ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

Develop an Audio Search Engine for Amharic Speech web Resources

Arega Hassen Mohammed

Advisor: Solomon Atnafu (PhD)

A Thesis Submitted to the School of Graduate Studies of the Addis Ababa University in partial fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

October, 2019

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

Arega Hassen Mohammed

Advisor: Solomon Atnafu (PhD)

This is to certify that the thesis prepared by Arega Hassen Mohammed, titled: Design and Development of a Model for an Audio Search Engine for Amharic Speech Web Resources and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

APPROVED BY:
EXAMINING COMMITTEE:

1. Advisor: Dr. Solomon Atnafu, _____
2. Examiner: _____
3. Examiner: _____

ABSTRACT

Most general purpose search engines like Google and Yahoo are designed bearing in mind the English language. As non-resource rich languages have been growing on the web, the number of online non-resource rich speakers is enormously growing. Amharic, which is a morphologically rich language that has strong impact on the effectiveness of information retrieval, is one of the non-resource rich languages with a rapidly growing content on the web in all forma of media like text, speech, and video. With increasing number of online radios, speech based reports and news, retrieving Amharic speech from the web is becoming a challenge that needs attention. As a result, the need to develop speech search engine that handles the specific characteristics of the users' Amharic language query and retrieves Amharic languages speech web documents becomes more apparent.

In this research work, we develop an Audio Search Engine for Amharic speech Web Resources that enables web users for finding the speech information they need in Amharic languages. In doing so, we have enhanced the existing crawler for the Amharic speech web resources, transcribed the Amharic speech, indexed the transcribed speech and developed query preprocessing components for user text based query. As base line tools, We have used open source tools (JSpider, and Datafari) for web document crawling, parsing, indexing, ranking and retrieving and sphinx for speech recognition and transcription.

To evaluate the effectiveness of our Amharic speech search engine, precision/recall measures were conducted on the retrieved speech web documents. The experimental results showed that the Amharic speech retrieval engine performed 80% precision on the top 10 results and a recall of 92% of its corresponding retrieval engine. The overall evaluation results of the system are found to be promising.

Key Words: *audio search engines, audio information retrieval, Information Retrieval in Amharic language, Speech Crawler, Amharic speech Identification.*

Dedication

-To my grandmother ***Zewdie Endeshaw Yalew***

and

-To all people **who have played a role in shaping me into the person I am today.**

Acknowledgement

First and for most I would like to thank God for giving me health and patience in completing my thesis work. Next to this, I wish to express my sincere gratitude to my advisor Dr. Solomon Atnafu for his continuous and constructive advice and guidance throughout this work. Since the period of this thesis work, he has been showing me the way I have to go through and has also provided me important resources, such as related research papers. He has been also tackling with me in showing the route when we encounter challenges in relation to tools.

I would like to thank Mekonnen Assefaw and Sara Abebe for reviewing and offering crucial feedbacks.

I am also highly thankful to Hafte Abera for his advice to work on this problem area, my classmate Masereshaye and Tehitena, for their technical support and advice. I also need to express my special thanks to my friend Dani Bekele.

Since I cannot list all, I thank all who helped me in the accomplishment of my work directly or indirectly.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Acronyms and Abbreviations

ASR	Automatic Speech Recognition
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HTK	Hidden Markov Model Tool Kit
IDF	Inverted Document Frequency
IR	Information Retrieval
KNN	K-nearest-neighbor
LVCSR	Large Vocabulary continuous Speech Recognition
MFCC	Mel Frequency Cepstral Coefficient
MIME	Multipurpose Internet Mail Extension
MPEG	Moving Picture Expert Group
OOV	Out of Vocabulary
SCR	Spoken Content Retrieval
SHINX	Site-oriented Processor for HTML INformation eXtraction
STFT	Short Time Fourier Transform
TF	Term Frequency
VSM	Vector Space Model
WWW	World Wide Web

Chapter One: Introduction

1.1 Background

Everybody needs information in his/her day to day life to make better decisions. In each of our personal activities, decisions are required and information is needed to support the decisions. Information is needed in fundamentally every field of human thought and action; and is stored in different media forms, including the web (WWW) to make it accessible to everyone and everywhere.

The web is not centralized by design. This is not a problem for casual browsing, but introduces difficulties for users attempting to locate specific information relating to topics and interests. Locating specific information related to topics and interest need is usually addressed by the use of a search engine [1].

A search engine is a reasonable solution for finding information from a dynamic, heterogeneous and vast information collection. Search engines are software that retrieve information from the web. Most of them are general purpose search engines and works only for the resource rich languages like English. There are other Language Specific search engines which are designed and developed to work with a particular language [1]. Amharic is one of those languages a search engine is designed for [2, 3, 4, 5]. To use search engines, users usually submit a query, typically a list of keywords, and receive a list of web pages that may be relevant to their query, usually pages that contain the keyword(s). Nowadays, search engines become a first choice for finding new information on the web.

Search engines like Google, Yahoo or Bing are very popular because they are easy to use [5] and achieve good results for everyday common search tasks. Most of these general purpose search engines are optimized for English language [2, 5, 6]. However, the web is designed not only for English. Currently, the total number of web pages in non-English languages is much more than that of English pages [2] and Amharic is one of these languages. As stated in [7] Amharic language is a morphologically complex language that has impact on the effectiveness of information retrieval.

The web contains audio data which is available at various sources such as recordings of meetings, broadcasts, stories, novels, lectures, seminars, conversational, party, blogs, counseling, religious and political speeches and songs etc. and the amount of audio data on web is increasing. By indexing and searching the digitized audio content users can get the most out of this materials. In the past, it was difficult to recognize, interpret, and analyze digitized speech, companies had to create and manually analyze written transcripts of audio content [8, 9].

Moreover, most of audio data is unstructured and unorganized. Just as the text data could be accessed quickly and easily with the support of text indexing and search, audio data can also be accessed quickly by providing the query in text/audio modes. Organizing, indexing, retrieving and summarization of large corpora of audio data raised challenging research issues for engineers and scientists as most of the future digital libraries may consist of audio and video data [9].

The increasing amounts of multimedia data in Amharic brings new challenges to information retrieval (IR). Traditional IR, familiar from the popular web search engines, offers the ability to quickly locate and browse large amounts of text using the familiar search and ranked by similarity interface. Unfortunately, for multimedia/speech there are no widely used similar techniques [10].

There are different approaches for multimedia information retrieval. The first is to generate textual indices automatically, semi automatically or manually and then use traditional IR. The other approach is to use content-based retrieval were the query is non-textual and a similarity measure is used for searching and retrieval.

In retrieval by similarity system for sound data, users can search for and retrieve sounds by perceptual and acoustical features, can specify classes based on these features and can ask the engine to retrieve similar or dissimilar sounds.

The indexing of a text corpus is done by using inverted file index format where words are used as units. One way to index the speech data is also at the word level or subword level provided we have the transcription or could generate the transcription. Other methods of indexing the speech data such as using automatically derived units, prosodic (the study of rhythm, intonation, stress, and related attribute in speech) information, acoustic-phonetic hints, global phone set, could also be explored.

Microphone, broadcasting mediums such as TV, Radios, studio quality recordings as in movies, telephone, mobile-phones, wireless devices used in military and defense applications are some of the mediums through which the speech data could be collected. Until recently, we don't have a better chance to access, once a program had finished its broadcast. Recording radio broadcasts and offering these archives over the Internet has created a second option for such programs. Unfortunately, the end-user does not get a chance to listen interesting programs. Clearly there is a need for a means to search and browse audio.

1.2 Motivation

General purpose search engines are easy to use and accomplish good results for everyday common search tasks and most of these general purpose search engines are optimized for English language. Recently, Google developed a voice-based interface to enter a query in voice form by using a microphone and retrieve information from the web. i.e., it has a voice based search interface. But this does not work for the Amharic language speech [11].

A major search engine, Google, announced the world's first billion-page index in 2015. The number of search knows about over 130 trillion pages in 2013 and indexed web contains 4.66 billion as of May 2015. These figures imply a text collection and a corresponding index in the order of hundreds of terabytes, which can only be stored in clusters of tens of thousands of computers. Clearly, this evolution of Web data puts more pressure on satisfying the user requirements; i.e., finding accurate results from the largest possible coverage of the Web, and doing it fast.

Subsequently, in the last two decades, a number of methods are proposed to improve the efficiency and scalability of a search engine. Paradigms from parallel and distributed processing are exploited for all components of a search engine, to cope up with the growth of data [12]. Furthermore, new approaches for crawling, indexing and query processing are introduced to improve the efficiency.

One such paradigm is prioritizing the Web pages and crawling only the valuable regions of Web, where the definition of a page's value depends on the specific application. Such focused crawlers cover only a specialized portion of Web and avoid the cost of crawling the entire Web, which is far beyond the capacity of individuals or institutions other than the largest players in the industry.

The idea of focused crawling can be used to generate topical search engines that aim to provide high quality and up-to-date query results in a specific area for their users.

Such new approaches are also proposed for the other two components, namely indexer and query processor, of the search systems. For instance, a survey on inverted index files demonstrates that it is possible to significantly optimize these components by a number of techniques from recent research [12] . In comparison to a straightforward implementation, such techniques can reduce the disk space usage up to a factor of five, memory space usage up to a factor of twenty, query evaluation time in CPU by a factor of three or more, disk traffic by a factor of five in volume and two or more in time and index construction time by a factor of two [12].

The efficiency issues for search engines are also important to be able to provide higher quality results [12]. That is, developing efficient strategies for the components of a search engine may allow reserving more computing, storage and/or networking resources for improving the result quality. For instance, efficient crawling strategies may increase the coverage of the search engine, or efficient query processing strategies may allow more sophisticated ranking algorithms.

A considerable amount of speech resources are available in the Amharic language on the web, users can not fully search and retrieve these resource in the language. This is due to Amharic language websites are developed using different character encodings. The use of different encodings create an additional challenge for information retrieval on the web [3].

Due to the increase in Internet user by Amharic speakers within the country and abroad, the number of web documents that are written in Amharic language and Amharic speech resources are also increasing. For this to justify, it suffices to mention the increasing number of online radio websites in Amharic language such as Sheger (FM 102.1), Bisrate (101.1), Fana (98.1), etc. However, there is no audio search engine that is specifically designed for these Amharic audio resources.

From the above paragraphs we can clearly see that all the general purpose search engines do not consider audio query of Amharic audio resources. Hence, the increasing number of Amharic audio resources on the web and lack of audio search engine that is specifically designed for Amharic language motivates us to do this research.

1.3 Statement of the Problem

Each natural language has its own characteristics and features [2]. It is difficult to follow the same searching pattern and apply the same searching rules for all languages [4]. Different audio searching, as well as individual exceptions, need special handling and a careful formation of a frame with specific rules, applied on the particular language. There exists some literature on the principles, methodologies, and problems involved in the application of audio searching algorithms. However, little attention has been given to audio searching of Amharic language.

Currently, there are a number of audio web documents available in the web that are in Amharic language. Since general search engines are optimized mainly for English, they do not search well those documents that are in Amharic [2]. Even if the increasing availability of audio information in Amharic language on the web, there have been only a few attempts to discover issues related to how search engines can handle these Amharic audio resources on the web to the effects of the characteristics of the language for information retrieval.

Speech processing covers a set of technologies including speech coding, text to speech or speech to text conversion, speaker recognition and, speech recognition. Even though speech coding techniques are universal, the other technologies are highly dependent on the language [13].

Amharic language is morphologically complex and has many distinct features that affect the information retrieval of the language's documents on the web [7]. General purpose search engines don't handle non–English queries [4] such as Amharic, Arabic or Hindi. Moreover, there is no search engine that is developed to handle Amharic speech resources. Therefore, speech search engine that fulfill the mentioned requirements need to be developed for Amharic language.

At the end of this study the following research questions are answered.

- What are the challenges in building speech search engine for Amharic language web resources?
- Is it possible to design effective and efficient algorithm that identifies the Amharic speech?
- To what extent the proposed search engine retrieve Amharic speech from the web?

1.4 Objectives

General objective

The general objective of this work is to develop a search engine of audio web resources that specifically deals with Amharic speech searching.

Specific objectives

Specifically, the purposes of this thesis are to:

- review related works in audio based web retrieval,
- review Amharic speech processing and identify specific characteristics of the language,
- identify the audio language identification methods and adopt the one that fits for Amharic speech identification,
- To construct a general architecture of web based Amharic speech search engine.
- develop a prototype for Amharic speech search engine, and
- Test and evaluate the prototype developed.

1.5 Methods

A design science methodology is followed as research method. Accordingly, this work involved the study of issues and areas closely related to the thesis work in order to gain a deeper understanding of the problem and process of information retrieval and speech recognition. Major activities performed in this research work are:

- **Literature review**: Comprehensive literature review is conducted to get a deeper understanding of crawler, indexer, query engine and review works done on Amharic ASR.
- **Tools**: The programming part of the search engine is done using Java programming language. The reason to use Java is the exposure of the researcher to the language and to the fact that more Web-based tools that can be used for this research are java based. Solr, 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. Apache Tomcat server has been used to host Datafari that has been used to create the search interface.

- **Data Preparation Techniques**: speech data corpus is needed for automatic speech recognition system. The speech data corpus will be divided into training data and test data. The test data will further divided into: words selected from the training data and words not included in the training data. These data will be manipulated using Sphinx.

1.6 Scope and Limitations

Scope

The study focuses on the development of a speech search engine model for Amharic web speech resources.

We have attempted to carry out four major activities. First of all, we have tried to investigate the basic units of crawler for Amharic and has identified possible challengers. Second, Amharic sub-word speech recognizers have been built and the Amharic speech is transcribed. Third, the recognized speech has been indexed. Fourth, text based query and user interface has been designed. The prototype is designed, tested and evaluated. Further examination and conclusions have been made based on the results obtained from the investigation.

The scope of the study is limited to Amharic speech resources, and it is based on sub-word based automatic speech recognition. The study will not necessarily consider speeches in a video, text and image contents of the web and will not also return these results.

Limitation

The main limitation of this study was the absence of a ready-made Amharic reference database or corpus for speech recognition researches. We have prepared speech database and deal with the rest of the experiment based on this database. Though every possible effort has been made to impart good coverage and phonetic balance into the database, experimental results have demonstrated that more effort is required to prepare a phonetically balanced database with good coverage.

1.7 Application of Results

Amharic speech search engine is specifically used for retrieval of Amharic audio documents available on the web. As a result, the search engine can be used by the users of the language and those who want to use Amharic language documents on the web for their purpose.

1.8 Organization of the Thesis

The remaining part of this work is organized as follows. Chapter 2 and Chapter 3 present literature review and related works, respectively in the domain. The fourth chapter deals with the proposed solution of the spoken document search engine of Amharic web resources. Chapter 5 presents the Experimentation/Prototype and Results. Chapter 6 presents discussion of the results. Finally, conclusions, recommendations, and future work are given in chapter 7.

Chapter Two: Literature Review

2.1 Introduction

In this chapter, we review literature in the area of spoken document search engines. To describe issues related to the subject at hand, we investigate the literature on basic concepts, theory and practices behind the implementation of search engines by focusing on a wide range of topics including information retrieval, speech recognition, search engine and spoken document retrieval.

2.2 Information Retrieval

Currently, there are challenges of dealing with a huge amount of electronic documents that are in text, audio, video, and image. We are good at creating of them, but not as capable at managing their information content which are stored in some devices such as hard drives and CD-ROMS to form large collections. It should be stored in an organized format such that the access of such information is easy and well presented to users. An information retrieval system informs on the existence or non-existence and locations of documents relating to a request [14]. Therefore, the aims of information retrieval is to help people find out whether the information they are interested in exist or not.

Information retrieval (IR) has become an important problem for searching and managing massive amounts of information available on the web [15]. Much effort in IR has been devoted to textual information developed in [2, 3, 4] for Amharic, but since information comes in the forms, such as music, lectures, speech, radio and television broadcasts, etc., retrieval has become a topic of interest [16].

Information retrieval (IR), defined by [17], is designed to facilitate access to stored information. Moreover, it is concerned with the representation, analysis, searching, storage, retrieval and organization of information items [18]. The components of an IR consists of a set of information items, a set of requests and a mechanism to determine which information items are most likely to meet the requirements of the requests. Mostly, users state their information needs in the form of queries and submit their queries to the information retrieval system. Based on their queries, the system outputs information items that are relevant to the user query. This takes place when a matching exists between queries and information items. To attain a match between these two

components, the items in the documents as well as in the query should be represented in some form.

Differences in IR systems can be made based on their effectiveness in relation to users query. The effectiveness of an IR system depends on the indexing and searching techniques employed. The search result from a given query can be an indication of how effective a given information retrieval is. The higher the number of correct responses to users query, the more effective is the IR system and vice versa.

In information retrieval, the stored information items and the incoming search requests are represented by sets of content identifiers known as keywords, index terms, or simply terms [17]. A typical development of information retrieval is to find out a particular interested document from a group of over million or billion documents by matching a user-specified information, normally a query. An information retrieval system is unlike a question-answering system [19], the results returned by the IR system to the user are a list of ranked document rather than an exact answer to the query. The IR system may return a document of a short title with a brief abstract. In this case, users can judge whether a retrieved document is relevant by reading the short title or brief abstract, before they proceed to read the contents of the whole document.

As mentioned, ordinary Information Retrieval (IR) research has been mainly based on text [20, 21], which is familiar through the popular web search engines such as google, yahoo, MSN, Lycos or AltaVista. The common IR problem is to locate desired text documents using a search query consisting of a number of keywords. Typically, matching documents are found by locating query keywords within them. If a document has a high number of query terms, it is regarded as being more relevant to the query than other documents with fewer terms. Documents can then be ranked by relevance and presented to the user for further exploration, as the web search engines do. On the other hand powerful IR algorithms are available for text, it is clear that for audio, or multimedia in general, common term-matching approaches are useless due to the simple lack of identifiable words or comparable entities in audio documents. The problem becomes even more undecided when one considers audio, such as music, voice, which may have no speech.

As multimedia is becoming increasingly available on the WWW, more and more applications are being developed to process multimedia objects. This process generally consists of storage,

indexing, retrieval and presentation of multimedia. Much of the research in this area deals with the technically pointed and yet-to-be-resolved task of content-based retrieval which refers to the automatic recognition of the content of the medium [22].

Even when a desired audio document can be located in a large archive, another problem to overcome is the linearity of audio files. In order not to miss important parts of an audio, the entire audio file must be heard from start to finish, which takes significant time. In contrast, the transcription of a minute-long message is typically a paragraph of text. A typical interface treats audio as an undifferentiated stream: the tape recorder symbol (with stop, play, rewind, and forward buttons) is ubiquitous. In contrast, most text processing software has a “find" command that does simple string-match word searching to locate desired information in large files. An audio interface of similar flexibility is impossible unless suitable indexing entities, analogous to “words" or “pages" can be located in the audio. Queries are analyzed by means of natural language processing and tokens have to be mapped to the corresponding concepts [23]. Using the common approach to measuring effectiveness of retrieval in terms of the relevance of retrieved documents, the documents are divided into two mutually exclusive sets, relevant and non-relevant. Precision and Recall are then calculated for the system according to the following formulae:

$$Precision = \frac{relevant\ documents| \cap |documents\ retrieved|}{|document\ retrieved|} \quad \text{and} \quad (1)$$

$$Recall = \frac{|relevant\ documents| \cap |documents\ retrieved|}{|relevant\ documents|}$$

In equation 1 above, the precision provides an indication of the quality of the answer set. For example, for a text search on a set of documents, precision is the number of correct results divided by the number of all returned results. Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system.

Precision is used with recall, the percent of *all* relevant documents that is returned by the search. For example, for a text search on a set of documents, recall is the number of correct results divided by the number of results that should have been returned. Recall can be viewed as the probability that a relevant document is retrieved by the query. It is trivial to achieve recall of 100% by returning

all documents in response to any query. Therefore, recall alone is not enough but one needs to measure the number of non-relevant documents also, for example by also computing the precision.

Retrieval systems, search engines, browsing tools, and other information management software are selecting relevant documents or information from the collections. When present-day retrieval and information selection tools operate on the content of document texts or make it accessible, they are not sufficiently powerful to identify documents or information that might be relevant to their users.

Content-based access to spoken audio information is a desirable technology due to spoken audio information which is a wealth of information available to us only in audio format. Obvious examples are audio broadcasts but audio also forms an important component in retrieval from video.

Audio information can be very difficult to analyze due to speaker difference, various types of background noises, the speech is continuous and quality of audio, like in telephone interview broadcast over the radio, can be poor. In developing a system to perform retrieval from digital audio information we have avoided some of these problems by using radio news bulletins [24], whereas the quality of studio broadcast is good.

2.3 Information Retrieval from Audio Documents

IR from spoken document is very different to retrieve from text, this is because of the fact that audio is a continuous stream, not just the medium and the errors inherent in any speech recognition system. Most IR on text is performed on sets of individual documents which are regarded as being independent of each other. Documents may be broken down into smaller logical units for retrieval if they are large.

In spoken audio recording applications, natural boundaries between independent elements of the audio such as stories in a news do not form an inherent part of the audio itself and individual units are concatenated together without markup or significant pauses. For instance, in a radio news bulletin there is no indicator of the boundary one story and the next although an individual bulletin will contain a relatively small number of stories. This would be analogous to concatenating individual text documents together and removing markup indicators such as title fields, author names, etc.

One of the first spoken document retrieval systems was developed in [25]. Here a recognizer was developed for Swiss-German radio news broadcasts that recognized sub-word units of pronunciation and broadcast were divided into fixed length overlapping windows which were indexed by tri-phones or triples of adjacent phones. A single broadcast contains multiple concatenated stories. The entire broadcast is divided into fixed length overlapping windows and to treat each window as an independent document for indexing and retrieval. This is due to using individual phones would be equivalent to retrieving text based on matching individual letters in queries and documents [24].

Using tri-phones as a representation for audio broadcasting has been reported in [26] which indexed an archive of spoken news bulletins by mono-phones, bi-phones and tri-phones and ran a series of experiments measuring retrieval effectiveness against the basic unit of representation. Their results confirmed a tri-phone representation of query and broadcast to produce the most effective retrieval performance.

