ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

DEPARTMENT OF INFORMATION SCIENCE

SEARCHING IN AMHARIC DOCUMENT IMAGE CORPUS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

By:

Abreham Gebretsadik Teklu

July 2010

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

DEPARTMENT OF INFORMATION SCIENCE

SEARCHING IN AMHARIC DOCUMENT IMAGE CORPUS

By:

Abreham Gebretsadik Teklu

Name and Signature of Members of the Examining Board

	Name	**Signature**
Advisor	Million Meshesha (Ph.D)	_____
Chair Person	_____	_____
Examiner	_____	_____

DEDICATION

To my family: which is a wonderful gift from God.

I love you all.

ACKNOWLEDGMENT

First of all, I would like to praise God who has been there with me all the time, even at the lowest moment of my life. Next, I would like to thank my adviser Dr. Million Meshesha for his constructive comments and encouragement on my work. This work would not be possible without the unreserved and careful technical assistance of my advisor.

My unlimited special thanks go to my parents Gebretsadik Teklu and Mebrat Gebrezigabher for their wholehearted financial and moral support. My greatest love and gratefulness goes to them for always being there for me.

My deepest gratitude goes to my sisters Selamawit Gebretsadik and Fana Gebretsadik who supports me in every aspect of my life and also my brothers Kassa Gebretsadik and Yohannes Gebretsadik.

My boundless thank goes to Mr. Enzo and Mrs. Sophie Zoroni for their support of providing a laptop that helps me as an engine and for their excellent advices.

I would also thank W/o Medhine Teklu for her constructive encouragement and Ato Gebrewahid for his unlimited advice.

I am also indebted to Ato Awetahegn Kiros for the help he offered to me during my stay in the postgraduate school and Ato Mesfin Worku who gave me his code that helps me as an initial idea.

Finally, I would like to give my gratitude to people who are not mentioned in name but whose effort helped me much all along.

Thank you all

Abreham Gebretsadik

ABSTRACT

The introduction of World Wide Web has made access to digital information easier than ever before. Many information providers have therefore been started to digitize existing paper materials to enable access through networked information service. Nowadays, document retrieval becomes the main issue in information retrieval in order to search for relevant document as per users query. There are two types of document retrieval, text retrieval or document image retrieval. Document image retrieval can be recognition-based or without explicit recognition.

There are a number of researches that have been done on document image retrieval throughout the world but there is few research in Amharic document image retrieval. As a result, this study deals with searching from Amharic document image corpus without explicit recognition.

This study aims at improving efficiency and effectiveness of the retrieval system from document image collection. To this ends an inverted index file is created to store index terms after removing stopwords and grouping together variant words. Prefix and suffix of word variants are detected by modifying cosine similarity measure. The index file is constructed using inverted file structure. The search result of the system is displayed in ranked order based on TF*IDF weight, and performance evaluation of the system shows a promising result. However there is a need to solve issues related to feature extraction, word variation detection and noise detection and removal. Accordingly, further research works are recommended.

Table of Contents

List of Tables

List of Figures

List of Equations

List of Algorithms

List of Acronyms

CF:	Cumulative Frequency
DF:	Document Frequency
Dpi:	Density Per Inch
DTW:	Dynamic Time Wrapping
FREQ:	Frequency
Id:	Identification
IDF:	Inverse Document Frequency
IR:	Information Retrieval
OCR:	Optical Character Recognition
TF:	Term Frequency
TOTFREQ:	Total Frequency

CHAPTER ONE

INTRODUCTION

1.1 Background

Individuals use information for different walks of lifestyles. One of the most common ways to define information is to describe it as one or more statements of facts that are received by a human and that have some form of worth to the recipient (Losee, 1998).

The world has changed and hundreds of millions of people are engaged in an information retrieval every day using search engines via the Internet. As we have vast amount of information which is stored on the Internet, accurate and speedy access to the relevant information is becoming more difficult than ever. So the problem of information retrieval creates attention on the minds of researchers. Information retrieval is a process of searching documents that satisfy for an information users from large collection of unstructured documents. Information in this context can be composed of text, images, audio, video and other multimedia objects (Manning et al., 2008)

In the past 20 years, the area of information retrieval has grown well beyond its primary goals of indexing text and searching for useful documents in a collection. Nowadays, research in information retrieval includes, among others modeling, document classification, filtering and document retrieval (Beaza-Yates and Ribeiro-Neto, 1999).

Document retrieval first emerged as a field of both inquiry and application. Later, in the late1940's the number of scientific literature that were being published has grown at alarming

rate and it raised concerns as to how scientists would be able to stay informed of new developments as they were being reported in the scientific literature. Researchers began to consider how computers might be programmed to accomplish some of the laborious human tasks of representing and retrieving relevant sources. Initially, document retrieval means citation retrieval (Liddy, 2004), that is, a search against bibliographic fields such as title, author, source, and subject-based keywords from a controlled vocabulary that were humanly assigned to documents. The documents themselves were neither actually stored nor accessible by computers.

Retrieval is an operation of accessing information from collection of documents. Document retrieval matches some stated user query against useful parts of document image records; these records could be any type of mainly unstructured image, such as bibliographic records, newspapers, books or paragraphs in a manual. Recent years have seen a rapid increase in the size of document image collections. Every day, both military and civilian equipment generates Giga-bytes of images (Manning et al., 2008; Rui et al., 1999). A huge amount of information is out there, and we would now like to be able to search collections that total in the order of billions to trillions of words. (Manning et al., 2008; Rui et al., 1999). To retrieve information from document image, we can use either optical character retrieval or document image retrieval (Liddy, 2004).

Optical character retrieval refers to the process by which scanned images are electronically processed and converted to an editable text (ASCII or Unicode), where as document image retrieval aims at finding relevant documents from a corpus of digitized pages relying on image features only (Marinai, Soda, 2009; Cole et al. 1996).

1.2 Statement of the Problem and Justification

The prevalence of the World Wide Web makes easy to access digital information than ever before. Many information providers have therefore been inspired to digitize existing paper materials that could be accessible through networked information services.

Amharic is an official working language of the federal government and most regional states of Ethiopia, with more than 20 million speakers (Getachew and Derib, 2006). Amharic is today probably one of the five with largest speaker of the language on the continent (Samual, 2005). Accordingly, there are huge amount of hardcopy Amharic documents that are available in government and non-government offices.

There are a number of researches done to explore the application of optical character recognition in order to convert document image into their equivalent textual formats (Worku, 1997; Ermias, 1998; Dereje, 1999; Million, 2000; Wondewosen, 2004). But the development of optical character recognition is a long term solution. So in parallel there are researches on Amharic document image retrieval without explicit recognition as a short term solution (Million, 2008; Mesfin, 2009). Such works help searching relevant documents directly from collection of image, and this kind of retrieval works by comparing the query with the document in the form of word image comparison.

Million (2008) employs a feature of extraction scheme that extracts local features by scanning vertical strips of the word image. A hierarchical indexing scheme is also designed to speed up searching for a relevant document images. A word set is identified by clustering them into different groups based on their similarities as measured using dynamic time warping (DTW).

Mesfin (2009) attempted to design document image retrieval that provide response to users query. To identify word images from document collection, threshold segmentation technique was employed. In the study, he used a word shape analysis or parallel bar vertical technique in order to extract feature. To measure the similarity between a query and the document image, Euclidean similarity measurement with a parallel bar vertical line feature extractions was used and the result of mean average precision was 82% and suffix detection is performed to remove image at the end pixels of the word image. However, word formulation in Amharic language involves both suffix and prefix detection, and hence there is a need to consider prefix detection in addition to suffix detection. Mesfin (2009) notes that, searching a document from 483 pages takes about 22 seconds. Since the image documents can be in thousands or even in millions, the system needs further improvement to speed up searching in order to satisfy users in addition to this, the relevant documents should be retrieved in ranked order to suit information need of users.

The area of document image retrieval needs further investigation to come up with a practical system that enable searching with faster speed and with better result that provide more relevant documents for the user. Research in document image retrieval raises many issues related to efficiency and effectiveness. Fast searching, stemming word variants and partial matching are the crucial challenges, which require detailed experimentation. It is therefore the aim of the present research to explore searching techniques to enhance efficiency and effectiveness of Amharic document image retrieval system.

1.3 Objectives

This thesis work has a general objective and a list of specific objectives that are achieved as mentioned below.

1.3.1 General Objective

The general objective of this study is to design an effective and efficient document retrieval system that searches relevant documents from Amharic image corpus and displays documents in ranked order as per users query.

1.3.2 Specific Objectives

To achieve the general objective, the study accomplishes the following specific objectives.

- To review a literature on related works for conceptual understanding and investigate what have been done and what have not been done.
- To investigate word formation in Amharic writing sytem which helps to cluster word with the same meaning but varies in structure.
- To study different techniques used for indexing, similarity measure and weighting scheme inorder to select applicable algorithms for retrieval of Amharic document images.
- To design the architecture of document image retrieval system consisting of indexing and searching subsystems.
- To develop a retrieval system that searches from document image corpus.

- To evaluate effectiveness of the system using the popular methods of information retrieval evaluation such as recall, precision and F-measure.

- To pass concluding remarks and to forward recommendation for further study in the area of document image retrieval.

1.4 Methodology of the Study

In order to achieve the general and specific objectives of the study, the following methods have been used.

1.4.1 Literature Review

Related literature from different sources (books, journals, articles and the Internet) have been reviewed to understand in detail the field of document image retrieval, to select tools and procedures suitable for developing the system. Literature is also assessed to identify the characteristics of Amharic words and their variation.

1.4.2 Dataset Preparation

The nature of the study indicates that input of the system is digitized Amharic documents image corpus. In the course of this research, some Amharic document images are collected and digitized to evaluate the performance of the system. A flatbed scanner is used for the digitalizing process.

1.4.3 Implementation Procedures

Designing the system requires an integration of indexing and searching subsystems, which passes through many steps. The main once are image preprocessing, segmentation, feature extraction, indexing, similarity measure and searching in Amharic document image corpus.

To implement each of the steps, advanced Java image-programming language is used. The reason why we chose Java programming language is, first, this research is a continuation of Mesfin's (2009) work, which was implemented using Java Programming Language. The other reason is, familiarity of the researcher with Java Programming Language, and Java has a number of built in packages which helps for image processing and also supports to write module source code which is reusable and easily modifiable (James and Henry, 1997).

1.4.4 Testing Procedures

There is a need to evaluate the system using data corpus, which is prepared for this purpose. The effectiveness of the system is evaluated using recall, precision and F-measure. Recall evaluates the ability of the system to retrieve relevant documents from total number of relevant documents, where precision evaluates the ability of the system to retrieve only the relevant documents from the retrieved documents (Baeza-Yates and Ribeiro-Neto, 1999). Finally, F-measure is used to analyze the maximum recall and precision that is registered in this study (Baeza-Yates and Ribeiro-Neto 1999).

1.4 Scope and Limitation of the Study

This study stands from the continuation of Mesfin's (2009) research in the area of document image retrieval. The main purpose is being to improve the performance (effectiveness and

efficiency) of the Amharic document image retrieval by designing an indexing and searching subsystem. During indexing and searching, the system considers stemming word images in order to group words with the same meaning into one cluster, exploring further study for fast searching of documents, and similarity measure from scanned Amharic document. The stemming algorithm works to detect prefix and suffix of a word. Other word variations and synonymous words are not detected. This is because of time constraint.

The corpus collection of the study is limited to printed Amharic documents, synonymous and word variation like infixes are not detected and the query provided to the system is limited to single word query.

1.6 Application of the Study

Nowadays, the need for using information technology in information storage and retrieval is becoming unquestionable.The following are some of the application of Amharic document image retrieval system.

- Digital labraries can use the result of the study to facilliate searching of Amharic document images by providing query to the system.
- It helps users (organizations and individuals) to search for relevant documents, that matches their search query.

In general, the result of the study helps search engine developers in designing application of search engine for Amharic language.

1.7 Organization of the Study

This thesis is organized into six chapters. The first chapter includes the background, the statement of the problem and its justification. It also presents objectives, methodologies and scope of the study.

The second chapter discusses the basic methods of preprocessing techniques of document image retrieval system and the phases in order to increase performance of the document image retrieval system. It also presents related works on document image retrieval system in general and Amharic document image retrieval in particular.

The third chapter presents the nature, characteristics and fundamental problems of Amharic writing system for the case of document image retrieval systems, including formation of Amharic word variants.

The fourth chapter deals on the basic techniques for document image retrieval, including the algorithm used in order to construct the proposed model of the study.

Chapter five deals with the experimentation of the retrieval system implemented in the study. The chapter also presents a thorough performance evaluation of the system based on which analysis and finding are done for identifying further research to improve efficiency and effectiveness of the system.

The last chapter summarizes the overall task performed and the results achieved in this study. This chapter also forwards a set of recommendations for further study in the area of document image retrieval.

CHAPTER TWO

REVIEW OF LITERATURE ON DOCUMENT IMAGE RETRIEVAL

2.1 Introduction

Information retrieval is the process of searching documents of unstructured in nature (usually text, image, audio and video) that satisfies information need of users from within large collections (Manning et al., 2008). Document Retrieval produces a list of documents that are relevant to an inquirer's request by comparing the user's request to the content of documents in the system (Liddy, 2004).

Retrieval systems have grown dramatically during recent years due to very rapid increase of available storage capacity, increased performance of all types of processors and exponential growth of the global networks that provide an enormous source of different documents (Martynov and Novikov. 1996).

These days with the emergence of digital libraries, high-speed digitization process is held all over the word. This results in a huge collection of scanned document images of book, articles, etc. To access from such corpus, document image retrieval is an active research.

2.2 Document Image Retrieval

According to Marinai (2006), there are two types of document image retrieval system: Recogintion Based Retrieval and Document Image Retrieval with out recognition.

2.2.1 Recognition Based Retrieval

Recognition-based document image retrieval is highly dependent on optical character recognition (OCR) engine and it works by scanning source documents and performing character analysis on the resulting images, giving a translation to ASCII or Unicode text, which can then be stored and manipulated like any standard electronic documents (Steven et al. 2003).

Optical Character Recognition is the process of examining printed or handwritten characters on paper and determining their shapes by detecting patterns of dark and light. Once the scanner or reader has determined the shapes, character recognition methods and pattern matching with stored sets of characters are used to translate the shapes into computer text or computer readable format (Wondewosen, 2004).

Optical Character Recognition reading devices are fundamentally classified into two categories, Text Input and Data Capture. Text input devices are page readers or document scanners that scan an entire document or large portion of document. The source data is entered with the intention of someone editing it during or after it is scanned. Text input devices have varying degrees of automation from hand-fed to having automatic feeding, reading, sorting, and stacking capabilities. Data Capturing devices are designed to capture repetitive data and perform formatting functions on the data as it is being entered. The data delivered from the scanner to the computer must be very accurate because it has entered without the intention of being edited later, so, accuracy must be higher than the text input.

There are two type of system with respect to the way the input is provided to the system. The first being online Optical Character Recognition where the recognition task is performed concurrently with the writing process. The other one is offline Optical Character Recognition

process is performed after the whole text is a scanned and bit map representation of a text is supplied to the system (Wondewosen, 2004).

A number of researches have been done in order to get better recognition of Amharic characters. Among them, Million (2000) has done topology feature-based recognition for different fonts of printed character. Messay (2003) attempts statistical approach for handwritten postal address. Wondewosen (2004) has used Neural Network approach for the recognition Yekum Tsehuf.

2.2.2 Document Image Retrieval without Recognition

The recognition-based approach has some limitations when dealing with documents having a high level of noise or containing multi-lingual text printed with non-standard fonts with a variable layout. This means that document image retrieval works best when the optical character recognition encounters a limited range of type styles, with little or no variation within each style. Where the characters are less predictable, feature, or topographical analysis is superior (Marinai, 2006). As a result, document image retrieval without explicit recognition is proposed as a short-term solution. Figure 2.1 presents the architecture of document image retrieval (Million and Jawahar, 2007).

