BOSTON UNIVERSITY
COLLEGE OF ENGINEERING

Thesis

REDUCING THE EFFECTS OF LINEAR CHANNEL
DISTORTION ON CONTINUOUS SPEECH RECOGNITION

by

REBECCA ANNE BATES

M.T.S., Harvard Divinity School, Harvard University, 1993
B.S. Biomedical Engineering, Boston University, 1990

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science
1996
Approved by

First Reader
Dr. Mari Ostendorf, Associate Professor,
Department of Electrical, Computer and Systems Engineering
Boston University

Second Reader
Dr. J. Robin Rohlicek, Vice President, Engineering,
PureSpeech
Research Associate,
Department of Electrical, Computer and Systems Engineering
Boston University

Third Reader
Dr. William C. Karl, Assistant Professor,
Department of Electrical, Computer and Systems Engineering
Boston University
Acknowledgments

It takes a village to raise a child, it took a village to get this thesis done and written. Apologies for the length of the acknowledgments.

Primary acknowledgments must go to Mari Ostendorf, my advisor and friend, who has been a wonderful source of advice, support, and encouragement. I must point out that this work, both the research and the writing, would not have been possible without her considerable input. If I mentioned all she has done for me, we’d never get to the thesis. I would like to thank my readers, Robin Rohlicek and Clem Karl for their patience with my eternal thesis as well as their insightful comments. Thanks also to Chin-Hui Lee and Ananth Sankar for their encouragement and suggestions.

For technical support, I must thank Yogen Patel and BBN who helped get the work started by providing initial waveforms as well as assistance with experimental set up. Mucho grande thanks to the Cambridge-based HTK crew. Without help and advice from everyone in the support loop, this work would never have been finished. I know I will never be able to repay Julian Odell for all of his help. My frantic, tearing-out-my-hair email messages were always answered quickly and well. Julian, the next few ciders are on me.

I would like to thank Jordan Cohen for enjoyable conversations as well as for his help with resources. Thanks to Jordan and the CAIP Center at Rutgers University, a large set of experiments was quickly finished with the help of CAIP Center computers and disk space. I am also appreciative of the trip to Austin, TX for the 1995 SLT workshop as well as the opportunity to edit the conference proceedings.

Thanks especially for the technical and emotional support from SPIlab people: Cam Fordyce, Rukmini Iyer, Ashvin Kannan, Orith Ronen and Nanette Veilleux. Proofreading at various stages and conversations that ranged from chocolate to compiling errors were necessary and appreciated. Thanks to John Kaufhold for putting together much of the initial work on this project. Special thanks also to Mary Hendrix
for being a wonderful officemate and a continuing source of girl scout cookies, recipes and conversation. Also thanks to SPIlab legends Fred Richardson and Owen Kimball for all that I was able to learn from them about programming.

Thanks to John Cooney for patient listening, all kinds of advice, sanity checks, and any Red Sox statistic I could ever want to hear. I would also like to thank my dear friend Lori Harris for providing wonderful breaks at her home in Alabama each year. The Gang deserves thanks as well for helping me keep one foot in a very pleasant reality. Thanks also to the Doucette family for everything from football games to chicken soup.

My family deserves special thanks for putting up with the one who “may never return to Montana”. Sarah knows I will, if only because I miss the couch. My brothers have kept me in touch with home by sharing their motorcycle-riding, mountain-trekking, rock-a-billy lives with me, collect of course. Saving the best for last, I am eternally grateful to my dad for all of the support and love he has showered me with over the past 10 years away from home.

This research was supported by the NSF under contract number IRI-9408896 in association with ARPA/ONR N00014-92-J-1778.
REDUCING THE EFFECTS OF LINEAR CHANNEL DISTORTION
ON CONTINUOUS SPEECH RECOGNITION

(Order No. )

REBECCA ANNE BATES

Boston University, College of Engineering, 1996
Major Professor: Mari Ostendorf, Associate Professor,
Electrical, Computer and Systems Engineering

Abstract

Speech recognition over telephone channels introduces challenges not present when speech is recorded using a known, high quality microphone. The goal of this work is to mitigate some of the effects of the telephone channel which reduce accuracy relative to the high quality situation. For practical purposes, a telephone channel is often characterized as an unknown linear, time-invariant system. Given an acoustic model of speech unaffected by channel distortion, it is possible to modify either the model or the input speech based on an estimate of the channel to reduce mismatch in training and test conditions. This work focuses on channel compensation and extends previous results through the introduction of a prior distribution of the channel. The main contributions include: 1) finding a prior distribution for the telephone channel while “cleaning” training data recorded over various telephone lines, 2) assessing the usefulness of maximum likelihood and maximum a posteriori channel estimates, implemented in the signal space, and 3) assessing the usefulness of Bayesian learning to modify the model means and covariances for channel compensation in comparison to maximum likelihood estimation of the model transformation parameters. A multi-pass recognition strategy is used, where the channel estimate is based on the first pass of recognition. The various methods are evaluated using the Macrophone Natural Numbers corpus. The best results are obtained with methods using a prior channel distribution with an acoustic model trained on “cleaned” data; reductions in word error rate of up to 15% over cepstral mean subtraction are shown. The model space transformations give only a small improvement over signal space modification.


## Contents

1 **Introduction** 1

2 **Background Work in Channel Compensation** 7

2.1 Basic Recognition System 8

2.2 Acoustic Pre-processing Methods for Channel Compensation 11

2.2.1 Band-liming of Data 11

2.2.2 Post-Processing of Cepstral Features 12

2.3 Model Modification 18

2.3.1 Parallel Model Combination (PMC) 19

2.3.2 Stochastic Matching of Models 20

2.4 Limitations of Existing Techniques 21

3 **Baseline Work** 23

3.1 The Microphone Corpus 23

3.2 HTK, a Hidden Markov Model Toolkit 25

3.2.1 Hidden Markov Models 27

3.2.2 System Set Up 29

3.2.3 Model Training 32
3.2.4 Recognition and Analysis .......................... 36
3.3 Baseline Results ........................................ 37
   3.3.1 Model development .............................. 38
   3.3.2 Results on the Development Test Set and Subset ........ 39
   3.3.3 Gender-dependent Results ....................... 41
   3.3.4 Summary ......................................... 42

4 Adaptation in Feature Space ......................... 43
   4.1 Channel Estimation Theory ....................... 43
      4.1.1 ML Estimates ................................ 45
      4.1.2 MAP Estimates ................................ 47
   4.2 Implementation .................................... 49
   4.3 Channel Prior Distribution Estimation ............ 51
   4.4 Experimental Results and Discussion ............ 61
   4.5 Summary ......................................... 65

5 Transformations in the Model Space ................. 66
   5.1 Model Modification Theory ....................... 67
      5.1.1 Stochastic Matching ......................... 67
      5.1.2 Bayesian Learning ............................ 69
   5.2 Implementation .................................... 72
   5.3 Experimental Results and Discussion ............ 73
   5.4 Comparison of Feature-based and Model-based Techniques .... 78

6 Conclusion ............................................. 79
   6.1 Summary ......................................... 79
6.2 Suggestions for Future Directions .......................... 81

A Test Subset Lexicon and Grammar 83
A.1 Lexicon ......................................................... 83
A.2 Finite State Grammar ................................. 83
List of Tables

2.1 Percentage word error for comparison experiments using SRI simultaneous telephone and high quality Sennheiser microphone recordings (from [26]). .................................................. 12

2.2 Comparison of RASTA and CMS on the verbalized pronunciation (VP) data set with a 5k vocabulary. Models were trained on the WSJ0 training set. .................................................. 14

2.3 Comparison of CMS and channel estimation on two different data sets. .................................................. 16

2.4 Stochastic matching percentage word error results on telephone speech for two speakers (from [35,36]). .................................................. 21

3.1 Signal processing parameters used for coding speech. .................. 33

3.2 Tuning the insertion penalty: results for a portion of the full test set. 39

3.3 Results for full test set using full Macrophone training data on a gender-independent model. .................................................. 40

3.4 Results for full test set and subset using 7 mixture, gender-independent model with different language models. ............................. 41

3.5 Results for test subset using the finite state network and open and close vocabulary bigram language models. ............................. 41

3.6 Results for gender-dependent 6-mixture models using full Macrophone training data. .................................................. 42
4.1 \( R^2 \) values when predicting channel error components with all other channel error components. ........................................ 60

4.2 Channel error prior distribution parameters, assuming a Gaussian distribution with a diagonal covariance. ................................. 61

4.3 CMS, ML and MAP channel estimation results on telephone speech, using the full test set and the open-vocabulary bigram language model. 63

4.4 CMS, ML and MAP channel estimation results on telephone speech, using the full test set and the closed-vocabulary bigram language model. 63

4.5 MAP and weighted MAP channel estimation results on telephone speech, using the cleaned acoustic model and the open-vocabulary bigram language model. ........................................ 63

4.6 CMS, ML and MAP channel estimation results on telephone speech, using the test subset and three types of language models. ........... 64

5.1 Bayesian learning results on telephone speech, using the test subset, two types of language models, and a cleaned acoustic model with associated feature space transformation. The CMS baseline is 90.36% for finite state LM and 86.92% for open-vocabulary bigram. .... 75

5.2 Bayesian learning results on telephone speech, using the full test set, the open-vocabulary bigram language model, and cleaned acoustic model. Associated feature space transformations are given to show the effects of the covariance transform. The CMS baseline is 84.06%. .......................... 76

5.3 Stochastic Matching results on telephone speech, using the test subset, two types of language models, and a cleaned acoustic model with associated feature space transformation for reference. The CMS baseline is 90.36% for finite state LM and 86.92% for open-vocabulary bigram. .......................... 77
5.4 Stochastic Matching results on telephone speech, using the full test set, the open-vocabulary bigram language model, and cleaned acoustic model. Associated feature space transformations are given to show the effects of the covariance transform. The CMS baseline is 84.06%. . . . 77


**List of Figures**

2.1 Basic speech recognition system.  

2.2 Generation of feature vectors from processed speech.  

3.1 A histogram of SNRs for Macrophone natural numbers (from [20]).  

3.2 HTK software structure (from [41]).  

3.3 HTK overall structure (from [41]).  

3.4 A state diagram of a 3 state hidden Markov model.  

3.5 Sample dictionary format, including alternate pronunciations and optional output symbols.  

3.6 Sample label files:  

   a.) individual file with both monophone and word labels,  

   b.) individual file with alternate word labels,  

   c.) master label with recognizer output including words and boundary times.  

4.1 Block diagram for recognition with channel estimation.  

4.2 Distribution plots of the means of the first component of the cepstral feature vector.  

4.3 Distribution plots of the first component of the ML channel error estimated vectors.  

4.4 Histograms of the ML channel estimates (cepstra 1-6) from the training data.
4.5 Histograms of the ML channel estimates (cepstra 7-12) from the training data. .................................................. 57
4.6 Quantiles of standard normal plots of four of the ML channel error estimate components. .......................... 58
4.7 Scatterplot of four of the ML channel error estimate components. ... 59
Chapter 1

Introduction

Taking as a given that human communication is important, it is also important that humans be able to communicate with machines. Humans communicate with many machines by learning the languages of those machines. We have remote controls for televisions, steering wheels and gas pedals for cars, and keyboards for computers. If we get the machines to understand human language, we would be able to communicate with them more easily. For machines such as televisions, remote controls and knobs work fine, but for machines like computers, where keyboards and typing are our main means of communication and where the types of interaction can be much more complicated, it makes sense to build a more natural human interface. Since most people can speak but some can neither read nor write, a spoken language interface would make communicating with computers open to far more people, especially people with arthritis or RSI (Repetitive Stress Injuries), the blind, and young children. Examples of current speech recognition by computers include maneuvering through an operating system ("open directory", "get file"), getting account information from credit cards over the telephone, and doing telephone surveys. In these examples, a human listening to the speech would have no difficulty understanding speech spoken over the telephone. Unlike humans, however, the computer is sensitive to the fact that
the recording environment is different. This thesis will focus on improving recognition of speech that is recorded in a particular non-ideal environment, that of telephone lines.

Speaker-independent continuous speech recognition systems for read speech are currently at a state where word recognition results are quite good, meaning around 90% accuracy on unlimited vocabularies and for high quality recordings [30]. Unfortunately, the results are not as good when the speech being recognized is recorded from a telephone line, which is where most applications of speech recognition systems are currently being fielded. This is due to both additive noise and channel distortion from the telephone handsets and lines. In order to improve recognition results on telephone speech, recognition systems need to be robust to this noise and distortion. Specific difficulties with telephone speech in particular include: a narrowing of typical bandwidth from 100-8000Hz to 300-3400Hz, low frequency tones added at approximately 180Hz, additive stationary noise, impulse noise, channel distortion, which can mean an unknown but fixed channel or non-stationary distortion such as amplitude and phase jitter. This thesis involves modifying an existing speech recognition system to use explicit channel modeling in order to better recognize speech which has passed through a telephone channel. On a broader scale, this thesis involves determining an unknown channel from an unknown signal. Research in channel modeling will also benefit such areas as speaker identification, where dealing with the effects of an unknown channel is considered the “most important technical challenge for applications” [12].

Continuous speech recognition systems rely on acoustic models of speech which require large amounts of training data. The best recognition results typically occur when the recording environment of the training data matches the recording environment of the recognition data. Since telephone speech varies depending on the distance of the call, the particular handset, and the type of phone lines as well as other sources of noise, it is impossible to have enough training data in any one environment to match all possible recognition cases. Compensating for individual channels by
explicitly estimating them in both the data used for training acoustic models and the speech to be recognized will decrease the mismatch between training and recognition environments. Because there is a lot of available data that has been recorded over long-distance and local phone lines, it will be used here to allow acoustic models to be robust for some of the variability in telephone speech.

Techniques for dealing with known distortion are relatively straightforward. Handling unknown channel and noise distortion is more difficult. Most techniques make assumptions that the telephone is a linear time-invariant channel, either with no additive noise or with stationary, independent noise. In other words, when a given set of true speech \( x(n) \) has passed through a linear, time-invariant channel \( h(n) \) and is combined with stationary additive noise \( n_a(n) \), the speech observations, \( y(n) \), can be represented by the following equation:

\[
y(n) = x(n) \ast h(n) + n_a(n)
\]  

(1.1)

where \( \ast \) indicates convolution. If \( h(n) \) and \( n_a(n) \) were known, then this problem could be solved with standard statistical signal processing techniques. However, they are generally not known and the problem is complicated by the fact that \( x(n) \) is not stationary. If \( h(n) \) and \( n_a(n) \) were fixed but unknown, parallel recordings of speech in both a noisy and high-quality environment would allow models of the channel and noise to be estimated from observed clean versus distorted differences. Although parallel recordings can be helpful when the environment is fixed, such as recordings made over a particular microphone, they are not appropriate for unpredictable environments such as telephone lines. For example, different telephone handsets may be connected to the same phone line. All calls from this line would then have different characteristics depending on the handset used. For long distance calls, the same route may not be used for all calls between two numbers. Although the most difficult possible task would be working with a non-linear or time-varying channel as well as additive noise as in Equation 1.1, this work focuses on a \textit{fixed} but unknown \textit{linear}
channel without additive noise where

\[ y(n) = x(n) * h(n) \]  \hspace{1cm} (1.2)

because the problem of channel compensation alone has not yet been solved and
because the channel appears to be a more important factor than noise for telephone
speech, since telephone signal-to-noise ratios are typically greater than 20dB.

