AIN SHAMS UNIVERSITY

Spoken Audio Files Search based on Keyword Spotting

Andira Hayder Ibrahim
Menna Allah Salah El-Wakiel
Mostafa Sayed El-Fouly
Rana Fawzy El-Sayed

Supervised by:
Dr. Amira Hamdy
Dr. Mohamed Abdeen

Faculty of Computer and Information Sciences
Department of Computer Science

June 2009

Abstract

The amount of data is increasing vastly in our times and with this increase in the amount of data, the need to have access to this data is also increasing. Some of this data is easily accessible like textual data but other are more complex like media data. Our project is concerned with the content of spoken audio media. It enables searching inside the content of audio files.

Acknowledgements

We would like to thank everyone who helped us in accomplishing this project, specially our project supervisors; Dr. Amira Hamdy and Dr. Mohamed Abdeen.

Contents

List of Figures

Abbreviations

ASR	**A**utomatic **S**peech **R**ecognition
KWS	**K**ey**w**ord **S**potting
OOV	**O**ut **O**f **V**ocabulary
HMM	**H**idden **M**arcov **M**odel

Chapter 1

Introduction

- Problem Definition
- Motivation
- Approach
- Related Work
- Outline of Documentation

1.1 Problem Definition

Our project is a search engine that searches for a given word inside the content of spoken audio files or recorded speech e.g. recorded lectures, audio books, audio pod casts, etc. As it is widely recognized that the transformation from Speech to Text is possible so the audio files will be uploaded by the users and then transformed into text to be searched. Users will use text words to search inside the website which will be matched to the audio files where the word was found. The keywords for each search can be found in the title of the audio file or inside its content.

1.2 Motivation

The idea was initiated by the observation of the increasing amount of data each day on the web and the need to accessing this data. Search engines have made this data easier to access only for text files while accessing other formats of data like Images, Audio and Video is not as efficient. So our spoken audio files search engine will help our user to find exactly what he was looking for inside the content of an audio file which was not available for him before.

1.3 Approach

The recognition system depended on keyword spotting (Filler-based) using hidden markov models. The aim of keyword spotting system is to detect a small set of keywords from a continuous speech. It is important to obtain the highest possible keyword detection rate without increasing the number of false insertions in this system. Modeling only keywords is not enough. To separate keywords from non-keywords, models for out-of-vocabulary words are needed, too. Since the structure and type of garbage model has great effect on the entire system performance, out-of-vocabulary modeling is done by the use of garbage models. The Hidden Markov Models is the most widely used recognition algorithm in the past fifteen years, it is a finite set of states, each of which is associated with a probability distribution. Transition probabilities are assigned to the transitions among the states. In a particular state an outcome or observation can be generated, according to the associated probability distribution. The external Observer can only see the outcome, not the states. Therefore states are hidden to the outside. By using The Hidden Markov Models toolkit and the key word spotting algorithm the recognizer was build.

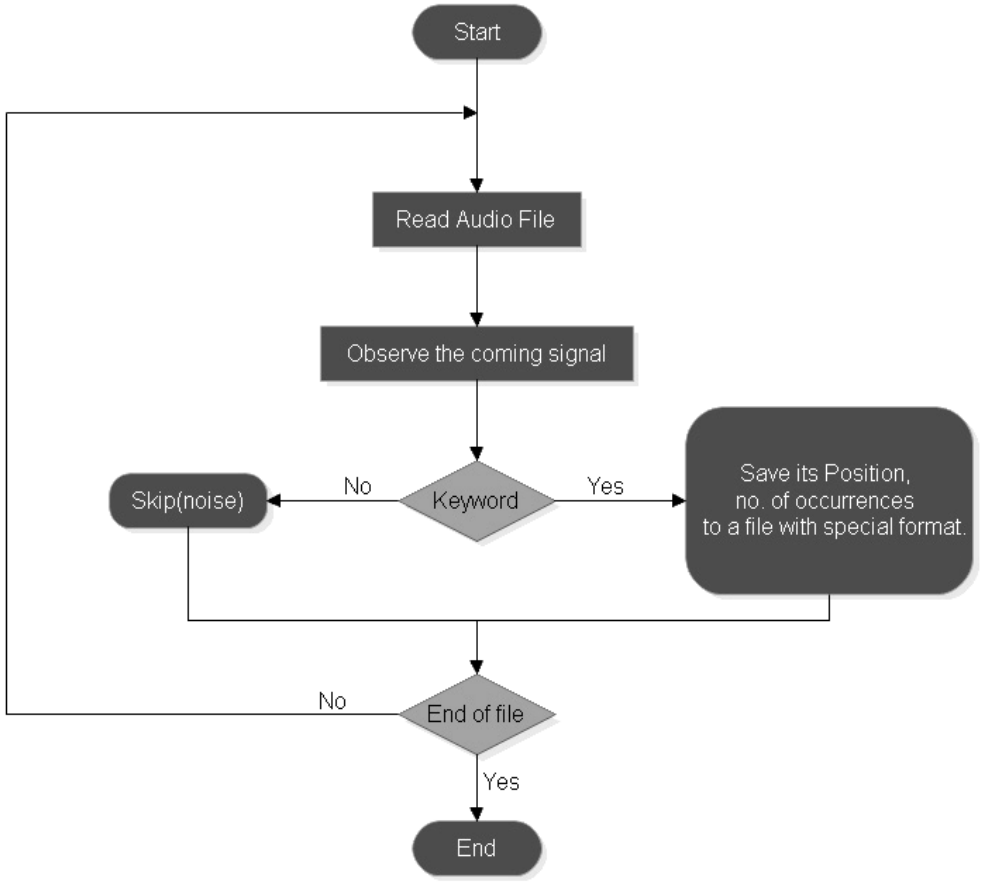

FIGURE 1.1: Recognition Process Example

1.4 Related Work

There is, of course, a number of systems for performing audio search already available, most of them are experimental systems. Commercial systems operate both on the web and are used in local settings, such as on home computers. Meanwhile, there are a vast number research projects both ongoing and completed that attempt to tackle the challenge of searching among images in various ways.

There are commercial systems including Nexidia/Fast- Talk (www.nexidia.com), Virage/AudioLogger (www.virage.com), Convera (www.convera.com) as well as research systems like AT&T DVL (Cox et al., 1998), AT&T ScanMail (Hirschberg et al., 2001), BBN RoughnReady (Makhoul et al., 2000), CMU Informedia (www.informedia.cs.cmu.edu), SpeechBot (www.speechbot.com), among others. Also between 1997 and 2000, the Test REtrieval Conference (TREC) had a spoken document retrieval (SDR) track with many participants (Garofolo et al., 2000). NIST TREC-9 SDR Web Site (2000) states that: The results of the TREC-9 2000 SDR evaluation presented at TREC on November 14, 2000 showed that retrieval performance for sites on their own recognizer transcripts was virtually the same as their performance on the human reference transcripts. Therefore, retrieval of excerpts from broadcast news using automatic speech recognition for transcription was deemed to be a solved problem - even with word error rates of 30%.

PhD Theses written on this topic include James (1995), Wechsler (1998), Siegler (1999) and Ng (2000). Jones et al. (1996) describe a system that combines a large vocabulary continuous speech recognition (LVCSR) system and a phone-lattice word spotter (WS) for retrieval of voice and video mail messages (Brown et al., 1996). Witbrock and Hauptmann (1997)

present a system where a phonetic transcript is obtained from the word transcript and retrieval is performed using both word and phone indices. Wechsler et al. (1998) present new techniques including a new method to detect occurrences of query features, a new method to estimate occurrence probabilities, a collection-wide probability re-estimation technique and feature length weighting. Srinivasan and Petkovic (2000) introduce a method for phonetic retrieval based on the probabilistic formulation of term weighting using phone confusion data. Amir et al. (2001) use indexing based on confusable phone groups and a Bayesian phonetic edit distance for phonetic speech retrieval. Logan et al. (2002) compare three indexing methods based on words, syllable-like particles, and phonemes to study the problem of OOV queries in audio indexing systems. Logan and Van Thong (2002) give an alternate approach to the OOV query problem by expanding query words into in-vocabulary phrases while taking acoustic confusability and language model scores into account.

1.5 Outline of Documentation

Chapter 2 gives a background information about Speech Recognition Systems and Media Serch Systems.

Chapter 3 presents the Analysis and Design of the project. It contains the use case, the sequence diagrams, activity driagrams and class diagram.

Chapter 4 gives a detailed description of the implementation of the projects and tools used.

Chapter 5 presents the performance measures for the project.

Chapter 6 presents the conclusions and the future works.

Chapter 2

Background Information

- Speech Recognition Systems
  - Overview
  - Speech Recognition Basics
  - Speech Recognition Problem
  - Types of Speech Recognition
  - Acoustic Modeling and Training
  - Hidden Markov Models
  - Artificial Neural Nets
  - Dynamic Time-Warping
- Keyword Spotting
- Audio Content Search
  - Overview
  - Architecture for Spoken Media Search
  - Applications of Audio Content Search

2.1 Speech Recognition Systems

2.1.1 Overview

Over the last few decades there has been increasing research activity in speech recognition (SR) technology. Research in automatic speech recognition (ASR) aims to develop methods and techniques that enable computer systems to accept speech input and to transcribe the recognized utterances into orthography. Current SR systems are based on the principles of statistical pattern recognition [1]. The speech signal is a time-varying and therefore nonstationary signal. When the speech signal is divided into block of samples (called frames) it can be considered stationary and some analysis can be performed with this frame methodology. The first step of speech recognition is to convert an unknown speech waveform into a sequence of acoustic vectors, O $= o_1; o_2; \cdots ; o_t$ (here each vector denotes the parametric equivalent of a frame in the acoustic data). If we assume that the acoustic data is produced by a sequence of words, $W = w_1; w_2; \cdots ; w_n$, then we can define the aim of a speech recognition system as determining the most probable word sequence, W, given the observed acoustic signal, O.

Speech recognition is a difficult problem, largely because of the many sources of variability associated with the signal.

First, the acoustic realizations of phonemes, the smallest sound units of which words are composed, are highly dependent on the context in which they appear. This phonetic variability is exemplified by the acoustic differences of the phoneme in two, true, and butter in American English. At word boundaries, contextual variations can be quite dramatic. Second, acoustic variability can result from changes in the environment as well as in the position and characteristics of the transducer. Third, within-speaker variability can result from changes in the speaker's physical and emotional state, speaking rate, or voice quality. Finally, differences in sociolinguistic background, dialect, and vocal tract size and shape can contribute to across-speaker variability.

This Figure shows the major components of a typical speech recognition system. The digitized speech signal is first transformed into a set of useful measurements or features at a fixed rate, typically once every 10–20 msec. These measurements are then used to search for the most likely word candidate, making use of constraints imposed by the acoustic, lexical, and language models. Throughout this process, training data are used to determine the values of the model parameters.

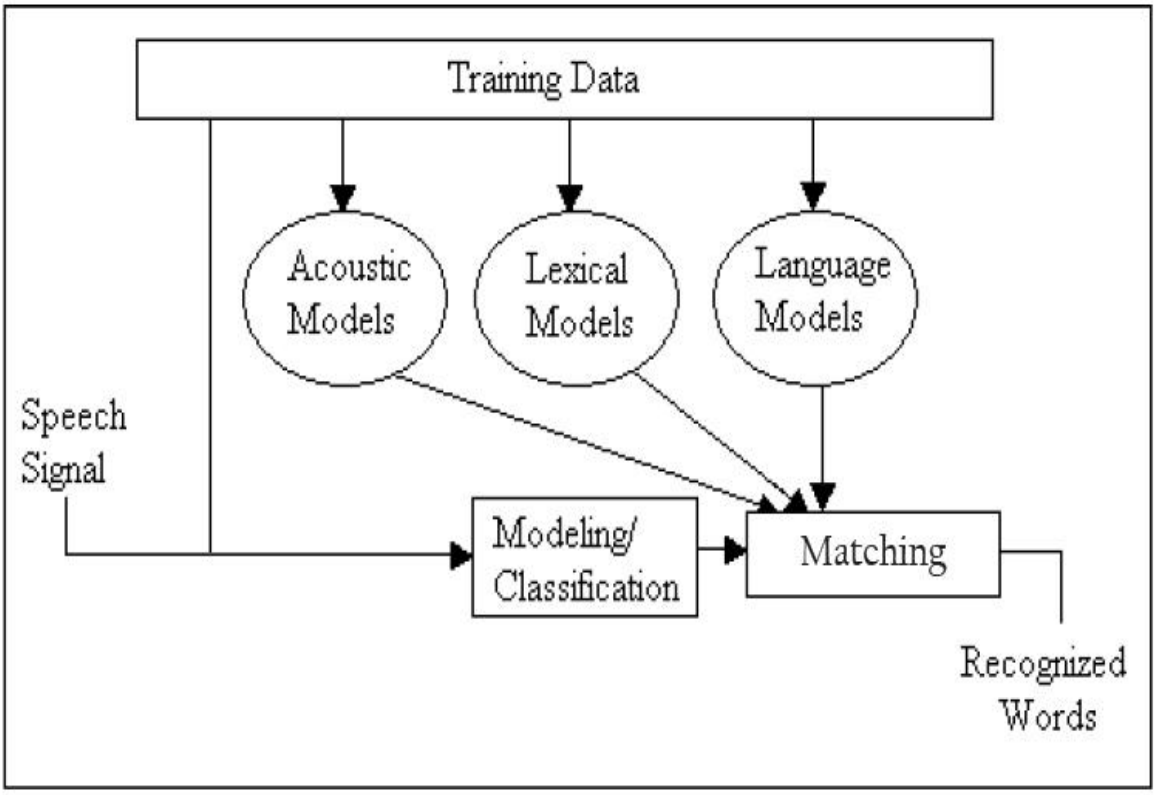

FIGURE 2.1: Generic Architecture for Spoken Media Search Systems

2.1.2 Speech Recognition Basics

The following definitions are the basics needed for understanding speech recognition technology.

