THE 1999 CMU 10X REAL TIME BROADCAST NEWS TRANSCRIPTION SYSTEM

Mosur Ravishankar, Rita Singh, Bhiksha Raj, and Richard M. Stern

Department of Electrical and Computer Engineering and School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

CMU's 10X real time system is the HMM-based SPHINX-III system with a newly developed fast decoder. The fast decoder uses a subvector clustered version of the acoustic models for Gaussian computation and a lexical tree search structure. It was developed in September, 1999, and is currently a first-pass decoder, capable of generating word lattices. It was designed to optimize speed, recognition accuracy as well as memory requirements. For the 1999 Hub 4 evaluation task, the system used two sets of acoustic models - full-bandwidth and narrow-bandwidth. The acoustic models were 6000 senone, 32 Gaussians per state, 3-state HMMs with no skips permitted across states. The system used a single 39 dimensional feature stream consisting of cepstra and cepstral differences. The lattices generated were rescored using a DAG algorithm. The DAG-rescored hypotheses were designated as those of the primary system. The contrastive system consisted of the output of the first pass Viterbi search, with no DAG rescoring of lattices. A trigram language model consisting of 57,000 unigrams, 10 million bigrams and 14.9 million trigrams was used. No adaptation passes were done. In this paper we describe the various components of the primary system. The first-pass word error rate on the 1998 Hub 4 evaluation set was 20.4% with this system. The overall word error rate scored by NIST for the 1999 Hub 4 evaluation set was 27.6%.

1. Introduction

The 10X spoke of the 1999 Hub 4 Broadcast News evaluation was aimed at the development of automatic speech recognition (ASR) systems that can give the best recognition performance at below 10 times real time on specified state-of-the-art computers. The task consisted of recognizing a total of 3.08 hours of speech in less than 30 hours of decoding time. The system was run on a 450-MHz Pentium-III processor machine with 256-MB RAM.

In HMM-based ASR systems higher recognition accuracies generally result from detailed models with a large number of parameters, and by a detailed search of all possible hypotheses during recognition. Since computation time increases with the number of parameters in the acoustic models, and also with the number of hypotheses considered during recognition, the requirement of high recognition accuracy is usually at cross purposes with the requirement of high recognition speed. The compromise in accuracy involved in achieving higher speeds is lower if a large amount of system memory is available to store intermediate results during recognition, which can be used to achieve higher accuracies. However, this places an additional burden on the computer since the memory used for recognition is now unavailable to other processes running concurrently.

The CMU 10X decoder was developed in September 1999. It has been designed to strike a good compromise between recognition accuracy, speed, and memory requirements, to give the best possible hypothesis in a single pass. The CMU 1999 Hub 4 system was trained keeping the requirements of the decoder in mind. The acoustic models were trained to have the lowest number of parameters that could be used without significant loss in accuracy. The lexicon was modified to use a lower number of phones. The language model (LM) was also trained with a relatively small vocabulary to reduce the search space.

This paper is divided into two sections. In Section 2 we describe the 10X system used for the evaluation including our results and observations. In Section 3 we describe the CMU 10X decoder in detail.

2. System Description of the CMU 10X Hub 4 1999 system

In this section we describe the various components of the CMU 10X broadcast news transcription system in detail. Specifically, we look at the following components of the broadcast news transcription system:

- Signal processing, in which appropriate feature vectors are extracted from the speech waveforms
- Segmentation, in which the broadcast news shows are segmented into manageable segments and non-speech regions are discarded
- Acoustic models, which are trained keeping the task and decoding requirements in mind
- Language models, which are also trained keeping the task and decoding requirements in mind

The following subsections describe each of these components.

2.1. Signal Processing

Cepstral features were computed from the speech waveforms by a standard algorithm using a log-linear frequency warping function. 13 Mel frequency cepstra were computed for each window of 25 ms, with adjacent windows overlapped by 15 ms, resulting in about 551,000 vectors for Show 1 and 555,000 vectors for Show 2. Broadband and narrowband analyses were both performed, resulting in two sets of cepstra for each show. 40 Mel filters covering the frequency range 150 Hz to 6400 Hz were used for broadband parametrization. For narrowband parametrization, 31 Mel filters covering 200 Hz to 3400 Hz were used. Signal processing took about 0.03 times real time.

