Discriminative Training of Acoustic Models in a Segment-Based Speech Recognizer

by

Eric D. Sandness

S.B., Massachusetts Institute of Technology (2000)

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 19, 2000

Author ....
Department of Electrical Engineering and Computer Science
May 19, 2000

Certified by ....
I. Lee Hetherington
Research Scientist
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Discriminative Training of Acoustic Models in a Segment-Based Speech Recognizer

by

Eric D. Sandness

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2000, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis explores the use of discriminative training to improve acoustic modeling in a segment-based speech recognizer. In contrast with the more commonly used Maximum Likelihood training, discriminative training considers the likelihoods of competing classes when determining the parameters for a given class's model. Thus, discriminative training works directly to minimize the number of errors made in the recognition of the training data.

Several variants of discriminative training are implemented in the SUMMIT recognizer, and these variants are compared using results in the Jupiter weather information domain. We adopt an utterance-level training procedure. The effects of using different training criteria, optimizing various model parameters, and using different values of training control parameters are investigated. Consistent with previous studies, we find the most common objective criteria produce very similar results. We do, however, find that with our training scheme optimizing the mixture weights is much more effective than optimizing the Gaussian means and variances.

An extension to the standard discriminative training algorithms is developed which focuses on the recognition of certain keywords. The keywords are words that are most important for the proper parsing of an utterance by the natural language component of the system. We find that our technique can greatly improve the recognition word accuracy on this subset of the vocabulary. In addition, we find that choices of keyword lists which exclude certain unimportant, often poorly articulated words can actually result in an improvement in word accuracy for all words, not just the keywords themselves.

The accuracy gains reported in this thesis are consistent with gains previously reported in the literature. Typical reductions in word error rate are in the range of 5% relative to Maximum Likelihood trained models.

Thesis Supervisor: I. Lee Hetherington
Title: Research Scientist

Acknowledgments

I would first and foremost like to thank my thesis advisor, Lee Hetherington, for his help throughout this project. He was always available to help me with whatever problems came up and he provided me with many insightful suggestions. I would also like to thank Victor Zue and the entire Spoken Language Systems group for making this a wonderful place to conduct research. This group has given me a great deal and I appreciate it very much. I can not imagine a better place to have spent my time as a graduate student.

Finally, I would like to thank everyone who has helped me to survive (somehow) my years at MIT. My friends here have made my collegiate experience a fun and memorable one. While I am thrilled to finally be done, I will also miss the good times I have had at the Institute.

This research was supported by DARPA under contract DAAN02-98-K-0003, monitored through U.S. Army Natick Research, Development and Engineering Center; and contract N66001-99-8904, monitored through Naval Command, Control, and Ocean Surveillance Center.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Discriminative Training

1.1 Introduction

Modern speech recognition systems typically work by modeling speech as a series of phonemes. These phonemes are described using some sort of probability density functions, often mixtures of multivariate Gaussian components. Various recognition hypotheses are ranked by determining the likelihood of a set of input waveform measurements given the distributions of the hypothesized phonemes, combined with syllable and word sequence likelihoods imposed by the grammar of the language. The goal of the speech recognition process is to produce a transcription of the speaker's utterance with the minimum number of errors.

A major problem in building a successful speech recognition system is how to find the likelihood functions for each sub-word class. Given a set of transcribed training data, a set of modeling parameters must be found which will minimize the number of recognition errors. The standard approach is to train the model parameters using Maximum-Likelihood (ML) estimation. This approach aims to maximize the probability of observing the training data for each class given that class's parameter estimates. However, maximizing this probability does not necessarily correspond to minimizing the number of misclassifications. ML estimation does not take the likelihoods of incorrect classes into account during training, so potential confusions are ignored. Therefore, Maximum-Likelihood training does not always do a good job of discriminating near class boundaries.

Discriminative training aims to correct this deficiency by taking the likelihoods of potential confusing classes into account. Rather than trying to maximize the probability of observing the training data, it works to directly minimize the number of errors that are made in the recognition of the training data. More attention is given to data lying near the class boundaries, and the result is an improved ability to choose the correct class when the likelihood scores are close.

The goal of this thesis is to comparatively evaluate several discriminative training techniques in a segment-based speech recognition system. The next section provides an overview of previous research which serves as an introduction to some of the issues involved. Following that is a precise statement of the objectives of this thesis and an outline of the remaining chapters.

1.2 Previous Research

Much research has already been done on discriminative training of speech recognition parameters. Discriminative training procedures always employ an objective function which is optimized by some sort of update algorithm. The objective function should measure how well the current set of parameters classifies the training data. The update algorithm alters the system parameters to incrementally improve the objective score. The calculation of the objective function and subsequent alteration of the system parameters are repeated until the objective score converges to an optimum value.

The most common discriminative training criteria are Maximum Mutual Information (MMI) [21] and Minimum Classification Error (MCE) [7]. The basic aim of MMI estimation is to maximize the a posteriori probability of the training utterances given the training data, whereas MCE training aims to minimize the error rate on the training data. MMI focuses on the separation between the classes, while MCE focuses on the positions of the decision boundaries.

The standard MMI objective function [27] encourages two different optimizations for each training sample: the probability of observing the sample given the correct model is increased, and the probabilities of observing the sample for all other models are decreased. Classification errors will hopefully be corrected since the likelihood of the correct model for each sample is increased relative to the likelihoods of all other models. However, the amount of optimization is not dependent on the number of errors; even in the absence of classification errors on the training data significant optimization of the objective function can be performed. This is because MMI does not directly measure the number of classification errors on the training data, but rather the class separability.

By contrast, a wide variety of different objective functions are used for MCE estimation. MCE objective functions measure an approximation of the error rate on the training data. While a simple zero-one cost function would of course measure this error rate perfectly, it is a piecewise constant function and therefore cannot easily be optimized by numerical search methods. A continuous approximation to the error rate must be used instead. Some simple objective functions (perceptron, selective squared distance, minimum squared error) are discussed in [16]. In most practical MCE training procedures [7, 14, 16] a measure of misclassification is embedded into a smoothed zero-one cost function such as a translated sigmoid. Thus almost no penalty is incurred for data that is classified correctly, and data that is misclassified incurs a penalty that is essentially a count of classification error. MCE directly acts to correct classification errors on the training data; data that is classified correctly does not significantly influence the optimization process.

A comparison between the MMI and MCE criteria is done in [25]. It is shown that both criteria can be mathematically expressed using a common formula. The criteria are found to produce similar results in practice.

The most common update algorithms used to iteratively alter the system parameters are Generalized Probabilistic Decscent (GPD) and Extended Baum-Welch (EBW). GPD [6] is a standard gradient descent algorithm and is usually used to optimize MCE objective functions. EBW [13] is a procedure analogous to the Baum-Welch algorithm for ML training that is designed for MMI optimization. Both algorithms have step sizes which must be chosen to maximize the learning rate while still ensuring convergence. A comparison between the two optimization methods is done in [24], showing that the differences between the two are minimal.

Many variations of discriminative training have been implemented and reported in the literature. Numerous combinations of parameters to optimize, optimization criteria, and recognition frameworks are possible. In [3] the mixture weights of a system based on Hidden Markov Models (HMMs) are optimized using MMI with sentence error rates. In [27], the means and variances of the HMM state distributions are adjusted using MMI. Corrective MMI training, a variation on MMI in which only incorrectly recognized sentences contribute to the objective function, is used in [19]. A less mathematically rigorous version of corrective training is used in [2] to optimize HMM transition probabilities and output distributions. In [14], MCE training is used to compute means and variances of HMM state likelihoods. Relative reductions in word error rate of 5-20% compared to the same systems trained with the ML algorithm are typical.

Note that almost all of the previous research on discriminative training has been done using HMM-based recognition systems. By constrast, the research for this thesis will be performed using SUMMIT [10], a segment-based continuous speech recognition system developed by the Spoken Language Systems group at MIT. The opportunity to measure the effectiveness of discriminative training in a segment-based recognition framework is one of the original characteristics of the research that will be performed for this thesis.

1.3 Thesis Objectives

The principal goal of this thesis is to compare the effectiveness of various discriminative training algorithms in a segment-based speech recognizer. While many previous experiments have utilized somewhat simple acoustic models (e.g., few classes or few

Gaussian components per mixture model) the experiments in this thesis will evaluate recognition performance using the full-scale SUMMIT recognizer in the medium-vocabulary Jupiter domain. This should provide a more realistic view of the acoustic modeling gains that may be possible in real speech recognition systems. The data from these experiments will allow a direct comparison of the effectiveness of MCE and MMI training. Also, the relative merits of altering different system parameters (i.e., mixture weights versus means and variances) can be compared. Finally, it will be possible to observe the effects of using different values for several training parameters, such as the step size and the number of hypotheses used for discrimination.

Another goal of this thesis is to extend the existing body of discriminative training techniques for speech recognition. To this end a keyword-based discriminative training algorithm is developed. The goal of this technique is to shape the acoustic models to allow optimal recognition of the words that the system most needs in order to understand an utterance. This technique should reveal the extent to which the alteration of acoustic model parameters can be used to achieve specific recognition goals.

Chapter 2 describes the experimental framework for this thesis. The Jupiter domain is described and an overview of the SUMMIT speech recognizer is provided. A detailed discussion of the acoustic modeling component of the recognizer is given.

Chapter 3 describes the technique of Gaussian Selection, which is used to reduce test set evaluation times in this thesis.

Chapter 4 discusses the details of the discriminative training algorithm. The objective functions that will be used in this thesis are presented along with the algorithms used for parameter optimization. The implementation of discriminative training in SUMMIT is also discussed.

Chapter 5 presents results of various discriminative training experiments. Performance using both the MCE and MMI criteria for altering mixture weights, means and variances is evaluated.

Chapter 6 describes the new keyword-based discriminative training technique. Alterations to the standard discriminative training algorithm are introduced. Results of some experiments are presented which show that the keyword-based technique actually produces the best results reported in this thesis using either a keyword-error rate metric or the overall word-error rate.

Finally, Chapter 7 summarizes the lessons of this thesis and presents some suggestions for future work in this area.

Chapter 2

Experimental Framework

This chapter contains background information for the experimental work presented in this thesis. First, the Jupiter weather information domain, which is used for all experiments in this thesis, is discussed. Next, an overview of the SUMMIT speech recognition system is presented that illustrates how the acoustic modeling component fits into the recognizer as a whole. Finally, the acoustic modeling component is described in more detail.

2.1 The Jupiter Domain

Recognition experiments were performed using the corpus associated with the Jupiter telephone-based weather information system [30]. Users can call and receive weather information from various online sources over the phone. Jupiter has information on approximately 500 cities, 60% of them in the United States. The system can provide weather reports for a given day or for a period of several days. It can also answer specific questions about such weather phenomena as temperature, wind speed, precipitation, sunrise and sunset times, pressure, and humidity. An example dialogue between Jupiter and a real user is given in Table 2.1.

The Jupiter corpus contains quite a realistic body of data. Jupiter has a medium vocabulary of 1957 words, of which nearly half are geography-related words. The spoken utterances come mostly from calls to a toll-free number; almost all of these

Jupiter: Hi! Welcome to Jupiter, the MIT Lab for Computer Science weather information system. This call is being recorded for system development. You may hang up or ask for help at any time. How can I help you?
User: Can you tell me what's the weather like in Boston today? Jupiter: In Boston this afternoon. Partly sunny early. Then becoming cloudy with 30 percent chance of showers toward evening. Tonight: rain. Possibly heavy at times. Near 100 percent chance of rain. Can I help you with something else?
User: Yes what about San Francisco? Jupiter: In San Francisco today: patchy fog near the ocean morning. Otherwise mostly sunny. Tonight: areas of coastal fog and low clouds. Otherwise fair. Is there something else I can do for you?
User: Yes, how cold is it in Dallas Texas today? Jupiter: In Dallas today: high upper 40s. Tonight: lows near 32. What else would you like to know?
User: I'd like to know whether it's snowing in Detroit. Jupiter: I'm sorry, I didn't understand you. Please rephrase your query.
User: What is the weather in Detroit? Jupiter: In Detroit this afternoon: occasional light rain and drizzle. Tonight: rain showers likely. Becoming snow showers by late tonight. 70 percent chance of precipitation.

Table 2.1: Example of user dialogue with the Jupiter weather information system.

consist of spontaneous speech from untrained or self-trained users. Only spontaneous, in-vocabulary utterances are used for training in this thesis.

In this thesis, three sets of training data are used. The first, *train_6000*, is used for the majority of experiments with traditional discriminative training and contains 6000 utterances. The second, *train_12000*, is used for comparative purposes and contains 12000 utterances. The third, *train_18000*, contains about 18000 utterances and is only used for the keyword-based training experiments in Chapter 6. During training, the training sets are subdivided into two functional groups: eighty percent of the data is used for training, and twenty percent is set aside for development evaluations.

Two test sets are used. The first, *test_500*, is used for the majority of recognition experiments and contains 500 in-vocabulary utterances. For the more successful experiments, performance is further measured using the 2500 utterance set *test_2500*, which contains a full sampling of in-vocabulary and out-of-vocabulary utterances.

2.2 Overview of SUMMIT

SUMMIT is a segment-based continuous speech recognition system developed at MIT [10]. This type of recognizer attempts to segment the waveform into predefined subword units, such as phones, sequences of phones, or parts of phones. Frame-based recognizers, on the other hand, divide the waveform into equal-length windows, or frames [23]. The first step in either approach is to extract acoustic features at small, regular intervals. However, segment-based approaches next attempt to segment the waveform using these features, while frame-based recognizers use the frame-based features directly.

The result of the segmentation is a sequence of hypothesized segments together with their hypothesized boundaries. As discussed in section 2.2.2, the signal can either be represented with measurements taken inside the segments or measurements centered around the boundaries. In this thesis, only boundary-based measurements are used. Thus the recognizer must take a set of acoustic feature vectors $A = \{\vec{a_1}, \vec{a_2}, \cdots, \vec{a_N}\}$, where N is the number of hypothesized boundaries, and find the most likely string of words $\vec{w}^* = \{w_1, w_2, \cdots, w_M\}$ that produced the waveform. In other words,

$$\vec{w}^* = \arg\max_{\vec{w}} \Pr(\vec{w}|A) \tag{2.1}$$

where $\vec{w}$ ranges over all possible word strings. Each word string may be realizable as different strings of units $\vec{u}$ with different segmentations s, but since all hypotheses are evaluated on the same set of boundaries when using boundary-based measurements, the likelihood of s need not be considered when comparing the likelihoods of different word strings. Thus equation (2.1) becomes:

$$\vec{w}^* = \arg\max_{\vec{w}} \sum_{\vec{u}} \Pr(\vec{w}, \vec{u}|A) \tag{2.2}$$

where $\vec{u}$ ranges over all possible pronunciations of $\vec{w}$. To reduce computation SUMMIT assumes that, given a word sequence $\vec{w}$, there is an optimal unit sequence which is much more likely than any other $\vec{u}$. This replaces the summation by a maximization in which we attempt to find the best combination of word string and unit sequence given the acoustic features:

$$\{\vec{w}^*, \vec{u}^*\} = \arg\max_{\vec{w},\vec{u}} \Pr(\vec{w}, \vec{u}|A) \tag{2.3}$$

Applying Bayes' Rule, this can be rewritten as:

$$\begin{aligned} \{\vec{w}^*, \vec{u}^*\} &= \arg\max_{\vec{w},\vec{u}} \frac{\Pr(A|\vec{w}, \vec{u}) \Pr(\vec{u}|\vec{w}) \Pr(\vec{w})}{\Pr(A)} && (2.4) \\ &= \arg\max_{\vec{w},\vec{u}} \Pr(A|\vec{w}, \vec{u}) \Pr(\vec{u}|\vec{w}) \Pr(\vec{w}) && (2.5) \end{aligned}$$

The division by $\Pr(A)$ can be omitted since $\Pr(A)$ is constant for all $\vec{w}$ and $\vec{u}$ and therefore does not affect the maximization.

The estimation of the three components of the last equation is performed by, respectively, the acoustic, lexical, and language modeling components. The following four sections briefly describe the major components of the recognizer: segmentation, acoustic modeling, lexical modeling, and language modeling. Afterwards we describe how SUMMIT searches for the best path and the implementation of SUMMIT using finite-state transducers.

2.2.1 Segmentation

The essential element of the segment-based approach is the use of explicit segmental start and end times in the extraction of acoustic measurements from the speech signal. In order to implement this measurement extraction strategy, segmentation hypotheses are needed. In SUMMIT, this is done by first extracting frame-based acoustic features (e.g., MFCC's) at small, regular intervals, as in a frame-based recognizer. Next, boundaries between segments must be hypothesized based on these features. There are various ways to perform this task. One method is to place segment boundaries at points where the rate of change of the spectral features reaches a local maximum; this is called acoustic segmentation [9]. Another method, called probabilistic segmentation [17], uses a frame-based phonetic recognizer to hypothesize boundaries. In this thesis, acoustic segmentation is used.

2.2.2 Acoustic Modeling

There are two types of acoustic models that may be used in SUMMIT: segment models and boundary models [26]. Segment models are intended to model hypothesized phonetic segments in a phonetic graph. The observation vector is typically derived from spectral vectors spanning the segment. Segment-based measurements from a hypothesized segment network lead to a network of observations. For every path through the network, some segments are on the path and others are off the path. When comparing different paths it is necessary for the scoring computation to account for all observations by including both on-path and off-path segments in the calculation [5, 11].

In contrast, boundary models are intended to model transistions between phonetic units. These are diphone models with observation vectors centered at hypothesized boundary locations. Some of these landmarks will in reality be internal to a phone, so both internal and transition boundary models are used. Every path through the network incorporates all of the boundaries through either internal or transition boundary models, so the scores for different paths can be directly compared.

Boundary models are used exclusively in this thesis for two reasons. First, since every recognition hypothesis is based on the same set of measurement vectors, collecting statistics for discriminative training is much easier. Second, performance in Jupiter has been shown to be better using boundary models than when using context-independent segment models [26]. With considerably more computation, context-dependent triphone segment models can improve performance further.

SUMMIT assumes that boundaries are independent of each other and of the word string, so that

$$\Pr(A|\vec{w},\vec{u}) = \prod_{l=1}^{N} \Pr(\vec{a_l}|b_l) \tag{2.6}$$

where b_l is the lth boundary label as defined by $\vec{u}$. During recognition, the acoustic model assigns to each boundary in the segmentation graph a vector of scores corresponding to the probability of observing the boundary's features given each of the boundary labels. In practice, however, only the scores that are being considered in the recognition search are actually computed.

The details of how the likelihoods of a given measurement vector are computed for each boundary label are presented in section 2.3.

2.2.3 Lexical Modeling

Lexical modeling is the modeling of the allowable pronunciations for each word in the recognizer's vocabulary [29]. A dictionary of pronunciations, called the lexicon, contains one or more basic pronunciations for each word. In addition, alternate pronunciations may be created by applying phonological rules to the basic forms. The alternate pronunciations are represented as a graph, the arcs of which can be weighted to account for the different probabilities of the alternate pronunciations. The probability of a given path in the lexicon accounts for the lexical probability component in (2.5).

2.2.4 Language Modeling

The purpose of the language model is to determine the likelihood of a word sequence $\vec{w}$. Since it is usually infeasible to consider the probability of observing the entire sequence of words together, n-grams are often used to consider the likelihoods of observing smaller strings of words within the sequence [1]. n-grams assume that the probability of a given word depends on only $n-1$ preceding words, so that:

$$\Pr(\vec{w}) = \prod_{i=1}^{N} \Pr(w_i | w_{i-1}, w_{i-2}, \cdots, w_{i-(n-1)}) \tag{2.7}$$

where N is the number of words in $\vec{w}$. The probability estimates for n-gram language models are usually trained by counting the number of occurrences of each n-word sequence in a set of training data, although smoothing methods are often needed to redistribute some of the probability from observed n-grams to unobserved n-grams.

In all of the experiments in this thesis, bigram and trigram models (n=2 and 3, respectively) are used.

2.2.5 The Search

Together, the acoustic, lexical, and language model scores form a weighted graph representing the search space. The recognizer uses a Viterbi search to find the best-scoring path through this graph. Beam-pruning is used to reduce the amount of computation required. In the version of SUMMIT used in this thesis, the forward Viterbi search uses a bigram language model to obtain the partial-path scores to each node. This is followed by a backward A^* beam search using the scores from the Viterbi search as the look-ahead function. Since the Viterbi search has already pruned away much of the search space, a more computationally demanding trigram language model is used for the backward A^* search. This produces the final N-best recognition hypotheses.

2.2.6 Finite-State Transducers

The recognition experiments in this thesis utilized an implementation of SUMMIT using weighted finite-state transducers, or FSTs [12]. The FST recognizer R can be represented as the composition of four FSTs,

$$R = A \circ D \circ L \circ G \tag{2.8}$$

where A is a segment graph with associated acoustic model scores, D performs the conversion from the diphone labels in the acoustic graph to the phone labels in the lexicon, L represents the lexicon, and G represents the language model. Recognition then becomes a search for the best path through R. The composition of D, L, and G can be performed either during or prior to recognition. In this thesis $D \circ L \circ G$ is computed as $D \circ \min(\det(L \circ G))$, and the composition with A occurs as part of the Viterbi search.

2.3 The Acoustic Modeling Component

As seen in the previous section, the job of the acoustic modeling component of the recognizer is to compute $\Pr(A|\vec{w}, \vec{u})$. This reduces to a product of individual terms of the form $\Pr(\vec{a_l}|b_l)$, where b_l is the lth boundary label defined by $\vec{u}$. This section describes how the probability density functions for each boundary label are modeled, and how the model parameters are trained given a set of training data.