2.3.1 Retrieval Models

Model is an idealization or abstraction of an actual process and the Mathematical models are used to study the properties of the process, draw conclusions, and make predictions using Mathematical concepts and languages [27]. The conclusions derived from a model depend on whether the model is a good approximation of the actual situation while a statistical models represent repetitive processes, make predictions about frequencies of interesting events. Models of information retrieval can serve as a blueprint to implement an actual retrieval system [28]. Retrieval models can describe the computational process, e.g. how documents are ranked, and Retrieval models can attempt to describe the human process, e.g. the information need, and interaction. Retrieval models have an explicit or implicit definition of relevance.

An IR model is a technique by which a relevance measure is obtained between a query and a document, and outputs a ranked by closeness to the query list of results. The difference between the frameworks lie in their choice of ranking function, which computes a ranking score for each of the result items. The Retrieval System component is most often taken to be a standard IR system, implementing one of the standard IR frameworks.

Boolean search: Boolean searching is named after George Boole who wrote about logical ways to formulate precise queries using connectors or logical operators between concepts. The true-false nature of Boolean logic makes it compatible with binary logic used in digital computers. Most old library systems have a long history of Boolean retrieval. It has become the conventional basis for searching most computerized systems. The basic operators are **and**, **or**, and **not**. **Near** is a modification of AND. The **Near** operator allows for the search of two terms that are near to each other without any requirements on the order of the words. Parentheses are used to organize the sequence and groups of concepts.

In the Boolean search, an item is returned as potentially relevant if its contents match very complex query constructed using Boolean operators. Since Boolean search does not generate a relevance score for each item, the set of returned results is not ordered by potential relevance. The searcher must either make use of other information, for example, date of creation, or abbreviated representation of their content to decide which items to inspect further.

A basic form of Boolean search query is a simple AND construction requiring all features in the query to present in the item in order for it to be retrieved. Taking a simple AND query, the Boolean search framework basically addresses a "finding mentions" task, looking for the presence of certain features in the items to be retrieved. AND returns too few documents (low recall), OR return too many document (low precision) and NOT eliminates many good documents (low recall).

Vector Space Model: This model is introduced due to the Boolean model cannot be able to rank documents well [28] due to simple queries do not work well and complex query languages confuse end users on one hand, and OR return too many document (low precision), AND returns too few documents (low recall), and NOT eliminates many good documents (low recall) on the other hand. Vector-space model (VSM) is one of the most commonly used strategy proposed by Salton in 1975. It was the first widely used ranked IR framework, and has been used in a number of spoken content retrieval (SCR) studies [29, 30, 31].

VSM generally involves the indexing on the documents, sum of the vectors corresponding to terms in the document and represented as bags of words, collection to form a list of indexed terms from the document. This list of indexed terms, i.e., document vectors are represented as a vector when used computationally where each component is an indexing term [32]. A vector is like an array of

floating point, which has direction and magnitude and each vector holds a place for every term in the collection. Therefore, most vectors are sparse (mostly represented by a linked list of nodes, each of which contains an integer index, and a pointer to the next node).

VSM makes use of a vector representation containing indexing features. One such representation is made for each item in the collection of available items. The elements of the vector contain the weights of the individual indexing features (i.e., terms). Each weight represents the importance of that feature for the item. The VSM is based on the assumption that the closeness of vectors within the vector space reflects the semantic similarity of the items that they represent. Within the VSM, the ranking score of an item, with respect to the query, is calculated as the similarity between a vector representing the item and a vector representing the query. Query document similarity in the VSM is calculated using the dot product between the vectors.

Documents (D) and queries (Q) are both vectors

$$D_i = (w_{i1}, w_{i2}, \dots, w_{ij}) \; and \; Q = (w_{q1}, w_{q2}, \dots, w_{qj}) \quad (2)$$

Each w_{ij} is a weight for term j in document i

Each dimension corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero. Several different ways of computing these values, also known as (term) weights, have been developed. One of the best known schemes is tf-idf weighting.

The definition of *term* depends on the application. Typically terms are single words, keywords, or longer phrases. If words are chosen to be the terms, the dimensionality of the vector is the number of words in the vocabulary, the number of distinct words occurring in the corpus.

Relevance rankings of documents in a keyword search can be calculated, using the assumptions of document similarities theory, by comparing the deviation of angles between each document vector and the original query vector where the query is represented as a vector with same dimension as the vectors that represent the other documents.

In practice, it is easier to calculate the cosine of the angle between the vectors, instead of the angle itself. Similarity (sim) of a document (D) vector to a query (Q) vector = cosine of the angle between them

$sim(D_i, Q) = \cos\theta = \frac{D_i.Q}{|D_i||Q|}$ Cosine is a normalized dot product (3)

Documents ranked by decreasing cosine value

$similarity(D, Q)$ = 1 when Document (D) = Query (Q)

$similarity(D, Q)$ = 0 when Document(D) and Query(Q) share no terms, Vector Space Relevance Measure is given by $\text{Document}(D_i) = W_{d_{i1}}, W_{d_{i2}}, \ldots, W_{d_{it}}$

$Query(Q) = W_{q1}, W_{q2}, \ldots, W_{qt}$ Where t is the total number of terms in the system and

W_{qi} is the weight associated with the i^{th} term in query q.

If term weights normalized: $sim(Q, D_i) = \sum_{ti \in Q,D} W_{tiQ}.W_{tiD}$ (4)

Where W_{tiQ} is the value of the i^{th} component in the query vector Q, and $W_{tiD}$is the i^{th} component in the document vector D. The summation can only be done over the terms that are common in the query and the document since the word that is not present in either the query or the document has a value zero.

There are several alternative schemes that can be used for calculating the weights of each term. Usually, these schemes make use of statistics calculated on the basis of term occurrences in the collection. A common Weights for Term Components, referred to as *tf-idf* uses *tf*, term frequency (the number of occurrences of a given term in a document), as the representative component.

How well does a term describe its document? If a term *t* appears often in a document, then a query containing *t* should retrieve that document

$$tf_{ij} = \frac{f_{ij}}{\max_j f_{ij}} \quad (5)$$

In the case of the term frequency tf_{ij}, the simplest choice is to use the *raw count* of a term in a document, i.e., the number of times that term *i* occurs in document j. Other possibilities include the inverse document frequency (IDF) measures the rarity of a term *t* in the document collection which is the inverse of the number of documents in the collection containing the given term, as the discriminative component. A detailed explanation of the VSM and presentation of alternative *tf* and *idf* component functions is given in [33]; this account is extended to incorporate a more effective method of query and document length normalization than in the standard cosine function

in [34]. So inverse document frequency (idf) is a discriminating measure for a term *i* in collection, i.e., how discriminating term *i* is. $idf_t = \log(\frac{N}{n_i})$, where *N* is the number of documents and n_i is the number of documents containing the query term t.

tf-idf weighting is the most common term weighting approach for VSM retrieval.

$$W_{dt} = tf_{dt} * idf_t \quad (6)$$

A term that appears in many documents should not be regarded as more important than one that appears in few documents. A document with many occurrences of a term should not be regarded as less important than a document with few occurrences of the term.

Normalize document length is a long documents having higher term frequencies i.e., the same term appears more often, but more terms: increases the number of matches between a document and a query Long documents are more likely to be retrieved. The cosine normalization in equation (3) lessens the impact of long documents.

Probabilistic Retrieval: In the Boolean or vector space models of IR, matching is done in a formally defined but semantically imprecise calculus of index terms. Probabilistic Retrieval is based on the Probability Ranking Principle [35], which states that the most effective retrieval system given the available data is the system that ranks items according to the probability of their relevance to the user's information need. In order to calculate the probability of an item being relevant to the information need, a particular model must be adopted. Retrieval status value (RSV) is computed by

$$rsv(d_j, q) \approx \frac{P(d_j|R)}{P(d_j|\bar{R})} \quad \text{where } \bar{R} \text{ is the set of irrelevant documents} \quad (7)$$

In equation 7, given a query Q, there exists a subset of the documents R which are relevant to Q, but membership of R is uncertain. A Probabilistic retrieval model ranks documents in decreasing order of probability of relevance to the information need.

Language Modeling Framework: Statistical language models estimate the distribution of words in an input language. In the context of information retrieval, a document is generally viewed as a sample from an underlying language model; terms in the collection are generated with specific probabilities. Language modelling approach has been known to be very effective for information

retrieval not only for text, but for spoken content as well [36]. Language modeling framework is one of the major IR frameworks [28] which is first proposed by Ponte and Croft in 1998. This framework takes a very different approach to the generation of a ranked list of potentially relevant items in response to a query. Documents are ranked by the likelihood that each document language model could have generated the user's query terms. The model essentially estimates the probability of a language model associated with each item generating the query.

$$RS = P(D|Q) = \frac{P(Q|D)P(D)}{P(Q)} \quad (8)$$

In equation 8, the basic idea for language modelling approach is that both documents and queries are respectively represented as language models D and Q, and the relevance score function RSL (D, Q) used for ranking the documents D where query Q is the inverse of the divergence between Q and D: This value is returned by the system as the ranking score.

The prior probability of all items is the same, and the prior probability of the query is the same for all documents so does not impact on the ranking of documents with respect to the ranking score. Since the language model used here is one that returns the likelihood of a query given a particular item, the Equation in 2 is referred to as the query likelihood model.

Subword Based Language Modeling: Since ASR system is able to recognize only the words that are in the recognition vocabulary, coverage is a critical point in language modeling. Amharic suffers from data sparseness and out of vocabulary words problems as it is a morphologically rich language. In Amharic, it is possible to obtain many different words from a single stem, by adding prefix, suffixes and postfix. As a result, word-based models result in high OOV rate in agglutinative languages; increasing the vocabulary size is a solution but it requires more memory and processing power [37]. In addition, high vocabulary size causes the language model to be non-robust due to the data sparseness. Much more amount of training data is needed to model a large number of words. Therefore, the feasible solution is using subwords as the language modeling unit, instead of words [37, 38].

2.4 Segmentation

Segmentation is accomplished by applying the acoustic analyses to the signal and looking for transitions. The transitions define segments of the signal, which can then be treated like individual

sounds. For example, a recording of a concert could be scanned automatically for applause sounds to determine the boundaries between musical pieces. Similarly, after training the system to recognize a certain speaker, a recording could be segmented and scanned for all the sections where that speaker was talking.

Segmentation techniques create information about spoken content that is useful for IR. Segmentation of spoken content can take place either based on direct analysis of the audio content or by using the ASR transcripts. The importance of forming audio and topical segments has been recognized in management of speech content. Segmentation using the audio content prior to recognition can help to improve the quality of the ASR transcripts. In general, ASR systems do not process a continuous string of speech, but rather disassemble it into smaller pieces, generating a hypothesis string for each. Segments created for the purposes of ASR may be of fixed length or may be divided at pauses (the speech signal drops off in energy). Segments may approximately correspond to utterances, but whether they will also be useful for SCR will depend on the application.

In general, the quality of the segmentation will be strongly dependent on the segmentation algorithm used and on the nature of the audio signal being segmented [36].

The use of segmented word can increase precision, thus can improve the quality of those added terms. Therefore, terms for extraction are indexed with segmented word within the side collection for selection purposes. The segmentation is based on the lexicon knowledge from audio data. However, for retrieval of the spoken document, indexing is still based on overlapping character. All segmented words will also form overlapping character before added back to the spoken document [39].

2.4.1 Audio Classification

One part of a speech retrieval system concerns identifying different audio classes. The main classes considered are speech, music, noise, and silence but depending on the application more specific classes such as noisy speech, speech over music, and different classes of noise, have been considered [40].

The task of segmenting or classifying audio into different classes has been implemented using a number of different schemes. Two aspects that must be considered are feature and classification model selection.

Different features have been proposed, based on different observations on the characteristics that separate speech, music and other possible classes of audio. The features are generally divided based on the time perspective they are extracted.

The simplest features proposed include time domain and spectral features. Time domain features typically represent a measure of the energy.

Cepstral coefficients have been used with great success in speech recognition systems, and subsequently have shown to be quite successful in audio classification tasks.

The other aspect to be considered is the classification scheme to use. A number of classification approaches have been proposed, that can be divided into rule-based and model-based schemes. The rule-based approaches use some simple rules deducted from the properties of the features. As these methods depend on thresholds, they are not very robust to changing conditions, but may be feasible for real-time implementations [40].

Model-based approaches have included Gaussian Mixture Model (GMM), K-nearest-neighbor (KNN), Hidden Markov Models (HMM), and the time sequence of features, or the probability model.

2.5 Speech Recognition

Speech Recognition is the process of converting speech signal to a sequence of words by means of algorithm implemented as a computer program. Speech recognition process aims to give a machine the ability to hear, understand and act upon spoken information. Speech recognition concentrates on recognition of spoken word, recognition of speakers, recognition of spoken language and recognition of emotion. Music information retrieval works on analyzing the structure of music and retrieving similar pieces of music, instruments and musical genres. Environmental sound retrieval includes types of sounds neither music nor speech [41].

There are four basic approaches for developing automatic speech recognition system:

Template Matching: it is a whole-word matching which is the most widespread, and commercially available approach to automatic speech recognition. The method involves obtaining one or more utterances of every word that is to be recognized from one or more speakers and storing the spectrographic representations of these utterances via a voice coding (filtering and digitizing) process.

A whole-word template matching recognizer can only be successful on those words with which it has been trained. With a reasonably large vocabulary, it is obviously inefficient [42].

Stochastic Approaches: is the process of making a sequence of non- deterministic selections from among sets of choices. They are non-deterministic because the selections are not specified in advance but governed by the characteristics of the input.

Speech recognition can be viewed as a situation where selections have to be made based on uncertain information and stochastic modeling is a flexible and general method for handling situations where uncertainty overcomes.

Like template matching, stochastic processing requires the creation and storage of models of each of the items that will be recognized. However, unlike template matching stochastic processing involves no direct matching between stored models and input. Instead, it is based upon complex statistical and probabilistic analyses that are best understood by examining the network like structure in which those statistics are stored [43]. The hidden Markov model (HMM) is one popular stochastic approach that can be used to deal with speech recognition problems [42].

Neural Networks: are described as connectionist systems because of the connections between the individual processing nodes. They are also known as adaptive systems, because the values of these connections can change so that the neural network performs more effectively. People also refer to them as parallel distributed processing systems after the way in which the many nodes or neurons in a neural network operate.

Neural Networks are best classification systems. They specialize in classifying noisy, patterned and variable data streams containing multiple, overlapping, interacting, and incomplete signals [43]. Speech recognition is a classification task that has all of these characteristics, making neural networks a possible alternative to speech recognition.

Like stochastic and template matching techniques, neural networks require training. However, neural networks do not require that a complete specification of a problem be created prior to developing a network-based solution. Instead, networks learn patterns solely through exposure to large numbers of examples, making it possible to construct neural networks for auditory models and other poorly understood areas [42].

Neural networks use manually-segmented or labeled speech elements and often store speech items as whole units rather than as time sequences.

Knowledge-Based Approaches: The main idea is to compile and incorporate knowledge from a variety of knowledge sources so that the system imitators a human being.

The knowledge sources that are considered include [44]:

Acoustic knowledge – evidence of which sounds are spoken on the basis of spectral measurements and presence or absence of features.

Lexical knowledge – the combination of acoustic evidence so as to nominate words as specified by a lexicon that maps sounds into words or equivalently decomposes words into sounds.

Syntactic knowledge – the combination of words to form grammatically correct strings, according to a language model, such as sentences or phrases.

Semantic knowledge – understanding of the task domain so as to be able to validate sentences or phrases that are consistent with the task being performed, or which are consistent with previously decoded sentences.

Pragmatic knowledge – inference ability necessary in resolving ambiguity of meaning based on ways in which words are generally used.

All these knowledge sources interact in order to arrive at a decision about a speech sample. The crucial question, with such a complex scheme is “how do these levels interact?” [42].

Automatic speech recognition (ASR) has the following components.

Front End: The front end extracts data from the digital representation of the spoken words, setting it into a form the decoder can use. To make pattern recognition easier, the digital audio (spoken words) is transformed into frequency domain. In the frequency domain, frequency components of

a sound can be identified. From the frequency components, it is possible to approximate how the human ear distinguishes the sound. The transformation results in a graph of the amplitudes of frequency components, describing the sound heard.

Acoustic model: is building statistical models for some meaningful speech units based on the feature vectors computed from speech. The speech signal is first chunked into overlapping 20-30ms time windows at every 10ms and the spectral representation is computed from each frame. A commonly used feature vector consists of mel-frequency cepstral coefficients (MFCC) which is discussed in section 2.5.1.

The meaningful units of speech that are most often used in ASR are phones. Phones are the minimal units of speech that are part of the sound system of a language, which serve to distinguish one word from another.

The dominant approach to acoustic modeling in speech recognition is to use Hidden Markov Models (HMMs). An alternative to the standard HMM approach is a hybrid approach in which Artificial Neural Networks (ANN) and HMMs are employed. In order to recognize speech, the acoustic models should first be trained. During training, the parameters for the models are estimated from recorded speech material which has been orthographically transcribed (i.e., at word level). Moreover, a phonetic transcription of the words is needed. Transforming a word sequence to a phone sequence is accomplished by looking up the phonetic transcription for a word in the lexicon [44].

Pronunciation Model: consists of the orthography of words that occur in the training material and their corresponding phonetic transcriptions. It specifies the finite set of words that may be output by the speech recognizer. The transcriptions can be obtained either manually or through grapheme-to-phoneme conversion. A pronunciation dictionary can be classified as a canonical or alternative on the basis of the pronunciations it includes. For each word, a canonical pronunciation dictionary includes only the most probable pronunciation that is assumed to be pronounced in read speech. It does not consider pronunciation variations such as speaker variability, dialect, or co-articulation in conversational speech. On the other hand, an alternative pronunciation dictionary includes all the alternate pronunciations that are assumed to be pronounced in speech [45].

Language Model: which includes the dictionary, is used by the decoder to determine the most likely suggestion. The language model describes to the decoder, the relationship between words and the probability of words appearing in a particular order. Language models may be domain specific. The language model is referred to as the grammar when referencing context free implementations.

Its goal is to predict the likelihood of specific words occurring one after another in a given language. Typical recognizers use n-gram language models, most commonly trigram. An n-gram contains the prior probability of the occurrence of a word (unigram), or a sequence of words (bigram, trigram etc.):

unigram probability $P(w_i)$
bigram probability $P(w_i|w_{i-1})$
n-gram probability $P(w_n|w_{n-1},w_{n-2}, \ldots,w_1)$.

Decoder: The acoustic model is produced or prepared by the ASR training tools. The trainer processes audio files with their text transcriptions to extract common acoustic characters for each individual phoneme (context independent phoneme) as well as each phoneme with its context (context dependent phoneme). Actually, each phoneme has been divided into 5 states in a time sequence, since the characters are slightly different from begin to end. All phoneme-state based characters form a voice model.

Hence, the decoder is the speech engine that takes the acoustic model, the pronunciation model, the language model, and observation sequence and outputs the most likely sequence of words.

Feature Extraction: which is discussed in broad in section 2.5, converts the speech signal into a sequence of acoustic feature vectors. The goal is to extract a number of features from the signal that have a maximum of information relevant for classification. That means features that are robust to acoustic variation but sensitive to linguistic content are extracted.

2.6 Feature Extraction

Feature extraction is the process of obtaining sequence of vectors that is used to represent acoustic data in which input sound is digitized (sampled) with sampling rates 8 KHz for telephone speech and 16 KHz for direct microphone output [37]. A signal is sampled by measuring its amplitude at a particular time; the sampling rate is the number of samples taken per second. In order to

accurately measure a wave, it is necessary to have at least two samples in each cycle: one measuring the positive part of the wave and one measuring the negative part. More than two samples per cycle increases the amplitude accuracy, but less than two samples will cause the frequency of the wave to be completely missed [46]. Audio feature extraction deals with the analysis and extraction of meaningful information from audio signals in order to obtain a compact and expressive description. It transforms the input waveform into a sequence of acoustic feature vectors, each vector representing the information in a small time window of the signal [46]. The speech signal is transformed into a sequence of acoustic feature vectors before it is processed by the speech recognition system. The vectors contain spectral features, which encode how much energy is present at different frequencies in the speech signal [46, 47]. These vectors are extracted from overlapping windows of the speech signal. Each vector is extracted from a signal window that is small enough to support the assumption that the speech signal is stationary (non-changing) in its duration. A typical window is 25ms in length and for each vector, the window is shifted forward by 10ms [48]. This overlap seeks to ensure that rapid changes in the input signal are captured in the feature vectors. In general, acoustic vectors are 39 components in length [48]. Commonly, the components are mel-frequency cepstral coefficients (MFCCs). Mel frequencies are frequency bands warped to approximate the sensitivity of the human ear. Perceptual linear prediction is also commonly used for spectral vectors, this is a linear prediction method that retains information in the signal that is relevant for human perception.

Feature extraction can be used to characterize a segment of audio by computing a numerical representation. This numerical representation, which is called the feature vector, is used as a fundamental building block of various types of audio analysis and information extraction algorithms. This vector has typically a fixed dimension and therefore can be thought of as a point in a multi-dimensional feature space. When using feature vectors to represent audio two main approaches are used. In the first approach the audio file is broken into small segments in time and a feature vector is computed for each segment. The resulting representation is a time series of feature vectors which can be thought of as a path of points in the feature space. In the second approach a single feature vector that summarizes information for the whole file is used. The single vector approach is appropriate when overall information about the whole file is required whereas the trajectory approach is appropriate when information needs to be updated in real time. For

example, classification of radio signals might use the trajectory approach. Typically the signal is broken in small chunks called analysis windows. Their sizes are usually around 20 to 40 milliseconds. That way the signal characteristics are relatively stable for the duration of the window [49].

Feature extraction is achieved using Time-Frequency analysis technique such as the Short Time Fourier Transform (STFT). Time-Frequency analysis technique basically represent the energy distribution of the signal in a time-frequency plane and differ in how this plane is subdivided into regions.

Short Time Fourier Transform Based Features

Features based on the Short Time Fourier Transform (STFT) are very common and have the advantage of fast calculation based on the Fast Fourier Transform (FFT) algorithm. Although the exact details of the STFT parameters used to calculate them differ from system to system their basic description is the same. The following features are based on the short-time magnitude of the STFT transform of the signal.

