iTalk

Index your talk

by

Archana Adma

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2006

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Abstract

iTalk is an integrated tool developed to enable search through audio content recorded from voice conferences. In this digital world, where VoIP and text based search engines rule the user community, iTalk comes handy by saving the user's contribution to voice conferences and indexing them with the help of manipulated speech recognition technologies, thereby providing the users a gateway to archived audio conferences. A 3-tier indexing algorithm enables retrieval of small chunks of audio from the recorded conversation, which saves user's time in skimming through the audio content.

This project is aimed at the integration of the voice conferencing tool Vocal Village developed at the University of Toronto and the file system based search engine Wumpus developed at the University of Waterloo. The resulting system provides a back stream to record, index and archive the audio for retrieval based on the associated XML based index file. The XML index file is generated using transcriptions obtained by the 3-tier algorithm to which each audio file is fed. The output accuracy is dependent on the training files obtained from the users. The more training length is, the better is the output.

In this thesis we present the design and development of the tool; we also provide the necessary background information on speech recognition tools and VoIP technologies.

Acknowledgements

My supervisor Charlie Clarke is duly thanked for his enthusiasm, encouragement, and support throughout the development of the iTalk project and for his assistance and advice in writing this thesis. His care in making my masters program as easy as it can get, cannot be sufficiently thanked. Special thanks to my thesis readers: Gordon Cormack and Olga Vechtomova, for their time, guidance, availability and understanding. I would also like to graciously acknowledge funding from IBM Canada and Centre for Communications and Information Technology Ontario.

Many thanks to Mark Chignell, Dr. Sharon Straus, MIE lab mates and KTP group from University of Toronto, who gave me an opportunity to conduct my research from their place along with a lot of support by involving me in several of their events. To Helen Evans, who helped me edit my thesis and made it into a good work as it looks today. Thanks to all the PLG and IR group members, especially Stephen Büttcher and Philip Tilker, whose research advices and help made way to the development of iTalk system in a short time. My thanks are due to Ashif Harji and Mike Patterson, who have provided the necessary technical support throughout and sometimes even at odd hours. To Maheedhar Kolla, who helped in proofreading and providing valuable technical writing advises.

As brief as I would like to make my acknowledgements, I would not want to forget anyone who is responsible for this thesis. To all the administrative staff and all my friends at Waterloo and to all of those whom I have talked to, met with, debated with, and even smiled to: I would like to thank you all, for the small things that give a lot of inspiration and keep the courage to reach our goals. Always mentioned at the last, just because they are family and they know that irrespective of where I mention them, they hold a very prestigious position in my life and I can never sufficiently thank them all for everything they are, they did, and they will do for me. Thank you for being my family. A special group of friends who are more than just friends ABCD-IT are all thanked in person for their direct and indirect moral support, especially Sravanthi Boocha and Anuhya Cheruku. Last but not the least, my little niece Dhanya Karra, who used to lift the stress off this thesis with her child talk, I am smiling even as I am writing this, I guess that says everything.

Dedication

To the one without whose artistic talent this thesis would have been yet another thesis with lines of text and figures that are just boxes, without whose support I wouldn't have successfully traveled all the way to completing my thesis, who has been encouraging me despite spending days of lacking attention from me during my masters program, to you dear husband, Shashidhar Reddy Methuku, I dedicate my thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Technology has taken a front seat in communications with the advancement of Internet. VoIP technology has become one of the leading communications technologies today. However, the dream of recording conversations for future revisiting has been complex - the reasons include technological complexity associated with recording of the VoIP conferences, lack of proper audio indexing tools and techniques. Today, VoIP technology has advanced to the stage of recording conferences in real time. However, the need for audio indexing tools, such as visiting a particular point of time in the conference audio, has not been achieved successfully primarily due to the lack of proper tools to index the audio files. Mere indexing with the metadata of the file, even though helpful in retrieving the correct audio file, is not very effective in skimming through the content of the audio files.

1.1 Need for iTalk System

Church [20] assesses the possible trends in development of future storage to be in the range of petabytes. The availability of such a huge storage is expected to cause major dislocations, as everyone will be able to store large amounts of text, speech, and video. Going even further, Church [20] predicts that in the future, search will become a more important application than speech coding and data entry. As speech is the key to information retrieval from audio and video clips, speech-based research is expected to increase in the near future. Church [20] presents milestones in speech and multimodal technology and research as follows:

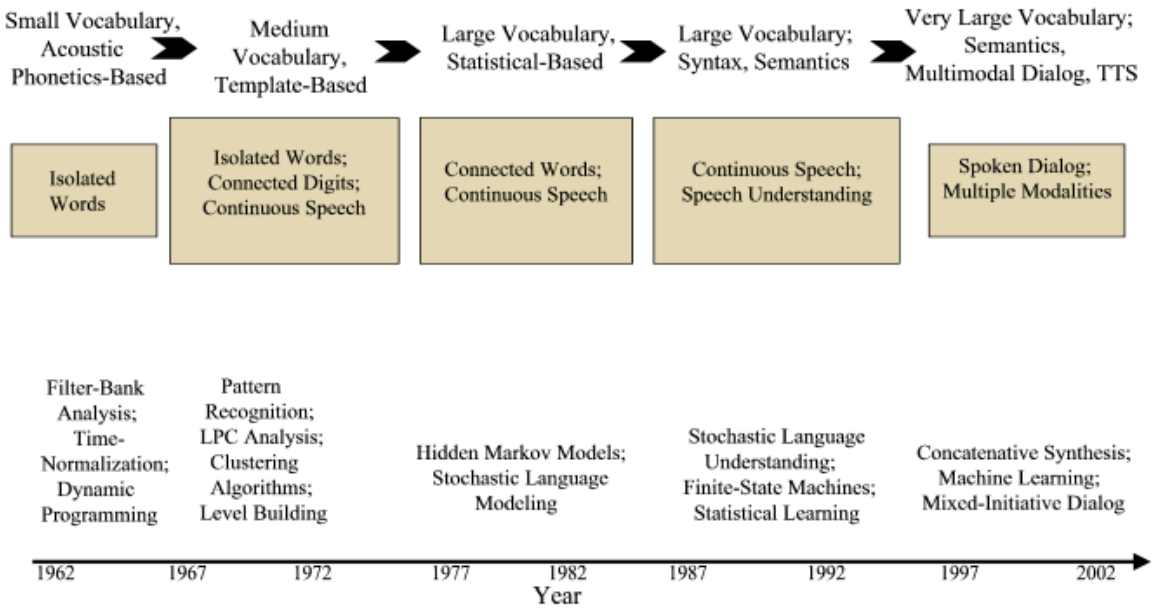

Figure 1: Milestones in speech and multimodal technology and research [20].

Though lot of work has already been done in the direction of audio indexing, most of it was focused on the read-speech, broadcast news domain, and very few, in contrast, on spontaneous conversational speech. Methods to cope with informal speech containing misstatements, corrections and disfluencies, which are more complex to handle, are yet to be developed. The biggest challenge, however, is to design user interfaces for people to sift through spoken documents as rapidly as they can pick through clean text [21]. The need for tools, which provide an interface for users to sift through their audio conferences for spoken words as they would through a text document for some specific words, is hence desired. iTalk system is developed to satisfy this need: archive, index, and retrieve spontaneous speech.

1.2 Hypothesis for the iTalk system

As will be explained in chapter 2 of this thesis, typical spoken document retrieval involves generating a transcript for a recorded audio file by running it through a speech recognition tool. Further, text-based indexing techniques are applied to this transcript to search for specific words in an audio file. Such spoken document retrieval would retrieve the audio file that contains a specific keyword from among thousands of audio files. However, such retrieval is not very effective considering the number of false positive documents and the amount of time spent in skimming through the audio file looking for the keyword. A more desired approach is to retrieve either the exact position in the audio file where the keyword occurs or the sets of keywords with associated concepts. This requires time-stamping the audio file and the generated transcript. To achieve this, iTalk hypothesis uses a 3-tier feed

and multiple hypothesized transcript generation methodology that is further described in chapter 3.

This method ensures better indexing of all positions in the audio file and also benefits the recognition accuracy.

1.3 Organization

The remainder of this thesis presents the iTalk spoken snippet retrieval system, the techniques and design involved in the development of this system and its applications.

Chapter 2 presents the required background on the speech recognition process, standard audio mining approach, out of vocabulary words and their effect on spoken document retrieval, VoIP based audio conferences, and information retrieval process using text-based search engines. This information is followed up with relevant tools and techniques developed in the field of spoken document retrieval.

Chapter 3 presents the audio indexing methodology for automatic archiving, indexing, and retrieving of recorded VoIP based audio conferences.

Chapter 4 describes the architecture of iTalk and the initial design with informal user feedback obtained from a demo at CASCON 2005 conference followed by improved design developed based on the criticism received.

Chapter 5 emphasizes the potential future direction for the development of iTalk tool and Chapter 6 concludes the contributions of this work followed by references in bibliography.

Chapter 2

Background and Related Work

2.1 Speech Recognition

Speech is a combination of information obtained from discourse, semantics, syntax, phonologies, phonetics and acoustics [1]. Speech recognition, however, is the process of converting an acoustic signal into a set of words. In general, speech recognition involves the ability to match a voice pattern against a provided or acquired vocabulary. The most successful model to match speech patterns in Automatic Speech Recognition (ASR) is the Hidden Markov Model (HMM).

2.1.1 Hidden Markov Model

Hidden Markov Models [2] are referred to probabilistic functions of Markov chains in the communications literature. This section presents the theory of discrete Markov chains and extends it to the ideas behind HMM.

Discrete Markov Processes:

Consider a system which may be described at any time as being in one of a set of N distinct states, $S_1, S_2, ..., S_N,$ (where N=5), as illustrated in Figure 2 [2].

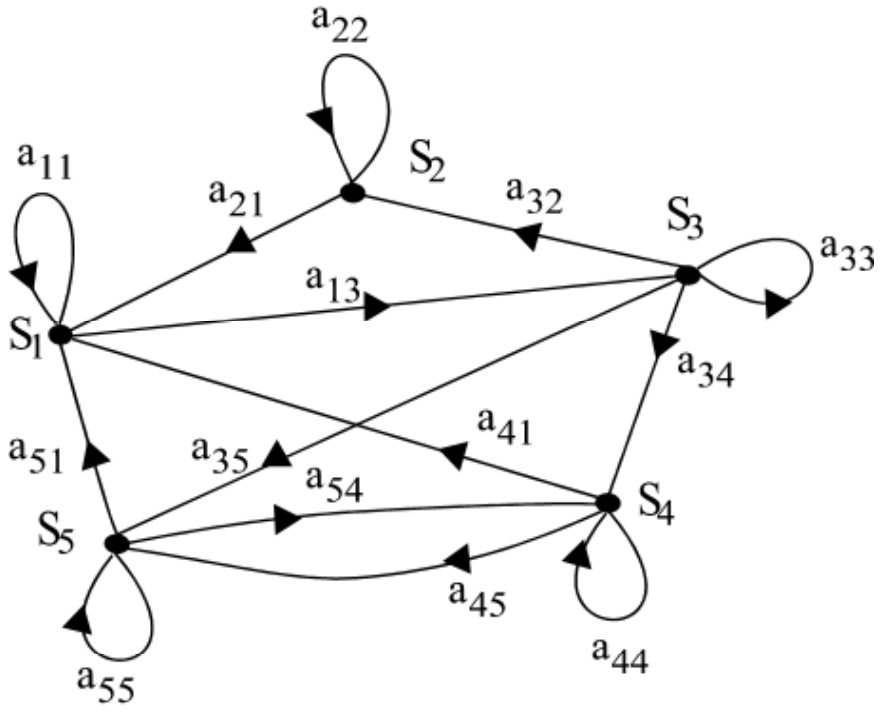

Figure 2: A Markov chain with 5 states and selected state transitions [2].

At regularly spaced discrete times, the system undergoes a change of state, even back to the same state, according to a set of probabilities associated with the state. The time instants associated with the state changes are denoted by $t = 1, 2, ...,$ and the actual state at time t is denoted by q_t . A complete probabilistic description of the above system would, in general, require specification of the current state (at time t) as well as all the predecessor states. For a discrete first order Markov chain, this probabilistic description would require just the current and the predecessor state. This can be represented as follows:

$$P[q_t = S_j / q_{t-1} = S_i \, , q_{t-2} = S_k , ...] = P[q_t = S_j / q_{t-1} = S_i] \qquad \text{Eqn-1}$$

Only those processes in which the right hand side of Eqn-1 is independent of time are considered. This leads to the set of state transition probabilities a_{ij} of the following form:

$$a_{ij} = P[q_t = S_j / q_{t-1} = S_i], \; 1 \le i, \; j \le N \qquad \text{Eqn -2}$$

with the state transition coefficients having the properties:

$$a_{ij} \geq 0 \qquad \text{Eqn-3a}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \text{Eqn-3b}$$

since they obey standard stochastic constraints. The above stochastic process could be called an observable Markov Model, since the output of the process is the set of states at each instant of time, in which each state corresponds to a physical observable event.

Hidden Markov Model:

The concept of Markov model [2] includes the case where the observation is a probabilistic function of the state. The resulting model is a doubly embedded stochastic process with an underlying invisible stochastic process that can only be observed through another set of stochastic processes that generate the sequence of observations. This model is generally referred to as the HMM [2].

An HMM is characterized by the following:

1. N, the number of states in the model. Generally, the states are interconnected in such a way so that a needed state can be reached regardless the current state. Individual states are denoted as $S = \{S_1, S_2, ..., S_N\}$, and the state at time t is denoted as q_t.
2. M, the number of distinct observation symbols per state, i.e. the discrete alphabet size. The observation symbols correspond to the physical output of the system being modeled. Individual symbols are denoted as $V = \{V_1, V_2, ..., V_M\}$.
3. The state transition probability distribution $A = \{a_{ij}\}$ where

$$a_{ij} = P[q_{t+1} = S_j / q_t = S_i], \ 1 \leq i, j \leq N \qquad \text{Eqn-4}$$

$a_{ij} > 0$ for all *i,j* considering the case where any state can reach any other state in a single step.

4. The observation symbol probability distribution in state *j*, $B = \{b_j(k)\}$, where

$$b_j(k) = P[v_k \text{ at } t / q_t = S_j],\ 1 \le j \le N,\ 1 \le k \le M \qquad \text{Eqn-5}$$

5. The initial distribution $\pi = \{\pi_i\}$ where

$$\pi_i = P[q_1 = S_i], 1 \le i \le N \qquad \text{Eqn-6}$$

Given appropriate values of *N, M, A, B* and π, the HMM can be used as a generator to give an observation sequence of the following form:

$$O = O_1 O_2 \ldots\ldots\ldots O_T \qquad \text{Eqn-7}$$

where each of the observations O_t is one of the symbols from *V*, and *T* is the number of observations in the sequence. The whole procedure can be subdivided into the following stages [2]:

1. Choose an initial state $q_1 = S_i$ according to the initial state distribution π.
2. Set *t = 1*.
3. Choose $O_t = v_k$ according to the symbol probability distribution in state S_i, i.e., $b_i(k)$.
4. Transit to a new state $q_{t+1} = S_j$ according to the state transition probability distribution for state S_i, i.e., a_{ij}.
5. Set *t= t + 1*; return to step 3 if *t < T*; otherwise terminate the procedure.

The above procedure can be used both as a generator of observations, and as a model for how a given observation sequence was generated by an appropriate HMM. Thus, we can say that a complete specification of an HMM requires specification of two model parameters (N and M), specification of observation symbols, and the specification of the three probability measures A, B, and π. The complete parameter set of the model can be represented as:

$$\lambda = (A, B, \pi) \qquad \text{Eqn-8}$$

Thus, a HMM is defined [2] as a pair of stochastic processes *(X,Y)*, where *X* is a first order Markov chain and is not directly observable; *Y* is a sequence of random variables taking values in the space of acoustic parameters or observations.

Assumptions in HMMs specific to speech recognition [2]:

1. The first order Markov hypothesis states that history has no influence on the chain's future evolution if the present is specified.
2. The output independence hypothesis states that neither the chain evolution nor the past observations influence the present observation, if the last chain transition is specified.

Types of HMMs:

HMMs can be classified as discrete, continuous, and semi-continuous, based on the nature of the elements of the B matrix, which are the distribution functions [2].

In discrete HMMs, distributions are defined on finite spaces. The observations would be vectors of symbols in a finite alphabet of ***N*** different elements. For all ***Q*** vector components, a discrete density $\{w(k) \mid k = 1,2,\ldots,N\}$ is defined. The distribution is then obtained by multiplying the probabilities of each component. The components are assumed to be independent.

In continuous HMMs, distributions are defined as probability densities on continuous observation spaces. Strong restrictions have to be imposed on the functional form of the distributions, in order to have a manageable number of statistical parameters to estimate. The popular approach is to characterize the model transitions with mixtures of base densities **g** of a family ***G*** having a simple parametric form. The base densities $\boldsymbol{g} \in \boldsymbol{G}$ are usually Gaussian or Laplacian, and can be parameterized by the mean vector and the covariance matrix. To model complex distributions, a large number of base densities have to be used in every mixture. This procedure may require a large training corpus of data for the estimation of the distribution parameters. When the available corpus is not large enough, distributions can be shared among transitions of different models to reduce problems.

In semi-continuous HMMs, all mixtures are expressed in terms of a common set of base densities. Different mixtures are characterized only by different weights. In general, semi-continuous modeling involves interpretation of the input vector **y** in terms of several components $y[1], y[2], \ldots, y[Q]$. Each component is associated with a different set of base distributions. The components are assumed to be statistically independent, and, hence, the distributions associated with the model transitions are products of the component density functions.

Computation of probabilities with discrete models is faster as compared with continuous models. However, it is possible to speed up the mixture densities computation by applying vector quantization on the Gaussians of the mixture. Parameters of statistical models are estimated by iterative learning algorithms, in which the likelihood of a set of training data is guaranteed to increase at each step [2].

2.1.2 Methods in Automatic Speech Recognition

Sequences of vectors [3] of acoustic parameters are treated as observations of acoustic word models that are used to compute the probability of observing a sequence of vectors when a word sequence is pronounced. Given a sequence Y_1^T, a word sequence $\hat{W}$ is generated by the ASR system with a search process based on the rule:

$$\hat{W} = \arg\max_{w} \quad P(Y_1^T / W) P(W),$$

where $\hat{W}$ corresponds to the candidate having maximum a-posteriori probability (MAP). While $P(Y_1^T / W)$ is computed by the acoustic models, *P(W)* is computed by the language models.

Acoustic Models -A sequence of acoustic parameters, extracted from a spoken utterance, is seen as a concatenation of elementary processes described by the HMMs [3]. Acoustic models includes both word and unit models.

Word and unit models: Words are represented by networks of phonemes [3]. Each path in a word network represents a pronunciation of the word. The same phoneme can have different acoustic distributions of observations if pronounced in different contexts. Models of a phoneme in different contexts are called Allophone models. The decision as to how many allophones should be considered for a given phoneme depends on many factors such as the availability of sufficient training data to infer the model parameters. In principle, an allophone should be considered for every different word in which a phoneme appears. If vocabulary is large, it is unlikely that there is enough data to train all the allophone models. Hence, models for allophones are considered at a different level of detail i.e. word, syllable, triphone, diphone, context independent phoneme. Probability distributions for an allophone that have a certain degree of generality can be obtained by mixing the distributions of more detailed allophone models.

One other way allophones can be considered is to choose them by clustering possible contexts. This choice can be made by Classification and Regression Trees (CART) [3]. CART is a binary tree having a phoneme at the root and, each node n_i associated with a question Q_i about the context. There are algorithms for growing and pruning CARTs

based on automatically assigning questions to a node from a manually determined pool of questions. The leaves of the tree may be simply labeled by an allophone symbol. Each allophone model is an HMM made of states, transitions and probability distributions.

Language Models -The probability *P(W)* of a sequence of words $W = w_1, w_2, ..., w_L$ is computed by a Language Model. In general *P(W)* can be expressed as follows:

$$P(W) = P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i \mid w_0, ..., w_{i-1}) \quad [3]$$

2.1.3 Pattern Recognition approach to continuous speech recognition system

This section presents how specific aspects of HMM theory are applied to continuous speech recognition. Figure 3 [2], shows a block diagram of pattern recognition approach to continuous speech recognition system.

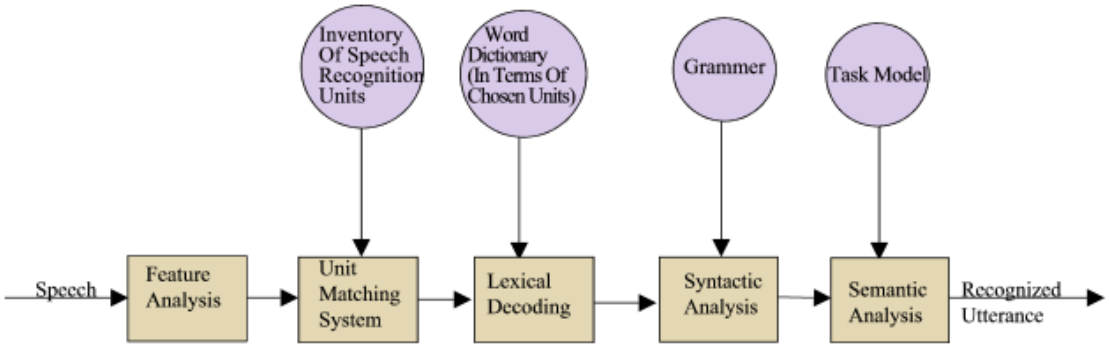

Figure 3: Block diagram of a continuous speech recognizer [2].

