Rapid Speaker Adaptation for Neural Network Speech Recognizers

Daniel Clark Burnett
B.S., Harvey Mudd College, 1990

A dissertation submitted to the faculty of the
Oregon Graduate Institute of Science and Technology
in partial fulfillment of the
requirements for the degree
Doctor of Philosophy
in
Computer Science and Engineering

April 1997

The dissertation "Rapid Speaker Adaptation for Neural Network Speech Recognizers" by Daniel Clark Burnett has been examined and approved by the following Examination Committee:

Mark Fanty
Associate Professor
Thesis Research Adviser

Ronald Cole
Professor and Director
Center for Spoken Language Understanding

Hynek Hermansky
Associate Professor

Jordan Cohen
Member of the Research Staff
Center for Communication Research
Institute for Defense Analyses

Acknowledgements

I most gratefully acknowledge the support of the National Science Foundation and the Office of Naval Research for their financial support on grants GER 9354959 and N00014-94-1-1154, respectively. In addition, there are many individuals to whom I owe a great deal of thanks for their support these past few years. In addition to thesis committee members Jordan Cohen and Hynek Hermansky, extra thanks go to:

Mike Noel, for his help with the **Huh?** corpus data collection and many, many discussions during the past six years.

Don Colton, whose behavior gave me many good examples of how to get things done.

Kay Berkling, who pointed me to books on how to write that I should have read five years ago.

Philipp Schmid, who was always there as a backdrop against which I could splatter my ideas and rebuild them into a more cohesive whole.

Ronald Cole, who both supported me through difficult times (for both me and CSLU) and gave me opportunities to show what I could and could not do.

Mark Fanty, who pushed without shoving.

Bill McDonald, who helped me remember and rediscover why.

And my wife, Marina, whose support cannot be measured, cataloged, or even comprehended.

Contents

List of Tables

List of Figures

Abstract

Rapid Speaker Adaptation
for
Neural Network Speech Recognizers

Daniel Clark Burnett

Supervising Professor: Mark Fanty

The most successful speaker adaptation approaches today are, first, designed for use in a Hidden Markov Model-based recognizer, and second, either too slow or require too much adaptation speech. This thesis examines approaches for rapidly adapting neural network-based speech recognition systems to a new speaker using little adaptation speech, in some cases only a single word.

Two methods are described, both of which are rapid enough to be used on-line. The first method is a model-based retraining of weights in the neural network, while the second is a feature-space parameter adjustment. In the first approach, the goal is to improve performance on a target word for a given speaker without seriously degrading performance on the other words in the vocabulary. This is accomplished by retraining the hidden-to-output weights using a combination of original training speech and new target-word speech. Training only weights to the target outputs dramatically reduces the number of vectors (and hence time) needed for retraining. To avoid hurting performance on non-target words, an incremental retraining scheme is developed which trains only after encountering an error on a target word. The initial retraining only slightly changes the weights, with each successive retraining being slightly more aggressive. On average, this approach reduced the target word error rate by 84% while increasing the non-target error rate by 3%.

In the second approach, the recognizer score for the adaptation speech is directly optimized by varying a *Bark offset* parameter, a parameter which raises or lowers frequency components on the Bark scale. In the first of three experiments, adapting an adult-speech trained recognizer to children's speech reduced the children's error rate on a connected digit task by 35% using only a single digit as adaptation speech. In the second experiment, normalization to adults' digits reduced the error rate for adults by 32% using only a short (3–5 digits) string. A final experiment on a medium-vocabulary isolated word task produced a word-level error rate reduction of 7.5% using only 8 words as adaptation speech. For those speakers whose baseline performance was between 40 and 70%, the improvement was 13.3%.

Chapter 1

Introduction

It is important that real-world speech recognition systems work for as many people as possible; otherwise someone may be denied a service or a business may lose a customer. Unfortunately, even high-performance recognition systems occasionally encounter individual speakers for whom performance is low. An important strategy for achieving the maximum user throughput is to rapidly (during one session) adapt the recognizer. Such rapid adaptation will necessarily be based on a small amount of data—perhaps a single word.

A simple example is that of a voice-only World Wide Web browser that allows one to say common commands such as *help*, *back*, *home*, and *reload*. For the sake of argument, assume that the recognizer accurately recognizes these words 99% of the time. Now a user comes along, dials this WWW browser on his phone, and the first time he says *back* the system gets it wrong but gives him a chance to correct it. The next time he says *back* the system messes up again, and the next time, and the next time, ... Imagine how frustrating this is—he will probably hang up after the third time it misses the word! For this person, the system is useless. In a case such as this, the need to improve performance for the speaker on this one word is critical. Moreover, the improvement must be done quickly.

For a slightly different example, consider a system designed to automatically collect U.S. Census information over the telephone. The goal is to have a phone number that people can call rather than having to write down information onto a form. For the system to succeed, it must obtain all of the information and must obtain it correctly. From the caller's point of view, the system should be easy to use and not require the caller to repeat himself or call back later. This system would need to work well for an extremely large number of people. What happens if the system misrecognizes a word in this case? After fixing the error, the caller probably will not say the same word again. However, the system might easily misrecognize a *different* word. If the system misses too many words, the caller may just give up. Again, it is critical that the system's performance rapidly improve for the speaker. Unlike in the previous example, here the system must improve performance for *any* word the person might say.

Both of the above examples illustrate the need for speaker adaptation, the first one in the restricted domain of target word adaptation for a speaker and the second one in the domain of general speaker adaptation. For Hidden Markov Model-based recognition systems the problem of speaker adaptation has been studied quite extensively. A common and successful adaptation approach for these systems is to adjust the means and variances of the various models. Since each model is typically specified by a few parameters exclusive to that model, one can individually adjust the different models. Systems following this approach encounter problems when adapting all of the models, since several examples for each model are typically required in order to accurately estimate appropriate new values for the parameters. However, this can be solved by computing and using correlations between the parameters (e.g., mean and variance differences) for different speakers.

It would be convenient to adapt these techniques for neural network-based recognizers. Unlike an HMM, though, a single global neural network does not have explicit mean and variance values for each of the categories being modelled. Speaker adaptation techniques for neural network-based recognizers have instead focused either on adding additional input, hidden, or output layers or on completely redesigning the network architecture.

Our goal is to develop adaptation methods that are simple, rapid, and effective for our neural network-based recognizer. Specifically, we examine two different approaches to adaptation: a model-based rapid retraining of the network and a parameter-optimization feature space method. In this dissertation we present descriptions of and experiments with these two methods. The first approach reduced the target word error rate on a confusable word task by 84% while increasing non-target word error rates by 3%. The second approach reduced the error rate for children's speech on an adult-trained recognizer by 35%. In both cases only a single adaptation word was needed.

1.1 Thesis Outline

In this dissertation we first discuss the use of neural networks for speech recognition, followed by a review of speaker adaptation methods. We then present two methods for adapting neural network speech recognizers to new conditions: rapid retraining and simple parameter optimization. After presenting our experiments and associated results, we discuss our results and present some conclusions and thoughts on ways to extend and/or combine the two approaches.

Chapter 2

Speech Recognition with Neural Networks

This chapter describes the speech recognition problem and how it may be addressed using neural networks as probability estimators. Each description will consist of two parts: a general description and a detailed description of what is used in the experiments for this thesis.

2.1 The Speech Recognition Problem

The speech recognition problem is one of decoding. Given a word, phrase, or sentence spoken by a person, a spoken language system (SLS) must decode the utterance into a sequence of known words or phrases or determine that such decoding is beyond the system's capabilities. Most such systems in use today can be described in terms of four components: signal processing, feature selection, subword modeling, and lexical search. These components, shown in Figure 2.1, are described below.

Signal Processing

The speech utterance, converted from pressure variations in the air into an electrical signal, is first digitized. The digitized signal, a simple time vs. amplitude waveform, is then analyzed to determine the strengths of the frequency components that make up the signal. Further analysis may be performed to emphasize, de-emphasize, shift, scale, or otherwise change these frequency component strength values to produce values more

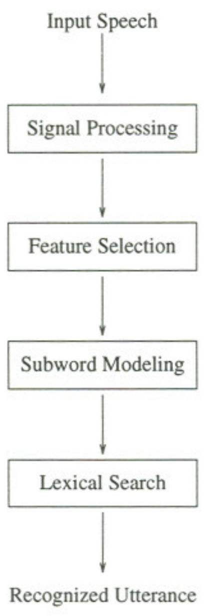


Figure 2.1: Components of a Spoken Language System.

suitable for later processing stages. Typically a new set of such values is computed at regular time intervals.

The recognizers used in this thesis work were originally designed to work with telephone speech utterances digitized at a rate of 8000 samples per second. Since the recognizers expect to receive samples digitized at this rate, any dataset used in the experiments which was not originally sampled at this rate was appropriately filtered and up- or down-sampled to 8kHz. Each 10ms section of this waveform (a *frame*) is analyzed for frequency components using either the PLP[22] algorithm or a variant of the algorithm known as RASTA-PLP[23]. This seventh-order PLP analysis produces 8 cepstral coefficients that compactly model a smoothed frequency spectrum.

Feature Selection

The values from the previous stage are then grouped to provide a set of features. To this set may be added other measures obtained from the original signal such as pitch and voicing information, energy estimates, change in frequency component strength values, and so on. This grouping is usually either *frame-based* or *segment-based.* Typically frame-based processing groups identically at regular intervals, while segment-based processing groups based on the notion of an acoustically or otherwise relevant segment within the speech. The segment may be a sentence, word, syllable, sub-syllable, or something even smaller.

The feature grouping in our system is frame-based. Each frame is processed using the same grouping; specifically, the 8 cepstral coefficients for the current frame, along with the 8 coefficients from each of six nearby frames (covering a total of approximately 160ms of speech), are connected to form a 56-element feature vector for the frame.

Subword modeling

Previously-trained subword models are next used to generate emission probabilities for a set of feature vectors. Typically there is a model for each of the designated subword units in the system (phone, phoneme, diphone, triphone, syllable, etc.) which estimates the probability that the set of vectors could have been generated by speech from that particular subword category. This subword modelling stage thus produces a set of emission probabilities over time.

Our system models context-dependent (CD) phonemes using a three-layer feed-forward artificial neural network. The network maps a single feature vector for each frame to a new vector of CD phoneme posterior probabilities. Dividing each probability by the corresponding prior probability for the subword category results in a vector of likelihoods that can be used as emission probabilities. Thus the input to the next processing stage is a set of CD phoneme probabilities over time. The structure and training of this network are described in Section 2.2.

Lexical Search

Word models and grammars specify how to represent words, phrases, and sentences in terms of the subword units. This knowledge of the allowed sequences (often with associated transition probabilities) of subword units, along with the time-varying probabilities for the subword units just computed, can be used to determine the most likely word, phrase, or utterance represented by the speech signal.

In our system, the lexical decoding is accomplished through a Viterbi search.[18] The search uses the CD phoneme probabilities, empirically-determined minimum and maximum durations for each CD phoneme, and a dictionary of word pronunciations to determine the most likely utterance. Details of the word models and grammars vary by experiment and are thus included in the appropriate experiment descriptions.

2.2 Hybrid Systems

Our system is actually a neural network/Hidden Markov Model (HMM) hybrid. Commonly, HMM systems model subword unit emission probabilities using multivariate Gaussian distribution functions or mixtures of these functions. This has both advantages and disadvantages. One advantage is that the statistical formulation of these functions is convenient to integrate into the statistical framework underlying the Hidden Markov Model. The word models and grammar can also be described within the same probabilistic framework, as can the lexical search. This framework allows one to treat the encoding and decoding of speech as a statistical process from beginning to end. On the other hand, a disadvantage of using these convenient yet arbitrary statistical distributions and model topologies is the assumption that the emission probabilities for each speech unit can be adequately modelled in this way, since the true distributions may in reality be quite non-Gaussian.

Instead of using these statistical distribution functions to encode the relationship between input feature likelihoods and individual subword units, one can use an artificial neural network. In fact, one can actually use a single network to simultaneously compute the probabilities for all of the subword units, as we do. This *hybrid* approach allows one to retain the convenient time sequence modelling properties of HMM's while gaining the less restrictive feature-modelling characteristics of neural networks. For a more detailed discussion of such hybrid systems, see [6].

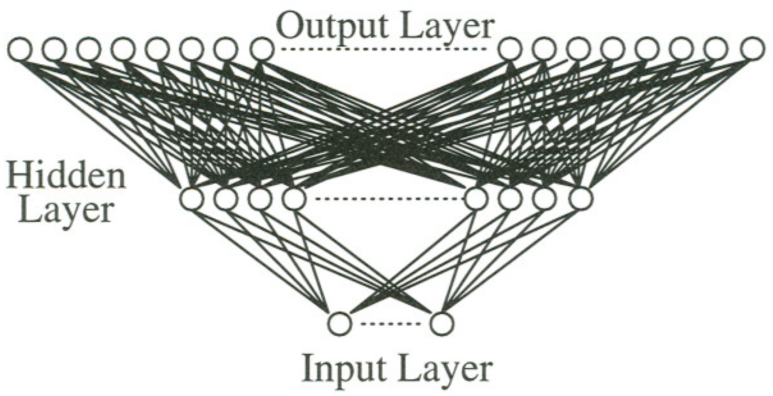


Figure 2.2: This figure shows a three-layer neural network. Weights connect all input units to all hidden units and all hidden units to all output units. All connections (weights) are feed-forward only.

2.2.1 Architecture

In this and following sections we give a brief description of our network architecture and how we train and use it for recognition. Our network architecture is shown in Figure 2.2. The network has 3 *layers*: an input layer, a hidden layer, and an output layer. Each layer consists of a number of *units*, each of which has a value. There are weights connecting every input unit to every hidden unit and weights connecting every hidden unit to every output unit.

All of the networks in our experiments have 56 input units and 200 hidden units. The number of output units varies for different experiments and is given in the appropriate experiment sections.

2.2.2 Evaluation and Training

During recognition, the feature vector for a given frame is presented as input to the network by setting the input units' values equal to their corresponding feature vector values. Feed-forward computation is illustrated in Figure 2.3. The value of any hidden unit h_x is computed as follows. For each input unit, the product of that input unit's value and the weight connecting the two units is computed. To the sum of these products is added an offset value. Finally, the function $f(x) = \frac{1}{1+e^{-x}}$ maps this value to one in the range (0,1) to produce the final value for the unit. The value of an output unit is computed similarly

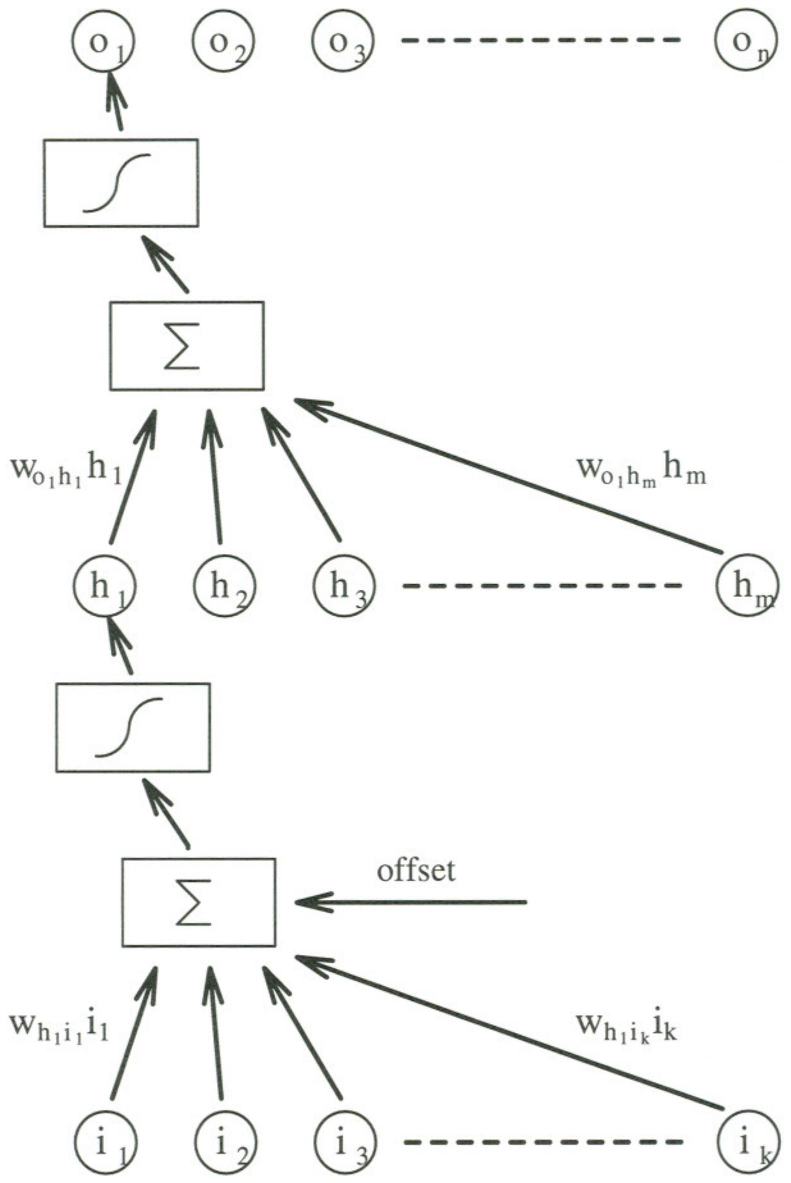


Figure 2.3: [Feed-forward computation.] This figure shows the computation of a hidden unit's value using the input unit values and the input-to-hidden weights. It also shows the similar computation of an output unit's value using the hidden unit values and the hidden-to-output weights. Each value is computed by summing the product of the prior layer's values with the corresponding weights, then compressing this value with a sigmoid function to ensure that the final value is within the range of 0 to 1. In the figure, k is the number of input units, m is the number of hidden units, n is the number of output units, l_z is the activation of the z^{th} unit in the l^{th} layer (where l is i,h, or o for the input, hidden, and output layers, respectively), and w_{xy} is the weight from the y^{th} unit to the x^{th}.

from the hidden unit values, but no offset is used to adjust the weighted sum before the sigmoidal compression. The output values, all between zero and one, are treated as an array of posterior probabilities for the categories modelled by the net. For example, the first output may represent the probability that the input values for the current frame came from a silent region of speech, the second output might represent the probability that the input values for this frame came from an /ε/ (the vowel found in the word "less") and so on.

Gradient descent (back-propagation) We use gradient descent (the back-propagation algorithm) to train our network.

The general idea behind the gradient descent algorithm is, first, to compute an error for the output of the network when given an input pattern and, second, to adjust the weights in the network in a manner which decreases this error. The change in the output error as a function of a change in any given weight is referred to as the *gradient* of the error with respect to the weight. To decrease the error, each weight is adjusted proportionately to the negative of its gradient. This is referred to as gradient descent because the repetition of this process for many input-output pairs results in a gradual "descent" through the multidimensional error function to a minimum of the function.

The back-propagation algorithm [51] is a procedure for computing these gradient functions and the corresponding weight updates. For each training input pattern, a forward evaluation produces output values. The error between the actual and desired outputs is then computed and "propagated back" to the inputs. The error of the hidden-to-output weights is based on the error at the outputs. Likewise, the error of the input-to-hidden weights is based on the error at the hidden units, which is computed using the error at the outputs and the hidden-to-output weights, hence the term back-propagation. For a good introduction to this algorithm, see [5, 39]. A more detailed mathematical analysis of this and other training algorithms can be found in [54].

Specifically, our network is trained as follows. For each feature vector in the training set,

- This vector is copied to the input units of the network, and the network outputs are computed as described earlier.

- The network error for this vector is computed using the cross-entropy error function

$$E = \sum_{j=1}^{n} d_j \log(o_j) + (1 - d_j) \log(1 - o_j)$$

where n = the number of output units, o_j is the actual value for output unit j, and d_j is based on the phonetic category c from which the feature vector was generated:

$$d_j = \begin{cases} 1 & \text{if the } j^{\text{th}} \text{ category is } c \\ 0 & \text{otherwise} \end{cases}$$

- For each weight w_{ji}, δw_{ji} is computed using the back-propagation algorithm. Each weight is then immediately updated by setting the new value of $w_{ji} = R(l) \cdot \delta w_{ji}$. $R(l)$, the *annealed learning rate*, is given by

$$R(l) = \frac{r}{1 + \frac{l}{5t}}$$

where r = nominal learning rate, t = total number of vectors used in training, and l = total number of vectors seen so far. For example, after 5 iterations through the training set the actual learning rate is half the nominal rate.

After the weights are updated, the process repeats for the next training pattern. In this thesis, a single pass over all of the training examples is referred to as an *iteration.*

Note that this approach uses stochastic, or on-line, gradient descent to update the weights and offsets, meaning that weight adjustments are performed after every training pattern. It is possible to accumulate weight updates over a large number of training examples and then apply them all at once, producing a more stable descent in the error space. We have found, however, that combining stochastic training with an annealing factor to gradually reduce the sizes of the weight adjustments results in more rapid training with only a slight reduction in performance. In fact, for the large networks used in this research this approach is the only one that allows training to complete within a reasonable amount of time.

Cross-validation One issue of concern when training neural networks is knowing when to stop. Given sufficient hidden units, a network will eventually learn the training patterns perfectly. More desirable for speech recognition, however, is for the network to generalize such that new unseen inputs will still be matched to the appropriate output values. The training needs to be stopped after it has (mostly) learned the training patterns but before it overtrains to a point where performance on new patterns suffers. In our training we use a *cross-validation set* to accomplish this. After each iteration, the network is tested on a separately-generated set of vectors, the cross-validation set. Performance typically increases on this set up to a point, levels off, and then drops again with more training iterations. The network with the highest cross-validation set performance is selected as the result of the training.

Chapter 3

Review of Adaptation Literature

Although the techniques described in this thesis are not limited in applicability only to speaker adaptation, the motivation for this work and the experiments performed primarily fall into this domain. In this chapter we present a review of speaker adaptation approaches, covering both a description of recent successful approaches and brief comments on why these methods may or may not be applicable to the problem we address. We conclude with an explanation of the context within which this thesis work should be viewed.

3.1 Speaker Adaptation/Normalization

Before discussing the history of speaker adaptation, it is appropriate to present both some relevant definitions and some dimensions often used to categorize adaptation algorithms.

