Multimedia Document Retrieval Using Speech and Speaker Recognition*

Mahesh Viswanathan, Homayoon S.M. Beigi, Satya Dharanipragada,
Fereydoun Maali† and Alain Tritschler

IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598, USA

Abstract

Speech and speaker recognition systems are rapidly being deployed in real-world applications. In this paper, we discuss the details of a system and its components for indexing and retrieving multimedia content derived from broadcast news sources. The audio analysis component calls for real-time speech recognition for converting the audio to text and concurrent speaker analysis consisting of the segmentation of audio into acoustically homogeneous sections followed by speaker identification. The output of these two simultaneous processes is used to abstract statistics to automatically build indexes for text-based and speaker-based retrieval without user intervention. The real power of multimedia document processing is the possibility of boolean queries in the form of combined text- and speaker-based user queries. Retrieval for such queries entails combining the results of individual text and speaker based searches. The underlying techniques discussed here can easily be extended to other speech-centric applications and transactions.

Keywords: audio indexing, speech recognition, speaker recognition, speaker segmentation, spoken document analysis

1 Introduction

The goal of an audio-indexing system is to provide the capability of searching and browsing through audio (and video) content. The system is formed by integrating text retrieval methods with large vocabulary continuous speech recognition and speaker recognition techniques. A large vocabulary continuous speech recognition system is used to produce time-aligned transcripts of the speech. The speech signal is also divided into acoustically homogeneous segments, classified and labeled. Information retrieval techniques are then employed on these recognized transcripts, combined with the labeled speaker segments to identify locations in the text that are relevant to the search request. These locations with time-alignments then specify regions of the speech that are relevant for the request. The time information is used to extract the appropriate audio and video segments from the source media.

The indexing and retrieval components of this system are indeed two distinct applications. The indexer uses real-time speech recognition of streaming audio, and real-time acoustic segmentation followed by speaker recognition to analyze streaming audio in real-time, which is followed by a

*To Appear: International Journal of Document Analysis and Recognition, June 2000
†Signal Recognition Corporation, P.O. Box 7010, New York, NY 10128, USA

very swift index generation step after the audio terminates. The indexing subsystem is completely integrated with the audio analysis step. Since the index formation has to be complete before use, the combined audio analysis and indexing steps are considered to be a single off-line process. The index files thus engendered form the seed for the retrieval engine. The retrieval subsystem consists of two parts, an interface to accept queries and reflect results back to the user, and a retrieval engine to evaluate the queries and interact with the user interface. The results are presented in the form of video or audio clips with the relevant portions pinpointed for easy access and playback. Additionally, a transcription of the audio is also available.

Conventional information retrieval mainly focuses on retrieving text documents from large collections of text. The basic principles of text retrieval are well established and have been well documented [Salton89]. The index is a mechanism to match descriptions of documents with descriptions of queries. The indexing phase describes documents as a list of words or phrases, and the retrieval phase describes the query as a list of words or phrases. A document (or a portion thereof) is retrieved when its description matches the description of the query.

Data and retrieval models required for multimedia objects are quite different from those required for text objects. Typically, the indexing phase analyzes the multimedia object for certain characteristics which are then abstracted into feature vectors for comparison with the the feature vectors derived from the retrieval station. Then, similarities between feature vectors of these objects are computed to rank the data set. There is little consensus on a standard set of features for multimedia data. One approach to index an audio database might be to use certain audio cues such as an applause, music or speech. Similarly, for video it might be key frames, or shot changes. However, for real world audio derived from radio sources, or commercial TV broadcasts which are predominantly speech, the text can be generated using speech recognition, the speaker labeled using speaker recognition, and the turns and segments so derived can be used for indexing the associated audio and video.

The rest of the paper is organized as follows. A brief review of prior work in related fields is followed by a description of the speech recognition and information retrieval systems that constitute this audio indexing system. The tests conducted to evaluate just this subsystem are presented next. The subsequent section describes a speaker segmentation algorithm which is a necessary prerequisite step to speaker identification and verification. Two different algorithms for speaker recognition are discussed along with the implementation and performance details for the same. A description of the design and architecture of the audio analysis system which incorporates the three modules provides an overview of the implementation test-bed. Finally, the algorithm for combined text and speaker index creation, and retrieval is discussed along with some results.

2 Prior Work

Several systems for multimedia content indexing and retrieval have been documented in recent years. What these systems have in common with ours is the focus on using audio or audio transcripts in building indexes for information retrieval. But our system goes further as will be evident.

For issues relating to building user interfaces for audio browsing and retrieval systems, see [Hirschberg99]. An editorial review contrasting two video archival indexing systems is in [Grosky97]. The first system uses close-captioning information to build a video index. The second system includes indexing of video content and implies automatic speech transcription, which is also a salient feature of the system described in this paper. Video sequences are retrieved by textual queries. Yet another system uses a combination of face recognition derived from a video sequence and name extraction via close-captioned text and transcripts [Satoh99]. Wold et al. describe a system for search, identification and retrieval of audio based on a host of features that define

and characterize sound in an intuitive manner. The emphasis is on using acoustic and perceptual features for short and single-gestalt sound retrieval [Wold96].

CueVideo employs video key frames detection and speech recognition for audio/video browsing and indexing [Srinivasan99]. The necessity of speaker-based segmentation as a first step for several indexing tasks is emphasized in [Delacourt99]. A two-pass approach is used – a distance-based measure for speaker change detection, followed by BIC-based validation. Another BIC-like speaker segmentation scheme is reported in [Liu99]. However, they provide no details on speaker indexing. A preliminary version of our multimedia retrieval system was presented in [Viswanathan99].

3 Text-Based Information Retrieval

The current text-based information retrieval system consists of two ancillary components: (1) A large vocabulary continuous speech recognition system, and (2) a text-based information retrieval system. Below is a brief description of the two systems, followed by an independent evaluation of this information retrieval system. A discussion of the implementation and components that make up the system is also provided.

3.1 Speech Recognition System

Speech recognition systems are typically guided by three components: a vocabulary, a language model and set of pronunciations for each word in the vocabulary. A vocabulary is a set of words that is used by the recognizer to translate speech to text. As part of the decoding process, the recognizer matches the acoustics from the speech input to words in the vocabulary. Therefore, the vocabulary defines what words can be transcribed. If a word that is not in the vocabulary is to be recognized, it must first be added to the vocabulary. A language model is a domain-specific database of sequences of words in the vocabulary. A set of probabilities of the words occurring in a specific order is also required. The output of the recognizer will be biased towards the high probability word sequences when the language model is operative. Correct decoding is therefore a function of whether the user speaks a sequence of words that have high probability within the language model. Which is why when the user speaks an unusual sequence of words, the decoder performance will degrade. Word recognition is based entirely on their pronunciation, the phonetic representations of the word [Rabiner93].

The IBM speech recognition system for broadcast news uses acoustic models for sub-phonetic units with context-dependent tying. The instances of context dependent sub-phone classes are identified by growing a decision tree from the available training data and specifying the terminal nodes of the tree as the relevant instances of these classes. The acoustic feature vectors that characterize the training data at the leaves are modeled by a mixture of Gaussian pdf's with diagonal covariance matrices. Each leaf of the decision tree is modeled by a single-state Hidden Markov Model (HMM) with a self loop and a forward transition. The IBM system expresses the output distributions on the state transitions in terms of the rank of the leaf instead of in terms of the feature vector with a mixture of Gaussian pdf's modeling the training data at the leaf. The rank of a leaf is obtained by computing the log-likelihood of the acoustic vector using the model at each leaf, and then ranking the leaves on the basis of their log-likelihoods [Bahl94, Bahl95, Gopalakrishnan95].

A baseline acoustic model with 90,000 Gaussians was built using 70 hours of training data by rebuilding the Gaussian mixtures for about 3500 HMM states. Two new decision trees were built for context clustering, one based on training data from prepared and spontaneous speech and

Speech Conditions	WER (%)
Prepared speech	22.2
Spontaneous speech	29.6
Low-fidelity speech	39.6
Speech+Music	37.5
Speech+Background noise	35.1
Non-native speakers	29.7
Overall	29.7

Table 1: Word Error Rates (WER) for IBM's 1997 broadcast news speech recognition system using the 1997 Hub4 evaluation test set.

the other based on all the training data (including telephone channel speech, speech from non-native speakers, and speech in the presence of background noise and music). Gaussian mixtures were then estimated using the EM algorithm and the performance for various model sizes were evaluated. The language model (LM) has a vocabulary of 65000 most frequent words from a broadcast news language model corpus. The baseline LM is a deleted interpolation trigram model which is also trained on the 70 hours of acoustic training data transcriptions plus 400 million words of broadcast news text made available by the Linguistic Data Consortium in the context of the DARPA sponsored Hub4 evaluations of Large Vocabulary Continuous Speech Recognition systems on broadcast news [Pallet97]. The speech data exhibits a wide variety of speaking styles, environmental and background noise conditions, intended to model real-world spontaneous audio.