2.2. Segmentation

Segmentation was done in four stages. First, the cepstral vectors for the entire show were classified using a speech/non-speech classifier. All contiguous segments of 3 seconds or longer that were marked as non-speech and that had speech/non-speech likelihood ratio below a preset threshold were discarded. The distributions for the two classes were trained using speech and non-speech segments selected from the 1997 BN training data released by LDC.

Next, the cepstral vectors were classified using a broadband-narrowband classifier. The likelihood ratio was smoothed by median filtering with a 3-second window, and all segments whose smoothed likelihood ratio lay below a threshold were marked as narrowband speech. The transition points from broadband to narrowband speech and vice-versa were marked as segmentation points. These classes were trained using broadband and narrowband segments from the 1997 LDC training data.

The cepstral vectors were then classified using a male/female classifier. All points where the difference in the log-likelihoods of the classes changed by more than a threshold were marked as segmentation points.

Even after the above three steps, several extremely long segments remained. These were further segmented using the CMUseg algorithm [9]. All classifiers used a mixture of 32 Gaussians to model the distribution of a class. The segmentation process took about 0.1 times real time.

2.3. Acoustic Training and Overall Decoding Procedure

Acoustic models were built using a subset of the 200 hours of BN training data distributed by the LDC during 1997 and 1998. In order to reduce the total training time, different amounts of training data were used at different stages in the training. For the broadband models, decision trees for state tying were built using only 25 hours of training data from the F0 and F1 conditions of the 1997 training data. This is not a recommended procedure in standard training because the structure of the decision tree changes with changing noise conditions, and parameters distributed using data from one recording domain may not be optimal for data from other recording domains [10]. This was done in our case because of other time constraints at CMU. The decision trees were pruned to give 6000 tied states in all. Models with one Gaussian per state were also trained using only 25 hours of data, and approximately 20 hours of data were added for each subsequent splitting of the Gaussians per state. The final 32-Gaussian-per-state models were trained using 125 hours of training data.

The narrowband models were trained using the same decision trees used by the broadband models. 62 hours of speech from the 1998 training data were filtered down to telephone bandwidth, and narrowband analysis (31 Mel filters covering 100 Hz to 3400 Hz) was performed to compute narrow band cepstra. The entire 62 hours were used at each stage of training. The final narrowband models had 6000 tied states, each modelled by a mixture of 32 Gaussians.

After segmentation of the test data, all segments marked as broadband are decoded using the broadband models. Segments marked as narrowband were similarly decoded using narrowband models. Lattices were generated using a trigram LM. The acoustic models used in both cases were triphone-based HMMs, with cross-word and within-word triphones separately modelled. The HMMs had a 3-state topology with transitions permitted only between adjacent states. Both models had 6000 tied states, with a mixture of 32 Gaussians modelling the distribution in each tied state. The lattice generation step took about 7.6 times real time.

The lattices were collapsed into a directed acyclic graph (DAG) and a best-path search [7] was performed using a higher language weight for the final hypothesis. This step took about 0.2 times real time.

2.4. Lexicon and Grammar Training

In order to reduce the total number of parameters, the dictionary was manually redone to use only 44 phones (instead of the usual 50 used by the CMU lexicon). Pronunciations for the more frequently-occurring words were chosen from a pre-selected set of pronunciations to maximize the likelihood of their instances in the training data. The reduced phoneset and the redone dictionary resulted in a reduced, and much more compact, set of triphones needed to model all possible combinations of phones that could be generated by the dictionary. Linguistic questions used for decision-tree generation were automatically learned from the data [1]. The final recognition lexicon consisted of 45,700 words, selected on the basis of the occurrence frequency in the LM training data. Additional emphasis was given to words that were expected to occur in the news epoch mentioned in the NIST specifications. Some words occurring in news programs in that epoch that did not occur in the training text at all were also added to the lexicon. These were modelled as unseen unigrams by the LM. However, a cursory audio analysis of the evaluation data after the evaluation revealed that there were approximately 400 OOVs in the final recognition lexicon.

A standard, trigram backoff LM was built using the CMU-Cambridge statistical language modelling toolkit [1]. The LM was trained using approximately 169 million words of text obtained from the following sources:

1. Spoken Document Retrieval text, Jan-Feb 98
2. BN LM “test” data from LDC BN CD-ROMs, 1992-1996
3. BN LM “train” data from LDC BN CD-ROMs, 1992-1996
4. BN acoustic training data transcriptions, July 1997 to Jan 1998

Frequent word sequences such as news program names, such as “A. B. C.” and names of newsreaders for these shows, and other expected common names such as “Bill Clinton” were compounded into single words. Significant reconditioning of the text was also performed to eliminate spelling mistakes, inappropriate separation of word prefixes, etc. The final trigram LM had a vocabulary of 57,000 words, and included 10 million bigrams and 14.9 million trigrams.

