

Confidence and Rejection in Automatic Speech Recognition

Larry Don Colton

B.S., Brigham Young University, 1976

M.B.A., Brigham Young University, 1978

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

August 1997

© Copyright 1997 by Larry Don Colton
All Rights Reserved

The dissertation “Confidence and Rejection in Automatic Speech Processing” by Larry Don Colton has been examined and approved by the following Examination Committee:

Mark Fanty, Advisor
Research Associate Professor

Ronald A. Cole
Professor

Misha Pavel
Professor

Wayne H. Ward
Research Professor
Carnegie Mellon University

Dedication

To my family: My wife, Lois, who gracefully endured the move to Portland and financial stress of my studenthood. My children, Larissa, Joseph, Benjamin, Daniel, Stacia, and Isaac, who insisted that life go on anyway. My father, Lawrence B. Colton, who set many fine examples. My mother, Jean Colton, who first suggested that I should teach.

To my teachers, advisors, and fellow students: Mark Fanty, Ron Cole, Misha Pavel, Hynek Hermansky, Etienne Barnard, and many others at Oregon Graduate Institute, who patiently reached down and lifted me up.

But most of all, to my God, who granted me with understanding and excited me to this pursuit. I agree with the ancient prophet: “Yea, I know that I am nothing; as to my strength I am weak; therefore I will not boast of myself, but I will boast of my God, for in his strength I can do all things.” — Alma 26:12

Acknowledgements

My time at OGI was supported by a Graduate Research Fellowship grant from the National Science Foundation, and by U S West, Texas Instruments, and the Center for Spoken Language Understanding.

This research was supported in part by a National Science Foundation Graduate Fellowship. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the NSF or the U. S. Government.

Contents

Dedication	iv
Acknowledgements	v
Abstract	xiii
1 Introduction	1
1.1 Research Goals	1
1.2 Male/Female Versus Last Names	2
1.3 Scaling Up: 58 Phrases	4
1.4 Vocabulary Independence	5
1.5 Thesis Overview	6
1.6 Tutorial on Automatic Speech Recognition	7
1.6.1 A Setting for Automatic Speech Recognition	7
1.6.2 Overview of Speech Recognition	8
1.6.3 Artificial Neural Network	12
1.6.4 Context-Dependent Modeling	13
1.6.5 Viterbi Search Example	14
1.6.6 What Can Go Wrong?	17
1.6.7 Wanted: Absolute Probabilities	18
1.7 Definitions Used in This Thesis	19
1.7.1 Speech Recognition	19
1.7.2 Confidence Measurement	23
1.7.3 Algorithm Names	25
1.7.4 Miscellaneous Terms	26
2 Prior Research	29
2.1 Major Sources of Research Literature	29
2.2 Scope of Interest	29
2.2.1 Vocabulary Independence	30
2.2.2 Discriminative Training	30

2.3	Research Results of Interest	30
2.3.1	Logarithmic Averaging	31
2.3.2	Hierarchical Averaging	31
2.3.3	Filler Normalizing	32
2.3.4	Rank-Based Schemes	33
2.3.5	Creative Averaging	33
2.3.6	Role of Perplexity	33
2.3.7	Creation of Probabilities	33
2.4	Confidence Work at Other Institutions	34
2.4.1	Confidence Work at AT&T and Lucent	34
2.4.2	Confidence Work at Verbmobil and CMU	34
2.4.3	Confidence Work at SRI	35
2.5	Conclusions	35
3	Vocabulary-Dependent Experiments	36
3.1	Introduction	36
3.2	The Frame-Based Classifier	38
3.3	Male/Female: Out-of-Vocabulary Rejection	39
3.3.1	Baseline System	39
3.3.2	Second Pass Rejection	40
3.3.3	Results	41
3.4	58 Phrases: Improved Closed-Set Recognition	42
3.4.1	Baseline System	43
3.4.2	Second Pass Rescoring	43
3.4.3	Results	44
3.5	Conclusions	44
4	Vocabulary-Independent Methodology	45
4.1	ANN-based Recognizers	45
4.1.1	Phonetic Units	46
4.1.2	Oct96: Oct 1996 MFCC-based recognizer	46
4.1.3	May96: May 1996 PLP-based recognizer	48
4.2	Performing One Experiment	49
4.2.1	p^r : raw probabilities	49
4.2.2	Hypothesis	50
4.3	Design	50
4.3.1	Scenario	50
4.3.2	Algorithm	51

4.4	Experiments and Trials	51
4.4.1	Density Function for Trues	52
4.4.2	Density Function for Impostors	53
4.4.3	Comparison to OOV Rejection	53
4.4.4	Implications of Random Vocabulary	54
4.4.5	Summary	55
4.5	Corpora	55
4.5.1	Names: OGI Names corpus	57
4.5.2	PhoneBook: NYNEX PhoneBook corpus	58
4.5.3	Corrections	60
4.6	Pronunciation and Word Modeling	61
4.6.1	Worldbet Symbols	62
4.6.2	Orator: word models from Orator TTS	63
4.6.3	CMU: word models from CMU dictionary	63
4.6.4	Wordspotting Grammars	63
4.7	Results: Raw Score Histogram	64
4.7.1	Total Verification Error	65
4.7.2	EER Statistics	67
4.8	Comparing Several Experiments	69
4.8.1	Hypothesis	70
4.8.2	p^n : normalized probabilities	70
4.8.3	$p^n/(1 - p^n)$: likelihood ratio (odds)	71
4.8.4	Mean, Standard Deviation, and Confidence Intervals	72
4.8.5	Distance Chart	72
4.8.6	EER Across Algorithms	73
4.9	Impostors and Perplexity	74
4.9.1	True Words	74
4.9.2	Impostors	74
4.9.3	Perplexity	75
4.10	Recognition by Viterbi Alignment	76
4.10.1	Frames and Words	76
4.10.2	ANN Probability Profiles	76
4.11	Statistical Issues	77
4.11.1	Sampling and Trials	77
4.11.2	Controlling Recognizer Error	77
4.11.3	Out-of-Vocabulary versus In-Vocabulary Recognition Errors	79
4.11.4	Out-of-Vocabulary versus In-Vocabulary Recognition Errors	80

4.11.5	Histogram Creation	80
4.11.6	The ROC Curve	81
4.11.7	Alternatives to EER: MVE and FOM	82
4.11.8	Bootstrap Parameter Estimation	83
5	Baseline Vocabulary-Independent Experiments	85
5.1	Different Corpora	85
5.1.1	An Easier Corpus	86
5.1.2	An Averaged Corpus	87
5.1.3	Conclusions	88
5.2	On-Line Garbage Modeling	89
5.2.1	Estimating the Target Median Rank	90
5.2.2	Initial Knot-point Experiments	92
5.2.3	Dramatically Fewer Knot Points	93
5.2.4	Conclusions	95
5.3	Log Averages	96
5.4	Segmental Averaging	98
5.5	On-Line Garbage Improved	105
5.5.1	Initial Experiments	106
5.5.2	High Ranks	106
5.5.3	Averages of High Ranks	107
5.5.4	Wider Averages	107
5.5.5	Experimental Details	108
5.5.6	Discussion and Conclusions	109
6	Vocabulary-Independent Rank-Based Algorithms	110
6.1	Estimating Probability Three Ways	111
6.1.1	$p(\text{true})$, $p(\text{false})$	111
6.1.2	Cubic Polynomial Smoothing	115
6.1.3	Likelihood Ratio	116
6.1.4	Simple Probability	116
6.1.5	Cumulative Probability	116
6.1.6	Estimating Simple Probability	117
6.2	Probability Training Corpus Selection	117
6.3	Weighted Alternatives to Mean Accumulation	119
6.3.1	Mean Averaging	119
6.3.2	Triangular Averaging	119
6.3.3	Trapezoidal Averaging	120

6.3.4	Parabolic Averaging	120
6.3.5	Usage	120
6.4	Averaging Ranks	121
6.5	Averaging Probabilities	121
6.6	Conclusions	122
6.7	Vocabulary-Independent Final Results	123
7	Confidence	125
7.1	Continuous Versus Discrete	125
7.1.1	Accept, Verify, or Try Again	125
7.2	True Confidence	126
7.2.1	Estimating $p(\text{Impostor})$	127
7.2.2	Estimating $p(\text{True})$	128
7.2.3	Estimating the Likelihood Ratio	130
7.3	Application to a Real-World Problem	130
8	Conclusions	133
8.1	General Conclusions	133
8.2	Noteworthy Points	135
8.3	Future Work	135
	Bibliography	137
	Index	142
	Biographical Note	146

List of Tables

3.1	Utterance Verification Accuracy for 6 Feature Sets	42
4.1	Oct 1996 MFCC-based recognizer ANN Outputs	47
4.2	May 1996 PLP-based recognizer ANN Outputs	48
4.3	Worldbet Symbols	62
4.4	Mean, Std Dev, and Confidence Interval for the p^r Algorithm	67
4.5	Mean, Std Dev, and Confidence Intervals for the p^r Family	71
4.6	Distance Chart for Algorithms in the p^r Family	72
4.7	Differences Across Selected Algorithms in the p^r Family	73
5.1	Differences Across Corpora for Algorithms in the p^r Family	86
5.2	Corpus Differences Change Algorithm Rankings in the p^r Family	88
5.3	Mean, Std Dev, and Confidence Intervals for the $g(a,b,c,\dots)$ Family	93
5.4	Distance Chart for Algorithms in the $g(a,b,c,\dots)$ Family	95
5.5	Distance Chart for Algorithms in the $\log(p^r)$ Family	97
5.6	Accumulation methods pairwise comparison	102
5.7	Distance Chart comparing accumulation methods	103
5.8	Distance Chart for the $\log(p^r/g(low))$ and $\log(p^r/g(high))$ Families	107
5.9	Distance Chart for Algorithms in the $\log(p^r/g(few))$ Family	108
6.1	Distance Chart for Algorithms in the $f^P(R)$ Family	117
6.2	Distance Chart for Algorithms in the $f^M(R)$ Family	118
6.3	Distance Chart for Algorithms in the $f(R)$ Family	118
6.4	Distance Chart for Algorithms in the $f(av(R))$ Family	121
6.5	Distance Chart for Algorithms in the $av(f(R))$ Family	122
6.6	Distance Chart for the Top Algorithms	124

List of Figures

1.1	A Setting for Automatic Speech Recognition	8
1.2	Overview of Speech Recognition	9
1.3	Artificial Neural Network	12
1.4	Context-Dependent Modeling	14
1.5	Viterbi Search Example	15
1.6	Outdid outdid Midafternoon	17
1.7	Wanted: Absolute Probabilities	18
4.1	Illustration of Experimental Design	52
4.2	Histogram for p^r	65
4.3	Various Error Rates for p^r	66
4.4	Annotated ROC Curve	82
4.5	Bootstrap EER distribution for p^r	84
5.1	Distribution Variation across corpora	87
5.2	Histogram of Algorithms, 1: $g(0,2,4,8\dots)$; 2: $g(0,10,20\dots)$; 3: $g(0,2,4,6\dots)$. .	92
5.3	Histogram of Algorithms, 1: $g(4,16)$; 2: $g(0,4,16)$; 3: $g(0,10)$	94
5.4	Distribution Variation for the $\log(p^r)$ Family	98
5.5	Hierarchical Averaging	99
5.6	Histogram variation across accumulation methods	101
6.1	Cumulative Probabilities for Phoneme 21	112
6.2	Variation Among Phonemes in Probability per Rank	113
6.3	Likelihood Ratios for Phoneme 21	115
7.1	Log-Scale Histograms at Various Perplexities	128
7.2	Histograms at Various Perplexities	129
7.3	True Histograms at Various Perplexities	130
7.4	Probabilities from Likelihood Ratios	131

Abstract

Confidence and Rejection in Automatic Speech Recognition

Larry Don Colton

Supervising Professor: Mark Fanty

Automatic speech recognition (ASR) is performed imperfectly by computers. For some designated part (e.g., word or phrase) of the ASR output, rejection is deciding (yes or no) whether it is correct, and confidence is the probability (0.0 to 1.0) of it being correct. This thesis presents new methods of rejecting errors and estimating confidence for telephone speech. These are also called word or utterance verification and can be used in wordspotting or voice-response systems. Open-set or out-of-vocabulary situations are a primary focus. Language models are not considered.

In vocabulary-dependent rejection all words in the target vocabulary are known in advance and a strategy can be developed for confirming each word. A word-specific artificial neural network (ANN) is shown to discriminate well, and scores from such ANNs are shown on a closed-set recognition task to reorder the N-best hypothesis list ($N=3$) for improved recognition performance. Segment-based duration and perceptual linear prediction (PLP) features are shown to perform well for such ANNs.

The majority of the thesis concerns vocabulary- and task-independent confidence and rejection based on phonetic word models. These can be computed for words even when no training examples of those words have been seen.

New techniques are developed using phoneme ranks instead of probabilities in each frame. These are shown to perform as well as the best other methods examined despite the data reduction involved.

Certain new weighted averaging schemes are studied but found to give no performance benefit. Hierarchical averaging is shown to improve performance significantly: frame scores combine to make segment (phoneme state) scores, which combine to make phoneme scores, which combine to make word scores. Use of intermediate syllable scores is shown to not affect performance. Normalizing frame scores by an average of the top probabilities in each frame is shown to improve performance significantly.

Perplexity of the wrong-word set is shown to be an important factor in computing the impostor probability used in the likelihood ratio. Bootstrap parameter estimation techniques are used to assess the significance of performance differences.

Chapter 1

Introduction

Automatic speech recognition (ASR) is the activity of taking in utterances, processing them by computer, and correctly identifying (recognizing) what words were said. Ideally, of course, ASR would do a perfect job of identifying those words. But ASR is not perfect.

Since it falls short of perfection, it would be useful to know when the recognition was likely correct and when it was not. This capability is called “rejection.” Unfortunately even rejection cannot be done reliably. It would be useful to know how likely it is that a given recognition event is correct. This capability is called “confidence.”

In the design and implementation of ASR projects, the availability of an accurate confidence and rejection process would be very useful. Consider the example of a telephone-based system that asks, “Will you accept a collect call from (insert name here)?” and waits for a “yes” or “no.” Because the ASR system is not perfect, one can never be absolutely certain that it has correctly identified the response. But if the system could report that there is 95% certainty that the answer is “yes,” the telephone company’s statisticians and business analysts could decide whether to go along with the answer or not. A “break-even” threshold could be determined in advance, allowing the ASR system to perform useful work despite the uncertainty that remains.

1.1 Research Goals

The goal of this research is to develop new methods of estimating confidence in order to reject errors.

Two major areas are explored in this thesis. The first area is vocabulary-dependent

rejection, where all words in the target vocabulary are known in advance (such as the “yes” and “no” example given above) and a strategy can be developed for confirming each word. The second area is vocabulary- and task-independent open-set rejection, where the words in the vocabulary may be specified at recognition time, and may include new words for which phonetic models exist, but no training examples have previously been seen.

One major challenge is the selection of features used for discrimination between correct recognitions and incorrect ones. There are a number of subsidiary issues (including corpus selection) that are also involved. These are presented in detail later in the thesis.

As an introduction to this thesis the next several sections of the chapter present examples of the research problem, the vocabulary used to discuss it, and some methodological issues.

1.2 *Male/Female Versus Last Names*

The first task in this confidence and rejection research is a simple problem. It involves the two-word vocabulary “male” and “female.” This vocabulary comes up in the context of census-taking. The task is to discriminate between the true words and other words falsely recognized. In particular, the question would be put: “What is your sex, male or female?” When answered with either of those two words, the recognizer has an accuracy of 98.8%. However in an actual census study (Cole, Novick, Fenty, Vermeulen, Sutton, Burnett, and Schalkwyk 1994) 1.6% of the utterances did not contain either target word. The goal is to reject such non-target utterances. This is open-set out-of-vocabulary rejection.

Although a careful explanation of the recognition process is presented in section 1.6, it is useful to briefly introduce it here. The recognizer operates by comparing the actual utterance (digitally recorded) with a computer model of the target word. This comparison results in a score that represents the similarity between utterance and word model. This recognition score (also called the Viterbi score) is computed for each of the word models, and the model with the best score is selected. Note that this approach fails to account for out-of-vocabulary (OOV) pronouncements.

Two speech corpora were used in this research. The gender corpus is a collection of

several thousand actual, valid responses collected in the census study. Because there were few invalid utterances in the gender corpus, another corpus was used to provide impostors. (Informally, this is like a police lineup where the criminal must be identified from a field that includes random people who happened to be available.) The impostor corpus is a collection of persons' surnames (family names). Each utterance from either corpus was forced to be recognized as "male" or "female." Each utterance from the gender corpus was assumed to be correctly recognized. Each utterance from the impostor corpus was considered to be an out-of-vocabulary utterance that had been forced to be (incorrectly) recognized as either "male" or "female." Wordspotting (explained in section 4.6.4) was used to allow recognition within simple embeddings such as "I'm male." These embeddings occurred in 1.4% of the gender responses. Some errors were expected but believed to be so uncommon as to not need attention. These include the 1.2% of gender responses that are incorrectly recognized but treated as though they were correct, and the occasional last name (such as "Mailer") that embeds something recognizable as one of the key words but which would be treated as though they were incorrect.

I hypothesized that two word-specific artificial neural networks, each trained to accept or reject a recognition event, could be used to separate true recognitions from false or out-of-vocabulary ones. The two outputs of each artificial neural network are "confirm" and "deny." Each network is called a "verifier."

Various feature sets were tested, including phoneme¹ duration alone, phoneme center energy alone, PLP coefficients equally spaced through the word, PLP taken at phoneme centers, and PLP from before and after the word. Phoneme centers were especially interesting because I expected that at the central frame the phoneme would be at its most reliable (i.e., recognizable) point. In each case an artificial neural network was trained for the word, yielding confirm/deny outputs.

The most accurate results came from phoneme durations with PLP taken at phoneme centers and 50 msec before and after the word. This achieved a 95.2% accuracy rate when equal numbers of true words and falsely recognized words were evaluated. This

¹See section 1.7 for definitions of this and other terms

confirmed the hypothesis that word-specific neural networks could be used to separate true recognitions from false or out-of-vocabulary ones.

The male/female experiments and results are presented in section 3.3.

1.3 Scaling Up: 58 Phrases

The second research task is to improve the recognition rate on a larger but closed set of words and phrases. Closed-set verification ignores the possibility of out-of-vocabulary utterances. The chosen words and phrases are related to the telephone services industry and include "cancel call forwarding," "help," "no," and 55 others. As before, the recognizer matches the utterance against various word models and develops a score for each. The highest score becomes the putative recognition.

For this task, when the top-scoring recognition was wrong, the true answer was often among the next few choices. The engineering goal was to improve the existing 93.5% recognition rate on 58 words and phrases. This was to be done by selecting the correct answer from among the top three choices returned by the recognizer. The research goal is to evaluate the male/female approach of training a separate verifier for each word, not just against the out-of-vocabulary option, but as an indicator of relative confidence in each recognition.

I hypothesized that word-specific neural networks, each trained to accept or reject a recognition event, can be used to evaluate the relative confidence of in-vocabulary alternatives better than the original Viterbi recognition scores do.

To explain why this might be, it is useful to briefly introduce a few more aspects of the recognition process. Recognition scores are computed with an equal contribution from each "frame" of the utterance. For recognition each utterance is divided into frames of fixed duration and each frame is recognized separately. Then the recognition results for the frames are strung together to match the target word model. Although this method is efficient and gives good results, it can be fooled in various ways and I thought that taking a more careful look at each of the top contenders might give a more accurate ranking.

Building on the previous research, 58 individual artificial neural networks (one per

word or phrase) were constructed, each giving confirm/deny outputs. As before, each artificial neural network took as input the phoneme durations and PLP taken at phoneme centers and ± 50 msec from the word based on the recognizer output. The top three contenders were each evaluated by their individual artificial neural networks, and a winner declared based on the original ranking and the newly computed scores. The recognition rate improved to 95.5%, which is a 30% reduction in the error rate (from 6.5% to 4.5%).

This confirmed the hypothesis that word-specific artificial neural networks could be used to measure relative confidence of in-vocabulary recognition alternatives.

The 58-phrase experiments and results are presented in section 3.4.

1.4 Vocabulary Independence

The two-word and fifty-eight-phrase experiments provide background leading up to the major research task, which is to study confidence and rejection on the set of all possible words. Creating word-specific artificial neural networks is not feasible, so an alternative was sought. The hypothesis is that confidence and rejection can be based on the set of phonemes from which word models have been defined and on which recognition itself is based.

The advantage of dealing with all possible words is that new words can be added to an “active vocabulary” (those words potentially recognizable at a point in time) without extra training. (Vocabulary-dependent systems generally require special training on each individual word of the vocabulary. This results in higher accuracy rates than a vocabulary-independent system can achieve, but does not easily accommodate previously unknown words. For speaker-independent systems, such training can require many samples of each utterance from a variety of talkers.) It becomes possible to create, for example, a robotic telephone attendant for an automatic voice-response-based switchboard that can connect incoming calls to a person based on an utterance of the person’s name. That is, the caller would say the name rather than spelling it on the touch-tone pad. This can be made to work even for calls to the new persons that have recently joined the staff of the organization and may have been unknown in the system a day before. To minimize the

number of wrong connections in such a system it is necessary to have a confidence measure for each recognition and a dialogue manager that confirms or re-prompts in the case of low-confidence recognitions.

The present research is primarily an out-of-vocabulary or open-set rejection study. Random-vocabulary systems such as those used in this research ignore a large amount of vocabulary dependence present in many authentic recognition settings. Language modeling has long been known to provide a substantial improvement to recognizer performance. Vocabulary-independence restricts the availability of any language modeling benefits. Further performance improvements can be expected in vocabulary- and task-dependent situations.

My previous research also took advantage of phonemes by looking at characteristics at the center of each phoneme, and the duration of each phoneme. This new research broadens the scope to treat transitional parts of phonemes (i.e., states within phonemes) as separate entities. That is, in the word “fox” the first part of the /ah/ sound is “colored” by the fact it is following an /f/. It differs from first part of /ah/ as seen in “cox.” By identifying up to eight different transitions into and out of each phoneme, the total number of phonological segment types used in these experiments comes to 544.

1.5 Thesis Overview

The experiments summarized above provide a general sense of the content and direction of the thesis. The following sections in this chapter give a tutorial introduction to automatic speech recognition, and definitions of terms used throughout the thesis.

Chapter 2 reviews prior research that is related to confidence and rejection. Chapter 3 examines vocabulary-dependent utterance verification, and reports the experiments with vocabularies of two and fifty-eight words.

In chapter 4 the scope is broadened to examine vocabulary-independent measures of confidence and rejection. It covers general and methodological information, such as the overall experimental design and a description of the corpora that are used. Each section of chapter 5 studies a technique known in the research literature, pushing it to peak

performance for comparative purposes. For each experiment it tells the motivation and results and provides some discussion and conclusions.

Totally new research is reported in chapter 6, which examines the area of rank-based probability estimation. Chapter 7 completes the discussion of rejection by developing an actual confidence score that can be used to guide higher-level decisions about dialogue processing. Chapter 8 presents overall results, discussion, and conclusions.

1.6 Tutorial on Automatic Speech Recognition

For the benefit of the reader who is less familiar with speech recognition as well as the reader who may be familiar with different terminology than is used in this thesis, it is useful to present a tutorial introduction to speech recognition. This section does not give a rigorous and carefully referenced treatment. Such a textbook presentation is beyond the scope of this thesis, and the interested reader is referred to Rabiner and Juang (1993), Deller, Proakis, and Hansen (1993), or other fine textbooks.

The purpose of this presentation is simply to establish and illustrate the methodology of speech recognition, since it is helpful to have a concrete and moderately detailed understanding of the process, even when those details go beyond what would be required to understand the rest of the thesis. Accordingly the presentation is approximately true; less important details may be overlooked and simplifications are made in the interest of giving a good first approximation to the speech recognition process.

Figures 1.2, 1.3, 1.4, and 1.5 were created by John-Paul Hosom and are used by his kind permission. They are part of a tutorial on the WWW.²

1.6.1 A Setting for Automatic Speech Recognition

Figure 1.1 illustrates automatic speech recognition as a process involving several parts. First there is the talker or speaker who is producing the utterance. In this illustration, the utterance is captured by a telephone handset. Next there is the speech recognition system that is connected to the telephone system and is recording the utterance. It performs

²At <http://www.cse.ogi.edu/CSLU/toolkit/documentation/recog/recog.html>, as of July 1997.

