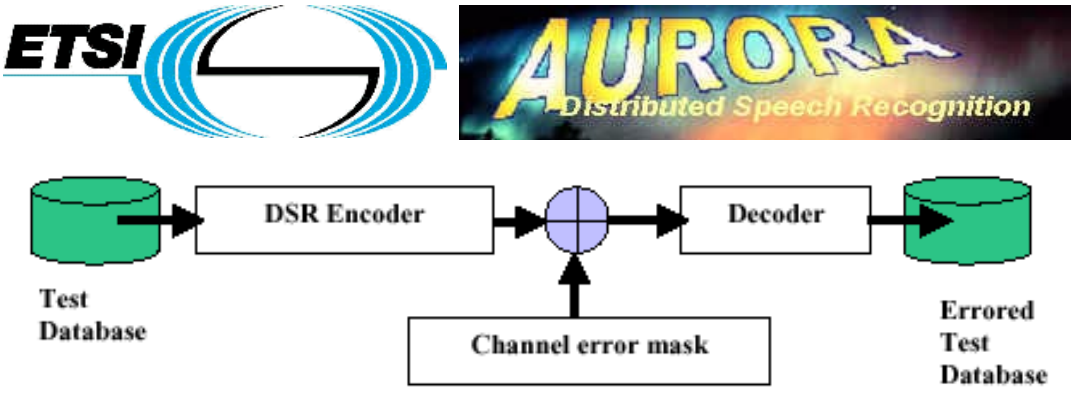
Aurora Working Group:
DSR Front End LVCSR Evaluation
AU/384/02

submitted to:

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

December 06, 2002



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 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. The 5000-word closed-loop vocabulary task based on the DARPA Wall Street Journal (WSJ0) Corpus was chosen for these evaluations. The experiments were designed to test the following focus conditions:

1. **Additive Noise**: six noise conditions collected from street traffic, train stations, cars, babble, restaurants and airports were 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 was calibrated.
3. **Microphone Variation**: performance for two microphone conditions (Sennheiser and Second Microphone) contained in the WSJ0 corpus was analyzed.
4. **Compression**: degradations due to data compression of the feature vectors was evaluated.
5. **Model Mismatch**: the degradation due to a mismatch between training and evaluation conditions was calibrated.
6. **Utterance Detection**: performance improvement due to end pointing the WSJ0 corpus.

The first step in this project was to design a baseline system that provides 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 330 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 involved 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 was expected to be consistent with our previous results. We evaluated a total of 154 conditions that involved 7 noise types, 2 compression types, 3 training sets, 2 sample rates, 2 compression types and utterance detection. The results from these experiments are summarized in this report.

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 well under various noise conditions. Further, since many of these portable devices use lossy compression to conserve bandwidth, system performance must also not degrade when subjected to compression, packet loss, and other common wireless communication system artifacts. Speech compression, for example, is known to have a negative effect on the accuracy of speech recognition systems [2,3].

Aurora, a working group of ETSI, has been formed to address many of the issues involved in using speech recognition in mobile environments [4]. 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) were 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) were randomly chosen. For each of the six test conditions, the SNR was randomly chosen between 5 and 15 dB in steps of 1 dB. G.712 filtering [5] was used to simulate the frequency characteristics at an 8 kHz sample frequency. P.341 [6] was used to simulate frequency characteristics at a 16 kHz sample frequency. Filtering was applied to both noisy and clean data.
- **Sample Frequency Reduction:** Two sampling rates, 16kHz and 8kHz, were 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-414 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 [4] which compresses features to 4800 b/s was 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 was performed. One set of models was trained on only the Sennheiser microphone data with no additive noise (often referred to as clean data). A second set of models was trained on a combination of clean and noisy data using both microphone conditions. Both models were evaluated on the same test conditions that contain a broad range of noise conditions.

- **Utterance Detection:** Endpoint detection is one of the most important aspects of speech recognition that can vary the word error rate significantly. The WSJ training and evaluation utterances were exercised to study the effect of end pointing on the overall WER on the various noisy conditions.

Our most important goal was to develop a baseline recognition system to conduct these evaluations. We defined these milestones in accordance with this goal:

- **Develop a baseline WSJ0 system:** The baseline system should be fairly simple yet deliver near state-of-the-art performance. It would use a bigram language model and a 5000-word closed-loop vocabulary (no out of vocabulary words in the evaluation set). This system would initially use the ISIP front end with standard mel cepstral features. It would be trained using a mixture of clean and noisy data described in [7] and evaluated on clean data. This baseline system provided a comparison point to insure that future results were credible.
- **Provide an HTK feature file interface:** We would develop an enhanced version of the ISIP prototype system, which would be identified as v5.10, that would provide direct support for reading and writing of HTK-formatted binary features. This version also would be capable of computing the first and second derivatives of features on the fly, rather than requiring these to be stored in a feature file.
- **Generate a short set:** We would generate a short list of utterances which could be used to run small-scale pilot experiments and achieve rapid turnaround. Short sets can often be misleading but were required for this evaluation since many participants had limited computing resources.
- **Integrate and evaluate the ETSI front end:** The final baseline system would use an implementation of the industry-standard mel frequency-scaled cepstral coefficient front end (DSR-MFCC) developed in previous Aurora standards activities [4]. This system would be trained in a manner identical to the MFCC baseline system described above. The system would be evaluated on 14 test conditions (6 noise conditions plus clean speech and using each of the two microphone conditions).
- **Develop parallel processing software:** Scripts that automatically partition jobs across multiple CPUs would be developed. Users would be able to set appropriate parameters from the command line, as well as provide a list of CPUs to be used.
- **Evaluate the effects of utterance detection:** Conduct experiments on speech data for which the leading and trailing silence has been removed.
- **Evaluate performance at 8 kHz:** The experiments described above would be repeated on the same data resampled at 8 kHz. Downsampling would be performed using the procedure described in [7].
- **Evaluate the impact of compression:** Repeat these experiments using the DSR front end compression algorithm [4].

Obviously, there were a large number of conditions that were evaluated, and processing time was a critical issue. In the following sections we describe the design of these experiments and the corresponding results of these evaluations. We also analyze these results later. The baseline system with ETSI front-end and the data sets employed to the generate the performance baselines for the Aurora evaluations have been summarized in the APPENDIX A for the convenience of the readers wishing a comprehensive and a brief overview of the final system. The APPENDIX B has been added that explains the theory and basics behind the statistical significance testing used for this project.

2. AN OVERVIEW OF THE EVALUATION CORPUS

The first step in evaluating front ends for telecommunications applications was 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 [7]. WSJ is a large vocabulary continuous speech recognition corpus consisting of high-quality recordings of read speech. Two-channel recordings of the same utterances 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) [8] is also distributed with the corpus as a reference language model.

In our experiments for the baseline system with ISIP front end, we used 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 used two data sets: a 330 subset (described below) of the 1206 utterance dev test set, and the complete November 92 NIST evaluation set [9]. 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 needed 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 (v0.6) [10] 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 goal was to conduct all experiments with no OOVs, 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) [11]. 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.

For the Aurora Evaluations for the ETSI front end, we were provided with processed versions [7] of the training and evaluation utterances for both the filtered and additive noise conditions. These were generated at 16 kHz and 8 kHz. G.712 filtering [5] was used to simulate the frequency characteristics at an 8 kHz sample frequency and P.341 filtering [6] was used for simulation at 16 kHz. The filtering was applied to the noisy data as well. As shown in Figure 1, Training Set 1 consists of the filtered version of the complete SI-84 training set (7138 utterances) recorded with the Sennheiser microphone.

Training Set 2 was used to study the effects of variation in microphone and noise. Its data distribution is shown in Figure 2. The filtered 7138 training utterances are divided into two blocks: 3569 utterances (half) recorded with the Sennheiser microphone, and the remaining half recorded with a different microphone (18 different microphone types were used). No noise is added to one-fourth (893 utterances) of each of these subsets. To the remaining three-fourths (2676 utterances) of each of these subsets, 6 different noise types (car, babble, restaurant, street, airport, and train) were added at randomly selected SNRs between 10 and 20 dB. The goal was an equal distribution of noise types and SNRs. Thus, we had one clean set (893 utterances) and 6 noisy subsets (446 utterances each) for both the microphone conditions.

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.

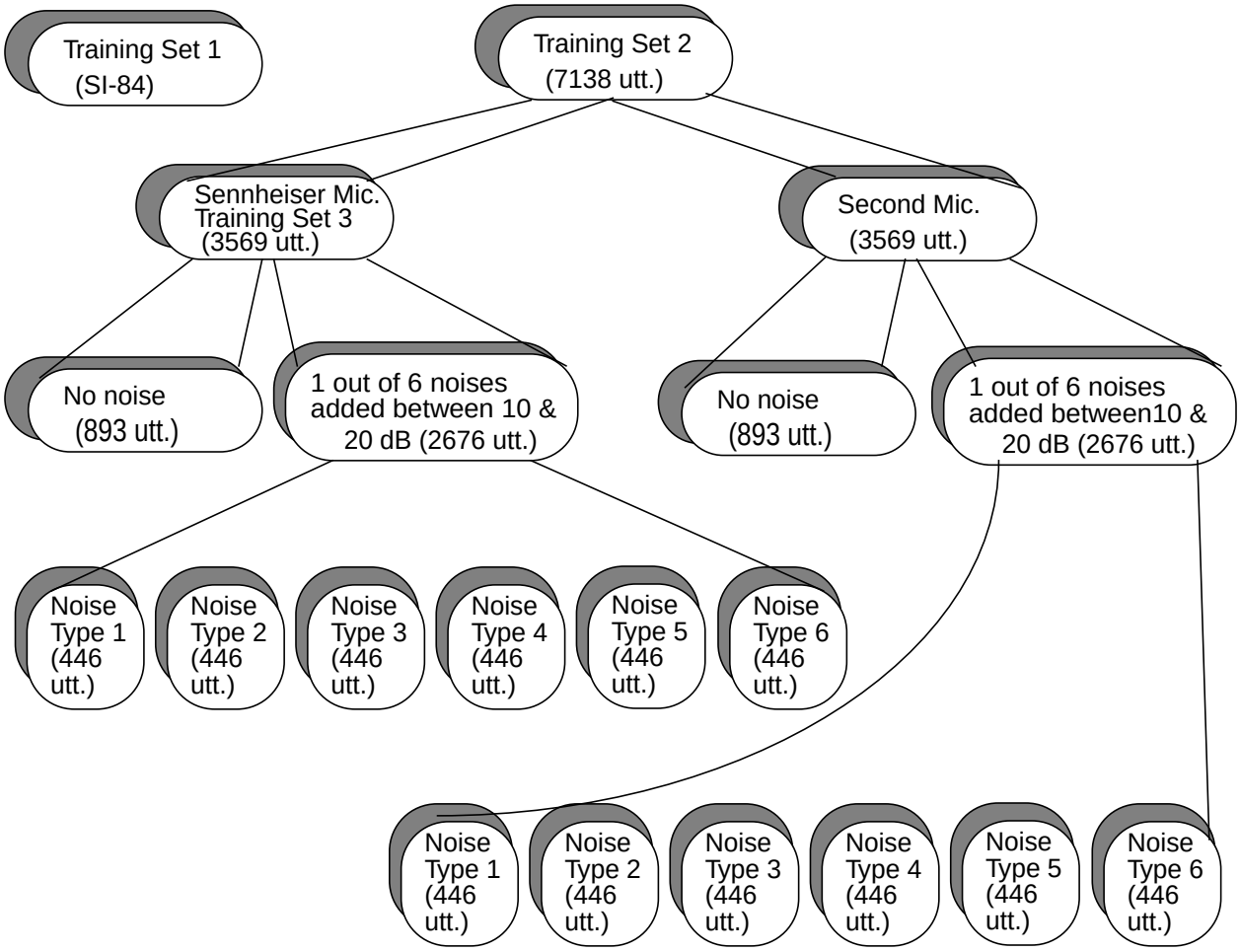


Figure 1. Definition of Training Set 1 (Clean Training) and Training Set 2 (Multi-condition Training).

There is one irregularity in Training Set 2. The speech file 408o303.wv1, recorded with the Sennheiser microphone exists, but the file 408o303.wv2, recorded with a second microphone, does not exist on the original WSJ0 CDs. To keep the number of files constant across both the training sets (1 and 2), the file 408o302.wv2 has been selected instead of 408o303.wv2. Thus both the files 408o302.wv1 and 408o302.wv2 are used in Training Set 2.

Training Set 3 was defined to study the impact of using utterances recorded only on Sennheiser microphone for training. The Sennheiser microphone block of the Training Set 2 was called as the Training Set 3.

Fourteen evaluation sets were defined in order to study the degradations in speech recognition performance due to microphone conditions, filtering and noisy environments. Each of the filtered versions of the evaluation set recorded with Sennheiser microphone and secondary microphone were selected to form the two eval sets. The remaining 12 subsets were defined by randomly adding each of the 6 noise types at randomly chosen SNR between 5 and 15 dB for each of the microphone types as shown in Figure 2. The goal was to have an equal distribution of each of the 6 noise types and the SNR with an average SNR of 10 dB.

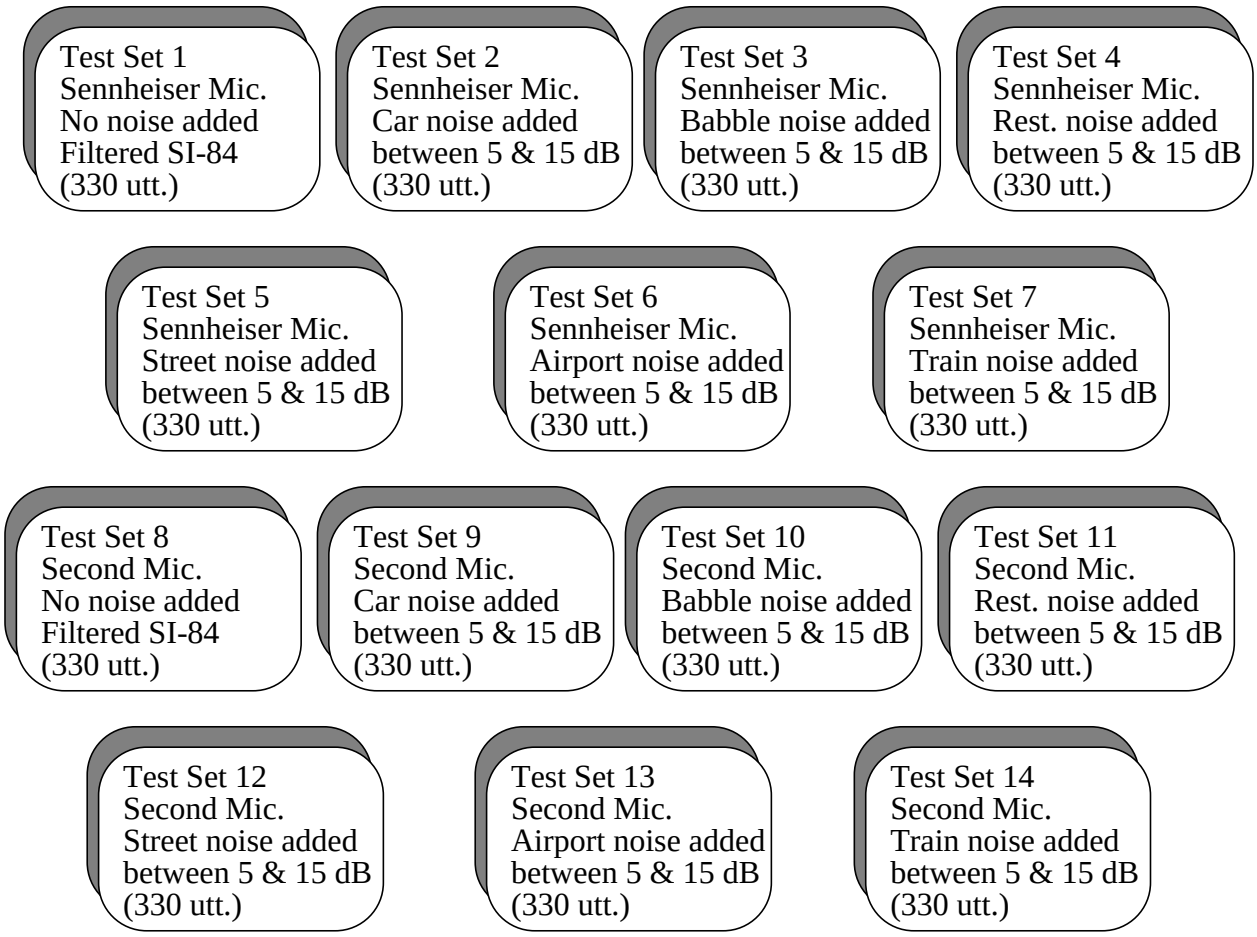


Figure 2. Definitions of 14 Testing Sets that include 6 noise types and different mic types.

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 [12] since it was the first recognition system we built and served as a testbed to develop ideas for implementing conversational speech recognition 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 [13,14] and the Naval Research Laboratory [15]. The prototype system is predecessor of a more grandiose attempt to build a modular and flexible recognition system known as the production system [16]. 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 in early 2002.