Utterance An utterance is the vocalization (speaking) of a word or words that represent a single meaning to the Computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences.

Speaker Dependence Speaker dependent systems are designed around a specific speaker. They generally are more accurate for the correct speaker, but much less accurate for other speakers. They assume the speaker will Speak in a consistent voice and tempo. Speaker independent systems are designed for a variety of speakers. Adaptive systems usually start as speaker independent systems and utilize training techniques to adapt to the speaker to increase their recognition accuracy.

Vocabularies Vocabularies (or dictionaries) are lists of words or utterances that can be recognized by the SR system. Generally, smaller vocabularies are easier for a computer to recognize, while larger vocabularies are more difficult. Unlike normal dictionaries, each entry doesn't have to be a single word. They can be as long as a sentence or two. Smaller vocabularies can have as few as 1 or 2 recognized utterances (e.g."Wake Up"), while very large vocabularies can have a hundred thousand or more!

Accurate The ability of a recognizer can be examined by measuring its accuracy ? or how well it recognizes utterances. This includes not only correctly identifying an utterance but also identifying if the spoken utterance is not in its vocabulary. Good ASR systems have an accuracy of 98

Training Some speech recognizers have the ability to adapt to a speaker. When the system has this ability, it may allow training to take place. An ASR system is trained by having the speaker repeat standard or common phrases and adjusting its comparison algorithms to match that particular speaker. Training a recognizer usually improves its accuracy. Training can also be used by speakers that have difficulty speaking, or pronouncing certain words. As long as the speaker can consistently repeat an utterance, ASR systems with training should be able to adapt.

2.1.3 Speech Recognition Problem

The speech signal is different if input is given with isolated words or the speech is continuous. If the speaker knows that a computer will try to recognize the speech, then he/she may pause between words. However in continuous speech some sounds will disappear and sometimes there will be no silence between words. It may be hard to say a word in a different context. An exaggerated example may be a tongue twister. Even in normal cases there is great difference in the characteristics of the speech signal. Figure 2.1 shows the same /e/ sound In ev and Ben eve. The waveforms are shown at the top of the figure. The spectrograms at the bottom show the energy at different frequencies versus time. The darkness

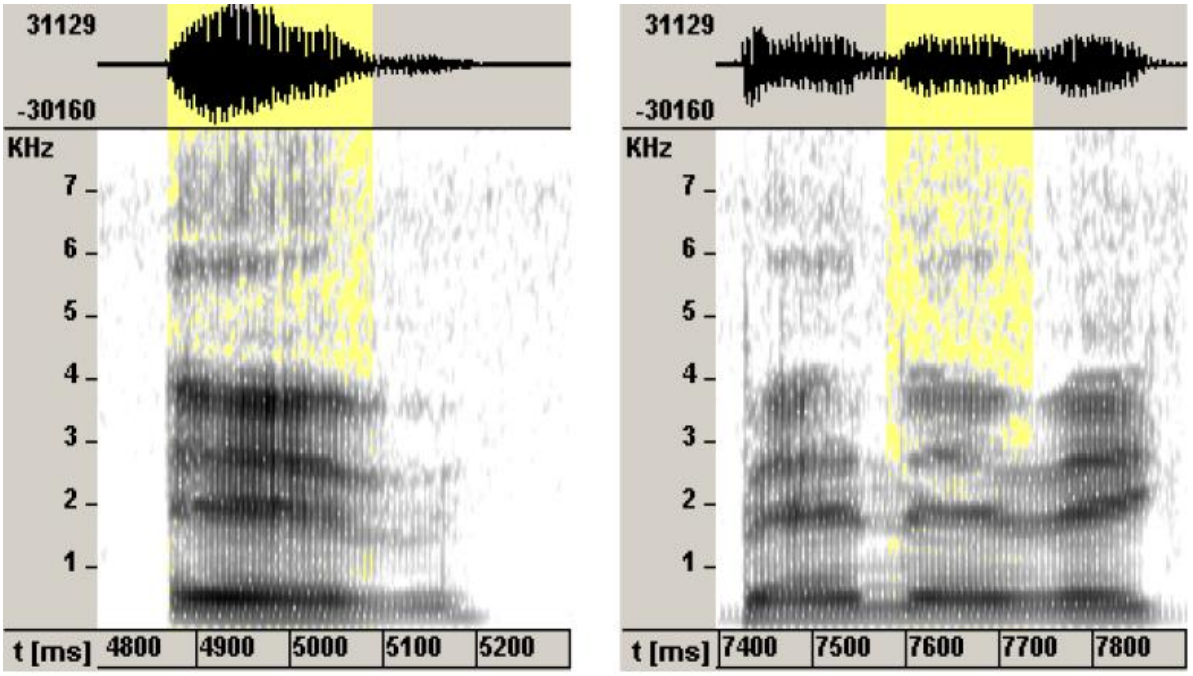

FIGURE 2.2: The waveform and spectrogram of ev on the left and Ben eve on the right.

shows the amplitude. The effect of the context can be seen on the characteristics of the /e/ sound. The effect of the context leads us to model each phoneme according to the neighboring phonemes.

Spontaneous speech may contain other fillers that are not words like ee or hImm. It is another difficulty in continuous speech recognition. The task should be known while designing the algorithm. If we are to use a recognizer in continuous speech recognition, the training data should consist of

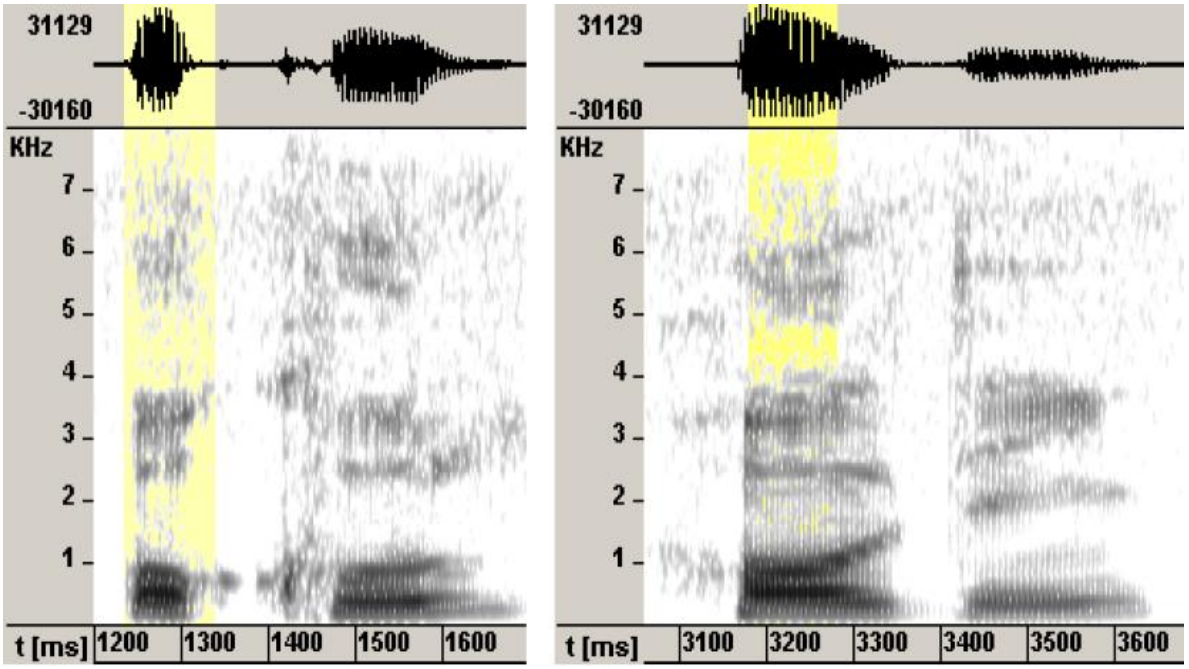

FIGURE 2.3: The waveform and spectrogram of okul on the left and holding on the right

continuousSpeech as well. The main difficulty of the speech recognition problem comes from the variability of the source of the signal. First,the characteristics of phonemes, the smallest sound units, are dependent on the context in which they appear.

The environment also causes variability. The same speaker will say a word differently according to his physical and emotional state, speaking rate, or voice quality. The difference in the vocal tract size and shape of different people also causes

variability. The problem is to find the meaningful information in the speech signal. The meaningful information is different for speech recognition and speaker recognition. The same Information in the speech signal may be necessary for some application and redundant for some other application. For speaker independent speech recognition we have to get rid of As much speaker related features as possible.

2.1.4 Types of Speech Recognition

Speech recognition systems can be separated in several different classes by describing what types of utterances they have the ability to recognize. These classes are based on the fact that one of the difficulties of ASR is the ability to determine when a speaker starts and finishes an utterance. Most packages can fit into more than one class, depending on which mode they're using.

- Isolated Words Isolated word recognizers usually require each utterance to have quiet (lack of an audio signal) on BOTH sides of the sample window. It doesn't mean that it accepts single words, but does require a single utterance at a time. Often, these systems have "Listen/Not?Listen" states, where they require the speaker to wait between

utterances (usually doing processing during the pauses). Isolated Utterance might be a better name for this class.

- Connected Words Connect word systems (or more correctly 'connected utterances') are similar to Isolated words, but allow separate utterances to be 'run?together' with a minimal pause between them.
- Continuous Speech Continuous recognition is the next step. Recognizers with continuous speech capabilities are some of the most difficult to create because they must utilize special methods to determine utterance boundaries. Continuous speech recognizers allow users to speak almost naturally, while the computer determines the content. Basically, it's computer dictation.
- Spontaneous Speech There appears to be a variety of definitions for what spontaneous speech actually is. At a basic level, it can be thought of as speech that is natural sounding and not rehearsed. An ASR system with spontaneous speech ability should be able to handle a variety of natural speech features such as words being run together, "ums" and "ahs", and even slight stutters.
- Voice Verification/Identification Some ASR systems have the ability to identify specific users. This document doesn't cover verification or security systems.

2.1.4.1 Dependence and Independence in Speech Recognition

Speech recognition may be speaker dependent or speaker independent. If the application is for home use, and the same person will use the same microphone at the same place, then the problem is simple and you dont need a robust algorithm. But if it is an application that will recognize speech over a public telephone network where speaker variability and the environment that speech passes through are different among different calls, you need a robust algorithm. If recognition of isolated words or phrases is the problem, then you will have less of a problem as far as the speakers only give the required input. If the speakers also use other words in addition to the keywords you require, you need to perform keyword Spotting which means recognizing the keywords among other non-keyword filler words. If we go further, recognition from a large vocabulary where you have to recognize all of the words, it is called dictation, which is a harder task.

2.1.5 Acoustic Modelling and Training

Modelling of the acoustic data is generally performed in a statictical framework. An inventory of elemantary probabilistic models of basic linguistic units is used to build word representations. Acoustic Models (AM) are stochastic models

used with language models to find the correct transcription for the given acoustic data.

There are mainly three methods used in acoustic modelling:

- Hidden Markov Models(HMM)
- Dynamic Time Warping(DTW)
- Artificial Neural Networks(ANN)

The most eficient and widely used method for the past decade is HMM method and also in this work HMMs are used for acoustic modelling.

2.1.5.1 Hidden Markov Models

The Hidden Markov Model Toolkit (HTK) is a portable toolkit for building and manipulating hidden Markov models. HTK is primarily used for speech recognition research although it has been used for numerous other applications including research into speech synthesis, character recognition and DNA sequencing. HTK is in use at hundreds of sites worldwide HTK consists of a set of library modules and tools available in C source form. The tools provide sophisticated facilities for speech analysis, HMM training, testing and results analysis. The software supports HMMs using both continuous density

mixture Gaussians and discrete distributions and can be used to build complex HMM systems.

Speech recognition systems generally assume that the speech signal is a realisation of some message encoded as a sequence of one or more symbols. To effect the reverse operation of recognising the underlying symbol sequence given a spoken utterance, the continuous speech waveform is first converted to a sequence of equally spaced discrete parameter vectors. This sequence of parameter vectors is assumed to form an exact representation of the speech waveform on the basis that for the duration covered by a single vector, the speech waveform can be regarded as being stationary. Although this is not strictly true, it is a reasonable approximation. Typical parametric representations in common use are smoothed spectra or linear prediction coefficients plus various other representations derived from these.