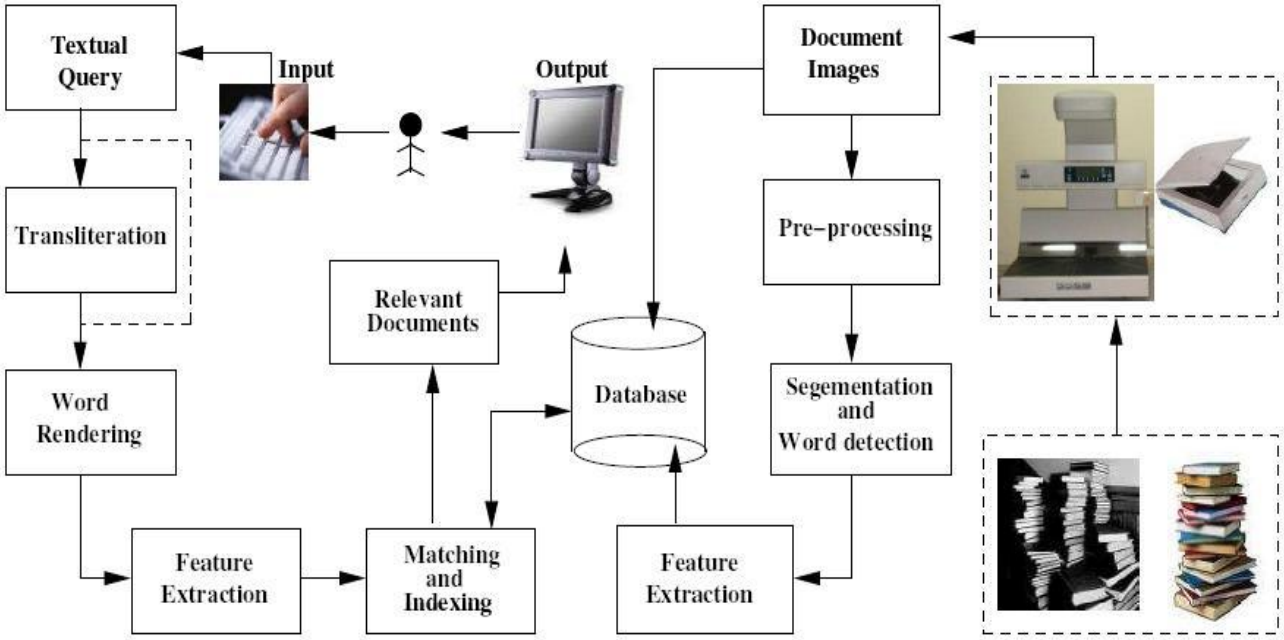

Figure 2.1: Conceptual Diagram of the Searching Procedure

Document images should be preprocessed offline to threshold, skew-correct, remove the noise and thereafter segment into words. Offline preprocess helps to increase efficiency at the time of searching. Word images are also normalized so that word representations become insensitive to variations in size, font and various degradations popularly present in the text documents. For proper search, they need to identify similar words. Distance or similarity between words is computed using the features and documents are ranked according to their similarity (Million and Jawahar, 2007).

2.3 Preprocessing Methods

Digitization is the process of converting information into a digital format, the information can be text, audio, image, video. In this format, information is organized into discrete units of data (called bit) that can be separately addressed (usually in multiple-bit groups called bytes). Text

and images can be digitalized using a scanner, scanner captures an image (which may be an image of text) and converts it to an image file. Document image is preprocessed to make it ready for further processing. The goal of image preprocessing is to increase both the accuracy and interpretability. The preprocessing includes binarization, normalization, noise removal, etc (Young et al., 2007).

2.3 1 Thresholding or Binarization

In the process of analyzing objects in an image, it is essential to distinguish between the objects of interest and "the rest." The technique that is used to find the object of interest is called thresholding (Young et al., 2007).

This technique is based upon a simple concept. A parameter *Ø* called brightness threshold is chosen, brightness is chosen somehow in the middle of not dark or bright and applied to the image a[m,n] as follows (Young et al., 2007):

If a[m,n] > Ø or a[m,n]= Ø	*a[m,n]= object = 1*
Else	*a[m,n]= background = 0*

Algorithm 2.1: Finding fixed threshold

The concern of the algorithm is to identifying light objects on a dark background. The output is the label "object" or "background", which has a representation as a Boolean variable "1" or "0" respectively. The algorithm works good for clean image. For noisy image, there is a need to further preprocess before applying binarization.

2.3.2 Noise Removal and Normalization

Noise removal is the process of restoring blurred pixel from the image. The most frequent noise removal techniques are image filtering by using mean (average) or median filter. In mean filter, the value of each pixel is replaced by the average of all pixel values in a local neighborhood. In median filter, the value of each pixel is replaced by a median value calculated in a local neighborhood. (Jadwiga and Mark, 2001).

Normalization is the process of making the different image to fit to a standard template by identifying commonalities and differences between groups. There is brightness normalization for instance, if the intensity range of the image is 50 to 180 and the desired range is 0 to 255 the process entails subtracting 50 from each of pixel intensity, making the range 0 to 130. Then each pixel intensity is multiplied by 255/130, making the range 0 to 255 (Martin, 2004).

2.4 Image Segmentation Techniques

According to Rahimizadeh, et al. (2009) Image segmentation is the initial step in image analysis and pattern recognition; and image segmentation partitions an image into non-overlapping regions. A region is defined as a homogeneous group of connected pixels with respect to a chosen property. There are different ways to describe homogeneity of a region like color, gray levels, etc.

According to Chakravarty and Mitra (2007), there are about four types of popular approaches for two dimensional image segmentation those are pixel-based segmentation, region-based segmentation, edge-based segmentation and physics-based segmentation.

2.4.1 Pixel-Based Segmentation

According to Koschan and Skarbek (1994), images are assumed to be composed of regions with different gray levels that are partitioned into a number of different levels of intensity, each intensity corresponding to one region. There are two methods of pixel-based segmentation, cluster and histogram methods (Koschan and Skarbek, 1994):

- Cluster method: One of the method used for clustering K-means algorithm. This algorithm works as an iterative technique by partitioning the image into K clusters (Ohlander et al., 1978).

- Histogram method: It is computed from all of the pixels in the image, and the peaks and valleys in the histogram are used to locate the clusters in the image. Color or intensity can be used as measurement (Shapiro and Stockman, 2001).

2.4.2 Area Based Segmentation

This segmentation algorithm uses uniform criteria that are calculated in regions of image domain. It has two type of techniques (Koschan and Skarbek, 1994):

- Region growing: In this class of algorithms, a number of basic homogenous regions are being given and different strategies are applied to join surrounding neighborhoods; to identify this group from the next one it is important that regions here are not resulting from splitting or subdivision processes of heterogeneous regions.

- Split and merge: This algorithms start from heterogeneous regions, subdivide them until homogenous ones are obtained, and then apply some integrating heuristics to fit them to maximal possible homogenous area.

2.4.3 Edge Based Segmentation

The result of edge detection could be an edge image, in which the worth of each pixel reflects how strong the corresponding pixel in the original image meets the requirements of being an edge pixel (Lei et al., 1999).

Extraction of edge-end-pixels is an important step for the edge linking process to achieve edge-based image segmentation, Edge detection techniques have therefore been used as the base of another segmentation technique and the edges identified by edge detection are often disconnected, to segment an object from an image (Pathegama and Göl, 2004).

There are two edge detection techniques, such as local and global techniques. The local technique to determine an edge point needs only information in the neighborhood of that point. The global technique, on the contrary, makes a sort of global optimization, and therefore the given edge point could be identified after many optimization steps involving changes in large areas (Koschan and Skarbek, 1994).

2.4.4 Physics Based Segmentation

Physics based segmentation techniques allow the segmentation of real images based on physical models for image formation. The segmentation technique employs physical models to partition an image into regions that correspond to surfaces or objects in the scene. The objective of this technique is to segment a multispectral image at object boundaries and not at the edges of

highlights and shadows in an image (Koschan et al., 1994). This is a difficult task since the image measurements corresponding to a single surface can have considerable variation due to effects such as highlights, shading, sensor noise, non-uniform illumination, and surface texture (Koschan et al., 1994).

Physics based segmentation is so limited to determine changes in materials whose reflection properties are well known and can be modeled properly. Classification is needed to obtain segmentation results representing surfaces and objects in the scene (Koschan et al., 1994):

2.5 Feature Extraction

Feature extraction is the process of creating a representation, or a transformation from the original data. That means, it is the identification of pixels in an image that have some distinctive characteristics (Hendriks and Reinders, 1999).

According to Lei et al. (1999) features are classified as follows:

- *General features:* According to the abstraction level, they can be further divided into:
    - *Pixel-level features:* Features calculated at each pixel, e.g. color, location.
    - *Local features:* Features calculated over the results of subdivision of the image band on image segmentation or edge detection and it can be extracted directly from the original images
    - *Global features:* Features calculated over the entire image or just regular sub-area of an image and its extraction must be based on low-level features.

- *Domain-specific features:* Application interrelated features such as human faces, fingerprints, and conceptual features.

There are different methods to calculate features. According to Lester (1998) there are four groups.

- *Regularization or Iterative:* image is modeled as a function f(x, y). There is a function, which measures both closeness to the original data and smoothness. This method is performed to reduce noise in an image while preserving the relevant features such as edges. This approach is an iterative method, which involves several free parameters. These parameters control the level of smoothness, edge preservation, and the reliability of the solution to the input data.

- *Multi-scale feature extraction:* This method use neighborhood operators of various sizes for smoothing or polynomial approximation. The larger the scale, the more the image will be smoothed and edges will be displaced from their actual location in the image. However, the smaller the scale, the more noise remains in the image. As Lester (1998) suggests by using multiple scales and combining them, a complete analysis of the features can be performed.

- *Mathematical morphology:* In this approach, morphological operators such as erosion, dilation, openings, and closings are used to detect and classify edge types in range data. Edges are detected by the residues obtained from openings and closings with various shaped morphological operators.

- *Differential geometry* in range images is locating and classifying of geometric properties of the scene by means of estimating the derivatives of the digitized range data points and using these estimations to infer the geometry of the surfaces in the range image. By predicting, the depth values of the adjacent pixels and adjusting the normal based on these adjacent depths. This is the approach which have taken there feature extraction techniques for range images.

Zhou et al. (2002) proposes edge-based structure feature extraction. They use water-filling algorithm to extract edge/structural features. This method uses a linear-time algorithm and this algorithm extracts features from the edge map directly without edge linking or shape representation. They proposes three ways of calculating the feature. The first one is Filling-time. This calculates the time to fill the set of connected edge. The second one is Fork count. This calculates the number of branches the waterfront has forked during the filling of a set of edges. The last algorithm is Loop count. This counts the number of simple loops or simple cycles in a set of connected edges.

According to (Million and Jawahar, 2008) three categories of features are proposed: word profiles, structural features and transformation of domain representations.

- *Word profiles:* Words are represented using a common profile image matching. The profile is done using upper word, lower word, projection, density and ink-to-background transition. Upper and lower word profiles capture some sort of the outlining shape of a word, while projection and transition profiles capture the distribution of data that is, the pixel which has the image along one of the two dimensions in a word image.

- Structural feature: This method helps for matching purpose of two image similarities, and for artifacts like pepper and salt noise where, structural features are found to be very useful. This method employed in their work for describing the structure of the word.

- Transform domain: Using Fourier's transformation a compact representation of a series of observations or they call it profiles can be derived. Among the coefficients few of them can represent a word accurately and these coefficients are compared at a coarse level for matching.

2.6 Document Image Indexing

To speed up searching from a document collection, it is necessary to develop an index, which provides an access to segments in a file. Index is particularly important for certain kinds of computer storage devices such as disks. Disks permit rapid access to consecutive records, but access to particular region of the disk is a slow process. An index file helps to locate the selected region of the disk that contains the record of interest for a given query. Those records can then be scanned using different search algorithms (Salton and McGill, 1983).

In reality, to retrieve a document from a collection without performing indexing, it is time consuming and too expensive. Hence, it is customary to characterize each document by assigning a short description or profile to the document which can be used to obtain access whenever the document is wanted (Salton and McGill, 1983).

2.6.1 Selecting Index Terms

If a full document image representation is adopted then all words in the document image are used as index terms. An alternative method is to adopt a more abstract view in which a set of image terms are selected in order to represent the document (Baeza-Yates and Ribeiro-Neto, 1999).

There are two types of indexing: manual and automatic indexing. Manual indexing takes place when analysis operation has carried out by personal judgment. Automatic indexing is the assignments of the content identifier when it is carried out with the aid of modern computing equipment, without the intervention of human expert in order to create index file in the process (Baeza-Yates and Ribeiro-Neto, 1999).

According to Kowalski and Maybury (2002), there are two major techniques for creation of the index: statistical and natural language. Statistical techniques can be based upon vector models and probabilistic models with a special case being Bayesian models. They are classified as statistical because their calculation of weights uses statistical information such as the frequency of occurrence of words and their distributions in the searchable database. Natural language techniques also use some statistical information, but it performs more complex parsing to define the final set of index concepts.

Most automatic indexing starts with the observation of the frequency of occurrence of individual word type (that is distinct words) in natural language text has something to do with the importance of the words randomly across the content representation. The steps in selecting image word index terms are as follows (Salton and McGill, 1983).

- From a collection of n documents, calculate for each document the frequency of each unique image term in that document. This is the frequency of image term k in document i $FREQ_{ik}$.

- Determine the total collection frequency $TOTFREQ_{ik}$ for each word by summing the frequency of each unique image term across all n documents, that is

$$Total\ Frequency = \sum_{i=1}^{n} FREQ_{ik}$$

Equation 2.1: Total Frequency

- Arrange the image words in decreasing order according to their collection frequency. Decide on some suitable high threshold value and remove all image words with a collection frequency above this threshold. This eliminates high frequency once from image words.

- In the same way, eliminate from consideration a low frequency word that is, choosing some low threshold and remove all words from a collection whose frequency is below the threshold. This deletes image terms occurring so infrequently in the collection whose presence does not affect the retrieval performance in a significant way.

- The remaining medium frequency words are now used for assignment to the document as index image term.

The other type of determining term weight of a document is inverse document frequency (IDF) which is commonly used in Information Retrieval system in order to determine weight of a term in building index file (Sparck, 1972). Inverse Document Frequency is defined as

$$IDF = log_2 N/df_w$$

Equation 2.2: Inverse Document Frequency

Where df_w is the document frequency, which shows the number of documents containing term w, N is total number of document in the collection (Kenneth and William, 1995).

Image words, which are too frequent among the documents in the collection, are not good discriminators. As a consequence, a word that occurs in 80% of the documents in the collection is useless for searching. Therefore, elimination of stop words reduces the size of index structure (Baeza-Yates and Ribeiro-Neto 1999).

2.6.2 Indexing Structure

Generally, in indexing, there is some data structure that helps to store the document images after indexing task is employed. These are inverted file, signature file, suffix tree, etc (Baeza-Yates and Ribeiro-Neto 1999).

2.6.2.1 Inverted file

Inverted file (or inverted index) is a word-oriented mechanism for indexing a document collection in order to speed up the searching task. The inverted file structure is composed of two elements (Baeza-Yates and Ribeiro-Neto 1999). A record level inverted index (or inverted file), which contains a list of references to documents for each word. A word level inverted index (or inverted list) additionally contains the positions of each word within a document (Baeza-Yates and Ribeiro-Neto 1999).

An inverted file consists of two components (Frieder et al.,1999): vocabulary file and posting file. Vocabulary file contains a list of distinct terms, where as posting file contains list of identifiers where the terms exist. Figure 2.2 shows the general structure of inverted file.

