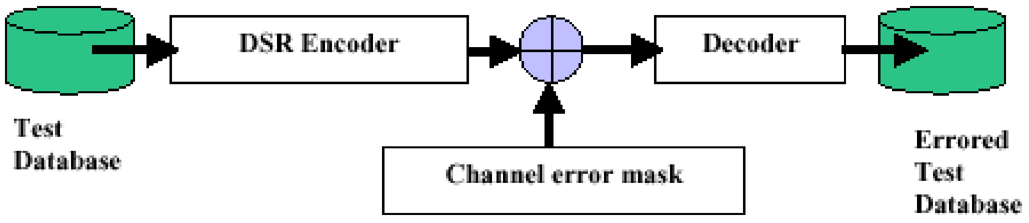
Aurora Working Group: DSR Front End LVCSR Evaluation — Baseline Recognition System Description

submitted to:

David Pearce
European Telecommunications Standards Institute (ETSI)
650, Route des Lucioles
06921 Sophia Antipolis CEDEX
FRANCE

July 21, 2001



submitted by:

N. Parihar 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: {parihar, picone}@isip.msstate.edu

EXECUTIVE SUMMARY

In this document we describe the features of the baseline system to be used in the Distributed Speech Recognition (DSR) front end large vocabulary continuous speech recognition (LVCSR) evaluations being conducted by the Aurora Working Group of the European Telecommunications Standards Institute (ETSI). The objective of these evaluations is to determine the robustness of different front ends for use in client/server type telecommunications applications. As such, our experiments are designed to test the following focus conditions on the DARPA Wall Street Journal (WSJ0) corpus using a 5000-word closed-loop vocabulary and a bigram language model:

1. **Additive Noise**: six noise conditions collected from street traffic, train stations, cars, babble, restaurants and airports will be digitally added to the speech data to simulate degradations in the signal-to-noise ratio of the channel.
2. **Sample Frequency Reduction**: the reduction in accuracy due to decreasing the sample frequency from 16kHz to 8kHz will be calibrated.
3. **Microphone Variation**: performance for both microphone conditions contained in the WSJ0 corpus will be analyzed.
4. **Compression**: degradations due to data compression of the feature vectors will be evaluated.
5. **Model Mismatch**: the degradation due to a mismatch between training an evaluation conditions will be calibrated.

The first step in this project was to design a baseline system that will provide a stable point of comparison with state-of-the-art WSJ0 systems. This baseline system was trained on 7,138 clean utterances from the SI-84 WSJ0 training set. These utterances were parameterized using a standard mel frequency scaled cepstral coefficient (MFCC) front end that uses 12 FFT-derived cepstral coefficients, log energy, and the first and second derivatives of these parameters. From these features, state-tied cross-word triphone acoustic models with 16 Gaussian mixtures per state were generated. The lexicon was extracted from the CMU dictionary (version 0.6) with some local additions to cover the 5000 word vocabulary. Recognition was performed using a single pass dynamic programming-based search guided by a standard backoff bigram language model.

The NIST Nov’92 dev test and evaluation sets were used for our evaluations. Our initial experiments used a 300 utterance subset of the 1206 utterance dev test set, and also used a set of pruning thresholds and scaling parameters based on our Hub-5E conversational speech evaluation system. The baseline system yielded a word error rate (WER) of 10.8% on the dev test set. Tuning various parameters decreased the error rate to 10.1% on the dev test subset and 8.3% on the evaluation set. State-of-the-art systems developed by other sites such as the HTK Group at Cambridge University have achieved a 6.9% WER on the same task. The principal difference between their system and the system presented here seems to be a proprietary lexicon which was developed to improve performance on the WSJ task.

The next step in this project will involve the completion of benchmarks using the ETSI standard front end. Since this front end is very similar to the MFCC front end described above, its performance is expected to be consistent with our previous results. We will evaluate a total of 112 conditions: 6 noise types + clean data x 2 compression types x 2 sampling rates x 2 microphone conditions x 2 training conditions. The system resulting from these experiments will serve as the baseline for all subsequent experiments.

TABLE OF CONTENTS

1. INTRODUCTION

State-of-the-art speech recognition systems have achieved low error rates on medium complexity tasks such as Wall Street Journal (WSJ) [1] which involve clean data. However, the performance of these systems rapidly degrades as the background noise level increases. With the growing popularity of low-bandwidth miniature communication devices such as cell phones, palm computers, and smart pagers, a much greater demand is being created for robust voice interfaces. Speech recognition systems are now required to perform at a near-zero error rate under various noise conditions. Further, since many of these portable devices use lossy compression to conserve bandwidth, speech recognition system performance must also not degrade when subjected to compression, packet loss, and other common wireless communication system artifacts. Speech coding, for example, is known to have a negative effect on the accuracy of speech recognition systems [2].

Aurora, a working group of ETSI, has been formed to address many of the issues involved in using speech recognition in mobile environments [6]. Aurora's main task is the development of a distributed speech recognition (DSR) system standard that provides a client/server framework for human-computer interaction. In this framework, the client side performs the speech collection and signal processing (feature extraction) using software and hardware collectively termed as a *front end*. The processed data is transmitted to the server for recognition and subsequent processing. The exact form and function of the front end is a design factor in the overall DSR structure. Our collaboration with the Aurora Working Group focuses on evaluating the performance of different front ends on the WSJ task for a variety of impairments:

