TECHNIQUES TO ACHIEVE AN ACCURATE REAL-TIME LARGE-VOCABULARY SPEECH RECOGNITION SYSTEM

Hy Murveit, Peter Monaco, Vassilios Digalakis, John Butzberger

SRI International
Speech Technology and Research Laboratory
333 Ravenswood Avenue
Menlo Park, California 94025-3493

ABSTRACT

In addressing the problem of achieving high-accuracy real-time speech recognition systems, we focus on recognizing speech from ARPA's 20,000-word Wall Street Journal (WSJ) task, using current UNIX workstations. We have found that our standard approach—using a narrow beam width in a Viterbi search for simple discrete-density hidden Markov models (HMMs)—works in real time with only very low accuracy. Our most accurate algorithms recognize speech many times slower than real time. Our (yet unattained) goal is to recognize speech in real time at or near full accuracy.

We describe the speed/accuracy trade-offs associated with several techniques used in a one-pass speech recognition framework:

- Trade-offs associated with reducing the acoustic modeling resolution of the HMMs (e.g., output-distribution type, number of parameters, cross-word modeling)
- Trade-offs associated with using lexicon trees, and techniques for implementing full and partial bigram grammars with those trees
- Computation of Gaussian probabilities are the most time-consuming aspect of our highest accuracy system, and techniques allowing us to reduce the number of Gaussian probabilities computed with little or no impact on speech recognition accuracy.

Our results show that tree-based modeling techniques used with appropriate acoustic modeling approaches achieve real-time performance on current UNIX workstations at about a 30% error rate for the WSJ task. The results also show that we can dramatically reduce the computational complexity of our more accurate but slower modeling alternatives so that they are near the speed necessary for real-time performance in a multipass search. Our near-future goal is to combine these two technologies so that real-time, high-accuracy large-vocabulary speech recognition can be achieved.

1. INTRODUCTION

Our techniques for achieving real-time, high-accuracy large-vocabulary continuous speech recognition systems focus on the task of recognizing speech from ARPA's Wall Street Journal (WSJ) speech corpus. All of the speed and performance data given in this paper are results of recognizing 169 sentences from the four male speakers that comprise ARPA's November 1992 20,000-word vocabulary evaluation set. Our best performance on these data is 8.9% (10.3% using bigram language models). Our standard implementation for this system would run approximately 100 times slower than real time.[1] Both these systems use beam-search techniques for finding the highest-scoring recognition hypothesis.

Our most accurate systems are those that use HMMs with genonic mixtures as observation distributions [3]. Genonic mixtures sample the continuum between fully continuous and tied-mixture HMMs at an arbitrary point and therefore can achieve an optimum recognition performance given the available training data and computational resources. In brief, genonic systems are similar to fully continuous Gaussian-mixture HMMs, except that instead of each state having its own set of Gaussian densities, states are clustered into *genones* that share these Gaussian codebooks. Each state, however, can have its own set of mixture weights used with the Gaussian codebook to form its own unique observation distribution. All the genonic systems discussed in this paper use a single 39-dimensional observation composed of the speech cepstrum and its first and second derivatives, and the speech energy and its first and second derivatives. All Gaussians have diagonal covariance matrices.

2. MODELING TRADE-OFFS

The speed/accuracy trade-off of our speech recognition systems can be adjusted in several ways. The standard approaches are to adjust the beam width of the Viterbi search and to change the

[1] We define real-time systems as those that process 1 second of speech per second.

output-distribution modeling technique. Table 1 shows, for

System Type	Cross-Word Modeling	Word Error (%)	Lattice Search Speed
Genone	yes	11.6	50.4
Genone	no	13.4	19.8
Phonetically Tied Mixtures	yes	13.9	43.9
Phonetically Tied Mixtures	no	16.6	6.8
VQ	no	19.2	~1

Table 1: Effect of model type on speed and accuracy

instance, that eliminating cross-word modeling can significantly improve the speed of our recognition systems at about a 20% cost in word error. In this table, lattice speed refers to recognition accuracy when decoding from precomputed word lattices [8]. That is, this is only performing a subset of the search. Actual full grammar recognition time could be from a factor of 3 to an order of magnitude higher. However, it is useful to compare relative lattice decoding speeds from the various techniques.

A technique frequently used at SRI to achieve relatively fast speech recognition demonstrations is to downgrade our acoustic modeling by implementing a discrete density (VQ) HMM system without cross-word acoustic modeling. This system is then searched using a Viterbi search with a small beam width. Table 2 shows the full grammar speed accuracy trade-off when modifying the beam width if a Silicon Graphics Incorporated[2] (SGI) UNIX workstation with a 150-MHz MIPS R4400 CPU[3] is used to

Beam Width	Word Error (%)	Hypotheses per Frame	Full Search Speed
600	29.5	981	3.2
700	21.5	3089	8.3
800	19.2	7764	16.0

Table 2: Speed/accuracy trade-off for a beam search

perform the computation.

