# Afaan Oromo Search Engine

Computer Science (Wollega University)


ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

# AFAAN OROMO SEARCH ENGINE

By: **Tesfaye Guta Debela**

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF THE ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTERS OF SCIENCE IN COMPUTER SCIENCE

November 2010

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

# AFAAN OROMO SEARCH ENGINE

By: **Tesfaye Guta Debela**

ADVISOR: **Dida Midekso (PhD)**

APPROVED BY

EXAMINING BOARD:

1. Dr. Dida Midekso, Advisor _____ _____
2. _____ _____
3. _____ _____

# *Dedication*

To my baby girl, ***Naataatii,*** who was born during the proposal stage of this thesis.

# Acknowledgements

Praises and thanks to my **God**; ***"Yaa Rabbi galatni kee hin badiin!"*** Many people contributed to this research work directly or indirectly. **Dr. Dida Midekso**, my advisor, I thank you very much for your critical and timely comments. You are also my model in the field of academia. **Seyaw** and **Heni** – the ***"Caannees",*** thank you for your inspirations during title selection and your continuous encouragement. **Tessema**, thank you for your resources on Habesha search engine and your cooperation when I come with questions. Thank you **Debela** for your $11^{th}$- hour source code on stemmer for Afaan Oromo texts. **Adugna** and **Desta** from Afaan Oromo department, thank you for your cooperation in the linguistic aspect. **Mequannint Muniye**, special thanks for your technical help. I would also like to thank **Getasew Tsedalu**, **Moges Ahmed**, **Seada Hussen**, **Mequannint Muniye, Teklay G/her**, **Selama G/Meskel**, **Abel T/Mariam**, **Mandefro Kejela**, **Abebe Abeshu**, **Amsale Zelalem**, **Fetiya Beshir,** and **Zenebe Nigussie** for being my good friends as project/lunch/tea/lab partners during our graduate study. I hope our friendship will even get stronger after campus too. **Shimelis Bekele** and **Dejene Tsegaye**, I thank you that you were continuously asking me about my thesis status. Now, it seems that it is over. **Hamer**, my love, I don't think that this would have been possible without your understanding. Thank you very much for managing all the burdens that I left on you. ***Iftuu waaqjiraa***, my mam, and ***Guutaa Dabalee***, my dad, thank you that you sent me to school. Last but not least, **Dereje**, my brother, really I am lucky to have brother like you. May God bless you for what you did and is doing for me and for the rest of our families.

TABLE OF CONTENTS

LIST OF TABLES ..... IV

LIST OF FIGURES ..... V

ACRONYMS & ABBREVIATIONS ..... VI

ABSTRACT ..... VII

**CHAPTER ONE: INTRODUCTION ..... 1**

1.1 BACKGROUND ..... 1

1.2 MOTIVATION ..... 1

1.3 STATEMENT OF THE PROBLEM ..... 3

1.4 OBJECTIVES ..... 4

1.4.1 General Objective ..... 4

1.4.2 Specific Objectives ..... 4

1.5 SCOPE AND LIMITATIONS OF THE STUDY ..... 4

1.6 METHODOLOGY ..... 5

1.7 APPLICATION OF RESULTS ..... 6

1.8 THESIS ORGANIZATION ..... 6

**CHAPTER TWO: LITERATURE REVIEW ..... 7**

2.1 INFORMATION RETRIEVAL (IR) ..... 7

2.1.1 Approaches to IR or Models in IR ..... 8

2.1.2 Evaluation of IR Performance ..... 11

2.2 SEARCH ENGINES ..... 12

2.2.1 The Crawler Component ..... 12

2.2.2 The Indexer Component ..... 13

2.2.3 The Query Engine Component ..... 14

**CHAPTER THREE: RELATED WORKS ..... 16**

3.1 TAMIL SEARCH ENGINE ..... 16
3.2 ARABIC SEARCH ENGINE ..... 17
3.3 AMHARIC SEARCH ENGINE ..... 18
3.4 CHINESE SEARCH ENGINE ..... 19
3.5 SUMMARY ..... 20
**CHAPTER FOUR: AFAAN OROMO ..... 21**
4.1 INTRODUCTION ..... 21
4.2 CONSONANT AND VOWEL PHONEMES ..... 21
4.3 MORPHOLOGY ..... 23
4.4 SHORT FORMS OF COMPOUND WORDS ..... 24
**CHAPTER FIVE: DESIGN ..... 25**
5.1 DESIGN REQUIREMENTS ..... 25
5.2 ARCHITECTURE OF AFAAN OROMO SEARCH ENGINE ..... 25
5.2.1 The Crawler Component ..... 27
5.2.2 The Indexer Component ..... 27
5.2.3 The Query Engine Component ..... 28
**CHAPTER SIX: IMPLEMENTATION ..... 29**
6.1 THE CRAWLER COMPONENT ..... 29
6.1.1 Downloader and Link Extractor ..... 29
6.1.2 Text Extractor ..... 29
6.1.3 Categorizer ..... 30
6.2 THE INDEXER COMPONENT ..... 31
6.2.1 Normalizer ..... 32
6.2.2 Stopwords Remover and Stemmer ..... 33
6.2.3 Lucene ..... 35
6.3 THE QUERY ENGINE COMPONENT ..... 38
6.3.1 Query Preprocessing ..... 39
6.3.2. Searching Using Lucene ..... 39

6.3.3 Ranking ..... 40
6.4 SUMMARY ..... 41
**CHAPTER SEVEN: EXPERIMENTATION AND RESULTS ..... 43**
7.1 TESTING ENVIRONMENT AND CRITERIA ..... 43
7.2 EVALUATION OF THE CATEGORIZER ..... 43
7.3 PRECISION AND RECALL EVALUATION OF THE SEARCH ENGINE ..... 44
7.4 MEETING DESIGN REQUIREMENTS ..... 45
7.4.1 Morphological Variants of a Word ..... 46
7.4.2 Short Forms of Compound Words ..... 47
7.5 DISCUSSION ..... 49
**CHAPTER EIGHT: CONCLUSIONS AND RECOMMENDATIONS ..... 51**
8.1 CONCLUSIONS ..... 51
8.2 RECOMMENDATIONS ..... 52
REFERENCES ..... 53
APPENDIX: LIST OF AFAAN OROMO STOPWORDS. ..... 56

LIST OF TABLES

Table 1.1: Results obtained from three general search engines for selected Afaan Oromo queries. .....3

Table 6.1: Comparison of original stemmer and adjusted stemmer. .....35

Table 7.1: Evaluation result of the categorizer on Afaan Oromo documents. .....44

Table 7.2: Evaluation result of the categorizer on non-Afaan Oromo documents. .....44

Table 7.3: Precision-Recall Evaluation Result of Afaan Oromo search engine. .....45

LIST OF FIGURES

Figure 2.1: A General architecture of a search engine.....12

Figure 5.1: Architecture of Afaan Oromo Search Engine.....26

Figure 6.1: Algorithm for Afaan Oromo document categorizer.....31

Figure 6.2: Algorithm for Normalizer component.....33

Figure 6.3: A typical application integration with Lucene.....36

Figure 6.4: Search interface for Afaan Oromo search engine with sample query results.....38

Figure 7.1: Screen shot of results for the query ***Barataa.***.....46

Figure 7.2: Screen shot of results for the query ***Barattoota***.....47

Figure 7.3: Screen shot of results for the query ***I/gaafatamaa.***.....48

Figure 7.4: Screen shot of results for the query ***Itti gaafatamaa***. .....49

ACRONYMS & ABBREVIATIONS

| | |
|---|---|
| API | Application Programming Interface |
| DOM | Document Object Model |
| HITS | Hypertext Induced Topic Search |
| HTML | HyperText Markup Language |
| IDF | Inverse Document Frequency |
| IR | Information Retrieval |
| JSP | Java Server Page |
| TF | Term Frequency |
| URL | Uniform Resource Locator |
| VSM | Vector Space Model |
| XML | eXtesible Markup Language |

ABSTRACT

The Web is a repository of huge amount of information among other sources of information used in the day-to-day activities of human being. Moreover, this information may be presented in different languages. Retrieving information from the Web requires the presence of search engines. There are general purpose search engines like Google, Yahoo, and MSN. These general purpose search engines are mainly designed for English language. Shortcomings of these search engines are reflected when they are applied to non-English languages such as Afaan Oromo as they lack specific characteristics of such languages.

This research work came up with design and prototype of a search engine for Afaan Oromo texts. The search engine mainly consists of three components – crawler, indexer, and query engine that are optimized for Afaan Oromo.

The crawler downloads documents and then filtering of these documents for Afaan Oromo is done by the categorizer subcomponent of the crawler. Next, documents that are identified as Afaan Oromo are preprocessed and stored in an index for later retrieval. Finally, queries supplied in an interface to the query engine component are preprocessed, checked for a match in the index, and matched documents are displayed through an interface in a ranked order.

Performance evaluation of the search engine is conducted using selected set of documents and queries. According to precision-recall measures employed, 76% precision on the top 10 results and an average precision of 93% are obtained. Experiment on some specific features of the language against the design requirements is also made.

**Key words***:* Information Retrieval, Search Engine, Categorizer, Afaan Oromo

# CHAPTER ONE: INTRODUCTION

## 1.1 BACKGROUND

Information is very essential in the day to day activities of human being. The source of such information is diverse with World Wide Web being the dominant one. At the heart of information over the Web is a search engine. A Search engine is software that is used to retrieve information over the Web.

Since the early days of the Web, English has been the lingua franca of Web documents, technology, and search engines and tools [1, 5]. However, the use and spread of other languages is by no means negligible, a fact that contributes to the complexity and importance of investigating information retrieval on the Web. General search engines on the Web such as Google, Yahoo, and MSN are among the popular tools to search for, locate, and retrieve information, and their use has been growing fast. These engines handle English queries more or less in the same way, but their handling of non-English queries is different from how these queries are handled by non-English search engines – engines that were designed for specific languages [1].

Most general purpose search engines allow users to limit their searches to specific languages, and some of them provide local versions including interfaces as in Google Ethiopia (an interface provided by Google) for some local languages of Ethiopia including Afaan Oromo. Though they don't provide interfaces, some general search engines like Yahoo and MSN also allow users to search in some local languages. However, there is no means to evaluate their performances so far. One of the reasons for this is the lack of local search engines which take the language's special characteristics into consideration, to compare with [2].

## 1.2 MOTIVATION

Web is the source that provides large number of references for information that we are looking for. The access to this large number of information over the Web is possible through the help of search engines. Most general search engines enable people to search using English language. But people usually want to get the information they need in the language

that they better understand. Google, one of the well known search engines, has Ethiopian version interface that enables searching documents written in Afaan Oromo. However, it tends to miss some language specific aspects, as the following scenarios demonstrate.

Firstly, Google Ethiopia returns different number of results for the acronymic form ***I/Gaafatamaa***[1] (boss) and expanded form ***Itti Gaafatamaa*** of the same phrase in the language. But a user should be able to get the same result if he/she expresses his/her query in either of the two ways.

The second case is related to stemming which involves reducing a word into its root or stem. Afaan Oromo is a morphologically rich language [7]. Morphological variations of a word can be either derivational or inflectional. Derivational morphology transforms words from one syntactic class into another (such as *compute* (verb) can produce *computer* (noun)), and it usually results in change of meaning, which may create a negative effect on the performance of an information retrieval system. On the other hand, inflectional morphology (such as number, gender, case, and tense) does not change the category or general meaning of the word [4]. Here, users may use different inflectional morphemes of the same word as in ***Barataa*** (student, male) and ***Barattoota*** (students) and that should not greatly affect the results of the query. But most general search engines including Google fail to incorporate this. Specially, for languages like Afaan Oromo that do not have large number of documents on the Web, considering inflectional morphemes of a word will have significant contribution on the documents returned by the search engine.

Some queries with the corresponding number of results over three general search engines to illustrate the two scenarios described are shown in Table 1.1.

[1] As far as this document is concerned, all words or phrases that are **BOLD** and *ITALIC* are in Afaan Oromo.

**Table 1.1:** Results obtained from three general search engines for selected Afaan Oromo queries.

| Afaan Oromo Queries | Google's Results | Yahoo's Results | MSN's Results |
|---|---|---|---|
| ***I/Gaafatamaa*** | 47 | 81 | 134 |
| ***Itti Gaafatamaa*** | 214 | 262 | 86 |
| ***Barataa*** | 13,100 | 16,400 | 1,010 |
| ***Barattoota*** | 842 | 1,850 | 177 |
| ***Mana Jireenyaa*** | 313 | 663 | 140 |
| ***Manneen Jireenyaa*** | 115 | 49 | 17 |

These sample variations and the fact that no Afaan Oromo search engine exists has motivated the researcher to undertake this research work.

## 1.3 STATEMENT OF THE PROBLEM

Most general purpose search engines are mainly designed to handle English queries. These search engines tend to be unsatisfactory when they come to most non-English language queries including Afaan Oromo as they lack considering some special characteristics of the language. English is a morphologically simple language and when extending search capabilities to non-English languages, morphological variations have to be taken into account [3]. Search engines that are designed for specific language are more effective at handling queries of that language than general purpose search engines. Afaan Oromo is spoken as a first language by more than 25 million Oromo and neighboring people in Ethiopia and Kenya and it is being used as a working language of the Regional Government of Oromiya in Ethiopia. Moreover, a number of literatures, newspapers, magazines, education resources, official documents and religious writings are being written and published in this language on the Web [6, 7]. However, the general search engines that enable searching in this language are not flexible enough and they suffer from missing some language specific features that we believe Afaan Oromo search engine should consider and integrate into its searching and ranking techniques. The main aim of this thesis is, thus, to investigate how search engine for the language can be developed.