Both of these system share the same general computational and algorithmic frameworks. We use hidden Markov model based context-dependent acoustic models [17], lexical trees [18] for cross-word acoustic modeling, N-gram language models with backoff probabilities [19] for language modeling (finite state networks are also supported), and a tree-based lexicon for

pronunciation modeling [20]. The core of the system is a hierarchical dynamic programming-based time synchronous network search engine [21,22] 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 [23] along with several short courses and training materials [24,25]. Annual workshops [26,27] 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 the Signal Modeling paper [28]. The most popular front end in use today employs mel frequency-scaled cepstral coefficients. This front end is summarized on Figure 3. 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).

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 [29,30] approach to computing the derivatives. A five-frame window, as shown in Figure 4, 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.

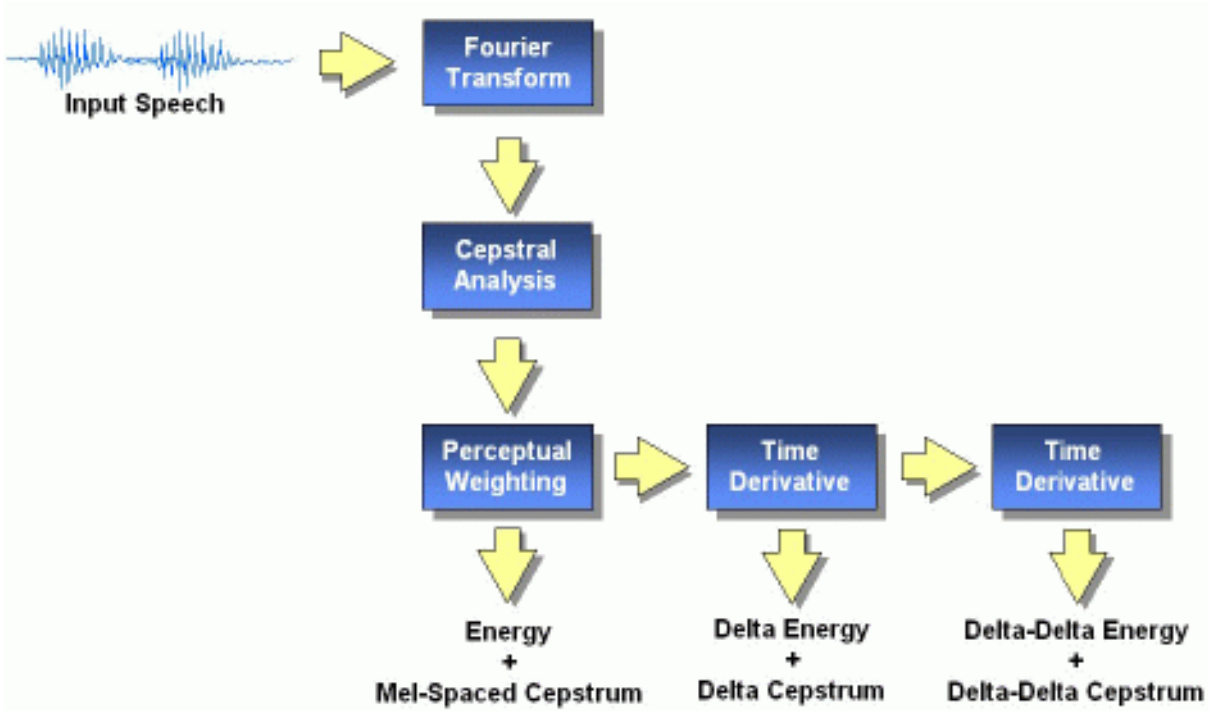


Figure 3. An overview of the standard cepstral coefficient based front end in use in modern speech recognition systems. Absolute feature based on spectral measurements are combined with differential features that measure the stationarity of the spectrum.

To adjust to varying channel and speaker conditions, cepstral mean subtraction [31] was 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 [32] and variance normalization [33], were not used in this study even though they are supported in our prototype system. Further, adaptation techniques, such as Maximum Likelihood Linear Regression (MLLR) [34] and Linear Discriminant Analysis (LDA) [35], were 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 5. 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. A typical training schedule we use is summarized in Table 3. However, for baseline system, we changed the force alignment step. Instead of force-aligning the word transcription on monophone models to get the monophone transcriptions, we produced a cross-word triphone transcriptions by aligning the word-transcription to the cross-word triphone models that were previously generated from the best performing system tuned on the Eval set. These aligned triphone transcriptions were then converted to monophone transcriptions by

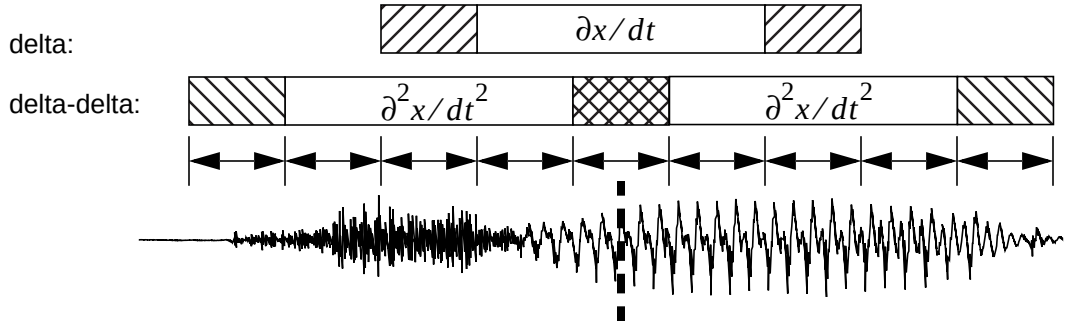


Figure 4. Each temporal derivative is computed using a five frame window (at 10 msec per frame). Hence, the second derivative computation, which requires five frames of first derivative data, involves data extending over nine frames of the input signal.

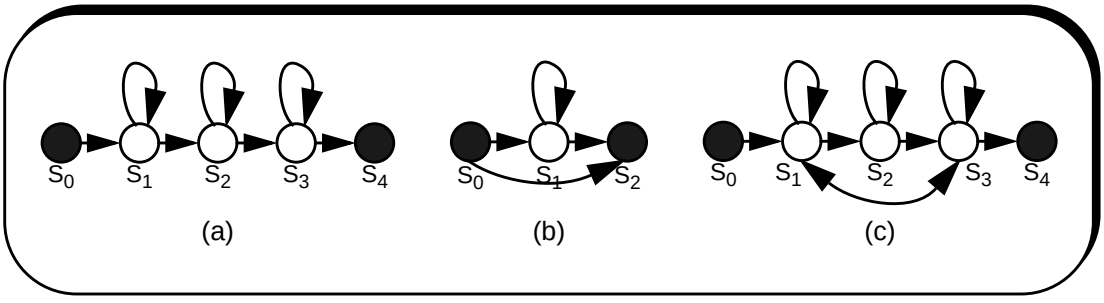


Figure 5. Typical HMM topologies used for acoustic modeling: (a) typical triphone, (b) short pause, and (c) silence. The shaded states denote the start and stop states for each model.

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 [36]. 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.

removing the left and the right context for each central phone. Models for all possible triphone contexts were 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 were 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 [22,36]. 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 [36] 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 [36].

All the experiments for baseline system were conducted on SI-84 training set and Nov'92 eval set. Table 4 shows the performance improvement due to varying the number of states after tying. 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 one 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 [22] was set to 12 and the word insertion penalty [22] 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

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 before state tying.

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.

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 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 [22], which allows users to trade off search errors and real-time performance. Tight beams result in fast decoding times but less accuracy. Beam pruning is 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

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.

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).

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 SI-84 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 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.

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 SI-84 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.

Site	Acoustic Model Type	Language Model	Adaptation	WER
ISIP	xwrd/gi	bigram	none	8.3%
CU [37]	wint/gi	bigram	none	8.1%
UT [38]	wint/gd	bigram	none	7.1%
CU [37]	xwrd/gi	bigram	none	6.9%
LT [39]	xwrd/gi	bigram	none	6.8%
CU [37]	xwrd/gd	bigram	none	6.6%
UT [40]	xwrd/gd	bigram	none	6.4%
UT [40]	xwrd/gd	bigram	VTLN	6.2%
LT [41]	xwrd/gi	trigram	none	5.0%
LT [41]	xwrd/gd	trigram	none	4.8%
LT [41]	xwrd/gd/tag	trigram	none	4.4%

Table 8. A comparison of performance reported in the literature on the WSJ0 SI-84/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.

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 since we need to conduct additional experiments. 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 with state-of-the-art systems for other tasks we have run (Resource Management and OGI Alphadigits).

5. EXTENSIONS TO THE BASELINE SYSTEM

The recognition system used for these evaluations was modified to make it compatible with the ETSI front end that generates HTK formatted features. We also added to this system the capability to compute first and second derivatives of the features on the fly during training and decoding. This was necessary since the ETSI front end was not capable of computing derivatives. This modified system was formally released as v5.11 [42] of the prototype system.

5.1. Single-CPU Scripts

Our standard Tidigits and Alphadigits Tutorials are a comprehensive examination of the features of the ISIP prototype ASR system. These allows one to experiment with Baum-Welch and Viterbi training of word, monophone, word-internal triphone and cross-word triphones on a single processor. However, a significant limitation of these tutorials is their inability to change various training and testing parameters from the command line. These scripts produce use force-alignment to get the phonetic alignments during the monophone stage of the training stage. For Aurora Evaluations we used a pre-aligned phonetic transcriptions that doesn’t need the force-alignment step.

With an objective that a novice speech researcher should be able to run Aurora experiments easily, we developed a software interface that would enable a user to run a complete experiment using a simple command, and to allow a user to change significant parameters from the command line. A sample command line is shown in Figure 6. The various command line options that are supported are shown in Figure 7.

To get the monophone phonetic transcriptions, we took our best 16-mixture cross-word triphone models and the word-level transcriptions, and then ran forced-alignment. The aligned cross-word triphone transcriptions obtained from this force-alignment step were then converted to

TRAINING AND TESTING

wsj_run -num_features 13 -mode acc -feature_format htk -mixtures 1
-train_mfc_list ./train_1792_v1.4.multi.list -cpus_train isip216 isip210 isip210 isip211
-feature_format htk -lm devel -test_state_beam_pruning 200
-test_model_beam_pruning 150 -test_word_beam_pruning 150
-test_mfc_list ./devtest_0030_v1.4.list -models_path ../exp_085
-cpus_test isip210 isip211 isip212 isip213 isip214 isip215 isip216

TRAINING

wsj_run -num_features 13 -mode acc -feature_format htk -mixtures 1
-train_mfc_list ./train_1792_v1.4.multi.list -cpus_train isip216 isip210 isip210 isip211

TESTING

wsj_run -num_features 13 -mode acc -feature_format htk -lm devel
-test_state_beam_pruning 200 -test_model_beam_pruning 150
-test_word_beam_pruning 150 -test_mfc_list ./devtest_0030_v1.4.list
-models_path ../exp_085 -cpus_test isip210 isip211 isip212 isip213 isip214 isip215 isip216

Figure 6. Typical command lines for training and evaluation using the multiple CPU WSJ script.

monophone transcriptions by retaining the central phone of the triphone unit (deleting the left and right phones). These monophone phonetic transcriptions typically produce a better alignment than the transcriptions obtained from alignments that use monophone model. The monophone phonetic transcriptions that resulted from this process were bundled with this software and used throughout the project to reduce the complexity of the training process.

This single-CPU version [42] runs a complete experiment, starting from model initialization, continuing through training, and concluding with recognition and scoring. These scripts were written in PERL, a popular language for such things. All the required training parameters, such as beam_pruning, state-tying [36], word penalties, etc., can be specified from the command line. All the command line parameters are handled by a special parsing script (*command_line.pm.in*). The subroutines used for various processes for speech recognition such as *init_triphones, run_bw_train, run_trace_projector, run_tie_state_train,* etc. are contained in subroutine files (*wsj_subs.pm.in*).

Inter-process communication is performed using the standard Unix protocol remote shell (*rsh*). The master script, *wsj_run.pl.in*, reads the command line, parses it, and executes the appropriate subtask (e.g., the training script *wsj_train.pl.in*). All training steps except the forced-alignment procedure are contained in the training script *wsj_train.pl.in*. The state-tying utility is run in test mode and decoding is run by the decoding script *wsj_decode.pl.in*.

-num_features <number>: number of features per frame (default is 39)

-mode <mode>: none, delta (compute deltas on the fly), and acc (compute deltas and accelerations on the fly). The default is none.

-feature_format <value>: htk (HTK formatted features), isip_proto (prototype system format).

-mixtures <number>: number of Gaussian mixtures components per state.

-split_threshold <value>: the minimum increase in likelihood to split a node during the state tying process (default is 165).

-merge_threshold <value>: the maximum decrease in likelihood to merge two nodes during the state tying process (default is 165).

-num_occ_threshold <value>: the minimum number of states for splitting a node during the state tying process (default is 840)

-lm <value>: devel (WSJ development test set), eval (evaluation test set).

-lm_scale <value>: the language model weight (default is 18).

-word_penalty <value>: the penalty applied to each word that is hypothesized (default is 10).

-train_beam_pruning <value>: the beam pruning threshold for training (default is 1000).

-test_state_beam_pruning <value>: the state-level beam pruning for testing (default is 300).

-test_model_beam_pruning <value>: the model-level beam pruning for testing (default is 250).

-test_word_beam_pruning <value>: the word-level beam pruning for testing (default is 250).

-train_mfc_list <list>: the list of files for training.

-test_mfc_list <list>: the list of file for testing.

Figure 7. An explanation of the command line arguments for the single-CPU scripts.

5.2. Multiple-CPU Scripts

The baseline system requires about 10 days for training and 2 days for decoding on an 800 MHz Pentium processor. Running 126 test conditions and 7 training conditions on a single processor is virtually impossible since it would require about 94 days for each training condition. Therefore, it is necessary to run these tasks in parallel on multiple processors to meet the aggressive time lines of this evaluation.

We therefore proceeded to develop a multiple-CPU version [42] of the single-CPU scripts. Parallel processing methodologies are often site-dependent and tool-dependent, creating a support nightmare. Our strategy in developing the multiple CPU scripts was to make the implementation highly portable and simple to use. We leveraged our many years of experience in this area to make these scripts as useful as possible.

The interfaces for these scripts are shown in Figure 6. The scripts allow training and decoding to be run separately, which is useful when evaluating models across a wide range of noise conditions. Alternately, the entire process can be run from a single command line. The training and testing CPU's can be specified from the command line using the options "*-cpus_train*" and "*-cpus_test*." The scripts split the files to be processed across these machines, and handle all inter-process communications using standard Unix protocols such as rsh. There is one big

assumption in our approach — **all data used by these scripts must be available from the same mount point on all processors**. File lists should use full pathnames such as /isip/data/aurora/foo.raw, and this file must be visible from all machines using that path.

The first processor specified in the list takes on special significance, and is referred to as the master processor. Jobs are dispatched to each machine by the master processor and the results are similarly collated by the master processor. The information written to disk by each processor consists of intermediate calculations performed by the Baum Welch training [47, 48] (e.g., state occupancies). The master processor then combines these accumulators to get an estimate of the transitions and the state distributions, updates the model parameters, and resumes another iteration of training.

During the course of the project we found out that the results fluctuate when varying the number of training processors as shown in Table 9. We have also included results from Sun Sparc processors, since their floating point arithmetic differs slightly from the Intel/AMD processors. A post-evaluation analysis showed that these fluctuations were due to the lack of precision involved in storing the accumulators and then combining these accumulators to get the estimate of the state distributions and transitions. Using a single-precision floating-point representation (32 bits) to store the intermediate information in the accumulators added too much computational noise and resulted in these fluctuations. The use of a double-precision floating-point representation (64 bits) to store intermediate information in the accumulators resolved this problem.

However, there is a trade-off in terms of the memory required to store these intermediate results. A typical states file in single-precision floating-point format (32 bits) requires about 30 Mbytes per processor for our typical conversational speech models. We typically use 16 processors for training for WSJ0, which implies the total disk space required for one experiment (for these

Processor	float / double	Number of Training Processors	WER
Intel / AMD	float	1	28.0%
Intel / AMD	float	2	28.2%
Intel / AMD	float	10	28.2%
Intel / AMD	double	1	26.9%
Intel / AMD	double	2	26.9%
Intel / AMD	double	10	26.9%
Sun Sparc	float	1	26.7%
Sun Sparc	float	2	27.5%
Sun Sparc	float	10	26.9%
Sun Sparc	double	1	26.9%
Sun Sparc	double	2	26.9%
Sun Sparc	double	10	26.9%

Table 9. A comparison of WER's when the intermediate information in accumulators was stored in single or double precision floating point. Training was conducted on the short-1792 (filtered) training set and testing was conducted on the filtered devtest-30 set.

accumulators) is about 0.5 Gbytes. By promoting this data type to double precision (64 bits), we require about 1 Gbyte of disk space. Fortunately, since, we overwrite states and transitions files after each iteration of training, the disk space required for these accumulators is not cumulative.

The baseline system used in this study incorporates more state-tying than our typical conversational speech systems. Much less space is required to store models because we use fewer real states. Hence, our WSJ models require only about 0.3 Mbytes per processor, and the overall disk space requirement, even in double precision, is fairly small.