The key signal processing steps for pattern recognition approach include [2]:

1. Feature Analysis: A spectral and/or temporal analysis of the speech signal is performed to generate observation vectors, which can be used to train the HMMs that characterize different speech sounds.

2. Unit Matching System: First a choice of speech recognition unit must be made. Possibilities include linguistically based sub-word units such as phones, diphones, demisyllables, as well as derivative units such as fenemes, fenones, and acoustic

units. Other possibilities include whole word units, and even units, which correspond to a group of 2 or more words. Ex: - and an, in the, of a etc. Generally, the less complex the unit is, the fewer of them are in the language, and the more complicated their structure is in continuous speech. For large vocabulary speech recognition (involving 1000 or more words), the use of sub-word speech units is almost mandatory, as it would be quite difficult to record an adequate training set for designing HMMs for units of the size of words or larger. However, for specialized applications such as small vocabulary, it is both reasonable and practical to consider the word as a basic speech unit. Independent of the unit chosen for recognition, an inventory of such units must be obtained via training. Typically, each such unit is characterized by some types of HMMs, whose parameters are estimated from a training set of speech data. The unit matching system provides the likelihood of a match of all sequences of speech recognition units to the unknown input speech. Techniques for providing such match scores, and in particular determining the best match score (subject to lexical and syntactic constraints of the system) include the stack decoding procedure, various forms of frame synchronous path decoding, and a lexical access scoring procedure.

3. Lexical Decoding: This process places constraints on the unit matching system so that the paths investigated are those corresponding to sequences of speech units present in a word dictionary (a lexicon). This procedure implies that the speech recognition word vocabulary must be specified in terms of the basic units chosen for recognition. Such specification can be deterministic (Ex: - one or more finite state networks for each word in the vocabulary) or statistical (Ex: - probabilities attached to the arcs in the finite state representation of words). In the case when the chosen units are words or word combinations, the lexical decoding step is essentially eliminated and the structure of the recognizer is greatly simplified.

4. Syntactic Analysis: This process places further constraints on the unit matching system so that the paths investigated are those corresponding to speech units which comprise words (lexical decoding) and for which the words are in a proper sequence, as specified by a word grammar. Such a word grammar can then be represented by a deterministic finite state network, in which, all word combinations which are accepted by the grammar are enumerated, or by a statistical grammar. Ex: - a trigram word model in which probabilities of sequences of 3 words in a specified order are given. For some command and control tasks only a single word from finite set of equiprobable is required to be recognized, which leads to grammar being either trivial or unnecessary. Such tasks are often referred to as isolated word speech recognition tasks. For other applications, such as digit sequences, very simple grammars are often adequate. Ex: - any digit can be spoken and followed by any other digit. Finally, there are tasks for which grammar is the dominant factor and, although it adds a great deal

of constraint to the recognition process, the procedure also improves recognition performance imposing the resulting restrictions on the sequence of speech units, which are valid recognition candidates.

5. Semantic Analysis: This process, similar to the steps of syntactic analysis and lexical decoding, adds further constraints on the set of recognition search paths. One way in which semantic constraints are utilized is via a dynamic model of the state of the recognizer. Depending on the recognizer state, certain syntactically correct input strings are eliminated from consideration. This allows making the recognition task easier, which, in turn, results in higher performance of the system.

An additional factor that affects the implementation process of speech recognizer refers to the method used to separate background silence from the input speech. There are three reasonable ways to accomplish this task [2]:

1. Explicit detection of speech presence via techniques that discriminate background from speech on the basis of signal energy and signal durations. Such methods have been used for template-based approaches because of their inherent simplicity and their success in low to moderate noise backgrounds.

2. Construction of a model of the background silence, e.g. a statistical model, and present the incoming signal as an arbitrary sequence of speech and background in the following form:

   $$\text{Signal} = (\text{silence}) - \text{speech} - (\text{silence})$$

   where, the silence part of the signal is optional in that it may not be present before or after the speech.

3. Extension of speech unit models so that background silence is included (optionally) within the first and/or the last state of the model, and, consequently, silence inherently is included within all speech unit models.

All three techniques have been utilized in speech recognition systems.

2.1.4 Word Error Rate (WER)

The performance of speech recognition systems is generally measured based on word error rate. In simple terms, word error rate refers to the measure of the average number of word differences between the recognized word sequence and the reference of correct word sequence. The general difficulty of measuring the performance lies in the fact that the recognized word sequence can have length different from the reference word sequence [31].

The WER is derived from the Levenshtein distance, working at word level instead of character. This problem is solved by aligning the recognized word sequence with the reference sequence using dynamic string alignment. Word error rate could then be computed as [31]:

$$WER = \frac{S + D + I}{N}$$

where

N is the number of words in the reference,

S is the number of substitutions,

D is the number of the deletions,

I is the number of the insertions,

Word Recognition Rate (WRR) reports the performance of speech recognition tools in terms of the number of words in the recognized word sequence that matches with the ones in the referenced word sequence, in which the reference word sequence is considered to be correct. The WRR can be obtained from the WER following the rule [31]:

$$WRR = 1 - WER = \frac{N - S - D - I}{N} = \frac{H - I}{N},$$

where

H is the number of correctly recognized words.

2.1.5 Summary of Automatic Speech Recognition Process

Speech recognition [4] is the process of analyzing an audio signal to determine the words uttered by the speaker. Speech recognition involves audio and lexical processing. Audio processing creates a network of possible phonetic sequences that could match the input speech. Lexical processing refines the network by using application context information captured in grammars. A typical speech recognition process is represented in Figure 4 [4].

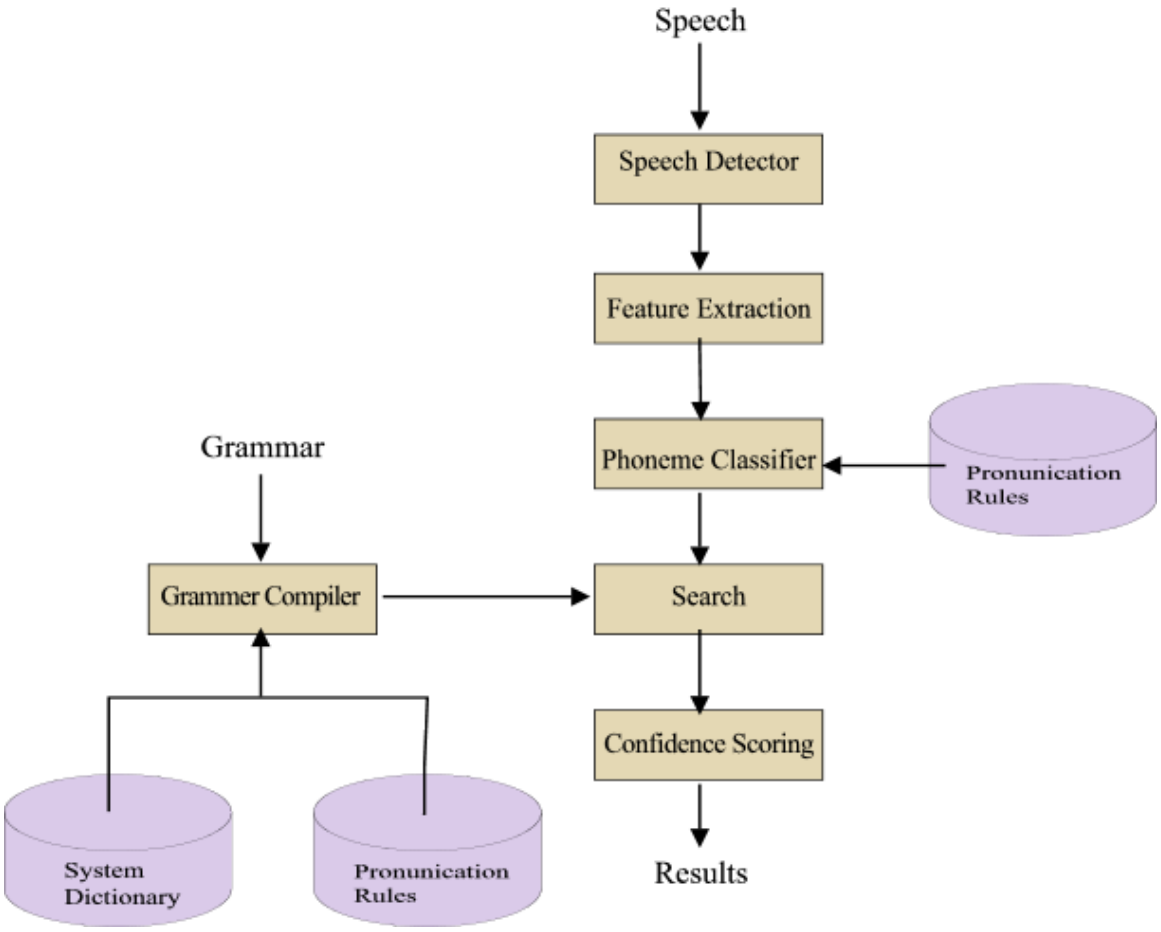

Figure 4: A typical speech recognition process [4].

The automatic speech recognition process begins with capturing the audio signal, converting it into electrical form, and digitizing it for computer analysis. A speech detector, or end-pointer, identifies where the speech starts and stops in the signal. This eliminates non-speech noise like silence and background noise from being processed. Spoken language is composed of a sequence of words, and words are composed of a sequence of phonemes. Since the phonemes are the smallest distinctive acoustic units, altering a single phoneme will transform one word into another. Ex:- by changing "mah" to "rah", mat becomes rat. HMMs identify the phoneme sequence in the end-pointed speech signal by calculating the most probable phoneme sequences that best describe the speech input. These probabilities are derived using mathematical acoustic models that represent archetypical phonemes. The acoustic models are built by distilling large amounts of recorded speech in order to capture the essential characteristics of each phoneme. A large body of training data is required to overcome the variability inherent in speech of every individual. The variability in speech can be attributed to the differences in regional accents, speaking rate, background noise, channel characteristics, linguistic content, physiological differences, etc. Sometimes similar sounding phrases are also confusing. This is especially common in unusual contexts, noisy situations,

and lyrics in music. Speech recognition does not attempt to distinguish a single phrase that was spoken, but proposes a phoneme lattice n-best list of similar sounding phrases ranked by likelihood [4].

The speech recognizer further searches through the network of possible phoneme sequences to find the most likely phrase that was spoken. Most of the possible phrases in the n-best list are meaningless to the application because they are completely out of context. Application context is captured using a recognition grammar. A grammar is a set of allowable phrases at a given point in an application. Speech recognition grammars are not meant to represent all of the phrases in a language. Instead, each prompt in a speech application may reference a unique grammar that constrains the associated speech recognition to a specific range of phrases. As the application context changes, different grammars are loaded. Grammars are built by listing the allowable phrases with shorthand to simplify common constructs. For example, shorthand might be used to indicate that one word out of a set can appear in a given location, as opposed to solution that requires an explicit list of all complete phrases.

Example:

[I am working] [today][tonight]

is more compact than:

[I am working today]

[I am working tonight]

Paths in the grammar can include weights to indicate that the word sequence "*Toronto, Ontario*" is more likely to be said than "*Surrey, Ontario*". All the words in a grammar must be converted to phonemes as used in speech recognition process. First, each word is looked up in a pronunciation dictionary that provides the corresponding phonemes. Custom pronunciation dictionaries can be used to control the pronunciations used in general applications. If a word is not found in the dictionary, pronunciation rules are applied to sound out the answer similar to the way a person would. The result of the conversion process is a network of phoneme strings that represent phrases allowed by the grammar. The final step in the speech recognition process is matching the network of acoustic segments with their phoneme labels to the phoneme strings allowed by the grammar [4].

2.2 Audio Mining

Audio Mining [6], also referred to as audio searching, takes a text-based query and locates the search term or phrase in an audio/video file. In simple terms, Audio Mining is a technique that is used to search audio/video files for occurrences of specific words or phrases.

While speech recognition technology is used to recognize the words or phonemes spoken in an audio or video file, audio mining is aimed at identification of location of specific words or phonemes within audio or video files that contain speech. There are two main approaches to audio mining:

1. LVCSR Audio Mining
2. Phonetic Audio Mining

LVCSR Audio Mining - Large Vocabulary Continuous Speech Recognition (LVCSR) audio mining has two stages – pre-processing or indexing stage and the search stage [6]. During the indexing stage, a large vocabulary recognizer processes the speech content in the input file to generate a searchable index file with information about the words spoken in the input file. During the search stage, firstly, a search term is defined and then one or more index files are searched for all occurrences that match the specified search term. The audio or video files that result from the search can be either displayed as "search hits", or can be played to the user after being subdivided into parts based on relevancy.

Phonetic Audio Mining - Phonetic audio mining is also a two-stage process that contains indexing and search stages. During the indexing stage, speech from the input file is processed with a phonetic recognizer to create a phonetic-based index file [6]. This index file stores the phonetic content of the speech. During the search stage, a search term is defined and converted into a phoneme string. A number of phonetic index files are searched to identify all occurrences of the search term.

Phoneme-based searches can result in more false matches than the text-based approach. The issue is particularly relevant for short search terms, as many words sound alike or sound like parts of other words. As pointed out by Dan Ellis [6], it can be complicated for a phonetic system to accurately classify a phoneme, except by recognizing the entire word that it is a part of or by understanding that a language permits only certain phoneme sequences. On the contrary, phonetic indexing can still be useful if the analyzed material contains important words that are likely to be missing from a text system's dictionary, such as foreign terms and names of people and places [6].

2.2.1 How Audio Mining Works?

LVCSR audio mining systems and phoneme-based audio mining systems work in much the same way, except that the LVCSR system uses a text-based dictionary and the latter uses a phonetic dictionary [6]. One of the successful models for audio mining, the ScanSoft's audio mining development system, is represented in the Figure 5 [6].

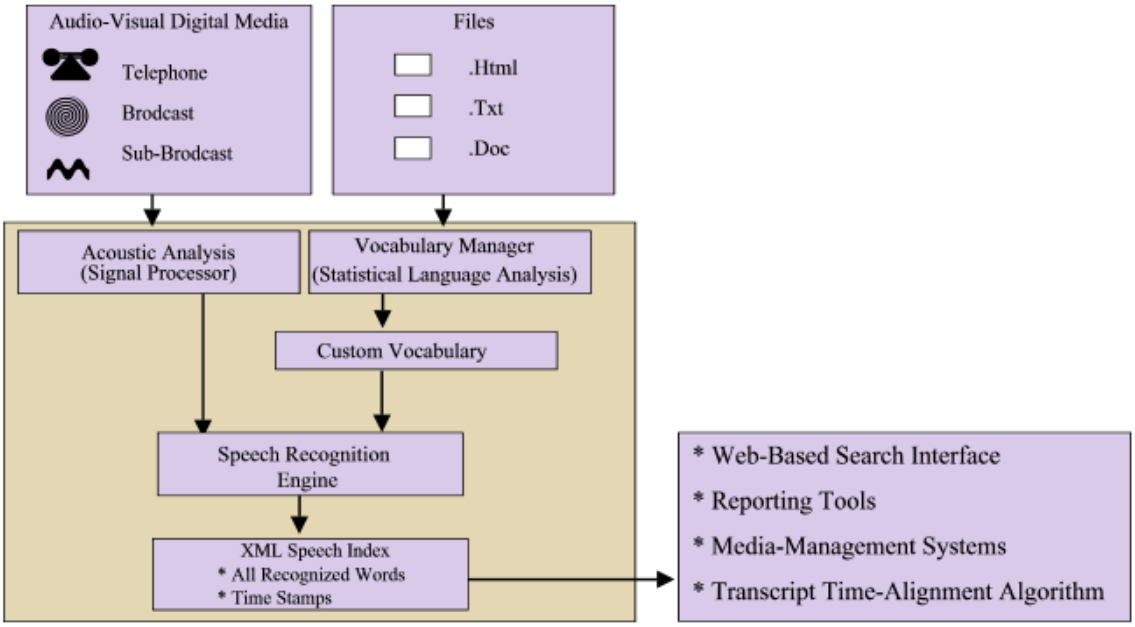

Figure 5: Scansoft's audio mining development process [6].

The audio/video file is given to the acoustic analyzer, the outputs of which alongside with the custom vocabulary information are processed by the speech recognition engine to generate an XML-based speech index file. The XML speech index file consists of all the words recognized by the speech recognizer along with time stamps of their occurrence in the audio/video file. These XML speech index files are accessible by either the web-based search interfaces or other media management systems for retrieving specific audio information.

2.2.2 Out of vocabulary (OOV) words in audio mining or spoken document retrieval

The general approach for information retrieval from audio sources is to use a large vocabulary continuous speech recognition (LVCSR) system to generate word-level transcriptions and then apply standard text based information retrieval techniques to those transcripts [19]. This LVCSR approach to SDR raises the issue of identifying new words that are not a part of the large vocabulary recognition system. Regardless how large the vocabulary of the recognition system is, there will always be new terms that occur in the input audio stream. Such words in the audio stream that are outside the particular speech recognition vocabulary used are called out of vocabulary (OOV) words. Woodland et al. [19] have investigated the effects of OOV words in spoken document retrieval. The speech recognizer interprets each audio segment as a sequence of items from the recognition vocabulary. The recognizer will make at least one word error whenever an OOV word is in the input audio stream. Hence, OOV words are actually replaced by alternatives that are most probable based on the acoustic model and the recognition language model of the recognizer. Size and nature of the vocabulary recognizer is hence an important factor to the recognizer

performance. One way to minimize the expected OOV rate is to choose the vocabulary by taking the most frequent word forms from a large text corpus. Other researchers [21] also indicate that long documents alleviate the expected problems due to "out of vocabulary" and otherwise misrecognized words.

2.3 Information Retrieval

Information Retrieval (IR) [8] is mainly concerned with the identification of information sources that are related to a user's request. Information retrieval is different from information extraction, in which content is derived directly from the information sources, and question answering, in which a user's question is answered based on knowledge of information [8]. Of the three types mentioned above, information retrieval is the easiest to generalize, since knowledge of the content is unnecessary [7]. Identification of relevant segments of speech in a large corpus of digitized audio is different from the task of finding a relevant document among several text documents. Hence, accurate identification of the specific location of a word or phrase from a large corpus of XML index files might require better search tools than the regular text-based search engines.

File system search engines index the file system ahead of time and keep an index that, for every term that appears in the file system, contains a list of all files with the relevant term and the exact positions of it within the files [9]. Thus, usage of the file system search engine technique for the XML index files seems to provide a better retrieval interface for the audio indexing system in question.

The iTalk system was developed with Wumpus search engine tool [*] [38] as the retrieval interface. Wumpus is a file system based search engine developed by Stephen Büttcher under the supervision of Prof. Charles Clarke from the Information Retrieval Group at University of Waterloo. Wumpus search engine indexes the content of all files in the file system and, upon a query, returns all the indices that correspond to the query keywords. The index positions that rank highest among all the indices based on BM25 technique [32] will be returned to the user in the decreasing order of the ranks.

2.4 Audio Conferencing

Audio conferencing is a widely used distance collaboration tool [10]. It is quite economical and is used in several formal business settings. Audio conferencing involves real time dialogue exchange among the conference participants and is one of the current forms of communication among major professional groups and teams. Researchers have been working on improving the audio conferencing experience leading to effective communication by creating new forms of audio conferencing tools. Vocal Village (VV) [36] is a result of such research by Prof. Mark Chignell and his team from the University of Toronto. This system uses VoIP technology to connect collaborative groups. Unlike other VoIP applications, the

[*] More information about the Wumpus Search Engine is available at www.wumpus-search.org

VV system enhances traditional audio conference environments by binaurally presenting auditory location cues that cause the voices of individual participants to appear as if they are coming from different positions in space, distributed from left to right around the listener's head. This spatialized presentation format closely simulates the acoustical qualities of face-to-face collaboration, allowing VV users to feel as if they are meeting in a common physical location. Figure 6 [10] presents the graphical user interface of the VV audio conferencing tool.

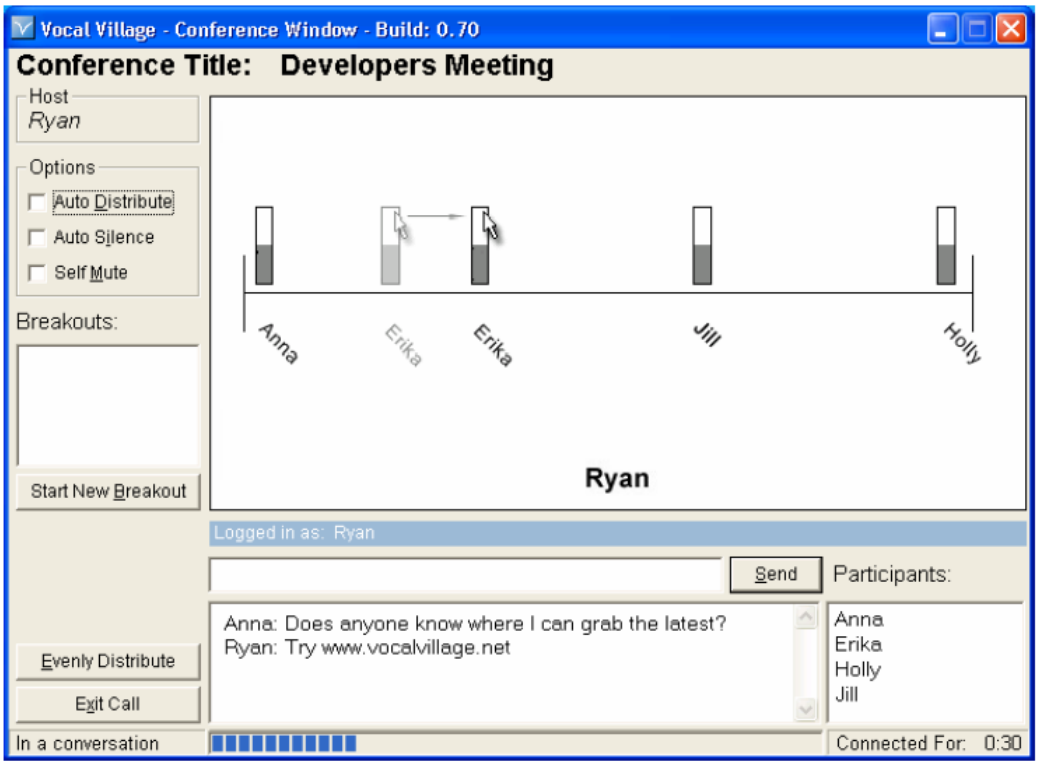

Figure 6: Sample screen shot of vocal village voice conferencing tool [10].