We have found that this technique gives an unsatisfactory speed/accuracy trade-off for this task and we have investigated other techniques as described below.

[2] All product names mentioned in this paper are the trademark of their respective holders.

[3] This workstation scores 85 and 93 for the SPECint92 and SPECfp92 benchmarks. For our tests it is roughly 50% faster than an SGI R4000 Indigo, and 50% faster than a SPARC 10/51. It should be between 1/2 and 2/3 the speed of an HP735. SGI R4400 systems cost about $12,000.

3. LEXICON TREES

We explored the use of lexicon trees as a technique for speeding up the decoding times for all modeling techniques. Lexicon trees represent the phonetics of the recognition vocabulary as a tree instead of as a list of pronunciations (lists of phones). With a tree representation, words starting with the same phonetic units share the computation of phonetic models. This technique has been used by others, including the IBM [10], Phillips [7], and CMU groups, and it is also currently used at LIMSI. Because of the large amount of sharing, trees can drastically reduce the amount of computation required by a speech recognition system. However, lexicon trees have several possible drawbacks:

- Phonetic trees are not able to use triphone modeling in all positions since the right phonetic context of a node in a tree can be ambiguous.
- One cannot implement an admissible Viterbi search for a single lexicon tree when using a bigram language model, because the word being decoded (*w2* in the bigram equation $P(w2/w1)$) may not be known until a leaf in the tree is reached—long after certain Viterbi decisions are typically made.

The first concern can be addressed by replicating nodes in the tree to disambiguate triphone contexts. However, even this may not be necessary because the large majority of right contexts in the tree are unambiguous (that is, most nodes have only one child). This is shown in Table 3, where the concentrations of triphone and biphone models are compared for tree- and linear-lexicon schemes.

Lexicon Type	Triphone Models (%)	Biphone Models (%)
Tree	73	27
Linear	85	15

Table 3: Model allocation for the SRI WSJ system with and without lexicon trees

The second concern, the ability to model bigram language models using an admissible search strategy, is a problem. As shown in Table 4, moving from a bigram to a unigram language model more than doubles our error rate. Ney [7] has proposed a scheme where lexicon trees are replicated for each bigram context. It is possible that this scheme would generalize to our application as well. For the three recognition systems in Table 2, on average 7, 13, and 26 different words end each frame. This is the minimum average number of copies of the lexicon tree that the system would need to maintain.

We have decided to pursue a different approach, which is shown in the figure below. We refer to this technique as *approximate bigram trees*.

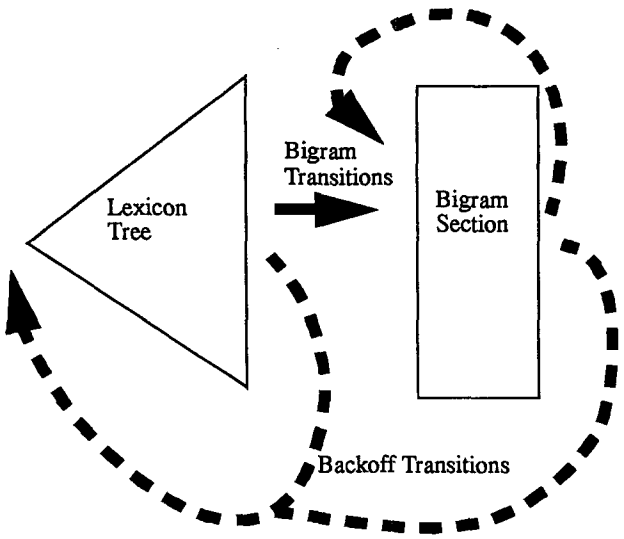

Figure 1: Approximate bigram trees

In an approximate bigram tree, the aim is to model the salient portion of the backed-off bigram language model [11] in use. In an approximate bigram tree, a standard lexicon tree (incorporating unigram word probabilities) is combined with a bigram section that maintains a linear (non-tree) representation of the vocabulary. Bigram and backoff language model transitions are added to the leaves of the tree and to the word-final nodes of the bigram section.[4] When the entire set of bigram is represented, then this network implements a full backed-off bigram language model with an efficient tree-based backoff section. In fact, for VQHMM systems, this scheme halves our typical decoding time for little or no cost in accuracy. Typically, however, we need further reduction in the computational requirement. To achieve this we represent only a subset of the group of bigram transitions (and adjust the backoff probabilities appropriately). This degrades the accuracy of our original bigram language model, but reduces its computational requirements. The choice of which bigrams to represent is the key design decision for approximate bigram trees. We have experimented with four techniques for choosing bigram subsets to see which make the best speed/accuracy trade-offs:

Count x means only use bigrams where $P(w1) * P(w2/w1) > 10^x$.

