New Developments in the Lincoln Stack-Decoder Based Large-Vocabulary CSR System[1]

Douglas B. Paul [2]

Lincoln Laboratory, MIT
Lexington, Ma. 02173-9108

Abstract

The system described here is a large-vocabulary continuous-speech recognition (CSR) system developed using the ARPA Wall Street Journal[15] (WSJ) and North American Business (NAB) databases. The recognizer uses a stack decoder-based search strategy[1, 7, 14] with a left-to-right stochastic language model. This decoder has been shown to function effectively on 56K-word recognition of continuous speech. It operates left-to-right and can produce final textual output while continuing to accept additional input. The recognizer also features recognition-time adaptation to the user's voice. The new system showed a 48% reduction in the word error rate over the previously reported Nov. 92 system[16]

1. The Basic HMM CSR System

The "initial system" reported at ICASSP 93[16] was a two observation-stream (mel and Δ-mel cepstrum) Gaussian tied-mixture[3, 6] (TM) HMM CSR using cross-word sex-dependent semiphone models. Larger machines have since allowed the use of triphones and three observation streams. More recently, there has been evidence that combined observation streams using a larger number of mixture groups with fewer Gaussians in each group produces better results[4, 23] and therefore the system has been changed to use monophone-tied mixtures[9]. Monophone tied mixtures (MTM) were chosen because the decoder needs lower-context models in the fast match and if too many mixture groups were used, the efficiency of the fast match would be compromised. The MTM systems are also much smaller than the corresponding TM systems, primarily due to an order of magnitude reduction in the number of mixture weights. During this time period, the amount of available speaker-independent (SI) training data has also increased from 16 hours (SI-84) to 82 hours (SI-284).

2. The Trainer

The trainer has been improved in several ways: initialization, parameter estimation, and quantization error reduction.

2.1 Model initialization

The pdfs for the triphones are initialized by a monophone bootstrapping procedure. For instance, for a TM system, the sequence is:

1. Initialize Gaussians (data subset)
2. Train monophone models from a flat start using the Baum-Welch algorithm (data subset)
3. Train monophone models (all data)
4. Initialize triphones from monophones
5. Train triphones using the Baum-Welch algorithm

It was found that a binary-splitting K-means (i.e. Top-1 EM) was preferable to a binary-splitting EM to initialize the Gaussians because the EM procedure produced degenerate sets of Gaussians whereas the K-means procedure produced non-degenerate (i.e. no two Gaussians were identical) sets.

The MTM systems required a modification of the bootstrapping procedure:

1. Train single Gaussian per state monophone models from a flat start using the Baum-Welch algorithm (data subset)
2. alternate binary splitting and training until the desired number of Gaussians is reached (data subset)
3. Train monophones (all data)
4. Initialize triphones from monophones
5. Train triphones using the Baum-Welch algorithm

It was found that a much larger subset of the data was required in the early stages to produce the best recognition results. Several bootstrapping procedures based upon training single-Gaussian per state triphones and then clustering the Gaussians to form mixtures were also tested but failed to outperform the above.

2.2 Gaussian Variance Bound Addition

A well-known problem in ML estimation of Gaussian-dependent variances in Gaussian mixtures is variance values that go to zero. Two common methods for preventing this singularity are lower bounding or using a grand variance. Simple addition of a constant to each variance has been found to be a superior alternative to lower bounding: it is equally trivial to apply and has yielded superior recognition performance on several recognition tasks. For instance, for several tasks using single observation stream Gaussian-dependent variances:

	Error Rate (std dev)	
System	Var lim	Var add
SI-84 CSR	16.7% (.5%)	15.6% (.4%)
Spkr ID[18]	29.0% (1.4%)	26.0% (1.4%)

In both tasks, the performance was improved by over two standard deviations by the use of variance addition. While not needed to insure non-singularity, variance addition was also found to improve recognition in a grand variance system:

	Error Rate (std dev)		
System	none	Var lim	Var add
SI-84 CSR	25.2% (.5%)	20.5% (.5%)	17.5% (.5%)

In spite of the robustness of the estimate of the grand variance, the performance is improved significantly by variance limiting and even more by the variance addition.