2.6.1 Mel-Frequency Cepstral Coefficients

There are various psychoacoustic models of pitch perception scales [50]. However, Cepstral Coefficients based on the Mel-scale are the most popular variant used today. The reason for MFCC being most commonly used for extracting features [50, 39] is that it is most nearest to the actual human auditory speech perception. MFCC [51] is used to recognize numbers automatically spoken into a telephone, airline reservation, voice recognition system for security purpose etc. MFCC is commonly used in speech recognition and speaker recognition systems. However, MFCC also proved to be useful in discriminating between speech and other sound classes, which explains its wide usage in the audio analysis and processing literature [52].

MFCC is performed on an interval of sampled speech data (frame) and returns a coefficient vector. The method includes a Fast Fourier Transform, the computation of MEL frequency cepstrum coefficients, and a cepstral analysis [44]. The final step of the signal processing stage is often a vector quantization that returns a vector code for every frame [53].

Some researchers have proposed modifications to the basic MFCC algorithm to improve robustness, such as by raising the log-mel-amplitudes to a suitable power (around 2 or 3) before taking the Discrete Cosine Transform (DCT), which reduces the influence of low-energy components. One common model in MFC is the **mel** scale [54]. A mel is a unit of pitch in which pairs of sounds which are perceptually equidistant in pitch are separated by an equal number of mels [46].

A variety of representations for speech signals have been proposed in different literatures. So far the most common in feature representation is the mel frequency cepstral coefficients (MFCC). MFCC [55, 56] is also the most popular features for large vocabulary continuous speech recognizer (LVCSR). The feature captures the short time segment statistics based on models of human auditory perceptive system with respect to different frequency range. These are based on the important idea of the cepstrum.

Most information in human speech is in frequencies below 10,000 Hz; thus a 20,000 Hz sampling rate would be necessary for complete accuracy [57]. But telephone speech is filtered by the switching network, and only frequencies less than 4,000 Hz are transmitted by telephones [46]. Thus an 8,000 Hz sampling rate is sufficient for telephone-bandwidth speech.

Mel-Frequency Cepstral Coefficients (MFCC) features that are also based on the STFT. After taking the log-amplitude of the magnitude spectrum, the FFT bins are grouped and smoothed according to the perceptually motivated Mel-frequency scaling. Finally, a Discrete Cosine Transform is performed.

The log spectral amplitudes are first mapped onto the perceptual, logarithmic *mel-scale*, using a triangular band-pass filter bank. Then, the output of the filter bank is transformed into MFCC using the discrete Cosine transform (*DCT*).

MFCCs are usually computed as follow. At first the Fourier transform of the window input signal is computed (a short-time Fourier transform). Then a mel-filter bank, consisting of logarithmically positioned triangular base-pass filter is applied. After taking the logarithm of the magnitude of the band-pass filtered amplitudes, the cosine transform is taken in order to obtain MFCCs [50] .

2.6.2 MPEG based features

The Moving Picture Experts Group (MPEG) is a working group of the development of standards for digitally coded representation of audio and video.

Most of audio data available on the web is existing in compressed form following the MPEG audio compression standard. There are several standards of MPEG: MPEG-1 standard is used for Video CDs and also defines several layers for audio compression, one of which is the very popular MP3 format. MPEG-2 is another standard for video and audio compression and is used in DVDs and digital TV broadcasting. MPEG-4 is a standard for multimedia for the fixed and mobile web. MPEG-7 defines the Multimedia Content Description Interface and is the standard for description and search of audio and visual content. MPEG-21 defines the Multimedia Framework. The MPEG-7 standard [49, 40] describes a number of low level audio descriptors as well as some high-level description tools.

The five defined sets for high-level audio description are partly based on the low-level descriptors and are intended for specific applications; description of audio signature, spoken content as well as for general sound recognition and indexing.

The low-level audio descriptors comprise 17 temporal and spectral descriptors, divided into seven classes [40]. Some of them are based on basic waveform or spectral information while others use harmonic or timbral information.

MPEG audio compression is a loss perceptually-based compression scheme that allows audio files to be stored in approximately one tenth of the space they would normally require if they were uncompressed [49].

This enables large numbers of audio files to be stored and streamed over the network. Using current hardware the decompression can be performed in real-time and therefore in most cases there is no reason to store the uncompressed audio signal. Traditional Computer Audition algorithms operate on uncompressed audio signals therefore would require the signal to be decompressed and then subsequently analyzed. In MPEG audio compression the signal is analyzed in time and frequency for compression purposes. Therefore using this analysis not only for compression but also to extract features directly from compressed data is a good idea. Because the bulk of the feature

calculation is performed during the encoding stage this process has a significant performance advantages if the available data is compressed.

This idea of calculating features directly from MPEG audio compressed data was explored in [58] and evaluated in the context of music/speech classification and segmentation.

2.7 Search Engines

The first web search engine was the World Wide Web Wanderer, started in 1993, at which it possessed a crawler which traversed thousands and tens of thousands of web sites available on the World Wide Web, and creating own index of their content [59]. Search engines (SEs) [60] are the largest data management systems in the world; although there are larger databases in total storage there is nothing close in query volume. A search engine is the practical application of information retrieval techniques to large-scale data collections. Search engines can be found in many different applications, such as desktop search or enterprise search.

The general workings of web search engines are quite similar to each other [59]. In a first step, a web crawler, starting from a number of pages, follows outbound links, and so attempts to crawl the entire web, copying the content of the visited pages into the search engine's storage. These contents are then indexed, and made available for retrieval. On the other hand, the user enters search terms, so a ranking mechanism attempts to find results in the index which will most probably satisfy the user, and to rank them in a way that will maximize this satisfaction [61].

Search engine is often used to describe crawler-based search engine, human power directories, Hybrid Search engine and Meta Search Engines. These types of search engines gather their listings in different ways as in the crawler-based search engines, such as google, yahoo, create their listings automatically. They crawl or spider the web and users search through what they have found, while human-powered directories, such as the open directory (http://dmoz.org), depending on humans for its listing. Users submit a short description to the directory their entire site, or editors write one for sites they review. A search looks for matches only in the descriptions submitted. These are rarely used at large scale. But these are useful in the organizations where small scale of data is dealt with [62]. Hybrid Search engine or Mixed Results used to be either presented crawler-based results or human-power listings. Today, it is common for both types of results to be presented. Usually, a hybrid search engine will favor one type of listing over another. For example, MSN

search is more likely to present human-powered listings from Looksmart. However, it does also present crawler-based results as provided by Inktomi, SubmitExpress. Meta search engines fetch results from other search engines. The fetched results are combined and ranked again according to their relevancy. These search engines were useful when each search engine had a significantly unique index and search engines were less practicality. Because the search has improved a lot, the need for these has reduced. Meta Crawler and MSN Search are some examples.

There are three parts (components) of a Search Engine.

• Crawler, spider, ant, robot, wanderer or worm

• Index, catalog or database

• Query engine component

The explanation of these are shown below:

2.7.1 Crawlers

A web crawler is a computer program that collects or downloads information or documents (such as audio, text, video, image, etc.) on the web and FTP servers whereas crawling or spidering is finding and downloading web pages automatically [18]. Web crawlers [63, 64] are almost as old as the web itself and the first crawler was written in 1993 by Matthew Gray Wanderer. Crawlers are used to index documents and supply the user with a way of searching the web.

The original Google crawler [63, 64] developed at Stanford consisted of five functional components running in different processes. Firstly, a URL server process read URLs out of a file and forwarded them to multiple crawler processes. Second, each crawler process ran on a different machine, was single-threaded, and used asynchronous Input/output to fetch data from up to 300 web servers in parallel. Third, the crawlers transmitted downloaded pages to a single store server process, which compressed the pages and stored them to disk. Forth, the pages were read back from disk by an indexer process, which extracted links from HTML pages and saved them to a different disk file. Finally, a URL resolver process read the link file, removes the URLs contained, and saved the absolute URLs to the disk file that was read by the URL server. Typically, three to four crawler machines were used, so the entire system required between four and eight machines [63, 64].

The web crawler has two jobs: downloading pages and finding URLs. The crawling is traversing the web by recursively traversing links from a starting URL(s) which are the starting point though which any crawler begins searching procedure. This set of starting URL is known as Seed URLs. The crawling method starts with a given URL (seed), extracting links from it and adding them to an un-visited list of URLs. This list of un-visited links or URLs is known as, Frontier (Processing Queue). Each time, a URL is picked from the frontier by the Crawler Scheduler. This process continues until the crawler either runs out of disk space to store pages or runs out of useful links to add to the request queue. Once a page has been fetched, we need a parser to parse its content to extract information that will feed and possibly guide the future path of the crawler. The job of any parser is to parse the fetched web page to extract list of new URLs from it and return the new un-visited URLs to the Frontier.

Fetching many pages at once is good for those running the web crawler, but not necessarily good for the person running the web server on the other end. If the web server is not very powerful, it might spend all of its time handling requests from the crawler instead of handling requests from users. So, the crawler carefully chooses at each step about which page to index. Some policies were introduced to guide the crawler. They are [65, 66, 67]

- Selection policy states which pages to download
- Revisit policy states when to check for changes in web pages
- Politeness policy states how to avoid overloading of web sites, and
- Parallelization policy states how to coordinate the different web crawlers distributed.

Selection policy

As a crawler always downloads just a fraction of the web pages from the enormous web pages, it is highly desirable that the downloaded fraction contains the most relevant pages and not just a random sample of the web. Designing a good selection policy has an added difficulty: it must work with partial information, as the complete set of web pages is not known during crawling [67]. This requires a metric of importance for prioritization of web pages. The importance of a page is a function of its quality, popularity in terms of links or visits and even of its URL. Therefore, one possible selection policy can be crawling web pages with high PageRank from different communities in less iteration in comparison with a crawl starting from random seeds.

Revisit policy

Due to the fact that pages change over time as it has a very dynamic nature and crawling a fraction of the web can take a really long time, usually measured in weeks or months, By the time a web crawler has finished its crawl, many events could have happened such as creation, updating and deletions could have happened by the time the crawler has finished its crawl. So, crawlers have to revisit pages already indexed. Revisiting policies can be split into two categories, Uniform policy and Proportional policy. Uniform policy revisits every page with the same frequency regardless to the rate of change. Proportional policy revisits pages with a higher rate of change more often. For the second strategy the frequency of change has to be estimated. From the search engine's point of view, there is a cost associated with not detecting an event, and thus having an outdated copy of a resource. There are two most commonly used cost functions.

- Freshness and
- Age

Freshness

Web pages are constantly being added, deleted, and modified. To keep an accurate view of the web, a web crawler must continually revisit pages it has already crawled to see if they have changed in order to maintain the *freshness* of the document collection. The opposite of a fresh copy is a *stale* copy, which means a copy that no longer reflects the real content of the web page.

The freshness of a page p in the repository at time t is defined as:

$$F_p(t) = \begin{cases} 1 & \text{if p is equal to the local copy at time t} \\ 0 & otherwise \end{cases}$$

Age is a better metric to use. It is a measure concerned with how outdated the local copy is. The age of a page p in the repository at a time t is defined as:

$$A_p(t) = \begin{cases} 0 & \text{if p is not modified at time t} \\ \text{t} - \text{Modification time of p} & \text{otherwise} \end{cases}$$

The crawler should maintain the average age of pages as low as possible and the average freshness of the pages as high as possible.

Politeness policy

This states how to avoid overloading web sites [67]. Crawling the web has high costs for the public. These costs include network resources because crawlers use a lot of bandwidth over a long period of time. They can cause Server overload due to the number of requests to a given server. Some crawlers can even crash a server, if they are written badly. Robots that get deployed by individuals, may interfere with the network, if they are used by to many people. This behavior is no different to those of DOS (Denial of Service) or DDOS (Distributed Denial of Service) attacks. A partial solution to this problem is the robots exclusion protocol, which lets system administrators define which part of the server is accessible to the robots. This protocol does not define the interval of revisits or the amount of parallel downloads the crawler starts. And robots have to be written in a way, that they follow the servers' robot exclusion settings.

Parallelization policy

Some Crawlers run multiple processes in parallel. This is done to maximize download rates. To avoid downloading the same URL multiple times, the crawler needs a system for assigning new URLs during the crawl, because the same URL can be found by different crawling processes. Two types of policies where studied by [68] static and dynamic assignment. Dynamic assignment uses on central process to assign new URLs to the different crawler processes dynamically. The server can also balance the workload of the crawlers. Static assignment assigns each crawler a range of URLs (usually a range of hash values). Whenever a crawler finds a new URL, it has to assign the new URL to the corresponding crawler, making inter-process communication which is an important issue in the crawlers design.

The standard high level architecture of a web crawler is presented in figure 2.1.

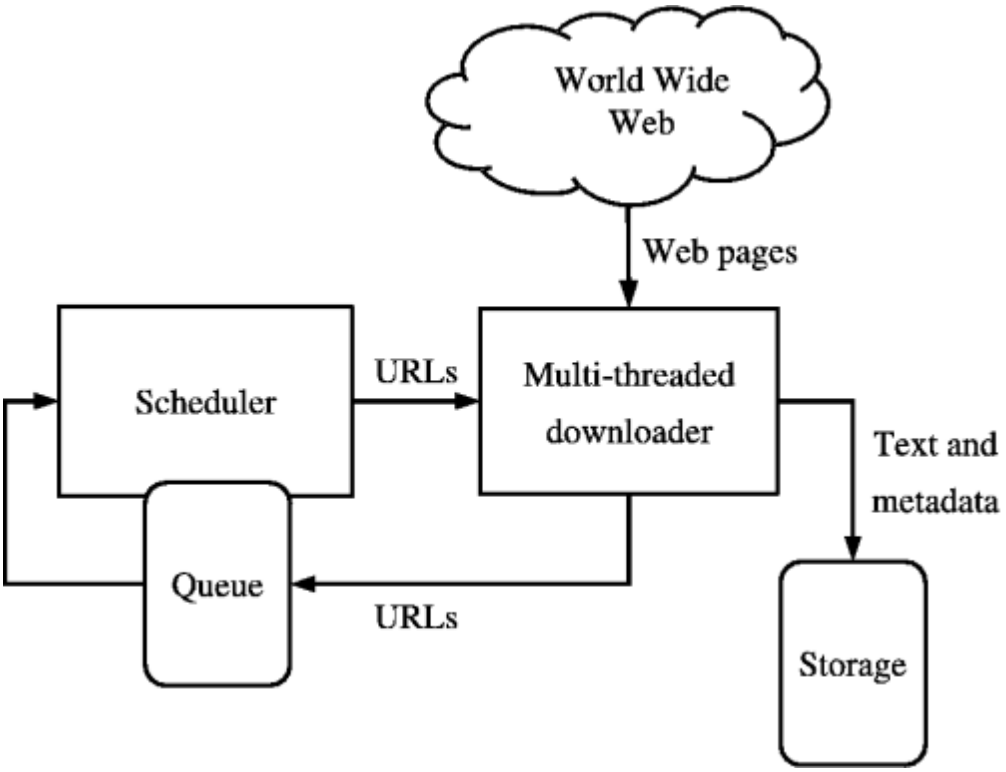


Figure 2- 1: Components of a web-crawler adapted from [62, 64, 69]

The structure of a basic crawler shown in figure 2.1 plays a major role in this component. It starts its work from the web by downloading multiple URLs simultaneously as it is muti-threaded downloader and also as a programs that browses the Web on the search engine's behalf, similarly to how a human user would follow links to reach different pages. The program is given a starting URLs, called seed URLs. It extracts URLs appearing in the retrieved pages, and based on the extracted URLs it determines what links to visit next. The crawlers also pass the retrieved pages into a *page repository* which is used by another component. Crawlers continue visiting the Web until local resources, such as storage, are exhausted or some conditions are met.

Common web crawler implements method composed from following steps:

The basic working of a web crawler can be discussed as follows [62, 69]:

1. Select a starting seed URL or URLs
2. Add it to the frontier
3. Pick the URL from the frontier
4. Fetch the web-page corresponding to that URL

5. Parse that web-page to find new URL links
6. Add all the newly found URLs into the frontier
7. Go to step 2 and repeat while the frontier is not empty

Thus a crawler will recursively keep on adding newer URLs to the database repository of the search engine [62]. Hence, we can see that the main function of a crawler is to add new links into the frontier and to select a new URL from the frontier for further processing after each recursive step.

2.7.2 Indexing

Finding important relevant pages on the web for indexing is a priority for search engines. Indexing is the process of selecting keywords for representing a document. It can be done manually or automatically. In manual indexing the process of selecting the keywords is done by trained indexers, who are knowledgeable about the subject matter of the document, through scanning of the entire text or selected portions of the text, like titles and abstracts. When indexing is carried out by using computers, it is known as automatic indexing. Automatic indexing assumes the storage of information items in a database in a computer [17].

Indexing process creates a massive index of all the words of the pages it crawled, which results in a database consisting of millions of web pages. This index facilitates searching with a reduced cost. Indexing parses the web pages and extracts terms from them by removal of stop words, stemming and builds an inverted index containing a word and the list of documents in which it occurs [70].

The process of automatic indexing has three parts [71]: removal of high frequency words, suffix stripping, and detecting equivalent stems. Removal of high frequency words involves the removal of frequently occurring words (stop words) which have least significance in representing a document. Suffix stripping, which commonly done by stemming algorithm, is a conflation procedure that reduces different variants of the same word to single common term by removing suffixes. Detecting equivalent stems, deals with selection of index terms in which a class name is assigned to a document if and only if one of its members occurs as a significant word in the text of the document and then becomes a list of class names. These are often referred to as the documents index terms or keywords

2.6.2.1 Spoken Content Indexing

After the spoken documents are converted in some suitable format the next step is to index the terms. The well-accepted and the most used format for storing and indexing the terms is Inverted File Structure (IFS) [72]. The basic format of this structure is as follows:

$$\langle t, (d_1; p_1, p_2, \dots), (d_2; p_1, p_2, \dots), \ \dots , \ (d_i; p_1, p_2, \dots) \rangle \quad (9)$$

Where t = term (represented as discussed in the previous section)

d_i = document i

p_j = position j in that document

Equation 11 shows that, if there are too many positions where the term is occurring, then it will increase the space overhead.

In that case, instead of storing positions, we can divide the document in blocks of some terms and use the block number to represent the position of a term. While searching, we can reach to the block directly, but we will have to do linear search in that block. Thus, this block-level approach is a trade-off between space and time.

Improvements in speech recognition technology and computing power have enabled the development of usable indexes for vast spoken audio repositories. A standard technique is to use speech recognition to transcribe the audio and then to build an index using this transcription.

However, this approach suffers from the fact that a speech recognizer has a limited vocabulary so the system cannot retrieve out of vocabulary (OOV) queries. A popular technique to confront this problem is to use phoneme recognition system is used to transcribe the spoken audio. Word queries are then converted to phoneme sequences and searched for in the transcription [73].

Weighted automata indexation is an efficient method for the retrieval of uncertain inputs [74]. Alternative ASR hypotheses together with their probabilities, are represented as weighted automata. These automata are processed to extract all of the possible substrings that are contained in the automata.

In this process the automata are turned into transducers where the inputs are the original labels of the automata and the outputs are the utterance numbers. Next, these transducers are combined by taking their union. The final transducer is optimized using weighted transducer determination,

resulting in optimal search complexity, linear in the length of the input string. The weights in the index transducer correspond to expected counts.

The process of indexing involves generation of an index, and can be defined as follows: Spoken content indexing is the task of generating representations of spoken content for use in a retrieval system. These representations include indexing features i.e., terms consisting of words and phrases derived from the spoken content, weights for the indexing [75].

The exact form of the index depends on the domain and the application. For example, for some applications, speech media is pre-segmented into documents and only entire items are returned to the user in the results list. In this case, a time code information returned by the speech recognizer may be discarded, and the index may contain only the information regarding which indexing term is associated with which document. In other cases, the system might return a time point within an item as a result. In this case, time code information must be retained in the index. The structure of the index is optimized for efficient calculation of matches between the query and the speech media item [76].

The information generated by an audio processing modules can be stored in an XML[1] resource descriptor file that takes into account the kind of data to be indexed and retrieved and the various modules operating on them. It is based on the concept of segment and provides generic but powerful mechanisms to: characterize segments, and group segments into sections. Each segment is characterized by a set of features and consists of a sequence of words, multi-words and acoustic events, in any order. Words and multiword may also include phonetic, lexical and morpho-syntactic information. Additionally, it includes metadata describing where the audio resources were taken from, how they were processed and key features that allow to play their contents.

2.7.3 Query engine Component

This component examines through the results and ranks them according to their relevancy. It has two interfaces, namely search interface and display interface. Since it directly interacts with the user, it accepts the user query, analyses it and consults the indexer to display the results to the user by some ranking. Some basic principles are followed by all search engines to determine the

[1] http://www.w3.org/XML/Schema.

relevancy of results. They are [65]: the location of key words in a web page is a factor for determining the relevancy. The pages containing the search term in its HTML (Hyper Text Markup Language) tag, at the beginning of the page, in the links or subheadings and Meta tags are more relevant. The second one is frequency of key words in the page is another factor for determining the relevancy. The page with more occurrences of a search term is said to be more relevant. Each search engine has its own method for assigning weights. Because of this, for the same query, different search engines provide differently ordered results.

2.8 Spoken Document Retrieval

2.8.1 English

Spoken document retrieval was first introduced in the Sixth Text REtrieval Conference (TREC-6) at 1997. All experiments were performed in English. The transcription generated from the recognizer can be phone-based, syllable-based or word-based. The transcription is then indexed and searched using text retrieval techniques which are widely adopted in the TREC community. Data used by this community are mostly newswire services, video sources and radio sources, and such data is provided by the Linguistic Data Consortium (LDC).

2.8.2 Amharic

Amharic is one of the Ethio-Semitic languages, which belongs to the Semitic branch of the Afroasiatic family [77] and is related to Hebrew, Arabic, and Syrian. Amharic, which is spoken mainly in Ethiopia, is the second most spoken Semitic language, after Arabic. It is spoken by several million people as a first language and as a second language throughout different regions of Ethiopia. Currently, Amharic is the official working language of the federal democratic republic of Ethiopia.