The role of the recogniser is to effect a mapping between sequences of speech vectors and the wanted underlying symbol sequences. Two problems make this very difficult. Firstly, the mapping from symbols to speech is not one-to-one since different underlying symbols can give rise to similar speech sounds. Furthermore, there are large variations in the realised

speech waveform due to speaker variability, mood, environment, etc. Secondly, the boundaries between symbols cannot be identified explicitly from the speech waveform. Hence, it is not possible to treat the speech waveform as a sequence of concatenated static patterns.

The second problem of not knowing the word boundary locations can be avoided by restricting the task to isolated word recognition. Despite the fact that this simpler problem is somewhat artificial, it nevertheless has a wide range of practical applications. Furthermore, it serves as a good basis for introducing the basic ideas of HMM-based recognition before dealing with the more complex continuous speech case. Hence, isolated word recognition using HMMs will be dealt with first[1].

The Hidden Markov Model (HMM) is a powerful statistical tool for modeling generative sequences that can be characterized by an underlying process generating an observable sequence. HMMs have found application in many areas interested in signal processing, and in particular speech processing.

Practically all ASR systems in use are based on hidden Markov

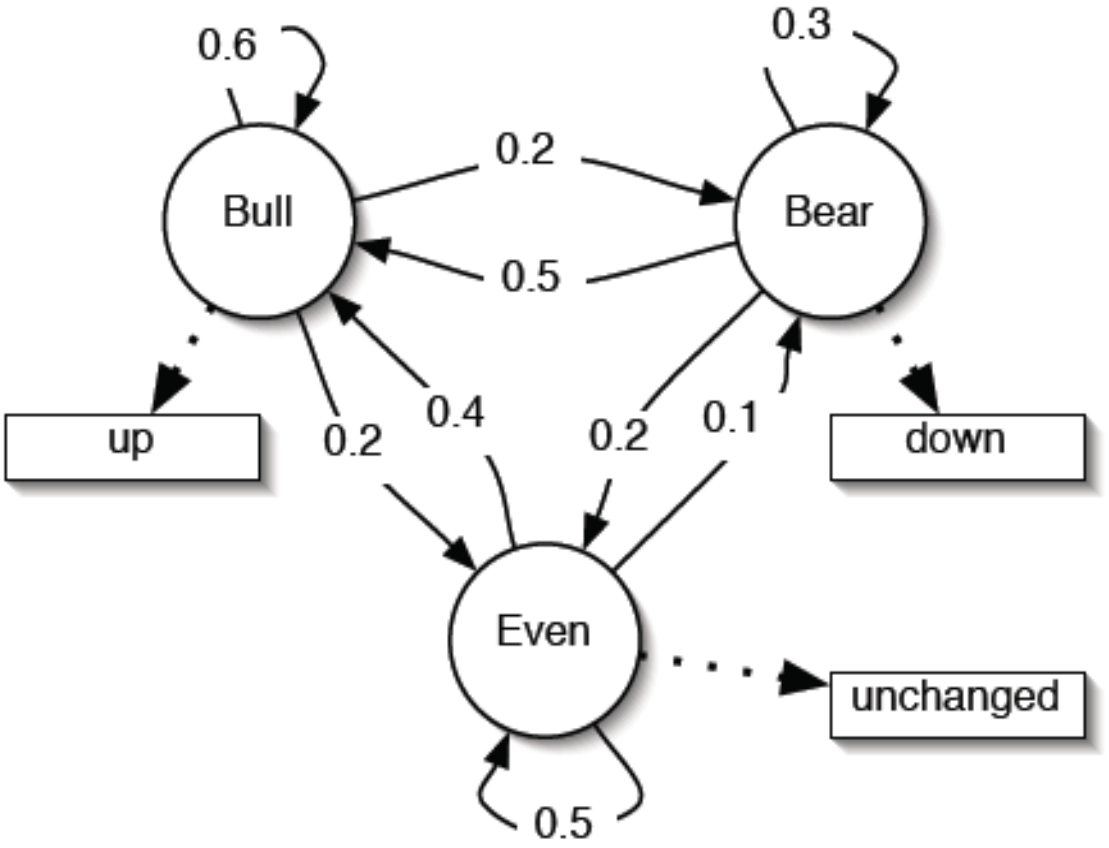

FIGURE 2.4: Markov Process Example

models HMMs. A hidden Markov model is a statistical representation of a speech event like a word, model Parameters are typically trained on a large corpus of labeled speech data. This approach has proved successful not Only for large-vocabulary recognition systems, but for keyword-spotting systems.

The Baum-Welch Algorithm

To determine the parameters of a HMM it is first necessary to make a rough guess at what they might be. Once this is

done, more accurate parameters can be found by applying the so-called Baum-Welch re-estimation formulae.

The algorithm has two steps:

1. Calculating the forward probability and the backward probability for each HMM state;

2. On the basis of this, determining the frequency of the transition-emission pair values and dividing it by the probability of the entire string. This amounts to calculating the expected count of the particular transition-emission pair. Each time a particular transition is found, the value of the quotient of the transition divided by the probability of the entire string goes up, and this value can then be made the new value of the transition.

2.1.5.2 Artificial Neural Nets

Neural Networks can be seen as a succesful modelling technique since they produce accurate recognition results with some of the most diffcult phoneme identifications, such as stop consonant pairs, nasals and sonarants and fricatives. The best performance has been obtained from Recursive or Time-Delay network approaches, which are well suited to ignoring the temporal variability that is undesirable in speech,

while capturing the temporal features that mark prosodic distinctions within words. Although neural network algorithms provides significant progress in phoneme-based recognition, they need further improvement to be as accurate as HMMs in modelling of speech signals. In ANN approach there exist several layers and each layer includes several nodes (Each layer can have different number of nodes). The coded speech enters the network and the network is updated synchronuously at some time intervals, receiving new inputs and propagating information through the rest of the network. The output nodes represent output status, or response, which signals that a particular event has been recognized. Between the input and output nodes there are one or more hidden layers of nodes. Connections between nodes may go forward towards output nodes, sideways within hidden layers, or backwards, and can be recursive or have delays built in.

2.1.5.3 Dynamic Time-Warping

The two speech signal of an utterance produced by the same speaker can be quite different. The length of the utterance can distort in time non-linearly. DTW is used to derive the overall distortion between two speech templates. In these template-based systems, each speech template consists of a sequence of speech vectors. The overall distortion measure is

computed from the accumulated distance between two feature vectors that are aligned between two speech templates with minimal overall distortion. Indeed DTW is a special kind of Dynamic Programing Technique. Let's consider two array in time domain; $A = a_1, a_2 \cdots a_M$ and $B = b_1, b_2, \ldots b_N$. If we assume that the start time is same for both array, $a_1 = b_1$, then the aim is to overlap the end times, $a_M and b_N$. To make the and times equal a funcion is defined by using dynamic programming techniques. And this function is applied on the sound data iteratively untill the boundaries are equal.

2.2 Keyword Spotting

Keyword (or word) spotting refers to a proper detecting of any occurrence of a given word in a speech signal. The aim of keyword spotting system is to detect a small set of keywords from a continuous speech. It is important to obtain the highest possible keyword detection rate without increasing the number of false insertions in this system. Modeling only keywords is not enough. To separate keywords from non-keywords, models for out-of-vocabulary words are needed, too. Since the structure and type of garbage model has great effect on the entire system performance, out-of-vocabulary modeling is done by the use of garbage (or filler) models.

Keyword spotting becomes a very important branch of speech recognition. Latest research experiences show that it is nearly impossible to design a recognizer that covers all words uttered during the practical application.

Keyword spotting can be regarded as a sub-field of automatic speech recognition (ASR), as it uses the same mathematical concepts and techniques. The difference is that while ASR systems provide a transcription hypothesis for the whole utterance1, KWS systems output a binary decision given a keyword, and (optionally) the begin and end points of the

keyword segment in the utterance (which is called a segmentation). KWS is most useful in domains where a full transcription is not required and where conventional ASR systems are either unavailable or introduce an unnecessary complexity to the application[7].

When spotting keywords in unconstrained speech, the keyword to be recognised is assumed to be embedded in extraneous speech, and may begin and end at any time frame of the utterance, making keyword spotting a non-trivial task. There are two basic approaches to tackle this problem (without relying on transcription hypotheses from ASR systems), the filler model method and the sliding model method. Every keyword has its unique model in both methods, but they differ in the way they treat non-keyword speech segments[2].

The filler/garbage model method aims to model the whole utterance by explicitly modeling non-keyword segments using one or several filler models in addition to the keyword models.

The sliding model method only models keyword segments, and calculates the likelihood of a keyword occurrence for every possible segments, resulting in a more robust, but complex solution. Most keyword spotting algorithms used today are based on one of these two basic approaches.

2.2.1 Keyword Spotting Problem

In the keyword spotting problem a continuously spoken utterance is tested for the existence of a keyword. The speech signal will contain any combination of silences, keywords and non-keywords. The words that are not keywords are called garbage or out of-vocabulary words. We need to model the out-of-vocabulary words in some form.

2.2.2 Filler Models

To compute confidence measures for accepted commands, we used another model - a filler model - to make a new likelihood comparison. The filler model is trained with all database utterances, commands and phrases, in order to represent a generic speech segment. If the command and filler likelihoods are almost the same, then there is a low confidence on the result. On the contrary, if the likelihoods are very different, the confidence on the recognition result should be high.

The design philosophy of filler models should be consistent with their common purpose, filling the holes of the grammar in the normal speech recognizer. In order to do this the acoustic world is modelled in some units and these unit models are connected to each other within a loop structure (null-grammar). Thus the recognition process is made free

from any effect of constrained unit networks like, grammar or language model. In other words, filler models output the best unconstrained acoustic path from the unit networks. In this thesis we have defined filler models consisting of units in difierent details[3].

In general, the confidence of an utterance is determined by a comparison in between the best path from the acoustic unit network and the best path from the regular decoder. The desired ratio between the two should be equal to one or greater than one. This means that the filler model result is more trustworthy than that of the regular decoder. This is because filler model matches all units belonging to the observation with models in the loop without any dependency. However the normal decoder makes its matching by considering the connectivity of the trained models. Even if the observation sequence is not in the grammar or in the LM, according to the dependency of the models it generates an output in any way. For example let's assume that the word work is not included in our grammar or LM but the word verb is included. Also let's design our filler model as all-phone network, in which all phones are connected to each other without considering any dependency. For the acoustic observation of the word work, the normal decoder will give a transcription such as "verb". However in filler model decoding each phone is matched with

a phoneme model and the concatenation of these best models are returned as "work" even if the work is not in the grammar/LM. Then it can be concluded that filler models are well on detecting the out of vocabulary errors. Also in grammar-based cases, they can successfully detect the out of grammar sentences or divisions[4].

2.3 Audio Content Search

2.3.1 Overview

Text search remains dominant while at the same time the generation of multimedia content is growing at a fast pace. When someone uses a search engine to find information about a subject of interest, after entering a set of keywords, is presented with a list of potentially relevant sources, mainly in the form of web pages, documents or images. Yet until recently, search engines were unable to process information preserved in the form of speech interviews, lectures, voicemails and radio newscasts even though this medium allows easy and affordable storage of millions of hours of human knowledge and interactions. Spoken media content combines information from multiple levels (phonetic, acoustic, syntactic, semantic and discourse) and is far richer than what a simple textual transcription can capture. Current approaches

in searching and browsing speech, image and video archives are expensive and time consuming creating incentives to extend search capabilities into these media in a similar way to text. The focus of this contribution is on the field of spoken media[5].

Automatic systems for indexing, archiving, searching and browsing of large amounts of spoken communications have become a reality in the last decade. Most such systems use an automatic speech recognition (ASR) component to convert speech to text which is then used as an input to a standard text based information retrieval (IR) component. This strategy works reasonably well when speech recognition output is mostly correct or the documents are long enough so that some occurrences of the query terms are recognized correctly. Most of the research has concentrated on retrieval of Broadcast News type of spoken documents where speech is relatively clean and the documents are relatively long. In addition it is possible to find large amounts of text with similar content in order to build better language models and enhance retrieval through use of similar documents. We are interested in extending this to telephone conversations and teleconferences. Our task is locating occurrences of a query in spoken communications to aid browsing. This is not exactly spoken

document retrieval. In fact, it is more similar to word spotting. Each document is a short segment of audio. Although reasonable retrieval performance can be obtained using the best ASR hypothesis for tasks with moderate ($\tilde{2}0\%$) word error rates, tasks with higher (40-50%) word error rates require use of multiple ASR hypotheses. Use of ASR lattices makes the system more robust to recognition errors. Almost all ASR systems have a closed vocabulary. This restriction comes from run-time requirements as well as the finite amount of data used for training the language models of the ASR systems. Typically the recognition vocabulary is taken to be the words appearing in the language model training corpus. Sometimes the vocabulary is further reduced to only include the most frequent words in the corpus. The words that are not in this closed vocabulary the out of vocabulary (OOV) words will not be recognized by the ASR system, contributing to recognition errors. The effects of OOV words in spoken document retrieval are discussed by Woodland et al. (2000). Using phonetic search helps retrieve OOV words[6].

2.3.2 Architecture for Spoken Media Search

A generic architecture of Spoken Media Search systems is as follows (See Fig. 2.5):

Speech recognition systems can label automatic transcriptions with exact time stamps, their output can be viewed as a form of notation with which the other tasks can synchronise.