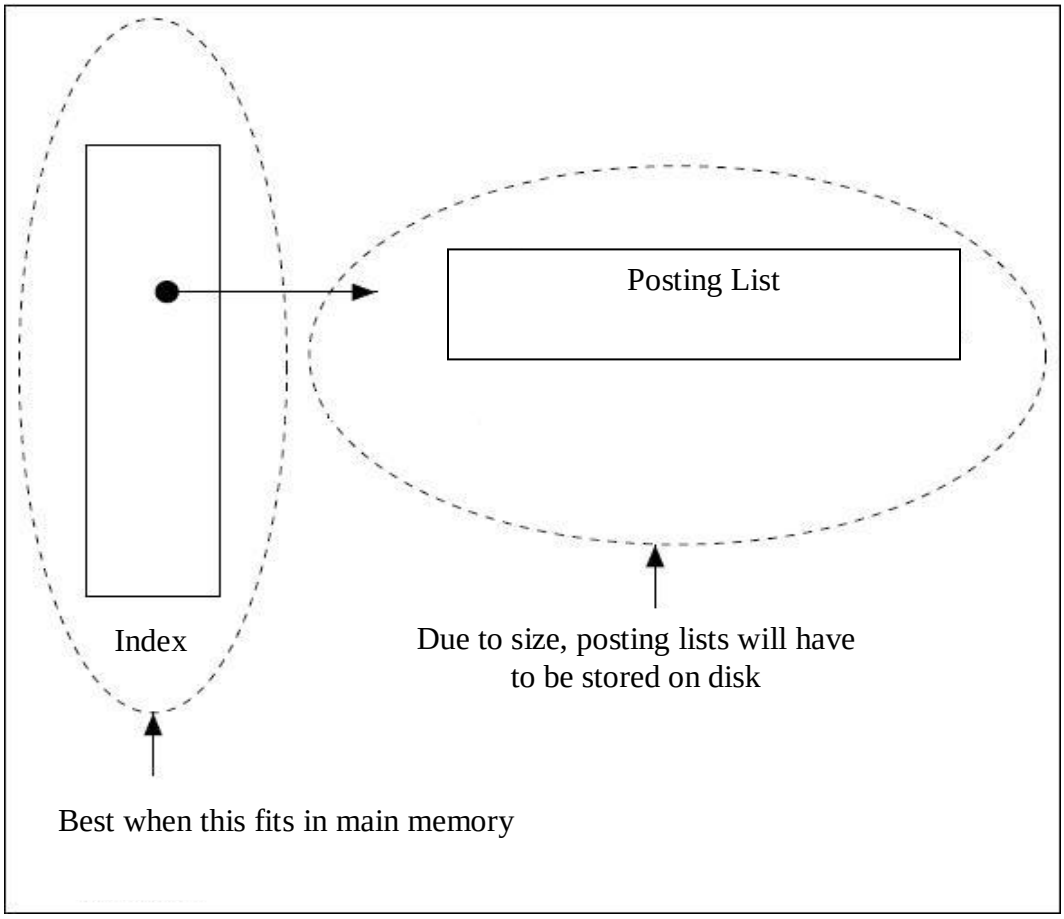

Figure 2.2: Inverted File Structure

Consider a document collection from which a list of terms and their statistical information is captured. Vocabulary file contains the list of terms and their collection, document frequency and pointer to the posting file. The vocabulary grows by $O(n^{\beta})$, where β is a constant between 0 – 1. The posting list, on the other hand is, simply a linked list that is associated with each of the index

terms. Structure of a posting list entry does vary from implementation to implementation. It always includes the document identifier but can also include entries for term frequency, and possibly position data and it needs for the postings pointer an extra space of O(n).

Entries in the posting lists are typically more efficient to insert at the head of the list than in any other location. Clearly, the construction of this inverted index is expensive, but once built, queries can be efficiently executed.

The search algorithm on an inverted file follows three general steps (Baeza-Yates and Ribeiro-Neto 1999). The first is searching from vocabulary file using term and patterns present in the query where phrases and proximity queries are split into single terms. The second one is retrieval of occurrence of all the terms matching with query terms. The third one is manipulation of occurrence, where the occurrence is processed to solve phrases, proximity or Boolean operation. If block addressing is used, it may be necessary to directly search the document to find the information missing from the occurrences.

2.6.2.2 Signature File

The goal of a signature file structure is to provide a fast test to eliminate the majority of items that are not related to a query. Items in document collection are represented in a highly compressed form that facilitates fast searching. Because file structure is highly compressed and disordered, and new items can be concatenated to the end of the structure versus the significant inversion list update. Since items are seldom deleted from index file, it is typical to leave deleted items in place and mark them as deleted. Signature file search is a linear scan of the compressed version of items producing a response time linear with respect to file size. The items that satisfy

the test can either be evaluated by another search algorithm to eliminate additional false hits or delivered to a user to view (Kowalski and Maybury, 2002).

The surrogate signature file is created via superimposed coding computed using hash function. The coding is based upon terms in the item. The terms are mapped into a word signature. A word signature is a fixed length code with a fixed number of bits set to "1". The bit positions that are set to one are determined via a hash function of the terms. The term signature are tied together to create the signature of an item. To avoid signatures being too dense with "1"s, a maximum number of words are specified and an item is partitioned into blocks of that size. Table 2.1 presents how signature file is constructed by hashing term signature and block signature

WORD	Signature	Block Signature
Computer	0001 0110 0000 0110	1001 0111 1110 0110
Graduate	1000 0101 0100 0010	
Science	1001 0000 1110 0000	
Students	0000 0111 1000 0100	
Study	0000 0110 0110 0100	

Table 2.1: Signature file

As shown in table 2.1, the block size is set at five words, the code length is 16 bits. The block signature is computed by ORing term signature of words existing in that block. Signature file

structure has false drop problem, which it is possible that all the corresponding bits are set, even though the word is not there (Kowalski and Maybury, 2002).

2.6.2.3 Suffix Tree

Suffix tree is suffixes of a given string where all nodes with one child are merged with their parents (Andersson and Nilsson, 1995). That means for instance figure 2.3 shows the graphical represention of a word “abebe” in Suffix Tress and the word “abebe” should end with the symbol “$” like “abebe$”. So we have {“$, e$, be$, ebe$, bebe$, abebe$,”} of suffix word.

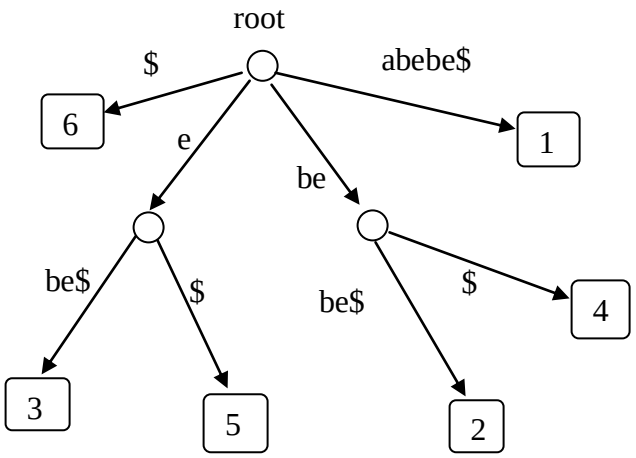

Figure 2.3: Suffix Tree

Based on figure 2.3 interpretation of the Suffix Tree is as follows: if there is $ at the end, it is the end of a word.

To build a suffix tree for a string it needs $O(n^2)$ time complexity, and to search in suffix tree it needs a linear length of the string size. It takes a lot of storage space to store the words (Andersson and Nilsson, 1995).

2.7 Document Image Matching Methods

Document retrieval is all about matching of some stated user query against useful parts of document images (Ronald et al., 1996). There are different types used to measure similarity between images (Nguyen et al., 2009).

2.7.1 Dynamic Time Warping

Dynamic Time Warping is used to compute a distance between two time series. A time series is a list of samples taken from a signal, ordered by the time that the respective samples were obtained (Nguyen et al., 2009).

Dynamic Time Warping (DTW) performs discrepancy between intuition and calculated matching distance by recovering optimal alignments between sample points in the two time series. The alignment is optimal in the sense that it minimizes a cumulative distance measure consisting of "local" distances between aligned samples.

The distance between two time series $X_1 \ldots X_M$ and $Y_1 \ldots Y_N$ is calculated in a dynamic programming approach (Million, 2008; Nguyen et al., 2009), using the following formula.

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

Equation 2.3: Local continuity constraint

The particular choice of recurrence equation and "local" distance function d (·, ·) varies with the application. Using the given three values D(i, j − 1), D(i − 1, j) and D(i − 1, j − 1) in the calculation of D(i, j) realizes a local continuity constraint.

According to Nguyen et al. (2009) Dynamic Time Warping algorithm can be run to determine a warping path between X and Y. The length K of the warping path ($(i_1, j_1), \dots, (i_K, j_K)$) biases the determined distance

$$D(X,Y) = \sum_{k=1}^{K} d(x_{ik}, y_{jk})$$

Equation 2.4: Dynamic Time Wrapping

The words are similar if the cost approaches to zero or dissimilar if the cost is near 1 (Nguyen et al., 2009).

2.7.2 Cosine Distance

According to Dengsheng and Guojun (2003) the cosine distance computes the difference in direction. The distance is given by the angle between two vectors; 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. Therefore, the similarity between the query document image Q and the archived document image Y is computed as follows (Tan, 2004):

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

Equation 2.5: Cosine Distance

Where, q is the document vector of the image Q, and y is the document vector of the image Y. n is the dimension of the query and document vector (Tan, 2004) .

It is said to be similar when the cosine distance approaches to '1' or different when it approaches to '0' (Tan, 2004).

2.7.3 Euclidean Similarity

The Euclidean similarity score between two image words is defined as their difference by their sum, a difference is calculated through summing up the products of the corresponding elements and Euclidean similarity value of zero or very near to zero means that the query and word images are similar (Haque et al., 2005). If the Euclidean similarity value is one very near to one, it means the word image is dissimilar. Euclidean distance formula is given as follows,

$$d_{\text{Euclidean}}\ (T, Q) = \frac{\sqrt{\sum_{i=0}^{m-1}(t_i - q_i)^2}}{\sum_{i=0}^{m-1} t_i + \sum_{i=0}^{m-1} q_i}$$

Equation 2.6: Euclidean distance

Where, t_i is the vector value of the word image T at i^{th} position and q_i is the vector value of the query word image Q at i^{th} position (Haque et al., 2005):

2.8 Evaluating Retrieval System

Interest in the evaluation techniques for information retrieval systems has significantly increased with the commercial use of information retrieval technologies in the everyday life of the millions of users of the Internet. In recent years, the evaluation of information retrieval systems and techniques for indexing, sorting, searching and retrieving information has become increasingly important (Manning et al., 2008). This growth in interest is due to two major reasons, the growing number of retrieval systems being used and additional focus on evaluation methods themselves (Manning et al., 2008).

When we see evaluation from an academic perspective, measurements are focused on the specific effectiveness of a system and usually are applied to determining the effects of changing a system's algorithms or comparing algorithms among systems.

According to Manning et al. (2008), to measure information retrieval effectiveness in the standard way, we need a test collection consisting of three things:

- A document collection of text, image, audio and/or video
- A test suite of information needs, expressible as queries
- A set of relevance judgments that is, a standard binary assessment of either relevant or non-relevant for each query–document pairs.

With respect to a user information need, a document in the test collection is given a binary classification as either relevant or non-relevant (relevance judgment) before evaluation takes place.

Any information retrieval system requires the evaluation of how precise is the answer set to the given query, and the evaluation to be considered depends on the objective of the retrieval system. Any software system has to provide the functionality it was conceived for. Thus, the primary type of evaluation that should be considered is function analysis in which the specified system functionalities are tested one by one. After evaluating the functional analysis, the performance evaluation is then proceed (Baeza-Yates and Ribeiro-Neto 1999).

There are two major types of performance evaluation: Effectiveness and Efficiency measurement (Frieder et al., 1999).

Efficiency measures how fast a result is obtained within a given time period during searching and indexing. Searching time is the required time that takes to respond a result after a query is provided to the system, where as indexing time is the time taken to build the index file. This is computed using standard algorithmic complexity analysis (e.g., the "Big Oh" notation) or more empirically measured with statistics such as response time. Most of the time effectiveness is given more attention than efficiency (Frieder et al., 1999).

2.8.1 Effectiveness Measurement

Any retrieval system should accept a query from a user and provide a result according to their relevance in a sorted or ranked way to the user. Relevance of ranked result to the given query is measured to calculate effectiveness of the system. The most important measures are recall, precision and F-measure (Mandl, 2007; Baeza-Yates and Ribeiro-Neto, 1999). Before measuring effectiveness of any system, we need to determine relevant documents to the query.

Consider in a collection, r is relevant document that is retrieved, R is relevant document that is not retrieved, m is irrelevant documents that are not retrieved, and N is total collection of documents as shown in figure 2.4.

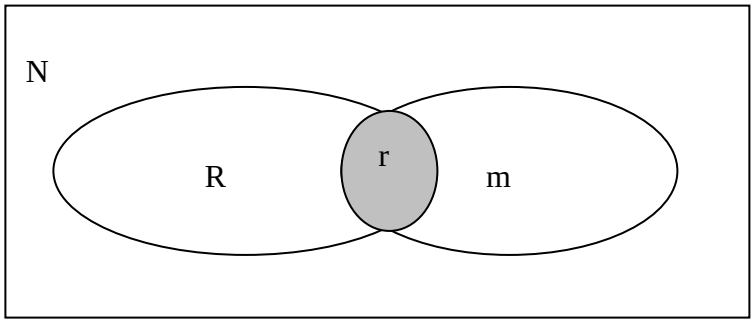

Figure 2.4: Retrieval scenarios for a specific query

Recall is the fraction of relevant documents that is retrieved from the total number of relevant document in the collection; it is defined as Baeza-Yates and Ribeiro-Neto (1999).

$$Recall = \frac{r}{r + R}$$

Equation 2.8: Recall

Precision is the fraction of relevant document retrieved from the total number of documents retrieved. It is defined as: (Baeza-Yates and Ribeiro-Neto, 1999).

$$Precision = \frac{r}{r + m}$$

Equation 2.8: Precision

Recall indicates the ability of a system to find relevant documents and it requires knowledge of all the relevant documents in a collection. The number of known relevant documents is usually used to calculate the value at each position. For instance a query is provided to the system and results are listed in according to their ranking position and if all the relevant document are retrieved at any position then the Recall value will be one. On the other hand, if all are non-relevant result to the query provided, then recall will be zero.

Precision measures how good a system is in finding only relevant documents without ballast. Precision shows how well the system retrieves only relevant data to the provided query. For instance, if the system provides search results in ranked order that are all relevant to the query, then its precision value will be one. Nevertheless, among the result, if all are non-relevant to the query provided, then precision will be zero.

There is an inverse relationship between recall and precision. If you try to increase the recall value, precision of the system tends to decrease and if you try to increase precision, recall of the system decrease. Therefore, it is difficult to identify cut point by considering only one of them, then the two metrics are often combined as their harmonic mean, known as the *F-measure*, which can be formulated as follows (George and Adam, 2005):

$$F - measure = \frac{2 * (recall * precision)}{precision + recall}$$

Equation 2.9: F-measure

The maximum F-measure is selected for the threshold.

2.9 Attempts on Document Image Retrieval

Nowadays, attempts are being conducted in the field of document image retrieval, especially in the application of OCR for document retrieval. If OCRs are not available, then scanned images will be inaccessible. Million and Jawahar (2008) proposed that it is possible to search a document without an intermediate textual representation. The following are some of the noticeable work in document image retrieval.

2.9.1 Word Shape Recognition for Chinese Image-Based Document Retrieval

Weihua et al. (2002) propose a word shape recognition method for retrieving Chinese image-based documents. First, they segmented the document images at the word level, and then they detect local maximum and minimum points in word segments to form vertical bar patterns. These vertical bar patterns form the feature vector of a document. Scalar product is used to measure similarity of between two document image feature vectors. The proposed method is

insensitive to change occurring from fonts and styles variation. They recommend further study of language independent system should designed.