The graphical user interface for VV displays contextual information for the users, showing who is present in the audio space and where they are located, as well as providing a visual cue when participants enter and leave the conference [*] [10]. The apparently new technology in audio conferencing with added quality benefits and ease of use perfectly fits the iTalk requirements. Hence, the iTalk system is developed with Vocal Village as the audio conferencing tool at the user's end. Vocal Village generates a high quality raw audio file for each participant at the end of the conference that is further processed by iTalk tool kit to be indexed for accurate retrieval of audio snippets from within the audio file.

[*]More information on vocal village VoIP conferencing tool is available at www.vocalvillage.net

2.5 Related Work in Audio Indexing Systems

The iTalk tool that will be discussed in detail in Chapter 3 is not the first of its kind that performs indexing of audio information. The main research effort towards spoken document retrieval (SDR) was the SDR-TREC evaluations [22]. Recently, there seems to be a lot of research in this direction outside the SDR-TREC as well. Several approaches have been developed for spoken document indexing and retrieval for different applications. Some of those pioneering and marking approaches that perform audio indexing and searching are described in this section. These audio indexing systems are not intended to support voice conferences; however, they are aimed at effective skimming through audio information obtained from different sources.

2.5.1 TREC Spoken document retrieval efforts

The Text Retrieval Conference (TREC) SDR track has provided an infrastructure for development and evaluation of SDR technology [22]. SDR is accomplished by using a combination of automatic speech recognition and information retrieval technologies.

TREC 6 1997 Spoken Document Retrieval Track – The goal of TREC 6 SDR *known-item retrieval* task is to generate a single correct document for each topic rather than a set of relevant documents as in an ad-hoc task. The TREC 6 SDR process is presented in Figure 7 [39]. The evaluation of SDR technology [22] showed that relatively good known-item retrieval could be achieved for a small collection of broadcast news spoken documents. However, retrieval performance or recognition accuracy could not be analyzed due to the small corpus used for testing.

TREC 7 1998 Spoken Document Retrieval Track – The goal of TREC 7 SDR was to evaluate ad-hoc relevance retrieval from a much larger corpus than the 1997 SDR task. The typical SDR process was defined from these proceedings, see Figure 8 [22]:

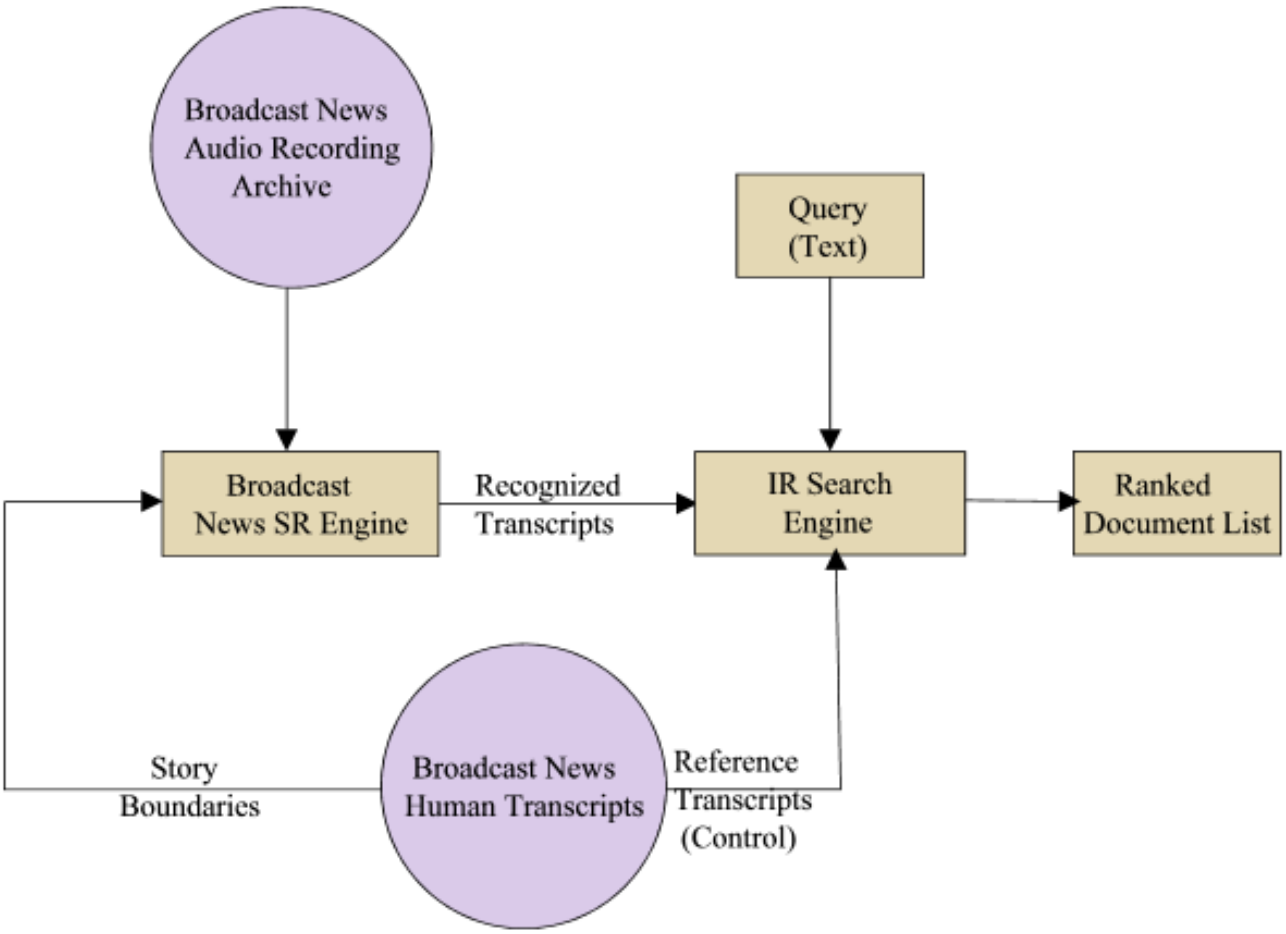

Figure 7: TREC 6 SDR process [39].

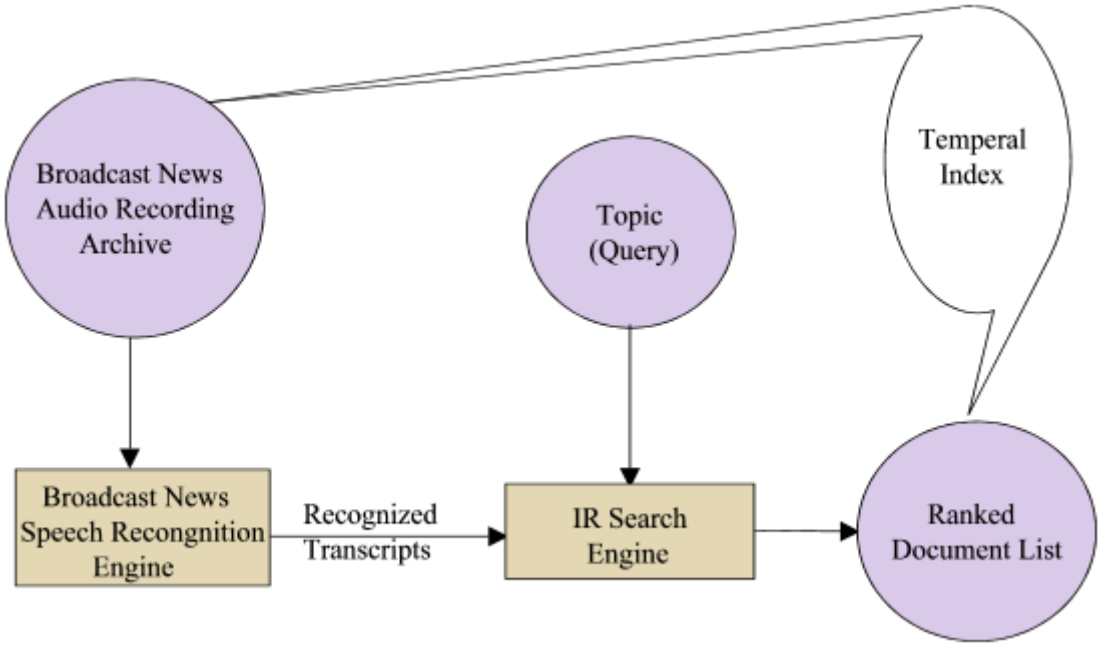

Figure 8: A typical SDR process [22].

A speech recognizer is applied to an audio stream and a time-marked transcription of the speech generated. The transcription may be phone or word-based in a lattice (probability network), n-best list (multiple individual transcriptions) or more typically a 1-best transcript (the most probable transcription as determined by the recognizer). The transcript is then indexed and searched by a retrieval system [22]. The results retrieved for a query is a list of temporal pointers to the audio stream ordered by decreasing similarity between the content of the speech being pointed to and the query.

The evaluations of TREC 7 revealed that the best performance for retrieval using a speech recognizer (.5120 MAP) approached that of retrieval using perfect human-generated reference transcripts (.5668) [40]. It is estimated that there is a near-linear relationship between recognition word error rate and retrieval performance. Named-entity word error rate had high correlation with retrieval performance than mere word error rate. These evaluations are the basis for spoken document retrieval and many research efforts evolved from these results.

TREC 8 Spoken Document Retrieval Track – [26] The 100 hours of TREC-7 data with 2866 stories and 23 queries limited the conclusions that could be drawn about SDR and hence a larger collection was needed to confirm the results. TREC-8 provided 500 hours of audio data, 21,574 stories and 50 queries. The TREC-8 SDR task contains two main tasks [26]:

1. Story-Known (SK) SDR with audio from American broadcast radio and TV news programs along with manually generated story boundaries. Participants must

present a list of relevant documents and their ranking for a given natural language text query, by running a recognizer and retrieval system on the audio data.

2. Story – unknown SDR with no knowledge of story boundaries at both recognition and retrieval time. Endpoints of the shows are given with start time of "first" story and end time of the "last" story. Retrieval would then produce a ranked list of shows with time stamps that are mapped to their corresponding story identifiers based on the standard scoring procedure.

TREC 8 results [22] show that the SDR technology is robust to handle large spoken document collections. Once again, the TREC-8 track results revealed that there is a near linear relationship between recognition errors and retrieval accuracy and that the degradation of retrieval performance for increased recognition errors is relatively gentle. TREC-8 made way for further experimental research in conditions where story boundaries are unknown and that was taken up in TREC-9.

Though TREC-SDR track experiments have been marking in the field of SDR, all the TREC experiments applied SDR technology within the broadcast news domain. Broadcast news domain is quite different from conversational speech; consequently, TREC SDR experimental systems might not provide effective results in the conversational speech domain.

2.5.2 Tools for automatic transcription and browsing of audio-visual presentations

Cyphers, Hazen and Glass [5] developed a tool for web-based transcription service to which users can upload their audio files for automatic transcription and indexing. As a part of this MIT Spoken Lecture Processing Tool, Glass et al. [15] have developed a server and an accompanying lecture browser for the users to locate and browse audio segments appropriate to their query. Personal digital audio recorder iRiver N10 [41] was the source for collecting audio data in this experiment [15].

The key steps in the transcription process are:

1. Language Model Adaptation - Adaptation of a topic independent vocabulary and language model using the supplemental text data provided by the user.
2. Segmentation - Automatic segmentation of audio files into 6 to 8 second chunks of speech separated by pauses.
3. Automatic annotation of audio chunks using a speech recognition system.

Language Model Adaptation- In this process, the vocabulary of any supplemental topic-dependent texts provided by the user is extracted and added to the topic-independent vocabulary. The recognizer would then combine the topic independent word sequence statistics from the corpus of existing lecture material with the topic dependent statistics of the supplemental material to create a topic-adapted language model.

Segmentation- In this process, the audio file is arbitrarily broken down into ten second chunks and is processed with an efficient vocabulary and a speaker independent phonetic recognizer. Contiguous regions of speech are identified from the phonetic recognition output. The segmentation process constructs 6 to 8 second audio segments of speech from these contiguous regions. Speech is broken down into 6 to 8 second segments, as experiments have shown that longer segments tend to be computationally burdensome for searching audio files. At the same time, shorter segments are harder to recognize accurately as they provide less context information to the language model. Further, language model adaptation is performed with supplemental topic dependent vocabulary being added to topic-independent vocabulary. SUMMIT speech recognition system [42] is used to transcribe the segmented audio.

The audio data is then indexed based on the recognition output in preparation for availability on browsing by the user, as shown in the Figure 9 [5].

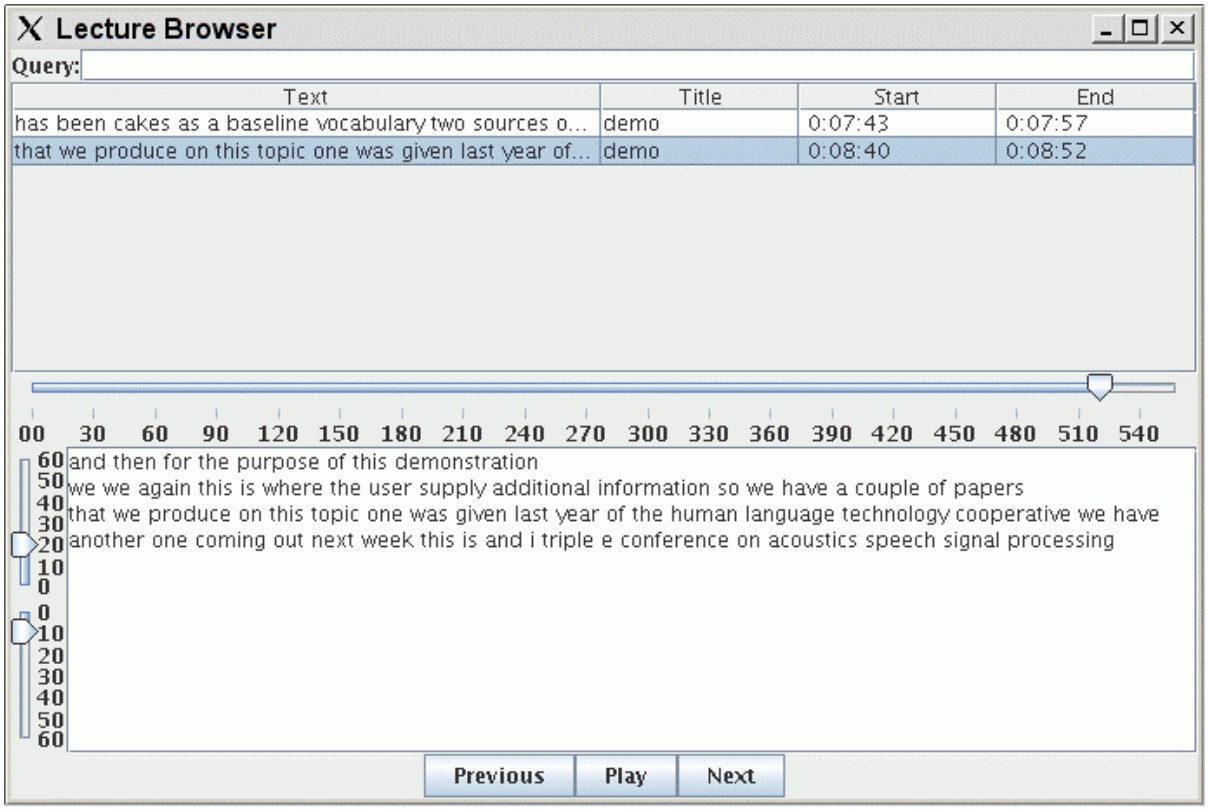

Figure 9: Sample screen shot of the lecture browser [5].

The lecture transcriptions are split into segments separated by two or more seconds of non-speech segments, as determined by the recognizer. Lucene text search engine [43] library is used to index the segments with additional support from Berkeley database containing information such as timing for the segments and the words within the segments [5].

The browser presents the list of hits within the indexed lectures for a given text query. When a hit is selected, it is shown in the context of the lecture transcription. User can then adjust

the duration of the context preceding or following the hit, navigate to and from the preceding and following parts of the lecture, and listen to the displayed segment.

Glass et al.'s [15] experimental results that are based on 10 computer science lectures present that:

1. Retrieval of short audio segments containing important keywords and phrases can be reliably obtained even with high word error rates.
2. The language model adaptation process of adding spontaneous speech data to subject-specific written materials increases transcription accuracy but has a marginal effect on the retrieval performance [14].
3. Good audio IR results can be obtained even with high errors in the transcriptions, given that the relevant keywords are reliably detected.
4. When the vocabulary is not exhaustive, reducing the OOV word rate by increasing the word size improves recognition accuracy and precision, but reduces recall.

As mentioned above, this project is aimed at processing the lecture speech. Despite the similarity with the conversational speech in terms of spontaneity, lecture speech involves a small topic dependent vocabulary that is rarely used in every day conversations. Hence, the language adaptation model, as performed by Glass et al. [15], though appropriate for lecture speeches, might not be effective for human to human conversations. Human to human conversations might not only be boundless in terms of the topics discussed, but also in terms of vocabulary being used.

2.5.3 Indexing spoken documents using self-organizing maps

Kurimo [11] created a searchable index for spoken audio documents by selecting the best index terms for each document with the help of the other documents close to it using a semantic vector space. First, the audio stream is converted into a text stream by a speech recognizer. Then, the text of each story is represented in a vector space as a document vector, which is a normalized sum of the word vectors in the story. A large collection of such document vectors is used to train a self-organizing map (SOM) [11] to find latent semantic structures in the collection. As the stories in spoken news are short and normally include speech recognition errors, smoothing of the document vectors using the semantic clusters determined by the SOM is used as a way to enhance the indexing. Kurimo's application is the indexing and retrieval of broadcast news on radio and television.

2.5.4 An integrated automatic transcription and indexing tool for spoken medical reports

Happe et al. [12] developed an integrated tool for automatic transcription and indexing of spoken medical reports using the NOMINDEX program [44] and the commercially available speech recognition tools: Naturally Speaking 5 [45] and IBM VIA VOICE 8 professional

[46] versions. NOMINDEX [44] is an indexing tool that extracts concepts from medical text in natural language and is based on the MeSH French medical lexicon thesaurus [47]. This indexing tool also has the matching process that extracts thesaurus concepts in a sentence. In this section we present the method and evaluation of the tool developed by Happe et al. [12].

Method - For each document, the automatic concept extraction process consists of three steps:

1. Dictation and digital audio recording.
2. Speech Recognition and
3. Automatic Indexing

Figure 10 [12] provides an overview of the method used in this experiment.

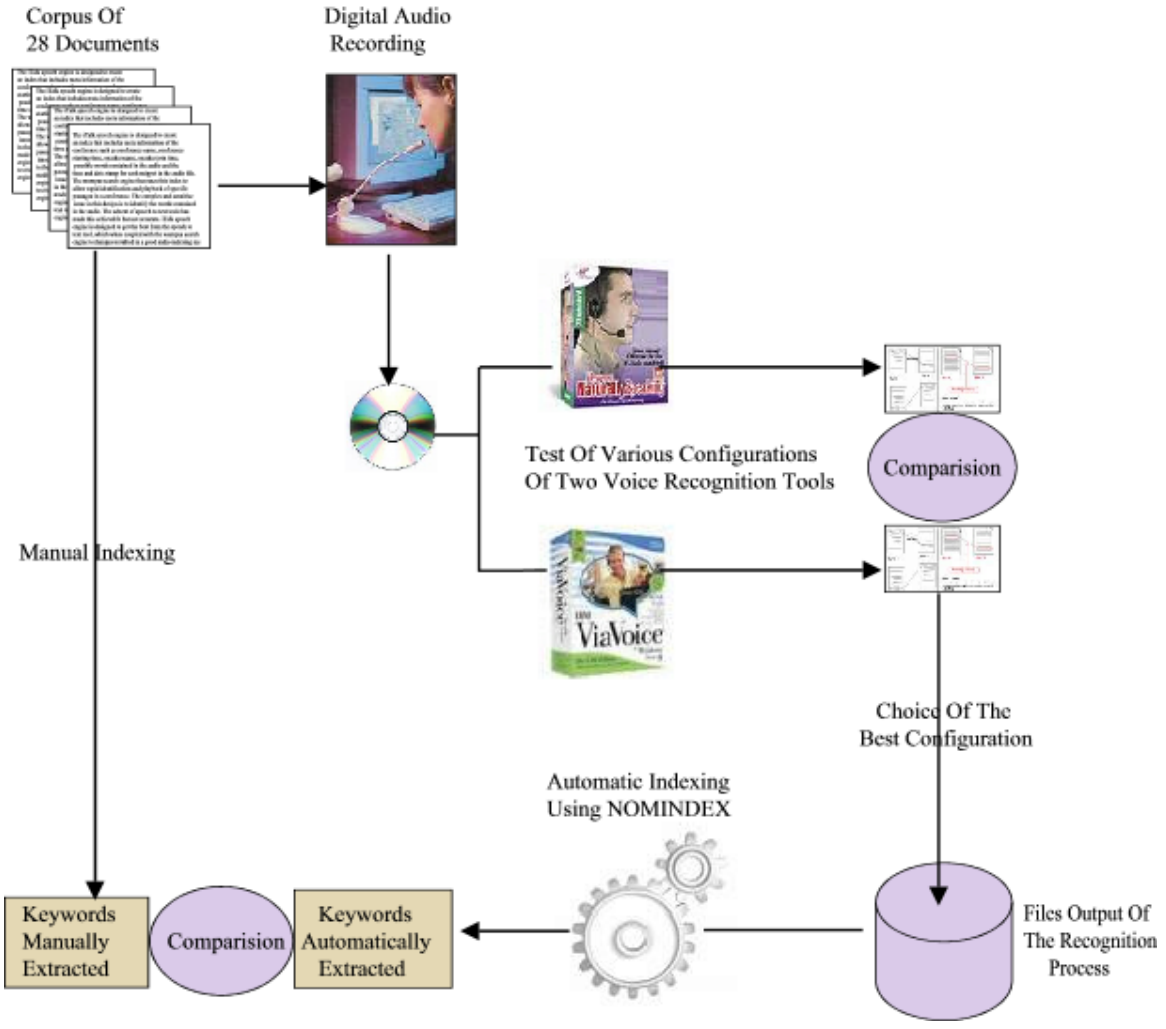

Figure 10: Overview of the automatic concept extraction study process [12].