Clearly the variance addition is doing something more than just preventing singular variances. One possible viewpoint is that variance addition is a soft limiting function. A simple bound throws away all information about the original variance while the addition retains some of the original information. Another possible view is that the variance addition is providing signal-to-noise (S/N) ratio compensation. Each component of the observation vector contains both useful signal and noise. Variance addition might

[1]This work was sponsored by the Advanced Research Projects Agency. The views expressed are those of the author and do not reflect the official policy or position of the U.S. Government.

[2]Now with Dragon Systems Inc., 320 Nevada St., Newton, MA 02160.

0-7803-2431-5/95 $4.00

act like a Wiener filter in adjusting the gain on each component appropriately:

$$-\frac{1}{2}\sum_i \frac{V_i}{V_i + lim_i}\frac{(x_i-\mu_i)^2}{V_i} = -\frac{1}{2}\sum_i \frac{(x_i-\mu_i)^2}{V_i + lim_i}$$

where the second term on the left is the normal summation term in the exponent of a diagonal covariance Gaussian and the first term on the left is analogous to a Wiener filter if lim_i represents the noise power.

2.3 Trainer Quantization Error Reduction

The SI-284 training condition of WSJ1 uses 30M frames of training data and, in the Baum-Welch training procedure, significant fractions of these frames are summed into single numbers resulting in quantization error. Due to space limitations, the accumulators are stored as two-byte log-probs. Multi-layer sums were used to reduce the quantization error without unduly increasing the datas-pace requirements.

Since there were relatively few (diagonal covariance) Gaussians in these systems quantization in estimating them was reduced adequately by the use of double-precision accumulators.

3. The Recognizer

The stack decoder is organized as described in reference [14]. The basic paradigm used by the stack decoder is:

1. Pop the best theory (partial sentence) from the stack.
2. Apply acoustic and LM fast matches[2, 5] to produce a short list of candidate next words.
3. Apply acoustic and LM detailed matches to the candidate words.
4. Insert surviving new theories into the stack.

Each theory has an ending time likelihood distribution and the most likely time on this distribution [12, 14] will hereafter be be referred to as the theory ending time. The stack is a sorted list of theories where the theories are ordered primarily shortest first based upon their ending times, and secondarily by liklihood at these ending times (best first). Any theory whose ending time likelihood is less than that of the best for the same ending time by more than some threshold is pruned from the stack. Thus the shortest theories are expanded first which has the net effect of working on a one to two second active region of the input and moving this active region left-to-right through the data.

The "extend each partial theory with one word at a time" approach allows the use of a particularly simple interface to the LM. All requests to the LM are of the form: "Give me the probability of this one-word extension to this theory." This has been exploited in order to place the LM in an external module connected via pipes[10][3]. Since the N-gram LMs currently in use are trivial to compute, the LM fast match probability is currently just the LM detailed match probability.

This stack decoder, since all information is entered into the search as soon as possible, need only pursue a "find the single best path and output it" strategy. It is also possible to output an N-best list of sentences with trivial modifications[12, 14].

Given this search strategy, it is very easy to produce output "on the fly" as the decoder continues to operate on the incoming data. Any time the first N words in all entries on the stack are the same they may be output. (This is the analog of the "confluent node" or "partial traceback" algorithm [21] in a time synchronous decoder.) No future data will alter this partial output.

Similarly, since the active zone moves left-to-right though the data, the stack decoder can easily be adapted to unbounded length input since the various bounds and the like need only cover the active region.

3.1. The Fast Match

The acoustic fast match (FM) uses a two pass strategy. Both passes search a left-diphone tree generated from the recognition vocabulary. The first pass takes all theories for which end at the current time and combines their log-likelihood distributions to create the input log-likelihood for the decoder. This decode produces two outputs: pruning thresholds for the second passes and marks on all diphone nodes for words whose FM output log-likelihood exceeds the FM output threshold envelope.

The second pass is applied once for every theory which was included in the above combination. It applies the exact log-likelihood from the detailed match as input to the left-diphone tree using pruning thresholds from the first pass and searching only the marked diphone nodes. The word output log-likelihoods are added to the LM log-probabilities to produce the net word output log-likelihoods. The cumulative maximum of these net log-likelihoods plus a (negative) threshold now produces the FM output threshold envelope. Any word whose output log-likelihood exceeds this threshold envelope is placed on the candidate word list for the detailed match.

