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
PO Box 218, Yorktown Heights, NY 10598, USA
maheshv@watson.ibm.com

November 10, 1999

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 retrieval 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-based analysis consisting of the segmentation of audio into acoustically homogeneous sections followed by speaker identification. The output of these two processes is used to abstract statistics to automatically build indexes for text-based and speaker-based retrieval. The real power of multimedia document processing is in 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 retrieval. The experiments described here are in the broadcast news domain, but the underlying techniques can easily be extended to other speech-centric applications and transactions.

Keywords: audio indexing, multimedia document retrieval, 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 information 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 and 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.

*Submitted to IJDAR

†Signal Recognition Corporation, PO Box 7010, New York, NY 10128, USA

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 very swift index generation step after the audio terminates. The indexing subsystem is completely automated. Since the index formation has to be complete before use, the indexing subsystem as a whole is considered to be an 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 return the results back to 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, transcribed text of the audio is also available.

Conventional information retrieval has mainly focused 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 may be generated using speech recognition (and the speaker labeled using speaker recognition) and the turns (and segments) so derived can be used for indexing the associated audio (and video).

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.

For issues relating to building user interfaces for audio browsing and retrieval systems, see [Hirschberg99]. An editorial review contrasts two video archival indexing systems in [Grosky97]. The first system uses close captioning information to build a video index. The second system includes indexing of video content and implying 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. describes a system for search, identification and retrieval of audio based on a host of features that define and characterize sound in an intuitive manner [Wold96]. The emphasis is on using acoustic and perceptual features for short and single-gestalt sound retrieval.

Cue Video 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 was emphasized in [Delacourt99]. A two-pass approach is used - using a distance-based measure for speaker change detection, followed by BIC-based validation. Another

BIC-like speaker segmentation scheme is also reported in [Liu99]. However, they give no details on speaker indexing. A preliminary version of our multimedia retrieval system is presented in [Viswanathan99].

The rest of the paper is organized as follows. A description of the speech recognition and information retrieval systems that constitute this audio indexing system follows. The tests that were conducted to evaluate just this subsystem are presented next. The following section describes a speaker segmentation algorithm which is a prerequisite step to speaker identification and verification. Two different algorithms for speaker recognition are discussed along with the implementation details and performance of of this subsystem. 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 retrieval is discussed along with some results.

3 Text-Based Information Retrieval

The current text-based information retrieval system consists of two 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 on a Hub4 corpus. 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 [Bahl94, Bahl95]. 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 [Gopalkrishnan95]. 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 1-state Hidden Markov Model 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 and the 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.

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 prepared and spontaneous speech training data and 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 65K most frequent words from the 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 mismatch 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 on a 9-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 runs using the broadcast news systems 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%.

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.

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 which deduces 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 the broadcast news domain which targets a listening and viewing audience and further handles live audio or video feeds.) 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 taken.) 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 offline indexing phase, where relevant statistics about the textual documents are gathered to build an index, and an on-line searching and retrieving phase, where the index is used to perform fast query-document matching followed by returning the top 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 words, is processed to derive a document description which 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 performance as measured by retrieval speed 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:

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

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, $\bar{l}$ is the average length of the documents in the collection, and $idf(q_k)$ is the inverse document frequency 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 inverse document frequency term thus favors terms that are rare among documents. (For unigrams, $\alpha_1 = 0.5$ and $\alpha_2 = 1.5$.) 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 over a span of 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.

The entire speech collection is recognized with a large vocabulary speech recognizer to produce transcripts with time-alignments for each word in the transcript. Unlike a standard information retrieval scenario, there are no distinct documents in the transcripts and therefore one has to be artificially generated. The simple solution used was 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 potential problem if query words span two documents. (An alternative to fixed-sized document segments is to use topic identification schemes for subject classification [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].

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.

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%.

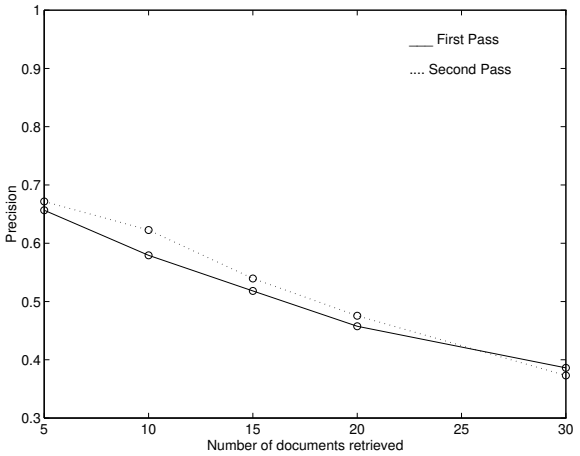


Figure 1: Precision vs Number of Retrieved Documents.