5.3. Endpointing

An important goal of this project was to study the effects of end pointing on the performance of the ASR system under various noise conditions. It had been observed in previous Aurora evaluations that the word accuracy decreases dramatically for all the three test conditions (A, B and C) when training on multi-condition (noisy) data as compared to training on clean data. This phenomenon is mentioned in Table 8 of the previous experimental framework report [46]. Most of the errors on noisy data are due to the insertion of a word in the hypothesis. The inserted word is typically aligned to a noisy segment of data (i.e. a segment where there is no speech but where the noise level is sufficiently high for the speech models to match it well). The rate of insertions can be reduced by removing silence from the beginning and end of an utterance in a process known as endpointing.

For the current evaluations it was decided to have each site use a common set of endpoints. The common endpoints were generated by force-aligning all utterances using our best WSJ models — a set of 16-mixture cross-word models that gave a WER of 8.3% on the Nov'92 evaluation set. Note that these models were more precise than the models used in the actual evaluations, so the quality of endpoints should be higher than what would be generated from the evaluation system. In this way, we make the accuracy of the endpoints less of an issue.

After performing forced alignment, the endpoints were set such that 200 milliseconds of silence was retained preceding the beginning of the first word and following the end of the last word. Additional details on the endpointing process and algorithm employed is provided in the Section 6.4 of this report. To facilitate endpointing, a simple script was provided that processed a file list. An example of the command line for this script is shown in Figure 8. The command line options for this script are shown in Table 9.

ENDPOINTING

```
wsj_strip -utterance_timing_list ./eval_0330.text
-input_mfc_list eval01_0166_v1.4.1.list
-output_path /isip/data/corpora/wsj0/13mfc/etsi_fe/test_ep/test_clean_wv1
-extension endpt -format htk -dimension 13 -frame_duration 0.01
```

Figure 8. A typical command line for invoking the endpointing script that generates excised feature files.

-*input_mfc_list <list>*: the list of files to be processed.

-*feature_format <value>*: htk (HTK formatted features), isip_proto (prototype system format).

-*utterance_timing_list <file>:* contains segmentation information for all utterances in the input list.

-*output_path <path>*: path location where the output files should be written (default: same as the input)

-*extension <value>*: file extension for the output files (default is .endpt).

-*dimension <value>*: the dimension of the input feature vectors (default is 13).

-*frame_duration <value>*: the frame duration in seconds (default is 0.01).

Figure 9. An explanation of the command line options available in the endpointing script. Note that this script requires segmentation information as input — something that is generated by the recognizer.

The endpoint utility was folded into the multiple-CPU scripts that were developed as part of this project. Users must have already installed v5.11 of the ISIP prototype system before running the multi-CPU scripts. Installation of this package is straightforward using standard Unix configure and make utilities. Detailed instructions are included in the release's *AAREADME.text* file. We also released the endpointing information for the standard data sets [45] as a separate distribution that is available on the project web site.

6. SHORT SET DEFINITIONS

LVCSR experiments are computationally expensive and require a fairly large amount of infrastructure. Most of the sites participating in Aurora evaluations did not have such a large infrastructure. Hence, a goal was established to define a small subset that would provide results comparable to a full evaluation and yet run in a single day's worth of CPU time on an 800 MHz Intel CPU. Though such short sets are notoriously misleading, it was considered a priority to provide such sets to the working group. Below we describe the development of various short sets, and other modifications made to the standard evaluation data set to meet the needs of the Aurora evaluation.

6.1. Training Subset Selection

The WSJ0 SI-84 training set consists of 7,138 utterances, 83 speakers and over 14 hours of data. There are more than 129,000 word tokens and about 10,000 unique words. The average number of words per utterance is 17.8, and the average utterance duration is 7.6 secs. The average speaking rate is about 2.4 words per second. The training set includes utterances with verbalized punctuation. The distribution of the number of words per utterance for the entire training set is shown in Figure 10. Figure 11 summarizes the distribution of the utterance durations.

One major design decision in the construction of the short set was to preserve all 83 speakers since we are concentrating on speaker independent recognition. Given the constraints on CPU time to run a typical tuning experiment in a single day on an 800 MHz Intel CPU, we decided on selecting 415 training utterances and 30 dev test set utterances. Since training on full SI-84 set takes about 10 days, training time on 415 utterances linearly scales down to about 16 hours. Similarly, the decoding time of 50 hours on 330 utterances scales down to 5 hours on 30 utterances. Thus, the total time for one complete experiment would be about one day. This, in

turn, motivated a second major design decision: uniformly sample each speaker, including 5 utterances per speaker. Since the average number of words per utterance was 18, we decided to throw out utterances that were extremely short (less than 8 words) and long (greater than 24 words) with respect to the average length. This reduced SI-84 to 4944 utterances. We then randomly sampled the remaining utterances from each speaker to obtain a total of 415 utterances (83 speakers x 5 utterances per speaker). We will refer to this as short-415 [42]. Its word count and duration statistics are compared to the full training set in Figures 12 and 13 respectively. Both the distributions for the short-415 are peaky compared to the distributions for the SI-84 training set because extremely short and long utterances with respect to average length were not included in the short-415 set.

Unfortunately, performance of system trained on 415 utterances even for the simplest of systems (1-mixture cross-word models) was poor — 44% WER as shown in Table 10. This system was tested on a short development set consisting of 30 utterances. Hence, we followed a similar paradigm but doubled the training set size to 830 utterances (short-830). The performance on short-830 for 1-mixture cross-word models was also poor — 36.0% WER. We then decided to increase the training set size to a quarter of the total training utterances in SI-84. This yielded short-1785 with an error rate of 25.5%.

Since the training subset has to be consistent for both the Training Set 1 (clean) and Training Set 2 (multi condition data), we decided to sample one-fourth of each of the clean and noisy utterances from both the Sennheiser as well as the second microphone conditions. The distribution of data in Training Set 2 is shown in Figure 1. We alternately picked 112 utterances from each of the 12 noisy blocks and 224 utterances from each of the two clean blocks to obtain 1,792 utterances. We refer to this set as short-1792. This is what was used for acoustic training in the evaluations. A summary of the word count and duration statistics are shown in Figures 14 and 15 respectively. Note that the word count distribution as well as the utterance duration distribution for the short-1792 is very similar to the respective distributions for the SI-84 training set.

Key statistics for all short training sets are provided in Table 11. Although the average duration and speaking rate is almost constant across all the training sets, the total number of unique words drastically reduce as the number of utterances decreases. This often results in undertrained acoustic models since there are an insufficient number of instances of each phonetic context to support reliable training.

6.2. Devtest Subset Selection

The Nov’92 development test set consists of 1206 utterances, and includes 10 unique speakers, and totals over 134 minutes of data. Similarly, the Nov’92 evaluation set consists of 330 utterances, 8 speakers, and about 40 minutes of data. Our goal was to produce a short set that is a reasonable match to the statistics of both of these sets. Following the same strategy described

Acoustic Models	Training Set	Devtest-30
CI-Mono-1-mix	415	46.0%
CD-Tri-1-mix	415	44.1%
CI-Mono-1-mix	830	46.6%
CD-Tri-1-mix	830	36.0%
CD-Tri-1-mix	1785	25.5%

Table 10. Performance as a function of the training set for the baseline system (with ISIP’s standard front end and unfiltered audio data).

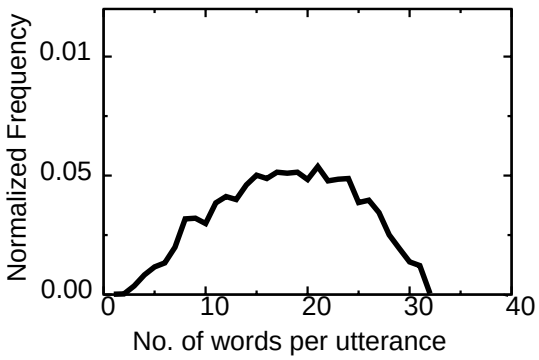


Figure 10. A histogram of the number of words per utterance for the full training set (SI-84).

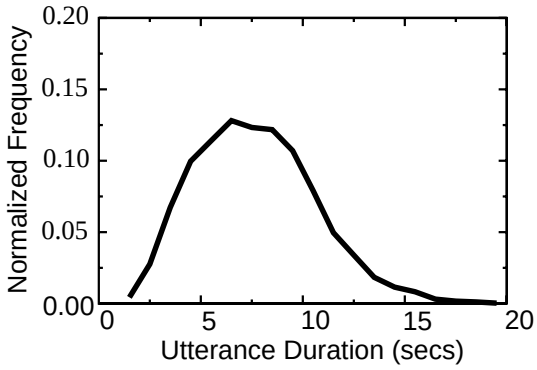


Figure 11. A histogram of the utterance durations for the full training set (SI-84).

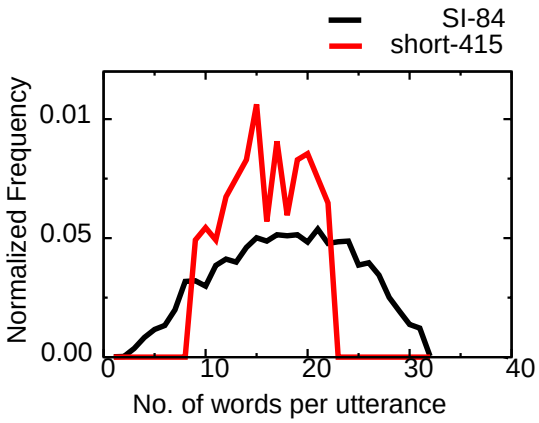


Figure 12. Comparison of the histograms of the number of words per utterance for the full training set (SI-84) and the short-415 training set.

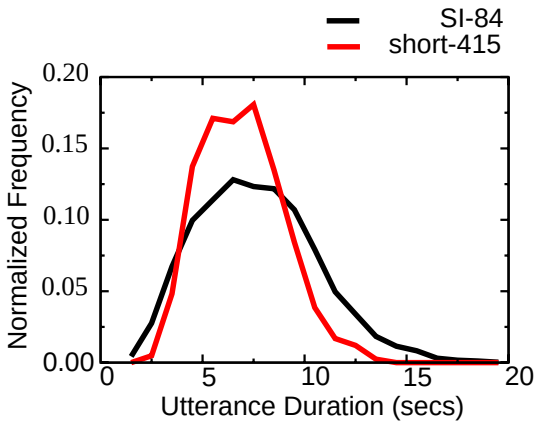


Figure 13. Comparison of the histograms of the utterance durations for the full training set (SI-84) and the short-415 training set.

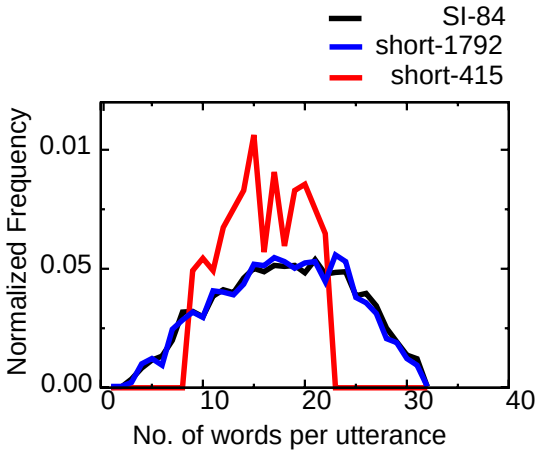


Figure 14. Comparison of the histograms of the number of words per utterance for the full training set (SI-84) and the short-1792 training set.

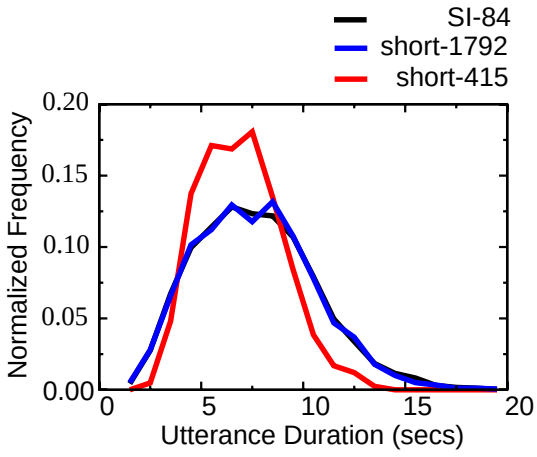


Figure 15. Comparison of the histograms of the utterance durations for the full training set (SI-84) and the short-1792 training set.

above, we selected 3 utterances per speaker for a total of 30 utterances. (This is a dangerously small development test set.) We also reduced the 1206 utterance set to a 330 utterance set for internal testing and tuning of the baseline system.

We sampled the 1206 utterance Nov'92 dev test set to obtain devtest-330. We decided to make devtest-30 a subset of devtest-330. More importantly, we decided not to throw out long or short utterances this time, since that might create a mismatch between the language model and the test set. Hence, we attempted to sample the entire distribution. In Figures 16 and 17, we compare the word counts and duration statistics for these short sets to the full Nov'92 dev test set.

In Table 12, we analyze the statistics of these three sets. Most of the important statistics such as the number of speakers, average utterance duration and average speaking rate for these three sets are constant. Although the number of unique word tokens reduces as the size of the test set decreases, the perplexities of the dev test sets are comparable.

Training Set Size	Total Number of Words	Average No. Words/ Utterance	Number of Unique Words	Average Duration (secs)	Average Speaking Rate (words/sec)
415	6,797	16.4	2,242	6.85	2.4
830	14,996	18.1	3,626	7.67	2.4
1785	32,085	18.0	5,481	7.59	2.4
1792	32,012	17.9	5,444	7.63	2.4
SI-84	128,294	18.0	8,914	7.62	2.4

Table 11. A comparison of some vital statistics for various training subsets.

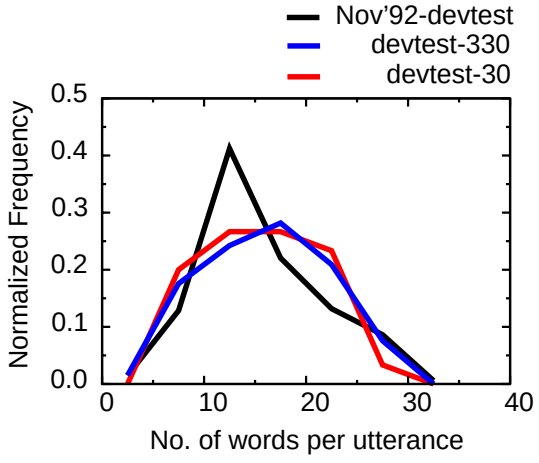


Figure 16. Comparison of the histograms of the number of words per utterance for the full development test set and two dev test subsets.

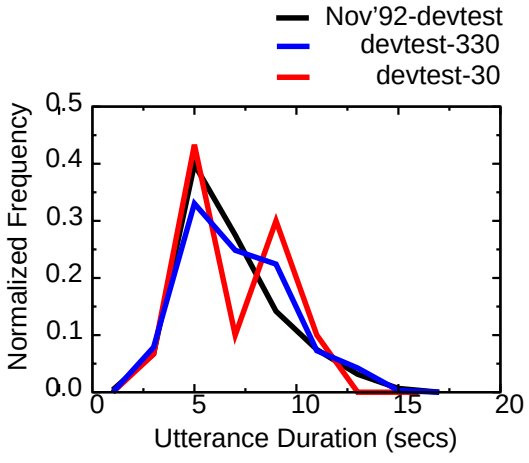


Figure 17. Comparison of the histograms of the utterance durations for the full development test set and two dev test subsets.

6.3. Eval Subset Selection

In order to reduce the computing requirements for the evaluations, we decided to reduce the size of the evaluation set by 50%. To obtain this shortened version of the evaluation set, we began by sampling in such a way that every speaker in the eval set was represented in the shortened set. This is shown in Table 13. We randomly sampled utterances from each of the speakers. This random sampling process was repeated four times to get four different eval short lists (A, B, C, D). Next, we computed the WER for these short sets based on results for the complete evaluation set for a series of experiments on various noise conditions. These results are shown in Table 14.

We then analyzed each set using a number of statistical distance measures to determine the set that is closest to the original eval set. These results are shown in Table 15. We chose subset "A" since this was closest to the results for the full evaluation set for the normalized statistical measures 3 and 4. The word count and duration statistics are compared to the full evaluation sets in Figures 18 and 19 respectively. Both the words count distribution and utterance duration distribution for eval-166 match closely to the respective distributions for Nov'92-eval set.

Description	Nov'92 eval	eval-166	Nov'92 devtest	devtest-330	devtest-30
No. of Speakers	8	8	10	10	10
No. of Utterances	330	166	1206	330	30
Amount of Data (mins.)	40.19	20.69	134.42	38.33	3.35
No. of Word Tokens	5,353	2,715	19,254	5,468	493
No. of Unique Words	1,270	936	2,404	1,444	290
Avg. No. of Words Per Utt.	16.2	16.3	16.0	16.6	16.1
Avg. Utt. Duration (secs)	7.3	7.5	6.7	7.0	6.7
Avg. Spk. Rate (words/sec)	2.2	2.3	2.4	2.4	2.4
Test Set Perplexity	134.9	139.0	146.8	143.7	151.5

Table 12. A comparison of the complexity of several subsets of the evaluation and devtest data.