In Chapter 1, we presented some motivations for speaker adaptation. The process of adaptation is illustrated in Figure 3.1. The basic idea is to use some amount of speech, called *adaptation* or *enrollment* speech, to improve the recognition accuracy of an existing SLS on other utterances by the same speaker. The speech may be used in several different ways: to change other incoming speech from the speaker, to change the internals of the recognition system, to change the output from the recognition system, or even to select the ideal from several different recognition systems. With so many varied ways of adapting to a speaker it can be difficult to categorize any given system. However, there are several dimensions and distinctions frequently used in the literature:

Normalization or Adaptation For adaptation, one takes an existing trained SLS and, using some amount of speech from the new speaker, performs a transformation on

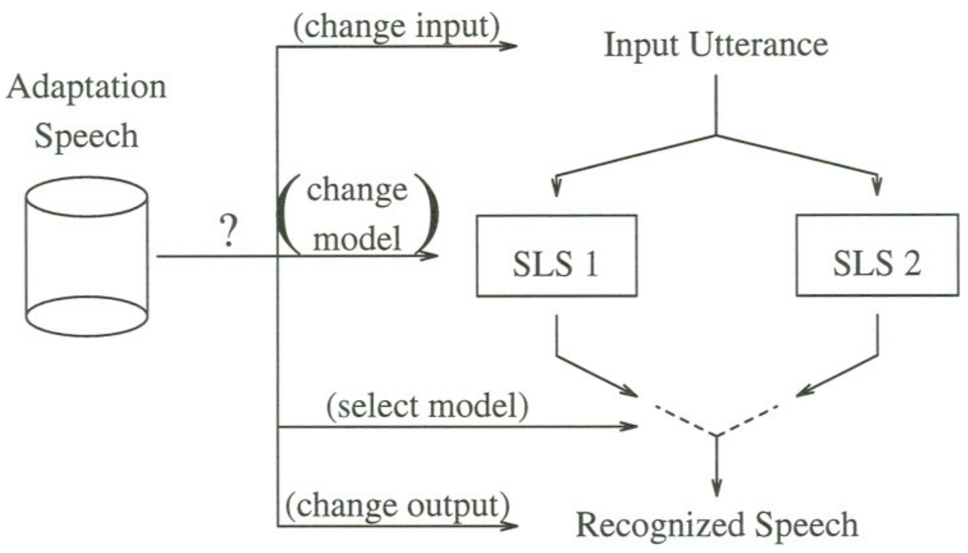
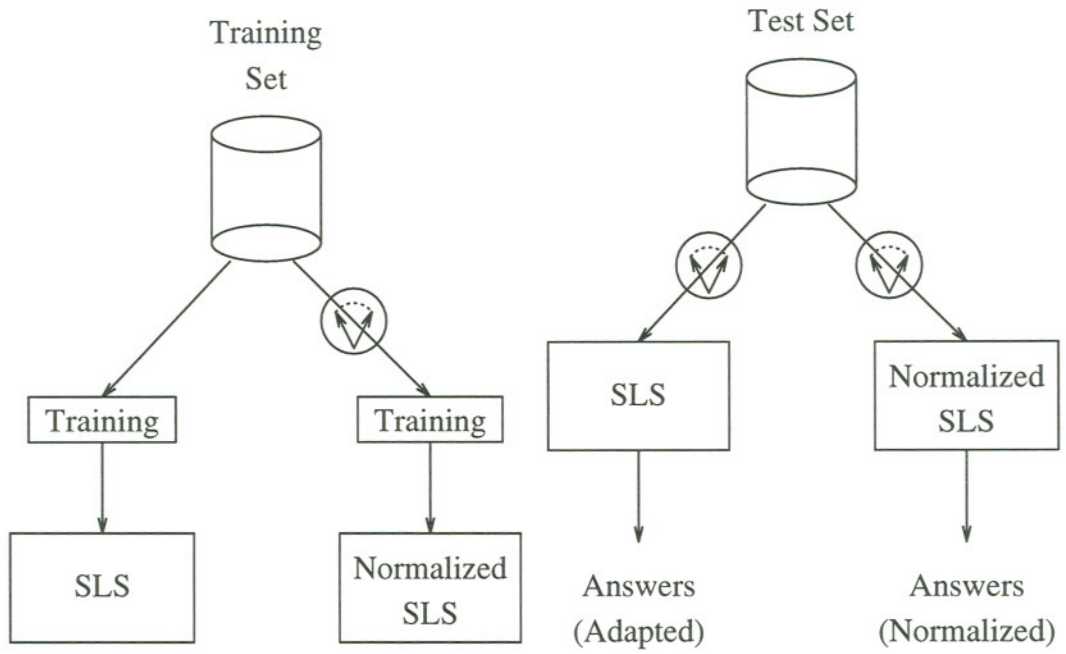


Figure 3.1: This figure shows the process of adaptation. Adaptation speech can be used to modify other incoming speech, the recognition system itself, or the recognizer output. It can also be used to select the recognition system out of a set.

the system, the new speaker's speech, or both to improve evaluation performance on the speaker. With speaker *normalization*, the transformation is performed during training, i.e. all speakers are normalized to appear to be the same to the SLS. The normalization process is, of course, needed for the evaluation of the speech of the new speaker as well. Note that for a normalized system it would be unnatural *not* to perform the transformation. Adaptation, on the other hand, is an extra (optional) procedure that one executes to improve performance for a given speaker. These two differences are presented diagrammatically in Figure 3.2.

Supervised or Unsupervised *Supervised* adaptation approaches require that the identity of the enrollment speech be known, i.e. the system must know the words the person said. To know unequivocally the words used in the adaptation speech, it may be necessary for the system either to direct the speaker in what to say, to confirm with the user a guess made by the system, or to have the user spell, type, or write what was said. *Unsupervised* adaptation approaches, on the other hand, do not require the system to know what the person said. Unsupervised adaptation is usually more desirable, since it can be performed without any special requirements on the



(a) Training (b) Evaluation

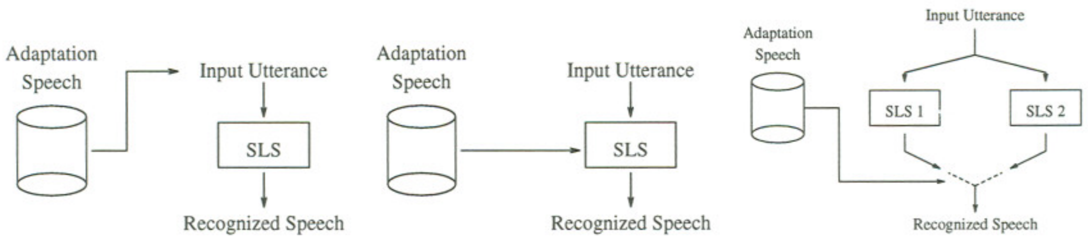
Figure 3.2: Adaptation vs. Normalization. This figure shows the difference during (a) training and (b) evaluation between adapation and normalization. The difference is that adaptation is also performed when training a normalized system.

speaker.

On-line or Off-line With off-line methods, the adaptation speech is separate from the speech to be recognized and must be uttered and processed ahead of time. On-line methods, on the other hand, use the utterances to be recognized as adaptation utterances.

Data Transformation, Model Transformation, or Speaker Categorization When adapting to the speech of a given individual, one can generally either transform the data or the model(s). However, it is also possible to merely build different systems for different groups of speakers. This distinction is shown in Figure 3.3.

For ease of comparison we will primarily categorize the recent work along the last



(a) Data transformation (b) Model transformation (c) Clustering

Figure 3.3: This figure shows the differences between the three primary methods of adaptation. One can transform the data or the model, or one can select a system based on a speaker (or other) clustering.

dimension, commenting on the other dimensions as appropriate. Since speaker categorization is not adaptation or normalization in the normal sense of *modifying* a system, it is a special case and will be treated first.

3.1.1 Speaker Categorization Approaches

Probably the simplest way to incorporate speaker-specific information into a speaker-independent system is merely to make several different systems, each trained on a group of similar speakers. During recognition an utterance by a speaker would be used to identify the cluster expected to best model that speaker's speech characteristics. Then, the system corresponding to that cluster would be used to decode that and further speech by that speaker. One common clustering based loosely on vocal tract length is a gender-based clustering, such as that done by Abrash, *et al.*[1] Most recent approaches attempt to share speaker-independent and -dependent data within their models. Ljolje [40] experimented with adjusting mixture weights of the Gaussian mixture phone models in his HMM but found little improvement. A popular approach that is fast and effective is hierarchical speaker clustering.[31] The primary goal of this approach is to rapidly select a model that has been trained on speakers similar to that of the new speaker. The system is initially constructed by creating an HMM model for each of several reference speakers. A model is also created for the cluster of all speakers together. Then, a distance measure is used to split, recursively, each cluster into sub-clusters until the individual models are reached.

During recognition/adaptation, at every node of this tree the most likely node (using a maximum likelihood criterion) is selected using the input speech, starting at the top and stopping at one of the leaves. The HMM associated with the most likely node in the path is then used for recognition. Thus, the HMM with the best combination of general and speaker-specific traits is selected with only order $log\ n$ model evaluations, where n is the number of reference speakers.

It is also possible to include speaker-identifying information as merely another input parameter to be modelled. Konig and Morgan [30] have experimented with using neural networks to provide this information. For each of n clusters, they trained a network to estimate the probability $P(cluster|Data)$ using LPC cepstral values and estimates of the speaker's fundamental frequency. They then added n binary input units to their phonetic probability estimator $(P(Phone|cluster, Data))$. Features generated from an individual's speech were evaluated once by each of the cluster networks and then n times by the recognizer network to produce outputs that could be used to estimate $P(Data|Phone, cluster)$ for each cluster. Unfortunately, there was very little performance improvement on the Resource Management task [46] used for evaluation. Witbrock and Haffner [59], instead of explicitly clustering speakers, used a neural network to compute a two-dimensional *Speaker Voice Code* representing the speaker's location in "speaker space". This vector was made available as additional inputs to an MS-TDNN recognizer and resulted in a substantial drop in the error rate on a telephone digit recognition task. There are two convenient features of this SVC network approach. First, computation of the SVC for a new speaker can be performed even if some phone categories are not represented in the available adaptation speech. Second, the value of this code stabilizes after only a few words of adaptation speech have been presented to the network. The combination of these two features allows for robust adaptation with only a small amount of (arbitrary) adaptation speech.

3.1.2 Data/Feature Transformation Approaches

Feature transformation approaches can be either direct or indirect. In a direct approach, a direct mapping is computed between the features computed from the speech of a new speaker and the features computed from the corresponding speech of a target speaker. This mapping may then be used either to convert speech examples of the target speaker for use in training a speaker-dependent recognizer for the new speaker or to convert the new speaker's speech to that of the target speaker, for whom a trained recognizer already exists. Probably the oldest technique, *spectral mapping* is still one of the most popular. Early work focused on primarily linear methods of creating the mapping between reference and new speakers (see [17]), although more recent work has focused either on piecewise linear mappings [3, 41] or explicitly nonlinear mappings such as that provided by neural networks.[24, 16, 29] One interesting aspect of the work by Fukuzawa, *et al.* [16] is that the neural network is segment-based. For either a frame-based or a segment-based system, it can be tricky to optimally align speech (or feature vectors) between new and reference speakers, since a different alignment produces a different mapping, which produces a different alignment, etc. With this in mind, Gong, *et al.* [20] have developed an iterative process to minimize the phonetic alignment error during the training of a spectral-transform neural net. With similar goals in mind, Knohl and Rinscheid [28] claim that self-organizing feature maps can be used to simultaneously deal with the alignment problem and reduce the amount of adaptation data that would normally be required to train a neural network to map from one speaker's spectral space to another. Various other models designed explicitly with speaker adaptation in mind have been described in the literature.[53]

In an indirect approach, the transformation between speakers either occurs earlier in the computation of the features or is accomplished by adjusting the value of speaker-specific (possibly speaker-identifying) parameters. Some forms of speaker clustering, especially those using neural networks, could also be classified as indirect feature transformation approaches (*cf.* [30, 59] in section 3.1.1). Another example of the indirect approach can be found in the recent work on frequency axis warping for speaker normalization. Since this work is closely related to our own, it is presented in Section 6.2.1 after the introduction of our method.

3.1.3 Model Transformation Approaches

HMM

Model transformation approaches, both for HMM- and neural network-based systems, have enjoyed recent popularity, due in large part to a 1991 paper by Lee, Lin, and Juang [34] presenting a Bayesian procedure for adapting continuous-density hidden Markov model parameters to the speech of a new speaker. This widely-cited paper demonstrates how to adjust model means in a way that optimally integrates new speech data with the existing speaker-independent models. Note that although adaptation is gradual and hence slow, this method will asymptotically converge to a speaker-dependent system if given enough data. Also, the theoretical development of this technique requires that one have adaptation examples for each speech unit in order to adapt all of the models, which may be practically impossible for certain adaptation tasks. A more general theoretical expansion of this work can be found in [19]. Recently a number of researchers have focused on the problem of lacking or insufficient examples for all models. Cox [11], for example, models each sound class as a linear regression on a set of different sound classes. Ohkura, *et al.* have introduced *Vector Field Smoothing* [45], a popular technique in Japan, to accomplish the same task. VFS not only estimates mappings for which there are no data; it also includes a smoothing step to adjust all mappings based on the surrounding (in model space) mappings, relying more heavily on mappings for which more examples exist. Kosaka, *et al.* [32] have combined this method with two simpler methods to improve performance when given small amounts of adaptation data. Leggetter and Woodland [36, 37] have also addressed the no-examples problem, as have Digalakis, *et al.* [13] and Zavaliagkos, *et al.*[61, 62] Earlier work in this area has been done by Stern and Lasry.[55] Having experimented with both supervised and unsupervised MAP adjustment of the means of feature vectors in CMU's FEATURE [10] system, they claim to be the first researchers to have "include[d] statistical correlations across decision classes in a speaker adaptation procedure explicitly."

There has recently been substantial research on ways to combine various adaptation methods. After pointing out that most statistical approaches can be categorized into either Maximum Likelihood feature transformation approaches or Bayesian adaptation approaches, where the former require little adaptation data but cannot adequately use more and the latter require more data but asymptotically approach speaker-dependent performance with increases in the data set size, Digalakis and Neumeyer [12, 44] present a method of combining the two families to create an adaptation approach with both short-term and asymptotically good performance. Earlier work by Zhao [63, 64] also focused on a similar combination of methods.

For a more detailed discussion of this topic, George Zavaliagkos' doctoral dissertation [60] contains an excellent review of recent HMM speaker adaptation techniques.

Neural Network

The model-based adaptation/retraining methods for neural network-based systems can be broken down into data/model transformations, new architectures, and new techniques.

Data/model transformation Speech recognition researchers have examined connectionist feature-space transformations for neural network-based systems. Frequently, this approach allows the feature-space transform to be directly included into the architecture, training, and/or use of the existing network model. Two such methods can be found in the European ESPRIT project [21, pp. 38–46] and work by Watrous.[57] In the ESPRIT project, researchers examined adding a speaker-dependent linear transformation layer to either the input or the output of the network. Although the linear transformation might not perform as well as a nonlinear transformation, the former can be directly integrated into the existing nonlinear weights, thereby eliminating the need for any extra computation when using the new speaker-dependent system. On the Resource Management (RM) Task [46], the linear output network reduced the word error rate on the Speaker Dependent (SD) speakers by 6% using 100 adaptation sentences. The linear input network reduced the word error rate by about 22%.

Watrous, on the other hand, used a set of second-order units to transform the inputs to his feedforward network of first-order units. A different transformation layer was created for each new speaker. On a ten-vowel classification task with 76 speakers, creating the speaker-dependent layer reduced the vowel error rate from 11.7% to 4.7%. Jointly optimizing the transformation layer and the rest of the network resulted in an error rate of 2.5%.

New architecture Also interesting is the work describing different network architectures that are more suited to adaptation or retraining. A notable example in the speech recognition literature is Hampshire and Waibel's META-PI architecture.[25, 26] The system consists of a limited number of speaker-dependent subnetworks whose outputs are combined through other networks to produce a speaker-dependent recognizer for *any* speaker. Although they do not explicitly perform adaptation, they claim that this architecture enables them to obtain approximately speaker-dependent recognition rates on a voiced-stop discrimination task using only speech from other speakers.

New technique Rather than changing the feature space or the architecture, one can instead use a new method of training or retraining. There is much work in this area within the neural network community. Chen developed a network weight optimization algorithm intended to rapidly locate global minima.[8] Such an algorithm would allow for continuous on-line training in a way that eliminates the chance of being trapped into a local minimum during continued training. While not specifically designed for adaptation, this approach suggests the possibility of a mathematical framework that supports retention of learned information when presented with new training examples.

The problem of learning new patterns without losing the old has been directly addressed in the neural network community under the term "catastrophic forgetting". A recent paper by Robins [50] discussed this problem in detail and presented several solutions. Pedreira and Roehl [47] also discussed this problem. Their solution was the development of a training algorithm that allowed a network designer to specifically control the tradeoff "between fitting new incoming data and causing minimum damage to the information related to the original data set."

3.2 This Thesis in Context

Of the above methods, the most popular and most successful by far are the model-based methods for HMM-based systems. Faced with adapting a neural-network system, however, it is not immediately obvious how to translate these methods for use in such a system. Most of the HMM methods perform adjustments to the parameters, e.g. means and/or variances, of the probability density functions of the subword models. Without changing the architecture of our network, we do not have the option of changing such parameters. Unlike the "catastrophic forgetting" research [50], we are not attempting to learn totally new information. Rather, we are attempting to slightly alter some of the information that has been learned without affecting other patterns. Thus, we feel that rapid speaker adaptation using only one or a few words of speech is still a substantial challenge for a neural network-based system.

In this thesis, we present two methods for adapting an already-trained neural network recognizer to the speech of a new speaker. The first of these methods, rapid retraining of the hidden-to-output unit weights, is a model-based method that actually produces a new network tuned to the speech it has been given. The second method, a parametric optimization approach, is an indirect feature space-based method that is not necessarily specific to neural network systems but is nevertheless appropriate because it is rapid and relies only on the existence of a recognizer score for an utterance. This approach requires only the tuning of a single parameter based on the final output score of the system and does not result in a "new" system.

Chapter 4

Datasets

This chapter gives brief descriptions of the datasets used in our experiments. Only the corpora themselves (along with default divisions into training, development, and test sets) are described here; descriptions of how they were used or further subdivided for our experiments are described in the experiment sections themselves (*cf.* Sections 6.4 and 5.3). Wherever possible, we have included references to more complete descriptions of the datasets.

4.1 The *OGI 30000 Numbers* corpus

Release 1.0 of the **OGI 30000 Numbers** corpus [9] is a telephone speech database consisting of 15000 number-string utterances. The utterances were taken from birthdates, phone numbers, zip codes, and street addresses collected in various OGI CSLU data collections. All calls have been transcribed at the word level (without time alignment) according to the conventions in [33]. From this corpus we selected all speakers for whom we had both an address and zipcode that consisted only of (possibly) connected digits, i.e. we only allowed sequences containing the words "oh", "zero", "one", "two", "three", "four", "five", "six", "seven", "eight", and "nine". We then divided these speakers into training, development, and test speakers based on call number as follows

$$\text{dataset} = \begin{cases} \text{training} & \text{if call number mod } 5 = 0,1, \text{ or } 2 \\ \text{development} & \text{if call number mod } 5 = 3 \\ \text{test} & \text{otherwise} \end{cases}$$

This resulted in 1147 training speakers, 406 development speakers, and 376 test speakers.

4.2 The *TIDIGITS* corpus

The **TIDIGITS** corpus [38] consists of studio-quality connected-digit utterances read by men, women, boys, and girls. Each speaker (adult or child) read 22 single-digit utterances and 11 each of 2-, 3-, 4-, 5-, and 7-digit utterances, for a total of 77 utterances per speaker. To allow us to use our existing telephone speech recognizers, we downsampled the **TIDIGITS** data from the original 20kHz sampling rate by upsampling to 40kHz, low-pass filtering with a 127-point Finite Impulse Response filter, and finally decimating to the desired 8kHz rate. The numbers of speakers in the training and test sets are shown in Table 4.1.

Table 4.1: Number of speakers in the **TIDIGITS** dataset.

	Number of speakers			
Dataset	Man	Woman	Girl	Boy
Train	55	57	26	25
Test	56	57	25	25

4.3 The *PhoneBook* corpus

The **PhoneBook** corpus [48] consists of both read speech and spontaneous speech utterances spoken over the telephone. Our experiments used only the read speech portion. In this portion, each speaker was given a list of either 75 or 76 isolated words to read. There are a total of 1358 speakers and 106 lists, so a word list was used by several speakers. In all, the word lists contained 7979 distinct words. Only 5 of these words were present on more than two lists; all others were present on only one list.

Each word list was labeled with a two-character alphabetic identifier, so we used this to partition the corpus into train, development, and final test sets based on the second letter of the identifier as follows

$$\text{dataset} = \begin{cases} \text{training} & \text{if } \textit{letter} \text{ is an odd-numbered letter (a, c, e, \ldots)} \\ \text{development} & \text{if } \textit{letter} \text{ is an even-numbered letter (b, d, f, \ldots)} \\ & \text{and is in the range a–l} \\ \text{test} & \text{otherwise} \end{cases}$$

This resulted in 688 training, 316 development, and 354 test speakers.

4.4 The *Huh?* corpus

There are currently many small-vocabulary tasks for which speech recognizers are being used. An example of such a task is a voice-only interface to the World Wide Web on the Internet. Certain words occur very frequently and must be recognized with high accuracy, even though they may be confusable. To test adaptation under such conditions, we have collected a database of common, highly confusable words.

We expected baseline recognition performance on this system to be poor since the words were chosen to be acoustically similar.

4.4.1 Collection

We collected the database over the telephone using digital T1 lines. Callers registered by filling out a form on our data collection Web page (shown in Figure 4.1) which asked for the following information: email address, native/non-native, and optionally their mailing address, age range, and gender. Each caller was then sent, by email, an instruction sheet containing the phone number to call and the list of words the person was to say when prompted. Although the order was different (selected at random) for each caller, the caller was expected to say the same set of words. The words and their frequencies are given in Table 4.2.

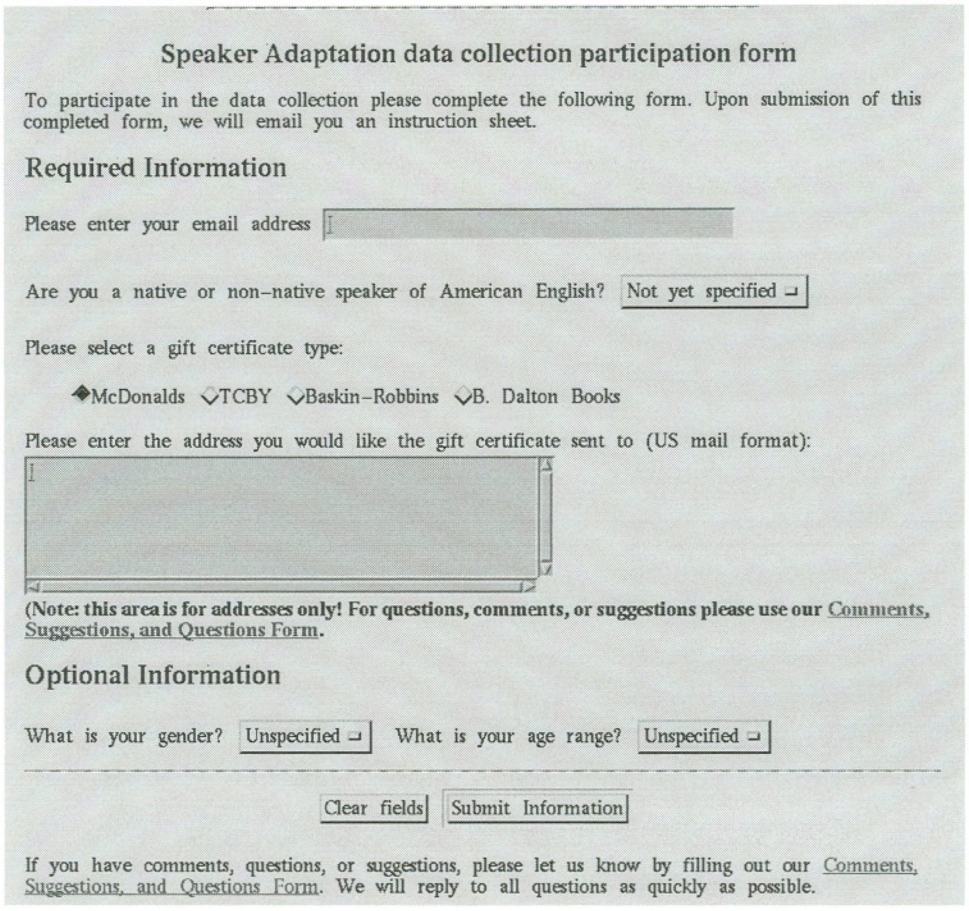
Thus each caller was supposed to say 125 words. To avoid monotony, the list of words was broken up into three sections. Section A contained the first 50 words, Section B contained the next 40 words, and Section C contained the final 35 words. A sample call list is shown in Appendix A.

Native speakers were given one number to call and non-native speakers another.

The collection protocol (list of prompts) is also given in Appendix A.

4.4.2 Transcription

The calls have all been labelled at the word level using the conventions in [33].



Speaker Adaptation data collection participation form

To participate in the data collection please complete the following form. Upon submission of this completed form, we will email you an instruction sheet.

Required Information

Please enter your email address

Are you a native or non-native speaker of American English? Not yet specified

Please select a gift certificate type:

[✓ McDonalds] TCBY Baskin-Robbins B. Dalton Books

Please enter the address you would like the gift certificate sent to (US mail format):

(Note: this area is for addresses only! For questions, comments, or suggestions please use our Comments, Suggestions, and Questions Form.

Optional Information

What is your gender? Unspecified What is your age range? Unspecified

If you have comments, questions, or suggestions, please let us know by filling out our Comments, Suggestions, and Questions Form. We will reply to all questions as quickly as possible.