- **Additive Noise**: Six noise types collected from street traffic, train terminals and stations, cars, babble, restaurants and airports at varying signal-to-noise ratios (SNRs) are artificially added. For a training utterance, the noise (one of six types) and SNR conditions (between 10 and 20 dB in steps of 1 dB) are randomly chosen. For each of the six test conditions, the SNR is randomly chosen between 5 and 15 dB in steps of 1 dB. G.712 filtering [4] is used to simulate the frequency characteristics at an 8 kHz sample frequency. P.341 [5] is used to simulate frequency characteristics at a 16 kHz sample frequency. Filtering is applied to both noisy and clean data.
- **Sample Frequency Reduction**: Two sampling rates, 16kHz and 8kHz, will be evaluated. Current telecommunications technology operates at a sampling rate of 8 KHz but a goal of next-generation technologies is to increase this to 16 KHz in order to increase quality.
- **Microphone Variation**: The data was recorded using a two-channel recorder. All speakers used a head-mounted Sennheiser HMD-410 close-talking microphone, which was recorded on one of these two channels. The second channel contained the same acoustic data recorded using a second microphone that was selected from a group of 18 microphones which included such common types as a Crown PCC-160, Sony ECM-50PS, and a Nakamichi CM100.
- **Compression**: A vector quantization-based compression scheme defined by the Aurora Working Group [3] which compresses features to 4800 b/s is being used to evaluate the degradation in performance due to compression.
- **Model Mismatch**: Mismatched training and testing conditions are inevitable in a rapidly evolving area such as wireless communications. To better understand this issue, a simple evaluation of mismatched conditions is being performed. One set of models is trained on only the Sennheiser microphone data with no additive noise (often referred to as clean data). A second set of models is trained on a combination of clean and noisy data using both microphone conditions. Both models are evaluated on the same test conditions that contain a broad range of noise conditions.

In the present project, our goal is to develop a baseline recognition system to conduct these evaluations. We have defined these milestones in accordance with these goals:

- **Develop a baseline WSJ0 system:** The baseline system should be fairly simple yet deliver near state-of-the-art performance. It will use a bigram language model and a 5000-word closed vocabulary (no out of vocabulary words in the evaluation set). This system will initially use the ISIP front end with standard mel cepstral features. It will be trained using a mixture of clean and noisy data described in [6] and evaluated on clean data. This baseline system provides a comparison point to insure that future results are credible.
- **Provide an HTK feature file interface**: We will develop an enhanced version of the ISIP prototype system, which will be identified as v5.10, that provides direct support for reading and writing of HTK-formatted binary features. This version also is capable of computing the first and second derivative for features on the fly, rather than requiring these to be stored in a feature file.
- **Integrate the ETSI front end:** The final baseline system will use an implementation of the industry-standard mel frequency-scaled cepstral coefficient front end developed in previous Aurora standards activities [3] (DSR-MFCC). This system will be trained in a manner identical to the MFCC baseline system described above. The system will be evaluated on 14 test conditions (6 noise conditions plus clean speech and using each of the two microphone conditions).
- **Optimize system performance:** The final configuration of the baseline system will be determined by optimizing the parameters of the system with respect to the clean version of the development test set using clean training data collected with the Sennheiser microphone. This system, whose parameters will be fixed, will then be evaluated across a range of training and test conditions involving the six noise sources previously described.
- **Evaluate performance at 8 kHz:** The experiments described above will the repeated on the same data resampled at 8 kHz. Downsampling will be performed using the procedure described in [6].
- **Evaluate the impact of compression:** The preceding experiments will be repeated using the DSR front end compression algorithm [3].

In this report we focus on the first two milestones listed above. We have conducted a set of baseline experiments to provide a point of comparison with state-of-the-art WSJ0 systems. In the following sections we describe the baseline system and provide details of our experiments.

2. AN OVERVIEW OF THE EVALUATION CORPUS

The first step in evaluating front ends for telecommunications applications is to define an evaluation paradigm. The Aurora Working Group decided to follow a standard common evaluation paradigm used extensively in the speech research community. Further, it was decided to leverage existing resources, namely the DARPA Wall Street Journal Corpus (WSJ) [1], and to evaluate noise conditions by digitally adding noise [6]. WSJ is a large vocabulary continuous speech recognition corpus consisting of high-quality recordings of read speech. Two-channel recordings were made at 16 kHz. Channel 1 consisted of the same microphone for all speakers — a Sennheiser HMD-414 close-talking microphone that was extremely popular at the time. The second channel included a sampling of 18 different types of microphones. The text material for this corpus was drawn from newspaper articles appearing in the Wall Street Journal. A portion of the data included utterances containing verbalized punctuation (“John COMMA who came home early COMMA decided to read the newspaper PERIOD”).

The data is divided into a sequence of training (train), development (dev test) and evaluation (eval) sets. Further, the Aurora Working Group decided to focus on the 5000 word evaluation task. This is an interesting task in that the evaluation set is defined in such a way that a 5000 word vocabulary, which is distributed with the corpus, is sufficient to give complete coverage of the evaluation set. This means there are no out of vocabulary words (OOVs) in the evaluation set. This task is often referred to as the 5k closed vocabulary task. It is a popular approach when one wants to focus on acoustic modeling problems, and remove language modeling issues from the evaluation. A bigram language model (LM) [7] is also distributed with the corpus as a reference language model.

In our experiments we are using the standard SI-84 training set. This set contains 7,138 utterances from 83 speakers, totaling 14 hours of speech data. The SI-84 set contains a mixture of utterances with and without verbalized punctuation. A typographic error in the training transcriptions, “EXISITING” instead of “EXISTING”, was fixed before training. For evaluation, we are using two data sets: a 330 subset (described below) of the 1206 utterance dev test set, and the complete November 92 NIST evaluation set [8]. The latter is a speaker-independent evaluation set consisting of 330 utterances from 8 speakers. All data sets are drawn exclusively from the channel 1 data (Sennheiser microphone).

Due to time constraints and the large number of experiments that need to be run to effectively tune a system, we decided to reduce the 1206 utterance dev test set to a 330 utterance set which was comparable in size to the evaluation set. To do this, we decided to preserve all 10 speakers represented in the dev test, and select 33 utterances per speaker. These utterances were selected such that the duration profile of the 330 utterance subset was a good model of the entire 1206 utterance set (measured in words per utterance). Since we had a large number of utterances per speaker, we did not pay too much attention to finer details such as the entropy of the word N-gram distributions and the speaking rates.