2.9.2 Document Image Retrieval Techniques for Chinese Newspapers

Tseng and Douglas (2004) present experimental results for retrieval from a collection of scanned article clippings from Chinese newspapers. The byte length normalization is found to be performing good result over cosine normalization and indexing of a combination of unigrams and bigrams perform better result over individual indexing. They recommend a differential handling for document with many recognition errors.

2.9.3 Searching in Document Images

Million and Jawahar (2008) proposed a technique for retrieval from document images without an intermediate textual representation; They achieve effective retrieval from Amharic document databases by matching at word-level using image features. For characterizing the word images they undertaken word profiles, structural features and transform domain representations. They attempt to apply partial matching approach based on dynamic time warping to take care of word form variations. They recommend handling of word variants needs to be improved and fast searching.

2.9.4 Amharic Document Image Retrieval without Explicit Recognition

Mesfin (2009) has done a research on Amharic document image retrieval without using Optical Character Recognition. The researcher uses a gray scale image quality, and a fixed threshold value to segment the image from the background. During document image segmentation, first the

line is segmented to identify the lines that hold document image word and then image words are identified.

During searching the system accepts query and then the text query is converted to image (query rendering) in order to be compared with the set of document images. To represent word image the researcher extract an image feature based on the pixel values in the predefined area. The idea is to calculate vector value for the image and represent each word image in one vector. Then feature extracted is used for matching using Euclidean distance to analyze the similarity of word images. The performance of the system is 82% precision and it takes 22 second for searching from 483 documents. As the number of documents increase, it requires more time to search and retrieve relevant documents, which makes the system inefficient and they recommend enhancement of efficiency and effectiveness in order to make better system.

Hence there is a need to design an efficient indexing scheme in the image domain. In addition, a good retrieval system should not be affected by word variants during searching. This also requires to detect word variants for a given query term and accordingly design a strong stemming algorithm that groups together word variants into one cluster. Therefore, the present research attempts to investigate such issues to come up with a better search engine for Amharic document images retrieval.

Chapter Three

The Amharic Writing System

3.1 Introduction

Every language has its own writing system and every sound is represented by some set of strokes. These combinations of strokes make up the character set of the language. The character sets or symbols used to represent the sounds of a language are called scripts of that language. After formulating and being modified through the set of characters through time, the speakers of the language will get used to these symbols so that they can use the written language approach to communicate as well as to preserve knowledge (Bethlehem, 2002).

3.2 Amharic Language

In Ethiopia, more than 80 languages are used in day-to-day communication. Among them Amharic is the dominant one in that it is spoken as a mother tongue by a number of the population and it is the most commonly learned second language throughout the country. It is also the official language of the country and medium of communication and working language in most of the regional states (such as Addis Ababa, Amhara, etc) (Tessema et al., 2010).

Amharic is one of the laguage which has it own writing system. The language was adpated from the Ancient Ge'ez writing system. Ge'ez language belongs to the class of sematic language, which was drived from the south Arabian alphabet called sabaean. Figure 3.1 depicts the genetic structure of the Amharic language (Bender et al., 1976; Yaregal, 1976)

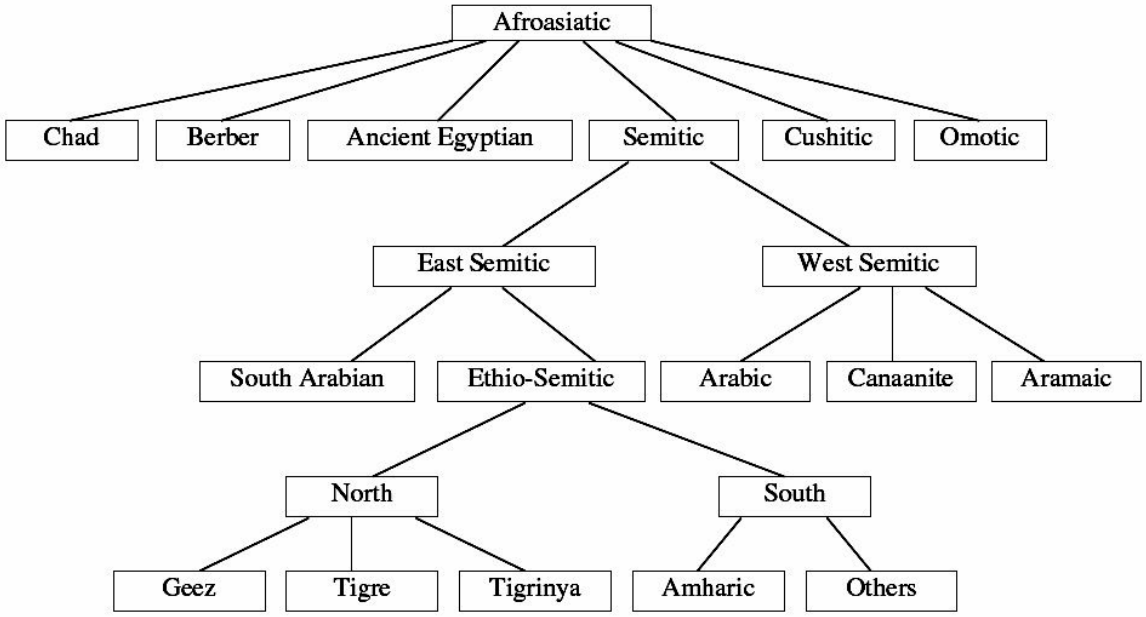

Figure 3.1 The Genetic structure of Amharic language:

Amharic writing system took all the symbols from Ge'ez writing system and it adds some new symbols to the writing system. Sabaean is not used currently, where as Ge'ez is still used especially as a language in liturgy of the Ethiopian orthodox and in church literature (Bender et al., 1976).

The Sabaean alphabet has twenty nine symbols, out of which Ge'ez took twenty four. In Ge'ez two new symbols where created to represent sounds of Greek and latin loan words. When Ge'ez was abondoned as the spoken language, other languages like Tigrignya and Amharic came into the society by adding new symbols in addition to the existing Ge'ez alphabets (Tessema et al., 2010; Bethlehem, 2002). The new symbols added in the Amharic script are ሸ(š),□ (z') ቸ(c) ኘ (n') ዸ (c') ቨ(v) ኀ(h£) □ (j). Amharic language is written from left to right and there is no Capital or Lower case distinction between characters (Tessema et al., 2010).

3.2.1 The Alphabets

Amharic writing system consists of thirty three characters (fidel) as a core characters. The thirty three characters occur in one basic form and in six other forms know as orders, as shown in table 3.1. These orders are drived from the basic forms by more or less regular modification (Hudson, 2001). The seven orders of the Ethiopic represent the different sounds of a consonant-vowel combination known as syllabic. The 33 core characters produce 231 distinct symbols and 51 labialized characters. The complete Amharic symbols are found in the Appendix 2.

1st order	2nd order	3rd order	4th order	5th order	6th order	7th order
ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
Hä	hu	Hi	ha	He	h	Ho
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
Lä	lu	Li	La	Le	l	Lo

Table 3.1 The Seven order of Amharic writing system

Each symbol represent a consonant together with its vowel. The vowels are fused to the consonant form in the form of diacritic markings. The diacritic markings are strokes attached to the base characters to change their order(e.g the first order le "ለ") is transformed into the second order symbol lu "ሉ" by attaching "ु" to the right middle of it; the same first order is transformed into the third order symbol li "ሊ" by attaching "ि" to the right end of it; and so on. This means the Ethiopic does not use independent symbols for vowels in representing a syllable(Bender et al., 1976).

3.2.2 The Numbering System

The basic numbering system in a Amharic consists from one to ten and a combination of this numbers will produce other numbers. This characters are drived from Greek letters, and some were modified to look like Amharic fidel, for instance each of the symbols has a horizontal stroke above and below (Bender,1976). There is no symbol for zero in Amharic script. Arthimetic computations using the Amharic symbols are very difficult. As a result, most of the time Hindu-Arabic numerals are used, where as Ethiopic numerals are used in writing dates and Page numbers in text (Bethlehem et al., 2002).

3.2.3 Punctuation

In Amharic writing system there are a number of symbols for punctuation. According to Betetu (1982), as quoted in Zelalem (2001), there are about 17 punctuation marks and only some of them are commonly used. Among them Two square arranged like a colon (: hulet netib) are word delimiters. The equivalent of the full stop is four dots arranged as for dots (:: arat netib). The equivalent for comma is (netela serez) and the semi-colon (and dirib serez). There are some borrowed symbols like (?, !, “, ”, ‘, /, \, +, -, *, etc) but the most common used symbol in handwriting is (: hulet netib). Where as in computer writing style it is common to use a space as a word separator instead of using (: hulet netib).

3.3 Problems in Amharic Writing System

In Amharic language, a number of problems are observed with the writing system that have an effect to produce words with structural variation, though they have the same meaning and sound. The major problems are summarized below.

3.3.1 Redundancy of Consonants in Different Forms

In Amharic Alphabets there are some different symbols having the same pronunciation (sound). Although in Ge'ez language, these different symbols give each word different meanings, in the Amharic language they are used interchangeably (Rewords et al., 2003; Bender et al., 1976). The presence of these redundant characters with the same sound in the language creates problem, especially in terms of matching words with retrieval systems. Literally different words can be formed by combining the different forms of the same sound character, For instance the same word 'tsehay' □ሀ□, can be written differently using the variant characters of '□' and 'ሀ' as □ሃ□, □ሀ□, □ሐ□, □ኀ□, □ሓ□, □ሃ□, □ሐ□, □ሓ□, □ኀ□, etc (Nega and Peter, 2002; Tessema et al., 2010).

The class of symbols with the same sound falls into two. The first class includes characters with the same sound for the first and fourth order. These are ሀ and ሃ, ሐ and ሓ, አ and ኣ, ወand □. The second class includes characters with different alphabets that share the same sound. These characters are ሀ, ሐ and ኀ, ሰ and ሠ, አ and ዐ, □and □. This reduncancy has been recognized in literature as a problem of the language (Bethlehem, 2002; Zelalem, 2001), which is also a challenging task during retrieval system design.

3.3.2 Multiple Writing Systems for Single Amharic Word

When foreign terms are translated in to Amharic, different spellings may be used as varied as the number of possible pronunciations. For example, the word "television" may be translated as "ቴሌቭዥን" (television) or "ቴሌቪዥን" (television), and the word 'electric' as "ኤሌክትሪክ" (electric) or "ኤሌትሪ□" (eletiric). Such different transilations usually result in text, especially

when using loan words (words borrowed from other languages and that do not have their own transilation in Amharic)(Bender 1970; Zelalem, 2001).

3.3.3 Compound Words

Tessema et al. (2010) indicates compound nouns are sometimes written as two separate words and have the same meaning with the single word. The word 'kitchen' can be written as either "□ዓ ቤት" or "□ዓ ቤት" and the word 'temple' as "ቤተ መቅ□ስ" and "ቤተመቅ□ስ" (Zelalem, 2001).

Occasionally, the constituent terms may have completely different meaning from the compound word formed from them. For example, the word 'hode-sefee' ሆ□-ሰ□ which means 'tolerant' has different meaning from the constituent terms 'hode' which means 'stomach' and 'sefee' which means 'wide'. During searching the constituent terms of the compound noun are considered as independent and a document that contains one of these terms is treated as relevant. This phenomena result in retrieval of irrelevant materials for a query which contains one of the constituent terms (Dumias, 1992; Deerwester et al., 1990).

3.3.4 Abbreviations

In Amharic writing system there is no consistency of spelling abbreviations. For example, when abbreviating the phrase ዓመተ ምህረት (in the year AD), one can find □ም., □ም or □ም ; አዲስ አበባ (Addis Ababa the capital city of Ethiopia) is also written as አአ, አ.አ abbreviations. The use of hyphen is not consistent through out the writing system (Zelalem, 2001).

3.4 Amharic Word Formation

Amharic language is one of the most inflectional languages; in which a given word can have several word variation. Affixing is used to derive nouns by adding prefixes, infixes or suffixes to basic nouns, adjectives, verbs, stems and roots. It is obvious that nouns are most important for retrieving relevant documents for a given query (Nega and Peter , 2002; Zelalem, 2001).

3.4.1 Suffixes

There are different kinds of suffix that can be attached to a word. To list some of them **ን**, **ም**, **ና** etc (Zelalem et al., 2001). Their attachment is made to show possession (**ን**), emphasis (**ም**), and object marker (**ና**). However, this attachment to the root word does not change the original meaning of the word (Nega and Peter, 2002).

One or more of the above suffixes can be attached to a word in a number of ways. Table 3.2 presents an example of the possible combination of suffixes to a word.

Before Adding Suffixes	After Adding Suffix
አዋጅ + ን	**አዋጅን**
አዋጅ + ን + ና	**አዋጅንና**
አዋጅ + ን + ም	**አዋጅንም**
አዋጅ + ና	**አዋጅና**
አዋጅ + ም	**አዋጅም**

Table 3.2 Sample Suffixes attached to the word **አዋጅ**

3.4.2 Prefixes

Prefixes are characters that are attached to the starting point a word. To list some of them are **በ**, **ለ**, **ከ**, **□ ስለ** etc (Zelalem, 2001).

Prefixes such as **አ** 'I'-, **ት** 't'-, and □ 'y'- are used for the first, second, and third person future forms and there are a number of prefix which can be added on the starting character of the word (Nega and Peter, 2002). Table 3.3 shows some examples of how prefixes are attached to a word that have the same meaning to the root word.

Before Adding Suffixes	After Adding Suffixes
በ + ሉዋጅ	**በሉዋጅ**
ለ + ሉዋጅ	**ለሉዋጅ**
ከ + ሉዋጅ	**ከሉዋጅ**
□ + ሉዋጅ	**□ሉዋጅ**
□ + ሂ□	**ስለሂ□**
ት + ሂ□	**ትሂድ**
አት + ሂ□	**አትሂድ**

Table 3.3 Sample Prefixes attached to the word **ሉዋጅ** *and* **ሂ□**

3.4.3 Infixes

Infixes occur by modifying or adding some characters at the middle of the word root. To list one of them "ም" (Nega and Peter, 2002). Table 3.4 presents same example of infixes derivation of a root word.

Before Modification	After Modification
ለበሰ + ባ	ለባበሰ
አንበሳ + ናብስ	አናብስ
ነገረ + □	ነጋገረ
□ለ□ + ላ	□ላለ□

Table 3.4 Sample Infix attached to the different words

An investigation of the features of Amharic writing system and word formation helps for select proper algorithms for stemming word variation and identifying stopwords in the course of indexing and searching document images.

CHAPTER FOUR

Document Retrieval Techniques

4.1 Introduction

Document Retrieval is a computerized process of retrieving relevant documents in ranked order in response to an inquirer's request (Liddy, 2004). In order to achieve document retrieval system there are different techniques integrated together. In the present study, algorithm for stemming, indexing and stopword detection are carefully designed to achieve the intended objective.

4.2 Similarity Measure

In this study, cosine similarity measurement is used to compare the degree of similarity between two word images. Given the normalized feature vector of two word images $q_1, q_2, \ldots, q_n$ and $p_1, p_2, \ldots, p_n$ cosine similarity is compared as using algorithm 4.1.

1. *Read the image vectors from a file*
2. *While end of file*
   A. $S(\bar{Q}, \bar{Y}) = \frac{\sum_{k=1}^{n} q_k \cdot y_k}{\sqrt{\sum_{k=1}^{n} q_k^2 \ \sum_{k=1}^{n} y_k^2}}$

//whrere q_k is the n^{th} firstimage vector value and y_k is the n^{th} secondimage vector value

Algorithm 4.1: Determining cosine similarity between two images

4.3 Stemming

Morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of IR applications. For this reason, a number of stemming Algorithms, or stemmers, have been developed, which attempt to reduce a word to its stem or root form (Nega and Peter, 2002). In Amharic writing system, most of the word variations are formed by adding extra characters as suffix and/or prefix.