Traditionally, channel compensation is performed during the signal processing
of speech signals. During signal processing, speech signals are parameterized into
feature vectors that can be easily used to create acoustic models of speech. Typical
usage of recognition systems involves using models trained on data recorded using
high quality microphones in noise-controlled environments. These models are not
necessarily the best models to use for speech that has been recorded over telephone
lines or even low quality microphones. The current method for dealing with channel
distortion is to remove the mean of the feature vectors for an utterance. Theoretically,
a better statistical estimate of the channel is the maximum likelihood (ML) estimate
[26, 35, 5], which has already been implemented at other sites working on the problem
of telephone channel distortion, but it has not shown improved performance over
simpler mean normalization approaches using test sets with long utterances. However,
the negative results might be explained by a couple of observations: 1) since the
feature mean only converges to the channel estimate for long utterances [21], short
utterances, such as digits or "yes" or "no", may be better recognized when a more
exact channel estimate is used, and 2) the acoustic model of speech is based on
data that is processed differently from the recognized speech. In addition, we know
that telephone channels are not without some constraints, and so we may be able to
improve on ML estimation by using prior information about telephone channels.

The goal of channel compensation is to improve the match between the char-
acteristics of speech used for acoustic models and those of recognized speech. The
limitations of previous work in ML channel compensation raises several questions that
this work addresses.

1. Telephone channels have constraints, and these constraints can be expressed in terms of a prior distribution. Does the use of a prior channel distribution, as in a maximum \textit{a posteriori} (MAP) estimate, improve recognition performance?

2. Given that there are negative results for long utterances, do statistical estimation techniques (ML and/or MAP) out-perform feature mean subtraction in short utterances? For short utterances, the channel estimate has higher variance and mean removal could remove some speech information.

3. Does “cleaning” the training data, or estimating the channel and subtracting it from the training observations, improve the performance for either the ML or MAP method? If each utterance in the training data has the ML channel estimate removed, the resulting model should be closer to an acoustic model from utterances recorded in an ideal environment.

4. Is there an advantage to modifying the acoustic models rather than simply subtracting a constant vector from the features? Modifying the acoustic models accounts for changes in the variance of the observed acoustic features due to the added random channel distortion.

This thesis begins with a description of current approaches to the problem of recognizing telephone speech, in particular approaches for statistical channel estimation that do not involve parallel speech recordings, as well as an overview of general speech recognition systems in Chapter 2. A discussion of baseline speech recognition results, including a description of the speech corpus and recognition system used here, is in Chapter 3. Chapter 4 describes the feature-based adaptation methods and experiments along with the channel prior distribution estimation for use in the MAP channel estimate and Bayesian learning technique. Chapter 5 discusses model-based
adaptation methods and experiments. The thesis ends with conclusions and future directions in Chapter 6.
Chapter 2

Background Work in Channel Compensation

In this chapter, after a brief description of a basic recognition system, current approaches to recognition of telephone speech will be described. The approaches are divided into three aspects according to where a continuous speech recognition (CSR) system is impacted: 1) filtering of input data, 2) cepstral feature transformations, including cepstral mean removal techniques and other feature post-processing techniques (feature space transformations), and 3) model modification techniques (model space transformations). The first two areas affect the system during the pre-processing stage, while the third affects the system during the recognition stage. Techniques that require parallel recordings will not be examined in this work because parallel recordings are difficult to obtain for a wide variety of telephone channels. Although additive noise will not be addressed in this thesis, some methods presented in this section will compensate for both channel distortion and additive noise.
2.1 Basic Recognition System

The major components of a speech recognition system are signal processing and pattern recognition as shown in Figure 2.1. Pattern recognition involves finding the highest scoring word string, \( W \), for input features \( \hat{y}(n) \), the output of signal processing. The system recognizes words which are composed of phonemes, the basic unit of speech. Most CSR systems are statistical and use both an acoustic model and a language model. Labeled training speech is used to develop acoustic and language models. This section will first describe signal processing then basic models used in pattern recognition.

The signal processing component of a recognition system transforms speech into feature space vectors which are then used in training and recognition. Figure 2.2 illustrates the parameterization of a speech waveform into feature vectors. For each frame period, a windowed segment of speech is processed to obtain a vector of points. Cepstral analysis [27, 28] is the most commonly used signal processing procedure for speech recognition. Cepstral analysis is a simple means of deconvolving the excitation information and the vocal tract shape information, retaining only the more important vocal tract information. One implementation is to take the Fourier transform of the observations, \( y(n) \), the log of the transformed observations, the inverse Fourier
transform, and then a windowing operation to give $\hat{y}(n)$.\footnote{Other implementations use a filter bank rather than the first Fourier transform step, or linear prediction (LP) analysis instead of the windowing step to remove excitation information.}

In speech recognition, a given utterance is generally compared to acoustic models of speech. The models are estimated using a large amount of training data. It is important to have enough training data in order to model speaker variability and to create robust models. In general, probabilistic models are used for the acoustic models of sub-word units which are concatenated to represent $p(x|W)$ where $x$ is the observed clean training speech and $W$ is the associated text. A common type of model is the hidden Markov model (HMM) \cite{3, 32}, although segment based models are becoming more common \cite{29}. This work will use hidden Markov models because of the availability of a commercial HMM tool kit, HTK. Training acoustic models on speech recorded in the same environment in which test speech is recorded generally yields the best results, however it is not always possible to get enough training data in the test environment to make this practical.

Acoustic model modification can be done to avoid training new models and when there is a scarcity of data in the test environment. Basically, model modification
allows information about the interfering additive noise and channel distortion to be incorporated into the parameters of a clean speech model. Model modification makes possible the use of clean training speech, or training data from multiple environments, to develop a robust acoustic model.

**Language models** represent word strings $p(W)$ instead of phonemes. A typical language model is a Markov model where the probability of a word is found given the $n$ previous words, which is useful for large vocabulary tasks. For small vocabulary tasks, such as digits, a finite state network which maps out specific probabilities for the language can be used. An ideal model would take into account all previous words. Since this is rarely possible given the necessary constraints on computer storage and recognition search costs, bigram and trigram models are typically used. A bigram model represents the probability of a word string, $w_1, w_2, \ldots, w_T$, using the probability of a word given the one previous word,

$$P(w_1, w_2, \ldots, w_T) = P(w_1) \prod_{i=2}^{T} P(w_i|w_{i-1}),$$

and a trigram model conditions the probability of a word on the two previous words. Training of the language model is done using the text of the training data to compute the necessary probabilities.

**Pattern recognition search** involves finding the maximum likelihood word string for an utterance given the acoustic and language models, $W = \max_{W} p(x|W)p(W)$. The final hypothesized word string for a set of observations is the word string with the maximum probability given the acoustic model and the language model, where the acoustic model describes the distribution of the features and the language model describes the probability of a word given the word or words previous to it. Various efficient search methods, such as the Viterbi algorithm, can be used in this task to search through a network of words, which themselves are networks of phonemes, to find the most probable sequence. One way to reduce the search cost is to use a multi-pass search, where the first pass narrows the space and subsequent passes incorporate
more information such as more detailed models. Because of the size of the search space for large vocabulary recognition, various fast search algorithms are used in the pattern recognition step.

2.2 Acoustic Pre-processing Methods for Channel Compensation

This section examines current approaches to compensate for channel distortion in the feature space of the observations. These can generally be used for both training and testing speech data sets.

2.2.1 Band-limiting of Data

Incoming data may be band-limited at both the training and testing stages of recognition. At the training stage, band-limiting clean (non-noisy) speech during the signal processing step approximates the band-limiting effects of the telephone channel. Because it is easier to collect clean training speech than telephone speech, band-limiting makes it possible to build a relatively good acoustic model for telephone speech using training data that does not match the test data. Researchers at SRI [26, 40] and BBN [2] have shown that training on band-limited data improves recognition performance on telephone data because the data represents some of the effects of the telephone channel, at least for local lines. The results in Table 2.1 from SRI on parallel recordings² [26] show that band-limited clean speech gives better results when used for training data than telephone speech given the same amount of clean speech and telephone speech. During the recognition stage, it is also useful to band-limit

²The parallel recordings consisted of 3000 sentences for training and 400 sentences for testing of ATIS (Air Travel Information System) sentences using 10 different telephone handsets and recorded over local telephone lines by 13 male speakers.
Table 2.1: Percentage word error for comparison experiments using SRI simultaneous telephone and high quality Sennheiser microphone recordings (from [26]).

<table>
<thead>
<tr>
<th>Acoustic Model</th>
<th>% Word Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training Data</td>
</tr>
<tr>
<td>Sennheiser</td>
<td>7.8</td>
</tr>
<tr>
<td>Sennheiser</td>
<td>9.0</td>
</tr>
<tr>
<td>Telephone</td>
<td>10.0</td>
</tr>
</tbody>
</table>

the telephone test data to remove the low frequency tones that are added by the telephone line, leaving only the band-limited speech information.

2.2.2 Post-Processing of Cepstral Features

Telephone speech, $y(n)$, can be modeled as the convolution of speech, $x(n)$, and a telephone channel, $h(n)$, which is a simplification of (1) that assumes no noise. In this case, the Fourier transform of $y(n)$ is taken to give

$$ Y(f) = \mathcal{F}[y(n)] = X(f)H(f), $$

and then the log of the magnitude transform is computed

$$ \ln |Y(f)| = \ln |X(f)| + \ln |H(f)|. $$

After the inverse Fourier transform is performed, the cepstral observation, $\tilde{y}(n) = \mathcal{F}^{-1}[\ln |Y(f)|]$, is simply a linear combination of the speech $\tilde{x}(n) = \mathcal{F}^{-1}[\ln |X(f)|]$, and the channel information, $\tilde{h}(n) = \mathcal{F}^{-1}[\ln |H(f)|]$:

$$ \tilde{y}(n) = \tilde{x}(n) + \tilde{h}(n), $$

The stationary channel $\tilde{h}(n)$ can be subtracted out after it is estimated from the observed speech. The additive term $\tilde{h}(n)$ is sometimes referred to as convolutional
noise as well as channel distortion.

The following techniques are all performed in the cepstral domain, before the recognition search. The techniques can be grouped in terms of the problems they address: 1) channel distortion alone, 2) additive noise alone, and 3) both channel distortion and noise. Since this thesis will focus on channel modeling, additive noise modeling alone will not be reviewed.

**Cepstral mean normalization** or **cepstral mean subtraction** (CMN/CMS)\(^3\) is a commonly used technique in speech recognition that can normalize for different microphones as well as telephone channels. Basically, a sample mean of the cepstrum vector is computed over an utterance. The mean is subtracted from the vector at each frame,

\[
\hat{y}'(n) = \hat{y}(n) - \bar{y},
\]

where

\[
\bar{y} = \frac{1}{N} \sum_{n=0}^{N-1} \hat{y}(n) \cong \bar{h}(n),
\]

which gives an estimate, \(\hat{x}(n)\), of the original speech \(\hat{y}'(n) \cong \hat{x}(n)\), if the speech process is zero mean in the transform domain. If it is not zero mean, \(\hat{y}'(n) \cong \hat{x}(n) - \bar{x}\). In this technique, speech frames are not distinguished from noisy frames of silence, which can affect this basic channel distortion estimate. CMS works well for unknown channels but does produce a distortion in the presence of additive noise. CMS is similar to early work done by Gish *et al.* in [10, 11], but CMS is a simplified version of their work.

**RASTA**, or RelAtive SpecTrAl processing, developed by Hermansky and Morgan [15, 16, 17], makes speech analysis less sensitive to steady-state spectral factors in speech and is particularly effective when the channel distortion is slowly varying. RASTA is essentially a band-pass filtering of the time trajectories of cepstra. The

\(^3\)The terms CMS and CMN are used interchangeably in the literature, depending on the author. CMS will be used in this thesis.
Table 2.2: Comparison of RASTA and CMS on the verbalized pronunciation (VP) data set with a 5k vocabulary. Models were trained on the WSJ0 training set.

<table>
<thead>
<tr>
<th>System</th>
<th>% Word Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sen.</td>
</tr>
<tr>
<td>Baseline</td>
<td>12.0</td>
</tr>
<tr>
<td>RASTA</td>
<td>12.5</td>
</tr>
<tr>
<td>CMS</td>
<td>11.8</td>
</tr>
</tbody>
</table>

high-pass filter portion alleviates the effects of convolution noise while the low-pass filter helps in smoothing out some frame-to-frame analysis artifacts. Because the filtering is in the cepstral (or log) domain, RASTA deals with convolutional noise and not additive noise. Further developments in RASTA techniques (referred to as J-RASTA [18]) have allowed for additive noise compensation at low SNR and will be discussed later.

RASTA and CMS are techniques for dealing with channel distortion and both are computationally inexpensive. BBN results [2], as summarized in Table 2.2, show that CMS and RASTA give similar percentage word error on test data with unknown and varying microphones, and thus, various channel distortions. However, on clean speech, RASTA does not necessarily help the results while CMS does not hurt and will often improve the performance. Results in [15] found that RASTA needs detailed models to work well, i.e. the models cannot be context-independent. Other work in cepstral filtering, including similar comparisons of RASTA and CMS, has been done by Hanson and Applebaum [13, 14].

Neumeyer, Digalakis and Weintraub [26] and Sankar and Lee [35, 36] present a maximum likelihood channel estimate that can be subtracted from a set of cepstral observation vectors, that is more accurate but also more computationally expensive than CMS. The maximum likelihood (ML) channel estimate is the channel
that maximizes the probability of the utterance, assuming a fixed but unknown stationary channel for each utterance. The ML channel estimate is:

$$\hat{h} = \arg\max_h p(Y|S, \Theta, h),$$

where $Y$ is the set of cepstral observations for $N_y$ frames of the utterance, $S$ is the HMM state sequence, $s_1, \ldots, s_{N_y}$, $\Theta$ is the set of model parameters, and $h$ is the fixed but unknown channel. If the recognition model uses Gaussian output distributions, where

$$p(\tilde{x}(n)|s_n) \sim N(\mu_{s_n}, \Sigma_{s_n}),$$

with the region-dependent mean, $\mu_{s_n}$, and covariance $\Sigma_{s_n}$, then the estimate of the channel, $\hat{h}$, is given by

$$\hat{h} = \left[ \sum_n \Sigma_{s_n}^{-1} \right]^{-1} \sum_n \Sigma_{s_n}^{-1} (\tilde{y}(n) - \mu_{s_n}).$$

Since the model region sequence $S$ is generally unobserved, $\hat{h}$ can be found iteratively using the EM algorithm\(^4\) for unknown $S$.

Results from SRI show that ML channel estimation is comparable to CMS on two different data sets. On both a Wall Street Journal (WSJ) test and using the Credit Card corpus, the percentage word error for CMS is similar to that of subtracting the ML channel estimate as seen in Table 2.3. Similar results were found in the BU implementation of this approach [4]. These results suggest that ML channel estimation is not cost-effective since it is computationally more expensive than CMS. However, the tests are on relatively long utterances where the sample mean may be a reasonable channel estimate, and the conclusions may be different for short utterances such as natural numbers.

\(^4\)The Expectation-Maximization (EM) algorithm [6] is an iterative algorithm used for dealing with missing or unobserved data, which in this case is the channel, $h$. It consists of an expectation step (E-step) and a maximization step (M-step). The E-step finds the expected log-likelihood of the complete data, given an estimated starting point. The M-step re-estimates the missing parameters by maximizing the log-likelihood. The steps are repeated until the estimates converge.
Table 2.3: *Comparison of CMS and channel estimation on two different data sets.*

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>% Word Error</th>
<th>WSJ Dev Set</th>
<th>CC corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS</td>
<td>21.6</td>
<td>45.4</td>
<td></td>
</tr>
<tr>
<td>Channel Est.</td>
<td>21.4</td>
<td>46.4</td>
<td></td>
</tr>
</tbody>
</table>

More recently, Chien, Lee and Wang [5] included the use of a prior channel distribution in their work to obtain a heuristic MAP channel estimate\(^5\) in an attempt to improve the match between clean speech models and a given telephone utterance. The results showed that their version of a MAP channel estimate is slightly better than the ML estimate, which itself is slightly worse than simply using CMS on the speech.\(^6\)

Rahim and Juang [33] have also implemented a ML channel estimate with improved recognition results. Their implementation does not incorporate the model covariances from the acoustic models but does iteratively calculate and remove ML channels for the same utterance. They also present a method for calculating channel estimates without knowing the entire utterance for use in near real-time applications.

