summary of deliverables for

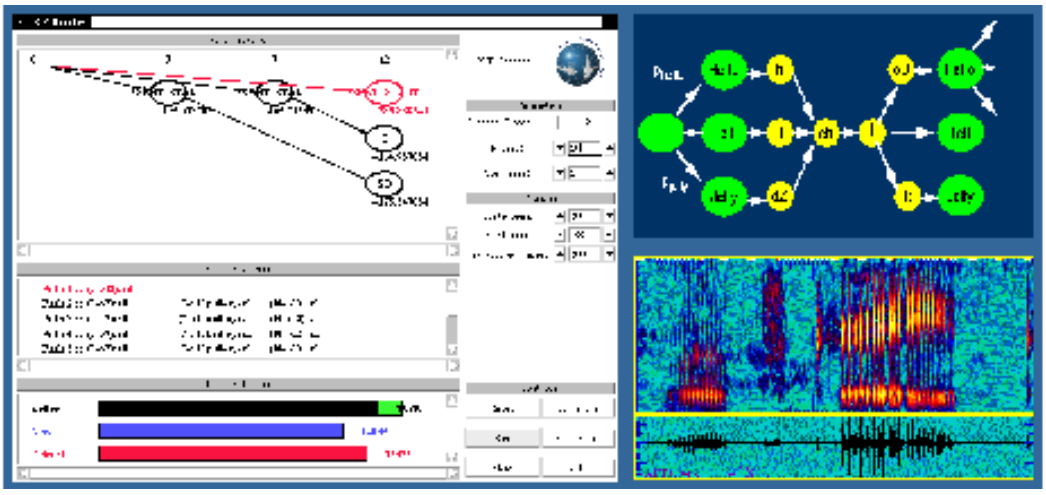
Robust Low Perplexity Voice Interfaces

Subcontract No. 43556

submitted to:

Dr. Fred J. Goodman
Signal Processing Center
The MITRE Corporation
1820 Dolley Madison Blvd., M/S W622
McLean, Virginia, USA 22102-3481

May 15, 2001



submitted by:

F. Zheng and J. Picone
Institute for Signal and Information Processing
Department of Electrical and Computer Engineering
Mississippi State University
Box 9571, 413 Simrall, Hardy Road
Mississippi State, Mississippi 39762
Tel: 662-325-3149, Fax: 662-325-3149
primary email contacts: {zheng, picone}@isip.msstate.edu

EXECUTIVE SUMMARY

This report summarizes the first set of deliverables for our one-year collaboration with the MITRE Corporation on a project titled “Robust Low Perplexity Voice Interfaces.” The two main goals for the first portion of this project were to develop a real-time Resource Management speech recognition system, and to experiment with its robustness to speech coding impairments on a limited recognition task (TIDigits). The longer term goal of this collaboration is to produce a prototype of a near real-time system that provides a robust and flexible command and control voice interface in realistic tactical noisy environments.

In the first phase of this project, we have successfully completed the first two milestones:

- Evaluated our baseline system on coded speech generated by the Mixed Excitation Linear Prediction (MELP) and Continuously Variable Slope Delta Modulation (CVSD) algorithms. The baseline system achieved a performance of 0.4% WER on clean speech; MELP coded speech produce a WER of 0.7% while CVSD coded speech produced a WER of 2.1%. Though the relative increase in error rate is considerable for both of these algorithms, the absolute performance is still respectable.
- Developed a real-time Resource Management recognition system that delivers a WER of 5.0% at 1.1 xRT on a 600 MHz processor. The baseline recognition system for this application is an error rate of 3.4% (very competitive with the state of the art) at 9.7 xRT.

In order to investigate the effects on recognition performance of different types of speech compression algorithms, we have used a simple small vocabulary industry standard application, TIDigits, as a testbed. In addition to a studio quality recording condition used to collect the original database, two types of compression algorithms, MELP and CVSD, were included in this experiment. We also added several mismatched conditions. For example, we trained on studio quality data and evaluated on MELP coded data. These experiments were conducted using two types of baseline recognition systems: a 16 mixture per state HMM-based recognizer that uses whole word acoustic models, and a similar system that uses cross-word triphones for its acoustic models. The performance of these baseline systems on the studio data was 0.4% and 0.6% WER respectively. The results for comparable versions of these system on MELP coded data was 0.7% and 0.8% respectively; for CVSD data results were 2.1% and 2.0% respectively. MELP seems very robust to mismatched training conditions: the performance on MELP data using the models trained on studio quality data was 1.1%. CVSD performance under mismatched conditions was much worse.

Our second major activity involved developing a real-time system on a task comparable in complexity to the DARPA Resource Management database (1000 words, bigram perplexity of 60). We used a 6 mixture crossword triphone model system with a bigram language model. This represents a reasonable trade-off between complexity (computations and memory), performance, and training data limitations. We developed a baseline system with a WER of 3.4% running at 9.7 xRT on a 600 MHz Pentium processor. We achieved this optimum baseline performance by tuning parameters related to the number of tied states, the language model scaling, and the word insertion penalty. To make this system real-time, we increased the frame duration from 10 msec to 15 msec, and methodically tuned the search algorithm using several beam width parameters. The resulting system achieves a 1.1 xRT and uses 46 MBytes of memory. Performance degrades from 3.4% to 5.0%. We believe with further effort we can significantly reduce this gap in performance.