Both passes of the fast match use a beam pruned depth-first (DF) search of the diphone tree. The DF search is faster than a time-synchronous search due to its localized data. At any one time, it only needs an input array (which was used very recently), and output array (which will be used very soon), and the parameters of one phone. This allows the DF search to stay in cache (~1 MB on many current workstations) and to page very efficiently.

A goal of recognition system design is to minimize the overall run time without loss of accuracy. In the current system, this minimum occurs (so far) with the relatively expensive fast match described above. It is the largest time consumer in the recognizer. Using pruning thresholds that reduce the number of fast match pruning errors to below a few tenths of a percent, this fast match allows only an average of about 20 words of a 20K word vocabulary to be passed to the detailed match.

3.2 The Detailed Match

The detailed match (DM) is also currently implemented as a beam-pruned depth-first searched triphone tree. The tree is precompiled for the whole vocabulary, but only triphone nodes corresponding to the FM candidate words are searched. The LM log-probabilities are integrated into the triphone tree to apply the information as soon as possible into the search. Because the right context is not available for cross-word triphones, the final phone is dropped from each word and prepended to the next word.

4. Component Algorithms

This recognition system includes a variety of algorithms which are used as components supporting the major parts described above.

4.1 Bayesian Smoothing

In a number of situations it is necessary to smooth a sparse-data estimate of a parameter with a more robust but less appropriate estimate of the parameter. For instance, the mixture weights for sparse-data triphone pdfs might be smoothed with corresponding mixture weights from the corresponding diphones and monophones[19]. The following smoothing weight estimation algorithm applies to parameters which are estimated as a [weighted] average of groups of training data.

A standard Bayesian method for combining new and old estimates of the same parameter is

$$x = \frac{N_n}{N_n + N_o}x_n + \frac{N_o}{N_n + N_o}x_o$$

where x is the parameter in question and N is the number of counts that went into each estimate and the subscripts n and o denote new and old. Similarly if one assumes the variance v of each estimate is inversely proportional to N (i.e. $v \propto \frac{1}{N}$),

$$x = \frac{v_o}{v_n + v_o}x_n + \frac{v_n}{v_n + v_o}x_o.$$

The above assumes x_n and x_o to be estimates of the same parameter. However, in the case of smoothing, the purpose is to use data from a different but related old parameter to improve the estimate of the new parameter. Thus

$$E[x] = E[x_n] \neq E[x_o].$$

If one assumes that the expected values of x and x_o differ by a zero mean Gaussian representing the unknown bias,

$$E[x] - E[x_o] = G(0, v_d)$$

[3] This interface has also been used to integrate an MIT TINA NLP[20] into a single-pass search CSR[22].

then a corrected estimate for the old variance is

$$v'_o = v_o + v_d.$$

If we now substitute the new value for v_o and return to the initial form of the estimator,

$$x = \frac{N_n}{N_n + N'_o} x_n + \frac{N'_o}{N_n + N'_o} x_o \quad \text{where} \quad N'_o = \frac{N_o N_d}{N_o + N_d}.$$

Note that $N'_o \leq N_d$ and thus the smoothing equation discounts the value of the old data according to N_d which, for the above examples, may be determined empirically. This equation can be trivially extended to include multiple old estimates for smoothing a triphone with the left diphone, right diphone, and monophone. In this recognition system, symmetries and linear interpolation across states have been used to reduce the number of N_d's for triphone smoothing from twelve to three. This smoothing scheme has also been used for speaker adaptation and language modeling.

4.2 Pdf cache

Tied mixture pdfs are relatively expensive to compute and a pdf cache is necessary to reduce the computational load. The cache must also be able to grow efficiently upon demand and discard outdated entries efficiently. Algorithms such as hash tables do not grow efficiently and have terrible memory cache and paging behavior. Instead, the pdf cache is stored as a dynamically allocated three dimensional array:

$$pdf[t/T][s][t\%T]$$

where t is time, s is the state, and % is the modulo operator. Only the first level pointer array ($[t/T]$) is static, both the $[s]$ pointer arrays and the actual storage locations $[t\%T]$ are allocated dynamically. Outdating is simple: remove all pointer arrays and storage locations for $t/T < t'/T$ (integer arithmetic), allocation occurs whenever a null pointer is traversed, and access is just two pointers to a one dimensional array. It is also a very good match to a depth-first search since such a search accesses the states of a phone sequentially in time for a number of time steps which gives very good memory cache and paging performance. This caching algorithm is used in both the trainer and the recognizer.