For this experiment, Happe et al. [12] have used the MENELAS Patient Discharge Summaries (PDSs) [13] as the audio source. Most patient relevant medical information is generally stored in narrative format in the PDS. These PDSs would contain minimal yet sufficient information to be used with the following visits of the patient. MENELAS [13] has the audio version of these PDSs, which are dictated by a secretary and stored on the computer.

A single operator recorded the PDSs on CD-ROM. The audio PDSs are run through both the Naturally Speaking 5 [45] and IBM VIA VOICE [46] speech recognition tools as per the initialization phase requirements for each of the tools. The files obtained from the best configuration of speech recognition process are further processed by NOMINDEX program [44] to generate keywords/ index terms.

Evaluation – A comparison was done between the set of concepts extracted by NOMINDEX [44] after the speech recognition phase and the set of keywords manually extracted from the initial document. The method was evaluated on a corpus consisting of 28 hospital medical reports corresponding to in-patients admitted for Coronarography. The same corpus was used for testing speech recognition and indexing.

- Speech Recognition: Two speech recognition packages, i.e. Naturally Speaking 5 Professional [45] and IBM via voice 8 Professional [46], were tested to measure their relative performances and evaluated against their ability to recognize words.
- Indexing: For each document, a set of keywords was manually extracted by one of the authors. Automatic indexing was performed by the NOMINDEX program [44] on the transcriptions generated from the speech recognition tool.

From Happe et. al [12] evaluation of the speech recognition tools and the effect of speech recognition errors in audio indexing, one can assume that:

- "Misspellings" would not affect the concept of the documents and can be easily corrected with a spellchecker.
- Contexts cannot always be used for disambiguation; therefore, homophone disambiguation might not be helpful.
- Word sense ambiguity needs to be improved for automatic concept extraction to be successful.
- The number of indexing errors resulting from speech recognition errors is low. Consequently, combination of speech recognition and automatic indexing techniques should be beneficial.

In the experiment, Happe et al. [12] employed a corpus of 28 dictated documents. This experiment is limited, since the data is recorded by a dictation specialist/secretary rather than a normal user/physician. This difference in the recording method is worth noting, as the dictation from a written document is similar to carefully spoken speech, i.e. broadcast news. Spontaneous dictation, on the other hand, has more similarities with the conversational speech in that both types contain spontaneous speech effects such as word contractions, extraneous filler words, partial words, and false starts. Further more, any person with clear speech habits can get 97% recognition accuracy without much training with the speech recognition system. However, the same system might have comparatively lower recognition accuracy resulting from speech habits of the speaker regardless of the amount of training. Speech habits are continuously formed and refer to a certain pattern of speech frequently used by the speaker. Ex: – fast, slow, long pauses, short pauses, etc. On the contrary, iTalk captures the audio from normal spoken conversations by any user. Consequently, the recognition accuracy is affected. iTalk has an integrated search feature, which enables retrieval of contents of the audio file based on the search query. Happe et al.'s [12] experiment was focused primarily on comparison between the manually extracted keywords with the automatically generated index terms. Despite the found differences and similarities, Happe et al.'s [12] experimental results strengthen the motivation for tools similar to iTalk.

2.5.5 A spoken query information retrieval system

Garza et al. [16] proposed architecture for a spoken information retrieval from multimedia databases. The architecture focuses on information retrieval from multimedia documents, using spoken queries and user interface.

Being based on client server architecture, the spoken query information retrieval system is shown in Figure 11 [16].

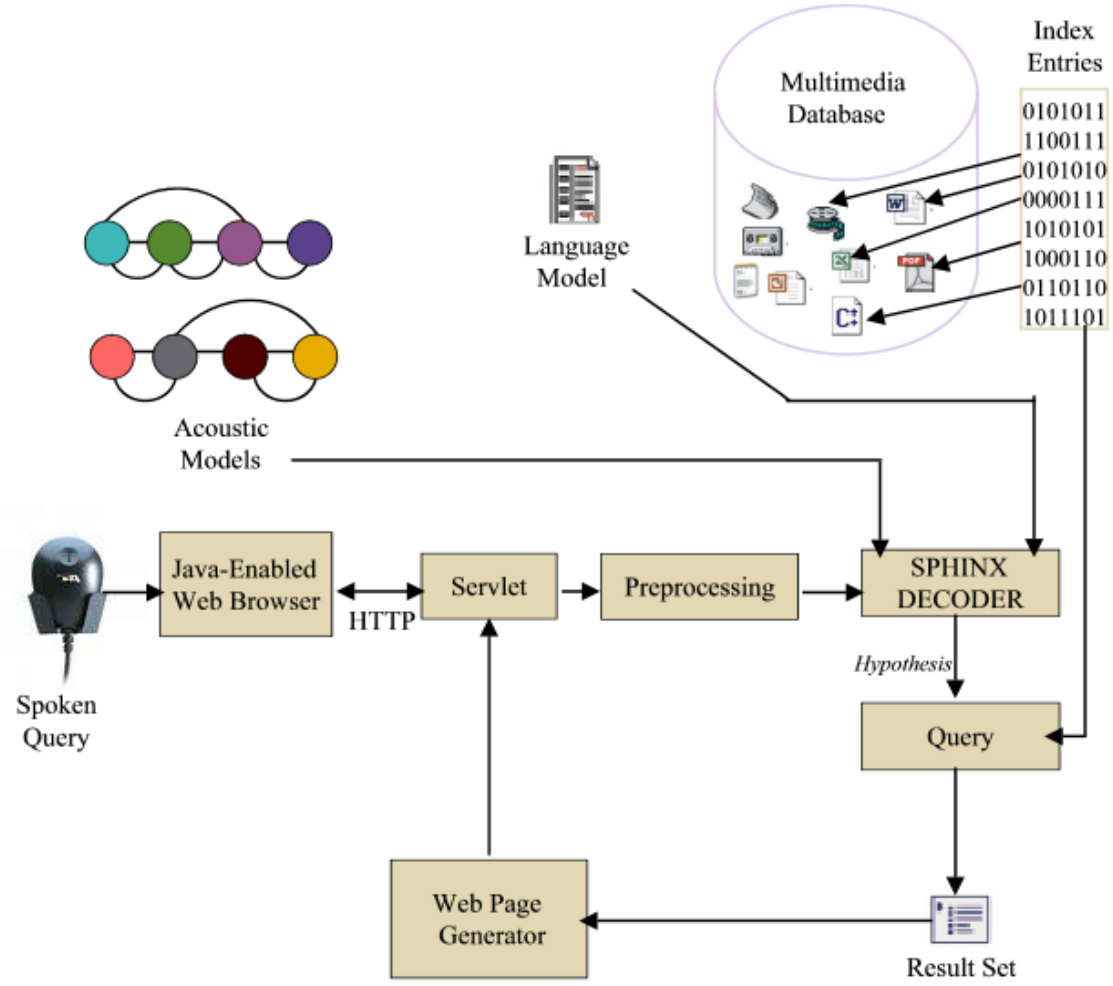

Figure 11: General spoken query information retrieval system architecture [16].

The system has a Java enabled front end that captures acoustic information from the query. This acoustic information is then transmitted to a server-side servlet that invokes the audio preprocessing module and is then forwarded to the speech decoder. A query system, Managing Gigabytes (MG) [48], that takes the CMU SPHINX III [49] system decoder hypothesis, was generated using the acoustic and language models obtained from the training stage. The Input produces a set of pointers to the documents in the multimedia database, in which the queried words were found. The results obtained are presented to the user on a web page, which includes links to document segments or audio/video file frames.

The multimedia database used in this architecture is a set of indexed documents such as plain-text, MS-Word, MS-PowerPoint, PDF, and audio/video files. Non-text documents are preprocessed to produce a raw transcription that can be added to the indexed archive. In general, the database contains two file categories: text-sourced and audio-sourced. In the preprocessing stage, plain text is extracted from text-sourced files by ignoring images, fonts,

document style formats, and special characters. For audio-sourced files, i.e. audio and video files, the preprocessing stage involves automatic speech recognition to extract the audio transcriptions as shown in the Figure 12 [16].

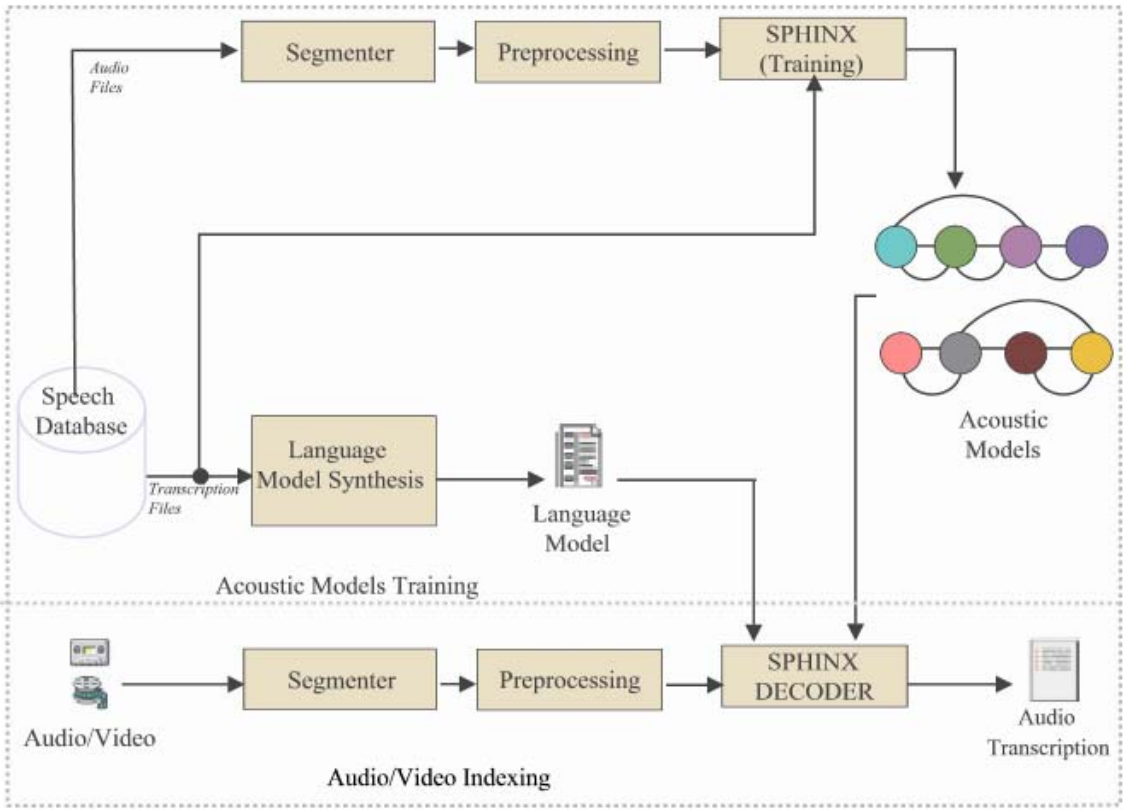

Figure 12: Audio/video file preprocessing using a speech decoder, acoustic and language models [16].

The audio-sourced files preprocessing involves generation of speech segments of the acoustic signals using the CMUseg tool [50]. These speech segments are preprocessed to generate static and dynamic information condensed as a feature vector, which is given as input to the SPHINX III [49] decoder to produce the best transcription for the audio signal. Training of the acoustic models was performed using a database of several hours of audio produced by different speakers. Finally, the audio transcription files are indexed using an inverted index based text retrieval system, Managing Gigabytes [48].

From the results obtained, Garza et al. [16] proposed that the integration of speech recognizer, text based document indexing and retrieval tools comprise an effective architecture for a spoken query system on multimedia databases.

2.5.6 Acoustic indexing via phone lattices

Brown et al. [17] presented an audio indexing approach that uses an offline HMM system. This offline HMM system generates a number of likely phone sequences, which may then be rapidly searched to find phone strings that consists the desired word. Audio indexing using phone lattices involves three stages:

1. Model training
2. Lattice generation
3. Lattice scanning

Model training - Using a text transcription and dictionary, a phone sequence is generated for every training utterance. HMM parameters were estimated using the phone sequences. Speaker-dependent models were trained using a corpus containing:

- 20 natural speech messages ("p" data): Responses obtained for 20 unique voicemail prompts.
- 77 read sentences ("r" data): Sentences constructed to contain a keyword minimum of five times.
- 170 keywords ("i" data) spoken in isolation.
- 150 read sentences ("Z" data): phonetically-rich sentences from the TIMIT corpus [51].

For speaker-independent acoustic models, additional training data is obtained from the WSJCAM0 British English corpus [52] that consists of spoken sentences read from the Wall Street Journal.

Lattice generation - Given unknown acoustic data, it is possible to find the most likely sequence of phones using a set of phone HMMs and a network of possible phone interconnections. An enhanced Viterbi algorithm [17] is used to generate multiple hypotheses, representing the *n* most likely paths through the models. The multiple hypotheses can be stored as a directed acyclic graph, in which nodes represent phone start/end times; whereas arcs represent hypothesized phone occurrences between the nodes. This graph is referred to as phone lattice. Arcs are labelled with an acoustic score. Acoustic score indicates the likelihood of correspondence between the acoustic data over a given interval with the particular phone model. An additional constraint is added to the phone lattices: all paths must start at a particular node and end at another node. The "depth" of a phone lattice is the number of phone hypotheses that are active at a given point in time. A shallow lattice leads to

poor performance resulting from unavoidable phone recognition errors. Considering that the best phone recognition systems exhibit accuracy levels of approximately 70% , even a smaller depth, such as 1-deep lattice, has a poor chance of correct identification of a given phone string. Furthermore, storage requirements and search time increase substantially with lattice depth. At any given point in time during lattice generation, the number of active tokens represents the *n* best paths through the model lattice. The more tokens are being used, the deeper is the lattice. Figure 13 [17] represents a lattice generated for the single utterance "damage".

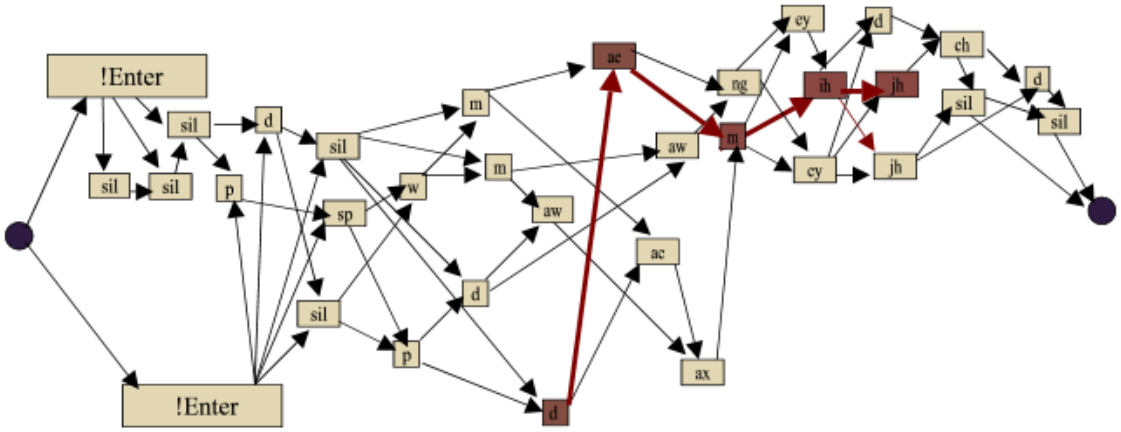

Figure 13: Phone lattice for the word "damage" (d ae m ih jh) [17].

Lattice scanning - The generated phone lattices are scanned for a phone sequence composing the given word. An acoustic score is computed from this phone sequence as the sum of the phone arc scores normalized by the best-path score. Deep lattices have different paths starting or ending within a few milliseconds and resulting in many hypotheses for a given word. In the case shown in Figure 13, there are two possible paths for the sequence d ae m ih jh. The best scoring hypothesis is then used as the resultant hypothesis.

The phone accuracy [17] is defined as: *"the ratio of correctly recognized phones, minus deletions and insertion, to the actual number of phones for the best path through the lattice*". For word spotting, an acceptable figure of merit is defined [17] as: "*The average percentage of correctly detected words as the threshold is varied from one to ten false alarms per word per hour*". Word spotting errors affect both speech recognition and information retrieval performance. The lattice scan approach uses lattices of depth necessary for identification of longer words and results in larger number of false alarms for shorter words. The lattice scan method is not robust enough for short words.

Brown et al. [17] conducted experiments on the Wall Street Journal WSJCAM0 read-speech corpus [52]. The results indicate that natural speech corpus is more difficult to recognize than the read-speech corpus.

2.5.7 Spoken document indexing and search using PSPL

Chelba and Acero [30], in their study *Position Specific Posterior Lattices (PSPL) for indexing speech,* have proposed a new representation of automatic speech recognition lattices that naturally lend themselves to efficient indexing of position information and subsequent relevance of spoken documents using proximity. The PSPL is a lossy but compact representation of a speech recognition lattice that lends itself to the standard inverted indexing conducted in the text search: position as well as other contextual information for each hit is retained.

The position information needed for recording a given word is not readily available in ASR lattices. However, ASR lattices do contain the data needed to evaluate proximity information. On a given path through the lattice, position index to each link/word is assigned in a regular way. Each path occurs with a given posterior probability that is easily computable from the lattice. Soft-hits [30] that specify the tuple <*document id, position, posterior probability*> for each word in the lattice are then indexed.

A simple dynamic programming algorithm [30] which is a variation on the standard forward-backward algorithm is employed for computing the sum of overall possible paths in a lattice that contains *a* given word at *a* given position. In *N*-gram lattices where $N \geq 2$, all links ending at a given node *n* must contain the same word *word(n)*, so the posterior probability of a given word *w* occurring at a given position *l* is calculated using the following formula:

$$P(w,l \mid \mathrm{LAT}) = \sum_{n\ s.t.\ \alpha_n[l]\cdot\beta_n>0} \frac{\alpha_n[l]\cdot\beta_n}{\beta_{start}} \cdot \delta(w, word(n)) \quad [30]$$

The position Specific Posterior Lattice is a representation of the $P(w,l \mid \mathrm{LAT})$ distribution: for each position bin *l*, the words *w* are stored along with their posterior probability $P(w,l \mid \mathrm{LAT})$.

For indexation of the speech content, speech files are first segmented and for each segment a corresponding PSPL lattice is generated. Each document and each segment in a given collection are mapped to an integer value using a *collection descriptor file*, which lists all documents and segments. Each soft hit will store the PSPL position and posterior probability. The relevance ranking scheme for a given Query $Q = q_1 \ldots . q_i \ldots . q_Q$ and document D represented as PSPL is described as follows [30]:

For all query terms, a 1-gram score is calculated by summing the PSPL posterior probability across all segments s and positions k. The results are aggregated in a common value $S_{1-gram}(D,Q)$ [30]. Logarithmic tapering off is used for discounting the effect of large counts in a given document. Similar to 1-gram, an expected tapered-count is calculated for

each N-gram in the query and the results are aggregated in a common value $S_{N-gram}(D,Q)$ [30]. By taking the inner product with a vector of weights, the different proximity types, one for each N-gram order allowed by the query length, are combined:

$$S(D,Q) = \sum_{N=1}^{Q} w_N \cdot S_{N-gram}(D,Q)$$

Only documents containing all the terms in the query are returned. Further more, Chelba and Acero [30] indicate that the transcription for any given segment might also be represented as a PSPL with exactly one word per position bin and the relevance scores calculated correspond to the ones specified by TF-IDF score for the query.

2.5.8 SpeechLogger: A coversational speech retrieval system

SpeechLogger [18] is a system that enables search from spoken communications such as teleconferences and telephone conversations. Begeja et al. [18] focused on improvement of the navigational interface for better information retrieval of spoken documents. The possibility to improve the ASR system has not been considered, given that the data quality of conversational speech audio files is not always sufficient for production of objective ASR results.