For this audio indexing task for broadcast news the recognizer was trained on data that is different from the Hub4 (the annual NIST sponsored evaluations of speech recognition systems) training data. This enabled us to judge the performance of our system under mismatched conditions where one wishes to retrieve, as is often the case in practice, spoken documents that come from a domain different from the domain that the speech recognizer is trained on. The acoustic space is parameterized by 60-dimensional feature vectors which are obtained by performing a linear discriminant analysis (LDA) on a nine-frame window of 24-dimensional cepstral coefficients vectors. The decision tree for identifying the context-dependent sub-phone classes was grown using the Wall Street Journal (WSJ) data. The decision tree has around 6000 leaves.

3.2 Speech Recognition Performance

The speech recognition system described above runs at real-time or better on a 266 MHz Pentium II personal computer. The system was tested on the 1997 Hub4 evaluation test set which consists of three hours of broadcast news. The word error (deletions, substitutions, and insertions) rates for various speech and background conditions are given in Table 1. The error rates cited can further vary depending on the audio quality. For very poor quality audio, in some cases the error rate may be as high as 70%.

3.3 Some Issues for Retrieval Systems Using Speech Recogntion

Retrieval systems that use the text generated by speech recognition systems face several issues and constraints. Unless otherwise derived from the acoustic information (lip smacks, pauses) or special language constructs, all sentence structure information is lost in the decoding. (IBM does

have a system that punctuates sentences as it transcribes deducing periods and some commas, but it is nowhere near as comprehensive as a well-punctuated piece of English text [Chen99]. All current dictation systems including the IBM ViaVoice product line for office dictation expect users to dictate punctuation along with their text, but this option is not feasible in broadcast news which targets a listening and viewing audience.) Speech recognition systems recognize punctuation symbols with several different verbal representations such as the symbol '.' which may be "period", "dot", or "point". In real-world broadcast news speech there is no representation for punctuations which explains some of the reduced accuracy in the table above between prepared speech and spontaneous speech.

With audio and video, the granularity of the retrieved element is another issue to contend with. One approach is to assume that the input audio or video clip has to be retrieved as is, that is, it corresponds to a single document. Then, in response to a query, the relevant audio or video clip would be retrieved from a repository of such clips. Alternately, the transcript from each audio or video clip in the repository may be broken down into several chunks (documents) during indexing. In this case, the search results would be obtained as a time-aligned document within a specific audio or video clip. (This is the approach we have pursued.) A third approach would be to retrieve the relevant clip and the points of interest within it.

There are several measures that can be undertaken to compensate for the lower accuracy of spontaneous speech material transcription and lack of sentence structure. One approach would be to also include alternate words from the recognition results. The recognizer outputs the best path through a sentence as the best word matches even if individual words score better because it uses a weighted sum of the acoustic and language model scores. Alternate words may be used as weighted index terms for the document. Words that are out of the vocabulary (OOV) cannot be recognized and therefore cannot be index terms. Such words have to be handled separately. Domain mismatch must also be handled in terms of the content being indexed. Overall prior work indicates that spoken document retrieval systems approach 80% of that of full text retrieval systems.

3.4 Text-based Information Retrieval

It is typical that information retrieval systems work in two phases – an off-line indexing phase, where relevant statistics about the textual documents are gathered to build an index, and an on-line search and retrieval phase, where the index is used to perform fast query-document matching returning the N best matched documents (and additional information) to the user. In the indexing phase, the text output from the speech recognition output – a continuous stream of time-stamped words – is processed to derive a document description. This is used in the retrieval phase for rapid searching. There are several operations performed in sequence in this processing: tokenization to detect sentence boundaries, part-of-speech tagging for morphological analysis, and then stop-word removal using a standard stop-word list. Most systems use stemming and filtering for the above operations. Morphological analysis, which is widely used in natural language processing, is a form of linguistic signal processing. In morphological analysis, nouns are decomposed into their roots along with a tag to indicate the plural form. Verbs are decomposed into units designating person, tense and mood, along with the root of the verb. For example, the statement
Today I am launching an international effort to ban anti-personnel land mines
after processing becomes
Today I be launch an international effort to ban anti-personnel land mine

In general, our retrieval scheme uses a two-pass approach, but in the system discussed in this paper, we use just the first-pass, principally to improve response time albeit at the cost of lower average precision. The following version of the Okapi formula [Robertson95], for computing the

matching score between a document d and a query q is used:

$$S(d,q) = \sum_{k=1}^{Q} c_q(q_k) \frac{c_d(q_k)}{\alpha_1 + \alpha_2 \frac{l_d}{\overline{l}} + c_d(q_k)} \, idf(q_k).$$

Here, q_k is the *kth* term in the query, Q is the number of terms in the query, $c_q(q_k)$ and $c_d(q_k)$ are the counts of the *kth* term in the query and document respectively, l_d is the length of the document, $\overline{l}$ is the average length of the documents in the collection, and for unigrams, $\alpha_1 = 0.5$ and $\alpha_2 = 1.5$. The inverse document frequency, $idf(q_k)$, for the term q_k which is given by:

$$idf(q_k) = log(\frac{N - n(q_k) + 0.5}{n(q_k) + 0.5}),$$

where N is the total number of documents, and $n(q_k)$ is the number of documents that contain the term q_k. The *idf* is pre-calculated and stored as are most of the elements of the scoring function above except for the items relating to the query.

Each query is matched against all the documents in the collection and the documents are ranked according to the computed score from the Okapi formula above. The scoring function takes into account the number of times each query term occurs in the document normalized with respect to the length of the document. This normalization removes bias that generally favor longer documents since longer documents are more likely to have more instances of any given word. This function also favors terms that are specific to a document and rare across other documents. (If a second pass is used, the documents would be re-ranked by training another model for documents, using the top-ranked documents from the first pass as training data.)

3.5 Evaluation and Results

Approximately 20 hours of Voice of America radio news broadcasts were collected during May-June 1996. In order to avoid replication of stories at most three broadcasts were recorded each day at eight hour intervals. There were 10 main speakers, male and female, with scores of correspondents and interviewees, both American and foreign speakers of English. Each broadcast was typically 6 or 10 minutes long.

Rank	# Known Items
≤ 1	37
≤ 5	41
≤ 10	46
≤ 20	47
≤ 100	47
Not found:	0

Table 2: IR system performance on reference transcripts – known item retrieval.

The entire speech collection is transcribed with our large vocabulary speech recognizer to produce transcripts with time-alignments for each word. Unlike a standard information retrieval scenario, there are no distinct documents in the transcripts and therefore one has to be artificially generated. A simple solution is to automatically break the text into overlapping segments of a fixed number of words and treat each segment as a separate document. The overlap eliminates the

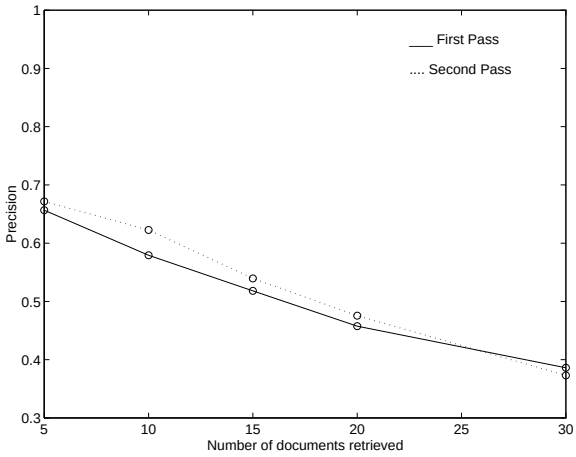

Figure 1: Retrieval performance showing precision the versus the number of retrieved documents.

potential problem of query words spanning two documents. (An alternative to fixed-sized document segments is to use topic identification schemes for subject classification, and then generating one or more documents for each topic in the transcript [Dharanipragada99a]).

Search requests were generated by having users read the documents and compose queries. This approach is also known as "known item retrieval." Overall there were 49 topics or questions and 1451 documents. The performance of our system is shown in Table 2. The average precision after the first pass was computed to be 69.92%. For a more detailed evaluation of the speech recognition system used in indexing and retrieval in this paper see [Dharanipragada97].

Another way of presenting retrieval performance is by plotting the precision versus the number of retrieved documents. This is shown in Figure 1. For example, the precision when the top 10 documents are retrieved is 57.92%.

Rank (R)	% queries with at least one relevant document in top R ranks
5	86.79%
10	96.25%
15	98.11%
20	98.11%
30	100 %

Table 3: Rank (R) vs percentage queries with at least one relevant document in the top R ranks after the first pass.

A third method of measuring retrieval performance is by the percentage of queries that have relevant documents within a given range of the ranked list of retrieved documents (Table 3).

4 Speaker-Based Information Retrieval

The speaker-based information retrieval system consists of two components: (1) an acoustic-change detection system called speaker segmentation, and (2) a language-independent, text-independent speaker recognition system. The following is a discussion of the constituent elements.

4.1 Speaker Segmentation

In order to completely automate the process of speaker identification for speaker retrieval in audio-indexing systems, it is necessary to detect the boundaries (turns) between non-homogeneous speech portions, viz., speaker-change detection. An example of early progress in speaker segmentation is in [Gish91]. Each homogeneous segment must ideally correspond to the speech of a single speaker. Once delineated, each segment – if it meets the minimum segment length requirement required for speaker recognition – can be classified as having been spoken by a particular speaker. These segments are in some sense the equivalent of "documents" in that they form individual units of retrieval.