Figure 4.1: WWW registration form for **Huh?** data collection. The form asked for the following information: email address, native/non-native speaker of American English, and optionally mailing address, age range, and gender.

Table 4.2: Words collected in **Huh?** corpus. Also shown next to each word is the number of times each caller was asked to say the word.

word	*frequency*
east	10
help	15
hope	10
last	10
less	10
low	10
nest	10
next	10
no	15
yes	15
yo	10

Table 4.3: Statistics—**Huh?** corpus.

Category		Number of Speakers	Percentage of Speakers
Development and test set callers		160	100
Gave at least one example of each word		149	93.1
Gave all desired examples of each word		49	37.5
Callers with recognizable ID's		120	75.0[1]
Native speakers of American English		117	97.5[2]
Males		56	50.9[3]
Age[4]	< 10	1	0.8
	10–15	6	5.0
	16–20	19	16.0
	21–30	43	36.1
	31–40	29	24.4
	41–50	8	6.7
	51–60	4	3.4
	61–70	2	1.7

4.4.3 Corpus statistics

We collected speech from 160 callers. Table 4.3 shows some statistics about the calls received. It is interesting to note that only about one-third of the callers actually provided exactly the number of examples of each word that were requested.

We divided the callers into three sets:

Test These were the callers who gave us exactly the requested numbers of examples of each word. There were 49 in this set.

Development These were the callers who gave us at least one example of each of the words. Test set callers were not included in this group, which had a total of 100 callers.

Remaining These were the callers who at least missed giving us even one example for all of the words. There were 11 callers in this group.

[1]Out of all 160 callers.

[2]Out of all 120 callers with recognizable ID's.

[3]Out of all 110 callers who volunteered gender information.

[4]Percentages are out of all 119 callers who volunteered age information.

4.4.4 Perceptual Study

Our stated goal in creating the **Huh?** database was to have a database consisting of a small set of confusable words. These are common words chosen for their similarity to one another. Our claim is that these words are difficult for our general-purpose recognizer to distinguish yet easy for humans to distinguish. In this section we present the results of a small-scale perceptual study which may help indicate the range of human performance on this task.

Data The utterances evaluated by the test subjects were randomly selected from the evaluation utterances of the test set. The 725 utterances, approximately one-fifth of the available utterances, were chosen to provide the same word frequencies present in the overall data set. Thus, there were 58 examples each of the words east, hope, last, less, low, nest, next, and yo and 87 examples of the words help, yes, and no.

The study Ten subjects participated in the study. All subjects were native speakers of American English. Each subject heard all 725 utterances in a random order, one at a time, and was asked to decide which of the eleven allowed words the utterance contained. The graphical user interface, shown in Figure 4.2, allowed the subject to play each word as many times as desired before deciding which word to select. The word performances for each speaker are presented in Table 4.4.

Performance on hope and last was noticeably poor, while performance on east and yes was noticeably good.

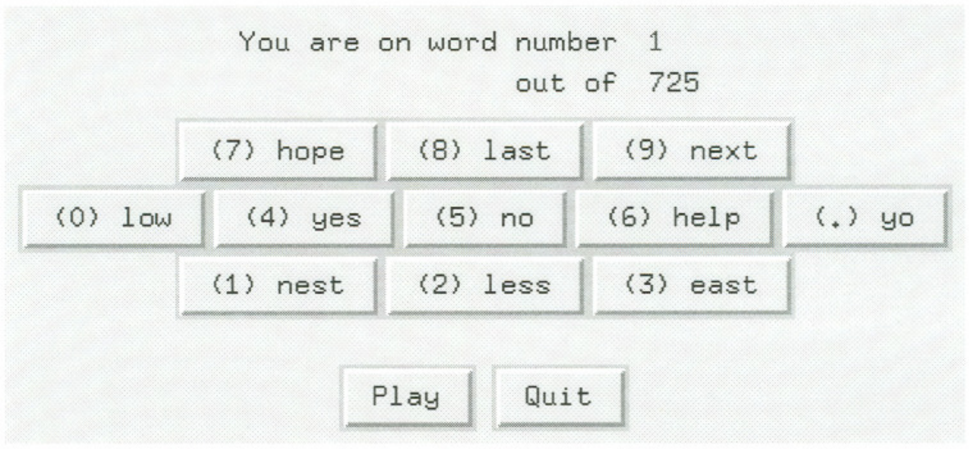


Figure 4.2: User interface for **Huh?** corpus perceptual study. Subjects listened to 725 utterances, one at a time, and clicked on the word contained in the utterance. Subjects could also play each utterance again as often as desired or stop and continue later.

Table 4.4: **Huh?** corpus perceptual study subject performance.

Adaptation Word	Percent correct for subject 1	2	3	4	5	6	7	8	9	10	% correct overall
east	100.0	100.0	100.0	100.0	100.0	100.0	100.0	98.3	100.0	94.8	99.3
help	95.4	95.4	94.3	93.1	94.3	93.1	98.9	94.3	97.7	86.2	94.3
hope	98.3	91.4	89.7	96.6	89.7	98.3	91.4	87.9	87.9	86.2	91.7
last	96.6	87.9	96.6	91.4	81.0	89.7	93.1	91.4	94.8	70.7	89.3
less	93.1	96.6	93.1	94.8	87.9	94.8	94.8	87.9	94.8	89.7	92.8
low	100.0	93.1	91.4	96.6	93.1	94.8	100.0	82.8	98.3	82.8	93.3
nest	98.3	93.1	94.8	91.4	82.8	94.8	96.6	96.6	100.0	82.8	93.1
next	96.6	98.3	98.3	98.3	96.6	98.3	98.3	98.3	98.3	98.3	97.9
no	100.0	100.0	100.0	100.0	98.9	98.9	100.0	100.0	100.0	93.1	99.1
yes	100.0	100.0	98.9	96.6	100.0	100.0	100.0	96.6	100.0	93.1	98.5
yo	98.3	98.3	98.3	98.3	91.4	96.6	98.3	96.6	100.0	93.1	96.9
OVERALL	97.9	96.1	96.1	96.1	93.0	96.4	97.7	94.1	97.7	88.6	95.4

Chapter 5

Rapid Retraining of the Neural Network

As described in Section 3.1.3, prior work on model-based adaptation for neural networks has consisted primarily of ways to add an extra processing layer to the inputs or the outputs or to add additional hidden units. The thesis work presented in this chapter focuses instead on ways to directly change the neural network's weights using only a small amount of speech from a new speaker.

5.1 Description

We are specifically interested in rapid adaptation using only small amounts of adaptation speech, preferably as little as one word. In particular, we have the following goal: in a small-vocabulary task, our method shall improve the performance of a single word for a given speaker

- using only one example of that word to adapt
- rapidly
- without substantially degrading performance on the other words in the vocabulary

This will be accomplished by adjusting the weights of the neural network. The adaptation procedure begins with a trained general-purpose, speaker-independent neural network and trains for a few iterations on a small amount of speaker-dependent data.

There are some difficulties in using only a single word to adapt an entire network. A single word will only consist of a small number of phonemes and hence be represented in the system (more precisely, the neural network probability estimator) by only a few context-dependent phonemic output categories. The adaptation vectors that can be generated from this word will only provide positive training examples for a small number of network output units. Unfortunately, our normal training paradigm for neural networks requires pattern vectors for all of the outputs in order to adjust all of the weights. Because the system is only trying to adapt to this one word, though, it is reasonable to consider only adapting the weights to the outputs for which speaker-dependent training examples have been computed (the *target* outputs).

An important issue must be addressed for this to succeed. It is critical that the network does not unlearn most of the patterns it has learned; in particular, it is unacceptable for the network to forget how to recognize the other words in the vocabulary. Due to the structure of the network, however, every output unit's value depends on the values of *all* of the hidden units. Adapting all of the weights leading to the target outputs would require changing the weights from the input to the hidden units as well. This would change output values for all of the units! Our solution to this problem is to change only the weights from the hidden layer to the outputs. This provides two benefits: first, it is possible to change weights only for the target outputs without affecting the other outputs, and second, the adaptation will be faster (fewer weights to change) and ideally more robust (fewer parameters to adjust based on the input data).

In the next two sections we describe the adaptation task we have chosen and our experiments on this task.

5.2 Confusable-word task

For our experiments we collected the **Huh?** database, a small-vocabulary database of confusable words. The design, development, collection, and labelling of this database are presented in Section 4.4.

Development and test sets The experiments used the 100 development set and 49 test set speakers from the **Huh?** corpus. To guarantee that the evaluation utterances were separate from the adaptation utterances, each speaker's files (utterances) were divided

Table 5.1: This table shows the division of a speaker's utterances into adaptation and evaluation sets.

	Number of utterances		
Word	Adapt	Eval	Total
east	4	6	10
help	6	9	15
hope	4	6	10
last	4	6	10
less	4	6	10
low	4	6	10
nest	4	6	10
next	4	6	10
no	6	9	15
yes	6	9	15
yo	4	6	10
ALL	50	75	125

into adaptation and evaluation portions for both the development and test sets. For each caller, each word ("less", "east", etc.) was assigned a unique index which counted, in order, the number of examples of that word spoken by the caller. Thus, each index ideally ranged between 0 and 9 or 0 and 14, depending on how many examples there were of each word. Any utterance whose word index mod 5 was

- 0, 2, or 4 was in the evaluation set
- 1 or 3 was in the adaptation set

Each speaker said up to 125 words, so this scheme resulted in up to 50 words in the adaptation set and up to 75 words in the evaluation set. Word counts are shown in Table 5.1.

The approximate number of words in the adaptation and evaluation subsets of the development and test sets are given in Table 5.2. The totals are approximate because some of the speakers did not give the requested number of examples of each word (see Table 4.3).

Table 5.2: This table shows the approximate sizes of the development adaptation, development evaluation, test adaptation, and test evaluation sets. Sizes are approximate because not all callers gave all the requested examples of each of the words.

		Approximate number of words		
Dataset	Speakers	Adapt	Eval	Total
Development	100	$100 * 50 = 5000$	$100 * 75 = 7500$	$100 * 125 = 12500$
Test	49	$49 * 50 = 2450$	$49 * 75 = 3675$	$49 * 125 = 6125$

5.3 Experiments

This section describes the experiments performed on the **Huh?** corpus. The experiments fall into three groups corresponding to the three subgoals of our overall goal as described in the previous section. The primary goal of the first experiment was to demonstrate that it *was* possible to improve performance on a target word through retraining. The primary goal of the second experiment was to reduce the computational load (and therefore time) of the adaptation without adversely affecting performance. The primary goal of the third experiment was to reduce any adverse effect on non-target words that might easily be confused with the target word. Although each experiment had a primary goal, it was important to keep the overall goal in mind. For example, at times it was necessary to sacrifice performance for speed.

In running the experiments, it rapidly became apparent that there were a number of interdependent parameters that had to be set reasonably to obtain acceptable performance. Keeping in mind the goals of speed and performance, the various parameters were quickly adjusted to move the system into the right performance region. The decisions on which parameters to adjust in any given experiment were made heuristically. The parameters varied were

Number of speaker-independent vectors It was important to establish the amount of original speaker-independent data to use. Some early experiments to determine the best way to train indicated that giving only examples from the new speaker to the trained net and retraining all of the outputs rapidly destroyed the performance of the net on all other words. Including some of the original speaker-independent training data would tend to reduce this effect. In addition, this general-purpose data may be used to recreate the same ratio (for each output) of positive to negative training examples present during the original training of the network, and more importantly, can provide the needed variety in negative training examples. We therefore explored adding various amounts of data selected randomly from the speaker-independent vectors originally used to train the net.

Number of speaker-dependent vectors In our experiments presented in this chapter vectors from the general-purpose speaker-independent set are replaced with speaker-dependent ones generated from the adaptation utterance. An important issue is finding a balance between the number of vectors (and the corresponding time needed to train) and the performance on the target word.

Number of training iterations It usually takes more than one training pass through the data for the network to learn the patterns. However, training for too many iterations can cause the network to overlearn the target patterns. In fact, this is a major concern in these experiments because it is unacceptable for the network to *unlearn* all of the input/output patterns already encoded—rather, they should merely be adjusted slightly. A common way to deal with this problem is to have a separate cross-validation set that is used to evaluate the performance of the network after each iteration. When the performance on the cross-validation set reaches a maximum and then degrades, training is stopped. In the situation presented earlier, though, no such dataset is available. Ideally, the single word of adaptation speech will be the only speech used to adapt, and in fact may be entirely used in the retraining of the network itself.

Learning rate This parameter controls the size of the step taken in the gradient descent algorithm. When far away from the eventual solution, a large learning rate can help speed up the rate at which the network nears the eventual solution. If close, though, a large value may actually delay or even prevent the algorithm from achieving the best solution by forcing the network weight changes to be larger than necessary, thereby overshooting the ideal weight values. Clearly the initial setting of this parameter can affect the rapidity of the retraining (adaptation).

It was also discovered early on that the performance on the word "help" was particularly poor. To ensure that this important case was addressed, many of the analyses described in this chapter initally focused on the performance of this word.

In Sections 5.3.1–5.3.3 the three development set experiments are described. Section 5.3.4 then presents results for the three experiments on the final test set. All experiments used a 3-layer feed-forward neural network with 56 input units, 200 hidden units, and 534 output units.

5.3.1 Experiment 1: Improving target word performance

An important issue to be addressed immediately was whether to adapt all of the output units or only the target output units. Given adaptation examples for, say, only 10 of the output units, it might be more appropriate to only update the weights to these output units rather than all of the weights. Although it was eventually decided that training all of the output units was computationally too costly given that the same performance was achievable by only training target outputs, both the all-output and target-only experiments are presented here.

Training all outputs

Speaker-independent data We first selected 200, 100, 50, and 25 examples per output category from the speaker-independent data. Then, for the worst-performing speaker in the dataset, we selected the first example of the word "help" available in the adaptation set, generated feature vectors, and substituted[1] them into the speaker-independent sets. Training on these different sets showed that at least 50 vectors/category were needed to keep from destroying performance on all other categories. Using more than 50 did not make much difference. Hence, we selected 50 vectors per category as the number of general-purpose examples to use.

[1]By *substituted* we mean that we replaced (at random) existing speaker-independent vectors in our set with speaker-dependent vectors, for each category. For example, if we started with 50 SI /ε/ vectors and 3 SD /ε/ vectors our final set would contain 47 SI and 3 SD /ε/vectors.

Speaker-dependent data After establishing the number of speaker-independent vectors needed to keep most of the outputs from changing, the next task was to determine the number of speaker-dependent vectors needed to affect the performance on a designated word. Since the word "help" was the most poorly recognized word overall, we focused on improving that word's performance. We first created a list of all the speakers who had no correct "help" recognitions. Of these we selected speaker #091, the speaker with the highest performance on all the other words, as our experimentation speaker. Not only would this point out any improvement in "help" performance; it would also point out any *decreases* in performance on the other 10 words.

We retained the 50 vector per category set of speaker-independent data from the previous experiment. We then generated feature vectors for all six of the "help" utterances for caller #091 in the adaptation set, duplicated this set of speaker-dependent vectors 10, 100, and 200 times, and substituted them into the speaker-independent set as before. Note that for the outputs representing the word "help" we would in some cases have notably more than 50 vectors per category. For all other outputs we would still only have the 50 speaker-independent vectors.

Baseline performance on the evaluation set was zero correct "help" recognitions and 94.2% correct non-help recognitions. Although all four vector sets improved performance on "help" to 100% correct, recognition performance on all other words went to 92.8%, 88.4%, 69.6%, and 91.3% for 1x, 10x, 100x, and 200x, respectively. The 200x result was a bit surprising, since the trend for the other vectors seemed to indicate that excessive numbers of target training vectors overbalanced the training enough to hurt performance on the other words. We concluded that a single copy of the vectors from all six adaptation examples was sufficient for adaptation.

Training targets only

In the previous experiments all of the output units were trained, even though there were only adaptation vectors for the outputs used in the word "help". In all following experiments only weights from the hidden units to those output units that would be expected to change were adjusted. For the word "help", this meant that only 9 of the 534 outputs would have their weights adjusted.

Experiments on the word "help" In this preliminary experiment vectors were still generated from all six of the adaptation examples of the word "help" from speaker #091. After training for 7 iterations, the baseline performance (same as before) improved to perfect "help" recognition and 94.2% non-help recognition.

Since the eventual goal was to adapt using only one utterance, we performed the same experiment using ten copies of the data from a single adaptation utterance. Post-retraining performance was perfect "help" recognition with no decrease in non-help recognition. For this speaker, 10 training iterations were required.

Experiments on all target words (Development set) We then tested on all the speakers in the development set with the parameters established so far:

- up to 50 speaker-dependent vectors per category obtained by generating feature vectors from one adaptation word and duplicating them 10 times, with any lacking (up to 50 total) vectors per category filled in with speaker-independent data
- training target outputs only
- training for 10 iterations

We did this for each of the 11 adaptation words. The results, given in Table 5.3, indicate that the performance on the target word always improved, in some cases dramatically, with only a slight degradation in performance on the non-target words.

5.3.2 Experiment 2: Reducing training time

The goal of the second experiment group was to reduce the computational load, and hence time, of the retraining. An obvious candidate for reduction was the number of speaker-independent vectors. Remember that the 50 vectors per category (for all 534 outputs!) of speaker-independent data used in the earlier experiments was established when still training all of the outputs. In this experiment group we develop a way to reduce the

Table 5.3: This table shows target and non-target recognition rates for all speakers in the development set. One example of the target word was used to generate training vectors. These vectors were duplicated 10 times to provide up to 50 speaker-dependent vectors for each category present in the training word, which were substituted into 50 speaker-independent vectors/category. Training (target outputs only) was stopped after 10 iterations. This is the development set result for Experiment 1.

	Percent correct			
Adaptation	Target		Non-target	
Word	Baseline	**Exp1**	Baseline	**Exp1**
east	96.2	99.3	69.7	70.6
help	35.4	95.1	76.7	75.3
hope	62.6	96.1	72.5	69.8
last	77.6	95.4	71.2	69.4
less	67.6	95.7	72.1	69.1
low	70.5	92.9	71.8	72.5
nest	61.0	95.9	72.6	69.2
next	73.2	94.2	71.6	71.1
no	71.6	96.8	71.7	68.6
yes	91.5	99.0	69.0	66.2
yo	91.5	99.7	70.0	66.8
OVERALL	71.7	96.4	71.7	69.9

number of training vectors without seriously reducing the target-word performance gain obtained in Experiment 1.

Varying the numbers of speaker-independent and -dependent vectors The first experiment again used caller #091. We generated feature vectors from the first example of the word "help" by the speaker in the adapt set and then duplicated this set of vectors enough times to obtain at least 50 vectors for each of the nine outputs represented in the word "help", producing an entirely speaker-dependent vector set. To form the training set, a given number of vectors per category were selected from the speaker-dependent set and the rest from the speaker-independent set. We experimented with an overall number of 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50 vectors for each of the 534 output categories. For each of these we tried 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, or 50 speaker-dependent vectors, up to the total number of vectors per category. In other words, we evaluated dependent/independent combinations of 0/5, 5/5, 0/10, 5/10, 10/10, and so on. The networks were trained for 5 iterations. As one might expect, the best performances

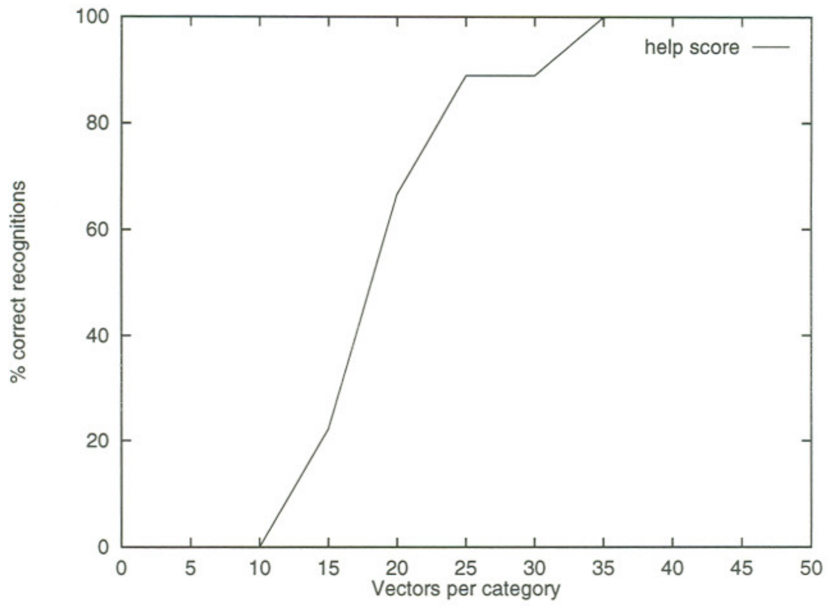


Figure 5.1: This figure shows recognition rates for the word "help" for speaker #091 as a function of the number of training vectors. One example of the word "help" was used to generate training vectors. These vectors were duplicated to provide 5, 10, ..., 45, 50 speaker-dependent vectors for each category present in the training word. An equivalent number of vectors/category (for missing categories only) of speaker-independent data were added to this set. Results are after training for 5 iterations.

occurred for the largest percentages of speaker-dependent data — the 5/5, 10/10, 15/15, and similar cases. Recognition rates for the word "help" in these cases are plotted in Figure 5.1. Performance on the non-help words remained near 100%.

Varying learning rate and number of vectors These recognition rates seem to indicate that a larger number of vectors is better. However, the speaker-dependent vectors present in all the cases are essentially identical—some datasets merely have more copies. Since the training algorithm performs weight updates for all of the vectors in the training set during each iteration, the network will definitely learn the desired mapping in fewer iterations with more training vectors, even if they are only copies. To reduce this effect, we examined adaptation performance when varying the *learning rate*, the multiplier used to determine how much of a weight's error should be corrected in any one weight change. We selected three callers from the list of callers who had no correct "help" recognitions.

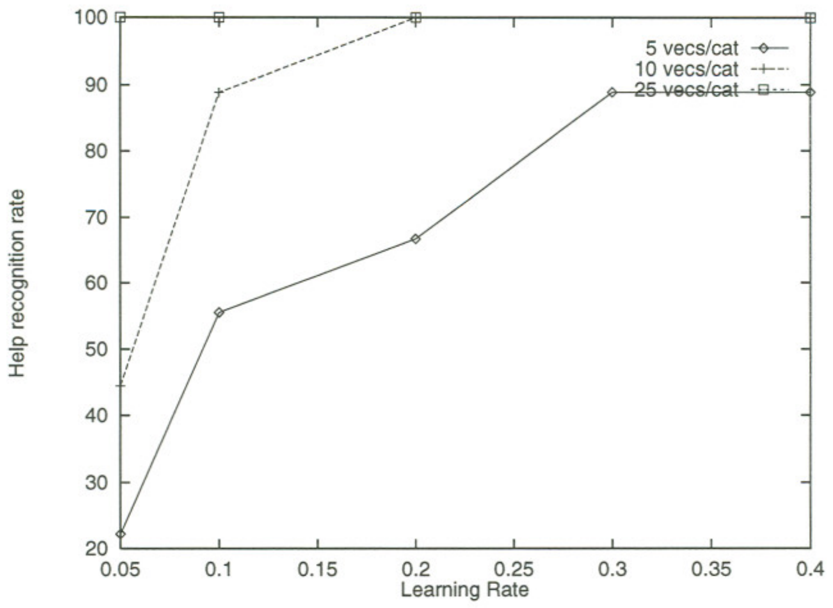


Figure 5.2: This figure shows recognition rates for the word "help" for speaker #091 as a function of both the learning rate and the overall number of training vectors per category. One example of the word "help" was used to generate training vectors. These vectors were duplicated to provide 5, 10, 15, 20, or 25 speaker-dependent vectors for each category present in the training word. An equivalent number of vectors/category (for missing categories only) of speaker-independent data were added to this set. Learning rates varied between 0.05 and 0.4. Results are after training for 6 iterations.

Specifically, we selected the best-performing (#091), the worst-performing (#077), and one from the middle (#128). For each of these callers, we simultaneously varied the learning rate from 0.05 to 0.4 and the overall number of vectors per category from 5 to 25. The results from training for six iterations are shown in Figures 5.2 through 5.4.