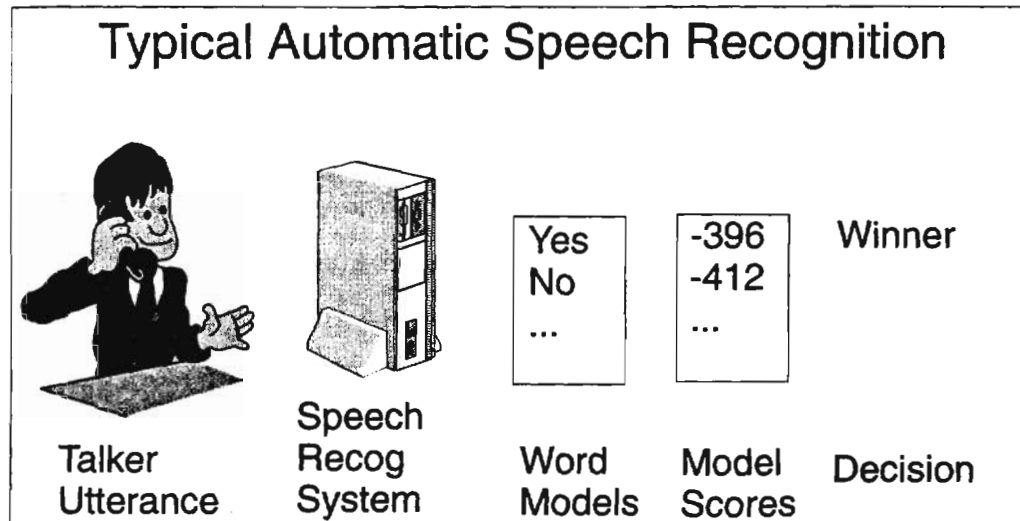


Figure 1.1: A Setting for Automatic Speech Recognition

an analysis of the utterance and compares it to the various word models in its active vocabulary. There may be other words and word models that are known by the ASR system which are not included in the active vocabulary because they do not represent expected inputs at this point in time. The word models are indicated by the words “Yes” and “No.” The full model includes an actual string of phonemes (a pronunciation) that must be present in the utterance for recognition to occur. For each of these word models a score is computed that reflects the goodness of the match between the utterance and the word model. The method for calculating the score is given below. All word models, even wrong models, will create some score. Finally a decision is made and the model that achieves the highest score is “recognized” as the (putative) winner. The following figures illustrate these steps in greater detail.

1.6.2 Overview of Speech Recognition

Figure 1.2 gives an overview of automatic speech recognition.

Upper Left: The utterance is transformed by the telephone (or microphone) into an electric signal. The waveform of the utterance “two” shows the electrical signal from the

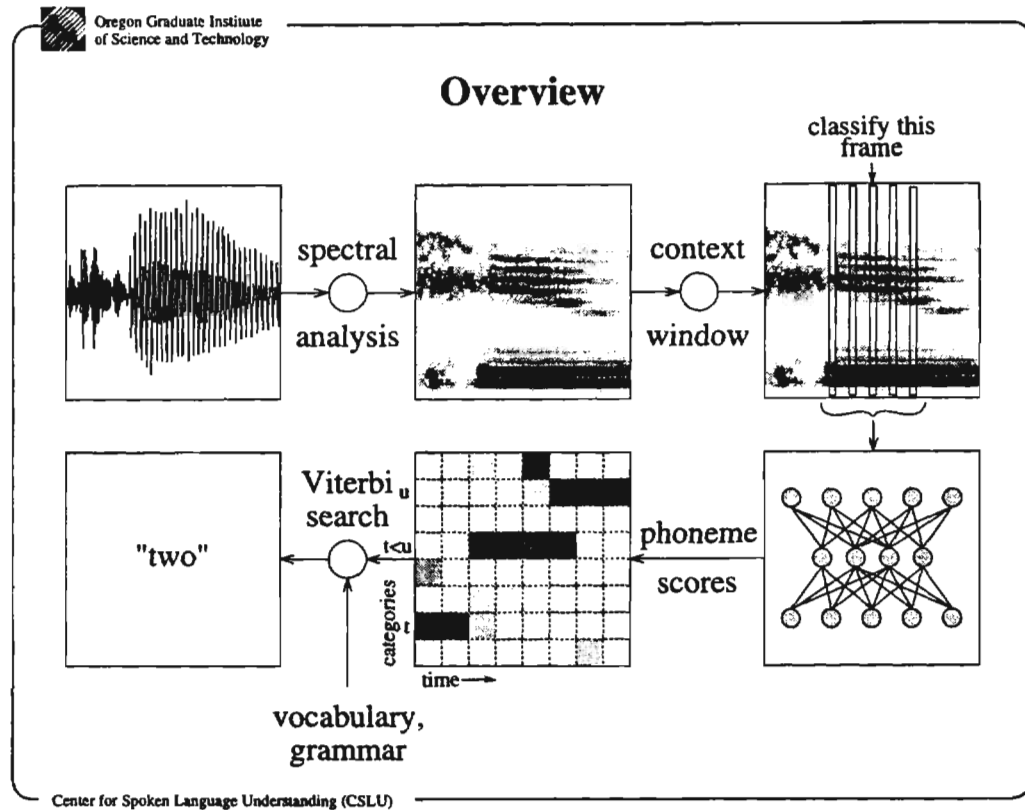


Figure 1.2: Overview of Speech Recognition. Used by permission.

microphone as a jagged line. Each corner represents the voltage that was present at a particular moment in time. The samples are taken each $\frac{1}{8000}$ of a second. Two types of error are introduced at this stage. First the sampling process does not record a continuous history of the original signal, but only takes samples (snapshots) at certain points in time. This is called discretization or sampling error. The second error is quantization error. Each sample is represented as an eight-bit number. To do this the entire range of possible voltages is divided into 256 ranges and each range is assigned a number. At playback time these numbers are converted into a typical voltage for that range. The actual voltage value is lost. These errors do not seem to be serious problems because the samples occur often enough and have a fine enough quantization. Nearly everyone is familiar with the fact that television and movies are produced as a sequence of still frames and that playing them in sequence at a rate of about 30 frames per second gives the appearance of smooth

motion; also that discrete cones and rods in the eye capture the video image so light falling between the receptors is lost. This same effect seems to apply to speech, although the speech sampling rate of 8000 per second is much higher than the video sampling rate, and the cochlea of the inner ear uses fine hairs to detect different frequencies in sound, together with their amplitudes, rather than cones and rods.

The waveform is next divided into frames (not shown), typically 10 msec in length. In some research the frames actually overlap. The step size is the number of samples or msec between adjacent frame starts. The frame size is the number of samples or msec within each frame. For this research the step size and frame size were both 10 msec (80 samples). Durations and steps as short as 3 msec or as long as 30 msec are used by some researchers.

Upper Middle: The waveform frames are passed through a spectral analysis, yielding a spectrogram. The spectrogram shows where the individual frames have been converted into a spectral representation using some kind of Fourier transformation. Dark bands are shown as energy concentrations or resonances in the simulated spectrogram. Linear Predictive Coding (LPC), Perceptual Linear Prediction (PLP) and Mel-scale Frequency Cepstral Coefficients (MFCC) are typical spectral transformations. A set of spectral features is created for each frame.

Experimental measurements suggest that human perception of frequency is not linear, but is roughly linear below 1000 Hz, and roughly logarithmic above. Perceptual LP and Mel-scaling account for this.

The cepstrum is the inverse Fourier Transform of the logarithm of the absolute value of the Fourier Transform of the signal. It is used to separate the speech signal into the glottal pulse train (roughly pitch) and the vocal cavity resonances. The word cepstrum is just the word spectrum with the first four letters reversed, suggesting that the two meanings are nearly the same. In the case of cepstrum the taking of the logarithm provides the significant additional “twist” for which the letters are reversed.

The resonances (called formants) shown correspond to different parts of the vocal tract. These parts include the space from the larynx to the back of the tongue, the space from

there to the hump or tip of the tongue, and the space from there to the teeth or lips. It can be helpful to think of a trombone or a child's whistle that produces different pitches by moving a rod in or out. As the tongue moves in the mouth, the vocal cavities change shape and size resulting in a changed set of resonances. All these simultaneous resonances taken together produce a composite sound that is perceived as linguistic or phonetic.

The lowest formant is called f-zero (written F_0 or $F0$). It corresponds to the glottal pulse frequency and is generally perceived as the pitch of the speech. Higher formants are numbered from f-one and up, and correspond to the resonances from lowest frequency to highest. Formants through f-three are generally transmitted in telephone speech. Higher formants exist and are transmitted in radio, television, cinematic, and CD-music speech. The higher formants seem less important for speech recognition but contribute to speaker identification and recognition of whether for instance the speaker has a cold.

Upper Right: Context windowing is the process of selecting information from the spectrogram to be used in the phonemic classification. In this case, information from five frames located at offsets of -6, -3, 0, 3, and 6 from the frame to be classified is used to identify the phoneme represented in the central of those frames.

Lower Right: A neural network converts these features into classification estimates, which are probability estimates, telling how likely the frame is to be an example of some phoneme. Each neural network output gives the probability for a different phoneme.

Lower Middle: The phoneme scores are shown in a grid where the darkness of each cell indicates the probability of that phoneme at that timeframe. Dark cells are high-probability classifications. Viterbi search is used to compare various vocabulary possibilities against this grid. Grammars can be used to control the sequence of words recognized.

Lower Left: The Viterbi search results in a putative recognition. In this case the recognition is the word "two."

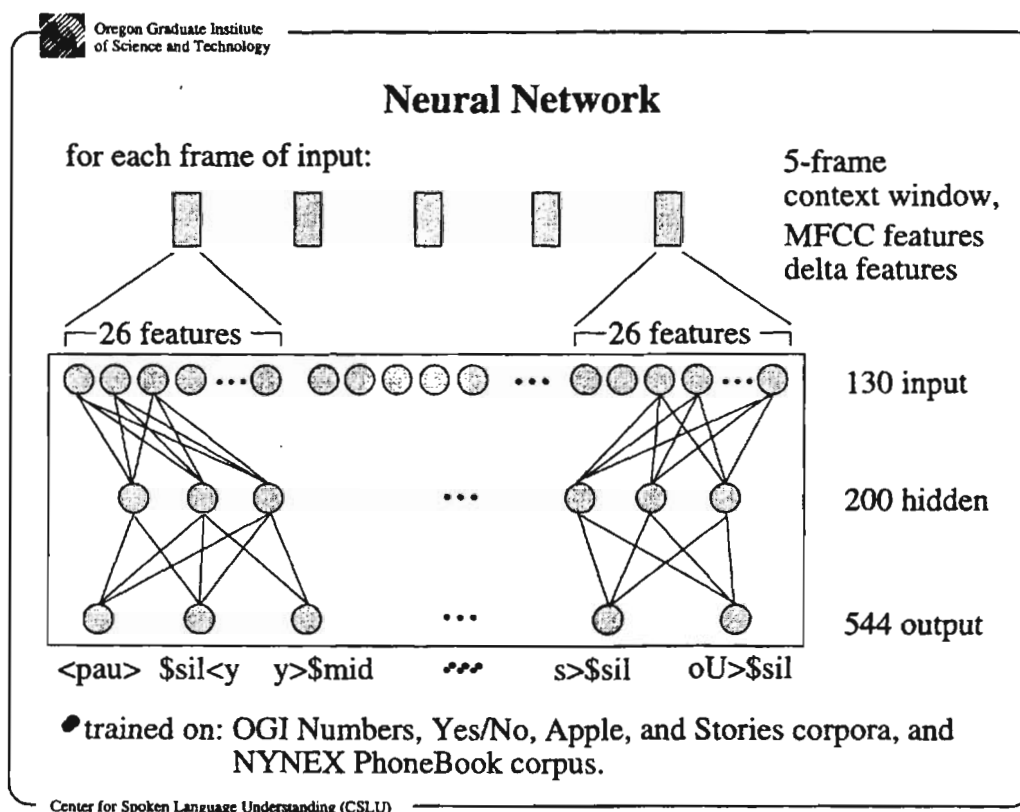


Figure 1.3: Artificial Neural Network. Used by permission.

1.6.3 Artificial Neural Network

Figure 1.3 illustrates the artificial neural network that is used in the recognition process. For each frame of input, a 5-frame context window is used, with frames offset -6, -3, 0, 3, and 6 from the current frame. The features include MFCC features and delta (first-order difference) MFCC features at the rate of 26 per frame. The first 12 are mfcc coefficients for that frame of speech, and with those is one energy coefficient indicating the amount of energy in the signal at that frame. Energy is the RMS (root mean squared) value for the samples in the frame. It is computed by squaring all the sample values, summing the result, and taking the square root.

The other 13 numbers are called delta coefficients and give the difference between the mfcc and energy in this frame versus the previous frame. That is, the delta values tell

how much the coefficients have changed. The particular input features used will possibly differ from recognizer to recognizer.

This totals 130 inputs (plus one hard-wired to a constant value of 1.0). The neural network has 200 hidden nodes and 544 outputs. Table 4.1 on page 47 lists the outputs.

Five frames of feature values are used in building a 130-number feature vector for input to the neural network. With each frame occupying 10 msec of the input speech, the total window is 130 msec, about 1/7 of a second. Half of that is involved in look-ahead. That is, the ANN does not make a decision about the current frame until it has seen the next six frames after it.

The conversion of input features into phoneme probabilities progresses on a frame-by-frame basis. On the top of the diagram the input features are indicated. Each input value is a real-valued number. Conceptually it is loaded into a node in the top row. To compute the value for a node in the center row, each top-row value is multiplied by a weight. The results are added together and then run through a sigmoid function to limit the values to the range -1 through 1. The weights correspond to the lines that connect the nodes. If there are n inputs and m hidden nodes, there will be $n \times m$ weights between those two layers in a fully-connected neural net architecture. In this way the values of all the nodes in the hidden layer are computed. Then the bottom row is computed from the hidden-layer values and the next set of intervening weights.

The values in the final layer are outputs. In this figure there are only three layers. Each output corresponds to a single phoneme or to a single phoneme state (half or a third of a phoneme). Taking the first output, /pau/, for example, the value there might be .34. This would indicate that the inputs provided have about a 34% chance of representing the /pau/ (pause or silence) phoneme. A chart showing the phoneme set appears on page 62.

1.6.4 Context-Dependent Modeling

Figure 1.4 examines context-dependent modeling. Each context-dependent phoneme is divided into one, two, or three parts. For example, the word “yes” is given as three phonemes: /j E s/. (/j/ is the Worldbet symbol for the “y” sound. See Table 4.3 on page 62

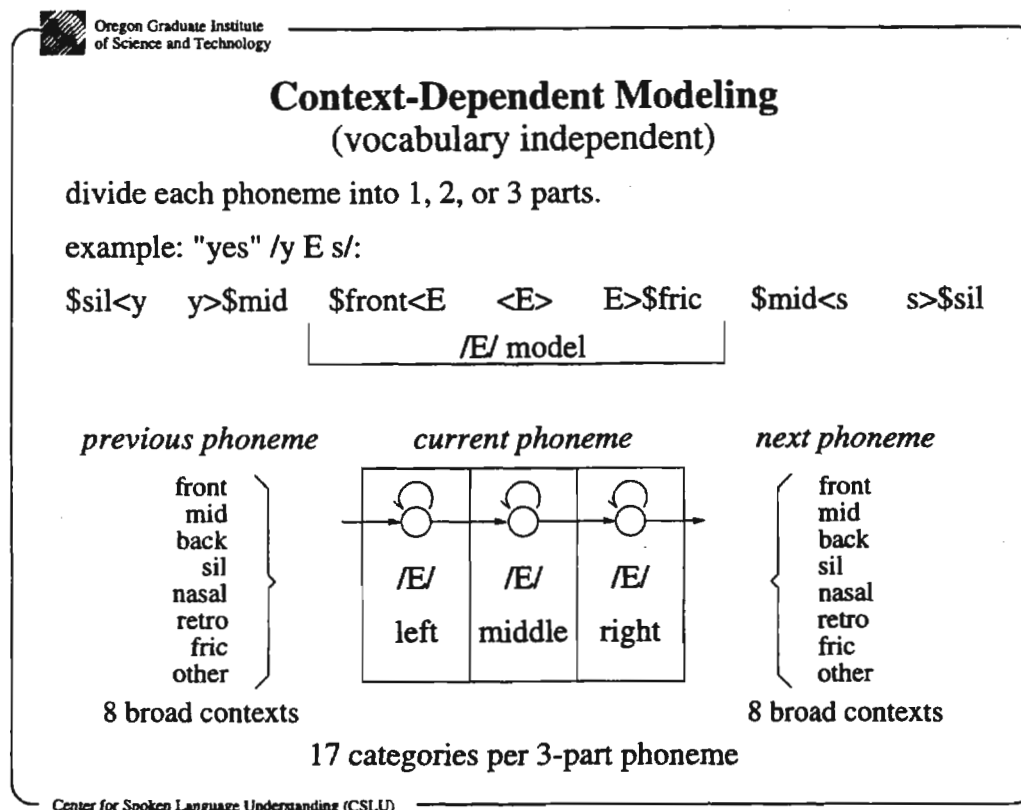


Figure 1.4: Context-Dependent Modeling. Used by permission.

for a presentation of those symbols.) The /j/ phoneme is divided into two parts: j-after-silence and j-before-mid-vowel. The /E/ phoneme is divided into three parts: E-after-front-vowel, central-E, and E-before-fricative. The /s/ phoneme is divided into two parts: s-after-mid-vowel and s-before-silence. Including mid-vowel, front-vowel, and fricative, there are eight broad contexts with which to identify the previous phoneme and the next phoneme. This list of contexts can vary by phoneme for maximum usefulness.

1.6.5 Viterbi Search Example

Figure 1.5 illustrates two search paths created by Viterbi search. In this example, the search paths for "yes" and "no" are shown, on a field of thirteen ANN outputs and twenty speech frames. In each cell, the darkness indicates the probability contribution from that cell. The Viterbi algorithm searches through the cells to find the path with the highest

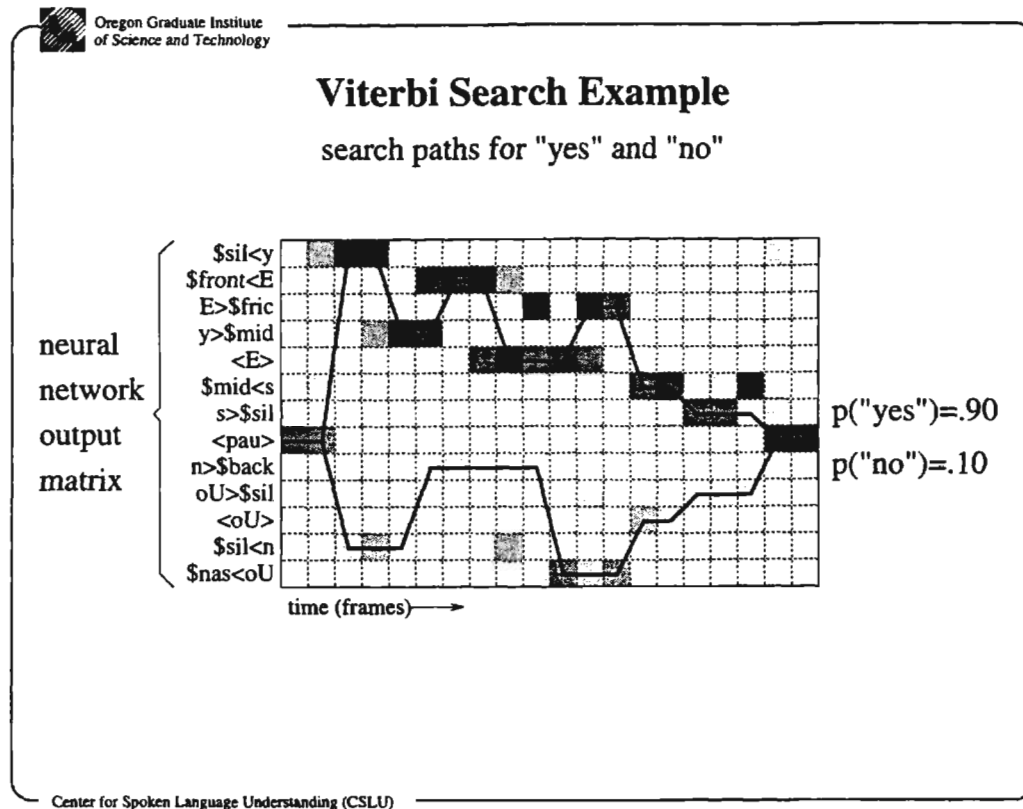


Figure 1.5: Viterbi Search Example. Used by permission.

probability. This is the path with the darkest cells. In this figure, the phonetic segments of "yes" use 2, 2, 2, 3, 2, 2, and 3 frames respectively. Similarly the segments of "no" use 3, 5, 3, 2, and 3 frames respectively. It can readily be seen that the cells in the "yes" path tend to be darker than those in the "no" path. This is further reflected in the final probabilities, with "yes" being 90% probable, and "no" being 10% probable. The first and last two frames are assigned to /pau/, which represents pause or silence. These four cells are permitted or required by the grammar, but are not part of the word model for either "yes" or "no."

The rule for aligning the model to the utterance is this: each frame must be used. The first frame in the utterance must belong to the first phoneme in the model. The last frame must belong to the last phoneme. Between the frames must be assigned to the phonemes in the same order as the phonemes appear in the model.

There are many ways the frames could be assigned to the phonemes. Each way has its own score. There are a finite number of ways, yielding a finite number of scores, of which one is highest (more than one in case of ties). The best-scoring alignment is saved, along with its score.

The number of alternatives may seem quite large. Fortunately there is a dynamic programming solution to this problem that finds one of the optimal alignments in linear time. That means if the length of the utterance doubles, then it will take twice as long to find the optimal alignment. This is much better than the exponential explosion in running time that occurs using some alternative search algorithms. The dynamic programming algorithm used in cases like this is the Viterbi algorithm. (The Viterbi algorithm computes the score in cell(i,j) as the maximum score from all transitions from cell($i-1,k$). This yields the best alignment up to any point in the word model, given the ANN outputs seen so far. It avoids exponential explosion by keeping only the best path to each cell(i,j).) Other algorithms exist that are nearly as efficient but experience has shown that Viterbi performs as well as these competitors.

In an actual recognition attempt there are normally several word models that are evaluated. In this example, the models for “yes” and “no” are evaluated. There can be many more such models. Each will have a best alignment and a score to go with it.

The actual computation of Viterbi scores, such as -396 for “yes” and -412 for “no” is done by converting frame probabilities into their logarithms and then adding along the path that gives the highest score. The use of logarithms serves two purposes. First, logarithms of probabilities can be added rather than being multiplied. On some computers addition is faster than multiplication. Second and more importantly, because all the probabilities are smaller than one, when a large number of them are multiplied together the result can be very very near zero, with many zeroes in front. This can result in “underflow” in the computer, where the number becomes so close to zero that the computer does not represent it accurately and makes the answer zero, or stops and complains that the numbers are getting too small to work with. When logarithms are added the number becomes negative but it does not underflow. Remember that this is not a fundamental part of the alignment process, but is an efficiency trick that is commonly employed.

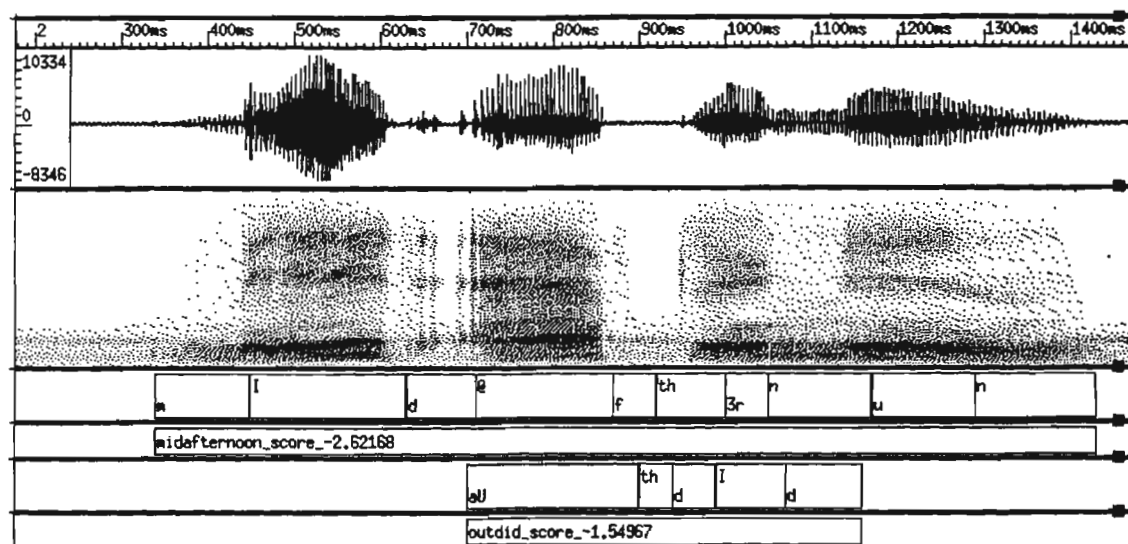


Figure 1.6: Outdid outdid Midafternoon: This example shows a misrecognition. The top portion shows the waveform running from the 300 msec point to the 1400 msec point. The middle part shows a spectrogram for the same time range. The bottom of the spectrogram indicates frequencies near zero. The top indicates frequencies near 4000 Hz. The next two lines show the correct recognition (“midafternoon”) together with its division into phonemes. The last two lines show an incorrect recognition (“outdid”) together with its division into phonemes.