The pronunciations contained in the lexicon were prepared using the publicly available CMU dictionary (v 0.6) [9] with some local additions made to give full coverage of the training set. The additions needed for the training lexicon are shown in Table 1. All stress markers in the CMU dictionary were removed and the words “*!SENT_START*” and “*!SENT_END*” were added to follow our lexicon format. Each pronunciation was replicated twice in the lexicon (one ending with the *sil* phoneme and one with *sp*) to model both long and short interword silences (a requirement for the technology being used in the baseline system). Similarly, an evaluation lexicon was prepared from the CMU dictionary with local additions as shown in Table 2.

The 5K bigram LM and associated lexicon do not give complete coverage of the dev test set. Since our interest is in experiments with a 0% OOV rate, we decided to augment the LM with the missing words. There are several ways this can be done. We chose an interpolation technique supported in the SRI Language Modeling Toolkit (SRILM) [10]. We constructed an interpolated bigram LM by generating an LM on the test set, and interpolating it with the existing bigram such that the overall perplexity of the modified LM was comparable to the original LM. The original LM had a perplexity of 147. The interpolated LM was constructed by setting the interpolation factor such that the final perplexity was the same. The resulting value of this interpolation factor was 0.998. This interpolated LM was only used for tuning experiments on the dev test set.

Word	Pronunciation
PHILIPPINES	F IH L IH P IY N Z
PHILIPS	F IH L AH P S
PURCHASING	P ER CH AH S IH NG
ROUTE	R AW T R UW T
ROUTINE	R UW T IY N
ROVER	R OW V ER

Table 1. Local additions to the CMU lexicon needed for coverage of the training data set.

Word	Pronunciation
PURCHASING	P ER CH AH S IH NG
ROUTES	R AW T S R UW T S
ROUTINELY	R UW T IY N L IY
ROVING	R OW V IH NG

Table 2. Local additions to the CMU lexicon needed for coverage of the evaluation set.

3. A DESCRIPTION OF THE BASELINE SYSTEM

The baseline system to be used for the Aurora evaluations is based on a public domain speech recognition system that has been under development at the Institute for Signal and Information Processing (ISIP) at Mississippi State University for several years. We refer to this system as the prototype system [11] since it was the first recognition system we built and served as a testbed to develop ideas for implementing conversational speech systems. This system is implemented entirely in C++ and is fairly modular and easy to modify. It has been used on several evaluations conducted by NIST [12,13] and the Naval Research Laboratory [14]. The prototype system is predecessor of a more grandiose attempt to build a modular and flexible recognition system known as the production system [15]. The production system is in early stages of alpha release, and is not mature enough for a formal evaluation. The production system will be compatible with the prototype system, and should be released by the end of the year.

Both of these system share the same general computational and algorithmic frameworks. We use hidden Markov model based context-dependent acoustic models [16], lexical trees [17] for cross-word acoustic modeling, N-gram language models with backoff probabilities [18] for language modeling (finite state networks are also supported), and a tree-based lexicon for pronunciation modeling [19]. The core of the system is a hierarchical dynamic programming-based time synchronous network search engine [20,21] that implements a standard beam-pruning approach for maximizing search accuracy while minimizing memory requirements. Numerous papers describing these algorithms in more detail can be found at ISIP's web site [22] along with several short courses and training materials [23,24]. Annual workshops [25,26] are held at which we train users and provide more details on the theory behind this technology.

The focus of this project is the signal processing component of the system. A good overview of signal processing in speech recognition can be found in [27]. The most popular front end in use today employs mel frequency-scaled cepstral coefficients. This front end is summarized on Figure 1. The speech signal is preemphasized to enhance high frequencies and is then analyzed using a 10 ms frame and a 25 ms Hamming window. For each frame of speech data, we compute 12 mel-scaled FFT-derived cepstral coefficients and a single log energy feature for a total of 13 base features. We refer to these features as absolute features since they measure the absolute magnitude of the spectrum (or energy).