2.5. Word Error Rates on the Evaluation Set

The first pass decodes of the evaluation set were designated as CMU's secondary 10X system, and the hypotheses generated from DAG-rescoring of the lattices generated in the first pass were designated as CMU's primary system. Table 1 shows the word error rates (%) and Table 2 shows the timing information related to the two systems for all the Hub 4 focus conditions. Following the usual convention, these are coded as follows:

- F0: Prepared broadcast speech
- F1: Spontaneous broadcast speech
- F2: Speech over telephone channels

- F3: Speech in the presence of background music
- F4: Speech under degraded acoustic conditions
- F5: Speech from non-native speakers
- FX: All other speech and mixtures of conditions

	F0	F1	F2	F3	F4	F5	FX
Pass 1	14.6	24.7	30.4	26.2	26.0	32.0	59.1
DAG	14.4	23.7	29.9	26.0	26.4	30.5	57.4

Table 1. Word error rates for both evaluation sets for all Hub 4 focus conditions. The overall WER was 26.7% after Pass 1 and 26.3% after DAG rescoring.

Component	Time (seconds) Show 1	 Show 2	Memory Used
Signal Processing	192	192	< 5 MB
Segmentation	544	545	< 5 MB
Lattice Generation	41494	42949	190 MB
DAG Rescoring	988	1048	190 MB
Total CPU Time	43218	44734	≤ 190 MB

Table 2. Timing and memory usage information for all components of the CMU 10X system.

The computational resources used consisted of a 450-MHz Pentium II processor with 256-MB RAM and 512-MB SWAP.

3. Decoder Architecture

3.1. Outline

CMU's 10X decoder was designed to give the highest speed with minimum loss in accuracy, in conjunction with a minimum memory requirement. It is a single-pass decoder capable of generating word lattices in its first pass, which can be rescored using adapted models. Decoding is performed using a full set of acoustic models, and it is currently designed to work with a full trigram language model. However, larger LMs can be used with only minor modification of the code.

Several innovations were introduced in the standard decoding strategy in order to speed up the decoding. Both, acoustic likelihood computation and search were modified to reduce computation time. It is important to note at the outset that in this decoder, the reduction in decoding speed was achieved entirely at an algorithmic level, through implementational fine-tuning. Almost no hand-coding was done in the decoder to speed up the decoding process.

In the following sections we briefly describe the computation of acoustic likelihoods, and the search component of the decoder.

3.2. Computation of Acoustic Model Score

The acoustic models used for the CMU Hub 4 1999 system consisted of 6000 *senones* or shared states, each modelled by a Gaussian mixture with 32 component densities, and each density consisting of a 39-dimensional mean vector and corresponding diagonal covariance vector. During decoding, on average over half the senones are considered for each incoming feature vector. For a set of 6000 senones with 32 Gaussians per senone, this would require the evaluation of 3000x32 Gaussian densities for every feature vector in the utterance. Several *Gaussian selection* algorithms have been suggested in the literature to reduce this computational requirement (*e.g.* [3]). Most of them rely on using one or more layers of coarser acoustic models to obtain a *shortlist* of densities in each mixture, in each frame. The remaining densities are not evaluated.

A similar approach was used in the CMU 10X decoder as well. A *subvector quantized* version [8] of the acoustic model was created and used to select the most likely Gaussians in each mixture. The selected Gaussians were then re-evaluated using the original detailed acoustic model parameters. In the following subsections we describe the Gaussian selection procedure, and the evaluation of its performance.

3.2.1. Gaussian Selection

The general principle of building a subvector quantized acoustic model has been described in [8]. The mean and variance vectors representing each Gaussian in the model were combined and segmented into three subvectors, the first representing the means and variances of all the cepstral components in the vector, the second representing the means and variances of the delta-cepstral components, and the third representing the corresponding components for the double-delta features. The 6000x32 densities of the original model thus resulted in three sets of subvectors, each with 192,000 elements, which were then clustered into 4096 clusters using the K-means algorithm and quantized. This resulted in three codebooks, each with 4096 entries.