## 1.4 OBJECTIVES

### 1.4.1 General Objective

The general objective of this research work is to develop Afaan Oromo search engine.

### 1.4.2 Specific Objectives

The specific objectives of this research work include:

- Reviewing literature with regard to search engine.
- Studying the linguistics aspects of Afaan Oromo with respect to search engine specifics.
- Designing an algorithm that is used to identify Afaan Oromo documents.
- Identifying a stemming algorithm and adapting it to suite for the purpose of this research.
- Developing a general architecture of Afaan Oromo search engine.
- Developing a prototype for the proposed system.
- Evaluating the performance of the prototype.

## 1.5 SCOPE AND LIMITATIONS OF THE STUDY

Short forms of compound words may be composed of more than two words that are separated using either '/' or dot (.). However, this study considers only short forms that are of two words and separated by '/'. This study also focuses on text documents despite the fact that documents may be available as text, audio, video, and image formats.

Absence of compiled list of short forms of compound words and shortage of published literatures on Afaan Oromo are among the limitations during this study.

## 1.6 METHODOLOGY

In order to meet the general and specific objectives of this research work, the following methods and tools have been used.

**Literature Review**

For better understanding of search engines, literatures on different aspects of this thesis work have been studied. Basic concepts about IR have been studied as search engine is an information retrieval tool. Study on specific characteristics of Afaan Oromo that may have impact on search engine has been conducted. Review of four search engines that are designed for different languages has also been conducted in order to come up with architecture of Afaan Oromo search engine.

**Tools**

The programming aspect of the search engine is done using Java programming language. The reason to use Java is on one hand, the exposure of the researcher to the language and on the other hand, due to the fact that more Web-based tools that can be used for this research are java based. Lucene, an open source Java based API, is utilized for its indexing and searching capability. JSpider, an open source Java based crawler, has been employed for the crawling purpose. PDFBox, NekoHTML, and TextMining.org Java based tools have been used to extract contents of pdf, html, and word documents in a format that they are consumable by Lucene during indexing. Apache Tomcat server has been used to host Java Serve Page (JSP) that has been used to create the search interface.

**Evaluation**

Performance of the categorizer subcomponent has been evaluated on a set of 300 documents that are composed of English and Afaan Oromo. In order to evaluate the performance of the search engine, precision-recall measures have been used along with 10 selected queries on a total of 70 documents.

## 1.7 APPLICATION OF RESULTS

Afaan Oromo search engine is specifically used for effective retrieval of documents that are written using Afaan Oromo. The users of this search engine can be mainly those that wish to utilize electronic documents in Afaan Oromo that fit to their requirements.

## 1.8 THESIS ORGANIZATION

The rest of this thesis is organized as follows. Chapter Two presents literature review. It mainly focuses on basics of information retrieval and a description about general architecture of a search engine. Chapter Three presents works related to some non-English search engines. Chapter Four discusses Afaan Oromo. Chapter Five and Chapter Six focus on design and implementation of the proposed search engine respectively. Evaluation results are presented in Chapter Seven. This chapter mainly focuses on evaluation of one subcomponent of crawler in particular and performance evaluation of the proposed search engine in general. It also contains test results done against design requirements. The last chapter, Chapter Eight, presents concluding points about the thesis and also points out some future work directions.

# CHAPTER TWO: LITERATURE REVIEW

## 2.1 INFORMATION RETRIEVAL (IR)

The need to store and retrieve written information became increasingly important over centuries, especially with inventions like paper and the printing press. Soon after computers were invented, people realized that they could be used for storing and mechanically retrieving large amounts of information. Large amount of information is becoming available electronically, usually on the Web. The concept of information retrieval was coined with these activities [15].

The meaning of information retrieval can be very broad. But according to [16], it is defined as follows:

*Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).*

Although IR usually addresses unstructured documents, as stated in the definition given above, it can also cover structured, semi-structured, and a mixture of these types. Unstructured document refers to a document that does not have clear, semantically overt, easy-for-a-computer structure. A document is structured if it consists of named components, organized according to some well-defined syntax. A typical example of structured record is a relational database. If the given component has a definite semantics, then it is always possible for the search engine to find data with that semantics. An example can be finding the ages of all employees in a personnel database. A document is semi-structured in cases where a collection of textual documents may share a common structure and semantics. For example, in a collection of documents containing facts about countries, each document may contain data about a different country. The first paragraph may contain the name, location and population of the country in sentences that follow a fairly consistent form. Similarly, the second paragraph may list the principal industries and exports, again in sentences that follow a fairly standard form [16, 10].

IR focuses on retrieving documents based on the content of their unstructured components. An IR query may specify desired characteristics of both the structured and unstructured components of the documents to be retrieved. An example can be "The documents should be about 'Information retrieval' and their author must be 'Smith'". In this example, the query asks for documents whose body (the unstructured part) is about a certain topic and whose author (a structured part) has a specified value [10].

IR typically seeks to find documents in a given collection that are about a given topic or that satisfy a given information need. The topic or information need is expressed by a query, generated by the user. Documents that satisfy the given query in the judgment of the user are said to be relevant whereas documents that are not about the given topic are said to be non-relevant. An IR engine may use the query to classify the documents in a collection, returning to the user a subset of documents that satisfy some classification criterion. Naturally, the higher the proportion of documents returned to the user that he/she judges as relevant, the better the classification criterion. Alternatively, an IR engine may rank the documents in a given collection. To say that document *D1* is higher ranking with respect to a given query *Q* than document *D2* may be interpreted probabilistically as meaning that *D1* is more likely to satisfy *Q* than *D2*. Or it may be interpreted as meaning that *D1* satisfies *Q* more than *D2*. The latter could mean that *D1* is more precisely focused on the need expressed by *Q* than *D2*. Or it could mean that more of *D1* satisfies *Q* than *D2*. For example, *D1* might be entirely devoted to the need expressed by *Q* while *D2* might deal with a number of topics so that only a single paragraph of *D2* satisfies *Q* [10].

### 2.1.1 Approaches to IR or Models in IR

There are two major approaches to IR technology and research: *semantic* and *statistical*. Semantic approaches attempt to implement some degree of syntactic and semantic analysis. That is, they try to reproduce to some degree the understanding of the natural language text that a human user would provide. In statistical approaches, the documents that are retrieved or that are highly ranked are those that match the query most closely in terms of some statistical measure. By far the greatest amount of work to date has been devoted to statistical approach and hence a brief look at it is given next [10].

Statistical approaches break documents and queries into *terms*. These terms are the population that is counted and measured statistically. Most commonly, the terms are words that occur in a given query or collection of documents. The words often undergo pre-processing. They are *stemmed* to extract the *root* of each word. The objective is to eliminate the variation that arises from the occurrence of different grammatical forms of the same word, e.g., "retrieve," "retrieved," "retrieves," and "retrieval" should all be recognized as forms of the same word. Hence, it should not be necessary for the user who formulates a query to specify every possible form of a word that he/she believes may occur in the documents for which he/she is searching. Another common form of preprocessing is the elimination of stopwords that have little power to discriminate relevant from non-relevant documents, for example, "the", "a", "it" and the like in the English language. Hence, IR engines are usually provided with a *stop list* of such noisy words. Note that both stemming and stop lists are language-dependent. Statistical approaches fall into a number of categories: c*lassical Boolean*, *extended Boolean*, *vector space*, and *probabilistic*.

**Classical Boolean Approach to IR**

In the Boolean case, the query is formulated as a Boolean combination of terms. A conventional Boolean query uses the classical operators AND, OR, and NOT. The query "$t_1$ AND $t_2$" is satisfied by a given document $D_1$ if and only if $D_1$ contains both terms $t_1$ and $t_2$. Similarly, the query "$t_1$ OR $t_2$" is satisfied by $D_1$ if and only if it contains $t_1$ or $t_2$ or both. The query "$t_1$ AND NOT $t_2$" satisfies $D_1$ if and only if it contains $t_1$ and does *not* contain $t_2$. More complex Boolean queries can be built out of these operators and evaluated according to the classical rules of Boolean algebra. Such a classical Boolean query is either true or false. Correspondingly, a document either satisfies such a query (is relevant) or does not satisfy it (is non-relevant). Several kinds of refinement of this classical Boolean query are possible when it is applied to IR. First, the query may be applied to a specified syntactic component of each document, for example, the Boolean condition may be applied to the title or the abstract rather than to the document as a whole. Second, it may be specified that the condition must apply to a specified position within a syntactic component, for example, to the words at the beginning of the title rather than to any part of the title.

**Extended Boolean Approach**

Boolean conditions remain classical in the sense that they are either true or false. Such an all-or-nothing condition tends to have the effect that either an intimidatingly large number of documents or none at all are retrieved. A number of extended Boolean models have been developed to provide ranked output, that is, provide output such that some documents satisfy the query condition more closely than others.

Extended Boolean operators make use of the weights assigned to the terms in each document. A classical Boolean operator evaluates its arguments to return a value of either true or false. These truth values are often represented numerically by zero (false - or in IR terms "doesn't match a given document") and one (true - or in IR terms "matches a given document"). An extended Boolean operator evaluates its arguments to a number in the range zero to one, corresponding to the estimated degree to which the given logical expression matches the given document.

**Vector Space Model**

The vector space model (VSM) has been a standard model of representing documents in information retrieval for almost three decades. Let D be a document collection and Q be the set of queries representing users' information needs. Let also $t_i$ symbolize term i used to index the documents in the collection, with i = 1, .., n. The VSM assumes that for each term $t_i$, there exists a vector $\vec{t_i}$ in the vector space that represents it. It then considers the set of all term vectors $\{ \vec{t_i} \}$ to be the generating set of the vector space. If each $d_k$, (for k = 1, .., p) denotes a document of the collection, then there exists a linear combination of the term vectors $\{ \vec{t_i} \}$ which represents each $d_k$ in the vector space. Similarly, any query q can be modeled as a vector $\vec{q}$ that is a linear combination of the term vectors.

In the standard VSM, the term vectors are considered pair-wise orthogonal, meaning that they are linearly independent. But this assumption is unrealistic, since it enforces lack of relatedness between any pair of terms, whereas the terms in a language often relate to each other. Provided

that the orthogonality assumption holds, the similarity between a document vector $\overrightarrow{d_k}$ and a query vector $\overrightarrow{q}$ in the VSM can be expressed by the cosine measure as:

$$cos(\vec{d_k}, \vec{q}) = \frac{\sum_{j=1}^{n} a_{kj} q_j}{\sqrt{\sum_{i=1}^{n} a_{ki}^2 \sum_{j=1}^{n} q_j^2}}$$

Where $a_{kj}$, $q_j$ are real numbers standing for the weights of term j in the document $d_k$ and the query q respectively. A standard baseline retrieval strategy is to rank the documents according to their cosine similarity to the query [11].

**Probabilistic Model**

The probabilistic model attempts to capture the IR problem within a probabilistic framework. The vector-space model does not take the existing term relationships in a document into account. The probabilistic model takes these term dependencies and relationships into consideration. The whole idea is, given a user query, there is a set of documents which contain exactly the relevant documents and no other [2].

### 2.1.2 Evaluation of IR Performance

At the heart of IR evaluation is the concept of *relevance*. Relevance is an inherently subjective concept in the sense that satisfaction of human needs is the ultimate goal, and hence the judgment of human users as to how well retrieved documents satisfy their needs is the ultimate criterion of relevance. IR systems can be evaluated with respect to efficiency (operational issues like cost, time factor, space, etc.) as well as effectiveness (how well the retrieved documents satisfy the user request). Recall and precision are two measures of evaluating the effectiveness of an IR system.

- **Recall**: is the fraction of relevant documents which has been retrieved.
- **Precision**: is the fraction of the retrieved documents which is relevant.

Recall indicates completeness whereas precision refers to the system's ability to find only *relevant* documents. High precision means that the retrieved documents are highly *relevant* to the subject of the query [10].

## 2.2 SEARCH ENGINES

Search engines are special software tools that are designed to help people find information stored on the Web. There are differences in the ways various search engines work, but they all perform three basic tasks:

- They search the Web based on important words using *crawler* component.
- They keep an index of the words they find using *indexer* component.
- They allow users to look for words or combinations of words found in that index using the *query engine* component.

Early search engines held an index of a few hundred thousand pages and documents, and received one or two thousand inquiries each day. Today, a top search engine will index hundreds of millions of pages, and respond to tens of millions of queries per day [13]. A general architecture of a search engine is shown in Figure 2.1.

