., & Tan, C. L. (2007). Keyword Spotting and Retrieval of Document Images Captured by a Digital Camera. *Ninth International Conference on Document Analysis and Recognition* , (pp. 994-998). Singapore;.

Mandl, T. (2008, April). Recent Developments in the Evaluation of Information Retrieval Systems: Moving Towards Diversity and Practical Relevance. Hildesheim, Germany: Information Science University of Hildesheim Marienburger.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press .

Maureen Lewis, M. L. (2008, March). Social Exclusion and the Gender Gap in Education. *Policy Research Working Paper* . The World Bank Human Development Network.

Million, M. (2000). A generalized Approch To Optical Character Recognition (OCR) of Amharic Texts. *Master's Thesis,* .

Million, M., & Jawahar, V. C. (2008). Matching word images for content-based retrieval from printed document images. *International Journal on Document Analysis and Recognition , Volume 11* (Number 1), 29-38.

Mulegeta, T. B. (2002). Text Retrieval Using Self – Organized Document Map: The Case Of ILRI Digital Library. *Master's Thesis* .

Raymond G. Gordon, J. (2005). *Languages of the World* (15th ed.). Ethnologue.

Saba, A. (2001, July). The Application Of Information Retrieval Techniques For Amharic Documents On The Web. *Master's Thesis* .

Saeed, K., Pejas, J., & Long, R. M. (2006). *Biometrics, computer security systems and artificial intelligence application.*

Seethalakshmi R., S. T. (2005). Optical character recognition for printed Tamil text using Unicode. *Journal of Zhejiang University - Science A* , 1297-1305.

Skarbek, W., Koschan, A., Bericht, T., Veroffentlichung, Z., & Klette. (1994). Color Image Segmentation A Survey.

Solomon, T. A., Menzel, W., & Tafila, B. (2005). *An Amharic Speech Corpus for Large Vocabulary Continuous Speech Recognition.* Hamburg: Universit¨at Hamburg.

Spitz, A. L. (1997). Duplicate Document Detection. 88-94.

Sravana Reddy, G. C. (2006). A Document Recognition System for Early Modern Latin. Chicago: The University of Chicago .

Stauffer, C. (2004). Learning a Probabilistic Similarity Function for Segmentation. *Computer Vision and Pattern Recognition Workshop , Volume 4*, p. 50. Massachusetts.

Supatra, S., & Nualsawat, H. (2007). Unsupervised Image Segmentation Using Automated Fuzzy c-Means. *7th IEEE International Conference on Computer and Information Technology*, (pp. 690-694). Aizu-Wakamatsu City, Fukushima, Japan.

Tan, Lu, Y., & Lim, C. (2004). Information Retrieval in Document Image Databases. *Knowledge and Data Engineering, IEEE Transactions* , 1398 - 1410.

Team, i. M. (2008, April). *iProspect Blended Search Results Study.* Retrieved June 24, 2009, from iProspec web site: www.iprospect.com

Tewodros, H. (2003). Amharic Text Retrieval: An Experiment Using Latent Semantic Indexing (LSI) With Singular Value Decomposition (SUD). *Master's Thesis* .

Tim Kam Ho, J. J. (1992). A word shape analysis approach to lexicon based word recognition. *Pattern Recognition* , 821-826.

Wikipedia. (2009, June 19). *Amharic.* Retrieved June 19, 2009, from wikipedia: http://en.wikipedia.org/wiki/Amharic

Wikipedia. (2009 , April 14). *Ranking function.* Retrieved June 10, 2009, from Wikipedia web site: http://en.wikipedia.org/wiki/Ranking_function

Wikipedia. (2009, June 21). *Rendering (computer graphics).* Retrieved June 22, 2009, from Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Rendering_(computer_graphics)

Wikipedia, t. f. (2009, June 12). *Amhara people*. Retrieved June 16, 2008, from wikipedia.org: http://en.wikipedia.org/wiki/Amhara_people/

Wondewosen, M. (2004). OCR For Special Type Of Hand Written Amharic Text (Yekum Tsifet) Neural Network Approach. *Master's Thesis* .

Wu, T. (1999). *IMAGE SEGMENTATION: THE FIRST STEP IN 3-D IMAGING.* Retrieved June 9, 2008, from 3D-DOCTOR: http://www.ablesw.com/3d-doctor/3dseg.html

XoXus, E. S. (2003, May 01). *SearchCIO-Midmarket.com Definitions* . Retrieved July 10, 2008, from SearchCIO-Midmarket.com: http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci214132,00.html

Y. He, Z. J. (1999). Content Based Indexing and Retrieval Method of Chinese Document Images. *fifth International conferance on Document Analysis and Recognition*, (pp. 685-688).

Y.Y Tang, C. Y. (1994). Document Processing for Automatic Knowledge Acquisition. *IEEE Trans. Knowledge and Data Eng* , 3-21.

Yalemisew, M. (2005). Application Of Hierarchical Document Cluster Based Information Retrieval System For Amharic Documents. *Master's Thesis* .

Yimam. (1986). **የ ዓማርኛ ሰዋሰው**". Addis Ababa : **ት.መ.ማ.ማ.ድ**.

Yoseph, S. (2005). Application Of Multilingual Thesaurus For Cross Language Information Retrieval (CLIR) [Amharic –English Clir For The Legal Environment. *Master's Thesis* .

Young, I., Gerbrands, J. J., & van Vliet, L. J. (2007). *Image Processing Fundamentals.* Quantitative Image Group.