Prob x means only use bigrams where $P(w2/w1) > 10^x$.

Improve X means only use bigrams where $P(w2/w1) > Backoff(w1) * P(w2) / 10^x$.

Top *x* means only use bigrams $P(w2/w1)$ where $w2$ is one of the most frequent x words.

Table 4 shows speed/accuracy trade-offs for approximate bigram trees.

Tree Type	Number of Bigrams Used (thousands)	Word Error (%)	Full Search Speed (x RT)
Unigram tree	0	42.3	0.6
(non-tree) Bigram	3500	21.5	8.5
count, -6	93	30.4	1.5
count, -5	10	35.8	0.9
count, -4	.6	39.2	0.7
prob, -3	1250	28.2	0.9
prob, -2.5	671	29.2	0.8
prob, -2	219	31.5	0.7
prob, -1	20	36.6	0.7
improve, 2	908	29.7	1.6
improve, 3	191	37.1	0.8
top 10	113	39.5	0.7
top 50	320	36.0	0.7
top 100		35.2	0.7
top 1000	1500	31.4	1.1
top 5000	2624	25.3	1.9
top 20000	3500	21.0	~3

Table 4: Performance of "approximate bigram" trees

The top two lines of the table show that the bigram language model improves performance from 42.3% word error to 21.5% as compared with a unigram language model. The rest of the table shows how approximate bigram trees can trade off the performance and speed of the bigram model. For instance, in several techniques shown—such as *prob 2.5*—that maintain more than half of the benefit of bigrams for little computational cost, CPU usage goes from 0.6 to 0.8, when the error rate goes from 42.3% to 29.2%. The rest of the improvement, reducing the error rate from 29.2% to 21%, increases the required computation rate by a factor of four.

Table 4 also shows that the number of bigrams represented does not predict the computation rate.

[4] In the actual implementation, word-final nodes in the bigram section are merged with their counterparts in the tree so that the bigram transitions need be represented only once. For simplicity, however, we show the system with two sets of bigram probabilities.

The square root of the perplexity of these language models seems to predict the recognition performance as shown in Table 5.

Top X	Perplexity	Perplexity Square Root	Word error (%)
0	1248	35.3	42.3
10	954	30.9	39.5
50	727	27.0	36.0
100	631	25.1	35.2
1000	401	20.0	31.4
20000	237	15.4	21

Table 5: Grammar Perplexity for top X trees

4. REDUCING GAUSSIAN COMPUTATIONS

SRI's most accurate recognition systems, using genonic mixtures, require the evaluation of very large numbers of Gaussian distributions, and are therefore very slow to compute. The baseline system referenced here uses 589 genonic mixtures (genones), each with 48 Gaussian distributions, for a total of 28,272 39-dimensional Gaussians. On ARPA's November 1992 20,000-word Evaluation Test Set, this noncrossword, bigram system performs at 13.43% word error. Decoding time from word lattices is 12.2 times slower than real time on an R4400 processor. Full grammar decoding time would be much slower. Since the decoding time of a genonic recognition system such as this one is dominated by Gaussian evaluation, one major thrust of our effort to achieve real-time recognition has been to reduce the number of Gaussians requiring evaluation each frame. We have explored three methods of reducing Gaussian computation: Gaussian clustering, Gaussian shortlists, and genonic approximations.

4.1. Gaussian Clustering

The number of Gaussians per genone can be reduced using clustering. Specifically, we used an agglomerative procedure to cluster the component densities within each genone. The criteria that we considered were an entropy-based distance and a generalized likelihood-based distance [6]. We found that the entropy-based distance worked better. This criterion is the continuous-density analog of the weighted-by-counts entropy of the discrete HMM state distributions, often used for clustering HMM state distributions [5], [3].

Our experiments showed that the number of Gaussians per genone can be reduced by a factor of three by first clustering and then performing one additional iteration of the Baum-Welch algorithm as shown in Table 6. The table also shows that clustering followed by additional training iterations gives better accuracy than directly training a system with a smaller number of Gaussians (Table 6, Baseline2). This is especially true as the number of Gaussians per genone decreases.