5. WSJ Recognition Results

The 5K and 20K word vocabulary WSJ recognition results are:

System	Train	Pho	Pdf	Str x Gauss	5K WErr	20K WErr
Nov 92	SI-84	Semi	TM	3x257	13.5%	21.8%
	SI-84	Tri	TM	3x257	9.9%	16.9%
Nov 93	SI-284	Tri	TM	3x257	7.9%	14.2%
Nov 94	SI-284	Tri	MTM	1x64	7.0%	13.0%

5K Word err: p=62 NVP trigram LM, std dev=.3–.4%,
20K Word err: p=160 NVP trigram LM, std dev=.4–.5%,
WSJ0/WSJ1 training, WSJ0 test, closed vocab,
X-word sex-dependent semi/triphones, cep+Δcep+ΔΔcep

Both show about two-thirds of the improvement between the Nov. 92 and Nov. 93 systems to be due to the algorithmic improvements and about one-third to be due to the increased training data. Small additional performance improvements (along with a substantial size reduction) were obtained by switching to MTM pdfs for the Nov. 94 system. The total improvement is 48% for the 5K word system and 40% for the 20K word system.

6. The NAB Task

We have recently begun working on the ARPA North American Business (NAB) task, which is similar in form to the WSJ task, except that the LM training and the test prompts are derived from several newspapers. It has been structured as an open vocabulary task. Results are shown for the CMU-supplied 20K baseline compact trigram backoff LM and a 56K compact trigram backoff LM.

LM	Vocab	Wd err	perplex	OOV wds
baseline	20K	25.7%	135.4	2.6%
56K trigram	56K	23.6%	147.2	.9%

MTM systems, NAB dev test1, WSJ1 training, 2 obs streams,
non-X-word, non-sex-dependent triphones, std dev ∼.5%

The 56K vocabulary was chosen as the intersection of the 64K most frequent words and a CMU-supplied 100K word dictionary used for the task.

7. Additional Techniques

Several other techniques have been explored, but were not used in the above systems.

7.1 Language modeling

A potentially new form of interpolated N-gram LM has been developed. The basic interpolated trigram LM is of the form:

$$p(w_i) = \lambda_0 p_0 + \lambda_1 p_1(w_i) + \lambda_2 p_2(w_i|w_{i-1}) + \lambda_3 p_3(w_i|w_{i-1}, w_{i-2})$$

where $p(w|x) = count(w,x)/count(x)$, w is the predicted word and x is the context word[s]. However, p_2 and p_3 are undefined if their contexts were not observed. Thus, since all p_i must sum to one for each context, something must be done to avoid using the undefined terms. One method is to substitute lower order probabilities for the undefined terms. The potentially new strategy is to back off to a lower order (i.e. a backed-off interpolated LM) using several sets of λs. (This is distinct from a back-off LM.) This strategy uses several sets of $\lambda_{j,k}$s:

if the bigram $w_{i-1}w_{i-2}$ exists:	use $p_0 - p_3$ and $\lambda_{3,k}$
else if the unigram w_{i-1} exists:	use $p_0 - p_2$ and $\lambda_{2,k}$
else	use $p_0 - p_1$ and $\lambda_{1,k}$.

This avoids the need for heuristic strategies to substitute for the undefined terms and produces slightly lower perplexities. All of the results listed below for interpolated LMs use this scheme.

The original choice of the compact back-off form of an N-gram LM for the ARPA LVCSR baseline LMs was made by the author with little scientific justification[15]. Many papers on language modeling have been published using perplexity to compare LMs, but there is significant evidence that perplexity is a poor predictor of recognition performance[13]. Thus we undertook a series of experiments comparing the actual recognition performance of a recognizer using several different N-gram LMs.