4.3.1 Prefix Detection

Prefix detection is the process of identifying characters attached at the beginning of the root word Nega and Peter (2002). By using the length of the root word image, we can identify the prefix attached to the root word. The following algorithm is used to detect prefix of a word.

1. *While end of file read the image vectors*
    - A. *Select two image vectors*
    - B. *If size of image one is greater than size of image two, then minimum is assigned the size of image two*
    - C. *else minimum is assigned the size of image one*
    - D. *For n starts from minimum minus one down to zero do*
        1. *Calculate cosine similarity // Algorithm 4.1*

Algorithm 4.2: Detecting Prefix in a word

4.3.2 Suffix Detection

Suffix detection is the process of identifying the suffix character attached at the end of the root word Nega and Peter (2002). Also using the length of the root word image, we can identify the suffix attached to the root word. The following algorithm is used to detect suffix of a word.

1. *While end of file read the image vectors*
   - A. *Select two image vectors*
   - B. *If size of image one is greater than size of image two, then minimum is assigned the size of image two*
   - C. *else minimum is assigned the size of image one*
   - D. *For n starts from zero to minimum minus one do*
     1. *Calculate cosine similarity //Algorithm 4.1*

Algorithm 4.3: Detecting Suffix in a word

The system compares up to the end of the minimum image vector length and calculates the similarity score between the two words. If both the suffix and the prefix detection scores greater than the similarity threshold, the word with larger vector is word variant; otherwise, it is unrelated word.

Challenge

4.4 Stopword Removal

This step is used to remove stopword images before building the index file. In this, weight of the indexed term images are calculated using the inverse document frequency. The inverse document frequency is calculated using algorithm 4.4.

1. $Cutoff = log_2 N/(N * .8)_w$ // Calculate the cutoff value
2. *For each clustered word images*
   A. $idf_w = log_2 N/df_w$

// Where N is total number of documents and df_w *is number of document frequency that contain the* w^{th} *term*

Algorithm 4.4: Calculating the inverse document frequency

After inverse document frequency of the term image and cutoff value is calculated, then a comparison is performed between inverse document frequency of a word image and the cutoff value. If the cutoff value is greater than the word image, then the word image is considered to be stopword and it is not taken to the index file, otherwise it is taken to the index file. Algorithm 4.5 presents comparison of inverse document frequency and the cutoff value.

1. *For i starts from zero to total number of terms do*
   A. Caluculate IDF Weight //algorithm 4.4
   B. If cutoff is greater than IDF for the i^{th} *term clustered image*
      1. *Do nothing*
   c. else Write the term image to the index file

Algorithm 4.5: Stopword detection and removal

4.5 Indexing

Indexing is the process of identifying and creating index file by clustering similar word images together into one group. In this study, inverted index structure is followed to organize index terms. The following algorithm is followed in constructing inverted index file.

1. *Read the image vectors from a file*
2. *While q starts from the first word to last word do*
   - A. *While p starts from q + 1 to last word do*
     - 1 *Caluculate similarity of image vector q and p// Algorithm 4.1, 4.2 and 4.3*
     - 2. *If similarity is greater than ø, then*
       - a. *If size of image vector one is less than size of image two, then take image one as a root word*
       - b. *else take image vector two as a root word*
       - c. *get document id*
       - d. *increment tem frequency*
       - f. *increment cumulative frequency.*
       - g. *If the document is occurred for the first time, then increment document frequency,*
   - B. *Remove stop word //Algorithm 4.5*
   - C. *Store the root image vector term including document frequency, cumulative frequency, document id, term frequency.*

Algorithm 4.6: Inverted file structure construction

After the suffix and prefix is detected using the cosine similarity measurement, the shorter width term image is stored in the index file, which is implemented using the inverted file data structure method. The inverted data structure consists of the term itself, term frequency, document frequency, document identification.

4.6 Document Retrieved

Ranking is used to sort the retrieved documents according to their degree of relevance to the query provided by the users.

To rank the retrieved document we use TF*IDF weighted scheme. The reason why the researcher selects TF*IDF is, if you take term frequency (TF) for weighting documents, it only consider the frequency existence of the term in the document. Term frequency gives more weight to term with high frequency of appearance. With this weight we don't know in how many documents it appears.

On the other hands, if you take inverse document frequency (IDF) for weighting term images, you only consider frequency appearance of the term across the documents; but you don't know the term frequency of the document in one document.

So, it is better suggested to use term frequency * inverse document frequency (TF*IDF). At this time, the TF*IDF weight considers both term frequency in one document or page and document frequency across the documents or pages. TF*IDF is calculated using the algorithm 4.7.

1. *Read and buffered the index image vector*
2. *While end of indexed image vector*
   A. *Measure similarity //using Algorithm 4.1, 4.2 and 4.3*
   B. *If similarity is greater than ø, then break the loop and retrieve the index term image vector*
3. *While end of the document frequency of the retrieved index term calculate*
   A. $IDF = log_2 N/df_w$ *// calculated inverse document frequency of the retrieved document*
   B. *Calculate Term Frequency*Inverse Document Frequency Weight of the individual document terms*

// Where N is total number of documents and df_w *is number of document frequency that contain the* w^{th} *term*

Algorithm 4.7: Calculating weight of retrieved pages

Using TF*IDF weight the retrieved documents are sorted in ranked order using the following algorithm 4.8.

1. *For m starts from zero to the end of document frequency of the retrieved index term*
   A. *For k starts from m +1 to the end of document frequency of the retrieved index term*
      1. *If the weight of* k^{th} *image vector is less than to the* m^{th} *weight, Swap the two image terms*
2. *Display the result*

Algorithm 4.8 :Document retrieval

Identifying the algorithm required to construct a search engine in the image domain is crucial. Once the algorithm are identified and clearly designed, Java code is written to implement the indexing, stemming and stopword detection modules.

CHAPTER FIVE

EXPERMENTATION

5.1 Introduction

As a continuation research of Mesfin (2009), the present work concentrate on enhancing the efficiency and effectiveness of the system. This chapter briefly describes the outcome of the experimentation that took place in implementing document image searching system. Discussion of each stage of the development process described along with the respective result achieved and the difficulties faced.

5.2 Architecture of the Proposed Document Image Searching

Mesfin (2009), attempts to design document image retrieval without explicit recognition. On top of that, the present study further investigates techniques for enhancing document image searching. The architecture of the system is depicted in Figure 5.1

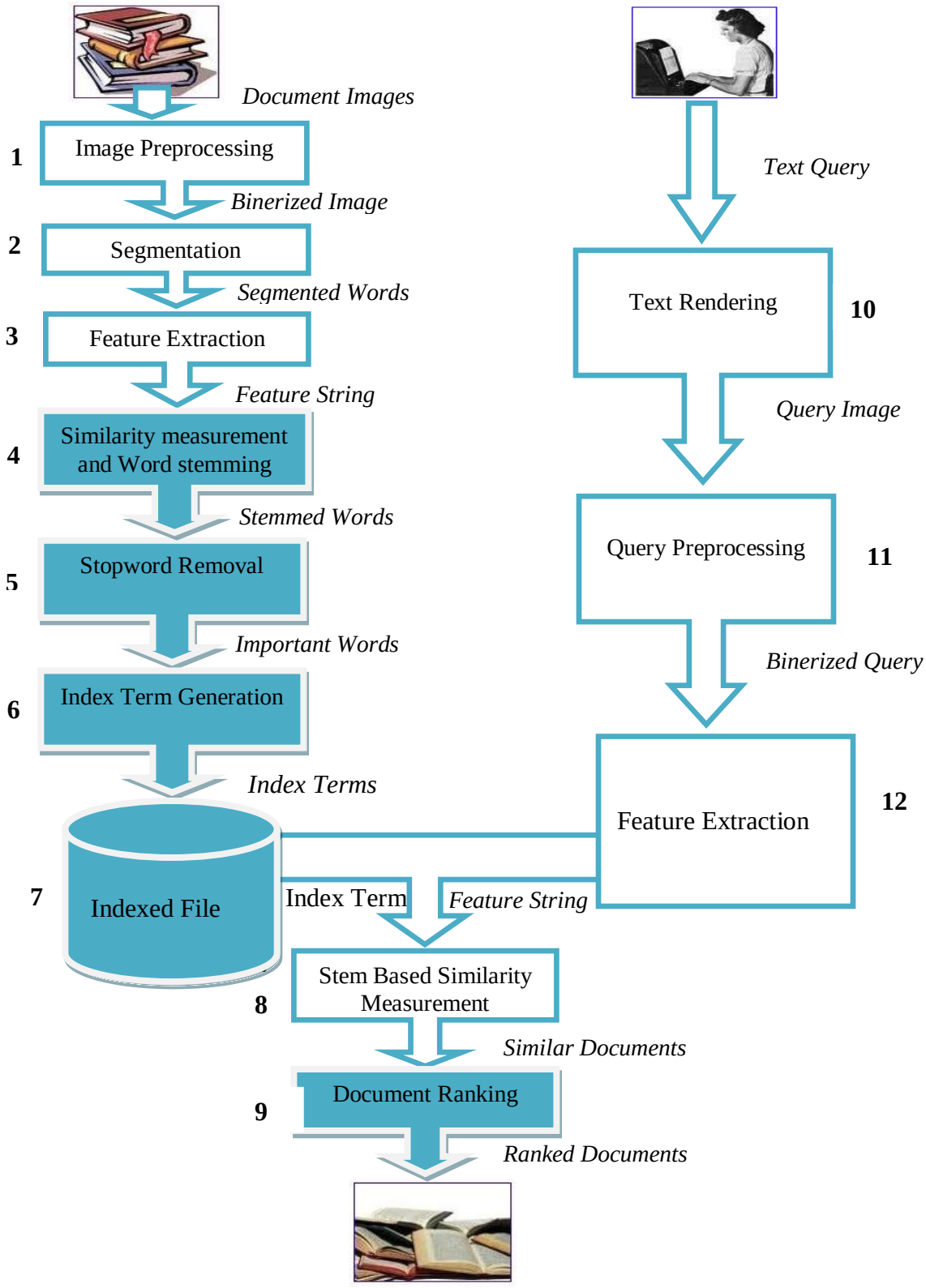

Figure 5.1: Architecture of the proposed document image retrieval system.

Figure 5.1 shows the architecture of document image retrieval system and those with gray color boxes are the contribution of the present study.

The system start by preprocessing digitized image corpus. The preprocessing stage creates binerized image. Images are then segmented in order to get the boundary of the word images. Feature extraction is followed in order to represent word images. This list of word images are indexed to ease searching. Before indexing word variants are grouped together stopwords are removed. Those distinct terms are finally organized using inverted index file to make them ready for searching. From the query side, the user enters a query to the system then the query text is converted to image using text rendering, the query is binerized, then feature is extracted in order to identify the query image word. Matching between the query word and the indexed term image is performed to identify similar word from index term after identifying relevant documents, they are displayed in ranked order for the user.

5.3 Preprocessing

The first step in document image searching process is acquisition of digital image. There are different kind of document capturing devices, including digital cameras, scanners, etc. In this study, flat-bed scanner is used to digitize document at 300 density per inch(dpi) with grayscale color format.

The document images are binarized to identify the foreground and background pixels of the image. The algorithm assigns ‘1’ for foreground and ‘0’ for background.

After binarization is performed on the digital images, the next step is segmenting the document image in to a set of word images. The segmentation algorithm has two phases: line segmentation

followed by word segmentation. Line and words in a document are identifies using horizontal and vertical boundary segmentation.

After word image is segmented from the documents, the next step is extracting the features of the individual word image. To extract feature, word shape analysis of vertical bar pattern is performed. The extracted feature uniquely identify each of the word in our collection.

5.4 Similarity Measure

In this study, the image similarity is measured using cosine, and it produces a value between zero and one. Figure 5.2 shows the Java code used to measure the similarity between two terms.

```
for(int i=0; i < featurelength-1; i++){
    datatemp = Double.parseDouble(data[i]);
    datacomparetemp = Double.parseDouble(comparedata[i]);
    dotproduct = (datatemp * datacomparetemp) + dotproduct;
    sqrdata = (datatemp * datatemp) + sqrdata;
    sqrdatacompare = (datacomparetemp * datacomparetemp) + sqrdatacompare;    }
suffixsimilarity = dotproduct / Math.sqrt(sqrdata * sqrdatacompare);
```

Figure 5.2: Measuring similarity using cosine

Here the main challenge is measuring similarity of different type of formats like Bold, Italic and size produce dissimilarity result while they are similar in meaning.

5.5 Stemming Word Images

In Amharic language most word variants are formed by adding suffix and prefix to the root word. Grouping such words together helps to increase recall of the system. In this study an attempt is

made to detect suffix and prefix of the word image. This is done during matching feature of the word image. To match a pair of word images cosine similarity measurement is used.

5.5.1 Suffix Detection

The suffix removal first checks the size of two vector images then a short length image vector is selected. The reason of selecting the short size image vector is that, the longer size image vector is compared until the end size of a shorter image vector.

For example, to compare two word images, say "**አዋጅ**" with "**አዋጅን**", their vector size is compared. Since vector size of "**አዋጅ**" (95) is less than the size of "**አዋጅን**" (128) as shown in figure 5.3, the size of "**አዋጅ**" is used to control the similarity measure between "**አዋጅ**" feature vector and "**አዋጅን**".

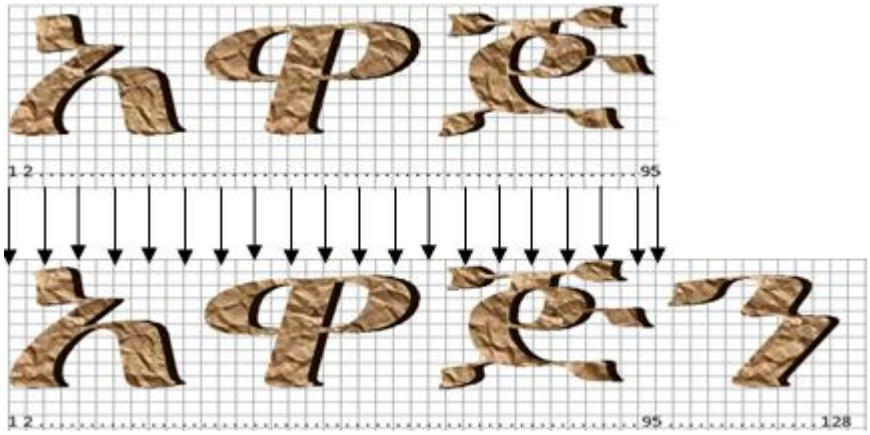

Figure 5.3: Suffix detection of two images vector size

The Java code shown in figure 5.4 is written to detect suffix.

```
If (data.length >= comparedata.length)
        featurelength = comparedata.length;
 else
     featurelength = data.length;
For (int i=0; i < featurelength-1; i++)
 {
     datatemp = Double.parseDouble(data[i]);
     datacomparetemp = Double.parseDouble(comparedata[i]);
     dotproduct = (datatemp * datacomparetemp) + dotproduct;
     sqrdata = (datatemp * datatemp) + sqrdata;
     sqrdatacompare = (datacomparetemp * datacomparetemp) + sqrdatacompare;
  }
```

Figure 5.4: Java code for suffix detection

At the time of comparing, the outer loop have the size of shorter word vector and it is compared up to size of 95 of the longer image vector, which ignores the rest of the vector of the longer size word. This means that the first word image“**አዋጅ**” is compared with the second word image up to “**አዋጅ**” by ignoring vector of “**ን**”.

5.5.2 Prefix Detection

Prefix detection helps to collect the similar kinds of word images that varies at the beginning position of image words.

The prefix detection is implemented as follows. First the word image size is identified and compared to each other to select a word with shorter size as a controller.