2.3.1 Acoustic Phonetic Models

Acoustic phonetic models are probability density functions over the space of possible measurement vectors, conditioned on the identity of the phonetic unit. Normalization and principal components analysis are performed on the measurement vectors prior to modeling. Then the whitened vectors are modeled using mixtures of diagonal Gaussian models, of the following form:

$$p(\vec{a}|u) = \sum_{i=1}^{M} w_i p_i(\vec{a}|u) \tag{2.9}$$

where M is the number of mixture components in the model, $\vec{a}$ is a measurement vector, and u is the unit being modeled. Each $p_i(\vec{a})$ is a multivariate normal probability density function with no off-diagonal covariance terms, whose value is scaled by a weight w_i. The mixture weights must satisfy

$$\sum_{i=1}^{M} w_i = 1 \tag{2.10}$$

$$0 \leq w_i \leq 1 \tag{2.11}$$

Thus the parameters used to model a given phonetic unit are M weights, Md means, and Md variances, where d is the dimensionality of the whitened measurement vector.

The score of an acoustic model is the value of the mixture density function at the given measurement vector. For practical reasons, the logarithm of this score is used during computation, resulting in what is known as the log likelihood score for the given measurement vector.

2.3.2 Maximum-Likelihood Training

Before the Gaussian mixture models can be trained, a time-aligned phonetic transcription of each of the training utterances is needed. These are automically generated from manual word transcriptions using forced phonetic transcriptions, or forced paths. Beginning with a set of existing acoustic models, we perform recognition on each training utterance using the known word transcription as the language model. Next, the principal components analysis coefficients are trained from the resulting measurement vectors, producing a set of labeled 50-dimensional vectors. The data are separated into different sets corresponding to each lexical unit.

Training a Gaussian mixture model for a given unit is a two-step process. In the first step, the K-means algorithm [8] is used to produce an initial clustering of the data. In this thesis the number of clusters generated (i.e., the number of mixture components) is either 50 or the number possible with at least 50 training tokens per cluster, whichever is smaller. This ensures that each mixture component is trained from sufficient data. In the second step, the results of the K-means algorithm are used to initialize the Expectation-Maximization (EM) algorithm [8]. The EM algorithm iteratively maximizes the likelihood of the training data and estimates the parameters of the mixture distribution. It converges to a local maximum; there is no guarantee of achieving the global optimum. The outcome is dependent on the initial conditions obtained from the K-means algorithm.

The result of the training is a set of model parameters which maximizes the likelihood of observing each class's training data given the parameters. Note that only the training data belonging to a given class affects that class's parameter estimates. Discriminative training aims to incorporate additional knowledge by also using the data associated with potential confusing classes.

Chapter 3

Gaussian Selection

This chapter discusses the technique of Gaussian Selection [4, 18]. First, the motivation for using Gaussian Selection and mathematical formulation of the technique are presented. Next, the results of applying Gaussian Selection to the acoustic models in SUMMIT are discussed. Finally, the most important features of Gaussian Selection are summarized.

3.1 Description of Gaussian Selection

Gaussian Selection is a technique which reduces the time required to estimate acoustic likelihood scores from Gaussian mixture models. This section discusses the motivation for using this technique and the details of its implementation.

3.1.1 Motivation

In a real-time conversational system such as Jupiter, not only the accuracy but also the computational requirements of the acoustic models are critical. Overall, up to 75% of the recognizer's time is spent evaluating the likelihoods of individual Gaussians in the various Gaussian mixture models. Thus it is necessary to keep the number of Gaussians evaluated as low as possible in order to perform recognition in real time. Previously, techniques such as beam pruning during the Viterbi search have been used to reduce the number of scores that need to be computed, but these techniques result in a large degradation in recognition accuracy. We need to find a way to prune away computations that do not have much effect on recognition accuracy.

In practice, the majority of the individual Gaussian likelihoods are completely negligible. The total likelihood score for a Gaussian mixture model is the sum of these individual likelihoods. There are usually a few Gaussians which are "close" to a given measurement vector and have much higher likelihoods than the rest of the Gaussians. Thus, the score for the mixture is approximately equal to the sum of the these few likelihoods. The likelihoods for the rest of the Gaussians do not need to be computed to maintain a high degree of accuracy in the overall score.

The motivation behind Gaussian Selection is to efficiently select the mixture components that are close to a given measurement vector, so that only the likelihoods for these Gaussians must be computed. The algorithm for accomplishing this is described next.

3.1.2 Algorithm Overview

To evaluate a model, the feature vector is quantized using binary vector quantization, and the resulting codeword is used to look up a list of Gaussians to evaluate. The lists of Gaussians to be evaluated for each model m and codeword k are computed beforehand.

A Gaussian is placed in the shortlist for a given m and k if the distance from its mean to the codeword center is below some threshold Θ. The distance criterion used is the squared Euclidean distance normalized by the number of dimensions D. Thus we select Gaussian i in model m if

$$\frac{1}{D}\sum_{d=1}^{D}(\mu_{m,i}(d) - c_k(d))^2 \leq \Theta \tag{3.1}$$

There are two other ways a Gaussian can be placed in shortlist (m, k). A Gaussian is automatically placed in the list if its mean quantizes to k. This means that every Gaussian will be selected for at least one codeword. If m has no Gaussians associated with k based on the above criteria, the closest Gaussian is placed in the list. This ensures that at least one Gaussian will be evaluated for every model/codeword pair, preserving a reasonable amount of modeling accuracy.

During model evaluation likelihood scores are computed for the Gaussians at each of the indices specified in the shortlist. Only these scores are combined to produce the total likelihood score for the model.

3.2 Results in SUMMIT

The effectiveness of Gaussian Selection was evaluated in Jupiter. Two similar metrics of computational load were used. The first metric is the computation fraction as defined in [4] and [18]:

$$C = \frac{N_{new} + VQ_{comp}}{N_{full}}, \tag{3.2}$$

where N_{new} and N_{full} are the average number of Gaussians evaluated per feature vector in the systems with and without Gaussian Selection, respectively, and VQ_{comp} is the number of computations required to quantize the feature vector. The second metric is a ratio of the overall recognition times with and without Gaussian Selection. Word accuracy in Jupiter is plotted against these metrics in figures 3-1 and 3-2. Curves are plotted for codebook sizes of 256, 512, 1024, and 2048. Each curve is swept out by varying the distance threshold Θ. The data for these curves was collected running the recognizer on a set of 500 test utterances.

As can be observed from the figures, very little recognition accuracy is lost for values of Θ down to 0.7. Lowering Θ further causes accuracy to degrade quite rapidly. Performance does not appear to be terribly sensitive to the choice of codebook size.

At the optimum point for this data (512 codewords, $\Theta = 0.7$), a 67% reduction in Gaussian evaluations results in an overall reduction of recognition time of 45%, a speedup by a factor of 1.8. Thus it is obvious that reducing computation in the acoustic modeling component has a large, direct impact on the speed of the recognizer, with little degradation of accuracy.

We have tried the more complex weighted distance criteria described in [18], but

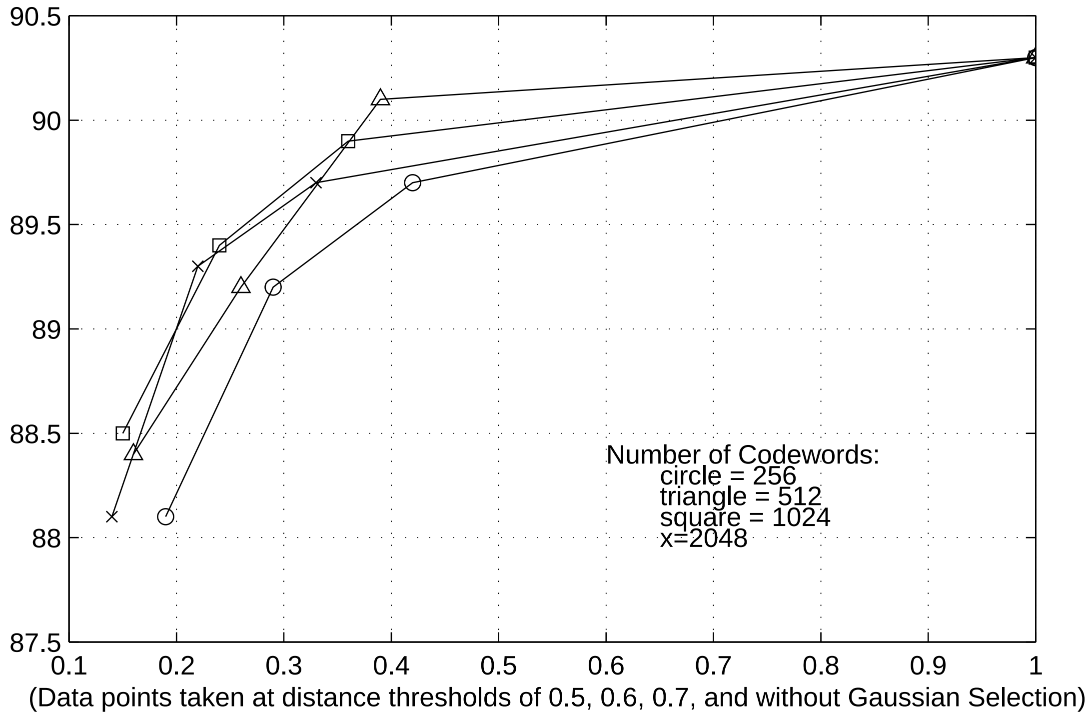

Figure 3-1: Test Set Accuracy vs. Gaussian Logprob Computation Fraction

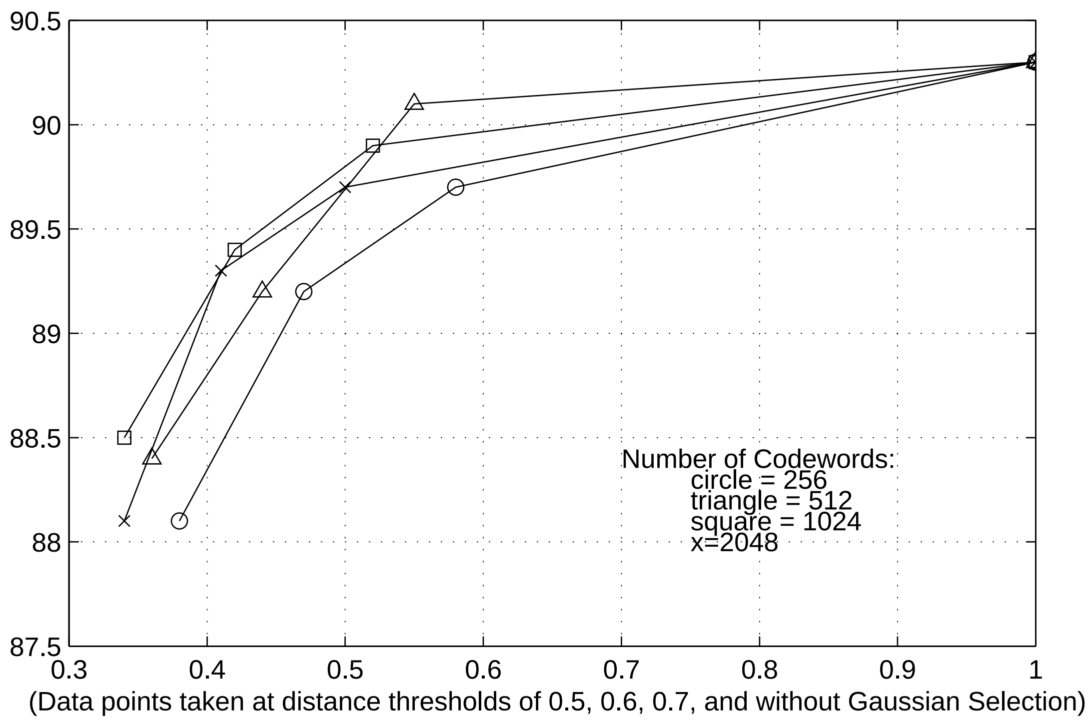

Figure 3-2: Test Set Accuracy vs. Overall Recognition Time Ratio

these did not offer any advantages in our system. We also tried using a Gaussian log probability in place of squared distance without noticing any improvement. Finally, we tried limiting the size of the shortlists as in [18], but this did not improve upon the speed/accuracy tradeoff [26].

3.3 Summary

This chapter discussed the technique of Gaussian Selection. By vector quantizing the feature vector and only evaluating Gaussians which are within some distance threshold from the codeword center, the computation required to score acoustic models can be greatly reduced. In SUMMIT, we found that the overall recognition time can be decreased by 45% with a very small decrease in word accuracy.

Gaussian Selection has now been incorporated into the mainstream SUMMIT recognizer; for an example of how it is used see [26]. Since this thesis aims to use the most realistic recognition conditions possible, Gaussian Selection is used in all recognition experiments. One major reason for discussing Gaussian Selection here is to emphasize that this introduces a small amount of extra uncertainty in the test set accuracies. For example, a gain obtained from altering the acoustic models may be at least partially accounted for by a coincidental decrease in errors caused by Gaussian Selection in the new acoustic space. However, we have seen that Gaussian Selection is not terribly sensitive to the exact values of the parameters in the models; thus, for the incremental changes used in discriminative training, the extra uncertainty in comparing accuracies should be quite small.

Chapter 4

Implementing Discriminative Training in SUMMIT

This chapter discusses the procedures used for discriminative training in this thesis. The training process requires a body of training data, an objective function, and a parameter optimization method. The training data is first scored using the current classifier parameters. The objective function then takes in these scores and provides a measure of classification performance. The parameter optimization method provides a way to alter the chosen parameters in order to maximize or minimize the objective function. If the objective function is a good measure of classifier accuracy, then optimizing it will minimize the number of recognition errors made. The first section describes two different ways of dividing up the training data and computing scores to be fed into the objective function. The next two sections describe the objective functions and parameter optimization formulas used in this thesis. Finally, we give an overview of the implementation of the training process as a collection of routines in SUMMIT.

4.1 Phone- Versus Utterance-Based Training

The first decision that must be made is how to divide up the training data into functional units that can be scored using the classifier parameters. The two possibilities considered here are the use of phone-level scores and the use of utterance-level scores.

If a phone-level discriminative criterion is to be used, the training data is broken down into a set of labeled phonemes. Each observation vector is sorted into a group based on its assigned label. The scores to be fed into the objective function are the pure classifier scores for each observation vector. The correct score is the score for the model corresponding to the vector's label, and the competing scores are the scores for all of the other models. This criterion attempts to maximize the acoustic score of the correct phone model for each observation vector independent of the context from which it was derived.

With an utterance-level discriminative criterion, on the other hand, the training data is organized into utterances. Each utterance consists of a sequence of observation vectors. The scores to be fed into the objective function are the complete recognizer scores for each utterance, i.e., the sum of the acoustic, lexical, and language model scores. The correct score is the recognition score for the actual spoken utterance, and the competing scores are the scores for all other possible utterances. Of course, it is not practical to train against every possible utterance, so an N-best list of competing utterances is used. This criterion attempts to adjust the acoustic models to maximize the chances of the correct utterance hypothesis being chosen, taking into account the effects of the non-acoustic components of the recognizer.

Some preliminary tests were run using several of the objective functions and optimization methods to be discussed in the next two sections. While phone-level discriminative training decreased the number of classification errors made on unseen observation vectors, it actually *increased* word error rates when the updated acoustic models were incorporated into the recognizer. We believe this to be due to the fact that the assignment of the correct label to a given boundary does not necessarily occur when $g_c - g_h > 0$, where g_c and g_h are the acoustic scores for the correct and best competing models, respectively. Instead, proper assignment occurs when $g_c - g_h > \Theta$, where Θ is some threshold determined by the relative non-acoustic scores of the hypotheses differing only at that boundary. In fact, Θ can be $-\infty$ if linguistic constraints make the competing label impossible at the boundary. Thus,

Models	Word Accuracy
ML Models	89.6
Phone-Level Training	89.2
Utterance-Level Training	90.2

Table 4.1: Word Accuracies for Preliminary Test Comparing Phone-Level and Utterance-Level Criteria

phone-level discriminative training can make many alterations that do not improve the recognizer and, in fact, may degrade its accuracy. Utterance-level discriminative training, which takes non-acoustic scores into consideration when altering the acoustic parameters, improved both classifier performance and the word accuracy of the recognizer. In light of this, utterance-level scores were chosen for use in this thesis.

Table 4.1 gives word accuracies from one of the preliminary tests illustrating the superiority of the utterance-level criterion. These accuracies were obtained training the mixture weights with an MCE objective function (to be discussed in the next section). Exactly ten training iterations were performed, and the resulting models were evaluated using the *test_500* set. Clearly, training with a phone-level criterion has a negative impact on word accuracy, while the utterance-level criterion produces the desired effect.

Thus, one set of scores is passed to the objective function for each utterance: the combined recognition score for the correct hypothesis, and recognition scores for the N hypotheses in some N-best list. This set of scores is very similar to that which was used for sentence-level training in [3].

4.2 The Objective Function

A variety of different objective functions are possible in discriminative training. There are two primary characteristics that are desirable in any choice of objective function. First, as the function's value is improved, the classifier performance should also improve. Second, it should be continuous in the parameters to be optimized so that numerical search methods can be employed. Most discriminative training algorithms reported in the literature use a variant on one of two broad classes of objective functions: a Minimum Classification Error (MCE) discriminant or a Maximum Mutual Information (MMI) discriminant. Descriptions of these classes of objective functions as well as the particular realizations that are used in this thesis are given next.

4.2.1 The MCE Criterion

A large variety of different objective functions are classified into the category of MCE discriminants. The defining characteristic of this group is that the functions are designed to approximate an error rate on the training data. A simple zero-one cost function would measure this error rate perfectly, but it violates the constraint that the function should be continuously differentiable in the classifier parameters. As noted in Chapter 1, most practical MCE training procedures in the literature use a measure of misclassification embedded into a smoothed zero-one cost function. An example of a common misclassification measure [16] is:

$$d_s(X_s, \Lambda) = -g_{c,s}(X_s, \Lambda) + \left[\frac{1}{N_{h,s}} \sum_{h=1}^{N_{h,s}} g_{h,s}(X_s, \Lambda)^{\eta} \right]^{\frac{1}{\eta}} \tag{4.1}$$

where X_s denotes the sequence of acoustic observation vectors for the sentence, Λ represents the classifier parameters, $g_{c,s}$ is the log recognizer score for the sentence's correct word string, $g_{h,s}$ is a log recognizer score for a competing hypothesis, and $N_{h,s}$ is the number of competing hypothesis in the sentence's N-best list. From this point forward $g_{c,s}$, $g_{h,s}$, and $N_{h,s}$ are shortened to g_c, g_h, and N_h for notational convenience, but it should be remembered that these are all different for every sentence. η is a positive number whose value controls the degree to which each of the competing hypothesis is taken into account.

This misclassification measure can be embedded in a number of smoothed zero-one cost functions. In this thesis we choose:

$$f_s(d_s) = \frac{1}{1 + e^{-\rho(d_s + \alpha)}} \tag{4.2}$$

where ρ is the rolloff, which defines how sharply the function changes at the transition point, and α defines the location of the transition point. In this thesis α is always set to zero. The contribution of each utterance to the total objective score is given by f_s.

It turns out that the misclassification measure of Equation (4.1) is not very sensitive to the value of η for the numbers we use. For simplicity, then, η is set to 1. After moving the g_c term inside the sum and substituting the new formula for d_s into the equation for f_s, we get:

$$f_s(X_s, \Lambda) = \frac{1}{1 + e^{\rho \frac{1}{N_h} \sum_{h=1}^{N_h} (g_c(X_s,\Lambda) - g_h(X_s,\Lambda))}} \tag{4.3}$$

Equation (4.3) is still just the cost function of [16], with η set to 1 and α set to 0. We decided to experiment with another equation similar to (4.3), but with the sum over competing hypotheses done outside instead of inside the sigmoid:

$$f_s(X_s, \Lambda) = \frac{1}{N_h} \sum_{h=1}^{N_h} \frac{1}{1 + e^{\rho(g_c(X_s,\Lambda) - g_h(X_s,\Lambda))}} \tag{4.4}$$

This function still ranges from 0 to 1, but it allows each competing hypothesis to make a more individual contribution to the value. In the limiting case as $\rho \to \infty$, the function in Equation (4.3) can only take on two distinct values (0 and 1), but the function in Equation (4.4) can take on $(N_h + 1)$ different values $(\frac{k}{N_h}, 0 \leq k \leq N_h)$. For moderate values of ρ, however, equations (4.3) and (4.4) produce very similar behaviors. This is illustrated in figure 4-1, which traces the curves of (4.3) and (4.4) with typical values for the competing scores. The only difference between the two curves is that the one defined by (4.4) transitions a bit more slowly. We choose to use the form in (4.4) since it is easier to calculate the derivative when the sum is on the outside.