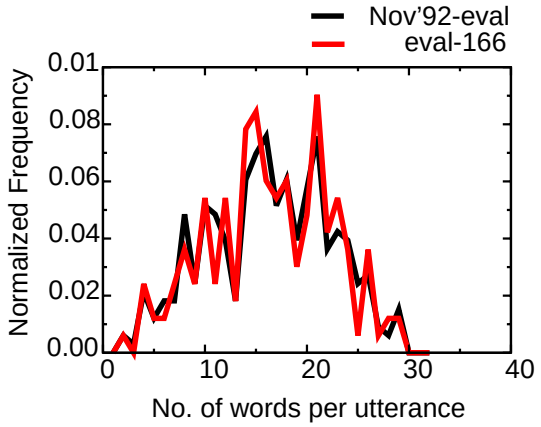


Figure 18. Comparison of the histograms of the number of words per utterance for the full Nov'92 evaluation set and eval-166.

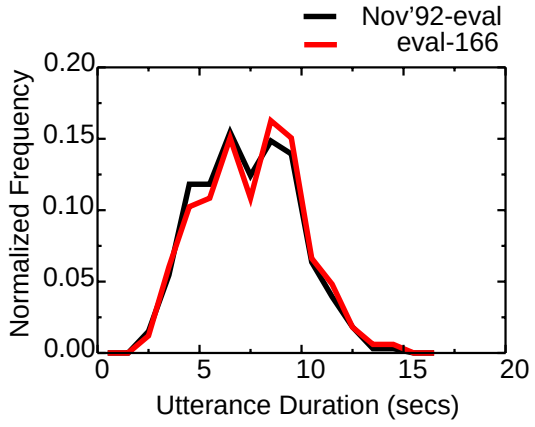


Figure 19. Comparison of the histograms of the utterance durations for the full Nov'92 evaluation set and eval-166.

6.4. Endpointed Speech Data

When speech recognition systems are subjected to severe amounts of noise, the nonspeech data preceding and following the utterance tends to cause insertion errors. The insertion errors can often be the dominant reason for an increased WER. It was decided that these nonspeech segments should be removed so that the evaluation can focus on recognition performance on valid speech data.

To remove the effects of this noise, it was decided to endpoint all speech data, so that complete experiments (training and evaluation) could be performed on endpointed data. We generated endpoint information using our best overall recognition system in forced-alignment mode. The endpoints were then extended by 200 msec on each side of an utterance. Table 16 summarizes the amount of silence in WSJ data. The first row represents the average number of seconds of audio data removed from each file (amount of data removed per utterance averaged across all utterances). The second row represents the amount of silence removed as a percentage of the total available audio data. The third row represents the total amount of data discarded as a percentage of the total audio data. Note that the eval data tended to have more silence than the training data. This somewhat supports

Spk. ID	No. of Utt. (full eval)	No. of Utt. (eval-166)
440	40	20
441	42	21
442	42	21
443	40	20
444	41	21
445	42	21
446	40	20
447	43	22
Total	330	166

Table 13. Distribution of the number of times each speaker appears in the evaluation set.

Test Set	Training Set	Eval Set	A	B	C	D
1	1	10.1%	10.2%	10.4%	9.7%	9.4%
2	1	55.4%	56.1%	54.9%	56.2%	58.0%
3	1	64.6%	64.8%	63.2%	66.1%	66.9%
4	1	58.4%	59.2%	59.5%	62.0%	58.8%
6	1	61.0%	61.7%	60.4%	63.5%	62.4%
8	1	53.7%	54.6%	52.5%	54.6%	55.5%
9	1	71.6%	72.5%	71.5%	72.8%	73.2%
10	1	76.2%	77.5%	75.5%	77.4%	79.5%
11	1	76.7%	78.5%	74.0%	77.5%	77.5%
13	1	74.5%	77.1%	74.1%	75.8%	76.9%
1	2	27.2%	27.9%	28.0%	28.0%	26.9%

Table 14. WER on various noisy conditions for Eval set and the subsets A, B, C, and D.

Distance Measure	A	B	C	D
$\sum abs(x - x_i))$	10.70	9.80	15.00	17.60
$\sum (x - x_i)^2$	15.47	13.90	28.92	37.84
$\sum abs\left(\frac{(x - x_i)}{x_i}\right)$	0.18	0.19	0.28	0.33
$\sum \left(\frac{(x - x_i)}{x_i}\right)^2$	0.22	0.23	0.48	0.60

Table 15. Results of several distance measures applied to select a subset of the full evaluation set.

Data Set	SI-84	dev-330	eval
Average Silence/ Utt. (secs)	0.82	0.97	1.37
Average Silence/ Utt. (%)	11.2	16.4	25.3
Total Silence (%)	10.8	13.9	18.8

Table 16. A summary of the amount of silence detected in the WSJ Corpus.

the approach of conducting the evaluation on endpointed data so that training and evaluation conditions were matched. The endpoint information was released to the project web site along with related supporting software [45].

7. BASELINE EXPERIMENTS AND RESULTS

We conducted a series of experiments to study the impact in performance of the baseline system over a variety of noise conditions. Summaries are provided in Tables 17 and 18. Table 17 provides results for experiments conducted without any feature value compression, and Table 18 provides results with compression. The compression algorithm used is described in detail in [49]. Each row in these tables consists of seven different test conditions: clean data plus six noise conditions. The original audio data for test conditions 1-7 was recorded with a Sennheiser microphone while test conditions 8-14 were recorded using a second microphone that was randomly selected from a set of 18 different microphones. These included such common types as a Crown PCC-160, Sony ECM-50PS, and a Nakamichi CM100. Noise was digitally added to this audio data to simulate operational environments. Refer to Figure 2 for a detailed description of the construction of these test sets.

All training sets were digitally filtered using either P.341 [6] at a 16 kHz sampling frequency or G.712 [5] at an 8 kHz sampling frequency. This is described in more detail in Figure 1. The impact of using endpointed speech, described in Section 6.4, was also evaluated as an independent variable. For the no compression case, the seven test conditions were then evaluated for several combinations of these conditions, resulting in a total of 98 conditions: 7 noise conditions x 2 microphone types x (3 training conditions for Training Set 1 + 3 conditions for training set 2 + 1 condition for Training Set 3). For the compression case, the seven test conditions were then evaluated using only endpointed speech, resulting in a total of 56 conditions: 7 noise conditions x 2 microphone types x (2 training conditions for Training Set 1 + 2 conditions for Training Set 2). Hence, a total of 154 test conditions were evaluated. These tables constitute a total of 4,580 hours (191 days) of CPU time on a 800 MHz Pentium processor.

Performance Summary (Without Compression)																
Training Set			Test Set													Average
Set	Sampling Frequency	Utterance Detection	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
1	16 kHz	No	14.9	65.2	69.2	63.1	72.3	69.4	73.2	61.3	81.7	82.5	75.4	83.8	81.0	84.1	69.8
	16 kHz	Yes	14.0	56.6	57.2	54.3	60.0	55.7	62.9	52.7	74.3	74.3	67.5	75.6	71.9	74.7	60.8
	8 kHz	Yes	16.2	49.6	62.2	58.7	58.2	61.5	61.7	37.4	59.7	69.8	67.7	72.2	68.3	67.9	57.9
2	16 kHz	No	23.5	21.9	29.2	34.9	33.7	33.0	35.3	49.3	45.2	49.2	48.8	51.7	49.9	49.0	39.6
	16 kHz	Yes	19.2	22.4	28.5	34.0	34.0	30.0	33.9	45.0	43.9	47.2	46.3	51.2	46.6	50.0	38.0
	8 kHz	Yes	18.4	24.9	37.6	39.3	38.8	38.2	40.4	29.7	37.3	48.3	46.1	50.6	44.9	49.3	38.8
3	16 kHz	No	20.6	23.2	34.4	40.1	38.2	34.7	41.3	46.8	49.1	53.5	53.4	57.2	53.2	56.1	43.0

Table 17. A summary of results (in terms of WER) obtained by the Aurora baseline system on the WSJ task. This series of evaluations did not use any feature value compression. Training Set 2 with endpointed data and16 KHz sampling frequency is the overall best condition.

Performance Summary (With Compression)																
Training Set			Test Set													Average
Set	Sampling Frequency	Utterance Detection	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
1	16 kHz	Yes	14.5	58.4	58.8	53.8	62.5	56.9	65.5	53.3	75.1	76.3	68.5	77.8	73.5	75.9	62.2
	8 kHz	Yes	15.4	49.4	60.6	59.0	57.4	61.9	62.0	36.6	59.9	71.6	67.8	72.5	70.2	69.5	58.1
2	16 kHz	Yes	19.1	23.4	31.7	35.5	35.3	33.1	36.4	40.9	47.4	50.3	48.9	54.7	49.3	51.8	39.8
	8 kHz	Yes	20.7	26.4	38.6	41.6	43.8	41.1	43.4	30.9	38.7	47.1	50.1	53.6	47.3	50.7	41.0

Table 18. A summary of results for the baseline system with feature value compression. Training Set 2 with endpointed data and16 KHz sampling frequency is still the overall best condition, and performance trends are similar to the no compression case.

7.1. Reduction of CPU Requirements For Experimentation

Training a standard 16-mixture cross-word context-dependent phone HMM system involves 36 passes of Baum-Welch training. This requires approximately 275 hours, or 10 days, on an 800 MHz Pentium processor. Decoding the 330 utterances that constitute the Nov'92 eval utterances for Test Set 1 (clean data) requires about 50 CPU hours. Each noisy test set takes approximately three times longer because of the poor acoustic match between the models and the data. Hence, the total decoding time for 14 test sets is 150 x 14 hours = 84 days. For the 11 training conditions mentioned in Tables 17 and 18, the total CPU time required would have been 94 x 11 = 1034 days. This computational load was not feasible for most of the sites involved in the Aurora evaluations. Hence, we explored three ways to reduce this time without compromising the integrity of the system or the results:

- **Evaluation set size:** In Section 6.3 we described the selection process used to reduce the evaluation set from 330 utterances to 166 utterances. This resulted in a 50% reduction in runtime requirements.
- **Number of mixtures:** Mixture generation and training is another time consuming process, since it involves multiple passes through the data. For example, reducing the number of mixtures from 16 to 4 would reduce the number of training passes from 36 to 28. Hence, the computation time during training by a factor of 7/9, and result in minimal degradations in performance. An analysis of performance as a function of the number of mixtures is given in Table 19. We decided to select 4 mixtures for the final baseline system.
- **Beam pruning:** Decreasing beam widths in the search process is a straightforward way to reduce computational complexity. In Table 7, we evaluated performance as a function of a selected number of combinations of beam pruning parameters. We selected the settings "200 150 150" because it was observed that these settings reduced runtime by a factor of 6 with minimal degradations in performance.

Hence, after incorporating these optimizations, we were able to reduce the expected total computation time required to generate Tables 17 and 18 from 1,034 days to 163 days. In the following sections, we analyze the results for specific contrastive conditions. All results were generated using the standard NIST scoring software [50], and the NIST MAPPSWE significance test [44], which is included in this software, to determine statistical significance.

7.2. Sample Frequency Reduction

Most telephony applications use a sample frequency of 8 kHz even though state-of-the-art ASR systems use speech data digitized at a sample frequency of 16 kHz. Spectral information above 4 kHz can be exploited to provide modest improvements in performance. For example, the third formant for several speech sounds, such as the consonant "s", has significant energy above 4 kHz. In state-of-the-art systems, a sample frequency of 16 kHz is often used in conjunction with a Sennheiser close-talking microphone to achieve

Number of Mixtures	xRT	WER
2	115	11.8%
4	113	9.5%
8	116	8.7%
16	114	8.0%

Table 19. A summary of performance on the Nov'92 dev test set using the SI-84 training set as a function of the number of mixtures. The baseline evaluation system used 4 mixtures.

better performance. Hence, we measured performance at both 8 kHz and 16 kHz to analyze whether trends in recognition performance were consistent at both sample frequencies.

A comparison of performance for Training Sets 1 and 2 is shown in Figure 20 for the "no compression" case. A similar comparison for the compression condition is shown in Figure 21. For Training Set 1, degradations due to a reduction in sampling frequency did not follow any trend. However, for Training Set 2, statistically significant degradations in performance were observed on the Sennheiser microphone conditions (Test Sets 3-7) in both the "no compression" and "compression" cases. The Sennheiser HMD-414 is an expensive close-talking microphone which does a good job of maintaining a relatively flat frequency response from DC to 8 kHz. The spectrogram of a typical utterance (e.g., *441c020b*) recorded with the Sennheiser microphone demonstrates that this microphone preserves high frequency information better than the microphones used for the second channel condition. This observation is supported by Figure 23, which provides the overall frequency response of the microphones on speech and non-speech data, respectively.

However, no significant improvement is observed when the sampling frequency is increased from 8 kHz to 16 kHz on matched conditions — training on Training Set 1 and decoding on Test Set 1, as shown in Figures 20 and 21. These sets are matched since both consist of clean utterances recorded on Sennheiser microphone. The additional information provided by high frequencies (between 4 kHz and 8 kHz) does not contribute to any additional improvement in performance. The spectral information provided by low frequencies (below 4 kHz) is sufficient to reach the upper bound on performance.

7.3. Utterance Detection

In addition to the investigation whether the trends in the recognition performance were consistent at both sampling frequencies, we also investigated whether the recognition performance improved due to utterance detection. The non-speech segments in noisy environments often result in an increase in insertion errors. These non-speech segments were removed from the audio data using the methodology known as endpointing, described in Section 6.4, with an expectation that the insertion errors would reduce in noisy environments. As expected, the utterance detection resulted in a significant improvement in performance on Test Sets 2-14 when the system was trained on Training Set 1, as shown in Figure 24. Table 20 shows that the reduction in insertion errors is primarily responsible for improvement in the performance. In this case, the "silence" model learned only pure silence during training because Training Set 1 consists of only clean data, and hence did not represent a good model of the actual background noise. Without endpointing, the noisy silences were interpreted as the non-silence words instead of silences, resulting in insertion errors. Endpointing reduced the amount of non-speech data and hence reduced insertion errors.

In contrast to Training Set 1, for Training Set 2, a significant improvement in performance was detected only for Test Set 8. A reduction in the number of deletions, rather than insertions, was primarily responsible for this improvement in performance. In other words, because the training conditions contained ample samples of the noise conditions, the non-speech segments were modeled adequately by the silence model and hence the insertion error rate did not increase significantly on the noisy test conditions.

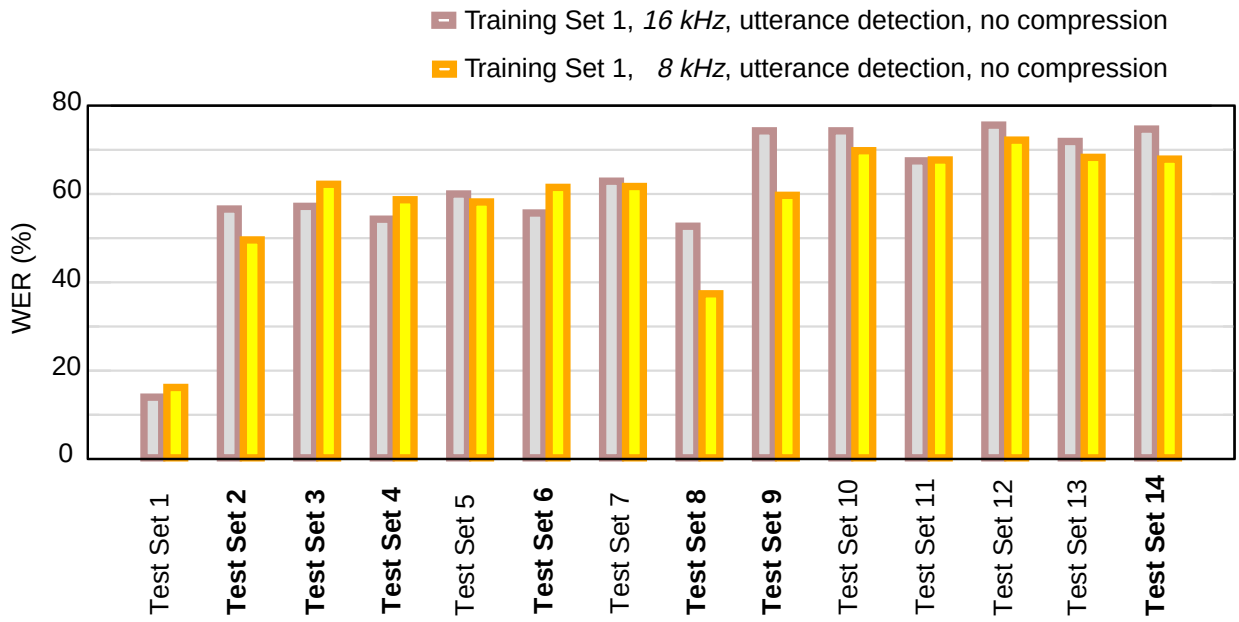


Figure 20(a). A comparison of the WER for *16 kHz* and *8 kHz* sample frequencies for Training Set 1 without feature vector compression. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. There is no significant trend in the results — overall performance at 16 kHz was not better than 8 kHz, as we might expect.