1.6.6 What Can Go Wrong?

Figure 1.6 illustrates a recognition gone awry. The utterance is “midafternoon,” and the correct model is m I dc d @ f tc th &r n u n. This model achieves a recognition score (average frame score) of -2.6. Another word in the vocabulary, “outdid,” with a word model of aU tc th dc d I dc d achieves a recognition score of -1.5, which is higher than the true score. This results in a misrecognition.

The spectrogram shows a steady hum in the lower frequencies. This probably contributes to poor average recognition across the entire utterance. In addition, it is just the middle of the word that is misrecognized. The actual recognition might be closer to “mid-outdi-doon.” This is an example of a misrecognition that is difficult to identify and reject. (This particular recognition error is discussed further in Figure 4.1 on page 52.)

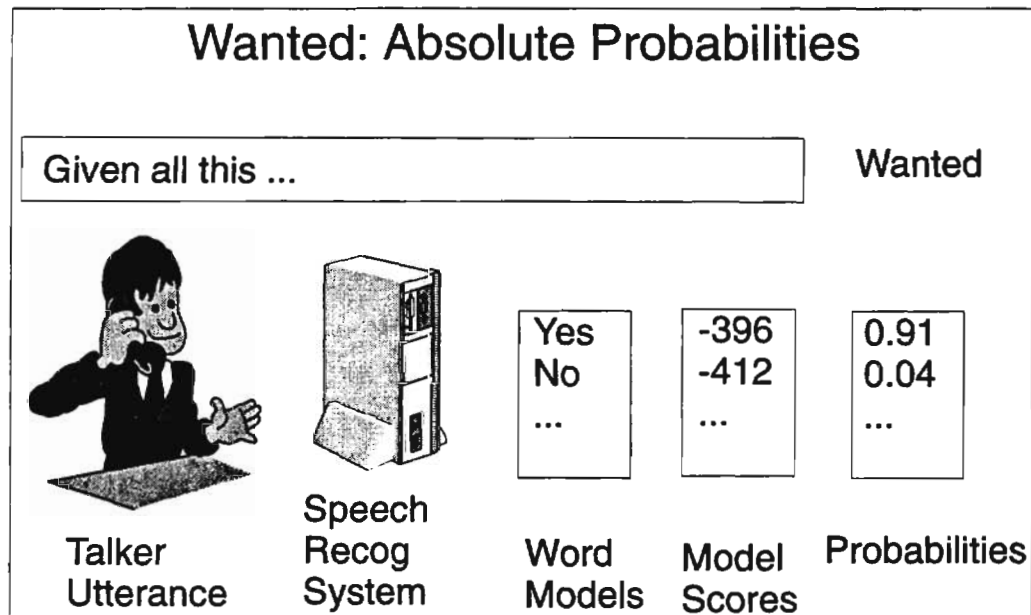


Figure 1.7: Wanted: Absolute Probabilities

1.6.7 Wanted: Absolute Probabilities

Figure 1.7 points out that a result like “yes scores best” or “no scores -412” is not the most desirable result. The meaning of -396 or -412 is not clear. Certainly one is better than the other. But can we tell that either is right? Perhaps both are wrong together.

Instead it is desirable to know that with the inputs provided (that is, the utterance from the talker, and the word models stored in the speech recognition system), there is a 91% chance that “yes” is correct, and a 4% chance that “no” is correct. The remaining 5% would represent the total chances for other vocabulary if any, plus the chance that the talker did not say any of the words in the vocabulary. Probabilities like these can be used in real settings such as the “collect call” example given at the start of this chapter.

Such percentages as these can be computed by first recognizing and scoring a large number of sample utterances called a training set. From the behavior of this group it can be determined that when some particular raw score is seen there is a .91 chance that it is correct and a .09 chance it is wrong. The exact method of doing this is given in chapter 7.

1.7 Definitions Used in This Thesis

This section contains a list of terms and their definitions as used in this thesis. A small amount of discussion is provided where it seems appropriate. Terms are presented in a conceptual order appropriate for direct reading, and are also mentioned alphabetically in the index.

1.7.1 Speech Recognition

utterance: Something said or uttered. Sound itself. In the context of this thesis, an utterance is meant to be recognized as one or more words.

wavefile: A collection of amplitude samples from a digital recording of an utterance (or other sound).

ASR: Automatic speech recognition. This is speech recognition performed by mechanical means, especially that done by computers. Generally it matches an utterance to a word model, where the utterance is given and the word model must be selected from a list called the active vocabulary.

ANN: Artificial neural network. A method sometimes used in ASR for computing the probability that a particular frame of speech represents a particular phoneme.

frame: A unit of input signal typically 10 msec in duration. The microphone inputs occurring during a frame are converted together to give a spectral representation of that signal.

HMM: Hidden Markov model. A statistical model of the acoustic production of speech used for recognition. States generally represent phonemes or portions of phonemes.

phonetic units: Units of speech based upon phonemes. The recognizer used for the research reported in this thesis uses phonemes, phone halves, and phone thirds as its units of recognition.

phoneme: A phoneme is a simple sound in some language (in this case English) that is used to distinguish between words. The various vowel sounds in “bead,” “bid,” “bed,” “bad,” “baud,” “bode,” “booed,” and “bud” are each identified by a different phoneme. Diphthongs, such as the vowel sounds in “cute,” “kate,” “kite,” “coat,” “couch,” and “boy” are each generally identified as single phonemes. The /k/ sounds in “king” and “kung” are somewhat different but are generally identified in English as being examples of the same phoneme (that is, allophones of the same phoneme). Some might argue whether there is a significant (i.e., phonemic) difference between the vowels in “suit” and “boot” or “caught” and “cot.” A phoneme chart appears on page 62.

allophone: A phoneme variant. A classic example is the /t/ in “struck” versus “truck.” Following /s/ the sound of /t/ is different than in a word-initial position. Although these sounds are acoustically distinct, they are not generally distinguished by native speakers of English.

phone state: A phoneme may be modeled as a sequence of one or more states. The word “state” is used in its normal meaning in the subject of automata, state machines, regular expressions, and the like. Each state in an ANN recognizer corresponds to one of the ANN outputs. A sequence of frames that are assigned to the same state are called a segment in this thesis.

phone halves: This is a phone state in a phoneme model having two states. The durations need not be of equal length.

phone thirds: This is a phone state in a phoneme model having three states. The durations need not be of equal length.

CI: Context independent. Generally refers to a phone state such as the middle third of /A/ that is modeled independently of the phoneme that occurs before or after it.

CDL: Context dependent toward the left. Generally refers to a phone state such as the first third of /A/ that is modeled differently depending on the phoneme that occurs just before it in time (to its left in English orthography).

CDR: Context dependent toward the right. Generally refers to a phone state such as the last third of /A/ that is modeled differently depending on the phoneme that occurs next after it in time (to its right in English orthography).

word model: A phonetic model for a word (or phrase). The model for the word “yes” could be /j E s/, where /j/, /E/, and /s/ are three phonemes from the Worldbet phonetic alphabet.

active vocabulary: The set of word models that will be compared to an utterance. The goal of the comparison is to find one that matches best.

best: In the context of active vocabulary, the best match is the vocabulary word that gets the highest recognition score. The Viterbi algorithm is used to find the highest-scoring alignment between a word model and the utterance to be recognized. Among all word models considered, the model with the highest score is “best.” This highest-scoring word model is the putative recognition. It is either a true recognition (if correct) or an impostor (if incorrect).

putative: Hypothesized. A putative recognition is a recognition that has been hypothesized but not yet fully verified, accepted, or rejected. It is under consideration.

wordspotting: Given the speech signal and the particular target word to be found, wordspotting scans the entire signal and finds locations where the target word might appear.

grammars: A grammar specifies the sequence of words that can be decoded from an utterance. Each word is defined by its word model. A simplified grammar for time might

run something like this: (hour) (minute or o'clock) (AM or PM), where hour is a number from one to twelve, and minute is a number from one to fifty-nine.

filler model: In wordspotting and similar applications, the entire utterance is accounted for by assigning each frame to a part of some word model. The filler model is a special word model that accounts for any sequence of phonemes. Given a grammar of (filler) (hello) (filler), some frames will be assigned to the word “hello” and frames before and after will be assigned to the filler model.

any model: Another name for the filler model.

garbage model: Another name for the filler model.

Worldbet: A phonetic alphabet designed by Jim Heironymus. It is presented in detail in Table 4.3 on page 62.

perplexity: For this research, perplexity is defined as the number of words in the active vocabulary when an out-of-vocabulary utterance is encountered. In general this is the branching factor by which the number of possible alternatives grows across time (without pruning).

pruning: The process of removing some branches from the search tree based on their low scores and extremely small likelihood that they will eventually be found to be part of the best-scoring interpretation of the utterance. Careful pruning can reduce the running time of the search algorithm from exponential time down to linear time without introducing too much error.

closed-set rejection: In-vocabulary rejection. Section 3.4 deals with this type of rejection. The rest of the thesis is concerned mainly with open-set rejection.

rejection: The decision that a putative recognition is wrong.

IV: In-vocabulary.

in-vocabulary: A situation where the true word model for recognition is guaranteed to be in the active vocabulary.

open-set rejection: Out-of-vocabulary rejection.

OOV: Out-of-vocabulary.

out-of-vocabulary: The situation where the correct word model for an utterance is not among those word models considered (the active vocabulary). Hence the utterance is out-of-vocabulary.

impostor: When an utterance is out-of-vocabulary with respect to the active vocabulary, but a best match is selected anyway, that best match is called an impostor. Generally any of the incorrect words in the active vocabulary may be called impostors, but the best-scoring such word is particularly called the impostor.

impostor score: The score assigned to an incorrectly recognized utterance, especially the score assigned to the impostor.

true recognition: The recognition of an utterance as a particular word model, when that recognition is correct.

true score: The score assigned to a correctly recognized utterance.

1.7.2 Confidence Measurement

confirm: To state that something is true. To accept a putative recognition.

deny: To state that something is false. To reject a putative recognitions.

verifier: An algorithm that evaluates a putative recognition and identifies it as correct or wrong. Also such an algorithm that assigns a confidence score to a putative recognition.

confidence: The probability of being correct.

alpha error: Also α error. The error of rejecting (denying) something that is actually true. Also called Type I error.

Type I error: The error of rejecting the truth. Also called alpha error.

beta error: Also β error. The error of accepting (confirming) a falsehood. Also called Type II error.

Type II error: The error of accepting a falsehood. Also called beta error.

EER: Equal error rate.

equal error rate: There is an operating point at which the Type I and Type II errors are equal. The rate of Type I or Type II error occurring at that point.

FOM: Figure of Merit.

figure of merit: The area beneath the ROC curve. For a perfect verifier it is 1.0. For a random verifier it is 0.5.

TVE: Total verification error.

total verification error: The sum of the Type I and Type II errors at a given operating threshold.

MVE: Minimum verification error; minimum TVE.

ROC curve: Receiver operating characteristics curve, showing the performance tradeoff between signal and noise as the reception threshold is varied. In the context of verification schemes, it shows the proportion of errors remaining and the proportion of correct recognitions remaining.

1.7.3 Algorithm Names

p^r : Raw probability. p stands for probability, and the superscript r indicates the type of probability, which is “raw.”

p^n : Normalized probability: all raw probabilities for the same frame are summed, and the sum is divided into each of the raw probabilities. This results in normalized probabilities, the sum of which is 1.0.

$p^n/(1 - p^n)$: Odds. The normalized probability of success divided by the normalized probability of failure.

fw: Frame, word hierarchical averaging. This is the degenerate case and could also be called non-hierarchical averaging.

fpw: Frame, phoneme, word hierarchical averaging.

fsw: Frame, segment, word hierarchical averaging.

fspw: Frame, segment, phoneme, word hierarchical averaging.

fspsw: Frame, segment, phoneme, syllable, word hierarchical averaging.

geometric averaging: The n th root of the product of n quantities. Can be implemented as the anti-logarithm of the average of the logarithms of the quantities in question.

likelihood ratio: Odds.

odds: The number of successful (or correct) instances divided by the number of unsuccessful (or incorrect) instances of some type of event.

knot points: Points at which two segments of a piecewise linear model come together.

1.7.4 Miscellaneous Terms

TTS: Text to speech synthesis. TTS is used in this research to generate word models for those words not present in a dictionary of “hand-generated” word models. Hand-generated models are those written by humans on a word-by-word basis.

Bayes rule: A statement of conditional probabilities, $p(a|b) = \frac{p(b|a)p(a)}{p(b)}$. It is easily derived from $p(a|b)p(b) = p(a, b) = p(b, a) = p(b|a)p(a)$.

ARPA: (US) Advanced Research Projects Agency, also:

DARPA: (US) Defense Advanced Research Projects Agency.

DoD: (US) Department of Defense.

NSF: (US) National Science Foundation.

DEC: Digital Equipment Corporation.

NYNEX: New York New England telephone company.

CSLU: Center for Spoken Language Understanding at OGI.

OGI: Oregon Graduate Institute.

CSLUrp: CSLU rapid prototyper.

CSLUsh: CSLU shell; a collection of Tcl procedures in the CSLU Toolkit.

toolkit: The CSLU toolkit, which includes CSLUrp and CSLUsh, and is designed to make easier research and construction of computer-based speech systems.

comp.speech: A newsgroup available on the Internet. This newsgroup publishes a FAQ each month listing many resources of interest in the field of computer speech recognition.

FAQ: Frequently asked questions. Many newsgroups on the Internet publish a message each month or so listing frequently asked questions by newcomers to that newsgroup, and giving the consensus answer of the group. In the comp.speech faq many useful questions are addressed, including the location and availability of speech corpora and software.

mfcc: Mel-frequency cepstral coefficients. Mel-scaling accounts for the experimental fact that human perception of pitch is not linear, but is roughly linear below 1000 Hz, and roughly logarithmic above.

cepstrum: The inverse DTFT (discrete-time Fourier transform) of the logarithm of the absolute value of the DTFT of the signal. It is used to separate the speech signal into two components: the glottal pulse train (roughly pitch; excitation) and the resonances of the vocal chambers. The word cepstrum is just the word spectrum with the first four letters reversed, suggesting that the two meanings are nearly the same. In the case of cepstrum the taking of the logarithm provides the significant additional “twist” for which the letters are reversed.

DTFT: Discrete-time Fourier Transform. This is a method for converting a discretely sampled time-domain signal into a frequency-domain representation.

time domain: This is the original form of the digitized signal from the microphone. Each sample represents one instant in time, and the value of that sample is the voltage present at the microphone at that time.

frequency domain: This is the form of the signal after Fourier transformation has been done. In this form each sample represents a frequency band and the value of the sample is the amount of energy present in that frequency band. In this domain the spectrum of the signal can be clearly seen. The size of the frequency band depends on the number of time-domain samples that are used in the Fourier analysis. If there are more samples used, then the band is tighter.

PLP: Perceptual linear prediction, a method for representing a speech signal as a list of perceptual features. PLP coefficients are introduced and defined in Hermansky (1990).

bootstrap: A procedure for estimating the statistical properties of some measurement (Efron and Tibshirani 1993). In this thesis it is used to estimate the variance of the equal error rate of a verification algorithm.

trial: In this thesis a trial is a recognition attempt, with its accompanying confidence and rejection computations. In a trial there is exactly one utterance drawn at random from a corpus. There are also some number of word models drawn from the same corpus. Many trials are performed and their results are considered together in the evaluation and comparison of confidence and rejection algorithms.

corpus: A body of recorded speech (or other sounds). Each recording may be transcribed.

Chapter 2

Prior Research

This chapter provides details of the state of the art surrounding this research on confidence and rejection, as available from the research literature. In particular, the focus is on measures of confidence, improvement of such measures, performance of rejection, and the closely related area of keyword spotting.

2.1 Major Sources of Research Literature

For this research area, results are typically reported in the proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) held each spring. The major journals are the IEEE Transactions on Speech and Audio Processing, starting January 1993, and its predecessor, the IEEE Transactions on Acoustics, Speech, and Signal Processing. Additional work is reported in the proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH) held in late summer on odd-numbered years starting in 1989, and in the proceedings of the International Conference on Spoken Language Processing (ICSLP) held in late summer on even-numbered years starting in 1990, and in the proceedings of annual ARPA / DARPA workshops.

2.2 Scope of Interest

Confidence and rejection comprise a large field of research. In this present thesis the field of interest has been necessarily narrow. Several aspects of that restriction are mentioned in this section.

2.2.1 Vocabulary Independence

The majority of this research is dedicated to vocabulary independence. Hon and Lee (1990) give a good discussion of such modeling. Hon (1992) presents a vocabulary-independent speech recognition system. Hetherington (1995) discusses the problems that lead to the need for vocabulary independence.

Much other research is focused on vocabulary-dependent settings where the words can be known in advance and training samples can be acquired. Some research focuses on class-based vocabulary dependence, where a city-name class may be treated all at once, or where vocabulary words may be classed by their broad-category phonetic spelling. Such research is beyond the scope of this thesis.

It is worth noting that in a vocabulary-independent setting, there are only minor differences between in-vocabulary and out-of-vocabulary verification. That is because the random selection of individual words makes them independent of each other. This allows the in-vocabulary verification problem to be decomposed into the separate problems of out-of-vocabulary verification (impostor rejection) and correct-word verification (correct acceptance). More is said about this in section 4.11.1 starting on page 77. Note that this differs from typical in-vocabulary rejection which is often studied in the context of a specific task and therefore of a specific vocabulary. In that typical task-dependent case results for in-vocabulary rejection and out-of-vocabulary rejection can be quite different.

2.2.2 Discriminative Training

Several researchers have focused on the improvement of the recognition process itself by using confidence results in the training of the recognizer. Such integration approaches are interesting and promise improved performance, but are beyond the scope of this thesis, where the assumption is that a recognition result is to be measured for confidence.

2.3 Research Results of Interest

Each of the headings in this section mentions an area of research where an interesting result is achieved in this thesis. Each also observes related contributions from other researchers.

2.3.1 Logarithmic Averaging

It will be shown (section 5.3) that frame scores which are probabilities can be averaged to advantage if they are first converted to the logarithmic domain. This same result should apply to likelihoods as well. Averaging in the linear probability domain was shown to work less well.

This is not a surprising result, as probabilities are typically combined by multiplication. Lleida-Solano and Rose (1996a) average likelihood ratios and demonstrate logarithmic and other transformations (see section 2.4.1 below).

2.3.2 Hierarchical Averaging

It will be shown (section 5.4) that hierarchical averaging works. Frame scores can be averaged across segments (frames with the same ANN output identity, also called phone-states) to make segment scores, and those can be averaged across phonemes and then words to make word scores. Figure 5.6 on page 101 illustrates the improved separation of true scores from impostors using this scheme.

It appears that most researchers use a whole-word approach to scoring and thresholding. This may be motivated by ease of computation (simply subtracting the Viterbi scores at the start and the end of the word). It will be shown that the whole-word approach gives much worse performance than hierarchical averaging for the corpora and recognition methods used in this thesis.

Rivlin, Cohen, Abrash, and Chung (1996) show that normalizing by phone durations improves performance. They argue that “to get the best recognition match, these [incorrect] phones will have minimal duration in the Viterbi backtrace. . . . Furthermore, since these recognized phones are incorrect, they [typically] have very poor likelihood scores.” This supports a scoring method that does not dilute the badness of such scores.

Segment-Based Scoring: Austin, Makhoul, Schwartz, and Zavaliagkos (1991) use an HMM for segmentation, and then use an ANN to score each entire segment. They call this a Segmental Neural Network (SNN). They reported a word error rate reduction from 9.1% for the HMM system to 8.5% using the additional SNN stage. Austin, Zavaliagkos,

Makhoul, and Schwartz (1992) reports for a different task a reduction from 4.1% to 3.0% which is significant at the 95% level.

Lleida-Solano and Rose (1996a) (see section 2.4.1 below) do whole-word and one-step sub-word averaging of frame scores.

2.3.3 Filler Normalizing

It will be shown (section 5.5) that normalizing the ANN outputs by an average of the top several scores in each frame gives an improved separation of true scores from impostors, as compared to not doing this normalization. This resulted in a “best score” among all algorithms tested. Normalizing using lower-ranked ANN outputs was shown to worsen performance.



On-Line Garbage: Boite, Boulard, D’hoore, and Haesen (1993) and later Boulard, D’hoore, and Boite (1994) introduce an on-line garbage model defined for each frame “as the average of the N best local scores of the CI or CD phonemic models.” In their work this average is modified with a word entrance penalty to prevent the garbage model from scoring better than the keywords. In the present thesis garbage scores are used to normalize keyword phoneme scores rather than to compete against them. This is the same as the all-phone model normalization approach if all phonemes are considered in the N best list. The all-phone model is also used by Young (1994) as an estimate of $p(A)$, the probability of the acoustics, in Bayes equation $p(W|A) = p(A|W)p(W)/p(A)$.

Filler Normalizing: Cox and Rose (1996) use filler models to normalize keyword model likelihoods. They call this a likelihood ratio and show that it approximates a probability. (It should be noted that likelihood ratio is multiply-defined throughout the literature, the commonality being that likelihoods are similar in nature to probabilities but need not sum to 1.0.) They present the use of the highest Viterbi path probability for normalization on a whole-word basis, and find this “to exhibit poor discrimination between classes C and I.”

Other Garbage Models: There are a number of other research efforts using garbage models. Specially-trained garbage models do not play a large part in this thesis, and they are not discussed further here.

2.3.4 Rank-Based Schemes

It will be shown (chapter 6) that throwing away ANN scores and using just the corresponding ranks also results in a “best score” among all algorithms tested. Review of the literature has not identified any similar research using this approach.

2.3.5 Creative Averaging

Weighted averaging schemes (triangular, trapezoidal, and parabolic) are examined in section 6.3 and found to give no additional discriminative benefit. Review of the literature has not identified any similar research using this approach.

2.3.6 Role of Perplexity

It will be shown (section 7.2.1) that perplexity of the impostor set plays an important role in computing the impostor probability used in the likelihood ratio. Jelinek (1981) defines perplexity and relates it to entropy.

2.3.7 Creation of Probabilities

It will be shown that likelihood ratios (odds) and probabilities can be estimated from raw scores (section 7.2.3) and that these can be used to solve a typical problem such as might be encountered by a real business in a principled and vocabulary-independent way. It is important to be able to solve such problems if the technology is to be useful in practice.

Underlying Theory: Duda, Hart, and Nilsson (1976) and Pearl (1990) provide excellent treatments of probabilities and odds (likelihood ratios). Deller, Proakis, and Hansen (1993) includes a brief discussion and Fukunaga (1990) includes a longer discussion of likelihood ratios. Cox and Rose (1996) discuss the creation and evaluation of confidence measures in general.

Comparison of Distributions: Fetter, Dandurand, and Regel-Brietzmann (1996) discusses the use of *eigen* and *fremd* distributions on a vocabulary-dependent basis for estimating probability. Young and Ward (1993) also use vocabulary-dependent distributions and word-class distributions to estimate confidence.

2.4 Confidence Work at Other Institutions

2.4.1 Confidence Work at AT&T and Lucent

The work presented in Lleida-Solano and Rose (1996a) is similar to the work shown in this thesis from the standpoint of general approach and methods of measurement. They present whole-word and segment-based confidence measures, and study several methods for accumulating frame scores into confidence measures. Their accumulation methods include m_1 linear, m_2 logarithmic, m_3 geometric, m_4 sigmoidal, and m_5 harmonic averaging. (Preliminary results following their more exotic approaches did not perform as well as other methods, so no final results are developed for this thesis.)

In Lleida-Solano and Rose (1996b) this work is extended and it is shown that geometric averaging is superior to arithmetic averaging. This is expected because it prevents extreme values from dominating the scoring. The sigmoidal transformation is shown to perform equally well compared to geometric averaging although they expect the sigmoid to be better at damping extreme values. Their emphasis is on development of a one-pass procedure for identifying and scoring word hypotheses.

Sukkar, Setlur, Rahim, and Lee (1996) and related work uses this same geometric averaging to combine several scores in the modeling the likelihood of the incorrect recognitions.

2.4.2 Confidence Work at VerbMobil and CMU

Chase, Rosenfeld, and Ward (1994) use negative n-grams as a language modeling step to prevent the survival of invalid word sequences. The technique is applied to out-of-vocabulary misrecognition.

Schaaf and Kemp (1997) discusses a confidence tagger JANKA for use in the VERB-MOBIL project. The context is large-vocabulary continuous speech recognition for translation purposes. The most important feature found was “A-stabil” which measures the number of times the proposed word occurs in a set of alternative hypotheses.

Both of these approaches make explicit use of language models and is beyond the scope of this present research which is limited to acoustic-based information only.

2.4.3 Confidence Work at SRI

Rivlin, Cohen, Abrash, and Chung (1996) shows that normalizing by phone durations improves performance.