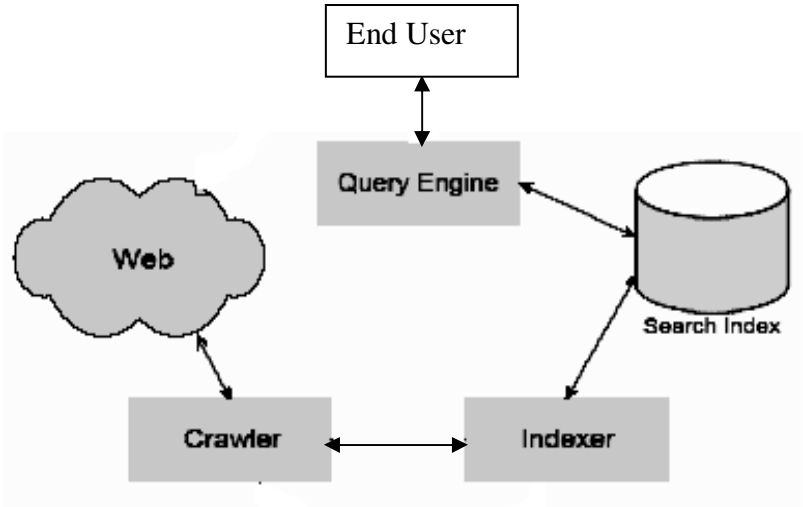


**Figure 2.1:** A General architecture of a search engine

### 2.2.1 The Crawler Component

The crawler component gathers pages from the Web for later analysis by the indexer component. How does any crawler start its travels over the Web? The usual starting points are lists of heavily used servers and very popular pages. The crawler will begin with a popular site, indexing the words on its pages and following every link found within the site. In this way, the crawling system quickly begins to travel, spreading out across the most widely used portions of the Web.

The large volume and rate of change of pages on the Web, as there is a huge amount of pages being added, changed and removed every day, made Web crawling a difficult task. The large volume implies that the crawler can only download a fraction of the Web pages within a given time, so it needs to prioritize downloads. The high rate of change implies that by the time the crawler is downloading the last pages from a site, it is very likely that new pages have been added to the site, or pages have already been updated or even deleted.

There are two basic ways by which a given crawler traverses the Web: b*readth first* and d*epth first*. In breadth first case, the crawler first extracts all links in a given page before it moves to the next page, then extracts all links on the first page of the first link and so on, until each level of links has been exhausted. In depth first scheme, the crawler starts by finding the first link in the first page. It then crawls the page associated with that link, finding the first link in the new page, and so on until the end of the path has been reached.

### 2.2.2 The Indexer Component

An index keeps information about each document in a collection. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Most indices use variants of inverted file. An *inverted file* contains, for each term in the lexicon, an inverted list that stores a list of pointers to all occurrences of that term in the main text, where each pointer is, in effect, the number of a document in which that term appears. Some search engines use elimination of stopwords to reduce the size of the index. Stemming can also help reduce the size of the index. Another operation to be performed before indexing is tokenization – that involves dividing the stream of text into chunks of words. This operation requires considering white spaces to separate words for most languages like English, Amharic, and Afaan Oromo.

**Stemming**

Stemming removes word suffixes, perhaps recursively in layer after layer of processing. The process has two goals. In terms of efficiency, stemming reduces the number of unique words in the index, which in turn reduces the storage space required for the index and speeds up the search process. In terms of effectiveness, stemming improves recall by reducing all forms of the word to

a base or stemmed form. For example, if a user asks for *analyze*, he/she may also want documents which contain *analysis*, *analyzing*, *analyzer*, *analyzes*, and *analyzed*. Therefore, the document processor stems document terms to *analy-* so that documents which include various forms of *analy-* will have equal likelihood of being retrieved; this would not occur if the engine only indexed variant forms separately and required the user to enter all. Of course, stemming does have a downside. It may negatively affect precision in that all forms of a stem will match, when, in fact, a successful query for the user would have come from matching only the word form actually used in the query [16].

**Stopwords Removal**

Stopwords are common words that carry less important meaning than keywords. Usually, search engines remove stopwords from a keyword phrase to return the most relevant result. That is, stopwords drive much less traffic than keywords. Stopwords are part of human language and there is nothing you can do about it. But high stopwords density can make your content look less important for search engines. These words need to be removed before the document gets indexed. The removal will save disk space required for indexing [35].

### 2.2.3 The Query Engine Component

The query engine is the component that interacts with a user. The user types his/her query on the interface provided by the query engine component and the results to the query are also displayed to the user through the help of this component but on a different interface. The query engine accepts the user query, consults the indexer component, makes some ranking, and finally presents the results to the user [2].

**Ranking**

Ranking of documents returned for a user query involves different techniques, among which PageRank and HITS (Hypertext Induced Topic Search) are common. PageRank is a numeric value that represents how important a page is on the web. This algorithm is an important part of the Google's ranking function. PageRank is a recursive process in that a page with high PageRank is a page that is referenced by many pages with high PageRank. This PageRank is defined as [14]:

*We assume page A has pages T1...Tn which point to it (i.e., are citations). C(A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:*

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

*The parameter d is a damping factor which can be set between 0 and 1. The PageRank theory holds that even an imaginary surfer who is randomly clicking on links will eventually stop clicking. The probability, at any step, that the person will continue is a damping factor d. Various studies have tested different damping factors, but it is generally assumed that the damping factor will be set around 0.85*

*Note that the PageRanks form a probability distribution over web pages, so that the sum of all web pages' PageRanks will be one.*

Another technique of ranking is the use of HITS (Hypertext Induced Topic Search). Unlike the PageRank, it is query dependent and calculated at the time of the search. HITS determines two values for a page: its *authority*, which estimates the value of the content of the page, and its *hub* value, which estimates the value of its links to other pages. That is, a hub refers to the number of links pointing from the page and authority depends on the number of pages with links pointing to it. The starting point for the HITS algorithm is a set of pages that are expected to be relevant to the given query [15].

# CHAPTER THREE: RELATED WORKS

A number of search engines for non-English languages have been developed over the years. This chapter briefly discusses some of these search engines.

## 3.1 TAMIL SEARCH ENGINE

Tamil is one of the fast growing and highly inflectional languages on the Internet among the Indian languages. Many websites in the language are being available and locating required information from these sites has become problematic. As a result, a Tamil search engine that enables users to retrieve relevant pages in the language was developed. The Tamil search engine incorporated the basic components that many search engines have [17].

As a search approach, Boolean search and phrasal search approaches were considered. The Boolean search works the way it is described in Chapter Two as classical Boolean approach to IR. In phrasal search, the user specifies his/her query as a phrase between two quotes. It was also indicated that the phrasal option requires huge processing power and large memory in the database.

The ranking consideration used in the Tamil search engine included:

- The proximity of the query terms with each other, when the query contains more than one term.
- Frequency of the query term (term frequency) in the page normalized by the total number of pages having the query term (document frequency) in the database.
- The presence of keywords (i) at the beginning of the text (ii) in bold or italics.
- The presence of any of the query terms in the content.

In the ranking process, hypertext vector method of ranking where a site is ranked based on the number of other sites having a link that points to the current site was not considered as it needs high computing power.

Morphological analyzer was incorporated in the Tamil search engine to take advantage of the inflectional nature of the language and its high accuracy (95%) to simple word stemming. Stopwords removal was also incorporated in the search engine.

Two user interfaces were incorporated to allow the user to choose suitable mode of typing his/her query:

- Tamil keyboard – for users who can type in Tamil, and
- Phonetic keyboard – for users who know English but not familiar with any of the Tamil keyboards. Here, the user uses transliterated English.

## 3.2 ARABIC SEARCH ENGINE

Arabic is a language with the largest number of speakers in the Semitic family. It is spoken by more than 280 million people as a first language and more than 250 million people as a second language [18]. More and more Arabic documents are being published on the web. In order to help utilize these documents, several Arabic search engines were also developed (Al-Bahar, Ayna, Hahooa, Konouz, Maktoob, Barq, etc.). Here, we focus on Barq: a distributed multilingual search engine with focus on Arabic language [19].

The Barq search engine is composed of backend and front-end subsystems sharing replicated index databases. The backend subsystem includes focused crawler, stemmer, indexer, categorizer, and broker. The focused crawler gathers documents that contain at least one Arabic word from the Web and caches it for offline processing. The stemmer then retrieves the cached documents and converts each word into the corresponding root word that is used to build the index. The indexer in Barq contains for each document its words, root words, number of occurrences, positions, color, font, whether they appear in title, body of the text, or caption, as well as a comprehensive set of metadata about the document itself (these comprise the creation date, the refresh date, the size, and the format (like pdf, word, HTML)). The categorizer component plays the role of categorizing the indexed documents into one or more of predefined classes to help users build sophisticated queries and narrow down their search. The performance of the categorizer is 75.6%. HITS (Hypertext Induced Topic Search) technique that uses *hubs* and *authorities* was used for ranking purpose. Broker, the other back-end component, is used to

offer a location transparent access to redundant index databases and to manage pools of connections to them. The broker also serves URLs to be fetched by the crawler threads.

The front-end subsystem accepts user queries and consults the index databases through the front-end brokers to respond to the queries. The front-end broker offers location-transparent access to index databases and manages pools of connections to the index databases.

Barq lets users to choose among the following types of search: exact key word based search, concept based search, and image search. The scope of the search can be set to either the whole Arabic Web or a particular site. And it employs Boolean information retrieval model. Stopwords removal was also taken into account by the search engine.

### 3.3 AMHARIC SEARCH ENGINE

Amharic is the working language of the Federal Government of Ethiopia and some regional governments of the country. Tessema [2] has designed an Amharic search engine that enables searching documents on the Web that are written in the language.

Among the things that were taken into consideration during the design of Amharic search engine are:

- Morphological variants of words: stemming of inflectional morphemes of a word for tense, number, case, and gender was considered.
- Short form of compound words: short form of compound words and expanded form of the compound words were treated as being referring the same thing.
- Repetitive alphabets: Amharic alphabets that have same pronunciation and use, but different representations were made to be treated as being the same by the search engine.
- Encoding issues: The work considered only Amharic documents written using fonts that have Unicode encoding.

The categorizer that was developed to identify Unicode encoded Amharic pages showed an accuracy of over 99%. The combination of text based ranking and link based ranking was used in the Amharic search engine to rank documents that are retrieved. In the link based ranking, after the crawler downloaded all the pages, the links in all the pages will be parsed. All the links will

be analyzed and the number of incoming Amharic links that a page has will be the link rank of that page. The text based similarity is calculated using the vector space model. And the sum of these two ranking results was used to rank a given page. Accuracy of the search engine using the 'OR' and 'AND' operators is 75% and 99% respectively.

### 3.4 CHINESE SEARCH ENGINE

The authors in [21] designed and implemented a search engine named Net-Compass that provides both Chinese and English information retrieval for web pages on sites of Chinese. The search engine employed vector space model as text information retrieval. It used inverted file indexing that stores each word and its occurrences in a document for ranking purpose.

Among the considerations made during the design of Net-Compass are Chinese text segmentation, Chinese character code detection, and catalogs.

Unlike English documents where white spaces and punctuation can be used as word delimiter, Chinese words don't have obvious word boundaries. To deal with Chinese text segmentation two known approaches, single character based approach and multi character based approach, were used. In single character based approach, a document is segmented into individual Chinese character to build word indexing. With multi character based approach, only words listed in a Chinese dictionary are used to build the word index. The same consideration is made on the users' query side. Combination of the two approaches was used in the Net-Compass search engine.

Net-Compass used GB (Guo-Biao) as a character encoding among other Chinese encoding standards (BIG5, ISO-2022-CN, HZ, etc.). Documents of other encoding standards were made to automatically be recognized and transformed to GB code. Network Chinese Filter (NCF) component of Net-Compass is responsible to handle the transformation. GB and BIG5 interfaces were provided for users to interact with the search engine.

The Internet is thought to be a large library without catalogs, as a result of which searching is considered challenging. Catalog was incorporated in the Net-Compass to be used as a guide for users so as to ease searching.

## 3.5 SUMMARY

The general working principle of search engines is almost the same. Mainly, crawler, indexer and query engine are incorporated in the architecture of every search engine. Differences are usually observed in some preprocessing stages. For example, stemming is being incorporated in search engines that are designed for morphologically rich languages. Most of the reviewed works also incorporated language identification module.

# CHAPTER FOUR: AFAAN OROMO

## 4.1 INTRODUCTION

Afaan Oromo is an Afro-Asiatic language, and the most widely spoken of the Cushitic sub-phylum. As indicated in Section 1.3, Afaan Oromo is widely used as both written and spoken language in Ethiopia and some neighboring countries, including Kenya and Somalia. Besides being a working language of Regional Government of Oromiya, Afaan Oromo is the instructional medium for primary and junior secondary schools throughout the region. Moreover, a number of literatures, newspapers, magazines, educational resources, official documents and religious writings are written and published in Afaan Oromo [6, 7, 22, 34].

There are also television and radio programs on which information in Afaan Oromo is being broadcasted in Ethiopia. These include Ethiopian Television (ETV), Oromiya Television (TV Oromiya), Ethiopian Radio, and Radio Fana.

Before 1991, Ge'ez script was used for writing Afaan Oromo documents. A Latin-based alphabet called ***Qubee*** has been adopted and became the official script of Afaan Oromo since 1991. There are about twenty-six consonants and ten vowels (five short and five long) in the language [22, 12].