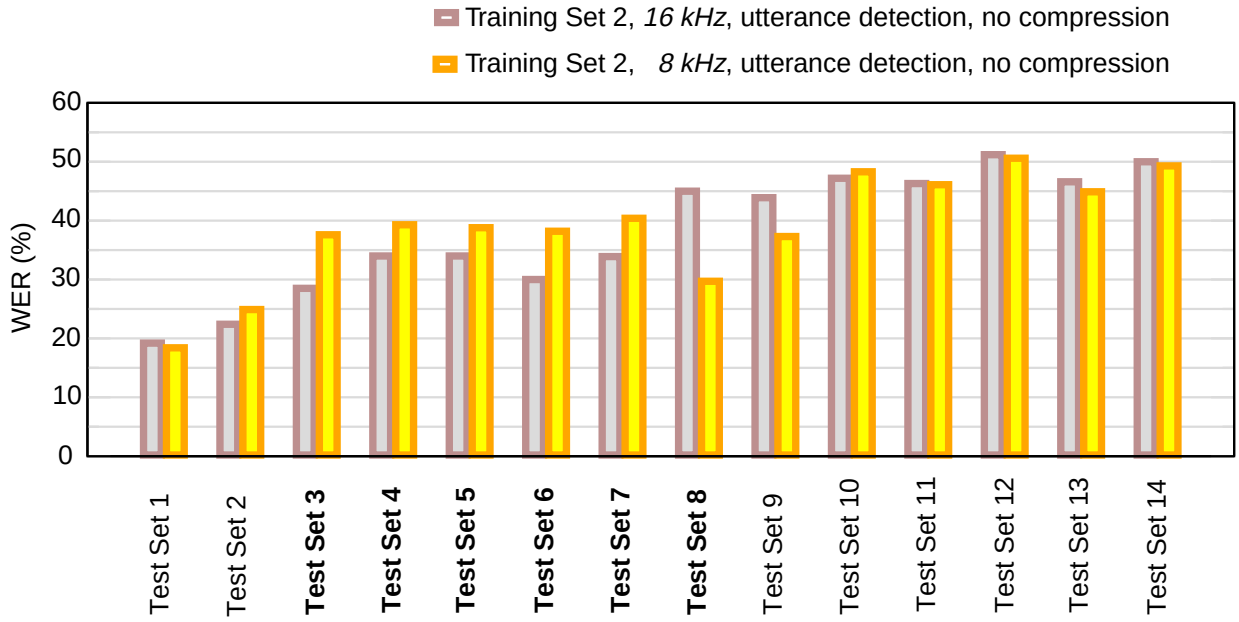


Figure 20(b). A comparison of the WER for *16 kHz* and *8 kHz* sample frequencies for Training Set 2 without feature vector compression. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. Performance was significantly better for *16 kHz* than *8 kHz* for data recorded with a Sennheiser microphone (Test Set 3-7) since Sennheiser microphone preserves the high frequency components. The high frequencies contain important spectral information that give a boost to the speech recognition performance in noisy environments when the speech is distorted by noise.

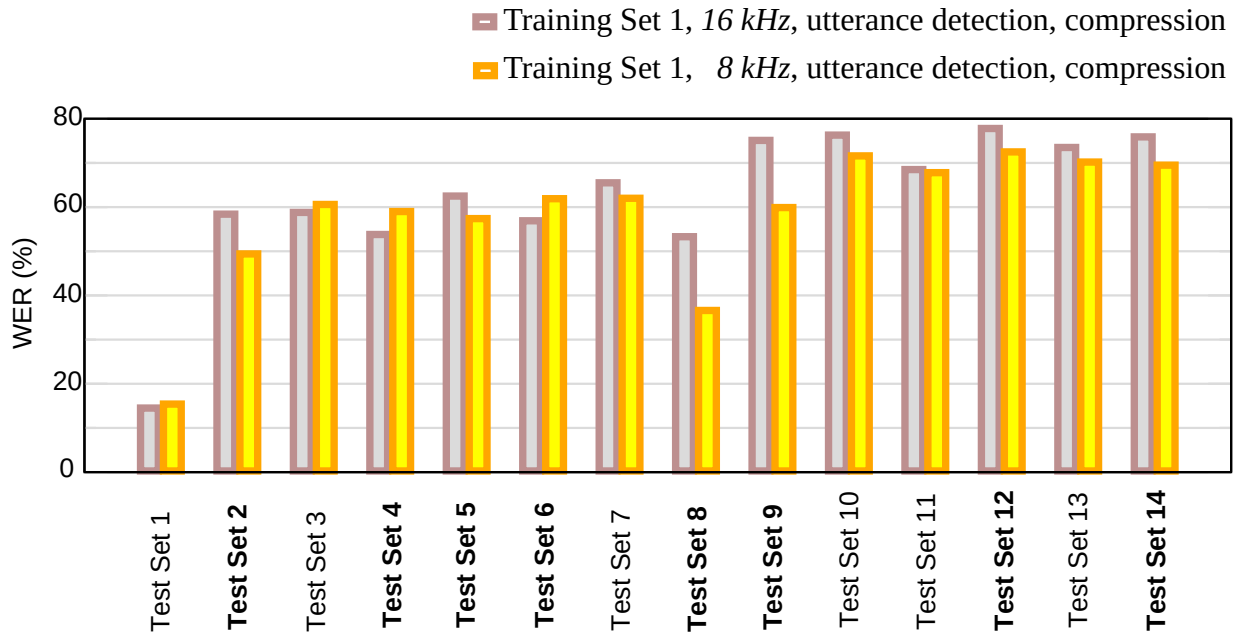


Figure 21(a). A comparison of the WER for *16 kHz* and *8 kHz* sample frequencies for Training Set 1 with feature vector compression.Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. There is no significant trend in the results — overall performance at 16 kHz was not better than 8 kHz, as we might expect.

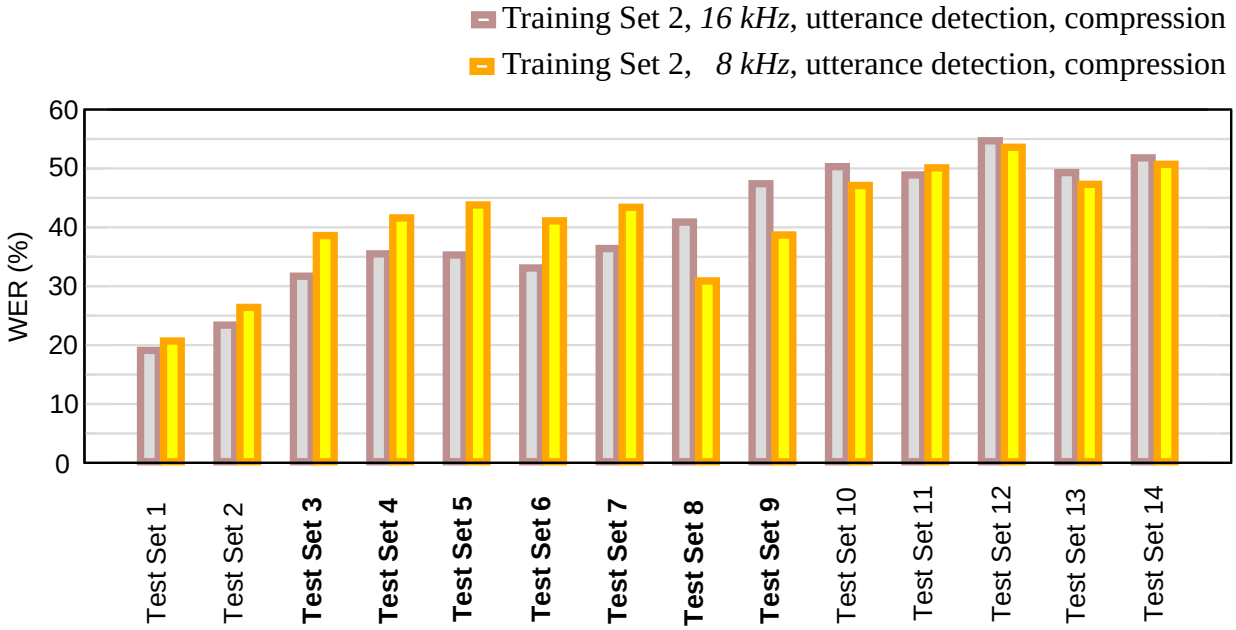


Figure 21(b). A comparison of the WER for *16 kHz* and *8 kHz* sample frequencies for Training Set 2 with feature vector compression. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. The trend is same as no-compression type. Sennheiser microphone gives significantly better performance for *16 KHz*, since it preserves the high frequency components. The spectral information in these high frequencies contribute to the improvement in the recognition results in noisy environments when the speech is distorted by noise.

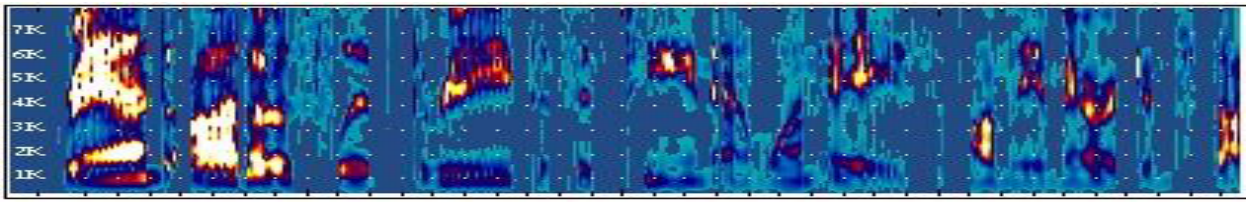


Figure 22(a). Spectrogram for utterance *441c020b* that is recorded on Sennheiser microphone, digitized at 16 kHz and filtered using P.341 [6]. High frequency energy up to 8 kHz are preserved. The high frequency components contain critical spectral information useful for recognition in the noisy environments when the low frequency components don't have accurate spectral information.

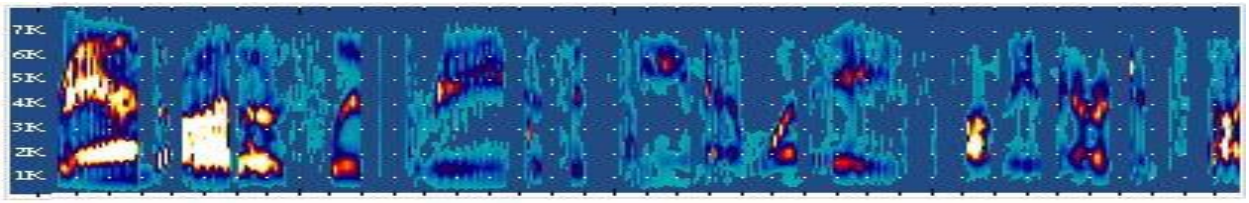


Figure 22(b). Spectrogram for utterance *441c020b* that is recorded on second microphone, digitized at 16 KHz and filtered using P.341 [6]. The high frequency components, especially above 4 KHz are not preserved as well.

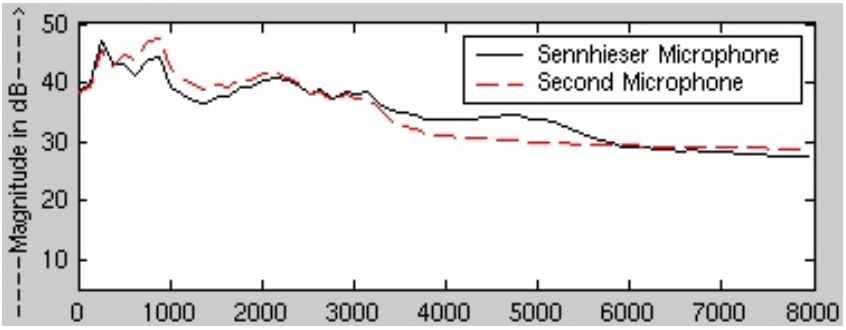


Figure 23(a). Comparison of the magnitude of the frequency response of the Sennheiser microphone and the second microphone derived from the speech segment from the utterance id *441c020b*. Both the utterances were digitized at 16 KHz and filtered using P.341 [6]. The Sennheiser microphone preserves the frequencies above 3.5 KHz by taking advantage of high sampling rate (16 KHz) while the second microphone filters the high frequencies.

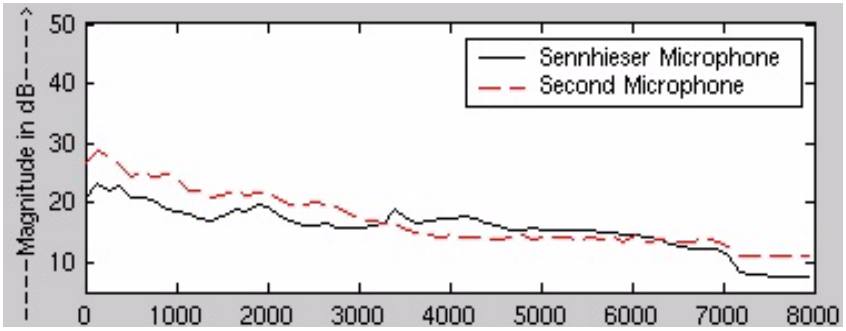


Figure 23(b). Comparison of the magnitude of the frequency response of the Sennheiser microphone and second microphone derived from the non-speech segment from the utterance id *441c020b*. Both the two utterances were digitized at 16 KHz and filtered using P.341 [6]. The Sennheiser microphone preserves the frequencies above 3.5 KHz, taking advantage of high sampling rate (16 KHz) while the Second microphone filters the high frequencies.

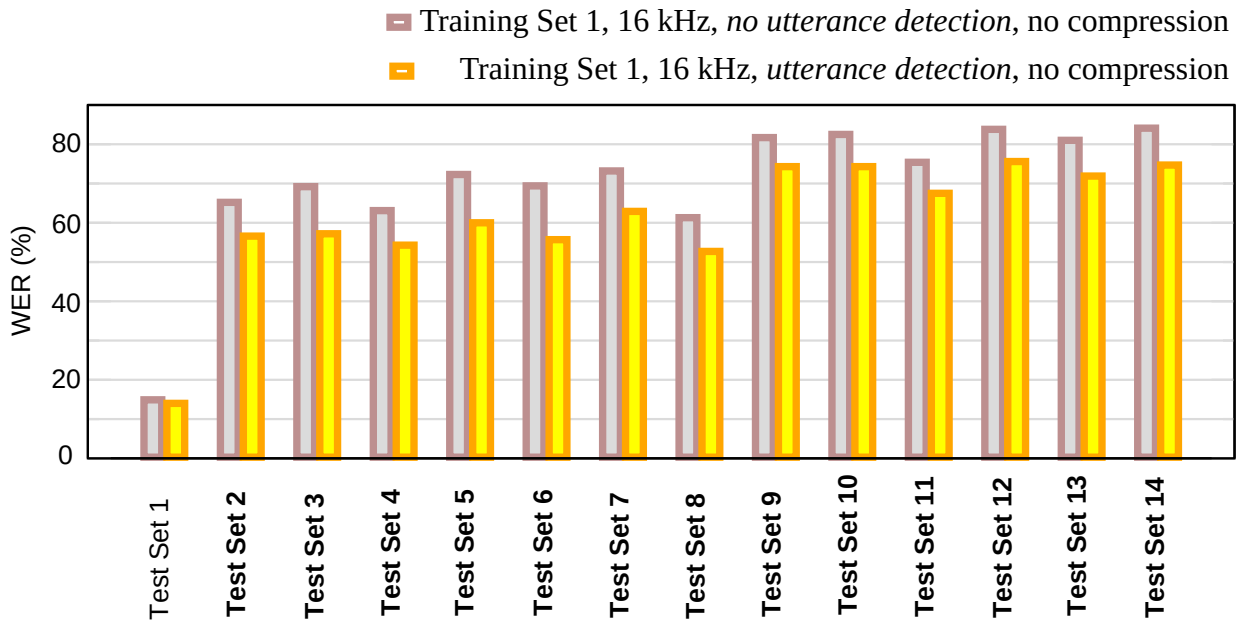


Figure 24(a). Comparison of the WER for *no endpointed* and *endpointed* data for Training Set 1 at 16 kHz with no feature vector compression. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. Significant reduction in performance is observed for all noisy conditions except the clean-Sennheiser microphone - as we might expect primarily because of the increase in insertion errors at the beginning and the end of test utterances.