In any given frame, all the component densities in a Gaussian mixture were first evaluated using the quantized versions of the means and variances. Since there are only 4096x3 codewords in all, this required the computation of only 4096x3 Gaussians. The best scoring candidates were then selected, based on a *selection beamwidth*. Selection was done relative to the best-scoring component within each mixture, independently of other mixtures. The components thus selected were then re-evaluated using the original acoustic model, producing a score for the corresponding senone in that frame. The average number of Gaussians selected from each mixture was approximately 2.5 to 3, depending on the selection beamwidth.

3.2.2. Performance Evaluation

The Gaussian selection algorithm was evaluated on the basis of two metrics:

1. *Selection accuracy*, or the accuracy with which the procedure selects the best Gaussian in each mixture. This is important because the acoustic likelihood for any mixture is dominated by the most likely Gaussian in that mixture.
2. The *recognition accuracy* obtained when the procedure is used for acoustic likelihood computation.

The DARPA Hub 4 1997 evaluation data were used for this evaluation. Segments tagged as F2 (telephone channel speech) were excluded from the set. The acoustic models used for this evaluation had 6000 senones, with 20 component densities in the mixture representing each senone. Table 3 shows the selection accuracy as a function of the selection beamwidth. Within each mixture, the selection beamwidth determines the Gaussian density with the worst score (in terms of the quantized parameters) that can be considered to be active, relative to the best scoring density in that mixture. Thus, when the selection beamwidth was 1.0, only the single best component in each mixture was chosen. In that case, the shortlist size in this case would be only 5% (*i.e.* only one out of the 20 densities in each mixture would be selected).

Shortlist Beam	Shortlist Size (%)	Selection Accuracy
1.0	5.0	54
0.1	8.2	69
0.01	12.8	80
0.001	18.4	88
0.0001	24.9	92
0.00001	31.8	95
0.00001	38.3	97

Table 3. Gaussian selection accuracy as a function of selection beamwidth.

We observe that when the selection beamwidth is 0.1, we still manage to select the correct (most likely Gaussian) in each state 69% of the time. While we do miss the correct Gaussian in 31% of the senones, the effect on the recognition may not be as pronounced. This is because, if the likelihoods of these senones are low compared to the most likely ones, the senones may not occur in the best path globally.

Table 4 shows the word error rates for the same Hub 4 1997 evaluation set, with and without the proposed Gaussian selection algorithm. The selection beamwidth for this table was set to be 0.1. The shortlist size was therefore less than 10% of the mixture size, on average. The overall error rate without Gaussian selection was 26.8% and the system ran at 53 times real time. With Gaussian selection the error rate increased to 27.5%, but the execution speed dropped to 11 times real time. In other words, the Gaussian selection algorithm speeds up the acoustic likelihood computation by approximately a factor of 5, while resulting in only about a 2.5% relative increase in word error rate.

3.3. Search Algorithm

The CMU 10X decoder uses a variant of the single lexical-tree (lextree) search algorithm for searching the space of hypotheses. A variety of lextree schemes have been proposed and implemented in the literature. They differ mainly in the way in which LM scores are incorporated into the search algorithm. Ortmanns *et al.* [5] use a lextree copy strategy to incorporate LM scores into the tree. Davenport *et al.* use an incremental approach [2]. The CMU SPHINX-II system [6] uses a single lextree copy, with post-tree LM incorporation.

	F0	F1	F3	F4	F5	FX
Base	17.2	26.2	36.1	31.4	35.1	63.3
Test	17.7	26.3	37.6	32.8	35.1	65.2

Table 4. Performance evaluation of the Gaussian selection algorithm on the DARPA Hub 4 1997 evaluation data set.

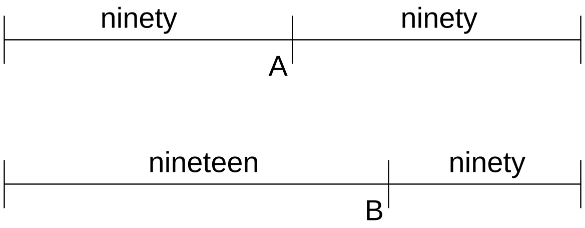

Figure 1. Example of Viterbi pruning.

Ortmanns' approach requires careful handling to prevent the copies from overwhelming the main memory available. Even so, its main memory size requirements are very large and there is considerable overhead for dynamic allocation of lextree nodes. The strategy used by Davenport *et. al.* requires the building of a much larger LM in order to handle the incremental score computation. The SPHINX-II approach, while compact and efficient, suffers from poor word segmentations and hence degraded recognition accuracy. The strategy used by Davenport *et. al.* is likely to have this drawback as well.