Topic segmentation/tracking and speaker detection/tracking are used as a basis for indexing relevant audio segments according to topic or speakers, respectively.

Specific information can also be extracted automatically from the transcriptions to facilitate more detailed analysis. In the retrieval phase, the focus is on selecting which terms from the text and annotations to compare, how they should be weighted, and how to compare the sets of weighted terms. An advantage of re-usable annotations is that they can be used to infer additional semantic relations.

Retrieval can also be facilitated by classifying content into categories. This not only simplifies the retrieval process itself but can also assist users to better understand and remember information as it is presented in the appropriate context.

The last and perhaps the least explored phase deals with the delivery of the retrieved content to users. Summarisation is a promising method to overcome the problems associated with information overload by presenting condensed versions of the content. User interfaces typically supports queries expressed in natural language or with Boolean expressions. Adaptive

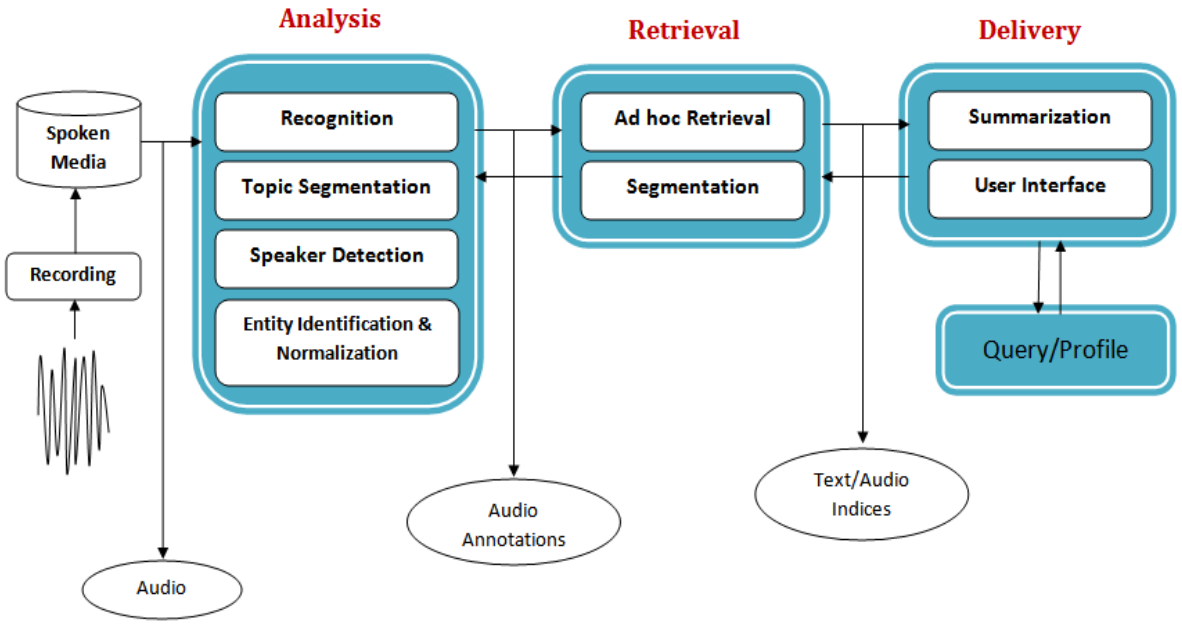

FIGURE 2.5: Generic Architecture for Spoken Media Search Systems

profiles that tend to reflect long term information needs can also be used to replace repeated queries and filter our irrelevant information[5].

2.3.3 Applications of Audio Content Search

Individual

- Personalized delivery of voicemail and newscasts
- Search in audio books and music (incl. similarity search)

- Search in personal audio recordings (meetings, presentations, telephone conversations)

Corporate

- Monitoring of television and radio news
- Retrieval of help desk calls
- Content management of corporate meetings and conference calls

Government

- Access to audio proceedings (parliamentary sessions, court of law archives)
- Access to cultural heritage archives
- Monitoring of unlawful conversations for security purposes

Education

- Access to audio visual material for training and research purposes

Chapter 3

Analysis and Design

- Overview
- Use Cases
- Sequence DiagramsActivity Diagrams
- Activity Diagrams
- Class Diagrams

3.1 Overview

Our System has three modules (See Fig. 3.1):

- Analysis

  Speech recognition systems can label automatic transcriptions with exact time stamps, their output can be viewed as a form of notation with which the other tasks can synchronise.

- Retrieval

  In the retrieval phase, the focus is on selecting which terms from the text and annotations to compare, how they should be weighted, and how to compare the sets of weighted terms.

- Delivery

  User interfaces typically supports queries expressed in natural language.

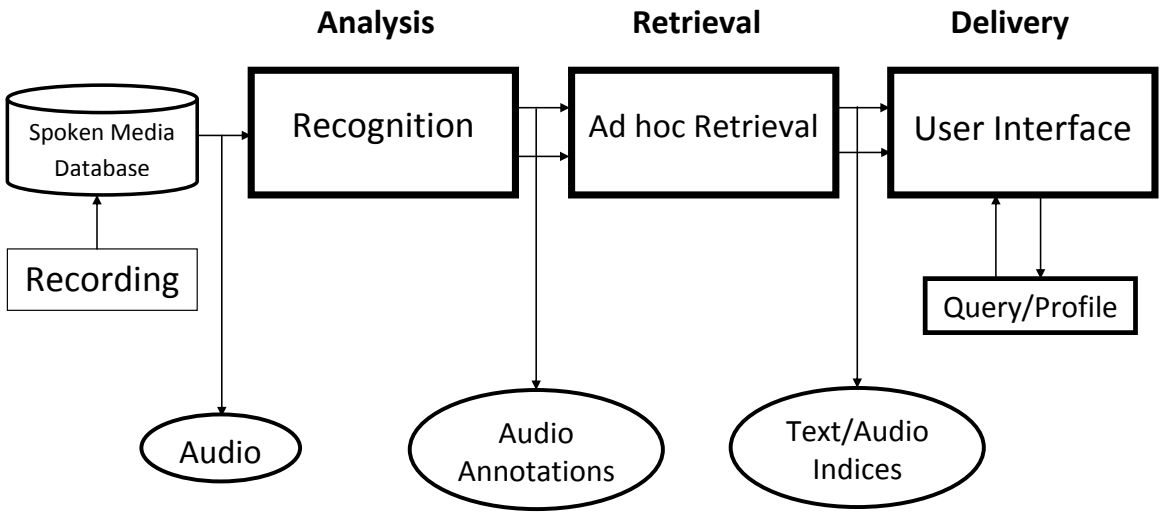

FIGURE 3.1: Overall System Architecture

3.2 Use Cases

```
Name: Register
Identifier: UC01
Actor: Visitor
Description:
Register into the website to have a member account.
Preconditions:
The user must not already have an account.
The account username should be unique.
Post conditions:
The user becomes a member and is treated as a member by the system.
Basic Course of Action:
The user inputs his/her username, desired password, real name, age, country
    .
An account is created for the user.
Alternative Course A:
1. The user already registered
2. The user is informed that the he s already registered or the username s/
    he has chosen is already taken.
3. The user can refill the registration form once again.
```

```
Name: Login
Identifier: UC02
Actor: Visitor
Description:
Log into the website.
Preconditions:
The visitor has to be a registered member.
Post conditions:
Basic Course of Action:
1. The user presses on the Login button.
2. The user enters his authentication information.
3. The system checks the validity of this information.
4. If the information is valid, the visitor become logged in and is treated
     as a member by the system.
Alternative Course A: Authentication information not valid
1. The system rejects the request to log in.
2. The user can input his/her information once again.
Alternative Course B: The user forgotten his/her password
1. The user presses the Forgot your Password button.
2. The user inputs his e-mail.
3. The system checks if there s an account corresponding to this e-mail.
4. The system sends the username and password of the corresponding account
    to the inputted e-mail.
```

```
Name: Logout
```

```
Identifier: UC03
Actor: Member
Description: Logout of the website.
Preconditions: The user has to be logged in.
Post conditions: The user is treated by the system as a visitor not a
    member.
Basic Course of Action:
1. The user presses on Logout button.
2. The user becomes no longer logged on and is not treated as a member
    anymore.
```

```
Name: Add Audio Files
Identifier: UC04
Actor: Member
Description: A member can add Audio Files to his/her account, these audio
    files become searchable by the system.
Preconditions: The user must be logged in.
Post conditions: The files are Transcript and indexed by the system.
Basic Course of Action:
1. The user pressed on the Add File button.
2. The user browses and chooses the Audio File to be uploaded.
3. The system checks the format and length of Audio Files.
```

```
Name: Remove Audio File
Identifier: UC05
Actor: Member
Description:  The user can remove an Audio File thats added by him.
Preconditions:  The user can only remove Audio Files uploaded by him/her.
Post conditions: The Audio File is removed from the users account.
The Audio File and its transcript are removed from the index.
Basic Course of Action:
1. User selects the file to be edited and then press the Delete File button
    .
```

```
Name: Delete Account
Identifier: UC06
Actor: Member
Description: The user deletes his/he account
Preconditions: The user must be logged in.
Post conditions:
The user account is deleted.
The users Audio Files and their transcripts are removed from the index.
Basic Course of Action:
1. The user his/her account page.
2. The user presses the Delete button.
The system removes the users Audio Files and their transcripts from the
    index.
```

```
Name: Search Audio Files
Identifier: UC07
Actor: Visitor/Member
Description: The user enters a word and the audio files containing that
    word are retrieved.
Preconditions: N/A
Post conditions: N/A
Basic Course of Action:
1. The user enters a keyword in the Search field.
2. The system searches the index for Audio Files containing this keyword in
     their transcript.
These Audio Files are retrieved to the user with a flag to where the word
    was found.
```

3.3 Sequence Diagrams

A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. 1- Sequence diagram for registration User enters his/her data in the form; system checks the database for the entered user name if not used before then the system create new account to the user, if used before or there is an error in the entered data system return error message and redisplay the registration form. 2- Sequence diagram for Upload Audio File User browses the audio file and enters description about it then enter upload; system checks the type of the uploaded audio

if it is correct then saved in the server and save the information in the database. After specific time the system start to recognize the file and convert then update the index. 3- Sequence diagram for Search User enters the word to search; system checks to be from the vocabulary if so then retrieves all files that contain the keyword. 4- Sequence diagram for Login User enters username and password; system checks for correct username and password if correct return his/her account.