Amharic has mainly five dialectical variations (Addis Ababa, north shoa, Wollo, Gojjam, and Gondar) spoken in different regions of the country. The speech of Addis Ababa has emerged as the standard dialect and has wide spread across all Amharic-speaking communities.

The main characteristic of Amharic is the complex morphology where many new words can be derived from a single stem by addition of several prefixes, infixes and suffixes. Although there is no a one to one correspondence between Amharic morphemes and English words, we can say that

one Amharic word may correspond to a group of English words. This complexity nature causes the vocabulary to expand significantly which is problematic for speech recognition.

Another characteristic of Amharic is the free words order where the order of constituents can be changed without affecting the grammaticality of a sentence. In terms of constituent orders, Amharic can be considered as a Subject-Object-Verb (SOV) type language, however, other constituent orders are also common. Free word order causes data sparseness which leads to non-robust language Model (LM) estimates.

As with all of the other languages, Amharic has its own characterizing phonetic, phonological and morphological properties. It has a set of speech sounds that is not found in other languages. For example the following sounds are not found in English: ጸ, ፀ, ኸ, ቐ, ጐ, etc. Amharic speech contains 38 different phones with 31 consonants and 7 vocal sounds (vowels) and at least 234 distinct CV (consonant-vowel) syllables. The consonants are generally classified as stops, fricatives, nasals, liquids, and semi-vowels [78].

Amharic spoken document retrieval is an important problem because Amharic is one of the key languages used by majority of population of Ethiopia, and much information exist in the form of Amharic spoken audio. The linguistic properties of Amharic speech are very different from that of English. This affects the techniques used in Amharic spoken document retrieval. Amharic suffers from data sparseness and out of vocabulary words problems as it is a morphologically rich language. Therefore, the development of subword based language models for Amharic is recommended [38].

2.9 Comparison of Audio with Text Information Retrieval

There are some similarities between term identification for text and spoken document retrieval.

For example, in the text case there may be false alarms on searches with multiple senses. If one of these terms is present in a query it will match any occurrence in a document regardless of the sense used in each case. This has been shown to have a minimal effect on retrieval effectiveness except for very short queries. There may also be misses on search terms which occur in the documents as synonyms of a query term; although there are techniques designed to overcome this problem. Finally, there may be query-document term matching errors, arising from spelling errors in the documents or query, or inappropriate term stemming.

These problems are also present in spoken document retrieval. However, there are two additional sources of potential errors similar in effect to those just described. Acoustic false alarms, which occur when the speech recognizer hypotheses the presence of a term which actually doesn't exist, and acoustic misses, where the occurrence of a term is not detected by the recognizer. Adding more search terms to the query may balance all these sources of search error [79].

2.10 Language Identification

This section is concerned with different language identification methods. Language identification is the process of identifying the language of a speech or text of a document. Language differences are only part of the observed differences arising from speakers, uttered messages and environmental conditions. In the following, we briefly describe the major language identification tasks.

The task of language identification serves in various areas of Natural Language Processing. Language identification program provides applications like: E-mail routing and filtering engines, Text mining applications, Identification of the language or encoding of WWW pages, Information retrieval systems, Content based and language specific web crawlers and search engines . With the increasing number of different languages in the Internet, the importance of language identification also grows. Language identification is also needed for morphological processing of document. The language of a document needs to be known before techniques such as stemming or parsing can be applied. Also if a lexicon is used for spell checking, the program needs to know which language specific rules to apply. While there are many methods for performing language identification discussed as follows, an N-gram approach, as proposed by [80] has proven effective.

Common Words and Unique Letter Combinations

The main idea of the common words approach is to store highly frequent words for each language in a database. The document to be classified is then compared to all the word lists in question. By the use of a scoring system the word list with the most occurrences in that document indicates the language of the document. This seems to be a good approach for large documents.

Several drawbacks have to be mentioned however. Especially for short texts, i.e. input of less than ten words, the probability is quite low that one of the common words occurs in the input. This leads to misclassification of the language. Also for languages as Amharic, where morphologically

complex and tokenization is not easily possible, this approach won't work in the desired way. An advantage of the common words approach is that it is easy to implement and that it requires less computational power than a tri-gram approach as there are fewer calculations to be done [81].

Statistical Approach

Statistical Approach uses Markov Models to calculate the probability that a document originated from a given language model. For statistical language identification a set of character level language models is prepared from training data, i.e. sample texts in different languages as a first step. This is done by segmenting the strings and entering them into a transition matrix which contains the probabilities of the occurrence of all character sequences [81].

Compression based approach with prediction by partial matching

Another method is the Compression Based Approach with prediction by partial matching. In this method sample text from various languages are compressed using the PPM (prediction by partial matching) algorithm. A document to be identified is then also compressed and the number of bits needed to encode this document are compared to the bits in the existing language models.

The main idea of PPM is to predict an upcoming character using the set of previous characters with a fixed context. Each upcoming character is assigned a probability. The amount depends on whether the character has appeared before in a sequence with the previous characters. If a certain sequence has not been observed so far, it uses the escape probability and the context size is reduced by one. The escape probability for a sequence receives the number of times the sequence has been observed being followed by a character.

In the lower order model again, a check is conducted to see, if a sequence with that character at the end exists. If the character has never been seen in a sequence the context size is set to -1. At this level all characters are predicted equally. This means the character receives a probability, which is calculated by 1 divided by the total numbers of characters - the already occurred characters [81].

The N-gram Approach

One of the most successful approaches is the N-gram approach. It is more concerned with text categorization, but they used also very well on the task of language identification. The main idea of using n-grams for language identification is that every language uses certain n-grams more frequently than others, hence providing a clue about the language [81].

N-grams can be seen as substrings of words and respectively words depending on the size of N. For language identification one calculates the N-gram profile of a document to be identified and compares it to language specific N-gram profiles. The language profile which has the smallest distance to our sample text N-gram profile then indicates the language. For N-grams at word initial and final positions, they add an underscore marker which forms part of the actual N-gram. In the example below all possible N-grams with N=2, 3 and 4 for the word **"አልወሰዱም"** are listed. Uni-grams, i.e. N = 1 are just the letters of the word themselves like አ, ል, ወ, ሰ, ዱ, and ም.

bi-grams: **_አ, አል, ልወ, ወሰ, ሰዱ , ዱም , ም_**

tri-grams: **_አል, አልወ, ልወሰ, ወሰዱ, ሰዱም, ዱም_**

quad-grams: **_አልወ, አልወሰ, ልወሰዱ, ወሰዱም, ሰዱም_**

The algorithm works in the following way. In the first step the sample texts in several languages read in only by one and all punctuation marks are deleted. Each word becomes a token delimited by white space before and after. All tokens are scanned and N-grams with N=1-5 are produced from these tokens. The N-grams are stored in a hash and for each occurrence the counter for the N-gram in question is increased. After that the hash is ordered starting with the most frequent N-grams. Usually uni-grams are at the top of the list, but are discarded as they just reveal information about alphabetical distribution of a language. They do not account too much for the information we are interested in. This procedure is repeated for each language. The N-gram hashes constitute our N-gram profiles for each language. In order to identify the language of a new document, repeat the above mentioned procedure for that document, yielding an N-gram profile for that document. Now they compare the document profile to the existing language profiles by calculating the distance between the N-gram profiles. The distance is calculated in the following way. For each N-gram in our test document, there can be a corresponding one in the current language profile we are comparing it to. N-grams having the same rank in both profiles receive a zero distance. If the respective ranks for an N-gram vary, they are assigned the number of ranks between the two with

a maximum distance of 3. Finally all individual N-gram rank distances are added up. This number is now the distance between the sample document and the current language profile. This step is repeated until the sample document has been compared to all language profiles in question. The smallest distance then indicates the language [81].

Chapter Three: Related Work

3.1 Introduction

A lot of work has been done on the development of audio information retrieval and multimedia in general for many languages of the world. In this chapter, we review the implementation of search engines by focusing on related research works on topics such as language-specific audio information retrieval.

3.2 Chinese Spoken Document Retrieval

3.2.1 The Use of Subword-based Audio Indexing in Chinese Spoken Document Retrieval

This research is based on local Hong Kong television news broadcasts in Cantonese, the predominant Chinese dialect used in Hong Kong, Macau, South China and many overseas Chinese communities [82]. Cantonese is monosyllabic with a rich tonal structure which are approximately 1,600 distinct tonal syllables with six lexical tones. The syllable unit seems very desirable for Chinese spoken document retrieval, by virtue of the monosyllabic nature of the language and its dialectal variations. Hence, syllables can fully characterize the language and provide high phonological coverage for spoken documents.

According to LI Yuk Chi [82], there are several thousand unique characters in the Chinese language that are used for the Chinese writing system. Unlike Amharic, the definition of a Chinese word is vague, as there is no explicit word delimiter in Chinese. Hence given a Chinese sentence (i.e. sequence of characters), they need to perform the procedure of word tokenization in order to segment the character sequence into a word sequence. Word tokenization involves referencing a Chinese word lexicon in order to segment into words.

The inherent ambiguity in Chinese word tokenization creates a problem for Chinese information retrieval. If a given word appears in both the query and the document but it is not segmented as such, there will also be a failure in matching the word during retrieval. This suggests that word-based indexing for Chinese retrieval may be problematic because the "correct" word is difficult to obtain. As a result, some have adopted the technique of subword-based indexing, i.e. indexing based on Chinese characters instead of words. Indexing solely based on single characters lose the

sequential information which captures lexical information. Overlapping characters to retain some sequential constraints for retrieval where, Sequential constraints offered by overlapping character bigrams achieve comparable retrieval performance as Chinese word-based retrieval.

Unlike Amharic, a Chinese word does not have much inflectional variations, hence stemming is generally unnecessary. Stop word removal for Chinese is possible. However, due to the ambiguity word tokenization, a Chinese character which constitutes a stop word in one context may be part of a content word in another context. As a result, stop word removal may not be applied in Chinese information retrieval.

The researcher uses overlapping Chinese syllable n-grams for indexing Chinese spoken documents. Indexing Chinese audio in terms of syllables imply the use of syllable recognition technology. If the audio document representation is in terms of syllables, the textual queries need to be transformed into syllables as well, and the retrieval mechanism should also involve matching based on syllables. Transformation of textual queries into syllables may be accomplished by pronunciation dictionary lookup. It should be noted that syllable errors from speech recognition are present in the audio indices, but are absent from the query transformation. This creates a mismatch in retrieval, i.e. the queries are "clean" but the documents are "erroneous".

The Chinese spoken document retrieval involves the use of a query (spoken query or textual query) to retrieve relevant documents from a spoken document collection like audio tracks from video clips and audio clips from radio broadcast. The author interested in the use of Chinese textual queries to retrieve Chinese spoken documents; where a list of documents are retrieved, and they are ranked according to their degree of relevance. The author maps the text into syllables by using pronunciation dictionary lookup for matching during retrieval. The author proposes to perform Chinese spoken document retrieval based on subword units, namely, syllable n-grams and to enhance retrieval engine by robust techniques, the author investigate two techniques: query expansion and document expansion. The author also used a vector space model for retrieval.

Chi evaluates the retrieval performance based on the ranked retrieval list output from the retrieval engine and used the Average Inverse Rank and the Mean Average Precision as evaluation criteria.

According to [82], the use of subword (syllable-based) indexing for Chinese spoken document retrieval includes:

- The incorporation of sequential constraints in syllable-based audio indexing by means of overlapping syllable n-grams, and the contribution of such constraints towards retrieval performance.
- The use of tone information that compare retrieval results between indexing with base syllables (tone excluded) and indexing with tonal syllables (tone included).

The Cantonese spoken documents are derived from a video archive of news broadcasts. Each video clip is accompanied by a short textual summary which is very brief in contents, and it is not a transcription of the audio track. The text are in Big5 encoding, which is the major encoding method used in Hong Kong. The textual summaries average 150 Chinese characters in length, and vary between 140 to 1,700 characters. The video clips average 1.5 minutes in duration, and vary between 11 seconds to 25 minutes.

The researcher transformed textual query in Big5 format into Cantonese syllables by looking up a pronunciation dictionary; indexed the audio tracks with a Cantonese syllable recognizer by using monosyllables, overlapping syllable bigrams, overlapping syllable trigrams and skipped syllable bigrams; and then estimated the Chinese word-based text retrieval benchmark to be Average Inverse Rank (AIR)=0.834. The use of overlapping syllable bigrams (tonal syllables) deliver a comparative performance with AIR=0.830. Results based on speech recognition outputs using overlapping syllable bigrams without tone information gave AIR=0.479.

The strength of this research work was use of different alternatives like monosyllables, overlapping syllable bigrams, overlapping syllable trigrams and skipped syllable bigrams to index the audio information of Chinese language, but this cannot be effective due to the inefficient method of automatic speech recognizer; it is better to use feature extraction methods like MFCC, etc.

3.2.2 Search by Voice in Mandarin Chinese

The authors in [83] build a Mandarin Chinese voice search system. They describe strategies for data collection, language, lexicon and acoustic modeling, as well as issues related to text normalization that are an integral part of building voice search systems and show their performance on typical spoken search queries under a variety of accents and acoustic conditions.

The authors selected more than 100,000 queries from anonymized google.cn logs after some filtering to remove offensive terms. More than 1200 speakers with different cultural and educational backgrounds where selected for the data collection and asked them to read the queries under varied acoustical conditions such in the office, in the street, in restaurants, etc. They also made an effort to select speakers from a variety of language backgrounds, whose native dialects include several varieties of Mandarin, Wu, Xiang, Gan, Kejia and Cantonese. More than 250,000 utterances were ultimately collected from these speakers [83].

To build the language model for Mandarin the authors selected simplified Chinese queries from domain google.cn, user segmentations were rejected and the data reformatted by running segmented, queries where further processed removing some unwanted terms, such as, very long queries were removed, punctuation was removed, non-Chinese and non-English, mostly words that are sufficiently common in daily Chinese usage, queries were further removed, URLs were normalized, i.e. mapped from their written form to their spoken form, offensive terms.

The acoustic models used are standard 3-state context dependent (tri-phone) with a variable number of Gaussians per state. These are trained on a 39-dimensional vector composed of cepstral coefficients and their first and second order derivatives. The frontend also uses Linear Discriminant Analysis (LDA).

Finally, different accuracy metrics used to analyze the quality of the system. They measure character error rate (CER), with different training sets and get 35.3% in 4-gram language model and 25.9% in 5-gram language model. On the web metrics such as web score at one, they compare the top web search results produced by the recognition hypothesis with the top web result produced by the reference transcript and with different training sets, get 63.3% in 4-gram language model and 59.4% in 5-gram language model. They pointed that adding more acoustic data further improves the performance of their system. The conclusions were not so reliable because of the limited amount of data.

3.2 Speech Retrieval for Turkish Broadcast News

Turkish is the most widely spoken of the Turkic languages, with a distinctive characteristics vowel harmony, word order of subject-object-verb, and extensive agglutination. Due to Turkish agglutinativity, text IR methods are inadequate to index ASR output, especially when the ASR

system has low accuracy. However, Spoken information retrieval has been investigated for several languages, but did not considered the agglutinative languages like Turkish [37]. Siddika Parlak [37] has designed speech retrieval for Turkish broadcast news that enables searching documents that are spoken in the language.

Among the things that were taken into consideration during the design based on Turkish broadcast news corpus were:

- Develop two types of systems for the retrieval of Turkish Broadcast News: a Spoken Term Detection system and a Spoken Document Retrieval system. They both combine ASR and IR components to retrieve spoken data by using WFSA indexation to retrieve STD and VSM for SDR index.
- To alleviate the effect of ASR errors on retrieval, the authors used grammatical and statistical subword units as well as lattices introduced to STD.
- NIST-based tools are used for evaluation.

As the authors mentioned, the best scores of STD was obtained with the word-morph hybrid, even better with term-specific thresholding in detection. They construct a baseline SDR collection. The test set is segmented into news stories, relevance judgments were made by human assessors. Use of confusion networks, improved the system performance for only a few cases.

Comparing the two tasks, it can be concluded that, the methods to improve the retrieval performance resulted in different effects on STD and SDR. Since STD is based on term matching, its performance is directly affected by OOV words and WER variation. For this reason, subword based recognition and lattice based indexing were useful for STD, but for SDR. On the other hand all of the stemming approaches provided improvements in SDR scores.

The first weakness we observed is that they used human agents to assess news stories, as discussed in section 1.1 the humans may vary greatly in their descriptions of material or may be inconsistent, introducing errors. The second one is the query result is text instead of audio which degrades the validity of the information as ASR is not perfect and text information is less expressive than audio.

3.3 Arabic Audio News Retrieval System

Arabic is a language with the largest number of speakers in the Semitic family. It is spoken by more than 422 million people as a first language and a second language making it one of the six most spoken language in the world[2]. 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. Here, Arabic Audio News Retrieval System Using Dependent Speaker Mode, Mel Frequency Cepstral Coefficient and Dynamic Time Warping Techniques (ARANEWS) was developed by Hassen et al., 2014. The main aim of this study was to build an Arabic audio news search engine that could satisfy the search requirements of the Arabic users. ARANEWS is content based audio retrieval system; it creates an abstraction of audio Arabic queries and Arabic audio keywords in term of features. The extracted features are examined to determine the perceptually similar audio content.

The proposed architecture is based on two bases [41]: the first one is the interaction between retrieval system and human which would be by using human voice directly or file contains recorded human voice. The second base is the retrieved ranked list which would be audio files not text files. Therefore, the researchers proposed an architecture which consists of three major components: input module, query module and retrieval module. The Input module stores the extracted features from Arabic audio news within designed feature database. Query module extracts features from formulated and refined Arabic queries that are submitted from users. Retrieval modules measure the similarity between the extracted features from queries and extracted features from Arabic news. The proposed system supports both Query by Humming (QBH) and Query by Example (QBE).

The researchers also proposed a technique for selecting keywords which is based on news text files, where each selected keyword will be recorded many times by the user of system to prepare the training data at the input module. The authors also added the reason why they use text files in selecting keywords process instead of analyzing the audio news and extracting the audio keywords directly is that simply lack of researches that work on analyzing the long Arabic audio clip.

[2] en.m.wikipedia.org/wiki/Arabic

The recording process in both QBH and QBE may be affected by the nature of environment "clean vs. noisy" and the quality of microphone, which in turn affects the accuracy of extracted MFCC features, the accuracy of extracted MFCC will affect by its role on the calculated distances and the retrieved audio news. Hence, the authors used WIENER filter to remove silence gate within ARANEWS system.

As pointed by the authors, the performance of ARANEWS system is based on the accuracy of speech recognition component that has been used. MFCC feature and Dynamic time warping (DTW) are used in speech recognition component and three factors have been chosen to be examined in the experiments. The first factor is the size of dictionary "number of audio keywords" that indexed Arabic Audio news. The second factor is the noise and the third one is the number of templates "how many times each keyword will be recorded". The length of query in all experiments is two seconds.

According to the researchers, two thousands Arabic news have been collected to evaluate the performance of ARANEWS system. Each one hundred Arabic news is classified under one category and each category is indexed by one Arabic audio keyword. Each news is collected as audio news and as text news. Sound Forge tool has been used to extract audio tracks from some video files that contain the Arabic audio news.

Therefore, the results show that the values of recall, precision and F-Measure increases when the number of templates increases. The number of templates affect the retrieval time. When the number of template increases, the retrieval time increases. Content based audio retrieval systems that use template based approach must achieve the tradeoff between the number of templates and the retrieval time.

In the literature that discusses ARANEWS, there is no explanation about how the language is identified, how the search engine ranks its results.

3.4 Search Engine for Amharic language

Amharic language is one of the Semitic languages. It is the official language of the federal government of Ethiopia and some other regional states and widely spoken by millions of people as a first language and as a second language. Amharic language is a morphologically rich language. Due to most search engines are designed for English language, and do not consider the specific

natures of Amharic language. Tessema Mindaye [2] has developed Amharic search engine. In this work, the author studied the morphological variations of the Amharic language in detail to make the search engine efficient.

As other search engines, Amharic search engine consists of the three components such as: crawler, indexer and query engine that adapt the nature of the Amharic language. The components of the search engine are all language specific and consider only documents that contain Amharic language texts. The crawler has a language identifier or categorizer. For each component, the author developed algorithms and implemented them. For implementing the proposed algorithms, the author used lucene as a tool, Java programming language and other assisting tools such as, wordextract, NekoHTML and PDFBOX for text extraction. The author developed a language specific crawler from scratch.

The author used am.wikitionary, www.waltainfo.com, www2.dw-world.de/Amharic and www.ethiopiawakeu.net as seed URLs to begin the initial crawling. In the indexing component, words are tokenized using a whitespace, normalized, stop words are removed and stemming is taken place before indexing. For tokenizing the Amharic words, Amharic punctuation marks and white space are used. In addition to this, hyphenated words are tokenized as they are. Short forms of compound and single words were treated using "/" or ".".

The author evaluated the search engine using precision-recall method and some selected queries to guarantee the requirements are met. The crawler is also tested using two runs on different days and it has processed 7500 pages. It has selected 1803 Amharic pages in the first run. The researcher has also helped the crawler in deleting non-Amharic URLs from the frontier manually. In the second run, the crawling was done on a single host and it has processed 22,276 pages and 12,276 pages were with Amharic content. The rest pages were link pages.

The categorizer was tested against Amharic textual news documents to check whether it can identify Amharic Unicode encoded pages from the document collection and it was achieved 99.41%. Its accuracy against identifying non-Unicode encoded pages was perfect, 100%. In this case, the author did not test the categorizer against other languages' pages to assure whether or not it can identify as Amharic pages.

The results of precision and recall using the "OR" and "AND" operators were:

- Using “OR” precision 65%, recall 95%.
- Using “AND” precision 99%, recall 52%

As a modification of the Amharic search engine, Hassen Redwan [3] has increased the efficiency of the crawler by employing multi-threading. The author also converted non-Unicode fonts into Unicode encoding (UTF-8) font, upgraded the normalizer to handle short form words separated by more than one slash or dot and incorporated the different Amharic aliases.