For all three speakers, quadrupling the learning rate resulted in a rapid increase in performance with only a small number of training iterations, suggesting that the earlier low performance was indeed due to the small weight adjustment steps. Note that for all three speakers only ten vectors per category were necessary to obtain this performance, much less than the 50 needed when training all of the outputs.

Varying learning rate and number of training iterations The results on the three speakers were promising, so we retained the 10 vectors per category setting and varied the learning rate and number of training iterations for all of the speakers in the development

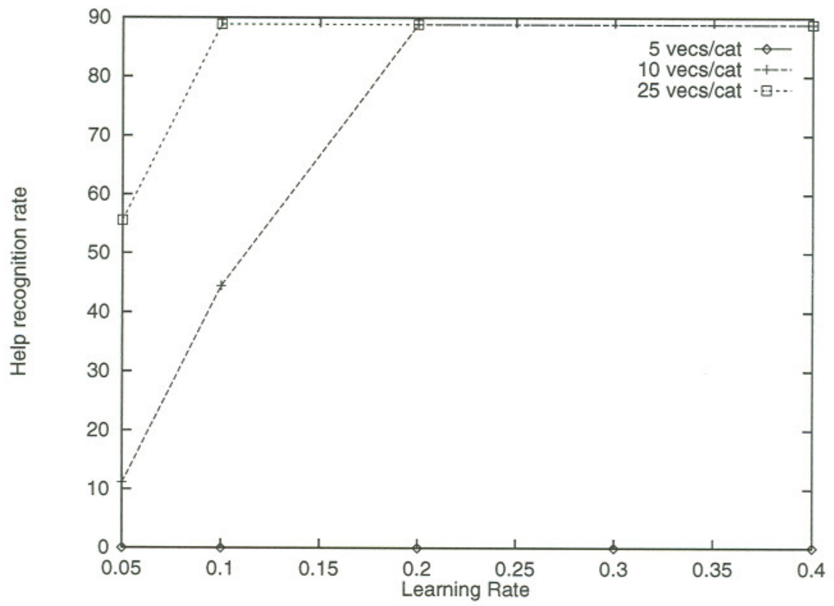


Figure 5.3: This figure shows recognition rates for the word "help" for speaker #077 as a function of both the learning rate and the overall number of training vectors per category. One example of the word "help" was used to generate training vectors. These vectors were duplicated to provide 5, 10, 15, 20, or 25 speaker-dependent vectors for each category present in the training word. An equivalent number of vectors/category (for missing categories only) of speaker-independent data were added to this set. Learning rates varied between 0.05 and 0.4. Results are after training for 6 iterations.

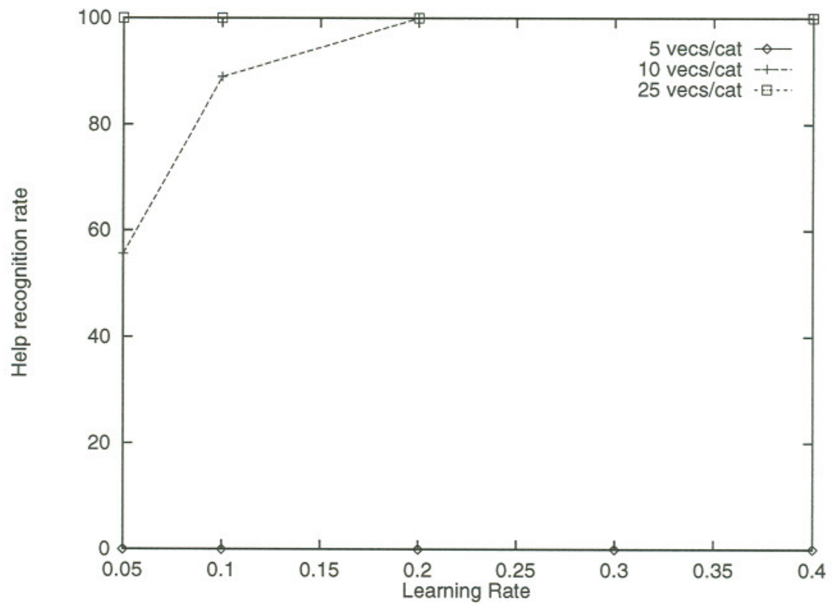


Figure 5.4: This figure shows recognition rates for the word "help" for speaker #128 as a function of both the learning rate and the overall number of training vectors per category. One example of the word "help" was used to generate training vectors. These vectors were duplicated to provide 5, 10, 15, 20, or 25 speaker-dependent vectors for each category present in the training word. An equivalent number of vectors/category (for missing categories only) of speaker-independent data were added to this set. Learning rates varied between 0.05 and 0.4. Results are after training for 6 iterations.

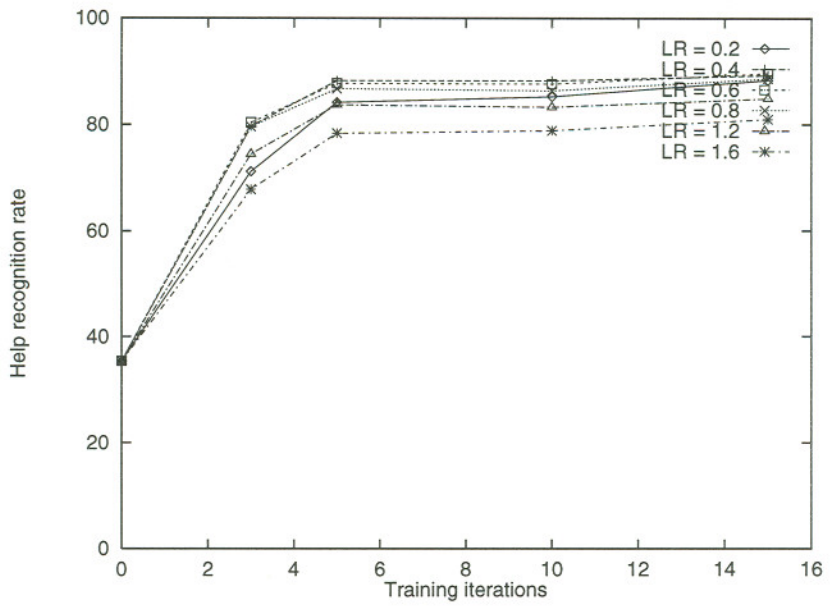


Figure 5.5: This figure shows recognition rates for the word "help" for the speakers in the development set as a function of both the learning rate and the number of training iterations. For each speaker, one example of the word "help" was used to generate training vectors. These vectors were duplicated to provide 10 speaker-dependent vectors for each category present in the training word. For the missing categories, 10 vectors/category of speaker-independent data were added to this set. Learning rates varied between 0.2 and 1.6. System was tested after training for 3, 5, 10, and 15 iterations.

set. We used learning rates of 0.2, 0.4, 0.6, 0.8, 1.2, and 1.6 and training iterations of 3, 5, 10, and 15. One example of the word "help" was used to generate features as described in the previous paragraphs. After training, the new system was tested on all of the evaluation utterances for the speaker. The results for each condition, averaged over all speakers, are shown in Figure 5.5. The results clearly indicated that 0.4 was the best learning rate for this subset of the data. Moreover, the majority of the performance improvement was obtained with only 5 training iterations. Although there was a slight improvement at 15 iterations, it did not seem worth the added computational cost.

Varying amount of speaker-dependent data A complete test was run on all target words for all speakers in the development set using 10 vectors per category, a learning rate of 0.4, and only 5 training iterations. Results are shown in Table 5.4. When comparing to the results from Experiment 1, also shown in the table, one can see that the

Table 5.4: This table shows target and non-target recognition rates for all speakers in the development set. One example of the target word was used to generate training vectors. These vectors were duplicated to provide 10 speaker-dependent vectors for each category present in the training word. For the missing categories, 10 vectors/category of speaker-independent data were added to this set. The learning rate was set at 0.4 and training was stopped after 5 iterations.

Adaptation	Percent correct					
	Target			Non-target		
Word	Baseline	Exp1	**10vec**	Baseline	Exp1	**10vec**
east	96.2	99.3	98.7	69.7	70.6	70.2
help	35.4	95.1	88.2	76.7	75.3	75.8
hope	62.6	96.1	93.4	72.5	69.8	69.8
last	77.6	95.4	89.0	71.2	69.4	71.2
less	67.6	95.7	93.1	72.1	69.1	71.1
low	70.5	92.9	90.9	71.8	72.5	72.5
nest	61.0	95.9	91.0	72.6	69.2	71.0
next	73.2	94.2	91.8	71.6	71.1	72.2
no	71.6	96.8	94.6	71.7	68.6	70.3
yes	91.5	99.0	98.3	69.0	66.2	67.4
yo	91.5	99.7	99.1	70.0	66.8	67.8
OVERALL	71.7	96.4	93.5	71.7	69.9	70.9

target word performance was not quite as good. To fix this, we once again changed the number of speaker-dependent vectors. We began, as in the previous experiment, by generating features from one example of the target word. However, this time we duplicated the resulting vectors enough times to obtain at least 50 examples of each of the target categories. We then substituted exactly 50 per category into our 10 vectors per category of speaker-independent data, producing 50 speaker-dependent vectors for each target category and 10 speaker-independent vectors for each of the remaining categories. The results of rerunning the experiment with this set of vectors on the entire development set, keeping the learning rate at 0.4 and training for only 5 iterations, are shown in Table 5.5. Performance this time was excellent. It was actually slightly better than that obtained in Experiment 1, which used 50 vectors for each category. Note that even though the vector total has increased, there is still a tremendous reduction in computational time compared to Experiment 1. In each training iteration the system only needs to process $10 * 50 + 524 * 10 = 5740$ vectors rather than $50 * 524 = 26200$ vectors, a reduction by

Table 5.5: This table shows target and non-target recognition rates for all speakers in the development set. One example of the target word was used to generate training vectors. These vectors were duplicated to provide 50 speaker-dependent vectors for each category present in the training word. For the missing categories, 10 vectors/category of speaker-independent data were added to this set. The learning rate was set at 0.4 and training was stopped after 5 iterations. This is the development set result for Experiment 2.

	Percent correct					
Adaptation	Target			Non-target		
Word	Baseline	Exp1	**Exp2**	Baseline	Exp1	**Exp2**
east	96.2	99.3	99.6	69.7	70.6	70.8
help	35.4	95.1	95.4	76.7	75.3	74.4
hope	62.6	96.1	98.3	72.5	69.8	67.8
last	77.6	95.4	96.7	71.2	69.4	68.0
less	67.6	95.7	97.7	72.1	69.1	66.0
low	70.5	92.9	93.1	71.8	72.5	72.9
nest	61.0	95.9	97.3	72.6	69.2	66.7
next	73.2	94.2	95.4	71.6	71.1	69.3
no	71.6	96.8	96.2	71.7	68.6	68.0
yes	91.5	99.0	99.3	69.0	66.2	64.7
yo	91.5	99.7	99.3	70.0	66.8	66.9
OVERALL	71.7	96.4	97.1	71.7	69.9	68.7

approximately a factor of five. In addition, only 5 training iterations are needed instead of 10, further halving the training time. We have accomplished the goal of Experiment 2: to reduce the computational load without losing a substantial amount of the target word gain obtained in Experiment 1.

5.3.3 Experiment 3: Incremental retraining

The performance improvements described so far suggest that the retraining approach as it currently stands is rather successful. However, remember that the overall goal was to improve performance on a target word *without* hurting performance on the other words. Although the non-target performance numbers for Experiment 2 (Table 5.5) show only a small degradation from the baseline, a closer look revealed a rather significant problem. Table 5.6 shows the performance changes on several target words and the corresponding changes in specific confusable non-target words. It appears that much of the performance drop is borne by specific words that are difficult to distinguish from the target words.

Table 5.6: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Results are given for the baseline system as well as Experiments 1 and 2.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Expt.	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
Baseline	77.6	67.6	67.6	77.6	61.0	91.5	71.6	91.5	91.5	61.0	91.5	71.6
Exp1	95.4	52.7	95.7	56.6	95.9	77.5	96.8	75.5	99.0	36.1	99.7	45.8
Exp2	96.7	39.6	97.7	40.5	97.3	64.0	96.2	65.7	99.3	27.1	99.3	39.7

Further examination revealed that this performance drop often occurred for a speaker whose performance on the target word was already high. In other words, the system was over-adapting to a target word whose performance, *for that speaker*, was already good.

The last experiment, then, focused on determining a way to adapt only when necessary. Rather than explictly attempting to estimate the recognition performance on words from a new speaker, a modification to the approach would be to retrain on a target word whenever an error occurs on that word. In addition, it should be possible to initially retrain using only a fraction of the 50 speaker-dependent vectors per category, increasing this fraction for each further error that occurs. The benefits to this approach are:

- Adaptation (retraining) only occurs if the word was missed. If the performance is good, there is no need to adapt and risk damaging the performance on other words.
- Retraining starts off cautiously and increases only as necessary. If only a small push is needed to nudge the network into better recognition performance, why hit it with a sledge hammer?

Keep in mind that there are only four or six adaptation examples (per speaker) of any target word in this corpus (*cf.* Table 5.1). The consequent effect is that all retraining decisions must be made based on the correctness of recognition on the first four (or six) utterances. Experiments exploring this modified approach are described in the next three paragraphs.

Throughout the remainder of this chapter we will be using the concept of a *training progression*. As an example, consider the progression 5-10-15-20. This means that five speaker-dependent vectors (per category) are used to retrain after the first error, 10 after the second, 15 after the third, and 20 after the fourth. In each case the speaker-dependent vectors are substituted into the same 10 vectors per category of speaker-independent data. After each retraining, the new network is used in the following classifications, so the chance of errors on the target word should decrease over the course of the progression. Note, however, that the speaker-dependent vectors are generated, for all retrainings, using the original unadapted network. As before, the speaker-dependent vectors are generated from the first example of the target word (whether or not it was correctly classified).

Each of the networks produced during the course of the progression is tested on all of the evaluation utterances for the speaker. This experimental setup was designed to mimic the effect of being constrained to allow no more than 4 target-word errors within the first six utterances, with the effectiveness of the new network demonstrated on all utterances thereafter.

Initial training progressions We first tried several different training progressions on fifty speakers from the development set. The goal here was to get a general impression for how many speaker-dependent vectors should be used both early and late in the progression. Progressions of 5-10-15-20, 10-20-30-40, 15-30-45, and 20-40-50 were tested, with each training using a learning rate of 0.4 and lasting for 5 iterations. The confusable-pair results for all four progressions are given in Tables 5.7–5.10. The results show that using more speaker-dependent data adversely affects the performance on the confusable non-target words. More importantly, using more data sooner hurts more. In other words, the system should initially use as little speaker-dependent data as possible and only add more as necessary.

Table 5.7: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Scores are averaged over 50 development set speakers. Results are given for an incremental retraining progression of 5-10-15-20 speaker-dependent vectors for the first, second, third, and fourth errors.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Number of vectors	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
0	78.0	65.7	65.7	78.0	55.9	93.4	70.7	93.3	93.4	55.9	93.3	70.7
5	91.3	58.2	92.8	66.8	91.8	77.2	96.6	83.9	97.9	53.8	98.2	65.5
10	92.4	58.8	94.4	66.1	92.8	77.9	96.4	83.5	97.7	54.1	98.2	65.5
15	92.4	58.8	94.4	66.1	92.5	77.9	97.1	81.8	97.7	54.1	98.2	65.5
20	92.4	58.8	94.4	66.1	92.5	77.9	97.1	81.8	97.7	54.1	98.2	65.5

Table 5.8: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Scores are averaged over 50 development set speakers. Results are given for an incremental retraining progression of 10-20-30-40 speaker-dependent vectors for the first, second, third, and fourth errors.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Number of vectors	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
0	78.0	65.7	65.7	78.0	55.9	93.4	70.7	93.3	93.4	55.9	93.3	70.7
10	92.8	58.8	95.1	59.2	92.8	76.9	96.6	81.8	98.4	54.5	98.2	63.9
20	93.1	58.5	94.8	59.2	93.5	76.5	97.1	80.7	98.2	54.5	98.2	63.9
30	93.1	58.2	94.8	59.2	93.5	76.5	97.3	80.7	98.4	54.1	98.2	63.9
40	93.1	58.2	94.8	59.2	93.5	76.5	97.3	80.7	98.4	54.1	98.2	63.9

Table 5.9: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Scores are averaged over 50 development set speakers. Results are given for an incremental retraining progression of 15-30-45 speaker-dependent vectors for the first, second, and third errors.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Number of vectors	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
0	78.0	65.7	65.7	78.0	55.9	93.4	70.7	93.3	93.4	55.9	93.3	70.7
15	92.4	57.2	95.1	53.4	93.2	74.9	96.1	77.5	98.2	54.8	98.2	64.6
30	92.8	56.5	94.8	53.8	94.6	74.7	96.8	77.2	98.2	54.5	98.2	64.6
45	92.8	56.9	95.1	53.8	94.6	74.7	96.8	78.2	98.4	54.8	98.2	64.6

Table 5.10: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Scores are averaged over 50 development set speakers. Results are given for an incremental retraining progression of 20-40-50 speaker-dependent vectors for the first, second, and third errors.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Number of vectors	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
0	78.0	65.7	65.7	78.0	55.9	93.4	70.7	93.3	93.4	55.9	93.3	70.7
20	93.1	57.2	95.1	59.2	94.6	71.9	97.1	79.3	97.9	52.7	98.2	63.7
40	94.2	56.5	95.1	58.5	95.0	72.1	97.3	80.4	97.9	52.7	98.2	63.3
50	94.2	56.5	95.4	58.5	95.0	72.1	97.3	80.4	98.4	52.0	98.2	63.3

Improved training progressions The next set of progressions used a small initial number of speaker-dependent vectors that rapidly increased. The goal here was to catch most of the errors with a slight amount of retraining, catch essentially all of the others with the second retraining, and use the last retraining merely as a mop-up for the stubborn remainder. Progressions of 1-5-25, 2-6-18, and 3-9-27 were tried using the established values of 10 speaker-independent vectors/category, a learning rate of 0.4, and 5 training iterations. These progressions were also tested using only 50 of the development set speakers. The results, shown in Tables 5.11–5.13, were slightly better. We concluded that starting with 2 or 3 vectors was approximately correct, but a more rapid increase thereafter would be better.

Table 5.11: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Scores are averaged over 50 development set speakers. Results are given for an incremental retraining progression of 1-5-25 speaker-dependent vectors for the first, second, and third errors.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Number of vectors	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
0	78.0	65.7	65.7	78.0	55.9	93.4	70.7	93.3	93.4	55.9	93.3	70.7
1	80.1	68.0	79.7	79.1	73.5	87.7	83.2	87.7	95.4	54.5	95.8	65.1
5	84.1	68.6	89.9	73.6	88.2	80.1	95.2	82.5	97.0	54.8	96.8	65.3
25	88.8	64.7	91.5	72.9	87.8	80.1	94.6	81.4	97.3	54.8	96.8	65.3

Table 5.12: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Scores are averaged over 50 development set speakers. Results are given for an incremental retraining progression of 2-6-18 speaker-dependent vectors for the first, second, and third errors.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Number of vectors	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
0	78.0	65.7	65.7	78.0	55.9	93.4	70.7	93.3	93.4	55.9	93.3	70.7
2	83.4	67.6	79.7	80.9	89.6	84.0	93.9	81.8	96.6	52.3	97.2	63.3
6	88.8	66.7	89.2	71.8	91.8	82.2	93.4	82.5	97.7	52.7	97.9	63.5
18	91.0	64.7	89.5	72.2	91.8	82.2	93.0	82.1	97.7	52.7	97.9	63.5

Table 5.13: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Scores are averaged over 50 development set speakers. Results are given for an incremental retraining progression of 3-9-27 speaker-dependent vectors for the first, second, and third errors.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Number of vectors	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
0	78.0	65.7	65.7	78.0	55.9	93.4	70.7	93.3	93.4	55.9	93.3	70.7
3	83.4	68.3	90.5	69.7	90.3	83.3	91.2	84.6	97.7	52.3	97.9	64.4
9	91.3	66.3	92.8	68.2	91.4	82.4	92.5	84.6	97.5	52.7	97.9	64.6
27	91.7	66.0	92.8	68.2	91.4	82.4	92.5	84.2	97.3	52.7	97.9	64.6

Full development set With the range of acceptable progressions narrowed down, we tried only two progressions on the entire development set. The two progressions were 2-10-20 and 3-12-24. All other parameters were the same. Results on the development set speakers are given in Tables 5.14 and 5.15. We accepted the progression 3-12-24 as the most desirable. Overall performance results for this progression were computed on the last retrained network and are shown in Table 5.16. With this experiment we accomplished the last of our 3 major goals: we maintained a good performance improvement on target words and a rapid adaptation while reducing the damage done to confusable non-target words.

Table 5.14: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Scores are averaged over all development set speakers. Results are given for an incremental retraining progression of 2-10-20 speaker-dependent vectors for the first, second, and third errors.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Number of vectors	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
0	77.6	67.6	67.6	77.6	61.0	91.5	71.6	91.5	91.5	61.0	91.5	71.6
2	84.2	68.9	78.6	79.5	89.0	85.0	93.6	83.5	95.7	57.8	97.4	65.5
10	91.4	67.4	91.1	70.6	91.7	83.8	94.4	82.6	97.0	58.0	97.7	65.5
20	91.6	67.1	91.3	70.6	92.1	83.8	94.2	82.6	97.1	58.0	97.7	65.5

Table 5.15: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Scores are averaged over all development set speakers. Results are given for an incremental retraining progression of 3-12-24 speaker-dependent vectors for the first, second, and third errors.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Number of vectors	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
0	77.6	67.6	67.6	77.6	61.0	91.5	71.6	91.5	91.5	61.0	91.5	71.6
3	82.3	69.6	90.5	70.4	91.2	84.9	90.9	86.1	96.8	56.9	97.7	66.2
12	91.6	67.6	93.0	68.7	92.6	83.9	93.7	84.7	96.8	57.3	97.7	66.2
24	91.2	67.4	93.0	68.7	92.8	83.9	94.0	85.2	96.9	57.3	97.7	66.2

Table 5.16: This table shows target and non-target recognition rates for all speakers in the development set. One example of the target word was used to generate training vectors. These vectors were duplicated to provide 3, 12, or 24 speaker-dependent vectors for each category present in the training word. Speaker-independent vectors were added to bring the total for each category to 10 (if not already there or higher). This is the development set result for Experiment 3.

	Percent correct							
Adaptation	Target				Non-target			
Word	Baseline	Exp1	Exp2	**Exp3**	Baseline	Exp1	Exp2	**Exp3**
east	96.2	99.3	99.6	98.4	69.7	70.6	70.8	69.7
help	35.4	95.1	95.4	97.2	76.7	75.3	74.4	70.8
hope	62.6	96.1	98.3	92.3	72.5	69.8	67.8	71.1
last	77.6	95.4	96.7	91.2	71.2	69.4	68.0	70.7
less	67.6	95.7	97.7	93.0	72.1	69.1	66.0	70.9
low	70.5	92.9	93.1	91.7	71.8	72.5	72.9	71.7
nest	61.0	95.9	97.3	92.8	72.6	69.2	66.7	71.5
next	73.2	94.2	95.4	91.4	71.6	71.1	69.3	71.6
no	71.6	96.8	96.2	94.0	71.7	68.6	68.0	70.4
yes	91.5	99.0	99.3	96.9	69.0	66.2	64.7	68.6
yo	91.5	99.7	99.3	97.7	70.0	66.8	66.9	69.0
OVERALL	71.7	96.4	97.1	94.4	71.7	69.9	68.7	70.6

5.3.4 Final Results

Having found suitable parameter values, the 3 Experiment approaches were evaluated on the final test set. Table 5.17 shows the target and non-target recognition performance when adapting (individually) to each of the eleven words in the dataset. Table 5.18 gives the performance of a target word and its most-confusable non-target counterpart for several different target words. The next 3 paragraphs give a brief summary of the parameter values used in each of the 3 Experiments.

Experiment 1: Improving target word performance Adaptation proceeded as follows. Features were generated from one example of the target word. These vectors were duplicated 10 times, and from this set were selected no more than 50 vectors per category. The training set was formed by adding in enough speaker-independent data to bring the total for each category up to 50 vectors. With the learning rate set at the default value of 0.05, the network was trained for 10 iterations (starting from the already-trained general purpose network). Retraining required approximately 300 seconds on our workstations.