## 4.2 CONSONANT AND VOWEL PHONEMES

Like most other Ethiopian languages, whether Semitic, Cushitic, or Omotic, Afaan Oromo has a set of ejective consonants, that is, voiceless stops or affricates that are accompanied by glottalization and an explosive burst of air. Afaan Oromo has another glottalized phone that is more unusual, an implosive retroflex stop, "dh" in Afaan Oromo orthography, a sound that is like an English "d" produced with the tongue curled back slightly and with the air drawn in so that a glottal stop is heard before the following vowel begins [22, 34].

Afaan Oromo has the typical Southern Cushitic set of five short (a, e, i, o, u) and five long vowels, indicated in the orthography by doubling the five vowel letters (aa, ee, ii, oo, uu). The difference in length of vowels results in change of meaning.

Example:

| Afaan Oromo | English |
|---|---|
| ***hara*** | lake |
| ***haaraa*** | new |
| ***boru*** | tomorrow |
| ***booruu*** | dirty |

Gemination (doubling a consonant) is also significant in Afaan Oromo. That is, consonant length can distinguish words from one another.

Example:

| Afaan Oromo | English |
|---|---|
| ***badaa*** | bad |
| ***baddaa*** | highland |
| ***hatuu*** | to steal |
| ***hattuu*** | thief |

In Afaan Oromo alphabet, a letter consists either of a single symbol or a digraph (ch, dh, ny, ph, sh). Gemination is not obligatorily marked for the digraphs [22].

Words in Afaan Oromo sentences are separated by white spaces the same way as it is used in English. Different Afaan Oromo punctuation marks follow the same punctuation pattern used in English and other languages that follow Latin writing system. For example, comma (,) is used to separate listing of ideas, concepts, names, items, etc and the full stop (.) in statement, the question mark (?) in interrogative and the exclamation mark (!) in command and exclamatory sentences mark the end of a sentence[23].

## 4.3 MORPHOLOGY

Like in a number of other African and Ethiopian languages, Afaan Oromo has a very complex and rich morphology [20]. It has the basic features of agglutinative languages involving very extensive inflectional and derivational morphological processes. In agglutinative languages like Afaan Oromo, most of the grammatical information is conveyed through affixes, (that is, prefixes and suffixes) attached to the root or stem of words. Although Afaan Oromo words have some prefixes and infixes, suffixes are the predominant morphological features in the language. Almost all Afaan Oromo nouns in a given text have person, number, gender and possession markers which are concatenated and affixed to a stem or singular noun form. In addition, Afaan Oromo noun plural markers or forms can have several alternatives. For instance, in comparison to the English noun plural marker, *s (-es)*, there are more than ten major and very common plural markers in Afaan Oromo including: ***-oota***, ***-oolii***, ***-wwan***, ***-lee***, ***-an***, ***een***, ***-eeyyii***, ***-oo,*** etc.). As an example, the Afaan Oromo singular noun ***mana*** (house) can take the following different plural forms: ***manoota (mana + oota), manneen (mana + een), manawwan (mana + wwan)***. The construction and usages of such alternative affixes and attachments are governed by the morphological and syntactic rules of the language [20].

Afaan Oromo nouns have also a number of different cases and gender suffixes depending on the grammatical level and classification system used to analyze them. Frequent gender markers in Afaan Oromo include ***-eessa/-eettii***, ***-a/-ttii*** or ***–aa/tuu***.

Example:

| **Afaan Oromo** | **Construction** | **Gender, English** |
|---|---|---|
| ***obboleessa*** | ***obbol + eessa*** | male, brother |
| ***obboleettii*** | ***obbol + eettii*** | female, sister |
| ***beekaa*** | ***beek + aa*** | male, knowledgeable |
| ***beektuu*** | ***beek + tuu*** | female, knowledgeable |

Likewise, Afaan Oromo adjectives have case, person, number, gender, and possession markers similar to Afaan Oromo nouns. Afaan Oromo verbs are also highly inflected for gender, person, number, tenses, voice, and transitivity. Furthermore, prepositions, postpositions and article markers are often indicated through affixes in Afaan Oromo [20, 34].

The extensive inflectional and derivational features of Afaan Oromo are presenting various challenges for text processing and information retrieval tasks in the language. In information retrieval, the abundance of different word forms and lexical variability may result in a greater likelihood of mismatch between the forms of a keyword in a query and its variant forms found in the document index databases. Stemming, a technique that is used to bring morphological variants of a word into the root or stem word, plays an important role in this regard.

## 4.4 SHORT FORMS OF COMPOUND WORDS

Short form representation of compound words is common in Afaan Oromo similar to other languages like Amharic and English. In such representation, the use of forward slash (/) is much common although a dot or period (.) can also be used alternatively. In Afaan Oromo documents, either of the short or long form of compound words may exist. If such words exist in query terms in information retrieval, there needs to be a way to handle the situation so that both forms could be retrieved as alternative for the query.

Example:

| **Afaan Oromo (short form)** | **Afaan Oromo (long form)** | **English** |
|---|---|---|
| ***I/gaafatamaa*** | ***Itti gaafatamaa*** | Boss/Chief/Head |
| ***B/M*** | ***Bulchiinsa Magaalaa*** | City Administrator |
| ***W.A.*** | ***Waldaa Aksiyoonaa*** | Share Company |
| ***K.K.F*** | ***Kan Kana Fakkaatan*** | etc./ and so on |

# CHAPTER FIVE: DESIGN

## 5.1 DESIGN REQUIREMENTS

As design requirements for Afaan Oromo search engine, some characteristics of the language should be taken into consideration. Among these characteristics, the first one is stemming inflectional morphemes of words that basically do not have significant impact on the semantics. The second characteristic is devising a means of handling short forms of compound words so that they will be treated as being the same by the search engine. The third characteristic is devising a means by which Afaan Oromo documents are identified from other documents so as to minimize the domain from which searching is to be done and to increase the relatedness of the documents being retrieved.

Apart from the language characteristics, there should be a simple user friendly interface by which a user feeds his/her query to the search engine and an interface through which results will be displayed.

The following sections provide a brief discussion on the architecture of the proposed Afaan Oromo search engine.

## 5.2 ARCHITECTURE OF AFAAN OROMO SEARCH ENGINE

Afaan Oromo search engine has three major components like many other search engines. These are:

- Query engine
- Indexer
- Crawler

However, these components have subcomponents that are unique for the language they are designed for. Accordingly, detailed architecture of the proposed search engine is given in Figure 5.1 and the description of each of the components is given in sections 5.2.1, 5.2.2, and 5.2.3.

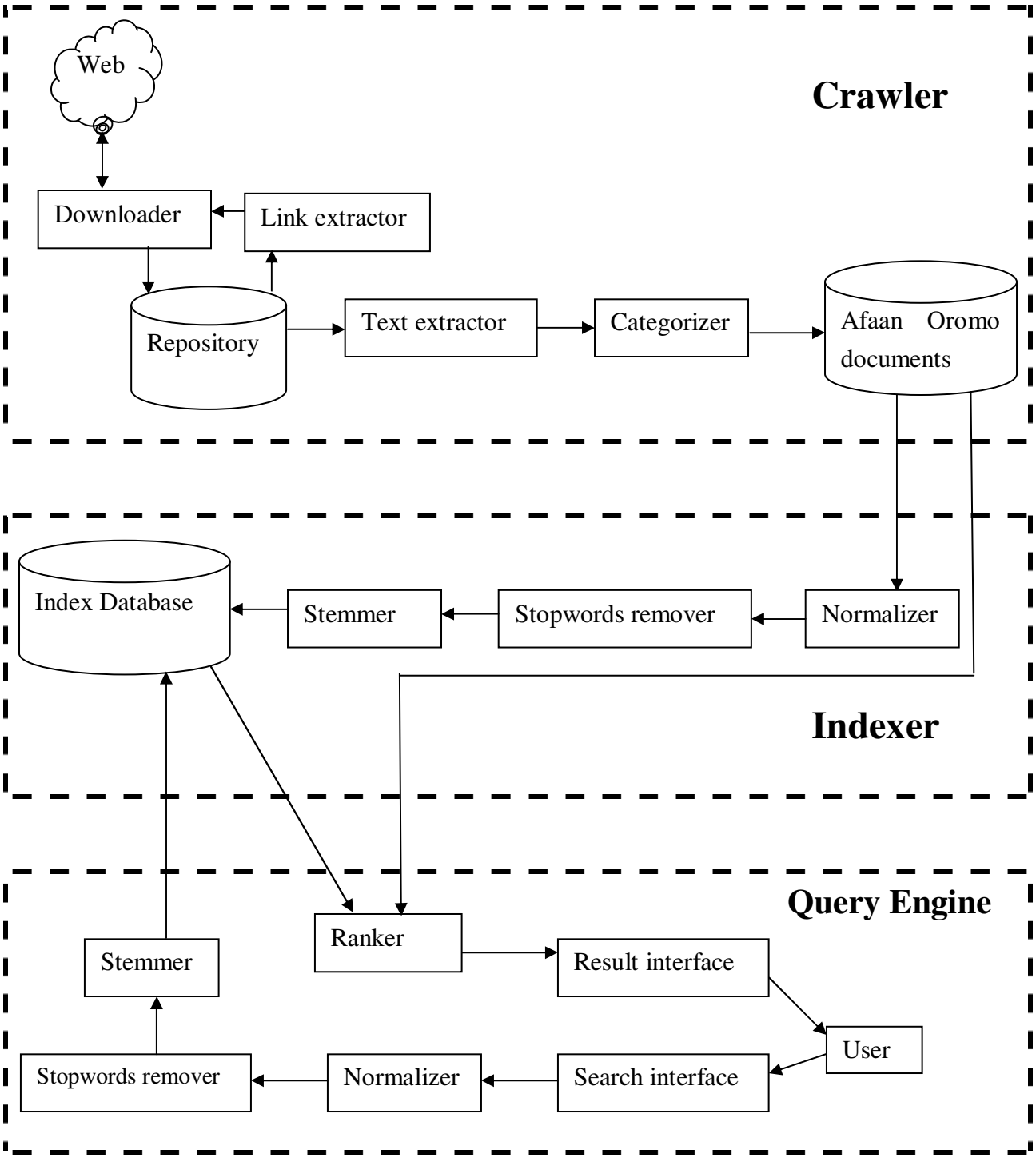


**Figure 5.1:** Architecture of Afaan Oromo Search Engine.

### 5.2.1 The Crawler Component

The crawler component has subcomponents that play different role in the crawling process. These subcomponents are: downloader, link extractor, text extractor, and categorizer.

The downloader starts with a selected URL and searches for documents on the Web for download. All the downloaded documents are then stored in a repository. Text extractor subcomponent is used to extract texts from pdf, word, and html documents in the repository as the format of these documents cannot be directly used by indexer and categorizer. Once the texts are extracted, the categorizer subcomponent takes its turn to categorize documents as being written in Afaan Oromo or non-Afaan Oromo. Those documents that are not in Afaan Oromo will be rejected while those written in Afaan Oromo will be stored as Afaan Oromo documents for further processing. The link extractor subcomponent extracts links from the repository of documents that is going to be used as the next URL.

### 5.2.2 The Indexer Component

The indexer component takes Afaan Oromo documents from the crawler component and undergoes some preprocessing steps to convert the document into a form that eases the searching process. This component includes normalizer, stopwords remover, and stemmer subcomponents.

The normalizer subcomponent takes Afaan Oromo documents and parses text of a document into its constituent words usually by considering white spaces and punctuation marks (punctuation mark usage in Afaan Oromo is similar to that of English). The normalizer also handles short forms of compound words. The stopwords removal subcomponent removes words that do not have significant impact on the document being searched from the normalized documents. The stemmer subcomponent will reduce inflectional words from the stopwords remover subcomponent into their most simplified form known as stem or root. Finally, the processed document will be stored in index database to be used later by the query engine component.

### 5.2.3 The Query Engine Component

The query engine component accepts user query, consults the index database, performs some preprocessing activities, and finally returns results to the query. This component encompasses normalizer, stopwords remover, stemmer, and ranker subcomponents.

A user enters his/her query through the help of search interface. The query will pass through the normalizer, stopwords remover, and stemmer subcomponents whose purposes are the same as the corresponding subcomponents in the indexer component. Then, the preprocessed query will be consulted with the index database for and Afaan Oromo documents for ranking. Finally, the ranked documents will be displayed as a result for the user through the result interface.

# CHAPTER SIX: IMPLEMENTATION

This chapter is devoted to describing the implementation of Afaan Oromo search engine whose architecture is discussed in Chapter Five. In different sections of the implementation, different off-the-shelf components are used. These components and their brief explanations are given within the corresponding sections that they are used. Section 6.1 presents the implementation of the crawler. Preprocessing of documents before they get indexed is described in Section 6.2. The implementation of the query engine component and a summary are given in Sections 6.3 and 6.4 respectively.

## 6.1 THE CRAWLER COMPONENT

Several open source crawlers that are written in different languages are available on the Web. The researcher has gone through some of the crawlers written in java and picked one that can fit to his requirement. This crawler is named **JSpider**. JSpider is a highly configurable and customizable Web spider engine developed under the LGPL (Lesser General Public License) open source license purely in Java [33]. The overall crawler subcomponents used in this research work are described in sections 6.1.1, 6.1.2, and 6.1.3.