Weintraub, Beaufays, Rivlin, Konig, and Stolcke (1997) develops confidence metrics based on numerous features combined by an ANN. Some of these features are similar or identical in nature to those used in the hierarchical averaging approaches of this thesis. These features include averaging by word, phone, phone-state, or any combination of these.

2.5 Conclusions

Based on the work done elsewhere, I concluded that on-line garbage would be a good baseline for comparison, and that extending the existing hierarchical averaging research from two steps to many steps was a promising direction.

Chapter 3

Vocabulary-Dependent Experiments

This chapter and those following provide details of a number of experiments that were performed. The vocabulary-dependent experiments focus on settings where the active vocabulary is known in advance and word-specific verification strategies can be employed.

The material in this chapter extends results previously reported in Colton, Fanty, and Cole (1995). It is further introduced in sections 1.2 and 1.3 of this thesis.

Section 3.3 reports on utterance verification of putative (hypothesized) recognitions in open-set recognition tasks using telephone speech. The focus is on rejection of out-of-vocabulary utterances. In a two-keyword task (“male” and “female”) using 50% out-of-vocabulary utterances, utterance verification reduced errors by 60%, from 12% to 4.8% compared to a baseline rejection strategy.

Section 3.4 reports on utterance verification of putative recognitions in closed-set recognition tasks using telephone speech. The focus is on re-ordering the N-best hypotheses. In a 58-phrase task, utterance verification reduced closed-set recognition errors by 30%, from 6.5% to 4.5%.

3.1 Introduction

Recognition based on the combination of phonetic likelihoods from short fixed-width frames is the dominate paradigm for speech recognition systems. While this approach has numerous advantages, it is reasonable to think that better word-level recognition is possible using whole-word classifiers. Building such recognizers presents a number of difficulties, such as finding word boundaries before performing the classification, and collecting

enough data to train the classifiers.

A frame-based classifier operates in a narrow context, and makes its judgment about the class to which a frame belongs based on information in that frame and other nearby frames. This classification information must be combined with similar information across many frames to arrive at a word score. The way the information is combined limits the kinds of relationships that can be seen between frame features and the final word recognition accuracy.

A whole-word classifier operates in a broad context, and makes its judgment about the identity of an entire word based on all the frames that belong to that word. Whole-word classifiers can model non-linear effects between word features and word recognition accuracy.

This chapter reports results on experiments with a two-pass strategy. The first pass uses a frame-based recognizer. The output is the recognized word (putative hit) or a list of the top N recognized words, along with the phonetic segmentation derived from backtrace information. This effectively solves the segmentation problem. For these experiments, ample training data was available for the entire vocabulary.

Given a putative match between a test utterance and a reference phrase, the match is verified (i.e., confirmed or denied) using word-specific classifiers. These are ANNs (artificial neural networks) with input features describing the whole word. Combining reclassification with an N-best recognizer allows us to improve recognition accuracy if the utterance verification score is more reliable than the initial recognition score. Out-of-vocabulary utterances can also be rejected by rejecting the entire set of top-scoring matches from the N-best list.

This chapter extends prior work at the Center for Spoken Language Understanding (CSLU) on two-pass Alphabet recognition by Fanty, Cole, and Roginski (1992). In the alphabet system, the frame-based first pass provides letter and broad-phonetic boundaries. The second pass uses an extensive set of knowledge-based features specifically designed for the alphabet. The second-pass classifier has 27 outputs: the 26 letters and an output for “not a letter” which was trained on false positives from the first pass in a development set (mostly noise, not extraneous speech). The second pass yielded much better recognition

than was achieved with a frame-based recognizer alone. The work presented here differs in several ways: the classifiers are word specific, so there are two outputs: word and not-word. This contrasts with having the whole vocabulary in a single ANN. Also, the feature set is generic and not based on careful study of the vocabulary.

This work also extends that of Mathan and Miclet (1991). They used word-specific ANNs to reclassify putative hits in an isolated word recognizer. Their feature vector included duration, average energy and the average first Mel frequency coefficient for each segment in the trace of the first-pass recognition as input features. This work is extended by examining a variety of feature bundles, and by combining reclassification with an N-best search list to improve keyword recognition accuracy.

In all these experiments, telephone speech was used. The speech was digitally sampled at 8000 Hz. For all these corpora, calls are serially numbered as they arrive, and are apportioned into training (60%), development test (20%), and final test (20%) sets according to the last digit of the serial number.

3.2 The Frame-Based Classifier

For both experiments, the first pass is a frame-based classifier which uses an ANN to estimate phoneme probabilities. Speech analysis is seventh order Perceptual Linear Prediction (PLP) analysis (Hermansky 1990), which yields eight coefficients per frame including energy. The analysis window is 10 msec and the frame increment (shift) is 6 msec. The inputs to the ANN are 56 PLP coefficients from a 160 msec window around the frame to be classified. The outputs of the ANN correspond to the phonetic units of the task. For the male/female task the net has only six outputs. For the 58-word task, a context-dependent net with sub-phoneme units (Barnard, Cole, Fanty, and Vermeulen 1995) was used. These units correspond to separate phoneme states in a hidden Markov model (HMM) context-dependent phoneme model. There were several hundred outputs. Section 4.1.2 describes a similar recognizer that is a successor to this one.

Vocabulary words are initially modeled as a sequence of phonemes. For recognition the word model is further refined into a sequence of context-dependent sub-phoneme units

each corresponding to one ANN output of the recognizer. The best alignment between a word model and the ANN probability estimates is found using a Viterbi search. Background sounds are modeled with a simple on-line garbage or filler model (Boite, Bourlard, D'hoore, and Haesen 1993). The model selects the n th ranking phoneme and uses its score instead of computing a garbage score from a trained garbage model. Background modeling increases robustness and provides some wordspotting ability. Wordspotting makes out-of-vocabulary rejection more difficult, as the vocabulary word need only align with part of the extraneous speech.

3.3 Male/Female: Out-of-Vocabulary Rejection

The first experiment sought to identify and reject out-of-vocabulary utterances using a second-pass, whole-word classifier. The task was gender recognition which consisted of two words: “male” and “female.” This is an easy task for which the frame based classifier does very well, but it is fairly difficult for rejection because the target words are so short.

All speech data in this experiment are from the OGI Census corpus (Cole, Fanty, Noel, and Lander 1994). Gender utterances and last name utterances were used. The gender utterances consist of more than 2000 responses to the prompt “What is your sex, male or female?” Of these, roughly 70% were the word “female” (including a few examples spoken by males!) and 30% were the word “male.” The last name utterances consist of responses to the prompt “Please say your last name.”

3.3.1 Baseline System

The baseline system was a frame-based ANN recognizer for the two words “male” and “female.” This recognizer was developed for and used in the OGI Census system (Cole, Novick, Fanty, Vermeulen, Sutton, Burnett, and Schalkwyk 1994). When in-vocabulary utterances are used, the baseline system’s accuracy is 99.5%. To detect low-confidence recognitions, the baseline system takes the ratio of the top two recognizer scores s_1 and s_2 , and compares this to an optimized threshold θ : $\frac{s_1}{s_2} \stackrel{acc}{\underset{rej}{\geq}} \theta$. For a recognition of “male” the test would be: $\frac{s_m}{s_f} \stackrel{acc}{\underset{rej}{\geq}} \theta$. For a recognition of “female” the test would be: $\frac{s_f}{s_m} \stackrel{acc}{\underset{rej}{\geq}} \theta$.

3.3.2 Second Pass Rejection

The approach is to take the Viterbi backtrace to identify the start and end times for each phoneme of the putative utterance. Features based on this time alignment are collected and used to train two new ANNs (one each for “male” and “female”). The new ANNs produce two outputs: “confirm” and “deny.”

The training set contained as many negative examples as positive. The Census corpus contained very few extraneous utterances, so the male-female recognizer was run against the Census corpus of last names (family surnames), forcing each to be recognized as “male” or “female,” and used these as negative inputs for training and testing.

The “female” utterance verifier was trained using 2000 examples, and (due to less available data) the “male” utterance verifier was trained using 1400 examples. In each case half of the training examples represented correct putative hits (drawn from the gender corpus) and half represented incorrect putative hits (drawn from the last name corpus). Similarly, half of the test set was “male” or “female” and half was last names. Using the Viterbi backtrace from the first-pass recognition, word and phoneme boundaries were identified (three phonemes for “male” and five for “female”). The following feature combinations were then examined.

1. [du] Phoneme durations alone. The durations for the phonemes are expected to be stable, but for mis-recognized phonemes the durations may be random. Those that vary from expected values can be eliminated.
2. [en] Phoneme center-frame energy alone. As with durations, the energies for the phonemes are expected to be stable, and those for incorrect phonemes are expected to be more random and therefore possible to identify as wrong.
3. [du.en.+] Phoneme durations, phoneme center-frame energies, plus the energy in the frame 50 msec before and the frame 50 msec after the word. This combines duration and energy and also looks at the context in which the word appears. The frame 50 msec before “male” is probably different than the frame 50 msec before the “male” in “female.” Using before and after context may help to identify these

cases.

4. [du.10p] Phoneme durations plus PLP from ten frames located at 5%, 15%, 25%, ..., and 95% across the word. This is a baseline approach taken from speech recognition literature. It assumes that relative durations of phonemes do not change much across the corpus. If this is true, then the extra overhead of segmenting the utterance can be avoided.
5. [du.5p] Phoneme durations plus PLP from five frames located at 5%, 25%, 45%, 65%, and 85% across the word. This will show the sensitivity of performance to the number of frames examined. Here five frames are used rather than the ten in the [du.10p] case above. If performance is much different, it may suggest using 20 or more frames instead of just 10 or 5. This approach was also chosen because the input features were already available from [du.10p] thus making this an inexpensive thing to test.
6. [du.sp.+] Phoneme durations, PLP from the center-frame of each phoneme, plus the PLP from the frame 50 msec before and the frame 50 msec after the word. This is expected to perform the best because it uses more information than just duration and energy. It uses the full PLP from the chosen frames. Also it allows the relative durations of phonemes to vary as might be expected.

3.3.3 Results

Setting the rejection threshold for the best overall performance on a development set which had an equal number of examples of in-vocabulary and out-of-vocabulary speech, the best performance achieved with the baseline system was 88% overall.

All but one of the feature sets used for second pass classification scored better than the baseline. Phoneme durations alone [du], a very small number of input features, do quite well. Durations and energies [du.en.+] scored about the same as durations alone. Energies alone [en] scored much worse. As expected, durations plus PLP from the center of each phoneme [du.sp.+] scored best. Sampling PLP equally across the word [du.10p] [du.5p] did not work as well as using the phonetic boundaries from the first pass.

Table 3.1: Utterance Verification Accuracy for 6 Feature Sets: Keyword and overall performance is shown along with its difference from the baseline. Notice that du.sp.+ returns the best performance. The test set contains 50% in-vocabulary and 50% out-of-vocabulary utterances.

Results	female	male	overall	gain
0. baseline			.880	
1. du	.948	.883	.928	.400
2. en	.875	.635	.803	(.642)
3. du.en.+	.943	.890	.927	.392
4. du.10p	.954	.911	.941	.508
5. du.5p	.935	.906	.926	.383
6. du.sp.+	.965	.922	.952	.600

Table 3.1 shows the utterance verification accuracy for each of the six feature vector sets, for each of the two keywords. An overall (weighted) average is also shown, and this is compared to the baseline accuracy of 88% to give a measure of error reduction.

In each case, putative hits for “female” were reclassified more accurately than those for “male.” This may be due to the smaller training set for “male” or because there are fewer phonemes on which to base a decision.

3.4 58 Phrases: Improved Closed-Set Recognition

The second experiment used reclassification to re-order an N-best hypothesis list in order to improve recognition accuracy. The closed set consisted of 58 words and phrases in the telephone services domain. Phrases varied in length from two to twenty-three phonemes. The task was to reclassify the top three choices and possibly change the identity of the recognized utterance.

More than 1000 callers said each of the 58 target words or phrases. Each utterance was verified by a human listener, and mistakes (for example, the wrong phrase or a partial phrase) were deleted from the corpus. There was no extraneous speech.

Similar work is reported in Setlur, Sukkar, and Jacob (1996) where the N-best list (for $N=2$) is re-ordered by confidence score. They report an 11% reduction in error rate using

an algorithm similar to that reported in this present section.

3.4.1 Baseline System

The baseline system was a frame based ANN classifier plus Viterbi search. Left and right context dependent modeling, with categories chosen specifically for this vocabulary, resulted in over 500 outputs. Each base phoneme was divided into three parts: left-context dependent, center, and right-context dependent. Using only in-vocabulary test utterances, with each of the 58 phrases equally likely, the accuracy is 93.5%. When there is an error, the correct phrase is often near the top of the N-best list. This is what prompted us to try a second pass classifier.

3.4.2 Second Pass Rescoring

An ANN was trained for each of the 58 keywords using a subset of the data. An equal number of positive and negative examples were used for each. Negative examples were chosen from the utterances for which the target word appeared high in the N-best list (i.e., the more easily confused utterances were selected from within the 58-word vocabulary).

Building on experience from the first experiment, the feature vector was based on the segmentation from the Viterbi backtrace on each putative hit in the N-best list. The following features were used for utterance verification:

- The average per-frame Viterbi score for the entire word (from the first pass recognizer).
- The average per-frame Viterbi score for each sub-phonetic segment.
- The duration of each sub-phonetic segment.
- The PLP from the center of the middle (context-independent) segments.
- The PLP from the frame 50 msec before and the frame 50 msec after the word.

The last three features are the same as those in the male/female verifiers. Newly added are the average Viterbi scores. The Viterbi scores were readily available and I believed

that they might give additional help in the classification process, and in any case would probably not hurt it.

By reviewing the development test scores, a manually optimized threshold was developed to select the best match from the reclassification scores of the top three outputs of the N-best classification: If scores one and two were both below 0.1, and score three was above 0.5, then the third match was selected (this was rare). Otherwise, if score two was 1.7 times greater than score one, the second match was selected. Otherwise the first match was selected.

3.4.3 Results

On the final test set, the error rate without utterance verification was 6.5%. The verification step error rate was 4.5%, which is a 30% improvement. It is interesting to note that when an early version of the first-pass recognizer was below 90% accuracy, the verification improved the performance to about 95%. As the first pass improved, the net result after the verification held steady.

3.5 Conclusions

Word-based reclassification showed promise in both experiments. For rejection, it worked better than the default scheme of using ratios. Although the default was no doubt not the best possible one-pass rejection strategy, the second pass could probably be improved as well. For example, (in the first experiment) no features based on the phonetic probabilities from the first pass were used. The biggest drawback of this approach is the large amount of training data needed to build the classifiers. It is possible to formulate word acceptance as a vocabulary-independent classification problem based on feature sets which can be defined for any word. This is investigated in the next few chapters.

Chapter 4

Vocabulary-Independent Methodology

This chapter contains three simple experiments. They illustrate the methods by which vocabulary-independent research was conducted, and the measures by which experiments are compared. The following chapters (5 and 6) present research results based on the methodology of the current chapter.

The purpose of presenting simple experiments is to focus attention on the research methodology. This includes discussion of the recognizers used, the speech recognition corpora, division of the corpora into training, development test, and final test sets, pronunciation modeling, recognition perplexity, the actions involved in a single recognition trial, the evaluation of results across many trials, computation of statistics by which significance can be determined, and the figures and tables by which the results will be presented.

The ANN-based recognizers used in these experiments represent a different approach in comparison to hidden Markov models (HMMs). The role of the recognizer is so crucial that it is presented first. Section 4.2 follows with information on the first experiment.

4.1 ANN-based Recognizers

A number of ANN-based recognizers have been used in these experiments. They are all general-purpose recognizers with dozens of inputs, a single hidden layer, and several hundred outputs that represent context-dependent phonemes. A tutorial on ASR including discussion of the ANN is given in section 1.6. The Oct 1996 MFCC-based recognizer is identified herein as “Oct96.” It is the recognizer that was used for all of the final experiments reported in this thesis. There are eight other recognizers with which the

confidence and rejection technology has been tested. Both “Oct96” and “May96” are briefly described below.

One reason for varying the recognizer is to see whether the rejection techniques are tied to a particular recognizer or whether they apply generally across several recognizers. Another reason is to test confidence and rejection techniques on the best available recognizer. The identity of that recognizer has changed periodically over the course of this research.

4.1.1 Phonetic Units

Three types of phonetic units are modeled: phones, phone halves, and phone thirds. Phones model an entire phoneme. They can be context independent (CI) (generally silence) or context dependent to the right (CDR) (generally stop consonants). (CDL is also possible but does not occur in the recognizers used in this research.) Phone halves model the left or right half of a phoneme. These are typically consonants. Left halves are context dependent to the left (CDL). Right halves are context dependent to the right (CDR). Phone thirds model the left, center, or right third of a phoneme (typically a vowel). Center thirds are context independent (CI). Left thirds are context dependent to the left (CDL). Right thirds are context dependent to the right (CDR). Up to eight left and right contexts are modeled for each phoneme.

4.1.2 Oct96: Oct 1996 MFCC-based recognizer

The October 1996 recognizer has 131 inputs, 200 nodes in the hidden layer, and 544 outputs. The outputs are listed in Table 4.1. The inputs are 12th-order mfcc (mel-scaled frequency cepstral coefficients, normalized using cepstral mean subtraction) plus energy, and the differences (deltas) of those values from the prior frame, for a total of 26 inputs per frame; taken across five frames (-6 -3 0 3 6) centered on the one to be classified. To this is added one input that is hard-wired to 1.0.

Training: Training was performed with a maximum of 1500 frames per class (for 544 classes). For each class, 500 examples were sought from the OGI Yes/No corpus, 500

Table 4.1: Oct 1996 MFCC-based recognizer ANN Outputs: Three types of phonetic units are modeled: phones, phone halves, and phone thirds. Models are context independent (CI), context dependent to the left (CDL), or context dependent to the right (CDR). For each phoneme, the Worldbet base symbol is given, followed by the numbers of contexts modeled. There are 544 total outputs. These are given in ANN order. Phones are defined in Table 4.3.

phones			phone halves			phone thirds			
phon	CI	CDR	phon	CDL	CDR	phon	CI	CDL	CDR
.pau	1		s	8	8	3r	1	8	8
uc	1		f	8	8	U	1	8	7
vc	1		S	8	8	u	1	8	8
b		8	T	8	8	oU	1	8	8
d		8	D	8	8	aU	1	8	8
g		8	v	8	8	A	1	8	8
ph		8	z	8	8	aI	1	8	8
th		8	h	8	8	>i	1	8	8
kh		8	d_(8	8	ˆ	1	8	8
tS		8	j	8	8	@	1	8	8
dZ		8	9r	8	8	E	1	8	8
			w	8	8	ei	1	8	8
			l	8	8	I	1	8	8
			m	8	8	i:	1	8	8
			n	8	8				

examples from the OGI Numbers corpus, and 500 examples from the OGI Apple corpus. If less than 1500 examples had been found, up to 1000 examples were taken from the OGI Stories corpus, and the remainder up to 1500 total examples were taken from the NYNEX PhoneBook corpus.

The strategy was to train an initial ANN using zero and one as output objectives. Then target reestimation was performed using two iterations of the forward/backward algorithm on the same training data but without the OGI Stories corpus data.

Performance: The closed-set word accuracy of this recognizer is 99.7% on the OGI Yes/No corpus (perplexity two), 95.3% on the isolated digits portion of the OGI Numbers corpus (perplexity eleven; zero through nine, plus oh), and 87.3% on the NYNEX PhoneBook corpus (perplexity 7979).

Table 4.2: May 1996 PLP-based recognizer ANN Outputs: Three types of phonetic units are modeled: phones, phone halves, and phone thirds. Models are context independent (CI), context dependent to the left (CDL), or context dependent to the right (CDR). For each phoneme, the Worldbet base symbol is given, followed by the numbers of contexts modeled. There are 534 total outputs, given in ANN order. Phones are defined in Table 4.3.

phones		phone halves			phone thirds				phones	
phon	CI	phon	CDL	CDR	phon	CDL	CI	CDR	phon	CDR
.pau	1	f	8	8	I	8	1	8	b	7
.br	1	v	8	8	i:	8	1	8	d	8
vc	1	T	8	8	E	8	1	8	g	8
uc	1	D	8	8	ə	8	1	8	th	8
		s	8	8	A	8	1	8	ph	8
		z	8	8	˜	8	1	8	kh	8
		S	8	8	U	7	1	8	tS	8
		h	8	8	u	7	1	8	dZ	8
		m	8	8	3r	8	1	8		
		n	8	8	ei	8	1	8		
		d_()	5	5	>i	8	1	8		
		l	8	8	aI	8	1	8		
		9r	8	8	aU	7	1	8		
		j	8	7	oU	8	1	8		
		w	8	7						

4.1.3 May96: May 1996 PLP-based recognizer

The May 1996 PLP-based recognizer is described here to illustrate the variation in recognizers used during the performance of this research. A total of nine different recognizers were used, with several of them being the best-performing recognizers of their time at CSLU. All recognizers were based on ANN technology and used context-dependent phoneme units. The exact definition of those units varied, as did the training regimen. By showing a few details of this eighth of nine recognizers it is intended that the reader can sense the range of recognizers across which this research was performed, and the applicability of these results to other settings.

The May 1996 PLP-based recognizer has 57 input nodes, 200 nodes in the hidden layer, and 534 outputs. The outputs are listed in Table 4.2. The inputs are eight values from each

of seven frames centered at the frame to be classified, and one additional input hardwired to 1.0. The eight values are the seventh-order PLP (Hermansky 1990) coefficients and one measure of energy.

The recognizer was trained using the OGI Stories corpus, the OGI Yes/No corpus, and the NYNEX PhoneBook corpus.

The Oct96 (ninth) recognizer has much better performance than the May96 (eighth) recognizer, but actual performance statistics are not available.

4.2 Performing One Experiment

The next few sections present and evaluate a simple algorithm. Section 4.8 evaluates two related algorithms, and presents the methodology by which performance is compared.

4.2.1 p^r : raw probabilities

The goal of every confidence algorithm is to create a discriminating raw score, meaning that true and impostor scores can be identified from among each other easily.

For the p^r algorithm the outputs of the recognizer Artificial Neural Network (ANN) are used. Ideally the outputs of an ANN are exactly the *a posteriori* probability that the phoneme is correct, i.e., the probability of a particular phonetic classification given the acoustic evidence (Bourlard and Wellekens 1989, Hampshire and Pearlmutter 1990, and Richard and Lippmann 1991). This requires that the ANN have an infinite amount of training data in natural proportions and an infinite number of hidden units to be trained. While the Oct 1996 MFCC-based recognizer ANN has already been used successfully to do closed-set recognitions (see section 4.1.2 above), it does not meet any of the conditions just mentioned, so at best the outputs only approximate true probabilities. They are designated p^r for “probability raw.”

p^r is used for another reason. It is a simple algorithm, and the purpose of this chapter is to illustrate the methodology rather than to present the best algorithm.

The raw score is computed as the average of the frame scores across the word model, neglecting preceding and following filler frames. Each frame score is simply p^r . Averaging

is done because it is simple. The performance will be seen to be very bad. Improvements will be introduced in the next chapter.

4.2.2 Hypothesis

The raw score will be effective at discriminating between correct and incorrect recognitions. Statistically, the equal error rate will be significantly different from the equal error rate of a process that assigns scores at random.

4.3 Design

Design is a major theme of this chapter. The design given here is typical of all the vocabulary-independent experiments throughout this thesis. The design is motivated by a particular recognition scenario. In this scenario, a word has been recognized by an ASR system and the confidence system has been asked to make a statement about the confidence that should be placed in this particular recognition.

4.3.1 Scenario

For simplicity it is assumed that only the utterance and the best word model are available. There is substantial challenge in looking at the alternate word models that were considered, or that may have been in mind by the talker that produced the utterance, but this challenge is beyond the scope of the current research.

Nothing special is known about the vocabulary or the task. (Of course, in a carefully designed application the vocabulary and task would be studied as well, but this present research seeks to be independent of the specific vocabulary.)

It is not known whether the recognition is correct. Confidence is the probability of a correct recognition, and is the result to be discovered.

It is not known how often the recognition would be correct. This is called the *a priori* or prior probability of correct recognition. Such information would be helpful but is task dependent and can be combined later with the acoustic match confidence.

The experiments are designed to meet the requirements of this scenario.

4.3.2 Algorithm

The computation of confidence is done in two steps. The first step is to compute a raw score using some appropriate algorithm. One such algorithm is p^r . Then given this raw score the second step is to convert it to a probability based on past experience with raw scores of that same value. The creation of a raw score constitutes one trial. The accumulation of experience with such raw scores constitutes an experiment.

4.4 Experiments and Trials

Figure 4.1 illustrates the recognition trial of an utterance. On the left, the utterance is recognized using a vocabulary of incorrect words. The actual words and pronunciations from one recognition trial are shown. Each of these alternatives is scored and the highest scoring incorrect word becomes the Impostor. The impostor is that word that would have been recognized given that vocabulary and that utterance. The goal is to reject the impostor. On the right, the same utterance is recognized using the correct word only as the recognition vocabulary. The recognition score received by this correct word is the same score it would have received had it been a member of some larger vocabulary; vocabulary size does not affect the score of the true word. The goal is to accept the true word and to reject the impostor. Figure 1.6 on page 17 shows the waveform, spectrogram, and segmentations for this recognition trial.