Experiment 2: Reducing computation time From one example of the target word, feature vectors were generated as in Experiment 1, then duplicated enough times to obtain 50 of each category. The training set was formed by adding in 10 vectors per category of speaker-independent data for each of the remaining (non-target) categories. This network was trained with a learning rate of 0.4, but only for 5 iterations. Retraining required approximately 30 seconds on our workstations.

Experiment 3: Incremental retraining As in Experiment 2, feature vectors were generated and duplicated from one example of the target word to produce 50 speaker-dependent vectors for each category. Sets of size 3, 12, and 24 of these vectors (per category) were randomly selected to use in the retraining. To each of these were added enough speaker-independent vectors to ensure that every category had at least 10 vectors. We evaluated our recognizer on the first of the adaptation utterances. If an error occurred, we retrained and used the new network to evaluate the next utterance. Otherwise, we used the original network to evaluate the next utterance. We did this for all six of the adaptation utterances available for a given target word and speaker. We then evaluated each network on all the evaluation utterances for the speaker. The first retraining used the 3-vector set, the second the 12-vector set, and the third the 24-vector set. Each training was for 5 iterations with a learning rate of 0.4. The first retraining required approximately 35 seconds, the second 40 seconds, and the third 50 seconds.

Table 5.17: This table shows target and non-target recognition rates averaged over all speakers in the final test set. Results are shown using the parameter settings of Experiments 1–3. Experiment 1 used 50 speaker-dependent vectors/category substituted into 50 speaker-independent vectors/category, a learning rate of 0.05, and 10 training iterations. Experiment 2 used 50 speaker-dependent vectors/category substituted into 10 speaker-independent vectors/category, a learning rate of 0.4, and 5 training iterations. Experiment 3 used a progression of 3, 12, and 24 speaker-dependent vectors substituted into 10 speaker-independent vectors/category, a learning rate of 0.4, and 5 training iterations.

	Percent correct							
Adaptation	Target				Non-target			
Word	Baseline	Exp1	Exp2	**Exp3**	Baseline	Exp1	Exp2	**Exp3**
east	99.7	99.7	99.7	99.7	72.6	72.8	73.0	72.6
help	32.0	95.2	98.2	97.1	80.6	78.3	77.4	74.8
hope	71.4	99.3	100.0	94.6	75.1	71.8	69.8	73.9
last	78.8	94.9	98.6	93.5	74.8	72.8	71.3	74.9
less	67.3	98.6	99.3	95.9	75.4	72.2	69.6	74.0
low	77.9	94.9	93.2	93.5	74.5	75.5	75.0	75.5
nest	75.2	99.0	99.0	93.5	74.7	71.1	69.3	74.8
next	78.9	97.6	100.0	93.9	74.4	73.6	71.8	75.2
no	73.5	98.0	98.4	95.2	75.0	71.2	71.2	73.9
yes	93.4	99.5	99.8	98.9	72.2	69.5	67.9	71.9
yo	91.2	99.7	99.7	98.6	73.4	69.5	69.4	72.9
OVERALL	74.8	97.8	98.7	96.0	74.8	72.6	71.4	74.1

Table 5.18: For each of six target words, this table shows the target word performance (after adapting) and the performance of a non-target word substantially affected by the retraining. Scores are averaged over all test set speakers. Results are shown using the parameter settings of Experiments 1–3. Experiment 1 used 50 speaker-dependent vectors/category substituted into 50 speaker-independent vectors/category, a learning rate of 0.05, and 10 training iterations. Experiment 2 used 50 speaker-dependent vectors/category substituted into 10 speaker-independent vectors/category, a learning rate of 0.4, and 5 training iterations. Experiment 3 used a progression of 3, 12, and 24 speaker-dependent vectors substituted into 10 speaker-independent vectors/category, a learning rate of 0.4, and 5 training iterations.

	Percent correct											
	Adapted on last		Adapted on less		Adapted on nest		Adapted on no		Adapted on yes		Adapted on yo	
Expt.	Tested on last	Tested on less	Tested on less	Tested on last	Tested on nest	Tested on yes	Tested on no	Tested on yo	Tested on yes	Tested on nest	Tested on yo	Tested on no
Baseline	74.8	67.3	67.3	74.8	75.2	93.4	73.5	91.2	93.4	75.2	91.2	73.5
Exp1	94.9	51.4	98.6	52.7	99.0	77.1	98.0	72.1	99.5	50.7	99.7	45.6
Exp2	98.6	37.1	99.3	35.7	99.0	66.0	98.4	62.9	99.8	38.8	99.7	37.9
Exp3	93.5	63.6	95.9	65.6	93.5	89.3	95.2	86.1	98.9	68.7	98.6	69.4

5.4 Summary

In this chapter we presented our rapid retraining approach to solving the target-word adaptation problem for a speaker. Our goal was to rapidly improve the performance of a neural network-based recognizer on a target word for a speaker using only one example of that word by the speaker without degrading performance on all other non-target words. We presented the results of applying this approach to the task of recognizing an isolated word out of a small-vocabulary set of acoustically-similar words. On this task, the approach resulted in an average 84% reduction in the target word error rate. On average, the non-target error rate increased by 3%. Even though our software implementation of the approach was not fully optimized for speed (and in fact was quite inefficient), adaptation required approximately 30 seconds of real-time processing on a single workstation. Using personal computers commercially available today, this processing time is estimated to drop to 15 seconds without any changes to the algorithm itself.

Chapter 6

Parameter Optimization

The neural network retraining approach described in the previous chapter was both vocabulary and speaker specific. The focus in that chapter was on using target-word speech from a speaker to improve the performance on just that word for the speaker. The goal with the experiments in this chapter, on the other hand, is to use a small amount of speech, preferably a single utterance, to rapidly improve performance of our neural network based recognizer on *all* of the utterances by a speaker. We begin this chapter by describing the motivation for using this parameter optimization approach. The next two sections then describe the parameter we attempt to optimize and the optimization method itself. Finally, we present our experiments exploring this method.

6.1 Motivation

Many adaptation methods, especially those for neural networks, attempt to estimate too many parameters, resulting in poorer estimation and longer adaptation times. A brief review of our frame-based recognition process illustrates how this can happen. The input speech is first divided into slices (frames) of a few milliseconds in duration. For each frame, a signal processing stage then produces a few values that compactly represent spectral properties of the speech from that frame. These values, along with values from other frames, are then combined to form a larger vector of feature values. A probability estimator then generates an even larger set of subword-unit probabilities. These probabilities are used by a lexical search to produce the final word string. Notice that at each of these stages the number of parameters has grown.

Most speaker adaptation approaches modify values at one of three points in this sequence: the feature vector itself, the probability estimator, or the output probabilities. Since these methods change a large number of parameters, they frequently require substantial amounts of adaptation data in order to reliably estimate new values for the parameters. Rather than adjusting a large number of parameters late in the processing, adjusting only a single parameter earlier in the computation could be done with less adaptation speech, perhaps as little as one word. Optimizing this single parameter would have the effect of appropriately adjusting the necessary parameters appearing later in the analysis process. For the task addressed in this chapter, an ideal parameter would be one that explicitly accounts for much of the variability among speakers.

In the next section we describe such a parameter: the *Bark offset.*

6.2 Bark offset parameter

Originally developed for the objective calculation of loudness, the Bark scale critical band function was proposed to the International Standards Organization in the late 1950's to standardize the locations and limits of auditory critical bands.[65] The proposal introduced as a unit the "Bark" (named after the German acoustician Barkhausen), where a frequency difference of 1 Bark corresponds to the width of one critical band. This Bark scale function maps the measured critical band locations to the frequency (Hertz) scale and has thus been described as the "natural" frequency scale of the ear. Note that although the end mapping is similar, this new frequency scale is constructed differently from the mel scale, a pitch-scaling based function. For a discussion of the critical bands and their importance in human auditory theory, see [52].

One factor that varies among speakers, especially between adult males, adult females, and children, is the length of the individual's vocal tract. One effect of the difference in vocal tract lengths can be seen in the speech spectra of vowels. Bladon, *et al.* [4] compared average spectra, for the vowel $/\varepsilon/$, for male and female speakers of Northern British English and noticed that when presented on the Bark scale, the spectra differed, on average, only in their location on the Bark frequency axis. They hypothesized that by

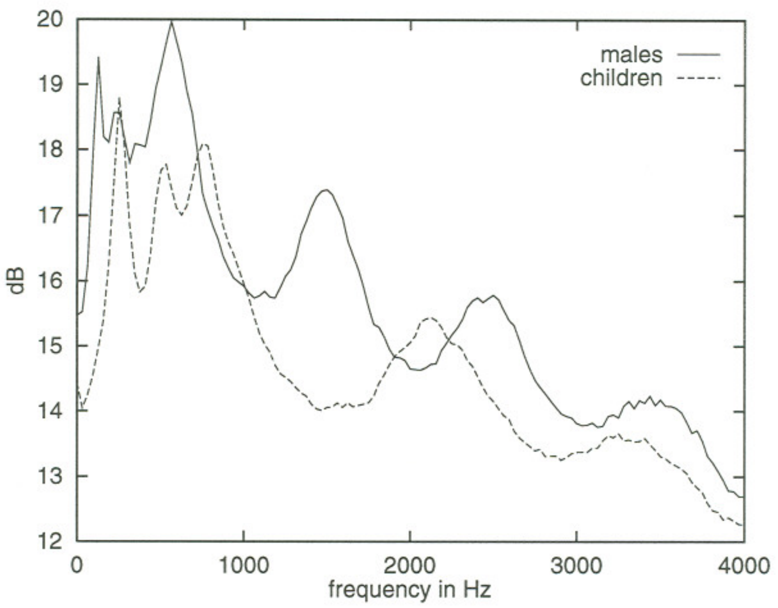


Figure 6.1: This figure shows average spectra for the vowel /ɛ/ from males and children. The average is computed over all male or child speakers in the **TIDIGITS** training set. Results are presented on the Hertz scale.

displacing all frequency components of the average female vowel downward by one unit on the Bark scale, the average spectrum of the same male vowel would be obtained. A more exaggerated difference can be seen between the spectra of males and children. Figures 6.1–6.3 show average spectra for the vowel /ɛ/ for both males and children. The figures show the spectra on the Hertz scale and the Bark scale and then illustrate how shifting the children's spectra by -1.5 Barks can produce a good alignment of the fundamental frequency and first formants and a better alignment of the second and third formants between the male and child speakers.

The experiments in this chapter explore the use of a *Bark offset* parameter. Briefly, this parameter is a single value, representing a number of Bark scale units, that specifies how to shift frequency components along the Bark axis. In the example shown in the figures, it is clear that a simple linear shift of spectral information along the Bark axis does not completely line up the two spectra. However, this kind of shift is both well-motivated and sufficient to provide some performance improvement. As will be described in Section 6.2.2, the parameter was also straightforward to implement within our recognition paradigm.

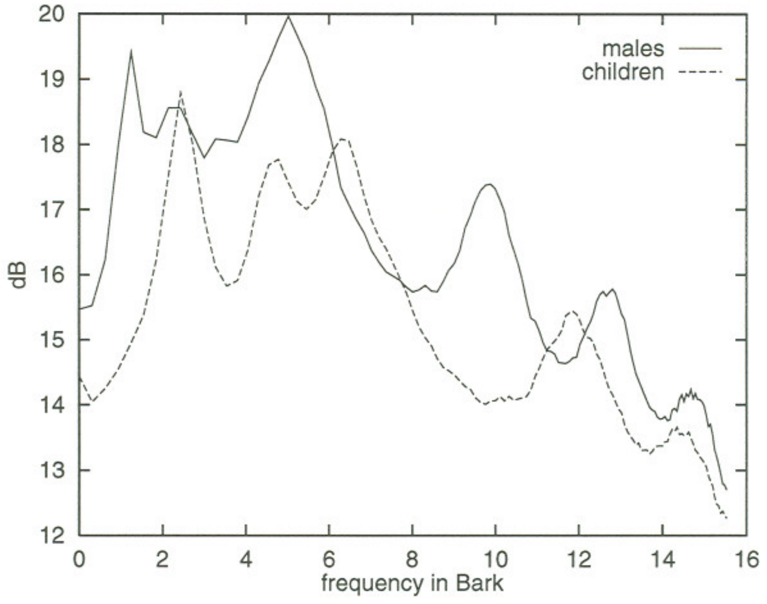


Figure 6.2: This figure shows average spectra for the vowel /ε/ from males and children. The average is computed over all male or child speakers in the **TIDIGITS** training set. Results are presented on the Bark scale.

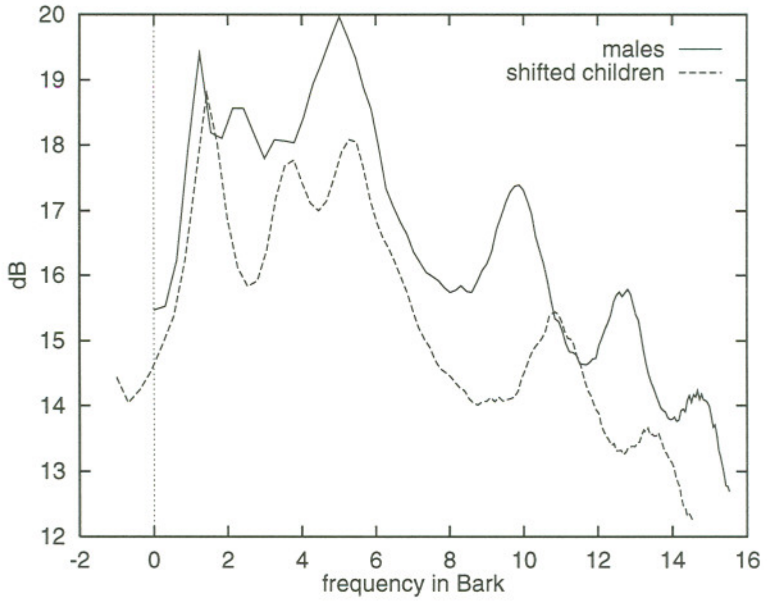


Figure 6.3: This figure shows average spectra for the vowel /ε/ from males and children. The average is computed over all male or child speakers in the **TIDIGITS** training set. The children's spectra have been uniformly shifted left (downward in frequency) by 1.5 Barks. Results are presented on the Bark scale.

Remember that the goal was to identify a parameter and optimization method that would allow for *rapid* tuning of a speaker-independent neural network recognizer to the speech of a specific individual.

6.2.1 Frequency scale warping

Our selection of this parameter was inspired by work done by Andreou, Kamm, and Cohen.[2] In fact, the idea of warping the frequency axis to normalize for different vocal tracts is not new. In 1977, Wakita [56] presented a method of vowel formant frequency normalization which used explicit estimates of the speaker's vocal-tract length and area functions obtained from the acoustic waveform. In a vowel identification task using steady-state portions of the vowels, normalization resulted in a 26% reduction in the identification error rate. Neuberg [43] examined several different transformations to the frequency scale. In an experiment on two speakers, he observed that piecewise-linear transformations of the first, second, and third formant regions of the frequency axis resulted in an improved correlation between vowel nuclei spectra from the two speakers. Recently, several papers have reported on experiments with various scalings of the frequency axis. Both Eide and Gish [14] and Lee and Rose [35] have proposed extensions to the work of Andreou, *et al.*[2] This work has also been extended by Kamm, *et al.*[27], who experimented with linearly scaling the frequency axis for speakers from the Switchboard corpus. The optimal scale factor was selected by trying different values and choosing the one that maximized the *a posteriori* probability of the data. Although most of the improvement occurred in this first step (adaptation), they also iteratively retrained their HMM by selecting an optimal scale factor for each training speaker, retraining using these scale factors, reselecting the optimal scale factor for each speaker, etc. This normalization reduced their error rate by approximately 9%. Similarly, Eide and Gish experimented with a simple exponential warping (designed to allow more adjustment at high frequencies than at low frequencies) of the speakers' frequency scales. However, to avoid the computational load of evaluating the system at many (arbitrary) warping parameter values for each speaker, they attempted to estimate the parameter value based on third formant values for the speaker relative to those for other speakers. Their error rates, also on Switchboard, showed an 8-10% drop for several different dataset sizes and conditions. Lee and Rose [35] proposed HMM-based procedures for estimating an appropriate scaling factor and described a simple implementation of the frequency warping using a direct modification of the filters in their front-end. Experiments on a telephone-based connected digit recognition task demonstrated a noticeable reduction in error rate. Wegmann, *et al.*[58] investigated an approach to vocal tract normalization based on an explicit voiced speech model. Their piecewise-linear mapping produced a 12% reduction in error rates on a Switchboard task.

6.2.2 Implementation

Within our system, the Bark offset parameter is implemented through a modification of the standard PLP [22] analysis. This analysis consists of a Hamming-windowed FFT, a warping of the power spectrum to the Bark scale, critical-band masking, equal-loudness pre-emphasis, intensity-loudness conversion, and finally LPC cepstral modelling. The original Hz-to-Bark transformation (an analytic approximation from Schroeder [15, p. 324]) and its inverse were

$$\Omega(f) = 6\ln\left\{\frac{f}{600} + \sqrt{\left(\frac{f}{600}\right)^2 + 1}\right\} \tag{6.1}$$

$$f(\Omega) = 600\sinh(\Omega/6) \tag{6.2}$$

A plot of this transformation is shown in Figure 6.4. Figure 6.5 shows a spectrum for an average male /ɛ/ when presented on the Hertz and Bark scales. Notice that the lower frequencies are expanded and the higher frequencies are compressed.

Critical-band masking filters

This transformation is used in the PLP implementation as follows. From each windowed frame of speech, 128 FFT coefficients are computed. Before the LPC analysis, these values are converted into 17 filter-bank coefficients. The shape of the auditory filter used is identical on the Bark scale for all filters, with each filter being separated from its neighbor by approximately one Bark in order to evenly cover the 15.6 Bark range. The filter shape is plotted in Figure 6.6. The Bark-to-Hz transformation (Equation 6.2) is used to precompute a) the range (in Hz) covered by each filter and b) the weighting factor

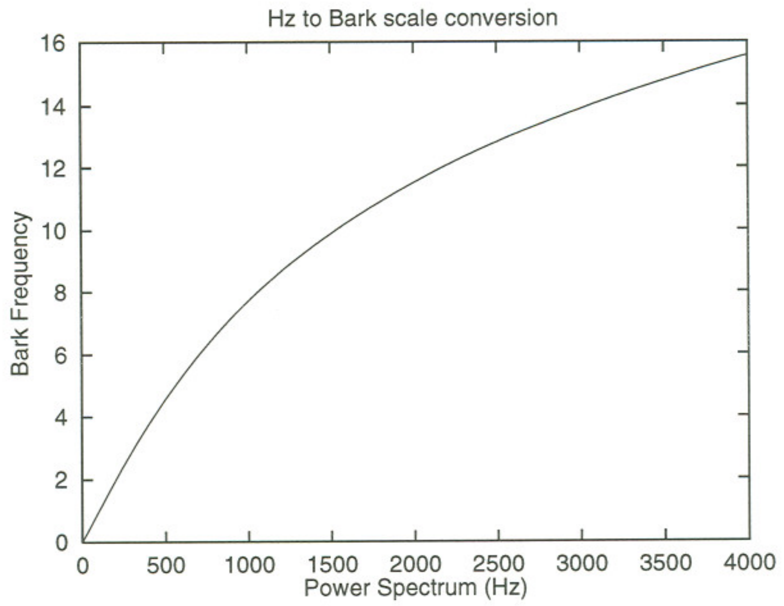
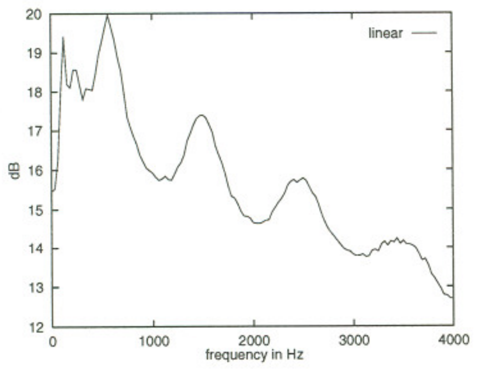
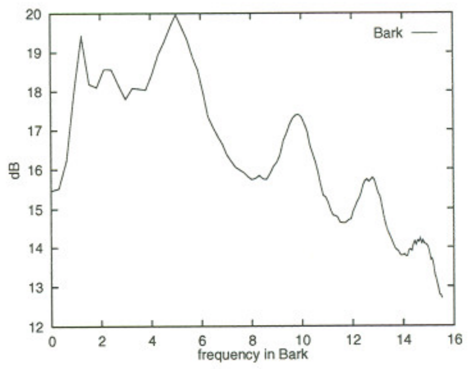


Figure 6.4: This figure shows a plot of the function $\Omega(f) = 6\ln(\frac{f}{600} + \sqrt{\left(\frac{f}{600}\right)^2 + 1})$ which converts from the Hertz scale to the Bark scale in PLP.



(a) Hz scale



(b) Bark scale

Figure 6.5: This figure shows spectra for the vowel $/\varepsilon/$ from males, averaged over all males in the **TIDIGITS** training set. Results are presented on the Hertz and Bark scales.

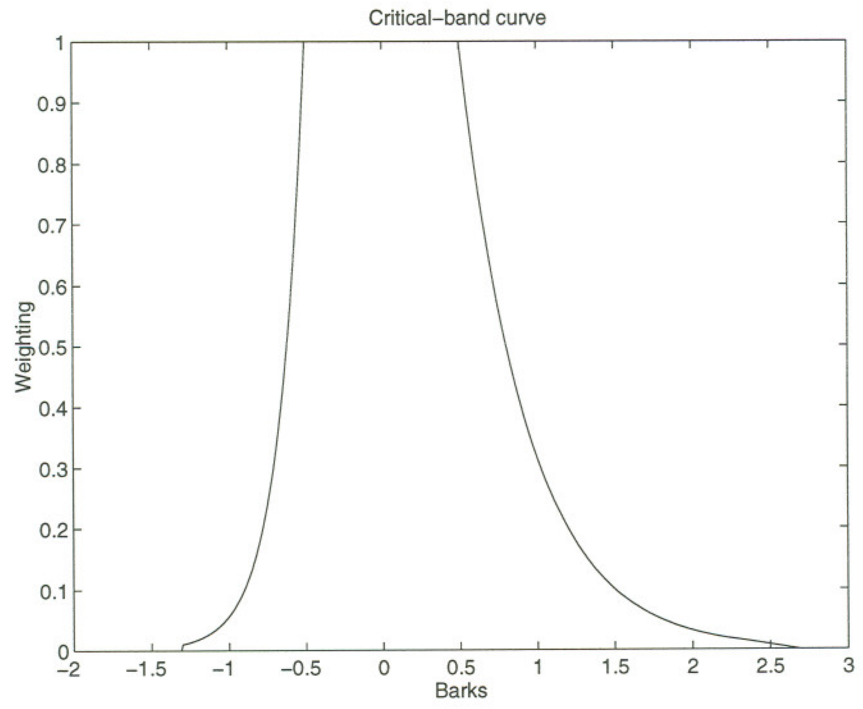


Figure 6.6: This figure shows a plot of the critical band filter used in PLP.

applied to each FFT sample point. Thus a change in the Hz-to-Bark transformation will affect, for each of the filter banks, both the range of FFT sample values on which the bank is computed and the actual weighting values for each sample.