Appendices 1 Java Implementation

```
package imagedocument;
/*
 * ImageInterface.java
 *
 * Created on Apr 8, 2009, 5:10:16 PM
 */
/**
 *
 * @author Mesfin
 *
 */
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedWriter.*;
import javax.media.jai.*;
import java.util.*;
import java.awt.image.BufferedImage.*;
import java.awt.image.*;
import java.awt.*;
import java.io.*;
import javax.media.jai.PlanarImage;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane.*;
import javax.swing.JToggleButton;
import javax.swing.SwingConstants;
import javax.swing.WindowConstants;

public class ImageInterface extends javax.swing.JFrame {

    int width, height, nbands = 0;
    int widthQ, heightQ;
    int dataStart = 0;
    int dataEnd = 0;
    int border = 0;
    int containt = 0;
    int Lnumber = 0;
    int count = 0;
    int count1 = 0;
    int wstart = 0;
    int wend = 0;
    int wdatastart1 = 0;
    int vdatastart1 = 0;
    int wdataend1 = 0;
    int vdataend1 = 0;
    int wborderstart1 = 0;
    int vborderstart1 = 0;
    int wborderend1 = 0;
    int vborderend1 = 0;
    int pixelValT = 0;
    int wordposT = 0;
    int pixelValM = 0;
    int wordposM = 0;
    int pixelValB = 0;
    int wordposB = 0;
```

```
int pixelValL = 0;
int wordposL = 0;
int nbandsQ = 0;
int w = 0;
int h = 0;
int hQ = 0;
int hordataEnd = 0;
int hordataStart = 0;
int pixelaverage = 0;
int pixelaverageQ = 0;
int[] pixel = new int[nbands];
int[] pixelQ = new int[nbandsQ];
PrintWriter pwvb;
PrintWriter pwwb;
PrintWriter pw;
PrintWriter pwwv;
FileWriter fw;
PrintWriter pwwbQ;
PrintWriter pwvbQ;
PrintWriter pwQ;
PrintWriter pwwvQ;
String linev = null;
String line3;
String line = "";
Vector<String> line2 = new Vector<String>(2, 2);
Vector<String> line4 = new Vector<String>(2, 2);
Vector VlinepixVal = new Vector();
File fileRend = new File("newimage3.jpg");
File imageDir = new File("C:/Users/Mesfin/Desktop/NetBe/ImageAccess/dist/Scanned/Final");
File image;
File Vector;
File imageQuery = fileRend;
String ImagenameQ = imageQuery.getPath();
PlanarImage piQ = JAI.create("fileload", ImagenameQ);
SampleModel smQ = piQ.getSampleModel();

ImageAccessMethod AllMethods = new ImageAccessMethod();
TextRendering RenderingMethods = new TextRendering();
/** Creates new form ImageInterface */
public ImageInterface() {
    initComponents();
}
    QuestionLabel = new JLabel();
    RetierevalButton = new JButton();
    ChangeButton = new JToggleButton();
    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    setBounds(new Rectangle(400, 400, 200, 200));
    QuestionLabel.setFont(new Font("Times New Roman", 1, 14));
    QuestionLabel.setHorizontalAlignment(SwingConstants.CENTER);
    QuestionLabel.setText("What Do You Want To Do ?");
    RetierevalButton.setFont(new Font("Times New Roman", 0, 12));
    RetierevalButton.setText("Prepare Document For Retierieval");
    RetierevalButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            RetierevalButtonActionPerformed(evt);
        }
    });
    ChangeButton.setFont(new Font("Times New Roman", 0, 12));
    ChangeButton.setText("Search for a Document ");
    ChangeButton.addActionListener(new ActionListener() {
```

```
  public int getValueZ(int z) {
     z = z + 1;
     return z;
  }
  private void ChangeButtonActionPerformed(ActionEvent evt) {
     showWindowRend AppWindowRend = new showWindowRend();
     AppWindowRend.showWindowRend();
  }
  public static void main(String args[]) {
     try {
        java.awt.EventQueue.invokeLater(new Runnable() {
           public void run() {
              new ImageInterface().setVisible(true);
           }
        });
     } catch (Exception e) {
        System.out.println(e.getMessage().toString());
     }
  }
  // Variables declaration - do not modify
  JToggleButton ChangeButton;
  JLabel QuestionLabel;
  JButton ReterievalButton;
  // End of variables declaration
}

class showWindowRend {

   public void showWindowRend() {
      TextRendering InterfaceRend = new TextRendering();
      InterfaceRend.setVisible(true);
   }
}
```

```
package imagedocument;

/**
 *
 * @author Mesfin
 */
import java.io.BufferedWriter.*;
import java.util.*;
import java.awt.image.BufferedImage.*;
import java.awt.image.*;
import java.io.*;

import javax.media.jai.PlanarImage;
import javax.media.jai.iterator.RandomIter;
import javax.media.jai.iterator.RandomIterFactory;
import javax.swing.JOptionPane.*;

public class ImageAccessMethod {
     public void printpixel(PlanarImage pi, int h, int height, int w, int width,
        int pixelaverage, PrintWriter pw, SampleModel sm, FileWriter fw) {
      try {
         fw = new FileWriter("mydata.txt", true);
         pw = new PrintWriter(fw);
         height = pi.getHeight();
         width = pi.getWidth();
         int nbands = sm.getNumBands();
```

```
        int[] pixel = new int[nbands];
        // Get an iterator for the image.
        RandomIter iterator = RandomIterFactory.create(pi, null);

        for (h = 0; h < height; h++) {
            for (w = 0; w < width; w++) {
                // Get the array of values for the pixel on the w,h coordinate.
                iterator.getPixel(w, h, pixel);
                pixelaverage = 0;
                //}
                for (int band = 0; band < nbands; band++) {
                    pixelaverage = pixel[band] + pixelaverage;
                }
                if (pixelaverage / nbands < 128) {
                    pw.print("0");
                } else {
                    pw.print(" ");
                }
            }
            pw.println();
        }
        pw.flush();
        pw.close();
    } catch (Exception e) {
        System.out.println(e.getMessage().toString());
    }
}

public void Horbord(PlanarImage pi, int h, int height, int w, int width,
        int pixelaverage, int nbands, int[] pixel, SampleModel sm) {
    try {
        FileWriter fwhb = new FileWriter("border.txt");
        PrintWriter pwhb = new PrintWriter(fwhb);
        nbands = sm.getNumBands();
        pixel = new int[nbands];
        height = pi.getHeight();
        width = pi.getWidth();
        RandomIter iterator = RandomIterFactory.create(pi, null);
        for (h = 0; h < height; h++) {

            for (w = 0; w < width; w++) {
                // Get the array of values for the pixel on the w,h coordinate.
                iterator.getPixel(w, h, pixel);
                pixelaverage = 0;
                for (int band = 0; band < nbands; band++) {
                    pixelaverage = pixel[band] + pixelaverage;
                }
                if (pixelaverage / nbands < 128) {
                    pwhb.println("D " + h);
                    break;
                }

            }
            if (pixelaverage / nbands > 128) {
                pwhb.println("B " + h);
            }
        }
        pwhb.flush();
        pwhb.close();
    } catch (Exception e) {
```

```
        e.getMessage().toString();
      }
    }

    public void VerBord(int h, int height, int w, int width, int pixelaverage,
        int nbands, int[] pixel, PlanarImage pi, SampleModel sm, PrintWriter pwvb, PrintWriter pwwb) {
      try {
        int dataStart = 0;
        int dataEnd = 0;
        int border = 0;
        int containt = 0;
        int Lnumber = 0;
        int count = 0;
        int count1 = 0;
        int wstart = 0;
        int wend = 0;
        int wdatastart1 = 0;
        int vdatastart1 = 0;
        int wdataend1 = 0;
        int vdataend1 = 0;
        int wborderstart1 = 0;
        int vborderstart1 = 0;
        int wborderend1 = 0;
        int vborderend1 = 0;
        String line = "";

        DataStartEnd(pi, line, dataStart, dataEnd, border, containt, Lnumber, width,
            pixelaverage, nbands, count, wend, count1, wstart, pixel, w, h, pwvb);
        WordBorder(wdatastart1, vdatastart1, wdataend1, vdataend1, wborderstart1, vborderstart1,
    wborderend1, vborderend1, pwwb);
      } catch (Exception e) {

        System.out.println(e.getCause());
      }

    }

    public void DataStartEnd(PlanarImage pi, String line, int dataStart, int dataEnd,
        int border, int containt, int Lnumber, int width,
        int pixelaverage, int nbands, int count, int wend, int count1,
        int wstart, int[] pixel, int w, int h, PrintWriter pwvb) {

      try {
        FileWriter fwvb = new FileWriter("HorizontalBord.txt");
        pwvb = new PrintWriter(fwvb);
        BufferedReader input = new BufferedReader(new FileReader("border.txt"));
        Vector<String> line2 = new Vector<String>(2, 2);

        while ((line = input.readLine()) != null) {
          line2.addElement(line);

        }
        for (int i = 0; i < (line2.size() - 1); i++) {
          if (line2.elementAt(i).charAt(0) == line2.elementAt(i + 1).charAt(0)) {
            continue;
          }
          while (line2.elementAt(i).contains("B")) {
            String[] data = line2.elementAt(i).split(" ");
            Lnumber = Integer.parseInt(data[1]);
            border = Lnumber;
```

```
            i++;
          }
          dataStart = (border + 1);
          while (line2.elementAt(i).contains("D")) {
            String[] data = line2.elementAt(i).split(" ");
            Lnumber = Integer.parseInt(data[1]);
            containt = Lnumber;
            i++;
          }