The problem with using the single lextree-based search in the CMU SPHINX-II system is the relatively small number of lextree root nodes. Viterbi pruning at these root nodes leaves few word segmentations (*i.e.* start times) intact, relative to the lextree copy scheme. When there is no LM score component before the Viterbi pruning decision, the result can often be unpredictable.

Figure 1 shows an example of two possible recognitions (and segmentations) where the spoken input was *nineteen ninety.* During recognition, at time *A*, the decoder hypothesizes the end of the word *ninety,* and transitions to the word *ninety.* Then, at time *B***,** the decoder hypothesizes *nineteen,* and again transitions to *ninety.* If LM scores were somehow incorporated during such transitions, assuming that $P(ninety|nineteen)$ is much greater than $P(ninety|ninety)$ the latter would most likely supersede the initial hypothesis and obtain the correct recognition. However, without the benefit of the LM, the outcome is more uncertain. (Note that the LM score for a word is only incorporated when the *end* of the word is reached.) Our solution to this problem is *unigram lextree replication*, which is described below.

3.3.1. Unigram Lexical Tree Replication

The main problem with the situation depicted in Figure 1 is that the transition from *nineteen* to *ninety* at time *B* may not succeed, especially given the absence of a guiding LM. It is desirable to have both the transitions at *A* and *B* survive, so that when the second word *ninety* is recognized, both predecessors can be considered for their LM probability. This was accomplished in the decoder by replicating the single lextree statically, and staggering the cross-word transitions to the lextrees across time, also statically. For example, in our actual evaluation configuration, three lextree copies were used, switching cross-word transitions to the next copy in a round-robin fashion every three frames.

This scheme allows many different segmentations for a hypothesized word to survive, at least one from each lextree. This partially overcomes the flaw in the SPHINX-II single lextree strategy. At the same time, the static nature of the replication places a reasonable bound on the memory requirements of the algorithm, without the overhead of dynamic lextree memory allocation.

3.3.2. Language Modelling

As mentioned earlier, the LM score for a word is incorporated only when the end of the word is reached (post-lextree LM integration).

Consider a transition occurring at time t_1 to a lextree root node. There could have been several word endings at t_1, one of which would propagate to the lextree root nodes, in accordance with Viterbi pruning. No LM score is associated with this transition, since lextree root nodes do not represent individual words. This transition would eventually lead to the end of some word *W* (*i.e.* a lextree leaf node) at some time t_2. At this time all possible LM histories for *W* are extracted from the *backpointer* that records LM histories at t_1. The LM score for *W* with respect to each of these histories is computed and incorporated into the exit score. The scores of the resulting paths and the new histories with *W* as the last word, are then included in the backpointer table at time t_2. Note that the single word *W* ending at t_2 could generate several entries in the backpointer table, one for each unique LM history. Since there is no restriction on the form of the LM history, a full trigram LM Viterbi search could now be performed.

In addition, we note that the lextree implementation does incorporate static unigram LM probabilities to provide some pruning and LM guidance during Viterbi search. The LM score at each node is the maximum of the unigram scores of all words that could be reached from that node. When a leaf node is exited, the unigram score component for that leaf is removed from the total path score, before incorporating the true LM scores (taking the histories into account).

3.3.3. Cross-Word Triphone Modelling

The CMU 10X decoder uses asymmetric cross-word triphone models. The lextree root nodes are modelled using full cross-word triphones. Each root node is therefore a *set* of all possible models derived from considering all possible left context phone. At run time, a transition to the root node enters the correct model based on the incoming left-context phone. This is important since word-beginnings are articulated most clearly and full cross-word modelling is desirable for these regions of speech.

At the lextree leaf nodes, however, *composite* HMM models are used, much like that in the BBN fast-match system [2]. Since very few words survive beam pruning until the final phone, the simplicity of using single (albeit composite) models at leaf nodes is considered to be beneficial.

3.3.4. Pruning

The CMU 10X decoder uses absolute pruning [4] [5] in addition to beam pruning to reduce the search size. Three separate pruning parameters are defined:

- Maximum active HMMs in each frame
- Maximum distinct words exiting in each frame
- Maximum histories recorded in backpointer table in each frame.

These parameters significantly improve the worst-case behavior of the decoder, in terms of recognition speed.