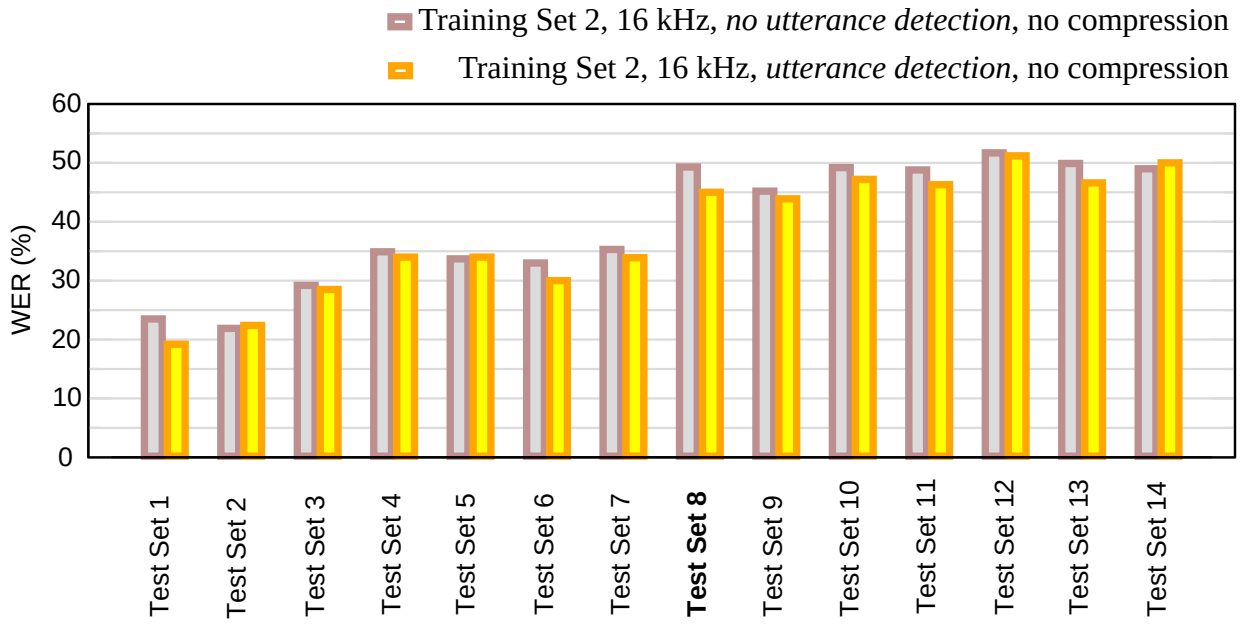


Figure 24(b). Comparison of the WER for *no endpointed* and *endpointed* data for Training Set 2 at 16 kHz with no feature vector compression. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. Significance is not detected for majority of the noisy conditions since the “silence” model learned the “silence + noise” from non-speech segments of the utterances in Training Set 2. Hence, noisy non-speech segments did not result in insertion errors.

Test Set	Training Set 1							
	Without Utterance Detection				With Utterance Detection			
	WER	Sub.	Del.	Ins.	WER	Sub.	Del.	Ins
1	14.9%	8.8%	1.0%	5.1%	14.0%	9.0%	0.8%	4.1%
2	65.2%	41.4%	3.6%	20.1%	56.6%	40.0%	3.6%	13.0%
3	69.2%	46.0%	6.5%	16.7%	57.2%	40.7%	6.2%	10.2%
4	63.1%	40.5%	12.0%	10.6%	54.3%	36.7%	10.8%	6.9%
5	72.3%	47.0%	11.2%	14.1%	60.0%	39.2%	13.8%	7.1%
6	69.4%	44.6%	7.8%	17.0%	55.7%	37.9%	8.2%	9.6%
7	73.2%	46.6%	14.1%	12.5%	62.9%	42.1%	13.7%	7.1%
8	61.3%	34.7%	14.6%	12.1%	52.7%	36.7%	8.7%	7.3%
9	81.7%	54.4%	12.3%	15.1%	74.3%	49.1%	15.1%	10.1%
10	82.5%	57.0%	12.2%	13.3%	74.3%	53.1%	13.0%	8.1%
11	75.4%	48.1%	17.9%	9.4%	67.5%	44.9%	17.5%	5.1%
12	83.8%	48.4%	26.7%	8.8%	75.6%	41.3%	30.5%	3.8%
13	81.0%	52.3%	15.5%	13.1%	71.9%	46.0%	18.4%	7.4%
14	84.1%	47.6%	26.2%	10.2%	74.7%	41.4%	28.5%	4.9%

Table 20. A comparison of experimental results for without and with endpointed data for Training Set 1 at 16 KHz with no feature vector compression. Test set conditions which are statistically significant at a 0.1% significance level are indicated by colored blocks. As expected, the reduction in insertion type errors is primarily responsible for the reduction in WER.

7.4. Compression

Continuing our investigation into the six focus conditions discussed in Section 1, we investigated the effects of compression on the features. It is desirable to compress feature values before transmission over a communications channel to conserve bandwidth. The compression algorithm employed in the DSR client-server application is a lossy split vector quantization (VQ) algorithm [49] that allows the quantized features to be transmitted at 4800 bps. Since this compression algorithm is lossy, the recovered features are a distorted version of the original features, and will result in a degradation in recognition performance. This degradation in performance was calibrated through a series of experiments described in Figures 25 and 26.

No significant degradation in performance due to compression was detected for Training Set 1 for both the 8 kHz and 16 kHz sampling frequencies. Since there is no significant degradation for Test Set 1 which is a matched condition, it is natural to draw a conclusion that the split VQ algorithm will not significantly degrade the performance of the system. However, Figure 26 shows that there was a significant degradation in performance for four noisy conditions at a 16 kHz sampling frequency and two noisy conditions at an 8 kHz sampling frequency on Training Set 2. We have not found a consistent explanation as to why these particular noise conditions were adversely affected.

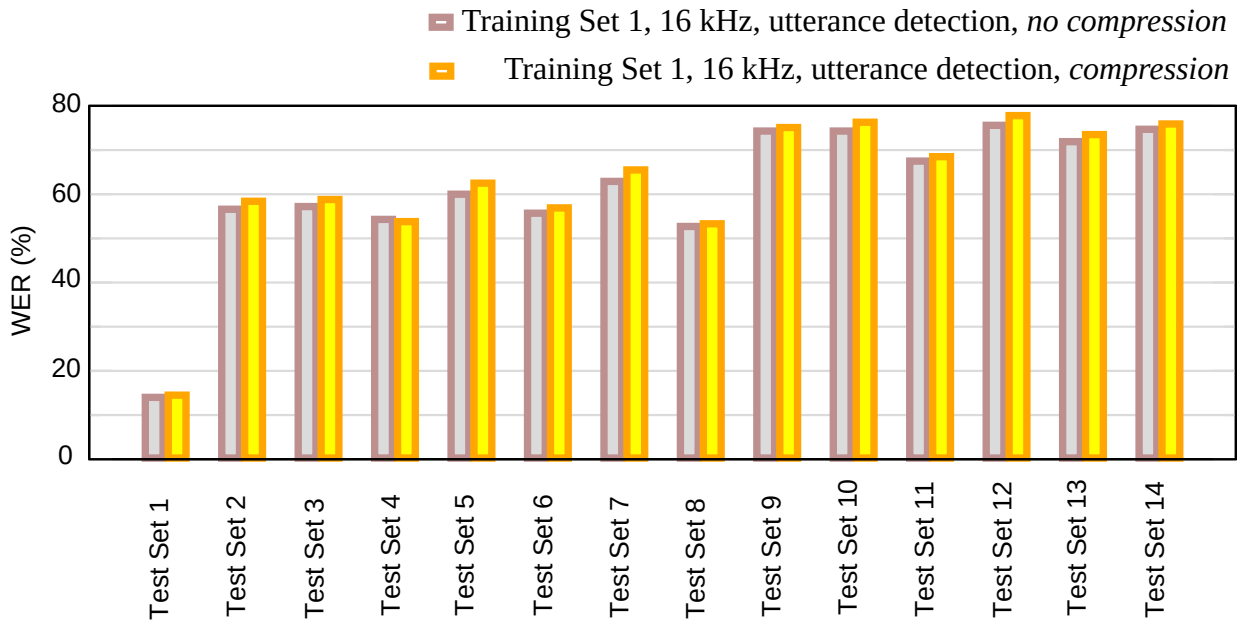


Figure 25(a). Comparison of the WER between *no compression* and *compression* of feature values on Training Set 1 at 16 kHz. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. The use of compression algorithm to compress feature values did not produce any significant degradations for any clean or noisy conditions.

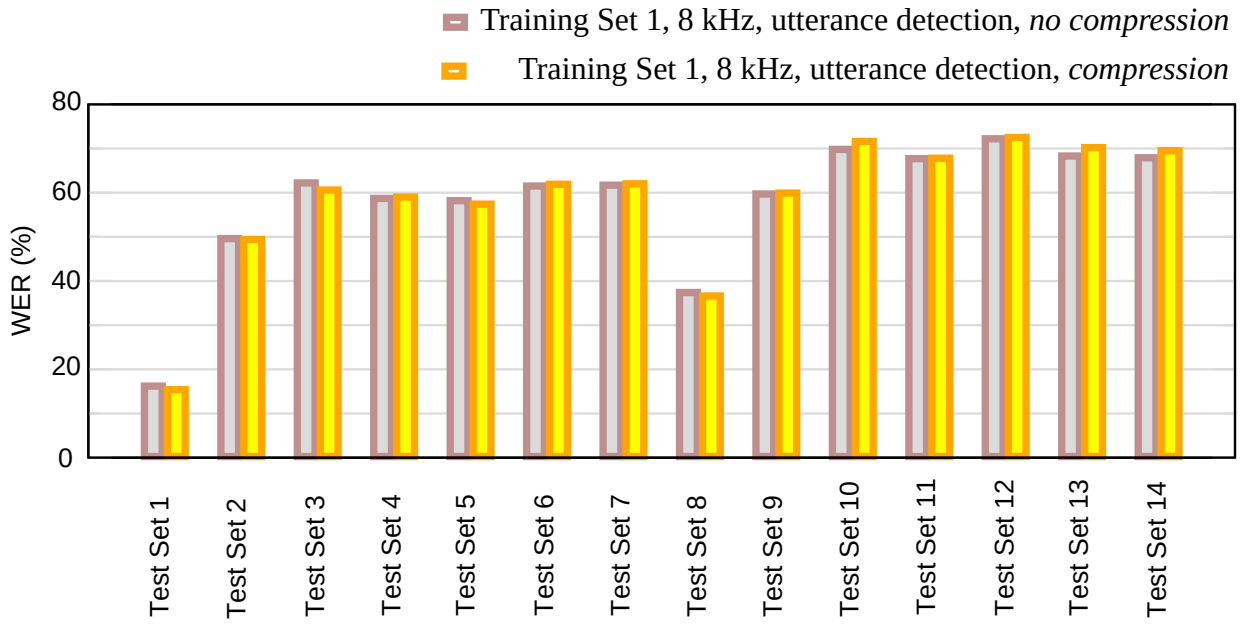


Figure 25(b). Comparison of the WER between *no compression* and *compression* of feature values on Training Set 1 at 8 kHz. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. The use of compression algorithm to compress feature values did not produce any significant degradations for any clean or noisy conditions.

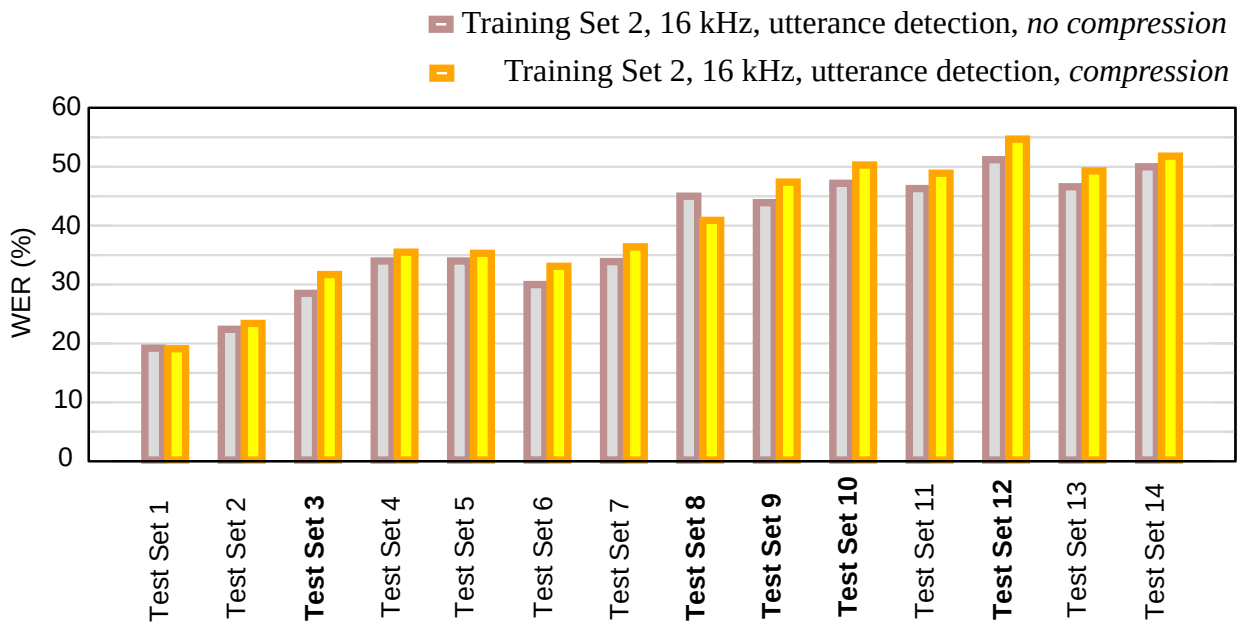


Figure 26(a). Comparison of the WER between *no compression* and *compression* of feature values on Training Set 2 at 16 kHz. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. There is no significant trend in the results.

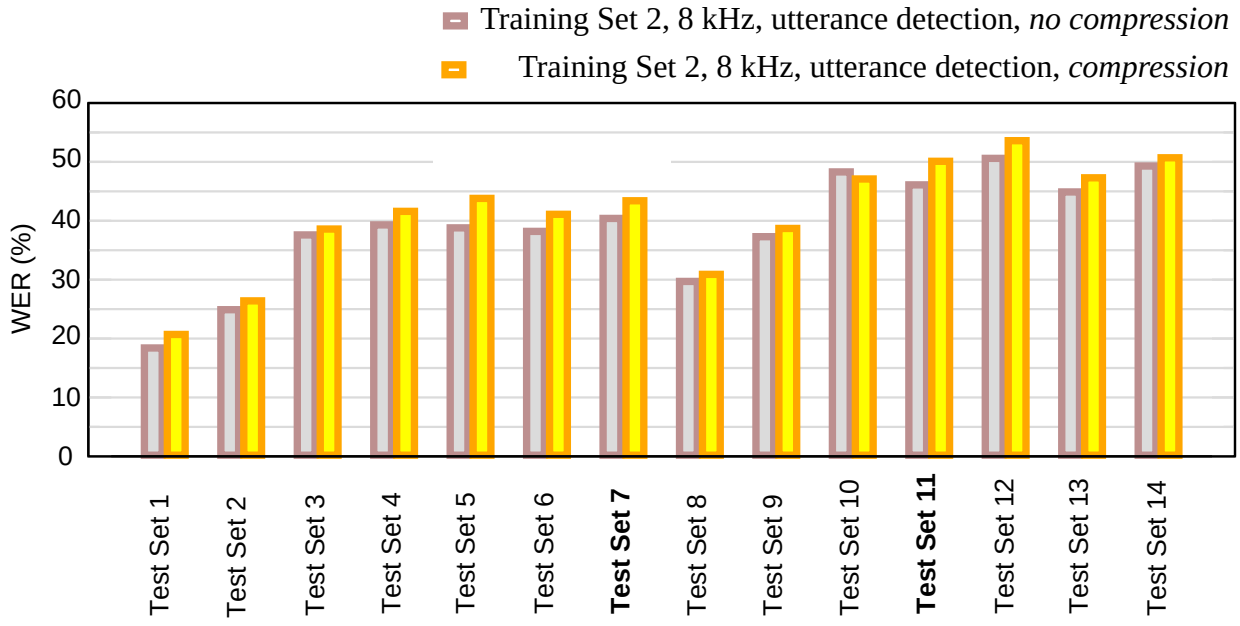


Figure 26(b). Comparison of the WER between *no compression* and *compression* of feature values on Training Set 2 at 8 kHz. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. There is no significant trend in the results.

7.5. Model Mismatch

In addition to investigating the trends in recognition performance due to reduction in sampling frequency, utterance detection, and compression of feature values, we also investigated the effects of model mismatch on the recognition performance. One would expect to attain high recognition performance on matched conditions, defined as an experimental condition in which both the training and the test data were recorded under identical conditions. Since training is based on a maximum likelihood parameter estimation process [19], high performance recognition can only be achieved when the test conditions generate feature vectors that are similar in terms of means, variances, etc. If there are consistent differences in SNR, background noise, or microphone, there will be a significant degradation in performance if some form of adaptation is not used. In these evaluations, it was decided not to consider adaptation within the recognition system. We calibrated the degradation in performance using a series of experiments summarized in Figures 27 and 28.

As expected, the best recognition performance was observed on matched training and testing conditions (Training Set 1 and Test Set 1) in which all utterances were recorded with a Sennheiser microphone. For all other conditions involving Training Set 1, recognition performance degraded significantly. Systems trained on Training Set 2 performed significantly better than those trained on Training Set 1 across all noise conditions. These trends were consistent for both sampling frequencies and both compression conditions. Reducing this degradation from mismatched conditions through front end processing was a major goal in this evaluation.

7.6. Microphone Variation

Next, we investigated the effects of microphone variation on speech recognition performance. In general, the Sennheiser microphone performed significantly better than the second microphone condition for all conditions, as shown in Table 21. The first cell in this table corresponds to Training Set 1, which consists of clean utterances recorded with a Sennheiser microphone, and Test Set 1, which consists of similar data. The second cell in the first row represents a mismatched condition in which the test set contained a different microphone. There was a significant increase in the word error rate, from 16.2% to 37.4%. The same argument of model-mismatch discussed in the previous section can be extended to explain the degradation in the performance. The same trend is observed on the car noise condition (Test Sets 2 and 9).