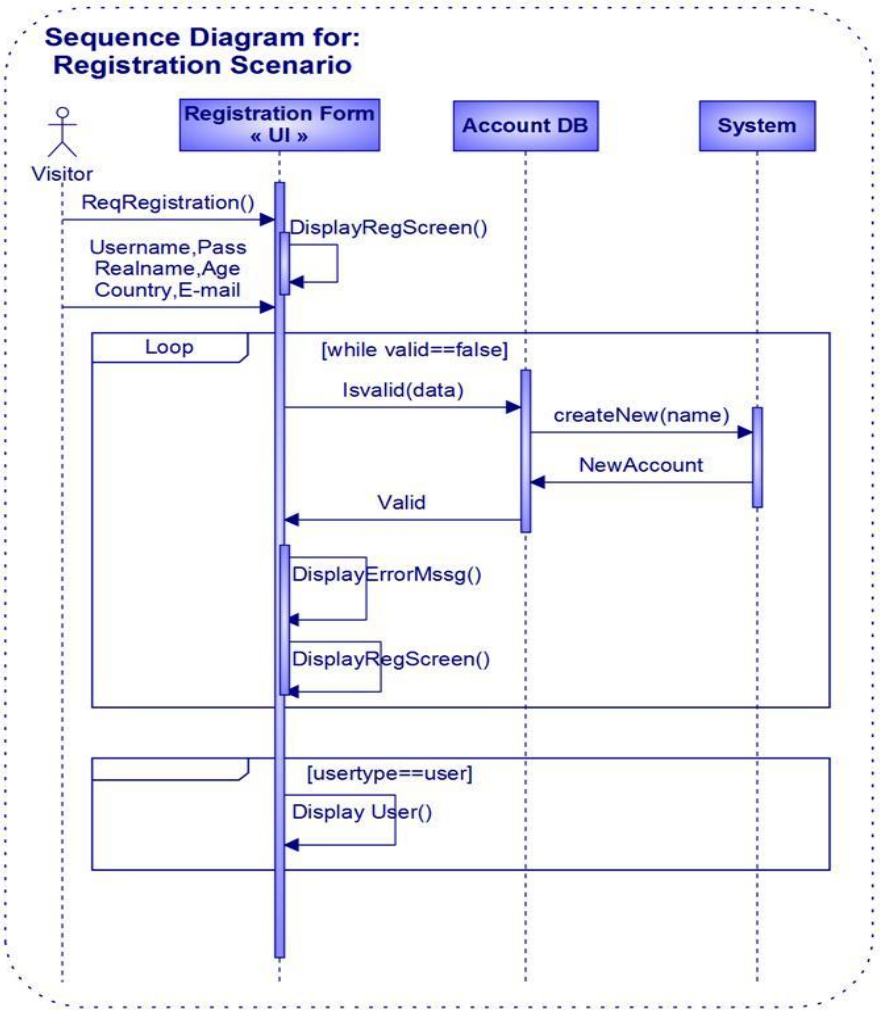

FIGURE 3.2: Sequence Diagram for Registration Scenario

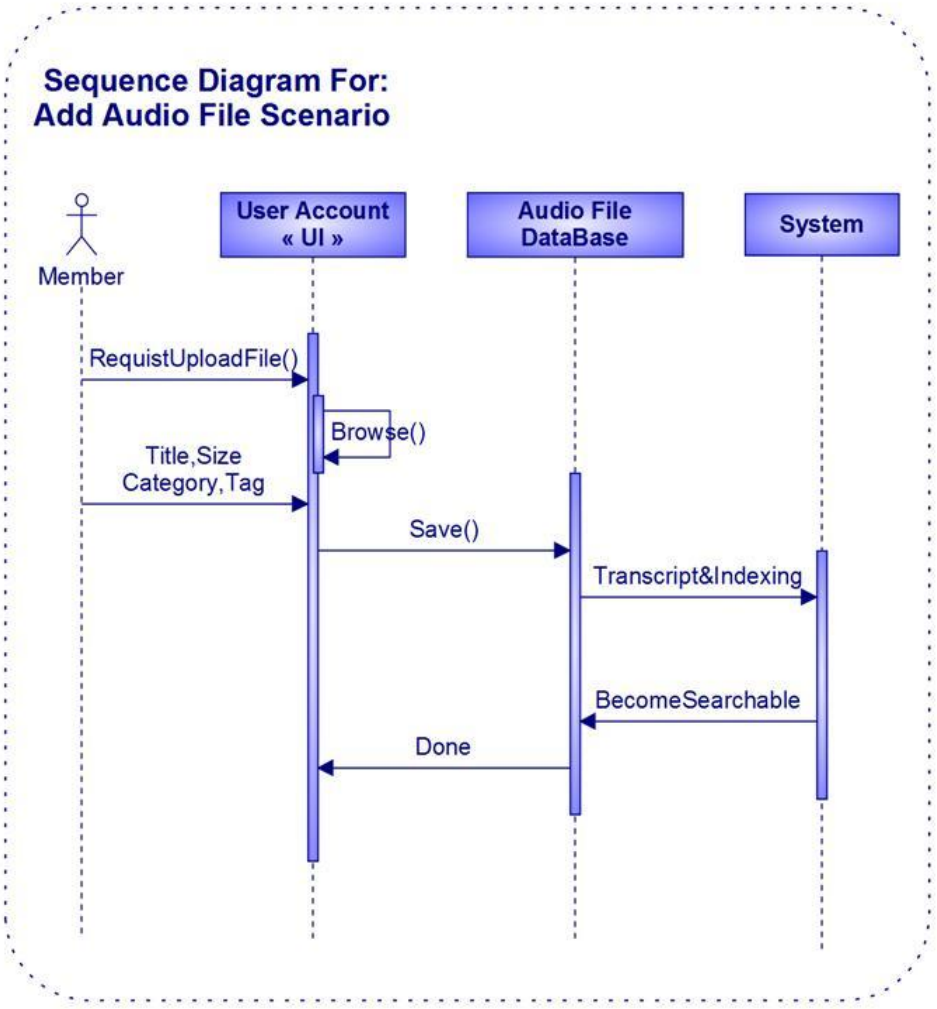

FIGURE 3.3: Sequence Diagram for Add Audio Scenario

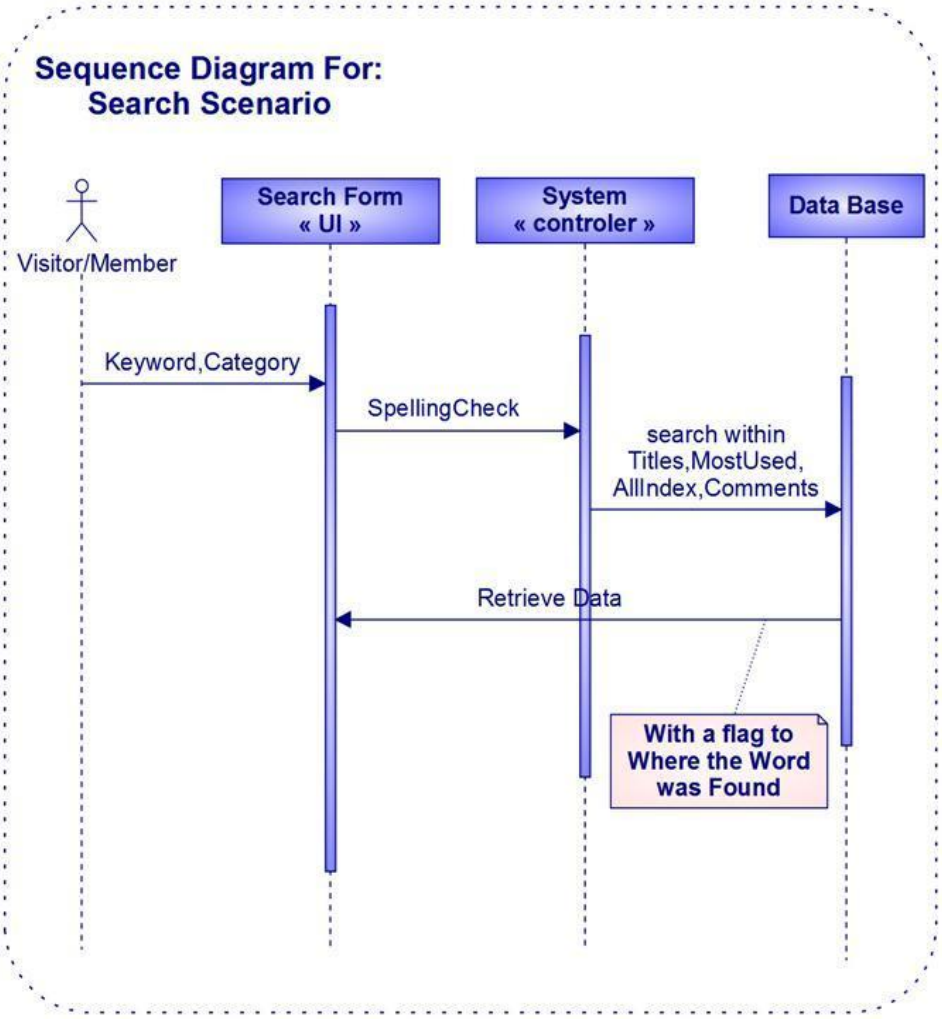

FIGURE 3.4: Sequence Diagram for Search Scenario

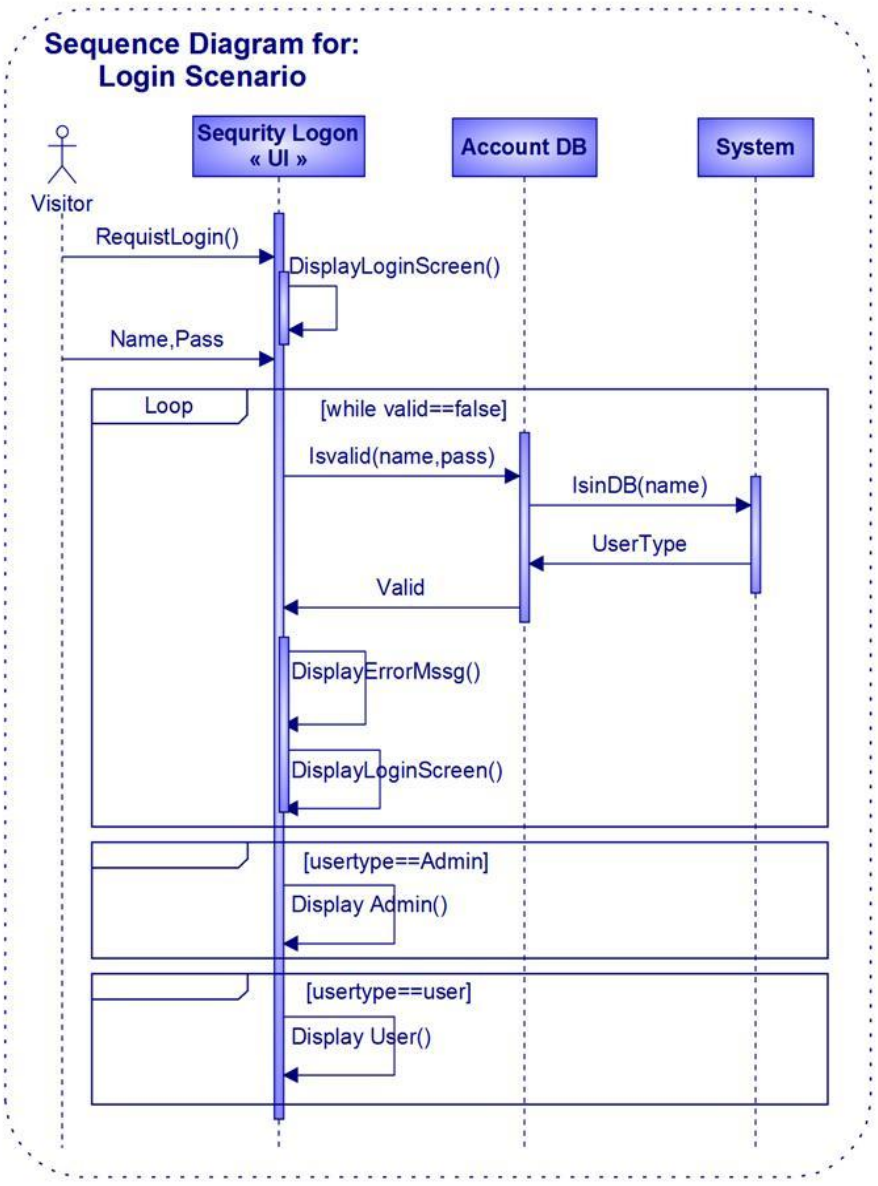

FIGURE 3.5: Sequence Diagram for Login Scenario

3.4 Activity Diagrams

Activity diagrams are a loosely defined diagram technique for showing workflows of stepwise activities and actions, with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