**Adaptive Lin-Log RASTA** [15, 16, 17, 18] is a modification of regular RASTA processing, and is referred to as J-RASTA. It is able to deal with both convolutional and additive noise because the transformation is done in an alternate spectral domain which is linear-like for small spectral values and logarithmic-like for large spectral values.

---

\(^5\)“Heuristic” is used here because of an apparent math error in their work that gave an increased weight to the prior information for long utterances. While this gave improved results over the ML estimate, it is not actually a MAP channel estimate.

\(^6\)In this work, it should be noted that the transformation is performed in the model space, but since the model means alone are modified it is effectively a feature space transformation.
values. The modeling assumption for this process is

$$Y(f) = \ln(1 + J X(f))$$

where $Y(f)$ is a set of speech observations, $X(f)$ is the true speech, and $J$ is a SNR-dependent positive constant. The inverse transform, which is used to recover the true speech, is

$$X(f) = (e^{Y(f)} - 1)/J$$

which can be approximated as

$$X(f) \approx e^{Y(f)}/J$$

to prevent negative spectral values. (This approximation is more inaccurate for small spectral values than for larger ones.) $J$ has different values for different noise levels, and the use of the SNR-dependent $J$ values yields better recognition accuracy than traditional RASTA. For example, smaller values of $J$ work better for noisier speech because most of the signal is in the logarithmic-like part of the non-linearity while the noise is in the linear-like part. $J$ can be adjusted by making it inversely dependent on measured mean noise energy, templates for which need to be available prior to recognition. Results using this technique are comparable to a system trained and tested on identical noise conditions with sufficient data to estimate the noise energy. However, in telephone conversations there is at least initially very little data for estimating noise levels [18].

Stern et al. [1, 19, 22, 23, 24, 25, 38] have described various non-linear cepstral mapping techniques that have been shown to work well in noisy conditions and where training data microphones do not match test data microphones. However, many of their algorithms require parallel recordings to find adaptation vectors. The basic method is to find an additive compensation vector using such factors as an instantaneous signal-to-noise ratio. A method that does not require parallel recordings is Codeword-Dependent Cepstral Normalization (CDCN) [1] which includes
information from the vector quantized codeword to determine the mapping between unobserved true speech and acoustic models. This procedure is a non-linear mapping of a noisy utterance into a vector space representing clean speech through an estimation of noise and channel equalization vectors. It is essentially a maximum likelihood estimation of both the channel and the noise and in its pure form, CDCN uses the EM algorithm to find the channel and noise parameters.

With CDCN, no prior knowledge of either the noise or channel distortion is needed and parallel training data is not necessary. Training data from a variety of microphones can be "cleaned" iteratively using the CDCN algorithm to produce a universal model. Cleaning training data essentially means removing the additive noise and channel distortion to simulate a uniform recording environment. This additional procedure can increase the already significant computation for using CDCN but does make the procedure robust to microphone/telephone channel distortion and additive noise. This also suggests that other model cleaning methods can improve recognition performance. Other cepstral normalization algorithms have been developed that have reduced computation from CDCN. However, these variations require parallel recordings to find adaptation vectors and are not applicable to this thesis. CMU has done a significant amount of work in the area of telephone speech recognition and has more recently shown significant improvements using the CDCN algorithm on test sets with multiple microphones [19]. Recent work has involved the use of a vector Taylor series to approximate the models of the additive noise and channel distortion followed by a minimum mean squared error estimate to find the clean speech given the observed noisy speech.

2.3 Model Modification

Model modification is an alternative to feature processing that is more flexible than the methods described above (in that there are more free parameters) and thus
could result in higher performance. Model modification methods previously proposed include Stochastic Matching [35, 36], which deals with channel distortion only, and Parallel Model Combination (PMC) [7, 8, 9], which deals with both channel distortion and noise.

### 2.3.1 Parallel Model Combination (PMC)

In PMC, an estimation of a noisy speech model is found, given information about the clean speech and interfering noise. This technique can be used for both additive and convolutional noise. The statistics for convolutional noise to build a channel model can be found from speech observations given a global speech and silence model \( \Theta_G \), an additive noise model \( \Theta_N \), and a corrupted-speech mean measured from the test channel. The EM algorithm used to find the channel estimate \( \hat{h} \) is as follows:

1. Make an initial channel estimate, with no knowledge of utterance transcription: 
   \[ h^{(0)} \].
2. Estimate the clean speech and transcription given \( \Theta_G \), \( \Theta_N \), and \( h^{(i)} \).
3. Maximize the log likelihood of the data to get \( \hat{h}^{(i+1)} \).
4. Repeat steps 2 and 3 until \( \hat{h} \) has converged.

The new mean of the noisy hidden Markov models, working in the log spectral domain instead of the cepstral domain, is then \( E[\hat{Y}(f)] \) instead of \( E[\hat{X}(f)] \) where the combination of the additive noise results in

\[
\hat{Y}(f) = \ln(HX(f) + N_A(f)).
\]

The mean and covariances of the new model are found using a numerical integration technique described in [9]. If there is no additive noise present, the process is essentially CMS. If there is no convolutional noise, an ML estimate of the corrupted
model can be found given $\Theta_G$ and $\Theta_N$. A disadvantage of PMC is that it requires expensive numerical integration to find the channel estimate, and it is currently impractical for applications where the models must be frequently adapted.

### 2.3.2 Stochastic Matching of Models

Stochastic matching of models involves a maximum likelihood comparison between a noisy utterance and a given clean speech model to decrease the acoustic mismatch. This is particularly useful when the channel distortion is not fixed but random over an utterance. The given clean speech model, $\Theta_X$, is transformed to a model fitting the acoustic characteristics of the noisy speech, $\Theta_Y$. The functional form of the model transformation is based on prior knowledge of the acoustic mismatch. In [35] and [36], convolutional noise is accounted for in the model space as a random Gaussian vector and the noise parameters, $\mu_N$, $\sigma_N^2$, (assuming a diagonal covariance matrix), can be estimated separately for speech and silence frames. If the acoustic mismatch is assumed to be channel distortion, which is additive in the cepstral domain, then the model transformation is a linear combination of the statistics of the clean model and the noise where

$$
\mu_y = \mu_x + \mu_N \\
\sigma_y^2 = \sigma_x^2 + \sigma_N^2.
$$

The noise parameters are jointly estimated with the recognized word string using the EM algorithm. A version of this method will be further described in Chapter 5.

This technique applied in feature space is essentially that of the channel estimation technique described in [26]. Results comparing CMS and the application of this transformation in both feature space (cepstral vectors) and model space are shown in Table 2.4. There are not enough speakers in the available results to draw strong conclusions from the data, but it appears that applying this transform in the model space rather than in the feature space gives a bigger improvement in the
Table 2.4: *Stochastic matching percentage word error results on telephone speech for two speakers (from [35,36]).*

<table>
<thead>
<tr>
<th>Speaker</th>
<th>Matched</th>
<th>Mismatched Training/Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No Processing</td>
</tr>
<tr>
<td>A</td>
<td>2.7</td>
<td>24.3</td>
</tr>
<tr>
<td>B</td>
<td>7.0</td>
<td>24.1</td>
</tr>
</tbody>
</table>

recognition rate. The key difference between the two is the adjustment to the variance in the model space. Applying the transform in the feature space does not necessarily improve the results at all, which is not unexpected given SRT’s results using a similar transformation.

## 2.4 Limitations of Existing Techniques

In order to develop telephone applications using speech recognition, basic work in dealing with telephone channel distortion and noise is needed. Because none of the above methods achieve the same results as training and testing on clean speech for channel distortion, there is still a need for further work in this area.

Cepstral mean subtraction is simple and effective. There are currently no feature-domain transformations that have out-performed it, particularly given cost considerations. However, the picture may be different for short utterances because: 1) CMS may not be as effective here because phoneme information may be subtracted when the mean is not taken over many contexts, and 2) the introduction of a prior is more important for short utterances, which would be expected for typical telephone recognition applications and may improve the feature-space transformation approach.
The results of Sankar and Lee [35, 36] suggest more is to be gained by working in the model space than in the feature space. A limitation of their work that is addressed here is the assumption that channel vectors are independent at each time frame. This is alleviated by assuming one random channel model throughout an utterance, where the distribution of the channel is found during training. Since telephone channels are relatively constant during short utterances, the single random vector model is a more realistic assumption. A furthering of Chien, Lee and Wang’s work [5] is to modify the acoustic model means and covariances. Parallel model combination, which is also in the model space, will not be considered here because it is essentially CMS for channel distortion only.

A likely area for decrease of mismatch between training data and test speech is the implementation of maximum likelihood or MAP channel compensation on training data corrupted by channel distortion. This is covered here by using training data that has been recorded over numerous long distance phone lines and iteratively “cleaning” the data before training the final models. CMS is normally performed on both test and training sets but for short utterances, it makes sense to try other channel estimates since speech information could be removed from the signal along with the mean. Cleaning training data using an ML channel estimate, which will be discussed in Chapter 4, should prove to create a more robust model for use in testing utterances that have either ML or MAP channels removed.
Chapter 3

Baseline Work

This chapter describes the work done to find baseline recognition results for the Microphone natural numbers task, for comparison with feature space and model space transformation results discussed in Chapters 4 and 5. This chapter will begin with a description of the Microphone corpus, followed by a description of HTK, the hidden Markov model toolkit used for signal processing, model training, and recognition, and will conclude with a discussion of the baseline recognition results.

3.1 The Microphone Corpus

The first issue in looking at telephone speech is that of an appropriate corpus for training and experimentation. A speech corpus representing many random telephone channels, especially over long distance lines, will yield the most insight into the approaches in this work. A small vocabulary, such as digits, (fewer than 1000 words or simple digit combinations) with more telephone channel distortion is important for removing such variables as complicated language models and dictionaries and allows a tighter focus on the problems specific to the telephone speech recognition task, particularly that of the channel distortion. With this in mind, the best available
data is the Macrophone corpus [39], available through the Linguistic Data Consortium (LDC). The Macrophone corpus consists of about 200,000 utterances spoken by about 5,000 American speakers and was recorded directly from T1 digital telephone lines. Macrophone was developed as a part of a project to provide telephone speech for use in the development of automatic voice-interactive telephone services. Utterances range from dictated newspaper sentences to short strings of digits, from names and addresses to spelled words. Overall, this corpus is a better test of efforts to deal with the problems of channel distortion and noise because it was recorded on actual phone lines, including many long distance lines. Since 95% of the utterances are at 22dB or higher SNR (see Figure 3.1) [20], additive noise is not a major problem, so it is reasonable to focus to be on channel distortion alone. This thesis focuses only on the natural numbers section of the corpus because the utterances are typically shorter in the natural numbers section. It was thought that the shorter utterances may benefit more from the use of prior channel information, in the form of the MAP channel estimate. The average utterance has about 6 words, with a maximum of 18 words per utterance. The natural numbers section also provides a simpler recognition problem because of the smaller vocabulary. Typical natural number utterances are “four thousand ten”, “fifteen lira”, and “seventy two acres”, although the section does include such utterances as “I don’t have a house number.”

The Macrophone corpus is divided into a training data set and development and evaluation test sets. The natural numbers training data has about 15,000 utterances and the development test data is about 2,000 utterances. Because this set may be used for an official benchmark, and because results are still at the point where they may be improved, only the development test set has been used for recognition in this thesis. All of the training and test data used here was recorded on varying telephone lines with unknown microphones. The training set is 49% adult female and 37% adult male with the difference being girls, boys or unknown gender (both adult and child). The development test set is 40% adult male, 53% adult female, with the remainder
boys and girls.

3.2 HTK, a Hidden Markov Model Toolkit

This section is a description of the modules of HTK, a Hidden Markov Model toolkit, that were used in this work. HTK was developed by the Cambridge University Engineering Department and Entropic Research Labs. Version 1.5 was used in initial system development but Version 2.0 was used for all results reported here. HTK has software library modules and user-level tools for speech analysis, model training, Viterbi recognition, results analysis as well as interactive speech labeling. HTK can be used to build systems which recognize isolated words, connected-words, or continuous speech. HTK can be used for one pass recognition or to produce lattices that can be used for N-best rescoring, essentially multiple passes of recognition. HTK was chosen for this work because it is a toolkit that can be used to create a basic speech recognition system from scratch and because the source code is available for
modification. Figure 3.2 shows the library modules that surround a particular tool. These modules can be modified to effect changes over all HTK tools that use the modules. The overall structure of HTK is shown in Figure 3.3. The square boxes contain the command names which all start with “H”. As with the library modules, the HTK tools can be modified. This figure can be referred back to during the following descriptions of various HTK tools. The baseline results described in this chapter are based on an unmodified version of HTK. The results described in Chapters 4 and 5 will be based on a modified version of HTK.

The main experimentation stages in the speech recognition are 1) training speech models, 2) recognition on development test data for parameter tuning, and 3) recognition on evaluation test data. After a brief description of hidden Markov models, this section describes system set up, model building for both acoustic and language models, and concludes with a description of how recognition and results analysis is done.
3.2.1 Hidden Markov Models

As described in Chapter 2, speech is divided up into segments called phonemes. Acoustic models are trained to describe these phonemes. The most general case of phoneme modeling is to model each phoneme independently, creating approximately 50 acoustic models. These models are context-independent, or monophone models, as they do not take into account the preceding or following phonemes. While they are robust for small amounts of training data, they do not take into account the variability associated with phonemes spoken in the context of other phonemes. For example, the sound *uh* in the word “rut” is affected by the *r* and *t* sounds, making it slightly different from the *uh* in “bubble”, and can be more effectively modeled as a context-dependent, or triphone, model. HTK can model the acoustics of whole words or sub-words, such as monophones, biphones, triphones, or broad classes of phonemes. This work began with monophone modeling and built up to triphones,
Figure 3.4: A state diagram of a 3 state hidden Markov model.

with some biphones for the beginnings and ends of words and monophones for short words such as “a” and “Γ”. (Cross-word triphones are not used because the original system building was based on HTK1.5, which did not have the capacity to model cross-word triphones.)

Hidden Markov models (HMM)[3] are a commonly used, statistical model for speech recognition. HMMs can be used to model a variable-length sequence of frame-based cepstral feature vectors. The HMM is composed of states that generate the feature vectors over time. This may be more easily seen in Figure 3.4. The models are defined by the probabilities of going from one state \( i \) to another \( j \) within the model, known as transition probabilities \( a_{ij} \), as well as the output probability distribution of that state, \( b_i(y_n) \) for \( s(n) = i \). The topology of the models can be defined by the user before training the model; however, all models have both an entry and an exit state which are used to connect the models. These states do not produce observations so do not have associated output distributions. The output distributions of the HMMs can be modeled by many types of distributions; the most common is to use some form of Gaussian or mixture of Gaussians. The models in this work have 3 states and are modeled by mixtures of diagonal covariance Gaussians.
3.2.2 System Set Up

Before model training and recognition can be done, the data needs to be processed both for use with HTK and for use in model training and recognition. This means that the speech waveforms need to be parameterized into feature vectors, a dictionary of words and their phoneme mappings (pronunciations) needs to be created, and transcriptions must exist for all training and test speech; and this must all be in HTK format. This section will describe how to prepare a dictionary, transcription files, and encoded speech.