To get the complete objective function, the cost functions of all of the training utterances are averaged together. Thus, the formula for the MCE objective function

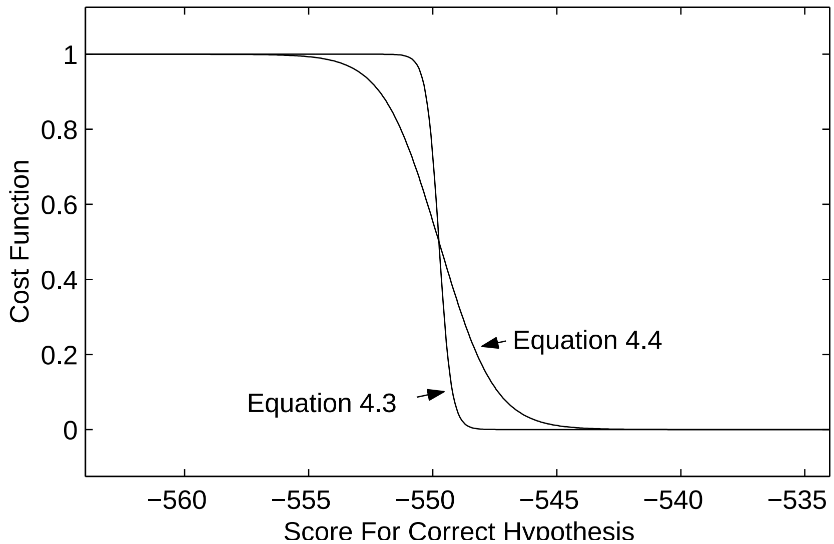

Figure 4-1: Comparison of cost functions of equations (4.3) and (4.4). Four competing hypotheses are used, whose scores are fixed at -549.0, -549.1, -550.3, and -550.6. This is a typical score pattern that might be observed for the top 4 hypotheses in an N-best list. ρ is set to 4.0. The curves are swept by varying the score of the correct hypothesis between -564.0 and -534.0.

is:

$$\mathcal{F}_{MCE}(\Lambda) = \frac{1}{N_s} \sum_{s=1}^{N_s} f_s(X_s, \Lambda) \tag{4.5}$$

$$= \frac{1}{N_s} \sum_{s=1}^{N_s} \frac{1}{N_h} \sum_{h=1}^{N_h} \frac{1}{1 + e^{\rho(g_c(X_s,\Lambda) - g_h(X_s,\Lambda))}} \tag{4.6}$$

where N_s is the total number of training utterances. This is the function we will use for MCE training experiments in this thesis.

4.2.2 The MMI Criterion

Unlike the MCE case, the MMI criterion comes with one standard objective function [27]. In place of a cost function to be minimized, there is a likelihood ratio to be maximized. This ratio is given by:

$$\ell_s(X_s, \Lambda) = \log \frac{p_c(X_s, \Lambda)}{p_c(X_s, \Lambda) + \sum_{h=1}^{N_h} p_h(X_s, \Lambda)} \tag{4.7}$$

where p_c and the p_h's are the linear versions of the recognizer scores, i.e.:

$$g_h(X_s, \Lambda) = \log p_h(X_s, \Lambda) \tag{4.8}$$

The log is used in the likelihood ratio to avoid computing with very small numbers. In terms of the log likelihood scores that the recognizer actually produces, (4.7) can be rewritten as:

$$\ell_s(X_s, \Lambda) = g_c(X_s, \Lambda) - log(e^{g_c(X_s,\Lambda)} + \sum_{h=1}^{N_h} e^{g_h(X_s,\Lambda)}) \tag{4.9}$$

In order to compare the behavior of ℓ_s in the MMI criterion with the behavior of f_s in the MCE criterion, figure 4-2 shows ℓ_s plotted as a function of g_c for the same competing hypotheses as in figure 4-1. As can be seen in the figure, ℓ_s is approximately linear up to the point when the correct score is slightly greater than the best competing score. At this point ℓ_s levels off and asymptotically approaches 0. Note that because of the logarithm, there is no lower bound on ℓ_s, which implies two things. First, ℓ_s can vary over a much larger range than f_s from the MCE case. Second, when the score for the correct hypothesis is much worse than the scores for competing hypotheses, ℓ_s can still have a significant derivative with respect to changes in parameter values. Thus MMI will encourage optimizations to improve very poorly scoring utterances, in contrast to MCE which is relatively insensitive to these utterances. This can have a negative effect on error rates if these optimizations never improve the targeted utterances enough to actually correct any errors, while at the same time detracting from optimizations meant to improve more typical utterances. In order to keep MMI training from encouraging these optimizations, a score threshold τ can be introduced. The highest score among the correct and competing hypotheses is the best score. Any of the other scores for which $p_h < \tau p_{\text{best}}$ are treated as if they were τp_{best}. In other words, a floor is placed on the score values. When small changes are made to these score values, they will still be treated as τp_{best}, and thus the objective score will no longer be sensitive to very poorly scoring hypotheses. Figure 4-2 also shows the behavior of ℓ_s when a score threshold is used.

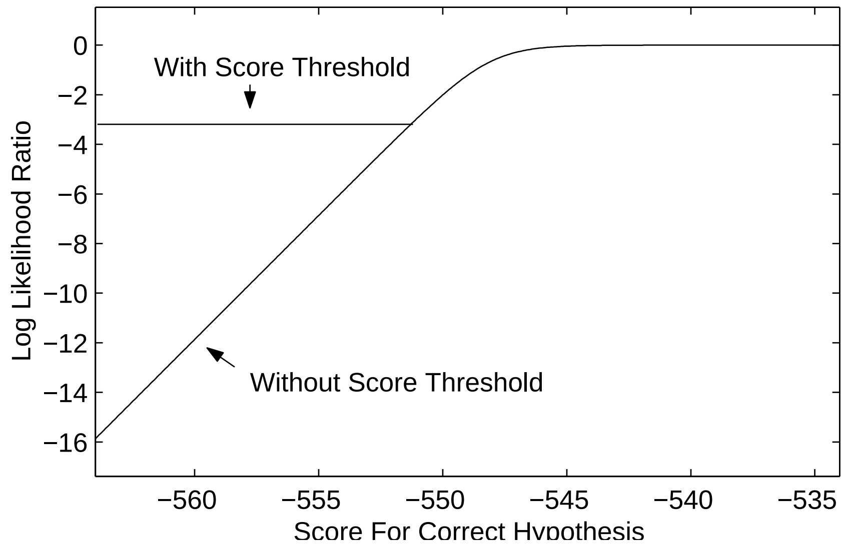

Figure 4-2: Plot of MMI log likelihood function of equation (4.9), with and without a score threshold. As in figure 4-1, four competing hypotheses are used, whose scores are fixed at -549.0, -549.1, -550.3, and -550.6. τ is set to 0.1. Again, the curves are swept by varying the score of the correct hypothesis between -564.0 and -534.0.

To get the complete objective function, the log likelihood functions of all the training utterances are averaged together. Thus, the final formula for the MMI objective function is:

$$\mathcal{F}_{MMI}(\Lambda) = \frac{1}{N_s} \sum_{s=1}^{N_s} \ell_s(X_s, \Lambda) \tag{4.10}$$

$$= \frac{1}{N_s} \sum_{s=1}^{N_s} \left(g_c(X_s, \Lambda) - log(e^{g_c(X_s,\Lambda)} + \sum_{h=1}^{N_h} e^{g_h(X_s,\Lambda)}) \right) \tag{4.11}$$

4.3 Parameter Optimization

Once an objective function is chosen, a procedure is needed for finding the parameter set Λ that optimizes its value. The nature of the problem suggests the use of a gradient-based approach. Two standard methods based on gradient search are discussed here: Generalized Probabilistic Descent (GPD) and Extended Baum-Welch (EBW) [24]. The search formulae are slightly different depending on whether mixture weights or means and variances are to be adjusted. The next two sections describe optimization for each of these parameter sets.

4.3.1 Adjusting the Mixture Weights

As discussed above, we will examine two different search techniques. The first, GPD, is a standard gradient descent algorithm usually used to optimize MCE objective functions. The GPD formula for updating the set of mixture weights $\vec{w}$ is [6]:

$$\hat{\vec{w}} = \vec{w} + \epsilon \nabla_{\vec{w}} \mathcal{F} \tag{4.12}$$

where ϵ is a constant that governs the learning rate and $\nabla_{\vec{w}} \mathcal{F}$ is the instantaneous gradient vector of $\mathcal{F}$. ϵ is positive for MMI training and negative for MCE training, since $\mathcal{F}$ is to be maximized in MMI and minimized in MCE. Thus, each individual weight is updated according to:

$$\hat{w}_{i,j} = w_{i,j} + \epsilon \frac{\partial \mathcal{F}}{\partial w_{i,j}} \tag{4.13}$$

where $w_{i,j}$ is the jth Gaussian of the ith mixture.

The second search technique, EBW, is a procedure analogous to the Baum-Welch algorithm for ML training that is designed for MMI optimization. The EBW formula for updating the mixture weights is [27]:

$$\hat{w}_{i,j} = \frac{w_{i,j} \left\{ \frac{\partial \mathcal{F}}{\partial w_{i,j}} + C \right\}}{\sum_{\hat{j}} w_{i,\hat{j}} \left\{ \frac{\partial \mathcal{F}}{\partial w_{i,\hat{j}}} + C \right\}} \tag{4.14}$$

The constant C governs the learning rate in this equation, and it is chosen such that all parameter derivatives are positive.

We automatically chose GPD to optimize the MCE objective function since EBW is only used with MMI in the literature. For MMI, we ran some preliminary tests to determine whether GPD or EBW performed better. Consistent with the findings in [24], the two optimization methods seemed to produce nearly identical results. We choose GPD in this thesis because it makes the calculation of the new weights much simpler. From this point forward EBW is no longer considered.

4.3.2 Adjusting the Means and Variances

When the means and variances are to be adjusted, the optimization method will be almost the same. There are, however, two differences: first, these parameters are multidimensional, and second, we will wish to take the variances into account when altering the means.

The multidimensionality of the parameters means that the alteration of each individual mean or variance will now be a vector alteration. This can be handled by adjusting each dimension separately. Partial derivatives can be taken with respect to each dimension; each dimension will then be altered based on its own partial derivative. This means that the amount of computation required will go up by a factor of D, where D is the dimensionality of the feature space. As we will see in Chapter 5, though, it is possible to alter only a subset of the dimensions and still get accuracy gains.

We wish to take the variances into account when altering the means so that each step produces comparable results. If a given dimension has a large variance in a Gaussian, its mean will need to be altered by a large amount to produce any noticeable changes in modeling effectiveness. Likewise, if the Gaussian has a very small variance, small alterations in its mean can produce large changes.

Taking into account these considerations, the GPD update formulas for the means and variances can be written as:

$$\hat{\mu}_{i,j,d} = \mu_{i,j,d} + \epsilon\sigma^2_{i,j,d}\frac{\partial\mathcal{F}}{\partial\mu_{i,j,d}} \tag{4.15}$$

and

$$\hat{\sigma}^2_{i,j,d} = \sigma^2_{i,j,d} + \frac{\epsilon}{2}\frac{\partial\mathcal{F}}{\partial\sigma^2_{i,j,d}} \tag{4.16}$$

where d represents a given dimension of the mean or variance vector. The stepsize is multiplied by the variance when a given dimension of a mean is altered. The stepsize of $\epsilon/2$ for the alteration of the variances is the same as was used in [24].

4.4 Implementational Details

This section provides an overview of the routines used to implement discriminative training in SUMMIT. The training process is divided into two major parts: collection of training statistics and the iterative alteration of parameter values. The statistics collection gathers the relevant parts of the data into a compact format; the parameter alteration iteratively improves the objective function's value on this data. The details of each one of these phases are described next.

4.4.1 The Statistics Collection

The goal of the statistics collection phase is to take in a set of transcribed utterances and write their acoustic data in a compact format to a file. Since we are doing utterance-level discriminative training, we not only need to write out the data for the correct hypothesis, but also the data for the competing hypotheses. The phone string for the correct hypothesis is known from the transcription, but the N-best competing hypotheses must be generated during the collection.

The statistics collection process is begun with a call to a Tcl script. This script goes through the file containing the utterances and loads them one by one. For each utterance, calls are made to SAPPHIRE objects which cause a measurement vector to be computed for each boundary. In addition, the reference transcription for the utterance is loaded from the transcription file. This word string is saved into the classifier object using one of the object's methods.

Data for the correct hypothesis and all of the competing hypotheses must then be collected. This data includes the non-acoustic score for the hypothesis and the phonetic labels assigned to each boundary. This information is first collected for the correct hypothesis running a forced Viterbi search. If this forced search is unable to produce a string matching the reference word string, the utterance is flagged to be thrown away. This usually happens when the search encounters no final states at the last boundary. Note that this means that the number of utterances actually used for training will turn out to be a bit less than the number of utterances in the training set. However, less than five percent of the utterances are thrown out in this manner. Otherwise, the non-acoustic score and sequence of labels assigned to each boundary are stored.

For those utterances on which the forced search is successful, a special method of the Viterbi object is called which generates an N-best list of hypotheses. For each of the hypotheses, a non-acoustic score and a list of models indices for each boundary are provided. If the correct hypothesis is among this list, it is thrown away. The non-acoustic scores and model index lists for the rest of the hypotheses are kept. This is all the information that is needed to completely characterize the hypotheses.

Note that in order to generate an N-best list of competing hypotheses, a set of starting point acoustic models must be provided for the statistics collection. In order to make these N-best lists as accurate as possible, we use a set of ML-trained models. These ML models were trained from the same pool of training data as we are using for the discriminative training. This strategy means that the hypotheses to be trained against are the best competing hypotheses when ML-trained acoustic models are used in the recognizer. It also means that the ML-trained parameters will eventually be the starting point for the parameter alteration. Thus, the baseline accuracy in training experiments will end up being the ML accuracy, and any gains in accuracy through the iterative process can be directly interpreted as gains relative to ML training.

To summarize, the following data gets stored in the statistics file. For each utterance, the sequence of observation vectors is first written. Next comes the non-acoustic score and a sequence of phone indices for the utterance's correct hypothesis. Finally, the non-acoustic score and phone indices are written for each of the competing hypotheses. The format of the statistics file is listed in Appendix A.

4.4.2 The Parameter Adjustment

The parameter adjustment phase needs to read in the data in the statistics file and use it to iteratively improve a given set of parameters. Every iteration of parameter adjustment consists of derivative computation, incremental parameter addition, and the calculation of new scores.

The training data is first read in from the statistics file and placed in an array of utterance structures. Each structure contains the following fields: the number of boundaries in the utterance, measurement vectors for each boundary, the number of hypotheses for the utterance, a list of phone indices for each hypothesis, and non-acoustic scores for each hypothesis. The values in these fields never change. The structure also has fields for acoustic scores for each hypothesis and boundary, and total scores for each hypothesis. The values in these fields are modified during the training process as the acoustic models change.

The training procedure takes in the initial acoustic models and the array of utterance structures. Before the iterative process begins, initial acoustic scores are computed for each utterance, hypothesis, and boundary, by passing the phone index and measurement vector for the boundary to the normal mixture scoring procedure. Total scores for each hypothesis are also computed by adding together the non-acoustic score and all of the acoustic scores. As noted in Chapter 2, about 20 percent of the training data is held out of the training process and used to evalute the training's progress. An initial objective function value and sentence error rate are computed for this development data to give a baseline reference for training performance.

Each iteration begins with a set of derivative calculations. Derivatives are calculated separately for each utterance's cost or likelihood function. A numerical derivative is used; a small change δ is made to the parameter value and new acoustic scores are computed for each boundary affected by the change. The cost/likelihood function is recomputed with the resulting new total scores, and the derivative with respect to the parameter is:

$$\frac{\partial f}{\partial \lambda_0} = \frac{f_{new} - f_{old}}{\delta} \tag{4.17}$$

where f stands for the MCE cost function or MMI likelihood function, and λ_0 represents the parameter being altered. Since the objective function is just the sum of the individual cost/likelihood functions, the derivatives for the separate utterances can simply be accumulated to produce the total derivative with respect to the parameter.

Next, the array of partial derivatives with respect to each parameter is used to alter the parameters. The alterations are done exactly according to the update formulas in section 4.3. It should be emphasized that all of the derivative calculations are done before any of the parameter alterations are done. The alternative is to alter each parameter immediately after calculating the corresponding derivative. The method used here has generally been found to lead to smoother convergence.

We have found that when parameter alteration is performed, the silence models can quickly come to dominate the acoustic space. Presumably this is because they make up a significant portion of every sentence and thus occur much more often in the training data than other models. After many iterations of adjustments, the recognizer begins to pick the empty hypothesis a disproportionate number of times. Besides, improved silence modeling does not offer much potential for improvement since the presence or absence of silences in the recognized string does not affect word accuracy. For these reasons, silence models are left unaltered during the iterative process.

Finally, all of the acoustic and total scores are updated to reflect the new acoustic model parameters. The objective function value and sentence error rate for the development data are again calculated to see if performance is still improving and if the parameter values are converging.

4.4.3 Implementational Summary

In summary, these are the features of the implementation that may have some effect on the results:

- During statistics collection, less than five percent of the training data gets thrown out due to a failed forced search. This means that the amount of data used for training is slightly less than the size of the training set.

- ML-trained models are used in order to generate N-best lists of competing hypotheses. The ML models then provide the starting point values for the parameters.

- Eighty percent of the available data is used for training, twenty percent for evaluation.

- Numerical differentiation is used, with a parameter δ that controls how far the parameters are perturbed.

- All derivative calculations are done before any alterations are done.

- Silence models are not altered.

4.5 Summary

This chapter discussed the algorithms used for discriminative training in this thesis. We chose utterance-based training over phone-based training to make the criterion more relevant to the metric of word error rate. Formulas for and properties of the MCE and MMI objective functions were presented. We showed how to adjust the parameters based on gradients of the objective functions. Finally, we discussed the implementation and some of its implications.

We are now ready to test the effectiveness of discriminative training. The next chapter analyzes the results of a large variety of training experiments.

Chapter 5

Discriminative Training Results

This chapter runs through a series of experiments designed to test the effectiveness of discriminative training in SUMMIT. We wish to test many combinations of model parameters to be optimized, training criteria to be used, and training control parameter values. The general methodology employed is to alter one degree of freedom in the training specifications at a time, and observe the results of each alteration. Each section in this chapter describes the effects of changing a single training control parameter.

Recognizer accuracies are measured after each training iteration by extracting the acoustic models at that point and using them on the 500 utterance test set, *test_500*. For the more successful training runs the recognizer accuracy for the final model set is measured on the 2500 utterance test set, *test_2500*. Word error rate is used exclusively with *test_500* since it does not have enough utterances to make a meaningful measurement of sentence error rate; both word and sentence error rates are measured when *test_2500* is used. These numbers can be compared with the sentence error rates on the development data to see if there is a strong correlation between them. We also want to know how quickly the parameters converge, which can be measured by looking at the changes in the objective function and the magnitude of the parameter alterations at each iteration.

The initial parameter values for all training experiments in this chapter come from the same set of ML-trained models. This allows direct comparisons to be made between the various training runs. We have, however, run experiments with other starting point models and observed similar gains.

5.1 Results of Mixture Weight Training

The first group of training runs to be examined will involve the alteration of the mixture weights. Later, we will look at the effects of altering the Gaussian means and variances. This section first describes MCE training of the weights and then proceeds to MMI training.

5.1.1 MCE Training of the Mixture Weights

We examine the effects of MCE weight training by first describing a base training run, and then comparing this base case with the results when various training parameters are changed.

Examination of a Base Case

As a base case we examine a training run using the *train_6000* set with N-best lists of 5 hypotheses (i.e., $N = 5$), ϵ (the step size) set to 2400, and ρ (the rolloff) set to 4.0.

Figure 5-1 shows the change in the unnormalized objective function measured on the 1200 development utterances at each iteration. The unnormalized objective function is just the expression of equation (4.6) without the $1/N_s$ term in the front; since the values are larger, they are easier to compare. Each bar in the figure represents the objective function value at the previous iteration minus the value at the current iteration. As can be seen, the changes get smaller and smaller over the long term. This shows that the objective function is converging to some value as the vector of weights is adjusted.