1 Activity diagram for login

a- Enter username and password

b- System check

c- If valid then manage audio files and his account

d- Else enter username and password again

2 Activity diagram for manage audio files

a- Sign in

b- Choose Audio file

c- Edit Audio file Name

d- Delete Audio file

e- Add new one

f- Save changes

3 Activity diagram for register

a- Enter his/her data

b- System check

c- Return new account

d- Manage audio files and manage account

4 Activity diagram for search

a- Write keyword

b- System check for priorities

c- Retrieve data

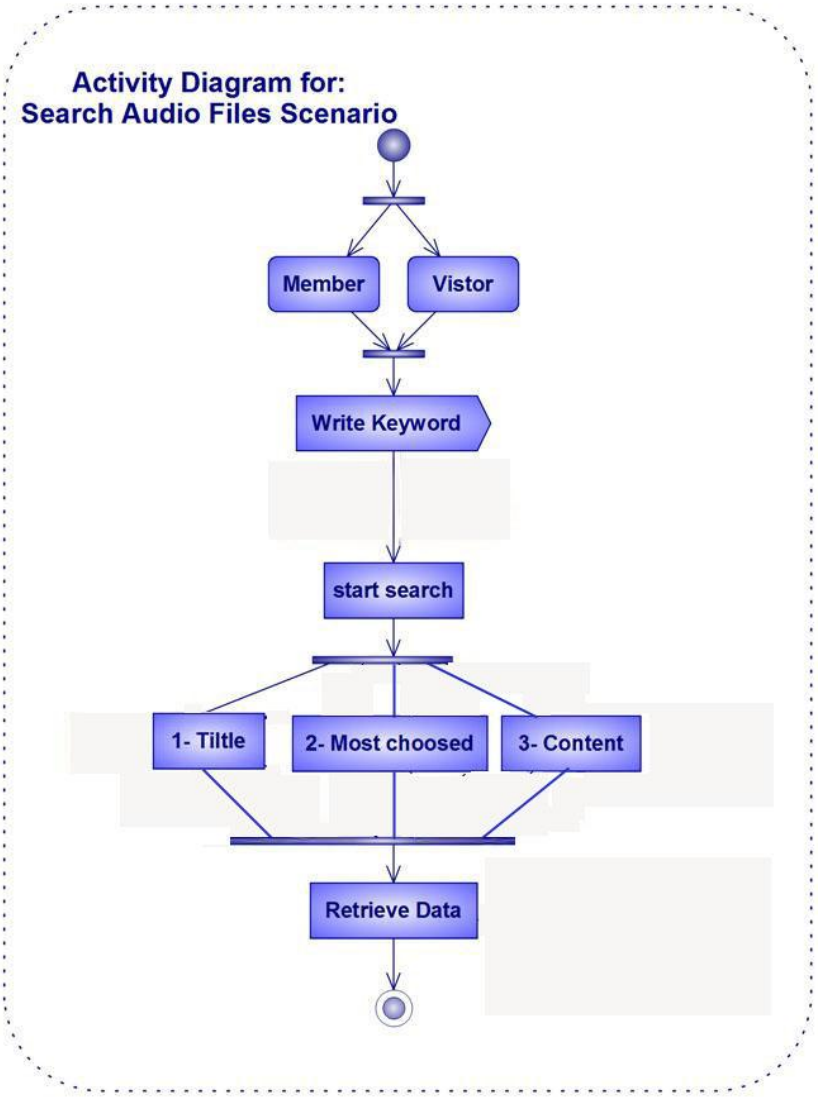

FIGURE 3.6: Activity Diagram for Search Audio Files Scenario

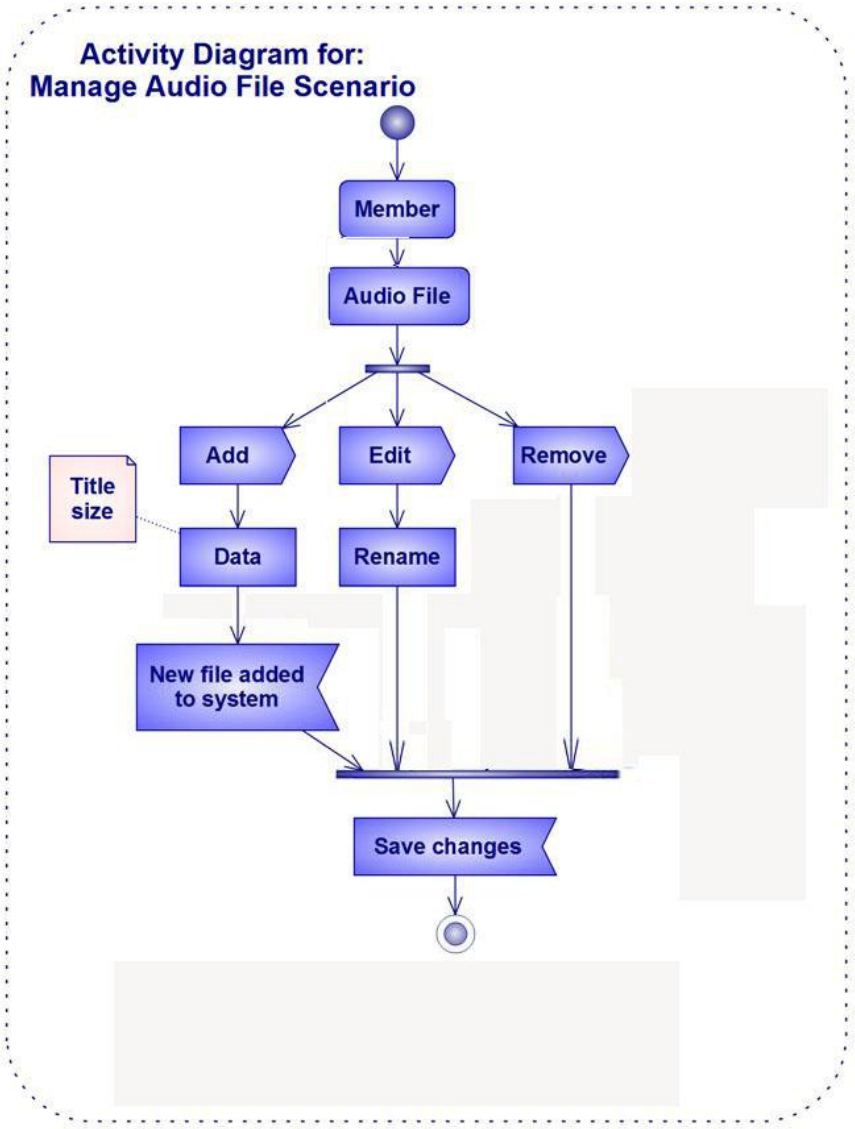

FIGURE 3.7: Activity Diagram for Manage Audio Files Scenario

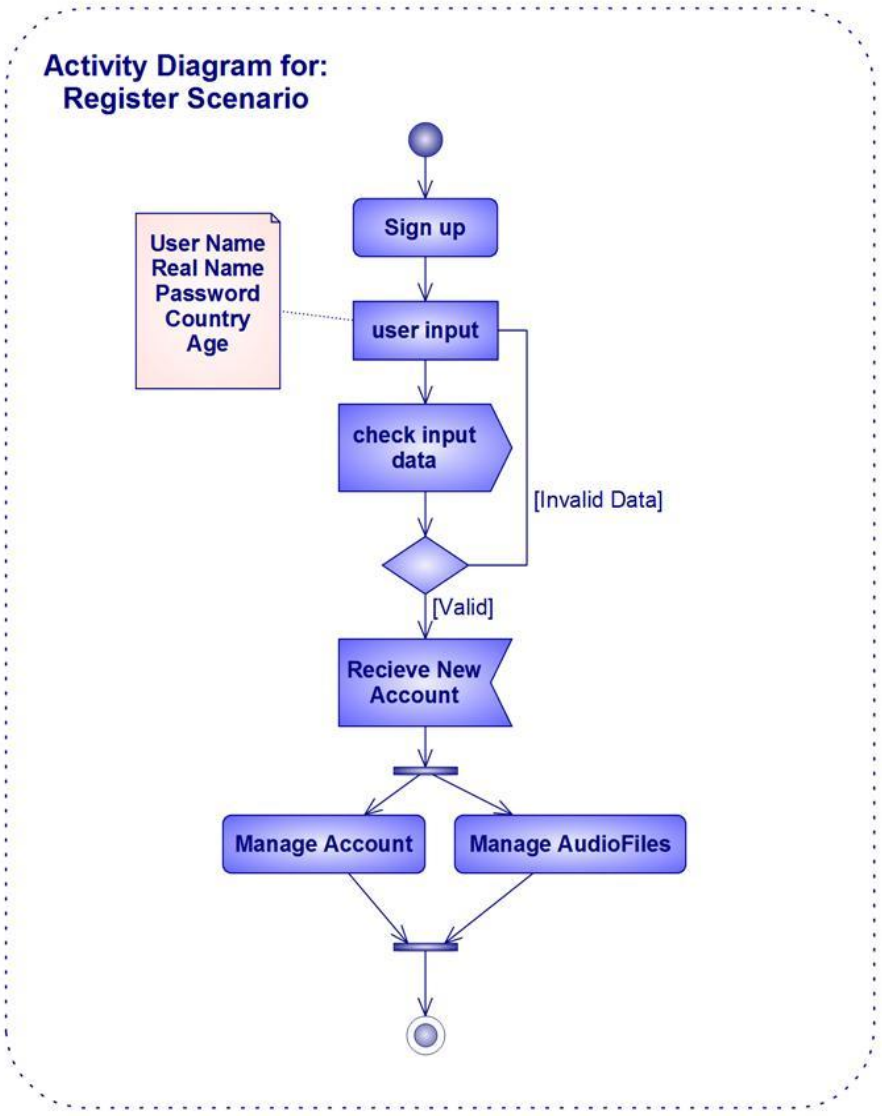

FIGURE 3.8: Activity Diagram for Register Scenario

3.5 Class Diagram

Class diagram in the Unified Modeling Language (UML), is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

1 Login

a- Check login data function.

b- User Account

2 Register

a- Check unique username

b- Create new account

3 Search

a- Check the keyword from vocabulary

b- Search in file names

c- Search in the content

d- Retrieve data

4 Profile

a- Edit profile

b- Delete account

c- Add/Edit/Delete Audio files

5 Database

Contain all information about users and audio files

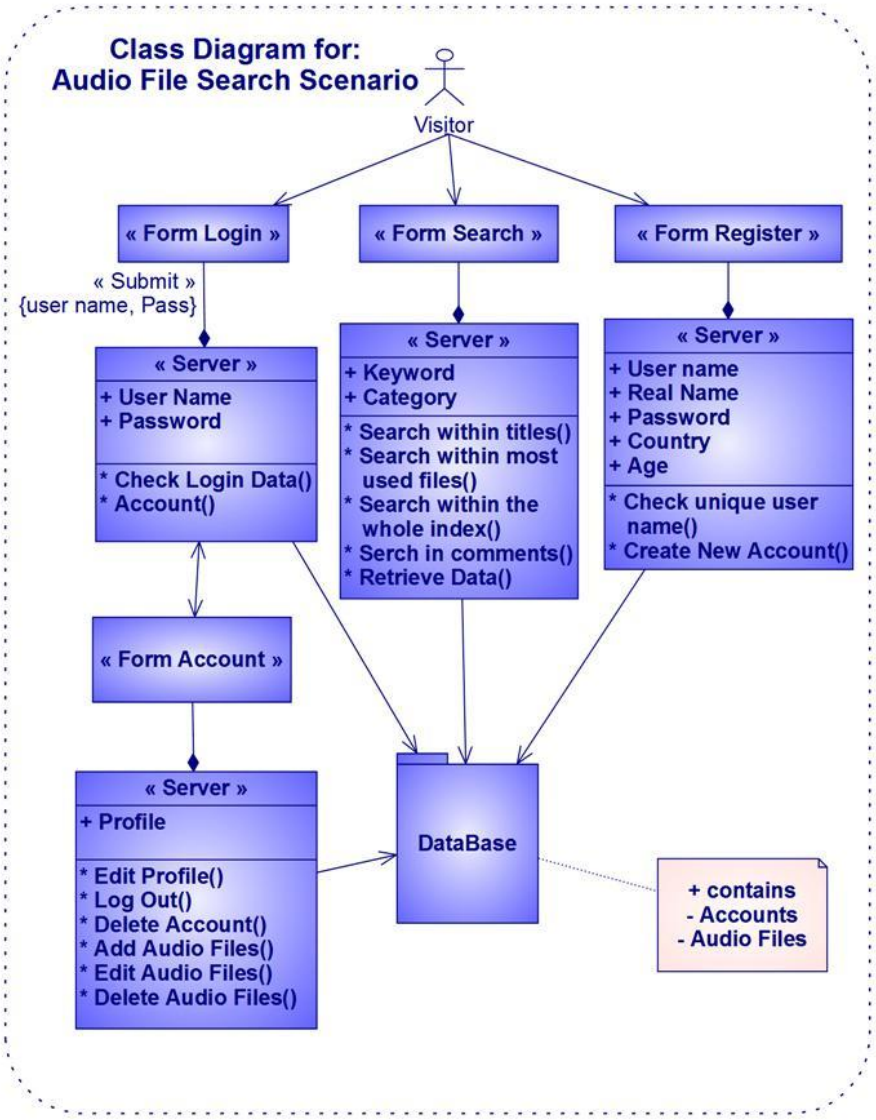

FIGURE 3.9: Class Diagram

Chapter 4

Implementation

- Analysis
  - HMM Training Phase
  - Grammar Specification Phase
  - Recognition Phase
- Retrieval
- Delivery
- Technology Used
  - PHP
  - MYSQL
  - Why PHP and MySQL?
  - Hidden Markov Model Toolkit (HTK)

4.1 Analysis

The Analysis module is the one that performs the Speech Recognition operation.

We've chosen **Keyword Spotting** as an approach for implementing the ASR system. It becomes a very important branch of Speech Recognition. Latest research experiences show that it is nearly impossible to design a recognizer that covers all words uttered during the practical application.

In the keyword spotting problem a continuously spoken utterance is tested for the existence of a keyword. The speech signal will contain any combination of silences, keywords and non-keywords. The words that are not keywords are called garbage or out-of-vocabulary words.

We've chosen the **filler-based** approach to implement the KWS system The advantage of this approach is that it preserves the simplicity of the Viterbi algorithm which is the dynamic programming algorithm for finding the most likely sequence of hidden states called the Viterbi path.

The implementation of the analysis module consisted of three successive phases:

1 HMM Training Phase

2 Grammar Specification Phase

3 Recognition Phase

4.1.1 HMM Training Phase

For the training phase, we've built a speech corpus. This database contains a collection of sentences. Acoustic-phonetic labelling involves segmenting the speech signal in terms of phonetic characteristics. The set of labels associated with a speech signal file constitutes a transcription and each transcription is stored in a separate label file. We have a set of 75 isolated word recordings, 9 carrier phrases recordings, 83 phonetically rich phrases as a training dataset. The raw speech waveform for training is parameterized into speech vectors. We have used a reduced set of 45 phones and build a HMM for each phone; the 45 HMMs include a silence and pause model. Together with a prototype HMM definition, the speech vectors and labels are fed into a training phase to estimate the parameters for the HMMs. The algorithms used in the training phase are the Baum- Welch re-estimation and the forward-backward algorithm. The output of this training

phase is the required set of HMM phone models used for the word spotting exercise.