The segmentation engine used the Bayesian Information Criterion (BIC) to partition frames produced by the front end. BIC is a model selection criterion and has evolved from an earlier Akaike's Information Criterion (AIC) [Akaike74]. BIC and BIC-like schemes have been used in other disciplines as well as in speech and speaker segmentation.

The basic problem may be viewed as a two-class classification where the object is to determine whether N consecutive audio frames constitute a single homogeneous window of frames (or segment), W, or two such windows, W_1 and W_2, with the boundary frame or "turn" occurring at the ith frame. The input audio stream can be modeled as Gaussian process in cepstral space. BIC is a maximum likelihood approach to detect (speaker) turns of a Gaussian process. The problem of model identification is to choose one among a set of candidate models to describe a given data set. It assumes the frames (10 ms) derived from the input audio signal are independent and result from a single-Gaussian process [Chen98, Tritschler99].

In order to detect if there is a speech change in a window of N frames, two models are built: one which represents the entire window (W_1) by a Gaussian characterized by its mean and covariance $\{\mu, \Sigma\}$, and a second which represents the first part of the window up to frame i with a first Gaussian $\{\mu_1, \Sigma_1\}$, and the second part of the window with another Gaussian $\{\mu_2, \Sigma_2\}$. The criterion is then expressed as:

$$\Delta BIC(i) = -\frac{N}{2} log|\Sigma| + \frac{i}{2} log|\Sigma_1| + \frac{N-i}{2} log|\Sigma_2| + \frac{\lambda}{2}(d + \frac{d(d+1)}{2}) log N$$

where, N is the number of frames in the window W, i is the number of frames of the first part of the window, $(N - i)$ is the number of frames of the second part, and d is the dimension of the cepstral vectors. λ is a penalty function which should be nominally 1, but for 24-dimensional feature vectors it is empirically determined that 1.3 yields better results. It should be noted that this formulation assumes independent feature vectors but not uncorrelated feature elements.

However, a turn is confirmed at frame i when i is not only the minimizer of ΔBIC but also drives it negative. Otherwise, the window size is increased incrementally by a fraction of N and the test is applied again. When the window exceeds a predefined size without a segment boundary, the window is viewed to be devoid of turns.

4.1.1 Implementation

The BIC algorithm has been implemented to make it fast without impairing the accuracy. The feature vectors used are simply mel-cepstra frames. No additional processing is done on these vectors. The algorithm works on a window-by-window basis, and in each window, a few frames are tested to check whether they are BIC-prescribed segment boundaries. If no segment boundary is found (positive ΔBIC), then the window size is increased. Otherwise, the old window location is recorded, which also corresponds to the start of a new window (with original size). The detailed set of steps is outlined below.

- The BIC computations are not performed for each frame of the window for obvious practical reasons. Instead, a frame resolution r is used, which splits the window into $M = \frac{N}{r}$ subsegments.
- Out of the resulting $(M - 1)$ BIC tests, the one that leads to the most negative ΔBIC is selected.
- If such a negative value exists, the detection window is reset to its minimal size, and a refinement of the point detected is performed, with a better resolution. (These refinement steps increase the total number of computations and impact the speed-performance of this algorithm. Hence, these should be tailored to the particular user environment, real-time or off-line.)
- If no negative value is found, the window size is increased from N_{i-1} to N_i frames using the following rule. $N_i = N_{i-1} + \Delta N_i$, with N_i also increasing when no change is found: $N_i - N_{i-1} = 2(N_{i-1} - N_{i-2})$. This speeds up the algorithm in homogeneous segments of the speech signal. In order not to increase the error rate though, the ΔN_i has an upper bound.
- When the detection window gets too big, the number of BIC computations is further reduced. If more than M_{max} subsegments are present, only $(M_{max} - 1)$ BIC computations will be performed – skipping the first.

4.1.2 Results

The performance of such a segmenter can be gauged by the extent that it tracks the true segments. This is measured in terms of over-segmentation, missed segments, and segmentation resolution. Different costs can be associated with each of these errors, based on the application at hand. In our system, the least desirable are the missed segments, as it can cause two distinct speaker segments to be merged into one, which in turn engenders a single identification tag for the whole segment, generating a label for the speaker in the initial portion and ignoring the second. This issue is touched upon again in the speaker identification section. Table 4 presents the performance of the BIC segmenter in testing five hours of broadcast news data with commercials excluded.

Missed segments	15%
Over segmentation	6%
Turn resolution	$\pm$ 50 frames

Table 4: Speaker segmentation system performance overview.

4.2 Speaker Identification

Speaker recognition has two principal components, speaker identification and speaker verification. For examples of prior work in speaker recognition, see [Furui89, Rosenberg92, Gish94]. Our speaker recognition engine has two different implementations for identification, a frame-based approach and a model-based approach with concomitant merits and demerits. The engine is both text and language independent which is essential for live audio indexing of material such as broadcast news. The engine also shares the signal processing front-end and the feature extraction phase with the IBM speech recognition engine.

In order to be able to index an audio stream by speaker, models derived from speakers' audio samples must be present in a speaker database. The process of adding speaker's voice samples to such a database is called enrollment. This is an off-line process and our audio indexing system provides the means to build a database for all speakers of interest. Ideally, about a minute's worth of audio is required from each speaker from multiple channels and microphones encompassing multiple acoustic conditions. The speaker recognition engine offers several functions: speaker enrollment, identification (frame-based and model-based), and verification.

4.2.1 Speaker Enrollment

Sample utterances from speakers of interest represented by a frame sequence are used to enroll speakers for subsequent identification and verification. Individual speakers are modeled as a mixture of Gaussians represented by the mean, covariance, and number of distributions within that model. A k-means clusterer generates the speaker models from their sample utterances each represented by a sequence of mel-cepstral feature vectors. Once the models are established, test speakers can be identified using either frame-based or model-based schemes. Speaker models by themselves will not suffice for the verification phase – a binary speaker tree is also needed which is built during enrollment. This binary tree is constructed bottom-up with speaker models as the leaves of the tree. Nodes at each level are merged into higher level nodes terminating with the root node. The enrollment-related information is all stored, in SVAPI parlance, in a "data store" for run-time use.

4.2.2 The Frame-By-Frame Approach

Let $\mathcal{M}_i$ be the model corresponding to the i^{th} enrolled speaker. $\mathcal{M}_i$ is entirely defined by the parameter set, $\{\vec{\mu}_{i,j}, \mathbf{\Sigma}_{i,j}, p_{i,j}\}_{j=1,\ldots,n_i}$, consisting of the mean vector, covariance matrix, and mixture weight for each of the n_i components of speaker i's Gaussian Mixture Model (GMM). These models are created using training data consisting of a sequence of M frames of speech, with the d-dimensional feature vector, $\{\vec{f}_m\}_{m=1,\ldots,M}$. If the size of the population is N_p, then the model universe is $\{\mathcal{M}_i\}_{i=1,\ldots,N_p}$. The fundamental goal is to find the i such that $\mathcal{M}_i$ best explains the test data, represented as a sequence of N frames, $\{\vec{f}_n\}_{n=1,\ldots,N}$, or to make a decision that none of the models describes the data adequately. The following frame-based weighted likelihood distance measure, $d_{i,n}$, is used in making the decision:

$$d_{i,n} = -log\left[\sum_{j=1}^{n_i} p_{i,j}\, p(\vec{f}_n | j^{th} \text{ component } of \mathcal{M}_i)\right], \text{ using a Normal representation,}$$

$$p(\vec{f}_n|\cdot) = \frac{1}{(2\pi)^{d/2}|\Sigma_{i,j}|^{1/2}} e^{-\frac{1}{2}(\vec{f}_n - \vec{\mu}_{i,j})^t \Sigma_{i,j}^{-1} (\vec{f}_n - \vec{\mu}_{i,j})}$$

The total distance, D_i, of model $\mathcal{M}_i$ from the test data is then taken to be the sum of all the distances over the total number of test frames.

$$D_i = \sum_{n=1}^{N} d_{i,n}.$$

Each class assignment is accompanied by a pseudo-distance (or score) which expresses the respective confidence. For model i: score = $\mathrm{D_i} \times \mathrm{N}$, where N denotes the number of frames in the respective utterance. Hence, each model in the data store can be accompanied by a set of rankings as identification result.

4.2.3 The Model-Based Approach

The test frames, representing the speaker whose identity is sought, are used to generate a model. The proximity of the generated test model to the prototype models is next examined and the candidate speakers are ranked in order of their proximity to the test model [Beigi98a]. The distances of the candidate speakers to the test model are also recorded. These indicate the extent of confidence in the class assignments and are used to generate the scores used in the indexing and retrieval phases.