Begeja et al. [18] have defined four interface elements that help in providing a better navigational system for spoken documents:

1. A timeline that represents every search hit in a spoken document with a tick mark. Tick mark, provides a better identification of false positive documents.
2. Keyword extraction that summarizes conversation and presents subtopics in large spoken documents thereby providing a means to differentiate among the search results.
3. Speaker segmentation and speaker identification that splits the spoken document into meaningful chunks of audio information.
4. Lattice search and phoneme search aimed at increasing the possible search space.

Figure 14 [18] provides an overview of the SpeechLogger system.

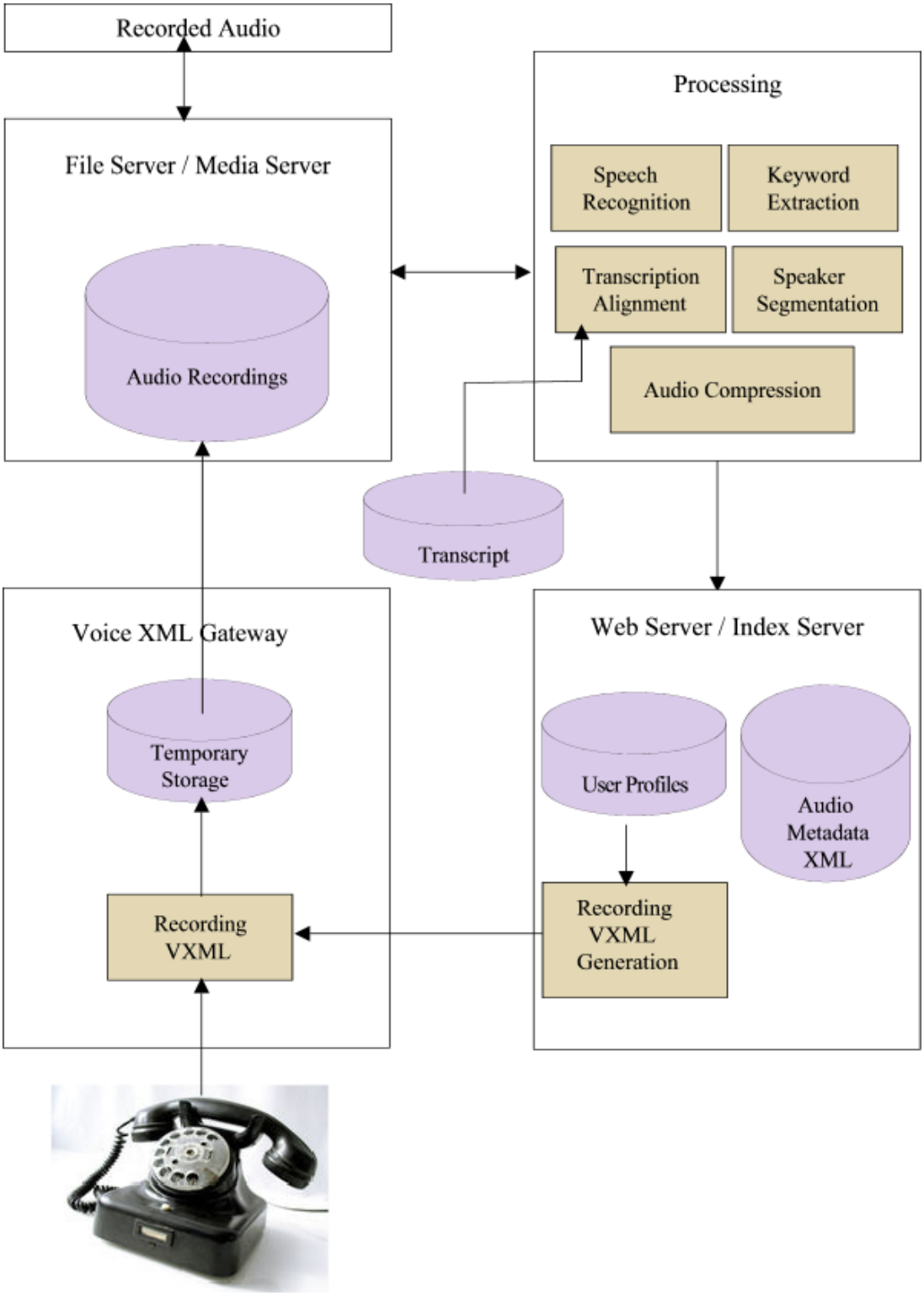

Figure 14: Overview of the SpeechLogger system [18].

As shown in the Figure 14, for every audio recording that is added to the File Server, SpeechLogger performs speaker segmentation, speech recognition, transcription alignment, keyword extraction, audio compression, and speech indexing.

- Speaker segmentation – This component separates the speakers from each other.
- Speech recognition – This component converts the audio into a word or phone sequence including alternate hypotheses in the form of lattices.
- Transcription alignment – This component aligns the corresponding transcripts for an audio file, if available, with the speech recognition output at a given point in time in the audio file.
- Keyword extraction – This component extracts the most salient words found in the speech recognition output (one-best word) or transcript. These keywords can be used to determine the nature of spoken communications.
- Audio compression – This component compresses the audio file and creates an MP3 audio file, which is copied into the Media Server.
- Speech Indexing – Text and lattice indexing is performed in this phase. Indices are created based on word and phone lattices or one-best word and one-best phones. User can browse and search the audio using either text index or lattice index. The audio is played back via media streaming or over the phone using the Voice Genie VoiceXML gateway [53].

In assessment of the effectiveness of the techniques featured above three different corpora were used: the DARPA broadcast news corpus, the switchboard corpus consisting of the two-way telephone conversations, and a corpus of multi-party teleconferences on various topics.

Begeja et al. [18] point out from their assessments that broadcast news data performs well at speaker segmentation where teleconferences data performs poorly. This performance difference can be attributed to the high audio quality and well controlled structure of the broadcast news program, in comparison to poor audio quality of teleconferences that contain a lot of spontaneous speech segments, which are less than 1 second long, such as “umm”, “ah”, “no”, etc. These characteristics make the teleconference data the most challenging for speaker segmentation.

ASR performance is evaluated based on word error rate and out of vocabulary word rate. The performance of ASR on broadcast news, switchboard corpus, and the teleconference corpus is presented in Table 1 [18]:

Corpus	Word Error Rate	Out of Vocabulary word rate
Broadcast News	~20%	0.6%
Switchboard	~40%	6%
Teleconference	~50%	12%

Table 1: ASR performance assessment for SpeechLogger system [18].

From the results summarized in Table 1, it is evident that both the word error rate and out of vocabulary word rate are high in the teleconference corpus. The reason can be attributed to the environment of the teleconferences, as they involve multiple users and different kinds of equipment for recording the audio. This, in turn, results in poor audio quality and, consequently, a poor speech recognition output.

2.5.9 Summary of related work

Previous sections introduced several spoken document retrieval efforts: TREC evaluations are concluded briefly; different audio indexing approaches using self-organizing maps, phone lattices and position specific posterior lattices are introduced; automatic tools for spoken document indexing and/or retrieval, i.e. the MIT spoken lecture processing tool; Happe et al.'s [12] integrated tool for concept extraction from spoken medical reports; Garza et al.'s [16] spoken query information retrieval tool and SpeechLogger tools are presented. Other research fields include the NoTIme system [23] that uses strokes on a sketching application to index audio, the Filochat system [25] that uses the stroke indexing for notes taken at meetings, the Audio Notebook tool [24], which is a paper-based portable note taker that indexes audio, etc. We tried to give an insight into the traditional and novel spoken document retrieval approaches for recorded conversational speech.

There are three classes of spoken documents as presented by Begeja et al. [18]:

Broadcast news – Broadcast news has excellent ASR output, as there is only one speaker at a time, the audio quality is high and generally involves a good speaker. There are several research and commercial systems aimed at broadcast news or similar read-speech data.

Telephone Conversations – Telephone conversations have considerably good ASR outputs as they involve only two speakers and have an acceptable audio quality. Though researchers did examine telephone conversations, attention mostly has been paid at voice mail data.

Teleconferences – Teleconferences, unlike telephone conversations and broadcast news, have poor ASR outputs resulting from extensive noise involved: there are multiple speakers communicating simultaneously. This leads to poor audio quality.

As previously stated, there are several research and commercial spoken document retrieval systems, some aiming at broadcast news, read-speech recordings, some addressing voicemail recording corpus, and some working on telephone and teleconference corpuses. iTalk, in its turn, is aimed at VoIP conferences. Similar to teleconferences, VoIP conferences would be vulnerable to poor audio quality. Consequently, there is a distinct need for better systems and techniques to compensate for a relatively poor ASR output.

James Allan [21] in his research estimated that even with 40% error rate in ASR system, the effectiveness of an IR system falls less than 10%. Consequently, tools similar to iTalk, that work with a good ASR tool and apply a better indexing approach specific to the application enhanced by text-based search techniques, provide a more effective way for skimming through VoIP based audio conversations.

Chapter 3

Design

In this chapter, we describe the components and associated changes in their structure that are focused on integration and development of iTalk system. Further, we discuss the hypothesis for indexing recorded audio conferences and retrieval of specific snippets of the audio file containing the query keyword.

3.1 Preliminary Steps

The first step in the development of any system is to determine what is already available, what is missing and what needs to be done to fill in the gap. In this section, we focus our analysis on the above stated issues.

As mentioned in Chapter 2, any basic spoken document retrieval system involves 3 tools:

- Audio recording
- Speech recognition
- Search tool

Technological innovation contributed to formation of efficient tools in all the three domains mentioned above. However, to achieve the goal of spoken snippet retrieval from an archive of audio files, integration of these tools must be completed in a way that minimizes the possibility of data loss, whereas accuracy of the end audio snippet retrieved must be sustained at the same level as the original input audio recorded.

The easiest way to integrate the model is to upload the recorded audio file from the source recorder to the speech recognition engine. The transcript generated from the speech recognition engine is then forwarded to the search tool to be indexed for searching. Figure 15 presents the flow of information for integration. Such an index would be able to retrieve an audio file from several audio files; however, retrieval of only a part of an audio file from

several files is unattainable. The reason for this is rooted in the fact that the transcript of an audio file is what is being indexed. When a query keyword appears to be positively present in a transcript say T_a, the search tool would point to T_a, which would in turn link to the source audio file. Hence, the search tool retrieves the entire audio file for the user. Though good enough for small documents say 1 to 5 minutes in length, retrieval of an audio file say of 60 minutes in length is but an incomplete search, since the user has to manually skim through audio file of 60 minutes in duration, in order to locate the exact place where the keyword was spoken. And if the retrieved document was a false positive, it would end up in an ineffective search.

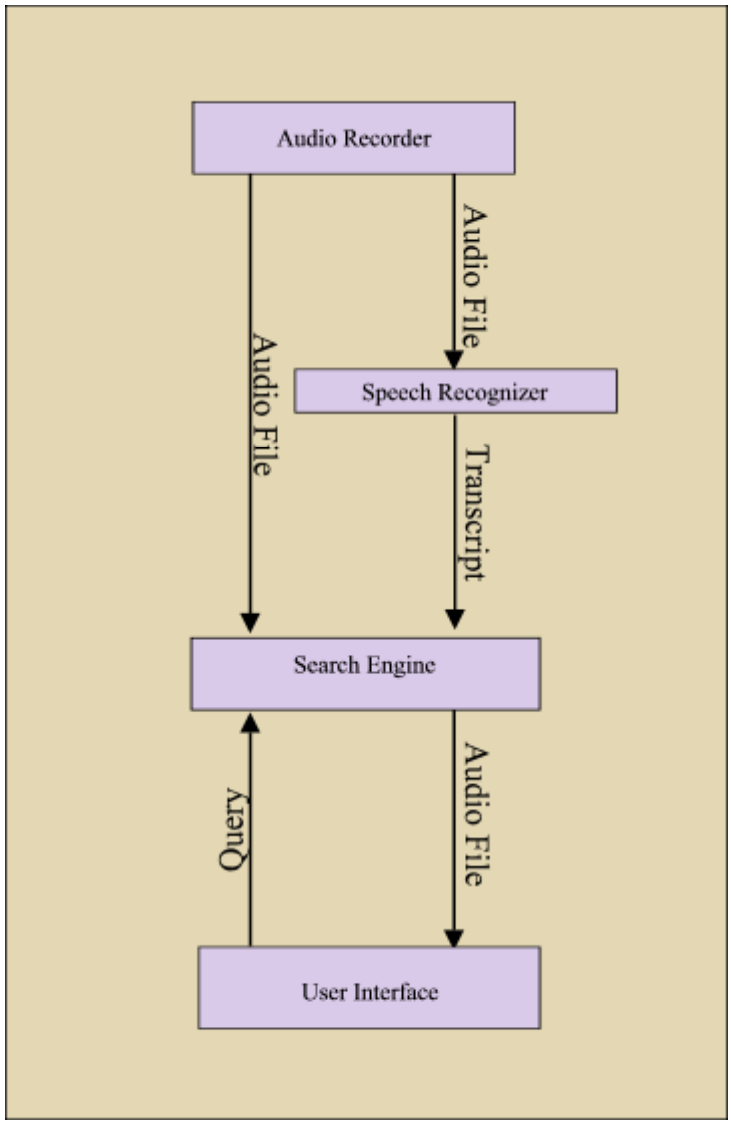

Figure 15: General SDR flow.

Moreover, the success of the system will be dependent on its ability to retrieve the specific position in an audio file from among several audio files, in which the text based query keyword appears.

Consequently, the basic need includes the following systems: recorder producing high quality audio files alongside with the speech recognition tool capable of generating time-stamped transcripts, which refer to specific positions in the audio file and a retrieval system, which returns relevant snippets as query results.

3.2 Recording high quality audio files

VV audio conferencing system, presented in section 2.4, has the capability to process good quality audio for broadcasting across the conference. A new version of this tool has been used: it saves the user's contribution to the conference on his/her system. Figure 16 presents the procedure. As evident from Figure 16, all clients participating in a conference communicate with a central VV server. The VV server enables the start and the end of a conference as well as the real time broadcasting of audio among the conference members or clients. As the conference continues, the client side VV software records the client's audio. At the end of the conference, recorded audio will be saved to a specific location on the client's system. Thus, at the end of each conference, all participants have their audio contribution to the conference stored on their systems in a specific location. As the audio is recorded and stored at the same location where it is generated, loss of audio data or quality is avoided. Thus, good quality audio recording is achieved. The achieved outcome is then used as the input for speech recognition engine.

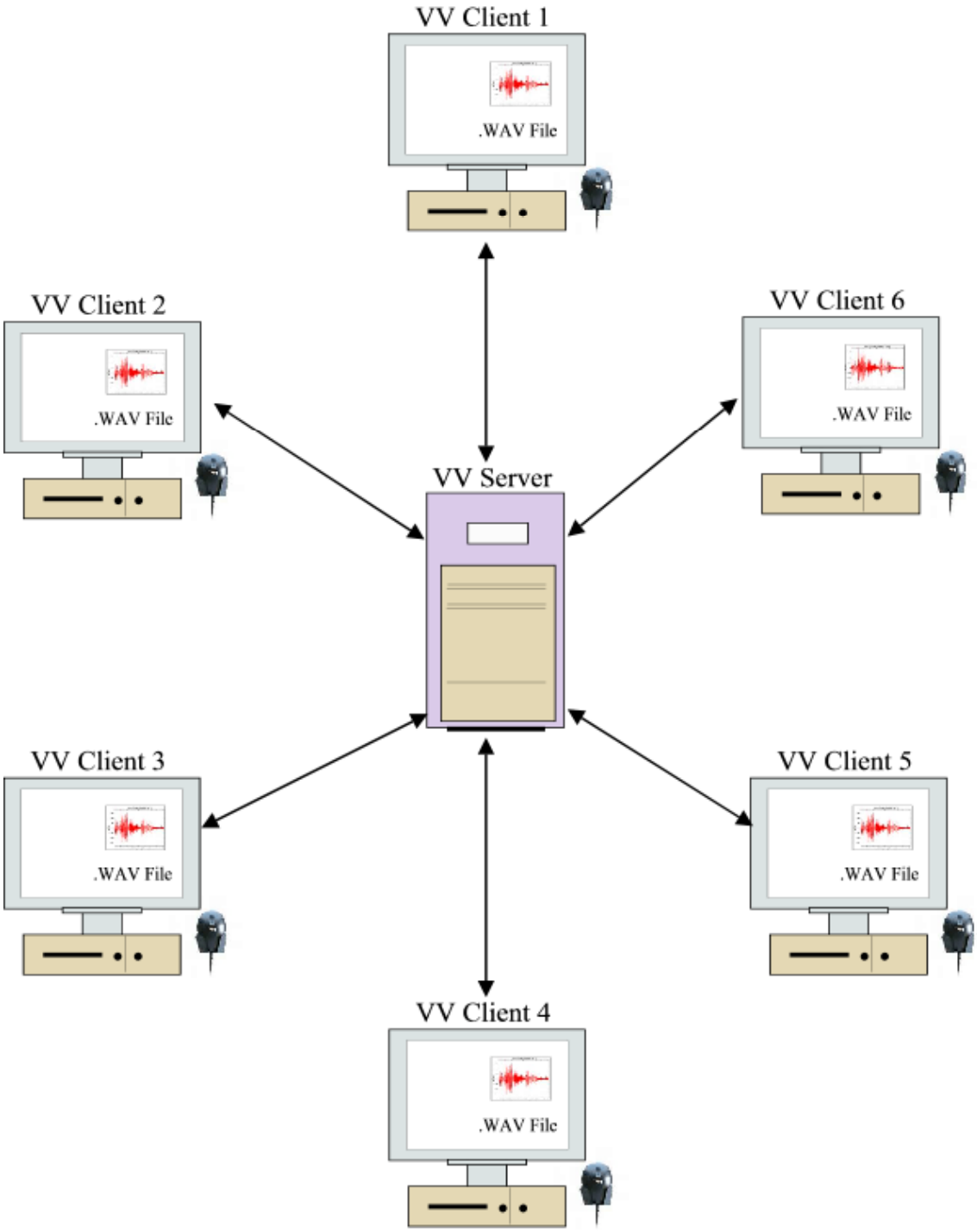

Figure 16: Overview of vocal village audio conferencing process.

Since the input audio quality is good, results of speech recognition will not be affected. Poor audio quality results in poor recognition accuracy, as it simultaneously increases the possibility for misrecognised words. Thus, ensuring good quality audio as an input helps in future processing of the audio file for better recognition accuracy, and high playback quality as well.

3.3 Speech recognition tool with time-stamped transcripts

Dragon Naturally Speaking 7.0 [45], here after referred to as DNS, is one of the leading commercially available software for speech recognition [34]. Having been long enough in the speech recognition market, the DNS has considerably evolved from a discrete speech recognition engine introduced in 1982 to a real-time continuous speech recognition system in 1997 that was further developed into a dictation system in 2001 followed by several enhancements that resulted in the current version, which is very competitive in the market of commercial speech recognition tools, aimed at providing powerful user-independent speech recognition solutions as well. Tools to create commands, i.e. macros, which are programming instructions for repetitive tasks and to control DNS, are also developed. The research in DNS extensions by Gould resulted in Natlink [27], a macro system for DNS. Natlink is a compatibility module that allows writing macros in python programming language [29] to control DNS. Also referred to as Natpython [28], Natlink provides a gateway to interact with the DNS system from the backend with capabilities to:

- Add speech recognition to Python programs
- Control DNS from Python
- Write voice macros in Python for DNS.
- Create grammars telling DNS what to listen for and get call-backs when speech matching the grammar is recognized.
- Access some of the functionality of DNS from Python including sending keystrokes to other applications, controlling the microphone, creating new DNS users, make DNS recognize speech from a file and write text to a file.

There are several other good speech recognition tools in the market. At the same time, DNS is one of the most efficient tools that claims 97% recognition accuracy [45]. All speech recognition tools are striving for better recognition accuracy, as there is no single system that can recognize any given audio input with 100% accuracy. Hence, much of the recognition tools have not yet focused on generating time-stamped transcripts for a given audio file.

Time-stamped transcript index files, as mentioned above, are a basic requirement for audio snippet retrieval systems. XML, the eXtensible Markup Language [33] that defines a

standard structural design for data representation and exchange on the Internet is a desired format for index files. We hence designed an XML file structure for documenting the transcripts with proper indices and timestamps to the audio file. Since an archival of audio files resulting from several audio conferences need to be indexed, a design that uniquely identifies and links the index files with respective audio files is required. In order to assist in audio snippet retrieval from archived audio conferences, the desired XML index file should contain the following information:

- Name of audio file
- Conference Starting Time
- Conference Ending Time
- Speaker
- Length of audio file
- Audio Clip

Name of audio file - The name of the audio file provides a unique identification to the audio file using conference related data such as name of the speaker, conference title, date and time of the conference along with the time when the speaker joined the conference. The filename format is proposed to contain information in the following format:

"*speaker_conference title_year_month_date_chours_cminutes_cseconds_jhours_jminutes_jseconds"*

where, chours, cminutes and cseconds correspond to the conference starting time and jhours, jminutes and jseconds correspond to the time at which the speaker joins the conference.

This information, being available in the filename, provides a unique index, since no two conferences clients can ever have all the information in the filename identical. This technique also helps to integrate conference files from multiple speakers or multiple conference files of a single speaker, etc.

Conference Starting time (CST) - Starting time refers to the time at which the conference was initiated and includes both date and time information.

Conference Ending time (CET) - Ending time refers to the time at which the conference is ended by the VV server.

Speaker – Speaker name is the username for VV client tool and the DNS username for training files as well. It is required to maintain integrity among the audio conferencing and speech recognition tools in order to obtain better results.

Length of the audio file (LA) - Length of the audio file is the time in seconds for which the audio conference has been recorded. Length of audio file can be obtained from the following equation:

$$LA = CET - CST \text{ seconds},$$

where CET is the conference ending time and CST is the conference starting time.