An experiment is composed of a number of separate trials. Each trial mimics the pattern in the scenario just given. A single utterance is chosen at random from some corpus. A matching word model is determined. Both the utterance and the word model are given to the confidence algorithm. The algorithm computes a raw score. If the probability density function is known for this type of score, the score can be converted into a probability.

The probability density is estimated on the basis of a number of these raw scores collected across a training set of utterances and word models.

Because the goal is to assign a given recognition to the class “true” or the class “impostor,” these two probability density functions are estimated. From this the odds or



	
Vocabulary of Incorrect Words bonzo bc b A n z oU chevrolats S E v 9r & l ei z clarifications kc kh l @ 9r & f & kc kh ei S & n z cook kc kh U kc kh drugstore dc d 9r ^ gc g s tc th > 9r handouts h @ n dc d aU tc th s kessler kc kh E s l &r leafing l i: f & N legwork l E gc g w 3r kc kh lonesome l oU n s & m makeshift m ei kc kh S I f tc th northland n > 9r T l @ n dc d occupying A kc kh j & pc ph aI & N outdid aU tc th dc d I dc d springfield s pc ph 9r I N f i: l dc d standoff s tc th @ n dc d > f textiles tc th E kc kh s tc th aI l z undershirts ^ n dc d &r S 3r tc th s unused ^ n j u z dc d wherever h w ei 9r E v &r	Correct Word Only midafternoon m I dc d @ f tc th &r n u n
Top Recognition = IMPOSTOR	Top Recognition = TRUTH
outdid aU tc th dc d I dc d	midafternoon m I dc d @ f tc th &r n u n

Figure 4.1: Illustration of Experimental Design: A single utterance (in this case, “mid-afternoon”) is matched to twenty impostor candidates shown on the left. The best scoring candidate is then returned as the designated impostor. The spectrogram and impostor phonetic alignment are shown in Figure 1.6 on page 17. The same utterance is also matched to the correct word model, shown on the right.

likelihood ratio for any given raw score can be computed easily.

4.4.1 Density Function for Trues

The density function for the true class is estimated by collecting scores from true recognitions. In this case the utterance is selected at random, with each utterance having an

equal chance of being drawn. Next the correct word model is determined by looking at the human transcription for that utterance. The transcription is converted into a word model by dictionary lookup, where the dictionary may be hand built (as in the case of the CMU dictionary) or machine generated using a TTS algorithm.

4.4.2 Density Function for Impostors

The density function for the impostor class is estimated by collecting scores from impostor recognitions. In this case both the utterance and the word models are selected at random subject to the restriction that no word model actually matches the utterance. This is done as follows. First the utterance is selected at random, with each utterance having an equal chance of being drawn. Next some number of word models are selected at random without replacement, with each word model having an equal chance of being drawn. If the true word model happens to be among those word models drawn yet another word model is drawn and is substituted for the true model. The result is a random collection of incorrect word models. These word models are submitted as the active vocabulary and the recognizer selects the one that best matches the utterance. It is designated the impostor and its raw score is used to estimate the density function for the impostor class.

The density function for the impostor class is known to depend on the number of incorrect word models competing in the recognizer. As more word models compete, the chance of getting a good match improves. This is shown in chapter 7.

4.4.3 Comparison to OOV Rejection

An alternate formulation is to select the word model first and use it to recognize a correct utterance. Then the utterance could be changed and recognition attempted again for the impostor case. This is the natural approach because the scenario (section 4.3.1) for a stable recognition task uses a fixed vocabulary across a large number of utterances.

The method used in this thesis was chosen for its substantially greater computational efficiency and theoretical equivalence with the alternate, natural formulation.

Comparative experiments were also performed using actual recognition scenarios to prove empirically that both methods yield the same results. The same performance was

observed when the vocabulary was assembled first and then an utterance chosen as when the utterance was chosen first and then a vocabulary was assembled.

The theoretical equivalence is demonstrated by the fact that the word models and utterances are selected independently and at random, so it cannot matter which is selected first. Thus the impostor selection scheme used in this thesis is exactly equivalent to the typical out-of-vocabulary recognition case where the active vocabulary is specified in advance and the utterance does not match any of the word models in that vocabulary.

Taking a more complex example, suppose that a 20-word vocabulary were constructed, and that 40 utterances were recognized, using 20 correctly-matching utterances and 20 out-of-vocabulary utterances. Each correct recognition would belong to the same distribution of correct recognitions as for all other correct recognitions in this part of the thesis. Further each incorrect recognition would belong to the same distribution of incorrect recognitions as for all other incorrect recognitions in this part of the thesis. Except for statistical problems that would result from the lack of independence among these recognitions (i.e., using the same word models over and over) the problem could be formulated either way and would give the same results. The formulation used in this thesis was chosen to avoid the independence problems present in the example of this paragraph.

4.4.4 Implications of Random Vocabulary

Notice also that vocabulary independence imposes a condition that is unlike natural recognition, where the active vocabulary can be inferred by context. Thus for example in the sentence “Let’s meet at four p.m. on (word)” the active vocabulary would probably consist of days of the week. Given knowledge of the speaker the list might change to include “my yacht,” exclude weekends, or favor certain days over others. Such task dependencies must play an important role in natural recognition but are beyond the scope of this research, which is limited to vocabulary independent recognition as a first approximation to unlimited vocabulary natural recognition.

4.4.5 Summary

In summary, an experiment involves a confidence and rejection algorithm (such as p'). The algorithm is evaluated by using it to score a number of recognitions from some corpus. Some of the recognitions are constrained to be wrong, and involve impostors that are generated using a randomized process. The scores from true and impostor recognitions are compared and the algorithm is characterized by the classification error rate that can be achieved.

4.5 Corpora

Each experiment uses some corpus. For this experiment the OGI Names corpus is used. It is expected to be particularly difficult mainly because name pronunciations generated by TTS tend to be less accurate than pronunciations for other words.

Corpora are collected bodies of recorded speech. The speech is encoded into wavefiles. For the present research, the corpora are required to have a word-level transcription, allowing evaluation of recognition results.

Telephone Speech: The present research has been directed towards telephone speech. An important characteristic of telephone speech is reduced frequency bandwidth. Telephone speech is filtered to occupy the frequency spectrum from 300 Hz to 3400 Hz (more or less) and is typically sampled at 8000 Hz.

Channel characteristics play a role with telephone speech. The quality of transmission has improved with the use of digital signaling on switch-to-switch connections, but analog segments remain in the telephone network, especially in the “first-mile” wiring from the customer to the local switch.

The corpora used in this research are actual telephone speech collected across the public telephone network in the US and Canada.

Two Extremes: Judging from recognizer performance the Names corpus presents a fairly difficult task, while the PhoneBook corpus presents a fairly easy recognition task.

This is attributed to the tendency of names to be shorter, more confusable, and more difficult to model from their spellings, whereas PhoneBook consists of longer words that are less confusable and for which the pronunciation models are provided with the corpus. Together these mark out interesting limits for the evaluation of confidence and rejection algorithms.

Partitioning to Train and Test: Each corpus is divided into several sets. Calls in the training set are used to train the parameters used in confidence and rejection algorithms. Calls in the development test set are used to evaluate and compare algorithms, arriving at preliminary conclusions. Calls in the final test set are “new” utterances never before seen by the system, and are used once at the end of research to verify the preliminary conclusions and create final results. Performance figures reported in this thesis are generally based on final results. Each table or figure tells which partition of the data was used.

To avoid tuning to the final test set, all evaluations and research were done using the development test set for scoring. Thesis writing, including comparison of algorithms and resulting in tentative conclusions was done using the development test set. Immediately before making the final copy of this thesis, the experiments were rerun using the final test set. The performance numbers from the final test set were cut into the thesis at that time, and it was seen that the conclusions based on development test set data continued to be reliable.

Most results from the final test set do not differ significantly from those for the development test set. For all results with a t value of 2.0 or greater, the final test set produces the better results. This suggests that the final test set is easier to recognize or confirm. The most significant difference was for perp 20 Mixed $-\log(p^n)$ fsw Various, where test set scored $.1750 \pm .0014$, which is 5% ($t=3.34$, $df=78$, $\alpha=.0013$) better than dev set which scored $.1840 \pm .0026$. Across the many experiments that were performed it is expected that a deviation this large could occur in a few cases. The next most significant difference had an α value of .0036. The huge majority of differences were insignificant.

4.5.1 Names: OGI Names corpus

The OGI Names Corpus¹ is described on its web page as follows: “The . . . Names Corpus is a collection of first and last name utterances. The utterances were taken from many other telephone speech data collections that have been completed at the CSLU, during which callers were asked to say their first and last names, or asked to leave their name and address to receive an award coupon. Each file in the Names corpus has an orthographic transcription”

- Its internal documentation states: “There is a large variability in the spelling of English names. In the case of common names, plausible spellings were intuitive. However, for the rarer names, we transcribed using an orthography which resembled the pronunciation as closely as possible. We have not attempted to standardize the name spellings. Over the whole corpus there are about 10570 unique names. No standard spellings are used so names such as ‘kerri’ and ‘kerry’ will be counted as two separate tokens. The corpus consists of about 6.3 hours of speech.”

This corpus is further described in Cole, Noel, Burnett, Fanty, Lander, Oshika, and Sutton (1994) and Cole, Fanty, Noel, and Lander (1994).

The current version (release 2) of the OGI Names corpus has 24 000 utterances distributed as follows: *firstname* 9727, *lastname* 11431, *whole* 2659, *other1* 151, *other2* 29, *other3* 2, *other4* 1. These labels are designated in the corpus and generally mean as follows. *Firstname* is the single first name of a person. *Lastname* is the single last name (family name) of a person. *Whole* is the entire name (first name and last name) of a person. Each label was used at most once per call, due to the way that file naming was done for the corpus. The “other” categories represent subsequent names whose category (*firstname*, *lastname*, or *whole*) was already in use for that call.

Special Characteristics: The OGI Names corpus is relatively difficult for recognition and confidence. Following are some possible reasons. Utterances may be excised from fluent speech rather than being isolated pronouncements. The mapping from orthographic

¹<http://www.cse.ogi.edu/CSLU/corpora/names.html> in July 1997

letter sequences to uttered phoneme sequences is less direct than for ordinary English words because of the ethnic origin and diversity of many names that have not been regularized for English pronunciation. The average file length is 943 msec.

Training and Test: The corpus is divided into several sets. By convention at CSLU (the Center for Spoken Language Understanding at Oregon Graduate Institute (OGI)) this division is made by call number. As each call arrives it is assigned a serial number. If the last digit is 0, 1, 2, 5, 6, or 7, the call is assigned to the training set. If the last digit is 3 or 8 the call is assigned to the development testing set. If the last digit is 4 or 9 the call is assigned to the final testing set.

From a total of 24000 utterances, there are 14380 utterances (60%) in the training set, 4854 utterances (20%) in the development test set, and 4766 utterances (20%) in the final test set. This partition is standard for CSLU, and allows different researchers here to share recognizers and other software without the risk of accidentally testing on the training set and thereby getting artificially inflated performance values.

All utterances in the same call are assumed to be by the same person. Each call is assumed to be by a different person. There are known to be exceptions. The actual amount of duplication is unknown but believed to be inconsequential.

4.5.2 PhoneBook: NYNEX PhoneBook corpus

PhoneBook is announced in Pitrelli, Fong, Wong, Spitz, and Leung (1995). As detailed in the PhoneBook Final Report (Pitrelli, Fong, and Leung 1995), "PhoneBook is a phonetically-rich, isolated-word, telephone-speech database . . . of American English word utterances incorporating all phonemes in as many segmental/stress contexts as are likely to produce co-articulatory variations, while also spanning a variety of talkers and telephone transmission characteristics. . . . The core section of PhoneBook consists of a total of 93,667 isolated-word utterances, totaling 23 hours of speech. This breaks down to 7979 distinct words, each said by an average of 11.7 talkers, with 1358 talkers each saying up to 75 words. All data were collected in 8-bit mu-law digital form directly from a T1 telephone line. Talkers were adult native speakers of American English chosen to be

demographically representative of the U.S.”

The words were organized into about 100 word lists, and each list was read by about 15 talkers. Five words (“examiners,” “hire,” “hutchins,” “sports,” and “your”) appear on more than one list.

With the exception of the five repeated words, there is no overlap between sets, either in the vocabulary words used or in the speakers themselves. This provides both speaker independence and vocabulary independence between the three sets.

Special Characteristics: The PhoneBook corpus is relatively easy for recognition and confidence. Following are some possible reasons. Utterances are isolated pronouncements. The pronunciations are screened to avoid rare or surprising variations. A pronunciation dictionary (modified from the CMU dictionary) is included with the corpus. The apparent randomness of the words themselves may cause the talker to enunciate them more carefully so as to avoid misrecognition as another word. The words were generated in a way that maximizes the phonological coverage of English, and guarantees that each word contains a unique phonological context not present in any of the other words. This unique context may make it easier to discriminate between correct and incorrect word models. The words also tend to be long because long words provide more phonological contexts. This tends to give more phonemes that will be wrong for an incorrect recognition. The average file length is 884 msec (about the same as OGI Names).

Training and Test: At CSLU the word lists are divided into a training set (50%), a development test set (25%), and a final test set (25%). AK68_M10 is a typical PhoneBook filename. Its second letter (e.g., K) is used to partition the corpus. Odd letters (A, C, E, G, I, K, M, O, Q, S, U, W, and Y) are assigned to the training set. The first half of the even letters (B, D, F, H, J, and L) are assigned to the development test set. The remaining letters (N, P, R, T, V, X, and Z) are assigned to the final test set. This partition is standard for CSLU, and allows different researchers here to share recognizers and other software without the risk of accidentally testing on the training set and thereby getting artificially inflated performance values.

4.5.3 Corrections

No wavefiles were eliminated from either corpus. Transcriptions were regularized in an automated way to facilitate the generation or lookup of pronunciation models. This involved removal of informative markings such as
 (indicates the presence of breath noise) and hypothesized portions of words (jonath[an] indicates the end of this name was cut off but believed to be as shown).

Sampling and Trials: In this thesis a trial is a single recognition attempt, with its accompanying confidence and rejection computations. In a trial there is exactly one utterance drawn at random from a corpus. There are also some number of word models drawn from the same corpus. Many trials are performed and their results are considered together in the evaluation and comparison of confidence and rejection algorithms.

A randomized sample of utterances and word models is drawn from the corpus. The sample size (number of trials) is chosen to be large enough that measured differences will be statistically valid. Section 4.11.1 presents more information. A short overview is given here.

In each of the n trials an utterance is selected at random. The word model is generated for the correct word, recognition is performed, and a raw word score is computed by averaging the frame scores. An impostor (section 4.9.2) is also generated and scored. The true word score and the impostor word score are kept. Eventually there are n of each score.

Note that it does not matter whether the utterance is chosen first or the impostor candidates are chosen first. In any case, the impostor score represents an out-of-vocabulary speech recognition event, where the candidates represent the active vocabulary and the utterance is out-of-vocabulary with respect to that set.

Impostors and Perplexity: Impostors are drawn from a best-of-20 pool, where each of the 20 candidates is drawn at random from words actually present in the corpus. The number of possible alternatives is called the perplexity of the task. The creation of impostors is further discussed in section 4.9.2. Effective use of impostors is made

difficult by the fact that this is open-set rejection.

ANN-based Recognizer: Each experiment uses a recognizer to identify what phoneme is being uttered at each point in time. The Oct 1996 MFCC-based recognizer is used in this and all other experiments. Preliminary research was conducted with a variety of other recognizers, including the May 1996 PLP-based recognizer.

4.6 Pronunciation and Word Modeling

This experiment uses word models from Orator TTS (described in section 4.6.2).

For each recognition attempt the Viterbi search algorithm aligns all available word models against the ANN outputs from the utterance. The model that scores best is declared winner. Word models that are not provided with the corpus must be derived in some other way.

Recognition (described more fully in section 4.10) is done by matching ANN outputs to a word model. The word model is specified as a list of phonemes. The Worldbet phoneme set (Table 4.3) is used to express word models and to identify phonemes. Using this scheme a word model can be constructed based on the pronunciation of the word. For example, the word “yes” can be modeled as /j E s/ and the word “no” can be modeled as /n oU/.

When phonetic word models are needed they are retrieved from a dictionary or generated using a Text-to-Speech (TTS) algorithm.² For the Names corpus Orator pronunciations are used. Other possibilities were also tried, including DECTalk and Rsynth pronunciations, but their performance did not surpass that of the Orator pronunciations. For the PhoneBook corpus CMU pronunciations (provided with the corpus) are used.

Phonetic word models must be further modified to produce ANN-specific word models

²Available public-domain or free dictionaries include the CMU dictionary and the Moby dictionary. Available TTS algorithms include Orator from Bellcore, DECTalk from Digital Equipment Corporation, and Rsynth from the University of Cambridge (UK). Other dictionaries and TTS programs exist, and a number of them are listed in the FAQ (frequently asked questions) of the comp.speech newsgroup. Alternately pronunciations could be gathered from phonetically labeled corpora such as the OGI Stories corpus, and with practice ordinary people could create word models just as they can now spell, but these approaches are not pursued in this research.

Table 4.3: **Worldbet Symbols:** The Worldbet symbols are used to form word models. OGI symbols are also given because they are familiar to many researchers. The samples are common English words that exhibit the specified phoneme.

Worldbet	OGI	Sample	Worldbet	OGI	Sample	Worldbet	OGI	Sample
i:	iy	be <u>e</u> t	iU		fe <u>u</u>	s	s	si <u>gn</u>
I	ih	bi <u>t</u>	aU	aw	ab <u>ou</u> t	S	sh	ass <u>u</u> re
E	eh	be <u>t</u>	oU	ow	bo <u>o</u> t	s	s	si <u>gn</u>
@	ae	ba <u>t</u>	ph	p	pa <u>n</u>	S	sh	ass <u>u</u> re
&	ax	ab <u>o</u> ve	th	t	ta <u>n</u>	h	hh	ho <u>p</u> e
u	uw	bo <u>o</u> t	kh	k	ca <u>n</u>	v	v	vi <u>n</u> e
U	uh	bo <u>o</u> k	b	b	ba <u>n</u>	D	dh	th <u>y</u>
^	ah	ab <u>o</u> ve	d	d	da <u>n</u>	z	z	re <u>s</u> ign
>	ao	ca <u>u</u> ght	g	g	ga <u>n</u> der	Z	zh	a <u>z</u> ure
A	aa	fa <u>t</u> her	m	m	me <u></u>	tS	ch	ch <u>u</u> rch
3r	er	bi <u>r</u> d	n	n	kn <u>ee</u>	dZ	jh	ju <u>d</u> ge
&r	axr	bu <u>tt</u> er	N	ng	si <u>n</u> g	l	l	le <u>n</u> t
ei	ey	ba <u>y</u>	d_ (dx	ri <u>d</u> er	9r	r	re <u>n</u> t
aI	ay	by <u>e</u>	f	f	fi <u>n</u> e	j	y	ye <u>s</u>
>i	oy	bo <u>y</u>	T	th	thi <u>gh</u>	w	w	w <u>e</u> nt
uc	unvoiced closure (before ph, th, kh, tS)					.br	breath noise	
vc	voiced closure (before b, d, g, and dZ)					.pau	pause or silence	

that identify the exact sequence of ANN outputs needed for that word. For instance, the transition /j E/ may be modeled as “first half of j after silence” followed by “second half of j leading into E” followed by “first third of E starting after j” followed by “central third of E.” Each of these context-dependent phonemes would correspond to some specific output of the ANN. These ANN outputs are presented in tables 4.1 and 4.2.

4.6.1 Worldbet Symbols

Table 4.3 shows Worldbet symbols used to specify word models.

4.6.2 Orator: word models from Orator TTS

The Orator web page³ presents the following information (June 1997). “Bellcore’s ORATOR(tm) Speech Synthesizer provides the tools for high quality, highly accurate telephone access to database-driven information services through the process of text-to-speech synthesis. ORATOR’s first commercial use is a popular reverse-telephone-directory service, currently available in Illinois.

“ORATOR’s Features Include: * Highest accuracy for name pronunciation available for American people, places, and businesses. * A high level of speech intelligibility - resulting in clear and natural sounding speech. * Excellent acronym pronunciation. * Words spelled out upon request, with human-like letter grouping. * Flexible, powerful facilities for customized pronunciation and intonation. * Ports to a variety of platforms.”

Orator is used with each name to produce a word model for that name. The word model consists of the sequence of phonemes that would have been uttered by Orator if it were attempting to pronounce that name.

4.6.3 CMU: word models from CMU dictionary

The CMU dictionary is described on its web page as follows (June 1997): “The Carnegie Mellon University Pronouncing Dictionary is a machine-readable pronunciation dictionary for North American English that contains over 100,000 words and their transcriptions. This format is particularly useful for speech recognition and synthesis, as it has mappings from words to their pronunciations in the given phoneme set. The current phoneme set contains 39 phonemes...” It is also cited in the `comp.speech` FAQ.

A version of the CMU dictionary is provided with the NYNEX PhoneBook corpus. Word models are determined by dictionary lookup for the words in this corpus.

4.6.4 Wordspotting Grammars

Before and after each word, a filler model is required by the grammar. It is sometimes referred to as the “any model” because it matches anything. It models the context around

³<http://www.bellcore.com/ORATOR/> in July 1997

the word in question. If the utterance were “I’m John (breath)” and the word model were “dZ A n” the filler model would need to account for the frames belonging to the preceding “I’m” and to the following breath noise. The grammar requires the first frame of the utterance to belong to the leading filler model, and the last frame to belong to the trailing filler model. More frames can be assigned to the filler models by the Viterbi search if it improves the overall word score.

In general, grammars are composed of word models, just as word models are composed of phonemes. Words are modeled as formal regular expressions, and grammars as formal context-free grammars. Grammars are particularly useful for specifying recognizers of connected-digit strings, dates, times, or other multi-word objects where repetition or branching are important within the model. In the context of the current research, grammars are used only to support wordspotting.

Filler Modeling: A typical wordspotting grammar is `<.any> word <.any>`. With this model the utterance is divided into three parts using an artificial phoneme `<.any>` in the scoring process. This phoneme is defined as having the same neural network output value as the median of the top n other phonemes (typically n defaults 30 or 50) or of the `silence` phoneme, whichever scores better. (The median approach is based on HMM work by Bourlard et al. 1994). In each frame, the phoneme with (counting from one) the 16th highest value ($30/2 + 1$) is identified, and the value is copied to become the value of the `<.any>` phoneme. If `silence` scores better, then it is copied instead. `<.any>` is used to account for phonemes outside the target word, and thus provides wordspotting capability.

4.7 Results: Raw Score Histogram

The results of these trials are shown by the histograms in Figure 4.2. (Section 4.11.5 gives details on histogram creation and smoothing.) The true scores have a median value of 0.12 and the impostors have a median value of 0.06. (These values are not important in themselves. It is important that they are different by a relatively large amount.) It is clear to see that there is a substantial difference between the two distributions, and

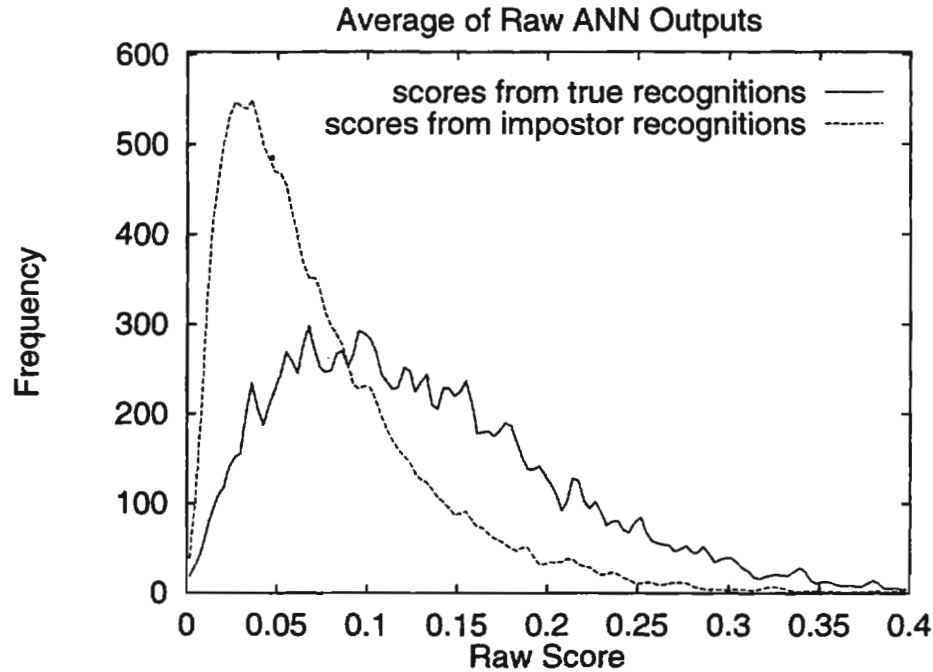


Figure 4.2: Histogram for p^r : from 0.0 to 0.4 in 128 steps. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, OGI Names corpus, raw probabilities, frame-to-word averaging, word models from Orator TTS, 16000 trials, final test set.

that the simple algorithm does distinguish to some extent between correct and impostor recognitions. The overlap seems rather large but improvements will be made in subsequent experiments. (The emphasis in this chapter is to identify the methodology.)