System	Gaussians per Genone	Word Error (%)
Baseline1	48	13.43
Baseline1+Clustering	18	14.17
above+Retraining	18	13.64
Baseline2	25	14.35

Table 6: Improved training of systems with fewer Gaussians by clustering from a larger number of Gaussians

4.2. Gaussian Shortlists

We have developed a method for eliminating large numbers of Gaussians before they are computed. Our method is to build a "Gaussian shortlist" [2], [4], which uses vector quantization to subdivide the acoustic space into regions, and lists the Gaussians worth evaluating within each region. Applied to unclustered genonic recognition systems, this technique has allowed us to reduce by more than a factor of five the number of Gaussians considered each frame. Here we apply Gaussian shortlists to the clustered system described in Section 4.1. Several methods for generating improved, smaller Gaussian shortlists are discussed and applied to the same system.

Table 7 shows the word error rates for shortlists generated by a variety of methods. Through a series of methods, we have reduced the average number of Gaussian distributions evaluated for each genone from 18 to 2.48 without compromising accuracy. The various shortlists tested were generated in the following ways:

- **None**: No shortlist was used. This is the baseline case from the clustered system described above. All 18 Gaussians are evaluated whenever a genone is active.
- **12D-256**: Our original shortlist method was used. This method uses a cepstral vector quantization codebook (12-dimensions, 256 codewords) to partition the acoustic space. With unclustered systems, this method generally achieves a 5:1 reduction in Gaussian computation. In this clustered system, only a 3:1 reduction was achieved, most likely because the savings from clustering and Gaussian shortlists overlap.
- **39D-256**: The cepstral codebook that partitions the acoustic space in the previous method ignores 27 of the 39 feature dimensions. By using a 39-dimensional, 256-codeword VQ codebook, we created better-differentiated acoustic regions, and reduced the average shortlist length to 4.93.
- **39D-4096**: We further decreased the number of Gaussians per region by shrinking the size of the regions. Here we used a single-feature VQ codebook with 4096 codewords.

For such a large codebook, vector quantization is accelerated using a binary tree VQ fastmatch.

- **39D-4096-min1**: When generating a Gaussian shortlist, certain region/genone pairs with low probabilities are assigned very few or even no Gaussians densities. When we were using 48 Gaussians/genone, we found it important to ensure that each list contains a minimum of three Gaussian densities. With our current clustered systems we found that we can achieve similar recognition accuracy by ensuring only one Gaussian per list. As shown in Table 7, this technique results in lists with an average of 2.48 Gaussians per genone, without hurting accuracy.

Shortlist	Shortlist Length	Gaussians Evaluated per Frame	Word Error (%)
none	18	5459	13.64
12D-256	6.08	1964	13.53
39D-256	4.93	1449	13.46
39D-4096	3.68	1088	13.64
39D-4096-min1	2.48	732	13.50

Table 7: Word error rates and Gaussians evaluated, for a variety of Gaussian shortlists

Thus, with the methods in Sections 4.1 and 4.2, we have used clustering, retraining, and new Gaussian shortlist techniques to reduce computation from 48 to an average of 2.48 Gaussians per genone without affecting accuracy.

4.3. Genonic Approximations

We have successfully employed one other method for reducing Gaussian computation. For certain pairs of genones and acoustic regions, even the evaluation of one or two Gaussian distributions may be excessive. These are cases where the output probability is either very low or very uniform across an acoustic region. Here a uniform probability across the region (i.e., requiring no Gaussian evaluations) may be sufficient to model the output probability.

To provide these regional flat probabilities, we implemented a discrete-density HMM, but one whose output probabilities were a region-by-region approximation of the probabilities of our genonic system. Since the two systems' outputs are calibrated, we can use them interchangeably, using a variety of criteria to decide which system's output to use for any given frame, state, acoustic region, or hypothesis. This technique, using variable resolution output models for HMMs is similar to what has been suggested by Alleva et al. [1].

We train this genonic approximation by averaging, for each acoustic region, the output of each genone across a set of observations. The resulting system can be used either by itself or in combination with the continuous system from which it was trained.

Table 8 shows the performance of the discrete approximate genone systems as a function of the number of regions used.