Dictionary Preparation

Given a phoneme set, there should be some phoneme mapping for every word in the training and test set vocabulary. HTK allows for multiple pronunciations as well as output symbols that do not necessarily match the word. This is useful for sentence begin and end markers. The HTK tool, HDMAn can be used to merge and alter old dictionaries to create an new dictionary. HDMAn can also be used to convert a monophone dictionary to one containing biphones and triphones. Dictionaries can also be automatically expanded before training or recognition for use with cross-word triphones.

A sample format of the dictionary is shown in Figure 3.5. It shows alternate pronunciations, each with their own line entry, as well as optional outputs, shown between square brackets.

Creating Transcription Files

Assuming that there are word transcriptions for the speech in some format, there are scripts available to convert them into HTK format files. The HTK tool HLEd can be used to manipulate the label files and will read in label files that are in TIMIT, Scribe, ESPS or HTK format. Depending on the level of detail desired,
label files can include boundary times, model labels (triphones or monophones), and confidence scores based (normally a log probability from the recognizer), as the format for training labels is the same as the format for recognized labels. For training, labels matching the model names are necessary but boundary times are not. Label files can also incorporate alternate word labels, which is used in the production of N-Best labels. Although the default in HTK is to create a separate label file for each waveform, they can be combined in a master label file (MLF). Sample label files are shown in Figure 3.6. An edit command script used in conjunction with HLEd can produce such changes as conversion from words to phonemes, or monophones to triphones (cross-word triphones or a combination of triphones, biphones, and monophones), renaming or deletion of phonemes if moving to a different set of phoneme labels, or to manipulate the labels in an MLF.
a. filename: woodchuck1.lab  b. filename: woodchuck2.lab
   w  woodchuck  woodchuck
   uw  ///  
   d  would  
   ch  chuck  
   ah  
   k

c. #!MLF!#
   
   “*/dt0000m-natnum05.rec”
   4900000 1100000 fifty
   .
   “*/dt0000m-natnum16.rec”
   5200000 8500000 six
   8500000 12900000 thousand
   12900000 14600000 nine
   14600000 18400000 hundred
   18400000 21900000 sixty
   21900000 24500000 four
   24500000 30300000 yards
   .

Figure 3.6: Sample label files: a.) individual file with both monophone and word labels, b.) individual file with alternate word labels, c.) master label with recognizer output including words and boundary times.
Speech Processing

The HTK tool \textbf{HCopy} can be used to convert waveforms to sequences of feature vectors. It can also be used to copy waveform files, or portions of the files, of different formats (NIST, Esignal, TIMIT) to HTK format. A configuration file setting various options is used to specify the parameters of the signal processing. The Macrophone data was in NIST waveform files and was converted using the option settings shown in Table 3.1. (See [41] for further options and equations for implementing the parameters.) The feature vectors for each 25 millisecond frame in this work have 13 coefficients, calculated using a 20 component Mel-Frequency filterbank. The feature vectors include 12 cepstral coefficients and a normalized log energy coefficient. The 12 cepstral derivatives, an energy derivative and cepstral mean subtraction are computed on the fly during all further use of the feature vectors. The library modules \textbf{HParm} and \textbf{HSigP} are used for all signal processing, especially in HCopy. HSigP functions are called by other tools to perform signal processing on the fly. By using a configuration file during training or recognition, HSigP can be called to compute cepstral derivatives or perform cepstral mean removal. Doing this kind of signal processing on the fly significantly reduces the amount of disk space required to store parameterized waveforms. \textbf{HList} can be used to view the parameterized speech waveform.

3.2.3 Model Training

Both language and acoustic models are trained to find the parameters that best describe their distributions. Together, they model the major aspects of speech. The language model defines the probability of a sequence of words, while the set of acoustic models describes the probability of a sequence of phonemes given the sequence of words. Together, these can be used to find the most likely word string. This section describes how the models are built using HTK.
Table 3.1: Signal processing parameters used for coding speech.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCEKIND</td>
<td>WAVEFORM</td>
<td>Source parameterization</td>
</tr>
<tr>
<td>SOURCEFORMAT</td>
<td>NIST</td>
<td>File format of source</td>
</tr>
<tr>
<td>TARGETKIND</td>
<td>MFCC,E</td>
<td>Target parameterization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mel-frequency cepstra with log energy</td>
</tr>
<tr>
<td>TARGETFORMAT</td>
<td>HTK</td>
<td>File format of output</td>
</tr>
<tr>
<td>TARGETRATE</td>
<td>100000</td>
<td>Target frame rate in 100ns units</td>
</tr>
<tr>
<td>ZMEANSOURCE</td>
<td>TRUE</td>
<td>Subtract out DC offset</td>
</tr>
<tr>
<td>USEHAMMING</td>
<td>TRUE</td>
<td>Use a Hamming window</td>
</tr>
<tr>
<td>PREEMCOEF</td>
<td>0.97</td>
<td>Set a pre-emphasis coefficient</td>
</tr>
<tr>
<td>CEPLIFTER</td>
<td>22</td>
<td>Cepstral lifting coefficient</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(for cepstra windowing)</td>
</tr>
<tr>
<td>NUMCHANS</td>
<td>20</td>
<td>Number of filterbank channels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(for cepstral analysis)</td>
</tr>
<tr>
<td>WINDOWSIZE</td>
<td>250000.0</td>
<td>Analysis window size in 100ns units</td>
</tr>
<tr>
<td>ENORMALIZE</td>
<td>TRUE</td>
<td>Normalize log energy</td>
</tr>
<tr>
<td>NUMCEPS</td>
<td>12</td>
<td>Number of cepstral coefficients</td>
</tr>
<tr>
<td>SCALE</td>
<td>1.0</td>
<td>Scale log energy</td>
</tr>
</tbody>
</table>
Acoustic Models

HTK uses the Baum-Welch (B-W) re-estimation algorithm [31], an instance of the EM algorithm, for estimating the parameters of the HMMs. Again, the parameters are the transition probabilities and the output distributions for each state of a given HMM. HTK models are created by iterating through the B-W algorithm, increasing the number of parameters in the model every few iterations to create a model that maximizes the likelihood of the data. Because there are so many parameters in an HMM set, a large amount of training data is needed to fully specify the parameters.

Initial parameters need to be found to bootstrap the models. There are two main ways of initializing the acoustic models for training. One is to begin with the function HCompV which computes the mean and covariance of the entire data set as well as a minimum variance vector which is used to prevent having variances go to zero. The prototype model from HCompV can be cloned to create models for a set of monophones. If training data with segmented phoneme labels is available, meaning labels with associated time boundaries, another method is to use the functions HInit and optionally HRest. HInit is used to create a prototype model based on a linear-in-time mapping of frames to states in each phoneme segment and subsequent Viterbi-style training. HRest may then be used for improving these models. If the time alignments are not that good, it may not be desirable to do all training using the original segmentations. The bootstrapped model from HInit and optionally HRest can then be sent to the tool HERest for further training. HERest does not require (and ignores) segment boundaries. If segmented training data is not available, HCompV works well. HERest is then used to perform iterations of the B-M algorithm. To fully train a level of a model, 2-5 passes of HERest should be made. HERest reads in a set of labels, usually in the format of a master label file, as well as the current version of the set of HMMs, usually in the format of a master model file (MMF) which is similar in set up to an MLF. Once a fully trained monophone model exists, the training data can be realigned using HVite (which will be described below) to take
multiple pronunciations of words into account.

The set of monophone models can be cloned to create triphone models using the model editing tool **HHEd**. After performing a few training iterations, the result is a significantly better model that takes up significantly more disk space. One way to deal with the disk space problem, as well as to prevent problems with sparse data for triphones that occur rarely or never, is to cluster the models. HTK has two clustering algorithms available for tying the parameters of model states, data-driven agglomerative clustering and tree-based divisive clustering. The agglomerative clustering continues until the largest weighted Euclidean distance between the means of any two states of a cluster reaches a threshold. For states with Gaussian mixtures, the criterion is based on the distance between mixture weights. The tree-based clustering, which can only be used at the level of a single-Gaussian output distribution, uses linguistically derived questions, making it possible to deal with unseen triphone combinations. This is useful for very large vocabulary systems but doesn’t necessarily improve the system if the recognition vocabulary is small and well-trained.

Once the models are clustered, the output distributions can be made more complex. HHEd can be used to increase the number of Gaussian mixtures in the output distributions. This is usually done in stages, with iterations of HERest being performed between each step where mixtures are increased. Because there are more free parameters as the number of Gaussian mixtures increases, there needs to be a large amount of training data to prevent variances from getting too small.

HERest has the useful option of running in parallel. Training data can be split up into smaller sections and sent to HERest, where the individual statistics for observations are accumulated. A final run of HERest processes the accumulator files and re-estimates the model parameters. If several cpus are available, this is extremely useful.
Language Models

HTK can be used to develop bigram language models (LMs), word-pair grammars, or simple word networks that are specific to a particularly recognition task. The output language models are in ARPA MIT-LI format. For small vocabulary recognition, a simple network may be best. For example, a network that allows all ten digits to follow each other in any order could be used for recognizing telephone numbers. For simple vocabularies under 1000 words, a word-pair grammar that simply allows some combinations of words works reasonably well. For larger vocabulary recognition, where there are many possible combinations but very different probabilities of different words following a given word, a probabilistic bigram model may give better results. The baseline results for this work result from a bigram model and a finite-state network.

The tool HLStats can generate bigram LMs given a word list and a master label file. HBuild is used to convert the matrix bigram file from HLStats to an HTK lattice file. HParse is used for generating word level lattices given a syntax file describing finite networks and word-pair grammars. To test a language model, HSGen can generate random utterances given a language model and the number of desired sentences.

3.2.4 Recognition and Analysis

In HTK, recognition and forced alignment of labels is performed using the tool HVite, which incorporates the Viterbi algorithm for pattern matching. The inputs to HVite are an HMM set, a word label file or a word network (language model) as defined by HParse or HBuild, a dictionary, and unknown parameterized speech. If a label file is used instead of a language model, forced alignment is performed. The outputs of HVite are either a single transcription in a label file or alternate transcriptions if N-best recognition is performed as shown in Figure 3.6. Depending
on the information desired, HVite can produce transcription files that include acoustic scores (normalized if desired), start and end times, words, models or even state information. Because HVite processes a single utterance at a time, the jobs can be split up to run on any available processors. Command line options that allow pruning and search thresholds to be set can greatly increase the speed of the program. There are also options to increase (or decrease) the weight of language model scores as well as an insertion penalty that should be optimized for a given set of data.

HResults is used to analyze the results of HVite. Although the final transcriptions are what we want from a recognizer, it is important to be able to gauge the fit of the speech models to real speech. Given a reference MLF, a word list and the results from HVite, HResults calculates the percentage correct for both sentences (utterances) and words as well as the percentage accuracy for words. Recognition performance is evaluated using percentage accuracy which is found by dividing the number of word errors by the total number of words in the correct transcriptions and subtracting from 1. The number of word errors is a sum of the number of substitutions, insertions and deletions. HResults can also be used for aligning test and reference transcriptions for sentences with errors, calculating confusion matrices and statistics for each file, calculating NIST format results (with or without NIST scoring), processing N-Best lists, and word spotting analysis.

3.3 Baseline Results

This section first describes the baseline model development and then describes the different recognition experiments performed to evaluate the models. Because CMS is the standard signal processing for dealing with any channel distortion, it will be the baseline channel estimate for this thesis. All training and test data has had the cepstral mean removed. A subset of the development test set was used to test the models on speech that included only numbers (See Appendix A.1 for the 31 word
vocabulary.) and no partial words or verbal hesitations. Results for both the full
test set and the subset will be included in these results. This work uses two bigram
language models based on different vocabulary sets, a 514 word open vocabulary and
a 559 word vocabulary closed on the test set, and a finite state network with a 31
word vocabulary. (See Appendix A.2 for the finite state grammar.) The perplexity\(^1\)
for the closed vocabulary test is about 118. The branching factor, a measure similar
to perplexity, for the finite state grammar is 28.73. The entire test set is composed of
1883 utterances (7529 words), and the subset includes 530 utterances (1514 words).

### 3.3.1 Model development

After the system set up is done, baseline work really begins with model develop-
ment. In this work, models were developed up to nine Gaussian mixtures per
state although some testing was performed at lower levels to insure that models were
giving acceptable results. Both the Natural Numbers (NN) training set and the
full Macrophone training set were used for training models. After 7 mixtures were
trained on the full data, only the NN data was used for higher mixture because it
was found that the full training set, which took up a large amount of disk space, did
not yield a significant improvement in recognition performance. At the 7 mixture
level, gender-dependent models were trained since gender-dependent models give a
recognition improvement in most reported results.

At the 1-mixture triphone level, both divisive (tree-based) and agglomerative
clustering were done to determine the best type of clustering to use. Recognition
experiments showed that for this particular test, the agglomerative clustering worked
slightly better than the divisive method. Clustering here is done separately on the
HMM states and there is no clustering based on word context.

\(^1\)Perplexity is a measure of the uncertainty of the next word for a given language model. A
lower number means that the next word is more certain, and is generally associated with a higher
recognition accuracy.
Table 3.2: Tuning the insertion penalty: results for a portion of the full test set.

<table>
<thead>
<tr>
<th>Insertion Penalty</th>
<th>Percentage Word Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>-120</td>
<td>65.10</td>
</tr>
<tr>
<td>-105</td>
<td>67.32</td>
</tr>
<tr>
<td>-90</td>
<td>68.90</td>
</tr>
<tr>
<td>-75</td>
<td>70.33</td>
</tr>
<tr>
<td>-68</td>
<td>70.91</td>
</tr>
<tr>
<td>-65</td>
<td>70.96</td>
</tr>
<tr>
<td>-60</td>
<td>70.80</td>
</tr>
<tr>
<td>-50</td>
<td>70.06</td>
</tr>
</tbody>
</table>

3.3.2 Results on the Development Test Set and Subset

Before recognition on the entire test set was performed, a random portion of the full test set (470 utterances) was used to tune the word insertion penalty. The results shown in Table 3.2 are from a 5 mixture model and have no other tuned parameters. Results for the full test set with the same model are shown in Table 3.3. The log probability of moving from one word to another is combined with an “insertion penalty” to discourage word insertions. The value of -65 will be used as the insertion penalty in all further experiments.

The results for the entire development test set in Table 3.3 show improvements with additional mixtures as well as with a larger weight for the bigram language model. The language model weight is an option that can be set in HVite to multiply the language model log likelihoods so that they have a greater or lesser impact on the final utterance score. These results all use a bigram language model. To reduce the computation time of recognition, a beam search parameter was set. This function
deactivates any models that contribute a log probability that is less than the current maximum by the amount of the threshold, called the pruning threshold. A smaller threshold makes the search faster, but may introduce search errors and may even cause the search to fail completely. In order to make sure that all utterances were recognized, the pruning threshold had to be at least 250.0. In other words, if any model has a maximum log probability that is 250.0 below the maximum probability for all models, it is deactivated from the recognition search.

Table 3.3: Results for full test set using full Macrophone training data on a gender-independent model.