Figure 5-2 contains more evidence of the convergence of the training process. This figure shows the average magnitude of the weight alterations at each iteration. It is apparent that the alterations are decreasing in magnitude roughly exponentially

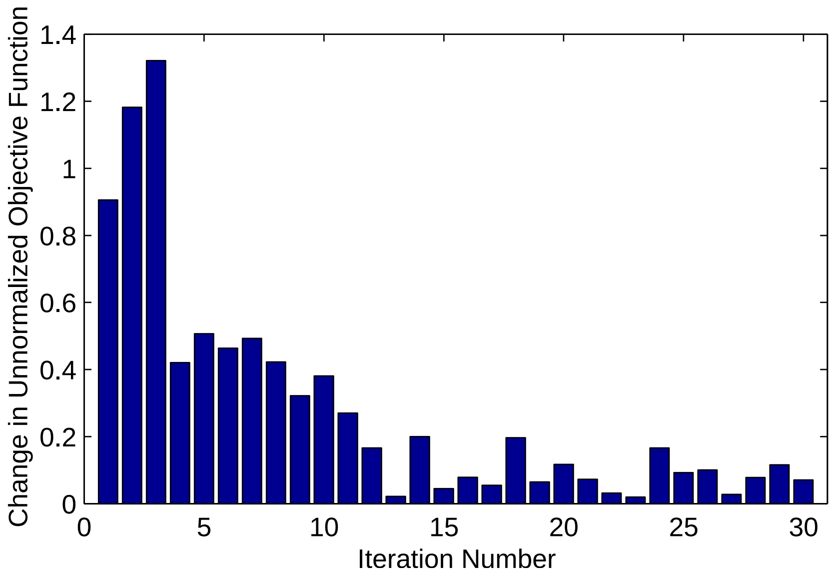

Figure 5-1: Change in Unnormalized Objective Function Value vs. Iteration Index for MCE Weight Training, Base Case

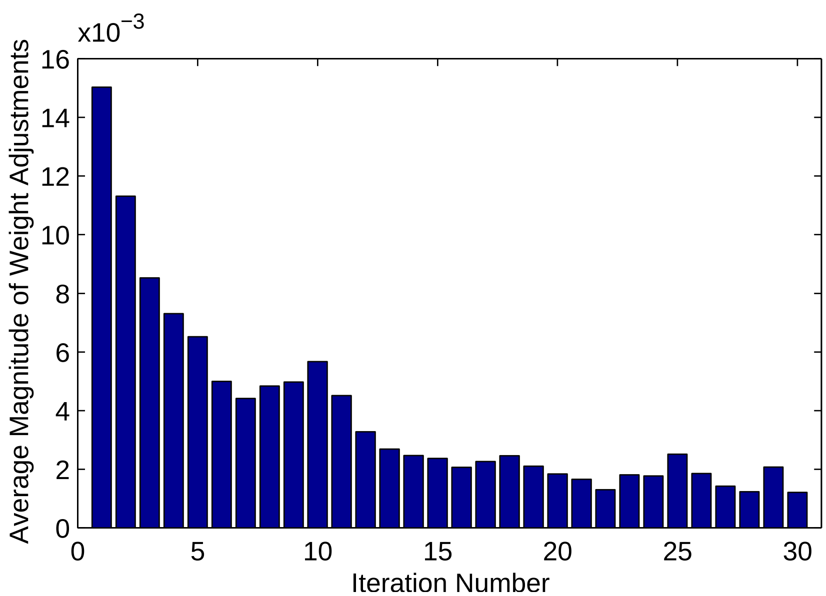

Figure 5-2: Average Magnitude of the Weight Alterations vs. Iteration Index for MCE Weight Training, Base Case

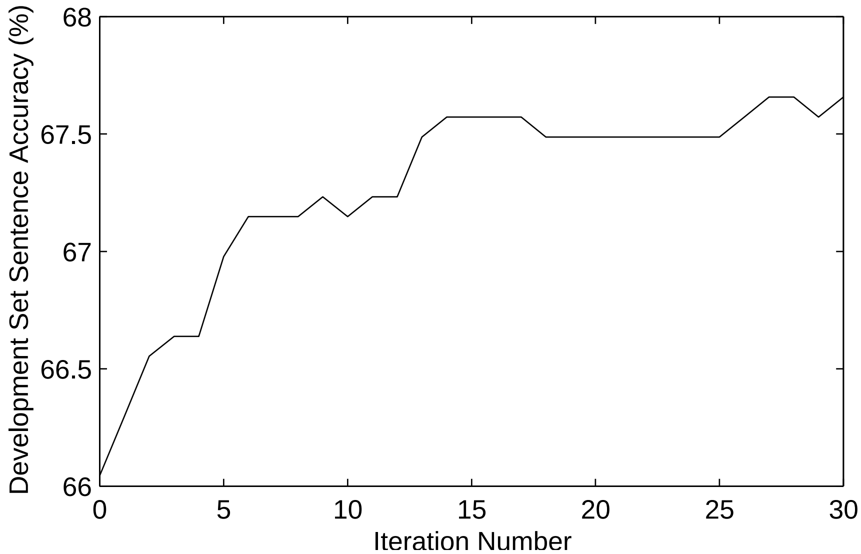

Figure 5-3: Sentence Accuracy on Development Data vs. Iteration Index for MCE Weight Training, Base Case

as the training proceeds. MCE seems to have a smooth, well-behaved convergence characteristic.

The effectiveness of the parameter alterations can be measured as the training proceeds using the sentence error rate on the development utterances. Figure 5-3 shows the sentence accuracy on this data as a function of the iteration index. A sentence is considered to be recognized correctly if its correct hypothesis scores higher than all of the hypotheses in the N-best list. Of course, this does not take into account the possibility of hypotheses not in the original N-best list eventually scoring better than the correct hypothesis, but for sufficiently large list sizes this should not happen very often. The accuracy is seen to increase fairly steadily and then settle around some final value. About five percent of the original sentence errors are corrected by the end of the training process.

Finally, we examine the test set accuracies after every iteration to see if the sentence-level gains on the development data translate to word error rate gains on test data. Figure 5-4 shows the word accuracy on the *test_500* set at each iteration. The accuracies exhibit a general increasing trend except for a period of about ten iterations where they fluctuate a bit. This fluctuation can perhaps be attributed to

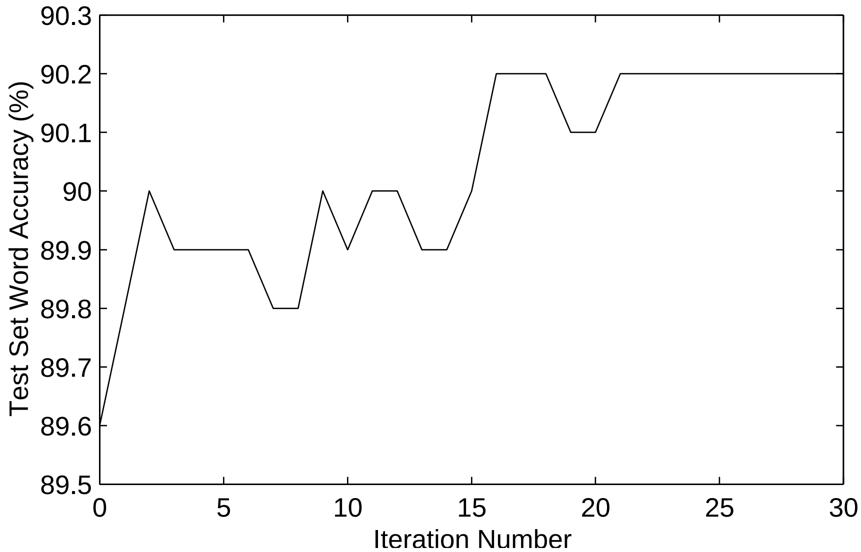

Figure 5-4: Word Accuracy on *test_500* vs. Iteration Index for MCE Weight Training, Base Case

the relatively small amount of training data used or the small size of the N-best lists; the following sections will investigate the effects of increasing these numbers. The accuracy settles at about 90.2%, which is a 5.8% relative error rate reduction compared to the baseline accuracy of 89.6%.

Altering the Step Size

Now that we have observed the outcome of one specific case, we can begin to test the effects of changing various training parameters. We start by trying different values of the step size ϵ. The step size controls the rate of learning; too small a value will lead to very long training times, and too large a value will make the training unstable. The base case used an ϵ of 2400, so we will perform the training with ϵ's of 1200 and 4800 and compare.

To get an initial indication of the effects of changing ϵ, we can observe what happens to the objective function. Figure 5-5 gives the values of the unnormalized objective functions on the development data at each iteration using the three different step sizes. As would be expected, the rate of the initial drop in the objective function is roughly proportional to the step size. With each of the step sizes the objective

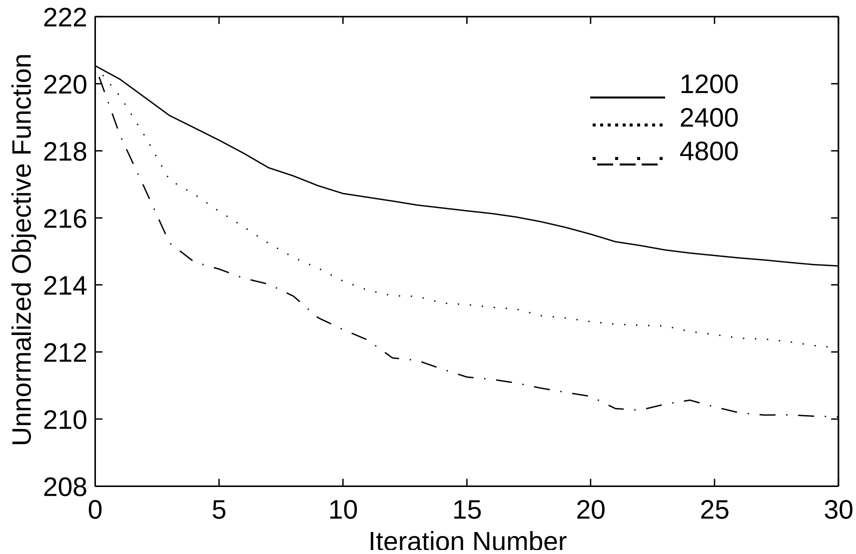

Figure 5-5: Unnormalized Objective Function vs. Iteration Index for MCE Weight Training with Various Step Sizes

function converges at a different point. Presumably this is due to the presence of many local minima in the parameter space that can attract training runs with lower values of ϵ. Note that the objective function curve when $\epsilon = 4800$ is not as smooth as the other two curves; in fact, it is not even monotonically decreasing. This may be a sign that for this step size the training is just barely stable, and the test set accuracies may fluctuate more wildly.

To test this hypothesis we can examine the test set accuracies after every iteration for each step size. Figure 5-6 shows the word accuracies on the *test_500* set for the two new training runs; the accuracies for the base case were shown in figure 5-4. As predicted by the objective function curve, the accuracies when ϵ is 4800 reach the highest point but also fluctuate the most wildly. The final accuracies when $\epsilon = 4800$ and $\epsilon = 2400$ are the same, and are a bit greater than the final accuracy when $\epsilon = 1200$; this is also to be expected since the objective function decreases more for the larger step sizes. In the end, the base step size of 2400 seems to provide a reasonable balance between achieving the largest gains and keeping the training process stable.

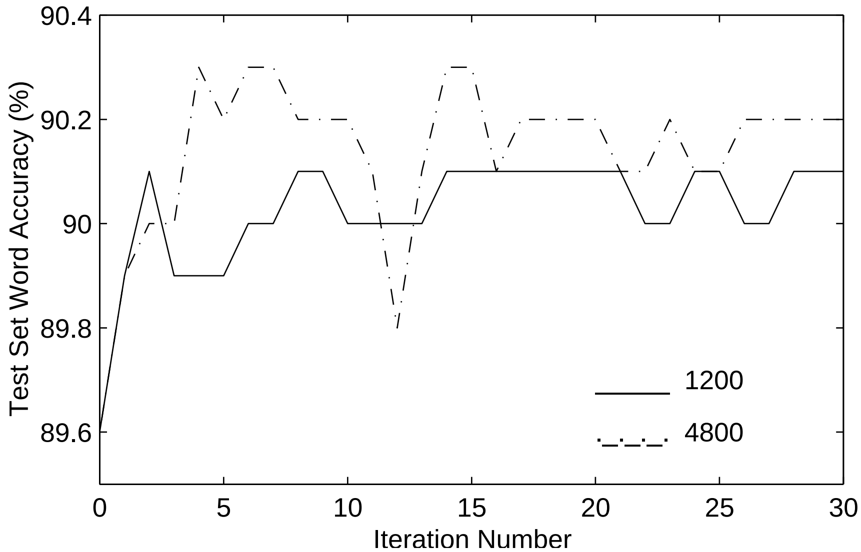

Figure 5-6: Word Accuracy on *test_500* vs. Iteration Index for MCE Weight Training with Various Step Sizes

Altering the Rolloff

Next we investigate the effects of changing the rolloff ρ. The rolloff controls the abruptness of the zero-one transition in the MCE cost function. Choosing a rolloff is a tradeoff between approximating a step more closely (high rolloff) and providing a smoother derivative (low rolloff). The base case used a ρ of 4.0; here we will also try $\rho = 1.0$ and $\rho = 10.0$.

The effects of rolloff on differentiation can be seen in the maximum alteration magnitudes. Larger rolloffs lead to greater derivatives in the transition region of the cost function, and this means that the resulting alterations can be larger. Figure 5-7 shows the magnitudes of the largest alterations at each iteration for each value of ρ. It is apparent that the largest alterations tend to be much greater as ρ is increased (though this is certainly not the case at every iteration). Excessively large alterations could have a negative impact on the resulting accuracies if those alterations come to dominate the training process.

On the other hand, when ρ is too low the cost function will no longer look like a step. Thus, it seems intuitively that improving the objective function may not have the desired effect on error rates. To test the impact of high and low rolloffs on

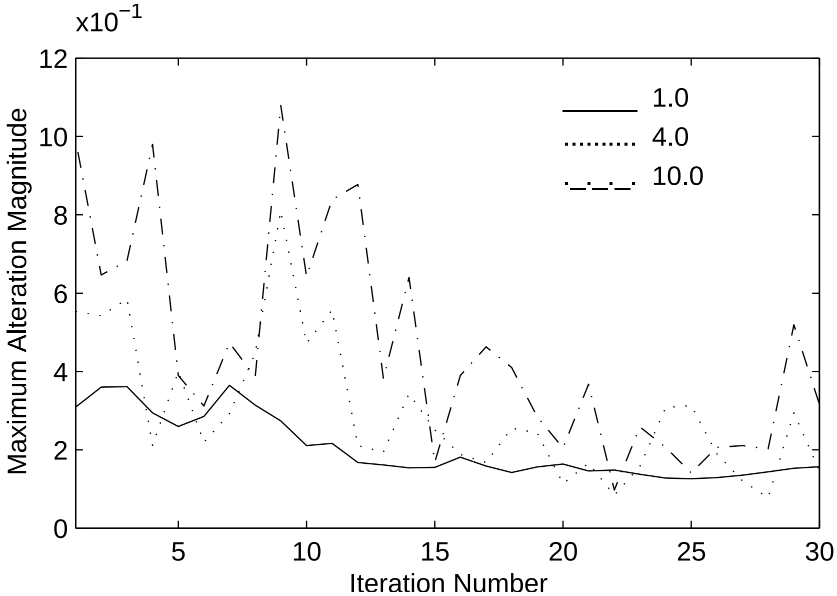

Figure 5-7: Maximum Weight Alteration Magnitude vs. Iteration Index for MCE Weight Training with Various Rolloffs

training effectiveness, we examine the test set accuracies for each value of ρ. Figure 5-8 gives the word accuracies on the *test_500* set for the two new rolloffs. With $\rho = 10.0$, the accuracies behave very similarly to the accuracies when $\rho = 4.0$ (figure 5-4). They fluctuate a bit more near the beginning, when the largest of the individual alterations occur, but the final accuracies are the same in both cases. With $\rho = 1.0$, accuracy gains are observed through the first 18 iterations of the training. After this point, however, the accuracies begin to fall off rapidly. This can be attributed to the lack of a steep transition region in the cost function curve. This does not cause any problems during the initial iterations when coarse adjustments are made, but as finer adjustments are made later on it is apparent that the minimization of the objective function is not directly connected to improvement in word accuracies. Thus, we see that the greatest danger to training effectiveness lies in making the rolloff too small.

Increasing the Number of Hypotheses Used for Training

Up to this point, N-best lists of 5 hypotheses have been used with the training utterances. This is a relatively small list size, and it is quite possible that important competing hypotheses are not included. If this is the case, the training might by

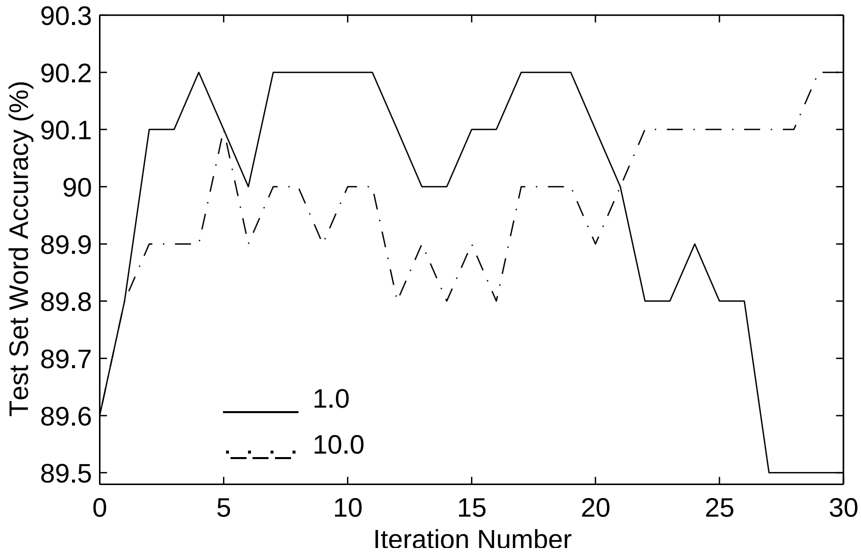

Figure 5-8: Word Accuracy on *test_500* vs. Iteration Index for MCE Weight Training with Various Rolloffs

coincidence make them score high enough to overtake even the correct hypotheses, thus causing a degradation in test set accuracies.

In this section we try training with N-best lists of 20 and 40 hypotheses, in addition to the lists of 5 hypotheses used previously. The behavior of the training indicators, i.e. the objective function and the accuracy on the development data, is virtually the same regardless of the list size. This is to be expected, since these measures are only functions of the data available, and not any data that might be outside the lists. Recognizer accuracy on the test set is the only measure that indicates whether it is helpful to use more competing hypotheses.

Figure 5-9 shows the word accuracies on *test_500* for the two new N-best list sizes. With 20 hypotheses, the accuracies after the early iterations are greater than with 5 hypotheses, but the final accuracy is the same. With 40 hypotheses, on the other hand, the final accuracy improves to 90.4%, compared to 90.2% for the base case. It seems that it is possible to achieve greater gains using more than 5 competing hypotheses.

When increasing the size of the N-best lists it is important to remember that the training time and memory requirements are each proportional to this size. If

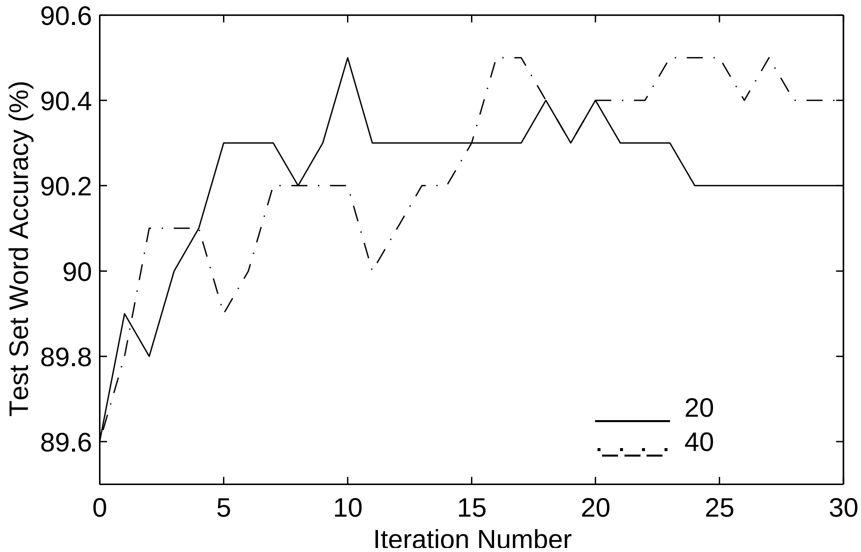

Figure 5-9: Word Accuracy on *test_500* vs. Iteration Index for MCE Weight Training with Various Numbers of Competing Hypotheses

computational factors are important, it may be undesirable to increase the list size past a certain point.

Increasing the Number of Training Utterances

It is also possible to increase the number of utterances in the training set. As is the case with standard ML training, increasing the number of training samples is expected to increase the predictive power of the discriminatively trained models. Therefore, we expect the test set accuracies to improve. Figure 5-10 shows the word accuracies on *test_500* for a set of models trained using the *train_12000* set. The training parameters are the same as they were for the base case. As can be seen, the final accuracy is 90.3%, compared to 90.2% for the base case. This is a bit of an improvement, but perhaps not as great as would be expected for a doubling of the number of training utterances. This lack of a major improvement seems to indicate that discriminative training does most of the good it can do with very few training utterances; in other words, discriminative training requires less training data than ML training. This fact means that discriminative training offers potential advantages in situations where the amount of training data is limited, as we will see in later chapters.

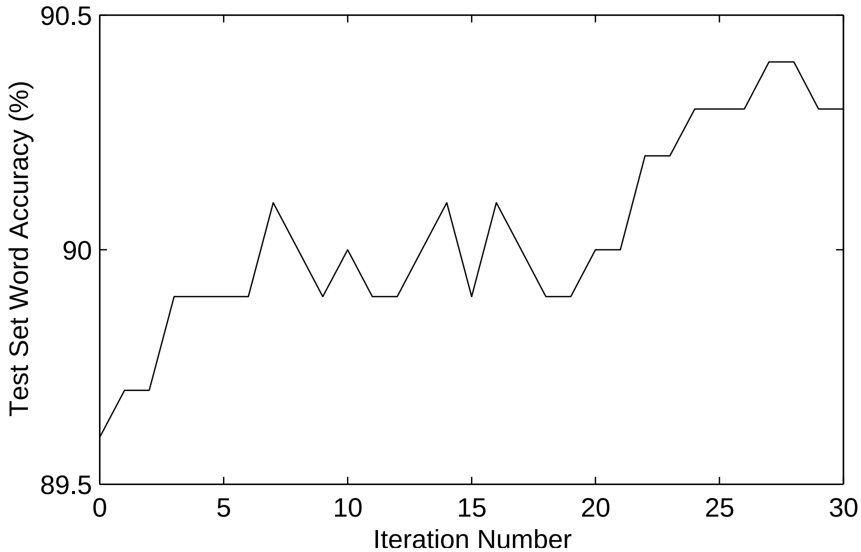

Figure 5-10: Word Accuracy on *test_500* vs. Iteration Index for MCE Weight Training using *train_12000*

As was the case when we increased the size of the N-best lists, it must be remembered that the training time and memory requirements are each proportional to the number of training utterances used. Thus, it is undesirable to increase the number of training utterances to the point where accuracy gains are minimal.