          dataEnd = containt;
          PrintStartEnd(pi, dataStart, dataEnd, width, pixelaverage, nbands,
              count, wend, count1, wstart, pixel, w, h, pwvb);
        }
        input.close();
        pwvb.flush();
        pwvb.close();
      } catch (Exception e) {
        System.out.println(e.getMessage());
      }
    }

    public void PrintStartEnd(PlanarImage pi, int dataStart, int dataEnd, int width,
        int pixelaverage, int nbands, int count, int wend, int count1,
        int wstart, int[] pixel, int w, int h, PrintWriter pwvb) {
      try {
        FileWriter fwvb = new FileWriter("HorizontalBord.txt", true);
        pwvb = new PrintWriter(fwvb);
        SampleModel sm = pi.getSampleModel();
        nbands = sm.getNumBands();
        pixel = new int[nbands];
        RandomIter iterator = RandomIterFactory.create(pi, null);
        width = pi.getWidth();
        for (w = 0; w < width; w++) {
          for (h = dataStart; h < dataEnd; h++) {

            // Get the array of values for the pixel on the w,h coordinate.
            iterator.getPixel(w, h, pixel);
            pixelaverage = 0;
            //}
            for (int band = 0; band < nbands; band++) {
              pixelaverage = pixel[band] + pixelaverage;
            }
            if (pixelaverage / nbands < 128) {
              count = 0;
              wend = w;
              count1++;
              break;
            }
          }
          if (pixelaverage / nbands > 128) {
            count++;
            count1 = 0;
          }
          if (count < (dataEnd - dataStart) / 3 && w < (dataEnd - dataStart) / 3 && count1 == 0) {
            wstart = w;
          } else if (count < (dataEnd - dataStart) / 3) {
            wstart = w + count;
          } else {
            wstart = w;
          }
```

```
            if (count >= (dataEnd - dataStart) / 3) {
                pwvb.println("B," + wend + "," + dataStart + "," + (wstart) + "," + dataEnd);
            } else if (count < (dataEnd - dataStart) / 3 && w < (dataEnd - dataStart) / 3 && count1 == 0) {
                pwvb.println("B," + wend + "," + dataStart + "," + (wstart) + "," + dataEnd);
            } else if (count1 != 0) {
                pwvb.println("D," + wstart + "," + dataStart + "," + wend + "," + dataEnd);
            }
          }
          pwvb.flush();
          pwvb.close();
        } catch (Exception e) {
          System.out.println(e.getMessage());
        }
      }

      public void WordBorder(int wdatastart1, int vdatastart1, int wdataend1,
          int vdataend1, int wborderstart1, int vborderstart1, int wborderend1, int vborderend1, PrintWriter pwwb)
{
        try {
          FileWriter fwwb = new FileWriter("wordborder.txt");
          pwwb = new PrintWriter(fwwb);
          BufferedReader inputv = new BufferedReader(new FileReader("HorizontalBord.txt"));
          String linev = null;
          Vector<String> line2 = new Vector<String>(2, 2);
          while ((linev = inputv.readLine()) != null) {
            line2.addElement(linev);
          }
          for (int i = 0; i < (line2.size() - 1); i++) {
            if (line2.elementAt(i).charAt(0) == line2.elementAt(i + 1).charAt(0)) {
              continue;
            }
            while (line2.elementAt(i).contains("B")) {
              String[] data1 = line2.elementAt(i).split(",");
              wborderstart1 = Integer.parseInt(data1[1]);
              vborderstart1 = Integer.parseInt(data1[2]);
              wborderend1 = Integer.parseInt(data1[3]);
              vborderend1 = Integer.parseInt(data1[4]);
              i++;
            }

            while (line2.elementAt(i).contains("D")) {
              String[] data1 = line2.elementAt(i).split(",");
              wdatastart1 = Integer.parseInt(data1[1]);
              vdatastart1 = Integer.parseInt(data1[2]);
              wdataend1 = Integer.parseInt(data1[3]);
              vdataend1 = Integer.parseInt(data1[4]);
              i++;
            }
            if (wdataend1 - wborderend1 > 20 && vdataend1-vdatastart1>10) {
              pwwb.println("C," + (wborderend1 + 1) + "," +
                  (vdatastart1) + "," + wdataend1 + "," + vdataend1);
            }
          }
          inputv.close();
          pwwb.flush();
          pwwb.close();
        } catch (Exception e) {
          System.out.println(e.getCause());
        }
      }
```

```
  public void PrintImageInfo(PlanarImage pi, SampleModel sm, int width, int height, PrintWriter pw, File
image, FileWriter fw) {
    try {

      // create the file called "Mydata.txt"  and "border.txt" for writing the outputs
      fw = new FileWriter("mydata.txt");
      pw = new PrintWriter(fw);
      pw.println("Image file size: " + image.length() + " bytes. ");
      System.out.println("Image file size: " + image.length() + " bytes. ");
      pw.println(" Dimensions ");
      System.out.print(" Dimensions ");
      pw.write(pi.getWidth() + " X " + pi.getHeight() + " pixels. ");
      System.out.print(pi.getWidth() + " X " + pi.getHeight() + " pixels. ");
      pw.print("( from " + pi.getMinX() + ", " + pi.getMinY() + " to " +
          (pi.getMaxX() - 1) + ", " + (pi.getMaxY() - 1) + ")");
      System.out.print("( from " + pi.getMinX() + ", " + pi.getMinY() + " to " +
          (pi.getMaxX() - 1) + ", " + (pi.getMaxY() - 1) + ")");
      System.out.println(" Number of bands: " + sm.getNumBands());
      pw.println(" Number of bands: " + sm.getNumBands());
      pw.println();
      pw.flush();
      pw.close();
      System.out.println();
    } catch (Exception e) {
      e.getMessage();
    }
  }

  public void CornerReader(PlanarImage pi, int w, int h, SampleModel sm,
      int pixelaverage, PrintWriter pwwv, String Imagename, int zz, File Vector) {

    try {
      String line3;
      BufferedReader inputB = new BufferedReader(new FileReader("wordborder.txt"));
      Vector<String> line4 = new Vector<String>(2, 2);


      while ((line3 = inputB.readLine()) != null) {
        line4.addElement(line3);
      }
      System.out.print(line4);
      for (int m = 0; m < (line4.size()); m++) {
        String[] data2 = line4.elementAt(m).split(",");
        int leftCorner = Integer.parseInt(data2[1]);
        int topCorner = Integer.parseInt(data2[2]);
        int rightCorner = Integer.parseInt(data2[3]);
        int bottomCorner = Integer.parseInt(data2[4]);
        RawPartitioning(pi, w, h, topCorner, bottomCorner, leftCorner,
            rightCorner, sm, pixelaverage, pwwv, Imagename, zz, Vector);
      }
      inputB.close();
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }

  public void RawPartitioning(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
Vector) {
```

```
try {
 FileWriter fwwv = new FileWriter("C:/ImageVector/"+ Vector.getName().replaceAll(".jpg", ".txt"),true);
   pwwv = new PrintWriter(fwwv);
   Vector VlinepixVal = new Vector();
   int boundgap = (bottomCorner - topCorner) / 3;
   int topbound = topCorner + boundgap;
   int midbound = topbound + boundgap;
   int bottombound = midbound + boundgap;
   int countTotal = 0;
   int countT, countM, countB, countL = 0;
   int nbands = sm.getNumBands();
   int[] pixel = new int[nbands];
   RandomIter iterator = RandomIterFactory.create(pi, null);
   for (w = leftCorner; w < rightCorner; w++) {
      countT = 0;
      countM = 0;
      countB = 0;
      countL = 0;
      int pixelValT = 0;
      int wordposT = 0;
      int pixelValM = 0;
      int wordposM = 0;
      int pixelValB = 0;
      int wordposB = 0;
      int pixelValL = 0;
      int wordposL = 0;
      for (h = topCorner; h < topbound; h++) {
         iterator.getPixel(w, h, pixel);
         pixelaverage = 0;
         for (int band = 0; band < nbands; band++) {
            pixelaverage = pixel[band] + pixelaverage;
         }
         if (pixelaverage / nbands < 128) {
            pixelValT++;// = pixelValT + wordposT;
         }
         countT++;
      }
      for (h = topbound; h < midbound; h++) {
         iterator.getPixel(w, h, pixel);
         pixelaverage = 0;
         for (int band = 0; band < nbands; band++) {
            pixelaverage = pixel[band] + pixelaverage;
         }
         if (pixelaverage / nbands < 128) {
            pixelValM++;// = pixelValM + wordposM;
         }
         countM++;
      }
      for (h = midbound; h < bottombound; h++) {
         iterator.getPixel(w, h, pixel);
         pixelaverage = 0;
         for (int band = 0; band < nbands; band++) {
            pixelaverage = pixel[band] + pixelaverage;
         }
         if (pixelaverage / nbands < 128) {
            pixelValB++;// = pixelValB + wordposB;
         }
         countB++;
      }
      for (h = bottombound; 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) {
               pixelValL++;
            }
            countL++;
         }
         countTotal = countT + (countM * 2) + (countB * 3) + (countL * 4);
         float VlineVal = (float) (pixelValL*4 + pixelValB*3 + pixelValM*2 + pixelValT) / countTotal;
         VlinepixVal.addElement(VlineVal);
      }
      for (int k = 0; k < VlinepixVal.size(); k++) {
         pwwv.print(VlinepixVal.elementAt(k) + ",");
      }
      pwwv.println();
      pwwv.flush();
      pwwv.close();
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