The choice of when to use the frame-based versus the model-based approach boils down to the size of the data store and the duration of the test utterance. One fact to remember is that almost always the frame-based scheme is more accurate and at best the model-based scheme may approach the accuracy of the former. For short utterances ($<$ 10 seconds), the frame-based approach yields more accurate results using less time than that for the corresponding model-based. (Note, however, that the model-based scheme implemented here uses a full search to find the best training model that matches the test utterance. The above speed performance issue is with respect to this implementation of model-based alone.) With longer utterances (upwards of 15 seconds or so), the model-based scheme approaches the accuracy of the frame-based but with a faster response time. This speech advantage becomes even more apparent as the size of the data store increases tipping the balance in favor of the model-based scheme. As a result, the system here contains both implementations.

4.2.4 Speaker Verification

Verification is a post-identification process. The speakers labeled during identification are claimants during the verification process. Let us introduce the notion of a "cohort set" as used in verification. A cohort set is a reference to a set of enrolled speakers or their associated models that satisfy some similarity criterion. Cohorts are specified by levels of the speaker binary tree, counting bottom-up since the individual speakers are the leaves of this binary tree. For instance, a cohort level of 3 would have $2^3 = 8$ speakers all emanating from the same node. The cohort set of any given speaker includes the target speaker.

During verification, the claimant's utterance is compared against those speaker models established during enrollment. The claimed identity is verified when this test model exhibits the shortest distance to the claimed model when compared with the rest of the cohort set augmented with a variety of background models. Otherwise, the claimed identity is rejected. A background model is a model representing speakers that are not of interest.

4.2.5 Results

About 30 seconds of speech from each speaker in the training population were collected from a broadcast news environment and used to enroll the speakers. (The data was clean speech with

a mixture of prepared and spontaneous speech.) For frame-based identification eight seconds of speaker data is required, of which only the middle six seconds are actually used. This is to compensate for the resolution of the speaker segmentation to avoid contamination with adjacent speakers. In model-based identification, we require 10 seconds of speaker data for the same test. For detailed comparison of results for the speaker recognition engine, see [Beigi98b].

Two different tests were run with both the model-based and frame-by-frame approaches using a enrolled population of 199 speakers. The speaker recognition engine was first tested on 104 individual PCM files. The test data was derived from US-based broadcast news sources, and was a mixture of prepared and spontaneous speech. All the files were at least 10 seconds running time. Table 5 shows the results of the tests.

Test Result	Frame-based	Model-based
Identified	101/104 (97%)	92/104 (88.5%)
Mis-identified	3/104 (3%)	12/104 (11.5%)
Verified (from identified)	101/101	91/92
Mis-verified (from identified)	0/101	1/92
Verified (from mis-identified)	3/3	6/12
Mis-verified (from mis-identified)	0/3	6/12
Overall Verification	101/104 (97%)	97/104 (93.3%)

Table 5: Speaker segmentation and identification test results on 199 individual PCM files. Verification can either uphold the identification result or reject it. In the model-based approach, one correct identification result was negated during verification. Out of the 12 mis-identifications, six were upheld, while six were erroneously rejected.

A more realistic test for speaker-based indexing is to run segmentation and speaker identification in tandem. The 104 cells from the test above were concatenated into a single PCM file. The resultant composite file was then submitted for segmentation, followed by speaker identification and verification of the resulting segments. Only the frame-based approach was used simply because it requires fewer samples than the model-based scheme. Of the original 104 segments in the test data, there were 84 segments reported by the segmentation module. Nine of these segments fell below are minimum required 8-second test window. The system was programmed to reject results from the speaker identification module if the input segment is smaller than 8 seconds with an "Inconclusive" tag. The results are summarized in Table 6.

Speaker recognition systems are subject to channel mismatch conditions which can drop the identification rate to around 25% based on the degree of mismatch. Channel mismatch conditions can occur for a host of reasons including microphone characteristics, room or outdoor acoustics, background noise, sampling rate discrepancies, and other similar conditions. These are very difficult to control in broadcast news settings. The recommended solution is enroll using multiple recording sessions in order the capture the full range of possible scenarios under which the speaker data may be used in identification [Rosenberg92, Chaudhari99]. It must be mentioned that our system is robust enough to handle involuntary voice changes such as emotional stress or colds. Being language independent, it permits enrollment in Spanish, for instance, and testing in English or Japanese.

Speaker segments	104
Segments reported	84/104 (80.8%)
Segments missed	25/104 (23%)
Oversegmentations	5/104 (4.8%)
Identified	70/75 (93.3%)
Mis-identified	5/75 (6.7%)
Inconclusive	9/84 (10.7%)
Verified (from identified)	70/70
Mis-verified (from identified)	0/70
Verified (from mis-identified)	4/5
Mis-verified (from mis-identified)	1/5
Overall Verification	74/75 (98.7%)

Table 6: Speaker segmentation and identification results on a single audio file with multiple speakers. Segments smaller than eight seconds are dismissed by the speaker identification engine without consideration and assigned a "Inconclusive" label. 75 segments were submitted for identification. 70 were identified and verified. Of the five mis-identifications, four were upheld, and one was erroneously rejected.

5 System Architecture and Application

Processing for speech and speaker recognition is quite compute intensive and yet it is important that the processing time for audio indexing be realistic. Figure 2 shows the system architecture we have designed for real-time analysis of broadcast news audio for indexing.

Audio from a live TV broadcast or equivalent audio source is the input to this system. (The video is encoded into MPEG-1 format in a separate pass. The MPEG-1 is required for video playback during retrieval.) The system uses a common front-end signal processing module which converts the input audio into mel-cepstral feature vectors (10 ms frames). These multi-dimensional feature vectors are simultaneously delivered to the speech, speaker segmentation, and speaker identification engines in a multi-process and multi-threaded programming environment. The three engines are all programmable via application programming interfaces called *SMAPI, SEGAPI,* and *SVAPI* respectively.

The speech recognition engine used in this application is an updated 1998 version of the broadcast news transcription engine described in the first sections of this paper with slightly better performance characteristics. The time-stamp of each transcribed word from the speech recognition is recorded along with the duration of the word. These word-times are correlated with each turn from the speaker segmentation module which tracks every frame in every speaker segment identified by the time of the feature vector (or frame) relative to the start of the audio stream. The speaker identification module receives the frames from the front-end and uses the segment start information from the speaker segmentation module to gather enough frames to form an "utterance" (SVAPI-parlance). We used 800 frames for the frame-based analysis. Each utterance is then scored against the speaker data store returning a set of labels, out of which one is then verified against the cohort set. This process continues until the audio terminates or is stopped by user intervention.

At the conclusion of the audio broadcast, the indexing API is invoked automatically to generate the index files from the data gathered in the previous processing stage described above. Figure 3 is a snapshot from our application after analysis and indexing exhibiting the remnants