TABLE OF CONTENTS

1. INTRODUCTION

State of the art speech recognition systems have achieved low error rates on low complexity tasks such as Resource Management [1] which involve clean data. Further, these systems are able to achieve near real-time performance. However, the performance of these systems rapidly degrades as the background noise-level increases. With the increasing popularity of low-bandwidth communication devices such as cell phones, a higher demand is being placed on the robustness of speech recognition systems. They are now required to perform at a near-zero error rate under various noise conditions and with real-time response. To date, this goal has not been achieved. In this project, our collaboration with the MITRE Corporation focuses on four primary research tasks to address this problem. In this first quarterly report we focus on the first of these:

- Develop a Near Real-Time Speech Recognition System: Deliver a near real-time system on a relatively small vocabulary of 1000 words which achieves a low WER of the order of 5%.
- Communicator Integration: Integrate the real-time system into the DARPA Communicator framework.
- Dynamic Run-Time Grammars: Develop a system capable of switching grammars on the fly.
- SPINE Evaluations: Evaluate the robustness of the system on the SPINE evaluation.

A major problem in applications such as military communications is the need to transmit over very low bandwidth channels. Speech communication applications employ coding algorithms to effectively deal with limited bandwidth. However, due to loss of spectral information, speech coding has a negative effect on the accuracy of speech recognition systems [2]. As the demand for mobile telephony increases, further reductions in codec bit rates are expected. This underscores the need for systems that are bit-rate insensitive. Several approaches have been proposed that deal with this problem. These approaches typically involve either regenerating the speech signal (decoding) prior to applying noise compensation or channel adaptation techniques [3, 4].

Once we have achieved robustness, there is still the problem of packaging this system into one running at real-time and in limited resources. There are numerous strategies to building a real-time system. These include using smaller acoustic models, tighter pruning, and fast approximation algorithms. A good example of this is the Hub-4E 10 xRT system evaluations [5, 6]. In these evaluations, groups took research systems running at 200-300 times real-time (xRT) and made them run at 10 xRT with little increase in error rate. The method for achieving real-time performance that we have explored in this quarter involves tuning the recognition parameters to achieve maximal performance. While this gives a good indication of the best performance a system will achieve on a particular data set, switching to another data set may require several iterations to optimize these tuned parameters.

In the first quarter of this project, we have developed a near real-time speech recognition system that supports the integration of improved signal processing and recognition technologies. We have used the ISIP ASR system — a public domain, cross-word, context-dependent, Hidden Markov Model (HMM) based system, freely available for both commercial and academic use with no licensing or copyright restrictions [7]. This deliverable consists of a baseline system that has a vocabulary on the order of 1000 words, and which can achieve word error rates (WER) on the order of 5% for tasks with perplexity less than 60. This system has been modified to run at near

real-time (~1 xRT) on a PC platform using a 600 MHz Pentium III processor while utilizing less than 128 Mbytes of memory. In this deliverable we include all software written in C++ that conforms to current standards supported by the GNU gcc compiler, thereby maximizing portability of the code to Unix and Windows platforms.

2. EXPERIMENTS ON CODED SPEECH

Motivated by the effects of speech compression algorithms on recognition performance, we have investigated the robustness of speech recognition systems on coded speech data. In particular, we have examined the effects of MELP and CVSD coding algorithms on recognition performance. For these experiments we used a connected digits task to analyze performance.

2.1. Data Preparation

The TIDigits corpus is a set of connected digits utterances collected by Texas Instruments (TI) [8]. The TIDigits corpus consists of more than 25 thousand digit sequences spoken by over 300 men, women, and children. The vocabulary contains eleven digits, “zero”, “one”, “two”,..., “nine”, and “oh”, under a loop grammar (any number of digits can occur in any sequence). The data was collected in a quiet studio environment and digitized at a 20 kHz sampling rate. The experiments discussed below begin by downsampling the data to 8 kHz. This data is referred to as STUDIO below.

Mixed Excitation Linear Prediction (MELP) [9] is a speech coding scheme based on the traditional LPC vocoder [10]. This coder offers a full parametric representation of the speech signal and can produce communication quality speech at a bit rate of 2.4 kbps [11]. It was selected by the United States Department of Defense Digital Voice Processing Consortium (DDVPC) [12] after an extensive testing program, as the best of seven candidates. The selection concentrated on four areas: intelligibility, voice quality, talker recognizability and communicability. MELP has also been found to be robust in difficult background noise environments that are frequently encountered in commercial and military communication systems.