Audio clip - Audio clip refers to an audio snippet which is a segment of an audio file that starts at some location in the audio file and ends after a specific duration: in this experiment the duration is 15 seconds. Indexing information outlined below is to be documented in the XML file for each audio snippet:

- Clip name – This is similar to the audio file name, with additional information to obtain snippet's relative position in the source audio file.

- Clip Starting time (CLST) - Clip starting time is the time in the audio file from which the clip is extracted.

- Clip Ending time (CLET) - Clip ending time is calculated using the clip starting time and applying the following equation:

  $$CLET = CLST + 15 \text{seconds},$$

  where CLET is the clip ending time and CLST is the clip starting time.

- Dragon Transcription - The output text generated by Dragon Naturally Speaking speech recognition tool when the audio clip is given as input.

The resulting XML index file for each client's conference audio file can be represented in the following form:

```
<conference>
          <starting time>   - conference starting time
          <ending time>    - conference ending time
          <audio file name>
          <speaker name>
          <audio clip  1>
          <audio clip 2>
          |
          |
          |
          |
          |
          <audio clip n>
</conference>
```

where, n is the number of snippets with 15 second duration generated from the given audio file.

The <audio clip> structure can be represented as:

```
<audio clip>
          <starting time>      - audio snippet starting time with reference to conference audio timeline.
          <ending time>        - audio snippet ending time with reference to conference audio timeline.
          <clip name>
          <transcription>
</audio clip>
```

where, clip name is the unique name given to the 15 second audio snippet file with information to identify the position of the snippet in the original audio file. Transcription refers to the output obtained from speech recognition system when the audio snippet file is given as an input. The structures of the <starting time> and <ending time> are similar and can be represented as follows:

```
< time>
        <year>
        <month>
        <date>
        <hour>
        <minutes>
        <seconds>
</time>
```

As mentioned earlier, the speech recognition tool does not automatically generate the desired XML index file. We hence designed the iTalk processor described in 3.3.1, which is integrated with the DNS speech recognition tool, to generate transcripts for every 15 seconds of audio, i.e., snippets, in the input audio file and document them together in a single XML index file.

A high level overview of the iTalk processor is shown in Figure 17.

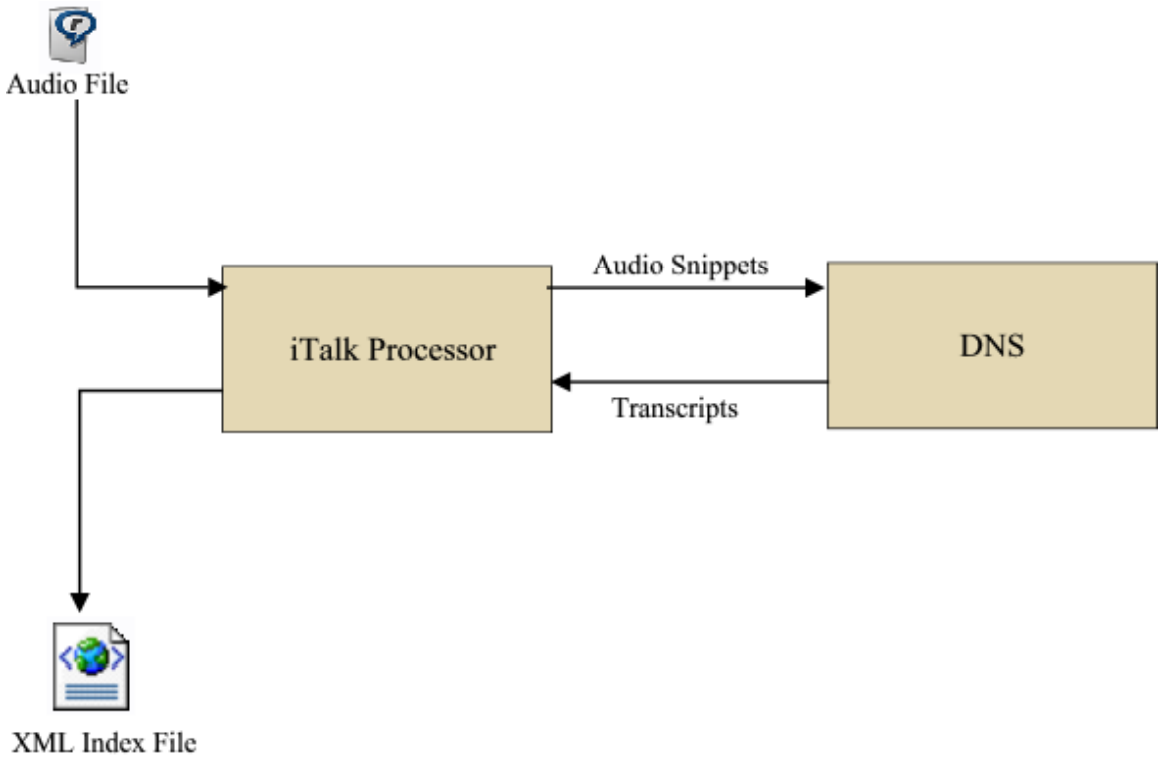

Figure 17: A high level overview of iTalk processor.

iTalk processor takes an input audio file, passes audio snippets to the DNS, and obtains transcripts to generate an XML index file. For a given audio snippet, when speech matching with the reference grammar is recognized, DNS returns the identified words to the iTalk processor to be written to the XML file.

To aid the processor, we developed an algorithm in Python that interacts with the Natlink module in order to programmatically send audio input to the DNS system and receive the text generated by the DNS system to a text file. Using the general transcription grammar to generate multiple hypotheses for every word recognized, the algorithm writes the results to a text file.

The following steps are involved in writing the results of DNS recognition to a text file, from an input audio file using the Natlink module:

Step 1. Identify the DNS user for the speaker of the audio file, ideally user = speaker.

Step 2. Connect the Natlink module with DNS, which launches DNS if it is not already running.

Step 3. Set the microphone state to "off". This step disables any other audio input sources to DNS.

Step 4. Open the specified DNS user which loads the user training files for recognition.

Step 5. Call DNS to take input from a wave file with the indicated filename simulating real time recognition and performing utterance detection by separating speech from pauses.

Step 6. Open the output text file to which the recognized results are to be written.

Step 7. Initialize and load the transcription grammar with the results object set to be obtained when the recognition is specific to the grammar and the result to contain partial recognition hypotheses, here on referred to as multiple hypotheses, which will be available during recognition.

Step 8. Activate the global dictation grammars.

Step 9. When a word is recognized, write all the partial recognition hypotheses to the output text file.

Step 10. Unload the transcription grammar, close the output text file, disconnect from DNS, and close the input wave file.

Thus, we can programmatically obtain transcription results to a text file for an input audio (wave) file.

3.3.1 The iTalk Speech Processor

The goal of iTalk speech processor is to generate an XML index file with all the necessary meta-information for archiving the audio conferences, and text version of the words spoken in those conferences for searching the audio archive and skimming through the audio files. The sensitive issue in this process is related to the speech recognition process. Despite their performance with dictation audio and broadcast news, the transcripts obtained from speech recognition have not proved to be successful for audio indexing. Main concerns include the misrecognised words, OOV words, and indices to the correctly recognized words. From the research presented in Chapter 2, we can assume that the effects of misrecognised words and OOV words are almost negligible for audio file retrieval. Hence, iTalk speech processor focuses on the recognized words and their position of occurrence in the audio file.

The DNS output is based on the input given. When given the same audio with different levels of background information, the output differs, as the speech recognizer tends towards the most accurate words that fill the sentence based on the reference grammar. If a previously recognized word is wrong, the probability of the next fit word to be wrong will increase. In order to index the audio file better, we have chosen to segment the audio file in 15 seconds snippets. 15 seconds of audio snippet is long enough to accurately recognize the words by providing sufficient context information and to contain a possible keyword but at the same time short enough not to be computationally burdensome. By passing the same audio file with differing background levels, i.e. from 0 sec, 5 sec, and 10 seconds in the audio, for only 15 seconds, we can expect an increase in the probability of occurrence of accurately recognized words. At the same time, proper indices are generated using the overlapping segments.

In order to provide differing levels of background information to the DNS and index the audio file at all possible indices, we present a 3-tier algorithm. “3-tier” refers to the 3 differing levels of audio segments sourcing from the same audio file that are given to the DNS system for transcription. Figure 18 presents the 3-tier segmentation of the audio file as it is fed to the speech recognition system.

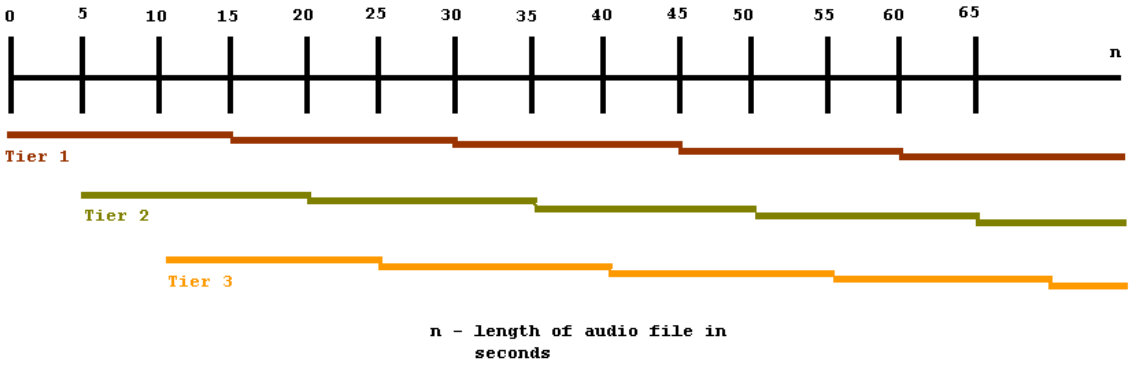

Figure 18: 3-tier feed.

This feed is the basic iTalk technique to index audio files for skimming through them. In tier-1, the audio file is fed from 0 to *n* seconds in segments of 15 seconds each, where *n* is the length of the audio file in seconds. So the tier-1 feed segments would be 0 to 15 seconds, 15 to 30 seconds, 30 to 45 seconds, 45 to 60 seconds, ……,(*n*-15) to *n* seconds in the source audio file. Tier 2 feeds the audio file from 5 to *n* seconds, and Tier 3 repeats the process from 10 to *n* seconds in segments of 15 seconds each, where *n* is the length of the audio file in seconds. Hence, tier-2 feed would be 5 to 20 seconds, 20 to 35 seconds, 35 to 50 seconds, ……., (*n*-10) to *n* seconds and the tier-3 feed would be 10 to 25 seconds, 25 to 40 seconds, 40 to 55 seconds, …, (*n*-5) to *n* seconds in the source audio file. Thus, the 3-tier feed to DNS results in transcripts that contain accurately recognized words reappearing as a consequence of the overlap in the segments. The overlap in the 3-tier feed is aimed at indexing the audio file with maximum indices to accurate keywords.

The iTalk processor is hence developed to give 3-tier feed to DNS and obtain multiple hypothesized outputs to generate the desired XML index file. A high-level algorithm for the iTalk processor is as follows:

Algorithm to generate time-stamped transcript index file.

```
Start
open new XML file in write mode
write metadata to XML file
for length(audio file):
        Tier-1
          splitaudio(0)
        Tier-2
          splitaudio(5)
        Tier-3
          splitaudio(10)
close all open files
disconnect from DNS
remove snippet wave files created.
End

splitaudio(starting position):
startpos = starting position
for length of audio file:
if startpos <= length of audio file:
   read frames from input audio and write to wave file from startpos for 15 seconds.
   startpos = startpos + 15
   close the wave file
   write the wavefile meta data to XML file
   do transcript()

doTranscript():
    generate multiple hypothesis()
    write to XML file
```

When the iTalk speech processor is triggered with an input audio file, the processor creates a new file of XML type and opens it in the write mode. The meta-information obtained from the audio file name (filename format described in section 3.3) is then added to the XML file with respective tagging data including the following variables: starting time, ending time, filename, and speaker of the audio file. The processor then prepares for 3-tier feed to the DNS by segmenting the audio file into chunks of 15 seconds in three steps:

Step 1. The processor reads the input audio file and writes 15 seconds of the audio to a new wave file. The new wave file or snippet is then given a unique id as the filename. The filename aids in the indexing providing position specific information from the input audio file. The clip id is defined to be in the format:

(clip number)audio file name(tier start position)

where, clip number is a value from 0 to m, m refers to the maximum number of snippets in the given audio file. The maximum number of snippets in a given audio file is obtained from the following equation:

m = length of audio file / snippet length

or simply

m = length of audio file in seconds / 15 seconds

Tier start position is 0 seconds in step-1, since the first audio snippet is generated from the initial or the 0^{th} second position in the source audio file. Hence, m audio snippets are generated in step-1, and each snippet starts at the position, where the preceding snippet ends for the length of the original audio file. So in tier-1, the first audio snippet would be identified as *(0)audio file name(0)*, the second snippet as *(1)audio file name(0),* and so on with the last audio snippet identified as *(m)audio file name(0).*

Step 2. Similar to step 1, the processor reads the input audio file and writes the data to a new wave file. However, this time the processor reads from 5^{th} second position in the source audio file and generates a new wave file of 15 seconds duration. All the snippets that follow start from the position at which the preceding snippet ends resulting in audio file segments from 5 to 20 seconds, 20 to 35 seconds, 35 to 50 seconds, and so on until (*n*-10) to *n* seconds. In tier-2, the start position being 5^{th} second, the first audio snippet will be identified as *(0)audio file name(5),* the second snippet as *(1)audio file name(5)*, and so on until *(m)audio file name(5).*

Step 3. Similar to the previous steps, the processor reads the input audio file and writes to a new wave file. However, this time the processor reads from 10^{th} second position in the source audio file and generates a new wave file of 15 seconds duration. All the snippets that follow start from the position at which the preceding snippet ends resulting in audio file

segments from 10 to 25 seconds, 25 to 40 seconds, 40 to 55 seconds and so on until (n-5) to n seconds. In tier-3, the start position being 10^{th} second, the first audio snippet will be identified as *(0)audio file name(10),* the second snippet as *(1)audio file name(10)*, and so on until *(m)audio file name(10).*

When a keyword is identified in a particular snippet, the clip id format helps in retrieving the position in the audio file. The snippet position can be calculated using the clip id applying the following rules:

Rule 1. snippet start position = (clip number + tier start position) * 15

Rule 2. snippet end position = snippet start position + 15

In each step, after the segmented snippet is generated, the meta-data of the snippet, such as the clip id, the starting and ending times of the audio snippet, is added to the XML file with respective tags. The audio snippet is then referred to the DNS to receive recognition results through the steps outlined in section 3.3. These results are also written to the XML file with appropriate tagging data.

Thus, the iTalk processor continues for 3m snippets generated by tiered segmentation. These 3m snippets have different background audio data as well as overlaps in audio data. With proper analysis the transcripts of these 3m snippets can provide useful indices to the source audio file. Hence, the XML file is loaded with the transcripts for all 3m snippets along with the necessary tagging information. Figure 19 presents the flow of information among the modules in iTalk processor:

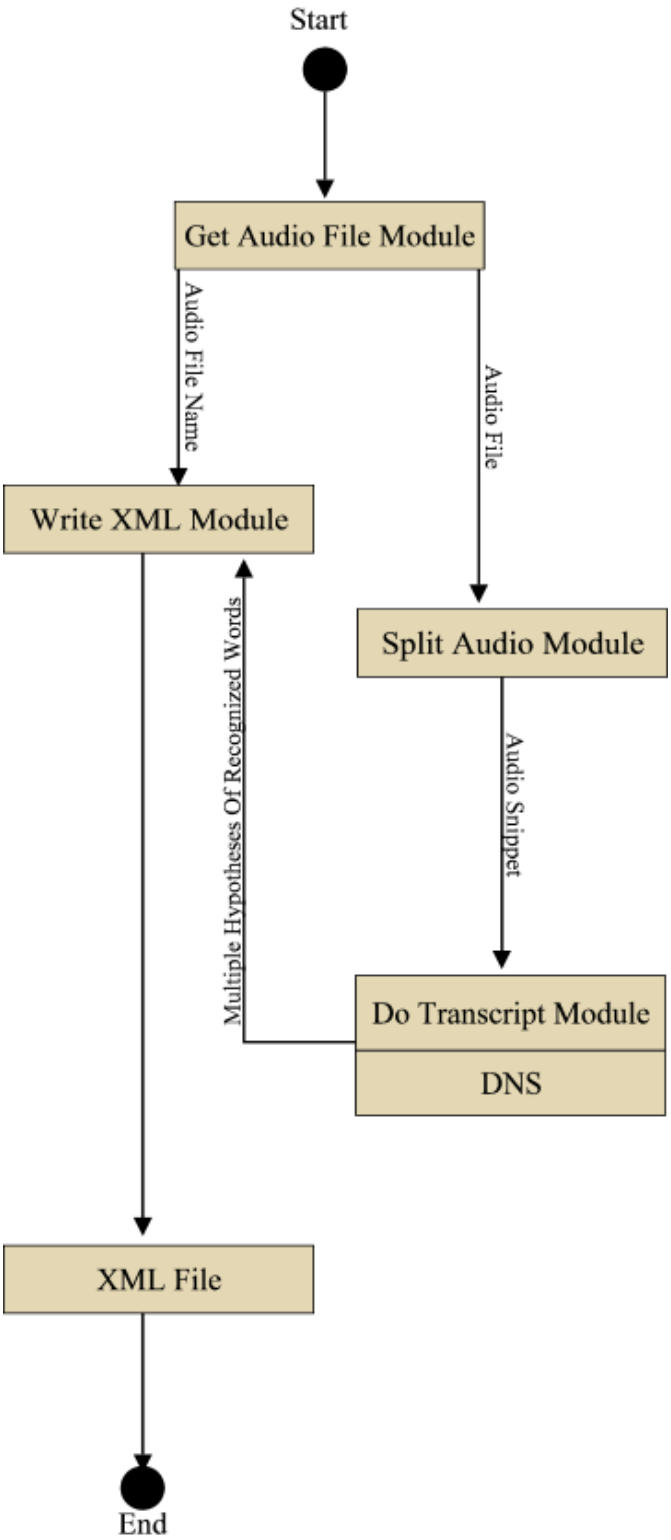

Figure 19: iTalk processor flow diagram.

The XML file and the source audio file are then forwarded to the iTalk file system. The iTalk file system, is further indexed by the Wumpus text-based file system search engine [38], whereas the snippets generated by the iTalk processor are deleted. Archiving the original audio file helps to maintain the sound quality of an audio file, when retrieved as a query result. Thus, the iTalk processor coupled with the DNS, is used as a speech recognition tool that generates time-stamped transcripts.

3.4 Audio Snippet Retrieval System

The Wumpus file system search engine is acted upon the iTalk file system to enable audio snippet retrieval with text-based queries. The back-end infrastructure for audio snippet retrieval with Wumpus search engine consists of the following three components:

- Update Server
- Wumpus Search Engine
- Http Server

Update Server - The update server monitors the "incoming" directory for new audio, XML, and log files and processes them as they arrive. When the update server receives a new file, the server follows the following steps to maintain the integrity of conference data:

1. The .wav file is converted into a single-channel raw audio file with 8 bits sample width and 22050 Hz sample rate using the sox sound converter tool [54]. This is to ensure high quality of the audio file.

2. Check for the presence of XML file for the given conference and speaker, if present, the new XML file is merged with the existing XML file so that there will be only one XML file for each speaker in a particular conference.

Hence, for each conference the update server contains the following information:

- One or more audio files and a single XML index file for each speaker in the conference and
- Log file with information about the joining and leaving time of each participant in the conference.

Wumpus Search Engine - Wumpus search engine [38] indexes the XML files and processes search queries.

HTTP server - HTTP server takes requests from the web browser, reformats them, forwards to wumpus search engine, and receives the wumpus response. The wumpus response is then forwarded to the web browser.

The retrieval system enables for keyword search query and audio file download query to wumpus. When a search query is given:

- The HTTP server translates the keyword query into a form that can be understood by wumpus, similar to the format below:
  rank "<audio>".."</audio>" by "word1", "word2", ..., "wordN"

- All 15 second snippets are ranked using a modified version of BM25 [32], similar to the method described in [35].

- Wumpus search engine then returns the top matching "<audio>".."</audio>" XML elements along with their full text to the HTTP server. The HTTP server modifies the look and feel of the information and returns it to the web browser.

When a download query ("get audio") is given:

- The request for an audio snippet contains the name of the conference, the name of the speaker, the start timestamp of the snippet, and its length. The latter two are given in seconds. The start offset is the number of seconds from the beginning of the audio stream for that speaker.

- Wumpus uses the above information to identify the raw audio file for the conference and the speaker, extracts the requested snippet, uses sox [54] to convert the audio data back into .wav format, and returns it to the HTTP server, which, in turn, returns it to the web browser.

- When the user requests multiple streams (i.e. streams of all speakers in the conference mixed together) instead of only a single audio stream, wumpus uses the information in the conference's .log file to find out when the individual speakers entered the conference. The relative start time of the participant's audio streams is then used to translate the start offset in the selected audio stream to the corresponding start offsets in other participant's audio streams. These audio streams are then mixed together into one single

.wav audio stream, containing audio data for all speakers with the help of sox tool [54].

Thus, the wumpus audio snippet retrieval system not only returns a particular snippet of speaker's conversation in a conference but with a little bit of added complexity also generates an integrated snippet with the contributions of all participants in the conference at that particular point of time in the conference.