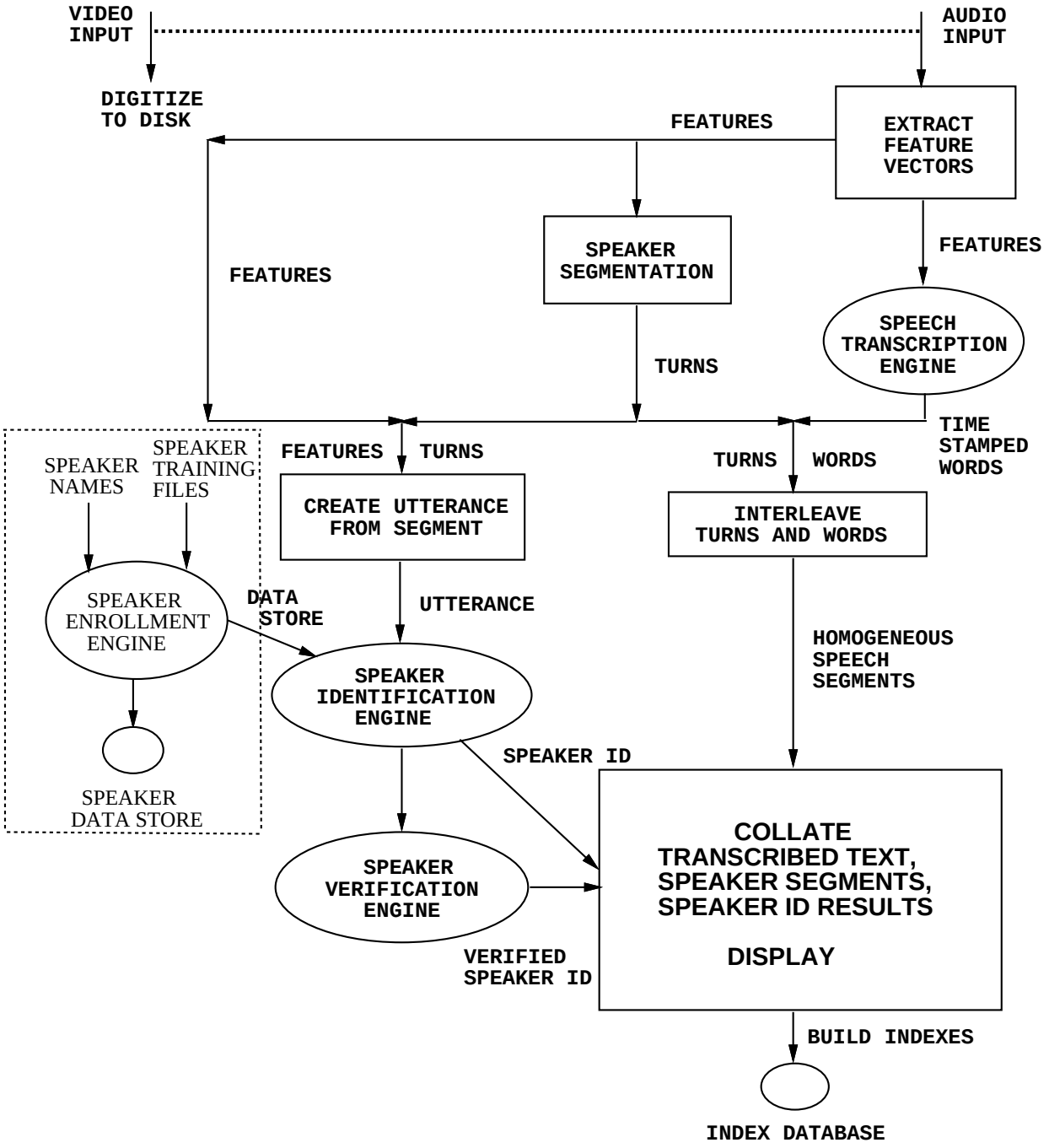

Figure 2: The architecture of a system for real-time speech recognition, speaker segmentation, and speaker identification for audio indexing and retrieval. This architecture is realized in an application which analyzes the input audio using three engines in real-time on a 400MHz Pentium-class machine. Indexing is triggered automatically when the audio terminates. Note, that the data store is generated off-line – indicated by the boxed area.

of the execution of our application. The figure shows the transcription, the result of the speaker segmentation as separate paragraphs, and the assigned labels on the left. The image thumbnails that appear at the left of the figure are single video frames of each speaker extracted from the video source to go with the assigned speaker label. This optional step is extraneous to the analysis, but tends to enhance the appearance of the user interface. The indexing process is completely automatic from beginning to end and executes without user intervention. The user is called upon to fill out some mandatory fields prior to the inception of the broadcast. This user information includes essentials such as index-name, date of broadcast, and so on.

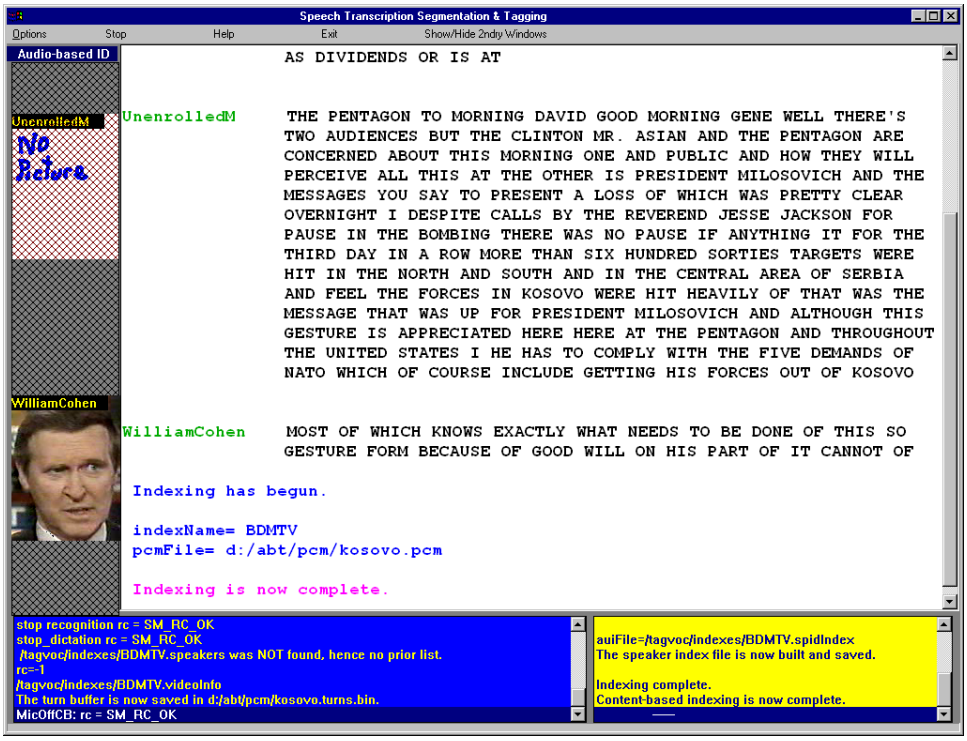

Figure 3: This plate shows a screen-shot of the application. The text corresponds to the real-time transcription of the input audio, the labels to the left are assigned by the speaker recognition module, and the paragraph-like breaks are introduced by the speaker segmentation module. Indexing is triggered when the audio is stopped. Indexing takes about 1-2% of the audio running time.

6 Retrieval Using Text and Speaker Information

6.1 Indexing for Text-Based Retrieval

The files that make up the text-based index are compiled after the audio has been completely transcribed using the speech recognizer. The time-alignments from the speech recognizer contain the start time of each word relative to the start of the audio or video clip and the duration of the word, both to hundredths of a second. (Part of the speech recognition API invocation involves using the operating system to produce a clock tick count just before the engine is ready to transcribe the first word. This clock time is then used as a base-line against which all other word-times are

aligned.) The entire transcript is then converted into text chunks of a fixed number of words (say, 100), each of which is considered a "document." The start and end times of each of such documents are recorded as these will be later used to retrieve the corresponding video segment from the user interface. Individual word times are also recorded for use by the speaker index and the user interface designer. After morphological analysis of each of these "documents," other statistics required by the Okapi equation are recorded, viz., term frequency and inverse document frequency. Other files include a standard 57,000 word vocabulary which is augmented by all the new words found in the recognized transcript. The time involved in generating the various index files (including the speaker index files) is around 1–2% of the time required in transcription which is real-time. The existing index can be augmented by processing new audio (or video) material using the APIs developed for this purpose. In order to update the index with additional audio files automatically, a separate file is maintained that registers the state of the index.

6.2 Indexing for Speaker-Based Retrieval

The index file for speaker-based retrieval is built after the input audio ends from the results of speaker identification and verification. Each identification result is accompanied by a score which is the distance from the enrolled speaker training model to the audio test segment, the start and end times of the segment relative to the beginning of the audio clip concerned, a label (the name of the speaker supplied during enrollment), and the name of the media source. All the segments assigned the same speaker label are gathered, then sorted by their scores and normalized by the segment with the best score. The segment with the best score is the one closest to the training model for that speaker. This step is repeated for each speaker identified in the audio. (Another approach to express the confidence in a recognized speaker in quantifiable terms would be to normalize across speakers. This can be done by running the training set as a test set, taking the mean of the top scores for all the correctly classified speakers, and then using this number to normalize all the scores.)

For every new audio clip processed by the system and added to the index, all the labeled segments, existing and the newly added, are gathered, re-sorted and re-normalized. The speaker index is therefore database of speakers, with multiple segments for each speaker. During retrieval, where the identity of the speaker is specified as part of the query request, the specified speaker's list of segments are selected from the database and scanned for the appropriate audio segment using the recorded start and end times. One view of a speaker index file is shown in Table 7.

6.3 The Retrieval System

Retrieval is performed by an engine with two distinct, non-overlapping steps, one for text-based and the other for speaker-based search. The two steps can be programmed to run concurrently using threads or processes since they are completely independent, however, in our implementation they run sequentially. Search requests can take one of three forms: search by text content, search by speaker name (matching the one provided during enrollment of the same speaker), or a search by text *and* speaker. The retrieval engine is primed by first loading into its memory the decision tree required for tokenizing the query, tables for part of speech tagging and morphological analysis, arrays containing the pre-computed portion of the Okapi formula, and other vocabularies and dictionaries. The engine is now ready for queries. Initializations take a few seconds on a 400MHz Pentium II personal computer with 64MB or more of RAM running Windows NT/95/98.

Each text-query phrase in the search string is tokenized word-by-word, appropriately stemmed and then scored against each of the fixed word-size documents in the index set using the index files

$Speaker_1$(name)	N_1(no. of segments)	start-time	end-time	score
1	$media_1$	t_a	t_b	S_{10}
2	$media_6$	t_e	t_f	S_{11}
3	$media_3$	t_c	t_d	S_{12}
$Speaker_2$	N_2	start-time	end-time	score
1	$media_4$	t_b	t_f	S_{20}
2	$media_6$	t_k	t_m	S_{21}

Table 7: Speaker index file layout and information. The index file consists of series of blocks of segments (the first three segments form a block for $Speaker_1$, while the next two form another block for $Speaker_2$), one for each speaker identified. Each speaker block is arranged by sorted match score. The first segment for $Speaker_1$ is therefore the closest to the speaker's training sample. For instance, when a new audio clip is processed, it might add one segment to the $Speaker_1$ block and two to $Speaker_2$ block. Hence, the four segments in the first block would have to be re-sorted by score, and then normalized by the new best scoring segment. This would be repeated for the four segments for $Speaker_2$.

for quick matching. (A vocabulary of words and an inverted index is generated when the indexes are generated in order to avoid scoring each query word against *all* the available documents.) All of the scored documents are then ranked and normalized with the most relevant document getting a match-score of 100. Generally, the top N documents alone are returned to the user. N has a default value of 10. Also available are the start and end times of each of the 10 documents in seconds (to two decimal places), the match-scores, and the matched words and morphs that contributed to the relevance score. (The last-mentioned is more useful to the user-interface builder.)