<table>
<thead>
<tr>
<th>Percentage Word Accuracy</th>
<th>1.0</th>
<th>7.0</th>
<th>10.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 mixture</td>
<td>74.28</td>
<td>82.27</td>
<td>82.38</td>
</tr>
<tr>
<td>6 mixture</td>
<td>—</td>
<td>83.48</td>
<td>83.80</td>
</tr>
<tr>
<td>7 mixture</td>
<td>—</td>
<td>83.98</td>
<td>84.06</td>
</tr>
</tbody>
</table>

Two vocabulary sets were also tested. The first was an open-vocabulary that included all words in the training set but did not include all of the words in the test set. The other was closed over the test set, meaning every word in the test set was in the dictionary and incorporated into the bigram language model. The results are shown in Table 3.4 and were found using a language model weight of 10. Even though the closed-vocabulary makes it possible to correctly transcribe every utterance, the larger vocabulary introduces confusability into the recognition system. The vocabulary words added to the closed set were typically proper names for streets and cities and since they were not in the training data, they may not have been modeled well by the available triphone set.
Table 3.4: *Results for full test set and subset using 7 mixture, gender-independent model with different language models.*

<table>
<thead>
<tr>
<th>Language Model Type</th>
<th>Percentage Word Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Vocab.</td>
<td>84.06</td>
</tr>
<tr>
<td>Closed Vocab.</td>
<td>83.17</td>
</tr>
</tbody>
</table>

Using the data subset (530 utterances), the results show another jump in improvement over the full test set. The results, as seen in Table 3.5 were significantly better on this test set than on the full test set. The best results on the subset of data come with the use of a finite state grammar because of the smaller vocabulary.

Table 3.5: *Results for test subset using the finite state network and open and close vocabulary bigram language models.*

<table>
<thead>
<tr>
<th>Language Model</th>
<th>Percentage Word Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite State</td>
<td>90.36</td>
</tr>
<tr>
<td>Open Bigram</td>
<td>86.92</td>
</tr>
<tr>
<td>Closed Bigram</td>
<td>86.59</td>
</tr>
</tbody>
</table>

### 3.3.3 Gender-dependent Results

When utterances spoken by adult male and female speakers were recognized with the appropriate gender-dependent models, the results were significantly better than using the gender-independent model. However, boys and girls did much better on the gender-independent model than on either gender model, and worse with the male
model than the female. There was more female training data so the female models may simply have been better trained. The best outcome from the gender-dependent results is 84.89% assuming known gender and using the gender-independent model for children, compared to the comparable 6-mixture result of 83.48% for the gender-independent model on the full test set. The experiments are summarized in Table 3.6. Because gender-dependent models are computationally more expensive to train and give only a small improvement, the remainder of the results reported in this thesis will be gender-independent.

Table 3.6: Results for gender-dependent 6-mixture models using full Macrophone training data.

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Data Test Set</th>
<th>Percentage Word Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Female</td>
<td>Male</td>
</tr>
<tr>
<td>Female</td>
<td>85.76</td>
<td>66.98</td>
</tr>
<tr>
<td>Indep.</td>
<td>84.70</td>
<td>83.62</td>
</tr>
<tr>
<td>Male</td>
<td>74.72</td>
<td>85.23</td>
</tr>
</tbody>
</table>

3.3.4 Summary

The baseline CMS results that will be used for comparison with the feature-based and model-based channel compensation results are the 7-mixture acoustic model, open-vocabulary bigram language model result of 84.06% word accuracy. For the full test set, a bigram language model with a scoring weight of 10.0 will be used. A pruning threshold of no less than 250.0 will be used along with an insertion penalty of -65.
Chapter 4

Adaptation in Feature Space

The processing of cepstral feature vectors to account for telephone channel distortion is the focus of this chapter. The first section will describe channel estimation theory and will show the derivation of the maximum likelihood (ML) and maximum a posteriori probability (MAP) channel estimates. This section will be followed by a description of the implementation of the feature based estimates, which is then followed by two sections describing the development of the channel prior estimate for use in the MAP channel estimate and the recognition experimental results.

4.1 Channel Estimation Theory

Recall that the basic equation describing speech features that have gone over a telephone line is

\[ \hat{y}(n) = \hat{x}(n) + \hat{h}, \]

where \( \hat{y}(n) \) is the observed speech, \( \hat{x}(n) \) is the unobserved true speech and \( \hat{h} \) is the channel distortion. Subtracting an estimate of \( \hat{h}(n) \) from each side gives

\[ \hat{y}'(n) = \hat{y}(n) - \hat{h} = \hat{x}(n) + \hat{h} - \hat{h} = \hat{x}(n) + \Delta h, \]
where

$$\Delta h = \bar{h} - \hat{h}.$$  

Since the object of this work is to get the observation $\tilde{y}'$ as close to $\tilde{x}$ as possible, an estimate of $\Delta h$ can be used to further reduce the mismatch. This results in

$$\tilde{y}'' = \tilde{y}' - \Delta \hat{h} = \tilde{x} + \Delta h'$$

where

$$\Delta h' = \Delta h - \Delta \hat{h}.$$  

Although further iterations can be used to reduce the mismatch between the observed speech and the true speech, in this work, the initial estimate, $\hat{h}$ is simply the cepstral mean, i.e.,

$$\hat{h} = \hat{h}_{CMS} = \bar{y}$$

and the estimate of the channel error, $\Delta \hat{h}$, will be found using ML and MAP techniques. For simplicity of notation, $\tilde{y}'$, an observation with the cepstral mean removed, will be represented by $z$ and $\tilde{Y}'$, the sequence of observations with the cepstral mean removed, will be represented by $\mathcal{Z}$.

CMS is used as a first pass of the channel estimate because so much research has shown the benefits of cepstral mean subtraction and because it is so computationally inexpensive. For these reasons, it is also not worth focusing on acoustic models created without CMS. Models created without CMS may not be as robust as comparable models with CMS, defeating part of the purpose of research into robust speech recognition. Given that all training and recognition data will have the cepstral mean removed, what is left to estimate is the difference between the additive channel vector and the mean. Given the cepstral mean, $\bar{h}_{CMS}$, the total channel estimate, or bias term, becomes

$$\hat{h}_{ML} = \hat{h}_{CMS} + \Delta \hat{h}_{ML}$$

where $\Delta \hat{h}_{ML}$ is the ML estimate of the difference vector or

$$\hat{h}_{MAP} = \hat{h}_{CMS} + \Delta \hat{h}_{MAP}$$
where $\Delta \hat{h}_{MAP}$ is the MAP estimate of the difference vector. The remainder of this section will describe how $\Delta \hat{h}$ is found using ML and MAP estimation.

### 4.1.1 ML Estimates

The channel estimates implemented in this work were found using a variation of the maximum likelihood approach first discussed in [26] and [35, 36] and previously in Section 2.2.2. Recall from Section 2.2.2 that the ML channel estimate is:

$$
\hat{h} = \arg \max_h p(\hat{\mathbf{y}}|\mathbf{s}, \Theta, h),
$$

or alternatively

$$
\Delta \hat{h} = \arg \max_{\Delta h} p(\mathbf{z}|\mathbf{s}, \Theta, \Delta h, \hat{h}_{CMS}),
$$

where $\mathbf{z}$ is the sequence of cepstral observations with the mean removed for the $N_y = N_z$ frames of the utterance, $\mathbf{s}$ is the associated model state sequence, $s_1, \ldots, s_{N_x}$, $\Theta$ is the set of model parameters trained on data with the cepstral mean removed, $h$ is the fixed but unknown channel, and $\Delta h$ is the difference between the true channel and the cepstral mean. The model state sequence is generally unknown but it can be found using the EM algorithm (iteratively re-estimate the channel and then re-estimate the state sequence) or through a forced alignment of an utterance given a transcription from a first pass of recognition. This does require a decent first pass of recognition to have a reasonable state sequence. Finding $\Delta h$ rather than finding $\Delta h'$ is supported by the work of Sankar and Lee [35, 36] where it was shown that the channel estimate converged after one iteration of ML estimation. For an acoustic model with Gaussian output distributions, $p(x(n)|s_n) \sim N(\mu_{s_n}, \Sigma_{s_n})$ or $p(z(n)|s_n) \sim N(\mu_{s_n} + \Delta h, \Sigma_{s_n})$, the channel difference is found to be:

$$
\Delta \hat{h}_{ML} = [\sum_{n=1}^{N_x} \Sigma_{s_n}^{-1}]^{-1} \sum_{n=1}^{N_x} \Sigma_{s_n}^{-1} (z(n) - \mu_{s_n}).
$$

Most HMM systems make use of Gaussian mixtures for the model distributions,
where for a frame of clean speech, $x(n)$,

$$p(x(n)|s_n) \sim \sum_{i=1}^{L} \lambda_i p(x(n)|\mu_{i,s_n}, \Sigma_{i,s_n}),$$

with

$$p(x|\mu, \Sigma) \sim N(\mu, \Sigma),$$

and where $\lambda_i$ is the mixture weight for the $L$ mixtures in each output distribution with the mixture mean, $\mu_{i,s_n}$, and the covariance $\Sigma_{i,s_n}$. In this case, the observed speech has the distribution

$$p(z(n)|s_n, \Delta h) \sim \sum_{i=1}^{L} \lambda_i p(z(n)|\mu_{i,s_n} + \Delta h, \Sigma_{i,s_n}).$$

When solving for the channel estimate, conditional independence of the observations is assumed given the state sequence and the channel vector. The log is taken to give

$$\Delta \hat{h}_{ML} = \arg \max_{\Delta h} \sum_{n=1}^{N_x} \log p(z(n)|s_n, \Delta h).$$

Taking the derivative with respect to $\Delta h$ and setting it equal to zero, the equation becomes

$$\sum_{n=1}^{N_x} \frac{1}{p(z(n)|s_n, \Delta h)} \sum_{i=1}^{L} \lambda_i p_i(z(n)|s_n, \Delta h) \Sigma_{i,s_n}^{-1}(z(n) - \mu_{i,s_n} - \Delta h) = 0$$

since

$$\frac{\delta}{\delta x} \log f(x) = \frac{\delta \Sigma f(x)}{f(x)}.$$

While there is no closed form solution to this equation, it can be solved iteratively to find $\Delta \hat{h}_{ML}^{(p)}$ where $p$ indicates the iteration number.

Solving for $\Delta \hat{h}_{ML}^{(p)}$ yields

$$\Delta \hat{h}_{ML}^{(p)} = \left[ \sum_{n=1}^{N_x} \sum_{i=1}^{L} \tilde{p}_i(z(n)|s_n) \Sigma_{i,s_n}^{-1} \right]^{-1} \sum_{n=1}^{N_x} \sum_{i=1}^{L} \tilde{p}_i(z(n)|s_n) \Sigma_{i,s_n}^{-1}(z(n) - \mu_{i,s_n})$$

(4.1)

where

$$\tilde{p}_i(z(n)|s_n) = \frac{\lambda_i p_i(z(n)|s_n, \Delta h = \Delta \hat{h}_{ML}^{(p-1)})}{\sum_{j=1}^{L} \lambda_j p_j(z(n)|s_n, \Delta h = \Delta \hat{h}_{ML}^{(p-1)})},$$

(4.2)
If only one iteration is performed, the cepstral mean, \( \hat{h}_{CMS} \), is used as the channel estimate and

\[
\Delta \hat{h}^{(p-1)} = 0.
\]

In the experiments reported here, only one iteration is performed in order to keep computation costs small.

The resulting feature vector becomes:

\[
\tilde{y}''(n) = z(n) - \Delta \hat{h}_{ML} = \tilde{y}(n) - \hat{h}_{CMS} - \Delta \hat{h}_{ML}
\]

or equivalently

\[
\tilde{y}''(n) = \tilde{y}(n) - \hat{h}_{ML}.
\]

\subsection{MAP Estimates}

As pointed out before, the ML estimate maximizes the likelihood of the channel for a given utterance, but if that utterance is not long enough to adequately describe the channel, using prior knowledge of the channel distribution may give a better estimate of the channel. For the natural numbers task, composed of many short utterances, the MAP channel estimate may be more reliable than the ML estimate. For this derivation, the prior distribution of the channel distortion error is assumed to be

\[ p(\Delta h) \sim N(\mu, \Sigma). \]

(Section 4.3 will examine the validity of this assumption.) For the maximum \textit{a posteriori} channel estimate,

\[
\Delta \hat{h}_{MAP} = \arg \max_{\Delta h} p(\mathcal{Z}|\mathcal{S}, \Delta h) p(\Delta h) p(\mathcal{S}|\Delta h)
\]

for all observations \( z(n) \) of an utterance of length \( N_z \) with a model region or state sequence, \( s_1, \ldots, s_{N_z} \). It is assumed that \( p(\mathcal{S}|\Delta h) = p(\mathcal{S}) \), where \( p(\mathcal{S}) \) is simply a constant term, in order to get a practical solution. This is a reasonable approximation
if the first pass of recognition is relatively good. This results in

$$\Delta \hat{h}_{MAP} = \arg\max_{\Delta h} p(\mathcal{Z}|\mathcal{S}, \Delta h) p(\Delta h).$$

Assuming that all of the observations are conditionally independent and taking the log, this gives

$$\Delta \hat{h}_{MAP} = \arg\max_{\Delta h} \left[ \log p(\Delta h) + \sum_{n=1}^{N_z} \log p(z(n)|s_n, \Delta h) \right].$$

For the single mode Gaussian case, where $p(z(n)|s_n, \Delta h) \sim N(\mu_{s_n} + \Delta h, \Sigma_{s_n})$, then

$$\Delta \hat{h}_{MAP} = \arg\max_{\Delta h} \left[ -\frac{1}{2} \log |\Sigma_N| - \frac{1}{2} (\Delta h - \mu_N)^T \Sigma_N^{-1} (\Delta h - \mu_N) \ight. \\
+ \sum_{n=1}^{N_z} -\frac{1}{2} \log |\Sigma_{s_n}| - \frac{1}{2} (z(n) - \mu_{s_n} - \Delta h)^T \Sigma_{s_n}^{-1} (z(n) - \mu_{s_n} - \Delta h) \bigg] .$$

To find the maximum over $\Delta h$, the derivative (or gradient) with respect to $\Delta h$ is taken and the equation is set equal to zero:

$$-\Sigma_N^{-1}(\Delta h - \mu_N) + \sum_{n=1}^{N_z} \Sigma_{s_n}^{-1}(z(n) - \mu_{s_n} - \Delta h) = 0$$

$$\Sigma_N^{-1} \mu_N + \sum_{n=1}^{N_z} \Sigma_{s_n}^{-1}(z(n) - \mu_{s_n}) = \Sigma_N^{-1} \Delta h + \sum_{n=1}^{N_z} \Sigma_{s_n}^{-1} \Delta h$$

$$\Sigma_N^{-1} \mu_N + \sum_{n=1}^{N_z} \Sigma_{s_n}^{-1}(z(n) - \mu_{s_n}) = \left( \Sigma_N^{-1} + \sum_{n=1}^{N_z} \Sigma_{s_n}^{-1} \right) \Delta h .$$

Solving for $\Delta h$, this gives the MAP estimate:

$$\Delta \hat{h}_{MAP} = \left( \Sigma_N^{-1} + \sum_{n=1}^{N_z} \Sigma_{s_n}^{-1} \right)^{-1} \left( \Sigma_N^{-1} \mu_N + \sum_{n=1}^{N_z} \Sigma_{s_n}^{-1}(z(n) - \mu_{s_n}) \right) .$$

This estimate is used to modify the feature space observations to get

$$\tilde{y}''(n) = z(n) - \Delta \hat{h}_{MAP} = \tilde{y}(n) - \hat{h}_{CMS} - \Delta \hat{h}_{MAP} ,$$

or

$$\tilde{y}''(n) = \tilde{y}(n) - \hat{h}_{MAP} ,$$
effectively modifying the mean of the model as seen by the observations. This transformation is similar to the ML estimate described above but the channel estimate is found using the prior channel distribution. Note that $\Delta \hat{h}_{MAP}$ is the same as $\Delta \hat{h}_{ML}$ if $N_y \rightarrow \infty$.