`

### 6.1.1 Downloader and Link Extractor

The downloader is made to start with seed URLs and to download the documents into a repository. The downloaded files are kept in their appropriate file formats as pdf, word, html, or text file formats. The link extractor subcomponent then extracts links from the repository of documents and feeds to the downloader to be used as the next URL to download documents from.

### 6.1.2 Text Extractor

In order for documents like pdf, word, and html to be consumed by some applications like lucene's index and our categorizer, they need to be converted into text. To carry out these conversions, different open source tools are available. The tools selected for this research work are briefly discussed below. These tools are selected mainly because they are java-based and open source.

**PDFBox**: is an open source Java PDF library for working with PDF documents. This project allows creation of new PDF documents, manipulation of existing documents and the ability to extract content from documents [30]. Version 0.7.3 is used for this work.

**NekoHTML**: is an open source HTML parser in java used to parse HTML documents with an XML DOM (standard way for accessing and manipulating XML documents) kind of representation of the parsed HTML from which the text contents can be retrieved [31]. Version 0.9.2 is used in this work.

**TextMining.org's API**: is also java based program that is capable of parsing and extracting useful information, including the body text, from Microsoft Word documents [25]. This work uses version 0.4.

### 6.1.3 Categorizer

The categorizer subcomponent is used to separate documents with Afaan Oromo contents from documents written in other languages. Since Afaan Oromo search engine is designed to respond to Afaan Oromo queries, the relevant documents are those documents with Afaan Oromo content. Documents with other languages are considered to be irrelevant for such queries. Hence, the categorizer retrieves documents that are written in Afaan Oromo while leaving out the rest from a repository of downloaded documents. According to [24], Afaan Oromo words do have any of the following four syllables (c = consonant, v = vowel):

| Pattern | Afaan Oromoo | English |
|---|---|---|
| cvc | ***sun, kun, kam*** | that, this, which |
| cv | ***na-ma, ku-ma*** | human, thousand |
| cvvc | ***maal, yoom*** | what, when |
| cvv | ***dii-maa, see-naa*** | red, history |

Below are examples of other Afaan Oromo words with possible combination of syllables described above:

| Afaan Oromoo | Pattern | English |
|---|---|---|
| ***Qul-lub-bii*** | cvc-cvc-cvv | onion |
| ***Suk-kuu-muu*** | cvc-cvv-cvv | to rub |
| ***Ba-reed-duu*** | cv-cvvc-cvv | beautiful |

Thus, taking into account these syllables, a categorizer that filters Afaan Oromo documents is developed. The algorithm is described in Figure 6.1.

1. Get **textual** form of a document
2. Split the texts into **words** using white space
3. Get a **word**
4. If **syllable of a word** matches any of **Afaan Oromo syllable category**, go to step 5 otherwise step 8
5. Increment **count** of that syllable category by 1
6. If **count** of each syllable category reaches (**12, 8**)** go to step 7 otherwise step 8
7. **Categorize** the document as **Afaan Oromo** and go to step 9
8. If **end of file** is not reached, go to step 3 otherwise go to step 9
9. **End** categorizing.

**This is the count of (**cvv**, **cvcc**) syllable match at which better categorization is obtained.

**Figure 6.1:** Algorithm for Afaan Oromo document categorizer.

## 6.2 THE INDEXER COMPONENT

As already stated in Chapter Five, the indexer component is responsible to store information about the categorized documents after going through some preprocessing tasks. These preprocessing tasks are handled by normalizer, stopwords remover, and stemmer subcomponents that are discussed in sections 6.2.1 and 6.2.2. The indexing service is handled by one of the most common indexing and searching java-based open source library named Lucene. A brief description of Lucene is given in section 6.2.3.

### 6.2.1 Normalizer

As already discussed in Chapter Six, the normalizer component mainly deals with parsing texts of a document into constituent words and handling short forms of compound words. Word demarcation in Afaan Oromo is handled following white space. Punctuation marks except those used for representing short forms of compound words are also removed during normalization.

The other aspect handled by the normalizer is short forms of compound words. This aspect was handled using Java's **HashMap** data structure in such a way that the beginning of the short form of words are stored as a key with their corresponding expanded form as a value. For example, the word ***I/gaafatamaa*** is stored as ***I*** as a key and ***Ittii*** as a value, because ***I/gaafatamaa*** is expanded as ***Itti gaafatama.*** Likewise, when both words before and after **'/'** are of one character long, the same concept is applied except that the character before and after **'/'** are handled on different HashMap for differentiating their position. For example, for the short word ***W/A***, ***W*** as a key and ***Waldaa*** as a value is stored in a HashMap different from a HashMap that stores ***A*** as a key and ***Aksiyoonaa*** as a value. The algorithm for normalizer is shown in Figure 6.2.

Set **destination file** to empty

Open **source file**

Get a character

**While** (not end of **source file**)

    If the character is not any of Afaan Oromo Punctuations (**. , ; ? ! “**)

        Append the character to **destination file**

    Read next character

**End while**

Split contents of **destination file** into words using white space

Set **destination file2** to empty

Open **destination file**

Get a word

**While** (not end of **destination file**)

    If the word contains the character **‘/’**

        Append the expanded form of the word to **destination file2**

    Else

        Append the word as it is to **destination file2**

    Read next word

**End while**

**Figure 6.2:** Algorithm for Normalizer component.

### 6.2.2 Stopwords Remover and Stemmer

Every language has its own list of stopwords, words that do not have discriminating value in IR. These words need to be removed during preprocessing stages. Among the techniques used to remove stopwords are IDF (inverse document frequency) value and dictionary lookup. The IDF

approach assumes words that appear in many documents as stopwords. Most of the existing stopwords removal techniques are based on a dictionary that contains a list of stopwords. This technique is much easier for well studied languages that have standard list of such words. Stopwords mainly consist of prepositions, conjunctions, articles, and particles. For the purpose of this research work, lists of around 100 stopwords that are compiled from Afaan Oromo books during implementation of a stemmer by Debela Tesfaye [32] are used. These words are given in Appendix.

The other subcomponent used before documents are indexed is stemmer. Stemmer reduces morphological variants of words into base or root form. According to [20], morphologies of a word, specially suffixes, can be composed of attached, derivational, and inflectional suffixes. Afaan Oromo attached suffixes are particles or postpositions. Derivational suffixes are mainly used for the formation of new words in the language from stem or base form of a word. Inflectional suffixes of a word may indicate tense, case, plurality (number), and gender differences. The most common order/sequence of Afaan Oromo suffixes (within a given word) is: <stem> <derivational suffixes> <inflectional suffixes> <attached suffixes>. For example the word ***barattootarratti*** (on the students) is composed of ***itti, irra*** (attached suffixes), ***oota*** (inflectional suffix), ***at*** (derivational suffix), and ***bar*** (the stem). A stemmer for Afaan Oromo text developed by Debela Tesfaye [34] is adapted for this research work with some adjustment. The original stemmer stems a word for attached, derivational and inflectional morphologies. The adjustment made to the stemmer only removes attached and inflectional suffixes as these suffixes do not change the general meaning of a word. Example that illustrates stemming of some words using the original stemmer and the adjusted stemmer is given in Table 6.1.

**Table 6.1:** Comparison of original stemmer and adjusted stemmer.

| Word | English | Result of original stemmer | Result of adjusted stemmer |
|---|---|---|---|
| ***barattootarratti*** | On the students | bar | barat |
| ***barattootaf*** | For the students | bar | barat |
| ***barattoota*** | students | bar | barat |
| ***barsiistuu*** | teacher (female) | bar | barsiis |
| ***barsiisaa*** | teacher (male) | bar | barsiis |
| ***beekna*** | We know | beek | beek |
| ***beektan*** | you know | beek | beek |

### 6.2.3 Lucene

Lucene is a high performance, scalable Information Retrieval (IR) library. It helps to add indexing and searching capabilities to applications. Lucene is a mature, free, open-source project implemented in Java. It is a member of the popular Apache Jakarta family of projects, licensed under the liberal Apache Software License [25]. Lucene can index and make searchable any data that can be converted to a textual format. Figure 6.3 presents how an application may be integrated with Lucene.

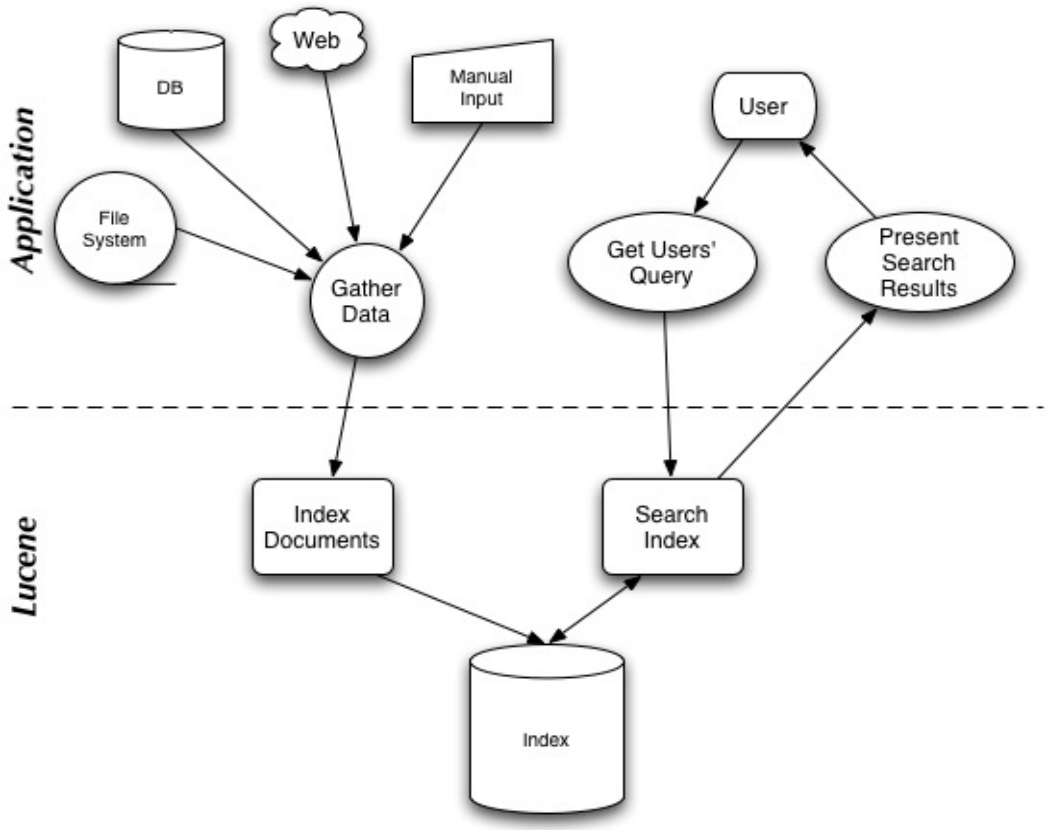


**Figure 6.3:** A typical application integration with Lucene [25].

**Lucene's Indexing Process**

Preprocessing operations (like normalization, stopwords removal, stemming, etc) are applied on the text before it is indexed in Lucene index. These pre-processing operations are called Analysis in Lucene terminology, where text is converted into its most fundamental indexed representation, terms. These terms are used to determine what documents match a query during searches. Analyzers are the encapsulation of the analysis process [25].

Indexing involves the processing of the original data into a highly efficient cross-reference lookup (index) in order to facilitate searching. The index stores statistics about terms in order to make term-based search more efficient. Lucene's index falls into the family of indexes known as an inverted index. An index contains a sequence of documents. A document is a sequence of fields where a field is a named sequence of terms (strings) [25, 26]. Lucene's indexer has five basic classes: IndexWriter, Directory, Analyzer, Document, and Field [25]. A brief overview of each class is given below:

**IndexWriter**

IndexWriter is the central component of the indexing process. This class creates a new index and adds documents to an existing index. The IndexWriter class can be considered as an object that provides us write access to the index but it does not let us read or search it.

**Directory**

The Directory class represents the location of a Lucene index. It is an abstract class that allows its subclasses to store the index as they see fit. IndexWriter then uses one of the concrete Directory implementations, FSDirectory or RAMDirectory, and creates the index in a directory in the file system.

**Analyzer**

Before text is indexed, it is passed through an Analyzer. The Analyzer, specified in the IndexWriter constructor, is in charge of extracting tokens out of text to be indexed and eliminating the rest. If the content to be indexed is not plain text, it should first be converted to it. Analyzer is an abstract class, but Lucene comes with several implementations of it. Some of them deal with skipping stopwords; some deal with conversion of tokens to lowercase letters, so that searches are not case-sensitive; and so on. Analyzer is an important part of Lucene and can be used for much more than simple input filtering.

The Analyzer class is language dependent as the preprocessing operations are language specific. Some of the analyzers incorporated in Lucene index are analyzers for English, Chinese, German, and French [27]. Hence, as none of these, including the AmharicAnalyzer developed by Tessema Mindaye [2], cannot be used for Afaan Oromo, AfaanOromoAnalyzer is developed to make use of Lucene indexer for the purpose of this research work.

**Document**

A Document represents a collection of fields. It can be considered as a virtual document –a chunk of data, such as a web page, an email message, or a text file – that we want to make retrievable at a later time.