```
package imagedocument;

/**
 *
 * @author Mesfin
 */
import java.io.BufferedWriter.*;
import java.util.*;
import java.awt.image.BufferedImage.*;
import java.awt.image.*;
import java.io.*;
import javax.media.jai.PlanarImage;
import javax.media.jai.iterator.RandomIter;
import javax.media.jai.iterator.RandomIterFactory;
import javax.swing.JOptionPane.*;

public class QueryAccessMethod {

    File ImageVector;
    File VectorValue;

    public void HorbordQ(PlanarImage piQ, int hQ, int heightQ, int wQ, int widthQ,
            int pixelaverageQ, SampleModel smQ, PrintWriter pwwbQ, int hordataStart, int hordataEnd, int vdatas,
  int vdataend, PrintWriter pwwvQ, String ImagenameQ) {

        try {
            FileWriter fwwbQ = new FileWriter("wordborderQ.txt");
            pwwbQ = new PrintWriter(fwwbQ);
            FileWriter fwwvQ = new FileWriter("wordvalueQ.txt");
            pwwvQ = new PrintWriter(fwwvQ);
            int counthor = 0;
            int nbandsQ = smQ.getNumBands();
            int[] pixelQ = new int[nbandsQ];
```

```
RandomIter iterator = RandomIterFactory.create(piQ, null);
Vector<String> horEndBorder = new Vector<String>(2, 2);
for (hQ = 0; hQ < heightQ; hQ++) {
   for (wQ = 0; wQ < widthQ; wQ++) {
      // Get the array of values for the pixel on the w,h coordinate.
      iterator.getPixel(wQ, hQ, pixelQ);
      pixelaverageQ = 0;
      for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
         pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
      }
      if (pixelaverageQ / nbandsQ < 128) {
         counthor++;
         break;
      }
    }
    if (counthor != 0) {
      horEndBorder.addElement("D," + hQ);
    } else {
      horEndBorder.addElement("B," + hQ);
    }
    counthor = 0;
  }
  for (int i = 0; i <= horEndBorder.size(); i++) {
    System.out.println(horEndBorder.elementAt(i));
    if (horEndBorder.elementAt(i).charAt(0) == horEndBorder.elementAt(i + 1).charAt(0)) {
      continue;
    }
    String[] datahor = horEndBorder.elementAt(i).split(",");
    if (datahor[0].equals("B")) {
      int dataS = Integer.parseInt((datahor[1]));
      hordataStart = dataS + 1;
      System.out.println("Data Start" + (hordataStart));
    } else if (datahor[0].equals("D")) {
      hordataEnd = Integer.parseInt(datahor[1]);
      System.out.println("DataEnd" + hordataEnd);
    }
  }
 } catch (Exception e) {
   e.getMessage().toString();
 }
 VerBordQ(piQ, hQ, heightQ, hQ, widthQ, pixelaverageQ, smQ,
      pwwbQ, hordataStart, hordataEnd,
      vdatas, vdataend, pwwvQ, ImagenameQ);
}