Summary of Accuracies for MCE Weight Training

In summary, the training is not terribly sensitive to the exact value of the step size parameter as long as it is in a moderate range. Likewise, as long as the rolloff is not too low its exact value is not critical. Test set accuracies seem to increase somewhat as more training data or more competing hypotheses are used. Since measurements on *test_500* could be fairly noisy, we can get a more precise measurement of the effects of increasing these values by using the *test_2500* set. Table 5.1 summarizes the word accuracies on both *test_500* and *test_2500* for the ML-trained models and the final model sets from various training runs. It also provides sentence accuracies for *test_2500* since this set contains enough utterances to make this measurement meaningful. The accuracies on the *test_2500* set seem to confirm the previous statement that increasing the size of the training set from 4800 to 9600 utterances does not

Training Run	*test_500* WA	*test_2500* WA	*test_2500* SA
ML Models	89.6	81.7	64.3
train_6000, $N = 5$	90.2	82.1	64.9
train_12000, $N = 5$	90.3	82.2	64.8
train_6000, $N = 20$	90.2	82.4	65.1
train_6000, $N = 40$	90.4	82.5	65.3

Table 5.1: Summary of Accuracies for MCE Weight Training with Various N-best List Sizes and Training Set Sizes.

improve the results much. In fact, the sentence accuracy actually goes down a bit with the extra training data. However, increasing the size of the N-best lists from 5 to 20 or 40 does have a significant positive impact. The data from *test_2500* makes this benefit much clearer than the previous 500 utterance data. Increasing N from 5 to 20 almost doubles the improvement in word accuracy over the ML models (0.4% improvement to 0.7% improvement). Increasing N again to 40 results in even better word accuracy, though it is only a bit of an improvement over $N = 20$. The sentence accuracies also go up when N is increased. This data makes it clear that it is best to use an N-best list size that is greater than 5.

From the data collected in this section, it seems that a reasonable set of parameters to carry forward is $\epsilon = 2400$, $\rho = 4.0$, $N = 20$, and a training set size of 4800 utterances. These choices provide a good tradeoff between accuracy gains and training time.

5.1.2 MMI Training of the Mixture Weights

We now switch objective functions and train the weights using the MMI criterion. We specifically wish to see how the MMI trained models compare with the MCE trained models, and whether there are any major differences in the training processes. Many of the issues involved, such as the choices of step size, N-best list size, and number of training utterances, are conceptually the same as in the MCE case. Thus, only the experimental results of these choices need be discussed here.

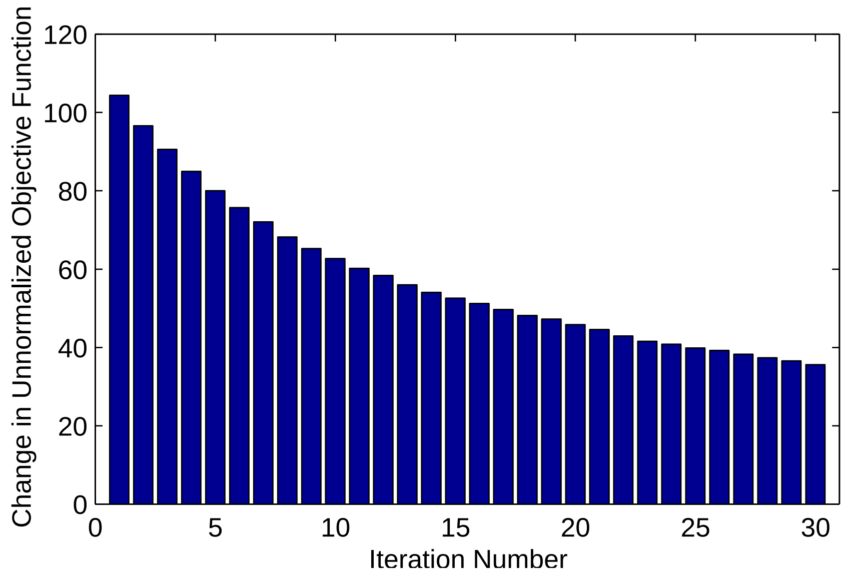

Figure 5-11: Change in Unnormalized Objective Function Value vs. Iteration Index for MMI Weight Training, Base Case

Examination of a Base Case

Again, we start with a base case to glean the basic properties of MMI training and establish a reference to compare against later. For the MMI base case we examine a training run using the *train_6000* set with N-best lists of 20 hypotheses, ϵ set to 480, and τ (the score threshold) set to 0.1.

Figure 5-11 shows the change in the unnormalized objective function measured on the development utterances at each iteration. This is the same information as was presented in figure 5-1 for the MCE base case. As before, the changes get smaller and smaller over the long term; however, they seem to be decreasing in a much more orderly fashion. Specifically, it looks as though the changes decrease roughly exponentially. While the decay is more orderly, it is also much slower: after 30 iterations the changes are still about 1/3 as big as the first few changes. In contrast, after 30 iterations the changes in the MCE objective function were less than 1/10 as big as the first few changes. This may suggest that the step size is a bit too small.

The theme of slow convergence is continued in figure 5-12. Here we see that the average magnitude of the weight alterations decreases monotonically as the training progresses. However, the rate of decrease is extremely slow (compare with figure 5-2,

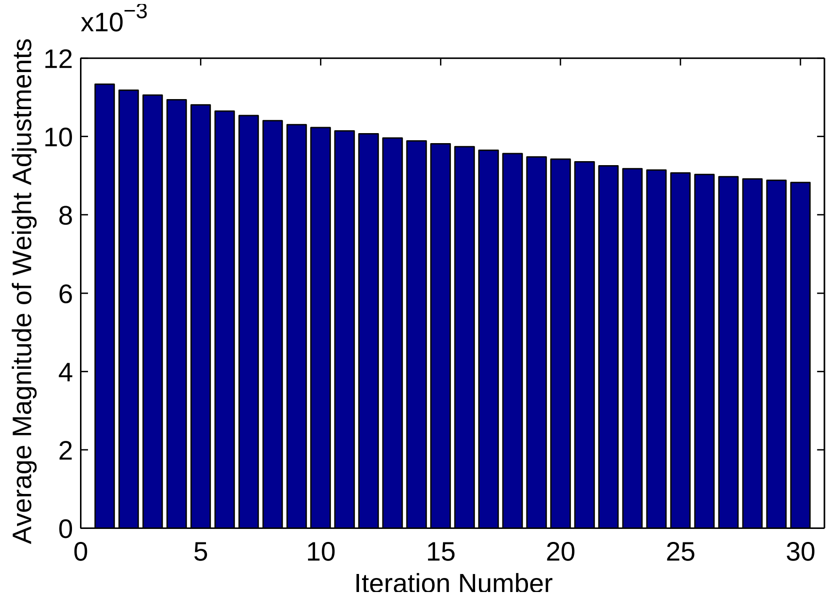

Figure 5-12: Average Magnitude of the Weight Alterations vs. Iteration Index for MMI Weight Training, Base Case

which has the same scale). It appears as if hundreds of iterations will be required before the average alteration magnitude approaches 0. Thus, we expect that the accuracies of the resulting models may not converge to any particular value over the iterations of interest.

As a first step towards determining if this is the case we examine the sentence error rate on the development utterances. Figure 5-13 shows the sentence accuracy on this data for each iteration index. Somewhat surprisingly, the accuracies do not fluctuate much more than those for the MCE base case. Moreover, they seem to increase by about the same amount and again settle around some final value. About four percent of the original sentence errors are corrected by the end of the training process, as compared to five percent for MCE.

Next we wish to see if the test set accuracies are as well behaved. Figure 5-14 shows the word accuracy on the *test_500* set at each iteration. This seems to follow the same kind of pattern as the MCE accuracies. In addition, the accuracy for the final model set is 90.5%. This is an 8.7% relative error rate reduction compared to the baseline accuracy of 89.6%. It is unknown why, on average, fairly large alterations are still made to the weights at later iterations, yet the resulting model accuracies seem

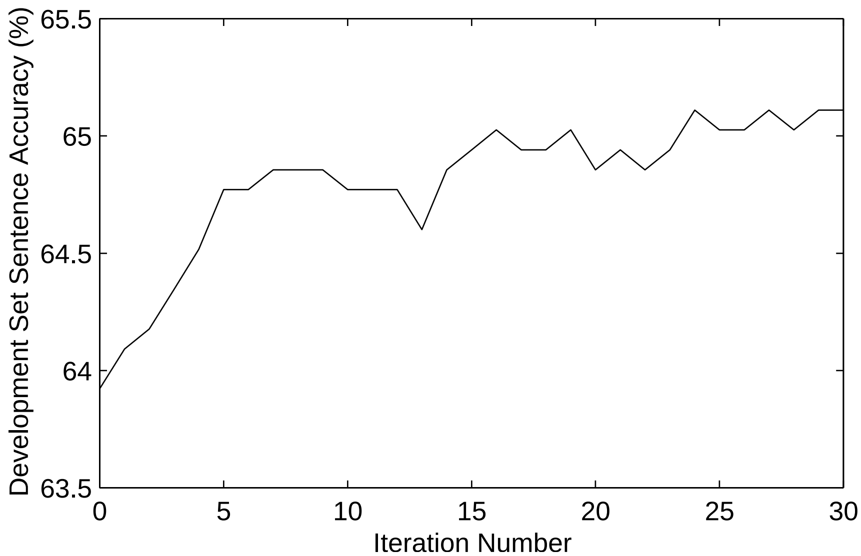

Figure 5-13: Sentence Accuracy on Development Data vs. Iteration Index for MMI Weight Training, Base Case

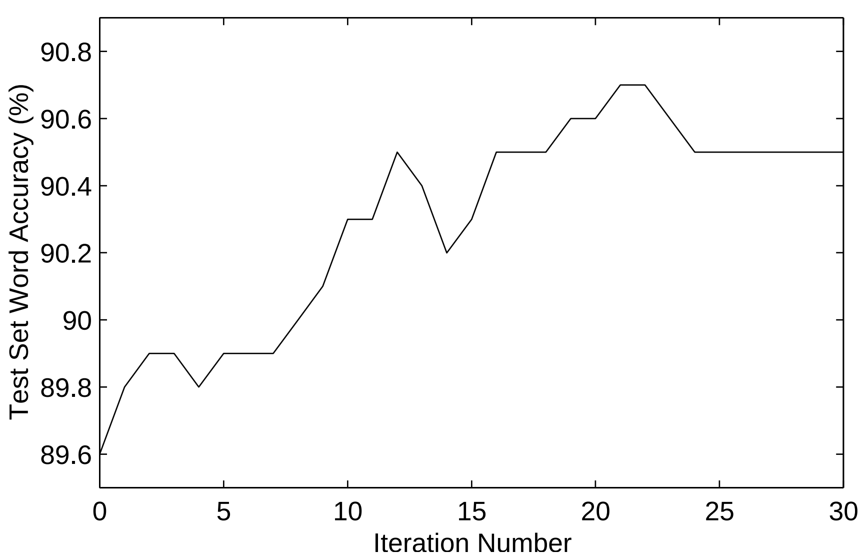

Figure 5-14: Word Accuracy on *test_500* vs. Iteration Index for MMI Weight Training, Base Case

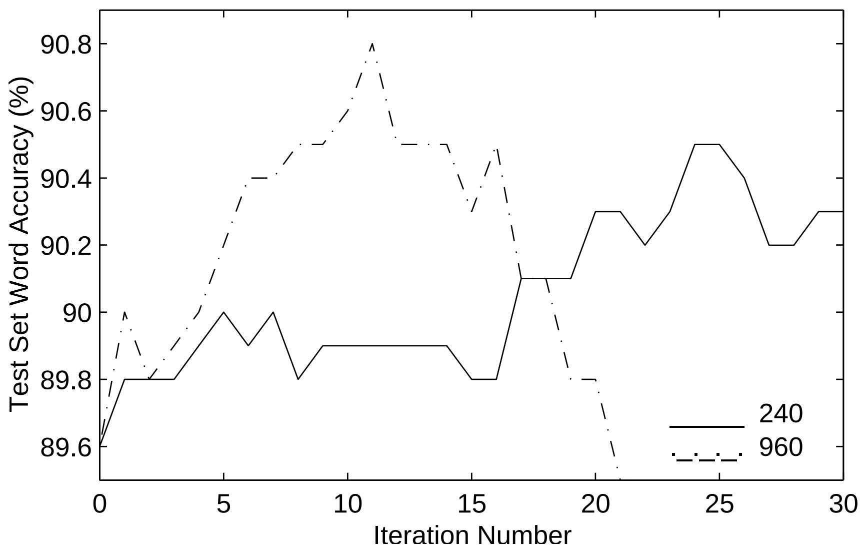

Figure 5-15: Word Accuracy on *test_500* vs. Iteration Index for MMI Weight Training with Various Step Sizes

to settle at certain values. Perhaps as the training proceeds some of the alterations begin to cancel each other out.

Altering the Step Size

Again, we would like to try different step sizes to determine the optimum value. The issues involved with the step size are the same as they were with MCE, and will not be discussed again here. The base case used a step size of 480, so we will perform training with ϵ's of 240 and 960 and compare. Note that the step sizes are much smaller than in the MCE case; this is because the MMI objective function has a larger magnitude, making its derivatives larger. Therefore, in order to get parameter alterations of comparable sizes, the multiplying factor ϵ must be smaller.

The qualitative behavior of training runs with different step sizes is similar to the behavior when changing the step size in MCE, so we will jump straight to the results. Figure 5-15 shows the word accuracies on *test_500* using the two new values of ϵ. With $\epsilon = 240$, the training seems to do almost as well as with a step size of 480. As expected, the rise in accuracies seems to occur about half as fast. The accuracies also seem to be settling at a slightly lower point than the 90.5% that the base case achieved. With $\epsilon = 960$, accuracies increase very rapidly for the first ten iterations. After this point, however, they drop just as rapidly and end up worse than the starting point accuracy. Clearly, this step size is too large. This does not cause any problems for the early iterations when coarse adjustments are needed, but later, when finer adjustments are required, the alterations are unable to settle the parameters at an optimum point. The fact that this step size is too large is a bit surprising in light of our earlier observation that the base step size results in very slow convergence. Perhaps an intermediate value would be the best choice, or even the use of a large step size for the first few iterations, followed by refinements using a smaller step size. Of the three step sizes used here, however, it appears that 480 is the best.

Altering the Score Threshold

Recall from section 4.2.2 that a lower bound can be set on the MMI log likelihood function using a score threshold τ. τ is a number between 0 and 1. Larger τ's reduce the range of the log likelihood function, but also make it less likely that derivatives of the function will be useful. The base case used a τ of 0.1; here we will also try $\tau = 0.05$ and $\tau = 0.5$.

It turns out that with these different thresholds there is almost no observable difference between the various training runs. The objective functions, average alteration magnitudes, and development set sentence error rates are nearly exactly the same on every iteration. To illustrate the sameness of the training runs, figure 5-16 shows the word accuracies on *test_500* using the two new values of τ. As can be seen, not only do the accuracies all settle at the same value, but they assume almost the same values at every iteration. It is apparent that the choice of score threshold does not have much effect on MMI training.

Changing the Number of Hypotheses Used for Training

Again, we wish to observe the effects of the N-best list size on training. The issues involved are the same as in MCE training. The MMI base case used lists of 20

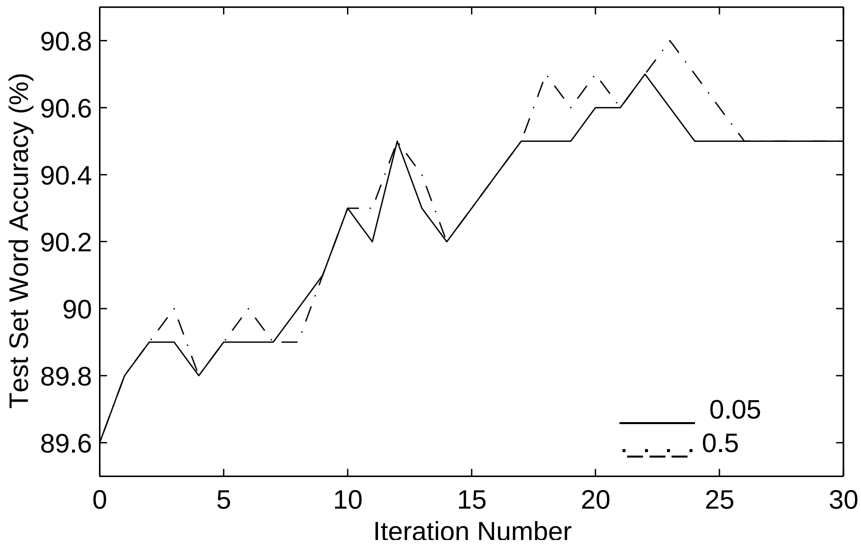

Figure 5-16: Word Accuracy on *test_500* vs. Iteration Index for MMI Weight Training with Various Score Thresholds

hypotheses, so here we consider lists of 5 and 40 hypotheses.

Figure 5-17 shows the word accuracies on *test_500* for the two new N-best list sizes. With 5 hypotheses, the accuracies appear to bounce around more than usual. In addition, they seem to be on a significant downward trend in the later iterations. It is reasonable to conclude that 5 hypotheses are insufficient for MMI training. The use of 40 competing hypotheses seems to produce similar accuracy gains as in the base case. Thus, the initial choice of 20 competing hypotheses appears to be a good one.

Increasing the Number of Training Utterances

We found with MCE that doubling the number of training utterances improved test set accuracies a bit, but perhaps not as much as would be expected. Now we wish to see if this same pattern holds for MMI.

Figure 5-18 shows the word accuracies on *test_500* for models trained using the *train_12000* set. The accuracies appear to rise to the same point as when *train_6000* is used, though they take longer to get there. This seems to provide additional evidence that discriminative training requires a relatively small amount of data, and using

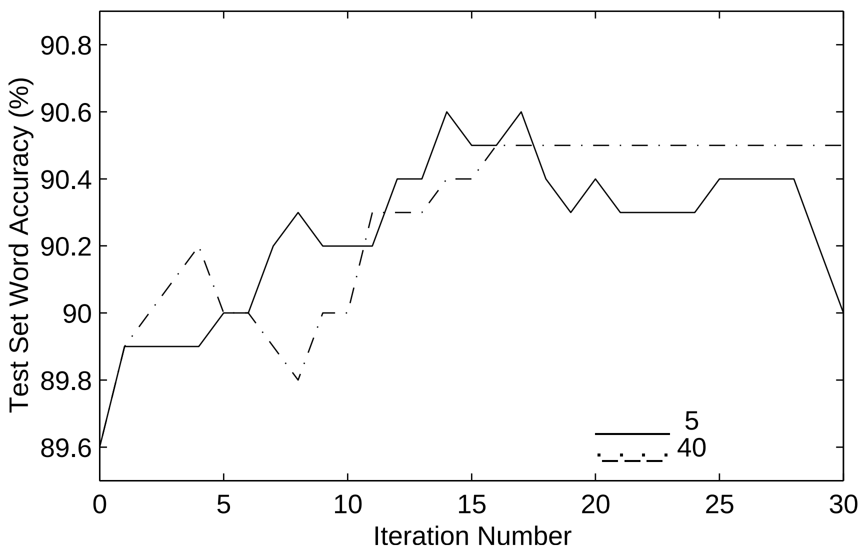

Figure 5-17: Word Accuracy on *test_500* vs. Iteration Index for MMI Weight Training with Various Numbers of Competing Hypotheses

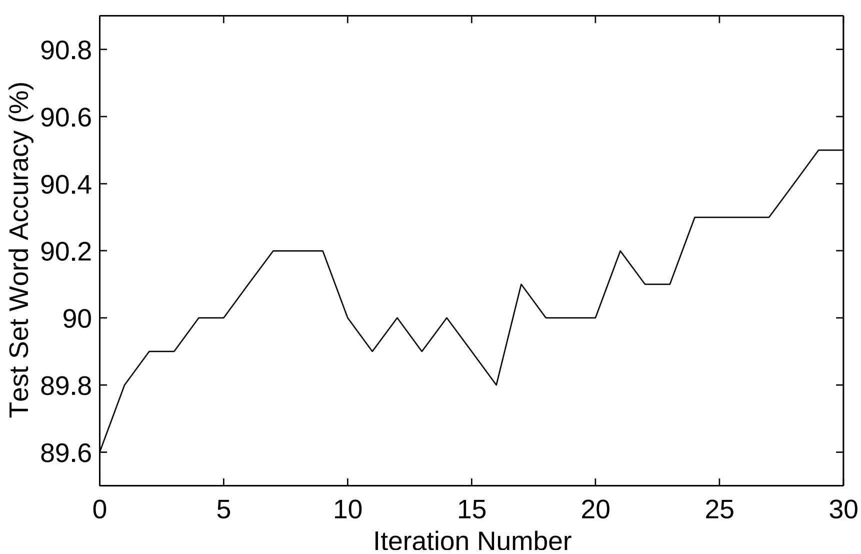

Figure 5-18: Word Accuracy on *test_500* vs. Iteration Index for MMI Weight Training using *train_12000*

Training Run	*test_500* WA	*test_2500* WA	*test_2500* SA
ML Models	89.6	81.7	64.3
train_6000, $N = 5$	90.0	81.9	64.7
train_6000, $N = 20$	90.5	82.6	66.0
train_6000, $N = 40$	90.5	82.6	66.0
train_12000, $N = 20$	90.5	82.6	66.1

Table 5.2: Summary of Accuracies for MMI Weight Training with Various N-best List Sizes and Training Set Sizes.

more provides little benefit.

Summary of Accuracies for MMI Weight Training

We find that the results of the training do not change much as the step size parameter and the score threshold are varied, as long as they assume moderate values. This is similar to the observation made earlier about the step size and rolloff in MCE. A table similar to table 5.1 can be constructed to get a more precise measurement of the effects of changing the number of competing hypotheses and the amount of training data. Table 5.2 summarizes the word accuracies on *test_500* and the word and sentence accuracies on *test_2500* for the ML-trained models and the final models from various training runs. Note that this data supports the previous observation that moving from *train_6000* to *train_12000* does not offer much accuracy improvement. We see that for MMI, like MCE, better results are achieved with an N-best list size that is greater than 5. However, for MMI recall that the accuracies do not seem to converge at all when $N = 5$, whereas with MCE they converged but at a lower point than when N was 20 or 40.