The following is the initial prototype of each HMM which is then re-estimated:

```
~o <VecSize> 25 <MFCC_0_D_N_Z>
~h "proto"
<BeginHMM>
  <NumStates> 5
  <State> 2
    <Mean> 25
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    <Variance> 25
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
 <State> 3
    <Mean> 25
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    <Variance> 25
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
 <State> 4
    <Mean> 25
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    <Variance> 25
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
 <TransP> 5
  0.0 1.0 0.0 0.0 0.0
  0.0 0.6 0.4 0.0 0.0
  0.0 0.0 0.6 0.4 0.0
  0.0 0.0 0.0 0.7 0.3
  0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

4.1.2 Grammar Specification Phase

The keywords are mapped to a standard phonetic grammar representation string using a standard phonetic dictionary. The grammar for the garbage patterns is also constructed. Both the grammars for the phonetic string of the query keyword and garbage, are fed into the grammar specification network file. Grammar is specified in BNF form then a word network is built out of the grammar.

4.1.3 Recognition Phase

For the recognition/word spotting phase, the speech archive which forms the target space for searching is digitised and parameterised to speech vectors. The speech vectors will be passed as tokens through the Viterbi algorithms which uses the previously trained HMMs and the grammar specification network file to produce a resultant transcribed output. The resultant transcribed output will consists of locations of the query keyword and locations of the garbage. Below is a sample of the transcribed output from word spotting. The number to the left is the end-time in second when the word label on the right has been spoken. The starting time of the label is taken to be the end-time of the previous label and 0 initially.

Note that the output will contain either the Keyword or the Garbage word.

The kewords that were chosen for the system are keywords from the Computer domain and they are: **Media, Windows, Linux, Programming, Network, Macintosh, Database, Java, Security, Web, Internet, Memory, System, Assembly and Framework.**

Thr keywords were chosen so that they have different length and different pronunciations and phoneme combinations and that are not subset of each other.

Sample Output of Recognition Phase:

```
#!MLF!#
"../test/carrier1.rec"
0.130000 121 NETWORK
0.530000 121 Garbage
0.920000 121 WINDOWS
2.760000 121 Garbage
3.460000 121 Garbage
4.500000 121 Garbage
4.970000 121 Garbage
5.520000 121 Garbage
5.920000 121 Garbage
6.660000 121 Garbage
6.920000 121 Garbage
8.120000 121 Garbage
8.510000 121 Garbage
8.680000 121 NETWORK
9.180000 121 Garbage
.
```

4.2 Retrieval

Our database consists of three main tables:

- Audio file table that contains all information about the uploaded audio file and the path where saved in server
- Member table that contains all information about the members
- Index table that stores for each file the occurences of each keyword of our vocabulary and it's automatically updated after end recognition.

Ranking is performed for the indexed data. Results are ranked according to the existence of the keyword in the name of the title and then according to the amount of occurences of the given word in the content of the audio file.

4.3 Delivery

Our GUI is a website that contains the following:

- Block for enter the username and password to log in the account or to sign up.
- Block for enter the words and choose the category to search.

- After log in, the site member will have a block to upload wav files.

Query Results:

The retrieved files are displayed and the user can listen to them. The occurences of the queried word is marked.

Member Privileges:

- All members have accounts.
- Each member Log in with username and password.
- Upload files and tag them.
- Choose the category of search.
- Search in the content of the audio files with computer domain.
- Edit/delete files, comments, and accounts.

4.4 Technology Used

4.4.1 PHP

PHP stands for Hypertext Pre-processor is a server-side scripting language, which can be embedded in HTML or used as a standalone binary. PHP is an official module of Apache HTTP Server, the market-leading free Web server that runs about 67 percent of the World Wide Web (according to the widely quoted Net craft Web server survey). This means that the PHP scripting engine can be built into the Web server itself, leading to faster processing, more efficient memory allocation, and greatly simplified maintenance. Like Apache Server, PHP is fully cross-platform, meaning it runs native on several flavors of UNIX, as well as on Windows and now on Mac OS X. All projects under the aegis of the Apache Software Foundationincluding PHPare open source software. PHP5 strives to deliver something many users have been clamoring for over the past few years: much improved object-oriented programming (OOP) functionality.

4.4.2 MySQL

SQL Relational Database Management System (RDBMS) which has more than 6 million installations. The program runs as a server providing multi-user access to a number of

databases. Early in its history, MySQL occasionally faced opposition due to its lack of support for some core SQL constructs such as sub selects and foreign keys. However, MySQL found a broad, enthusiastic user base for its liberal licensing terms, performance, and ease of use. MySQL is popular for web applications and acts as the database component of the WAMP server. Its popularity for use with web applications is closely tied to the popularity of PHP, which is often combined with MySQL to encourage its use through stable, well-documented modules and extensions. Several high-traffic web sites (including Flicker, Facebook, Wikipedia, Google, Nokia and YouTube) use MySQL for its data storage and logging of user data. MySQL isnt a database until you give it some structure and form.

4.4.3 Why PHP and MySQL?

- Cost

  Apache/PHP/MySQL combo runs great on cheap, low-end. Consider that although MySQL is open-source licensed for many uses.

- Open source software:

  Open source simply creates better software. Everyone collaborates, the best technology wins. Not just within

one company, but among an Internet-connected, world-wide community. New ideas and code travel the world in an instant. You can see the code, change it, learn from it. Bugs are found and fixed quickly.

- Ease of Use

  PHP is easy to learn, compared to the other ways to achieve similar functionality. Unlike Perl, which has been called a write-only language, PHP has a syntax that is quite easy to parse and human-friendly. And unlike ASP.NET, PHP is stable and ready to solve your problems today.
  Its entirely possible to use PHP just by modifying freely available scripts rather than starting from scratch.

- HTML-embedded

  PHP pages are ordinary HTML pages that escape into PHP mode only when necessary. When a client requests the page, the Web server pre-processes it. This means it goes through the page from top to bottom, looking for sections of PHP, which it will try to resolve. For one thing, the parser will suck up all assigned variables and try to plug them into later PHP commands If everything goes smoothly, the pre-processor will eventually return a normal HTML page to the clients browser. The HTML-embedded of PHP has many helpful consequences: PHP can quickly be added to code. PHP lends itself to a

division of labour between designers and scripters. PHP can reduce labour costs and increase efficiency due to its shallow learning curve and ease of use.

Perhaps the sweetest thing of all about embedded scripting languages is that they dont need to be compiled into binary code before they can be tested or usedjust write and run. PHP is interpreted.

- Cross-platform compatibility PHP and MySQL run native on every popular flavor of UNIX (including Mac OS X) and Windows. A huge percentage of the worlds HTTP servers run on one of these two classes of operating systems. PHP is compatible with the three leading Web servers: Apache HTTP Server for UNIX and Windows, Microsoft Internet Information Server, and Netscape Enterprise Server. Now that PHP runs on Macintosh, PHP is almost totally cross-platform. You can develop on almost any client OS using your favorite tools and then upload your PHP scripts to a server on almost any OS.

- Stability

  The word stable means two different things: The server doesnt need to be rebooted often. The software doesnt change radically and incompatibly from release to release.

Both of these connotations apply to both MySQL and PHP. Apache Server is generally considered the most stable of major Web servers, with a reputation for enviable uptime percentages. Apache HTTP Server seemingly never crashes. It also doesnt require server reboots every time a setting is changed. Apache Server with PHP handily beat both IIS/Visual Studio and Netscape Enterprise Server/Java for stability of environment. PHP and MySQL are also both stable in the sense of feature stability. Applications written in PHP3 will function with little or no revision for PHP4 and 5. And because of the standards-based SQL support, MySQL 3.x databases are easily moved to more current versions.

- Speed

  PHP is pleasingly zippy in its execution, especially when compiled as an Apache module, The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time. PHP5 is much faster for almost every use than CGI scripts. Although many CGI scripts are written in C- one of the lowest-level and therefore speediest of the major programming languages-. Although it takes a slight performance hit by being interpreted rather than compiled, this is far outweighed by the benefits PHP derives from its status as a Web

server module. When compiled this way, PHP becomes part of the http daemon itself. Because there is no transfer to and from a separate application server requests can be filled with maximum efficiency.

- Many extensions

  PHP makes it easy to communicate with other programs and protocols. The PHP development team seems committed to providing maximum flexibility to the largest number of users. In addition, PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

- Fast feature development

  Users of proprietary Web development technologies can sometimes be frustrated by the glacial speed at which new features are added to the official product standard to support emerging technologies. With PHP, this is not a problem. All it takes is one developer, a C compiler, and a dream to add important new functionality. PHP development is also constant and ongoing Compared the .NET transition.

- Popularity

  PHP is fast becoming one of the most popular choices for so-called two-tier development (Web plus data).

- Not proprietary

  PHP is in a position of maximum flexibility because it is, so to speak, ant proprietary. It is not tied to any one server operating system, unlike Active Server Pages. It is not tied to any proprietary cross-platform standard or middleware. It is not tied to any one browser or implementation of a programming language or database.

4.4.4 Hidden Markov Model Toolkit (HTK)

HTK has been developed in Cambridge University as an integrated suite of software tools for building and manipulating continuous density HMMs. It consists of a set of library modules and a set of more than 20 tools. It is written in ANSI C. Although HTK is general-purpose, it also encourages a particular approach to building speech recognition systems. Firstly, HTK is restricted to continuous density HMMs for the purpose of practical application believed to be more robust. Secondly, HTK provides a generalised mechanism which allows tying at all levels. Thirdly, HTK fosters an incremental

approach to model building whereby a system of HMMs is refined through a number of stages involving interleaved model manipulation and model re-estimation. Lastly, it provides a rich set of integrated tools to manipulate a diverse range of data.

Chapter 5

Evaluation

- Performance Measures
- Testing

5.1 Performance Measures

To measure the perfromance of our system, we've used the metrics used in information retrieval community recall and precision. For evaluating ASR performance we use the standard word error rate (WER) as our metric. Since we are interested in retrieval we use OOV rate by type to measure the OOV word characteristics. For evaluating retrieval performance we use precision and recall with respect to manual transcriptions. Let Correct(q) be the number of times the query q is found correctly, Answer(q) be the number of answers to the query q, and Reference(q) be the number of times q is found in the reference. $Precision(q) = \frac{Correct(q)}{Answer(q)}$

$Recall(q) = \frac{Correct(q)}{Reference(q)}$

We compute precision and recall rates for each query and report the average over all queries. The set of queries Q consists of all the words seen in the reference except for a stoplist of 100 most common words. The measurement is not weighted by frequency i.e. each query $q \in Q$ is presented to the system only once, independent of the number of occurences of q in the transcriptions. $Precision(q) = \frac{1}{|Q|} \sum_{q \in Q} Precision(q)$

$Recall(q) = \frac{1}{|Q|} \sum_{q \in Q} Recal(q)$

5.2 Testing

The system has been tested on long audio files that range from 5 to 15 minutes. The test files were recorded in a relatively quiet environment and with a limited number of speakers. The results of recognition were good enough but more training will yield better results.

On start up page you can find a link to the Registration page where the user can register to the website,It also contains a signin link where the registered user can sign into the website to view ,update or delete his personal data through his profile as shown in figure 5.1.As a user query one of the keywords ,the system retrieves relevant audio files satrting at the position of the first occurrence of the queried keyword as shown in figure 5.2.

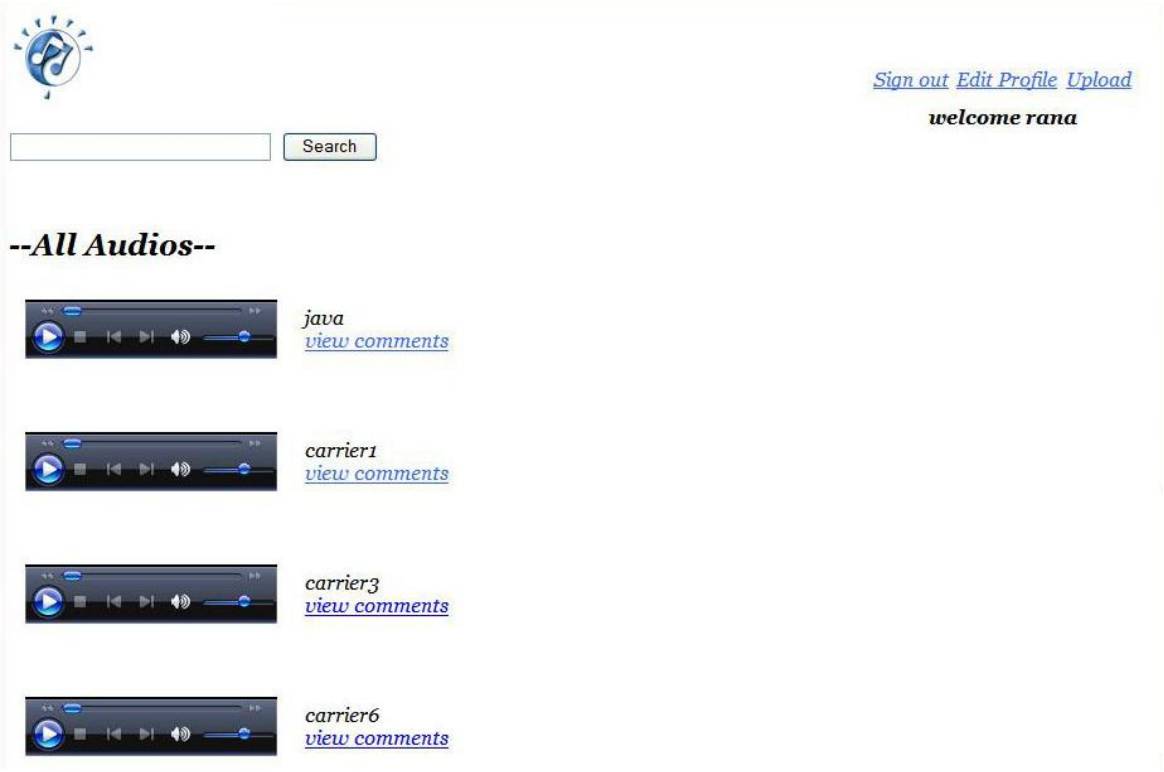

FIGURE 5.1: Website on startup

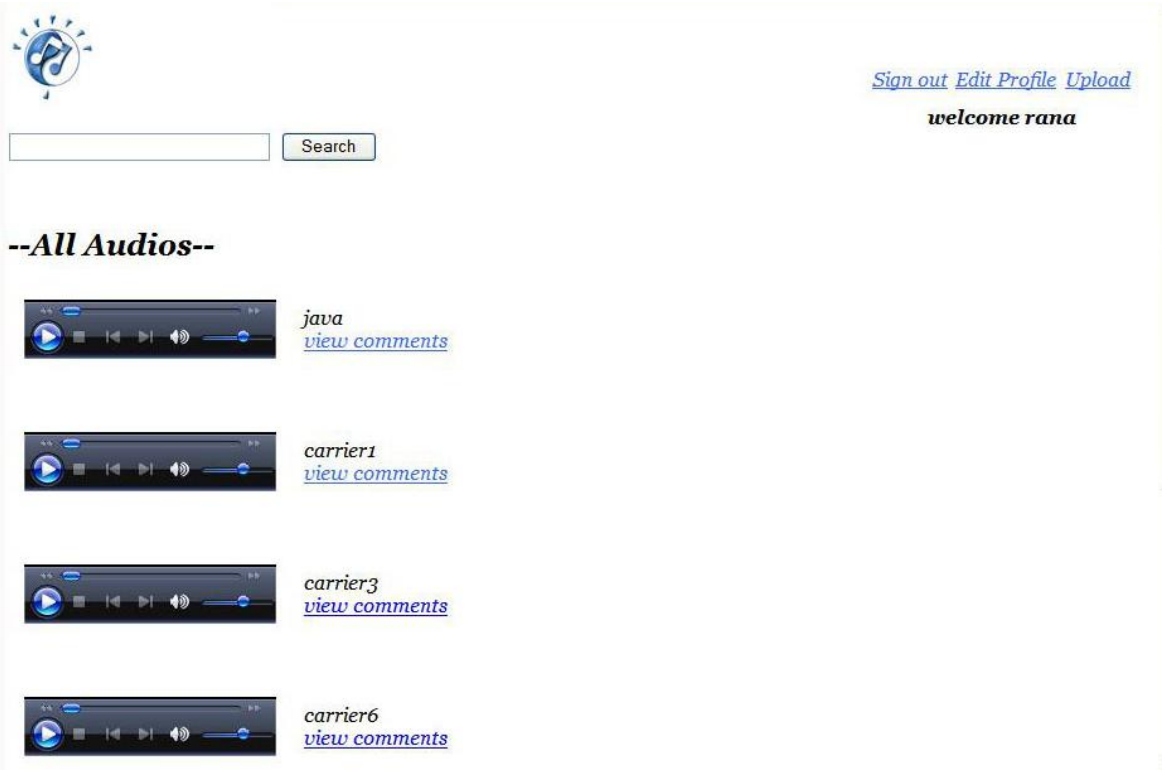

FIGURE 5.2: Website before query is made

Chapter 6

Conclusion & Future Work

- Conclusion
- Future Work

6.1 Conclusion

The system we have developed is able to perform content-based searches of sopken audio files. Much more testing is likely required to verify that this system can work using real-world, more complex data. However, it has been demonstrated that the project with it's current state can serve as the core of a full audio search system, and that such a system can deliver useful results. It has also been demonstrated that the system should be able to handle greater amounts of data after some modifications.

As such, while there are many aws and areas needing improvement in the current implementation of this system, the basic design seems to be sound. However, as mentioned, there is a need for more testing of the system. In particular, tests from a wider range of speaker with different accents and in different recording environments.

6.2 Future Work

To increase the accuracy in speech recognition systems, maybe using a more detailed phone-set can be useful. To be able to

observe the real contribution of this phone set, the experiment should be done on a suficiently large training data-set. Also, alternative filler model types can be examined.

To improve IR accuracy, we are investigating different relevance ranking algorithms and alternative ways of indexing the output of the speech recognizer. We also plan to improve the user interface. A better segmentation and categorization of audio files could allow search and retrieval of small sections of audio recordings that could be listened to as a whole, thus minimizing the navigation within a document.

Appendix A

Source Code

C# Code:

```
1
2  // Retrieve the files to Start Recognition
3  //-------------------------------------------------------
4  MySqlConnection mysqlCon = new MySqlConnection("server=localhost; user id=
       rana;password=; database=rana;");
5  mysqlCon.Open();
6  MySqlDataAdapter DA = new MySqlDataAdapter("SELECT FileName FROM newaudios
       ", mysqlCon);
7  DataTable DT = new DataTable();
8  DA.Fill(DT);
9  dGV.DataSource = DT;
10 // Delete Temp New Audio files from DB
11 MySqlCommand comm = new MySqlCommand();
12 comm.Connection = mysqlCon;
13 for (int i = 0; i < dGV.Rows.Count - 1; i++)
14 {
15      string temp = dGV[0, i].Value.ToString();
16      comm.CommandText = "DELETE FROM newaudios WHERE FileName='"+ temp +"'";
17      comm.ExecuteNonQuery();
18 }
19
20 // Start Recognition
21 //-------------------------------------------------------------------
22 for (int i = 0; i < dGV.Rows.Count - 1; i++)
23 {
24       string com = "-T 1 -C "+"C:/wamp/www/Projects/RecProject/SCR/Config"+"
       -S " + "C:/wamp/www/Projects/RecProject/SCR/java.scp";
25       Process.Start("C:\\wamp\\www\\Projects\\RecProject\\EXEs\\HCopy.exe",
       com);
26 }
```

```
27 //-------------------------------------------------------------------
28
29 // Read Output File
30 //--------------------
31 FileStream FSR = new FileStream("C:\\wamp\\www\\Projects\\Text\\" +
       filename[0] + ".out", FileMode.Open, FileAccess.Read);
32 StreamReader SR = new StreamReader(FSR);
33 St = SR.ReadLine();
34 SR.Close();
35 splitt = St.Split(c2);
36 while (splitt[index] != "</s>")
37 {
38     // check if the word== file name
39     if (splitt[index] == filename[0])
40     checkfilename = true;
41     //check if the word repeated in the same file
42     check = true;
43     for (int j = 0; j < count; j++)
44 if (Words[j] == splitt[index])
45 {
46     RepCount[j]++;
47     check = false;
48 }
49     // if the word is not repeated
50     if (check)
51     {
52 Words[count] = splitt[index];
53 RepCount[count] = 1;
54 count++;
55     }
56     index++;
57 }
58
59 //Update The Index In Database
60 //-----------------------------
61 int ID = 0;
62 comm.Connection = mysqlCon;
63 try
64 {
65     comm.CommandText = "INSERT INTO fileindex (FileName) VALUES ('" +
       filename[0] + "');SELECT last_insert_id();";
66     ID = Convert.ToInt32(comm.ExecuteScalar());
67     for (int j = 0; j < count; j++)
68     {
69                 comm.CommandText = "UPDATE  fileindex SET " + Words[j] +
       "=" + RepCount[j] + " WHERE ID=" + ID;
70                 comm.ExecuteNonQuery();
71     }
```

```
72          mysqlCon.Close();
73      }
74      catch (Exception ex)
75      {
76          MessageBox.Show(ex.Message);
77      }
```

PHP Code:

```
1  //Search
2  //----------------------------------------
3  // Retrieve data from database
4  //----------------------------------------
5            $query = "SELECT FileName,$tempArrSearch,$tempfilenametime FROM
       fileindex
6          WHERE $tempArrSearch>'$val'";
7            $result = mysql_query($query)
8    or die ("<br/>search text not in our domain");
9             while ( $row = mysql_fetch_array($result,MYSQL_ASSOC))
10            {
11                        //check for repeating audio files
12                        $check='true';
13  foreach ($tempsvefilename as $key => $Nam )
14  {
15                                      if($Nam==$row['FileName'])
16                                       $check='false';
17                              }
18                              if($check=='true')
19                              {
20                                  $tempsvefilename[]=$row['FileName'];
21          $tempArr[$row['FileName']]=$row[$tempArrSearch];
22                                      $tempfiletimeArr[$row['FileName']]=$row[
       $tempfilenametime];
23                              }
24                 }
25 //-----------------
26 //Check Priority
27 //-----------------
28                 foreach ( $tempArr as $FileName => $wordcount )
29                 {
30                     $res=$wordcount-floor($wordcount);
31                             if($res>'0')
32                             {
33                                     $tempArrFName[$FileName]=$wordcount;
34                                     arsort($tempArrFName);
35                             }
36                             else
37                             {
38                                 $tempArrRword[$FileName]=$wordcount;
39                                 arsort($tempArrRword);
40                             }
41                     }
42                     if($tempArrFName=="")
43                         $newArr=$tempArrRword;
44                     elseif($tempArrRword=="")
```

```
45                      $newArr=$tempArrFName;
46                  else
47                      $newArr=array_merge((array)$tempArrFName , (array)
    $tempArrRword);
48 //-----------------------------------------
49 //display data in windows media player
50 //-----------------------------------------
51              foreach ( $newArr as $File_Name => $Appcount )
52              {
53                  $Audio=$File_Name.".wav";
54                  $query = "SELECT Description FROM audiofile
55        WHERE AudioName='$Audio'";
56  $result = mysql_query($query)
57       or die ("incorrect query");
58  $row=mysql_fetch_array($result,MYSQL_ASSOC);
59  $Desc=$row['Description'];
60                  $filepath="RecFiles/".$Audio;
61                  ?>
62                          <br/><br/><br/>
63 <B>'.<?php echo $File_Name; ?>.'</B><br />
64 <object id= "Audio" width="320" height="47"
65         classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6" type="
    application/x-oleobject" >
66 <param name="URL" value=<?php echo $filepath; ?> >
67 <param name='AutoStart' value='False'/>
68 <param name='uiMode' value='full'/>
69 <param name='PlayCount' value='1'/>
70 <param name='Enable' value='True'/>
71 <param name='Volume' value='50'/>
72 <param name='CurrentPosition' value='<?php echo $tempfiletimeArr[$File_Name
    ];?>'/>
73 </object><br/>
74                          <B>Describtion: <?php echo $Desc;?></B><br/>
75                          <B>Mention: <?php echo floor($Appcount);?></B>
76                          <?php
77              }//end foreach
78 //--------
79 //Upload
80 //--------
81 if (!empty($_FILES['userfile']['type']))
82                         {
83     $extension = "";
84     if($_FILES['userfile']['type'] == "audio/wav"){ $extension = "wav"; }
85     else
86                             {
87        $errors['userfile'] = "Your file is not a supported format. Please
    upload .wav files only!";
88     }
```

```
89      $serverpath = "C:/wamp/www/Projects/Uploaded/" . $fileName;
90      $move_result = move_uploaded_file($_FILES['userfile']['tmp_name'],
        $serverpath);
91  }
92  //Insert the Information of the file in the Database
93  //-------------------------------------------------------
94  $dirToSaveIn="/Uploaded/".$fileName;
95  mysql_select_db($database);
96  $svename=@$_GET['sveusername'];
97  $query = "SELECT ID FROM member
98          WHERE UserName='$svename'";
99  $result = mysql_query($query)
100 or die ("incorrect username");
101 $row = mysql_fetch_array($result,MYSQL_ASSOC);
102 $userID=$row['ID'];
103 $query = "INSERT INTO audiofile (AudioName, FileSize, AudioFilePath,
        Description, memberID) ".
104             "VALUES ('$fileName', '$fileSize', '$dirToSaveIn','$fileDesc' ,
        '$userID')";
105  mysql_query($query)
106                          or die('Error, query failed');
107
108                          $query = "INSERT INTO newaudios (FileName) ".
109             "VALUES ('$fileName')";
110  mysql_query($query)
111                          or die('Error, query failed');
112  echo "<br>File $fileName uploaded<br>";
```

Bibliography

[1] A. Jansen and P. Niyogi *An Experimental Evaluation of Keyword-Filler Hidden Markov Models*, April 2009.

[2] Igor Szoke, Petr Schwarz, Lukas Burget, Martin Karaat, Jan Cernocky *Phoneme based acoustics keyword spotting in informal continuous speech*, Faculty of Information Technology, Brno University of Technology.

[3] Aydin Akyol *Filler Model based Word Level Confidence Measures for Spoken Dialogue Systems*, July 3003.

[4] M. A. Siegler, M. J. Witbrock, S. T. Slattery K. Seymore, R. E. Jones, and A. G. Hauptmann *An Experimental Evaluation of Keyword-Filler Hidden Markov Models*, Justsystem Pittsburgh Research Center.

[5] Konstantinos Koumpis *Opportunities and Challenges in Spoken Media Search*, Presented at the IST Consultation Workshop on Challenges of Future Search Engines, 2005.

[6] Wai Yat Wong, John Robertson *Application of phonological knowledge in audio word spotting*, 1998.

[7] K. M. Knill and S.J. Young *Speaker Dependant Keyword Spotting for Accessing Stored Speech*, October 1994.