public void VerBordQ(PlanarImage piQ, int hQ, int heightQ, int wQ, int widthQ,
    int pixelaverageQ, SampleModel smQ, PrintWriter pwwbQ,
    int hordataStart, int hordataEnd, int vdatas, int vdataend, PrintWriter pwwvQ, String ImagenameQ) {
  try {
    int verdataStart = 0;
    int verdataEnd = 0;
    int countVer = 0;
    int countspace = 0;
    int countblank = 0;
    int countspacerev = 0;
    int nbandsQ = smQ.getNumBands();
    int[] pixelQ = new int[nbandsQ];
    RandomIter iterator = RandomIterFactory.create(piQ, null);
    Vector<String> verEndBorder = new Vector<String>(2, 2);
    for (wQ = 0; wQ < widthQ; wQ++) {
```

```
        for (hQ = hordataStart; hQ < hordataEnd; hQ++) {
          // Get the array of values for the pixel on the w,h coordinate.
          iterator.getPixel(wQ, hQ, pixelQ);
          pixelaverageQ = 0;
          //}
          for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
            pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
          }
          if (pixelaverageQ / nbandsQ < 128) {
            //  countspace = 0;
            break;
          }
        }
        if (pixelaverageQ / nbandsQ > 128) {
          countspace++;
          verEndBorder.addElement("B," + wQ);
        } else {
          // countspace = 0;
          verEndBorder.addElement("D," + wQ);
        }
      }
      for (int i = 0; i < verEndBorder.size(); i++) {
        System.out.println(verEndBorder.elementAt(i));
        pwwbQ.println(verEndBorder.elementAt(i));
      }
      pwwbQ.flush();
      pwwbQ.close();
      DataStartEndQ(piQ, wQ, hQ, hordataStart, hordataEnd, vdatas,
          vdataend, smQ, pixelaverageQ, pwwvQ, ImagenameQ);
    } catch (Exception e) {
      e.getMessage().toString();
    }
  }

  public void DataStartEndQ(PlanarImage piQ, int wQ, int hQ, int hordataStart,
      int hordataEnd, int vdatas, int vdataend, SampleModel smQ,
      int pixelaverageQ, PrintWriter pwwvQ, String ImagenameQ) {
    int Lnumber = 0;
    int Lnumberrev = 0;
    int count = 0;
    int countrev = 0;
    try {
      String lineQ = "";
      BufferedReader inputQ = new BufferedReader(new FileReader("wordborderQ.txt"));
      Vector<String> lineQ2 = new Vector<String>(2, 2);
      while ((lineQ = inputQ.readLine()) != null) {
        lineQ2.addElement(lineQ);
      }
      int countStart = 0;
      for (int i = 0; i < (lineQ2.size() - 1); i++) {
        if (lineQ2.elementAt(i).charAt(0) == lineQ2.elementAt(i + 1).charAt(0) &&
lineQ2.elementAt(i).contains("B")) {
          count++;
          continue;
        }
        if (countStart == i - count) {
          Lnumber = count;
        } else if (count > (hordataEnd - hordataStart) / 3) {
          String[] data = lineQ2.elementAt(i).split(",");
          Lnumber = Integer.parseInt(data[1]);
```

```
            }
            count = 0;
        }
        System.out.println(Lnumber);
        int CountEnd = lineQ2.size() - 1;
        for (int i = lineQ2.size() - 1; i > 0; i--) {
            if (lineQ2.elementAt(i).charAt(0) == lineQ2.elementAt(i - 1).charAt(0) &&
lineQ2.elementAt(i).contains("B")) {
                countrev++;
                continue;
            }
            if (CountEnd == i + countrev) {
                //Lnumber = count;lineQ2.size() - 1) {
                Lnumberrev = i;//lineQ2.size() - countrev;
            } else if (countrev > (hordataEnd - hordataStart) / 3) {
                String[] data = lineQ2.elementAt(i).split(",");
                Lnumberrev = Integer.parseInt(data[1]);
            }
            countrev = 0;
        }
        vdatas = Lnumber + 1;
        vdataend = Lnumberrev;
        System.out.println(Lnumberrev);
        RawPartitioningQ(piQ, wQ, hQ, hordataStart, hordataEnd, vdatas, vdataend, smQ,
                pixelaverageQ, pwwvQ, ImagenameQ);
    } catch (Exception e) {

        System.out.println(e.getMessage());
    }
}
public void RawPartitioningQ(PlanarImage piQ, int wQ, int hQ, int hordataStart,
        int hordataEnd, int vdatas, int vdataend, SampleModel smQ,
        int pixelaverageQ, PrintWriter pwwvQ, String ImagenameQ) {
    try {
        FileWriter fwwvQ = new FileWriter("wordvalueQ.txt");
        pwwvQ = new PrintWriter(fwwvQ);
        Vector VlinepixValQ = new Vector();
        int boundgapQ = (hordataEnd - hordataStart) / 3;
        int topboundQ = hordataStart + boundgapQ;
        int midboundQ = topboundQ + boundgapQ;
        int bottomboundQ = midboundQ + boundgapQ;
        int countTotalQ = 0;
        int countTQ, countMQ, countBQ, countLQ = 0;
        int nbandsQ = smQ.getNumBands();
        int[] pixelQ = new int[nbandsQ];
        RandomIter iterator = RandomIterFactory.create(piQ, null);
        for (wQ = vdatas; wQ < vdataend; wQ++) {
            countTQ = 0;
            countMQ = 0;
            countBQ = 0;
            countLQ = 0;
            int pixelValTQ = 0;
            int wordposTQ = 0;
            int pixelValMQ = 0;
            int wordposMQ = 0;
            int pixelValBQ = 0;
            int wordposBQ = 0;
            int pixelValLQ = 0;
            int wordposLQ = 0;
            for (hQ = hordataStart; hQ < topboundQ; hQ++) {
```