While Training Set 1 consists of utterances recorded with a Sennheiser microphone, Training Set 2 has half of the utterances recorded on the same Sennheiser microphone and the other half on any one of the 18 microphone types described in Section 2. With the Baum-Welch training algorithm, a maximum likelihood based parameter estimation method, this fact implies that models trained on Training Set 2 quickly converge towards the Sennheiser microphone in terms of their means and the covariances [51]. Hence, both the clean and car test conditions for the second microphone result in significant degradation in recognition performance, as shown in the second row of the Table 21. Note also that the last three cells in the second row, which correspond to various noise conditions, show less of a degradation in performance than the corresponding conditions in the first row. So there is some value in exposing the models to noise during the training process, but not as much as had been hoped for in the initial design.

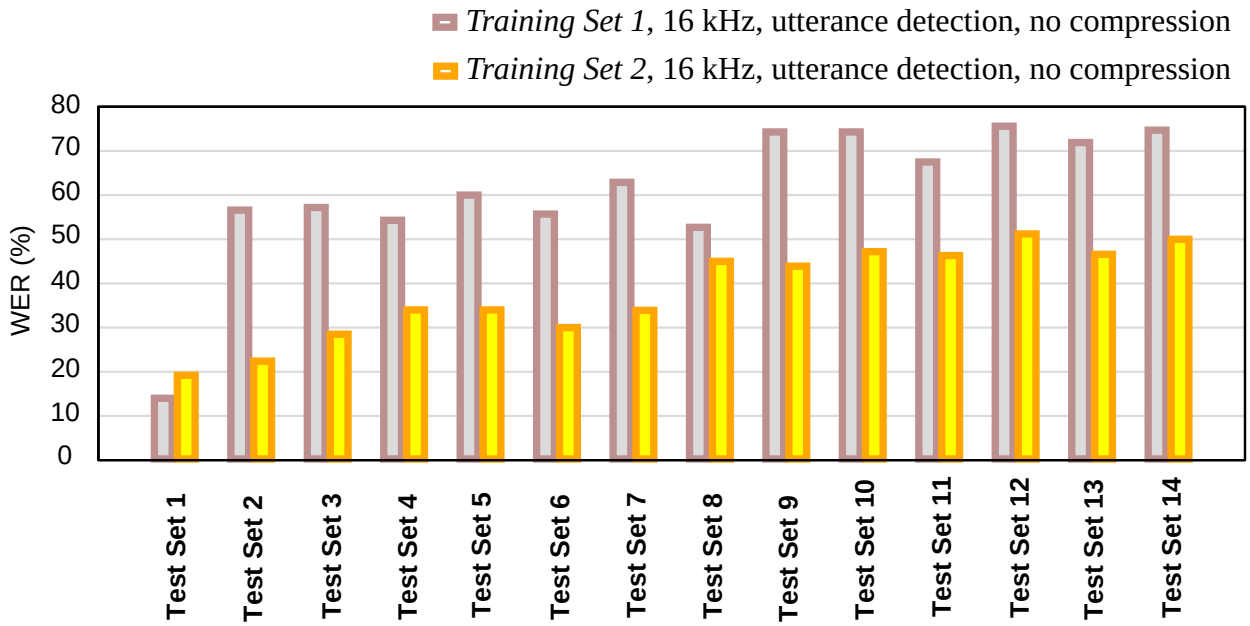


Figure 27(a). Comparison of the WER between *Training Set 1* and *Training Set 2* at 16 kHz with no feature value compression. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. Matched conditions without background noise and Sennheiser microphone (Training Set 1 and Test Set 1) resulted in the best recognition performance as one would expect. Training Set 1 is significantly worst than the Training Set 2 on all other test sets because it is a better match to the test conditions. Training Set 2 is representative of all the background noise and the microphone types.

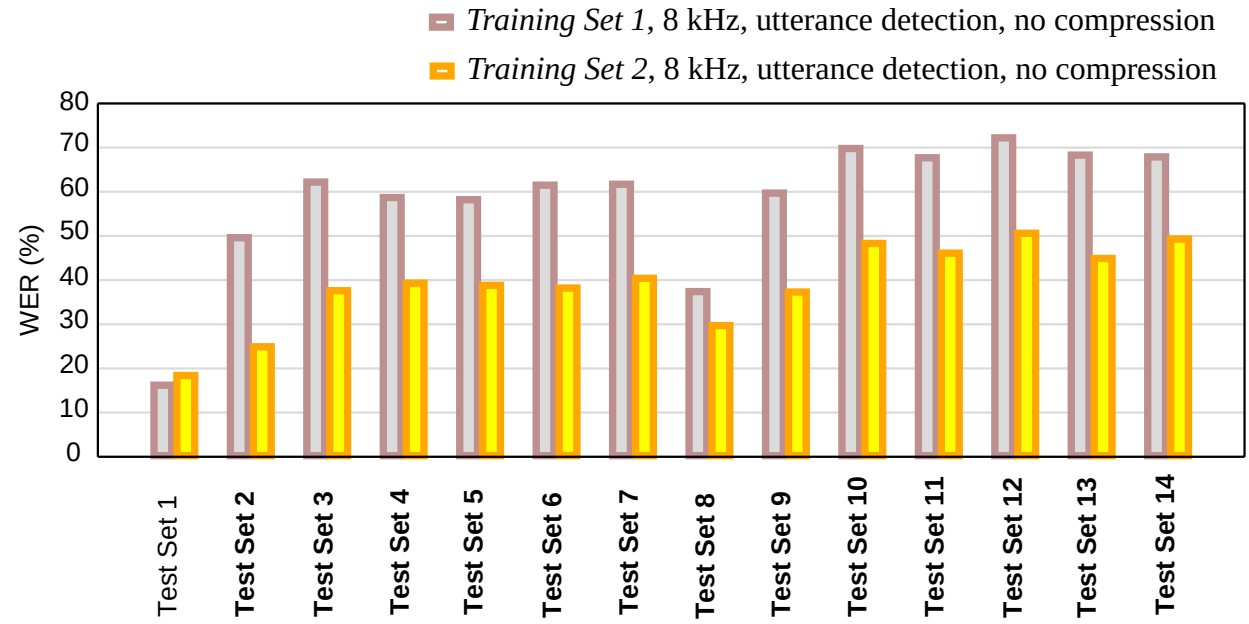


Figure 27(b). Comparison of the WER between *Training Set 1* and *Training Set 2* at 8 kHz with no feature value compression. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. Matched conditions without background noise and Sennheiser microphone (Training Set 1 and Test Set 1) resulted in the best recognition performance as one would expect. Training Set 1 is significantly worst than the Training Set 2 on all other test sets because it is a better match to the test conditions. Training Set 2 is representative of all the background noise and the microphone types.

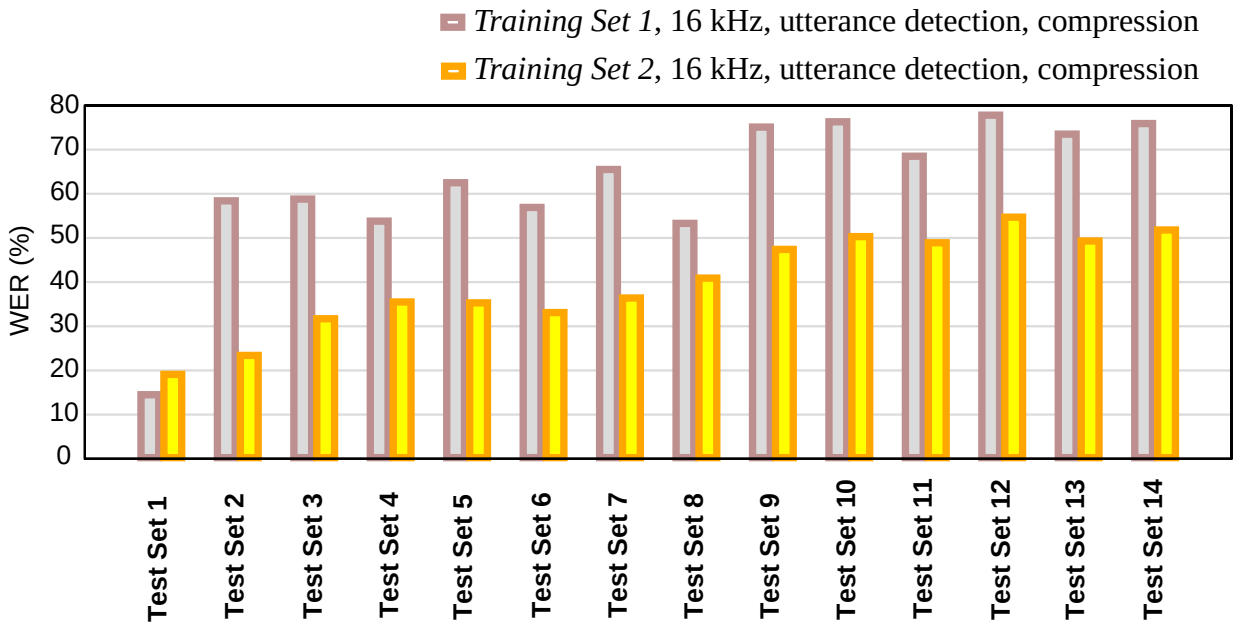


Figure 28(a). Comparison of the WER between *Training Set 1* and *Training Set 2* at 16 kHz with feature value compression. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. The trends are similar to the no compression case.

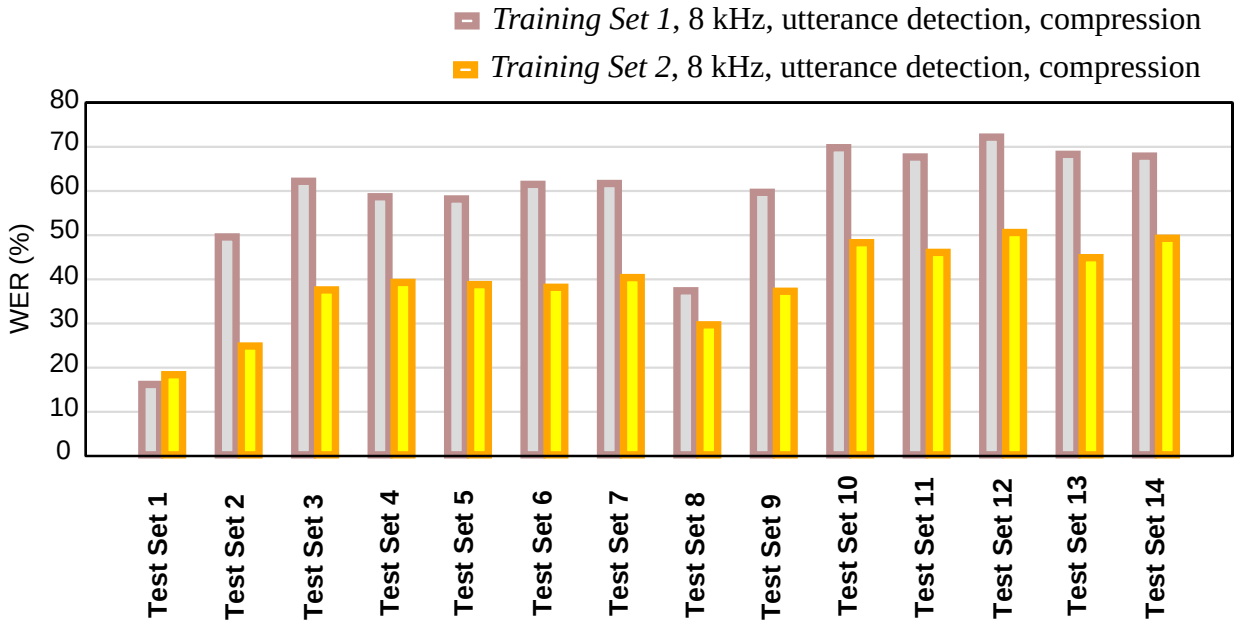


Figure 28(b). Comparison of the WER between *Training Set 1* and *Training Set 2* at 8 kHz with feature value compression. Test set conditions which are statistically significant at a 0.1% significance level are indicated by a boldface label. The trends are similar to the no compression case.

Performance (Without Compression)						
Training Set			Test Set			
Set	Sampling Frequency	Utterance Detection	1 (Sennheiser, Clean)	8 (Second, Clean)	2 (Sennheiser, Car)	9 (Second, Car)
1	8 kHz	Yes	16.2	37.4	49.6	59.7
2	8 kHz	Yes	18.4	29.7	24.9	37.3

Table 21. A significant performance degradation occurs for the second microphone condition on both training sets. No compression of feature values is employed. On Training Set 1, the models are representative of the Sennheiser microphone condition and hence, the performance drops significantly due to model mismatch on the remaining conditions. The models trained on Training Set 2, which have still been exposed to significant amounts of the Sennheiser data, do slightly better on the other noise conditions, but do not approach the matched training condition for a single microphone.

7.7. Additive Noise

In addition to calibrating the effects of on recognition performance of many signal processing issues such as sampling frequency reduction and utterance detection, we also calibrated recognition performance in presence of various background additive noise conditions. Figure 29 demonstrates the effect of these six noise conditions for two sample frequencies — 8 kHz and 16 kHz. As expected, severe degradation is observed at both sample frequencies.

However, the severity of this degradation can be limited by exposing the models to noise conditions during the training. In Figures 29(c) and (d), we demonstrate that the severity of the degradation in the noisy conditions is reduced by training the models on Training Set 2, which contains samples of the noise conditions. An important point to note is that these degradations are still significant compared to the clean condition. Similar trends were observed when the feature vectors were compressed.

8. COMPUTATIONAL ANALYSIS REVISITED

As discussed in Section 2, the SI-84 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 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: approximately 10 days 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

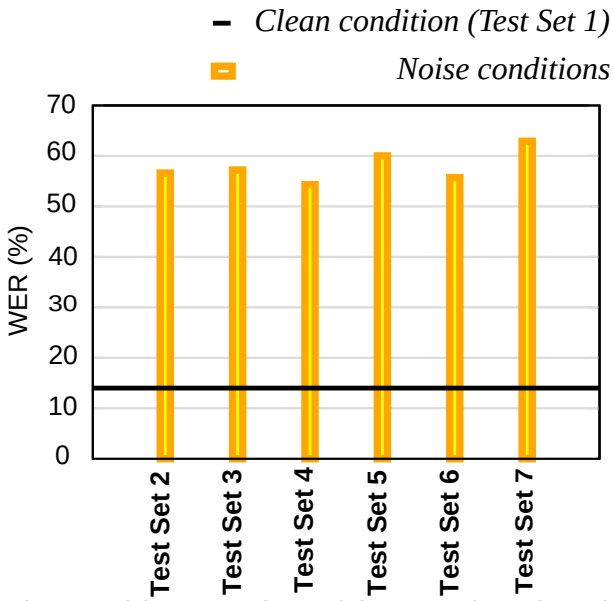


Figure 29(a). Comparison of the WER for selected noise conditions at *16 kHz* with *no feature value compression. Training Set 1* was used for training.

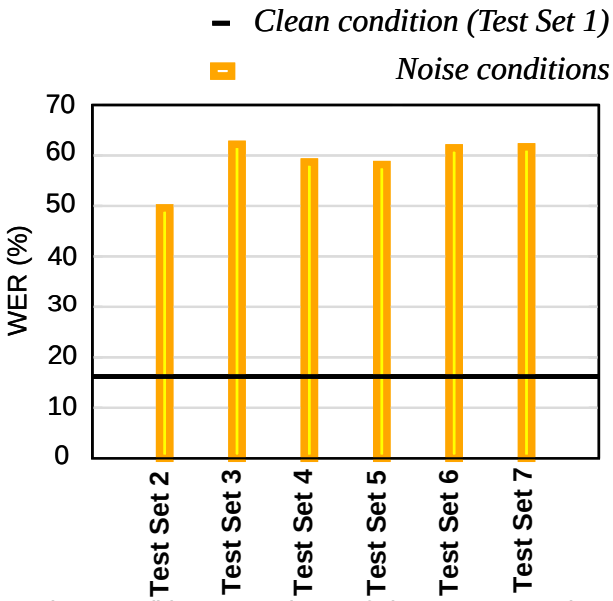


Figure 29(b). Comparison of the WER at *8 kHz* with *no feature value compression.*

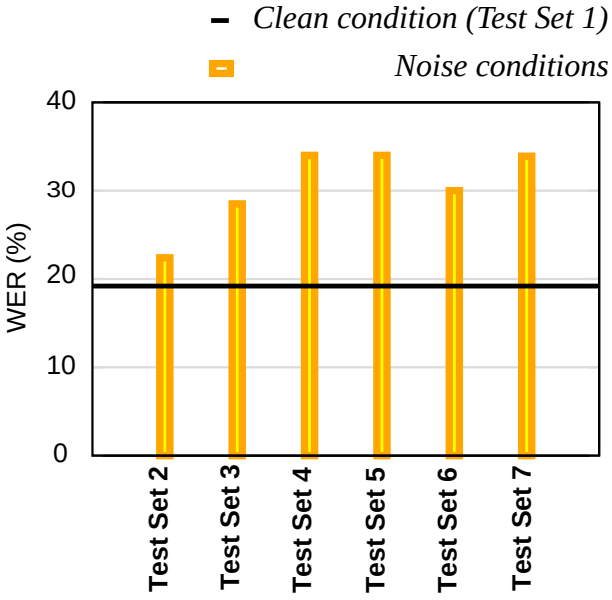


Figure 29(c). A similar comparison at *16 kHz* for *Training Set 2.*

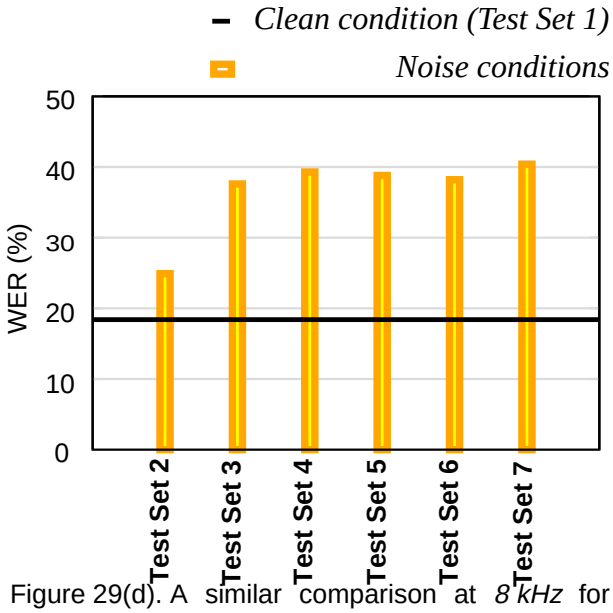


Figure 29(d). A similar comparison at *8 kHz* for *Training Set 2.*

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 [43]. A typical compute server 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 have recently acquired 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. 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 for Baseline system, 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 for the Baseline system. 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.