The *Continuously Variable Slope Delta-modulation* (CVSD) [13] algorithm is a type of delta modulation in which the size of the steps of the approximated signal are progressively increased or decreased as required to make the approximated signal closely match the input signal. CVSD algorithm gives a speech quality marginally inferior to the other delta-modulation systems. However, CVSD has an important practical advantage that it is much less sensitive to the channel errors (e.g. lost packets) that are common in low bit-rate transmission.

To generate the coded speech data, we began by downsampling the TIDigits corpus from 20 kHz to 8 kHz. The studio-quality data was then passed through the MELP and CVSD codecs respectively to generate new coded corpora. The MELP data was produced for us by MITRE off-site. The CVSD data was generated from the 8 kHz studio-quality data using the CVSD coder provided by ViaSat. Each corpus then underwent the same feature extraction procedure. This involved generating 12 FFT-derived cepstral coefficients and log-energy. These features were computed using a 10 ms frame duration with a 25 ms Hamming window. First and second derivative coefficients of the base features were appended to produce a 39-dimensional feature

vector. The 12 base cepstral feature were then debiased using utterance-based cepstral mean subtraction.

2.2. Coded Speech Experiments

For these experiments, we trained both word models and cross-word triphone models for each of the three data sets: STUDIO, MELP and CVSD. The word models consisted of an HMM with a left-to-right self-loop topology as shown in Figure 1(d). The number of states set for each word model was proportional to the average duration of the word as measured by a forced alignment process. The word models were context-independent (CI), iteratively trained up to 16-mixture Gaussians per state.

The cross-word triphone models used a 3-state left-to-right self-loop topology as shown in Figure 1(a). The special topologies shown in Figure 1(b) and Figure 1(c) were used for the interword short pause (sp) model and the silence (sil) model respectively. The training procedure involved first generating single-mixture monophone models. These monophone models were trained using the Baum-Welch algorithm for a number of iterations. The trained monophone models were then used to seed cross-word context dependent (CD) triphone models. A state-tying procedure was used to cluster those states and models that were statistically similar. The state-tied triphone models were iteratively trained from one mixture up to 16-mixture Gaussians per state.

Initially, we used CVSD data provided by MITRE but found that the performance of our system with this data was very poor: 16.9% WER for word models and 7.1% WER for cross-word triphone models. This problem was found to be due to a few problems with the CVSD coder used to generate the CVSD data. The first problem was that the coder was suited only for processing

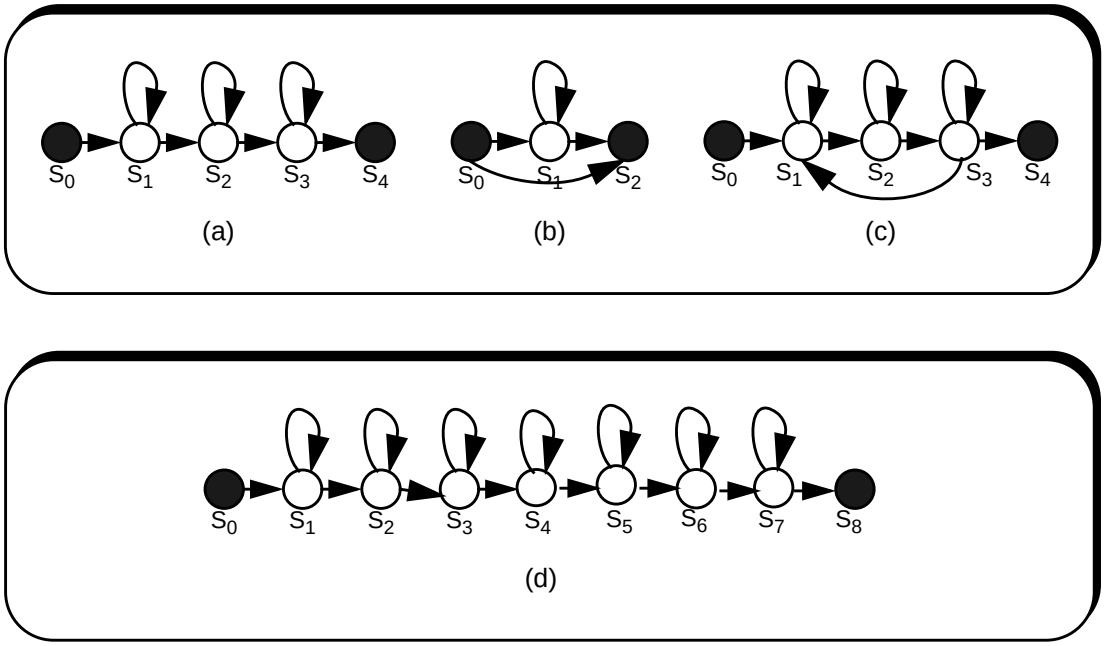


Figure 1. Some typical HMM topologies used for acoustic modeling: (a) typical triphone, (b) short pause, (c) silence, (d) typical word model. The shaded states denote the start and stop states for each model.