4.7.1 Total Verification Error

Figure 4.3 presents error rates at various raw score values. Three error rates are presented: Type I, Type II, and total (TVE). Two performance statistics are presented: the MVE (minimum point on the TVE curve) and the EER (the cross-over point on the Type I and Type II curves). The Type I error rate (defined for example in Spence, Cotton, Underwood, and Duncan 1992) is the proportion of true word scores that would be rejected at that threshold. It is also called the α error. The Type II error rate is the proportion of impostor word scores that would be accepted at that threshold. It is also called the β error. The sum of these is the Total Verification Error (TVE).

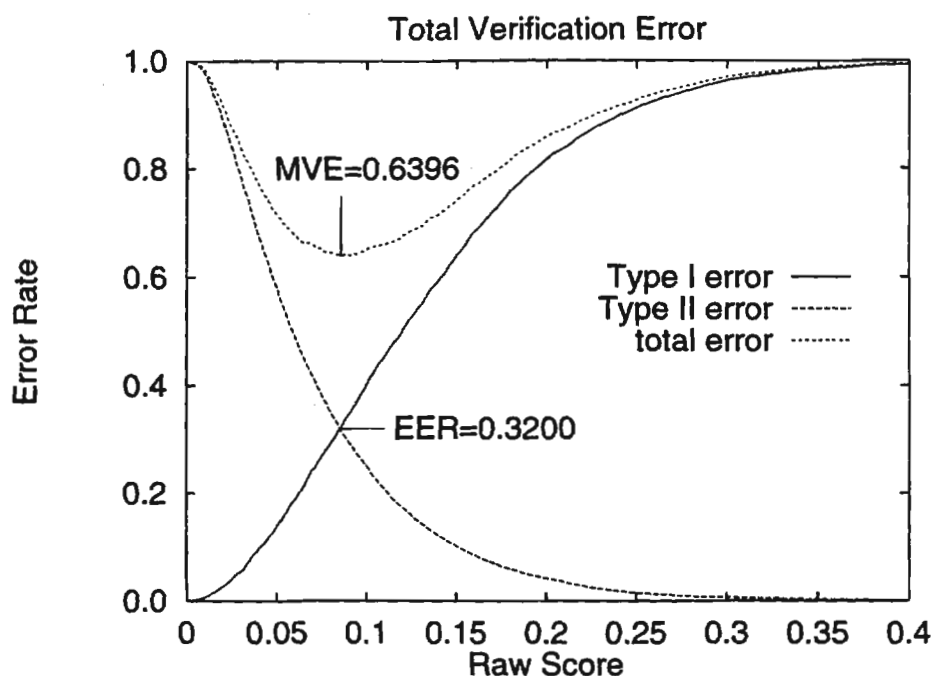


Figure 4.3: Various Error Rates for p' : from 0.0 to 0.4. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, OGI Names corpus, raw probabilities, frame-to-word averaging, word models from Orator TTS, 16000 trials, final test set. Type I error is rejection of truth. Type II error is acceptance of falsehood.

TVE dips and rises with a minimum verification error (MVE) of .6396 near a raw score of 0.085. MVE is the minimum point on the Total Verification Error curve.

In both figures (4.2 and 4.3) the better scores are toward the right. Scores to the right of a threshold would be accepted while those to the left of the threshold would be rejected.

At one extreme (in this case a threshold of 0.0) all scores are accepted. The Type I error rate is 0.0, since no true recognitions are rejected. The Type II error rate is 1.0, since all imposters are accepted.

At the other extreme (in this case a threshold of 1.0) all scores are rejected. The Type I error rate is 1.0 since all true recognitions are incorrectly rejected. The Type II error rate is 0.0 since all imposters are correctly rejected.

Between these two extremes there is a raw-score threshold (say 0.085) at which the error rates are equal. This rate is .3200. That means that .3200 of the true recognitions

Table 4.4: Mean, Standard Deviation, and 95% Confidence Interval for the p^r Algorithm. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, OGI Names corpus, frame-to-word averaging, word models from Orator TTS, 16000 trials, final test set, equal error rates.

Algorithm	mean $\pm s_{\bar{x}}$	n	95% confid
p^r	.3200 \pm .0023	200	.3155-.3245

would be incorrectly rejected, and .3200 of the impostor recognitions would be incorrectly accepted at that threshold. This is the Equal Error Rate (EER).

EER is used as the primary decision statistic in this research, but some interesting alternatives are discussed in section 4.11.6.

4.7.2 EER Statistics

Table 4.4 presents the estimated mean, standard deviation, and 95% confidence interval for algorithm p^r . Each of the terms used in the table and caption is explained below.

mean $\pm s_{\bar{x}}$: The mean is the mean equal error rate for the algorithm in question. It is defined as the equal error rate of the original raw scores before bootstrapping (see below) is performed. $s_{\bar{x}}$ is the standard deviation of the mean, which is the square root of the variance of the bootstrap estimates.

bootstrap iterations: The bootstrap procedure (Efron and Tibshirani 1993) is a statistical method for estimating the variance of quantities that may otherwise be hard to evaluate. Briefly the procedure involves treating the sample as though it were a population, and repeatedly drawing same-size samples from it (with replacement). The variance of these secondary (bootstrap) samples is an estimate of the true variance. See section 4.11.8, page 83 for more on bootstrapping.

n: This is the number of bootstrap iterations. Each iteration produces an estimate of the mean. (This is not the number of trials performed.)

95% confid: The estimated ERR is a random variable. The true EER is not known, and must be estimated by statistical means. The estimate may also be wrong, but it is possible to state a range (a confidence interval) in which the truth is likely to lie. For the confidence interval tables, these ranges indicate that 95 times out of 100 the truth will lie within the range given. This is a central range, which means that half the errors will be on each side of the range.

These central confidence intervals are computed by the standard-deviation method using Student's t distribution. See section 4.11.8 for more details.

impostors at perplexity 20: This is the perplexity used in these experiments. It is the number of randomly selected word models from which the best was chosen to be the impostor. This is described in section 4.9.3.

Initial experiments used a range of perplexity values: 2, 3, 5, 10, 20, 50, 100, 200, 500, 1000, and 2000. I saw that performance varied smoothly across perplexities, and that the comparison of one algorithm to another seemed stable even when perplexity was varied. Therefore it did not seem very important what perplexity was used. To increase the experimental throughput and to decrease the total number of experiments, perplexity was held at 20 for the experiments that are reported in this thesis.

In terms of the final application of this research, one major area would be confidence measurement among sets of alternatives that might be typed in by hand using the CSLUrP rapid prototyper. I believe that the number of alternatives present in these systems tends to be 20 or less. Therefore it seemed best to have performance measurements that are at their most accurate point near the perplexity that would be actually used in applications, all other things being equal.

Finally, the running time for experiments was a consideration. For larger and larger perplexities, the running time grew with about the square of the perplexity. (This is an empirical result.) Whether for memory page faulting or other reasons, the much longer completion time for experiments did not seem to justify the higher perplexities. But a perplexity of 20 executed about as fast as a perplexity of 2, due to fixed costs of running each trial within an experiment. 20 appeared to be at about the knee of the execution-time

curve, with perplexity 10 taking about as long; and perplexity 50 taking much longer.

For all these reasons the final experiments were conducted at perplexity 20, although earlier experiments were conducted at a wide range of perplexities.

Oct 1996 MFCC-based recognizer: This is the recognizer used in these experiments. It is described in section 4.1.2.

OGI Names corpus: This is the corpus is used in these experiments. It is described in section 4.5.1.

frame-to-word averaging: This is the method by which frame scores were accumulated into word scores. In this case the word scores were computed directly by averaging the individual frame scores within the word. Other ways of accumulating the word score are presented in section 5.4.

word models from Orator TTS: Word models are generated using the Orator text-to-speech system. It is described in section 4.6.2.

16000 trials: 16000 recognition trials (or some other number) are used to collect examples for scoring. This process is described in section 4.11.1. A larger number of trials generally results in a better estimate of the mean, as the standard deviation of the mean tends to decrease with the square root of the number of trials performed.

final test set: This is the test set used in these experiments. Test sets are described in section 4.5.

4.8 Comparing Several Experiments

How can comparison be made among several algorithms? The basic approach is to compare their equal error rates to identify the algorithm that performs best. To illustrate this comparison two additional algorithms are discussed and evaluated.

4.8.1 Hypothesis

Here I will speak in general terms. Throughout the next chapters many algorithms will be presented and discussed. The general hypothesis will be that such-and-such an algorithm is better than all others that have been considered. Some intuition will be given for why it is reasonable to expect it to be better. The hypothesis will be tested statistically by looking at the equal error rate (EER) of the varying algorithms, using bootstrap analysis to estimate how significant the differences actually are, or whether the differences are adequately accounted for by random chance. The EER measurement shows how good an algorithm is at discriminating between true recognitions and impostor recognitions.

Specific to the current chapter, two new algorithms are introduced next. They are p^n and $p^n/(1 - p^n)$. It is hypothesized that p^n will be an improvement upon the p^r algorithm already seen, and that $p^n/(1 - p^n)$ will be a further improvement.

4.8.2 p^n : normalized probabilities

The first of these algorithms modifies the p^r raw score by normalizing each frame so the scores sum to 1.0. These new scores are called p^n for “probability normalized.” The intuition for normalizing the raw scores is that true probabilities sum to 1.0, and at least to the extent the p^r probabilities do not sum to 1.0, they are flawed as probabilities. When the sum is one 1.0, it is not immediately clear whether the fault is localized to a few of the phoneme estimates or is general to the entire frame. Assuming that it is general to the entire frame, then normalization makes sense. The effect of normalization is to put all frames on a equal basis. Otherwise, frames with a total probability over 1.0 are weighted more heavily than frames with a total probability less than 1.0.

The methodology is exactly as stated for the p^r algorithm.

Table 4.5 shows a substantial decline in performance compared to p^r . In section 5.3 a variation on this algorithm will be shown, but even there a substantial decline in performance is seen. This indicates that something important has been lost or masked due to this normalization. I can reject the hypothesis that putting all frames on a equal basis helps discrimination. Instead, it appears that frames with a total probability over

Table 4.5: Mean, Standard Deviation, and 95% Confidence Intervals for Algorithms in the p^r Family. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, OGI Names corpus, frame-to-word averaging, word models from Orator TTS, 16000 trials, final test set, equal error rates. For more explanation see page 67.

Algorithm	mean $\pm s_{\bar{x}}$	n	95% confid
p^r	.3200 \pm .0023	200	.3155-.3245
p^n	.3421 \pm .0023	200	.3376-.3466
$p^n/(1 - p^n)$.3621 \pm .0024	200	.3573-.3669

1.0 should be weighted more heavily than frames with a total probability less than 1.0, and that the probabilities supplied by this ANN (Oct96) are more reliable when their frame-wise sum is greater.

4.8.3 $p^n/(1 - p^n)$: likelihood ratio (odds)

The second of these algorithms uses a likelihood ratio or odds formulation. The likelihood ratio is defined as the probability of truth (p) divided by the probability of error ($1 - p$). In this case the truth is represented by p^r and error by the sum of all other ANN scores in the same frame. This is mathematically equivalent to $\frac{p^n}{1-p^n}$. Since it is easy to convert both ways between p^n and $p^n/(1 - p^n)$ no information is lost. However, the emphasis changes to favor frames with high likelihoods and discount frames with low ones.

The intuition for using likelihood ratios is that such ratios taken individually are optimal decision boundaries, as shown in Fukunaga (1990). Additionally likelihood ratios are a transformation of scale with respect to the normalized probabilities. Using a likelihood-ratio formulation will tend to give strong positive scores a large emphasis, while weak scores will receive less emphasis. The result will be to favor the strong scores.

The same methodology is used. The results (given in Table 4.5) show a further decline in comparison to the performance of p^n . The decline in performance must be due to the change in emphasis which has resulted in scores that do not accumulate as accurately. Instead of emphasizing the strong positive scores it might be better to emphasize the weak scores, such as by using the reciprocal. In section 5.3 a variation on this algorithm will be shown to avoid this substantial decline in performance relative to p^n , and also to

Table 4.6: Distance Chart for Algorithms in the p^r Family. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, OGI Names corpus, frame-to-word averaging, word models from Orator TTS, 16000 trials, final test set, equal error rates.

$.3200 \pm .0023, p^r$			
9		$.3421 \pm .0023, p^n$	
18	7	$.3621 \pm .0024, p^n/(1 - p^n)$	

improve performance dramatically in comparison to the algorithm of this current section.

4.8.4 Mean, Standard Deviation, and Confidence Intervals

Table 4.5 presents the estimated mean, standard deviation, and 95% confidence interval for the three simple algorithms considered in this chapter.

4.8.5 Distance Chart

Table 4.6 presents a distance chart for the three simple algorithms considered in this chapter. It is styled after the distance (or mileage) charts often found on road maps, which give the distance in miles between cities. This distance chart gives a statistical “distance” between algorithms, telling how rarely such a difference in performance would arise by chance. This is explained below.

The caption information is described on page 67. Performance figures ($\text{mean} \pm s_x$), algorithm names, and other information that varies from case to case is listed along the main diagonal, best first, starting in the upper left corner. The distance between two algorithms is shown at the intersection of their respective row and column. Low numbers indicate the difference could be due to chance. High numbers indicate the difference is significant. A distance of **0** means that even when no difference exists in the true means, these estimates will have such a difference by accident more than 1 time in 10. **1** means 1 in 10 or less (two-tailed $\alpha \leq .1$). **2** means 1 in 100 or less ($.001 < \alpha \leq .01$). **n** means between 1 chance in 10^n and 1 chance in 10^{n+1} .

Table 4.6 indicates that p^r is better than p^n since that is their order on the main diagonal. The **9** indicates that their difference could happen by chance only 1 time in 10^9

Table 4.7: Differences Across Selected Algorithms in the p^r Family. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, OGI Names corpus, frame-to-word averaging, word models from Orator TTS, 16000 trials, final test set, equal error rates.

Better		Difference				Worse	
Algorithm	EER $\pm s_{\bar{x}}$	diff	t	df	α	EER $\pm s_{\bar{x}}$	Algorithm
p^r	.3200 \pm .0023	6%	6.85	398	.0000	.3421 \pm .0023	p^n
p^r	.3200 \pm .0023	12%	12.61	398	.0000	.3621 \pm .0024	$p^n/(1 - p^n)$
p^n	.3421 \pm .0023	6%	5.98	398	.0000	.3621 \pm .0024	$p^n/(1 - p^n)$

or less. Further, p^r is better than $p^n/(1 - p^n)$. 18 means the chance of getting such a difference by unlucky sampling is less than 1 in 10^{18} . And p^n is shown to be better than $p^n/(1 - p^n)$ unless a 1 in 10^7 mistake occurred. It is normally safe to trust numbers greater than 2. Distance numbers are computed by taking the two-tailed α value based on the computed t score⁴, and then reducing it by taking its logarithm to base 10.

4.8.6 EER Across Algorithms

The distance chart presents a large amount of information in a very compact format. For some pairs of algorithms a more detailed presentation may be appropriate. This is provided in the “EER Differences Across Algorithms” tables.

Table 4.7 compares the Equal Error Rate between pairs of algorithms in the set considered in this chapter. Each line in the table lists two algorithms and tells how they compare to each other. The algorithm mentioned first (on the left end of the line) has a better (or equal) EER than the the algorithm mentioned last (on the right end of the line). The percentage difference (improvement) over the algorithm on the right is shown, together with the statistical t score for such a difference, the number of degrees of freedom, and the probability of getting that difference or more by random chance (two-tailed α level).

The Rest of the Chapter: The remaining sections of this chapter present issues already touched upon, but do so in more depth. These details were deferred until now to

⁴ t score tail area is computed using the public-domain algorithm 27 from “Applied Statistics, Volume 19, number 1.”

make the initial presentation easier to follow.

4.9 Impostors and Perplexity

Closed-set means that the utterance is guaranteed to match one of the word models. With closed-set rejection, each word model can be considered in turn and if two word models both match well (e.g., “Doug” and “dog”) this fact can be known and used.

Open-set rejection is more difficult. Say for example that the active vocabulary is “dog,” “cat,” and “bird.” Say also that the actual utterance is “Doug.” What should be done? Is it close enough to be called “dog”?

Since one does not know which other words might be uttered, it is difficult to decide just how similar the utterance must be to the word model. The problem is approached here by the creation of impostors based on a perplexity parameter.

4.9.1 True Words

For each utterance the true identity of the word is recorded by a trained human listener. Since there is only one true word for each utterance, scoring is straightforward. The scores of randomly selected true words provide a histogram of the frequency distribution of all true words.

4.9.2 Impostors

There are some difficulties surrounding the generation of impostors. Ideally they would mimic the distribution of real-world impostors. This is much more difficult to know than for true words. It varies from task to task in ways that are beyond the scope of this thesis.

The approach taken here is illustrated in Figure 4.1 on page 52. It is to select impostors from the same corpus that provided the correct word. The actual selection of impostors is done by first selecting several word models at random, together with one utterance wavefile that does not match any of the word models. Viterbi search is used to identify the best-matching word model and it is declared to be “the” impostor for that utterance. It is scored as though it were a true word.

For simplicity all incorrect utterances and word models are assumed to be equally likely. This is a fairly gross simplification, as some names are rather frequent and others quite rare. However, all algorithms are tested under the same assumption and it is expected that relative rankings would be stable across any reasonable variation in the frequency of particular names. (To examine this hypothesis the algorithms are evaluated in section 5.1 with other corpora. The relative results appear to be stable.)

In the vocabulary-independent setting of this research it does not seem obvious how one might account for the varying likelihoods of different words in the vocabulary. These likelihoods can also vary by the task that is being undertaken. One example using prior probabilities is given in the collect-call scenario presented earlier, but detailed examination of such settings is beyond the scope of this thesis.

Scores from true words and impostors must be distributed differently in order for confidence and rejection to be better than random chance. It is the job of the algorithm to create such scores.

4.9.3 Perplexity

The term “perplexity” is used to refer to the number of word models from which the impostor was chosen. In a perplexity-20 setting, each impostor is the best match from a set of 20 word models drawn at random.

As the perplexity increases, the average goodness-of-match for the impostor also improves. With a large enough vocabulary it becomes almost certain to find an impostor that scores as well or better than the true word model does. In section 7.2.1 it is estimated that the Oct96 recognizer reaches a limit at 2795 random words. With that many words, the average impostor will score just as well as the correct word. This limit probably depends on the particular recognizer that is being used. At perplexity 2795 the average acoustic confidence score for true recognitions is 0.5. This is also the average acoustic confidence score for impostor recognitions.

This is important because the target scenario (section 4.3.1) involves a random list of word models and a random utterance all from the same corpus. The number of incorrect word models has a direct effect on the probability of getting a good score from an incorrect

model.

4.10 Recognition by Viterbi Alignment

The recognizer operates by aligning the actual utterance (digitally recorded) with a computer model of the target word. This alignment process creates (as a side effect) a score that gives the relative likelihood that the word model is correct for that utterance. This recognition score (also called the Viterbi score) is computed for each of the word models, and the model with the best score is selected.

4.10.1 Frames and Words

Each utterance is divided into frames of fixed length. This length is 10 msec. Researchers use a variety of frame sizes. The 10 msec length is inherited from the recognizer and is taken as a given. It is not optimized in any way for this confidence and rejection research.

Viterbi search is the process by which each frame is assigned to one of the parts of the word model. An example may help. Say an utterance is “yes” and has a duration of 0.90 seconds. The word model is “j E s” (word models and the phonetic alphabet are explained more fully in section 4.6) and the grammar is “any yes any.” At 10 msec per frame there are 90 frames in this utterance. They are numbered from 0 to 89. By Viterbi search, frames 0–7 are assigned to the filler model, frames 8–11 are assigned to the phoneme “j,” frames 12–26 to the phoneme “E,” and frames 27–44 to the phoneme “s.” The remaining frames, 45–89, are again assigned to the filler model. Each of these five parts is called a segment.

4.10.2 ANN Probability Profiles

The recognizer ANN is employed to estimate a score p^r for each frame of the utterance. This score is computed for all possible phonemes, giving not just a single score but an entire profile of scores at each moment in time. In the example above, the Viterbi search algorithm uses the ANN scores from “j” and “E” in frame 11 to assign that frame to the “j” segment. Note that the probability for “j” must be higher than the probability

for “E” or else the frame would have been assigned to the “E” segment instead.⁵ These probabilities approximate the true *a posteriori* probability of the phoneme classification given the acoustic evidence.

4.11 Statistical Issues

4.11.1 Sampling and Trials

A randomized sample of utterances and word models is drawn from the corpus. The sample size n (number of trials) is chosen to reduce the variance of performance statistics so that measured differences will be statistically valid. In each of the n trials an utterance is selected at random. The word model is generated for the correct word, recognition is performed, and a raw word score is computed by averaging the frame scores. An impostor (section 4.9.2) is also generated and scored. The true word score and the impostor word score are kept. Eventually there are n of each score.

Note that it does not matter whether the utterance is chosen first or the impostor candidates are chosen first. In any case, the impostor score represents an out-of-vocabulary speech recognition event, where the candidates represent the active vocabulary and the utterance is out-of-vocabulary with respect to that set.

4.11.2 Controlling Recognizer Error

If the task is to decide when the recognizer is right and when it is wrong, a simple decision can be rendered by simply looking at the recognizer accuracy based on past performance using training data. If it is highly accurate, then guess it is always accurate. If it is not as accurate, then guess it is always wrong. To normalize for recognizer accuracy the standard approach in this field is to look at correct and incorrect recognitions separately, and to compute a verifier error rate for each of these cases.

⁵This is a simplification. Segment duration and other constraints can also affect the score and segmentation assignments. There is a penalty applied for segments that are too short or too long compared to the proper duration for a segment of that type. For instance, if the proper duration is given as 30 to 200 msec, and the modeled duration is 250 msec, there will be 50 msec of too-long penalty applied to the word score. Similarly if the modeled duration is only 10 msec there will be 20 msec of too-short penalty applied to the word score.

In this thesis, a recognition event consists of matching a recorded utterance with a phonetic word model. The word model may be correct or incorrect. In any case, a score is generated. In a typical recognition setting the correct word model is not known in advance, and a number of possible word models (called the active vocabulary) are scored or partially scored until the best-scoring model can be determined. The score of any particular word model does not depend on the alternatives available. Each score is independent of all other scores, except that pruning of the search tree is performed to eliminate unlikely alternatives as soon as possible. Thus, some word models may not be scored at all. Pruning improves the speed of the search algorithm in return for a small risk of eliminating the true answer.

Because the score of any word model does not depend on the alternate word models, the correct word model score can be computed through a simple forced alignment procedure whereby the active vocabulary is restricted to just the correct word model. Scores from such alignments are called “true scores.”

Among incorrect recognitions, two alternatives are interesting. First is the case where the word that was spoken is not among the alternatives that were considered. That is, the word model that actually matches this utterance is not present in the active vocabulary. Second is the case where some incorrect word was recognized even though the correct word was also considered.

In the first case the utterance is called “out of vocabulary” because it does not match any of the models in the active vocabulary. In this thesis the word models and utterances are selected at random, and therefore it does not matter whether the utterance is selected first, and wrong word models are paired with it, or the word models are selected first and a wrong utterance is paired with them. In either case the recognition will result in the identification of one word model as “best.” This best word model, called the impostor, is still incorrect, but will be submitted to the verifier as though it might actually be correct.

The research task is to build, study, and improve verifiers that identify whether a recognition event represents a true recognition or the recognition of an impostor.

4.11.3 Out-of-Vocabulary versus In-Vocabulary Recognition Errors

The second case is where the word spoken is present in the active vocabulary, but one of the other word models (an incorrect one) scores better than the true word model. Such errors are called “in vocabulary” errors.

For word models and utterances taken at random, this second alternative is a subset of the first alternative. That is, the impostor score does not depend on the true score and vice versa. An impostor score does not depend on whether the true word is present in the vocabulary. The in-vocabulary recognitions simply represent a portion of one tail of the out-of-vocabulary distribution.

This is much different than recognition in a task-specific domain, where word models are not taken at random, but are related to the task. However it is the aim of this present research to look at the task-independent and vocabulary-independent evaluation of recognitions. No task-specific information is available. Therefore in-vocabulary recognition scores can be seen as a randomly generated subset of the out-of-vocabulary recognition scores, subject to the constraint that the impostor scores better than the true word. It is sufficient therefore to model the simple out-of-vocabulary case.