**Field**

Fields of a document represent the document or meta-data associated with that document. The original source (such as a database record, a Word document, a chapter from a book, and so on) of document data is irrelevant to Lucene. The meta-data such as author, title, subject, date modified, and so on, are indexed and stored separately as fields of a document. Each field corresponds to a piece of data that is either queried against or retrieved from the index during search.

## 6.3 THE QUERY ENGINE COMPONENT

This component enables a user to type his/her query, does some preprocessing on the query, and returns relevant documents after consulting the indexer. The query and result interfaces are designed using JSP (Java Server Page), a server-side programming technology that enables the creation of dynamic web pages and applications. The query interface, with sample query result, developed using JSP for Afaan Oromo search engine is given in Figure 6.4.

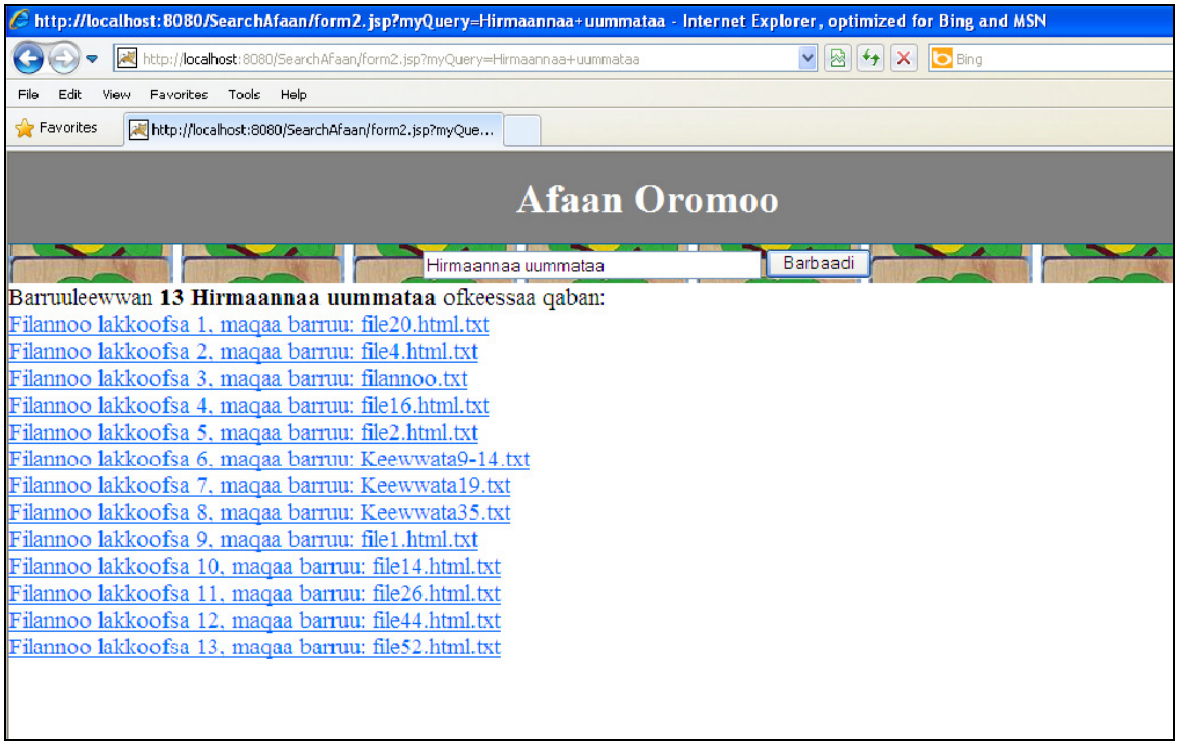


**Figure 6.4:** Search interface for Afaan Oromo search engine with sample query results.

Lucene's search capability again plays a vital role in consulting the index and returning documents that match the query supplied. Preprocessing on user's query and Lucene's searching contribution are discussed in sections 6.3.1 and 6.3.2 respectively. Section 6.3.3 presents ranking of returned documents for a user query.

### 6.3.1 Query Preprocessing

Once a query is supplied to the search engine component through the query interface, it needs to be preprocessed before a match is to be checked from the index. This stems from the fact that the index stored preprocessed contents of documents.

First, the query needs to be normalized by removing punctuation marks from it, tokenizing it following whitespaces, and expanding the query's short words of compound words, if any. Next, the query is passed to the stopwords remover and stemmer respectively. Finally, this preprocessed query is given to Lucene's searcher module, further processing.

### 6.3.2. Searching Using Lucene

Searching is the process of looking for words in the index and finding the documents that contain those words. Lucene's searching capability is equally important to its counterpart of indexing. Whatever data indexed in a Lucene index will be meaningless unless it is made convenient for searching. Like the indexing API, the search API has some core classes that facilitate searching. Some of the important searching classes are IndexSearcher, Term, Query, TermQuery, and Hits [25]. These classes are briefly discussed as follows:

**IndexSearcher** is to searching what IndexWriter is to indexing, the central link to the index that exposes several search methods. We can think of IndexSearcher as a class that opens an index in a read-only mode. It offers a number of search methods, some of which are implemented in its abstract parent class Searcher; the simplest takes a single Query object as a parameter and returns a Hits object.

A **Term** is the basic unit for searching. Similar to the Field object, it consists of a pair of string elements, the name of the field and the value of that field.

**Query** is an abstract base class for queries. Searching for a specified word or phrase involves wrapping them in a term, adding the terms to a query object, and passing this query object to IndexSearcher's search method. Lucene comes with a number of concrete Query subclasses: TermQuery, BooleanQuery, PhraseQuery, PrefixQuery, PhrasePrefixQuery, RangeQuery, FilteredQuery, and SpanQuery.

**TermQuery** is the most basic type of query supported by Lucene, and it is one of the primitive query types. It is used for matching documents that contain fields with specific values.

The **Hits** class is a simple container of pointers to ranked search results; that is documents which match a given query. For performance reasons, Hits instances do not load from the index all documents that match a query, but only a small portion of them at a time.

### 6.3.3 Ranking

All the documents that match a user query need not be returned in any order. Rather they need to be sorted based on their degree of relevancy to the query. This involves the use of ranking algorithms. The ranking algorithm employed in this research work is entirely that of Lucene. Lucene scoring uses a combination of the Vector Space Model (VSM) of Information Retrieval and the Boolean model to determine how relevant a given document is to a user's query. Lucene also adds some capabilities and refinements onto this model to support Boolean and fuzzy searching, but it essentially remains a VSM based system at the heart [28]. Among the most common factors used in Lucene's ranking model are Term Frequency (TF) and Inverse Document Frequency (IDF) which are described below [29]:

**Term Frequency (TF) -** The term frequency of a given term is the number of times that the term appears in a given document. The TF is usually normalized with respect to document length, that is, the frequency of term *t* divided by the frequency of the most frequent term in document *d*. This helps to avoid long documents to be ranked higher than short ones during ranking. Therefore, the normalized TF is given as:

$$tf_{t,d} = \frac{freq_{t,d}}{\max_l freq_{l,d}}$$

Where, $tf_{t,d}$ is the normalized frequency of term t in document *d*.

$freq_{t,d}$ is the frequency of term t in document *d*.

$\max_l freq_{l,d}$ is the frequency of the most frequent term in document *d*.

**Inverse Document Frequency (IDF) -** Inverse document frequency reflects how frequent a term is in the whole collection. The underlying principle is that a term that appears in a few documents gives more information than a term that appears in many documents. This means a term that appears in many documents has a weak discriminating power to select relevant documents over a document collection. If *N* is the number of documents and $n_t$ is the number of documents containing the query term *t*, then the inverse document frequency is given by:

$$Idf_t = \log \frac{N}{n_t}$$

Where, $Idf_t$ is the inverse document frequency of term t.

The most commonly used term weight is the composite of TF and IDF, known as Term Frequency-Inverse Document Frequency (TF_IDF), which gives importance both to the distribution of a term in the document in the collection and the frequency of occurrence of the term in the individual documents:

$$Tf\_Idf = tf_{t,d} \times Idf_t$$

## 6.4 SUMMARY

In this chapter, implementation of Afaan Oromo search engine was discussed. Different open source tools that are java-based were used in some parts. As a subcomponent of the crawler, JSpider was used for downloading documents and extracting links. PDFBox**,** NekoHTML**,** and TextMining.org's API were also used as text extracting tools from pdf, html and word documents. An algorithm developed by the researcher for categorizing Afaan Oromo documents was described.

As subcomponents of the indexer component, the normalizer, stopwords remover, stemmer and Lucene were described in detail.

Finally, in the query engine component, JSP was used for creating search interface, preprocessing on the query was discussed, usage of lucene again to search query matches from the index, including its text based ranking, was discussed.

# CHAPTER SEVEN: EXPERIMENTATION AND RESULTS

## 7.1 TESTING ENVIRONMENT AND CRITERIA

As discussed in Chapter Two, a common way of evaluating the effectiveness of IR system is using the precision-recall matrix. Here also, this technique is employed on some set of queries to evaluate the effectiveness of the prototype. The JSpider crawler used in this research work is fed with initial URLs of http://www.orto.gov.et (Website of Oromiya radio and television organization) and http://www.radiofana.com. The categorizer subcomponent that is used to filter Afaan Oromo documents from set of downloaded documents is evaluated independently. Some queries were also posed to the search engine to check against the design requirements set in Chapter Five.

The testing is done on a laptop computer with Windows XP Professional Edition SP2 (service pack 2) operating system, 1.5 GHz CPU, 1GB RAM, and 40 GB hard disk.

## 7.2 EVALUATION OF THE CATEGORIZER

The categorizer subcomponent categorizes documents as being written in Afaan Oromo by following the syllable pattern of words. Afaan Oromo shares the same Latin script with English language among some other languages. Words with the same syllable pattern can be found in both Afaan Oromo and English languages. But the researcher has realized that syllable patterns of **cvv** and **cvcc** are more frequent in Afaan Oromo documents than in English documents. Therefore, these two syllable patterns are used to evaluate the accuracy of the categorizer. Moreover, the test is conducted on a collection of documents containing Afaan Oromo and English documents. English is chosen among other languages sharing the same script with Afaan Oromo as many documents are published in this language. Accordingly, a total of 300 document collections with 161 Afaan Oromo and 139 English documents are used for the experiment. This experiment is also conducted on different cut-off values that are used as number of syllable matches of **cvv** and **cvcc**. However, performance evaluation of the categorizer on only two of these cut-off values at which better categorization occurs is shown in Table 7.1 and Table 7.2.

**Table 7.1**: Evaluation result of the categorizer on Afaan Oromo documents.

| Total Afaan Oromo documents in the collection | Categorized as Afaan Oromo | | Accuracy | |
|---|---|---|---|---|
| | Cut-off value (**cvv, cvcc**) = (12,8) | Cut-off value (**cvv, cvcc**) = (20,8) | Cut-off value (**cvv, cvcc**) = (12,8) | Cut-off value (**cvv, cvcc**) = (20,8) |
| "**A**" | "**B**" | "**C**" | "**(B/A)*100**" | "**(C/A)*100**" |
| 161 | 127 | 124 | 78.9% | 77% |

**Table 7.2:** Evaluation result of the categorizer on non-Afaan Oromo documents.

| Total non-Afaan Oromo documents in the collection | Wrongly categorized as Afaan Oromo | | Accuracy | |
|---|---|---|---|---|
| | Cut-off value (**cvv, cvcc**) = (12,8) | Cut-off value (**cvv, cvcc**) = (20,8) | Cut-off value (**cvv, cvcc**) = (12,8) | Cut-off value (**cvv, cvcc**) = (20,8) |
| "**A**" | "**B**" | "**C**" | "**((A-B)/A)*100**" | "**((A-C)/A)*100**" |
| 139 | 22 | 2 | 84.2% | 98.6 |

### 7.3 PRECISION AND RECALL EVALUATION OF THE SEARCH ENGINE

As already discussed in Chapter Two, recall is the fraction of relevant documents which has been retrieved and precision is the fraction of the retrieved documents which is relevant. Mathematically,

$$\text{Re}\,call = \frac{relevant\ \ documents\ \ retrieved\ by\ the\ search\ engine}{total\ number\ of\ \ relevant\ documents\ \ in\ the\ collection}, \text{ and}$$

$$\text{Pr}\,ecision = \frac{relevant\ \ documents\ \ retrieved\ by\ the\ search\ engine}{total\ number\ of\ \ retrieved\ \ documents\ \ by\ the\ search\ engine}$$

Evaluation of Afaan Oromo search engine using these two measures is conducted on 70 documents using 10 queries. Precision for the top 10 results is obtained by taking the ratio of relevant documents retrieved in the first 10 search results by 10. The evaluation result using precision-recall matrix is presented in Table 7.3.