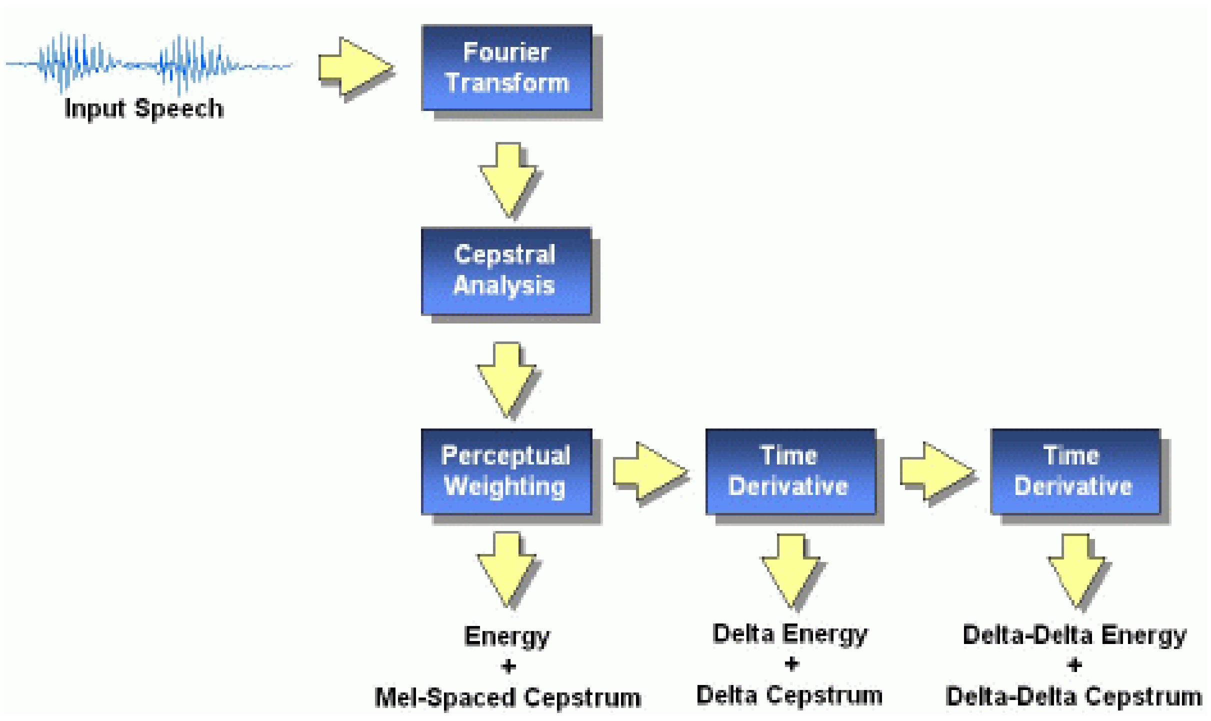

Input Speech
Fourier Transform
Cepstral Analysis
Perceptual Weighting
Time Derivative
Time Derivative
Energy + Mel-Spaced Cepstrum
Delta Energy + Delta Cepstrum
Delta-Delta Energy + Delta-Delta Cepstrum

Figure 1. An overview of the standard cepstral coefficient based front end in use in modern speech recognition systems. Absolute feature based on spectral measurements are combined with differential features that measure the stationarity of the spectrum.

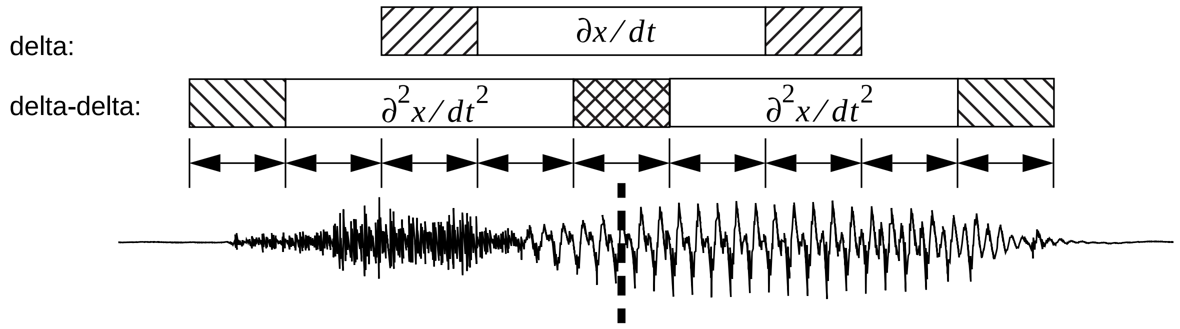


Figure 2. Each temporal derivative is computed using a five frame window (at 10 msec per frame). Hence, the second derivative computation, which requires five frames of first derivative data, involves data extending over nine frames of the input signal.

The first and second derivatives of these 13 features are appended to the features for a total of 39 features. We use a linear regression [28,29] approach to computing the derivatives. A five frame window, as shown in Figure 2, is used for the first derivative computation (two frames into the future and two frame into the past). Similarly, the second derivative uses the same derivative computation on the first derivative features, and hence extends over a region that ultimately includes nine frames of the signal.

To adjust to varying channel and speaker conditions, cepstral mean subtraction [30] is performed on the 12 cepstral features with the mean being computed and subtracted separately for each utterance. Other normalization techniques, such as vocal tract length normalization [31] and

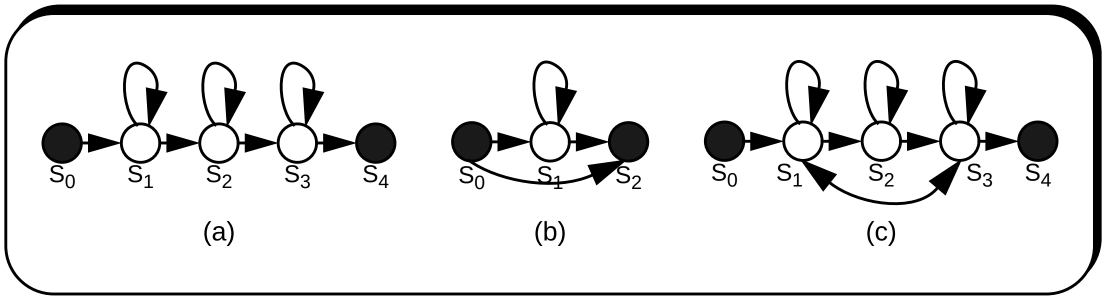


Figure 3. Typical HMM topologies used for acoustic modeling: (a) typical triphone, (b) short pause, (c) silence, and (d) typical word model. The shaded states denote the start and stop states for each model.

variance normalization [32], will not be used in this study even though they are supported in our prototype system. Further, adaptation techniques, such as Maximum Likelihood Linear Regression (MLLR) [33] and Linear Discriminant Analysis (LDA) [34], will not be employed.

Using the feature data, we trained a set of context-dependent cross-word triphone models. Each triphone model was a 3-state left-to-right model with self-loops with the exception of two models as shown in Figure 3. The silence model, *sil*, has a forward and backward skip transition to account for long stretches of silence containing transitory noises. The short, interword silence model, *sp*, contains a forward skip transition that allows it to consume no data when there is no silence between consecutive words. Each state in the models contains a Gaussian mixture model where the number of mixtures is initially set to one and is trained up to sixteen mixtures.

The triphone models were trained using a standard Baum-Welch Expectation Maximization (EM) training algorithm. The specific training schedule we use is summarized in Table 3. Models for all possible triphone contexts are generated using the decision trees produced during the state-tying phase of the training process — one of the distinct advantages of the decision tree based state tying approach. The trained models are then used in conjunction with a bigram language model to perform recognition on the evaluation data.

4. BASELINE SYSTEM EXPERIMENTS

Most HMM-based recognition systems provide a set of parameters that can be used to tune performance for a given application. These parameters include thresholds for state-tying and beam pruning, and scaling factors for the language model and word insertions. The first parameter we tune is the state-tying threshold [21,35]. 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 free parameters in the system. To avoid this problem the ISIP prototype system employs a maximum likelihood phonetic decision tree-based state-tying procedure [35] 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 [35].