```
            iterator.getPixel(wQ, hQ, pixelQ);
            pixelaverageQ = 0;
            for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
              pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
            }
            if (pixelaverageQ / nbandsQ < 128) {
              pixelValTQ++;
            }
            countTQ++;
          }
          for (hQ = topboundQ; hQ < midboundQ; hQ++) {
            iterator.getPixel(wQ, hQ, pixelQ);
            pixelaverageQ = 0;
            for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
              pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
            }
            if (pixelaverageQ / nbandsQ < 128) {
              //wordposMQ = 2;
              pixelValMQ++;// = pixelValMQ + wordposMQ;
            }
            countMQ++;
          }
          for (hQ = midboundQ; hQ < bottomboundQ; hQ++) {
            iterator.getPixel(wQ, hQ, pixelQ);
            pixelaverageQ = 0;
            for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
              pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
            }
            if (pixelaverageQ / nbandsQ < 128) {
              pixelValBQ++;
            }
            countBQ++;
          }
          for (hQ = bottomboundQ; hQ < hordataEnd; hQ++) {
            iterator.getPixel(wQ, hQ, pixelQ);
            pixelaverageQ = 0;
            for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
              pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
            }

            if (pixelaverageQ / nbandsQ < 128) {
              pixelValLQ++;
            }
            countLQ++;
          }
          countTotalQ = countTQ + (countMQ * 2) + (countBQ * 3) + (countLQ * 4);
          float VlineValQ = (float) (pixelValLQ*4 + pixelValBQ*3 + pixelValMQ*2 + pixelValTQ*1) /
countTotalQ;
          VlinepixValQ.addElement(VlineValQ);
        }
        for (int k = 0; k < VlinepixValQ.size() - 1; k++) {
          pwwvQ.print(VlinepixValQ.elementAt(k) + ",");
        }
        pwwvQ.flush();
        pwwvQ.close();
      } catch (Exception e) {
        System.out.println(e.getMessage());
      }
    }
```

```
public void ColPartitioningQ(PlanarImage piQ, int wQ, int hQ, int hordataStart,
    int hordataEnd, int vdatas, int vdataend, SampleModel smQ,
    int pixelaverageQ, PrintWriter pwwvQ, String ImagenameQ, PrintWriter pwwhQ) {

 try {
    FileWriter fwwvQ = new FileWriter("wordvalueQv.txt");
    pwwvQ = new PrintWriter(fwwvQ);
    FileWriter fwwhQ= new FileWriter("wordvalueQh.txt");
    pwwhQ= new PrintWriter(fwwhQ);
    Vector [] VlinepixValQ = new Vector[2];
    int boundgapQV = (hordataEnd - hordataStart) / 3;
    int boundgapQH=(vdataend-vdatas)/3;
    int topboundQV = hordataStart + boundgapQV;
    int leftboundQH=vdatas+boundgapQH;
    int midboundQV = topboundQV + boundgapQV;
    int midboundQH=leftboundQH+boundgapQH;
    int bottomboundQV = midboundQV + boundgapQV;
    int rightboundQH=midboundQH+boundgapQH;
    int countTotalQV = 0;
    int countTotalQH=0;
    int countTQV, countMQV, countBQV, countLQV = 0;
    int countLQH, countMLQH, countMRQH, countRQH = 0;
    int nbandsQ = smQ.getNumBands();
    int[] pixelQ = new int[nbandsQ];
    RandomIter iterator = RandomIterFactory.create(piQ, null);

    // For vertical vector
    for (wQ = vdatas; wQ < vdataend; wQ++) {
       countTQV = 0;
       countMQV = 0;
       countBQV = 0;
       countLQV = 0;
       int pixelValTQV = 0;
       int wordposTQV = 0;
       int pixelValMQV = 0;
       int wordposMQV = 0;
       int pixelValBQV = 0;
       int wordposBQV = 0;
       int pixelValLQV = 0;
       int wordposLQV = 0;
       for (hQ = hordataStart; hQ < topboundQV; hQ++) {
          iterator.getPixel(wQ, hQ, pixelQ);
          pixelaverageQ = 0;
          for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
             pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
          }
          if (pixelaverageQ / nbandsQ < 128) {
             pixelValTQV++;
          }
          countTQV++;
       }
       for (hQ = topboundQV; hQ < midboundQV; hQ++) {
          iterator.getPixel(wQ, hQ, pixelQ);
          pixelaverageQ = 0;
          for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
             pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
          }
          if (pixelaverageQ / nbandsQ < 128) {
             pixelValMQV++;
          }
```

```
                countMQV++;
            }
            for (hQ = midboundQV; hQ < bottomboundQV; hQ++) {
                iterator.getPixel(wQ, hQ, pixelQ);
                pixelaverageQ = 0;
                for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
                    pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
                }
                if (pixelaverageQ / nbandsQ < 128) {
                    pixelValBQV++;
                }
                countBQV++;
            }
            for (hQ = bottomboundQV; hQ < hordataEnd; hQ++) {
                iterator.getPixel(wQ, hQ, pixelQ);
                pixelaverageQ = 0;
                for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
                    pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
                }
                if (pixelaverageQ / nbandsQ < 128) {
                    pixelValLQV++;
                }
                countLQV++;
            }
            countTotalQV = countTQV + (countMQV * 2) + (countBQV * 3) + (countLQV * 4);
            float VlineValQV = (float) (pixelValLQV*4 + pixelValBQV*3 + pixelValMQV*2 +
pixelValTQV*1) / countTotalQV;
            VlinepixValQ[1].addElement(VlineValQV);
        }
        // For Horizontal vector
        for (hQ = hordataStart; hQ < hordataEnd; hQ++) {
            countLQH = 0;
            countMLQH = 0;
            countMRQH = 0;
            countRQH = 0;
            int pixelValLQH = 0;
            int wordposLQH = 0;
            int pixelValMLQH = 0;
            int wordposMLQH = 0;
            int pixelValRQH = 0;
            int wordposRQH = 0;
            int pixelValMRQH = 0;
            int wordposMRQV = 0;
            for (wQ = vdatas; wQ < leftboundQH; wQ++) {
                iterator.getPixel(wQ, hQ, pixelQ);
                pixelaverageQ = 0;
                for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
                    pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
                }
                if (pixelaverageQ / nbandsQ < 128) {
                    //wordposTQ = 1;
                    pixelValLQH++;// = pixelValTQ + wordposTQ;
                }
                countLQH++;
            }
            for (wQ = leftboundQH; wQ < midboundQH; wQ++)
            {
                iterator.getPixel(wQ, hQ, pixelQ);
                pixelaverageQ = 0;
                for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
```