We also see from table 5.2 that the word accuracies on *test_2500* are only slightly better with MMI than with MCE. This supports previous studies (e.g., [25]) which have concluded that MCE and MMI tend to produce similar gains. Strangely, however, the sentence accuracies show significantly more improvement using MMI. If sentence accuracy is the most important metric, MMI may be the better choice in consideration of this data. On the other hand, an advantage of MCE is that it requires less training time due to the extra logarithms and exponentials which must be calculated in the MMI objective function. In the end, it seems that either criterion will be an acceptable choice for most applications.

5.2 Results of Training the Means and Variances

Now that we have observed the results of altering the mixture weights, we proceed to the alteration of Gaussian means and variances. MCE will be used here since it was found to have slightly better convergence behavior in the last section, and because it is a bit faster than MMI. There are several inherent differences between altering weights and altering means and variances. The most obvious difference is that the means and variances are multi-dimensional. This means that there are potentially more degrees of freedom in parameter alteration, but also that the training can potentially take much longer. Another difference is that we wish to take the variances into account when altering the means, and vice versa. We begin by analyzing a training run which alters only one dimension of the means and variances, and then compare this to other training runs.

5.2.1 Altering One Dimension of the Means and Variances

The feature vectors used in this thesis are 50 dimensional. Thus, we expect alteration of the full mean and variance vectors to take 50 times longer than alteration of just the mixture weights. If weight training runs overnight, training the entire means and variances should take weeks. This is beyond the time scale that is practical for this thesis.

Luckily, it is possible to update each dimension of the means and variances independently of the other dimensions. We can begin by altering only the first dimension, and observing the accuracy gains this produces. If both the means and variances are altered, this should only take twice as long as weight training (because two parameters are being altered instead of one).

Equations (4.15) and (4.16), discussed in section 4.3.2, are used here to perform

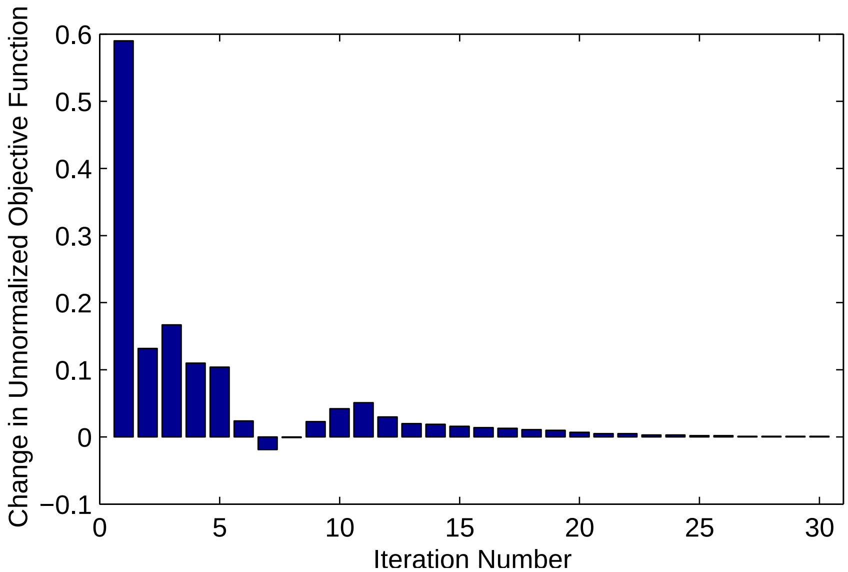

Figure 5-19: Change in Unnormalized Objective Function Value vs. Iteration Index for Mean/Variance Training, One Dimension

the parameter updates. We use *train_6000* with N-best lists of 5 hypotheses, ϵ equal to 960, and ρ set to 4.0. It turns out that a few very large alterations can have a negative effect on the training process, so we dynamically limit the size of the alterations at each iteration. This is discussed more in section 5.2.3.

Figure 5-19 shows the change in the unnormalized objective function measured on the development utterances at each iteration. Due largely to the limits placed on the alteration magnitudes, the objective function converges very smoothly and quickly in the later iterations. It is interesting to note, though, that the objective function actually gets worse on iterations 7 and 8. This never happened during weight training, and is perhaps a sign that training the means and variances may be plagued with more instability.

The average magnitudes of the parameter alterations do not contain much information in this case, because they are strongly influenced by the limits placed on the maximum alterations. We can jump straight to an analysis of the sentence error rate on the development utterances. Figure 5-20 shows the sentence accuracy on this data as a function of the iteration index. Notice that the sentence accuracy settles at a much lower value than we have previously seen for weight training. In addition, the

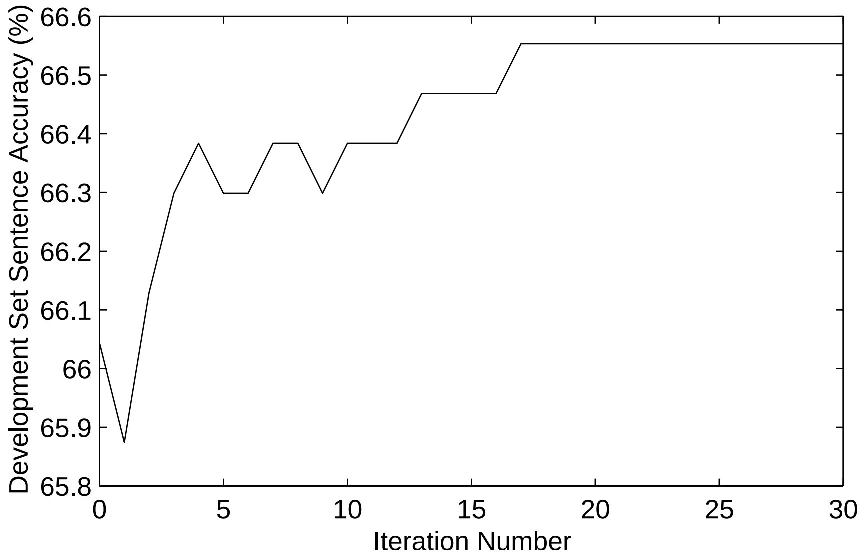

Figure 5-20: Sentence Accuracy on Development Data vs. Iteration Index for Mean/Variance Training, One Dimension

accuracy actually decreases on the first iteration! These are signs that perhaps this training will be less effective in improving test set accuracies than weight training.

We examine the test set accuracies after every iteration next. Figure 5-21 shows the word accuracies on *test_500*. The accuracies seem to be very erratic, dropping significantly at the beginning before eventually settling at a higher point. Also, the final accuracy (90.1%) is lower than for any of the weight training runs. It seems that training the means and variances is not translating to higher accuracies in our system; in fact, the effects seem quite unpredictable.

5.2.2 Altering Only the Means

A variation on the previous experiment is to try altering only the means. The variances are left constant throughout the training. The training parameters are kept the same as before, and still only one dimension is altered.

The results of this experiment are no better than those of the previous experiment. Figure 5-22 gives the word accuracies on *test_500* at each iteration. The accuracies do not follow any general upward trend, even over the long term. In fact, they are almost periodic, rising for a few iterations and then falling again. One possible explanation

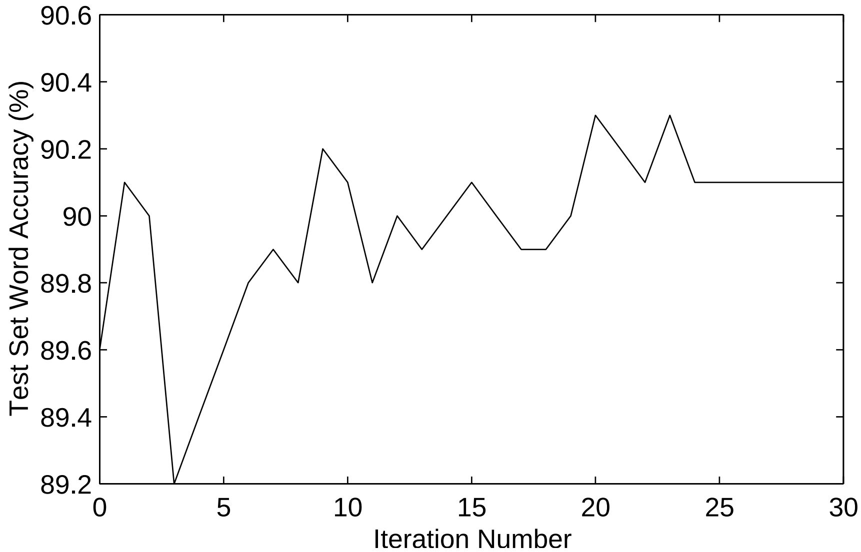

Figure 5-21: Word Accuracy on *test_500* vs. Iteration Index for Mean/Variance Training, One Dimension

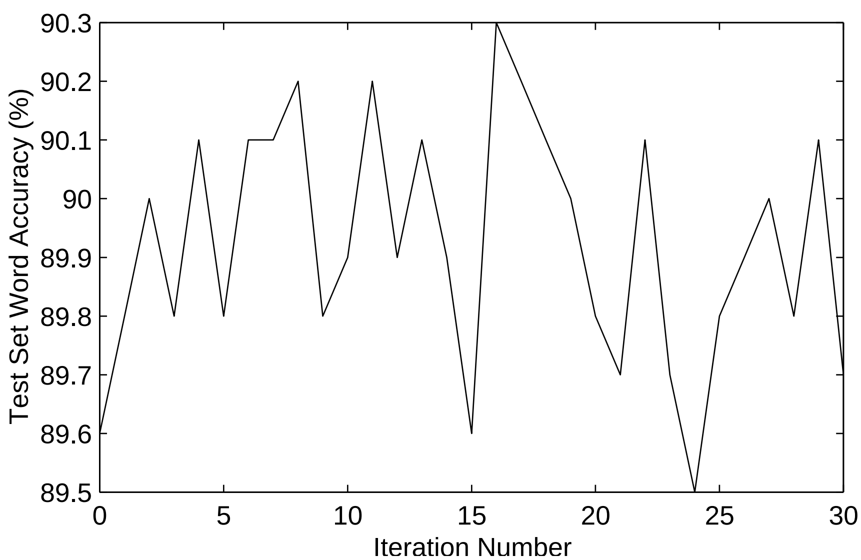

Figure 5-22: Word Accuracy on *test_500* vs. Iteration Index for Means Only Training, One Dimension

is that means are simply shifting between the same places again and again, with one Gaussian constantly taking the place of another Gaussian at any given point in the mean space. If this was the case, the accuracies might exhibit the periodic behavior seen in the figure. In any case, it appears that it is better to keep the variance alterations in the training.

5.2.3 Limiting the Maximum Alteration Magnitude

It was noted before that limits are placed on the maximum alteration magnitudes during training of the means and variances. This section explains why this is needed and how it is accomplished.

When training is performed without this maximum alteration limit, the maximum alteration magnitude does not tend to change very much from iteration to iteration. The average alteration magnitude does tend to decrease, however, so the ratio of the two gets greater and greater. Because of this, a few alterations begin to be much more important than any other alterations. This has a negative impact on the acoustic models. To illustrate this, figure 5-23 gives word accuracies on *test_500* for the same training run as in section 5.2.1 but without limits on the maximum alterations. An almost periodic pattern of the accuracies emerges, similar to the pattern when only means were altered. Clearly, the presence of the larger alterations causes the accuracies to become much more erratic, demonstrating the need to limit the size of any one alteration.

The technique used for limiting the alteration magnitudes is to employ the maximum alteration size from the previous iteration. With this scheme, the first iteration is allowed to proceed without limitation. For the second iteration, any alteration greater than some threshold in magnitude is reduced to have the threshold's magnitude. This threshold is set to be a constant (typically 0.75) times the maximum alteration magnitude from the first iteration. The maximum alteration magnitude for the third iteration is the constant times the maximum alteration magnitude from the second iteration, and so on for later iterations. Eventually, the maximum allowed alteration size will approach the average alteration size; by this time, however, con-

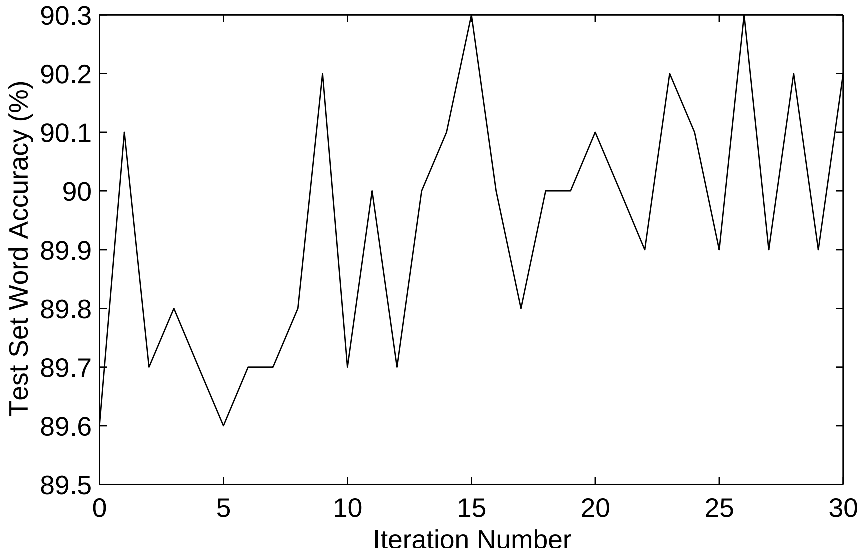

Figure 5-23: Word Accuracy on *test_500* vs. Iteration Index for Mean/Variance Training Without Maximum Alteration Limits, One Dimension

vergence of the parameters will have already taken place. Thus, most average-sized alterations are not affected by this magnitude limitation.

5.2.4 Increasing the Number of Dimensions Altered

Having been able to at least achieve a modest accuracy increase altering only one dimension of the means and variances, the next step is to try altering multiple dimensions at once. The hope is that the gains from optimizing various dimensions will to some extent be additive, resulting in higher test set accuracies than in previous experiments.

We again use the same training parameters as in previous mean/variance training experiments. For practical reasons, the number of dimensions altered is only increased from 1 to 3. Figure 5-24 shows word accuracies on *test_500* after each training iteration. Unfortunately, this training run does not seem to have been any more effective than the previous one; in fact, the test set accuracies settle at exactly the same value (90.1%). The potential for greater accuracy gains from optimizing larger numbers of dimensions seems limited at best.

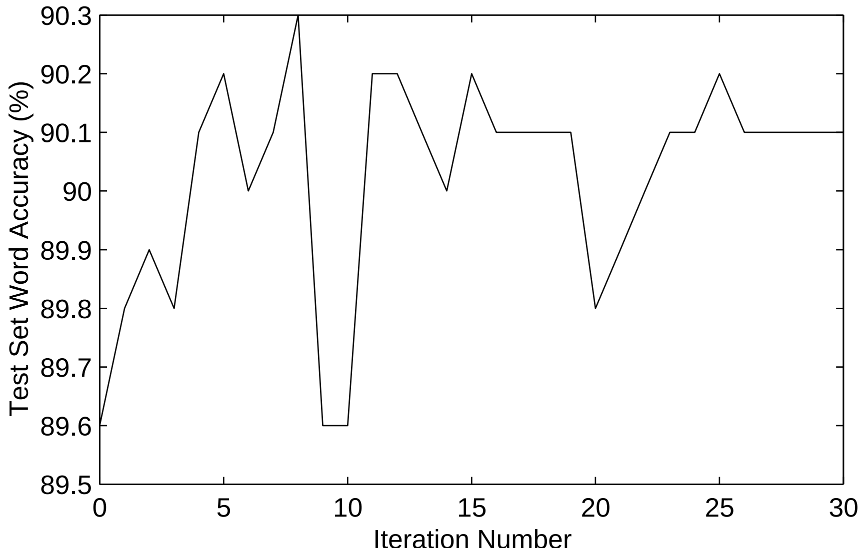

Figure 5-24: Word Accuracy on *test_500* vs. Iteration Index for Mean/Variance Training, Three Dimensions

5.2.5 Summary and Conclusions

In summary, altering the means and variances seems to be far less effective than altering the mixture weights. Accuracies of the output models tend to be very erratic, and a great deal of manipulation (e.g. limiting the maximum alteration magnitudes) is required to get the accuracies to converge at all. Training does not seem to improve as more and more dimensions are optimized.

In reality, the experiments here might even be an optimistic view of mean/variance training effectiveness. We have tried many different combinations of parameters and many slight changes to the training algorithm, always achieving little to no accuracy improvement. Usually the accuracies bounce around for many iterations and never seem to settle at any particular value. The reasons for this behavior are not completely understood. One possibility, that the Gaussians are merely shifting around and replacing each other at various points in the space, was mentioned above. The conclusion seems to be that this discriminative training algorithm is not well suited to the optimization of the means and variances.

5.3 Summary

This chapter discussed the results of a series of experiments designed to test the effectiveness of discriminative training in SUMMIT. First, we optimized the mixture weights. We compared the MCE and MMI criteria and found that they produced very similar accuracy improvements. The best relative reduction in word error rate achieved on the *test_2500* set was 5%. We found that for both MCE and MMI, increasing the number of competing hypotheses seemed to improve accuracies more than increasing the number of training utterances. Apparently this is because discriminative training does most of the good it can do with very few training utterances. We found that other training parameters could be set anywhere within wide ranges with negligible effect on the results. Training of the mixture weights generally seemed to have a smooth, well-behaved convergence characteristic.

We also attempted to optimize the means and variances using the same discriminative training algorithm. Very little, if any, accuracy gains were observed. Training seemed to be less well-behaved than when weights were optimized, and the resulting accuracies were usually very erratic. Optimizing the mixture weights is clearly preferable to optimizing the means and variances in this system.

Now that we have experimented with some of the established discriminative training algorithms, we may wish to see how they can be modified to better meet our ultimate goal of improved understanding of user queries. This is the focus of the next chapter.

Chapter 6

Keyword-Based Discriminative Training

Up to this point, the focus has been on improving overall word error rates in the SUMMIT recognizer. We have assumed that improving this metric will improve Jupiter's performance. But word error rate is not necessarily the most meaningful metric in a conversational system. Ultimately, it is only important that an utterance is properly understood, even if some words in the utterance are not recognized correctly. In this chapter, we attempt to focus our training efforts on this goal of better utterance understanding.

Certain words in any utterance are more important for correct understanding than others. For example, if the user says "weather in Boston please", the words 'weather' and 'Boston' must be recognized in order for the natural language unit to parse the sentence correctly. The words 'in' and 'please' are not very important; if the recognizer replaces 'in' with 'on', or 'please' with 'these', the natural language unit will still be able to understand the utterance and initiate the proper response. If we can somehow identify the important words and focus the model training on improving the recognition of just these words, perhaps we can maximize the overall system's effectiveness.

The following sections describe an attempt to implement such a training scheme. The first section defines hot boundaries, which are used to identify the more important segments of the training utterances. Next, we describe how we use these special boundaries to focus discriminative training on improving the recognition on a set of keywords. Finally, we analyze a keyword-based training experiment, and try to explain some unexpected results.

6.1 Hot Boundaries

As we have previously seen, when an utterance is to be recognized it is first broken down into a sequence of hypothesized boundaries, each with an associated feature vector. In our standard discriminative training experiments, no one part of a training utterance was more important than any other part, and thus there was no need to differentiate between the boundaries. Now we wish to focus the training on correcting only misrecognitions of keywords, so we will need a way to distinguish between the important and unimportant segments of the training utterances. The boundaries lying in the important segments are called the "hot boundaries."

The hot boundaries are determined during the statistics collection phase, and are chosen as follows. For each training utterance, a list of the keywords contained in the correct hypothesis is compared to lists of keywords in each of the competing hypotheses. When a competing hypothesis has the same keywords as the correct hypothesis, regardless of their position in the sentence, no hot boundaries are generated. When there is a mismatch in the keyword lists, all of the boundaries spanning all of the mismatched keywords are flagged as hot boundaries for the utterance. In this way, we hope to capture all of the regions of the utterance that have a high risk of causing a keyword recognition error.