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 shown in Table 3.

4 Speaker-Based Information Retrieval

The current speaker-based information retrieval system consists of two components: (1) an acoustic-change detection system also called speaker segmentation, and (2) a speaker-independent, 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 for audio-indexing systems, it is necessary to detect the boundaries (turns) between non-homogeneous

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.

speech portions. An example of early 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 full 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:

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

where, N is the number of frames in the window W, i is the number of frames of the first part of the window, and $(N - i)$ is the number of frames of the second part; 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 W for obvious practical reasons. Instead the window W is divided into coarse 50-frames (detection resolution) subsegments for a fixed maximum of 10 subsegments - as windows are not permitted to grow beyond a pre-defined threshold (500 frames). ΔBIC is only computed at the boundaries of these subsegments.

- Once a turn is coarsely detected, the ΔBIC is recomputed over a 100-frame window centered on the frame computed in the coarse turn determination step (above). A finer turn resolution is obtained by computing ΔBIC every 10 frames within this refined 100-frame window.

- The buffer is then refreshed to collect the N frames for the next window W.

- Let r denote the rth window considered for turn detection. Then, the number of frames in window W grows according to:

$$N_r = min(N_{r-1} + \Delta N_r,\ 500)\ ,\ r = 1, 2, ..9 \tag{2}$$

- The following rule governs the growth of N itself

$$\Delta N_r = \begin{cases} \Delta N_0 & \iff \Delta BIC < 0 \\ \Delta N_0 + 2^{(r-1)}\delta_0 & \iff \Delta BIC \geq 0 \end{cases} \tag{3}$$

  with $\delta_0 = 50$ frames. The following governs the changes in r.

$$r = \begin{cases} 1 & \iff & \Delta BIC < 0 \\ (r+1) & \iff & \Delta BIC \geq 0 \cap r < 9 \\ r & \Delta BIC \geq 0 \cap r \geq 9 & \end{cases} \tag{4}$$

- On the other hand, if no turn is revealed the window grows to a maximum of 10 sub-segments of 50 frames each. From then on the window size is retained until a turn is detected. Meanwhile, the least recently visited sub-segment is dropped from the list of sub-segments subjected to BIC testing.

- W can grow to a maximum size of 500 frames with each sub-segment not exceeding 50 frames (10 sub-segments). When the window grows beyind this limit, the statistics of all preceding frames are merged with that of the first sub-segment. In other words, the window W still consists of 10 sub-segments, but with an extra large first sub-segment - one that may exceed the prescribed 50 frames limit.

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. The resolution is generally around $\pm$ 10 frames, but in some rare cases extends to $\pm$ 75 frames.

4.2 Speaker Identification

Speaker recognition has two principal components, speaker identification and speaker verification. For examples of prior work in speaker recognition, see [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 are 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 and is SVAPI compliant.

In order to be able to index an audio stream by speaker, a database of 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 offline process and our audio indexing system assumes such a database exists for all speakers of interest. About a minute's worth of audio is required from each speaker from multiple channels and microphones encompassing multiple acoustic conditions. The 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. Each speaker model is represented by its mean, covariance, and the 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 a frame-based or model-based scheme. Speaker models by themselves will not suffice for the verification phase - a binary speaker tree is also needed which is built during enrollment.

The binary tree is constructed bottom-up with speaker models as the leaves of the tree. First, the leaves of the tree are arranged in such a way that the distance between successive model pairs

grows with the model similarity measure specified in [Beigi98a]. Each model pair is next merged into a higher level node. The resultant node, modeled much like its children, will have the same number of distributions as those of its children. The statistics of its constituent distributions is derived by first sorting and ranking the distribution pairs of the node's children using the Kullback-Liebler distance and then merging the statistics of the equi-rank distribution pairs. In the same fashion, the resulting nodes are merged into higher level nodes terminating when the root node is created. 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 $\{M_i \mid i = 1..I\}$ denote the models pertaining to the enrollees. Each model M_i can have N_i distributions associated with it. Let ω_{ij} refer to the jth distribution of model i. Also, let $\{\vec{x}_t \mid t = 1..T\}$ denote the frames constituting the test utterance, z, whose label is sought. During run-time, a test utterance $\mathbf{z}$ is identified with model q according to:

$$assign \;\; \mathbf{z} \longrightarrow M_q \;\; iff \;\; D_q = \min_{i=1..I} [D_i],$$

where

$$D_i = \sum_{t=1}^{T} d(i,t), \;\; i = 1..I,$$

and

$$d(i,t) = -log[\sum_{j=1}^{N_i} P(\omega_{ij}) \; p(\vec{x}_t|\omega_{ij})],$$

with $P(\omega_{ij})$ being the prior of the jth distribution of model i, and $p(\vec{x}_t|\omega_{ij})$ being the conditional pdf of utterance conditioned on the jth component of model i. Then,

$$p(\vec{x}_t|\omega_{ij}) = \frac{e^{-\frac{1}{2}(\vec{x}_t - \vec{\mu}_{i,j})^t \Sigma_{i,j}^{-1} (\vec{x}_t - \vec{\mu}_{i,j})}}{(2\pi)^{d/2} |\Sigma_{i,j}|^{1/2}}$$

Each class assignment is accompanied by a pseudo-distance (or score) which expresses the respective confidence. For model i, the score is: $score = -D_i over T$, where T 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 using the same k-means clusterer of the enrollment phase. The proximity of the generated test model to the prototype models is next examined using the inter-model distance in [Beigi98a]. The candidate speakers are next ranked in order of their proximity to the test model. 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 indexing and retrieval phase.

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.) With longer utterances (upwards of 15 seconds or so), the model-based scheme approaches the performance of the frame-based 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 pursued as a post-identification process. The speakers labeled during identification are claimants during the verification process. Let us introduce a 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 in here are specified through 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 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 (represented by the respective frame sequence) is transformed into a test model, as in the model-based identification above. The claimed identity is verified when this test model exhibits the shortest inter-model distance to the claimed model when compared with: one, the rest of the cohort set; two, the complementary set (to be described); and three, the background model if available. Otherwise, the claimed identity is rejected. The complementary model comprises the nodes that represent the tree minus the cohort set. A background model is a model representing speakers that are not of interest and hence not included in the speaker tree.

4.2.5 Results

About 30 seconds of speech from each speaker in the training set 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 for avoiding contamination with adjacent speakers. In model-based identification, we require 10 seconds of speaker data for the same test. Two different tests were run with both the model-based and frame-by-frame approaches on a database with 199 speakers. For a detailed set of results for the speaker recognition engine, see [Beigi98b].

The speaker recognition engine was first tested on 104 individual PCM files using the enrolled database of 199 speakers. The test data was derived from broadcast news sources in the USA, 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.

For speaker-based indexing a more realistic test 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.

Test Result	Frame-based	Model-based
Identified	101/104 (97%)	92/104 (88.5%)
Mis-identified	3/104	12/104
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 test results on 199 individual PCM files. Verification can either hold up the identification result or reject it. Here, in the model-based approach, one correct identification result was negated. Out the 12 mis-identifications, six were upheld, while six were (erroneously) not.