As non-English languages have been rising exponentially on the web with the increase of multilingual World Wide Web, the number of online non-English speakers who realizes the importance of finding information in different languages is enormously growing. Mequannint Munye [5] designed a model for an Amharic-English Search Engine and developed a bilingual web search engine based on the model that enables web users for finding the information they need in Amharic and English languages. In doing so, they identified different language dependent query preprocessing components for query translation and also developed a bidirectional dictionary-based translation system which incorporates a transliteration component to handle proper names which are often missing in bilingual lexicons. They used an Amharic search engine and an open source English search engine (Nutch) as their underlying search engines for web document crawling, indexing, searching, ranking and retrieving. To evaluate the effectiveness of their bilingual search engine, precision measures were conducted on the top 10 retrieved web documents. The experimental results showed that the cross-lingual retrieval engine performed 74.12% of its corresponding English monolingual retrieval engine and the cross-lingual retrieval engine performed 78.82% of its corresponding Amharic monolingual retrieval engine.

The overall evaluation results of the system are found to be promising. Even though three of them at [2, 3, 5] tried to solve the general search engine problem by designing text based Amharic search engine, they did not consider the audio information retrieval of Amharic web documents; as audio is a richer and more expressive medium than text; it contains more information than just the words. With speech, information such as the identity of the spoken language, the identity of the speaker, and the “mood" or “tone" of the speaker, expressed as prosodic cues, are captured in addition to the spoken words.

3.5 Audio Data Model for Multi-criteria Query Formulation and Retrieval

The rapid increase in multimedia requirements new demands in audio data management, search and retrieval. As a result, a number of research works have been conducted in the area, such as audio crawling, feature extraction, indexing, classification and segmentation. However, these techniques are not adequate and handling audio data content is still far from sufficient for most like search, retrieval tasks. In this subtopic, we have reviewed modeling audio data for multi-criteria query formulation, a work of Tizeta Zewide [10].

Tizeta Zewide [10] addressed different approaches that automatically analyzes audio content, like speech classification, music classification, retrieval of similar sounds, music genre classification. They pointed out two approaches that have been commonly employed to retrieve audio data. The first is to use Query-By- Example retrieval, where the query is a piece of audio and retrieval is made by using a similarity measure. The other approach is to generate textual indices and then use text-based information retrieval.

The authors pointed out the Query-By- Example approach is that similarity is computed on features that are automatically derived from the audio signal and can therefore be applied inexpensively on a large scale. But these systems make no presence of attaching semantics to the queries as they are not oriented towards audio semantics and audio information retrieval based on high-level features is certainly a reasonable answer to the semantic drawbacks of information retrieval based on low-level features. So, modeling the high-level meaning of audio requires semantic content. A very popular means of semantic audio retrieval is to annotate the audio with text, and use text-based database to perform the retrieval. But, this approach has major problems when encountered with large volumes of audio data due to its time consuming process and its subjectivity nature due to human intervention. The other most commonly used approach is to apply Speech Recognition techniques to extract spoken words from a sound or to use automatic text transcription of audio. Manual annotation is still widely applicable because Speech Recognition systems are mostly error prone due to problems such as Out Of Vocabulary words, challenges related to independent speakers and continuous speech, lack of language models, acoustic model, poor audio quality of Amharic speech etc. and Speech Recognition systems' application is restricted to speech.

The author proposes an audio data and audio repository model to fulfill user requirements in retrieving audio data from large collections. The audio data repository model they proposed enable to capture the audio itself, its low-level representation, all alphanumeric data associated to the audio and timing information. Thus, it enables both keyword-based and similarity-based operations on audio. In the proposed model, a generic audio repository model that can handle a general audio and a sub-repository model that can manipulate speech through its constituent units is discussed. The speech repository model enable them to capture all appropriate information associated to speech units, to keep track of hierarchical relationships between speech units and the speech that contains them. They used Object Relational scheme to manage these audio related data under a Database Management System.

The author develops audio retrieval by enabling users to apply high level features linked to audio signals in their query in order to match relevant audio in a database. Such an approach improves simple matching based on low level features, as the user need not have example sounds for querying. Moreover, an example audio can be used as a query since users may not always know exact search terms to achieve useful results.

The authors conduct an experiment by selecting an audio data from the selected application of medical domain of a heart sound and speech recordings, a custom database has been built from Internet searches and live recordings. They used a total of 78 sounds from these 48 were heart sounds and 30 were speeches. As these sounds vary in file lengths, compression type, sampling rates, and background noise levels, they used a common representation and formats to convert them into a fixed target format with predefined settings and eliminated background noises with the help of audacity which is off-the-shelf audio editing tool. Then, feature extraction was done using the MPEG-7 Low-Level Descriptors.

They used similarity matching using Euclidean Distance Metric for a query by example between an input audio and stored audio data on the basis of their respective features. The quality of the features was measured in terms of recall and precision pairs of the top ranked matches. In their experiment, one sample at a time was drawn from the database to serve as an example query and the rest were considered as samples to be compared to the query. A database sample was considered correctly retrieved, when the sample falls in the same class as that of the example sound and the correctness was verified using domain expert evaluation.

Their first experiment was made by combining all the mentioned features together and applying similarity matching between each category of the example and sample heart sounds. However, the aggregate features had given a recall and precision rate of 11.9% and 8.35%, which is very low. Since the combined features did not give promising results, further experiments were conducted using individual features. Firstly, similarity matching is made based on the distances computed between AudioSpectrumFlatness values of the query sound and sample sounds were computed for each sound category. AudioSpectrumFlatness yields relatively good results in that an average of 78.8% and 55.2% recall and precision pairs are attained respectively. Then, the same procedure has been applied for the other low-level features.

The authors' second experiment was concerned with keyword-based audio retrieval. Both heart sounds and speech based image descriptions were tested. To identify those keywords, their domain experts were involved for each medical image description considered in the experiment. Then, six physicians who had experience on Internet searching were involved in providing keyword-based queries. Fifteen such queries are used in the system so as to evaluate the applicability of the proposed model. Among the 15 queries applied for the speech based image description, 12 of them were supported by the system and 3 of the queries failed to be retrieved because the queries used keywords that were not in the system. But all the 12 queries have been retrieved with 100% recall and precision rate. They also conducted retrieval of heart sounds by providing the system with metadata information regarding heart sounds such as category, diseases that are believed to cause some type of abnormal heart sounds. All given queries were correctly retrieved with a 100% recall and precision rates.

3.6 Lessons Learned

Theses audio search engines have some similar components. Mainly, crawler, indexer and query engine are incorporated in the architecture of every search engine. Differences are usually observed in some preprocessing and index stages. However, in developing an audio search engine, especially a language specific one, one should take many factors to consider. In such case, there is a need to identify the features of the language that can affect the retrieval of the language's document. Apart from the language's features, the different encodings of the language's electronic documents play a vital role. Different information retrieval that are reviewed in this chapter

revealed that the above two factors i.e., language's features and encoding were given serious thoughts in their endeavor.

The development of language specific speech audio search engine needs a language oriented indexer, crawler, and a specific query engine for its operation. Different developers used different techniques to develop these components of the language specific tasks. In indexing, crawling, and query engine process the language's typical features that can affect the retrieval process must be taken in to consideration. In this chapter, we have seen what is done and what is not done regarding web audio information retrieval of Amharic web documents.

Chapter Four: The Proposed Amharic Speech Search Engine

4.1 Introduction

This chapter discusses the general architecture of Amharic Speech Search Engine. This search engine is designed to index and search only Amharic speech documents crawled from the web. Our search engine has four major components:

- The crawling component
- The Audio Processing component
- The indexing component and
- The query engine component

4.2 Overview of Search Engine System

Search engines are the main means of accessing the content on the Web. To satisfy its users, a search engine should answer the user queries accurately and quickly. This is a demanding goal, which needs for a good and fast retrieval model and a large and up-to-date coverage of Web content.

To accomplish these requirements, a search engine uses the components: a crawler, to collect the Web resources; an indexer, to create an index of the text content, and a query processor, to evaluate the user queries.

Crawling and indexing are carried out off-line, whereas query processing is conducted on-line. Given the magnitude of the data on the Web, efficiency and scalability for each of these components are of crucial importance for the success of a search engine [12].

4.3 Architecture of Amharic Speech Search Engine

The first component of the search engine, the crawler, visits a web page, reads it, and then follows links to other pages within the site. Everything the crawler finds is input to the second part of the search engine, which is the speech content processing. From the crawled content, the Amharic speech identifier, TIKA, identifies the Amharic speech, then it will be put to the Amharic speech transcriber and the Amharic speech is transcribed and store the transcribed file which will be indexed by the Amharic speech indexer. Query engine (Search engine software) is the fourth part

of the search engine. This is the program that examines the millions of pages recorded in the index to find matches to a search and ranks them in order of relevance.

These components and their sub-components are depicted in Fig. 4.2.

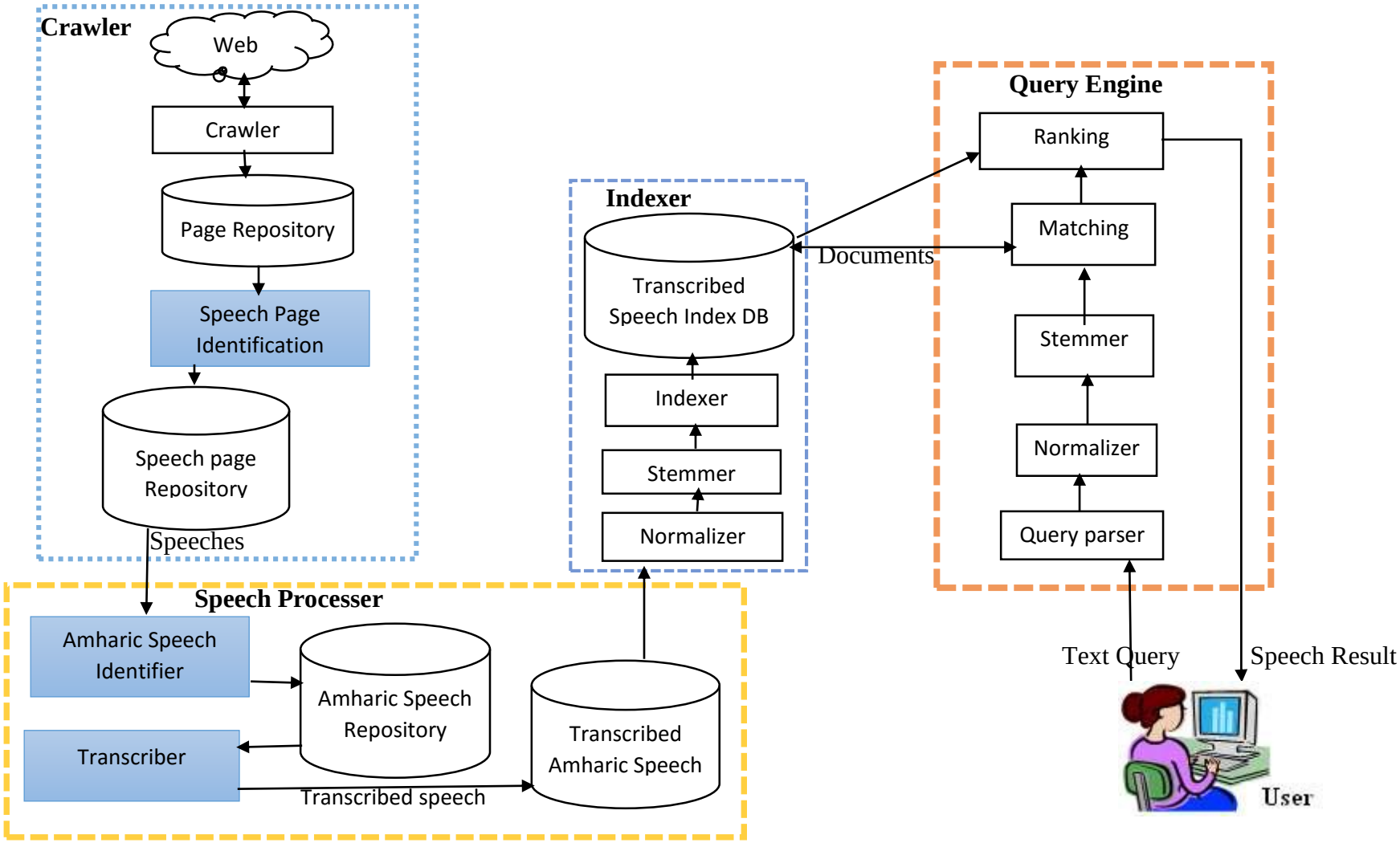


Figure 4- 1: Architecture of Amharic Speech Search Engine

As shown in Figure 4-1, the proposed Architecture of the Amharic speech search engine contains four components: the crawler, speech processor, indexer and query engine. The following sections describe the details of each component of the search engine.

4.4 The Crawler Component

The crawler is a part of search engine that retrieves the web pages from the Internet. Search engines retrieve web pages and download them or their representations to a local repository. The repository might be used for different tasks like the search engine as a source of searching documents for users' queries, mostly after passing through an indexing process. Search engines do not actually scan the web when they receive requests from users. They look up the keywords or phrases in their own database, which are compiled using spiders that continuously travel across the web looking for new and updated entries and creating links to them as shown in Fig. 4.2.

The crawler gathers the pages and stores them and then searches for links on the pages and collects the content from every link in the same way and put in the queue before it repeats this process until it has visited a depth of 10 links. This process is called crawling. The web crawling system has two main phases. First one determines the URLs and the second one retrieves the URLs and downloads the document. The crawler can be generic or focused. A generic web crawler retrieves everything regardless of any topic it finds and a focused crawler traverse only those document which are related to a specific topic.

A crawler should have the capability to manage the huge volumes of data internally. Also it must decide how frequently to visit the pages it already visited for frequent changes. The information retrieved by crawler includes information about pages or some search engine even store the copy of pages referred as cache. The aim of this process is to keep the information up to date. If any changes occur the indexes have to be up to date. To keep the index up to date the search engine schedule the crawling process. Some search engines crawl the web once in 24 hours some do it weekly, it depends on the search engine policy [84]. Generally, there are two spiders that participate in search. One is called discover, and the other is called harvest.

The main procedures are

1. the discover spider traverses through web pages and builds a URL database,

2. the harvest spider checks the URL database first, then begins automatically revisiting each web page the discover spider discovered,
3. the harvest spider extracts part or all of the content from the page, breaks it down into component words, integrates it into a master index of words from other web documents, which is the index database that the engine searches,
4. the search engine matches query keywords with the index in the index database, and the users get query results from the search engine

After downloading a particular page, a categorization task is invoked so that non-speech pages should be discarded and pages with speech content is stored in the speech page repository for further processing. Next to this, those speech pages that contain the speech document put to the Amharic page identifier and it is the task of the Amharic speech identifier component. As shown in Figure 4.1, the Amharic speech identifier takes in the crawled web pages from the repository and puts the pages with Amharic majority web pages into the Amharic speech repository that is to be used by the speech processor.

For the identification task, with certain modifications, an open source tool called Tika is used to classify contents in a given page as to Amharic majority content pages or not. Based on this tool, pages containing 60% or more Amharic speech contents are stored in Amharic speech repository.

4.5 Speech Content Processing

4.5.1 Speech Identification

The crawler downloads web documents of the given seed URLs as archived files. The crawled content includes document other than speech documents like text and image. We adopted apache Tika content extractor in a way to extract web content that filter documents collected other than the given audio document. After the audio feature extracted, Amharic web audio documents should be filtered and make them available for the next processing. The Amharic page identification module discards web documents and links with languages other than Amharic. Contents with a mixture of Amharic and other languages, where Amharic content exceeds 60% will be considered.

```
1. Input: crawled Web directory

2. Output: Identified audio Web documents

3. Add Web documents to queue

4. Read from the queue

5. If file type is doc, docx, xls, xlsx, pdf, gif, jpg,
jpeg, png, bmp, css, js, asf, txt,dll then

   Goto step 4;

 Otherwise

Extract html/xml tag if the file contains tags.

6. Return the file name extension

7. Insert the result: file name into the Repository

8. Repeat until all the documents are identified
```

Algorithm 4- 1: Algorithm for speech page Identifier

4.5.2 Amharic Speech Identification

The language identifier module is used to identify the language of the crawled web speech. In this research work, n-gram model is used to detect the language of the crawled web documents. The n-gram based language identifier is chosen for language identification due to its high accuracy in identification, its flexibility to typographical errors, and its minimal data requirement for Training Corpus. The n-gram language identifier used for this study is Tika. The G2LI method first creates n-bytes sequences of each training text. Then it checks the n-bytes sequences of the collected page with those of the training text. The language having the highest matching rate is considered to be the language of the web page. The general paradigm of language identification can be divided into two stages. First, a set of language models is generated from a training corpus during the training phase. Second, the system constructs a language model from the target document and compares it

to all trained language models, in order to identify the language of the target document during the identification phase. The algorithm of language identification is presented in Algorithm 4-2.

```
1. Input: speech directory

2. Output: Amharic speech

3. Generate a sequence of n-gram from the training corpus and
the target document match the generated n-gram

4. Return the file name, the language detected, the number of
match n-gram

5. Insert the result: file name and the detected language
into the database

6. Repeat until all the documents are identified
```

Algorithm 4- 2: Algorithm for Amharic speech page Identifier

If the speech is Amharic, then it is stored in the Amharic speech repository for further processing.

Amharic Speech Crawling Component

The crawler returns with new web pages, which are temporarily stored as full, complete web pages in the page repository. The new pages remain in the repository until they are sent to the indexing module, where their vital information is stripped to create a compressed version of the page. Popular pages that are repeatedly used to serve queries are stored here longer, perhaps indefinitely.

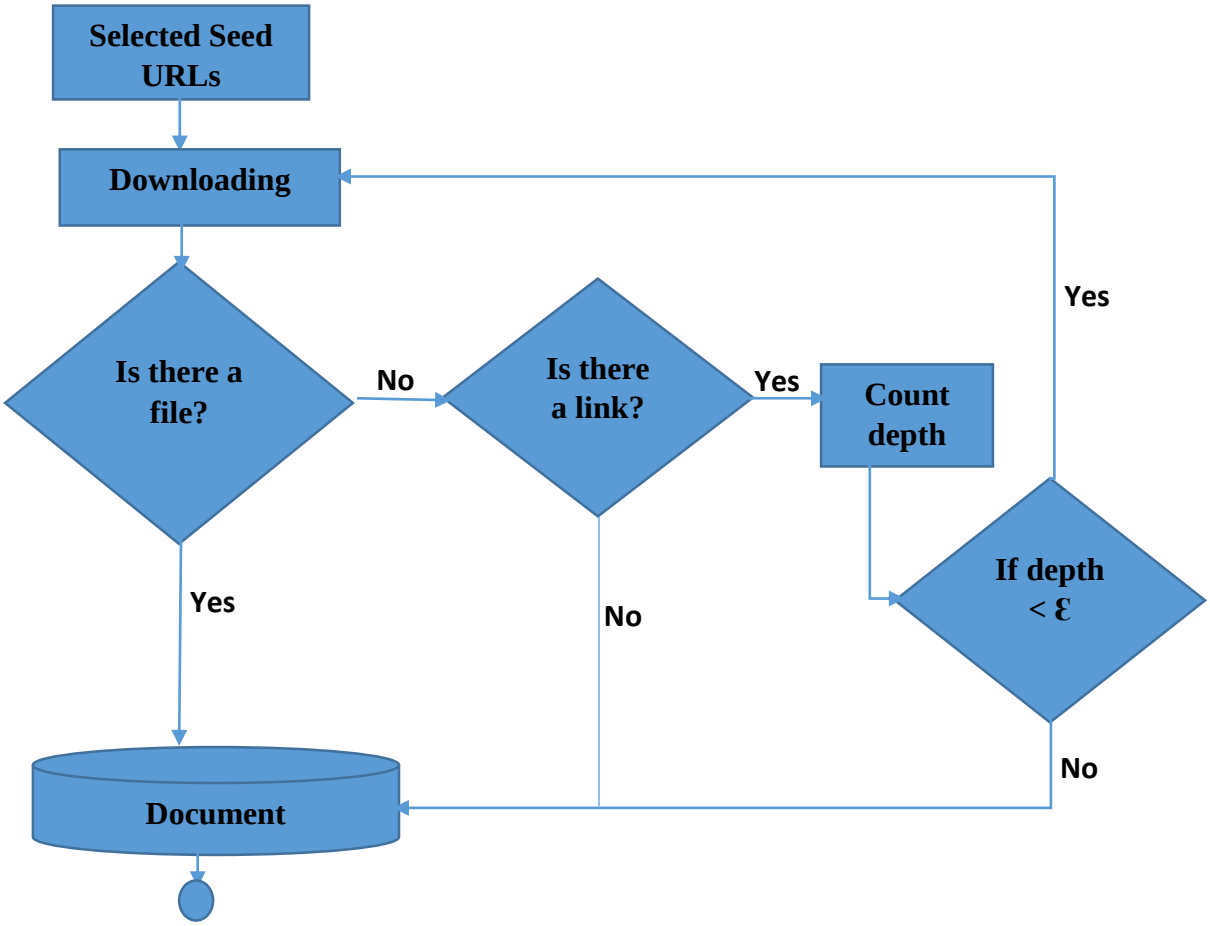


Figure 4- 2: Flow of process in Amharic Speech Crawling Component.

4.5.3 Transcription

Transcription is to use one of the speech recognition technology that exist on today's development frameworks and develop an application to support an automatic transcript generator for a podcast file so search engines can index each podcast content in greater detail related to their existing transcript hosted on the web deaf people and people with voice language understanding problem can take part of the transcript information related to the podcast.

The Auto Transcript Generator takes the podcast file as input, and if necessary, depending on the podcast format, the audio extraction and audio conversion needs to be done before sent over to the speech recognizer routine in pure wav format. In order to achieve this, the Auto Transcript

Generator use MPlayer in order to achieve necessary extraction and conversion and send the processed audio stream as pure wave format file to the speech recognizer module.