16 bits/sample input signal sampled at 16 kHz while we were using 8 kHz for the remainder of the experiments. We adjusted to this by simply downsampling to 16 kHz rather than 8 kHz. Second, the CVSD decoder converted the internal 8 bits/sample signal to a 16 bits/sample output signal but failed to rescale the output CVSD signal to match the max amplitude of the input signal. Thus, the decoder was outputting clipped data where many of the samples had an amplitude equal to the maximum possible amplitude. To solve this problem, we modified the source code to scale the output signal such that the maximum amplitude of the output signal matched the maximum amplitude of the input signal. Experiments showed that these fixes brought the error rate on the CVSD data to a reasonable range: 1.6% WER with word models and 1.7% WER with triphone models. Later, we obtained DoD's CVSD coder which has the ability to process 8 kHz input signal. The performance on 8 kHz CVSD speech generated by DoD's CVSD coder is marginally worse than the 16 kHz CVSD data, 2.1% and 2.0% on word models and triphone models respectively, but is comparable on the 8 kHz data.

Table 1 shows the coded speech experiments. All of these experiments were run using 8 kHz sampled data and without pruning to eliminate the effects of search errors. For the CVSD data, we used the DoD CVSD software. Experiments 1, 2, 3 were conducted on matched conditions where the training and test data were generated using the same coding scheme. The performance of the baseline systems on the STUDIO data was 0.4% and 0.6% WER respectively. The results for comparable versions of these system on MELP coded data were WER of 0.7% and 0.8% respectively, and on CVSD data were WER of 2.1% and 2.0% respectively. This would seem to indicate that MELP is a superior algorithm as compared to CVSD, and that MELP preserves more of the important spectral information contained in the input signal. This is an intriguing result since the bit rate for MELP is much lower than for CVSD.

The second set of experiments (4, 5, 6, and 7) tested mismatched training and test data. A desirable property of a robust system is that it be able to perform reasonably well no matter the channel degradation or coding scheme and to do so without training on the new channel condition. Clearly humans have the ability to normalize to vastly varying conditions, so we would

Experiment Number	Condition	Data		Word Error Rate	
		Training	Test	Word Models	Xwrd CD Models
1	Matched	STUDIO	STUDIO	0.4%	0.6%
2	Matched	MELP	MELP	0.7%	0.8%
3	Matched	CVSD	CVSD	2.1%	2.0%
4	Mismatched	STUDIO	MELP	1.1%	1.1%
5	Mismatched	STUDIO	CVSD	14.6%	14.2%
6	Mismatched	MELP	CVSD	22.2%	20.2%
7	Mismatched	CVSD	MELP	36.1%	48.1%

Table 1. Results of TIDigits experiments on coded speech. These experiments were performed on data sampled at 8 kHz using 16 mixture Gaussian HMM word models and cross-word triphone models. The full TIDigits training and test sets were used. The two cells that are highlighted demonstrate that recognition performance is fairly robust to the MELP coding algorithm (or equivalently, MELP does a much better job than CVSD at preserving the important aspects of the signal).

like our recognizers to do the same. It is interesting to note in Table 1 that there is only a slight degradation in WER when testing on MELP data for the case of matched training and mismatched STUDIO training. Again, this is a good indication that the MELP coding scheme is preserving most of the important information in the speech acoustic stream. Another somewhat surprising result is the difference between word models and triphone models on the different conditions. The word models were superior for all experiments which tested on either STUDIO or MELP data. This is to be expected since word models typically perform better than triphone models when there is sufficient training data. However, the triphone models performed better than the word models for the CVSD conditions.

3. TOWARDS A REAL-TIME SYSTEM

The primary goal of this quarter was to develop a near real-time system on a small vocabulary task. For this task, we chose to work with the DARPA Resource Management (RM) corpus [1]. The Resource Management corpus consists of prompted queries in very low background noise conditions. The prompts were chosen from a limited grammar with close to a 1000 word vocabulary and with a perplexity of less than 60. Recording was carried out using a headset microphone and simultaneously digitized at 20 kHz. Each recording session was then downsampled to 16 kHz for release.

The RM corpus is divided into training and test sets. The training data consists of 3990 training sentences (3.8 hours) from 109 speakers for speaker-independent speech recognition. The test set comprises four evaluation sets; *February 1989 evaluation, October 1989 evaluation, February 1991 evaluation,* and *September 1992 evaluation*. Each evaluation set consists of 300 sentences, adding up to a total of 1200 utterances (1.1 hours). The speaker sets for the training and test data are disjoint. Feature extraction for this data was similar to that used in the TIDigits experiments (Section 2.1) with the only difference being the number of filterbanks used. For data sampled at 8 kHz we normally use a 24th-order filterbank. The RM data, however, is sampled at 16 kHz. Further, we desired to keep the same features for the low frequency range (below 4 kHz) as we used in our TIDigits experiments. Thus, we divided the mel space evenly using the filter width specified by the TIDigits experiment. This produced a 32nd-order filterbank which is used to produce 12 cepstral coefficients.