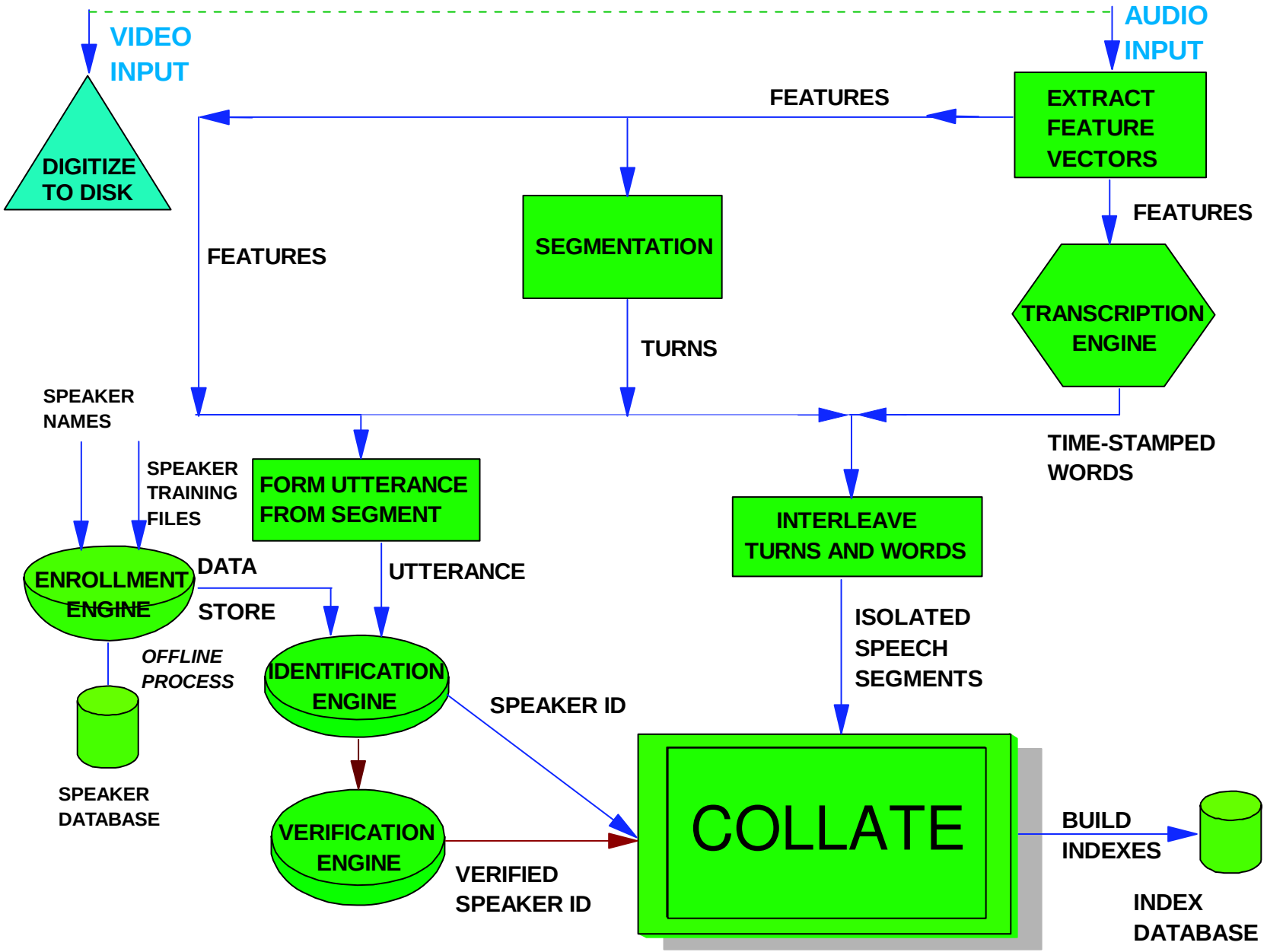
Speaker recognition systems are subject to channel mismatch conditions which can drop the identification rate to around 25mismatch. Channel mismatch conditions can occur for a host of reasons including microphone characteristics, room or outdoor acoustics, background noise, and other similar conditions. This is 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 [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.

5 System Architecture and Application

Speech and speaker recognition modules are quite compute intensive and yet it is desirable that the processing time for audio indexing be realistic. Figure 5 shows the architecture for the system 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 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 engines above in a multi-process and multi-threaded programming environment. The three engines are all programmable via application programming interfaces called SMAPI, SEGAPI, and SVAPI.

The engine used in our application is the 1998 update of the broadcast news transcription engine described in the first sections of this paper. The time-stamp at the start of each transcribed word from the speech recognition is recorded along with the duration of the spoken word. This time-stamp is relative to the beginning of the audio stream. 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" in SVAPI-parlance (800 frames for the frame-based analysis). The utterance is then scored against the speaker data store returning a label, which is then verified against the cohort set. This process


VIDEO INPUT
AUDIO INPUT
DIGITIZE TO DISK
FEATURES
EXTRACT FEATURE VECTORS
FEATURES
SEGMENTATION
FEATURES
TRANSCRIPTION ENGINE
TURNS
SPEAKER NAMES
SPEAKER TRAINING FILES
FORM UTTERANCE FROM SEGMENT
TIME-STAMPED WORDS
INTERLEAVE TURNS AND WORDS
ENROLLMENT ENGINE
DATA
STORE
UTTERANCE
OFFLINE PROCESS
IDENTIFICATION ENGINE
ISOLATED SPEECH SEGMENTS
SPEAKER ID
SPEAKER DATABASE
VERIFICATION ENGINE
COLLATE
BUILD INDEXES
VERIFIED SPEAKER ID
INDEX DATABASE

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 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.