3.5 Design summary of iTalk components

We have defined the essential tools and components for the development of iTalk system as audio recording component for the voice conferencing tool, time-stamped transcription component for the speech recognition tool, and the audio snippet retrieval component for the search engine tool. These components were defined specifically for iTalk, the design of which has been the focus of this chapter. Selected voice conferencing tool, VV, is equipped with the audio recording component. Section 3.2 outlined the design of the recording component. We then presented the design for time-stamped XML index file in section 3.3 followed by the design of iTalk processor with an algorithm to programmatically generate the defined XML index file using DNS. Finally, we described the steps involved in audio snippet retrieval component that uses the text based file system search engine wumpus to retrieve audio snippets either as a single stream specific to a speaker or as multiple streams specific to a conference, as outlined in section 3.4. The integration of VV audio conferencing system, the iTalk processor with DNS that generates time-stamped transcripts of audio recordings along with the wumpus text-based file system search engine results in the desired system for capture, archival and searching of audio conferences i.e; the iTalk tool kit.

Chapter 4

Architecture of iTalk Tool Kit

In this chapter, we present the architectural design for the iTalk tool kit that integrates the vocal village voice conferencing system, DNS speech recognition system, and Wumpus text based file system search engine tool to capture, archive, and search conference audio. We then describe this initial architecture with an informal user feedback obtained from our demo presented at CASCON 2005 conference in Toronto, Ontario, Canada. Further, we present an architectural evolution that was developed to overcome the limitations of the initial architecture.

4.1 Integration concerns

Having designed the three essential components (see Chapter 3), integration of these components appears to be a comparatively easier task. The main components and the flow of information among them are described in Figure 20.

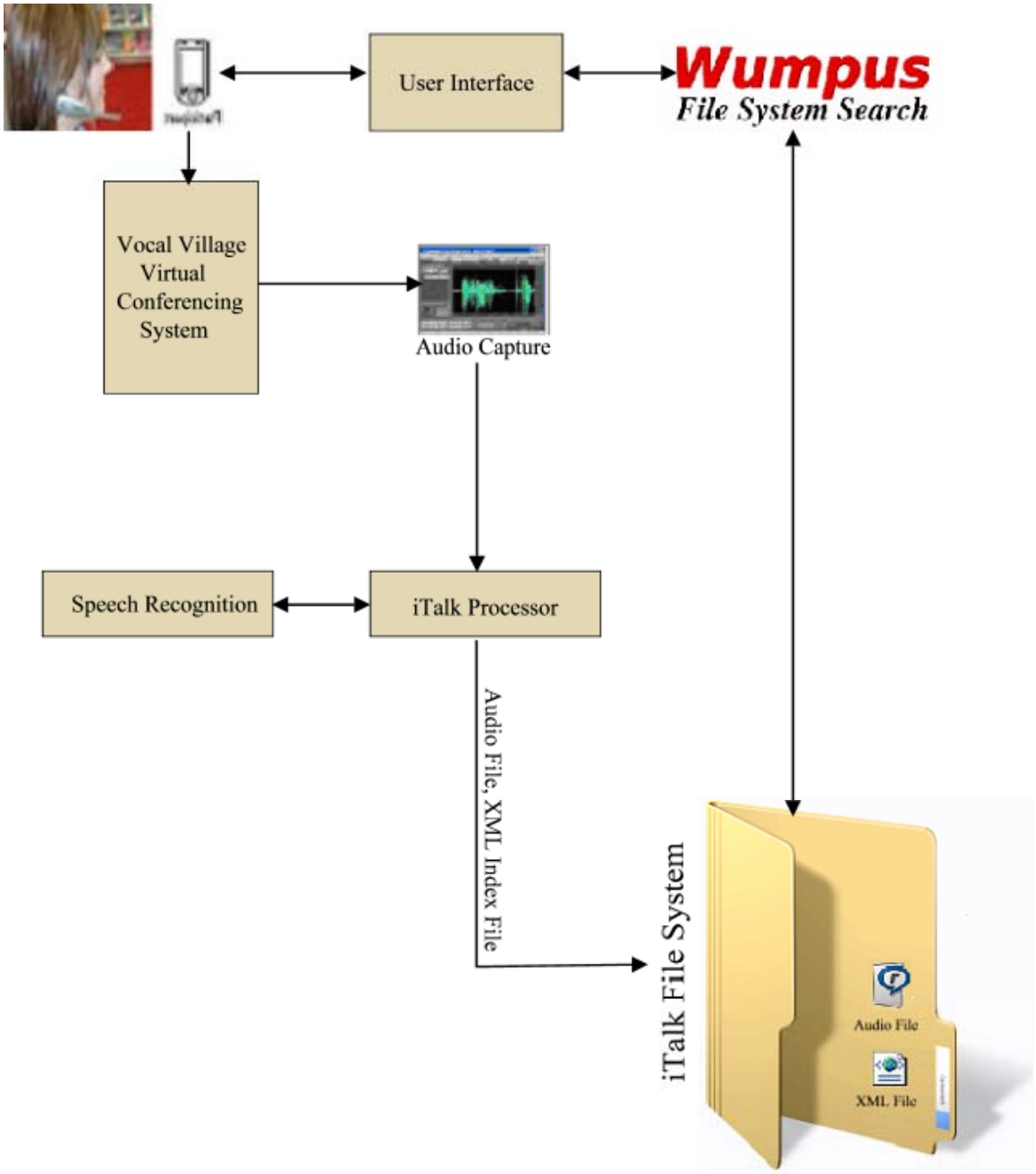

Figure 20: iTalk system – component level flow diagram.

As mentioned in Chapter 3, the iTalk processor interacts with DNS to generate the desired XML index file. DNS uses the speaker training files as a reference for speech recognition. Hence, each conference participant needs to train the DNS to generate the user file that is stored on the same system. This user file will be used as a reference by the DNS in order to recognize the words accurately. iTalk processor interacts with DNS as well as with this user file to generate an XML index file for the given conference audio input. iTalk system is an integration of technologies developed across different platforms and different programming languages. The VV conferencing tool is Windows based software developed in VB.Net. Whereas, the iTalk processor is developed in Python to control the DNS which is a Windows based tool, in order to obtain necessary recognition data. Wumpus is a file system search engine for Linux environment. Hence, integration design needs to consider several technical issues for the development of this system, which includes but is not limited to the file formats, security concerns, and possible loss of data. To achieve the desired results the VV client system, iTalk processor, and the DNS were merged to work on a single Windows system. The system is realized in the initial architecture described in the following section.

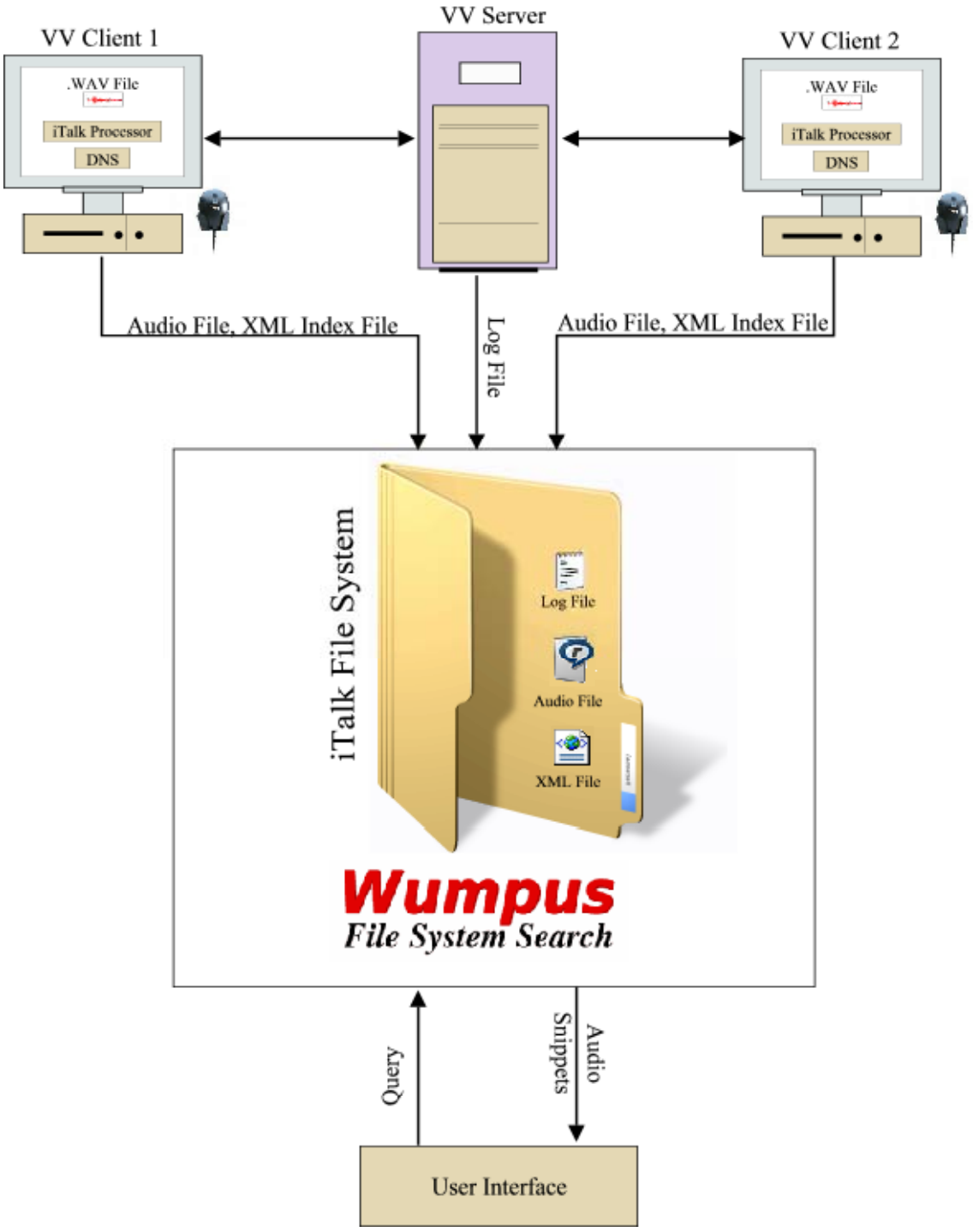

Figure 21: Overview of initial architecture for iTalk tool.

4.2 Initial Architecture

iTalk aims at recording, archiving, and searching audio conferences talks. An overview of the initial architecture for the iTalk tool is represented in the Figure 21.

As evident from Figure 21, iTalk architecture is developed by merging the VV client side software with the iTalk processor which in turn interacts with the DNS, as described in section 3.3.1. VV audio conferencing system connects all participants in a conference and broadcasts each participant's contribution to the conference in real time across all the participants in that conference with the help of a central VV server. When a conference is initiated by a VV client, the server creates a log of the conference by tracking the time at which participant is joining and leaving the conference. The VV server log file maintains data such as participant's username, time at which the user joined the conference, time at which the user left the conference. During the conference, VV client side tool records the user's speech contribution to the conference on the user's disk. At the end of the conference, the recorded speech is saved as a wave file and the iTalk processor is triggered in each of the VV client system. The processor, as described in the section 3.3.1, segments the wave file into audio clips of 15 seconds in 3 tiers and connects to DNS in order to receive partial recognition hypotheses for all the audio clips. The processor then generates the XML index file. The generated XML index file is then uploaded to the iTalk file system along with its associated client wave file. Thus, all participants of the conference have their speech contribution and its associated XML index file uploaded to the iTalk file system. The VV server, at the end of conference, uploads the log file created for that conference to the iTalk file system. The iTalk file system contains an audio file, an XML index file for each client that participated in the conference, and a log file for the same conference. The iTalk file system is then indexed by the Wumpus search engine for audio snippet retrieval, as outlined in section 3.4. This detailed process of the client side architecture is represented in Figure 22. This design requires each user to have DNS installed and trained in the system where VV client conferencing will be used. Further, at the end of the conference, users should wait for the iTalk processor to generate the XML file that is then forwarded for indexing to the Wumpus search engine. We had an opportunity to present this system at IBM CASCON 2005 conference in Toronto, Ontario, Canada. Informal user feedback obtained from the conference proved to be very effective for further analysis. Figure 23 presents an overview of the presentation at the conference.

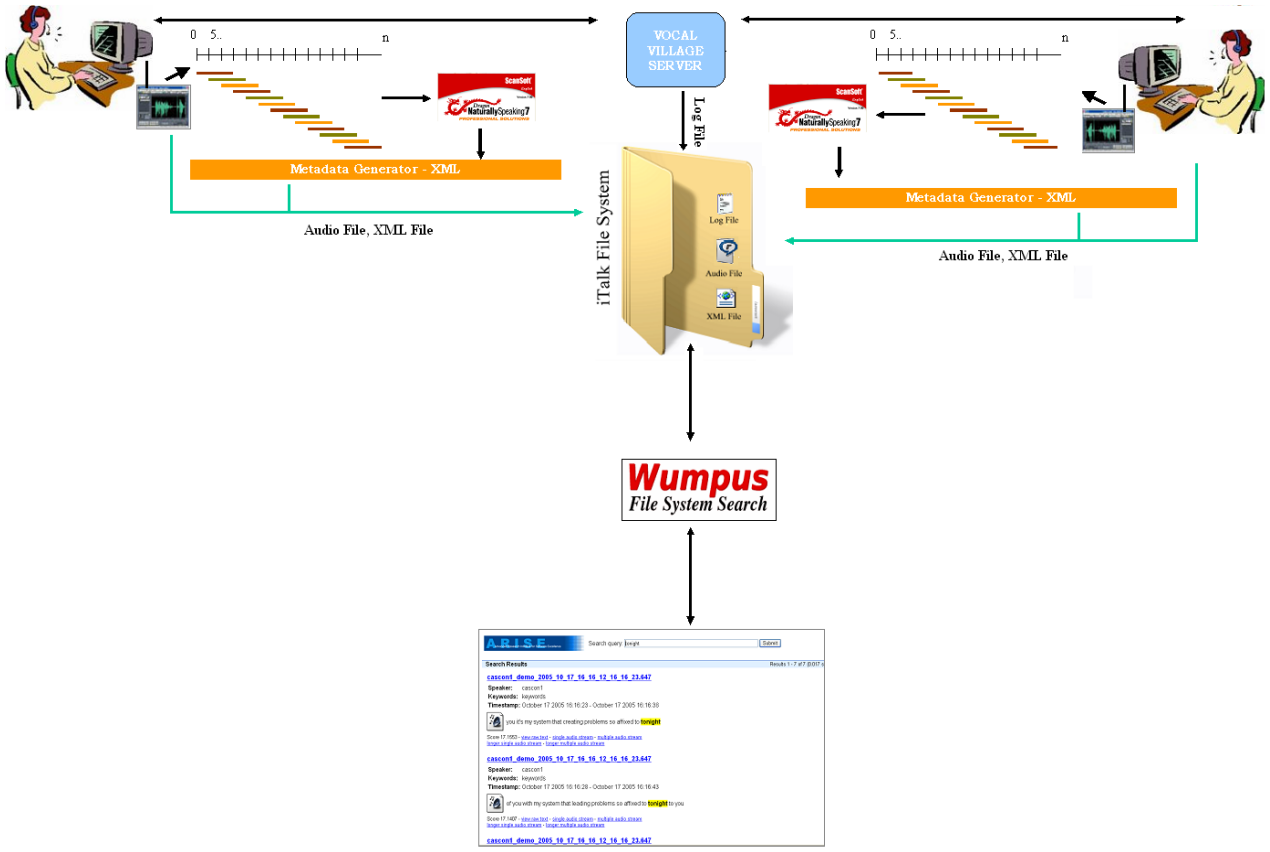

Figure 22: Detailed initial client side architecture.

Figure 23: Presentation of iTalk system at CASCON 2005 conference in Toronto, Ontario, Canada

In general, most users found the iTalk system to be more of a search engine for voice conferences. User interface of this system contains nothing but a search bar and needs to be enhanced to improve usability. Users were satisfied with the search results alongside with the snippet retrieval performance of the system; however there were few comments regarding the usability and overall performance of the system. The most noteworthy feedback is presented below:

1. The current iTalk processor requires a dedicated processor due to the involvement of DNS and leaves no space for the user to do mutli-tasking.
2. The longevity of the processing time at the end of the conference until the files are uploaded to the iTalk file system raised concerns. Though acceptable for short conversations of less than 5 minutes, conversations around 30 to 60 minutes extend the longevity of the wave file processing and the XML file generation processes.
3. Each client system in the conference should have the DNS and iTalk processor installed on it, which limits the user to use only that system for participating in the conference. The desired system should be available over the Internet for downloading in negligible time with easy installation. However, the involvement of DNS complicates this issue, as it is a commercial tool that requires user training for the minimum of 30 minutes to an average of 3 hours to generate the user training files. Even if the training files were to be downloaded from a different system, the time spent in mere installation of DNS and completion of the new user wizard itself is significant enough to be considered.
4. Changes and updates to any component of the system should be replicated for all the clients. Overall, the process is a very cumbersome task.
5. The client systems might not have the capability to perform huge audio file processing after installing tools like DNS that consume a lot of disk space. If a client was to participate in a conference for at least an hour, the resulting recorded audio file is large. Further more, the file requires extensive processing time alongside with the inevitable upload time, when being forwarded to the iTalk file system.

To overcome the limitations outlined above, we have worked towards easing the usability and saving time at the user's end. Efforts resulted in development of server architecture for the iTalk system to be described in the following section.

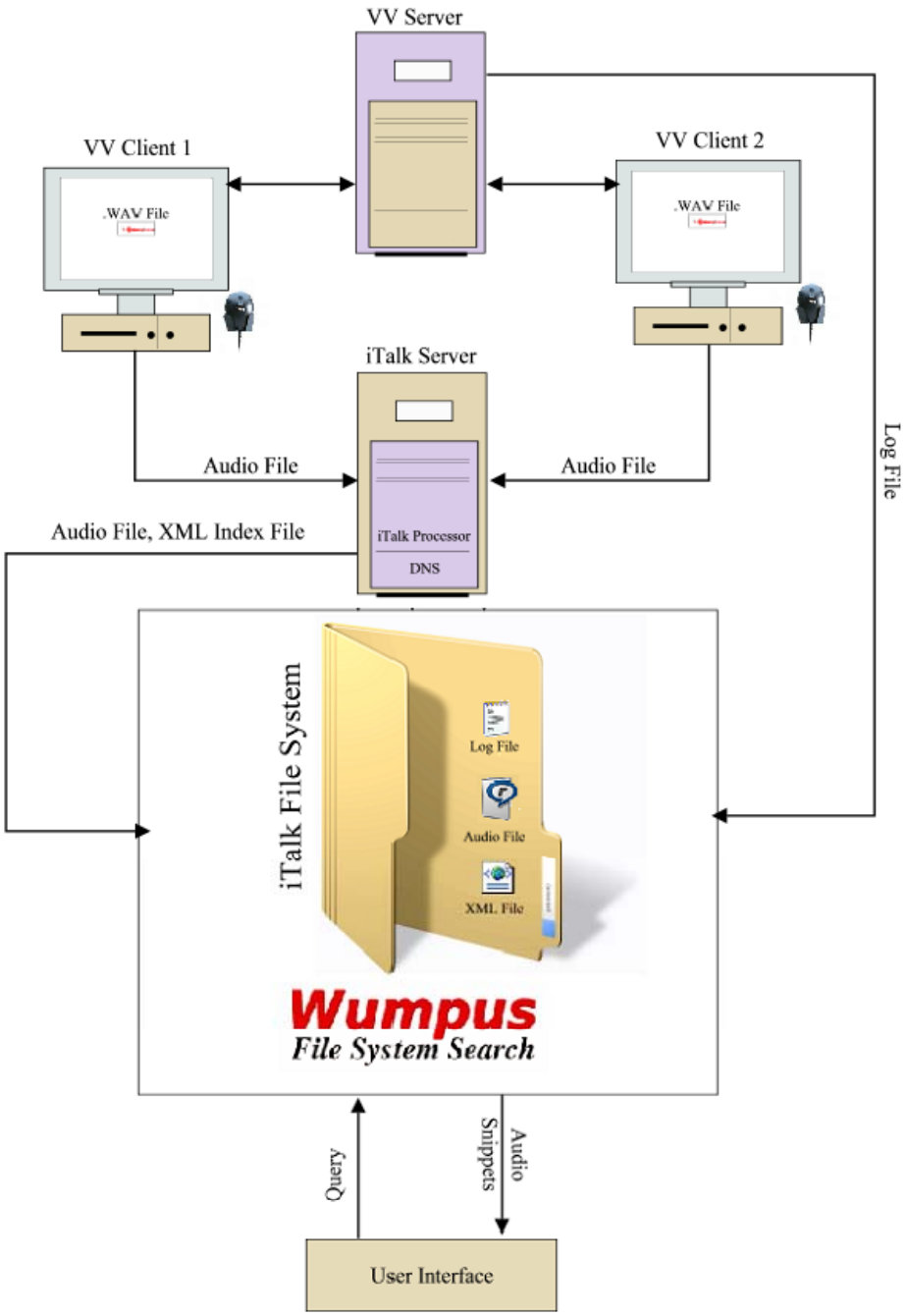

Figure 24: Current iTalk Server Architecture

4.3 Server Side Architecture

This design is aimed at enhancing the iTalk tool performance, while reducing the burden on client machines. Figure 24 presents an overview of the server side architecture.

A dedicated server is added to the initial architecture to run the iTalk processor with DNS. This migration of the processor from the client machine raised issues such as:

- DNS has to handle multiple users on a single machine.
- User training files must be obtained and archived for all participants.
- Input audio files must be archived on the iTalk server.
- The iTalk processor must be enhanced with a server module that keeps looking for incoming files and handles them to be processed by the iTalk processor as they are received.