The MAP channel estimate is easily extended for a model with Gaussian mixtures, as in the ML estimate case. For an acoustic model with $L$ Gaussian mixtures, the MAP channel estimate is

$$
\Delta \hat{h}_{MAP} = \left[ \Sigma_N^{-1} + \sum_{n=1}^{N_x} \sum_{i=1}^{L} \hat{p}_i(z(n)|s_n) \Sigma_{i,s_n}^{-1} \right]^{-1}
$$

$$
\left( \Sigma_N^{-1} \mu_N + \sum_{n=1}^{N_x} \sum_{i=1}^{L} \hat{p}_i(z(n)|s_n) \Sigma_{i,s_n}^{-1} (z(n) - \mu_{i,s_n}) \right)
$$

(4.3)

where

$$
\hat{p}_i(z(n)|s_n) = \frac{\lambda_i p_i(z(n)|s_n, \Delta h = 0)}{\sum_{j=1}^{L} \lambda_j p_j(z(n)|s_n, \Delta h = 0)}.
$$

(4.4)

### 4.2 Implementation

A new HTK tool, called HViteML, was written to implement feature-based channel estimation. These channel estimates, as well as any other feature-based channel estimate, fit into the BU-HTK system as shown in Figure 4.1. In order to get the mean and covariance matrix associated with the observation, it is necessary to have the model state sequence associated with the utterance. This comes from the transcription of a first pass of recognition. Although HVite can produce a state sequence as well as a transcription, for efficiency in experimentation, a middle pass of alignment in HViteML is used to generate the state sequence from the transcription. The channel estimation and subtraction is done during HViteML and the final pass of recognition is performed by HVite using the utterance feature vectors produced in HViteML.

Part of the implementation of the channel estimate involved “cleaning” the
training data. In cleaning training data, the first pass of recognition is skipped and the training transcriptions are used to generate the model state sequence and channel estimate with HViteML. This served two purposes, 1) to have new cepstral vectors that could be used to retrain the existing acoustic model to have "cleaner" parameters, and 2) to generate a body of ML channel estimates that could be used as a prior distribution estimate for the MAP estimate. The results of the ML and MAP channel estimates using a "cleaned" model for the final pass of recognition will be reported in the respective sections. The prior distribution calculations will be described below in Section 4.3.

The writing of HViteML involved using the alignment function of HVite to find the state sequence of an utterance given a word transcription found in a first pass of HVite, then calculating the ML or MAP channel and subtracting the channel estimate from the feature vector buffer. The modified feature vector was then saved in a new observation file for use in multiple experiments, i.e., for recognition with different acoustic models or language models. The implementation of the MAP channel estimate was essentially the same as that for the ML channel estimate. The difference was the inclusion of the channel prior distribution parameters. To save compute time, especially the costs of loading in acoustic models and feature vectors, the ML and MAP channel estimates were both implemented in the same version of HViteML.
At issue in this implementation was the fact that it was necessary to incorporate
extremely small likelihood values into the channel estimate. To prevent numerical
efforts in computation, the likelihood values were normalized over each mixture. So,
for Equations 4.1 and 4.3, Equations 4.2 and 4.4 are replaced by
\[
\hat{p}_i(z(n)|s_n) = \frac{\lambda_i p_i(z(n)|s_n, \Delta h)}{(\lambda p)_{\text{MAX}} \sum_{j=1}^L \frac{\lambda_j p_j(z(n)|s_n, \Delta h)}{(\lambda p)_{\text{MAX}}}}
\]
where
\[
(\lambda p)_{\text{MAX}} = \max_i \lambda_i p_i(z(n)|s_n).
\]

For the case of HMMs with diagonal covariances, as in this work, computation
is simplified. For a single component, \( k \), of the channel error estimate vector, the ML
channel estimate is
\[
\Delta \hat{h}_{\text{ML}}[k] = \left[ \sum_{n=1}^{N_x} \sum_{i=1}^L \hat{p}_i(z_k(n)|s_n)/\sigma_{i,s_n,k}^2 \right]^{-1} \sum_{n=1}^{N_x} \sum_{i=1}^L \hat{p}_i(z_k(n)|s_n)(\hat{z}_k(n) - \mu_{i,s_n,k})/\sigma_{i,s_n,k}^2.
\]
Similarly for the MAP case,
\[
\Delta \hat{h}_{\text{MAP}}[k] = \left[ \sigma_{N,k}^{-1} + \sum_{n=1}^{N_x} \sum_{i=1}^L \hat{p}_i(z_k(n)|s_n)/\sigma_{i,s_n,k}^2 \right]^{-1} \left( \mu_{N,k}/\sigma_{N,k} + \sum_{n=1}^{N_x} \sum_{i=1}^L \hat{p}_i(z_k(n)|s_n)(\hat{z}_k(n) - \mu_{i,s_n,k})/\sigma_{i,s_n,k}^2 \right).
\]

### 4.3 Channel Prior Distribution Estimation

Since there are so many different telephone channels, it is impossible to develop
models for each telephone channel. However, given a large amount of training data
recorded over many different channels, it is possible to get an idea of what a typical
telephone channel looks like. This section will describe the search for an appropriate
channel prior distribution.

Before any experiments were run, including baseline model development, the
signal processing function HSigP was altered so that the cepstral means were printed
during signal processing. The accumulated cepstral means of the entire Macrophone corpus training set, which includes both long and short utterances, were examined as a first pass estimate of the telephone channel to see what distribution would best fit. It became clear that a unimodal Gaussian would adequately describe the channel. Statistical plots for the first (of twelve) cepstral means can be seen in Figure 4.2. This figure shows 1) a histogram of the data, 2) a box plot, 3) a smoothed representation of the density, and 4) a normal probability plot. The histogram and density plots show a nice Gaussian shape, as desired. The box plot shows the median (the solid line inside the box) and the upper and lower quartiles are the upper and lower ends of the box. The normal plot shows a relatively linear match between the data and the standard normal distribution. These all point to a good match between the channel vector and a unimodal, multivariate Gaussian distribution, which will hopefully lead to a good fit of a Gaussian to $\Delta h$, as assumed in the previous sections.

Once HViteML was implemented, it was possible to run the program on the training data. Because the available training data was recorded over different telephone lines, the speech has added variability from channel and noise characteristics. It is possible to get a better model of the speech itself by “cleaning” the training data. There are two main reasons for cleaning the training data: 1) channel variability will be removed from the data used to train the model, and 2) information about the channel variability can be used during recognition to improve the channel estimation. Cleaning the training data can be done iteratively, using Maximum-Likelihood estimation as in the linear channel estimation technique described earlier where

$$\Delta \hat{h}_{ML}(i) = \arg\max_{\Delta \hat{h}} p(\mathcal{Y}_i | \Delta \hat{h})$$

where $\mathcal{Y}_i$ is a given training utterance. In other words, iteratively

1. estimate the channel vector ($\hat{h}_{CMS}$ for first iteration, $\Delta \hat{h}_{ML}$ for remaining passes) for each utterance given the current acoustic model and subtract it from the training data, and
Figure 4.2: Distribution plots of the means of the first component of the cepstral feature vector.
2. use the current training data to estimate an acoustic model.

In the experiments reported here, only one iteration of cleaning with the ML channel error estimate was done. The associated ML channel differences were then used to find the parameters for the channel prior distribution. Because the full Macrophone training set is so large, the ML channel error estimates were only calculated for the natural numbers section of the corpus.

S-Plus was used with the ML channel error estimates from the training data to insure that a unimodal multivariate Gaussian adequately represented the ML channel difference distribution. A set of distribution plots for the first component of the ML channel estimate is shown in Figure 4.3. The normal probability plot is not as straight for the ML error estimates as it was for the cepstral means. The significance of this is that the ML channel error estimates have longer tails than the actual Gaussian. However, it should be noted that, unlike the cepstral means shown in Figure 4.2, the ML channel error estimates were calculated only for the natural numbers portion of the data and they do not include the longer utterances of the entire Macrophone corpus. The longer tails suggest the use of a mixture distribution prior, but since the tails are less important for the MAP estimate, it was decided that the extra complexity was not necessary. The histograms of the components of the ML channel estimates are shown in Figures 4.4 and 4.5. While these may appear somewhat lopsided, a sampling of the data versus the normal distribution as seen in Figure 4.6 shows that a multivariate Gaussian is still a good approximation of the true channel error distribution over ±2 standard deviations.

Having decided on a unimodal multivariate Gaussian distribution as the appropriate form for the channel distribution, the next step is to find the correct parameters to describe the Gaussian. The means are easy to calculate but the covariance matrix form, whether full or diagonal, needs to be chosen. Figure 4.7 shows a pairwise scatter plot of four of the ML channel components. The fact that the spreads of the scatter plot are along the horizontal and vertical axes rather than along a tilted
Figure 4.3: Distribution plots of the first component of the MI channel error estimated vectors.
Figure 4.4: Histograms of the ML channel estimates (cepstra 1-6) from the training data.
Figure 4.5: Histograms of the ML channel estimates (cepstra 7-12) from the training data.
Figure 4.6: Quantiles of standard normal plots of four of the ML channel error estimate components.
Figure 4.7: Scatterplot of four of the ML channel error estimate components.

line indicates that the components are independent of each other. To further show this, a linear regression method was used to see if dependencies between components could be modeled linearly. From this method, the multiple $R^2$ coefficient, the ratio of the component variance explained by the regression over the total variance, is used to show any dependence between components [37]. The $R^2$ coefficients for the components are shown in Table 4.1. A number close to one indicates dependence. These results show that the components of the channel error estimate are well fit by a diagonal covariance matrix.

Table 4.2 gives the mean and diagonal covariance vectors used in the implementation of HViteML which included MAP channel estimation and subtraction.
Table 4.1: $R^2$ values when predicting channel error components with all other channel error components.

<table>
<thead>
<tr>
<th>Channel Components vs. Channel Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
</tbody>
</table>
Table 4.2: Channel error prior distribution parameters, assuming a Gaussian distribution with a diagonal covariance.

<table>
<thead>
<tr>
<th>Component</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1206</td>
<td>0.4210</td>
</tr>
<tr>
<td>2</td>
<td>-0.1453</td>
<td>0.9525</td>
</tr>
<tr>
<td>3</td>
<td>-0.2483</td>
<td>0.4206</td>
</tr>
<tr>
<td>4</td>
<td>-0.1574</td>
<td>1.4147</td>
</tr>
<tr>
<td>5</td>
<td>0.0897</td>
<td>0.4157</td>
</tr>
<tr>
<td>6</td>
<td>0.0057</td>
<td>0.5320</td>
</tr>
<tr>
<td>7</td>
<td>0.1095</td>
<td>0.3493</td>
</tr>
<tr>
<td>8</td>
<td>0.1054</td>
<td>0.5237</td>
</tr>
<tr>
<td>9</td>
<td>0.2361</td>
<td>0.2543</td>
</tr>
<tr>
<td>10</td>
<td>-0.0513</td>
<td>0.2717</td>
</tr>
<tr>
<td>11</td>
<td>-0.0033</td>
<td>0.1934</td>
</tr>
<tr>
<td>12</td>
<td>-0.0223</td>
<td>0.1957</td>
</tr>
</tbody>
</table>

The prior channel information found here is used in both the MAP channel estimate described earlier in this chapter and the Bayesian learning model adaptation described in Chapter 5.

4.4 Experimental Results and Discussion

Two main experiments were run using the ML and MAP channel estimates. One used the estimate on the test set to produce a set of modified cepstral vectors that are then recognized with the same acoustic model used to find the channel estimate. The other involved using the same set of modified cepstral vectors but with a cleaned acoustic model. These experiments were run using 7-mixture Gaussian
acoustic models. The clean model was based on the original 7-mixture model but was retrained on the natural numbers subset of the training data which had had the appropriate ML channels removed.

The ML and MAP channel results for these experiments using the full 1883 utterance test set with the open-vocabulary bigram language model are summarized in Table 4.3. Results using the closed-vocabulary LM are shown in Table 4.4. The decrease in accuracy for the closed-vocabulary LM is most likely due to the fact that the larger vocabulary introduces errors due to confusability of new, unlikely words. The use of the MAP channel error estimate did improve results over the ML channel estimate, as with Chien, Lee, and Wang [5], but not as much as their implementation. The implementation here is a correct implementation of the MAP channel estimate, however, rather than an approximation. Their implementation included an additional weighting of the prior information that did not allow the MAP estimate to go to the ML channel estimate as the number of frames increased. Their MAP estimate was

$$\Delta \hat{h}_{CLW} = \left[ \sum_{n=1}^{N_2} \sum_{i=1}^{L} \hat{p}_i(z(n)|s_n) \hat{\Sigma}_{i,s_n}^{-1} \right]^{-1} \left( \sum_{n=1}^{N_2} \mu_N + \sum_{n=1}^{N_2} \sum_{i=1}^{L} \hat{p}_i(z(n)|s_n)(z(n) - \mu_{i,s_n}) \hat{\Sigma}_{i,s_n}^{-1} \right).$$

For comparison, results for an implementation of this weighted MAP channel estimate using the cleaned acoustic model are shown in Table 4.5. As expected, the addition of the weight hurts the results. It may have helped in [5] because they were recognizing isolated words and probably had a very small frame length, $N_z$. Overall, model cleaning helps the results quite a bit. For the test subset, model cleaning decreased the word error rate by up to 14.4%. Although model cleaning hurt performance slightly for the full test set when using the closed-vocabulary bigram language model, model cleaning improved the results for the test subset with the same language model.

The test subset results shown in Table 4.6 show how much the constrained language model helps in this task. The improvements shown here may be partly attributed to the better initial transcriptions as seen in the original CMS results.
Table 4.3: CMS, ML and MAP channel estimation results on telephone speech, using the full test set and the open-vocabulary bigram language model.

<table>
<thead>
<tr>
<th>Percentage Word Accuracy</th>
<th>( h_{CMS} )</th>
<th>( h_{ML} )</th>
<th>( h_{MAP} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Model</td>
<td>84.06</td>
<td>84.09</td>
<td>84.10</td>
</tr>
<tr>
<td>Clean Model</td>
<td>X</td>
<td>84.38</td>
<td>84.36</td>
</tr>
</tbody>
</table>

Table 4.4: CMS, ML and MAP channel estimation results on telephone speech, using the full test set and the closed-vocabulary bigram language model.

<table>
<thead>
<tr>
<th>Percentage Word Accuracy</th>
<th>( h_{CMS} )</th>
<th>( h_{ML} )</th>
<th>( h_{MAP} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Model</td>
<td>83.17</td>
<td>83.21</td>
<td>83.21</td>
</tr>
<tr>
<td>Clean Model</td>
<td>X</td>
<td>83.09</td>
<td>83.11</td>
</tr>
</tbody>
</table>

Table 4.5: MAP and weighted MAP channel estimation results on telephone speech, using the cleaned acoustic model and the open-vocabulary bigram language model.

<table>
<thead>
<tr>
<th>Percentage Word Accuracy</th>
<th>( h_{MAP} )</th>
<th>( h_{CLW} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Set</td>
<td>84.36</td>
<td>84.14</td>
</tr>
<tr>
<td>Full Test Set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Subset</td>
<td>88.90</td>
<td>88.24</td>
</tr>
</tbody>
</table>
Table 4.6: CMS, ML and MAP channel estimation results on telephone speech, using the test subset and three types of language models.