3.3.5. Performance Evaluation

The performance of the decoder was tested on the DARPA Hub 4 1998 evaluation set. The acoustic models used for this test had 5000 tied states with 32 Gaussians per state. A trigram LM with 4.7M bigrams and 15.5M trigrams covering a vocabulary of 64,000 words was used. The baseline performance was established using the CMU's SPHINX-III decoder which performs full acoustic model evaluation and a full search. The tests were run on a 450-MHz Pentium-III processor Linux machine with 256-MB main memory.

Table 5 shows the speed and accuracy performance of the system on the chosen test set.

Configuration	WER	WER % incr	Search Time	Total Time
Baseline	21.5	0.0	42.7	49.3
1 Lextree	23.2	7.9	1.4	6.6
2 Lextree	22.4	4.2	2.0	7.3
3 Lextree	22.1	2.8	2.6	7.9

Table 5. Evaluation of the CMU 10X decoder on the DARPA Hub 4 1998 evaluation set.

The lextree based search is observed to result in large improvements in decoder speed. The decoding time is now dominated by the acoustic likelihood evaluation, and the search component takes less than a third of the total computation time. Also, lextree replication helps improve recognition accuracy. With 3 copies, the word-error-rate (WER) performance is only marginally worse than that of the baseline system. Finally, the decoder runs on a 256-MB machine, which is fairly small by current standards. In fact, the actual memory usage of the decoder is about 190 MB on this task.

4. Discussion

We have described the architecture of the new CMU SPHINX-III fast decoder and the various components of the recognition system designed for the 1999 DARPA Hub 4 evaluation task. In the decoder, the speed-up in decoding time has been achieved almost entirely at an algorithmic level. No processor-specific hand-crafting or code-tuning was done for this purpose. The CMU fast decoder optimizes for memory *as well as* for speed and recognition. It uses less than 190 MB of RAM during decoding. The search speed was observed to increase with better trained acoustic models with a greater number of parameters (limited, of course, by the standard bounds imposed by the amount of training data used) The senone computation time was observed to increase approximately linearly with an increasing number of model parameters. We do not give quantitative results for these observations in this paper, but they were used to optimize the size of the acoustic models.

Acknowledgments

The authors thank Kristie Seymore for organizing and pre-processing the LM text. This research was sponsored by the Department of the Navy, Naval Research Laboratory under Grant No. N00014-93-1-2005. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

References

[1] P. Clarkson, and R. Rosenfeld, “Statistical language modeling using the CMU-Cambridge toolkit,” *Proc. Eurospeech* 1997.

[2] J. Davenport, L. Nguyen, S. Matsoukas, R. Schwartz, and J. Makhoul, “Toward Realtime Transcription of Broadcast News,” *Proc. Eurospeech,* 1999.

[3] M. J. F. Gales, K. M. Knill, and S. Young, “State-based Gaussian Selection in Large Vocabulary Continuous Speech Recognition Using HMMs,” *IEEE Trans. on Speech and Audio Proc.,* March 1999, **7:**152-161.

[4] J. Odell, *The Use of Context in Large Vocabulary Speech Recognition*, Ph.D. Thesis, University of Cambridge, March 1995.

[5] S. Ortmanns, A. Eiden, and H. Ney, “Improved Lexical Tree Search for Large Vocabulary Speech Recognition,” *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Proc.*, 1998.

[6] M. Ravishankar, *Efficient Algorithms for Speech Recognition*, Ph.D. Thesis, Carnegie Mellon University, May 1996, Tech Report CMU-CS-96-143.

[7] M. Ravishankar, “Some Results on Search Complexity vs Accuracy.” *Proc. DARPA Speech Recognition Workshop*, Feb. 1997.

[8] M. Ravishankar, R. Bisiani, and E. Thayer, “Sub-vector Clustering to Improve Memory and Speed Performance of Acoustic Likelihood Computation,” *Proc. Eurospeech*, 1997.

[9] M. Siegler, U. Jain, B. Raj, and R. M. Stern, “Automatic Segmentation, Classification and Clustering of Broadcast News Audio,” *Proc. DARPA Speech Recognition Workshop*, Feb. 1997.

[10] R. Singh, B. Raj, and R. M. Stern, “Domain adduced state tying for cross domain acoustic modelling, *Proc. Eurospeech,* 1999.

[11] R. Singh, B. Raj, and R. M. Stern, “Automatic clustering and generation of contextual questions for tied states in hidden markov models, *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Proc.*, 1999.