**Table 7.3:** Precision-Recall Evaluation Result of Afaan Oromo search engine.

| Query | T-Rel | Ret | R-Ret | T-10P | P | R |
|---|---|---|---|---|---|---|
| ***Qonnaan bultoota*** | 9 | 36 | 9 | 0.6 | 0.25 | 1 |
| ***Manneen jireenyaa*** | 10 | 13 | 10 | 0.9 | 0.77 | 1 |
| ***Hirmaannaa uummataa*** | 9 | 10 | 7 | 0.7 | 0.7 | 0.77 |
| ***Misooma baadiyyaa*** | 3 | 33 | 3 | 0.3 | 0.1 | 1 |
| ***Madda eenyummaa*** | 1 | 2 | 1 | 0.5 | 0.5 | 1 |
| ***Filannoo biyyooleessaa*** | 11 | 16 | 9 | 0.9 | 0.69 | 1 |
| ***Maanguddoota biyyaa*** | 9 | 11 | 9 | 0.8 | 0.82 | 0.82 |
| ***Qabeenya uumamaa*** | 12 | 34 | 10 | 0.9 | 0.3 | 0.83 |
| ***Bineensoota bosonaa*** | 1 | 1 | 1 | 1 | 1 | 1 |
| ***Biiroo barnootaa*** | 21 | 33 | 19 | 1 | 0.58 | 0.9 |
| **Average** | | | | **0.76** | **0.57** | **0.93** |

Where, T-Rel – total number of documents that are relevant to the query in the collection

Ret - total number of documents retrieved by the system

R-Ret – total number of retrieved documents that are relevant to the query

T-10 P – precision with respect to the top 10 relevant documents retrieved

P – Precision

R- Recall

## 7.4 MEETING DESIGN REQUIREMENTS

The motivation of this work is the failure of general purpose search engines to handle queries in Afaan Oromo properly. As stated in Chapter One, these search engines fell short when they are given queries that reflect the typical features of the language. In this research work an attempt is made to handle these limitations. How the designed Afaan Oromo search engine behaves on queries that are morphological variants of a word and short forms of compound words are discussed in sections 7.4.1 and 7.4.2 respectively.

### 7.4.1 Morphological Variants of a Word

The search engine can display same results for queries that are morphological variants (inflectional morphology) of the same word. For example, results for queries ***barataa*** (student), ***barattoota*** (students), ***barataaf*** (for a student) must be the same, as their stem for inflectional morpheme is ***barat***. The screen shots for two of these morphemes are shown in Figure 7.1 and Figure 7.2. Here, in the Afaan Oromo search engine, as a matter of implementation of the prototype, links to file names in a ranked order are made to be returned for a query that is supplied to the engine. The user can then click on the links to see the contents.

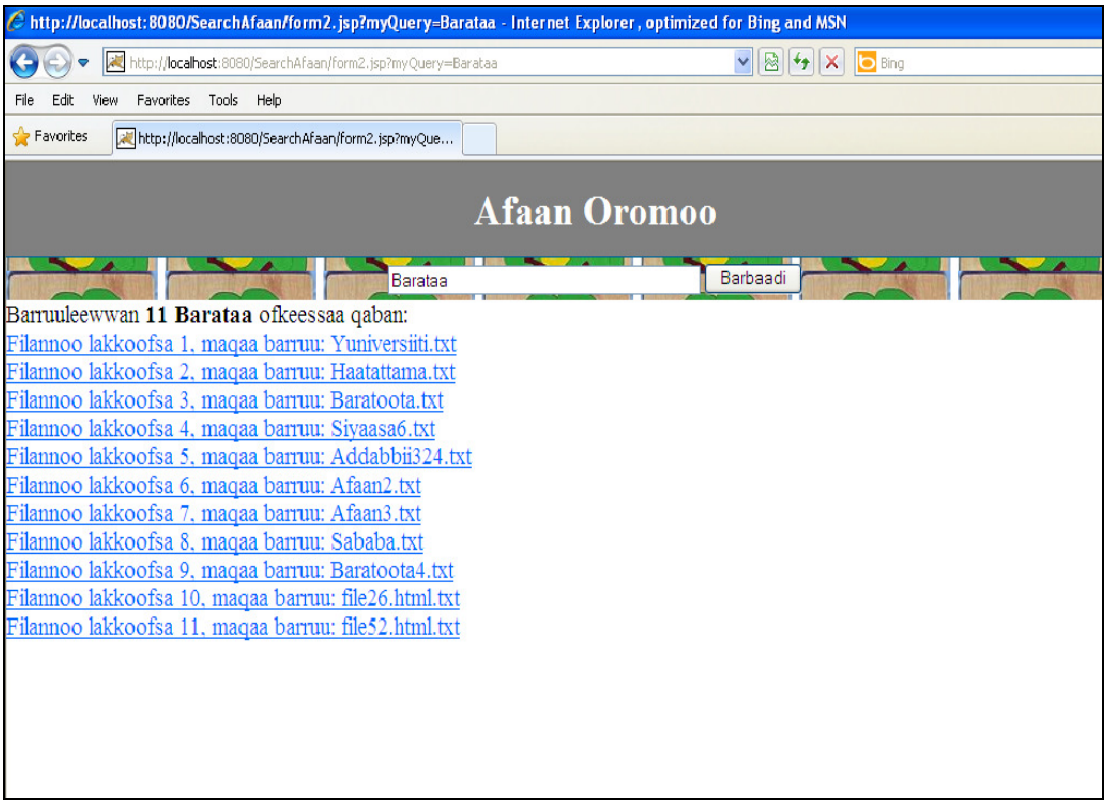


**Figure 7.1:** Screen shot of results for the query ***Barataa***.

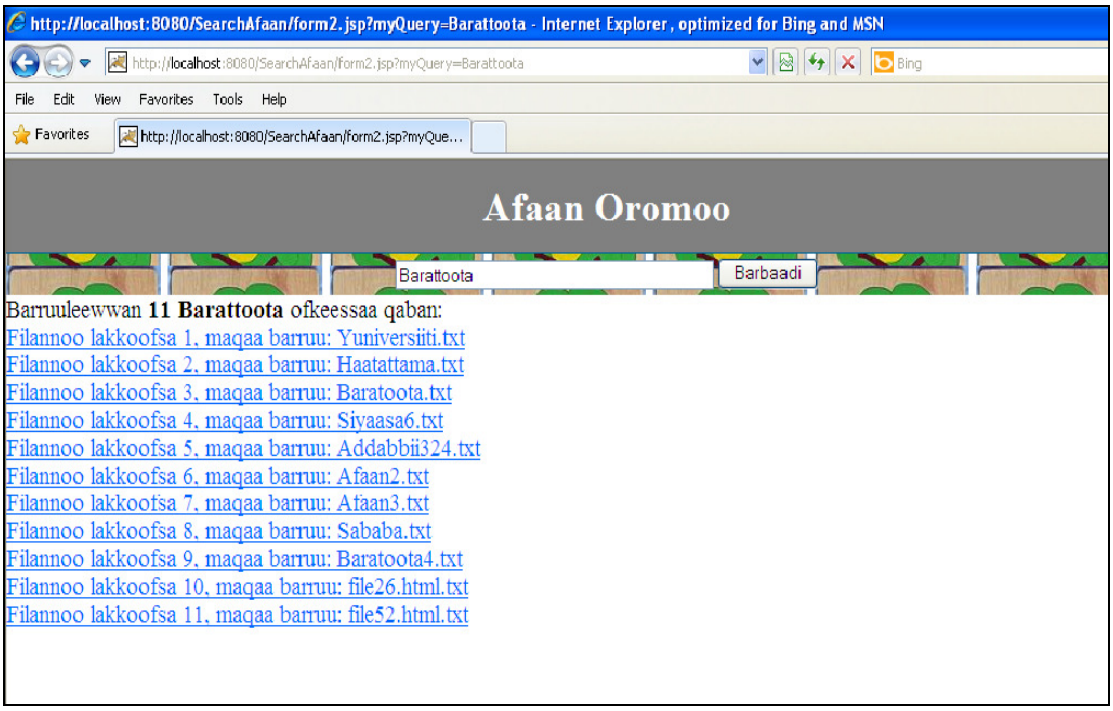


**Figure 7.2:** Screen shot of results for the query ***Barattoota***.

### 7.4.2 Short Forms of Compound Words

The other design requirement of Afaan Oromo search engine is handling short forms of compound words. The search engine can return similar results for queries that are supplied using short forms and expanded forms. For example, search results for the query ***I/gaafatamaa*** and ***Itti gaafatamaa*** (boss/chief/head) are the same. Screen shot of results for these queries are shown in Figure 7.3 and Figure 7.4.