The training procedure used was also similar to the training performed for TIDigits experiments. We obtained 6-mixture cross-word triphone acoustic models. The trained acoustic models were used for decoding along with a standard RM bigram language model. The initial experiment had a WER of 5.0% averaging over all the four evaluation sets, shown as experiment 04 in Table 2. An interesting observation is that the sep’92 set appears to be much harder than the other test sets.

3.1. System Tuning

Experiment 04 mentioned above was a simple first-cut experiment that allowed us to see where we stood compared to state of the art. However, the parameters used in that experiment were a mixture of those used for LVCSR experiments and TIDigits experiments. We were certain that this 5.0% was not the best performance possible for our system on this data. Further, for our real-time experiments to produce meaningful results, we needed to generate a tight baseline

performance number. Thus, we developed a tuned baseline system with a WER of 3.4% running at 8.3 xRT on a 600 MHz Pentium processor. We achieved this optimal baseline performance by tuning the following parameters: number of tied states, language model scaling, and word insertion penalty. During the tuning, we found the performance of the four test sets tended to increase or decrease independently (i.e. there was no globally best parameter setting across all test sets) so we maximized the overall performance.

Language Model Scaling and Word Insertion Penalty

The decoder uses a bigram backoff language model for the RM task. During the decoding process, the language model probability is multiplied by a scaling factor. This weights the importance of the language model probability to the overall path score for a hypothesis. Increasing the scale value causes the language model score to contribute more towards the overall path score — essentially boosting the importance of the language model relative to the acoustic model. Decreasing the scale causes the acoustic model score to dominate. Also during decoding a word insertion penalty is added to the overall path score for a hypothesis when a word is hypothesized. This parameter helps to balance deletions and insertions of words. Increasing this value will increase the number of insertions, decreasing it will decrease the number of insertions. Tuning these two parameters simultaneously allows one to effectively trade-off insertions, deletions and substitutions to yield an overall best error rate.

Our first experiment (experiment 04 in Table 2) on RM was performed by setting the language model scale to 12.0 and the word insertion penalty to -10. These are the settings we normally use for conversational speech systems. We then systematically tuned each parameter while holding the other parameters constant. We searched the language model scale values in increments of 2.0. A language model scale of 6.0 and a word insertion penalty of -10 gave the best performance of 4.5% WER, a 10% relative reduction.

Exp	LM Scale	Word Penalty	Word Error Rate				
			Feb89	Oct89	Feb91	Sep92	Average
01	18.0	20	4.3	5.3	4.1	7.5	5.3
02	16.0	20	4.1	4.7	3.9	7.0	4.9
03	14.0	20	4.0	4.3	3.7	7.0	4.7
04	12.0	-10*	4.1	4.5	3.4	8.1	5.0
05	12.0	20	3.9	4.4	3.5	7.0	4.7
06	10.0	20	3.7	4.6	3.3	6.9	4.6
07	8.0	20*	3.7	5.0	5.2	8.9	5.7
08	6.0	20*	4.1	4.6	3.4	8.2	5.0
09	6.0	-10	3.6	4.3	3.1	7.1	4.5
10	4.0	20*	4.7	5.1	4.8	9.8	6.1

Table 2. Comparison of experimental results by tuning the language model scale and word insertion penalty. 4.5% WER was the best error rate achieved. A '*' denotes those systems where deletion errors and insertion errors have not been balanced.

State-Tying

A problem often associated with training context-dependent models in a speech recognition system is the lack of sufficient training data for the large number of models in the system. To avoid this problem the ISIP system employs a maximum likelihood phonetic decision tree-based state-tying procedure [16] to pool HMM states. Each node of the phonetic decision tree is associated with a set of states. These states are iteratively separated into child nodes using phonetic questions. When the tree is trained, the states in a single leaf node of the tree represent acoustically similar states that can be tied together. This leads to better parameter estimates. The parameters governing the state-tying process are the thresholds for splitting and merging a node [16].

Experiments were performed by tuning the state-tying parameters to get the optimal number of tied states. Table 3 shows the performance improvement due to state-tying. All these experiments used the optimal language model scale and word insertion penalty found above: language model scale of 6.0 and word insertion penalty of -10. By tuning the parameters related to the number of tied states we were able to improve the performance from 4.5% to 3.5% WER, a 22% relative decrease. The optimal number of tied states found in experiment 15 was 3554. An interesting demonstration of the power of state-tying is experiment 19 where no state-tying was done. Here, one can see that the models are severely undertrained and this produced poor performance.