Linear shift (offset)

The Bark offset parameter is implemented in our system through a modification of Equation 6.1:

$$\Omega_{\mathbf{offset}}(f) = \Omega(f) + \mathbf{offset}$$

Thus a negative offset shifts all values to lower frequencies, while a positive offset shifts all values to higher frequencies. The effect of adding this parameter value is to allow the filter bank ranges and weightings to vary. At no time was a range allowed to extend outside of the range of the existing FFT sample points. For ease of implementation, the range of offset values was restricted to be such that the center portion of the filter (with a weighting value of 1.0) would remain at least partially within the originally defined range. For simplicity, this range of (-2.3,3.5) was further constrained to [-2,3]. Thus offsets were

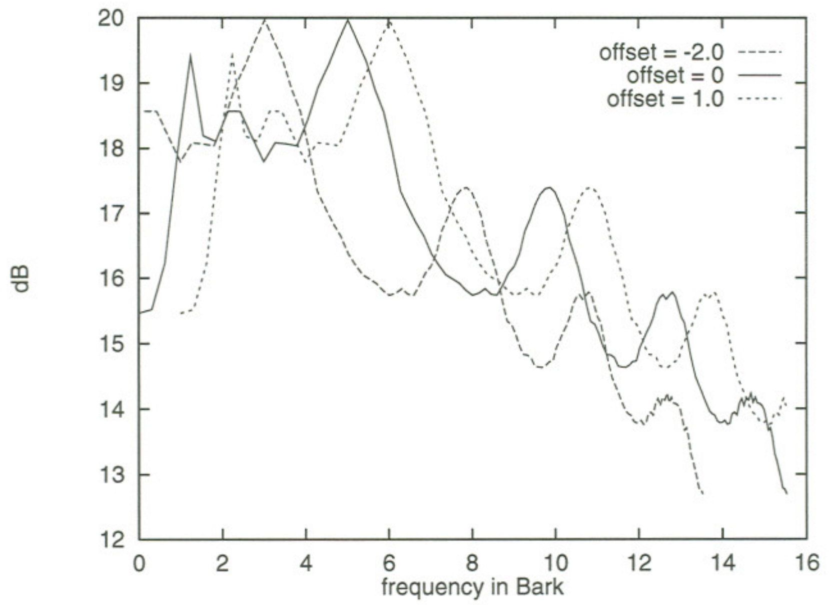


Figure 6.7: This figure shows the effects of different Bark offsets on a sample spectrum. The spectrum is an average for the vowel /ε/ over all males in the **TIDIGITS** training set.

only allowed in the range of -2 to +3 Barks. Examples of the effects of different offsets are shown in Figure 6.7.

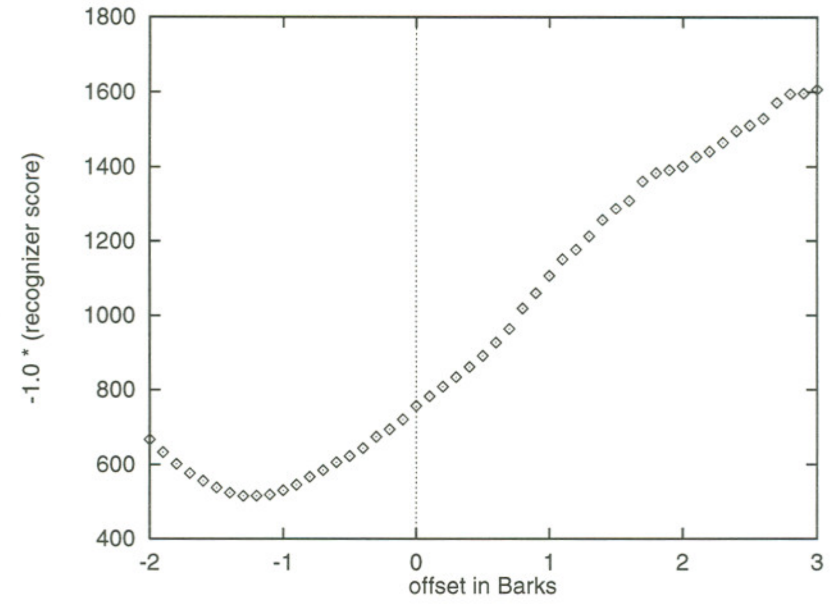


Figure 6.8: This figure shows the change in recognizer score as the Bark offset is changed for a seven-digit utterance spoken by a female child. The utterance is from the **TIDIGITS** training set.

6.3 Optimization

In the work done by Andreou, *et al.*[2], the optimal Bark scaling value was selected by evaluating the space of scale factors at regular intervals. In a preliminary analysis, we plotted the recognizer score as a function of the bark offset parameter. The plots indicated that, for each speaker, there was a single clear minimum region in a function that was approximately parabolic (see Fig. 6.8). This insight led us to select an optimization method that would take advantage of our knowledge of the general shape of the function. We selected Brent's algorithm [49, section 10.2], a cross between the golden section search and inverse parabolic interpolation, to perform the optimization. Given starting locations a, x_0, and b such that $a \leq x_0 \leq b$, $f(x_0) \leq a$, and $f(x_0) \leq b$, Brent's algorithm is guaranteed to find, to a specified precision of ϵ, the location x_{min} and value y_{min} of a local minimum of a function $y = f(x)$, where $a < x_{min} < b$, provided that the function has a continuous second derivative.

Excellent descriptions and analyses for the golden section search, inverse parabolic interpolation, and Brent's method can be found in Press, et al.[49, sections 10.1,10.2].

However, we present here a synopsis of their discussion and algorithm description. Briefly, the golden section search is a one-dimensional search for a local minimum within a specified range. The idea is based on the concept of successively bracketing a minimum. To illustrate the concept, we first describe how to locate a zero of a function by successively bracketing the zero. A zero can be bracketed with only two points (a, b) such that $f(a)$ and $f(b)$ are of opposite sign. Why? For a continuous function there must be at least one abcissa x_{zero} between a and b with an ordinate of zero. To find a zero, then, select an abcissa x between a and b at which to evaluate the function. If $f(x)$ has the same sign as $f(a)$, the new bracket should be set to (x, b). Conversely, if $f(x)$ has the same sign as $f(b)$, the new bracket should be set to (a, x). Continuing in this manner will eventually result in a bracket that is sufficiently narrow. Without any other information, the optimal point to select for the next function evaluation is the abcissa exactly halfway between the two bracketing points.

The same procedure may be used to locate a minimum of a function. However, bracketing a minimum requires a triplet (a, b, c) such that $f(b)$ is less than both $f(a)$ and $f(c)$. Why? Because the fact that the middle point is lower than the two outer points provides the guarantee of at least one point lower than the outer two. As above, the process of successively bracketing the minimum requires that the function be evaluated at a point x between a and c. Then a new triplet is selected such that the range is smaller yet the center point is still lower than the outer two. Again, without any other information about the shape of the function, the best point at which to evaluate next is the one located at a fraction 0.38197 into the larger of the two intervals (a, b) and (b, c), measuring from b, the center point of the triplet. Interestingly, this fraction (actually $\frac{3-\sqrt{5}}{2}$) of 1.0 and its complementary 0.61803 are the fractions of the *golden mean* or *golden section*. Hence the name of the search.

It would be convenient to have a more informed way of selecting each successive point for evaluation. Remember that the golden section search assumes *no other knowledge* about the shape of the function. Many functions, however, are parabolic, at least in the neighborhood of the minimum. If the function is parabolic in the region of the minimum being sought, why not fit a parabola through the three points of our triplet and jump straight to the minimum of the parabola, which should be close to the minimum of the function? This is the concept underlying *inverse parabolic interpolation.* (In this case, *inverse* merely refers to the fact that the abcissa corresponding to the minimum ordinate is desired, rather than the ordinate value itself.) Unfortunately, the equation giving the abcissa of the minimum of a parabola through 3 points will instead give the maximum of the parabola if the function is so shaped. Thus inverse parabolic interpolation should only be used along with a check to ensure that the new value is lower than (or equal to) the lowest of the three points.

The dilemma here is that inverse parabolic interpolation is fast but potentially inaccurate, while the golden section search is much slower but will definitely converge to the minimum. Which is best to use to set the next evaluation point? Brent's algorithm solves this problem by using built-in heuristics to decide when one approach is better than the other. Not only that, but it is spartan in its use of function evaluations. For the task at hand this is critically important, since a "function evaluation" corresponds to a complete recognizer pass over the adaptation utterance, including signal processing, feature generation, probability estimation, and Viterbi search to produce the function value. For information on the actual heuristics used in Brent's algorithm, see [7, Chapter 5].

We have illustrated the search procedure on a simplified example in Figure 6.9. Notice in this particular example that only parabolic interpolation is used to set the next evaluation point. For a less-parabolic (but more likely) function the algorithm would likely switch back and forth between interpolation and golden section. In summary, here is the procedure for Brent's algorithm:

1. Three points which bracket a minimum are given as the three starting points. In the diagram, these points are located at a, $x_i = x_0$, and b. The function is evaluated at x_0.

2. A new point, x_{i+1}, is selected either through a golden section division or by finding the minimum of a parabola through the three points.

3. The function is evaluated at this point.

4. Three new points are selected based on the location of the new point and the value of the function at the point. Intuitively, we select the lowest of the four points as the middle and keep the two that are closest to this middle one as the bounding points. Mathematically, if $x_{i+1} < x_i$, then

$$(a, x, b) = \begin{cases} (x_{i+1}, x_i, b) & \text{if } f(x_{i+1}) > f(x_i) \\ (a, x_{i+1}, x_i) & \text{if } f(x_{i+1}) < f(x_i) \\ \text{heuristic} & \text{otherwise} \end{cases}$$

Otherwise (i.e., if $x_{i+1} > x_i$), then

$$(a, x, b) = \begin{cases} (a, x_i, x_{i+1}) & \text{if } f(x_{i+1}) > f(x_i) \\ (x_i, x_{i+1}, b) & \text{if } f(x_{i+1}) < f(x_i) \\ \text{heuristic} & \text{otherwise} \end{cases}$$

In either case the search continues back at step 2 with the new points. Of course, if $x_{i+1} = x_i$ or the two outer points are equal to within $\pm\varepsilon \cdot x_i$ the search terminates with x_i as the answer.

Notice that the function is never evaluated at the original two endpoints. All the search needs to know when it starts is that the function values at those two points are both higher than that of the middle point.

6.3.1 Implementation

We have implemented Brent's method as a new Tcl function in the OGI Toolkit. This function takes as input a function to be minimized, a range within which to search, a starting value, and a tolerance value ε that determines when to quit searching. The search terminates when the outside points are less than $2 \times x \times \varepsilon$ apart, where x is the best abcissa found so far.

The adaptation must be both simple and rapid. Practically, a single utterance by a speaker will be used to estimate $p_{optimal}$, the optimal value of the parameter. This parameter value will then be used to recognize all of the remaining utterances by the speaker. In all experiments, the optimization is performed on the negative of (the log likelihood of) the recognizer score for the adaptation utterance.

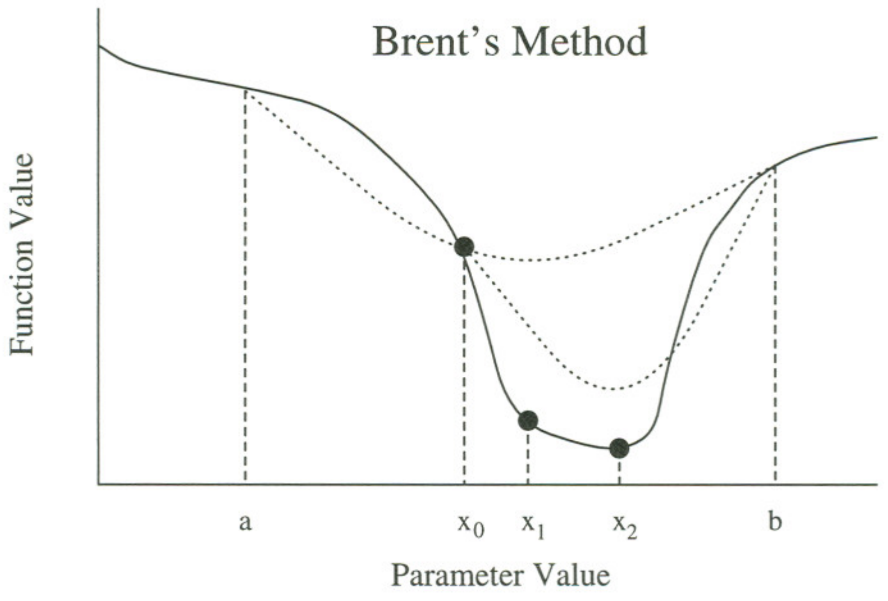


Figure 6.9: This figure shows an example optimization using Brent's method. In this example a parabola is fitted through the initial three points at a,x_0, and b. The minimum of this parabola is at x_1. A new parabola through x_0,x_1,and b gives a new minimum at x_2. This is the final value.

Note that the success of this method for rapid adaptation depends on several assumptions:

1. Brent's algorithm requires the initial specification of a range $[a,b]$ for the parameter value p within which the recognizer score $R(u,p)$ has a minimum.

2. For speed purposes, it must be possible to estimate $p_{optimal}$ with a small number of function evaluations (recognizer passes over the utterance).

3. The estimate of $p_{optimal}$ must be accurate enough to improve the recognizer performance on the remaining utterances for the speaker. Note that this implies that the score produced by the recognizer is fairly well correlated with the accuracy of the recognizer.

6.4 Experiments

This section describes the experiments in which a Bark offset parameter was optimized for each speaker as described in the previous section. Experiments were performed both on children's speech and adults' speech. All of the digit string adaptation experiments were unsupervised, meaning that the system did not presume to know the identity of the digit string. The utterance was, however, assumed to be a string of digits. Conversely, all experiments on the **PhoneBook** corpus were supervised—the system used its knowledge of the word's identity during adaptation.

6.4.1 Children's digit strings

Children's vocal tracts tend to be shorter than those of adults. Since the Bark offset parameter is intended to provide a rough normalization for the vocal tract length of the speaker, it is conceivable that with an appropriate setting of this parameter for each speaker the performance of an adult-trained recognizer on children's speech could be substantially improved. This was the goal in these experiments. The **TIDIGITS** corpus (described in Sec. 4.2) was selected for the experiments because it provides children's telephone speech in a controlled environment on a constrained task. The test set utterances from this corpus were used for all evaluations. There were 51 children in the test set.

The networks used in these experiments had 209 output units, one for each of the context-dependent phonemes present in the eleven allowed digit words ("zero" through "nine" plus "oh"). It had 56 inputs and 200 hidden units as described in Section 2.2.1. The language model allowed the recognized string to consist of one or more of the digit words, optionally separated by silence.

Recognizer trained on *Numbers* corpus

For this experiment the recognizer was trained on digits from the **OGI 30,000 Numbers** corpus (*cf.* Sec. 4.1). No age analysis was performed on this corpus, but the vast majority of callers appear to have been adults. Remember that the goal was to adapt an adult-trained recognizer.

The following procedure was performed for each child:

- For each 1-digit utterance (out of 11 spoken by the child), Brent's algorithm was used to obtain the offset that maximized the recognizer score for that utterance. The offset was then used to test, for digit-level accuracy, the remaining 76 utterances spoken by the child. For Brent's algorithm, an offset range of [-2.0,3.0] with a starting value of 0.0 was used.
- An average of these eleven accuracy scores was computed.

Computing the average, across all of the children, of these averaged accuracy scores produced the overall system error rate. The same procedure was followed using 7-digit utterances to adapt. The baseline performance was obtained by evaluating each child's utterances using the default offset of 0.0. Baseline, 1-digit adaptation, and 7-digit adaptation results are presented in Table 6.1. In comparison, the adult TIDIGIT performance was 96.17% digit accuracy. Also shown in the table is the average number of function evaluations, i.e. the number of times the recognizer evaluated the adaptation utterance during the search for the best offset. Although only presented for comparison, the table also includes the average CPU time (in seconds) needed to perform these evaluations. In general, longer strings required more time and more evaluations required more time.

Table 6.1: This table presents results for an **OGI 30000 Numbers**-trained recognizer adapted to children's **TIDIGITS** utterances. Results are shown both for using a single digit to adapt and for using seven digits to adapt. Also shown are the average number of recognizer evaluations required to obtain the optimal offset and the average number of seconds needed on our hardware to perform these evaluations.

	Digit		String	Average	Average time
Condition	Correct	Accuracy	Correct	recog. evals	(seconds)
Baseline	89.95%	75.41%	57.64%	N/A	N/A
1-digit	92.33%	83.52%	69.02%	10.4	21.8
7-digit	97.98%	89.41%	76.65%	9.6	34.5

Recognizer trained on *TIDIGITS* corpus

In the previous experiment the baseline performance for children was extremely poor. To see how much improvement was possible for a network trained on task data, we retrained

Table 6.2: This table presents results for an adult **TIDIGITS**-trained recognizer adapted to children's **TIDIGITS** utterances. Results are shown both for using a single digit to adapt and for using seven digits to adapt. Also shown are the average number of recognizer evaluations required to obtain the optimal offset and the average number of seconds needed on our hardware to perform these evaluations.

	Digit		String	Average	Average time
Condition	Correct	Accuracy	Correct	recog. evals	(seconds)
Baseline	95.74%	90.42%	80.21%	N/A	N/A
1-digit	96.98%	93.79%	87.41%	10.8	20.3
7-digit	99.00%	96.52%	90.96%	10.1	36.4

the neural network using speech from the adult training speakers in the **TIDIGITS** corpus. After training, baseline performance for adults on the test set (still **TIDIGITS**) was 99.3% digit accuracy.

We then performed the same adaptation procedure for children's speech as in the previous experiment. Results are in Table 6.2. There was still a substantial reduction in the average error rate. The number of recognizer evaluations increased slightly as well. The CPU time needed for the adaptation was approximately the same.

6.4.2 Normalization using adults' digit strings

In the last set of experiments the goal was to demonstrate that the same method could be used to improve performance on any speaker—not just children. Because the unsupervised adaptation performance improvement was not as large as expected, normalization was examined as well. The next paragraph describes this.

Normalization In the earlier experiments with children's speech there was a mismatch between the training and testing conditions. In both experiments the network was trained without any Bark offset parameter at all, which corresponds to a parameter value of 0.0. However, after adaptation the system was tested on Bark-shifted speech. While this may have been appropriate for the voiced regions of speech, the shift likely changed other regions of speech in ways never seen by the neural network. Essentially, the network had not been trained on appropriately-shifted speech. To account for this, a speaker-normalized recognizer was created by iteratively retraining the digit recognizer on adults' speech *using the optimal offset for each adult speaker.* The training procedure was as follows:

1. The original recognizer used in the adaptation experiment was set as the initial recognizer.

2. This recognizer was used to select the optimal offset for each utterance in the training and cross-validation sets.

3. Along with the known word transcription of the utterance and pronunciation models for the eleven allowed words in the dataset, this optimal offset for the utterance was used in the recognizer to automatically generate feature vectors for the utterance.

4. The new training vectors were used in our standard on-line stochastic training algorithm (see Section 2.2.2) to produce a new recognizer (network). The training started from random weights and was stopped as soon as the maximum performance on the cross-validation set was reached.

5. This new recognizer was the "norm1" recognizer. Repeating steps 2–4 produced the "norm2" recognizer.

Practically, one can envision using the speaker-independent baseline recognizer to initially recognize a few utterances by the speaker. In the background, the system would adapt the normalized recognizer using the initial speech (by selecting an apppropriate offset) to produce a much better recognizer which would then be used to recognize the remaining utterances by the speaker.

In the next section we present details of the experiment and the results obtained.

OGI 30000 Numbers

Since the baseline performance for adults on the **TIDIGITS** corpus was already 99.3% digit accuracy, this experiment was performed on the **OGI 30000 Numbers** corpus, which is noisy telephone speech rather than merely downsampled high-quality lab speech. As described in Section 4.1, only the address and zip code number strings were used. During evaluation, one utterance was used to adapt and one to test. During the iterative retraining, though, the optimal offset value for *each* utterance was found and subsequently used to generate features for the utterance.

For each of the three recognizers (unnormalized, norm1, and norm2), performance on the 376 speakers was evaluated both before and after adaptation. Adaptation was performed using Brent's algorithm to optimize the score for the adaptation utterance. The offset obtained was then used to recognize the test utterance, producing a binary correct/incorrect classification for the utterance. The same process was repeated by adapting to the second utterance and testing on the first. These two utterance scores together formed the score for the speaker. All speaker scores were averaged together to obtain the overall utterance score. Results for the three recognizers are shown in Table 6.3. The *street addresses* performance refers to the performance on street address utterances when using zip codes to adapt, and vice versa. The best result was obtained by adapting after only one normalization pass. The drop in baseline performance for the normalized recognizers was to be expected, since the "baseline" for each system is to use a Bark offset of 0.0 for all utterances. After training a normalized network, any unnormalized (unadapted)

Table 6.3: This table presents results for a normalized adult-speech **OGI 30000 Numbers** network adapted to and tested on adults. The *street addresses* performance refers to the performance on street address utterances when using zip codes to adapt, and vice versa.

	Digit level accuracy (%)					
	Unnormalized		Norm1		Norm2	
Eval set	Base	Adapt	Base	Adapt	Base	Adapt
Street addresses	97.9	98.2	97.2	98.2	96.6	97.9
Zip codes	95.8	96.8	95.7	97.7	94.7	96.6
Overall	96.9	97.5	96.4	97.9	95.7	97.3

speech the system processes will not match the training conditions and should therefore not be as well recognized.

6.4.3 Medium vocabulary isolated word task – the *PhoneBook* corpus

We also performed experiments using isolated words from the **PhoneBook** corpus. This corpus has a larger vocabulary than the digit corpora used in the previous experiments. The training, development, and test sets were partitioned in a way that for the most part ensured different training words than test. A description of the corpus can be found in Section 4.3. After training, the experiments used only the development and test set speakers.

Bark offset adaptation

The baseline recognizer used the same PLP input features as in the other experiments. The neural network was a three-layer network with 56 inputs, 200 hidden units, and 534 context-dependent phonemic outputs. Since each utterance consists of only a single word, the grammar (language model) was simple. It would accept a single word, which could be any of the allowed vocabulary words, optionally preceded and/or followed by silence or noise.

The goal was to use the same general adaptation approach used on the **TIDIGITS** utterances. For each of the callers, an optimal offset for the speaker would be determined using Brent's algorithm and then used when recognizing the evaluation utterances for the speaker. A major difference, however, between these experiments and the digit experiments is that this adaptation was *supervised*. Although unsupervised adaptation is possible, in this task it would require a vocabulary search for each recognizer evaluation. Since several recognizer evaluations are needed in order to determine the optimal offset value, this could be rather time-consuming. For this reason the experiments described here assume that the identity of the adaptation word(s) is known.

Some preliminary analyses indicated that the following issues would be important:

Using overall vs. just word score The overall utterance includes not only the word but silence and noise. Since the Bark offset parameter is really applicable only to speech, including non-speech noise into the calculation of the recognizer score to be optimized is not appropriate. This concern applied to the earlier digit string experiments as well, but the digit strings were typically longer than the isolated words (resulting in a larger percentage of the utterance that is speech) and the earlier adaptation was unsupervised (meaning that a change in the recognized word's identity as the parameter was varied could result in drastic, abrupt changes in the word score). In contrast, supervised adaptation allows the system to compare the changing scores of a fixed word. For these reasons, both the utterance and word scores were examined as values to optimize with Brent's algorithm.

Number of adaptation utterances It quickly became apparent that due to the increased vocabulary and number of network output categories, one utterance was typically not sufficient to provide an accurate estimate of the optimal offset value for a speaker. Thus, one of the areas explored was the number of adaptation utterances needed to obtain a successful offset value.

Offset-combination methods When using more than one utterance to adapt, an obvious question arises as to how to combine them. One possible approach is to concatenate the utterances into one adaptation utterance. Having to optimize several digit strings at once might produce a more robust offset estimate. Another approach, and the one taken in these experiments, was to assume that the optimal offset values for most of the individual utterances are close to one another. In that case, optimal offset values can be computed for each adaptation utterance and then combined.

The following paragraphs describe the experiments performed using the speakers in the development set. Note that only in the final experiments is adaptation actually being performed. The purpose of the preliminary experiments is to find out, first, whether it is possible to achieve a performance improvement using any number of utterances, and, second, to determine how few are needed.

Overall vs. word score The first experiment used a 30-speaker subset of the development set. For each of the (up to) 76 utterances by each speaker, Brent's algorithm was used to compute the offset that maximized the recognizer score for the utterance. This was done both when using the overall utterance score and when using only the word score.

Table 6.4: This table shows the percentage of utterances (words) correctly recognized when using the median offset for each of 30 development set speakers. The median is computed over all (potentially 76) utterances by a speaker. Both utterance score and word score were optimized. Results for both are shown.