Table 4 shows the performance improvement due to varying the number of states after tying.

1. **Flat-start models:** Initialize a set of monophone models to be single mixture Gaussian models. Seed the mean and variance of each Gaussian to be equal to the global mean and variance computed across a small set of the training data. This provides a reasonable starting point for the model optimization. Random initialization would also work but would converge less quickly.
2. **Monophone training:** The monophone models are trained for four iterations of Baum-Welch on the entire training set. In this phase, the *sp* model is not trained and it is assumed that *sil* only occurs at the beginnings and ends of utterances. This gives the *sil* model a chance to learn the parameters of clean silence before we attempt to force it to learn interword silence.
3. ***sp* model training:** The single state of the *sp* model is tied to the central state of the *sil* model. The monophone models are then trained for four more iterations of Baum-Welch on the entire training set. In this phase, it is assumed that the *sp* model occurs between every pair of sequential words while *sil* only occurs at the beginnings and ends of utterances. This allows the *sp* model to learn the parameters of interword silence.
4. **Forced alignment**: The transcriptions are force aligned to the acoustic data and the aligner is allowed to choose the most likely pronunciation for each word in the transcription. New phonetic transcriptions are generated from this forced alignment process and are used throughout the remainder of the training regime.
5. **Final monophone training:** The monophone models are trained against the new phonetic transcriptions for five iterations of Baum-Welch.
6. **Cross-word triphone training:** Cross-word triphone models are seeded from the monophone models. Only triphones seen in the training data are created. Four iterations of Baum-Welch training are used to get initial estimates of the triphone models.
7. **State-tying:** To reduce the parameter count and to provide sufficient training data to undertrained states, we employ a maximum likelihood decision tree-based state tying procedure [35]. Those states that are statistically similar to one another are tied into a single state and the training data previously attributed to each is now shared in the single tied state. The state-tied models are trained for four more iterations of Baum-Welch.
8. **Mixture training:** The single mixture models are successively split up to 16 mixtures with stages at 1, 2, 4, 8 and 16 mixtures. At each stage, four iterations of Baum-Welch reestimation are run on the multi-mixture models.

Table 3. An overview of the training paradigm used in the baseline cross-word context-dependent system.

Normally this parameter has a much more dramatic effect on performance. In this case, the improvements in performance were marginal. The number of tied states found to give best performance was 3,215. The number of initial states was 46,346, which implies that less than 1 out of every 10 states were preserved in the final models. The optimized value of this parameter is very close to what we use for our standard Hub 5E system.

The second parameter we optimized is the language model scaling factor. During the decoding process, the language model probability (as determined by the bigram language model) is computed for each bigram pair. This probability is multiplied by a language model scale factor that weights the relative contribution of the language model to the overall path score. Increasing this scale value tends to cause the language model to dominate the ranking of search paths — essentially boosting the importance of the language model relative to the acoustic model. Decreasing the scale causes the language model to play a lesser role. A word insertion penalty is added to the scaled language model score. This penalty is used to help inhibit the insertion of common, poorly articulated words such as “the”, “a”, and “uh”. Decreasing the value of this parameter will tend to decrease the number of words hypothesized. For the experiments presented in Table 4, the language model scale factor [21] was set to 12 and the word insertion penalty [21] set to -10.

In Table 5, we present results from varying the language model scale factor. A 0.6% absolute reduction in error rate was observed by adjusting this parameter. This is probably a result of the fact that the language model has decent predictive power for the WSJ data (more so than in a conversational speech application), and hence can be relied upon to a greater degree. Tuning the scale factor also reduced xRT by approximately 30%, which is advantageous.

Table 1. The next parameter to be tuned is the word insertion penalty. An interesting bit of folklore in speech research is that optimal performance is almost always achieved when one balances insertions, deletions, and substitutions. In Table 6, we summarize some experiments in

Number of Tied-States	**State-Tying Thresholds**			**xRT**	**WER**	**Sub.**	**Del.**	**Ins.**
	Split	**Merge**	**Occup.**					
1,157	1250	1250	2400	171	9.4%	7.1%	1.6%	0.7%
1,882	650	650	1400	151	11.0%	8.0%	1.7%	1.2%
3,024	150	150	900	149	10.7%	8.0%	1.6%	1.1%
3,215	165	165	840	138	8.6%	6.8%	1.1%	0.7%
3,580	125	125	750	123	8.9%	6.7%	1.4%	0.8%
3,983	110	110	660	120	8.7%	6.6%	1.0%	1.1%
4,330	100	100	600	116	9.1%	6.5%	1.4%	1.2%
5,371	75	75	450	106	9.0%	6.7%	1.0%	1.3%

Table 4. A comparison of experimental results obtained by tuning the number of tied states retained after the state-tying process. All experiments were conducted on the 330 utterance subset of the dev test data using identical conditions (language model scale = 12.0, word insertion penalty = -10 and pruning thresholds set to 300, 250 and 250). The number of initial states was 46,346.

LM Scale	**Word Penalty**	**xRT**	**WER**	**Sub.**	**Del.**	**Ins.**
12	-10	138	8.6%	6.8%	1.1%	0.7%
14	-10	108	8.2%	6.3%	1.2%	0.7%
16	-10	103	8.0%	6.1%	1.4%	0.6%
18	-10	85	8.1%	6.1%	1.5%	0.5%
18	10	85	8.0%	6.2%	0.9%	0.9%
20	10	85	8.0%	6.2%	1.0%	0.9%

Table 5. A comparison of experimental results for tuning the language model scale. 8.0% WER was the best error rate achieved. The first six experiments were run with the number of tied states set to 3,215, the word insertion penalty set to -10, and the pruning thresholds set to 300, 250 and 250. The last two experiments were run with a word insertion penalty of 10, and gave slightly better performance. An LM scale of 18 was chosen because insertions and deletions are balanced in addition to achieving the lowest overall WER.

which we optimized the value of this parameter. Though the best performance on this isolated experiment was obtained with a setting of -10, a word insertion penalty of 10 was selected because it produced near optimal results and balanced insertions and deletions. The fact that this was the optimum point was verified when these results were combined with other settings.