<table>
<thead>
<tr>
<th>Acoustic Model</th>
<th>Language Model</th>
<th>$\hat{h}_{CMS}$</th>
<th>$\hat{h}_{ML}$</th>
<th>$\hat{h}_{MAP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Model</td>
<td>Finite State</td>
<td>90.36</td>
<td>90.49</td>
<td>90.49</td>
</tr>
<tr>
<td></td>
<td>Open Bigram</td>
<td>86.92</td>
<td>87.12</td>
<td>87.12</td>
</tr>
<tr>
<td></td>
<td>Closed Bigram</td>
<td>86.59</td>
<td>86.79</td>
<td>86.79</td>
</tr>
<tr>
<td>Clean Model</td>
<td>Finite State</td>
<td>X</td>
<td>91.02</td>
<td>91.08</td>
</tr>
<tr>
<td></td>
<td>Open Bigram</td>
<td>X</td>
<td>88.97</td>
<td>88.90</td>
</tr>
<tr>
<td></td>
<td>Closed Bigram</td>
<td>X</td>
<td>87.78</td>
<td>87.85</td>
</tr>
</tbody>
</table>

Better transcriptions translate to a better state sequence which gives a more accurate channel error estimate. Additionally, since over 72% of the utterances in this subset are 3 words or less (78% are 4 words or less), this subset can be considered a subset of short utterances. The improvements shown here could thus be due to the impact of the better channel estimate on short utterances. Especially notable is the jump in word accuracy when going from the original model to the clean model in the open-vocabulary case. In this case, there is a 15.7% improvement in word error rate for the ML estimate and a 15.1% improvement for the MAP estimate with a 95.3% confidence rate that the ML and MAP estimates are better than the cepstral mean alone. With the finite state network, there is a 6.8% improvement in word error rate for the ML estimate and 7.5% for the MAP estimate. The fact that the performance gain in the open-vocabulary bigram LM case is bigger than for the finite state LM case suggests that the performance gain can be explained by the length of utterances and not simply the higher first pass accuracy.
4.5 Summary

This chapter addresses three of the questions raised in the introduction. With regards to the first question, a prior distribution for the channel error estimate has been established, and the additional use of the prior distribution in the channel error estimate has been shown to give a small gain in recognition. The investigation of the prior channel information supports the use of a single mixture, multivariate Gaussian as an acceptable model of the channel distribution. Although other sites have shown that an implementation of the ML channel does not necessarily improve results, which was supported by earlier BU results indicating a similar phenomena [4], the implementation of the ML channel error estimate here actually improved the word accuracy. Significant gains are observed for the cleaned models, which were not used in previous work. Question two deals with the issue of channel estimation for short utterances. The use of a test subset which contains many short utterances allows us to examine this issue. There is a consistent improvement in the results when moving from CMS to the channel estimates, especially in the case of the bigram language model based on the open-vocabulary set used with the clean acoustic model. The larger gains for the test subset over the full test set suggest that sentence length is an important factor, although increased first pass accuracy may also play a role. Answering the third question, “cleaning” the training data does improve the recognition results. One reason is that the removal of the ML channel from the training data allows for an acoustic model that better matches the recognition data. This result suggests further improvements could come from changing the acoustic model, as will be addressed next in Chapter 5.
Chapter 5

Transformations in the Model Space

This chapter focuses on modifying the acoustic models of speech to compensate for linear telephone channel distortion. Essentially, a new acoustic model is developed for each recognized utterance through a transformation that takes into account the convolutional noise of the given utterance. The first section will describe model modification theory and give an explanation for the Stochastic Matching (SM) technique developed by Sankar and Lee [35, 36] followed by a derivation of model modification through Bayesian learning (BL). This section will be followed by a description of the implementation of the model modification techniques followed by a discussion of the Stochastic Matching and Bayesian learning results. This chapter closes with a discussion of the relative advantages and disadvantages of model-space transformations in comparison to feature-space transformations.
5.1 Model Modification Theory

A simple modification of model means can be done in the feature space. Subtracting a constant additive bias in the feature space is equivalent to adding a bias term to every model mean and performing this in the feature space is computationally inexpensive. When a random bias is assumed, it can only be implemented in the model space because it requires a covariance transform. This section will describe the theory of Stochastic Matching, which requires no prior information about the channel distortion, and Bayesian learning, a technique for incorporating channel information into the acoustic model parameters.

5.1.1 Stochastic Matching

An acoustic model developed on clean speech can be made to match telephone speech in recognition by using a transformation that takes into account the convolutional noise of a given test utterance in terms of its effect on both the distribution means and covariances. In Stochastic Matching, the transformation between the observed speech with the cepstral mean removed, \( \tilde{y}'(n) \), here represented by \( z \), and the true speech, \( \bar{x}(n) \), is again assumed to be

\[
  z(n) = \bar{x}(n) + \Delta h(n),
\]

but now the channel error \( \Delta h(n) \) is a random vector that is drawn independently at each time frame from its distribution. Assuming the channel distribution is given by

\[
p(\Delta h(n)) \sim N(\mu_C, \Sigma_C)
\]

where \( \mu_C \) and \( \Sigma_C \) are unknown, a new model better representing the noisy utterance is created from the clean Gaussian model such that

\[
p(z(n)|s_n, \mathcal{Z}, \mathcal{S}) \sim N(\mu_{s_n} + \mu_C, \Sigma_{s_n} + \Sigma_C)
\]
where $\mu_C$ and $\Sigma_C$ are the parameters of the channel model $\Theta_C$. If the original model is Gaussian, the resulting model will be Gaussian. Likewise, if the original model is mixture Gaussian, the result will be a similar mixture Gaussian model. The unimodal, diagonal Gaussian case will be presented here for simplicity.

As for transformations in the feature space, the estimation of the new model parameters

$$\Theta'_C = \arg\max_{\Theta_C} p(\mathcal{Z}|\Theta, \Theta_C)$$

requires the EM algorithm because of the unknown state sequence. However, we will use an approximate solution here

$$\Theta'_C \approx \arg\max_{\Theta_C} p(\mathcal{Z}|\mathcal{S}, \Theta, \Theta_C)$$

where $\Theta$ is a clean speech model, $\mathcal{S}$ is the most likely state sequence of the given utterance, $\mathcal{Z}$, and $\Theta_C$ describes the noise parameters. Maximizing with respect to $\Theta_C = (\mu_C, \sigma_C^2)$ does not give a closed form solution. One solution, presented in [34] and [35, 36], uses an iterative re-estimation algorithm where

$$\mu'_C = \frac{1}{N_x} \sum_{n=1}^{N_x} \mu_C + \frac{\sigma_C^2}{\sigma_{sn}^2 + \sigma_C^2} (z(n) - \mu_{sn} - \mu_C)$$

and

$$\sigma_C'^2 = \frac{1}{N_x} \sum_{n=1}^{N_x} \left[ \frac{\sigma_{sn}^2 \sigma_C^2}{\sigma_{sn}^2 + \sigma_C^2} + \left( \mu_C + \frac{\sigma_C^2}{\sigma_{sn}^2 + \sigma_C^2} (z(n) - \mu_{sn} - \mu_C) \right)^2 \right] - (\mu'_C)^2.$$  

The assumptions used in the implementation of these equations here and in [35, 36] are that the initial $\sigma_C^2$ is a small positive number and the initial $\mu_C$ is 0. The same mean and covariance transformations are used when the remaining distortion is the channel error, $\Delta h$, rather than the channel $h$.

As presented by Sankar and Lee, the transformations can be different for each HMM state. In this work, however, the SM transformations will be assumed to be constant over the utterance for comparison with Bayesian learning. This assumption is based on the fact that the telephone channel varies slowly.
5.1.2 Bayesian Learning

As opposed to Stochastic Matching, Bayesian learning assumes that the channel is constant over the entire utterance and is thus estimated given the entire utterance. Here, the transformation between the recorded utterance and the true speech is

\[ z(n) = \hat{x}(n) + \Delta h. \]

As with SM, the transformed model has the same structure as the clean speech model (i.e., Gaussian with shifted mean and variance), but it is found using

\[ p(z(n)|s_n, \mathcal{Z}) = \int p(z(n)|s_n, \Delta h)p(\Delta h|\mathcal{Z})d\Delta h \]

where \( z \) is the test speech with the cepstral mean removed, \( \mathcal{Z} \) is the current utterance, and \( \Delta h \) is a possible value of the channel error. The channel error distribution is

\[ p(\Delta h) \sim N(\mu_N; \Sigma_N) \]

where \( \mu_N \) and \( \Sigma_N \) are known parameters. Using the posterior distribution \( p(\Delta h|\mathcal{Z}) \) in this way requires modification of both the model means and covariances, as opposed to using \( p(z(n)|s_n, \Delta h) \), which can be accomplished in feature space since it only involves modifying the means.

The posterior channel error distribution parameters, which are calculated for every utterance, become

\[ \mu_{new} = (\Sigma_N^{-1} + \sum_{n=1}^{N_s} \Sigma_{s_n}^{-1})^{-1} [\Sigma_N^{-1} \mu_N + \sum_{n=1}^{N_s} \Sigma_{s_n}^{-1} (z(n) - \mu_{s_n})] \]

and

\[ \Sigma_{new} = (\Sigma_N^{-1} + \sum_{n=1}^{N_s} \Sigma_{s_n}^{-1})^{-1}, \]

as will be shown below. Here, the transformation is an additive channel estimate where

\[ \Delta \hat{h} \approx \mu_{new} \]
and $\Sigma_{new}$ is a compensation term to deal with doubt about the channel error estimate. For the unimodal Gaussian case, the modified acoustic model is now

$$p(z|z) \sim N(\mu_s + \mu_{new}; \Sigma_s + \Sigma_{new}).$$

A major difference between this approach and Stochastic Matching is that the channel estimate takes into account the prior distribution of the channel found during training. If the prior distribution is indeed well modeled by a Gaussian and the channel is constant, this difference should in principle give an improvement over Stochastic Matching because it is a better model of channel distortion.

Using Bayesian learning, the distribution of the speech comes from

$$p(z(n)|s_n, z, s) = \int p(z(n)|s_n, \Delta h) p(\Delta h|z, s) d\Delta h$$

instead of $p(z(n)|s_n, \Delta \hat{h})$ (e.g. $\Delta \hat{h} = \Delta \hat{h}_{MAP}$ as found in Chapter 4). Here, $s$ is the HMM state sequence associated with the utterance $z$ and $\Delta h$ is the random channel error.

To solve for $\mu_{new}$ and $\Sigma_{new}$, $p(\Delta h|z, s)$ must be found. Bayes’ rule gives

$$p(\Delta h|z, s) = \frac{p(z|s, \Delta h) p(\Delta h)}{p(z|s)}$$

(5.1)

where $p(z|s)$ is a constant term with respect to $\Delta h$, $p(\Delta h) \sim N(\mu_N, \Sigma_N)$, and

$$p(z|s, \Delta h) = \prod_{n=1}^{N_s} p(z(n)|s_n, \Delta h)$$

where

$$p(z(n)|s_n, \Delta h) \sim N(\mu_s + \Delta h, \Sigma_{s_n})$$

as before.

Thus, equation 5.1 becomes

$$p(\Delta h|z, s) = K \exp \left[ -\frac{1}{2} (\Delta h - \mu_N)\Sigma_N^{-1}(\Delta h - \mu_N) \right.$$

$$\left. -\frac{1}{2} \sum_{n=1}^{N_s} (z(n) - \mu_{s_n} - \Delta h)\Sigma_{s_n}^{-1}(z(n) - \mu_{s_n} - \Delta h) \right]$$
where $K$ is a constant chosen such that $\int p(\Delta h|\mathcal{Z}) \, d\Delta h = 1$. This becomes

$$p(\Delta h|\mathcal{Z}, \mathcal{S}) = K \exp \left[ -\frac{1}{2} \left( \Delta h^T \left( \Sigma_N^{-1} + \sum_{n=1}^{N_s} \Sigma_{s_n}^{-1} \right) \Delta h \right.ight.$$ 

$$- 2 \Delta h^T \left( \Sigma_N^{-1} \mu_N + \sum_{n=1}^{N_s} \Sigma_{s_n}^{-1} \left( z(n) - \mu_{s_n} \right) \right) + k \right]$$

where $k$ is the combination of the terms independent of $\Delta h$. Setting $K' = K e^{-\frac{1}{2}k}$, the equation becomes

$$p(\Delta h|\mathcal{Z}, \mathcal{S}) = K' \exp \left[ -\frac{1}{2} \left( \Delta h^T \Sigma_{\text{new}}^{-1} \Delta h - 2 \Delta h^T \Sigma_{\text{new}}^{-1} \mu_{\text{new}} \right) \right],$$

where

$$\mu_{\text{new}} = \left( \Sigma_N^{-1} + \sum_{n=1}^{N_s} \Sigma_{s_n}^{-1} \right)^{-1} \left[ \Sigma_N^{-1} \mu_N + \sum_{n=1}^{N_s} \Sigma_{s_n}^{-1} \left( z(n) - \mu_{s_n} \right) \right]$$

and

$$\Sigma_{\text{new}} = \left( \Sigma_N^{-1} + \sum_{n=1}^{N_s} \Sigma_{s_n}^{-1} \right)^{-1}.$$

Thus, the channel distribution given observations $\mathcal{Z}$ is $p(\Delta h|\mathcal{Z}) \sim N(\mu_{\text{new}}, \Sigma_{\text{new}})$, where $\mu_{\text{new}}$ and $\Sigma_{\text{new}}$ are calculated for each utterance or each time window of adaptation. Note that $\mu_{\text{new}}$ is identical to $\hat{\mu}_{\text{MAP}}$ for the unimodal Gaussian acoustic model case as described in Chapter 4 and $\Sigma_{\text{new}}$ is a byproduct of the calculation of $\mu_{\text{new}}$.

Going back to the original integral,

$$p(z(n)|s_n, \mathcal{Z}, \mathcal{S}) = \int p(z(n)|s_n, \Delta h) p(\Delta h|\mathcal{Z}, \mathcal{S}) \, d\Delta h$$

and recalling that $p(z(n)|s_n, \Delta h) \sim N(\mu_{s_n} + \Delta h, \Sigma_{s_n})$, $p(z(n)|s_n, \mathcal{Z}, \mathcal{S})$ is the convolution of two distribution functions which corresponds to the sum of two independent random variables. Since the sum of two independent Gaussians is Gaussian, the new distribution is described as

$$p(z(n)|s_n, \mathcal{Z}, \mathcal{S}) \sim N(\mu_{s_n} + \mu_{\text{new}}, \Sigma_{s_n} + \Sigma_{\text{new}}).$$

A complicating factor here is the use of multiple mixture Gaussians for the acoustic model, yet $p(z(n)|s_n, \Delta h)$ and $p(\Delta h)$ need to be unimodal Gaussian for
\( p(\Delta h | \mathcal{Z}, \mathcal{S}) \) to be Gaussian and for Bayesian learning to be straightforward. Since multi-mixture Gaussian models generally give the best word recognition results, it is important to make sure that Bayesian learning can be used within this framework. One option is as follows. Given the basic framework

\[
p(z(n) | \mathcal{Z}) = \int p(z(n) | s_n, \Delta h) p(\Delta h | \mathcal{Z}) d\Delta h,
\]
a single mixture Gaussian is used to find \( p(\Delta h | \mathcal{Z}) \) such that

\[
p(\Delta h | \mathcal{Z}) = \prod_{n=1}^{N} \frac{p'(z(n) | \Delta h) p(\Delta h)}{p(\mathcal{Z})}
\]

where

\[
p'(z(n) | \Delta h) \sim N(\mu, \Sigma).
\]

\( p(z(n) | \Delta h) \) remains the \( L \)-mixture Gaussian used for recognizing the original utterance to get the best possible state sequence in a first pass with

\[
p(z(n) | s_n, \Delta h) \sim \sum_{i=1}^{L} \lambda_i p(z | \mu_{i, s_n}, \Sigma_{i, s_n}).
\]

The resulting \( p(z(n) | \mathcal{Z}) \) is the sum of \( L \) convolutions, so once the mean, \( \mu_{\text{new}} \), and covariance, \( \Sigma_{\text{new}} \), transforms are calculated for an utterance given a single mixture acoustic model, they can be used to transform multi-mixture acoustic models. The resulting multiple mixture Gaussian acoustic model is

\[
p(z(n) | s_n, \mathcal{Z}, \mathcal{S}) \sim \sum_{i=1}^{L} \lambda_i p(z | \mu_{s_n} + \mu_{\text{new}}, \Sigma_{i, s_n} + \Sigma_{\text{new}}),
\]

where each mixture covariance is modified by the same transform.