However, the migration proved worth handling the above issues, since the client machine burden was drastically reduced and usability has significantly improved:

- Users can use any machine with VV installed on it. The requirement for the client machine to have DNS installed on it will no longer be valid.
- When a user ends a conference, the audio file is uploaded to the iTalk server reducing the time for processing the audio file on the client machine.
- Users can transfer their DNS training files once initially to the iTalk server. When a user ends a conference, the audio file will be uploaded in the background to the iTalk server while the user can continue his work on the same machine.

Hence, we designed the iTalk server architecture to ease usability while maintaining the integrity of the system. The iTalk server was developed to continuously listen for incoming audio files. For every incoming audio file, the iTalk processor would be triggered to segment the audio files and receive transcripts from DNS to generate XML index files as described earlier. Each audio file and its associated XML index file are further uploaded to the iTalk file system indexed by the wumpus search engine. Figure 25 provides an overview of the detailed flow of processes for the iTalk server.

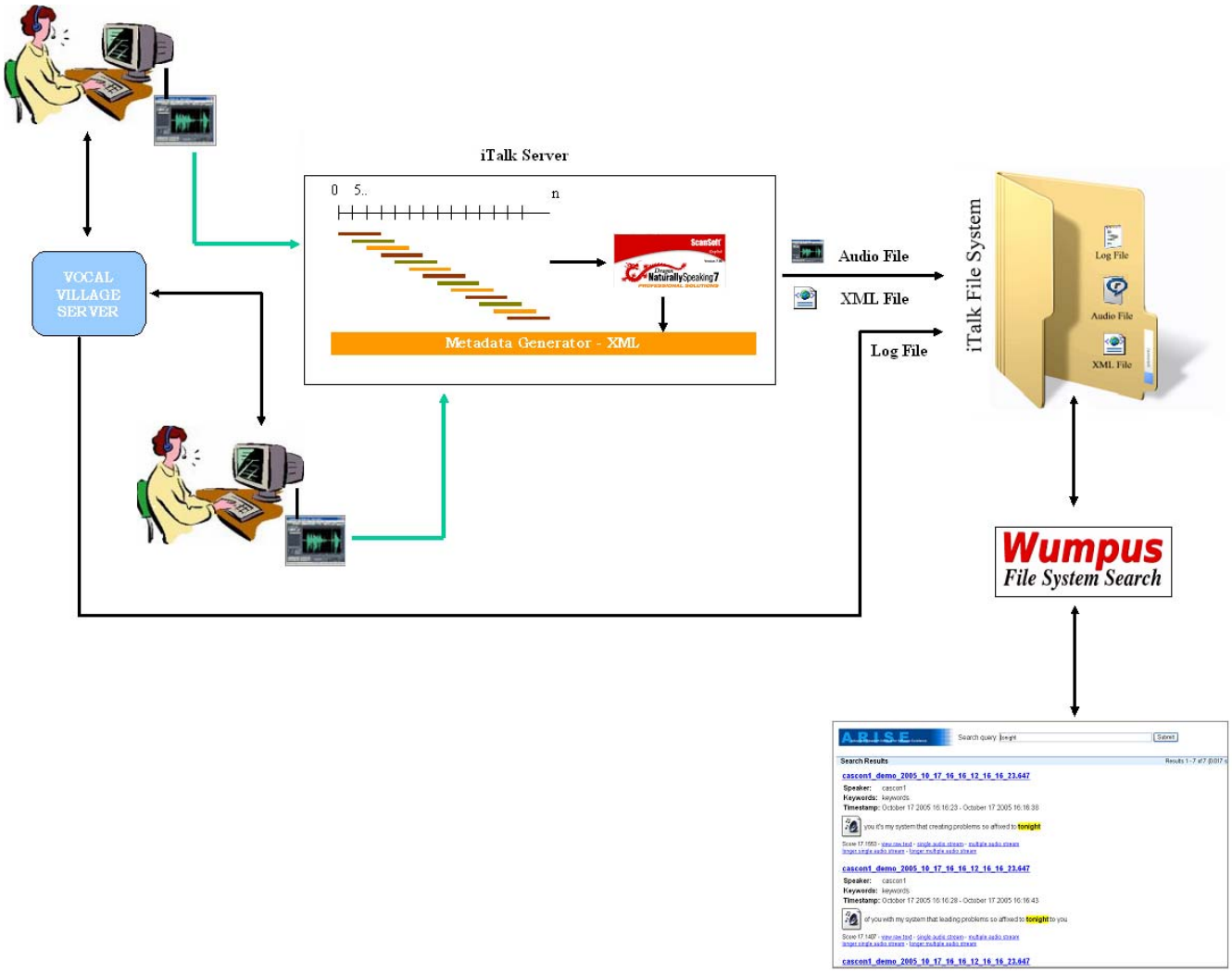

Figure 25: Detailed iTalk server architecture

At the end of a conference, the VV clients upload recorded audio file to iTalk server, while the VV server forwards the conference log file to the iTalk file system. Thus, the server side architecture is very transparent to the end user. Development of iTalk server clearly differentiated the 3 components – VV, iTalk server, and the wumpus search engine tools, as merely sharing data to provide the same integrated tool results. This architecture opened new domains: similar tools can handle the input to the iTalk server and output from iTalk server by sharing data in the same procedure. For example, VV can be replaced with Yahoo! Voice Calls [55], whereas wumpus search engine can be replaced with Google search engine [56] to provide similar or better results in audio conference indexing.

4.4 Summary of iTalk Architectures

We integrated the VV voice conferencing tool, the iTalk processor, DNS, and the Wumpus file system search engine to achieve the desired functionality of the iTalk system. We presented the integration concerns followed by the initial architecture for iTalk system, which merges the VV tool with the iTalk processor and DNS to generate XML index file that is uploaded to the iTalk file system which is indexed by the wumpus search engine. However, this tool posed several limitations that were discovered during our presentation of the system at the IBM CASCON 2005 conference in Toronto, Ontario, Canada. We further described the architectural evolution of the iTalk server which not only solved many usability and performance issues, but also opened doors for future developments of iTalk system with a perspective to be incorporated into several other applications. We hence achieved the goal to develop an audio indexing system for VoIP based audio conferences that enables recording, archiving, searching, and skimming through the audio data with effective snippet retrieval.

Chapter 5

Future Work

In this chapter, we focus on identifying specific areas that need improvement and should be considered in future research and development of iTalk tool. First, we concentrate on further evaluation of the iTalk system with consideration given to usability and speed of data retrieval. Later, we determine possible application domains for the iTalk tool.

5.1 Improving iTalk tool: usability and speed of data retrieval

iTalk server needs further evaluation in order to identify specific positions that can improve audio snippet retrieval performance with low rate of false positives. With the current design, the results of a query are audio snippets with transcripts that contain some or all the words in the query. The snippet that has all the query words occurring in the transcript, for the highest number of times will be ranked highest. It should be pointed out, that the rank largely depends on the search engine techniques used. However, the possibility that all the three tiers of an audio snippet have the requested query words happens more than once. This unavoidably leads to retrieval of all the three tiers of the snippet. Ex:- 30-45 seconds snippet of an audio might be retrieved as a result of a query and still contain 35-50 and 40-60 second snippets of the same audio in the search results list.

As such, better methods can be implemented to result in one audio snippet from one audio file to be retrieved. If the file retrieved corresponds to what the user was searching for, he or she can also listen to either the following snippet of the audio file or to the one prior to that. This procedure ensures that the list actually contains audio snippets from different audio files that resulted for a particular query, which ultimately results in effective search for audio information that the user would like to retrieve.

As mentioned in the introduction, there is a distinct need to develop a quality user interface for users to be able to sift through audio conferences for spoken words just as they would sift through text files. Unlike the current design that retrieves the related audio snippets as search

results, a more sophisticated version of user interface would enable the user to view all audio files that belong to one conference as a single search result. With such an interface, the user will be able to skim through different speakers in the conference at the exact point in time when the search keyword was spoken.

Several improvements can be made in the UI design to achieve better usability. However, the backbone of the iTalk system, i.e. the iTalk processor, can also be enhanced in different ways. Strategies of further improvement include: addition of semantic analyzers, phonetic recognizers, keyword extractors to aid in the retrieval performance.

5.2 Applications of iTalk system

The possible applications of the iTalk system include:

- iTalk could be improved into a personal audio search engine, where users can record their every day tasks or any other personal information they would rather talk and record, than type into their system. This recording will then be copied to the iTalk server to be further processed for indexing. The resulting XML index file can be returned back to the user's personal database to be indexed by a search engine. Thus, iTalk server can be enhanced to be very mobile and flexible for users.

- iTalk system could be used to retrieve a voice message from an archival of voice messages by searching for a keyword or a speaker. However, this requires that iTalk processor receive recognition results from speaker-independent speech recognition tool. The possibility, even though distinct, is realistic considering the pace at which speech recognition research is heading forward.

- Development of an interactive user interface with users being able to speak out their queries to the system would result in a complete audio search and retrieval system. The recognition results for the query audio can be passed on to the search engine for retrieval from iTalk system.

Even though development of the above mentioned application systems require a lot of improvement and enhancements to the iTalk system, the perspectives are positive and realistic if performance and stability of the system are improved.

Chapter 6

Conclusion

The first chapters of the research are focused on the background of speech recognition process, audio indexing, out of vocabulary words, information retrieval, and VoIP based voice conferences. Having presented different audio indexing research efforts, we estimated that there is a strong need for tools with functions similar to iTalk.

We, then, derived and outlined our approach for automatic recording, archiving and indexing of VoIP based audio conferences to enable search for words spoken in those conferences. We defined the time stamped transcript specific to voice conferences and the methodology for generating such transcripts using existing DNS speech recognition tool. We then described the developmental approaches of iTalk processor and the 3-tier algorithm that was designed to aid in obtaining relative indices from the transcripts generated by the DNS tool. We presented the two architectures developed for iTalk indexing system. From the research and development of iTalk system, we foresee a lot of enhancements that can be done to improve the retrieval performance, data security, processing speed, and the speech recognition accuracy. Suggestions for further research and analysis include evaluation of the benefits and performance of the 3-tier algorithm; development of a better user interface; integration of the iTalk system into a personal audio search system, and many others. Being limited in time, we could not successfully integrate all of those into the current system. Overall, we have estimated that iTalk system is a positive environment for enhancements that result in a good audio indexer for VoIP conferences.

Bibliography

[1] Woodland, P., "Speech recognition," *Speech and Language Engineering - State of the Art (Ref. No. 1998/499), IEE Colloquium on* , vol., no.pp.2/1-2/5, 19 Nov 1998.

[2] Rabiner, L.R., "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE* , vol.77, no.2pp.257-286, Feb 1989.

[3] De Mori, R., Brugnara, F. "A Survey of the State of the Art in Human Language Technology," *Eds. Ron Cole,* Chapter1 – Section 1.5, 1996, http://speech.bme.ogi.edu/HLTsurvey. Accessed September 9, 2006.

[4] Scansoft Inc, "Open speech recognizer: An intelligent, scalable and comprehensive speech recognition solution from SpeechWorks," *White Paper*, June 2005. http://www.nuance.com/recognizer/whitepapers/. Accessed September 9, 2006

[5] Cyphers, S., Hazen, T., and Glass, J., "Tools for Automatic Transcription and Browsing of Audio-Visual Presentations," *CSAIL abstract*, 2005.

[6] Leavitt, N., "Let's Hear It for Audio Mining," *Computer*, vol. 35, no. 10, pp. 23-25, Oct., 2002.

[7] Van Rijsbergen, C.J., "Information Retrieval*," Newton, MA, USA: Butterworth-Heinemann*, 1979.

[8] Siegler, M.A., "Integration of continuous speech recognition and information retrieval for mutually optimal performance," *Ph.D. Thesis, Carnegie Mellon*

University, 1999. http://www.cs.cmu.edu/~msiegler/publish/PhD/thesis.pdf. Accessed September 9, 2006

[9] Büttcher, S. and Charles, C.L.A., "A security model for full-text file system search in multi-user environments," In *4th USENIX Conference on File and Storage Technologies* (USENIX FAST), San Francisco, December 2005.

[10] Chignell, M. and Kilgore, R., "The Vocal Village: Enhancing Collaboration with Spatialized Audio conferencing," 2004. http://anarch.ie.utoronto.ca/IML/model/publications.php?ord=r. Accessed September 9, 2006

[11] Kurimo, M., "Thematic indexing of spoken documents by using self-organizing maps," *Speech Commun.* 38, 1, Sep. 2002, 29-45.

[12] Happe, A., Pouliquen, B., Burgun, A., Cuggia, M., Le Beux, P., "Automatic concept extraction from spoken medical reports," *International Journal of Medical Informatics,* July 2003, V. 70, Issues 2-3, Pp 255-263.

[13] MENELAS: http://www.med.univ-ennes1.fr/plaq/proj_an/menelas_an.html. Accessed August 31, 2006.

[14] Park, A., Hazen, T., and Glass, J., "Automatic Processing of Audio Lectures for Information Retrieval: Vocabulary Selection and Language Modeling," *Proc. Int. Conf. on Acoustics, Speech, and Signal Proc.*, Philadelphia, PA, March 2005.

[15] Glass, J., Hazen, T., Cyphers, S., Schutte, K., and Park, A., "The MIT Spoken Lecture Processing Project," *Proc. HLT/EMNLP*, Vancouver, October 2005.

[16] Salgado-Garza, L.R., Nolazco-Flores, J.A., Díaz-López, P.D., "A proposed architecture for a spoken information retrieval with multimedia databases," *Taller de Tecnologías del Lenguaje Humano, Encuentro Internacional de Computación ENC'04,* México, 2004.

[17] Brown, M. G., Foote, J. T., Jones, G. J., Jones, K. S., and Young, S. J, "Open-vocabulary speech indexing for voice and video mail retrieval," *In Proceedings of the Fourth ACM international Conference on Multimedia* (Boston, Massachusetts, United States, November 18 - 22, 1996). MULTIMEDIA '96. ACM Press, New York, NY, 307-316, 1996.

[18] Saraclar, M., Begeja, L.,Gibbon, D., Liu, Z., Renger, B., Shahraray, B., “A system for searching and browsing spoken communications,” *HLT/NAACL Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval*, Boston, MA, May 6, 2004.

[19] Woodland, P. C., Johnson, S. E., Jourlin, P., and Jones, K. S., “Effects of out of vocabulary words in spoken document retrieval (poster session),” *In Proceedings of the 23rd Annual international ACM SIGIR Conference on Research and Development in information Retrieval* (Athens, Greece, July 24 - 28, 2000). SIGIR '00. ACM Press, New York, NY, 372-374, 2000.

[20] Church, K.W.,”Speech and language processing: where have we been and where are we going?," *In EUROSPEECH-2003*, 1-4, 2003.

[21] Allan, J., “Perspectives on Information Retrieval and Speech,” *In information Retrieval Techniques For Speech Applications [This Book Is Based on the Workshop "information Retrieval Techniques For Speech Applications", Held As Part of the 24th Annual international ACM SIGIR Conference on Research and Development in infor* A. Coden, E. W. Brown, and S. Srinivasan, Eds. Lecture Notes In Computer Science, vol. 2273. Springer-Verlag, London, 1-10, 2002.

[22] Garofolo, J.S., Auzanne, G.P., Voorhes, E.M., “The TREC spoken document retrieval track: A success story,” *Proceedings of the Recherche d’Informations Assiste par Ordinateur: Content Based Multimedia Information Access Conference,* April 12-14, 2000.

[23] Lamming, M. G. and Newman, W. M., “Activity-based Information Retrieval: Technology in Support of Personal Memory,” *In Proceedings of the IFIP 12th World Computer Congress on Personal Computers and intelligent Systems - information Processing '92 - Volume 3 - Volume 3* (September 07 - 11, 1992). F. H. Vogt, Ed. IFIP Transactions, vol. A-14. North-Holland, 68-81, 1992.

[24] Stifelman, L. J., “Augmenting real-world objects: a paper-based audio notebook,” *In Conference Companion on Human Factors in Computing Systems: Common Ground* (Vancouver, British Columbia, Canada, April 13 - 18, 1996). M. J. Tauber, Ed. CHI '96. ACM Press, New York, NY, 199-200, 1996.

[25] Whittaker, S., Hyland, P., and Wiley, M., “FILOCHAT: handwritten notes provide access to recorded conversations,” *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Celebrating interdependence* (Boston,

Massachusetts, United States, April 24 - 28, 1994). B. Adelson, S. Dumais, and J. Olson, Eds. CHI '94. ACM Press, New York, NY, 271-277, 1994.

[26] Johnson, S.E., Jourlin, P., Spärk Jones, K., Woodland, P.C., "Spoken Document Retrieval for TREC-8 at Cambridge University," *In NIST special publication 500-246: The eighth Text REtrieval Conference (TREC-8). http://trec.nist.gov/pubs/trec8/t8_proceedings.html.* Accessed September 9, 2006

[27] Gould, J., Power Point slide presentation given to the Voice Coder's Group. http://www.synapseadaptive.com/joel/NatLinkTalk.ppt. Accessed September 7, 2006.

[28] Speech Recognition Wiki. NatPython/Natlink/Vocola: http://speechwiki.org/NL/HomePage.html. Accessed September 7, 2006.

[29] The Python Programming Language: http://www.python.org. Accessed September 7, 2006.

[30] Chelba, C., and Acero, A., "Position specific posterior lattices for indexing speech," *In Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, Ann Arbor, MI, 2005. ACL - 2005.

[31] Wikipedia - The Free Encyclopedia. Word error rate: http://en.wikipedia.org/w/index.php?title=Word_error_rate&oldid=71372259. Accessed August 31, 2006.

[32] Robertson, S.E., Walker, S., Hancock-Beaulieu, M., "Okapi at TREC-7," *Proceedings of the Seventh Text REtrieval Conference,* Gaithersburg, U.S.A.; November 1998.

[33] eXtensible Markup Language (XML): http://www.w3.org/XML. Accessed September 7, 2006.

[34] Wikipedia - The Free Encyclopedia. NaturallySpeaking: http://en.wikipedia.org/w/index.php?title=NaturallySpeaking&oldid=72866539. Accessed September 3, 2006.

[35] Büttcher, S., Clarke, C.L.A., Lushman, B., "Term Proximity Scoring for Ad-Hoc Retrieval on Very Large Text Collections," *Proceedings of the 29th ACM Conference*

on Research and Development in Information Retrieval (SIGIR 2006). Seattle, USA; August 2006.

[36] Interactive Media Lab and Collaborative Effectiveness Lab, Univ. of Toronto. Vocal Village: www.vocalvillage.net. Accessed September 6, 2006.

[37] Searchspell. Voice Recognition: http://www.searchspell.com/sites/wordv/voice%20recogntiion.php. Accessed September 6, 2006.

[38] Wumpus Search Engine. Wumpus File Search: www.wumpus-search.org. Accessed September 6, 2006.

[39] Garofolo, J., Voorhees, E., Stanford, V., Spärk Jones, K., "TREC-6 1997 spoken document retrieval track overview and results," *In NIST special publication 500-240: The sixth Text REtrieval Conference (TREC-6). http://trec.nist.gov/pubs/trec6/t6_proceedings.html.* Accessed September 10, 2006

[40] Garofolo, J., Voorhees, E., Stanford, V., Auzanne, G., Lund, B., "TREC-7 1998 spoken document retrieval track overview and results," *In NIST special publication 500-242: The seventh Text REtrieval Conference (TREC-7). http://trec.nist.gov/pubs/trec7/t7_proceedings.html.* Accessed September 10, 2006

[41] iRiver N10 personal digital audio recorder: http://www.iriver.com/html/product/prpa_product.asp?pidx=31. Accessed September 10, 2006.

[42] SUMMIT speech recognition system: http://groups.csail.mit.edu/sls//technologies/summit.shtml. Accessed September 10, 2006.

[43] Lucene text search engine: http://groups.csail.mit.edu/sls//technologies/summit.shtml Accessed September 10, 2006.

[44] NOMINDEX indexing program: http://www.med.univ-rennes1.fr/doc/nomindex/noomindex.html Accessed September 10, 2006.

[45] Dragon Naturally Speaking speech recognition system: http://www.nuance.com/naturallyspeaking/ Accessed September 10, 2006.

[46] IBM Via Voice speech recognition system: http://www-306.ibm.com/software/voice/viavoice/ Accessed September 10, 2006.

[47] MeSH – A French medical lexicon thesaurus: http://www.nlm.nih.gov/mesh/ Accessed September 10, 2006.

[48] Managing Gigabytes query system: http://mg4j.dsi.unimi.it/ Accessed September 10, 2006.

[49] SPHINX III speech recognition system: http://cmusphinx.sourceforge.net/html/cmusphinx.php Accessed September 10, 2006.

[50] CMUseg tool: http://www.cs.cmu.edu/~msiegler/ Accessed September 10, 2006.

[51] TIMIT acoustic-phonetic continuous speech corpus: http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1 Accessed September 10, 2006.

[52] Robinson, T., Fransen, J., Pye, D., Foote, J., and Renals, S., "WSJCAM0: A british english speech corpus for large vocabulary continuous speech recognition," *In Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 81--84, Detroit, May 1995. Institute of Electrical and Electronic Engineers.

[53] Voice Genie – Voice XML gateway: http://www.voicegenie.com Accessed September 10, 2006.

[54] SOX sound converter tool: http://sox.sourceforge.net/ Accessed September 11, 2006.

[55] Yahoo! Voice Calls – Audio conferencing tool: http://messenger.yahoo.com/ Accessed September 11, 2006.

[56] Google search engine: http://www.google.com Accessed September 11, 2006.