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 2 is a snapshot from our application after analysis and indexing exhibiting the remnants 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. Single frames of each speaker were separately extracted from the video source to enhance the appearance of the user interface, but this optional step is extraneous to the analysis. 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.

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. As mentioned before, the recognizer generates words along with time-alignments for each word. These time-alignments 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 words are time-aligned.) The entire transcript is then converted into segments of a fixed number of words each of which is 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 user interface designer. After morphological analysis of each of these "documents" other

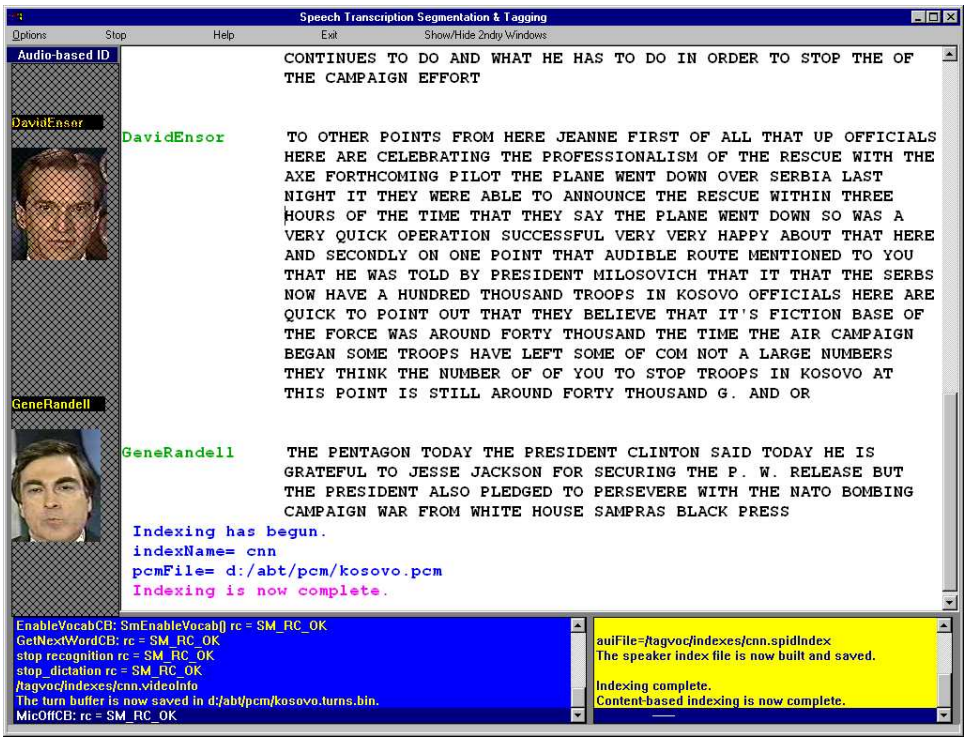


Figure 2: This plate shows a screen-shot of the application. The text corresponds to the real-time transcription of the input audio, the labels to the left are assigned by the speaker recognition module, and the paragraph-like breaks are introduced by the speaker segmentation module. Indexing is triggered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

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 augmented by the new words found in each recognized transcript. In order to update the index with additional audio files, a separate file that registers the state of the index is maintained. Indexing is an offline process which in our system implementation includes speech recognition and transcription generation. The time involved in generating the various 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.

6.2 Indexing for Speaker-Based Retrieval

The index file for speaker-based retrieval is built by taking a second pass over the results of speaker identification and verification. Each identification result is accompanied by a score which is the distance from the original enrolled speaker model the audio test segment, the start and end times of the segment relative to the beginning of the audio clip concerned, and a label (the name of the speaker supplied during enrollment). For any given audio clip, all the segments assigned the same speaker label are gathered. They are then sorted by their scores and normalized by the segment with the best score. The segment with the best score is closest to the training sample 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 again 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.

$Speaker_1$(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. Hence, the first segment for $Speaker_1$ is closest to the speaker's training sample. When a new audio or video 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 have to be re-sorted by score, and then normalized by the best scoring segment. Repeat for the four segments for $Speaker_2$.

6.3 The Retrieval System

Retrieval is performed by an engine with two distinct, non-overlapping modules, one for text-based and the other for speaker-based retrieval. The two modules 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 three forms: search by text content, search by speaker which entails specifying a speaker by 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, 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 or 95/98.