Once these basic parameters were adjusted, we can turn our attention to beam pruning [21], which allows users to trade off search errors and real-time performance. Tight beams result in fast decoding times but less accuracy. Beam pruning s a heuristic technique that removes low scoring hypotheses early in the search process so the computational resources associated with those hypotheses can be used for more promising paths. The decoder allows the user to specify a beam at each level in the search hierarchy (typically state, phoneme, and word-level). A higher beam threshold will allow more paths to be considered during the search process. Using too low of a threshold can result in search errors (i.e. where the correct hypothesis is pruned).

A summary of some basic experiments on the impact of the beam pruning thresholds are shown in Table 7. For these experiments, we trained on the SI84 training set, and evaluated on the 330-utterance Nov’92 dev test set. As can be seen in Table 7, there is a substantial impact on real-time rates by reducing the beam pruning thresholds. For the WSJ task, we find the combination of 300, 250, and 250 gives near-optimal performance at a reasonable real-time rate. Our standard Hub 5E system, and most of our other systems, use these same beam pruning values. Since CPU requirements are an issue for this evaluation due to the large number of experiments

Word Ins. Penalty	xRT	WER	Sub.	Del.	Ins.
-20	98	8.4%	6.3%	1.7%	0.4%
-10	103	8.0%	6.1%	1.4%	0.6%
0	107	8.1%	6.3%	1.0%	0.7%
10	117	8.2%	6.3%	0.9%	1.0%

Table 6. A comparison of experimental results for tuning the word insertion penalty. These experiments were run with the number of tied states set to 3,215, the LM scale factor set to 16, and the pruning thresholds set to 300, 250 and 250. A word insertion penalty of 10 was selected because it produced near optimal results and balanced insertions and deletions (the fact that this was the optimum point was verified when these results were combined with other settings).

Beam Pruning			xRT	WER	Sub.	Del.	Ins.
State	Model	Word					
200	150	150	14	8.6%	6.7%	0.9%	1.1%
300	250	250	85	8.0%	6.2%	0.9%	0.9%
400	350	350	230	8.0%	6.2%	0.9%	0.9%

Table 7. A summary of beam pruning experiments on the SI84 training set and the Nov’92 dev test set. These experiments were run with the number of tied states set to 3,215, the LM scale factor set to 18, and the word insertion penalty set to 10. Beam pruning thresholds of 300, 250 and 250 were chosen because they represent a nice trade-off between performance and real-time rate.

needed to be run, it is important to find ways to reduce computations without significantly impacting performance.

In Table 8, we compare the results of our tuned system to state of the art. Our overall best system, as shown in Table 7, achieves a WER of 8.0% on the dev test set, and 8.3% on the evaluation set. The best published results for comparable technology, highlighted in Table 8, are in the range of 6.8% WER. By directly tuning our system on the evaluation set, we have achieved error rates of 7.7%. However, tuning on the evaluation set is not a reasonable thing to do.

We believe that the primary difference that accounts for the discrepancy in the error rates is the lexicon used by the respective systems. When WSJ research was at its peak, most sites were using proprietary lexicons that had been tuned to optimize performance, normally by implementing some basic form of pronunciation modeling. We are in the process of contacting several of these sites (including one we can obtain from LDC) to obtain their lexicons so that we can better understand the difference in performance. We do not, however, believe that the lexicon is solely responsible for this large difference (18% relative). Diagnosing the reasons there is a performance gap will take more time, but should not hold up the evaluation efforts. The difference in performance is a fairly consistent bias that should not mask algorithm differences in the front end processing. Possible reasons for this gap include a difference in the results of the state-tying process, and issues in silence/noise modeling. We have not seen such a large difference for other tasks we have run (Resource Management and OGI Alphadigits).

5. COMPUTATIONAL CONSIDERATIONS

As stated previously, the SI84 training set consists of 7,138 utterances totaling 14 hours of speech data. The Baum-Welch HMM trainer in the prototype system is very fast — it runs approximately

Site	Acoustic Model Type	Language Model	Adaptation	WER
ISIP	xwrd/gi	bigram	none	8.3%
CU [36]	wint/gi	bigram	none	8.1%
UT [37]	wint/gd	bigram	none	7.1%
CU [36]	xwrd/gi	bigram	none	6.9%
LT [38]	xwrd/gi	bigram	none	6.8%
CU [36]	xwrd/gd	bigram	none	6.6%
UT [39]	xwrd/gd	bigram	none	6.4%
UT [39]	xwrd/gd	bigram	VTLN	6.2%
LT [40]	xwrd/gi	trigram	none	5.0%
LT [40]	xwrd/gd	trigram	none	4.8%
LT [40]	xwrd/gd/tag	trigram	none	4.4%

Table 8. A comparison of performance reported in the literature on the WSJ0 SI84/Nov’92 evaluation task. In order to keep the overall system complexity down, a cross-word, gender-independent bigram system with no adaptation was selected for the baseline system. The ISIP system achieves an error rate of 8.3% on the 330 utterance evaluation set, which is about 1.4% higher than other published results.

0.15 xRT on an 800 MHz processor. Nevertheless, because a typical training session involves 38 passes over the data, the overall training time is still considerable: 288 hours for a single 800 MHz Pentium III processor when you consider a typical run in a moderately loaded computer network. This means training must be run in parallel to be practical. We typically distribute training across 12 to 14 processors. Training uses minimal amounts of memory (less than 50M), so it can be run on less powerful machines. We normally run training in the background on our desktops, which have 800 MHz processors but only 0.5G of memory.