The user interface is a Java application on a PC platform that has two text fields, one for entering text queries and the other for specifying the desired speaker. An entry in either of the two text fields implies an isolated search in just that domain. Entries in both text fields triggers the combined search. The user interface has been engineered to accept speech queries as well if a microphone is attached. This system used the JavaSpeech API and runs the IBM ViaVoice Broadcast News Speech Recognition engine and IBM Text-to-Speech Synthesis (to echo the input queries back to the user). A simple dictation grammar is used to parse the voice input. After recognition, the recognized text is entered into the same text fields which would otherwise be used for typing the queries in. Figure 4 shows the query panel in the user interface.

If the query string contains an entry in the speaker name field, then the specified speaker string is compared a set of enrolled speaker list. If the name exists, the speaker index is searched and the top N segments are retrieved for that speaker. The segments are retrieved in the score-sorted order in which it is available in the speaker index. Also available for each segment is the name of the associated audio or video clip, start and end times, and a segment score (Figure 5). For each retrieved segment, the start and end times are used to recover the corresponding text from the text-based index file to create a "document" on the fly. This is then returned to the user. In the case of the text-based index each document is available at index time because of the fixed sized chunks that are used. The distinction here is that speaker segments can be of varied size and the compilation of the retrieved text is done only during retrieval.

Generally, for text string searching, the document length is fixed by a parameter set at index time such that it corresponds to a playing time of about 30 seconds. (This default playing time was arrived at because it conveys enough information with adequate context during playback of

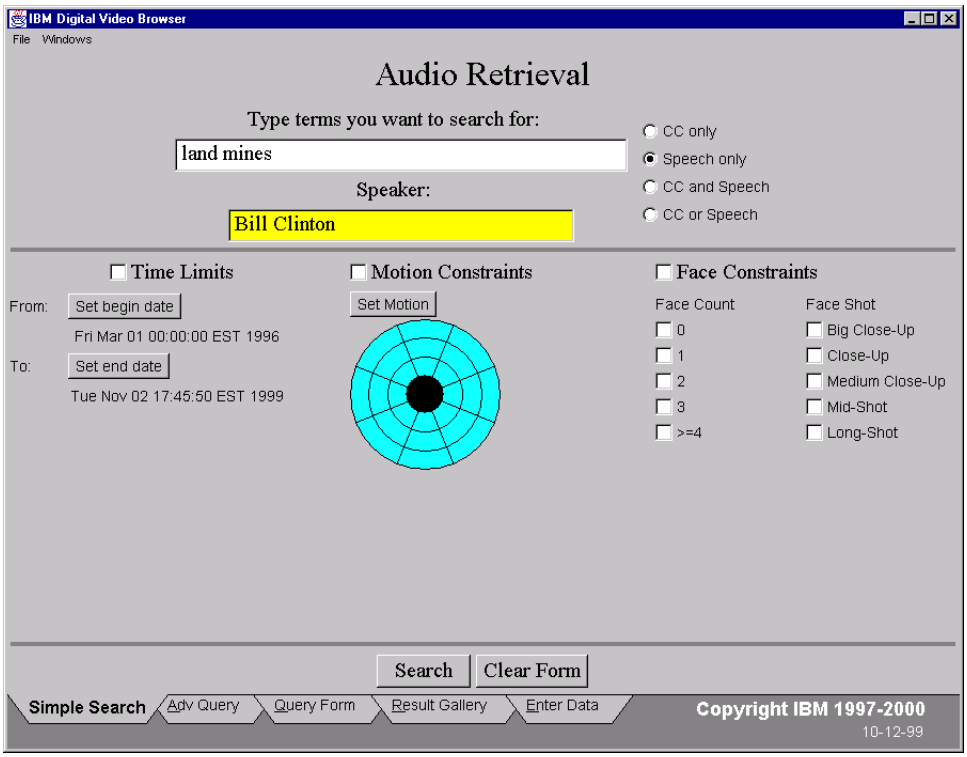

Figure 4: Query interface which permits text-based, speaker-based, or combined queries. Both these fields are voice-enabled. The other constraints shown have not been implemented.

broadcast news.) For speaker searching, the "document length" is determined at automatically during speaker segmentation. In order to avoid very long segments from being cued up for the media handler, the length of speaker segments are truncated at 60 seconds. Therefore, if $(endtime - starttime) > 60$, then $endtime = (starttime + 60)$. This can be modified via the the retrieval API.

The user interface is capable of showing all the relevant information for each of the N selections returned by the retrieval engine, and on further selection uses a media handler component, implemented using the Java Media Filter (JMF), to display MPEG-1 video (or audio only for radio broadcasts) via a VCR-like interface (Figure 6). The Java application is responsible for locating the video files (which can be on a server if the PC is networked), and then uses side information gathered during retrieval to embellish the results, such as displaying the retrieved document, the associated information such as media file name, start time, end time, rank, normalized score, a graphic view of where in the media file the retrieved segment lies, highlighting the query words and other morphs that contributed to the ranking of that document – this is significant only for text-based searching, or permitting highlighting of a portion of the displayed document for playback.

To play the video or audio clip corresponding to the retrieved document (or segment), the media handler component locates the media source file, advances to the start time specified, decompresses the stream (if required), and then initializes the media player with the first frame of the audio or video. The VCR-like interface permits the user to "play" just the retrieved video from start to finish or to stop and advance at any juncture.

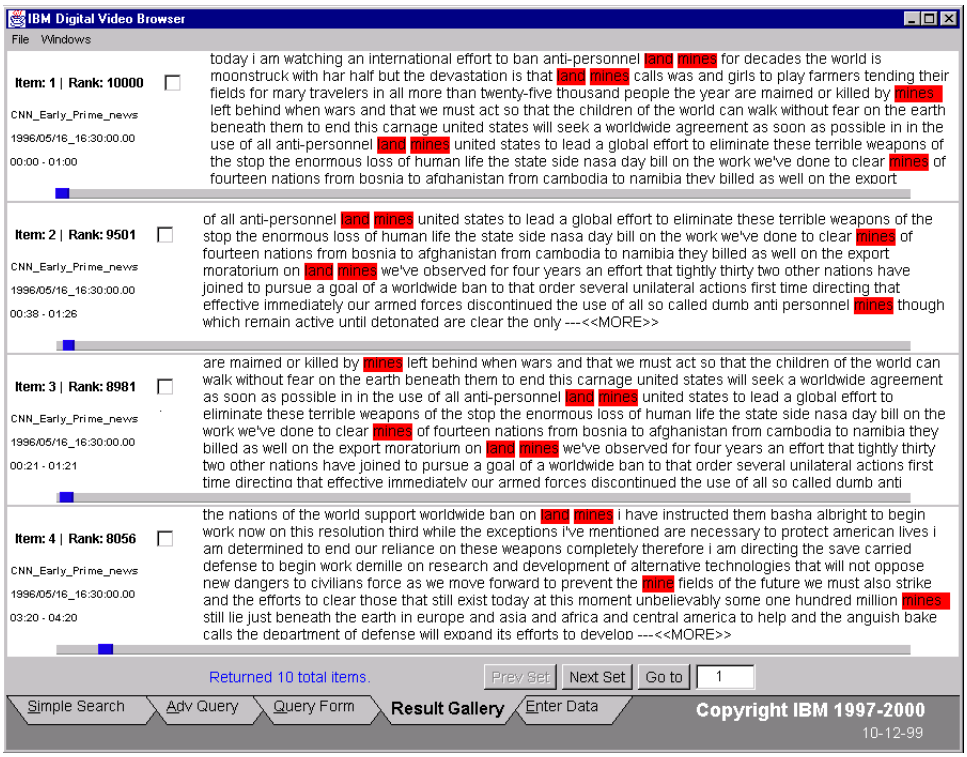

Figure 5: Summary panel in user interface which displays the top four selections out of a maximum 10 in response to the query in order of relevance. For speaker-only queries, the text of the audio portions which best match the speaker's training sample will be displayed in score sorted order. Note that for each document (segment), the name of the media source, the location of the displayed document (segment) within the media file, and the transcript is available. Clicking on the text in this panel provides the playback capability of the selected document (segment).

6.4 Combined Searching

How are retrieval results from two disparate search for text and speaker combined when queries contain both a text search string and a speaker. The user is looking for relevant audio/video segments which contain certain words spoken by a requested speaker. First, the text query is processed and *all* (not just the top N) the scored documents are collected. For the specified speaker, all the segments' information is available from the speaker index. The common elements between the segments for text-based search and speaker-based search are the start and end times – of document chunks in the text-based index, and segments in the speaker index respectively. If there is a time overlap between a document from the text search and a segment from the speaker search (when derived from the same audio/video source), then that portion of the audio stream satisfies the composite query. The degree to which the two segments overlap is also significant as are the match-scores of the two segments. A single segment from speaker retrieval may overlap with multiple segments from text retrieval or vice versa.