The fact that it is sufficient to model the simple out-of-vocabulary case may seem surprising until it is remembered that the research is restricted to vocabulary-independent and task-independent recognition. In this research each word model is independent of all others, including the correct word model.

In this thesis, performance is measured by the scoring of a large number of correct and incorrect recognitions. Each correct recognition is drawn at random from one of the corpora used (see sections 4.5.1 and 4.5.2). The corpora consist of recorded utterances and corresponding transcriptions created by skilled human listeners. The score for a word-model/utterance pair does not depend on what other word models or utterances might exist, so scoring can be done independently. In fact the score for the correct model and the score for the impostor are calculated separately, and indicated in Figure 4.1.

In this thesis the recognizer is forced to be right half the time (recognition is by forced alignment with only the correct word in the active vocabulary) and wrong half the

time (the active vocabulary does not have the correct word, but the size of this incorrect vocabulary can be set at various levels or “perplexities”).

This simplification avoids the confounding effects of recognizer accuracy. Some researchers attempt to measure the quality of a confidence measure by looking at confidence accuracy and then normalizing it for recognizer accuracy. This is reasonable since it is obvious that for a perfect recognizer, the confidence measure is trivial: always indicate 100% confidence in the recognition. By forcing some recognitions to be wrong in this research, I avoid the need to account for recognizer accuracy. Instead the performance of confidence and rejection algorithms is measured by the ROC curve which is independent of the recognizer accuracy.

4.11.4 Out-of-Vocabulary versus In-Vocabulary Recognition Errors

Several researchers distinguish between out-of-vocabulary (OOV) errors and in-vocabulary (IV) recognition errors. This may occur in a setting such as digit recognition. The present research does not make this distinction because the distributions of error scores do not seem to require such a split to account well for score distribution behavior. For example, the error distributions shown in Figure 7.1 on page 128 do not indicate bi-modality that requires separate underlying distributions. The uniformity of these curves may be a result of the vocabulary independence enforced in this research, and bi-modal distributions may apply in vocabulary-dependent task domains.

If the decision is made to distinguish between OOV and IV misrecognitions, several effects will naturally follow. The IV confusions will tend to have much better scores because they have been identified on the basis of having better recognition scores than the correct word models do. The OOVs (whatever is left over) will tend to have correspondingly worse scores. (This is simply the principle of adverse selection so familiar in the insurance industry.)

4.11.5 Histogram Creation

Histograms such as those in Figure 4.2 on page 65 are created in the following manner. All raw scores are reviewed and the highest and lowest are identified. The interval between

them is divided into n bins. Each raw score is examined and the appropriate bin count is incremented.

For the final histogram, smoothing is performed as follows. The count in each bin is reallocated with 25% going to the bin on the left, 50% to the original bin, and 25% to the bin on the right. This is done simultaneously for all bins.

The presentation of the histogram is done by connecting center-points of each bin. This method is chosen instead of the more common drawing of square corners for each bin because the squared histogram proved much more difficult to read, especially in areas where two lines were close to each other.

For some histograms the highest and lowest scores were not used, but a top and bottom of the range was chosen to better focus on the region of interest. This was helpful in cases where outliers caused the histogram to be compressed, thus obscuring interesting details.

4.11.6 The ROC Curve

The receiver operating characteristic (ROC) curve shown in Figure 4.4 arches from (0,0) to (1,1), showing the tradeoff between the Type I and Type II errors. The minimum verification error (MVE) is at the point of tangency on a 45° line that is tangent to the ROC. The equal error rate (EER) is at the point of intersection on a 45° line from (0,1) to (1,0). The figure of merit (FOM) is the area under the ROC curve.

Because the axes are error rates the presentation is normalized which makes it possible to visually compare two ROC curves to identify the better performance. Each raw score corresponds to some point on the curve, but raw scores are not presented explicitly.

Geometric characteristics of the ROC curve are used in comparing algorithms. The aspect used throughout this thesis is the equal error rate (EER). The MVE and FOM serve as alternatives to the EER in the comparison of algorithms.

In-depth discussion of ROC curves and likelihood ratios can be found in chapter 2 of Van Trees (1967).

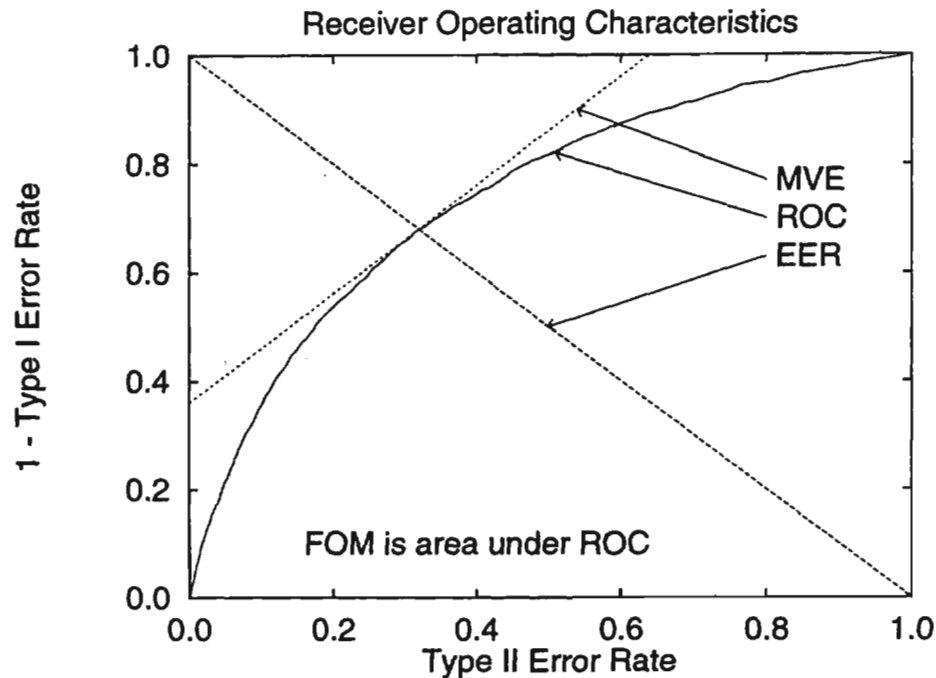


Figure 4.4: Annotated ROC Curve: The ROC arches from (0,0) to (1,1), showing the tradeoff between the two types of errors. The MVE is the point of tangency on a 45° line tangent to the ROC. The EER is the point of intersection on a 45° line from (0,1) to (1,0). The FOM is the area under the ROC curve.

4.11.7 Alternatives to EER: MVE and FOM

Figure of Merit: The figure of merit (FOM) is defined as the average accuracy across all Type I or Type II error rates. More simply this is the area under the ROC curve. Ideal performance produces a score of 1.0. As such it reflects total performance and not just the performance at one specific threshold. Random performance produces a score of .5. The residual error rate is 1-FOM.

Minimum Verification Error: The MVE generally occurs at or near the EER and is therefore approximately twice as great. The optimal decision point to minimize overall error depends on the relative frequency of impostor recognitions. When impostors occur half of the time the optimal point is the MVE. It lies on the equal-cost line which is $ec = T_1 + T_2$, where ec is chosen to make the line tangent to the ROC.

Minimum Cost Point: The optimal decision point to minimize overall cost depends on the relative costs of Type I and II errors. This varies by task and is beyond the scope of the current research. However, a variation in costs also has a geometric interpretation on the ROC curve. If the cost of a Type I error is \$5 and the cost of a Type II error is \$10, the equal-cost line will be $ec = 5T_1 + 10T_2$, where ec is chosen to make the line tangent to the ROC.

4.11.8 Bootstrap Parameter Estimation

In order to evaluate the stability of computed performance rates it is helpful to estimate the variance of the comparison metric. Because of the way the EER, MVE, and FOM are constructed it is difficult to give a closed-form specification of the variance. Instead statistical bootstrapping (Efron and Tibshirani 1993) is used to estimate variances and evaluate the difference in performance of two algorithms.

Bootstrap parameter estimation is performed as follows. Given a collection of n samples from which a single summary is computed (e.g., 16000 recognitions from which an EER is computed), select n samples from among the original n samples *with* replacement. Then compute the summary value again. Repeat this process a number of times (e.g., 200 times). The summary values thus computed can be examined to determine their distribution. In particular the summary values can be used to estimate their variance or a Monte Carlo confidence interval.

In the current research, central confidence intervals are computed by the standard-deviation method using Student's t distribution. Figure 4.5 shows the distribution in this one case is approximately normal. Due to the expense of computation only a limited number of experiments were evaluated to 200 bootstrap iterations, but in each case the distribution appeared to be normal. It is possible that by looking at more cases some non-normal distributions might be discovered.

it is seems reasonable to believe that most of the distributions are approximately normal. Further, the significance numbers reported in this thesis tend to extremes. They are either inconclusive (α is large) or highly conclusive (α is almost zero). The Monte Carlo confidence-interval method is not used because the distributions are believed to

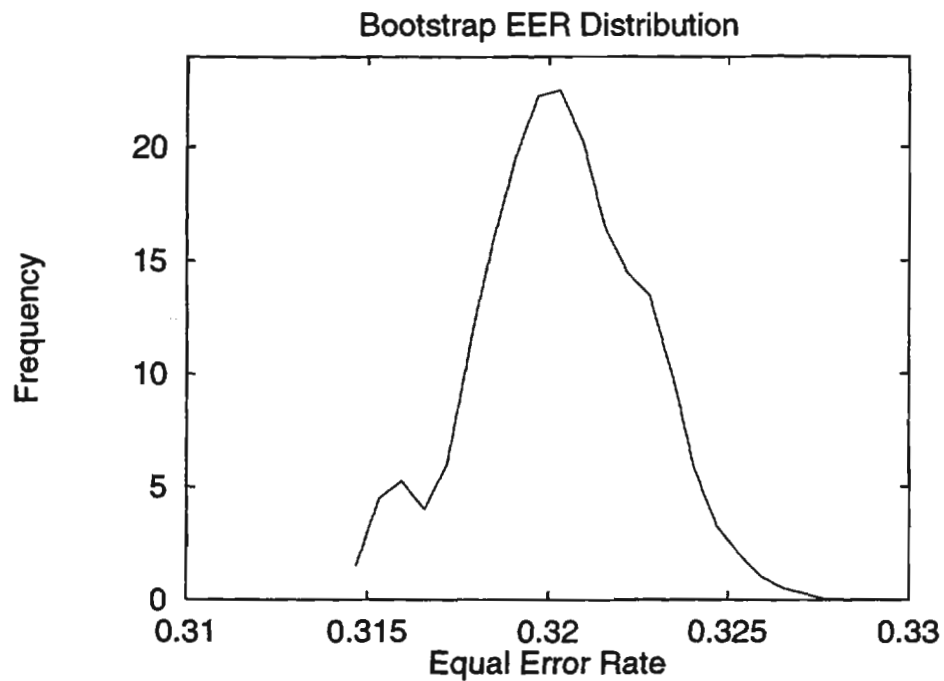


Figure 4.5: Bootstrap EER distribution for p^r : 200 bootstrap samples plotted from 0.31 to 0.33 in 32 steps. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, OGI Names corpus, raw probabilities, frame-to-word averaging, word models from Orator TTS, 16000 trials, final test set.

be approximately normal and the added cost of computing enough bootstrap scores was prohibitive.

Chapter 5

Baseline Vocabulary-Independent Experiments

In this chapter each section studies a technique known in the research literature, pushing it to peak performance for comparative purposes. For each experiment it tells the motivation and results and provides some discussion and conclusions.

The general methodology was described in chapter 4 and is not repeated here except to point out variations.

The following chapter (6) reports on new research in the area of rank-based probability estimation. The experiments in the present chapter form a baseline against which the performance of the rank-based approaches will be compared.

5.1 Different Corpora

Ideally the particular choice of a speech recognition corpus would not have any effect upon the ultimate evaluation of confidence or choice of thresholds for rejection. As much as possible the goodness of a particular raw score must be independent of the corpus from which it was drawn. One corpus may contain utterances that are difficult to recognize, due to recording conditions or to the nature of the utterances themselves. Another corpus may contain utterances that are enunciated more clearly and recorded under favorable circumstances. Among the best confidence and rejection algorithms, it is desirable that the ranking not depend upon the choice of corpus.

The two experiments in this section attempt to determine whether the results from sections 4.2 and 4.8 were likely to have been affected by the choice of the OGI Names

Table 5.1: Differences Across Corpora for Algorithms in the p^r Family. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, frame-to-word averaging, 16000 trials, final test set, equal error rates. OGI Names corpus uses word models from Orator TTS. NYNEX PhoneBook corpus uses word models from CMU dictionary.

PhoneBook		Difference				Names	
Algorithm	EER $\pm s_{\bar{x}}$	diff	t	df	α	EER $\pm s_{\bar{x}}$	Algorithm
p^r	.2216 \pm .0021	31%	31.60	398	.0000	.3200 \pm .0023	p^r
p^n	.2121 \pm .0020	38%	43.31	398	.0000	.3421 \pm .0023	p^n
$p^n/(1 - p^n)$.2453 \pm .0020	32%	37.10	398	.0000	.3621 \pm .0024	$p^n/(1 - p^n)$

corpus to perform the experiments. The first experiment in this section looks at the NYNEX PhoneBook corpus, and the second looks at an equal mix of OGI Names corpus and NYNEX PhoneBook corpus. It is concluded that the Mixed corpus is an appropriate base upon which to compare algorithms.

5.1.1 An Easier Corpus

The NYNEX PhoneBook corpus (see 4.5.2) provides another perspective from which measurements can be made. This corpus presents a relatively easy recognition task with utterances that are enunciated more clearly and recorded under more favorable circumstances than the OGI Names corpus.

Hypothesis: When experiments are rerun using the NYNEX PhoneBook corpus, the absolute results may vary but the relative results (one algorithm versus another) will be the same.

Results: Figure 5.1 shows a histogram of scores from the p^r algorithm on Names and PhoneBook. The impostor curves do not vary much, but the true curves do vary substantially. The PhoneBook corpus gives much better true scores. Does this translate into a better equal error rate?

Table 5.1 shows that performance on PhoneBook is 31% to 38% better than performance on Names by these algorithms. This indicates that PhoneBook is substantially easier to confirm or reject than Names.

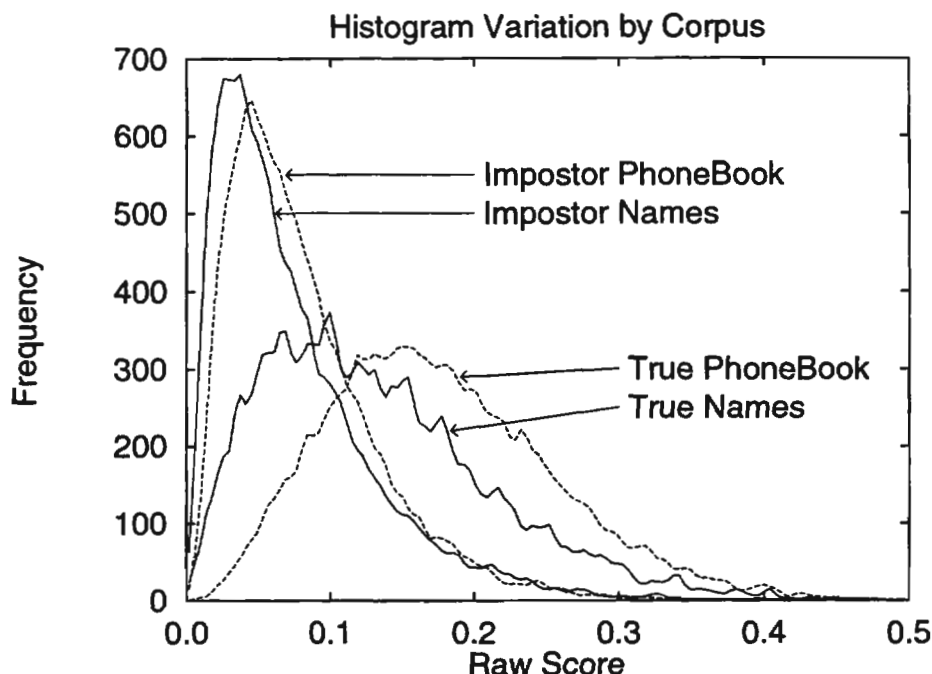


Figure 5.1: Distribution variation across Names and PhoneBook for Algorithm p^r . Trues show a large change in distribution between corpora. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, frame-to-word averaging, 16000 trials, final test set, equal error rates. OGI Names corpus uses word models from Orator TTS. NYNEX PhoneBook corpus uses word models from CMU dictionary.

Table 5.2 shows that the relative results (one algorithm versus another) are not the same: p^r and p^n swap positions in the line-up, although both beat $p^n/(1-p^n)$. This shows that the best-scoring algorithm may vary by corpus. Clearly there is some risk in doing all evaluations within just one corpus.

5.1.2 An Averaged Corpus

The relative performances of p^r and p^n are affected by the evaluation corpus used. Judging from the histograms the raw scores are compatible across corpora because the distributions almost coincide. A linear combination of the two corpora might serve better than either one alone. An equal mix will be examined.

Table 5.2: Corpus Differences Change Algorithm Rankings in the p^r Family. Note that p^r and p^n change positions in the rankings. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, frame-to-word averaging, 16000 trials, final test set, equal error rates. For more explanation see page 72.

NYNEX PhoneBook corpus using word models from CMU dictionary		
		$.2121 \pm .0020, p^n$
2		$.2216 \pm .0021, p^r$
17	11	$.2453 \pm .0020, p^n / (1 - p^n)$
equal mix of OGI Names corpus and NYNEX PhoneBook corpus		
		$.2755 \pm .0014, p^r$
2		$.2814 \pm .0015, p^n$
20	16	$.3063 \pm .0015, p^n / (1 - p^n)$
OGI Names corpus using word models from Orator TTS		
		$.3200 \pm .0023, p^r$
9		$.3421 \pm .0023, p^n$
18	7	$.3621 \pm .0024, p^n / (1 - p^n)$

Design: For this experiment no new recognitions are performed. Instead the raw scores from Names and PhoneBook are combined into a single list from which overall performance figures are determined. This analysis is equivalent to doing class-based recognitions where the separate corpora each represent a large class and the recognition is constrained to be within that class but the rejection thresholds are controlled globally. The results may have been different if the two corpora were mixed at recognition time because a different set of impostors might have been chosen.

5.1.3 Conclusions

Table 5.2 shows that the NYNEX PhoneBook corpus performs much better than the equal mix of OGI Names corpus and NYNEX PhoneBook corpus, which in turn performs much better than the OGI Names corpus. This shows that the rankings of algorithms one against another can change significantly based upon the corpus with which evaluations are done. (This may be an accident of the poor rejection capabilities of the algorithms viewed thus far.)

Compared to true performance in the field using real vocabularies, Names is believed to be too pessimistic and PhoneBook too optimistic. The combined corpus may more closely represent the actual recognition conditions that will prevail beyond the laboratory. It is not clear how this conjecture might be tested, so it will be taken as an assumption. **Combined performance is used hereafter for comparison among algorithms** because it is believed to be closer to expected real-world performance. Although some other linear combination of the two corpora is probably even better, it is not clear how to select the best combination. Therefore an equal mix (same amount from each) combination has been used.

The performance of p' ($.2755 \pm .0014$) is the best thus far.

5.2 On-Line Garbage Modeling

It is useful to compare rejection performance with existing methods. CSLU has an existing rejection mechanism installed in its CSLUsh toolkit and in CSLUrp, the CSLU Rapid Prototyper. The rejection system is based upon research by Boite, Bourlard, D'hoore, and Haesen (1993) using HMMs and has been in use at CSLU for several years. The experiments in this section are an attempt to express the CSLU rejection algorithm with a confidence score rather than a straight yes/no decision. Accordingly the score is defined as the threshold at which the yes/no decision changes. Except for that change, this is the CSLUrp rejection model.

The justification for examining this approach is that it works and is currently implemented. By measuring its performance it is expected that a baseline performance standard will be set against which future improvements can be compared.

In the discussion that follows, two word models will be considered. One is called the **target word model**. It represents a real word that is being evaluated for acceptance or rejection. It is scored by the recognizer and its Viterbi score becomes the **target word score**." The other is called the **<garbage> word model**. This is an artificial word composed of a sequence of **<garbage>** phonemes which are created similarly to the **<any>** model discussed in section 4.6.4. The **<garbage>** phoneme is defined as having the same neural

network output value as the median of the top n other phonemes (typically n defaults to 22). This is called the garbage median rank¹ or garbage rank. In each frame, the phoneme with the 12th highest value is identified, and its value is copied to become the value of the <garbage> phoneme.

Acceptance or rejection of the target word is based upon its whole-word Viterbi score, which includes all frames in the utterance. Specifically the frames that map to the <any> model (see section 4.6.4) are included in the score. Also since the score is made by adding the logarithms of the frame scores across the whole word and each frame score is a probability (i.e., usually less than 1.0) the scores tend to become more and more negative for longer and longer words.

To most closely follow the implementation of the CSLUrp system for rejection decisions, the <garbage> score is computed in the same way as the target word score, using the same number of frames. Then if the resulting target score is better than the garbage score the recognition is accepted. Otherwise it is rejected. For example, if the garbage median rank is 22, then the utterance will receive a garbage score based upon median rank 22. This utterance-specific score is the threshold for acceptance or rejection of any particular target word score for that utterance.

Adjustment of the rejection rate is achieved by changing the garbage median rank.

To compare this approach to others by using equal error rates it is necessary to convert each target word score to a common base. The most accurate way to do this is to compute for each target word the “garbage” median rank at which the word would be at the threshold between acceptance and rejection. This is called the “target” median rank. These estimated target median ranks are the unit of comparison across various utterances and target word models.

5.2.1 Estimating the Target Median Rank

One could compute the garbage score for all possible garbage median ranks, and then take the two scores closest to the target word score. Between these a simple linear interpolation

¹Median rank (mr) is related to rank (r) as follows: $mr/2+1=r$. $2(r+1)=mr$.

will result in an accurate estimated target median rank. (Alternately the closer, higher, or lower garbage median score could have been used. This would have resulted in quantization error and a loss of resolution, so it was not done.) In CSLUrp this is not done. Instead CSLUrp makes a simple accept/reject decision based on the threshold selection.

Computing several hundred garbage scores may be unnecessary. To discover exactly how many garbage scores should be computed, the actual plan is to select several different sets of ranks from which to interpolate, and then compare results. These chosen ranks are called “knot points” (or sample points) because they form the vertices along a piecewise linear curve that stretches from garbage median rank zero to garbage median rank 1000 (depending upon the number of ANN outputs in the recognizer). That is, they are the points at which the linear interpolation segments are “tied” together.

The intuition is that interpolation based on some knot-points may be more reliable and have higher rejection performance than interpolation based on other knot-points. This would be because some ranks are more reliable than others. In particular, a middle range of ranks, for example 10 and 30, might be more reliable than the top ranks (2, 4, 6) or the bottom ranks (beyond 40). Scores based on the top ranks are feared to jump around (be too variable). Scores based on bottom ranks are masked by the effects of the *<any>* and *<garbage>* models. This intuition will be tested to see whether accuracy differs depending on the set of knot-points employed.

The two knot-point garbage scores closest to the Viterbi score of the target word are used in linear interpolation (or extrapolation) to estimate the equivalent garbage rank, which is the threshold at which the word score would equal the garbage score.

The piecewise linear model may not perform as well as a fitted smooth curve might, but it is monotonic and relatively easy to compute. Since all scores are derived in the same way the piecewise nature is not expected to have a large effect upon the final results.

To specifically explore the sensitivity of this approach in terms of the “knot-points” at which the piecewise linear model is constructed, a variety of knot-point sets is examined. It is shown that the performance is not sensitive to the choice of points. That is, different point sets yield the same rejection performance.

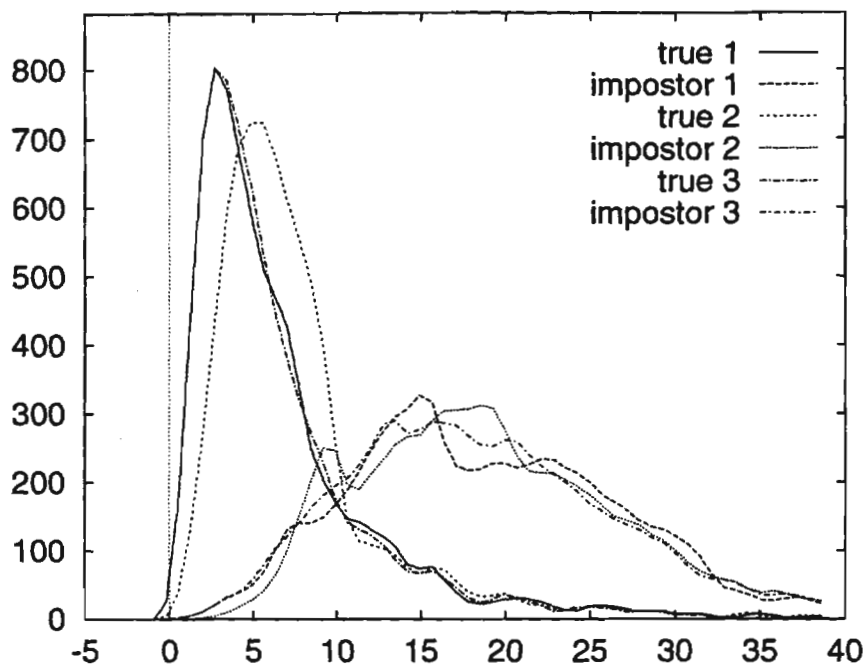


Figure 5.2: Histogram of Algorithms, 1: $g(0,2,4,8\dots)$; 2: $g(0,10,20\dots)$; 3: $g(0,2,4,6\dots)$. Notice that the histograms are nearly coincident for all three cases. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, whole-utterance on-line garbage scoring, word models depending on corpus, 32000 trials, final test set, equal error rates.