```
                pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
              }
              if (pixelaverageQ / nbandsQ < 128) {
                //wordposMQ = 2;
                pixelValMLQH++;// = pixelValMQ + wordposMQ;
              }
              countMLQH++;
            }
for (wQ = midboundQH; wQ < rightboundQH; wQ++)
            {
              iterator.getPixel(wQ, hQ, pixelQ);
              pixelaverageQ = 0;
              for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
                pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
              }
              if (pixelaverageQ / nbandsQ < 128) {
                pixelValMRQH++;// = pixelValBQ + wordposBQ;
              }
              countMRQH++;
            }
          for (wQ = rightboundQH; wQ < vdataend; wQ++)
            {
              iterator.getPixel(wQ, hQ, pixelQ);
              pixelaverageQ = 0;
              for (int bandQ = 0; bandQ < nbandsQ; bandQ++) {
                pixelaverageQ = pixelQ[bandQ] + pixelaverageQ;
              }
              if (pixelaverageQ / nbandsQ < 128) {
                pixelValRQH++;
              }
              countRQH++;
            }
            countTotalQH = countLQH + (countMLQH * 2) + (countMRQH * 3) + (countRQH * 4);
            float VlineValQH = (float) (pixelValRQH*4 + pixelValMRQH*3 + pixelValMLQH*2 +
  pixelValLQH*1) / countTotalQH;
            VlinepixValQ[0].addElement(VlineValQH);
          }
          for (int k = 0; k < VlinepixValQ[1].size() - 1; k++) {
            pwwvQ.print(VlinepixValQ[1].elementAt(k) + ",");
          }
          for (int k = 0; k < VlinepixValQ[0].size() - 1; k++) {
            pwwhQ.print(VlinepixValQ[0].elementAt(k) + ",");
          }
          pwwvQ.flush();
          pwwvQ.close();
        } catch (Exception e) {
          System.out.println(e.getMessage());
        }
      }
      public void cosienSimilarityQ(int wdatastart1Q, int vdatastart1Q, int wdataend1Q,
          int vdataend1Q, int wborderstart1Q, int vborderstart1Q, int wborderend1Q, int vborderend1Q,
    PrintWriter pwwbQ) {
        try {
          ImageVector = new File("c:/ImageVector");


          if (ImageVector.isDirectory()) {
            String s[] = ImageVector.list();
            for (int z = 0; z < s.length; z++) {
              VectorValue = new File(ImageVector.getPath() + "/" + s[z]);
```

```
 BufferedReader inputvQ = new BufferedReader(new FileReader("wordvalueQ.txt"));
BufferedReader inputvI = new BufferedReader(new FileReader(VectorValue));
String linevQ = null;
String linevI = null;
double SIres=0;
double SQres=0;
double difre=0;
double dotproduct = 0;
double sqrData1 = 0;
double sqrData2 = 0;
double Ires = 0;
double Qres = 0;
int dispnum = 0;
int countrel=0;
String[] dataQ = null;
String[] dataI = null;
Vector<String> lineQ5 = new Vector<String>(2, 2);
Vector<String> line2 = new Vector<String>(2, 2);
while ((linevQ = inputvQ.readLine()) != null) {
   lineQ5.addElement(linevQ);
}
for (int k = 0; k < (lineQ5.size()); ++k) {
   dataQ = lineQ5.elementAt(k).split(",");//elementAt(k);
}
while ((linevI = inputvI.readLine()) != null) {
   line2.addElement(linevI);
}
      int numberf=0;
for (int i = 0; i < (line2.size()); i++) {

   dataI = line2.elementAt(i).split(",");
    if (dataQ.length <= dataI.length) {
       for (int h = 0; h < dataQ.length; ++h) {
          Qres = Double.parseDouble(dataQ[h]);
          SQres=SQres+Qres;
          if (dataI[h].contains("[")) {
             dataI[h] = dataI[h].substring(1, dataI[h].length());
          } else if (dataI[h].contains("]")) {
             dataI[h] = dataI[h].substring(0, dataI[h].length() - 1);
          }

          Ires = Double.parseDouble(dataI[h]);
          dotproduct = (Qres * Ires) + dotproduct;
          sqrData1 = (Qres * Qres) + sqrData1;
          sqrData2 = (Ires * Ires) + sqrData2;
          difre=(Math.pow((Qres-Ires),2))+difre;
          SIres=SIres+Ires;
       }
    } if (dataQ.length > dataI.length) {
       for (int h = 0; h < dataI.length; ++h) {
          if (dataI[h].contains("[")) {
             dataI[h] = dataI[h].substring(1, dataI[h].length());
          } if (dataI[h].contains("]")) {
             dataI[h] = dataI[h].substring(0, dataI[h].length() - 1);
          }
          Ires = Double.parseDouble(dataI[h]);
          Qres = Double.parseDouble(dataQ[h]);
           SIres=SIres+Ires;
           SQres=SQres+Qres;
          dotproduct = (Qres * Ires) + dotproduct;
```

```
                    sqrData1 = (Qres * Qres) + sqrData1;
                    sqrData2 = (Ires * Ires) + sqrData2;
                    difre=(Math.pow((Qres-Ires),2))+difre;
                  }
                 }
                 double similarityC = 0;
                 double similarityE=0;
                 similarityE=(Math.sqrt(difre))/(SQres+SIres);
                 similarityC = dotproduct / Math.sqrt(sqrData1 * sqrData2);
                dispnum++;
                if (similarityE<0.030)
                 {
                    numberf=1;
                   countrel++;
                 System.out.println("SimilarityE " + similarityE + "," + (dispnum)+ "   "+VectorValue);
                 }
                 dotproduct = 0;
                 sqrData1 = 0;
                 sqrData2 = 0;
                 SQres=0;
                 SIres=0;
                 difre=0;
               }
               if (numberf==1)
               System.out.println(countrel);

            }
          }

      } catch (Exception e) {
         System.out.println(e.getMessage());
      }
   }
}
```

```
package imagedocument;

*
 * @author Mesfin
 */
import java.io.BufferedWriter.*;
import javax.media.jai.*;
import java.awt.image.BufferedImage.*;
import javax.media.jai.PlanarImage;
import javax.swing.JOptionPane.*;
import javax.imageio.*;
import java.io.BufferedWriter.*;
import java.awt.image.BufferedImage.*;
import java.awt.image.*;
import java.awt.*;
import java.io.*;
import javax.swing.JOptionPane.*;

public class TextRendering extends javax.swing.JFrame {

   int width;//= 0;
   int height;//= 0;
   int nbands = 0;
   int[] pixel = new int[nbands];
   int dataStart = 0;
```