Compared to suffix detection, this one starts from the last vector back to the beginning until the comparison reaches to the beginning vector image of the shorter word.

For example to compare the word "አ□ፀ" (of size 99) with the word "የአዋጅ" (of size 133), shown in figure 5.5, the word with shorter size "አዋጅ" is used as a controller and the comparison goes from the end of the two images till the starting point of "አዋጅ" is reached, where the extra character "□' is identified and ignored.

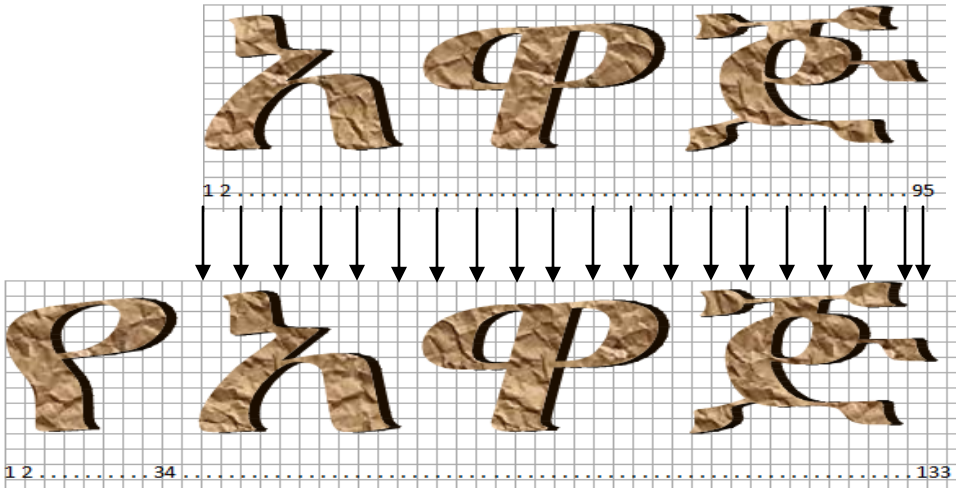

Figure 5.5: Prefix detection of two images vector size

The Java Code to implement prefix is presented in figure 5.6.

```
//To distinguish which one is longer and which one is shorter
If (data.length >= comparedata.length)
        featurelength = comparedata.length;
 else
     featurelength = data.length;
For (int ii= featurelength-2; ii >= 0; ii--)
 {
   datatemps = Double.parseDouble(data[ii]);
   datacomparetemps = Double.parseDouble(comparedata[ii]);
   dotproducts = (datatemps * datacomparetemps) + dotproducts;
   sqrdatas = (datatemps * datatemps) + sqrdatas;
   sqrdatacompares = (datacomparetemps * datacomparetemps)+sqrdatacompares;
 }
```

Figure 5.6: Java code for Prefix detection

The size of shorter image word "**አዋጅ**" is 95 and size of longer image word "**የአዋጅ**" is 133. The comparison controller assigns the size of shorter image that is 95 and it decrement counting until 1, which means up to size 34 of the longer image. This algorithm somehow ignores the prefix "□' from the longer image word.

The above suffix and prefix detection are crucial in grouping together word images that vary in structure but have the same meaning. This tremendously decrease the number of word images indexed.

The performance of stemming algorithm is tested on sample word images with word variants. Table 5.1 shows the accuracy of Amharic word images stemmer.

Word	Number of word variants	Similarity at 96%	Similarity at 98.5%	Similarity at 99%	Similarity at 99.5%
አዋጅ	60	79.74%	79.74%	69.62%	69.62%
ትምህርት	60	56.14%	56.14%	47.37%	47.37%
ኦሮሚጸ	60	100%	100%	100%	93.82%
ኢትዮጵያ	60	100%	100%	100%	93.65%
መንግሥት	60	100%	100%	100%	100%
ኢኮኖሚ	60	94.73%	94.73%	94.73%	94.73%
Average accuracy	60	88.44%	88.44%	85.3%	83.2%

Table 5.1: Performance of both suffix and prefix

The result shows, to measure the similarity between two image terms a threshold of 99% is used. This helps to cluster the image word much more precise to the same group. The suffix and prefix detection at the time of comparing two image terms shows 85.3% of average accuracy.

The result shows that variant words such as, "**በአዋጅ**", "**ከአዋጅ**" and "**□አዋጅ**" are group with the root word "**አዋጅ**". "**በትምህርት**", "**ከትምህርት**" and "**የትምህርት**" with the word "**ትምህርት**".

The main challenge during stemming is deciding, what threshold value of similarity to use. When we increase the threshold value near to'1,' some similar words left ungrouped and when we decrease the threshold value to near '0' dissimilar words may be grouped. For instance, the

word “የአዋጅ” can be grouped with the base word “አዋድ”, which have different meaning with the word “የአዋጅ”. There is also another challenge. Word images with the same base but different meanings are sometimes grouped together. For example, words like “ወጥቤት” is grouped to “ቤት” or “□ዓ” though all the three words have different meaning. Some word images are clustered to the same group that has different meaning such as “ሥራ” and “የአሥራ”. Based on the above, we select 99% of similarity measure between image words.

5.6 Stopword Images Detection and Removal

When preparing an index file, not all of the term images have equal weight in discriminating one document from the other. There are word images which exist in most of the documents in the collection. Such words cannot have discriminatory power and hence need to be ignored during constructing the index file. To remove high frequency word images, inverse document frequency is computed using the code given in Figure 5.7.

```
double IDF = (Math.log10(NumberDocument/DocumentFrequency))/Math.log10(2.0);
double cutoff = (Math.log10(NumberDocument/(NumberDocument*80/100)))/Math.log10(2.0);
if(cutoff  <  IDF)
 {
          //Indexing task will continue
 }
```

Figure 5.7: Java code for stopword removal

Here the biggest challenge is to set the threshold value of removing stopwords before constructing the index file. When the threshold value increase, there is an increase in the number of important word removal, which decreases recall of the system when we use for large

collection of documents. When the threshold value decreases, it considers less important word images as important. In our system, a word image is considered as a stopword if a word image occurred more than 80% of the documents.

There are some challenges here; some image terms are counted as a stopword because they occurred in more than 80% of the documents, but the word have the ability of discriminating the documents. such as **ኢትዮጵያ**.

5.7 Indexing

In this research work, an attempt is made to design indexing to enhance searching in document image collection. An inverted index structure is used to build the index file. After stopwords are detected and removed, important word images are left that need to be stored in the index file. To implement inverted index structure, the system stores the index term, cumulative frequency, term frequency and document frequency. Figure 5.8 shows the Java code to build the index file.

```
for (int y = 0; y < maximumfileholder.size()-1;y++ )
    EachListf.print(maximumfileholder.elementAt(y));
EachListf.print(1.5);
for (int r=0; r< temporaryfilewritten.size()-1; r++)
     EachListf.print(temporaryfilewritten.elementAt(r));
EachListf.print("," + inversedocumentfrequency);
```

Figure 5.8: Storing the index term using inverted file structure

The total number of document which we consider in the time of indexing are 70 and the total size of the documents are 20.2 Mega Byte, after indexing the total size of the index file reduce to 9.349 Mega Byte.

Table 5.2 shows the statistical information captured for sample word images in the course of building the index file.

Word Image	DF	CF	Document Id	TF
አዋጅ	2	5	doc1 doc2	3 2
ኢትዮጵያ	1	4	doc4	4
ትምህርት	3	13	doc10 doc13 doc19	3 4 6

Table 5.2: Inverted Index Structure

Here the main challenge is selecting of the index term from the clustered images. When selecting shorter in width, some problem like for the word □ዓ ቤት, two options are there. The word □ዓ can be selected to be the representative of the cluster, but □ዓ have different meaning with the word □ዓ ቤት. When selecting the longer in width to be the representative of the cluster, some problems occur words like የአዋጅን የአዋጅ and አዋጅ. The word የአዋጅን does not represent the cluster because የአዋጅን and አዋጅ have different feature in structure.

5.8 Document Retrieved

In this study, documents are retrieved in the order of their importance to the given user query. This is achieved by ranking documents based on their Term frequency and inverse document frequency (TF*IDF) weight. After sorting, relevant documents are displayed. Figure 5.9 presents Java code to compute TF*IDF weight for each word.

```
//To calculate weight or TF*IDF of a document to the query
double idf = (Math.log10 (NumberDocument/DocumentFrequency))/Math.log10(2.0);
double weight = idf * Double.parseDouble(dataI[d+1]);
// The Java Code to implement ranking
for(int p = 0; p < retrieveddocument.size();p++)
{
   String [] ranked = null;
   ranked = retrieveddocument.elementAt(p).split(",");
   double max = Double.parseDouble(ranked[ranked.length-1]);
   for(int k = p+1; k< retrieveddocument.size(); k++)    {
      String [] innerranked = null;
      innerranked = retrieveddocument.elementAt(k).split(",");
     double innermax =
     Double.parseDouble(innerranked[innerranked.length-1]);
     if(max< innermax)      {
        String temp = null;
        temp = retrieveddocument.elementAt(k);
        retrieveddocument.set(k, retrieveddocument.elementAt(p));
        retrieveddocument.set(p, temp);
        max = innermax;
     }
  }
  }
 System.out.println(retrieveddocument.elementAt(p));
```

Figure 5.9: Java code for document retrieval

5.9 Performance Evaluation

The efficiency and effectiveness of the system is tested using test cases. The efficiency is checked by measuring the time taken to search relevant documents from the index file. Further effectiveness of the system is measured using recall, precision and F-measure.

5.9.1 Test Cases

To confirm the soundness of the proposed system for searching based on users query experiments have been carried out using a size of 70 Amharic document image corpuses. Documents are collected from different categories as presented in table 5.3

Document images	**Size**
Proclamation	22
News Item	27
Guidelines	21

Table 5.3: Type of document collection

For testing the proposed system, Amharic language experts have selected 4 word queries from Amharic document corpus. The main font used in the document is Power geez Unicode 1 and the font size is 12.

5.9.2 Test Result

To measure the effectiveness of the system in retrieving relevant documents for a given query, the experiment is held at different threshold level. Table 5.4 depicts relevant and irrelevant documents retrieved at threshold values 90%, 91%, 92% and 93%.

Text Query	Documents that hold the text	Searching Threshold = 0.90			Searching Threshold = 0.91			Searching Threshold = 0.92			Searching Threshold = 0.93		
		Retrieved Documents	Relevant Document Retrieved	Non Relevant Document Retrieved	Retrieved Documents	Relevant Document Retrieved	Non Relevant Document Retrieved	Retrieved Documents	Relevant Document Retrieved	Non Relevant Document Retrieved	Retrieved Documents	Relevant Document Retrieved	Non Relevant Document Retrieved
አዋጅ	17	4	2	2	12	11	1	7	7	0	7	7	0
መንግሥት	19	4	3	1	4	3	1	4	3	1	4	3	1
ኦሮሚጸ	13	4	2	2	12	7	5	12	8	4	6	3	3
ሥራ	18	6	4	2	6	4	2	6	4	2	6	4	2

Table 5.4: Retrieved effectiveness at different threshold values

Based on the above experiment recall, precision and F-measure are computed at a threshold value of 90%, 91%, 92% and 93% for each of the query. Summary of the result is presented in table 5.5.

Text Query	Searching Threshold = 0.90			Searching Threshold = 0.91			Searching Threshold = 0.92			Searching Threshold = 0.93		
	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure
አዋጅ	11.76	50.00	19.05	64.71	91.67	75.86	41.18	100	58.33	41.18	100	58.33
መንግሥት	15.79	75.00	26.09	15.79	75.00	26.09	15.79	75.00	26.09	15.79	75.00	26.09
አርማጸ	15.38	50.00	23.53	53.85	58.33	56.00	61.54	66.67	64.00	23.08	50.00	31.58
ሥራ	22.22	66.67	33.33	22.22	66.67	33.33	22.22	66.67	33.33	22.22	66.67	33.33

Table 5.5: Precision, Recall and F-measure for Retrieved documents

As shown in table 5.5 a better performance is registered for all queries at threshold value 0.91 with this threshold value, the maximum F-measure registered is 75.86 for the word "**አዋጅ**". When clustering or grouping word images, dissimilar word images are clustered to the same group, for instance the word image "**ከአንድ**" is clustered to the group of the word image "**አዋጅ**" that is why we didn't get 100% of precision**.** Because the extracted feature of word image "**አንድ**" and "**አዋጅ**" are similar for the cosine similarity measurement. Cosine measures each of the normalized vector generated for the two word images that is "**አ**" with "**አ**", "**ን**" with "□" and "□" with "**ፀ** ". As shown in figure 5.10.

Figure 5.10: Comparing of two images

The calculated cosine similarity is greater than 99% in order to be grouped each other. Due to prefix or suffix detection, some word images are clustered to the same group that has different meaning such as "**ሥራ**" and "**የአሥራ**". The reason is that when detecting prefix of the word images, the similarity measurement starts from the right side of the vector image "**የአሥራ**" and continues until the end of the vector image "**ሥራ**". As a result, the vector image "**የአ**" is ignored. That is why we did not get 100% of precision. Figure 5.11 shows comparing of the two images "**ሥራ**" and "**የአሥራ**".

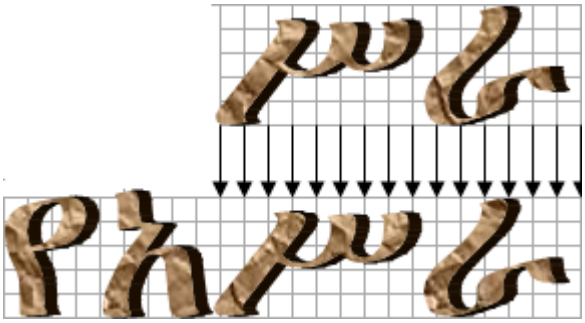

Figure 5.11: Comparing of two different size images

When we see the test result of the word **መንግሥት** shows less performance, because the language experts were selected **መንግሥት** and **መንግስት** as a same kind of words. But in this

research **መንግሥት** and **መንግስት** have different feature structure to each other, as a result these two word images are cluster in different group while they have same meaning.

Effectiveness of the retrieval is affected by the noise available within the document images. The existence of noise affects the selection of basic word images against word variants greatly. Because of noise in basic words, word variants are sometimes selected as basic words. This will affect recall and precision of the system.

5.9.3 Efficiency of the system

The time taken to search a document from the whole document is measured on Intel Pentium with 1.46 Giga Hertz Dual CPU and 1 Giga Byte of Random Access Memory. Table 5.6 presents searching time requirement before and after indexing.

Query	Time take in seconds before Indexing	Time taken in seconds after Indexing
አዋጅ	5 Seconds	4 Seconds
ትምህርት	5 Seconds	4 Seconds
ኦሮሚጸ	4 Seconds	3 Seconds
ኢትዮጵያ	4 Seconds	3 Seconds
መንግስት	5 Seconds	3 Seconds
ኢኮኖሚ	4 Seconds	3 Seconds
Average time taken	4.5 seconds	3.33 seconds

Table 5.6: A comparison of searching time required for a single query before and after indexing

As shown in Table 5.6, there is an improvement of average searching time by more than 26.6%. The time can be further enhanced by using more efficient data structures like binary tree. Because, the present search is done linearly starting from the beginning to the end, one by one. The worst case's running time depends on the size of the index list and with the use of binary tree logarithmic time, it requires less time as the size of the index list increases.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

The purpose of this work is to design a retrieval system that searches from Amharic document image corpus as per users query. In this study, an attempt has been made to introduce an index structure in order to speed up searching in document image.

The present research is a continuation of Mesfin's (2009) work that attempts to binarize document images and apply segmentation techniques to identify word images. Word images are represented using feature extraction method, based on which relevant document images are searched. To design an applicable system, the system needs to be enhanced in searching time and quality of documents retrieved. This study attempts to design an indexing scheme using inverted index file. Index terms are identified after stopwords are removed and variant words are detected. This shows that the index file contains only content-bearing terms which reduce the size of index file and enhance the relevance of document images retrieved by the system.