Genonic System	Number of Acoustic Regions	Word Error (%)
Continuous	N/A	13.64
Discrete	256	31.72
Discrete	1024	23.62
Discrete	4096	20.32
Discrete	16384	18.40

Table 8: Accuracy of genonic approximation systems

Even with 16384 acoustic regions, the discrete genonic approximation has an error rate of 18.40%, compared with the baseline continuous system at 13.64%. However, when these discrete systems are used selectively in combination with a continuous genonic system, the results are more encouraging. Our most successful merger combines the 4096-region discrete approximation system (20.32% error) with the 39D-4096-min1 genone system from Table 7 (13.50% error). In combining the two, instead of ensuring that a single Gaussian density was available for all shortlists, the genonic approximation was used for cases where no densities existed. In this way, we were able to eliminate another 25% of the Gaussian computations, reducing our lattice-based computation burden to 564 Gaussians per frame, with a word error of 13.36%.

In summary, we started with a speech recognition system with 28,272 Gaussian distributions that computed 14,538 Gaussian distributions per frame and achieved a 13.43% word error rate running 12.2 times slower than real time on word lattices. Using the techniques described in Section 4, we have reduced the system's computational requirements to 564 Gaussians per frame, resulting in a system with word error of 13.36%, running at 2.0 times real time on our word lattices.

5. MULTIPASS APPROACHES

The techniques for improving the speed of single-pass speech recognition systems can be combined to achieve other speed/accuracy trade-offs (e.g., trees using genone systems with reduced Gaussian computation rates). Furthermore, with multipass approaches [8,9] many of these techniques can be used independently as the different passes of the speech recognition system. For instance, a discrete density tree search may be used in a lattice building or a forward pass, and a Gaussian system may be used in the lattice and/or backward passes.

We have performed preliminary evaluations of several of the tree-based systems presented in Section 3 to evaluate their performance as forward passes for a forward-backward search [9]. Preliminary results show that forward tree-based systems with 30% word error would add at most 3% to the word error rate of a full accuracy backward pass (i.e., at most increase the error rate from

approximately 10% to approximately 13%). More detail on this work will be presented at the HLT conference and will be included in the final version of this paper.

6. CONCLUSIONS

Tree-based techniques, combined with appropriate modeling alternatives, can achieve real-time performance at about 30% error rate for ARPA's 20,000-word Wall Street Journal task. We have shown techniques that reduce the computational complexity of more accurate but slower modeling alternatives so that they are near the speed necessary for real-time performance in a multipass search. Our near-future goal is to combine these two technologies so that real-time, high-accuracy large-vocabulary speech recognition can be achieved.

ACKNOWLEDGMENT

We gratefully acknowledge support for this work from ARPA through Office of Naval Research Contract N00014-92-C-0154. The Government has certain rights in this material. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Government funding agencies.

REFERENCES

1. F. Alleva, X. D. Huang and M.-Y. Hwang, "An Improved Search Algorithm Using Incremental Knowledge for Continuous Speech Recognition," *Proc. ICASSP*, pp. II-307 - II-310, April 1993.

2. E. Bocchieri, "Vector Quantization for the Efficient Computation of Continuous Density Likelihoods," *Proc. ICASSP*, pp. II-692 - II-695, April 1993.

3. V. Digalakis and H. Murveit, "Genones: Optimizing the Degree of Tying in a Large Vocabulary HMM-based Speech Recognizer," to appear in *Proc. ICASSP*, 1994.

4. V. Digalakis, P. Monaco and H. Murveit, "Acoustic Calibration and Search in SRI's Large Vocabulary HMM-based Speech Recognition System," *Proc. IEEE ASR Workshop*, Snowbird, Dec. 1993.

5. M.-Y. Hwang and X. D. Huang, "Subphonetic Modeling with Markov States - Senone," *Proc. ICASSP*, pp. I-33-36, March 1992.

6. A. Kannan, M. Ostendorf and J. R. Rohlicek, "Maximum Likelihood Clustering of Gaussians for Speech Recognition," in *IEEE Transactions Speech and Audio Processing*, to appear July 1994.

7. H. Ney, R. Haeb-Umbach, B. Tran and M. Oerder, "Improvements in Beam Search for 10,000-word Continuous Speech Recognition," *Proc. ICASSP*, pp. I-9 - I-12, March 1992.

8. H. Murveit, J. Butzberger, V. Digalakis and M. Weintraub, "Large Vocabulary Dictation using SRI's DECIPHER™ Speech Recognition System: Progressive Search Techniques," *Proc. ICASSP*, pp. II-319 - II-322, April 1993.

9. L. Nguyen, R. Schwartz, F. Kubala and P. Placeway, "Search Algorithms for Software-Only Real-Time Recognition with Very Large Vocabularies," *Proc. ARPA Human Language Technology Workshop*, March 1993.

10. L. Bahl, S. De Gennaro, P. Gopalakrishnan and R. Mercer, "A Fast Approximate Acoustic Match for Large Vocabulary Speech Recognition," *Proc. Eurospeech 1989*.

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