In our implementation, 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. (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 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 that contributed to the relevance score. (The last-mentioned is more useful to the user-interface builder.)

We have developed an user interface, a Java application, on a PC platform that has two text fields, one for entering text queries for text-based search and the other for specifying the desired speaker for speaker-based search. An entry in either of the two text fields implies an isolated search on only 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 Synthesizer (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 3 shows the query panel in the user interface.

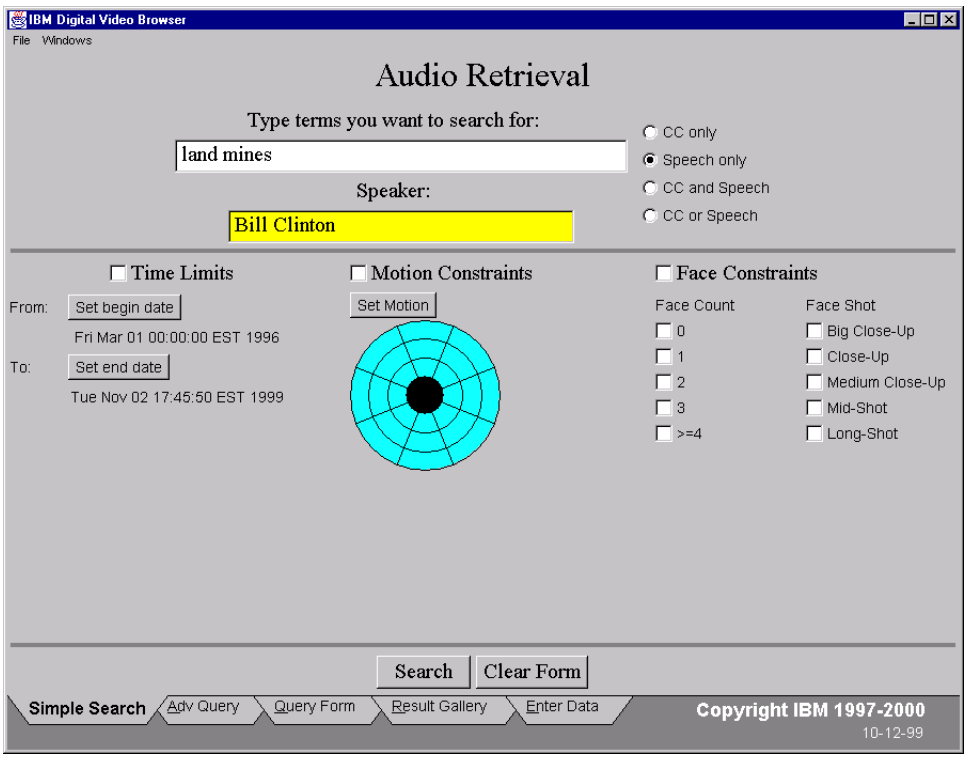


Figure 3: Query interface which permits text-based, speaker-based, or combined queries. Both these fields can also be voice-enabled.

If the query string contains an entry in the speaker name field, then the specified speaker string is compared a set of enrolled speakers' names 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 4). 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.