Baseline System

A final change to our baseline system involved fixing a small bug in our training process. During training, we tie the central state of the short pause model to the central state of the silence model. However, our state-tying procedure had previously been untying these two states. We simply fixed this process so that the short pause and silence models remain tied throughout the training process. Retraining the models and tuning the parameters as above, we get slightly different

Exp	Tied States	Word Error Rate				
		Feb89	Oct89	Feb91	Sep92	Average
11	529	4.0%	5.1%	4.7%	9.0%	5.7%
12	1006	3.4%	4.4%	3.6%	7.4%	4.7%
09	1378	3.6%	4.3%	3.1%	7.1%	4.5%
13	1946	2.9%	4.4%	3.1%	5.5%	4.0%
14	3073	2.9%	3.9%	2.3%	5.2%	3.6%
15	3554	2.8%	3.5%	2.6%	5.2%	3.5%
16	4004	3.3%	4.4%	2.6%	5.9%	4.0%
17	4902	3.1%	3.7%	2.9%	6.3%	4.0%
18	8392	7.6%	9.5%	7.1%	12.3%	9.1%
19	16299	35.4%	36.3%	34.3%	41.7%	36.9%

Table 3. Comparison of performance while tuning the number of tied states. All experiments were conducted using the identical conditions as exp 09 from Table 2: (language model scale = 6.0, word insertion penalty = -10). 3554 tied states delivers the local optimum performance of 3.5% WER.

values for the language model scale and number of tied states. Overall these changes yield only a 0.1% absolute change in WER. Our final baseline system (experiment 20 in Table 4), yields a 3.4% WER and runs at 8.3 xRT on a 600 MHz Pentium processor. The real-time rates do not include the time necessary to load the acoustic and language models.

Real-time system

To take this baseline system from 8.3 xRT to near real-time we explored two simple methods for increasing the decoding speed: increased pruning and increased frame duration. In most state-of-the-art decoders, pruning techniques are used to reduce the Viterbi search space and to improve the search speed and memory requirements. This pruning typically involves setting a threshold at each frame in the search where only paths whose score is higher than that threshold are extended to the next frame. Our decoder allows the user to set a separate threshold, or beam, at each level in the search hierarchy (typically state-level, model/phone-level and word-level). Tuning these beams can result in significant runtime savings without significantly degrading the performance.

Table 5 shows the performance of the RM baseline system at varying pruning rates. One can see that the performance of the system quickly degrades as more paths are pruned in the search space. We are able to achieve near real-time performance (1.8 xRT) but at a relatively high cost in the error rate. In experiments 24 and 25, for example, it is obvious that overpruning is taking place since there is a relatively minor change in the real-time rate but large losses in WER.

Exp	Tied States	LM Scale	Word Penalty	Word Error Rate					Runtime (xRT)
				Feb89	Oct89	Feb91	Sep92	Average	
15	3554	6.0	-10	2.8%	3.5%	2.6%	5.2%	3.5%	9.9
20	3565	7.0	-10	2.9%	3.4%	2.2%	5.2%	3.4%	8.3

Table 4. Baseline system: after tying sp to silence model, we achieve our baseline result of 3.4% using the parameter settings shown by experiment 20.

Exp	Beam Pruning			Word Error Rate					Memory (Mbyte)	Runtime (xRT)
				Feb89	Oct89	Feb91	Sep92	Average		
20	250	200	200	2.9%	3.4%	2.2%	5.2%	3.4%	111	9.7
21	200	200	200	3.5%	4.3%	3.2%	6.1%	4.3%	57	2.5
22	200	150	150	3.6%	4.6%	3.3%	7.4%	4.7%	44	2.3
23	200	150	100	4.1%	5.0%	3.5%	7.7%	5.1%	39	1.8
24	150	150	150	4.8%	6.2%	4.6%	10.4%	6.5%	36	1.4
25	200	100	100	5.8%	7.8%	6.4%	12.3%	8.1%	33	1.1

Table 5. Comparison of experimental results obtained by tuning the beam pruning parameters. All experiments are conducted using a language model scale of 7.0, a word insertion penalty of -10 and a number of tied states set to 3565. The best system decreased the real time rate by a factor of ~6 but also increased the word error rate by 1.7% absolute.

A second technique we tried for reducing the real-time rate of the decoder was to increase the frame duration. Normally we use a 10 ms frame duration in the front-end. This amounts to 100 frames evaluated per second. By increasing this to 20 ms, for instance, we could decrease by a factor of two the number of frames that would have to be evaluated. Since the memory and speed requirements of a decoder are proportional to the length of the utterance, the savings could be significant. For this experiment, we regenerated the feature data with a 15 ms frame duration and a 27.5 ms Hamming window. The models were then retrained and the parameters described above were all retuned to this frame rate. Our final system achieves a 5.0% WER at 1.1 xRT as shown in Table 6.

Exp	Beam Pruning			Word Error Rate					Memory (Mbyte)	Runtime (xRT)
				Feb89	Oct89	Feb91	Sep92	Average		
26	250	200	200	3.9%	4.1%	2.7%	6.4%	4.3%	109	5.2
27	200	200	200	4.0%	3.9%	2.7%	6.8%	4.4%	59	1.5
28	200	150	150	4.2%	4.5%	2.9%	7.7%	4.8%	49	1.3
29	200	160	100	4.3%	4.8%	3.1%	7.5%	4.9%	47	1.1
30	190	160	100	4.3%	4.8%	3.1%	7.7%	5.0%	46	1.1

Table 6. Comparison of experimental results obtained by changing the frame duration from 10 ms to 15 ms. All parameters were retuned to this frame rate.