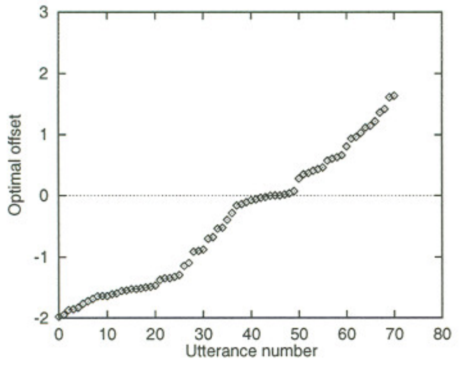
Condition	Percent correct
Baseline	74.6
Utterance	68.7
Word	76.7

Figures 6.10–6.13 show sorted plots of the optimal offsets found for the utterances of two female and two male speakers. The horizontal dashed line is at an offset of zero. The two things to note here are

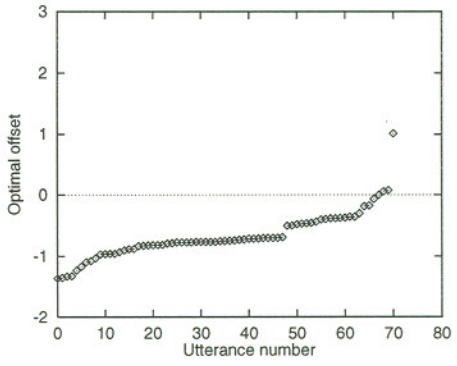
1. For each of the speakers, there is a definite offset value region into which most of the offsets fall. Assuming that an offset value close to the optimum will still improve the score for a word, these plots suggest that there is an offset (or a narrow range of offsets) that will result in improved scores for most of the words.

2. The offset value range is much narrower when using the word-only score than when using the overall utterance score.

The median offset The narrow offset range containing most of the optimal offsets suggested that finding a way to select a value within this range would be useful. Assuming that the optimal offset values for more than half the utterances fall in the desired range, simply selecting the median value should work. This experiment used the same 30 speakers. Both for the utterance-score and word-score approaches, all of the utterances for a speaker were tested using the median of the optimal offsets for the speaker's utterances. The summary results are shown in Table 6.4. Clearly, these results show a large difference in performance between the utterance-score and word-score methods, with the word-score approach showing a small improvement over the baseline. More interesting, however, are the individual performance changes shown in Figure 6.14. This figure is a scatter plot contrasting the before (offset of zero) and after (median offset) recognition rates for all 30 speakers. Although the average recognition rate is only slightly better after than before, it appears that most of this improvement is concentrated in a few of the poorer-performing

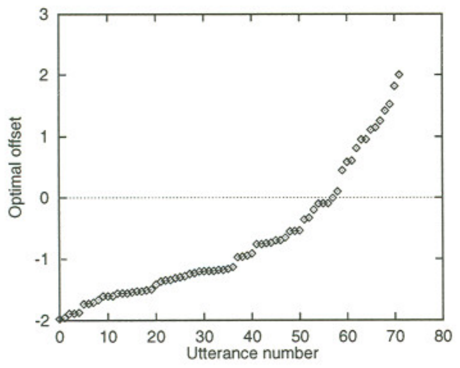


(a) Utterance

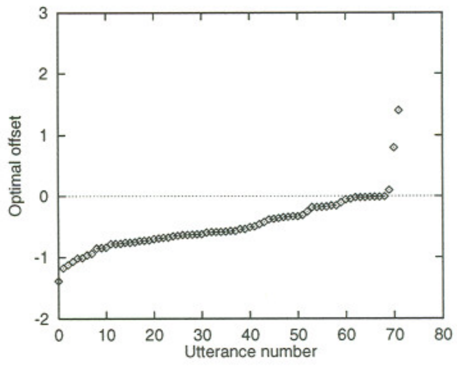


(b) Word

Figure 6.10: This figure shows the optimal offsets individually obtained for each of speaker f09's utterances using Brent's method on the recognizer score for the utterance.

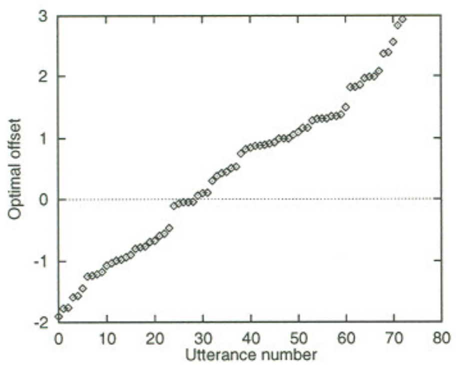


(a) Utterance

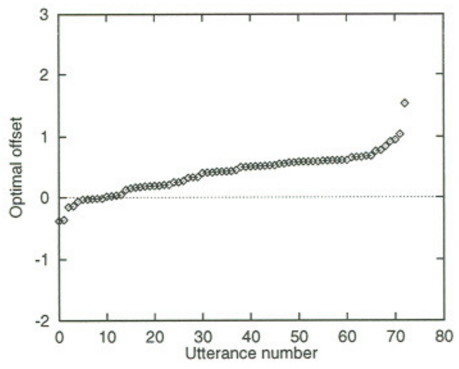


(b) Word

Figure 6.11: This figure shows the optimal offsets individually obtained for each of speaker f0d's utterances using Brent's method on the recognizer score for the utterance.

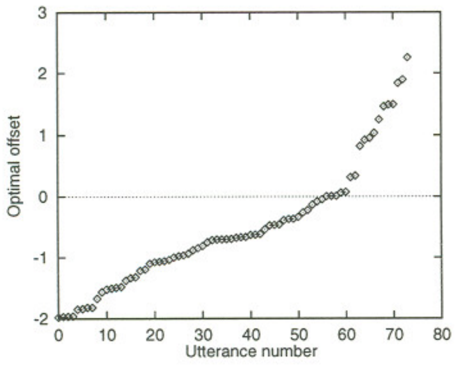


(a) Utterance

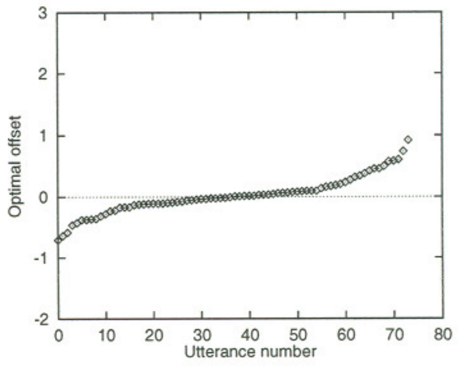


(b) Word

Figure 6.12: This figure shows the optimal offsets individually obtained for each of speaker mh7's utterances using Brent's method on the recognizer score for the utterance.

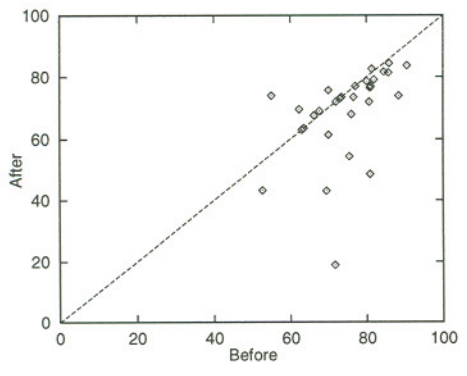


(a) Utterance

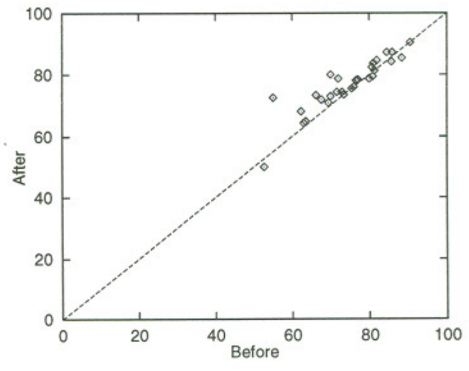


(b) Word

Figure 6.13: This figure shows the optimal offsets individually obtained for each of speaker mh9's utterances using Brent's method on the recognizer score for the utterance.



(a) Utterance



(b) Word

Figure 6.14: This figure shows a scatter plot of the percentage of utterances (words) correctly recognized before (offset of zero) and after using the median offset for each of 30 development set speakers. The median is computed over all (potentially 76) utterances by a speaker. Both utterance score and word score were optimized. Results for both are shown.

speakers. Two conclusions that can be drawn from this experiment are

1. Using the median of the optimal offsets for a speaker's utterances will result in a performance improvement (when optimizing the word-only score).

2. This improvement is the result of a dramatic improvement for a few of the worst-performing speakers with little change for the rest.

Number of adaptation utterances Since the eventual goal is rapid adaptation, preferably using only a single utterance, it is not reasonable to have to estimate an offset for 76 utterances by a speaker. This next experiment therefore focused on determining the approximate number of offsets (utterances) needed in order for the median to provide a good performance improvement. Note that this is not yet adaptation—it is merely another step towards identifying a method of obtaining the optimal offset value (for a speaker). For each of the 30 speakers, seven utterance subsets of sizes 64, 32, 16, 18, 4, 2, and 1, respectively, were randomly selected. For each subset the median offset value was computed and then used to test on *all* the utterances by the speaker. In this experiment the word score was optimized. Results, averaged over all 30 speakers, are shown in Table 6.5.

Table 6.5: This table shows the percentage of utterances (words) correctly recognized when using different numbers of utterances to select the offset for a speaker, for each of 30 development set speakers. The median offset for a speaker is computed using offsets from 1,2,4,8,16,32,or 64 utterances. Optimal offsets were obtained using word score.

Number of Adaptation Utterances	Percent correct
Baseline	74.6
1	74.7
2	75.2
4	76.2
8	76.6
16	76.4
32	76.7
64	76.7
76	76.7

It appears that a set of 32 words is sufficient to reach the final performance, while a set of 8 words is sufficient to achieve most of the performance gain.

Final results

This approach was tested on the final test set. Each speaker's data was partitioned by randomly selecting 8 utterances to be adaptation utterances, with the rest used as evaluation utterances. The optimal offset for each of the 8 utterances per speaker was obtained using Brent's algorithm on the word score. The median of these 8 offsets for each speaker was then used when testing on the evaluation utterances for the speaker. For comparison only, the median offset for each speaker was also computed from randomly-chosen subsets of 1, 2, and 4 of the adaptation utterances. Results for all adaptation-set sizes are shown in Table 6.6. The table also shows the results on the 64 speakers whose baseline performance was between 40% and 70%. Although the overall error rate only declined from 22.6% to 20.9% (7.5% reduction), the average error on speakers with medium-level baseline performance decreased from 39.8% to 34.5% (13.3% reduction). This effect is easier to see in Figure 6.15. This figure shows a scatterplot of pre- vs. post-adaptation performance for all of the speakers in the test set.

Table 6.6: This table shows adaptation results on the test set for the **PhoneBook** corpus. Results are shown for the condition of computing the median offset using 1,2,4, or 8 utterances by a speaker. Results for the subset of speakers whose baseline performance was between 40% and 70% are shown in column 3. Word score was optimized when obtaining individual offsets.

Number of Adaptation Utterances	Percent correct	
	Entire set	Middle
Baseline	77.4	60.2
1	78.0	63.3
2	77.7	63.1
4	78.6	64.8
8	79.1	65.5

6.5 Summary

This chapter examined a parameter optimization approach to solving the speaker adaptation problem. The goal was to rapidly improve the performance of a neural network-based

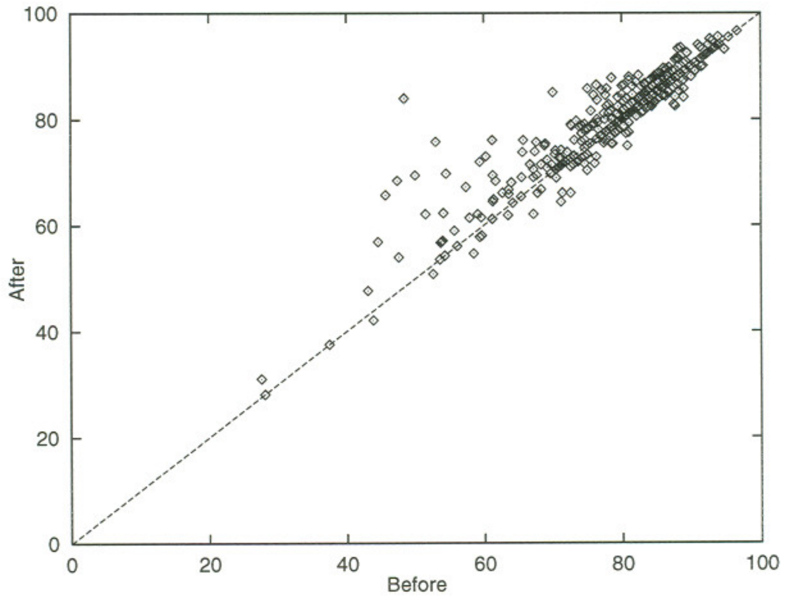


Figure 6.15: This figure shows a scatterplot of pre- and post-adaptation results on the test set for the **PhoneBook** corpus. Results are shown for the condition of computing the median offset using 8 utterances by a speaker. Word score was optimized when obtaining individual offsets.

recognizer on the speech of a new speaker using only one or a few utterances by the speaker. This approach was applied in three different tasks. First, a recognizer trained on connected digit speech from adults was adapted to children speaking connected digit strings. On average, the digit error rate decreased by 35% using only 1 digit to adapt (65% when using seven digits). Second, a recognizer trained on connected digits (spoken by adults) was normalized to new adult speakers by iteratively optimizing the training features and retraining. This resulted in a decrease of 32% in the digit error rate when using only a few digits to adapt. Third, a recognizer trained on isolated words was adapted to the speech of other speakers using eight words from the new speaker. Over all of the test speakers, the word error rate decreased by 7.5%. For speakers whose baseline performance was between 40% and 70%, the reduction was 13.3%. In each task, the time to perform adaptation was proportional to the time needed to recognize the adaptation words. In the experiments performed, this multiplier varied between 8 and 12.

Chapter 7

Discussion & Conclusions

7.1 Summary and Discussion

In this thesis work we investigated methods for rapidly adapting neural network-based speech recognition systems to new conditions. Specifically, we examined both a model-based approach and a feature space-based approach and found both to be successful within certain domains.

Rapid retraining

We tested our model-based approach, rapid retraining of the hidden-to-output layer weights of the neural network, in the domain of improving poorly-recognized words. We first collected the **Huh?** corpus, a collection of highly inter-confusable words spoken multiple times by over 100 callers. On this corpus we demonstrated that, indeed, it was possible to improve performance for a caller on a given word by combining feature vectors from a single example of the desired (target) word and feature vectors originally used in training the network, then retraining all of the hidden-to-output weights outputs. We then demonstrated that by training only weights to the target outputs, those corresponding to the phonemes present in the target word, one could reduce the overall number of training vectors while still maintaining the performance improvement. This dramatically reduced the time needed to train. These first two approaches had the drawback of decreasing performance on certain non-target words, some of which were initially well-recognized. To remedy this, we developed an incremental adaptation scheme that adapted only after misrecognizing an example of the target word. The first retraining used only a small amount of speaker-specific data, with each subsequent retraining using substantially more. This incremental approach had the desired behavior of minimizing the effect on non-target words while retaining most of the improvement on the target word by only doing as much adaptation as was necessary to improve the target word. On average, incremental retraining reduced the error rate on the target word by 84% while increasing the non-target error rate by only 3%.

The two greatest strengths of the rapid retraining approach are its speed and accuracy. The first retraining (in the incremental-retraining method), which covered most of the cases, was extremely fast. With optimization and/or hardware support, it would be possible to do the retraining within a few seconds, i.e. between utterances in a human-computer dialogue. This is tremendously useful. Typically during a telephone interaction the computer system is only working between the time it begins to receive the person's speech and the time it sends a reply back. Once the reply has been started, the computer is idle. If the reply only takes 1 or 2 seconds, it would be possible to adapt to a missed word within that time. The impressive accuracy obtainable with this approach is also of great benefit. Practically speaking, there are many tasks for which certain words must not be missed. On a voice-only Web browser, two extremely important words are *back* and *help*. In most tasks, *yes* and *no* are important. The ability to immediately boost performance on a given word can help improve error recovery.

The two greatest limitations of this approach are the requirement of supervised adaptation and the method's potential lack of scalability. In order to train the network, the system requires a time-aligned transcription of the adaptation word. Although the alignment can be automatically determined, the identity is more difficult to obtain, especially if only retraining when recognition errors have occurred. The system needs to know not only when an error occurred but also what the person intended to say, placing a heavy burden on the dialogue (error-recovery) component of the system. For many small-vocabulary tasks this may not be a problem, but it is important to consider. The scalability issue is a bit more serious. One reason the method is so rapid is that the system only needs to adjust weights going to outputs whose phonemes are present in a single target word. Although it is unclear whether simultaneously adapting to more than one target word would be successful, it is clear that adjusting more weights will take more time. The very factors responsible for the method being fast and effective also make it unlikely that the method will be useful outside the domain of rapid target-word adaptation.

Parameter optimization

Our feature space-based approach, parameter optimization based on the output score of the recognizer, is one that has enjoyed some popularity recently with the HMM community. We presented a motivation for the use of a specific simple parameter, the *Bark offset*, and then described our implementation of this parameter. We described a simple optimization method and then showed that it and the parameter were appropriate for the task of rapid adaptation of an adult speech trained recognizer to children's speech. We reduced the digit error rate on children's speech by over 30% using only a single digit to adapt. Directly performing this adaptation on adults' speech was only partially successful, so we adapted to the training set and retrained. Using this retrained network, adapting to adults' digits produced a 32% reduction in the error rate. Finally, we briefly examined the effectiveness of the technique on a medium-vocabulary isolated-word task. Although a single adaptation utterance was insufficient for adaptation in this domain, using 8 adaptation utterances (words) we reduced the error rate by 7.5%. Interestingly, this performance improvement was concentrated among speakers for whom the baseline performance was neither excellent nor terrible. For these average speakers, the error reduction was 13.3%.

The major strength of this approach is that it uses an extremely small amount of speech to adapt to a speaker. Even the 8 words required in the later experiments are not unreasonable for adaptation. For speakers encountering many recognition errors, one could just retain the speech for all words spoken so far and begin adapting after 8 words. If each new utterance could be processed between prompts, the minimal time needed for the adaptation itself (merely a parameter selection) would allow this approach to be used in a real interaction. Unfortunately, the greatest limitation of this approach is that the time to process an adaptation utterance depends on the recognizer speed. For all of the experiments (digits and isolated words), the time needed to process one adaptation utterance was 8–12 times the time needed to run the recognizer over the utterance. Thus, for a digit recognizer that could process an utterance in one-half real-time it would take about 2 seconds to adapt based on a single digit of approximately a half-second in length. Adaptation of longer words would take correspondingly more time.

7.2 Future work

There are of course many opportunities and avenues for extending this research. This final section presents a few ways each for both the rapid retraining and parameter optimization approaches and concludes with some suggestions on ways to combine the two approaches.

Rapid retraining

Throughout this thesis we have referred to the vectors generated from the target words as speaker-dependent, implicitly assuming that much of the benefit of retraining comes from a normalization to the speaker. Our goal, of course, was improved performance on a target word for a target speaker, and, while there may have been some benefit from adaptation to the speaker, it is more likely that the effective adaptation was to the vocabulary (in this case, the word). Even within the same task, it would be interesting to examine both vocabulary-dependent but speaker-independent adaptation and speaker-dependent but vocabulary-independent adaptation.

It might be possible to train outputs for which the system has no data by determining correlations between weight changes. These correlations could be estimated as follows. For each speaker in a designated dataset, separately adapt to target words spoken by the individual. After retraining is complete for a word, save the changes in the weight values. Over all speakers, these values (for a given target word) can be used to estimate each weight's average change as a function of the other weights' changes using any standard function estimation technique. These functions (one per output category) can then be used when adapting to a new speaker. After retraining to a target word for the speaker, the functions for that word can be used to estimate appropriate changes for the other (non-target) outputs. Although this approach might be simple to implement and execute for a small-vocabulary task, there may be some difficulties in extending it to larger or more-general recognition tasks because of the number of functions to be estimated.

Even when using 50 speaker-dependent vectors per output category, these 50 vectors were obtained by duplicating the vectors generated from only one utterance. This typically resulted in a maximum of only 10 unique vectors per category and sometimes only 1 vector for some categories. To provide greater robustness, it would be reasonable to select the 50 vectors per category from multiple examples of the target word. In the incremental retraining approach, this could be accomplished by saving all examples of the target word heard so far, whether or not they were correctly recognized. When the time came to retrain, the system would then generate features from all of the words saved so far. Further retrainings would use not only a larger ratio of speaker-dependent to generic data, but a larger variety of speaker-dependent vectors as well.

Although the goal of the experiments was to improve the performance on only a single word, one might want to use the retraining approach to improve either several words that are consistently being misrecognized or even all of the words. Earlier (Section 7.1) we pointed out that this will definitely increase the time needed to retrain. However, it would be most interesting to see how performance on the first target word is affected by adding other words.

In Section 5.3.3 we pointed out a critical implicit assumption made in deciding to retrain only after errors. In the dataset used, there were either four or six examples of each target word in the adaptation set. This meant the system could only retrain three times if there were three errors within the four or six utterances. We implicitly estimated the overall target-word error rate based on a sample of size four (or six)! While this is appropriate if the system user cannot accept three errors that close together, there may be tasks where the errors are preferable to overlearning a specific word. A dataset with fewer targets but more examples of each target per speaker would allow one to investigate this issue.

Parameter optimization

On the **TIDIGITS** task, there is a clear gap between the baseline adult and adapted child error rates not accounted for by the adaptation. Two obvious suspects are the accuracy of our estimate of the optimal offset and the optimality of our warping function (the Bark shift) itself. As described in Section 6.2.1, other researchers have investigated various different warping functions. We did perform one preliminary experiment with *scaling* (rather than shifting) that displayed an interesting effect—although the performance was slightly worse than for the comparable offset experiment, only about half as many recognizer evaluations were required to determine the optimal scaling factor! Even if the performance is not quite as good, it might be better to use a scaling factor just for speed purposes. This is clearly an important issue, but it is likely that there are prosodic or other differences between the speech of adults and children as well. A careful study of the effects of different warping functions and different optimization functions would help to show what can and cannot be accomplished through a simple warping.

All of our experiments used a search ending tolerance of 0.01 for Brent's algorithm. This means that the algorithm would finish whenever the range spanned by the two outer values of the triplet was within ± 0.01 of the location of the middle point. For most speakers this required between 10 and 12 recognizer evaluations. This value was set at 0.01 because the preliminary examination of the variation in recognizer score for different offset values indicated that a resolution of 1.0 or even 0.1 Barks might not be sufficient. However, the later experiments on the **PhoneBook** corpus seemed to indicate that the optimal offset for a speaker was less sensitive. It is our expectation that a more careful study to determine the appropriate tolerance value would suggest a larger tolerance value, most likely requiring significantly fewer recognizer evaluations and thus less time for adaptation.

Our approach to optimizing a parameter is of course not limited to the specific parameter we chose or even the domain of speaker adaptation. The technique is applicable to any task for which there is a single value that must be set appropriately for a speaker, phone line, etc. using only a small amount of speech. As an example, one such parameter would be the J in J-RASTA. J-RASTA[42] is a modification of the RASTA algorithm designed to overcome the latter's difficulties in handling additive noise. It turns out that the optimal J factor for any utterance is dependent upon the level of noise in the utterance. In the paper decribing this algorithm [42], the authors explicitly estimated the noise in the signal and set the value of J accordingly. Instead, using our approach one would directly find the value that optimizes the overall recognizer score. This reduces the risk of errors that might otherwise have been introduced by the noise estimation algorithm.

One implicit assumption made in the use of the Bark offset parameter was that shifting *all* of the speech would be acceptable. The motivation for using this parameter, however, was that it would help to account for differences in vocal tract lengths; in particular, it would normalize formant frequencies across speakers. However, unvoiced speech (fricatives, silence, etc.) is in general less affected by vocal tract length. A more appropriate use of this parameter would be to compute it based only on voiced regions of speech and, likewise, only to apply the corresponding shift to voiced regions. It might even be that a simple energy measure would be sufficient to determine the applicability of the shift. Our iterated normalization experiments accounted for this somewhat by guaranteeing that the network would have seen unvoiced regions shifted by varied amounts, but it would have been better not even to have shifted these regions of speech.