As discussed in Section 7.1, to further reduce the computational requirements for the experiments involving noisy data, we reduced the number of mixture components from 16 to 4. As anticipated, this reduction resulted in a drop in the total training time from 110 days to 86 days. However, the actual decoding time for the 14 noise conditions turned out to be approximately 203 days on a single 800 MHz Pentium processor, as compared to our estimate of 77 days. Note that this estimate of 77 days was based on projections from experimental runs involving only clean data. We estimated that the noisy data conditions would require three times longer than the clean data to decode. In reality, it required eight times longer to decode. When the acoustic models are not well-matched to the data, the recognizer retains a larger search space because the number of hypotheses that fall within the beam in the search process is larger. Accuracy and CPU time are often correlated — good acoustic matches allow the recognizer to be very selective in the paths it chooses to search. We also observed that the main memory requirements during decoding also increased to a maximum of about 900 MBytes due to an increase in the search space size.

9. CONCLUSIONS

The aim of the project was to generate the baselines for the noisy experiments for the Aurora Evaluations on the ETSI front-end [49]. We started by developing a baseline WSJ system that provides near state-of-the-art performance yet minimized complexity and maximizes ease of use. In the process of tuning the baseline WSJ system, we observed that excessive tuning does not significantly improve system performance. We began the tuning process by using the parameter values of our standard LVCSR system (based on the Switchboard Corpus). This operating point yielded a WER of 9.4%. Tuning this system yielded a WER of 8.0%. Most of this improvement is attributed to tuning of state-tying parameters (resulting in a WER of 8.6%). The remaining

improvement was achieved by tuning the language model scale and word insertion penalty parameters, as described in Section 4.

To enable the participating sites to run the evaluations, we released a series of packages including a package that allows a user to easily run complete experiments using multiple CPUs. We also released various short set definitions during the course of this project that were designed to provide a meaningful experimental result in a short amount of time [42]. We then proceeded to generate the baseline results for the Aurora evaluation that are shown in Tables 17 and 18.

We can draw several conclusions from these baseline experiments:

- **Sample Frequency:** increasing the sampling frequency from 8 kHz to 16 kHz resulted in a significant improvement in the performance only for the noisy test conditions.
- **Endpointing:** endpointing the data improved performance, as expected, only on the noisy test conditions. Endpointing resulted in a reduction of the insertion error rate. These improvements were not as visible on the multi-condition training because there is no increase in insertion errors on non-endpointed data. In this case, the “silence” model adapted to the channel conditions, and was able to control the insertion error rate.
- **Lossy Compression:** does not degrade performance. It can be argued that the fidelity of this compression technique is adequate for a recognition applications. The quality of compression algorithms has improved significantly in the past 20 years, perhaps crossing a threshold in which compression no longer contributes to poor recognition performance.
- **Model Mismatch:** performance improved when the training and the decoding conditions were matched — Training Set 1 and Test Set 1. Mismatched conditions (Training Set 1 and Test Sets 2-14) resulted in a significant degradation in performance.
- **Microphone Mismatch:** influences recognition performance in a similar fashion as the model mismatch conditions. The best performance is attained on matched Sennheiser microphone conditions (Training Set 1 and Test Sets 1-7), and performance degrades significantly for other mismatched microphone conditions (Training Set 1 and Test Sets 8-14). However, for the multi-condition training (Training Set 2), the performance also degraded on the second microphone conditions.
- **Additive Noise:** noise in test data degraded performance, even when the training conditions were exposed to similar amounts of noise. This is perhaps the most convincing argument for research into noise reduction techniques in the front end processing.

Note that all information related to the project, including this report and the related software, can be found at the project web site [42]: *http://www.isip.msstate.edu/projects/aurora*.

10. 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. This project in collaboration with ETSI’s DSR Aurora Working group has been very stimulating and thought provoking. The development of the baseline system enhanced our knowledge about the speech technology and the various parameters associated with the decoder. The Multi-CPU Eval package that we developed in the course of this project has been an excellent learning and starting-up tool for novice speech researchers that was used during our annual design workshops SRSDR02 and SRSTW02. The Aurora evaluations have helped us to

investigate issues involving the effects of a noisy environment, compression, sampling rate, microphone conditions, model mismatch on speech recognition performance. The knowledge we gained in these efforts will help us improve our technology for speech recognition in adverse conditions encountered during the wireless applications like mobile phones. These learning opportunities would not have occurred without funding and support from the DSR Aurora Working Group. We are also thankful to the ICSI and NIST for sharing with us the software for Significance tests. Finally, we would also like to acknowledge the users who provided us with valuable feedback regarding software bugs and features.

11. 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] F. Zheng and J. Picone, "Robust Low Perplexity Voice Interfaces," *http://www.isip.msstate.edu/projects/robust_low_perplexity/html/publications.html*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, December 2001.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] "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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] "Benchmark Tests, Matched Pairs Sentence-Segment Word Error (MAPSSWE)," *http://www.nist.gov/speech/tests/sigtests/mapsswe.htm*, Speech Group, NIST, USA, January 2002.

[45] N. Parihar and J. Picone, "Utterance Endpoints (v1.0)," *http://www.isip.msstate.edu/projects/aurora/html/downloads.html*, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA, October 2001.

[46] G. Hirsch and D. Pearce, "Second Experimental Framework for the Performance Evaluation of Speech Recognition Front-ends," STQ Aurora DSR Working Group, February 2000.

[47] V. Valtchev, Discriminative Methods in HMM-based Speech Recognition,Ph.D. dissertation, University of Cambridge, UK, 1995.

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

[49] "ETSI ES 201 108 v1.1.2 Distributed Speech Recognition; Front-end Feature Extraction Algorithm; Compression Algorithm," ETSI, April 2000.

[50] "Speech Recognition Scoring Toolkit (SCTK) Version 1.2c," *http://www.nist.gov/speech/tools/index.htm* Speech Group, NIST, USA, January 2001.

[51] R. Sundaram, "Effects of Transcription Errors on Supervised Learning in Speech Recognition," M.S. Dissertation, Institute for Signal and Information Processing, Mississippi State University, October 2000.

APPENDIX A. SUMMARY OF THE BASELINE SYSTEM AND EVALUATIONS

The objective of this project was two-fold — to develop an LVCSR system to be used by all sites participating in the Aurora evaluations, and to generate performance baselines for these evaluations using the ETSI-standard front end. In this summary, we briefly describe the baseline system and the data sets used for the evaluations.

The noisy baselines were generated on the noisy versions of the 5k DARPA Wall Street Journal (WSJ0) task [1], as described in Section 2. The evaluation data set consisted of two training sets and 14 test sets. Training Set 1 consisted of 7138 utterances belonging to the SI-84 subset of WSJ0. This data was filtered to simulate telephone quality speech. We refer to this as clean filtered speech. Training Set 2 consisted of the same filtered 7138 utterances, but three-fourths of these utterances were corrupted by one of the six noise conditions. In Training Set 2, half of the utterances were using a different microphone.

The 14 test sets are noisy versions of the 330 utterance **Nov'92 Eval Set**. Data from **seven noise conditions** (clean, car, babble, restaurant, street, airport and train-station) was digitally added to the clean data. To reduce computations, the 330 utterance set was reduced to **166 utterances**. Each of these seven tests sets was generated for both the microphone conditions. Hence, the final evaluation condition consisted of 14 sets of 166 utterances. These sets were generated at two sample frequencies: 8 kHz and 16 kHz.

The **ETSI standard** mel frequency scaled cepstral coefficient (MFCC) **front end** [49] was used to extract the features. From these features, **state-tied cross-word triphone acoustic models with 4 Gaussian mixtures per state** were generated using our standard LVCSR training procedure. One point to note is that the phone transcriptions required for training were generated using our best system (a 16-mixture version of the system described above) and fixed throughout the evaluation process. Monophone transcriptions were extracted from these triphone transcriptions. The lexicon was extracted from the CMU dictionary (version 0.6) [10] with some local additions to cover the 5k vocabulary. Recognition was performed using a single pass dynamic programming-based search guided by the **standard WSJ 5k backoff bigram language model** [8].

We then evaluated a total of 154 conditions that involved 7 noise types, 2 compression types, 3 training sets, 2 sample rates, 2 compression types and endpointing to benchmark the ETSI front end as shown in **Tables 17 and 18**. These conditions were evaluated using v5.12 of our prototype system (controlled by our multiple-CPU scripts). These evaluation results are described and analyzed in section 7. The total training time for generating models to support these experiments, which required 11 complete training runs, is 86 days of CPU time for an 800 MHz processor. The total decoding time, which required evaluating 14 noise conditions for 11 different model sets, required 203 days of CPU time for an 800 MHz processor. The peak memory utilization for the decoder, which is typically encountered during decoding of noisy utterances, was 900 Mbytes. Typical memory requirements for clean data is in the range of 300 MBytes.

Note that all information related to the project, including this report and the related software, can be found at the project web site [42]: *http://www.isip.msstate.edu/projects/aurora*.

APPENDIX B. SIGNIFICANCE TEST

The main goal of this project was to facilitate a comparison of front ends on a large vocabulary speech recognition task. To do this, we need to be able to say, definitively, that one front end performs better than another. It is tempting to conclude that if one system has a lower WER on a common task, then it must be better. However, there are many places in an experimental design in which noise can be introduced (e.g., computational noise from parallel processing, fluctuations due to a small evaluation data set). Thus, there is a need for a statistical measure which can help us determine if an experimental result is statistically significant.

The objective of statistical testing is to make inferences about a population given a finite sample set from the population. Since we lack complete knowledge of the population, we have a need to measure the uncertainty in our inferences. In statistics, this uncertainty is often referred as the probability of making an error (in our inference). Statistical testing provides a means of quantitative evaluation and control over the probability of making an error.

Statistical testing is often posed as a problem of hypothesis testing. A statistical hypothesis is speculation about the population's behavior or some established theory stated in terms of population parameters such as means and variances. Data is collected during an experiment, or set of experiments, and is assumed to be drawn according to the population distribution. Statistical tests provide a means for making judgements as to the truth of the statistical hypothesis based on the sample data. A control parameter known as the *significance level* is used to control the probability of making an error in the inference. The significance level measures our confidence in rejecting the hypothesis. The lower our confidence of rejection is, the more likely it is that our inference is true.

A common task in hypothesis testing is to compare statistics computed over samples of two distributions to determine how likely it is that the two distributions are equivalent. For example, we may want to compare the means and variances of two sampled distributions, each of which is assumed Gaussian with means μ_1 and μ_2, and variances σ_1^2 and σ_2^2, respectively. Consider the case for comparing the means of the two populations. We begin by forming the null hypothesis that the two means are equivalent:

- Null Hypothesis (H0): $\mu_1 = \mu_2$ or $\mu_1 - \mu_2 = 0$, (B.1)
- Alternate Hypothesis(H1): $\mu_1 \neq \mu_2$ or $|\mu_1 - \mu_2| > 0$. (B.2)

We randomly select n_1 samples from the first population and then draw n_2 samples independently from the second population. The difference between the two sample means $\overline{y_1} - \overline{y_2}$ is an unbiased point estimate of the difference of the true population means $\mu_1 - \mu_2$. According to the linear function of the random variables, the sampling distribution of statistic $\overline{y_1} - \overline{y_2}$ is a

normal distribution with a mean of $\mu_1 - \mu_2$ and variance of $\sigma_1^2/n_1 + \sigma_2^2/n_2$. Hence, the test statistic or z-statistic is given by

$$Z = \frac{\overline{y_1} - \overline{y_2}}{\sqrt{(\sigma_1^2/n_1) + (\sigma_2^2/n_2)}} \quad \text{(B.3)}$$

This test statistic's distribution can be approximated as a standard normal distribution shown in Figure B.1. A single right-tailed test can be used to reject the null hypothesis when $Z = z_p$ at a significance level of p. The rejection region or the probability of falsely rejecting the true null hypothesis (referred to as a Type I error) lies in the region from z_p to infinity. This region corresponds to the yellow region shown in Figure B.1.

The problem in our work is to specify an upper limit for performance (WER) for which a new design of a front end would be considered to be statistically significantly better than the baseline. Significance for proportions is suitable for such needs since WER is defined as a proportion. This leads to the same form as the z-test. An assumption is made that two experiments (baseline and new front end), each consisting of N independent trials, are run. To satisfy the independence assumption, it is necessary to consider each trial as the number of errors for each utterance in the corpus. This requires an assumption that the utterances in the corpus are independent of each other. For example, the utterances in the corpus should not be derived from discussions where one utterance is a response to another. We cannot use the words in the corpus as trials since we know that N-gram language models dictates that consecutive words are not independent of each other.

If, in our experiment, the first experiment resulted in y1 trials in error while the second experiment resulted in y2 trials in error, we can estimate the word error rates, p1 and p2, from a sample of size N in the sample population: $\hat{p}1 = y1/N$ and $\hat{p}2 = y2/N$. Since our goal is to determine if the second experiment WER, p2, is significantly better than the first experiment WER, p1, given N trials for each experiment, we consider the difference of the word error rates (proportions) to be zero as the null hypothesis, H0:

- Null Hypothesis (H0): $p1 = p2$ or $p1 - p2 = 0$, (B.4)
- Alternate Hypothesis (H1): $p1 \neq p2$ or $|p1 - p2| > 0$. (B.5)

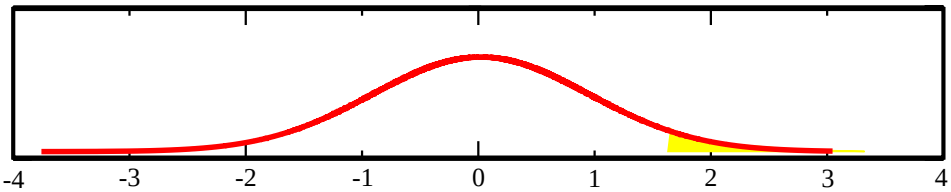


Figure B.1. A normal distribution with a mean of zero and a standard deviation of one. The hypothesis rejection region for a one-sided test is shaded in yellow.

To prove that the second experiment p2 is significantly better than the first experiment, we need to reject H0 at a given significance level. The normalized z-statistic for this test is given as:

$$Z = \frac{(\hat{p}1 - \hat{p}2)}{\sqrt{\left(\frac{\hat{p}1(1-\hat{p}1)}{N}\right) + \left(\frac{\hat{p}2(1-\hat{p}2)}{N}\right)}} \quad \text{(B.6)}$$

The assumption for this test is that according to the Central Limit Theorem, the distribution of this z-statistic is approximately normal given a large sample size. The single-tailed significance test is used to reject or fail to reject the null hypothesis. Note that the variance of $p1 - p2$ is estimated in the denominator of the equation above.

Let us consider a simple example to demonstrate these calculations. The baseline performance on the Short Test Set 1, which consists of 166 utterances, is 15.4% WER. Our goal was to specify the upper-limit of a WER that would be accepted as significantly better than the baseline. With N=166, p1=0.154, and a significance level of 1% (p=0.001), we iterate over decreasing values of p2 starting from 0.154 until the null hypothesis is rejected. It can be shown that when p2 reaches an error rate of 0.073 or a 7.3% WER, the z-statistic is given by

$$Z = \frac{(0.154 - 0.073)}{\sqrt{\left(\frac{0.154(1-0.154)}{166}\right) + \left(\frac{0.073(1-0.073)}{166}\right)}} = 2.35 \quad \text{(B.7)}$$

Since $z_{0.01} = 2.32 < 2.34$ and $z_{0.009} = 2.36 > 2.34$, we reject the null hypothesis at a 1% significance level. Similarly it can be shown that at a 10% significance level, the value of p2 is 0.106 or 10.6%. Thus, 7.3% WER and 10.6% WER specify the upper bound on significance results for the 1% and 10% significance levels, respectively.