The algorithm to compute the combined score from the two individual text search and speaker search is as follows. For each document from the text search results, run down the segments for the specified speaker computing time overlaps given by: $C_s = (c_s + (\lambda * s_s)) * (o_f)$, where c_s is the score

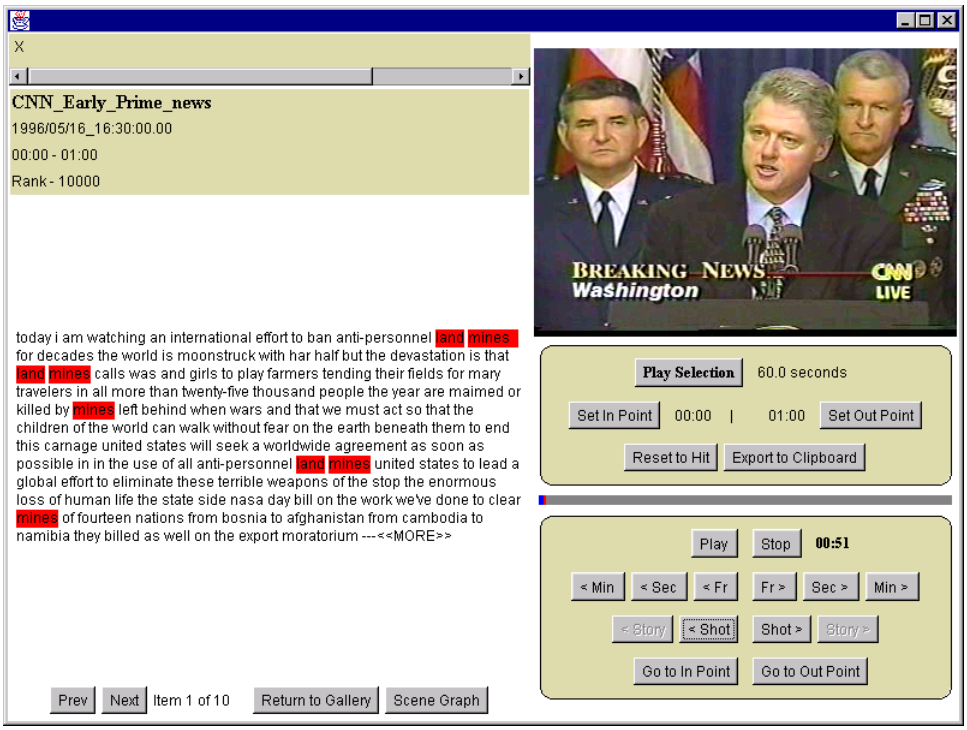

Figure 6: The VCR-like interface which permits playback of selected entity. This panel provides all the known information about the selected document (segment) – media name, location within media file, normalized score relative to other relevant documents retrieved, portion of transcript corresponding to document (segment) start and end times, and access to the audio/video source via the playback facility. Using "Play" instead of "Play Selection" enables video playback beyond the retrieved segment.

for the retrieved document from the text-based search, s_s is the score for the speaker segment, o_f is a fraction ($0 < o_f < 1$) that specifies by how much the speaker segment overlaps with the result of the text-based search and C_s is the combined score. λ is a factor which handicaps s_s based on confidence in the speaker scores. Currently, a λ of 0.75 is used. (When better cohort models in the speaker recognition system and more speakers become available, λ will be closer to 1.) The resulting combined scores are sorted and normalized so that the best score is 100. The retrieval engine then returns the requisite information about the top N composite segments to the caller for display to the user. The default start time of each combined result is the same as the start time for the corresponding document from the text-based search. (The other choice is to use the start time of the speaker segment.) The end time is set to the end of the speaker segment (to let the speaker finish the statement). However, for usability reasons, all of these alternatives can be tailored via API calls.

6.5 Results

We ran experiments on the entire system from initial audio analysis, index building, and then retrieval using five hours of broadcast news video data digitized from a VHS tape. (The amount of video data used was restricted simply because of the paucity of publicly distributed and freely

usable broadcast news video material.) The video and audio were digitized separately, the video using a MPEG-1 hardware encoder, and the audio at 22 KHz (PCM) which was streamed into the indexing application. Forty speakers were selected from the video material and enrolled into the speaker recognition system.

Each speaker was enrolled with 30 seconds of data from a single conversation or monologue derived from the training video clips. Two 30-minute video segments were set aside as test data. For speaker-based retrieval, the top 10 segments for 10 of the enrolled speakers were retrieved using both identification *and* verification. (The results show the sensitivity this system exhibits to channel acoustics. The degradation of speaker retrieval performance from expected levels is entirely due to channel mismatch conditions.) For text search, 20 user-defined queries from the video were used. The top 5 and top 25 results are presented below. Queries like "defense secretary" returned three hits for the whole phrase, with the remaining seven pertaining to just the word "defense" or "secretary." These were considered irrelevant for the purpose of scoring. In the combined search, errors were both due to speaker mis-identifications and irrelevant documents being retrieved. Sample queries include "land mines/Bill Clinton" and "Boris Yeltsin/Natalie Allen". The results are shown in Table 8.

Search	Relevant/Retrieved
Speaker	77/99
Text Top 5	198/200
Text Top 25	143/200
Combined Top 5	51/62
Combined Top 10	78/93

Table 8: Retrieval performance as measured by the number of relevant documents retrieved over 20 textual queries, 10 speakers, and 10 combined text-speaker queries. The speakers were enrolled using 30 seconds of voice data from a single conversation, and not from multiple sources to cover possible channel mismatch conditions.

7 Conclusion and Further Work

We have described our experience with large vocabulary speech recognition and speaker recognition for multimedia document retrieval. We have some good results with respect to our primary goal of extracting information from audio material using imperfect automatic transcription and speaker recognition, indexing it, and then running search queries against it. Both in the general version and in domain-specific versions, speech recognition accuracy tends to improve year after year. Already we deal only with systems that perform in real-time or better, removing one major hurdle to building practical and useful real-world applications.

The system we have built is modular and each module is a full-fledged PC-based application in its own right. The indexer consists of the audio analysis component with concurrent real-time speech recognition, speaker segmentation and speaker identification-verification functions, and the index compiling component. The audio analysis component runs apace with the input audio, and the index preparation takes about 1-2% of the duration of the audio processed. These index files are written to disk or stored in a database. From the inception of the audio analysis to index building and storage the system is fully automatic and functions without user intervention. The retrieval

system consists of a engine and a user interface. The user interface accepts queries from the user and returns responses back to the user with pointers to the actual video or audio clip relevant to the query. The retrieval engine uses the index and feeds the user interface with the appropriate information.

Further improvements can be made within the context of our approach to text-based information retrieval from audio. The current set of documents derived from the speech recognition output can be augmented by including the next-best guesses for each word or phrase from the recognizer. This information can be used for weighting the index terms, query expansion, and retrieval. Also, better recognition accuracy can be had by detecting segments with music or mostly noise so that only pure speech is indexed for retrieval. One limitation with the current approach to audio-indexing is the finite coverage of the vocabulary used in the speech recognizer. Words such as proper nouns and abbreviations that are important from an information retrieval standpoint are often found missing in the vocabulary and hence in the recognized transcripts. One method to overcome this limitation is to complement the speech recognizer with a word-spotter for the out of vocabulary words [Dharanipragada99b]. For this approach to be practical, however, one has to have the ability to detect spoken words in large amounts of speech at speeds many times faster than real-time. In speaker identification, the speed advantage of model-based over frame-based increases with utterance length and enrollee population. Its accuracy improves and approaches that of frame-based as test utterance duration increases. Thus the issue of which scheme to use depends on the problem at hand. In training, since more data is generally available, the model-based approach is preferred both for its accuracy and speed of comparisons.

A technique for combining results from two different modes of processing audio (and video) has been presented. Additionally, the design and implementation of the system and the application has also been described. Our results for the combined retrieval across systems is based on only five hours of video. However, the techniques presented here are scalable and larger video libraries can be indexed for search and retrieval using this system.

References

[Akaike74] H. Akaike, "A New Look at the Statistical Model for Identification," *IEEE Transactions on Automatic Control, AC–19*, 1974, pp. 716–723.

[Bahl94] L.R. Bahl, P.V. deSouza, P.S. Gopalakrishnan, D. Nahamoo, and M.A. Picheny, "Robust Methods for Context-dependent Features and Models in a Continuous Speech Recognizer," in *Proceedings, International Conference on Acoustics, Speech, and Signal Processing*, Adelaide, Australia, 1994, pp. 533–536.

[Bahl95] L.R. Bahl, S. Balakrishnan-Aiyer, J.R. Bellegarda, M. Franz, P.S. Gopalakrishnan, D. Nahamoo, M. Novak, M. Padmanabhan, M.A. Picheny, and S. Roukos, "Performance of the IBM Large Vocabulary Continuous Speech Recognition System on the ARPA Wall Street Journal Task," *Proceedings, International Conference on Acoustics, Speech, and Signal Processing*, Detroit, Michigan, 1995, pp. 41–44.