As mentioned in Section 6.4.3, only supervised adaptation was attempted for the **PhoneBook** experiment. An identical experiment using only the recognizer score for the top-scoring word (i.e., unsupervised), though, would be worthwhile to run. Another issue brought up in our experiments which should be addressed more thoroughly is the optimization of an overall utterance score vs. just the word score when searching for the correct offset. In our preliminary experiments it appeared as if the overall utterance score was less appropriate for our adaptation. However, that may be due to the fact that the score includes silence or noise, which may have been adversely affected by our shift. A more careful comparison of these two optimization scores could easily provide important insights into the usability of the scores for adaptation or other purposes.

The adaptation experiments on connected digit strings were *unsupervised* in the sense that the system did not know what digits the person spoke in the adaptation utterance, although the system did know that it was a string of digits. In addition, only one utterance was used to set the optimal offset. In experiments with the **PhoneBook** corpus, on the other hand, the adaptation was supervised and used multiple utterances to set the offset value. It would be interesting to see what effects these changes would have on the digits corpus. An experiment using supervised adaptation might help to point out the cases where maximizing the overall recognizer score is not as beneficial as maximizing the score for the correct word, or vice versa. An analysis of the number of utterances needed to accurately estimate the optimal offset, on the other hand, might illustrate that this number is indicative of the variability in the task, with a larger number indicating more variability. Although it should be clear that there is more phonetic and lexical variability in **PhoneBook** than in **TIDIGITS**, it might be more difficult to judge between two isolated word tasks or two short-utterance tasks.

In establishing the optimal offset for a speaker, the (one or more) utterance(s) selected for the adaptation set were selected at random. It would be interesting to see whether incorrectly-recognized utterances provide more reliable information about the optimal offset value. If so, a better tack might be to adapt only on errors. This would have the added advantage of only requiring adaptation for those speakers whose performance is not already good; the others would be unaffected.

Combinations

In the above sections we have separately discussed ways to extend the work done on the rapid retraining and parameter optimization approaches, but we have not discussed the possibility of combining the two approaches. Given both a general speaker adaptation approach and one designed only to improve speaker performance for a single target word, it is reasonable to consider first performing general adaptation and then, if appropriate for the task, to adapt to a specific word.

Here is an example, for an isolated-word task, of how a supervised version of this process might work. As each word (any word, i.e., not just the target) by the speaker is recognized, the system uses Brent's method to compute the optimal Bark offset for the utterance. Once several (an empirically-determined number, 8 in the case of **PhoneBook**) words have been processed in this manner, the median of these offset values can be used to process all remaining words. Offsets computed for succeeding words could then be included into the computation of the median to increase the robustness of this estimate of the optimal offset for the speaker. Concurrently with determining and using new offset values, the system will check to see if any of these words are target words. Upon encountering the first target word that is misrecognized, the system will retrain the network using speaker-dependent vectors generated from the first utterance of the target word by the speaker as described in Section 5.3.3. The system will then use this new network to classify all the following utterances until it misrecognizes a second target-word, at which point it will retrain again and the process will repeat.

What is missing from the sample process above is the interactions between the two approaches. Here is a list of the events that would require special action:

Retraining the network Since the optimal Bark offset for each of the utterances implicitly depends upon the network weights, all of the offsets computed so far would need to be recomputed. The median offset would need to be recomputed as well.

Changing the offset The features generated for use in retraining were generated using a specific Bark offset value. If this value changes (e.g., the median offset is recomputed), the features would have to be regenerated for use in the next retraining.

The sample process described above is simplistic, leaving out issues which would need to be addressed through experimentation. One concern is whether or not the retraining progressions should be restarted after changing the offset value. After changing the median offset, it might be appropriate to again only use a small percentage of speaker-dependent data on the next error rather than increasing the percentage from the last retraining. Another subtle issue is that of the implicit normalization that will happen by retraining after each adjustment to the offset value. Over time, the weights from the hidden units to the target outputs of the network will gradually be trained on features computed using an offset estimate based on earlier weight values (which were computed using an earlier offset estimate, etc.). As the adult digit string normalization experiments demonstrated (Table 6.3), it is possible that too many iterations in an iterative normalization scheme can actually *reduce* recognition accuracy. Finally, there is one other issue that is extremely important in any practical implementation: speed. Notice that this combination system now needs to perform *both* adaptation procedures within the timespan of a single prompt. In addition, they might need to be performed after each utterance. Remember that the median offset determination by itself only requires the estimation of one new offset value for each new utterance. With this combined system, any time the system retrains a network it needs to recompute the offset values for *all* of the utterances seen so far (or used in the computation of the median). For the combined system to be practically useful, these increased computation issues must be solved.

In this closing chapter we have summarized this thesis research and suggested possibilities for future extensions. Alone or in combination, these methods appear promising and suggest that rapid adaptation for neural network recognition systems may be feasible today within appropriate domains.

Bibliography

[1] Abrash, V., Franco, H., Cohen, M., Morgan, N., and Konig, Y. Connectionist gender adaptation in a hybrid neural network / hidden Markov model speech recognition system. In *Proceedings of the 1992 International Conference on Spoken Language Processing, Banff, Alberta, Canada, Oct. 12–16* (1992), vol. 2, pp. 911–914.

[2] Andreou, A., Kamm, T., and Cohen, J. Experiments in vocal tract normalization. In *Proceedings of the CAIP Workshop: Frontiers in Speech Recognition II* (CAIP, P.O. Box 1390, CoRE Building, Rutgers University, Piscataway, NJ 08855, 1994), Center for Computer Aids for Industrial Productivity.

[3] Bellegarda, J. R., de Souza, P. V., Nádas, A. J., Nahamoo, D., Picheny, M. A., and Bahl, L. R. Robust speaker adaptation using a piecewise linear acoustic mapping. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, Mar. 23–26* (1992), vol. 1, pp. 445–448.

[4] Bladon, R. A. W., Henton, C. G., and Pickering, J. B. Towards an auditory theory of speaker normalization. *Language & Communication 4*, 1 (1984), 59–69.

[5] Bourlard, H., and Morgan, N. *Connectionist Speech Recognition—A Hybrid Approach.* Kluwer Academic Publishers, 1994.

[6] Bourlard, H., Morgan, N., and Renals, S. Neural nets and hidden Markov models: Review and generalizations. *Speech Communication 11*, Nos. 2 and 3 (June 1992), 237–246.

[7] Brent, R. P. *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[8] Chen, L. A global optimization algorithm for neural network training. In *Proceedings of the 1993 International Joint Conference on Neural Networks, Nagoya, Japan, Oct. 25–29* (Oct. 1993), vol. 1, pp. 443–446.

[9] Cole, R. A., Noel, M., Lander, T., and Durham, T. New telephone speech corpora at CSLU. In *Proceedings of the 4^{th} European Conference on Speech Communication and Technology, Madrid, Sep. 18–21* (Sept. 1995), vol. 1, European Speech Communication Association, pp. 821–824.

[10] Cole, R. A., Stern, R. M., Phillips, M. S., Brill, S. M., Pilant, A. P., and Specker, P. Feature-based speaker-independent recognition of isolated English letters. In *Proceedings of the 1983 IEEE International Conference on Acoustics, Speech, and Signal Processing* (1983), pp. 731–734.

[11] Cox, S. Predictive speaker adaptation in speech recognition. *Computer Speech and Language 9*, 1 (1995), 1–17.

[12] Digalakis, V., and Neumeyer, L. Speaker adaptation using combined transformation and Bayesian methods. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing, Detroit, Michigan, May 9–12* (1995), vol. 1, pp. 680–683.

[13] Digalakis, V., Rtischev, D., and Neumeyer, L. Fast speaker adaptation using constrained estimation of Gaussian mixtures. *IEEE Transactions on Speech and Audio Processing 3*, 5 (Sept. 1995), 357–366.

[14] Eide, E., and Gish, H. A parametric approach to vocal tract length normalization. In *Proceedings of the 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing, Atlanta, Georgia, May 7–10* (1996), vol. 1, pp. 346–349.

[15] Fourcin, A., Ainsworth, W., Fant, G., Fujimura, O., Fujisaki, H., Hess, W., Holmes, J., Itakura, F., Schroeder, M., and Struebe, H. Speech processing by man and machine: Group report. In *Recognition of Complex Acoustic Signals*, T. H. Bullock, Ed., no. 5 in Life Sciences Research Report. Abakon Verlag, 1977, pp. 307–351.

[16] Fukuzawa, K., Komori, Y., Sawai, H., and Sugiyama, M. A segment-based speaker adaptation neural network applied to continuous speech recognition. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, Mar. 23–26* (1992), vol. 1, pp. 433–436.

[17] Furui, S. A training procedure for isolated word recognition systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing 28* (1980), 129–136.

[18] G. David Forney, J. The Viterbi algorithm. *Proceedings of the IEEE 61*, 3 (Mar. 1973), 268–277.

[19] Gauvain, J.-L., and Lee, C.-H. Maximum *a Posteriori* estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Speech and Audio Processing 2*, 2 (1994), 291–298.

[20] Gong, Y., Siohan, O., and Haton, J.-P. Minimization of speech alignment error by iterative transformation for speaker adaptation. In *Proceedings of the 1992 International Conference on Spoken Language Processing, Banff, Alberta, Canada, Oct. 12–16* (1992), pp. 377–380.

[21] H. Bourlard, e. Wernicke esprit project 6487: 1993–1994 progress report. Obtained from author at the International Computer Science Institute, Berkeley, CA.

[22] Hermansky, H. Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America 87*, 4 (Apr. 1990), 1738–1752.

[23] Hermansky, H., Morgan, N., Bayya, A., and Kohn, P. Compensation for the effect of the communication channel in auditory-like analysis of speech (RASTA-PLP). In *Proceedings of the 2nd European Conference on Speech Communication and Technology, Genova* (1991), European Speech Communication Association, pp. 1367–1370.

[24] Huang, X. Speaker normalization for speech recognition. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, Mar. 23–26* (1992), vol. 1, pp. 465–468.

[25] II, J. B. H., and Waibel, A. H. The Meta-Pi network: Building distributed knowledge representations for robust pattern recognition. Tech. Rep. CMU-CS-89-166, School of Computer Science, Carnegie Mellon University, Aug. 1989.

[26] II, J. B. H., and Waibel, A. H. The META-PI network: Connectionist rapid adaptation for high-performance multi-speaker phoneme recognition. In *Proceedings of the 1990 IEEE International Conference on Acoustics, Speech, and Signal Processing, Albuquerque, New Mexico* (Apr. 1990), pp. 165–169.

[27] Kamm, T., Andreou, A. G., and Cohen, J. Vocal tract normalization in speech recognition: Compensating for systematic speaker variability. In *Proceedings of the 15th Annual Speech Research Symposium* (Johns Hopkins University, June 1995), pp. 175–178.

[28] Knohl, L., and Rinscheid, A. Speaker normalization and adaptation based on feature-map projection. In *Proceedings of the 3rd European Conference on Speech Communication and Technology, Berlin* (1993), pp. 367–370.

[29] KOBAYASHI, T., UCHIYAMA, Y., OSADA, J., AND SHIRAI, K. Speaker adaptive phoneme recognition based on feature mapping from spectral domain to probabilistic domain. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, Mar. 23–26* (1992), vol. 1, pp. 457–460.

[30] KONIG, Y., AND MORGAN, N. Supervised and unsupervised clustering of the speaker space for connectionist speech recognition. In *Proceedings of the 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing, Minneapolis, Minnesota, Apr. 27–30* (1993), vol. 1, pp. 545–548.

[31] KOSAKA, T., AND SAGAYAMA, S. Tree-structured speaker clustering for fast speaker adaptation. In *Proceedings of the 1994 IEEE International Conference on Acoustics, Speech, and Signal Processing, Adelaide, Australia, Apr. 19–22* (1994), vol. 1, pp. 245–248.

[32] KOSAKA, T., WILLEMS, E., TAKAMI, J.-I., AND SAGAYAMA, S. A dynamic approach to speaker adaptation of hidden Markov networks for speech recognition. In *Proceedings of the* 3^{rd} *European Conference on Speech Communication and Technology, Berlin* (1993), pp. 363–366.

[33] LANDER, T. The CSLU labeling guide. Tech. Rep. CSLU-014-96, Center for Spoken Language Understanding, Oregon Graduate Institute, June 1996.

[34] LEE, C. H., LIN, C. H., AND JUANG, B. H. A study on speaker adaptation of the parameters of continuous density hidden Markov models. *IEEE Transactions on Signal Processing 39* (1991), 806–814.

[35] LEE, L., AND ROSE, R. C. Speaker normalization using efficient frequency warping procedures. In *Proceedings of the 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing, Atlanta, Georgia, May 7–10* (1996), vol. 1, pp. 353–356.

[36] LEGGETTER, C. J., AND WOODLAND, P. C. Speaker adaptation of continuous density HMMs using linear regression. In *Proceedings of the 1994 International Conference on Spoken Language Processing, Yokohama, Japan, Sep. 18–22* (1994), vol. 2, pp. 451–454.

[37] LEGGETTER, C. J., AND WOODLAND, P. C. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language 9*, 2 (Apr. 1995), 171–185.

[38] LEONARD, R. G. A database for speaker-independent digit recognition. In *Proceedings of the 1984 IEEE International Conference on Acoustics, Speech, and Signal Processing* (1984), vol. 3, p. 42.11.

[39] LIPPMANN, R. P. An introduction to computing with neural nets. *IEEE SSSP Magazine* (Apr. 1987), 4–22.

[40] LJOLJE, A. Speaker clustering for improved speech recognition. In *Proceedings of the 3^{rd} European Conference on Speech Communication and Technology, Berlin* (1993), pp. 631–634.

[41] MATSUKOTO, H., AND INOUE, H. A piecewise linear spectral mapping for supervised speaker adaptation. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, Mar. 23–26* (1992), vol. 1, pp. 449–452.

[42] MORGAN, N., AND HERMANSKY, H. RASTA extensions: Robustness to additive and convolutional noise. In *Proceedings of the Workshop on Speech Processing in Adverse Environments, Cannes, France* (Nov. 1992). Obtained from Hermansky at the Oregon Graduate Institute, P.O. Box 91000, Portland, OR 97291.

[43] NEUBERG, E. P. Frequency-axis warping to improve automatic word recognition. In *Proceedings of the 1980 IEEE International Conference on Acoustics, Speech, and Signal Processing* (1980), pp. 166–168.

[44] NEUMEYER, L., SANKAR, A., AND DIGALAKIS, V. A comparative study of speaker adaptation techniques. In *Proceedings of the 4^{th} European Conference on Speech Communication and Technology, Madrid, Sep. 18–21* (1995), pp. 1127–1130.

[45] OHKURA, K., SUGIYAMA, M., AND SAGAYAMA, S. Speaker adaptation based on transfer vector field smoothing with continuous mixture density HMMs. In *Proceedings of the 1992 International Conference on Spoken Language Processing, Banff, Alberta, Canada, Oct. 12–16* (1992), vol. 1, pp. 369–372.

[46] PALLETT, D., FISCUS, J., AND GAROFOLO, J. DARPA resource management benchmark test results June 1990. In *DARPA Speech and Language Workshop*. Morgan Kaufmann, 1990, pp. 298–305.

[47] PEDREIRA, C. E., AND ROEHL, N. M. On adaptively trained neural networks. In *Proceedings of the 1993 International Joint Conference on Neural Networks, Nagoya, Japan, Oct. 25–29* (Oct. 1993), vol. 1, pp. 565–568.

[48] Pitrelli, J. F., Fong, C., Wong, S. H., Spitz, J. R., and Leung, H. C. Phonebook: A phonetically-rich isolated-word telephone-speech database. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing, Detroit, Michigan, May 9–12* (1995), vol. 1, pp. 101–104.

[49] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes in C: The Art of Scientific Computing*, second ed. Cambridge University Press, 1994.

[50] Robins, A. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science 7*, 1 (1995), 123–146.

[51] Rumelhart, D., Hinton, H., and Williams, R. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. Rumelhart and J. McClelland, Eds., vol. I. The MIT Press, Cambridge, 1986.

[52] Scharf, B. Critical bands. In *Foundations of Modern Auditory Theory*, J. V. Tobias, Ed., vol. I. Academic Press, 1970, ch. 5.

[53] Schmidbauer, O., and Tebelskis, J. An LVQ based reference model for speaker-adaptive speech recognition. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, Mar. 23–26* (1992), vol. 1, pp. 441–444.

[54] Simpson, P. K. *Artificial Neural Systems.* Pergamon Press, New York, 1990.

[55] Stern, R. M., and Lasry, M. J. Dynamic speaker adaptation for feature-based isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing 35*, 6 (June 1987), 751–763.

[56] Wakita, H. Normalization of vowels by vocal-tract length and its application to vowel identification. *IEEE Transactions on Acoustics, Speech, and Signal Processing 25*, 2 (Apr. 1977), 183–192.

[57] Watrous, R. L. Speaker normalization and adaptation using second-order connectionist networks. *IEEE Transactions on Neural Networks 4*, 1 (Jan. 1993), 21–30.

[58] Wegmann, S., McAllaster, D., Orloff, J., and Peskin, B. Speaker normalization on conversational telephone speech. In *Proceedings of the 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing, Atlanta, Georgia, May 7–10* (1996), vol. 1, pp. 339–341.

[59] Witbrock, M., and Haffner, P. Rapid connectionist speaker adaptation. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, Mar. 23–26* (1992), vol. 1, pp. 453–456.

[60] Zavaliagkos, G. *Maximum A Posteriori Adaptation Techniques for Speech Recognition.* PhD thesis, Northeastern University, Boston, Massachusetts, May 1995.

[61] Zavaliagkos, G., Schwartz, R., and Makhoul, J. Batch, incremental and instantaneous adaptation techniques for speech recognition. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing, Detroit, Michigan, May 9–12* (1995), vol. 1, pp. 676–679.

[62] Zavaliagkos, G., Schwartz, R., McDonough, J., and Makhoul, J. Adaptation algorithms for large scale HMM recognizers. In *Proceedings of the* 4^{th} *European Conference on Speech Communication and Technology, Madrid, Sep. 18–21* (1995), pp. 1131–1134.

[63] Zhao, Y. Self-learning speaker adaptation based on spectral variation source decomposition. In *Proceedings of the* 3^{rd} *European Conference on Speech Communication and Technology, Berlin* (1993), pp. 359–362.

[64] Zhao, Y. An acoustic-phonetic-based speaker adaptation technique for improving speaker-independent continuous speech recognition. *IEEE Transactions on Speech and Audio Processing 2*, 3 (July 1994), 380–394.

[65] Zwicker, E. Subdivision of the audible frequency range into critical bands (frequenzgruppen). *Journal of the Acoustical Society of America 33*, 2 (1961), 248.

Appendix A

Huh? call list and protocol

This appendix presents both a sample call list and our collection protocol/prompts for the **Huh?** corpus data collection described in Section 4.4.

A.1 Sample call list

```
Date: Tue, 19 Mar 96 14:26 PST
From: nobody@cse.ogi.edu (uid no body)
To: burnett@cse.ogi.edu
Subject: Speaker Adaptation data collection instructions

To participate in this data collection call the following toll-free
telephone number and follow the instructions.  You will be asked to
tell us your ID number, so please locate it below.  You will also be
asked to say each word in the following three sections so it is
important that you have this list with you when you call.

Once again, thank you for helping us with our research.

The number to call is 1-800-123-4567 (native)

Your ID number is: 1787

                              *** Word List ***
                              -----------------
```

```
Section A:
---------

1.  last                    41.  next
2.  next                    42.  east
3.  yes                     43.  east
4.  last                    44.  yes
5.  last                    45.  nest

6.  east                    46.  next
7.  less                    47.  east
8.  less                    48.  next
9.  nest                    49.  yes
10.  last                   50.  east

.
.
.


Section B:
---------

1.  last                    36.  no
2.  east                    37.  no
3.  less                    38.  hope
4.  nest                    39.  yo
5.  nest                    40.  no

.
.
.

Section C:
---------
```

```
1.  no                         31.  help
2.  no                         32.  help
3.  help                       33.  hope
4.  yo                         34.  help
5.  no                         35.  help

.
.
.
```

If you have any questions or comments, feel free to contact Mike Noel at:

noel@cse.ogi.edu
(012) 345-6789

A.2 Collection protocol

Thank you for calling the Center for Spoken Language Understanding data collection system. For this data collection we will need you to read from the list of words you received. If you do not have your prompt sheet, please hang up now and call back later. After you have finished all the prompts from your sheet, please leave your name and the address to which you would like your gift certificate sent. This call will take about 15 minutes.

This data collection is for native (non-native) speakers of American English only. If you are not (are) a native speaker, please hang up now.

At the beep, please say the 4-digit I.D. number given on your instruction sheet.

We will now ask you to say the words in section A. You will be prompted for each word. Please say the word after the beep.

Say word 1.

Say word 2.

Say word 3.

.

.

.

Say word 48.

Say word 49.

Say word 50.

You have finished section A. We will now ask you to say the words in section B. You will be prompted for each word. Please say the word after the beep.

Say word 1.

.

.

.

Say word 40.

You have finished section B. We will now ask you to say the words in the last section, section C. You will be prompted for each word. Please say the word after the beep.

Say word 1.

.

.

.

Say word 35.

You have completed the data collection. At the beep, please leave your name and the address to which we should send your gift certificate. Thank you.

Biographical Note

Daniel Burnett was born in Phoenix, Arizona on November 29th, 1968. A National Merit Scholar, he attended Hermann Hesse Gymnasium in Calw, West Germany as an exchange student for a year before entering Harvey Mudd College in the fall of 1986. He received the Bachelor of Science degree in Mathematics in May, 1990.

Daniel's interests, both research and personal, have arisen from a desire to enhance our daily lives by appropriate application of computer technology. Personally, this desire has manifested itself in attempts to demonstrate to others ways in which the computer can be used to automate tedious and repetitive tasks. Professionally, his focus has been on methods of bringing human-like capabilities to computers, with research interests in artificial intelligence, neural networks, and most recently automatic speech recognition.

Daniel has authored or co-authored the following papers:

D. C. Burnett and M. Fanty. Rapid unsupervised adaptation to children's speech on a connected-digit task. In *Proceedings of the 1996 International Conference on Spoken Language Processing, Philadelphia, Pennsylvania*, volume 2, pages 1145–1148, 1996.

R. A. Cole, D. Novick, P.J.E. Vermeulen, S. Sutton, M. Fanty, L. Wessels, J. de Villiers, J. Schalkwyk, B. Hansen, and D. Burnett. Experiments with a spoken dialogue system for taking the U.S. Census. *International Journal of Human-Computer Interaction*, 1995.

R. A. Cole, D. G. Novick, M. Fanty, P. Vermeulen, S. Sutton, D. Burnett, and J. Schalkwyk. A prototype voice-response questionnaire for the U.S. Census. In *Proceedings of the 1994 International Conference on Spoken Language Processing, Yokohama, Japan, Sep. 18–22*, pages 683–686, 1994.

R. A. Cole, M. Noel, D. C. Burnett, M. Fanty, T. Lander, B. Oshika, and S. Sutton. Corpus development activities at the Center for Spoken Language Understanding. In *Proceedings of the ARPA Workshop on Human Language Technology, Plainsboro, NJ,*

Mar. 8–11, pages 31–36, 1994.

R. A. Cole, D. G. Novick, D. Burnett, B. Hansen, S. Sutton, and M. Fanty. Towards automatic collection of the U.S. Census. In *Proceedings of the 1994 IEEE International Conference on Acoustics, Speech, and Signal Processing, Adelaide, Australia, Apr. 19–22*, pages 93–96, 1994.

R. A. Cole, D. Burnett, and V. Weatherill. An evaluation guide for emergent technologies in automatic speech recognition. Technical Report CSLU-005, Center for Spoken Language Understanding, Oregon Graduate Institute, December 6, 1993.

R. A. Cole, D. G. Novick, M. Fanty, S. Sutton, B. Hansen, and D. Burnett. Rapid prototyping of spoken language systems: the Year 2000 Census Project. In *Proceedings of the International Symposium on Spoken Dialogue, Tokyo, Japan, Nov. 10–12*, pages 19–23, 1993.