3.2. Analysis of Real-Time Performance

We normally expect that, to take a system operating above real-time and make it operate in real-time, we should incur of cost of no more than a 25% increase in word error rate. Our best system presented above has a WER of 3.4% so our goal is to make our real-time system achieve ~4.3% WER. Currently we were only able to achieve 5.0% WER for a near real-time system. We believe, though, that the gap can be easily narrowed by considering faster algorithms rather than better parameter tuning.

First and foremost amongst these is to build more efficient search process. While we can always decrease the computation by using an aggressive beam search pruning strategy, our goal here is to decode the speech with the least amount of computation while still obtaining accuracy close to that of our baseline system. Simply depending on beam pruning will eventually hurt system performance quickly. A speech recognition system typically consists of four components: feature extraction, acoustic modeling, language modeling and search. Figure 2 shows the percentage of CPU time used by each of these components in our best RM system. Here, the search component and acoustic modeling (Gaussian evaluation) component consume the majority of the

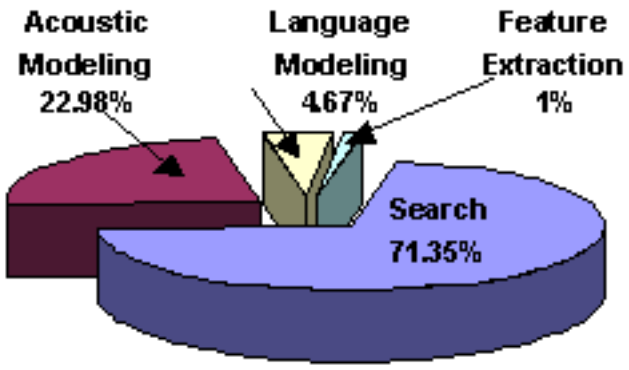


Figure 2. Percentage of CPU time each component takes in a real-time system.

resources. Thus, we are currently exploring techniques to more efficiently organize the search space in our decoder and ways to decrease the amount of resources related to Gaussian evaluation.

One project currently being pursued by our lab is the investigation of Fast Gaussian Computation (FGC) techniques. In particular, we are exploring the Bucket Box Intersection algorithm (BBI) [17]. The basic idea of BBI is to dynamically determine the most significant set of Gaussians for a particular input observation. The set is determined by searching a K-dimensional space partitioning tree. Experiments on a conversational speech corpus using 12 mixture cross-word models and a trigram language models have shown that BBI can give ~30% speedup in Gaussian evaluations with only 0.73% relative degradation in WER, and ~50% speedup in Gaussian with 3.4% relative degradation in WER.

3.3. Final Deliverable

In this quarter we are delivering two packages (download from [18]): one is our baseline RM system, and the other is our real-time RM system. Both packages consist of final models, list files, scripts, and experimental outputs.

Example Directories and files:

./rm_baseline/	(package directory;./rm_realtime/ for real-time system)
./rm_baseline/final_models/	(contains models, lexicon, phone, and bigram files)
./rm_baseline/params/params.text	(decoding parameter file)
./rm_baseline/lists/	(mfcc, raw, reference list files. these should be modified to have the correct file paths)
./rm_baseline/scripts/	(the scripts for generating feature data)
./rm_baseline/output/	(output directory of hypotheses)

Running Experiments:

- Install the prototype system [19].
- Generate the evaluation feature data: run ***extract_eval.sh*** scripts. Before running it, you must make sure the ***eval_mfc.list*** and ***eval_raw.list*** files have the right file paths. Note that the feature data of the baseline system and the real-time system are different, as the former is generated with 10ms frame duration and 25 ms window size, and the other is generated with 15ms frame duration and 27.5 ms window size.
- Decoding: run ***run_eval.sh*** scripts, which will output the runtime. The data size is ~1.1 hours, so in term of real-time the runtime will be (runtime / 1.1 hours) xRT;
  Notes: one can also manually set the parameters according to Section 3.1 to replicate those experiments described above. The parameters of the baseline and real-time systems must be configured according to Table 7.
- Scoring: run ***score_eval.sh*** scripts, which will output the scoring of four evaluation sets and the overall scoring.

System	lm_ scale	word_ penalty	beam_pruning			Word Error Rate					Memory (MBs)	Run-time (xRT)
						Feb89	Oct89	Feb91	Sep92	Avg		
Baseline	7.0	-10	250	200	200	2.9%	3.4%	2.2%	5.2%	3.4%	111	9.70
Real-time	14.0	40	190	160	100	4.3%	4.8%	3.1%	7.5%	5.0%	46	1.07

Table 7. Configuration of parameters, performance and runtime of baseline and real-time systems

4. FUTURE WORK

In this quarter, we have delivered a baseline system performing at near real-time with only a small WER performance degradation on the Resource Management task. In the next quarter we will continue pursuing this line of research. However, we will focus less the next quarter on parameter tuning and more on algorithm enhancement to produce a faster system with much lower degradations in error rate. Some avenues we plan to pursue include data structure enhancements in the search algorithm, fast Gaussian evaluation methods, and improved memory efficiency of the decoder.