### 5.2 Implementation

As noted above, the Bayesian learning model adaptation described here mirrors the MAP channel error estimate described in Chapter 4. The difference with model modification is that the covariances of the models are adjusted to better match
the statistics of the unobserved true speech. The model mean transformations are equivalent to a feature space shift. With this in mind, implementation becomes an easier task. The modified utterance cepstra from HViteML effectively incorporate the mean transformations for Bayesian learning when the MAP cepstra are used; what remains is to incorporate the covariance transform for each utterance. For this stage, HViteML was further modified to output the covariance transform for each utterance when calculating the initial MAP channel estimates. HViteMod was then written to incorporate the covariance transforms into the recognition algorithm by adding the utterance covariance transform to the covariances in the model before recognizing an utterance.

HViteSM, a modified version of HViteML, was used to calculate the Stochastic Matching transformations for a one mixture model. The mean transformation is still incorporated in the feature space and the covariance transform is saved for use in HViteMod. Although Sankar and Lee recommend modifying the covariances for the cepstral derivatives [35], that was not implemented here. This is left for future implementations of Stochastic Matching.

5.3 Experimental Results and Discussion

In order to take advantage of the best acoustic models available, the Bayesian learning model transformation parameters were based on a set of utterance transcriptions that came from a 7-mixture Gaussian acoustic model, \( p(x(n)|s_n, \Delta h = 0) \) trained on CMS data (not cleaned). A unimodal Gaussian model, \( p'(x(n)|s_n, \Delta h = 0) \), also trained on CMS data, was used to estimate the mean and covariance transformations representing \( p(\Delta h|Z) \). These transformations were then incorporated into a 7-mixture Gaussian model trained on ML cleaned data, \( p(x(n)|s_n, \Delta h = \Delta \hat{h}) \), for the final pass of recognition. To have a fair comparison with a feature space transform, recognition was performed using only the Bayesian learning mean transformation, which is simply
a MAP channel error estimate. The MAP estimate here will be denoted \( \hat{h}'_{\text{MAP}} \), because it is not the same result as in Chapter 4 where a 7-mixture acoustic model was used to estimate the MAP feature transformation.

Stochastic Matching model transformations were also based on transcriptions from a 7-mixture model \( p(x(n)|s_n, \Delta h = 0) \), but the state sequence and the parameters \( \mu_C \) and \( \Sigma_C \) were found using a unimodal Gaussian acoustic model, \( f'(x(n)|s_n, \Delta h = 0) \). The use of the unimodal model was mainly done for simplicity of implementation. The SM feature space transformation, where only the means are modified, is effectively a maximum likelihood channel estimate. Results using this transformation are denoted \( \hat{h}'_{\text{ML}} \) and are used to examine the effects of the covariance transformation. Future implementations of SM could use mixture models for the estimation of mean and covariance transforms, with potentially improved results.

The experiments described in this section include results on both the full test set and the subset, for both the Bayesian learning and Stochastic Matching adaptation schemes. The results are compared with baseline mean transformations (feature space transformations) for the channel error estimate calculated using a set of unimodal Gaussian HMMs implemented in the feature space as well as the MAP and ML results from Chapter 4. Recall that the baseline results using cepstral mean subtraction are 84.06% for the full test set with the open-vocabulary bigram LM and 90.36% for the subset with the finite state grammar. All of the techniques implemented here demonstrate an improvement over the baseline CMS results.

The recognition results for the Bayesian learning experiments on the test subset are shown in Table 5.1. Results for the full test set using the open-vocabulary bigram language model and the cleaned acoustic model are in Table 5.2. The results for Bayesian learning show that the model space transformation improves recognition accuracy over the feature space transformation in some cases. Specifically, for the open-vocabulary bigram language model condition, Bayesian learning gives the best results overall of the techniques implemented in this work. On the test subset, the
Table 5.1: Bayesian learning results on telephone speech, using the test subset, two types of language models, and a cleaned acoustic model with associated feature space transformation. The CMS baseline is 90.36% for finite state LM and 86.92% for open-vocabulary bigram.

<table>
<thead>
<tr>
<th>Language Model</th>
<th>Feature Space</th>
<th>Model Space</th>
<th>Bayesian Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite State</td>
<td>91.08</td>
<td>90.89</td>
<td>90.89</td>
</tr>
<tr>
<td>Open Bigram</td>
<td>88.90</td>
<td>88.90</td>
<td>89.04</td>
</tr>
</tbody>
</table>

The model space transformation decreases the word error rate with respect to CMS by 6% for the finite state grammar and 12% for the bigram LM. There are small but inconsistent differences between \( \hat{h}_{MAP} \) and \( \hat{h}'_{MAP} \). One might expect the more detailed \( \hat{h}_{MAP} \) estimate to be better, except that it requires an iterative solution while \( \hat{h}'_{MAP} \) does not.

The Stochastic Matching test subset results are in Table 5.3. With the finite state grammar, Stochastic Matching does as well as the ML feature transformation result from Chapter 4. When compared to the \( \hat{h}'_{ML} \) result, the use of a covariance transform decreases the word error rate by 2.8%. Results for the full test set using the open-vocabulary bigram language model and the cleaned acoustic model are given in Table 5.4. These results are better than the CMS baseline result but are not always as good as not as good as the feature-space ML results shown the previous chapter. A problem with the current implementation of SM is that the estimate of the covariance transform is slow to converge. Performing more iterations for each utterance could improve the recognition results, although this would increase the cost of the transformation. Also, the parameters are estimated using a unimodal Gaussian model. While the unimodal Gaussian decreases computation time, it does not give
Table 5.2: Bayesian learning results on telephone speech, using the full test set, the open-vocabulary bigram language model, and cleaned acoustic model. Associated feature space transformations are given to show the effects of the covariance transform. The CMS baseline is 84.06%.

<table>
<thead>
<tr>
<th>Percentage Word Accuracy</th>
<th>Model Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Space</td>
<td>Bayesian Learning</td>
</tr>
<tr>
<td>( \hat{h}_{MAP} )</td>
<td>( \hat{h}'_{MAP} )</td>
</tr>
<tr>
<td>84.36</td>
<td>84.43</td>
</tr>
</tbody>
</table>

the best possible estimates. The SM implementation here results in the same word accuracy as the ML channel error estimate shown in Chapter 4. It does, however, improve over the use of the SM mean transformation alone suggesting that either the use of a multiple mixture Gaussian model in the transformation calculations or performing more iterations to estimate the transformations could improve the SM results.

Because Stochastic Matching may require additional iterations to better specify the channel parameters, and because Bayesian learning improved the results for the larger vocabulary task, Bayesian learning appears to be a better way to implement model-space transformations. Only one iteration was used in calculating the SM transformations in this work. Further iterations could improve performance but would increase computation time.
Table 5.3: Stochastic Matching results on telephone speech, using the test subset, two types of language models, and a cleaned acoustic model with associated feature space transformation for reference. The CMS baseline is 90.36% for finite state LM and 86.92% for open-vocabulary bigram.

<table>
<thead>
<tr>
<th>Language Model</th>
<th>Feature Space</th>
<th>Model Space</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\hat{h}_{ML}$</td>
<td>$\hat{h}'_{ML}$ Stochastic Matching</td>
</tr>
<tr>
<td>Finite State</td>
<td>91.08</td>
<td>90.82</td>
</tr>
<tr>
<td>Open Bigram</td>
<td>88.97</td>
<td>88.18</td>
</tr>
</tbody>
</table>

Table 5.4: Stochastic Matching results on telephone speech, using the full test set, the open-vocabulary bigram language model, and cleaned acoustic model. Associated feature space transformations are given to show the effects of the covariance transform. The CMS baseline is 84.06%.

<table>
<thead>
<tr>
<th>Percentage Word Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Space</td>
</tr>
<tr>
<td>$\hat{h}_{ML}$</td>
</tr>
<tr>
<td>84.38</td>
</tr>
</tbody>
</table>
5.4 Comparison of Feature-based and Model-based Techniques

There are two main areas with which to compare feature based and model based techniques: performance and computation time. At this point model modification performance is only slightly better than feature space transformations, so computation issues must be examined. Since computation time is increased for model based adaptation, it is important to examine the task benefits before deciding to use model-based adaptation over feature based. Bayesian learning has the added computation costs of changing acoustic model covariances for each utterance. For a large acoustic model, the additional computation costs may not be practical. For small vocabulary tasks with triphone models, however, the increased cost of Bayesian learning over MAP feature transformations is not significant. In Bayesian learning, the mean transform is the same as the MAP channel estimate and the covariance transform is a by-product of the mean transform calculations. This makes the generation of feature vectors for use in Bayesian learning a simple task. Model modification does improve the recognition results, so it is probably worth the added cost for systems operating in a multi-pass search framework, where a recognition pass with model modification would be a smaller percentage of the overall recognition costs.
Chapter 6

Conclusion

This thesis examined the use of statistical methods for estimating linear telephone channel distortion of speech and for adapting acoustic models of speech to better match specific utterances of telephone speech. This chapter will summarize the key points of the research and the conclude with a discussion of possible future directions.

6.1 Summary

In the introduction of this thesis, four questions were posed. This section will examine the answers to each question as found in this work.

- Does the use of a prior channel distribution, as in a maximum a posteriori (MAP) estimate, improve recognition performance?

The answer to this question is sometimes. There are some cases where the MAP estimate improves the results over the ML estimate, but it is not consistent across all tests. As opposed to other sites, an improvement when using the ML estimate was observed here. In all cases, results are improved when a channel estimate is used. The use of a prior in Bayesian learning leads to better performance than Stochastic Matching for all conditions except the finite state grammar test, but further
study is needed because of simplifications in the implementation of SM. An important additional result of this work is establishing that a unimodal diagonal Gaussian model is a good fit to channel vectors. This prior channel distribution can be used in future work with channel estimation or as a basis for improving the channel distribution.

- Given the are negative results for long utterances, do statistical estimation techniques (ML and/or MAP) out-perform feature mean subtraction in short utterances?

Yes. The use of ML and MAP channel estimates improve the results for the test subset which consists of many short utterances, especially when using an acoustic model trained on data with ML channels removed. The combination of using a cleaned model and ML and MAP channel estimates yields a 15% decrease in percentage word error on a test set predominately comprised of short utterances. This is an important result as many telephone applications, such as activating a credit card, answering a survey, or checking a bank statement, would require short utterances.

- Does “cleaning” the training data improve the performance for either the ML or MAP method?

Yes. In all cases, but particularly for the case of short utterances, improvements were seen in this work when the models were cleaned using one iteration of the feature-space ML channel estimate. The recognized utterances had either an ML channel estimate or MAP channel estimate removed so that there was a better match between the training data and the recognized speech.

- Is there an advantage to modifying the acoustic models, i.e. transforming both means and variances, rather than simply subtracting a constant vector from the features?
Here, the answer is not clear. In almost all cases, model modification using Bayesian learning outperformed feature-space modification using the MAP channel estimate. Moreover, model modification with Bayesian learning gives the best overall results for the more difficult recognition task in this work, i.e. using the bigram language model with a 514 word vocabulary on a test set composed of long and short utterances. However, improvements were not clearly established for SM over ML channel estimation. The computation costs are slightly higher when using a covariance transform but this could be worthwhile for dealing with linear channel distortion in long utterances when working in a multi-pass search framework.

6.2 Suggestions for Future Directions

The Nobel peace prize winner Elie Wiesel once pointed out that in life, answering questions leads to better questions. The point of research and study is not simply to find answers but also to find the true questions. These questions are what draw us to the next step, leading us finally to the core mysteries of the universe, hopefully making some things clearer on the way. While this thesis can hardly claim to accomplish this, it is a start to finding some better questions about the effects of telephone channels on speech and speech recognition. Some of these new questions and directions are discussed below.

In the area of basic feature-space channel estimates, an implementation that estimates separate channel estimates for speech and silence could deal with non-linear distortion of speech, such as that associated with carbon microphones. Sankar and Lee [35] showed that using two estimates, one for speech and one for silence, improved performance. Continued work in establishing prior channel distributions could improve performance, for example, using more data to estimate the parameters. Better channel error estimates might be obtained by performing a second iteration of the channel error estimate.
A better acoustic model would give a more reliable state sequence to estimate the ML and MAP channels. Acoustic model improvements could involve: 1) increasing the number of mixtures in the model, 2) using cross-word triphones, especially for small vocabulary work, and 3) using gender-dependent models. As shown in the baseline results, gender-dependent models do help the general results. Using the gender-dependent acoustic models to estimate the channel may also result in better channel estimates for individual speakers. The model cleaning implemented here was only a one-pass cleaning but recognition accuracy did improve. This result brings up a new question: Would iteratively cleaning the training data, using the first “cleaned” model, further improve the results? Experiments to test different methods of model cleaning would be valuable future work.

Finally, because this work did not attempt to address the issue of channel distortion combined with additive noise, a logical future direction is to look at methods to ease the effects of both types of noise on continuous speech recognition. This work suggests that methods to jointly estimate the additive noise and channel distortion in a maximum likelihood framework are worth examining.
Appendix A

Test Subset Lexicon and Grammar

A.1 Lexicon

The words in the dictionary used for the 530 utterance subset of the Macrophone Natural Numbers development test set are all numbers or linking words. They are: a, and, eight, eighteen, eighty, eleven, fifteen, fifty, five, forty, four, fourteen, hundred, nine, nineteen, ninety, one, seven, seventeen, seventy, six, sixteen, sixty, ten, thirteen, thirty, thousand, three, twelve, twenty, two.

A.2 Finite State Grammar

$\text{posdigit} = \text{one} | \text{two} | \text{three} | \text{four} | \text{five} | \text{six} | \text{seven} | \text{eight} | \text{nine};$
$\text{digit} = \text{oh} | \text{zero} | \text{posdigit};$
$\text{teens} = \text{ten} | \text{eleven} | \text{twelve} | \text{thirteen} | \text{fourteen} | \text{fifteen} | \text{sixteen} | \text{seventeen} | \text{eighteen} | \text{nineteen};$
$\text{umpty} = \text{twenty} | \text{thirty} | \text{forty} | \text{fifty} | \text{sixty} | \text{seventy} | \text{eighty} | \text{ninety};$
$hposdigit = a \mid one \mid two \mid three \mid four \mid five \mid six \mid seven \mid eight
\mid nine;
$hposdigit\_no\_a = one \mid two \mid three \mid four \mid five \mid six \mid seven \mid eight
\mid nine;
$hposteens = eleven \mid twelve \mid thirteen \mid fourteen \mid fifteen \mid sixteen \mid seventeen \mid eighteen \mid nineteen;
$hposumpty = twenty \mid thirty \mid forty \mid fifty \mid sixty \mid seventy
\mid eighty \mid ninety;
$upto99 = $posdigit \mid $teens \mid ($umpty [$posdigit]);
$ones = ([and] $upto99) \mid ($hposdigit hundred [ [and] $upto99 ]);$
$hpos11to99 = $posteens \mid ($hposumpty $hposdigit_no_a);
$ktriad = a \mid $upto99 \mid ($hposdigit hundred [ [and] $upto99 ]);$
$htriad = $upto99 \mid ($hposdigit hundred [ [and] $upto99 ]);\mid
($hpos11to99 hundred);$

$bignumber = (($ktriad thousand) [$ones])|$htriad;

(!ENTER $bignumber !EXIT )
Bibliography