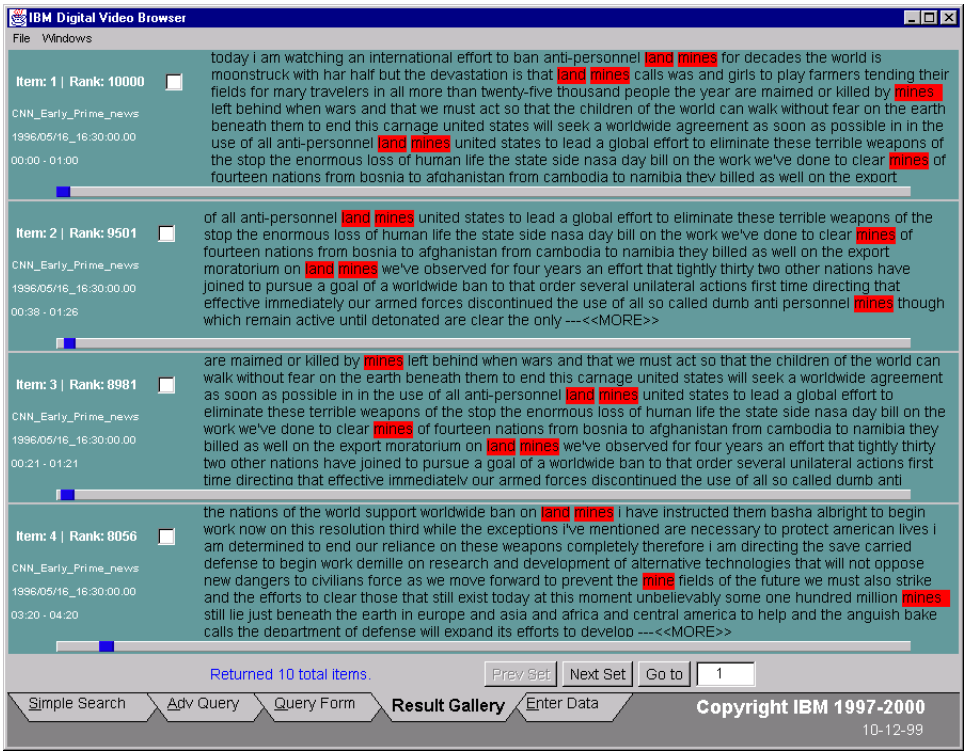


Figure 4: Summary panel in user interface which displays the top four selections in response to the query in order of relevance. In the case of speaker-only queries, the text of the audio portions which best match the speaker's training voice-print will be displayed in score sorted order. Note that for 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).

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, to display MPEG-1 video (or audio only for radio broadcasts) via a VCR-like interface (Figure 5). 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, 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.

Generally, for text string searching the document length (a parameter set at index time) is such that it corresponds to a playing time of about 30 seconds as determined by the fixed number of words assigned to each document. (This playing time was arrived at as being acceptable to convey enough information with adequate context during playback of 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

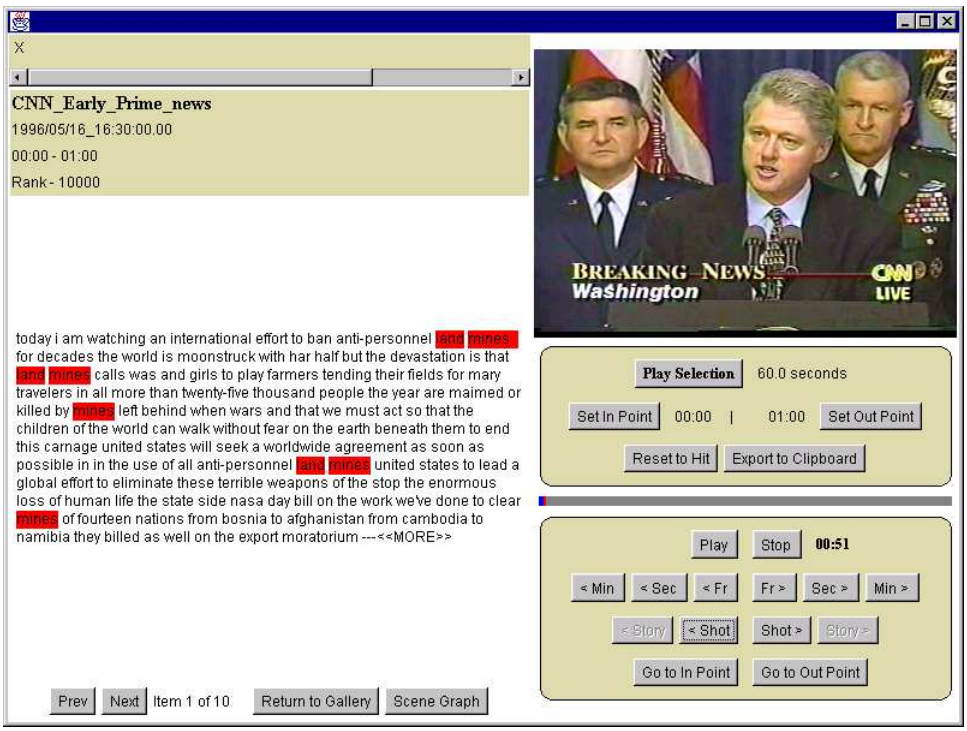


Figure 5: The VCR-like interface which permits playback of selected entity. This panel provides all the known information about the selected document (segment) - media name, location within media file, normalized score relative to other relevant documents retrieved, portion of transcript corresponding to document (segment) start and end times, and access to the audio/video source via the playback facility.