The process of choosing the hot boundaries is best illustrated using several examples. These examples are shown in figure 6-1. Notice in the first example that even though the keywords do not appear in the same positions in both hypotheses, the keyword lists for the hypotheses are the same, and thus this pair does not produce any hot boundaries. The second example is straightforward, and shows that hot boundaries are only produced where there are mismatched keywords, not mismatches

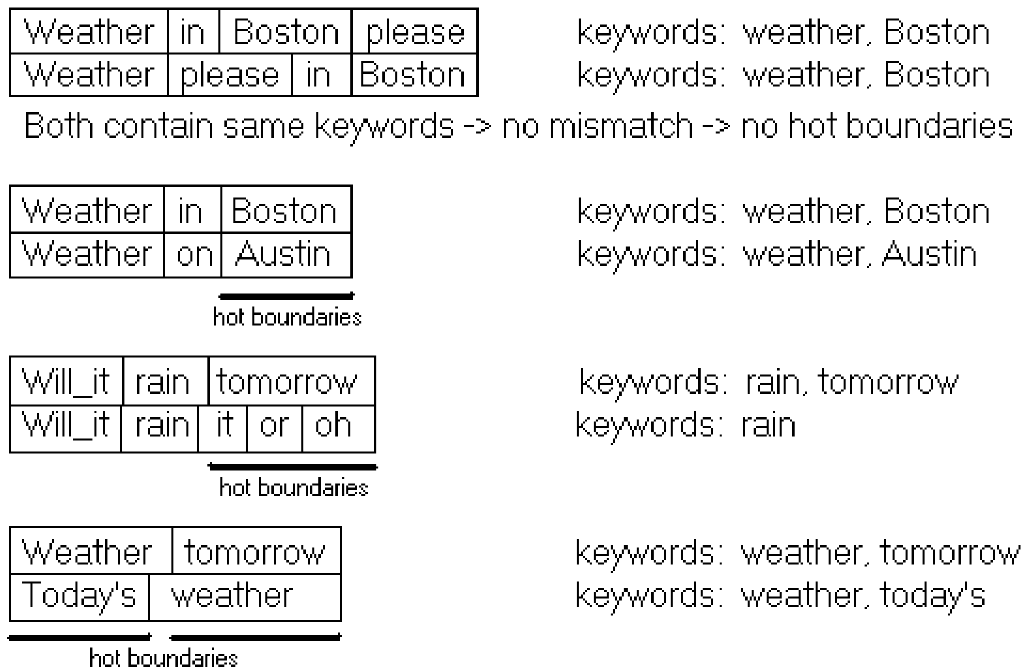

Figure 6-1: Examples of Choosing Hot Boundaries

in other words. The third example shows that insertions or deletions of keywords, not just keyword exchanges, produce hot boundaries. The fourth example shows that hot boundaries are produced at all positions spanned by a mismatched keyword in any hypothesis–in this case this includes almost the entire utterance.

A list of the hot boundary indices is saved out to the statistics file with the rest of the data for each utterance. The next section describes how the training procedure is changed to take advantage of this information.

6.2 Changes to the Training Procedure

Recall that on each iteration of discriminative training, three steps are taken: scores are computed from the training data, these scores are used to calculate an objective function, and derivatives of the objective function are used to alter the chosen parameters. We will modify the training procedure by changing only the first step, the computation of scores from the training data. This in turn affects the objective

Utterance has Non-Acoustic Score of -400
30 hot boundaries, sum of acoustic scores is -500
10 other boundaries, sum of acoustic scores is -300

Old Utterance Score = -400 + (-500) + (-300) = -1200
New Utterance Score = (-400)(30/40) + (-500) = -800

Figure 6-2: New Score Computation Using Hot Boundaries

function and the parameter alterations, and thus achieves the goal of focusing the training on the hot boundaries.

The scores used up to this point have been the sum of the acoustic and non-acoustic scores for each utterance hypothesis. The acoustic scores have been the sums of the scores at each boundary. To focus the training on the keywords, we can simply modify this score to be the sum of the scores at each *hot* boundary. Other boundaries do not contribute to the acoustic score. The data associated with those boundaries will then have no effect on the objective function, hence it can have no effect on the parameter alterations. Training is only being influenced by the important segments of the training utterances.

We should also scale down the non-acoustic score to account for the fact that less boundaries are contributing to the acoustic score. This keeps the relative importance of both types of scores roughly the same. In this scheme we simply multiply the true non-acoustic score by the fraction of the utterance's boundaries that are hot. This is not a perfect solution since it does not account for the influence of unimportant words on the language model, etc., but it turns out to be a reasonable approximation.

Figure 6-2 illustrates this new score computation process. With it in place, we are ready to run a training experiment to test the effectiveness of the keyword-based approach.

Place Names	Weather Terms	Dates/Times
Boston	snow	tomorrow
India	humidity	o'clock
m_i_t	weather	January
southeast_asia	advisories	sixth

Figure 6-3: Typical Keywords

6.3 A Training Experiment

The first issue to consider when running a keyword-based training experiment, which has not been addressed to this point, is how to obtain the list of keywords. We manually choose a list of 1066 words out of Jupiter's 1957 word vocabulary. Figure 6-3 illustrates typical words chosen as keywords; they are mostly place names, weather-specific terms, and date/time words. Some other words were included which were subjectively determined to be important for the natural language unit to parse the sentence. The complete list of keywords may be found in Appendix B. Automated means of determining keywords from the parse grammar may be possible, but have not been investigated here.

Once the keywords are in place, statistics can be collected and training can be performed. We will use the MCE objective function to adjust the mixture weights. The *train_18000* set will be used with N-best lists of 20 hypotheses, ϵ equal to 2400, and ρ set to 4.0.

The reason for choosing the *train_18000* set is that much of the training data goes unused. Any competing hypothesis with the same keywords as the correct hypothesis is thrown away during statistics collection, since it cannot result in any parameter optimizations. If every competing hypothesis gets thrown away, the entire training utterance gets thrown away. For this reason the number of training utterances and the average number of competing hypotheses will each be lower than the maximum amounts possible. As a first step it is useful to measure how much data is kept, since

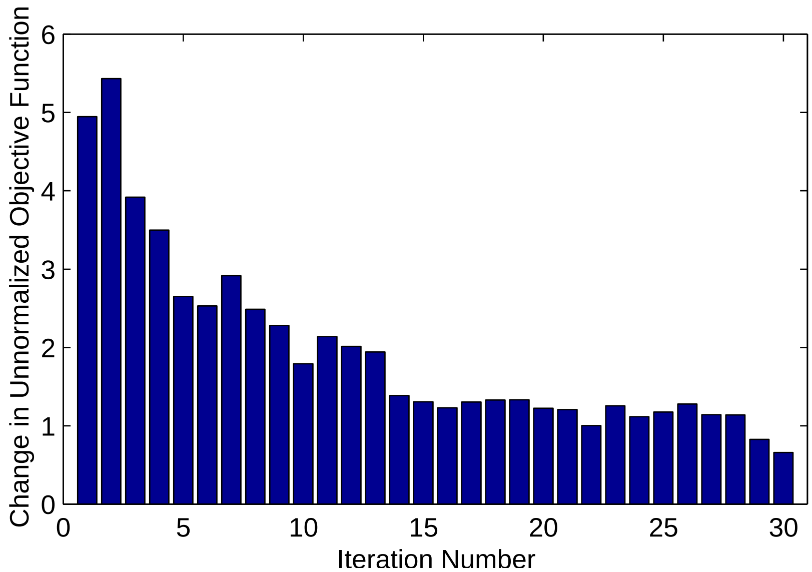

Figure 6-4: Change in Unnormalized Objective Function Value vs. Iteration Index for Keyword Training

this will establish which previous training experiments are most comparable. It turns out that 12,933 utterances survive the statistics collection phase (10,348 for training and 2,585 for evaluation). On average, the n-best lists for these utterances contain 5.2 hypotheses. Thus, the MCE weight training experiment which utilized *train_12000* and n-best lists of 5 hypotheses will be a good basis for comparison. Later on, we will examine results of keyword training on *train_12000* as well to verify that any extra accuracy gains are not solely due to the characteristics of *train_18000*.

Now we begin to analyze indicators of training performance taken during the training run. Figure 6-4 shows the change in the unnormalized objective function measured on the development utterances at each iteration. The pattern of convergence of the objective function looks very much the same as in previous training experiments. This is expected since the same training procedure is being used, only with different utterance scores. The average magnitude of the weight alterations at each iteration is shown in figure 6-5. The alterations also follow a familiar pattern of decrease. Overall, it seems that the use of the new utterance scores does not have much observable effect on the training indicators.

The next step is to see if this training really has the intended effect of producing

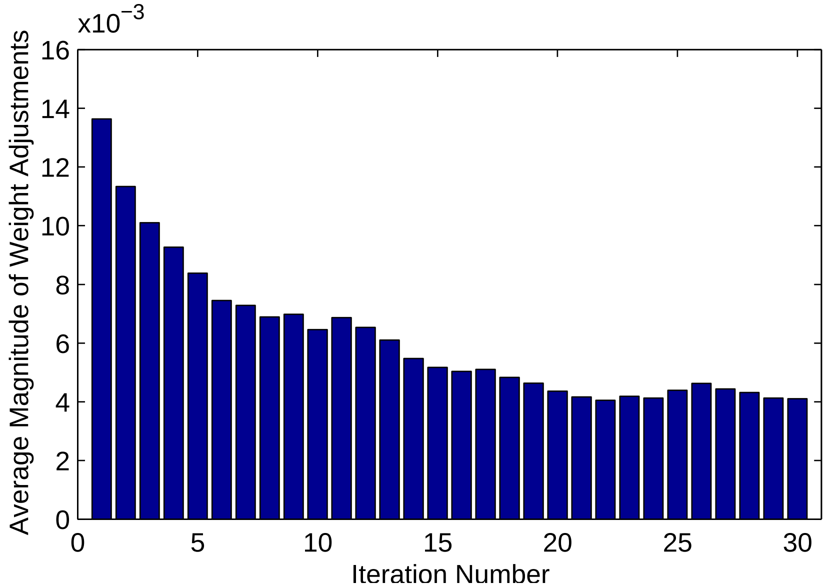

Figure 6-5: Average Magnitude of the Weight Alterations vs. Iteration Index for Keyword Training

Training Type	*test_2500* Keyword Accuracy
ML Models	86.1
MCE, *train_12000*, $n = 5$	86.2
MMI, *train_6000*, $n = 20$	86.4
MCE, *train_6000*, $n = 40$	86.4
Keywords, *train_12000*	86.9
Keywords, *train_18000*	87.1

Table 6.1: Summary of Keyword Accuracies for Various Types of Training.

higher keyword accuracies. Figure 6-6 shows the keyword accuracy on the *test_500* set at each iteration. The accuracies seem to be increasing fairly steadily. They increase quite a bit, starting at 94.0% and settling around 94.8%, a 13% relative reduction in keyword errors. This seems to give an initial indication that keyword training may be effective in improving the keyword accuracy.

For further corroboration of the training's effectiveness, we can observe the final accuracies on *test_2500*. We would like to see how much the keyword accuracy is improving relative to previous discriminative training runs. Table 6.1 gives the final keyword accuracies on this set for various types of training. An additional keyword training run using *train_12000* is included here; as mentioned earlier, this provides

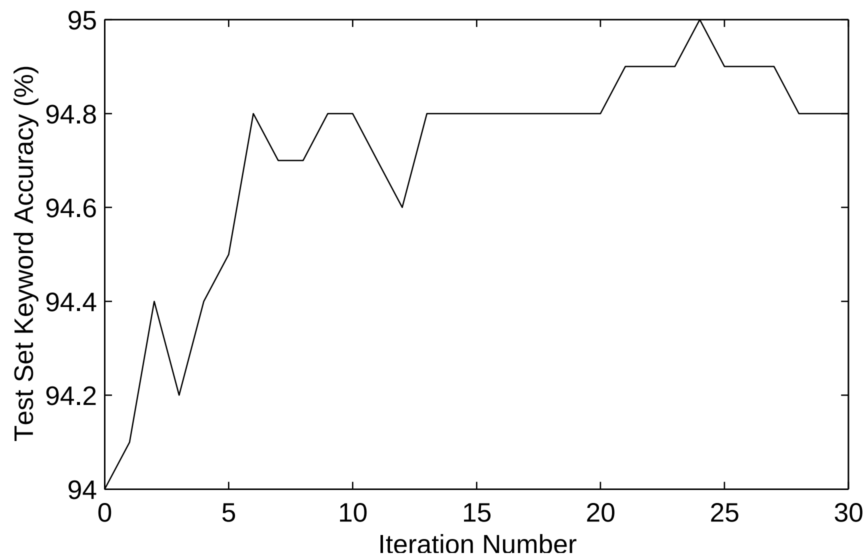

Figure 6-6: Keyword Accuracy on *test_500* vs. Iteration Index for Keyword Training

verification that any extra accuracy gains are not solely due to the characteristics of *train_18000*. As the table shows, the keyword-focused training improves keyword accuracies much more than the previous types of discriminative training. The previous training run with a comparable amount of data used (*train_12000*, $n = 5$) barely increased keyword accuracy at all relative to the ML models. The best that any previous training run did was to increase the keyword accuracy from 86.1% to 86.4%. Keyword-based training increases the keyword accuracy to 86.9% using *train_12000*, and all the way to 87.1% using *train_18000*, which means that about three times as many keyword errors are corrected. Clearly, our technique is causing the training to focus more attention on the most important words.

It is also informative to observe the effect of keyword-based discriminative training on overall word accuracy. We expect this accuracy to be roughly the same as the ML word accuracy. Since non-keywords are ignored during the training, we expect accuracy on these words to drop, offsetting the gains achieved on the keywords. This would result in little net change in the overall word accuracy. To test whether this is the case, we first observe the overall word accuracies on *test_500* at each iteration of the keyword training run using *train_18000*. These accuracies are shown in figure 6-7. Contrary to our expectations, the overall word accuracies improve very steadily as

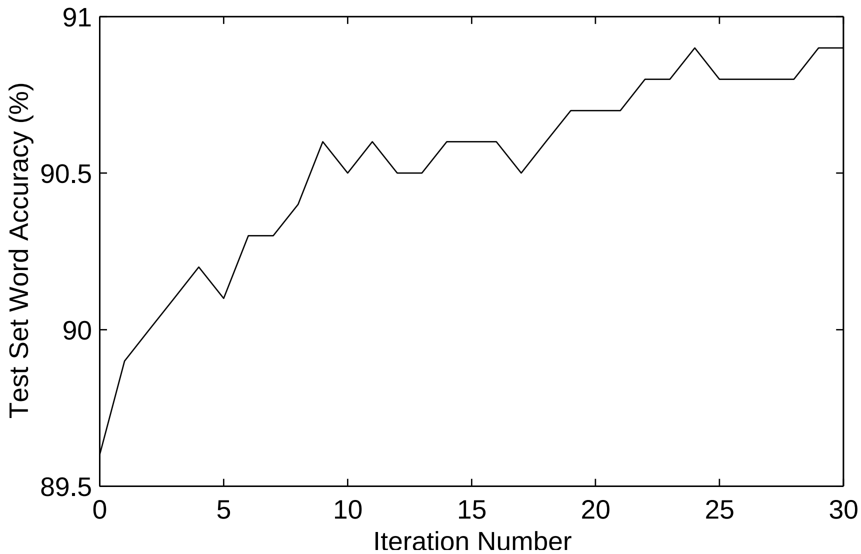

Figure 6-7: Overall Word Accuracy on *test_500* vs. Iteration Index for Keyword Training

Training Type	*test_2500* Word Accuracy
ML Models	81.7
MCE, *train_12000*, $n = 5$	82.2
MMI, *train_6000*, $n = 20$	82.5
MCE, *train_6000*, $n = 40$	82.5
Keywords, *train_12000*	82.6
Keywords, *train_18000*	82.7

Table 6.2: Summary of Overall Word Accuracies for Various Types of Training.

the iterations progress! They eventually settle at 90.9%; for previous training runs, this accuracy had settled at no higher than 90.5%. We seem to be improving the modeling not only for the keywords, but for all the words!

This point is verified in table 6.2, which gives the final overall word accuracies on *test_2500* for various types of training. This table shows that keyword training causes twice as much increase in overall word accuracy as the previous comparable training run (*train_12000*, $n = 5$). The accuracies of 82.6% using *train_12000* and 82.7% using *train_18000* are also higher than was achieved with any other previous training run. Keyword-based discriminative training not only optimizes recognition of the keywords, it also seems to optimize recognition overall.

It is quite surprising that modeling of words which are ignored during training is actually improved. The next section poses a hypothesis for why overall word accuracies increase for keyword-based training, and describes an experiment to test this hypothesis.

6.4 Omission of Unimportant Words

One hypothesis to explain why keyword-based discriminative training improves overall word accuracy is that training data from certain words actually makes the trained models less accurate. Suppose there are a set of words which are usually unstressed, poorly articulated, and not important to the meaning of an utterance. The acoustic features derived from within these words are likely to be erratic since a consistent manner of articulation is not always employed, i.e., the speaker may be sloppy in articulating these words. Thus the observation vectors for the various subword units may not tend to fall into neat clusters. When Gaussian means and variances are trained, they may be skewed away from their "correct" locations by the presence of these somewhat random observation vectors. Thus modeling of more precisely articulated realizations of the subword units may be harmed in an attempt to model realizations which are essentially unpredictable.

According to this hypothesis, the reason why keyword-based training would improve overall word accuracy is that it does not use data from these unimportant words when updating the model parameters. As a result, it should focus the training effort on ensuring that the more predictable vectors (the well-enunciated ones) are always classified correctly. This would not have much effect on the classification of vectors from the unimportant words, since these are spread across a large region of the feature space.

In this section, we run training experiments to see whether this hypothesis might be a good explanation of the gain in overall word accuracy. These experiments are run just like the previous keyword training experiments, except that a different list of keywords is used. This time, the list consists of all words in the vocabulary *except*

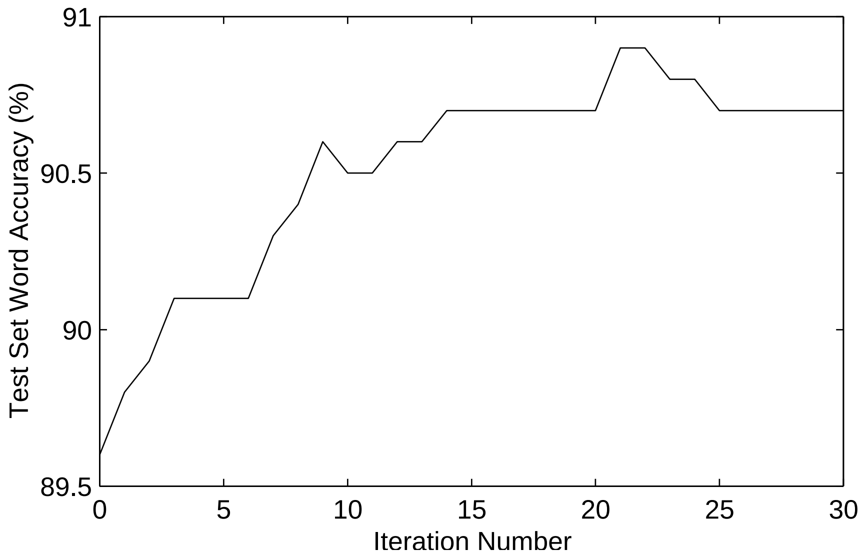

Figure 6-8: Overall Word Accuracy on *test_500* vs. Iteration Index for Omitted Words Training

a manually chosen set of 148 words. We placed any words in this set that we thought were unimportant to sentence understanding and were likely to be poorly articulated. The list of omitted words can be found in Appendix B. If our hypothesis is correct, we expect the overall word accuracies on the test sets to approach the levels achieved by the keyword training experiments, since the poor quality data is excluded from both training runs.

As a first indication, figure 6-8 shows the word accuracies on *test_500* at each iteration for omitted words training using *train_18000*. The pattern of accuracy increase looks similar to that of the corresponding keyword training experiment. In the end, the accuracies settle at 90.7%, which is slightly lower than the final accuracy for keyword training but higher than the accuracy for any previous training experiment. This is a good indication that the omission of the unimportant words is the most important factor in increasing overall word accuracies.

Again, we use final word accuracies on *test_2500* to confirm these results. Table 6.3 compares the final accuracies for omitted words training to the final accuracies of some previous experiments. Looking at the training runs using *train_12000*, the overall word accuracy jump from standard to keyword-based discriminative training

Training Type	*test_2500* Word Accuracy
ML Models	81.7
MCE, *train_12000*, $n = 5$	82.2
Omitted Words, *train_12000*	82.6
Keywords, *train_12000*	82.6
Omitted Words, *train_18000*	82.6
Keywords, *train_18000*	82.7

Table 6.3: Overall Word Accuracies for Omitted Words Experiments Compared to Other Types of Training.

seems to be completely accounted for by the fact that the unimportant words are omitted. With *train_18000*, the overall word accuracy is very nearly as high for the omitted words training as for the keyword training. One reason the accuracy might not go quite as high is that the manually selected list of unimportant words was not complete, and certain words that were used for the training still had a negative effect on overall accuracies. In any case, it seems clear that the omission of unimportant words from the training improves the overall quality of the acoustic models.

6.5 Conclusions

The experiments in this section have indicated that it is possible to improve the recognizer's accuracy on a subset of words in the vocabulary. This is useful for ensuring that the most important words in an utterance have the best chance of being recognized correctly. It is possible, however, that the benefits of this technique may be limited to domains similar to Jupiter: moderate-sized vocabulary, spoken dialog systems. Naturally, focusing on certain keywords seems less appealing in dictation systems, where it is equally important to recognize every word correctly. Also, as vocabularies get larger and larger, it becomes harder to identify a small set of keywords that are much more important than other words. For systems like Jupiter, though, keyword-based discriminative training seems to have the potential to offer substantial increases in utterance understanding rates. Since conversational systems like this are becoming more and more common, the technique might be applicable in many circumstances.

In our pursuit of lower keyword error rates we have also stumbled upon an insight into improving the acoustic models overall. It seems that preselecting certain higher quality parts of the training data can increase the general usefulness of the resulting models. This insight could have applications in types of training other than discriminative training. For example, if the training tokens derived from short, unimportant words are thrown out prior to ML training, it is possible that accuracy gains will still be observed. The experiments in this chapter support the idea that realizations of subword units can vary widely depending on their positions in an utterance, with some realizations being much more precise than others. Perhaps even greater accuracy gains are possible if this phenomenon is studied more closely and exploited to a greater degree.

Chapter 7

Conclusion

7.1 Thesis Overview and Observations

This thesis explored the use of discriminative training to improve word error rates in a segment-based speech recognizer. This was motivated by the fact that Maximum-Likelihood training does not take the likelihoods of incorrect classes into account, so potential confusions are ignored. We hoped to improve classifier performance by focusing the training directly on the minimization of erors on the training data.