```
    int dataEnd = 0;
    int border = 0;
    int containt = 0;
    int Lnumber = 0;
    int count = 0;
    int count1 = 0;
    int wstart = 0;
    int wend = 0;
    PrintWriter pwvb;
    PrintWriter pwwb;
    PrintWriter pw;
    FileWriter fw;
    PrintWriter pwwbQ;
    PrintWriter pwvbQ;
    PrintWriter pwQ;
    PrintWriter pwwvQ;
    int wdatastart1 = 0;
    int vdatastart1 = 0;
    int wdataend1 = 0;
    int vdataend1 = 0;
    int wborderstart1 = 0;
    int vborderstart1 = 0;
    int wborderend1 = 0;
    int vborderend1 = 0;
    String line = "";
    int pixelValT = 0;
    int wordposT = 0;
    int pixelValM = 0;
    int wordposM = 0;
    int pixelValB = 0;
    int wordposB = 0;
    int pixelValL = 0;
    int wordposL = 0;
    int nbandsQ = 0;
    int w = 0;
    int h = 0;
    int hQ = 0;
    int hordataEnd = 0;
    int hordataStart = 0;
    int pixelaverage = 0;
    int pixelaverageQ = 0;
    int[] pixelQ = new int[nbandsQ];
    File fileRend = new File("newimage3.jpg");
File("C:/Users/Mesfin/Desktop/NetBe/ImageAccess/dist/Parallel/Scanned/Amh001/2009-04-08 Doc4");
    String TextRend = "";
    QueryAccessMethod AllQueryMethods = new QueryAccessMethod();

    /** Creates new form TextRendering */
    public TextRendering() {
        initComponents();
    }

    private void initComponents() {

        jTextFieldWord = new javax.swing.JTextField();
        jLabelWord = new javax.swing.JLabel();
        jButtonWord = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
        jTextFieldWord.setBackground(new java.awt.Color(204, 255, 204));
        jTextFieldWord.setFont(new java.awt.Font("Power Geez Unicode1", 0, 12)); // NOI18N
        jTextFieldWord.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextFieldWordActionPerformed(evt);
            }
        });

        jLabelWord.setBackground(new java.awt.Color(255, 255, 0));
        jLabelWord.setFont(new java.awt.Font("Times New Roman", 1, 14));
        jLabelWord.setText("Insert word To search ");

        jButtonWord.setText("Search");
        jButtonWord.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonWordActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabelWord, javax.swing.GroupLayout.PREFERRED_SIZE, 172,
 javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextFieldWord, javax.swing.GroupLayout.PREFERRED_SIZE, 182,
 javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap(212, Short.MAX_VALUE)
                .addComponent(jButtonWord)
                .addGap(101, 101, 101))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(21, 21, 21)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabelWord, javax.swing.GroupLayout.DEFAULT_SIZE, 34,
 Short.MAX_VALUE)
                    .addComponent(jTextFieldWord, javax.swing.GroupLayout.DEFAULT_SIZE, 34,
 Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jButtonWord)
                .addGap(11, 11, 11))
        );

        pack();
    }// </editor-fold>

    private void jTextFieldWordActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButtonWordActionPerformed(java.awt.event.ActionEvent evt) {
        TextRend = jTextFieldWord.getText().trim();
        TextToimage TextConvert = new TextToimage();
```

```
    TextConvert.textRend(TextRend, fileRend);
    File imageQuery = fileRend;
    String ImagenameQ = imageQuery.getPath();
    PlanarImage piQ = JAI.create("fileload", ImagenameQ);
    SampleModel smQ = piQ.getSampleModel();
    int widthQ = piQ.getWidth();
    int heightQ = piQ.getHeight();
    AllQueryMethods.HorbordQ(piQ, hQ, heightQ, hQ, widthQ, pixelaverageQ,
            smQ, pwwbQ, hordataStart, hordataEnd, nbands, nbandsQ,
            pwwvQ, ImagenameQ);
pixelaverageQ, pwwvQ, ImagenameQ);
    AllQueryMethods.cosienSimilarityQ(widthQ, heightQ, nbands, nbands, widthQ, heightQ, nbands, nbands,
pwwbQ);
}
    // Variables declaration - do not modify
    javax.swing.JButton jButtonWord;
    javax.swing.JLabel jLabelWord;
    javax.swing.JTextField jTextFieldWord;
    // End of variables declaration
}

class TextToimage {

    public void textRend(String TextRend, File fileRend) {
        Font x = new Font("Power Geez Unicode1", Font.PLAIN, 12);
        int width = 400;
        int height = 100;
        // Create a buffered image in which to draw
        BufferedImage rendImage = new BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRAY);
        // Create a graphics contents on the buffered image
        Graphics2D g2d = rendImage.createGraphics();//
        g2d.scale(4.22,4.22);
        g2d.setBackground(Color.WHITE);
        g2d.clearRect(0, 0, width, height);
        g2d.setFont(x);
        g2d.setColor(Color.black);
        g2d.drawString(TextRend, 4, 16);
        g2d.dispose();


        try {
            // Save as PNG
            fileRend = new File("newimage3.png");
            ImageIO.write(rendImage, "png", fileRend);

            // Save as JPEG
            fileRend = new File("newimage3.jpg");
            ImageIO.write(rendImage, "jpg", fileRend);
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Appendices 2 **Amharic Alphabets (ፊደል)** (Writting System and language of the system)

Basic Alphabets

1st	2nd	3rd	4th	5th	6th	7th
ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

Letter Variants (labiovelars)

	ä	i	a	e	ə
kʷ	ቈ	ቍ	ቋ	ቌ	ቍ
kʰʷ	ኈ	ኍ	ኋ	ኌ	ኍ
gʷ	ጐ	ጕ	ጓ	ጔ	ጕ
kʷ	ኰ	ኵ	ኳ	ኴ	ኵ
gʷ	ጐ	ጕ	ጓ	ጔ	ጕ

ሏ	ሗ	ሟ	ሧ	ሯ
ሷ	ሿ	ቧ	ቷ	ቿ
ኗ	ኟ	ዟ	ዧ	ዷ
ጇ	ጧ	ጯ	ፏ	ፗ
ፏ	ᎏ	ᎇ	ᎍ	ᎏ

Punctuation

፣ ። ፤ ፥ ፦ ፧

comma | full stop / period | colon | semi-colon | preface colon | question mark (no longer used)

!, << >> :

Exclamation mark | Quotations | Word separater

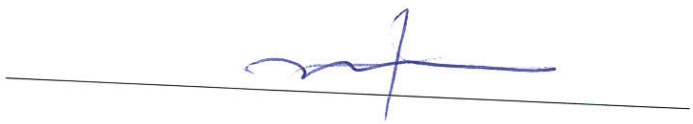
Numerals

These numerals developed from the Greek alphabet, possibly via Coptic.

፩	፪	፫	፬	፭	፮	፯	፰	፱	፲
1	2	3	4	5	6	7	8	9	10
፳	፴	፵	፶	፷	፸	፹	፺	፻	፼
20	30	40	50	60	70	80	90	100	10000

DECLARATION

This thesis is my original, has not been presented for a degree in any other university and that all sources of material used for the thesis have been duly acknowledged



Mesfin Worku

June 2009

The thesis has been submitted for examination with my approval as a university advisor

Million Meshesha