Fortunately, the Baum-Welch training algorithm is quite amenable to coarse-grain parallelism. The training file list is split across N machines in a way that equally balances the load. Each machine processes its list of files, and returns the accumulated results to a central location, where these results are merged to produce the final reestimated models. There is one complicating factor, however. Fourteen machines running at 0.15 xRT are moving the equivalent of 39 features/frame x 8 bytes/feature x 100 frames/sec x 14 processors x (1/0.15 xRT) = 2 Mbytes/sec of data. Even on a mildly congested network, processing such a large amount of data from a centralized location can tax the limits of your local computer network or file server. Hence, we routinely deploy local high-speed disks on our Pentium-class servers. Seagate Cheetah SCSI drives, which spin at 10,000 and 15,0000 rpm, and are available in a range of sizes (we use 9G and 18G drives). These drives are quite cost-effective for these tasks.

We also employ a bank of compute servers to perform computationally intensive tasks. Details of the configuration can be found at [42]. A typical compute server[1] consists of an 800 MHz dual processor Pentium III with 1 Gbyte of memory, 100 Mbits/s ethernet, and an 18 Gbyte local disk. We use the Sun Solaris operating system for x86 machines because we have found this environment to be very stable and productive (long before Linux became stable enough for speech recognition research). With this OS we are able to get very close to 100% utilization of each processor in a dual-processor configuration, making this architecture extremely cost-effective. We can run two training jobs (one per processor) on such a machine with no loss in performance.

The Nov’92 evaluation set consists of 40 minutes of speech data. The decoder runs 85 xRT with the beam pruning settings described previously, and therefore requires 57 hours of CPU time. We typically run decoding on 11 processors so that the entire job runs in 5.2 hours. Decoding requires between 300 Mbytes and 650 Mbytes of RAM depending on the length of the utterance (and the pruning thresholds previously described). This means a dual processor machine with 1 Gbyte of memory can only run one decoding job at a time. Fortunately, I/O bandwidth is not an issue with decoding since a great deal of time is spent processing each utterance. Hence, the data files can reside anywhere in the network and not hinder processing time. We normally partition our servers into groups and use one group for training, and another for evaluations, so that we minimize competition for the resources. We do not currently support any automatic job distribution software with our recognition software. We prefer users to distribute the jobs manually. For those interested in running these jobs in parallel, we will distribute example scripts shortly. These tend to be less portable due to many OS-dependent details required to handle files and directories.

1. As of 08/01, we are in the process of purchasing a bank for servers with dual 1.4 GHz AMD Athlon processors, a Tyan motherboard with on-board SCSI and ethernet, a 36G Cheetah disk, and 2 Gbytes of memory. These machines cost about $2,400 USD, which is quite a bargain.

6. CONCLUSIONS

We have presented a baseline WSJ system that provides near state-of-the-art performance yet minimized complexity and maximizes ease of use. We have incorporated support for HTK binary-formatted features, and added an ability to compute the derivatives of the features on-the-fly. We have also released a sample package that allows one to run decoding with the baseline system described above, and to easily verify the installation. Note that all information related to the project, including this report, can be found at the project web site [41]: *http://www.isip.msstate.edu/projects/aurora*.

We are now in the process of developing the final models for the evaluation by conducting experiments on the designated data set partitions provided by the Aurora Working Group [6]. Once these experiments have been completed, an analysis will be performed on the impact of multi-condition training and the degradations due to noise. We expect these results to be available by the end of August 2001.

7. ACKNOWLEDGMENTS

We wish to acknowledge David Pearce of Motorola Labs, Motorola Ltd., United Kingdom, and Guenter Hirsch of Niederrhein University of Applied Sciences for their invaluable collaborations and direction on this project.

8. REFERENCES

[1] D. Paul and J. Baker, “The Design of Wall Street Journal-based CSR Corpus,” *Proceedings of the International Conference on Spoken Language Systems (ICSLP)*, pp. 899-902, Banff, Alberta, Canada, October 1992.

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

[3] D. Pearce, “Enabling New Speech Driven Services for Mobile Devices: An overview of the ETSI standards activities for Distributed Speech Recognition Front-ends,” presented at the Applied Voice Input/Output Society Conference (AVIOS2000), San Jose, California, USA, May 2000.

[4] “Recommendation G.712 — Transmission performance characteristics of pulse code modulation channels,” International Telecommunication Union (ITU), Geneva, Switzerland, November 1996.

[5] “Recommendation P.341 — Transmission characteristics for wideband (150-7000 Hz) digital hands-free telephony terminals, International Telecommunication Union (ITU), Geneva, Switzerland, February 1998.

[6] G. Hirsch, “Experimental Framework for the Performance Evaluation of Speech

Recognition Front-ends on a Large Vocabulary Task," STQ Aurora DSR Working Group, June 2001.

[7] D. Paul and B. Necioglu, "The Lincoln Large-Vocabulary Stack-Decoder HMM CSR," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, Minnesota, USA, pp. 660-663, 1993.

[8] D. Pallett, J.G. Fiscus, W.M. Fisher, and J.S. Garofolo, "Benchmark Tests for the DARPA Spoken Language Program," *Proceedings from the Human Language Technology Conference*, Merrill Lynch Conference Center, Princeton, New Jersey, USA, March 1993.

[9] "The CMU Pronouncing Dictionary," *http://www.speech.cs.cmu.edu/cgi-bin/cmudict*, Speech at Carnegie Mellon University, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 2001.

[10] A. Stolcke, "The SRI Language Modeling Toolkit," *http://www.speech.sri.com/projects/srilm/*, Speech Technology and Research Laboratory, SRI International, Menlo Park, California, USA, July 2001.

[11] 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.

[12] R. Sundaram, A. Ganapathiraju, J. Hamaker and J. Picone, "ISIP 2000 Conversational Speech Evaluation System," presented at the Speech Transcription Workshop, College Park, Maryland, USA, May 2000.

[13] R. Sundaram, J. Hamaker, and J. Picone, "TWISTER: The ISIP 2001 Conversational Speech Evaluation System," presented at the Speech Transcription Workshop, Linthicum Heights, Maryland, USA, May 2001.