LM type	full LM		compact LM	
	Perplex	wd err	Perplex	wd err
GT disc. Back-off	60.1	11.18%	61.7	11.77%
FD disc. Back-off	60.2	11.35%	61.4	11.83%
Interpolated	64.7	11.84%	67.3	12.48%
Bucketed Interp	61.4	11.80%	67.1	12.95%

GT=Good-Turing discount, FD=fixed discount
TM-2 systems, closed 5K NVP vocab, trigram LMs,
WSJ0 training and test data, 2 obs streams
non-X-word, non-sex-dependent triphones, std dev ∼.4%,
compact LMs dropped the count=1 trigrams

The GT discount back-off LMs used the Good-Turing discount (as in[8]), the fixed discount back-off LMs used an n/(n+1) discount, the interpolated LMs are described above, and the bucketed interpolated LMs used multiple sets of of λs indexed by the context count. This experiment was performed using the stack decoder-LM interface[10] to allow the LM modules to be interchanged without modifying the recognizer.

These results show only a small difference between the two forms of discounting for the back-off LMs. The simple interpolated LM is no worse than the bucketed interpolated LMs. The back-off LMs perform about 1.5 standard deviations better than the interpolated LMs and the full LMs are about 1.5 standard deviations better than the compact LMs. All-in-all, the initial choice of compact back-off LMs for the baseline LMs was a reasonable trade-off of performance and size.

7.2 Fast Adaptation

Human listeners are capable of adapting very rapidly to a new speaker, but most ASR systems can only adapt slowly to a new speaker (e.g. [16]). A technique for very rapid adaptation of a TM system to a new speaker has been devised. Some earlier experiments in speaker adaptation used a training procedure which produced speaker-independent mixture weights and speaker-dependent Gaussians at one point in the training

process[16]. Using a model of this form, the recognition system can first perform speaker-identification to find the closest set of Gaussians to the current speaker and and then to recognize the speech using that set of Gaussians. Such a process is feasible in a TM system because the sets of Gaussians are fairly small and it is practical to have multiple sets of Gaussians (one per training speaker). The results are:

Training	Sets of Gaussians	Wd err rate
normal SI-84	1	11.8%
SI-12	12	12.7%
SI-84	84	11.3%
normal SI-284	1	10.6%
SI-284	284	10.2%

TM systems, closed 5K, NVP, p=62 trigram LM,
WSJ0/1 training and WSJ0 test data, 2 obs streams
non-X-word, non-sex-dependent triphones, std dev ~.4%

The multiple Gaussian set SI-84 system (~85 sentences per speaker) was a slight improvement over the normal SI-84 system and the multiple Gaussian set SI-12 (600 sentences per speaker) was slightly worse than the normal system. Clearly there are trade-offs between the number of speakers and the number of sentences per speaker which cannot be explored adequately with the given training data. The earlier work reported in Ref. [16] suggests that such a set of Gaussians will adapt better by traditional parameter adaptation methods than will a traditional SI set of Gaussians. However, the above results show the technique to be worthy of further exploration.

8. Discussion And Conclusions

The Nov. 92 system had been limited to two-observation-stream semiphones to limit the size of the system to one that would operate efficiently on the then available machines. The Nov. 93 system gave significantly better performance, but was too large to be practical. Finally, the MTM pdfs in the Nov. 94 system were much smaller and still yielded a small performance improvement.

Quantization error in the trainer is very subtle. The Baum-Welch training algorithm is sufficiently stable that it will only be found if one specifically looks for it. The primary effect in the systems described above is a "flattening" of the pdfs through a net upper-bounding of the mixture weight accumulation sums.

The stack-decoder has been shown to be an effective decoder for large vocabulary CSR both here and elsewhere[1]. Because it efficiently combines all information into a single unified search and it makes a zonal left-to-right pass though the input data, it can produce the recognized output continuously as more data is input as well as handle unbounded length input. Most of the computation in the above CSR is consumed by the acoustic fast match. (The stack itself is a very efficient mechanism for limiting the number of theories which must be expanded.) Thus the largest future speed-ups will probably result from faster fast matches. Significant speed-ups have already resulted from a mixture of strategies, such as pdf caching and covered theory elimination, and implementations which use the machine architectures efficiently without compromising portability.

The Bayesian smoothing fills in a long-standing gap in the smoothed triphone scheme[19] The smoothing weights must be computed by deleted interpolation[1] which requires at least several instances of the triphone or estimated by some non-data-driven method. The non-data-driven methods have generally been ad-hoc[11, 19]. This gives theoretical support for a functional form based upon the amounts of data available to train each object and the objects' similarity. This smoothing approach has also been used for acoustic adaptation and in language modeling.