Our primary task in the next quarter will be to focus on the deployment of our speech recognition technology. This implies the ability to communicate recognition results over a wide variety of communication architectures using different speech compression techniques and modulation/ encoding schemes. Such systems currently exist for wired LAN applications (such as the DARPA Communicator architecture). By August 15, 2001, we plan to deliver a system, which will integrate the baseline system into a thin-client wireless networking architecture under development at MITRE. We will provide the necessary C++ wrappers and event handlers that will allow the system to conform to an application programming interface defined by MITRE.

5. ACKNOWLEDGMENTS

We wish to acknowledge Drs. Fred J. Goodman, Bryan George, and George Shuttic of the Signal Processing Center at MITRE Corporation for their continued support and feedback.

6. REFERENCES

[1] P. Price, W. Fisher, J. Bernstein, and D. Pallett, “The DARPA 1000-Word Resource Management Database for continuous speech recognition,” *IEEE International Conference on Acoustics, Speech, and signal Processing*, pp. 651-654, 1988.

[2] J. M. Huerta, “Speech Recognition in Mobile Environments,” Ph.D. Dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA, April, 2000.

[3] European Telecommunication Standards Institute, “European digital telecommunications system (Phase 2); Full rate speech processing functions (GSM 06.01),” ETSI 1994.

[4] C. Mokbel, L. Mauuary, D. Jouvet, J. Monne, C. Sorin, J. Simonin, and K. Bartkova, "Towards Improving ASR Robustness for PSN & GSM Telephone Applications," *2nd IEEE Workshop on Interactive Voice Technology for Telecommunications Applications (IVTTA1994)*, Vol 1: 73-76, 1996.

[5] J. Davenport, L. Nguyen, S. Matsoukas, R. Schwartz, and J. Makhoul, "The 1998 BBN Byblos 10x Real Time System," 1999 DARPA Broadcast News Workshop, Herndon, VA, February 1999.

[6] M. Ravishankar, R. Singh, B. Raj, and R. Stern, "The 1999 CMU 10X Real Time Broadcast News Transcription System," 2000 NIST Speech Transcription Workshop, May 2000.

[7] N. Deshmukh, A. Ganapathiraju, J. Hamaker, J. Picone and M. Ordowski, "A Public Domain Speech-to-Text System," *Proceedings of the 6th European Conference on Speech Communication and Technology*, vol. 5, pp. 2127-2130, Budapest, Hungary, September 1999.

[8] R. G. Leonard, "A Database for Speaker-Independent Digit Recognition," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 3, pp. 42.11, Dallas, Texas, USA, 1984.

[9] "Analog to Digital conversion of Voice by 2,400 Bit/second Mixed Excitation Linear Prediction (MELP)," Federal Information Processing Standards Publication (FIPS PUB) Draft, June 12, 1997.

[10] W. Lin, S. Koh, and X. Lin, "Mixed excitation linear prediction coding of wideband speech at 8 kbps," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. 1137-1140, 2000.

[11] A. McCree, K. Truong, E. B. George, and T. P. Barnwell, "A 2.4 kbit/s MELP Coder Candidate for the New U.S. Federal Standard," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 200-203, Atlanta, Georgia, USA, May 1996.

[12] "The U.S. Department of Defense Digital Voice Processor Consortium," *http://www.plh.af.mil/ddvpc/index.html*, Department of Defense, USA, January 1998.

[13] P. E. Papamichalis, *Practical Approaches to Speech Coding*, Prentice-Hall Publishing, Englewood Cliffs, New Jersey, 1987.

[14] "Grammar_Compiler," *http://www.isip.msstate.edu/projects/speech/education/toolkits/asr/language_models/grammar_compiler/index.html*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, June 1999.

[15] N. Deshmukh, A. Ganapathiraju and J. Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition," IEEE Signal Processing Magazine, vol. 16, no. 5, pp.

84-107, September 1999.

[16] J. Zhao, X. Zhang, A. Ganapathiraju, N. Edshmuk, and J. Picone, "Decision Tree-Based State Tying for Acoustic Modeling," *http://www.isip.msstate.edu/publications/reports/isip_lvcsr/1999/decision_tree/doc/report_061599.pdf*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, June 1999.

[17] J. Fritsch and I. Rogina, "The Bucket Box Intersection (BBI) Algorithm for Fast Approximative Evaluation of Diagonal Mixture Gaussians," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. 837-840, Atlanta, Ga, USA, 1996.

[18] "Robust Low Perplexity Voice Interfaces," *http://www.isip.msstate.edu/projects/robust_low_perplexity/index.html*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, May 2001.

[19] "ISIP AUTOMATIC SPEECH RECOGNITION SYSTEM," *http://www.isip.msstate.edu/projects/speech/software/asr/download/asr/index.html*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, May 2001.