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 top N retrieved items are presented to the user in a compact form. This lets the user visually review the retrieved item for further action. It includes all the gathered information about the retrieved document including a portion of the text of the document (Figure 5). When one of the retrieved items is selected for perusal of the audio or video, media handler component is called upon to locate the media file, advance to the start time specified, decompress the stream (if required), and then initialize the media player with the first frame of the audio or video. The VCR-like interface permits the user to "play" the retrieved video from start to finish or to stop and advance at any juncture.

6.4 Combined Searching

How can retrieval results from the two disparate search schemes for text and speaker be combined when queries contain a text search string and a speaker name. The user is looking for relevant audio/video segments which contain certain words spoken by a certain speaker. First, the text query is processed and *all* (and 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 indexes for text-based search and speaker-based search are the start and end

times - of document chunks in the word-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's index (when derived from the same audio 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 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, we 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 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. 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 overall become available, λ will be closer to 1.) The resulting combined scores are re-sorted and normalized so that the best combined score is 100. The retrieval engine then returns the 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 re-visited via API calls. The media player user interface is designed to let the user continue the video past the end of the retrieved segment by selecting one of the VCR-like buttons on the user interface.

6.5 Results

We ran experiments for the entire system 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). Forty speakers were selected from the video material and enrolled into the speaker recognition system at least 30 seconds of the speaker's voice sample.

Each speaker was enrolled with 30 seconds of data from a single conversation or monologue derived from the training video clips set aside for this purpose. Two 30-minute video segments were used in testing. 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 mismatched conditions.) For text search, 10 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.

7 Conclusion and Further Work

We have described our experience with using large vocabulary speech recognition and speaker recognition for multimedia document retrieval. We believe we have some good results with re-

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 10 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.

spect to our primary goal of extracting information from audio material using imperfect automatic transcription and speaker recognition. 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 bottleneck 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. The indexer consists of the audio analysis component with the concurrent real-time speech recognition, speaker segmentation and speaker identification - verification functions. The results are captured into program memory as long as the audio is streamed into the system, and at the conclusion of which the results are re-cast into corresponding files. The audio analysis component run 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, this 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 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.

The model-based approach to speaker recognition is faster and more accurate than the frame-based approach for large speakers databases and longer test utterances. But, due to the parametric approach used in computing the distance between models given by the model-based approach, to obtain rich models sufficient amount of data should be supplied in building the models. If the

amount of data exceeds ten seconds, the model-based system gives better results. For less, the frame-based system gives more accurate and faster results. The computation speed of the model-based distance is significantly higher that the frame-by-frame technique. However, most of the computation of the frame-by-frame case may be done as the test data is being produced, where this is not possible with the model-based approach, which requires all the data to build a model. In training, since more data is generally available, the model-based approach is preferred both for its accuracy and speed of comparisons. Also, in the presence of noise, even more data is necessary for improving the performance of the model-based system. This is due to the fact that some of the Gaussian prototypes will be used to model the noise.

In speaker-based indexing we normalize the scores for each speaker. In order to express the confidence in a recognized speaker in quantifiable terms, we have 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.

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 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*, 1994.

[Bahl95] L. R. Bahl et al ., "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*, 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.

[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, pp.

[Chaudhari99] U.V. Chaudhari, H.S.M. Beigi, S.H. Maes, and J.S. Sorensen, "Multi-environment Speaker Verification," *Proceedings, AUTOID'99*, New Jersey, 1999, pp.

[Chen98] S. Chen and P.S. Gopalakrishnan, "Speaker, Environment and Channel Change Detection and Clustering Via the Bayesian Information Criterion," *Proceedings, DARPA Workshop*, 1998.

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

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

[Dharanipragada97] S. Dharanipragada, M. Franz and S. Roukos, "Audio-Indexing for Broadcast News," *Proceedings, DARPA Speech Recognition Workshop*, 1997, pp. 165–168.

[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*, 1995, pp.

[Grosky97] W. Grosky, "Pushing Streaming Video - Indexing Video Archives," *IEEE Multimedia*, Oct-Dec 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, EuroSpeech99*, 1999, pp.

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

[Pallet97] D. Pallet, "Overview of the 1997 DARPA Speech Recognition Workshop," *Proc., DARPA Speech Recognition Workshop*, Chantilly, VA, 1997, pp.

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

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

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

[Satoh99] S. Satoh, Y. Nakamura, T. Kanade, "Name-It: Naming and Detecting Faces in News Videos," *IEEE Multimedia*, 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*, 1999, pp.

[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*, 1999, pp. 576–581.

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