[14] B. George, B. Necioglu, J. Picone, G. Shuttic, and R. Sundaram, "The 2000 NRL Evaluation for Recognition of Speech in Noisy Environments," presented at the SPINE Workshop, Naval Research Laboratory, Alexandria, Virginia, USA, October, 2000.

[15] J. Picone, "ISIP Foundation Classes," *http://www.isip.msstate.edu/projects/speech/software/asr/download/ifc/index.html*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, July 2001.

[16] L.R. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.

[17] H. Murveit, P. Monaco, V. Digalakis and J. Butzberger, "Techniques to Achieve an Accurate Real-Time Large-Vocabulary Speech Recognition System," *Proceedings of the ARPA Human Language Technology Workshop*, pp. 368-373, Austin, Texas, USA,

March 1995.

[18] F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Cambridge, Massachusetts, USA, 1997.

[19] L. Lamel and G. Adda, “On Designing Pronunciation Lexicons for Large Vocabulary Continuous Speech Recognition,” *Proceedings of the International Conference on Speech and Language Processing*, pp. 6-9, Philadelphia, Pennsylvania, USA, October 1996.

[20] H. Ney and S. Ortmanns, “Dynamic Programming Search for Continuous Speech Recognition,” *IEEE Signal Processing Magazine*, vol. 1, no. 5, September 1999.

[21] N. Deshmukh, A. Ganapathiraju and J. Picone, “Hierarchical Search for Large Vocabulary Conversational Speech Recognition,” *IEEE Signal Processing Magazine*, vol. 1, no. 5, pp. 84-107, September 1999.

[22] J. Picone, “Internet-Accessible Speech Recognition Technology,” *http://www.isip.msstate.edu/projects/speech*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, June 2001.

[23] J. Picone, “Fundamentals of Speech Recognition,” *http://www.isip.msstate.edu/publications/courses/isip_0000*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, June 2001.

[24] A. Ganapathiraju, “ISIP LVCSR System Tutorial,” *http://www.isip.msstate.edu/projects/speech/education/tutorials/asr_alphadigits/current/index.html*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, June 2001.

[25] J. Picone, “Speech Recognition System Training Workshop”, *http://www.isip.msstate.edu/conferences/srstw01/index.html*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, May 2001.

[26] J. Picone, “Speech Recognition System Design Review”, *http://www.isip.msstate.edu/conferences/srsdr02/index.html*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, January 2002.

[27] J. Picone, “Signal Modeling Techniques in Speech Recognition”, *IEEE Proceedings*, vol. 81, no. 9, pp. 1215-1247, September 1993.

[28] F.K. Soong and A.E. Rosenberg, “On the Use of Instantaneous and Transitional Spectral Information in Speaker Recognition,” *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Tokyo, Japan, pp. 877-880, April 1986.

[29] J. Picone, “Adding Temporal Information: Derivatives,” *http://www.isip.msstate.edu/conferences/srstw01/program/session_04/signal_processing/html/sp_15.html*, Institute for

Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, May 2001.

[30] B.A. Hanson, T.H. Applebaum, J.C. Junqua, “Spectral Dynamics for Speech Recognition Under Adverse Conditions,” in *Advanced Topics in Automatic Speech and Speaker Recognition*, C.-H. Lee, K.K. Paliwal and F.K. Soong, Eds., Kluwer Academic Publishers, New York, New York, USA, 1995.

[31] T. Kamm, G. Andreou and J. Cohen, “Vocal Tract Normalization in Speech Recognition Compensating for Systematic Speaker Variability,” *Proceedings of the 15th Annual Speech Research Symposium*, Johns Hopkins University, Baltimore, Maryland, USA, pp. 175-178, June 1995.

[32] T. Hain, P.C. Woodland, T.R. Niesler, and E.W.D. Whittaker, “The 1998 HTK System for Transcription of Conversational Telephone Speech,” *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, Arizona, USA, pp. 57-60, March 1999.

[33] Y. Hao and D. Fang, “Speech Recognition Using Speaker Adaptation by System Parameter Transformation,” *IEEE Transactions on Speech and Audio Processing*, vol 2, no. 1, part 1, pp. 63-68, January 1994.

[34] R. Haeb-Umbach, H. Ney. “Linear Discriminant Analysis for Improved Large Vocabulary Continuous Speech Recognition,” *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, California, USA, vol. 1, pp. 13-16, March 1992.

[35] S. J. Young, J. J. Odell, and P. C. Woodland, “Tree-based State Tying For High Accuracy Acoustic Modelling, *Proceedings of the ARPA Workshop on Human Language Technology*, Plainsboro, New Jersey, USA, pp. 286-291, March 1994.

[36] P. C. Woodland, J. J. Odell, V. Valtchev, and S. J. Young, “Large Vocabulary Continuous Speech Recognition using HTK,” *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Adelaide, Australia, pp. II/125-II/128, April 1994.

[37] K. Beulen, and H. Ney, “Automatic Question Generation for Decision Tree Based State Tying,” *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 805-808, Seattle, Washington, USA, April 1998.

[38] W. Reichl, and W. Chou, “Decision tree state tying based on segmental clustering for acoustic modeling,” *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp 801-804, Seattle, WA, USA, April 1998.

[39] L. Welling, S. Kanthak, and H. Ney, “Improved Methods for Vocal Tract Normalization,” *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal*

Processing, pp. 761-764, Phoenix, Arizona, USA, March, 1999.

[40] W. Reichl, and W. Chiao, “Unified Approach of Incorporating General Features in Decision Tree Based Acoustic Modeling,” *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 573-576, Phoenix, Arizona, USA, March 1999.

[41] N. Parihar and J. Picone, “The Aurora Evaluations,” *http://www.isip.msstate.edu/projects/aurora*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, July 2001.

[42] J. Sesser and J. Picone, “Computer Resources,” *http://www.isip.msstate.edu/about_us/resources/technical/computer/index.html*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, July 2001.