Finally, variance addition is useful as a simple technique to reduce the error rate in many Gaussian mixture (or multiple Gaussian) based systems. Many standard techniques for dealing with varying S/N in the observation components perform a linear transform on the observation vector and then drop some of the resulting components. This all-or-nothing dropping of components throws away some signal with the noise. In contrast, variance addition attempts to weight each term according to its value. This technique appears to be related to the technique of "diagonal loading" (adding a constant to the diagonal of a matrix) that is sometimes used to increase the stability and/or noise immunity of a covariance matrix prior to inversion[17].

The above-described CSR system is well suited to handle the large vocabulary CSR problem. Many problems still need work—speed, size, accuracy, and robustness, to name a few—but improvements in all should be possible.

References

[1] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-5, March 1983.

[2] L. Bahl, S. V. De Gennaro, P. S. Gopalakrishnam, R. L. Mercer, "A Fast Approximate Acoustic Match for Large Vocabulary Speech Recognition," IEEE Trans. Speech and Audio Processing, Jan. 1993.

[3] J.R. Bellegarda and D.H. Nahamoo, "Tied Mixture Continuous Parameter Models for Large Vocabulary Isolated Speech Recognition," Proc. ICASSP 89, Glasgow, May 1989.

[4] J. L. Gauvain, L. F. Lamel, "The LIMSI Continuous Speech Dictation System: Evaluation on the ARPA Wall Street Journal Task," Proc. ICASSP 94, Adelaide, Australia, April 1994.

[5] L. S. Gillick and R. Roth, "A Rapid Match Algorithm for Continuous Speech Recognition," Proceedings June 1990 Speech and Natural Language Workshop, Morgan Kaufmann Publishers, June, 1990.

[6] X. D. Huang and M.A. Jack, "Semi-continuous Hidden Markov Models for Speech Recognition," Computer Speech and Language, Vol. 3, 1989.

[7] F. Jelinek, "A Fast Sequential Decoding Algorithm Using a Stack," IBM J. Res. Develop., vol. 13, November 1969.

[8] S. M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," ASSP-35, pp 400-401, March 1987.

[9] D. B. Paul, "The Lincoln Robust Continuous Speech Recognizer," Proc. ICASSP 89., Glasgow, Scotland, May 1989.

[10] D. B. Paul, "A CSR-NL Interface Architecture," Proc. ICSLP 92, Banff, Alberta, Canada, Sept. 1992.

[11] D. B. Paul, "The Lincoln Tied-Mixture HMM Continuous Speech Recognizer," ICASSP 91, Toronto, May 1991.

[12] D. B. Paul, "Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder," ICASSP 91, Toronto, May 1991.

[13] D. B. Paul, J. K. Baker, and J. M. Baker, "On the Interaction Between True Source, Training, and Testing Language Models," ICASSP 91, Toronto, May 1991.

[14] D. B. Paul, "An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model," Proc. ICASSP 92, San Francisco, California, March 1992.

[15] D. B. Paul and J. M. Baker, "The Design for the Wall Street Journal-based CSR Corpus," Proc. ICSLP 92, Banff, Alberta, Canada, Sept. 1992.

[16] D. B. Paul and B. F. Necioglu, "The Lincoln Large-Vocabulary Stack-Decoder HMM CSR," ICASSP 93, Minneapolis, April 1993.

[17] C. M. Rader, personal communication.

[18] D. A. Reynolds, personal communication.

[19] R Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech," Proc. ICASSP 85, Tampa, FL, April 1985.

[20] S. Seneff, "TINA: A Natural Language System for Spoken Language Applications," *Computational Linguistics*, vol. 18, no. 1, 1992.

[21] J. C. Spohrer, P. F. Brown, P. H. Hochschild, and J. K. Baker, "Partial Backtrace in Continuous Speech Recognition," Proc. Int. Conf. on Systems, Man, and Cybernetics, 1980.

[22] D. Tummala, S. Seneff, D. B. Paul, to be submitted to the ARPA Spoken Language Technology Workshop, Austin, Texas, Jan. 1995.

[23] P. C. Woodland, J. J. Odell, V. Valtchev, S. J. Young, "Large Vocabulary Speech Recognition Using HTK," Proc. ICASSP 94, Adelaide, Australia, April 1994.