At the time of building index file and during searching, the system handles either suffix or prefix attached to the index term or query term by modifying cosine similarity measure. The system removes words that occur most frequently and that do not have discriminating power among document images in Amharic corpus, using Inverse Document Frequency.

Before displaying the result, relevant document images are ranked using TF*IDF weight technique. Then the system displays the search result in ranked order based on their relevant weight to the query.

Effectiveness of the system shows F-measure value of 41.59%. Searching time of the system has improved after building index file by 26.6%. When the number of document increases, the efficiency gap between searching from indexed file and from the whole document also increases. Even by designing an efficient data structure the running time can be enhanced to logarithmic (if binary time is implemented) or constant (if hashing is used) functions.

The performance of the system is greatly affected by noise exist within real-life document images. This requires integrating noise detection and removal techniques. In addition to word variants in Amharic, there are multiple words with the same meaning. In this research an attempt is made to control suffix and prefix of a word with the same meaning written in multiple form, and this greatly affect recall and precision of the system.

6.2 Recommendation

This research contributes a lot in the area of document image retrieval for Amharic language. However, there are issues, which need to be further investigated to develop a real system.

- With the objective of grouping word variants with the same meaning, the present research considers suffix and prefix of the word image. However, there are synonymous and multiple words that needs to be identified to enhance the retrieval result. So further study needs to be done to group together synonymous and multiple words in the image document.

- A better feature extraction technique should be incorporated that is insensitive to different format of the word images like Bold, underlined, italic, sizes, background color etc

- Pre- processing techniques like noise removal, skew correction and normalization helps to increase the effectiveness and efficiency of the system. Especially noise detection and removal techniques need to be integrated to the information retrieval system.

- The present similarity measure expects two word images with the same size. However there are matching algorithms that aligns one word against another without considering size variation. Hence, exploring dynamic time warping and string editing algorithm is a necessary step to design a better similarity measure.

Reference

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

2 Andreas Koschan and Wladyslaw Skarbek, Color Image Segmentation: A Survey, Institute for Technical Informatics, University of Berlin, Germany, 1994

3 Arne Andersson and Stefan Nilsson, Efficient Implementation of Suffix Tree, Software-Practice and Experience, Vol. 25(2), 129–141, 1995.

4 B. J. Lei, Emile A. Hendriks and M.J.T. Reinders, On Feature Extraction from Images: Technical report on inventory properties for MCCWS, MCCWS project Information and Communication Theory Group TUDelft, 1999

5 Beaza-Yates Ricardo and Berthier Ribeiro-Neto, Modern Information Retrieval. A Division of the Association for Computing Machinery: Addison-Wesley, ACM Press, 1999

6 Bender, M.L, Sydney W. Head, and Roger Cowley, The Ethiopian writing System: Language in Ethiopia, London: Oxford University Press, 1976

7 Bender, M. L. and Fullass. H, Amharic Verb Morphology: a Generative Approach: East Lansing, MI; African Studies Centre, Michigan State University, 1978

8 Bethlehem Mengistu, N-Gram-Based Automatic Indexig For Amharic Text: Master's Thesis, Department of Information Science, Addis Ababa University, Ethiopia, 2002

9 C. V. Jawahar, Million Meshesha and A. Balasubramanian, Searching in Document Images: Center for Visual Information Technology, International Institute of Information Technology, Gachibowli, Hyderabad, India, 2008

10 Christopher D. Manning, Prabhakar Raghavan and Hinrich Schutze, Introduction to information retrieval, USA:Cambridge university press, 2008

11 Cole, R.A. Mariani, J. Uszkoreit, H. Zaenen, A. Zue, Survey of the State of the Art in Human Language Technology: Center for Spoken Language Understanding, Oregon Graduate Institute, USA University of Pisa, Italy 1996.

12 David Doermann, The Indexing and Retrieval of Document images: A Survey: Language and Media Processing Laboratory center for Automation Research University of Maryland. 1998

13 Dengsheng Zhang and Guojun Lu, Evaluation of Similarity Measurement For Image Retrieval: Gippsland School of Computing and Info Tech, Monash University, Churchill, Victoria 3842, 2003

14 Elizabeth D. Liddy, Automatic Document Retrieval: Center for Natural Language Processing, School of Information Studies, Syracuse University, 2004.

15 Eric D. Lester. Feature Extraction, Image Segmentation, And Surface Fitting: The Development of a 3D Scene Reconstruction System: Master Thesis, The University of Tennessee, Knoxville, 1998

16 Eyassu Samuel. Amharic Text Retrieval Using Neural Networks A Comparative Study Using News Items: Master's Thesis, Department of Information Science, Addis Ababa University, Ethiopia, 2005.

17 George Hripcsak and Adam S Rothschild. Agreement, the F-Measure, and Reliability in Information Retrieval: Department of Medical Informatics, Columbia University, 622 West 168th Street, VC5, New York, NY 10032, 2005

18 Gerald J. Kowalski and Mark T. Maybury. Information Storage and Retrieval System: Theory and implementation second edition. University of Massachusetts, Amherst. Kluwer Academic Publisher, 2002.

19 Getachew Haile. The Problems of the Amharic Writing System: A paper presented in advance for the interdisciplinary seminar of the Faculty of Arts and Education, Unpublished, 1967.

20 Hamid Rahimizadeh, M.H Marhaban, R.M Kamil and N.B Ismail. Color Image Segmentation Based on Bayesian Theorem and Kernel Density Estimation: European Journal of Scientific Research, 26(3): 430-436, 2009.

21 Hudson G. Aspects of the History of Ethiopic Writing, Bulletin of the Institute of Ethiopian Studies 25.1-12, 2001.

22 James Gosling and Henry McGilton. The Java™ Language Environment: Sun Microsystems, 2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A, 1997

23 Kenneth W. Church and William A. Gale. Inverse Document Frequency (IDF): A Measure of Deviations from Poisson, ATandT Bell Laboratories Murray Hill, NJ, USA, 1995

24 Linda G. Shapiro and George C. Stockman. Computer Vision: pp 279-325, New Jersey, Prentice-Hall, 2001

25 LUHN, H.P., 'The automatic creation of literature abstracts', IBM Journal of Research and Development, 2, 159-165, 1958

26 Mahinda P. Pathegama and Özdemir Göl. Edge-end Pixel Extraction for Edge-based Image Segmentation: Transactions on Engineering, Computing and Technology, vol. 2, pp. 213-216, 2004

27 Mandl T. Recent Developments in the Evaluation of Information Retrieval Systems: Moving Towards Diversity and Practical Relevance: Information Science, University of Hildesheim, 2007

28 Marinai Soda. A Survey of Document Image Retrieval in Digital Libraries: Department of system information University of Florence, Italy, 2006

29 Marinai, Soda. Tools for Document Image Retrieval in Digital Libraries: the AIDI System, Department of System Informatics University of Florence, Italy, 2009

30 Maxim Martynov and Boris Novikov. An Indexing Algorithm for Text Retrieval: University of St.-Petersburg, Russia, 1996

31 Mesfin Worku. Amharic Document Image Retrieval Without Explicit Recognition: Master's Thesis, Department of Information Science, Addis Ababa University, Ethiopia. 2009

32 Messay Hailemariam. Handwritten Amharic character Recognition the case of Postal Address: Master Thesis, Department of Information Science, Addis Ababa University, Ethiopia, 2003.

33 Million Mesheha. A generalized Approach to Optical Character Recognition (OCR) of Amharic Texts: Master's Thesis, Department of Information Science, Addis Ababa University, Ethiopia, 2000

34 Million Meshesha. Recognition and Retrieval from Document Image Collections: PhD Dissertation, International Institute of Information Technology, Hyderabad, India, 2008

35 Million Meshesha and C. V. Jawahar. Optical Character Recognition of Amharic Documents: Center for Visual Information Technology International Institute of Information Technology Hyderabad - 500 032, India, 2007

36 Million Meshesha and C. V. Jawahar. Matching word images for content-based retrieval from printed document images: International Journal on Document Analysis and Recognition, 11(1): 29-38, 2008

37 Nega Alemayehu and Peter Willet. Stemming of Amharic Words for Information Retrieval: University of Sheffield, Sheffield, UK, Literary and Linguistic Computing 17(1), 2002

38 Nhu Van Nguyen, Jean-Marc Ogier, Salvatore Tabbone and Alain Boucher. Text Retrieval Relevance Feedback Techniques for Bag of Words Model in CBIR: World Academy of Science, Engineering and Technology, 2009

39 O. Frieder, D. Grossman, A. Chowdhury, and G. Frieder. Efficiency Considerations for Scalable Information Retrieval Servers: Department of Computer Science Illinois Institute of Technology, Chicago, 1999.

40 Peter Kukol and Jim Gray. 2004. Sequential File Programming Patterns and Performance with .NET: Boston, MA: Kluwer Academic Publisher, 2004

41 Robert M. Losee. A Discipline Independent Definition of Information: Journal of the American Society for Information Science, 48(3): 254-269, 1997

42 Ron Ohlander, Keith Price, and D. Raj Reddy. Picture Segmentation Using a Recursive Region Splitting Method: Computer Graphics and Image Processing, volume 8, pp 313-333, 1978

43 Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, Victor Zue. Survey of the State of the Art in Human Language Technology: National Science Foundation European Commission and Center for Spoken Language Understanding, Oregon Graduate Institute, Italy, 1996

44 Salton and McGill. Introduction to modern information retrieval: McGram Hill Book Company, 1983

45 Sparck Jones, K. A Statistical Interpretation of Term Specificity and its Application in Retrieval: Journal of Documentation, 28(1), 11-21, 1972.

46 Steven M. Beitzel, Eric C. Jensen and David A. Grossman. Retrieving OCR Text A Survey of Current Approaches: Information Retrieval Laboratory Department of Computer Science, Illinois Institute of Technology 2003

47 Tessema, Mindaye, Meron Sahlemariam and Teshome Kassie. The Need for Amharic WordNet: Computer Science Department, Addis Ababa University, ISandT Division UN ECA, Ministry Finance and Economic, 2010.

48 Tewodros Hailemeskel. Amharic Text Retrieval: An Experiment Using Latent Semantic Indexing (LSI) with Singular Value Decomposition (SVD: Master's Thesis, Department of Information Science, Addis Ababa University, Ethiopia, 2003

49 Weihua Huang, Chew Lim Tan, Sam Yuan Sung and Yi Xu. Word Shape Recognition For Image-Based Document Retrieval: School of Computing, National University of Singapore, 2002

50 Wondewosen Mulugeta. OCR For Special Type of Hand Written Amharic Text: Master's Thesis, Department of Information Science, Addis Ababa University, Ethiopia, 2004

51 Xiang Sean Zhou, Ira Cohen, Thomas S. Huang and Qi Tian. Feature Extraction and Selection for Image Retrieval: Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana Champaign, 2002

52 Yaregal Assabie: optical character recognition of Amharic text: an integrated approach: : Master's Thesis, Department of Information Science, Addis Ababa University, Ethiopia, 2002.

53 Yong Rui A, Thomas S. Huang A and Shih-Fu Chang. Image Retrieval Current Techniques: Promising Directions, and Open Issues, Journal of Visual Communication and Image Representation, 10: 39–62, 1999

54 Young, I., Gerbrands, J. J., and van Vliet, L. J. Image Processing Fundamentals. Quantitative Image Group Delft University of Technology, 2007

55 Yuen Hsien Tseng and Douglas W. Oard. Document image Retrieval Techniques for Chinese: Department of Library and Information Science, Fu Jen Catholic University 510 Chung Cheng Road, HsinChuang, Taipei, Taiwan, 2004

56 Zelalem Sintayehu. Automatic Classification of Amharic News Items: The case of Ethiopian New Agency: Master's Thesis, Department of Information Science, Addis Ababa University, Ethiopia, 2001

57 Martin D. Levine, Maulin R. Gandhi, Jisnu Bhattacharyya Image Normalization for Illumination Compensation in Facial Images: Department of Electrical and Computer

Engineering and Center for Intelligent Machines McGill University, Montreal, Canada, 2004

57. Jadwiga Rogowska1 and Mark E Brezinski. Image processing techniques for noise removal, enhancement and segmentation of cartilage OCT images: Orthopedics Department, Brigham and Women's Hospital, 75 Francis St., Boston, USA, 2001