The speech recognizer is based on the Sphinx framework. It uses a trained acoustic model, pronunciation dictionary and language model of Solomon Tefera [85] for the input audio file in wave format to provide the recognized words and send them to the transcript smoother module. The recognized words from the speech recognizer is saved to a text file and smoothed up as much as possible by the transcript smoother to deal with removing unnecessary repeated words and finally save the transcript file.

```
1. Input: speech

2. Output: transcribed speech

3. Read the speech file from the speech Repository

4. Identify the file format

      If file type is not wav then

      Go to 3;

5. Recognize the speech

6. Return a transcribed speech

7. Insert the transcribed speech into the transcribed Amharic
speech Repository

8. Repeat until all the speeches are transcribed
```

Algorithm 4- 3: Algorithm for Speech transcriber

4.6 Indexer

Indexing is the process of converting documents in the repository into an efficient cross-reference lookup. For this work, we have used the solr indexer which uses an inverted index. This is because it can list, for a term, the documents that contain it. Solr index contains a sequence of documents, where a document is a sequence of fields and a field is a sequence of terms.

The indexer component takes Amharic speech documents from the transcribed Amharic speech repository component and undergoes some preprocessing steps to convert the document into a form that eases the searching process. It is composed of Amharic analyzer that considers the typical characteristics of the Amharic language and indexing module.

Once the spoken documents are converted in some suitable format, the next step is to index the terms. For this particular application, the well-accepted and the most used format for storing and indexing the terms is Inverted File Structure (IFS). The basic format of this structure is as follows:

$$\langle t, (d_1; p_1, p_2, \dots), (d_2; p_1, p_2, \dots), \dots., (d_i; p_1, p_2, \dots) \rangle$$

Where t = term

d_i = document i

p_j = position j in that document

If there are too many positions where the term is occurring, then it will increase the space overhead. In that case, instead of storing positions, we can divide the document in blocks of some terms and use the block number to represent the position of a term. While searching, we can reach to the block directly, but we will have to do linear search in that block. Thus, this block-level approach is a trade-off between space and time.

In this case, speeches are indexed by keywords that are automatically extracted or by other means of extraction from the transcribed speech. To index, Apache solr, a high performance full-featured search engine library written in Java is used. The collection of audio features are taken as input by the indexer to create a hierarchized structure of feature references. The index structure, which includes information for each feature, is dynamically updated each time as shown in Fig. 4.3.

Speeches are retrieved by searching the obtained free text which is obtained from transcribed Amharic speech repository file with the help of sub-words based speech recognition which is efficient methods for indexing and retrieving sequences. Proposed methods range from indexing fixed length subsequences such as triphones to full indexing of lattices represented as weighted automata. Finally, when user enters a text query, speeches are retrieved which match the exact or approximate match of substring of the query.

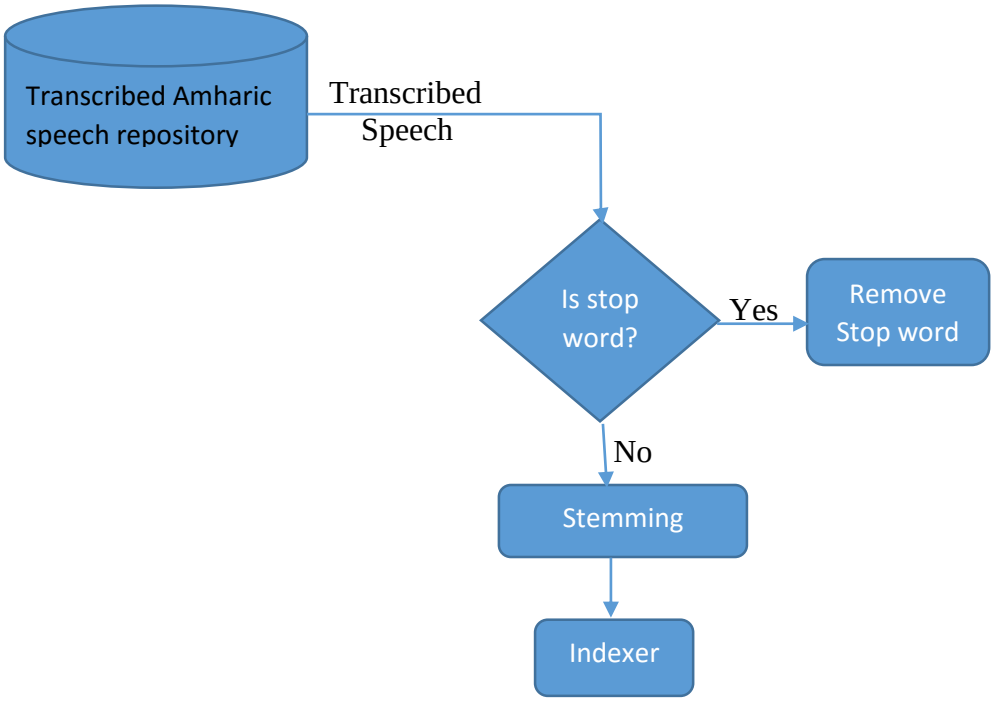


Figure 4- 3: Flow of Processes in Amharic Speech Analyzer

4.7 Query Engine

This component contains the search interface and query parser, and uses Amharic analyzer. The query parser uses the Amharic analyzer to do its best job to match the terms that were indexed. Ranking of search results is made by using the index and Amharic speech Page Repository.

4.7.1 Matching

After previous steps, we have stored the spoken documents. Now, our next step is to take a query and perform searching in the database for this query. Here, problem is to address the variability in duration of utterances. In ranking-queries or free-text queries, each document is assigned a matching score according to its similarity to the query using the vector space model. In this thesis, we concentrate on the ranking-queries, which are more frequently used in the Web search engines and IR systems. The cosine function is employed for query-document matching.

4.7.2 Ranking

Ranking is done by accessing the index. In order to rank pages, we propose to use the scoring factors. To calculate the score of a page we use the URL in suitable encoding format, the fetched

document and the in links. Scores are also distributed to out link, i.e., outgoing links from a page. Therefore, for proper ranking of Amharic speech web documents, we consider the following scoring factors.

1. The numbers of times the query terms are found in the specified transcribed speech. A transcribed speech which contains more of the query's terms receive a higher score.

2. The number of documents in which a given term appears in relation to the total number of documents in the index.

3. Giving different weighting factors for different fields of a document. A term's score in a document is itself the sum of the term run against each field that comprises a document.

Ranking Module. The ranking module takes the set of relevant pages and ranks them according to some criterion. The outcome is an ordered list of web pages such that the pages near the top of the list are most likely to be what the user desires.

The ranking module is perhaps the most important component of the search process because the output of the query module often results in too many (thousands of) relevant pages that the user must sort through. The ordered list filters the less relevant pages to the bottom, making the list of pages more manageable for the user.

In contrast, the similarity measures of traditional information retrieval often do not filter out enough irrelevant pages. Actually, this ranking which carries valuable, discriminatory power is arrived at by combining two scores, the content score and the popularity score. Many rules are used to give each relevant page a relevancy or content score. For example, many web engines give pages using the query word in the title or description a higher content score than pages using the query word in the body of the page [48]. The popularity score is determined from an analysis of the web's hyperlink structure. The content score is combined with the popularity score to determine an overall score for each relevant page [34]. The set of relevant pages resulting from the query module is then presented to the user in order of their overall scores.

```
1. Input: Query, Index
2. Output: Top-n matching documents
3. for each term in Query do
4.  Retrieve Index term from Index
5.  for each posting(d, f_{d,t})in index term do
6.  Disk access(d)=disk access(d)+ partialSimilarity(d,Query)
7. Extract top n matching documents.
```

Algorithm 4- 4: Algorithm for Ranking

4.8 Query Processor

The user interface of the system accepts text query which needs to be preprocessed before a match is to be checked from the index. First, the query needs to be tokenized, then normalized by removing punctuation marks from it, and expanding the query's short words of compound words, if any. Next, the query is passed to the stop-words remover and stemmer respectively. Finally, this preprocessed query is given to datafari's searcher module, for further processing and this will display the speech.

Text-based Speech Query

The Amharic query preprocessing component uses different techniques and produces words of a query in their normal forms. This component consists of subcomponents like normalization, tokenization, stop-words removal, and stemming. The output of this component is a set of Amharic bag of words which is used as an input for the Amharic query translation component.

Tokenization: it is the process of extracting words, discarding punctuations, removing common words, reducing words to a root form, or changing words into basic form from an input stream of characters [5].

Stop-words removing: which are filtered out before or after processing of natural language text. Most search engines and query translation systems do not consider common words in order to speed up search results. So, stop words should be removed from the query words. As a result, after the query is tokenized into words and the characters and short words are normalized, the next step is identifying and removing the Amharic stop words like **እንደ፣ እንዴት፣ ነው፣ ናቸው፣ ሆነ፣ ወደ፣ ግዜ፣ ውስጥ፣ ወይም፣** *etc* [5].

Stemming: it is the process of reducing inflected (or sometimes derived) words to their stem, base or root form. Many search engines use stemmer to compare the root forms of the search terms to the documents in their database. As Amharic language is morphologically complex, words are inflected through appended affixes (prefixes, suffixes and/or infixes). In this case, a stemmer is directly related to the information retrieval process, as having the stems of the words allows some phases of the information retrieval process to be improved, among which we can highlight the ability to index the documents according to their topics, as their terms are grouped by stems that are similar to concepts or the expansion of a query to obtain more and more precise results. It also reduces the size of the dictionary by avoiding the entry of different variants of a word in machine readable dictionaries [5].

In our Amharic query preprocessing component, after the Amharic stop words are removed, the stemming process has been performed on the remaining Amharic words. For example, the Amharic words *“”* **፣ "ወሰዱ ″፣ "ኮወሰዱ ″ ፣ "አልወሰደም ″ ፣ "አልወሰደችም ″,** etc. can be reduced to their citation word ***"ውስድ ″***. This helps the query processing component to handle morphological variations and to find matches of the query words as possible.

Chapter Five: Prototype and Results

5.1 Introduction

This chapter is dedicated to describe the implementation of Amharic speech search engine of which its architecture is discussed in Chapter Four. We describe in detail the development environment and use different open source web tool components. The tools used, the mechanisms of the tools use to crawl web documents and the configurations made on the tools for crawling, transcribe Speech and indexing Amharic speech web documents are described. These components and their brief explanations are given in the following sections. Development environment and tools are presented in section 5.1. Section 5.2 describe about the crawler modules. Preprocessing of documents before they get indexed is described in Section 5.3, 5.4 and 5.5. The indexer and implementation of the query engine component are given in Sections 5.6 and 5.7 respectively.

5.2 Development Environment and Tools

Development Environment

Developing a full-fledged speech search engine is resources intensive. High performance hardware, network bandwidth, and linguistic tools are among the basic resources that are required to determine the successful development and implementation of the speech search engines.

Crawling the web, storing the indexes, searching and retrieving web documents are network bandwidth, memory storage and high performance hardware intensive tasks. Preprocessing of user queries require the existence of linguistic tools for each language.

The development environment that we used in the development of our Amharic speech search engine is described as follow. Our system is developed and tested on a single personal computer of Intel core i3 processor with 2.50GHz speed, 6GB of RAM, 700GB of hard disk capacity, with Microsoft Windows 10 operating system. Software components used to develop and test our system are chrome, JDK, JSpider, Apache Tika, sphinx, solr, and Datafari.

Development Tools

During the development of the Amharic speech search engine prototype, we have selected and used the following tools.

Jspider

It is an open source project, written entirely in Java. It is an implementation of a highly flexible, configurable web robot engine. We have used version 0.5.0 for our purpose and discussed in the overview of crawler section [86].

Apache Tomcat 8.0

Apache Tomcat[3] is a web container which allows to run servlet and Java Server Pages based web applications. Apache Tomcat also provides by default a HTTP connector on port 8080, i.e., Tomcat can also be used as HTTP server. We used it as a container for our search engine application by handling HTTP requests and dispatching responses from different components of our search engine.

JDK 1.8.0_172

JDK 1.8.0_172[4] is the latest version open source software used for developing java based programs. It has been by far the most widely used Java SDK (Software Development Kit). With the Java 8 release, Java provides support for functional programming, running java based program, new JavaScript engine, new APIs for date time manipulation, new streaming API, etc.

We installed JDK1.8. 0_172 (Java Development Kit version 1.8.0_172) on a windows environment. Then we used Java programming language for implementing various components of the Amharic speech search engine architecture.

Apache TIKA

Tika was originally proposed in the Apache Nutch project. It provides a single uniform set of functions and a single Java interface to wrap around heterogeneous third-party parsing libraries. Tika should be embeddable within Java applications at low memory cost so that it's as easy to use Tika in a desktop class environment with large network and memory as it is with limited resources on which to operate. The necessity of detecting file formats and understanding them is ubiquitous

[3] http://www.vogella.com/tutorials/ApacheTomcat/article.html
[4] https://www.tutorialspoint.com/java8/java8_overview.htm

within software, and thus Tika should respond quickly when called upon. There are many existing metadata models to commonly describe files, and Tika has the burden of understanding all of the file formats that exist like: HyperText Markup Language, XML and derived formats, Microsoft Office document formats, Open Document Format, Portable Document Format, Electronic Publication Format, Rich Text Format, Compression and packaging formats, Audio formats, Image& Video formats, Java class files and archives, The mbox format, so it should in turn understand the formats' associated metadata models. Just as there are many metadata models per file format, there are also many parsing libraries. Tika should make it easy to use these within an application. MIME types provide an easy-to-use, understandable classification of file formats and Tika should control these classifications. There are numerous ways to detect MIME types based on their existing classifications, and Tika should provide a means for leveraging all or some combination of these mechanisms. Understanding what language a document's content is, it is one of the cornerstones of extracting metadata from it and its textual information, so Tika should make language identification [87].

Tika can detect several common audio formats and extract metadata from them. Even text extraction is supported for some audio files that contain lyrics or other textual content. The AudioParser and MidiParser classes use standard javax.sound features to process simple audio formats. The Mp3Parser class adds support for the widely used MP3 format.

5.3 The Crawler Modules

There are various open source crawlers differing from one another in terms of scalability, flexibility, politeness, distributed, freshness, quality, and their performance in different situation. Adaptation of particular crawlers by user or organization totally depends on their requirement.

Some of the most useful open source crawlers are Scrapy, Apache Nutch, Heritrix, WebSphinix, JSPider, GNUWget, WIRE, Pavuk, Teleport, WebCopier Pro, Web2disk, WebHTTrack etc. For our purpose, we have selected JSpider by looking the following features.

Key Features of JSpider [86]**:**

- Download complete web site, single page, some specific pages, etc.
- Easily configurable, scalable, multi-threaded, Java based and open source tool
- Check our site for errors (internal server errors)

- Check outgoing and/or internal link
- Besides, it easily checks web site errors within minutes (if the URL is wrong or resources are missing), lets us add our own configurations and rules, and its spidering speed is fast and configurable.

Taking the above discussed issues and features of the web crawlers, we have chosen JSpider for our Amharic speech Crawler.

5.3.1. The Amharic Speech Crawler

We used an adopted implementation of JSpider for crawling and downloading speech documents from the web. To doing so, first we should configure JSpider in implementing our work which are shown in Appendix-1. After completing the necessary requirements, to provide an appropriate environment to JSpider, we can now use to crawl in the web. As they are an open source which is fully written in a java programming language, with using it can crawl the web and find web pages in an automated manner and reduce lots of maintenance work, the JSpider web crawling can create a copy of all the visited web pages without duplicating and no need of SQL or others related to this as a database.

We used JSpider after changing the default configuration settings to meet our objectives. Thus, we set the configuration properties that fit to our Amharic speech search engine architecture.

For the purpose of experimentation, we selected the following Amharic URLs as seed URLs. These are

- http://www.fanabc.com/
- http://www.shegerfm.com/
- http://www.bisrate101.com/
- http://www.voaamharic.com/

The crawling and indexing processes are performed through a series of different steps. The process starts from a set of given base URLs, called seed URLs, which JSpider can crawl through the HTML structure in order to find links to speech files that are indexed by solr at the end. The entire process can be detailed in the following steps:

The main part of JSpider is the engine core that implements the most basic functionality.

All Jspider components configurations are kept in properties files that we can modify for our own use.

We are using Version 0.5 of the JSpider crawler released for Windows version. The JSpider is started from the command prompt via a startup script. After making important configurations, we type the command “jspider” followed by the seed URL we want to crawl and “download” command at the end. A sample snapshot of the crawling is shown in Figure 5.1 by supplying a sample seed URL www.shegerfm.com to start crawling the website of the Sheger FM radio.

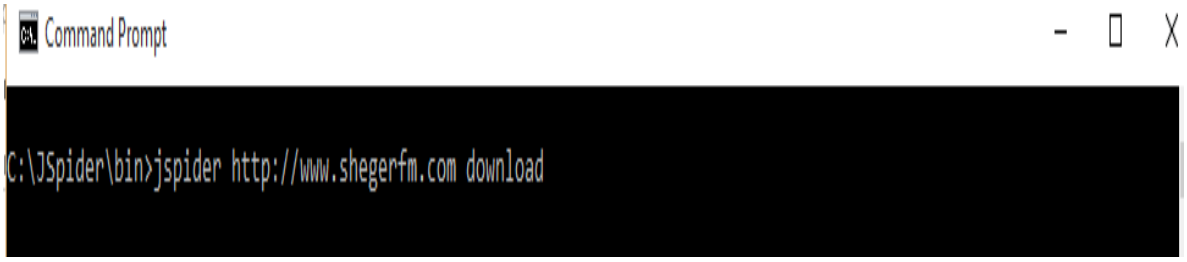


Figure 5- 1: Starting the JSpider web crawler

After we have downloaded pages from the selected seed URLs, we need to identify the pages with speech majority content.

5.4 Audio Extractor

Pages on the web can contain audio, video, image, text, etc. The page can also be in different formats, such as MP3, WAV, AU, MP4, Vorbis, Opus, Speex, MIDI, GIF, JPEG, text, Microsoft word, excel, power point, HTML, PDF, RTF and XML. Tika detect the file format, identify the language of the file, and extract it. Therefore, because most of the web audio documents are found in MP3, MP4, Vorbis, Opus, Speex, MIDI, Wav formats, we have used open source audio extraction tools; in our case Apache Tika is used for audio extractor.

Tika has different classes to parse different file formats. It also extracts the metadata of the document with the same procedure as in content extraction. So, it has a parser library that can parse the content of document formats and extract them. After detecting the type of the file, it selects the appropriate parser from the parser repository and passes the document [87] to the next phase of our speech search engine.

5.5 Amharic speech Web Document Identifier

The web is a huge repository with different types of data in different languages and it is difficult to browse the whole web, language specific search engines play a great role. The main component of such search engines is the language's documents categorizer. A given language's documents categorizer, also called language identifier, is a component of the crawler that assigns a given document to a specific category.

Multipurpose Internet Mail Extensions (MIME) standards are the best available standards for identifying document types. These standards help the browser during internal interactions. Tika supports all the Internet media document types provided in MIME. Whenever a file is passed through Tika, it detects the file and its type [87]. To detect media types, Tika internally uses the file extension mechanisms. Whenever you retrieve a file from a database or attach it to another document, you may lose the file's name or extension. In this case, the metadata supplied with the file is used to detect the file extension.

When a document is passed to Tika, it will detect the language in which it was written. It accepts documents without language annotation and adds that information in the metadata of the document by detecting the language. This needs the customization and reconfiguration of Apache Tike to do this for Amharic Speech.

To support language identification, Tika has a class called **Language Identifier** in the package **org.apache.tika.language**, and a language identification repository inside which contains algorithms for language detection from a given text. Tika internally uses N-gram algorithm for language detection [87].

So, after implementing this categorizer, we have tested it by using Amharic language audio documents and correctly identifies Amharic language audio documents from the collection. Sometimes, documents in the web can be multi-lingual and the proportion of the target language's documents can be very small. This percentage invalidates the language specifies of the search engine. Therefore, in a multi-lingual document the proportion (60% and above) must be kept. Due to this, we have decided to use both Tika and the Amharic language stop words based language identifier to categorize Amharic language documents so that these can be used for further processing.

Table 5- 1: Language Identifier

SNO.	**Constructors, Method and Description**
1	**LanguageIdentifier (LanguageProfile profile)** Instantiates the language identifier. Here you have to pass a LanguageProfile object as parameter.
2	**LanguageIdentifier (String content)** This constructor can instantiate a language identifier by passing on a String from text content.
3	**String** getLanguage () Returns the language given to the current LanguageIdentifier object.

Tika detects only eighteen languages [88] and does not include the Amharic language and need to customize this tool to identify the Amharic speech. To identify the language, we customized the constructors LanguageIdentifier (LanguageProfile profile) that instantiates the language identifier, passes LanguageProfile object as parameter, and LanguageIdentifier (String content) instantiate language identifier by passing on a string from text content. The language detection is performed with getLanguage() method of the LanguageIdentifier class. This method returns the code name of the language in String format.

5.6 Amharic Speech Recognizer

5.6.1. Audio process

Today audio is either in a video format with a composed video track and an audio track or they can be in an audio format with specific audio track. There are three possibilities that can take place before the audio can be processed, if it is a video file, then the audio needs to be extracted, and if it is not in wave format, then it needs to be converted to wave. The other possibility is that it is an audio file that is not in wave format, then it needs to be converted as well before being processed. For this purpose, MPlayer is going to be used since it can extract and convert any audio file to wave format that the speech recognizer supports [89].

MPlayer

MPlayer is open-source, latest media formats, which is easy embeddable in Java application using and it is ported for all the major operating systems like Windows, UNIX, and Mac. So, MPlayer is portability to embed into a Java application.

It can easily extract and convert an audio of any podcast file, it takes about 1-2 seconds fully optimized to extract an audio from a podcast file containing about 45 minutes content and at the same time get it to raw wave audio format [89].

The MPlayer command-line program is very efficient to extract and convert audio files stored on the podcast file, all it takes is the command “mplayer -quiet -ao pcm:fast:file=audio.wav –vo null -vc null -framedrop podcast” where “audio.wav" is the output file and “podcast” the input file. The "-quiet" switch will make the extraction and conversion process give faster since it does not have to output the results to the screen, the "-vo" and the "-vc" switches prohibits the possible existing video stream to be outputted to the screen, making the audio extraction and conversion to wave format at almost its full extraction and conversion performance.

5.6.2. Speech recognition

When the audio has been classified to contain speech, it is very useful to provide a text version of the speech. Conversion of speech to text is a necessary part of audio content- based system. Even though powerful IR algorithms are available to text, it is clear that for audio, or multimedia in general, common term matching approaches are useless due to the simple lack of identical words in audio documents [79].