Before discussing discriminative training, we presented some work on Gaussian Selection in Chapter 3. This technique reduced the number of Gaussian likelihood computations that needed to be evaluated to produce acoustic scores given a measurement vector. Recognition experiments showed that the number of Gaussians evaluated decreased by about 67%, resulting in a decrease in recognition time of about 45%.

Next we discussed our implementation of discriminative training in SUMMIT. We chose to use utterance-level scores on the training data in order to take linguistic context into account when updating model parameters. Both MCE and MMI objective functions were implemented, similar to those previously reported in the literature. We settled on a simple parameter update formula based on a step size and a derivative of the objective function with respect to the parameter being updated.

The results of a set of discriminative training experiments were similar to those previously reported in the literature. The MCE and MMI objective functions produced about the same accuracy gains when the mixture weights were updated. We found that updating the Gaussian means and variances did not produce very reliable gains; in fact the training behavior was usually quite erratic. The best result on *test_2500* for these experiments was a 5% relative reduction in word error rate compared to the ML-trained models.

In Chapter 6 we presented a modification to the discriminative training algorithm meant to focus more attention on the recognition of certain keywords. We modified the utterance score computation so that it was only composed of acoustic scores from certain "hot boundaries." Hot boundaries were defined as boundaries spanned by a mismatched keyword in one of the utterance hypotheses. This keyword-based training improved the accuracy gains on a manually chosen set of keywords by a factor of three relative to our previous experiments. We also found the keyword-based training produced higher gains in overall word accuracy than we had previously achieved. We attributed this to the fact that certain poorly articulated, acoustically unpredictable function words were no longer being used to update the model parameters.

A few observations arise related to the work in this thesis. First, it seems likely that discriminative training will be most useful in situations where the amount of training data is limited. As the amount of training data is subsequently increased, the discriminatively trained models do not get much better, while the ML-trained models tend to consistently improve. With this in mind, discriminative training might be used in initial deployments of a new conversational system, in order to produce the most accurate recognition as quickly as possible. Another use might be in more heavily context-dependent modeling (e.g. triphones or quinphones), where by nature the amounts of training data for certain models are always somewhat limited.

Another observation is that this work seems to confirm that there is little practical difference between the various forms of discriminative training. This suggests that research into new objective criteria and training procedures is unlikely to produce any substantial accuracy gains. The gains available from discriminative training can likely be exploited equally well with many different algorithms.

Finally, as discussed in Chapter 6, it seems possible to focus the training process on certain subsets of the training data. With different choices of the subset many different behaviors could possibly be obtained. For systems similar to Jupiter a wide range of applications of keyword-based training schemes can be imagined, which may all have different effects on the recognizer.

7.2 Suggestions for Future Work

As mentioned previously, there is likely not much more to be gained from research into discriminative training schemes similar to previously reported algorithms. Probably the most fertile area for future work related to this thesis lies in the exploration of topics related to our keyword-based training. Within the discriminative training framework presented here, much work could be done on refining the selection of keywords and the choice of hot boundaries. A different kind of non-acoustic score could also be developed based solely on the keywords in an utterance. Many different variations on the technique reported here are possible which could significantly improve the effectiveness of the training.

The concept of preselecting certain parts of the training data and throwing out the rest also need not be restricted to discriminative training. One could also try to select the training tokens from keywords and then use those for Maximum Likelihood training. If this produced gains in word accuracy similar to those of our keyword experiments, perhaps it would be more useful than discriminative training, due to the significantly reduced training time. Finally, further extensions could be explored where the keywords are modeled with one set of units and all other words are modeled with another set of units. This way the non-keyword data would not be thrown out, but instead used specifically to model phones from the set of non-keywords. This kind of scheme could prove the most useful in ensuring that the keywords are spotted while robustly recognizing the complete utterances.

Appendix A

Statistics File Format

The statistics file has the format of:

<*utterance-1*>

<*utterance-2*>

<*utterance-3*>

....

-300

The <*utterance*> has two possible formats, depending on whether hot boundary information is included. If it is not, the format is:

-200

<*hypothesis-1*>

<*hypothesis-2*>

<*hypothesis-3*>

....

<*dimensionality-1*>

<*vector-1*>

<*dimensionality-2*>

<*vector-2*>

....

If hot boundary information is included, an <*utterance*> has the format of:

-201

<*hot-boundary-1*>

<*hot-boundary-2*>

<*hot-boundary-3*>

....

-50

<*hypothesis-1*>

<*hypothesis-2*>

<*hypothesis-3*>

....

<*dimensionality-1*>

<*vector-1*>

<*dimensionality-2*>

<*vector-2*>

....

A <*hot-boundary*> is just an integer (a boundary index). A <*dimensionality*> is also just an integer–the number of measurements in the following vector. A <*vector*> is a sequence of floating point numbers.

The <*hypothesis*> has the format of:

-400

<*non-acoustic-score*>

<*class-index-1*>

<*class-index-2*>

<*class-index-3*>

....

-10 *or* -20

The <*non-acoustic-score*> is a floating point number. Each <*class-index*> is an integer–the index of the model corresponding to the next boundary in the hypothesis. A <*hypothesis*> is terminated with -10 if it is the correct hypothesis, and -20 if it is a competing hypothesis.

Appendix B

Lists of Words for Keyword Experiments

B.1 Keywords for Keyword-Based Training

a_m aalten aberdeen abidjan abilene abu_dhabi acapulco accumulation accumulation addis_ababa adelaide advisories afghanistan afternoon afternoon+s ainsworth airport akron alabama alaska albany alberta albuquerque alexandria algeria algiers algonquin_park allentown altus amarillo american_samoa amherst amman amsterdam anaheim anchorage anderson angola ankara ann_arbor annapolis anniston antigua appleton april argentina arizona arkansas arlington aruba asheville asia aspen astoria asuncion athens atlanta atlantic atlantic_city au_claire auckland august augusta augusta austin australia austria b_c b_w_i baghdad bahamas bahrain baja bakersfield bali baltimore bangalore bangkok bangladesh bangor bar_harbor barbados barcelona basel baton_rouge bc beaumont beijing beirut belarus belfast belgium belgrade belize bellevue bellingham belmont berkeley berlin bermuda bern bethesda billings binghamton birmingham bismarck bloomington boca_raton bogota boise bolivia bombay bonn bordeaux bosnia bosnia bosnia bosnia_herzegovina boston boulder bowling_green bozeman brasilia bratislava brazil brazzaville breckenridge bridgeport brisbane bristol britain british_columbia british_columbia brooklyn brownsville brunei brunswick brussels bucharest budapest buenos_aires buffalo bulgaria burbank burlington butte buzzards_bay bye cairo calcutta calgary cali california cambodia cambridge camden canary_islands canberra cancun cannes canton cape_cod cape_hatteras cape_may cape_town caracas caribou carmel carson_city casablanca casper cayenne cedar_city cedar_rapids celsius celsius centigrade central_asia champaign champaign_urbana chapel_hill charleston charlotte charlottesville charlottetown chatham chattanooga cheyenne chicago chile china christchurch christmas cincinnati cities city clearwater cleveland cocoa_beach coeur_d+alene cold cologne colombia colombo colorado colorado_springs columbia columbus columbus concord conditions congo connecticut copenhagen corpus_christi corsica costa_rica cozumel crete cuba cupertino curacao cyprus czech d_c d_f_w dakar dallas dallas_fort_worth damascus danbury danville dar_es_salaam darwin davenport davis day days daytime daytimes dayton daytona_beach dearborn death_valley december del_rio delaware delhi delmar denmark denver des_moines detroit detroit_lakes dewey_beach district_of_columbia district_of_columbia djibouti dodge_city dominican_republic dover dublin dubuque dulles duluth durango durham dusseldorf east east east_asia east_texas eastern eastern eastern_europe ecuador edinburgh edmonton egypt eight eighteenth eighth eighty el_dorado el_paso el_salvador eleven eleventh elkins elmira england england enid erie essen estonia ethiopia eugene europe evansville evening evening+s everett extended fahrenheit fairbanks fairfax falls_church falmouth fargo fayetteville february fergus_falls fifteenth fifth fifty fiji finland first five flagstaff flint florence florence florida forecast forecast fort_collins fort_knox fort_lauderdale fort_myers fort_smith fort_wayne fort_worth forty four fourteenth fourth framingham france francisco frankfurt fredericksburg french_guiana fresno friday friday+s gadsden gainesville galveston general geneva georgetown georgia germantown germany gibraltar glacier_park glasgow glenwood_springs goodbye grand_canyon grand_cayman grand_forks grand_island grand_rapids great_britain great_falls greece green_bay greenland greensboro greenville grenoble groton guadalajara guadaloupe guam guangzhou guatemala gulfport guyana hagerstown haiti halifax hamburg hampton hanoi harare harbin harriman harrisburg hartford havana hawaii heidelberg helsinki hidden_valley high high hilo hilton_head hingham ho_chi_minh_city hobart holiday holland hollywood honduras hong_kong honolulu hot hot hot_springs houston how_about humid humidity hungary huntington huntsville hutchinson hyannis iceland idaho idaho_falls illinois independence india indiana indianapolis indonesia innsbruck international_falls iowa iran iraq ireland irvine islamabad island israel istanbul italy ithaca ivory_coast j_f_k jackson jackson_hole jacksonville jakarta jamaica january japan jersey_city jerusalem johannesburg jonesboro jordan july june juneau jupiter kabul kahului kalamazoo kalispell kampala kansas kansas_city katmandu kenosha kentucky kenya key_west kiev kilimanjaro killington kingston kinshasa kittyhawk knoxville kodiak korea kourou kuala_lumpur kunming kuwait kyoto l_a la_guardia la_paz laconia lafayette lagos lahaina lake_powell lake_tahoe lancaster lansing las_cruces las_vegas latvia lawrence lebanon lebanon leesburg lewiston lexington lhasa libya lihue lima lincoln lisbon lithuania little_rock liverpool london london_heathrow long_beach long_island long_range los_altos los_angeles louisiana louisville low lowell luanda lubbock lusaka luxembourg lynn lyon macon madison madras madrid maine malaysia mali malta managua manchester manchester manhattan manila manitoba march marquette marseille martha+s_vineyard martinique maryland mass mass massachusetts massachusetts maui may medford melbourne memphis menlo_park meridian mexico mexico_city miami michigan milan milwaukee minneapolis minnesota minot minsk mississippi missoula missouri mobile mogadishu monaco monday monday+s montana monte_carlo montego_bay monterey monterrey montevideo montgomery month monticello montpelier montreal monument moorhead morning morning+s morocco moscow mount_mckinley mount_washington munich murmansk muskogee myanmar myrtle_beach nagano nairobi nanchang nanjing nantucket naperville naples nashua nashville nassau natchez nebraska nepal netherlands netherlands netherlands nevada new_bedford new_brunswick new_caledonia new_delhi new_hampshire new_haven new_jersey new_mexico new_orleans new_york new_york_city new_zealand newark newfoundland newport newport_news niagara_falls nicaragua nice nigeria nine nineteenth ninety ninth nome norfolk north north north_carolina north_dakota north_florida north_platte northern_europe northern_ireland northwest_territories norway nova_scotia november o+clock o+hare oakland ocean october ogden ohio oklahoma oklahoma_city olympia omaha one ontario orange_county oregon orlando osaka oslo ottawa oxford p_m pacific pakistan palm_beach palo_alto panama_city paraguay paramaribo paris park_city park_rapids pasadena pearson pegasus peking pendleton pennsylvania pensacola peoria perth peru philadelphia philippines phnom_penh phoenix pierre pine_bluff piscataway pittsburgh pittsfield pocatello poland port_au_prince portland portland portsmouth portugal pottersmill poughkeepsie prague presque_isle pretoria prince_edward_island princeton providence provincetown provo pueblo puerto_rico puerto_vallarta puson pyongyang qingdao quebec quebec_city queens quincy quito rabat rain rain rainfall raining raining raleigh raleigh_durham rapid_city redmond regina reno report reston reykjavik rhode_island rhodes richmond riga rio rio_de_janeiro rise rise riyadh roanoke rochester rockford rockville romania rome rotterdam russia rutland s_f_o sacramento saginaw saint_croix saint_george saint_john saint_kitts saint_louis saint_lucia saint_martin saint_paul saint_petersburg saint_thomas salem salina salisbury salt_lake salt_lake_city salzburg samoa san_antonio san_diego san_francisco san_jose san_jose san_juan san_luis_obispo san_salvador sanibel_island santa_barbara santa_clara santa_cruz santa_fe santiago santo_domingo sao_paulo sapporo sarajevo sarasota sardinia saskatchewan saskatoon saturday saturday+s saudi_arabia sausalito savannah scotland scotts_bluff scottsdale scranton seal_beach seattle second seneca_falls senegal seoul september serbia set seven seventeenth seventh seventy seville shanghai short_term shreveport sicily singapore sioux_city sioux_falls six sixteenth sixth sixty slovakia snow snowfall snowing snowmass sofia somalia south south_africa south_bend south_carolina south_dakota south_florida south_texas southeast_asia southern_europe spain spartanburg spokane springfield springfield springfield sri_lanka st_john st_johns st_petersburg stafford_springs stamford state_college stockholm stockville strasbourg stuttgart sun_valley sunday sunday+s sunnyvale suriname sweden switzerland sydney syracuse syria tacoma tahiti tahoe taipei taiwan tallahassee tallinn tampa tampere tanzania taos tashkent tasmania teen tegucigalpa tehran tel_aviv temecula temperature temperature ten ten tennessee tenth texarkana texas thailand thank_you thanks thanks thanksgiving third thirteenth thirtieth thirty thirty_first three thursday thursday+s tianjin tibet timbuktu today today today+s todays tokyo toledo tomorrow tomorrow+s tonight tonight+s topeka toronto torrence toulouse traverse_city trenton trinidad tripoli troy tucson tuesday tuesday+s tulsa tunis tunisia turkey tuscaloosa twelfth twelve twentieth twenty twenty_eighth twenty_fifth twenty_first twenty_fourth twenty_ninth twenty_second twenty_seventh twenty_sixth twenty_third twin_falls two u_k u_s u_s_a uganda ukraine united_arab_emirates united_kingdom united_states united_states_of_america urbana uruguay usa utah utica uzbekistan vail vancouver venezuela venice vermont vernal vero_beach victoria vienna vietnam vilnius virgin_island virginia vladivostok voyager waco wales warm warning warnings warsaw warwick washington washington_d_c waterbury waterloo watertown waterville weather wednesday wednesday+s week weekend weekend wellington west west_palm_beach west_texas west_virginia western western_europe what_about wheeling white_plains whitehorse wichita wichita_falls williamsburg williamsport wilmington windsor windy windy winnipeg winston_salem wisconsin woodland_hills woods_hole worcester wyoming xian yakima yangon yellowknife yellowstone yesterday yesterday+s yokohama yorktown yukon yuma zambia zimbabwe zurich

B.2 Omitted Words for Omission Experiment

<> <oh> <pause1> <pause2> <uh> <um> <unknown> a an and are as at be been blah but by can can_i can_you did didn+t do do_you does doesn+t doing don+t done else ever for from get give give_me go going going_on going_to got had has have having here hey hi how how_is i_am i_will i_would if in is isn+t it it_is it_will its let+s let_me like made make many may may_i more much my need next no not now of okay on once one ones only or our out said same say show so some sure than that that+s that_will the them then there there+s these they this those through to too towards until up us use very want want_to wanted was wasn+t we we+re were what what_are what_is what_will when when_is where where_is which who who+s why will with won+t would wouldn+t yeah yes you you+ve you_are your

Bibliography

[1] L.R. Bahl, F. Jelinek, and R.L.Mercer, "A maximum likelihood approach to continuous speech recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI 5(2), pp. 179-190, 1983.

[2] L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer, "Estimating Hidden Markov Model Parameters So As To Maximize Speech Recognition Accuracy", *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 1, pp. 77-82, January 1993.

[3] F. Beaufays, M. Weintraub, and Y. Konig, "Discriminative Mixture Weight Estimation for Large Gaussian Mixture Models", in *Proceedings of the 1999 International Conference on Acoustics, Speech, and Signal Processing*, pp. 337-340, March 1999.

[4] E. Bocchieri, "Vector Quantization for the Efficient Computation of Continuous Density Likelihoods", in *Proceedings of the 1993 International Conference on Acoustics, Speech, and Signal Processing*, pp. 692-695, April 1993.

[5] J.W. Chang, *Near-Miss Modeling: A Segment-Based Approach to Speech Recognition*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, June 1998.

[6] W. Chou, B.-H. Juang, and C.-H. Lee, "Segmental GPD Training of HMM Based Speech Recognizer", in *Proceedings of the 1992 International Conference on Acoustics, Speech, and Signal Processing*, pp. 473-476, March 1992.

[7] W. Chou, C.-H. Lee, and B.-H. Juang, "Minimum Error Rate Training Based on N-Best String Models", in *Proceedings of the 1993 International Conference on Acoustics, Speech, and Signal Processing*, pp. 652-655, April 1993.

[8] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, NY, 1973.

[9] J.R. Glass, *Finding Acoustic Regularities in Speech: Applications to Phonetic Recognition*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, 1988.

[10] J. Glass, J. Chang, and M. McCandless, "A Probabilistic Framework for Feature-Based Speech Recognition", in *Proceedings of the Fourth International Conference on Spoken Language Processing*, pp. 2277-2280, October 1996.

[11] J.R. Glass and T.J. Hazen, "Telephone-Based Conversational Speech Recognition in the JUPITER Domain", in *Proceedings of the Fifth International Conference on Spoken Language Processing*, pp. 1327-1330, December 1998.

[12] J. Glass, T.J. Hazen, and L. Hetherington, "Real-Time Telephone-Based Speech Recognition in the JUPITER Domain", in *Proceedings of 1999 International Conference on Acoustics, Speech, and Signal Processing*, pp. 61-64, March 1999.

[13] P.S. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo, "A Generalisation of Baum Algorithm To Rational Objective Functions", in *Proceedings of the 1989 International Conference on Acoustics, Speech, and Signal Processing*, pp. 631-634, March 1989.

[14] S.M. Herman and R.A. Sukkar, "Joint MCE Estimation of VQ and HMM Parameters for Gaussian Mixture Selection", in *Proceedings of the 1998 International Conference on Acoustics, Speech, adn Signal Processing*, pp. 485-488, May 1998.

[15] L. Hetherington and M. McCandless, "SAPPHIRE: An Extensible Speech Analysis and Recognition Tool Based on Tcl/Tk", in *Proceedings of the Fourth International Conference on Spoken Language Processing*, pp. 1942-1945, October 1996.

[16] B.-H. Juang and S. Katagiri, "Discriminative Learning for Minimum Error Classification", *IEEE Transactions on Signal Processing*, vol. 40, no. 12, pp. 3043-3054, December 1992.

[17] S.C. Lee, *Probabilistic Segmentation for Segment-Based Speech Recognition*, SM thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, May 1998.

[18] K. Knill, M. Gales, and S. Young, "Use of Gaussian Selection in Large Vocabulary Continuous Speech Recognition Using HMMs", in *Proceedings of the Fourth International Conference on Spoken Language Processing*, pp. 470-473, October 1996.

[19] Y. Normandin and S.D. Morgera, "An Improved MMIE Training Algorithm for Speaker-Independent, Small Vocabulary, Continuous Speech Recognition", in *Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing*, pp. 537-540, May 1991.

[20] Y. Normandin, R. Lacouture, and R. Cardin, "MMIE Training for Large Vocabulary Continuous Speech Recognition", in *Proceedings of the Third International Conference on Spoken Language Processing*, pp. 1367-1370, September 1994.

[21] Y. Normandin, "Maximum Mutual Information Estimation of Hidden Markov Models", *Automatic Speech and Speaker Recognition*, pp. 57-81, 1996.

[22] D. Povey and P.C. Woodland, "Frame Discrimination Training of HMMs for Large Vocabulary Speech Recognition", in *Proceedings of the 1999 International Conference on Acoustics, Speech, and Signal Processing*, pp. 333-336, March 1999.

[23] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, 77, pp. 257-286, 1989.

[24] R. Schluter, W. Macherey, S. Kanthak, H. Ney, and L. Welling, "Comparison of Optimization Methods for Discriminative Training Criteria", in *Proceedings of the 1997 European Conference on Speech Communication and Technology*, pp. 15-18, September 1997.

[25] R. Schluter and W. Macherey, "Comparison of Discriminative Training Criteria", in *Proceedings of the 1998 International Conference on Acoustics, Speech, and Signal Processing*, pp. 493-496, May 1998.

[26] N. Strom, L. Hetherington, T.J. Hazen, E. Sandness, and J. Glass, "Acoustic Modeling Improvements in a Segment-Based Speech Recognizer", in *Proceedings of the 1999 IEEE ASRU Workshop*, pp. 139-142, December 1999.

[27] V. Valtchev, J.J. Odell, P.C. Woodland, and S.J. Young, "Lattice-Based Discriminative Training for Large Vocabulary Speech Recognition", in *Proceedings of the 1996 International Conference on Acoustics, Speech, and Signal Processing*, pp. 605-608, March 1996.

[28] P.C. Woodland, C.J. Leggetter, J.J. Odell, V. Valtchev, and S.J. Young, "The 1994 HTK Large Vocabulary Speech Recognition System", in *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing*, pp. 73-76, May 1995.

[29] V. Zue, J. Glass, D. Goodine, M. Phillips, and S. Seneff, "The SUMMIT Speech Recognition System: Phonological Modelling and Lexical Access", in *Proceedings of the 1990 International Conference on Acoustics, Speech, and Signal Processing*, pp. 49-52, April 1990.

[30] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T.J. Hazen, and L. Hetherington, "JUPITER: A Telephone-Based Conversational Interface for Weather Information", *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 100-112, January 2000.