[Beigi98a] H.S.M. Beigi, S. Maes and J.S. Sorenson, "A Distance Measure Between Collections of Distributions and Its Application to Speaker Recognition," *Proceedings, International Conference on Acoustics, Speech, and Signal Processing*, Seattle, Washington, 1998, pp. 753–756.

[Beigi98b] H.S.M. Beigi, S. Maes, U.V. Chaudhari and J.S. Sorenson, "IBM Model-Based and Frame-By-Frame Speaker-Recognition," *Proceedings, Speaker Recognition and its Commercial and Forensic Applications*, Avignon, France, 1998.

[Chaudhari99] U.V. Chaudhari, H.S.M. Beigi, S.H. Maes, and J.S. Sorensen, "Multi-environment Speaker Verification," *Proceedings, 2nd Annual Workshop on Automatic Identification Advanced Technologies (Auto ID '99)*, Summit, New Jersey, 1999, pp. 19–22.

[Chen98] S.S. Chen and P.S. Gopalakrishnan, "Speaker, Environment and Channel Change Detection and Clustering Via the Bayesian Information Criterion," *Proceedings, DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, Virginia, 1998, pp. 127–132.

[Chen99] C.J. Chen, "Speech Recognition with Automatic Punctuation," *Proceedings, EuroSpeech99*, Budapest, Hungary, 1999, pp. 447–480.

[Delacourt99] P. Delacourt, D. Kryze, and C.J. Wellekens, "Speaker-based Segmentation for Audio Data Indexing," *Proceedings, EuroSpeech99*, Budapest, Hungary, 1999, pp. 1195–1198.

[Dharanipragada97] S. Dharanipragada and S. Roukos, "Experimental Results in Audio-Indexing," *Proceedings, DARPA Speech Recognition Workshop*, Chantilly, Virginia, 1997, pp. 2–5.

[Dharanipragada99a] S. Dharanipragada, M. Franz, J.S. McCarley, S. Roukos, and R.T. Ward, "Story Segmentation and Topic Detection for Recognized Speech," *Proceedings, EuroSpeech99*, Budapest, Hungary, 1999, pp. 2435–2438.

[Dharanipragada99b] S. Dharanipragada, M. Franz, and S. Roukos, "Audio Indexing for Broadcast News," *Proceedings, Seventh Text REtrieval Conference (TREC-7)*, E.M. Voorhees and D.K.Harman, editors, NIST Special Publication 500–242, 1999, pp. 115–120.

[Furui89] S. Furui, *Digital Speech Processing, Synthesis and Recognition*, Marcel Dekker, 1989.

[Gish91] H. Gish, M. Siu, and R. Rohlicek, "Segregation of Speakers for Speech Recognition and Speaker Identification," *Proceedings, International Conference on Acoustics, Speech, and Signal Processing*, Seattle, Washington, 1991, pp. 873–876.

[Gish94] H. Gish and M. Schmidt, "Text-Independent Speaker Identification," *IEEE Signal Processing*, Volume 11, Number 4, 1994, pp. 18–32.

[Gopalakrishnan95] P.S. Gopalakrishnan, L.R. Bahl and R. Mercer, "A Tree Search Strategy for Large Vocabulary Continuous Speech Recognition," *Proceedings, International Conference on Acoustics, Speech, and Signal Processing*, Detroit, Michigan, 1995, pp. 572–575.

[Grosky97] W. Grosky, "Pushing Streaming Video – Indexing Video Archives," *IEEE Multimedia*, October-December 1997, pp. 7–8.

[Hirschberg99] J. Hirschberg, S. Whittaker, D, Hindle, F. Periera, and A. Singhal, "Finding Information in Audio: A New Paradigm for Audio Browsing and Retrieval," *Proceedings, ESCA Tutorial and Research Workshop*, Cambridge, United Kingdom, April 1999.

[Liu99] D. Liu and F. Kubala, "Fast Speaker Change Detection for Broadcast News Transcription and Indexing," *Proceedings, EuroSpeech99*, Budapest, Hungary, 1999, pp. 1031–1034.

[Pallet97] D. Pallet, "Overview of the 1997 DARPA Speech Recognition Workshop," *Proceedings, DARPA Speech Recognition Workshop*, Chantilly, Virginia, 1997, pp. 1–2.

[Rabiner93] L. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.

[Robertson95] S.E. Robertson, S. Walker, K. Sparck-Jones, M.M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," *Proceedings, Third Text REtrieval Conference (TREC-3)*, D.K. Harman, editor, NIST Special Publication 500–226, Gaithersburg, Maryland, 1995, pp. 109–126.

[Rosenberg92] A.E. Rosenberg and F.K.Soong, in *Advances in Speech Signal Processing*, S. Furui and M.M. Sondhi, editors, Marcel Dekker, 1992, pp. 701–738.

[Salton89] G. Salton, *Automatic Text Processing*, Addison-Wesley, 1989.

[Satoh99] S. Satoh, Y. Nakamura, and T. Kanade, "Name-It: Naming and Detecting Faces in News Videos," *IEEE Multimedia*, Volume 6, Number 1, January-March 1999, pp. 22–35.

[Srinivasan99] S. Srinivasan, D. Petkovic, D. Ponceleon, and M. Viswanathan, "The CueVideo Spoken Media Retrieval System," *IBM Research Report RJ 10143 (95018)*, April 22, 1999.

[Tritschler99] A. Trischler and R.A. Gopinath, "Improved Speaker Segmentation and Segments Clustering Using the Bayesian Information Criterion," *Proceedings, EuroSpeech99*, Budapest, Hungary, 1999, pp. 679–682.

[Viswanathan99] M. Viswanathan, H.S.M. Beigi, S. Dharanipragada, and A. Tritschler, "Retrieval from Spoken Documents Using Content And Speaker Information," *Proceedings, International Conference on Document Analysis and Retrieval (ICDAR99)*, Bangalore, India, 1999, pp. 567–572.

[Wold96] E. Wold, T. Blum, and D. Keislar, "Content-based Classification, Search, and Retrieval of Audio," *IEEE Multimedia*, Volume 3, Number 3, Fall 1996, pp. 27–36.

Biographies

Mahesh Viswanathan is a Research Staff Member with the Human Language Technologies department at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. His research interests include multimedia indexing and retrieval, information retrieval, speech recognition, and document analysis. Viswanathan has a BS in Physics from Loyola College, Madras, India, BS in Electrical Engineering from the Indian Institute of Science, Bangalore, India, MS in Electrical and Computer Engineering from San Diego State University, San Diego, California, and PhD in Computer and Systems Engineering from Rensselaer Polytechnic Institute, Troy, New York. He has worked for IBM since 1991 in research, advanced technology, and product management and is a member of the IEEE.

Homayoon S.M. Beigi received his B.S., M.S. and D.Eng.Sc. from Columbia University in 1984, 1985, and 1990 respectively. He has been a Research Staff Member at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, since 1991 and is currently on a one-year leave of absence. He is also an Adjunct Associate Professor in both the EE and ME departments at Columbia University, New York. Beigi has published in the fields of Kinematics, Neural Networks, Optimization, Learning Control, Signal Processing, Image Compression, Handwriting, Speech and Speaker Recognition. He is Associate Editor of the Intelligent Automation and Soft Computing and is on the technical boards of three major international conferences.

Satya Dharanipragada received his B.Tech. degree from the Indian Institute of Technology, Kharagpur, India, his M.S. and Ph.D. degrees from the University of Illinois at Urbana-Champaign, all in electrical engineering, in 1988, 1990, and 1994, respectively. During 1994-1995 he was a post-doctoral fellow at the Center for Language and Speech Processing at the Johns Hopkins University, Baltimore. He is currently a Research Staff Member in the Human Language Technologies department at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. His research interests include robust speech processing and speech recognition, multimedia processing, constrained signal and image reconstruction, and image processing.

Fereydoun Maali is President of Signal Recognition Corporation, a Manhattan-based software engineering firm specializing in image processing and industrial vision. He has acted as consultant in image processing and related areas to a number of corporations including IBM, Nynex Science & Technology, United Technologies Northern Systems, RVSI, and Universal Instruments. He was earlier involved in research and development in image processing and robotic vision for Robotic Vision Systems, New York, and Vickers (Joyce-Loebl Div.), UK. Until 1979 he was a Commander in the Imperial Iranian Navy. He has a DIC, MSc, and PhD from Imperial College, U.K., and is a Chartered Electrical Engineer. He has received 5 US patents.

Alain Tritschler received his MS degree in Multimedia Communications in 1998, with emphasis on speech recognition, image analysis, and networking, jointly with the Institut Eurocom, Sophia-Antipolis, France, and the Institut National des Telecommunications, Evry, France. He joined the Human Language Technologies department at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, in 1998 and worked there until December 1999. Since January 2000, he has been working in echo and noise cancellation in wireless telephony in France. His current interests include speaker segmentation and clustering, speech recognition under noisy acoustic conditions, and multimedia retrieval.