Content-based retrieval of audio means retrieve a sound by specifying the exact numbers in the extract of the sound's sampled data; this is analogous to an exact text search, on the other hand it means match any sound containing the given extract without concern of the data's sample rate, quantization, compression, and so on. This is analogous to a fuzzy text search and can be implemented using correlation techniques. In order to implement audio content-based retrieval, the first step is the conversion of speech to text [79].

There are many tools that are used to recognize a speech like:

Google web search: which is introduced in Chrome version 25. Unfortunately, the speech API has not been released officially, though some reverse engineering solutions have been proposed by the community.

Kaldi: This is one of the newer speech recognition tool kits. Development began in 2009 at a workshop at John Hopkins University called "Low Development Cost, High Quality Speech Recognition for New Languages and Domains".

After working on the project for a couple of years, the code for Kaldi was released on May 14, 2011. Kaldi quickly gained a reputation for its ease to work with.

Julius: developed with C language in Japan by several projects and consortia since 1997. The project has focused on the Japanese language, thus it is available only with Japanese acoustic and linguistic Models.

Sphinx-4

Sphinx-4[5] is a pure Java speech recognition library. It provides a quick and easy API to convert the speech recordings into text with the help CMUSphinx acoustic models. It can be used on servers and in desktop applications, but so slow. Beside speech recognition Sphinx-4 helps to identify speakers, adapt models, align existing transcription to audio for time stamping and more. The sphinx recognition system is an open source, high performance system developed by the CMU sphinx project that can be used in building speech recognition applications. It also includes related resources such as acoustic model trainer, language model trainer, etc. Hence, Sphinx is available as a complete speech recognition system trainer and decoder. It uses HMMs with continuous output PDFs. As any library in Java all we need to do to use sphinx-4 is to add to jar and then we can write code using the API.

Sphinx stands for Site-oriented Processor for HTML INformation eXtraction is one of speech recognition engine that developed by The Sphinx works based on Hidden Markov Model (HMM) algorithm. The Sphinx that will use in this study is Sphinx-4 that written entirely in the JavaTM. Sphinx-4 is created via a joint collaboration between the Sphinx group at CMU, Sun Microsystems Laboratories, Mitsubishi Electric Research Labs, and Hewlett Packard, with contributions from the University of California programming language at Santa Cruz (UCSC) and the Massachusetts Institute of Technology.

Hidden Markov Model Toolkit (HTK)

[5] https://cmusphinx.github.io/wiki/tutorialsphinx4/

The HTK is open source software toolkit for building speech recognition systems. It is developed by the Cambridge University Speech Group. It has been improved and added properties over years since the beginning of the nineties.

The HTK is designed to be flexible enough to support both research and development of HMM systems. By controlling the tool, a speech recognition system can be implemented, tested and then its results can be inspected. A wide range of tasks can be performed, including isolated or continuous speech recognition using models based on whole word or sub-word units. HTK consists of a number of tools that perform tasks such as coding data, various styles of HMM training including embedded Baum-Welch re-estimation, Viterbi decoding, producing N-best lists or single recognition result. It can perform results analysis and editing HMM definitions, too. Especially, editing the HMM's externally enables the user to control the acoustic models with a considerable flexibility and also supports language model construction.

5.7 Indexer

The audio information retrieval needs to be transcribed before it is put to the indexer component. The term matching in text retrieval is not suitable to the audio information retrieval due to lack of identified keywords in the audio. Content-based audio retrieval could solve problems in the audio retrieval. Audio content based retrieval is different from text retrieval, and audio index is more complicated than text index [79]. So, we need to discuss the audio content and index, and compare text retrieval with audio retrieval.

Attempts to retrieve spoken documents encounter similar problems to those associated with text document retrieval, but there are further important issues that need to be considered. Most obvious among these is that the contents of spoken documents are unknown. The speech recognition system may also attempt to perform a full transcription of the contents of the documents using a large vocabulary recognizer. In speech recognition, the indexing vocabulary is limited to that of the recognizer. Many words, particularly proper nouns, cannot be recognized collectively by either recognition system since they are outside the domain of its vocabulary. This creates a significant search problem, which does not exist in text-based systems where new document terms are added into the file structure [79].

Additionally, speech recognition is not completely reliable. Even the very best systems will make recognition errors, often arising from variable pronunciation or events outside its domain.

The recognizer typically maps out-of-vocabulary words to something in its existing vocabulary, which will result in a recognition error. Short words are more susceptible to recognition errors than longer ones, both because of their inherent greater confusability and the greater tendency to poor articulation. Actual recognition is dependent on these and many other factors. The main significance is the degree of spontaneity in the speech, such as the amount of dialects, the formality of the linguistic structure, and the clarity of articulation. The effect of these factors is evidenced by the difference in recognition performance between formal dictation, where over 90% of the words can often be recognized correctly, and informal conversational speech, where performance may fall to less than 40% [79]. In any case it is important to realize that good text search terms may not be the best approach in the speech domain because of their acoustic properties.

Therefore, speech recognition is not the best approach to realize content-based retrieval. Other approaches are needed to improve audio retrieval. In this project, sphinx is used to extract audio contents.

The indexing service is handled by the most common indexing and searching java-based open source library called solr. Solr is a high performance, scalable Information Retrieval (IR) library. It helps to add indexing and searching capabilities to applications. Solr 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 [90]. Solr can index and make searchable any data that can be converted to a textual format.

Creating the Index

Solr defines six phonetic filters. Each one of them, given a sequence of characters, converts it into a tag according to its pronunciation. Those codes can be used to determine if two words sound alike. Filters supported by Solr were built on the basis of three most common solutions: Soundex, Metaphone and Carevphone [91].

There are three steps to complete the indexing process using solr. The first step is document converting where solr used to search and index any type of data which can be analyzed and extract textual information. The good thing about using Solr does not care about data format, types and

their languages, as long as we can convert the data to text. The second step is analysis where indexing and having created solr documents are preparing and sort the data indexed, while Solr makes the data more convenient for indexing. To do so, Solr works on the segmentation of textual data into parts. It works on remove all frequent but have no meaning tokens from the input, such as stop words. There is an important point about documents that contain metadata such as the author, the title, the last modified date, etc. The last step is storing the index in which Solr works to sort documents, such as words or numbers, to access to them quickly. Furthermore, Solr uses the (Inverted Index Feature) to sort the documents. Benefits to use inverted index, it is dividing Sentences into Words to speed up and optimize the search process [90].

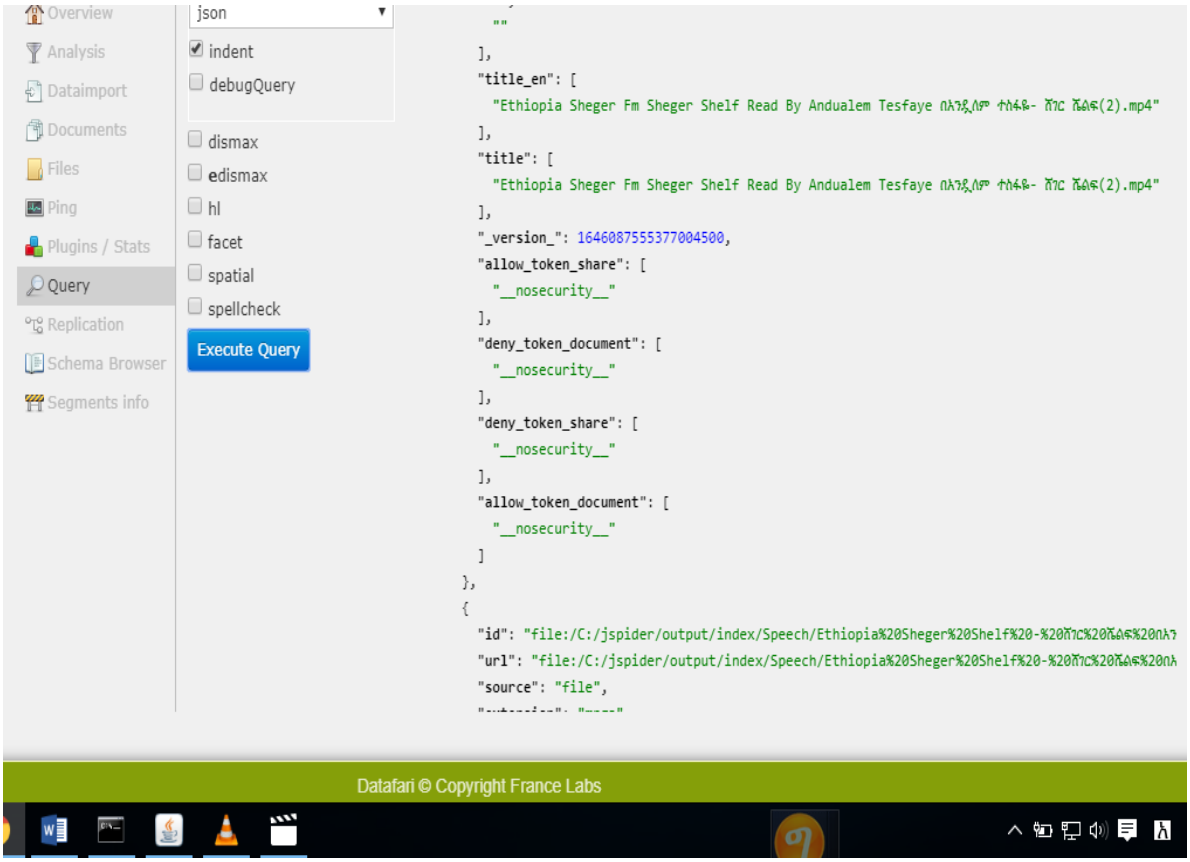


Figure 5- 2: Screen shot of the sample indexing result

5.8 Query Engine

When a user enters a query into a search engine, the engine examines its index and provides a list of best matching documents according to its criteria, usually with a short summary containing the document's title and sometimes parts of the document.

Query engine contains query interface for the user, analyzer, ranker and result interface. Query interface allows users enter their queries. The analyzer used, both in the indexer part and query engine, for the purpose of preprocessing the document. When matches to the keyword are found in the index, the documents that contain this keyword are displayed the speech on the interface.

5.8.1. Query Preprocessing

Once an index is built, we need to process the data in it to produce query results. Even with simple algorithms, processing queries using an index is much faster than it is without an index. However, better algorithms can enhance query processing speed over the simplest one. We will explore the simplest query processing tools and then move on to faster and more flexible modifications.

Our query engine is developed by using Datafari and it is hosted on Apache server (apache tomcat 8.0). It enables users to enter Amharic language text queries and has a search button that users submit their query to the system to obtain their information need. When the query is submitted, the query is preprocessed and the index is contacted. If the query gets a match in the index, the document that contains the keyword will be displayed on the display interface.

By using Apache Tomcat 8.0 web application server Datafari 2.1.4, the Amharic speech search engine user interface is displayed for a user to write text query and get the speech results. The result page contains links to the web pages containing the searched information along with their brief description in Amharic. Figure 5-2 shows the search user interface of our Amharic speech search engine. Our search engine is named **የአማርኛ ንግግር ሰርች ኢንጅን** “Amharic Speech Search Engine”.

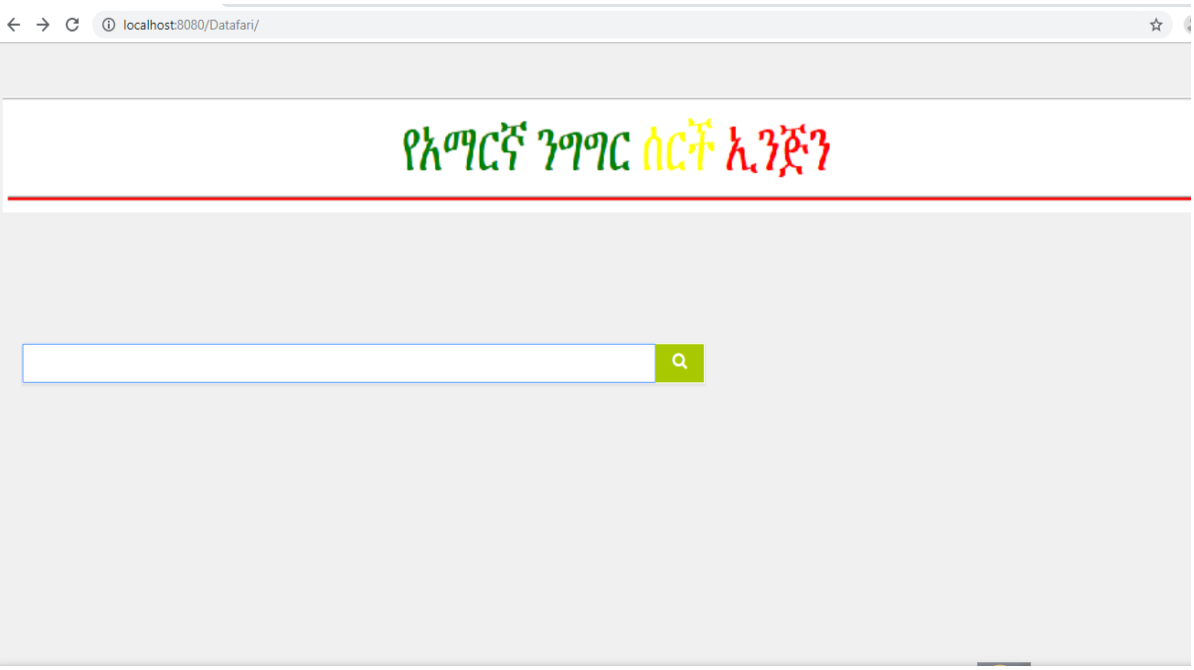


Figure 5- 3: የአማርኛ ንግግር ሰርች ኢንጂን “Amharic Speech Search Engine” User Interface

5.8.2. Ranking

Another important part in query Engine is ranking the results to the users. After storage the documents by Solr on local disk or hard disk, Solr works to removing the stop words and stemming words. Then Solr works to give ID and location to each document were stored and Solr has the Index Searcher. It is created using an analyzer. By it Solr take the user's query and analyze it, to find out where the user is seen. The Index Searcher can access to the documents and ranking the results according user's query. To satisfy the users’ interest, the retrieved documents must be returned in some order that brings documents that are important for the query term at the beginning of the display interface. IR systems like search engines, use the combination of text-based and link-based ranking to rank the retrieved relevant documents i.e. the sum of text-based and link-based ranking.

Text-based Ranking is a function that relies on the documents’ contents other than on the internal relations among the documents. Text-based ranking is explained in terms of TF-IDF based ranking [31] as discussed in section 2.2.1.

In the TF-IDF based ranking, documents and queries are expressed as vectors and their correlation is calculated using cosine similarity [31]. This method is used to find documents that are similar

to the query by calculating the cosine of the angle between them. In our case, the text based ranking is calculated using the solr scoring which is calculated by vector space model (term vector model) of solr library. Vector space model (VSM) is an algebraic model that is used to represent text documents as vectors. In this case, documents and queries are represented as vectors. As a result, solr calculates TF-IDF by using the cosine similarity by considering the angle between the document and query vectors.

Chapter Six: Experimental Result and Discussion

6.1 Introduction

In this chapter we discuss the experiment conducted to test the effectiveness of Amharic speech Search Engine. The test environment we used is a single personal computer of Intel Pentium IV Processor with Intel(R) Core(TM) i3-3120M CPU @ 2.50GHz, 2 Core(s), 4 Logical Processors, Microsoft Windows 10 operating system, 700GB Hard disk capacity, and 6 GB of RAM. We have tested our system using 11Mb/Second bandwidth. From the research objective point of view we will evaluate the following evaluation points.

6.2 Test Result of the Amharic Speech Crawler

We evaluated the performance of the crawler of the Amharic speech web resources. The crawler operation was invoked for seed URLs. We used depth of 10 and 50 threads to accomplish the crawling activity. The connection speed was 11Mbps. Crawling activities done two times, during the first run the crawler collect 85 documents in a minute. After extracting archived crawled web document using content extractor we verify that, the crawler downloaded different document types, videos, and audio and each documents are organized into its respective domains.

To run the crawler, we have configured JSpider with a thread and depth, which are the most important with the following URLs:

- www.fanabc.com/
- www.shegerfm.com/
- www.bisrat101.com/
- www.voaamharic.com/

These sites are selected because they contain large number of Amharic speech web resources. The crawling process continued until all the URLs are visited. Finally, the crawling process shows that the crawler has processed 1,519 Pages and 1047 of them are pages with Amharic language speech content. The sample report of the crawling process of the crawler conducted is shown in Appendix II.

6.3 Test Result of the Transcriber

When evaluating ASR systems the output sentence from the ASR-system must be aligned with the true transcription. When this is done, three error types can occur:

Substitutions: Cover words that are recognized wrongly. These include words that are in plural instead of singular.

Insertions: Extra words in the hypothesis that are not present in the reference.

Deletions: Are words present in the transcription but not in the sentence.

Table 6- 1: Transcriber Evaluation Results

Words	Matches	WER (%)	WA (%)
10852	10741	20%	80%

Quantification of the errors is done using the two measures, word error rate (WER) and word accuracy (WA), defined as:

$$\text{WER}=\frac{total\ word\ errors}{total\ words}$$

$$\text{WA}=\frac{total\ words-substitutions-delections}{total\ words}$$

Where, total word errors=substitutions + insertions + deletions and total words is the number of words in the reference.

6.4 Precision and Recall evaluation Results

In this section we measure the precision and recall of the Amharic speech search engine system in satisfying the user requirements in terms of the relevance of documents retrieved.

Precision is the ratio of the number of relevant documents retrieved to the total number of documents retrieved, i.e.

$$Precision = \frac{retrieved\ relevant\ document}{total\ retrieved\ document}$$

Recall is the ratio of the number of relevant documents retrieved to the total number of relevant documents (both retrieved and not-retrieved), i.e.

$$Recall = \frac{relevant\ documents\ retrieved}{total\ number\ of\ relevant\ documents}$$

So, we discussed the effectiveness of our speech search engine by taking 85 speeches and 10 randomly selected queries in terms of precision and recall as it is summarized in table 6-2.

Table 6- 2: Precision and Recall evaluation Results

No	Query	Relevant Documents in the collection	Retrieved Documents	Relevant Retrieved Documents	Precision	Recall
1	**አነጋጋሪ ንግግር**	35	38	34	0.89	0.97
2	**ያደረጉት ጨዋታ**	12	14	11	0.85	0.92
3	**አስገራሚ ነገር**	18	21	17	0.85	0.94
4	**የቋንቋችን ነገር**	25	27	24	0.89	0.96
5	**ህዝብን ያስደሰተዉ**	21	24	18	0.75	0.86
6	**አብይ አህመድ**	20	25	20	0.76	0.95
7	**መግባት ይችላል**	17	20	15	0.75	0.88
8	**ኢትዮጵያ ሕዝቦች**	12	15	11	0.73	0.92
9	**የትምህርት ጥራት**	10	12	10	0.75	0.90
10	**የማህበረሰብ ስጋት**	12	14	11	0.79	0.92
Average precision and recall					0.8	0.92

From the above table we have seen that the average precision is 80% and the average recall is 92%. Precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. From this, it can be understood that truly relevant result is 92%. Our search engine incorporates the typical features of the language both in its design and implementation.

Therefore, the users validated the results displayed by the Amharic speech search engine in response to the selected queries with that of their evaluation.

6.5 Testing of the Amharic Audio Language Identifier

Before using Tika, we have trained the module using the Amharic document. Because this text is Unicode (UTF-8) encoded, Ethiopic script document. Doing so, we subject this tool to categorize documents that are English, Tigrigna and Amharic documents. It is 99% efficient to identify that English document is not Amharic document. But it categorizes Amharic language documents as Tigrigna language documents if the Amharic language documents are Unicode encoded documents. Because we have trained Tika by Unicode encoded Ethiopic script, it does not categorize an Amharic language document as Tigrigna language document if its character encoding is out of UTF-8. So, this module is 99% efficient to identify the character encoding of the document in which it is trained for.

For multi-lingual web documents, we put some threshold in which Tika can categorize the document as the target language's document. Therefore, documents in which their Amharic content is above 60% are categorized as Amharic language documents.

6.6 Discussion

According to the evaluation of the Amharic speech search engine presented in section 6.3, the average precision and recall are 80% and 92% respectively.

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 operator of solr search on the terms in the query. This causes more number of speech to be retrieved. In effect, the precision will be negatively affected.

Chapter Seven: Conclusion

7.1 Conclusion

A large component of web content consists of audio information such as sound, music and speech in different languages and it is difficult to visit the whole web, there must be a software that enables us to fetch information from the web. So, search engines play a great role to make these information that the user needs accessible. There are a lot of search engines that are capable of searching information from the web. However, the existing search engines are not capable to handle the specific features of all languages' documents with several formats available on the web. The general purpose search engines are typically designed and developed to support English language and the Amharic search engine is designed primarily focusing on the textual web documents. Nevertheless, nowadays, different languages' documents are available on the web and users of these languages need to have the information they need in their desired language. Therefore, there must be a search engine that handles the Amharic language speech web documents. As a result, we have designed and developed a model for an audio search engine that handles Amharic language speech web documents.

The audio information on the web proves to be highly unstable. It requires a highly efficient automated search system in cataloging and retrieving information. The system regularly traverses the web, detects speech information, and indexes the speech in such a way as to allow efficient and effective search and retrieval. So, Amharic speech search engine has the four major components with their subcomponents.

The crawler is the components of the search engine that download audio files. The crawler spiders the web and puts the Amharic speech resources in the repository. The repository transfers the document stored in it to the second major component, the audio content processing, which identifies the Amharic speech and transcribe it.

The indexer has sub-components to achieve the analysis tasks. So, the indexer components creates the index in order to make it searchable form of the Amharic speech documents.

Query engine component is the fourth major component in which users interact with the search engine. It provides an interface that a user can enter queries in Amharic and the result of the query is also displayed to the user in the user interface.

The query results can be prioritized as all the documents gathered might not be equally important. So, we have used a ranking component that displays the query results in descending order. This means, the similarity of the query and the documents is measured by calculating the text-based similarity. Finally, the documents are shown in the order of descending the similarity values.

To implement these components and sub components, we have used different tools and development environments. We have used JSpider for crawling the web, Apache Tika for categorization, extraction and document format identification, sphinx for speech recognition and Solr for indexing.