Appendix 1: Partial Java code of the system

```
//This is the partial java code of the system
package imagedocument;
import java.io.BufferedWriter.*;
import java.util.*;
import java.io.*;
public class GeneralSequentialIndexing
{
   File ImageVector, ImageVectorf, Image, VectorValue;
    PrintWriter EachListf;
    File source;
    double NumberDocument = 0;
    int FlagDocumentFrequency = 0;
    public void GeneralDocumentIndexing()
    {
       try
       {
          //To open the folder Vector
          Image = new File("E:/abreham/ImageVector/");
          File f[] = Image.listFiles();
          FileWriter fileW = new FileWriter("E:/abreham/GeneralIndexFile/GeneralIndexFile.txt");
          EachListf = new PrintWriter(fileW);
          Vector<String> documentnamefrequency = new Vector<String>(2,2);
          Vector<Integer> documentlinefrequency = new Vector<Integer>(2,2);
          for (int x=0; x<f.length;x++)           // number of documents
          {
             Vector<String> line = new Vector<String>(2, 2);
             String BuffLine = null;
             String tempword = null;
             String [] data = null,  [] store = null, similarity, datatemp, datacomparetemp, dotproduct = 0, sqrdata =
             0, sqrdatacompare = 0, suffixsimilarity, datatemps, datacomparetemps, dotproducts = 0, sqrdatas = 0,
            sqrdatacompares = 0, preffixsimilarity;
             int featurelength, int outerflag;
             ImageVector = new File(f[x].getPath());
             BufferedReader Buff = new BufferedReader(new FileReader(ImageVector));
             while ((BuffLine = Buff.readLine()) != null)
             {
                line.addElement(BuffLine);
             }
             //number of words
             for (int g = 0; g<line.size();g++)
             {
                Vector<String> temporaryfilewritten = new Vector<String>(2,2);
                Vector<String> maximumfileholder = new Vector<String>(2,2);
                temporaryfilewritten.clear();
                double DocumentFrequency = 0;
                outerflag = 1;
                 //To check if the word is repeated or not
                 data = line.elementAt(g).split(",");
                 int sizedata = data.length;
                 store = line.elementAt(g).split(",");
                  for(int t=0; t<documentnamefrequency.size(); t++)
                  {
```

```
   if(ImageVector.equals(documentnamefrequency.elementAt(t)) &&
   documentlinefrequency.elementAt(t).equals(g+1))
   {
      outerflag = 0;
   }
}
if(outerflag == 1)
{
  //for the word to be compare with the root word
   for(int t=x;t<f.length;t++)
   {
      Vector<String> linecomparef = new Vector<String>(2, 2);
      Vector<String> linec = new Vector<String>(2, 2);
      String BuffLinef = null,
      String [] comparedata = null;
      int termfrequency = 0, flagofaddress = 0;
      String pageholder = null, innerpageholder = null;
      String [] comparedatatemp =null;
      ImageVectorf = new File(f[t].getPath());
      BufferedReader Bufff = new BufferedReader(new FileReader(ImageVectorf));
      while ((BuffLinef = Bufff.readLine()) != null)
      {
      linec.addElement(BuffLinef);
      }
      for (int h = 0; h < linec.size(); h++)//individual word image of  inner data
      {
          comparedata = linec.elementAt(h).split(",");
          if(comparedata[comparedata.length-1].equals(pageholder))
            ;
        else
           pageholder = comparedata[comparedata.length-1];
        // To find the number of page for document inverse calculation
        if(FlagDocumentFrequency == 0)
        {
           FlagDocumentFrequency = 1;
           File InnerImage = new File("E:/abreham/Final/");
           File Innerdocument[] = InnerImage.listFiles();
           for(int in = 0; in < Innerdocument.length; in++)
           {
              File Innerpage[] = Innerdocument[in].listFiles();
              NumberDocument = Innerpage.length + NumberDocument;
           }
        }
        //To select the smallest lenght of the word
        if(data.length >= comparedata.length)
           featurelength = comparedata.length;
        else
           featurelength = data.length;
        //for each feature vector of the word matching
        for(int i=0; i < featurelength-1; i++)
        {
           datatemp = Double.parseDouble(data[i]);
           datacomparetemp = Double.parseDouble(comparedata[i]);
           dotproduct = (datatemp * datacomparetemp) + dotproduct;
           sqrdata = (datatemp * datatemp) + sqrdata;
           sqrdatacompare = (datacomparetemp * datacomparetemp) + sqrdatacompare;
```

```
}
//cosine similarity measurment
suffixsimilarity = dotproduct / Math.sqrt(sqrdata * sqrdatacompare);
//for each feature vector of the word matching
for(int ii= featurelength-2; ii >= 0; ii--)
{
   datatemps = Double.parseDouble(data[ii]);
   datacomparetemps = Double.parseDouble(comparedata[ii]);
   dotproducts = (datatemps * datacomparetemps) + dotproducts;
   sqrdatas = (datatemps * datatemps) + sqrdatas;
   sqrdatacompares = (datacomparetemps * datacomparetemps) + sqrdatacompares;
}
//cosine similarity measurment
preffixsimilarity = dotproducts / Math.sqrt(sqrdatas * sqrdatacompares);
//selecting the best match
if(suffixsimilarity >= preffixsimilarity)
   similarity = suffixsimilarity;
else
   similarity = preffixsimilarity;
//more aproprite if similarity near to one
if(similarity > 0.97)
{
   if(sizedata > comparedata.length)
   {
      maximumfileholder.clear();
      for (int r=0; r< comparedata.length-1; r++)
      {
            maximumfileholder.addElement(comparedata[r]+",");
      }
          maximumfileholder.addElement("1.5");
   }
   termfrequency = termfrequency + 1 ;
   documentnamefrequency.addElement(f[t].getPath());
   documentlinefrequency.addElement(h+1);
}
sqrdata = 0;
sqrdatacompare = 0;
dotproduct = 0;
sqrdatas= 0;
sqrdatacompares = 0;
dotproducts = 0;
if(termfrequency >0)
{
   if(h==linec.size()-1)
   {
      innerpageholder = pageholder;
      if(flagofaddress==0)
      {
         temporaryfilewritten.addElement("," + 1.2 +","+ ImageVectorf + ","+
          innerpageholder+ ","+ termfrequency);
         // This help me to write the temporaryfilewritten to the file
         temporaryfilewritten.addElement("");
          DocumentFrequency = DocumentFrequency + 1;
          flagofaddress=1;
      }
      else
```

```
            {
               temporaryfilewritten.addElement("," + innerpageholder+ ","+ termfrequency);
               temporaryfilewritten.addElement("");
               termfrequency=0;
               DocumentFrequency = DocumentFrequency + 1;
            }
         }
         else
         {
            comparedatatemp = linec.elementAt(h+1).split(",");
            if(comparedatatemp[comparedatatemp.length-1].equals(pageholder))
            {                                   ;
            }
            else
            {
              innerpageholder = pageholder;
               if(flagofaddress==0)
               {
                  temporaryfilewritten.addElement("," + 1.2 +","+ ImageVectorf + ","+
                   innerpageholder+ ","+ termfrequency);
                  temporaryfilewritten.addElement("");
                  DocumentFrequency = DocumentFrequency + 1;
                  flagofaddress=1;
               }
               else
               {
                  temporaryfilewritten.addElement(","+ innerpageholder+ ","+ termfrequency);
                  temporaryfilewritten.addElement("");
                  DocumentFrequency = DocumentFrequency + 1;
                  termfrequency=0;
               }
            }
         }
      }
   }
}
double inversedocumentfrequency =
(Math.log10(NumberDocument/DocumentFrequency))/Math.log10(2.0);
// to check that if the data is less than or stem word or not
if (!(maximumfileholder.isEmpty()))
{
   for (int y = 0; y < maximumfileholder.size()-1;y++ )
      EachListf.print(maximumfileholder.elementAt(y));
   EachListf.print(1.5);
   for (int r=0; r< temporaryfilewritten.size()-1; r++)
   {
      EachListf.print(temporaryfilewritten.elementAt(r));
    }
   EachListf.print("," + inversedocumentfrequency);
}
else
{
   for (int r=0; r< store.length-1; r++)
   {
      EachListf.print(store[r] + ",");
   }
```

```
                    EachListf.print(1.5);
                    for (int sr=0; sr< temporaryfilewritten.size()-1; sr++)
                    {
                       EachListf.print(temporaryfilewritten.elementAt(sr));
                    }
                    EachListf.print("," + inversedocumentfrequency);
                 }
                 EachListf.println();
                 EachListf.flush();
              }
           }
        }
        EachListf.flush();
        EachListf.close();

     //   TermWeighting();
     }
     catch (Exception e)
     {
        System.out.println(e.getMessage());
     }
     TermWeighting();
   }
//To wieght and to remove less descriminating image term
  public void TermWeighting()
  {
     File docweight;
     File ImageVectorWeight;
     PrintWriter EachListWeight;
     String BuffLineWeight = null;
     String [] weightvector = null;
     try
     {
        Vector<String> lineweight = new Vector<String>(2, 2);
        FileWriter fileWeight = new FileWriter("E:/abreham/GeneralIndexFile/GeneralIndexFileWeighted.txt");
        EachListWeight = new PrintWriter(fileWeight);
        docweight = new File("E:/abreham/GeneralIndexFile/GeneralIndexFile.txt");
        ImageVectorWeight = new File(docweight.getPath());
        BufferedReader Buffweight = new BufferedReader(new FileReader(ImageVectorWeight));
        while ((BuffLineWeight = Buffweight.readLine()) != null)
        {
           lineweight.addElement(BuffLineWeight);
        }
        //To arrange the document in decending order
        for(int x = 0; x<(lineweight.size()-1);x++)
        {
           weightvector = lineweight.elementAt(x).split(",");
           double maximum = Double.parseDouble(weightvector[weightvector.length-1]);
           for(int y = x+1; y<lineweight.size();y++)
           {
              String [] weightvectorcompared = null;
              weightvectorcompared = lineweight.elementAt(y).split(",");
              double compared = Double.parseDouble(weightvectorcompared[weightvectorcompared.length -1]);
              if(maximum < compared)
              {
                 String temp = null;
```

```
                temp = lineweight.elementAt(y);
                lineweight.set(y, lineweight.elementAt(x));
                lineweight.set(x, temp);
                maximum = compared;
              }
            }
            String [] ss = null;
            ss = lineweight.elementAt(x).split(",");
            for(int i = 0; i<ss.length -1; i++)
            {
              if (i == ss.length - 2)
                EachListWeight.print(ss[i]);
              else
                EachListWeight.print(ss[i] + ",");
            }
            EachListWeight.println();
          }
        }
        catch (Exception e)
        {
          System.out.println(e.getMessage());
        }
      }
    }

//the index file and to rank the retrieved document is as follows
public void cosienSimilarityQ(int wdatastart1Q, int vdatastart1Q, int wdataend1Q,
          int vdataend1Q, int wborderstart1Q, int vborderstart1Q, int wborderend1Q, int vborderend1Q, PrintWriter
          pwwbQ)
      {
        try
        {
            VectorValue = new File("E:/abreham/GeneralIndexFile/GeneralIndexFileWeighted.txt");
            if (VectorValue.isFile())
            {
              BufferedReader inputvQ = new BufferedReader(new FileReader("wordvalueQ.txt"));
              BufferedReader inputvI = new BufferedReader(new FileReader(VectorValue));
              String linevQ = null, linevI = null;
              int eee = 0;
              double similarity = 0, datatemp, datacomparetemp, dotproduct = 0, sqrdata = 0, sqrdatacompare = 0,
              suffixsimilarity = 0, datatempsearch, datacomparetempsearch, dotproductsearch = 0, sqrdatasearch =
              0, sqrdatacomparesearch = 0, preffixsimilarity = 0, NumberDocument = 0;
              int featurelengthsearch, flag = 0;
              String[] dataQ = null, dataI = null;
              Vector<String> lineQ5 = new Vector<String>(2, 2);
              Vector<String> line2 = new Vector<String>(2, 2);
              Vector<String> retrieveddocument = new Vector<String>(2, 2);
              File InnerImage = new File("E:/abreham/Final/");
              File Innerdocument[] = InnerImage.listFiles();
              for(int in = 0; in < Innerdocument.length; in++)
              {
                File Innerpage[] = Innerdocument[in].listFiles();
                NumberDocument = Innerpage.length + NumberDocument;
              }
              while ((linevQ = inputvQ.readLine()) != null)
              {
```

```
   lineQ5.addElement(linevQ);
}
for (int k = 0; k < (lineQ5.size()); ++k)
{
   dataQ = lineQ5.elementAt(k).split(",");
}
while ((linevI = inputvI.readLine()) != null)
{
   line2.addElement(linevI);
}
//each word for document
int addressflag = 0;
for (int i = 0; i < (line2.size()); i++)
{
   if(line2.elementAt(i).equals(null))
   {
      continue;
   }
   addressflag = 0;
   dataI = line2.elementAt(i).split(",");
   //each word for query
   for (int u=0; u<dataI.length; u++)
   {
      if(Double.parseDouble(dataI[u])==1.5)
      {
         addressflag = u;
         break;
      }
   }
   //for each feature vector of the word matching
   if(dataQ.length > addressflag)
   {
      for(int h=0; h < addressflag; h++)
      {
            datatemp = Double.parseDouble(dataQ[h]);
            datacomparetemp = Double.parseDouble(dataI[h]);
            dotproduct = (datatemp * datacomparetemp) + dotproduct;
            sqrdata = (datatemp * datatemp) + sqrdata;
            sqrdatacompare = (datacomparetemp * datacomparetemp) + sqrdatacompare;
       }
       suffixsimilarity = dotproduct / Math.sqrt(sqrdata * sqrdatacompare);
       // for the preffixsimilarity
       int qq = dataQ.length-1;
       for( int hh = addressflag - 1; hh >= 0; hh--)
       {
          datatempsearch = Double.parseDouble(dataQ[qq]);
          datacomparetempsearch = Double.parseDouble(dataI[hh]);
          dotproductsearch = (datatempsearch * datacomparetempsearch) + dotproductsearch;
          sqrdatasearch = (datatempsearch * datatempsearch) + sqrdatasearch;
          sqrdatacomparesearch = (datacomparetempsearch * datacomparetempsearch) +
          sqrdatacomparesearch;
          qq--;
       }
       preffixsimilarity = dotproductsearch / Math.sqrt(sqrdatasearch * sqrdatacomparesearch);
      if(suffixsimilarity >= preffixsimilarity)
          similarity = suffixsimilarity;
```

```
    else
        similarity = preffixsimilarity;
}
if(dataQ.length <= addressflag)
{
   for(int hh=0; hh < dataQ.length; hh++)
   {
      datatemp = Double.parseDouble(dataQ[hh]);
      datacomparetemp = Double.parseDouble(dataI[hh]);
      dotproduct = (datatemp * datacomparetemp) + dotproduct;
      sqrdata = (datatemp * datatemp) + sqrdata;
      sqrdatacompare = (datacomparetemp * datacomparetemp) + sqrdatacompare;
   }
   //cosine similarity measurment
   suffixsimilarity = dotproduct / Math.sqrt(sqrdata * sqrdatacompare);
   // for the prefix similarity
   int hhindex =addressflag-1;
   for(int qq = (dataQ.length-1); qq >= 0; qq--)
   {
      datatempsearch = Double.parseDouble(dataQ[qq]);
      datacomparetempsearch = Double.parseDouble(dataI[hhindex]);
      dotproductsearch = (datatempsearch * datacomparetempsearch) + dotproductsearch;
      sqrdatasearch = (datatempsearch * datatempsearch) + sqrdatasearch;
      sqrdatacomparesearch = (datacomparetempsearch * datacomparetempsearch) +
      sqrdatacomparesearch;
      hhindex--;
   }
   preffixsimilarity = dotproductsearch / Math.sqrt(sqrdatasearch * sqrdatacomparesearch);

   if(suffixsimilarity >= preffixsimilarity)
      similarity = suffixsimilarity;
   else
      similarity = preffixsimilarity;
}
   //cosine similarity measurment
if (similarity >= 0.945)
{
   // To count the document frequency
   double DocumentFrequency = 0;
   for (int e=addressflag+2; e< dataI.length; e=e+1)
   {
      for(int m = e+1;m<dataI.length;m=m+2)
      {
         if(dataI[m].endsWith("1.2"))
         {
            e=m;
            break;
         }
         else
         {
            DocumentFrequency = DocumentFrequency+1;
            e=m+1;
         }
      }
   }
   for(int r=addressflag+2; r< dataI.length-1; r=r+1)
```

```
        {
            int l=r;
            int flagrank = 0;
            for(int d=r+1; d<dataI.length-1; d=d+2)
            {
                flagrank = 0;
                if(dataI[d].equals("1.2"))
                {
                  r=d;
                  break;
                }
                else
                {
                    //to check that the document is not repitted
                    for(int rr = 0; rr<retrieveddocument.size(); rr++)
                    {
                        String [] check = null;
                        check = retrieveddocument.elementAt(rr).split(",");
                        if(check[1].equalsIgnoreCase(dataI[d]) && check[0].equalsIgnoreCase(dataI[l]))
                        {
                            flagrank = 1;
                        }
                    }
                    if(flagrank == 0)
                    {
                        r=d+1;

                        double inversedocumentfrequency =
                        (Math.log10(NumberDocument/DocumentFrequency))/Math.log10(2.0);
                        double weight = inversedocumentfrequency * Double.parseDouble(dataI[d+1]);
                        retrieveddocument.addElement(dataI[l]+ "," + dataI[d] + "," + dataI[d+1] + "," +
                        weight);
                    }
                    else
                        r = d+1;
                }
            }
        }
    }
    dotproduct = 0;
    sqrdata = 0;
    sqrdatacompare = 0;
    sqrdatasearch= 0;
    sqrdatacomparesearch = 0;
    dotproductsearch = 0;
    suffixsimilarity = 0;
    preffixsimilarity = 0;
    similarity =0;
}
for(int p = 0; p < retrieveddocument.size();p++)
{
    String [] ranked = null;
    ranked = retrieveddocument.elementAt(p).split(",");
    double max = Double.parseDouble(ranked[ranked.length-1]);
    for(int k = p+1; k< retrieveddocument.size(); k++)
    {
```

```
                    String [] innerranked = null;
                    innerranked = retrieveddocument.elementAt(k).split(",");
                    double innermax = Double.parseDouble(innerranked[innerranked.length-1]);
                    if(max< innermax)
                    {
                       String temp = null;
                       temp = retrieveddocument.elementAt(k);
                       retrieveddocument.set(k, retrieveddocument.elementAt(p));
                       retrieveddocument.set(p, temp);
                       max = innermax;
                    }
                 }
                 System.out.println(retrieveddocument.elementAt(p));
              }
              System.out.println("Total number of pages retrieved =  "+retrieveddocument.size());
           }
        }
        catch (Exception e)
        {
          System.out.println(e.getMessage());
        }
      }
}
```

Appendix 2: Amharic Alphabets (ፊደል)

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

I, the undersigned, declare that 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 fully acknowledged.

Declared by:

Name: Abreham Gebretsadik Teklu

Signature: _____

Date: _____

Confirmed by Advisor:

Name: Million Meshesha (Ph.D)

Signature: _____

Date: _____

Addis Ababa Ethiopia

July 2010