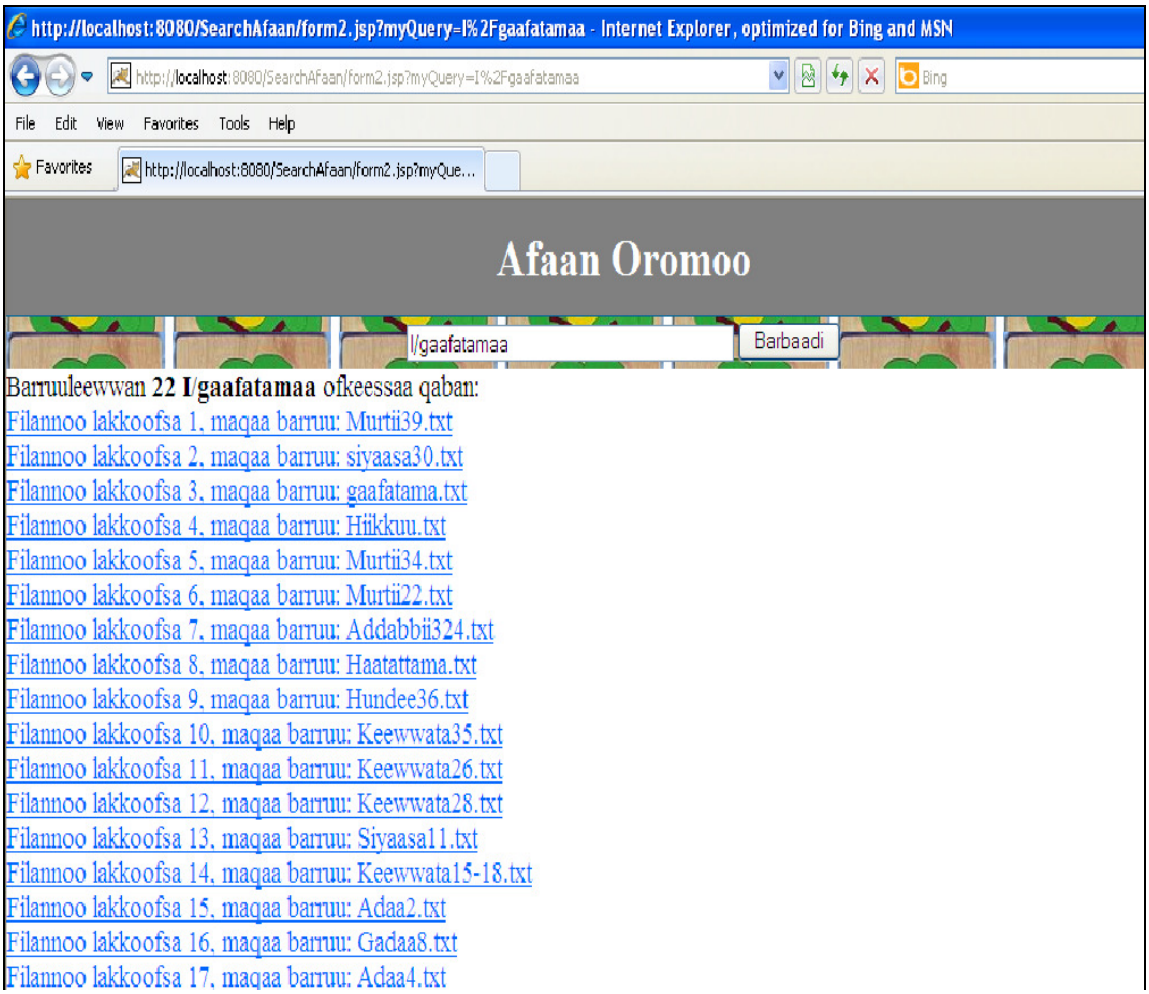


**Figure 7.3:** Screen shot of results for the query ***I/gaafatamaa***.

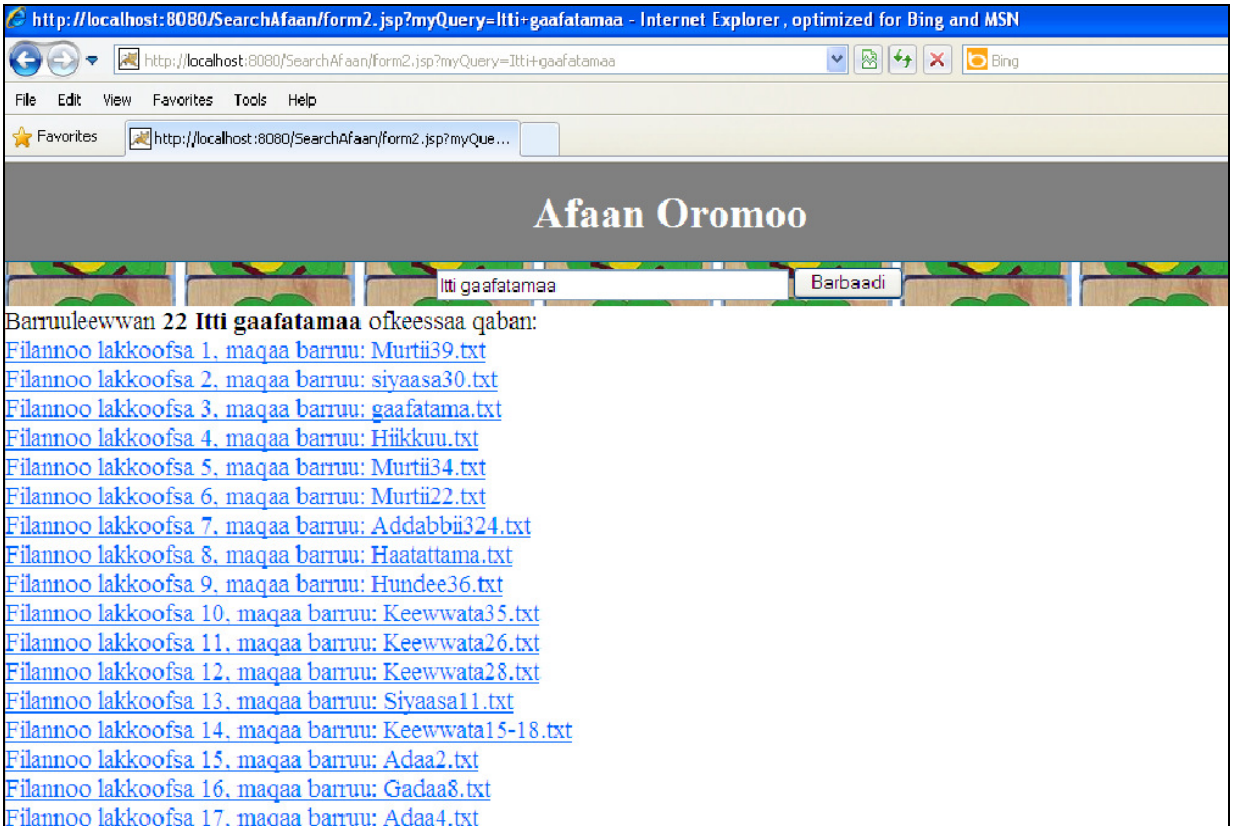


**Figure 7.4:** Screen shot of results for the query ***Itti gaafatamaa***.

## 7.5 DISCUSSION

When the categorizer is run on the 300 document collections, it identified 149 (127+22) and 126 (124+2) of the documents as Afaan Oromo documents on the two cut-off values respectively. This result is split in Table 7.1 and Table 7.2 for convenience of analysis. Accordingly, the categorizer has shown a performance of 78.9% in correctly identifying Afaan Oromo documents and it has shown a performance of 98.6% in correctly rejecting non-Afaan Oromo documents. The small decrease in performance from 78.9% to 77% in Table 7.1 is in favor of the reasonably large gap from 84.2% to 98.6% in performance increase of the categorizer on non-Afaan Oromo documents which indicates that more number of non-Afaan Oromo documents match **cvv** syllable pattern as compared to **cvcc** syllable pattern. As the cut-off values of **cvv** go below 12, more non-Afaan Oromo documents tend to be wrongly categorized as being Afaan Oromo documents and pushing values of this cut-off beyond 20, tend to eliminate more Afaan Oromo documents. During categorization, checking for syllable match for every word in the document

may require much time for large documents as well as it may favor large documents to have more matches to be checked.

According to the evaluation of the search engine presented in section 7.3, the average precision and recall are 57% and 93% respectively. The average precision for the top 10 results is 76%. This indicates that displaying only the first few top results, top 10 in this case, entails better user satisfaction as compared to displaying all the results. When a query is posed to the search engine, the search will take place using the default OR operator of Lucene search on the terms in the query. This causes more number of documents to be retrieved. In effect, the precision will be negatively affected.

# CHAPTER EIGHT: CONCLUSIONS AND RECOMMENDATIONS

## 8.1 CONCLUSIONS

Human beings need information in their day-to-day activities. The sources of such information can be diverse with the Web being the largest repository. Moreover, the contents of this information can obviously be in different languages. Retrieving information from the Web requires the presence of search engines. Search engines are software tools that may be designed for general purpose or specific use. Most general purpose search engines allow users to limit their queries to specific languages while others allow users to search in some local languages. Most of the general search engines are originally aimed to handle queries in English languages. This contributes to their shortcomings in considering specific characteristics of non-English languages like Afaan Oromo.

This research work came up with design and prototype of a search engine for Afaan Oromo texts. The search engine mainly consists of three components – crawler, indexer, and query engine that are optimized for Afaan Oromo.

For the crawling purpose, JSpider, an open source java based crawler, is used. The crawler is first made to start with initial (seed) URLs and download the contents. Next, links that can be used as next URLs are extracted from the collection of downloaded documents and again documents on these URLs are downloaded. Finally, Afaan Oromo documents are filtered from the downloaded documents using a categorizer that is developed during this research.

Lucene's indexing capability is utilized to index the filtered Afaan Oromo documents. Stemming for inflectional and attached suffixes, removing of stopwords, and handling of short forms of compound words are incorporated to modify the analyzer component of Lucene's indexer as it is language specific.

Lucene's searching capability to retrieve matched documents from the index after preprocessing on queries is also utilized in the query engine component of Afaan Oromo search engine. For

ranked display of matched documents in the index, the role of Lucene's TF-IDF was vital. JSP is used to create the search and result display interfaces.

Finally, to evaluate the performance of the search engine, 70 document collections are randomly selected from list of downloaded documents along with 10 queries. According to precision-recall measures employed, 76% precision on the top 10 results and an average precision of 93% are obtained.

## 8.2 RECOMMENDATIONS

In this research, an attempt is made to design and develop a search engine for Afaan Oromo texts. Arriving at a full-fledged search engine is time consuming and involves coordinated team effort from computer science/information science professionals that worked (or wish to work) on different subcomponents, linguists for clear understanding of the language characteristics, etc. The following are some of the recommendations for further research and improvement:

- Attempt to filter Afaan Oromo documents is made offline during this research work. As a result, the search is also made offline. But this requires large storage capacity for storing downloaded documents. So, mechanism to identify Afaan Oromo documents online may be researched and thereby enabling online search.
- When a user types a query, he/she may misspell a word and as a result no/wrong document may be retrieved. Thus, Afaan Oromo spell checker can be incorporated in the search engine.

## REFERENCES


[1]. H. Moukdad, "Lost In Cyberspace: How Do Search Engines Handle Arabic Queries?" The 12th International World Wide Web Conference, Budapest, Hungary, 2003.

[2]. Tessema Mindaye, "Design and Implementation of Amharic Search Engine", Master's thesis, Addis Ababa University, Department of Computer Science, 2007.

[3]. Judit Bar-Ilan and Tatyana Gutman, "How do Search Engines Handle Non-English Queries? - A case study". In Proceedings of the Alternate Papers Track of the 12th International WWW Conference, Budapest, Hungary, 2003.

[4]. Paul McNamee, Charles Nicholas and James Mayfield, "Addressing Morphological Variation in Alphabetic Languages". In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, 2009.

[5]. Borkur Sigurbjomsson and Jaap Kamps, "Blueprint of a Cross-Lingual Web Retrieval Collection", Journal of Digital Information management, Vol. 3, No. 1, 2005.

[6]. Oromo Language. http://en.wikipedia.org/wiki/Oromo_language; Last accessed on August 25, 2009.

[7]. Kula Kekeba Tune, Vasudeva Varma and Prasad Pingali, "Evaluation of Oromo-English Cross-Language Information Retrieval", IJCAI 2007 Workshop on CLIA, Hyderabad, India, 2007.

[8]. Amit Singhal, "Modern Information Retrieval: A Brief Overview", IEEE, 2001.

[9]. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, "An Introduction to Information Retrieval", Cambridge University Press, Cambridge, England, 2009.

[10]. Ed Greengrass, "Information Retrieval: A Survey", 30 November 2000.

[11]. G. Tsatsaronis and V. Panagiotopoulou, "A Generalized Vector Space Model for Text Retrieval Based on Semantic Relatedness", Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL - Student Research Workshop), pages 70–78, Athens, Greece, April 2009.

[12]. Wakshum Mekonnen, "Development of a Stemming Algorithm for Afaan Oromoo Text," Master's thesis, Addis Ababa University, School of Information Studies, 2000.

[13]. How Internet Search Engines Work, http://computer.howstuffworks.com/search-engine1.htm, Last Accessed on February 5, 2010.

[14]. Sergey Brin and Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", In Proceedings of the 7th International WWW Conference, Brisbane, Australia, April 1998.

[15]. Jon M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment", Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.

[16]. How a Search Engine Works, http://www.infotoday.com/searcher/may01/ liddy.htm, Last Accessed on December 25, 2009.

[17]. J. Deepa Devi, Ranjani Parthasarathi and T.V. Geetha, "Tamil Search Engine," Sixth Tamil Internet conference, Chennai, Tamilnadu, August 2003.

[18]. Arabic Language, http://en.wikipedia.org/wiki/Arabic_language#Influence_ of_Arabic_on_other_languages, Last Accessed on March 10, 2010.

[19]. T. Rachidi, O. Iraqi, M. Bouzoubaa, A. Ben Al Khattab, M. El Kourdi, A. Zahi, and A. Bensaid, "Barq: Distributed multilingual Internet search engine with focus on Arabic language," *Proceedings of IEEE Conf. on Sys., Man and Cyber., Washington DC, October 2003*.

[20]. Kula Kekeba Tune and Vasudeva Varma, "Oromo-English Information Retrieval Experiments at CLEF 2007", Working Notes of CLEF Workshop, Spain, 2007.

[21]. Z. Li, and L. Xing, "Search the Chinese Web — Design and the Operation of Net-Compass," *Proceedings of the First Asia Digital Library Workshop*, 1998.

[22]. Oromo Language: Encyclopedia, http://en.allexperts.com/e/o/or/oromo_language.htm, Last accessed on April 10, 2010.

[23]. Diriba Megersa, "An Automatic Sentence Parser for Oromo Language using Supervised Learning Technique," Master's thesis, Addis Ababa University, School of Information Studies, 2002.

[24]. Culture and Tourism Commission of Oromiya, "Caasluga Afaan Oromoo," 3rd ed. , Addis Ababa, Branna Printing Enterprise, 2006.

[25]. Otis Gospodnetic and Erik Hatcher, "Lucene in Action," Manning Publications Co., 209 Bruce Park Avenue, Greenwich, 2005.

[26]. Apache Lucene - Index File Formats, http://lucene.apache.org/java/2_4_0/fileformats.html, Last Accessed on August 20, 2010.

[27]. Lucene Based Search Engine, http://www.shadocms.com/shadozoom/company/blog/2007/lucene-based-search-engine.cfm, Last Accessed on August 15, 2010.

[28]. Apache Lucene – Scoring, http://lucene.apache.org/java/2_3_2/scoring.html, Last Accessed on August 29, 2010.

[29]. Gerard Salton and Christopher Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing and Management: an International Journal, Vol. 24, No. 5, pp. 513-523, Pergamon Press plc, Great Britain, 1988.

[30]. Apache PDFBox - Java PDF Library, http://pdfbox.apache.org/, Last Accessed on July 26, 2010.

[31]. CyberNeko HTML Parser, http://nekohtml.sourceforge.net/, Last Accessed on July 29, 2010.

[32]. Debela Tesfaye, "Designing a Stemmer for Afaan Oromo Text: Hybrid Approach", Master's thesis, Addis Ababa University, Department of Information Science, 2010.

[33]. JSpider, http://j-spider.sourceforge.net/, Last Accessed on August 18, 2010.

[34]. C. Griefenew-Mewis, "A Grammatical Sketch of Written Oromo", Druckerei Franz Hansen, Bergisch Gladbach, Germany, 2001.

[35]. Why Remove Stopwords, http://www.fromzerotoseo.com/stopwords-remove/, Last accessed on March 12, 2010.

## APPENDIX: LIST OF AFAAN OROMO STOPWORDS.

| | | | | |
|---|---|---|---|---|
| akka | hanga | jechuun | ol | waan |
| akkam | henna | kan | oliif | waggaa |
| akkasumas | hoggaa | kanaaf | oliin | woo |
| akkum | hogguu | kanaafi | oo | yammuu |
| akkuma | hoo | kanaafuu | osoo | yemmuu |
| ammo | illee | kee | otoo | yeroo |
| an | immoo | keenya | otumallee | ykn |
| ani | innaa | keenyaa | otuu | yommii |
| booda | inni | keeti | otuullee | yommuu |
| booddee | isaa | keetii | saniif | yoo |
| dura | isaan | koo | silaa | yookaan |
| eega | isee | kun | simmoo | yookiin |
| eegana | iseen | malee | sun | yookinimoo |
| eegasii | ishee | moo | ta`ullee | yoom |
| ennaa | isheen | nu | tahullee | |
| erga | itumallee | nuti | tanaaf | |
| fakkeenyaaf | ituu | nuyi | tanaafi | |
| fi | ituullee | odoo | tanaafuu | |
| fkn | Jechaan | ofii | tawullee | |
| garuu | Jechuu | oggaa | utuu | |

## Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

**Declared by:**

Name: _____

Signature: _____

Date: _____

**Confirmed by advisor:**

Name: _____

Signature: _____

Date: _____

Place and date of submission: Addis Ababa, November 2010.