As the effectiveness of any search engine must be evaluated, we have used precision–recall matrix to evaluate the effectiveness of our search engine. Consequently, we have obtained a precision value of 80% and a recall value of 92%.

Finally, it provides fundamentals to further develop audio searches in the future. The search would be improved if the implementation of the content-based retrieval can be added by an audio speech recognition, and audio clips are classified according to subjects such as music, speech, and so on.

7.2 Contributions

The contributions of this research work are:

- Developed algorithms for speech page identification, and Amharic speech identification.
- Proposed a general model for Amharic speech search engine.
- Customized different open source tools for crawler, language identification, and indexer to make a functional prototype.
- Developed an Amharic speech search engine based on the proposed architecture.

7.3 Recommendations

In this research, we designed and developed a model for an audio search engine for Amharic speech web resources. In order to develop a full-fledged audio search engine for Amharic audio web resources further research need to be done. Although the system had already demonstrated a good performance, there is still possibility for further improvement. Additional work may address the following recommendations in order to have a full- fledged audio search engine for Amharic speech web resources.

- A work on multimedia retrieval using the Amharic Audio search engine
- Investigation on web crawler that minimizes the amount of time required to crawl billions of websites
- Investigation on the Out Of Vocabulary (OOV) problem remains a major bottleneck for ASR and also for SCR. OOV issues can be addressed by better representations of uncertainty. Techniques that have been developed to deal with OOV include subword units, lattices and fuzzy matching, and a hybrid of them, which can be used to extract terms from the speech signal without necessitating full LVCSR
- Developing audio noise removal module
- Categorize web documents using machine learning approach
- A work on expansion techniques and of the exploitation of both manually and automatically generated metadata

References

[1] Ian Knopke, "AROOOGA: An Audio Search Engine for the World Wide Web," in *in Proceedings of the International Computer Music Conference*, Barcelona, Spain, 2004.

[2] Tessema Mindaye, "Design and Implementation of Amharic Search Engine," Addis Ababa University, Addis Ababa, 2007.

[3] Hassen Redwan, "Enhanced Design of Amharic Search Engine," Unpublished MSc thesis, Department of Computer science, Addis Ababa University, Addis Ababa, 2008.

[4] Hassen Redwan, Solomon Atnafu, Tessema Mindaye, "Searching the Web for Amharic Content," *Journal of Multimedia Processing and Technologies,* vol. vol. 1, pp. pp. 16-28, 2010.

[5] Mequannint Munye Zeru, "AMHARIC-ENGLISH BILINGUAL SEARCH ENGINE. Unpublished MSc Thesis, Department of Computer Science, ADDIS ABABA UNIVERSITY SCHOOL OF GRADUATE STUDIES," Addis Ababa, November, 2010.

[6] Florian Bäurle, "a User Interface for Semantic Full Text Search," Unpublished Master thesis, Faculty of Engineering, University of Freiburg, 2011.

[7] P. Nega A. and Willet, "Stemming of Amharic Words for Information Retrieval," *Literary and Linguistic Computing, Oxford, Oxford University press,* vol. 17, no. 1, pp. 1-17, 2002.

[8] N. A. Shruti Aggarwal, "Classification of Audio Data using Support Vector Machine," *International Journal of Computer Science and Technology,* vol. 2, no. 3, pp. 398-405, September 2011.

[9] S. V. G. B. Y. D. R. R. Kishore Prahallad, "Problems and Prospects in Collection of Spoken Language Data," (accessed on December 27, 2015). Available at: www.ulib.org/conference/2006..

[10] Solomon Atnafu, Tizita Zewide, " Audio Data Model for Multi-criteria Query Formulation and Retrieval," in *MoMM2009*, Kuala Lumpur, Malaysia, December 14–16, 2009.

[11] M. J. P. J. M. Yun-Hsuan Sung, "Deploying Google Search by Voice in Cantonese," INTERSPEECH, Google Inc., USA, 2011.

[12] I. Seng, "IMPROVING THE EFFICIENCY OF SEARCH ENGINES: STRATEGIES FOR FOCUSED CRAWLING, SEARCHING, AND INDEX PRUNING," A DISSERTATION SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING AND THE INSTITUTE OF ENGINEERING AND SCIENCE OF B˙ILKENT UNIVERSITY, July, 2009.

[13] C. M. a. M. M. Mansour Alghamdi, "Arabic Language Resources and Tools for Speech and Natural Language," (accessed on December 25, 2015). Available at: https://www.semanticscholar.org › paper › Arabic-Language-Resources-and-...

[14] F. W. Lancaster, Information Retrieval Systems: Characteristics, Test-ing and Evaluation, Wiley, New York, 1968.

[15] D. W. a. D. B. J. Oard, " A Survey of Multilingual Text Retrieval," Technical report, Technical Report UMIACS-TR-96-19 CS-TR-3615 of the Electrical Engineering Department, University of Maryland, USA, 1996.

[16] A. G. S. P. W. H. D. a. K. P. E. Hauptmann, "Multilingual Informedia: A Demonstration of Speech Recognition and Information Retrieval across Multiple Languages," in *In Proceedings of the DARPA Workshop on Broadcast News Understanding Systems,* , Lans-downe, VA, February 1998.

[17] G. M. N. Salton, Introduction to Modern Information Retrieval., New York: McGraw-Hill, 1983.

[18] D. M. T. S. W. Bruce Croft, Search Engines Information Retrieval in Practice, Pearson Education, Inc, 2015.

[19] E. Voorhees, "The TREC-8 Question Answering Track Report," in *In NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC-8)*, 1999.

[20] C. J. v. Rijsbergen, Information Retrieval, Butterworths, London, 2nd edition, 1979.

[21] W. F. a. R. Baeza-Yates, Information Retrieval: data structures and algorithms, New Jersey: Prentice Hall, 1992.

[22] A. Smeaton, "The TREC 2002 video track report. In E.M. Voorhees," in *The Eleventh Text Retrieval Conference (Trec-11),TREC 2002, NIST Special Publication 500-251*, NIST, Gaithersburg, Maryland, 2002.

[23] P. C. a. P. H. O. Celma, " Search Sounds: Anaudio crawler focused on weblogs.," in *Proc. 7th ISMIR*, 2006.

[24] M. M. G. Q. a. R. S. Alan F. Smeaton, "Taiscealai: Information Retrieval an Archive of Spoken Radio News," FORBAIRT under the Strategic Research Program grant ST-96-707, IRELAND (Accessed on April 17, 2016), 1998.

[25] P. W. M. Schauble, "First experiences with a system for content based Retrieval of Information from Speech recordings," In Proceding of the IJCAI workshop on Intelligent Multimedia Information Retrieval, 1995.

[26] K. Z. V. Ng, "An Invastigation of Subword Unit Representations for Spoken Document Retrieval," 1997 (available at https://groups.csail.mit.edu/kng/papers/sigir97.ps).

[27] www.wikipedia.org/wiki/Model. [Online].

[28] D. Hiemstra, "Using language models for information retrieval," Department of Centre for Telematics and Information Technology, PhD thesis,University of Twente, 2001.

[29] K. N. a. V. W. Zue, "Subword unit representations for spoken document retrieval," *in Proceedings of Eurospeech, ,* p. 1607–1610, 1997.

[30] K. N. a. V. W. Zue, "Subword-based approaches for spoken document retrieval," *Speech Communication,* vol. 32, no. 3, p. 157–186, 2000.

[31] A. W. a. C. S. Y. G. Salton, "A Vector Space Model for Automatic Indexing," *Communication of the ACM,* vol. 18, no. 11, pp. 613-620, 1975.

[32] G. a. B. C. Salton, "Term Weight Approaches in Automatic Text Retrieval.," *Journal of Information Processing and Management,* vol. 24, no. 5, pp. 513-523, 1988.

[33] G. S. a. C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management,* vol. 24, no. 5, pp. 513-523, 1988.

[34] C. B. a. M. M. A. Singhal, "Pivoted document length normalization," *in Proceedings of the International ACM Special Interest Group on Information Retrieval (SIGIR) Conference on Research and Development in Information Retrieval,* pp. 21-29, 1996.

[35] C. J. V. R. GIANNI AMATI, "Probabilistic Models of Information Retrieval Based on the divergence from Randomness," *ACM Transactions on Information Systems,* vol. 20, no. 4, p. 357–389, October 2002.

[36] K. C. S. H. L. a. H. T. N. Tee Kiah Chia, "Statistical lattice based spoken document retrieval," *ACM Trans. Inf. Syst,* vol. 28, 2010.

[37] S. Parlak, "Speech Retrieval for Turkish Broadcast News," Master's thesis, Bogazici University, Department of Electrical and Electronics Engineering, 2008.

[38] M. Y. T. a. W. Menzel, "Amharic Part-of-Speech Tagger for Factored Language Modeling," in *International Conference RANLP* , Borovets, Bulgaria, 2009.

[39] R. R. D. V. B. W. P. P. S. Pratik K. Kurzekar, "A Comparative Study of Feature Extraction Techniques for Speech Recognition System," *International Journal of Innovative Research in Science, Engineering and Technology,* vol. 3, no. 12, pp. 18006-18016, December 2014.

[40] V. Dimattia, "An Automatic Audio Segmentation System for Radio Newscast," Master thesis,Department of Signal Theory and Communications, UPC Terrassa, 2008.

[41] A. A.-A. T. K. S. A. a. M. A. Hasan Muaidi, "Arabic Audio News Retrieval System Using Dependent Speaker Mode, Mel Frequency Cepstral Coefficient and Dynamic Time Warping Techniques,"

Research Journal of Applied Science, Engineering and Technology, vol. 7, no. 24, pp. 5082-5097, 2014.

[42] Kinfe Tadesse, "Sub-word based Amharic speech recognizer: An experiment using Hidden Markov Model (HMM). Unpublished MSc Thesis, School of Information Studies for Africa," Addis Ababa University, Ethiopia, 2002.

[43] J. A. Markowitz, " Using Speech Recognition," Prentice Hall, Inc., Upper Saddle River, NewJersey, 1996.

[44] L. a. B.-H. J. Rabiner, "Fundamentals of Speech Recognition," Prentice Hall, Inc., Englewood Cliffs,New Jersey, 1993.

[45] T. Y. T. a. S. Y. Fukada, "Automatic generation of multiple pronunciations based on neural networks.," In proceeding of Speech Communication, Kyoto, Japan, 1999.

[46] D. J. a. J. H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, India: Prentice Hall, 2008.

[47] C. M. P. A. Jouni Pohjalainen, "Enhancing Noise Robustness in Automatic Speech Recognition Using Stablized Weighted Linear Prediction," in *ISCA ITRW, Speech Analysis and Processing for Knowledge Discovery*, Denmark, June 4-6, 2008.

[48] M. L. a. G. J. F. Jones, "Spoken Content Retrieval: A Survey of Techniques and Technologies," *Foundations and Trends in Information Retrieval,* vol. 5, no. 5, pp. 235-422, 2012.

[49] G. TZANETAKIS, "MANIPULATION, ANALYSIS AND RETRIEVAL," unpublished, PRINCETON UNIVERSITY, JUNE 2002.

[50] D. Mitrovic, "Features for Content-Based Audio Retrieval," *Advances in computers,* vol. 78, pp. 71-150, 2010.

[51] M. N. Stuttle, "A Gaussian Mixture Model Spectral Representation for Speech Recognition," Hughes Hall and Cambridge University, 2003.

[52] J. T. Foote, "Content-based retrieval of music and audio," *In Voice, Video, and Data Communications,* pp. 138-147, 1997.

[53] A. B. a. R. M. G. Y. Linde, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communication,* vol. 28, no. 1, pp. 84-95, 1980.

[54] S. S. a. V. J. Stevens, "The relation of pitch to frequency: A revised scale," *The American Journal of Psychology,* vol. 53, no. 3, pp. 329-353, 1940.

[55] S. B. D. a. P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. 28, no. 4, pp. 357-366, 1980.

[56] A. A. a. H.-W. H. X. Huang, Spoken Language Processing: A guide to theory, algorithm and system development, Prentice Hall PTR, 2001.

[57] I. M. a. N. F. Theodoros Theodorou, "Automatic Sound Classification of Radio Broadcast News," *International Journal of Signal Processing, Image Processing and Pattern Recognition,* vol. 5, no. 1, pp. 37-48, March, 2012.

[58] G. T. a. P. Cook, "Sound analysis using MPEG compressed audio. In," in *In Proc. Int. Conf. on Acoustics, Speech and Signal Processing ICASSP*, Istanbul, 2000.

[59] V. v. P. Sirotkin, "On Search Engine Evaluation Metrics," A dissertation submitted to the department of information science, Heinrich-Heine-Universität Düsseldorf, 2012.

[60] E. A. Brewer, "Combining Systems and Databases: A Search Engine Retrospective," in *Database System Fourth Edition*, University of California at Berkeley, 2005.

[61] W. G. Stock, "Information Retrieval: Informationen suchen und finden," Cambridge University Press, München, Oldenbourg, 2007.

[62] P. S. H. S. Yaduvir Singh, "Role of Web Crawler in Search Engine," *International Journal of Artificial Intelligence and Mechatronics,* vol. 2, no. 1, 2013.

[63] D. G. M.P.S.Bhatia, "Discussion on Web Crawlers of Search Engine," in *Proceedings of 2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008)*, 2008.

[64] R. A. Amit Chawla, "CRAWLING THE WEB: DISCOVERY AND MAINTENANCE OF LARGE-SCALE WEB DATA," *Proceedings of 2nd International Conference on Emerging Trends in Engineering and Management, ICETEM,* vol. 3, no. 3, pp. 62-66, 2013.

[65] M. D. K. Mr.K. Tarakeswar, "Search Engines:A Study," *Journal of Computer Applications (JCA),* vol. 6, no. 1, pp. 29-33, 2011.

[66] S. B. a. L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN systems,* vol. 30, no. (1-7), pp. 107-117, 1998.

[67] B. M. Swati Mali, "Focused Web Crawler with Page Change Detection Policy," in *International Journal of Computer Applications (IJCA)*, 2011.

[68] J. &. G.-M. H. Cho, "Parallel crawlers," in *In Proceedings of the eleventh international conference on World Wide Web, 124–135*, Honolulu, Hawaii, USA, 2002.

[69] R. D. K. R. C. D. Trupti V. Udapure, "Study of Web Crawler and its Different Types," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 16, no. 1, pp. 01-05, 2014.

[70] A. P. Mu. Annalakshmi, "Search Factors used by Search Engines," *International Journal of Advanced Research Trends in Engineering and Technology (IJARTET),* vol. 3, no. 20, pp. 652-655, 2016.

[71] C. J. V. Rijsbergen, INFORMATION RETRIEVAL, 2nd ed. London: Butterworth, 1979.

[72] C. a. Vina, "Fidel Cacheda and Angel Vina. Inverted files and dynamic signature files for optimization of Web directories," in *In The Eleventh International World Wide Web (WWW) Conference*, 2002.

[73] J. M. V. T. a. P. J. M. B. Logan, "Approaches to reduce the effects of OOV queries on indexed spoken audio," *IEEE Transactions on Multimedia,* vol. 7, no. 5, p. 899–906, 2005.

[74] M. M. a. M. S. C. Allauzen, "General-indexation of weighted automata-application to spoken utterance retrieval," in *In Proc. Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval at HLT-NAACL*, Boston, MA, USA, 2004.

[75] D. M. a. T. S. B. Croft, Search Engines: Information Retrieval in Practice., Addison Wesley, 1st Edition, 2009.

[76] A. M. a. T. C. B. I. H. Witten, Managing Gigabytes: Compressing and Indexing Documents and Images, Morgan Kaufmann, 1999.

[77] R. M. Voigt, "The classification of central semitic," *Journal of Semitic Studies, , no. 32, pp. 1–21, 1987.,* no. 32, pp. 1-21, 1987.

[78] W. Leslau, Introductory Grammar of Amharic,, Wiesbaden: Harrassowitz., 2000.

[79] Z. Ma, "Study of Audio Information Retrieval on the World Wide Web," Auburn University, Alabama, 2000.

[80] W. &. T. J. Cavnar, "N-gram based text categorization," in *In Proceedings of SDAIR-94. 3rd Annual Symposium on Document Analysis and Information Retrieval. Paper presented at SDAIR-94*, Las Vegas, NV, 1994.

[81] "Evaluation of Language Identification Methods," Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.720&rep=rep1&type=pdf (Accessed on june, 2018).

[82] L. Y. Chi, "The Use of Subword-based Audio Indexing in Chinese Spoken Document Retrieval," Masters Thesis, The Chinese University of Hong Kong, August, 2001.

[83] G. W. Z. H. X. T. M. J. P. J. M. Jiulong Shan, "Search by Voice in Mandarin Chinese," *INTERSPEECH,* pp. 354-357, 26-30 September 2010.

[84] D. M. a. D. J. S. Susan Maze, Authoritative Guide to Web Search Engines, New York: Neal-Schulman, 1997.

[85] Martha Yifiru, Ermias Abebe, Solomon Teferra, "Towards the Development of Speech Recognition Application," Addis Ababa University, School of Information Science, Addis Ababa, June 06, 2015.

[86] JSpider, Available at: http://j-spider.sourceforge.net/, Last Accessed on Dec 20, 2018.

[87] J. L. Z. Chris A. Mattmann, Tika in Action, Island: Manning Publications Co., 2012.

[88] "www.tutorialspoint.com," 2019. [Online]. Available: https://www.tutorialspoint.com/tika/tika_language_detection.htm. [Accessed 2019].

[89] A. Holst, "Automatic Transcript Generator for Podcast," Institutionen för datavetenskap, fysik och matematik, 2010.

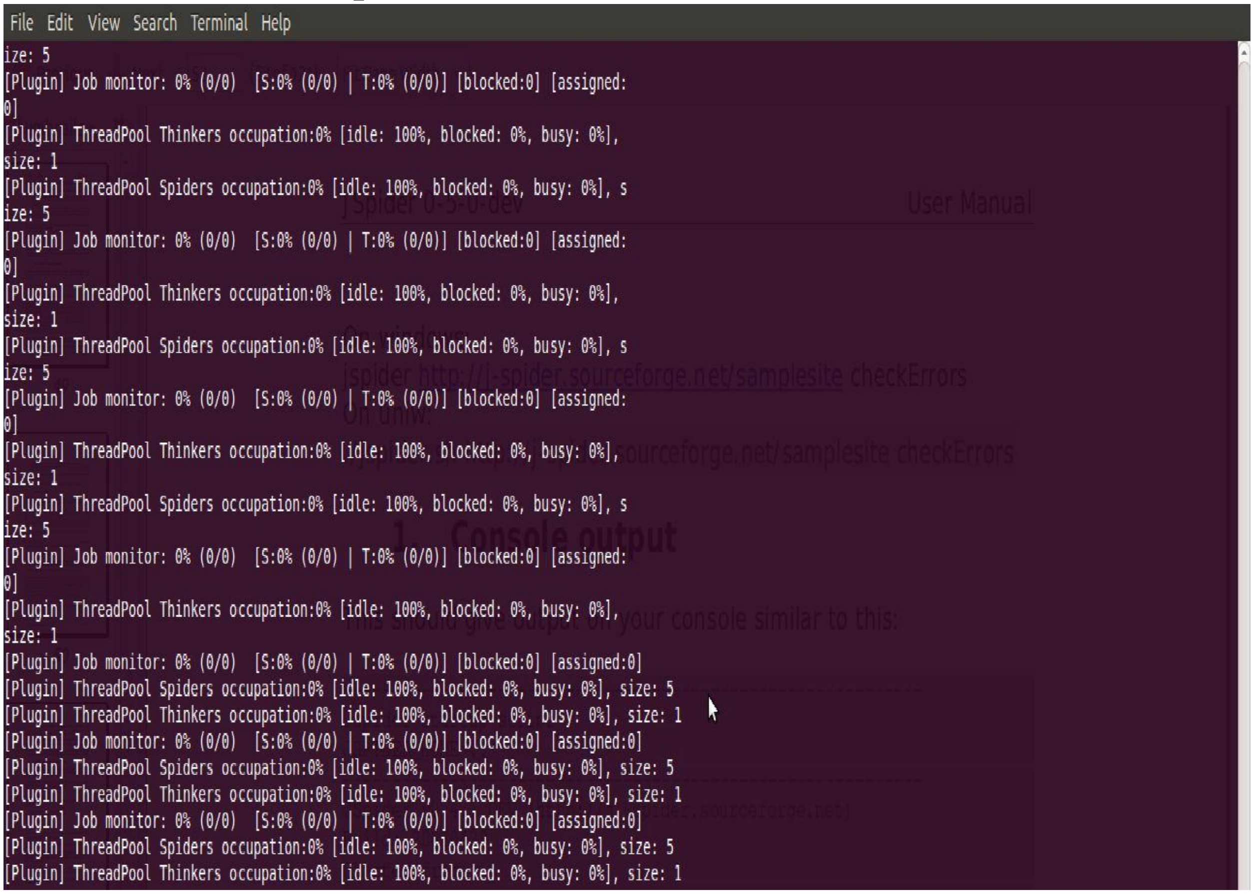
[90] N. ABDULWAHID, "CRAWLING THE WEB USING APACHE NUTCH AND LUCENE," ÇANKAYA UNIVERSITY, 2014. Available at: https://pdfs.semanticscholar.org ›.

[91] A. DUDZIEC, "Search system for an audio archive," DEGREE PROJECT, IN COMPUTER SCIENCE , SECOND LEVEL, STOCKHOLM, SWEDEN, 2015.

APPENDIX-I: Configuration of JSpider for Amharic speech Search Engine

Name	Value
jspider.proxy.use	True
jspider.proxy.host	
jspider.proxy.port	8080
jspider.proxy.authenticate	False
jspider.proxy.user	
jspider.proxy.user	
jspider.threads.thinkers.monitoring.enabled	True
jspider.threads.spiders.count	10
site.rules.spider.3.config.depth.min	0
site.rules.spider.3.config.depth.max	9

APPENDIX-II: Sample Crawl Test



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: Arega Hassen

Signature: _____

Date: _____

Confirmed by advisor:

Name: Solomon Atnafu (PhD)

Signature: _____

Date: _____

Place and date of Submission: Addis Ababa, October, 2019