5.2.2 Initial Knot-point Experiments

The following experiments were performed.

$g(0,2,4,6\dots)$: on-line garbage piecewise linear interpolation with knot-points at 0, 2, 4, 6, 8, 10, 14, 18, 22, 30, and 50: The best-estimate approach is to compute the garbage score for all possible median values. Due to the way the median value is mapped to an actual rank (right-shift by one) only even numbers need be tried. For each utterance, a Viterbi score is computed using a garbage model at each of the following median ranks: 0, 2, 4, 6, 8, 10, 14, 18, 22, 30, and 50. By observation it was discovered that most wrong scores are less than median rank 50. A histogram is shown in Figure 5.2. Performance is shown in Table 5.3 to be $.1557 \pm .0014$.

Table 5.3: Mean, Standard Deviation, and 95% Confidence Intervals for Algorithms in the $g(a,b,c,\dots)$ Family. Notice that performance is nearly identical for all cases. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, whole-utterance on-line garbage scoring, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 67.

Algorithm	mean $\pm s_{\bar{x}}$	n	95% confid
$g(4,16)$.1554 \pm .0013	50	.1528-.1580
$g(0,4,16)$.1554 \pm .0013	50	.1528-.1580
$g(0,10)$.1555 \pm .0014	50	.1527-.1583
$g(0,2,4,8,\dots)$.1556 \pm .0014	50	.1528-.1585
$g(0,10,20,\dots)$.1557 \pm .0014	50	.1530-.1585
$g(0,2,4,6,\dots)$.1557 \pm .0014	50	.1529-.1586

$g(0,2,4,8,\dots)$: on-line garbage piecewise linear interpolation with knot-points at 0, 2, 4, 8, 16, 32, and 64: This next selection of knot points is exponentially spaced across the region where scores are expected to fall. For each utterance, a Viterbi score is computed using a garbage model at each of the following median ranks: 0, 2, 4, 8, 16, 32, and 64. A histogram is shown in Figure 5.2. Performance is shown in Table 5.3 to be .1556 \pm .0014. This spacing seems to improve the accuracy slightly, but the difference is not statistically significant.

$g(0,10,20,\dots)$: on-line garbage piecewise linear interpolation with knot-points at 0, 10, 20, 30, 40, and 50: This selection of knot points is spaced equally (rather than exponentially) across the region where scores are expected to fall. For each utterance, a Viterbi score is computed using a garbage model at each of the following median ranks: 0, 10, 20, 30, 40, and 50. A histogram is shown in Figure 5.2. Performance is shown in Table 5.3 to be .1557 \pm .0014.

5.2.3 Dramatically Fewer Knot Points

None of the preceding knot-point sets varied much in its final performance result. A much smaller number of knot points may affect performance. It is not clear *a priori* whether

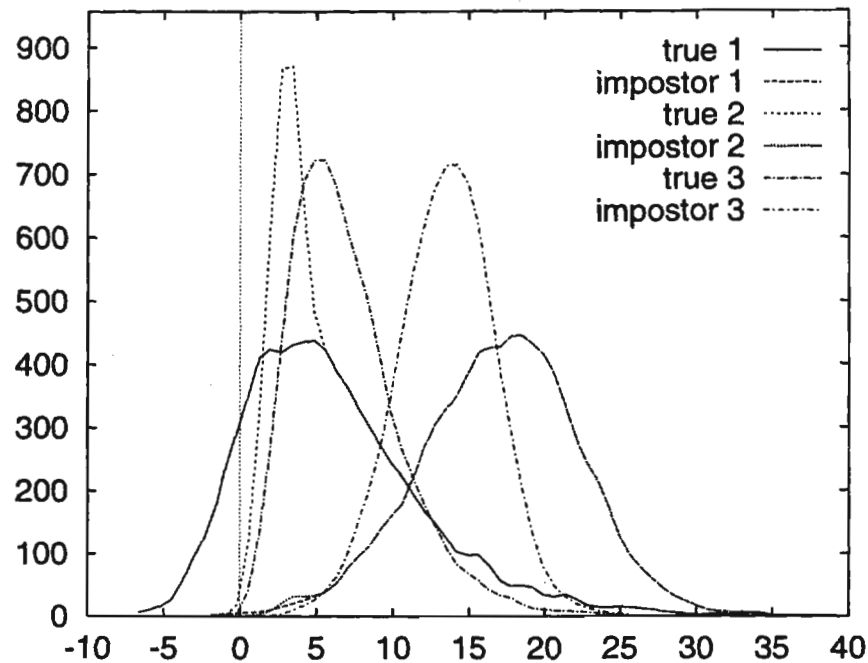


Figure 5.3: Histogram of Algorithms, 1: $g(4,16)$; 2: $g(0,4,16)$; 3: $g(0,10)$. Notice that the histograms are much different from those in Figure 5.2, which shows that the choice of knot points has a big influence on the eventual raw scores. However the performance does not change significantly. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, whole-utterance on-line garbage scoring, word models depending on corpus, 32000 trials, final test set, equal error rates.

the performance will be better or worse, as fewer points cannot follow the data as well, but more points may be overfitting. And ultimately it may not be statistically significant either way.

$g(0,4,16)$: on-line garbage piecewise linear interpolation with knot-points at 0, 4, and 16: This selection of three knot points is spaced exponentially across the region where most scores are expected to fall. For each utterance, a Viterbi score is computed using a garbage model at each of the following median ranks: 0, 4, and 16. The histograms in Figure 5.3 are much different from those in Figure 5.2, which shows that the choice of knot points has a big influence on the eventual raw scores. However the performance,

Table 5.4: Distance Chart for Algorithms in the $g(a,b,c...)$ Family. Notice that performance is nearly identical for all cases. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, whole-utterance on-line garbage scoring, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 72.

$.1554 \pm .0013, g(4,16)$					
0	$.1554 \pm .0013, g(0,4,16)$				
0	0	$.1555 \pm .0014, g(0,10)$			
0	0	0	$.1556 \pm .0014, g(0,2,4,8...)$		
0	0	0	0	$.1557 \pm .0014, g(0,10,20...)$	
0	0	0	0	0	$.1557 \pm .0014, g(0,2,4,6...)$

shown in Table 5.3 to be $.1554 \pm .0013$, has not changed significantly.

$g(4,16)$: on-line garbage linear interpolation with knot-points at 4 and 16:

This selection of two knot points is spaced exponentially across the region where most scores are expected to fall. For each utterance, a Viterbi score is computed using a garbage model at each of the following median ranks: 4 and 16. A histogram is shown in Figure 5.3. Performance is shown in Table 5.3 to be $.1554 \pm .0013$.

$g(0,10)$: on-line garbage linear interpolation with knot-points at 0 and 10:

This selection of two knot points is spaced linearly across the region where most true scores are expected to fall. 10 is near the dividing point between trues and impostors. For each utterance, a Viterbi score is computed using a garbage model at each of the following median ranks: 0 and 10. A histogram is shown in Figure 5.3. Performance is shown in Table 5.3 to be $.1555 \pm .0014$. The apparent slight loss in performance might be attributable to using 0 instead of 4 as the first knot-point.

5.2.4 Conclusions

The selection of knot points does not seem to affect the accuracy of the on-line garbage modeling technique. Table 5.4 tells the story. None of the differences is significant. In fact, each of the differences has better than 8 chances in 10 of occurring naturally even if

no actual difference exists. Because it is impossible to tell apart these performances based upon equal error rate alone, $g(4,16)$ is designated as the representative of this group based upon its simplicity of implementation, using only a single line to remap any Viterbi score into its estimated target median rank.

The on-line garbage approach of $g(4,16)$ achieves a performance of $.1554 \pm .0013$, which is dramatically better than the performance of p^r ($.2755 \pm .0014$). This is probably due to the summing of logarithms in computing the recognition score, as opposed to the simple-minded averaging of raw probabilities. Section 5.3 looks into this question.

5.3 Log Averages

The three simple algorithms presented in chapter 4 and in section 5.1 averaged probabilities directly. The impostor histograms in Figure 5.1 are sharply skewed, and the true histograms are somewhat skewed also. A logarithmic transformation may render curves that are more normal. Independent probabilities are always combined by multiplication to create joint probabilities, which suggests averaging in the logarithm domain. This type of averaging is also called geometric averaging. Because positive numbers are more convenient² for computation and logarithms of probabilities are not positive, the minus logarithm will be used.

The simple algorithms from chapter 4 will each be modified by taking the minus logarithm of the probability for the frame score. (Gillick, Ito, and Young (1997) refer to $-\log(p^n/(1-p^n))$ as the “logit” or “loglikelihood” function.)

This is shown to improve performance dramatically and will become a standard operation on probability-like frame scores.

Hypothesis: When experiments $-\log(p^r)$, $-\log(p^n)$, and $-\log(p^n/(1-p^n))$ are run, the histograms will be more normal and the performance will improve in comparison to p^r , p^n , and $p^n/(1-p^n)$.

²e.g., for taking another log, raising to a power, or geometric averaging of various types.

Table 5.5: Distance Chart for Algorithms in the $\log(p^r)$ Family Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, frame-to-word averaging, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 72.

$.1639 \pm .0012, -\log(p^r)$					
28	$.2110 \pm .0014, -\log(p^n)$				
28	0	$.2135 \pm .0015, -\log(p^n/(1-p^n))$			
43	32	31	$.2755 \pm .0014, p^r$		
43	33	32	2	$.2814 \pm .0015, p^n$	
47	39	38	20	16	$.3063 \pm .0015, p^n/(1-p^n)$

Because they are so different from each other, no hypothesis is made about the comparison of $-\log(p^r)$ with the $g(a,b,c,...)$ algorithms.

Results: Table 5.5 shows that $-\log(p^r)$ is clearly ahead of the other algorithms, and that $-\log(p^n)$ and $-\log(p^n/(1-p^n))$ are practically equal. It also shows that taking the logarithms of probabilities has produced a substantial improvement in rejection performance.

The histograms in Figure 5.4 show the raw scores the three algorithms. The curves are much more normal in shape than those in Figure 5.1. Notice that $-\log(p^n)$ and $-\log(p^n/(1-p^n))$ are very nearly equal. Transformation to the log domain has washed out most of the differences between normalized probability and odds. $-\log(p^r)$ has higher variance but is much better separated than the other two. The normalized probabilities reduce the variance but increase the overlap between trues and impostors.

The performance of $-\log(p^r)$ ($.1639 \pm .0012$) falls 5% behind the equal error rate of $g(4,16)$ ($.1554 \pm .0013$), the top whole-word on-line garbage model ($t = 4.82, \alpha \leq 10^{-5}$).

The performance of $g(a,b,c,...)$ algorithms is hurt by the use of the utterance frames before and after those in the word model. That is, the score is based upon the entire utterance, including frames that are assigned to the $\langle \text{any} \rangle$ model before and after the word. It does not seem reasonable that the $\langle \text{any} \rangle$ model portions of on-line garbage are helping. At best the $\langle \text{any} \rangle$ model portions would provide random noise into the

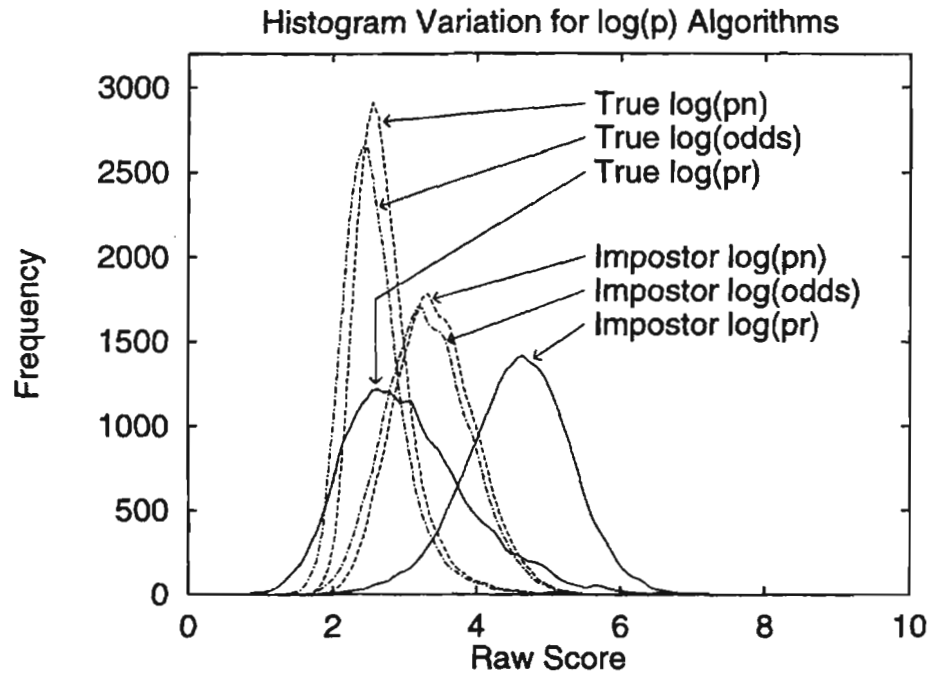


Figure 5.4: Distribution Variation for Algorithms in the $\log(p^r)$ Family. $\log(pn)$ is $-\log(p^n)$; $\log(odds)$ is $-\log(p^n/(1-p^n))$; $\log(pr)$ is $-\log(p^r)$. Notice that p^n and $p^n/(1-p^n)$ are nearly identical, and that p^r is substantially better. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, frame-to-word averaging, word models depending on corpus, 32000 trials, final test set, equal error rates.

measurements. It must be something else.

The other aspect is the normalization that is taking place in the $g(a,b,c,...)$ algorithms by using <garbage> phoneme scores as a point of comparison. This is examined further in section 5.5.

5.4 Segmental Averaging

Whole-word scoring is simple and effective, but there are alternatives that may perform better. Following is a list of ways that frame scores can be combined to make word scores. The method of averaging does make a substantial difference in performance, and is the focus of this section.

Hierarchical Averaging

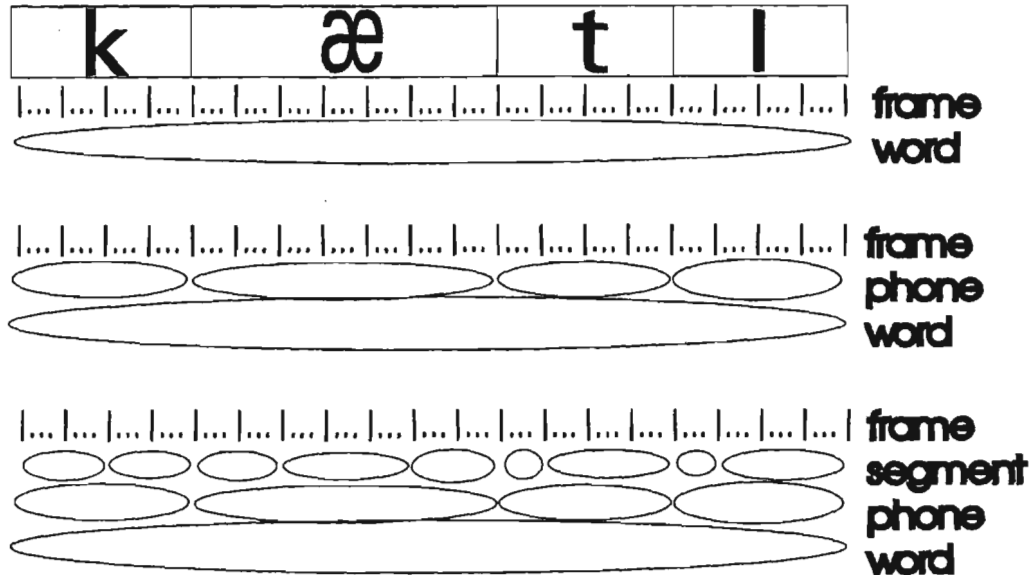


Figure 5.5: Hierarchical Averaging: The pronunciation model for the word “cattle” is shown. Below it frames are marked off. Three methods of averaging are illustrated. The first is frame-word averaging. In this method the scores from all frames are pooled directly to create one overall average. The second is frame-phone-word averaging. In this method the scores from all frames in a phoneme are pooled to create a phoneme average. These phoneme scores are then pooled to create the word score. The third is frame-segment-phone-word averaging. In this method the scores from all frames in a segment (phone state) are averaged to yield a segment score. The segment scores are averaged to yield a phone score. The phone scores are averaged to yield a word score.

Subsequent to the work done in this thesis, Weintraub, Beaufays, Rivlin, Konig, and Stolcke (1997) reported confidence metrics based on numerous features combined by an ANN. Some of these features are similar or identical in nature to those used in the hierarchical averaging approaches of this thesis. These features include averaging by word, phone, phone-state, or any combination of these. The interested reader is referred to section 6.1.1 of their paper.

Figure 5.5 shows three ways to accumulate frame scores into a final word score. The example shown is the word “cattle” using a word model of /k/, /æ/, /t/, /l/. Below

the word model there is marked off individual frames, with vertical lines separating the frames.

The simplest approach is to directly combine all the frame scores into a word score. This is illustrated by the oval that extends from the first frame to the last. This is called frame-word (fw) averaging.

Another approach is to average the frames across each phoneme, and then to average these phoneme scores across the word. The four smaller ovals represent phone averages. The larger oval below them represents a word average. This approach is called frame-phone-word (fpw) averaging. It illustrates a hierarchy of averages, where scores are rolled up a level at a time.

The final approach shown in this figure is fspw (frame-segment-phone-word) averaging. Each segment is a part of a phoneme. The /k/ is divided into two parts: closure (silence) and burst. The /ae/ is divided into three parts: ae following a k-class phoneme, central ae, and ae preceding a t-class phoneme. I say k-class and t-class because there are eight contexts used in the latest recognizer involved in this research. All possible phonemes that can precede ae are grouped into eight classes based on the similarity of their effect on the first part of the ae. The /t/ and /l/ are each modeled as two-state phonemes, with the frames of each state being collected into a unit called a segment.

The on-line garbage approaches of section 5.2 do not immediately lend themselves to a different accumulation strategy. Algorithm $-\log(p^r)$ is used as a baseline in this section because it is the best-performing other algorithm seen to this point.

Results for the $-\log(p^r)$ algorithm across five accumulation methods are presented in Figure 5.6 (histograms), Table 5.6 (pairs), and Table 5.7 (Distance Chart).

fw: frame-to-word averaging: Thus far raw word scores have been computed by averaging across whole words, with each frame contributing the same amount to the final score. This method is denoted fw for “frame to word.” Figure 5.6 shows that although fw has the smallest variances, it also has the worst separation of trues from impostors.

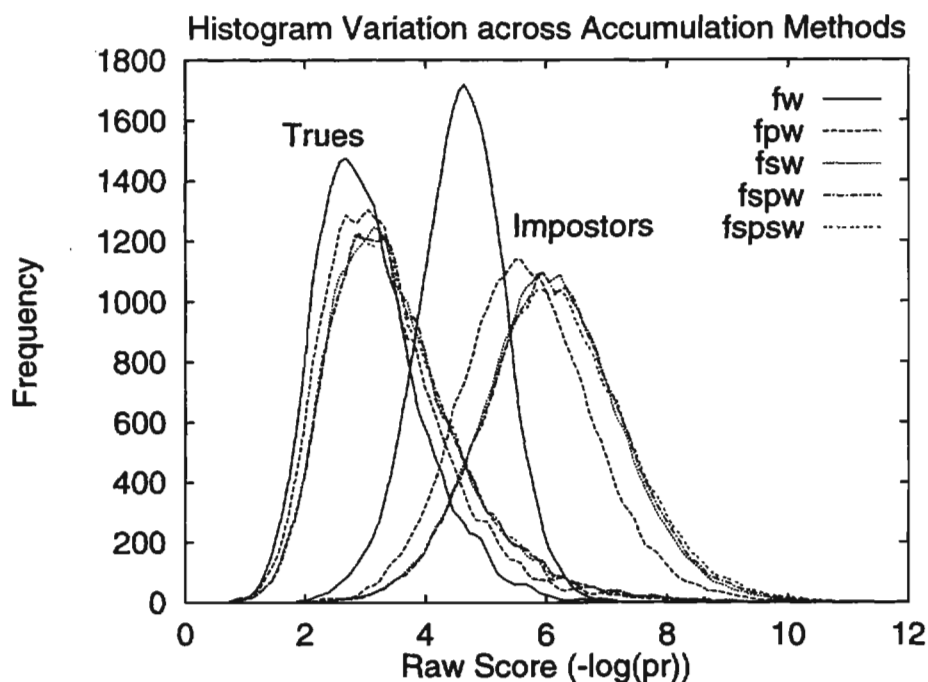


Figure 5.6: Histogram variation across accumulation methods for the $-\log(p^r)$ Algorithm. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, minus logarithm of raw probabilities, word models depending on corpus, 32000 trials, final test set, equal error rates. fw gets poor separation, fpw is better, and the best three are nearly identical.

fpw: frame/phoneme/word averaging: Rivlin, Cohen, Abrash, and Chung (1996) used a two-step averaging process to improve results. Their research averaged within phonemes to create a phoneme score, and then averaged the phoneme scores to get a word score. A phoneme is defined as a sequence of one or more frames that are associated with the same phoneme of the word model. This method is denoted fpw for “frame to phoneme to word.”

Figure 5.6 shows that fpw is dramatically better than fw, but all three of the other alternatives (fsw, fspw, and fspsw) are better still. Table 5.6 shows that fpw averaging improves results by 21% compared to fw averaging. This is a nice improvement. Table 5.7 shows that fpw is among the top group, and varies from the best methods by only a small amount.

Table 5.6: Accumulation methods pairwise comparison of performance for the $-\log(p^r)$ Algorithm. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, minus logarithm of raw probabilities, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 73.

Accum	Better	Difference				Worse	
	$EER \pm s_{\bar{x}}$	diff	t	df	α	$EER \pm s_{\bar{x}}$	Accum
fspw	.1233 \pm .0013	1%	0.96	98	.3418	.1252 \pm .0014	fspsw
fspw	.1233 \pm .0013	1%	0.98	98	.3303	.1252 \pm .0013	fsw
fspw	.1233 \pm .0013	5%	3.18	98	.0020	.1294 \pm .0013	fpw
fspw	.1233 \pm .0013	25%	22.35	98	.0000	.1639 \pm .0012	fw
fspsw	.1252 \pm .0014	0%	0.00	98	1.0000	.1252 \pm .0013	fsw
fspsw	.1252 \pm .0014	3%	2.16	98	.0331	.1294 \pm .0013	fpw
fspsw	.1252 \pm .0014	24%	20.87	98	.0000	.1639 \pm .0012	fw
fsw	.1252 \pm .0013	3%	2.21	98	.0292	.1294 \pm .0013	fpw
fsw	.1252 \pm .0013	24%	21.41	98	.0000	.1639 \pm .0012	fw
fpw	.1294 \pm .0013	21%	18.95	98	.0000	.1639 \pm .0012	fw

fsw: frame/segment/word averaging: Phonemes work well as an intermediate averaging point, but there are several other alternatives, including segments (ANN outputs) and syllables. Recognition itself is performed on the basis of ANN outputs which are sub-phonetic segments rather than directly with phonemes. A segment is defined as a sequence of one or more frames that are associated with the same ANN output in the word model. Segments may represent phonemes, phoneme halves, or phoneme thirds. Table 4.1 presents a list of these segments for the Oct96 recognizer. Computationally it is more convenient to work directly with segments. This method is denoted fsw for “frame to segment to word.” Table 5.6 shows that fsw averaging improves results by about 3% compared to fpw averaging.

fspw: frame/segment/phoneme/word averaging: Recognition can be viewed as a multi-level hierarchical activity, with frames collected into segments, segments into phonemes, phonemes into syllables, syllables into morphemes, morphemes into words,

Table 5.7: Distance Chart comparing accumulation methods for the $-\log(p^r)$ Algorithm. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, minus logarithm of raw probabilities, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 72.

<i>.1233±.0013, fspw</i>				
0	<i>.1252±.0013, fsw</i>			
0	0	<i>.1252±.0014, fspsw</i>		
2	1	1	<i>.1294±.0013, fpw</i>	
26	25	25	23	<i>.1639±.0012, fw</i>

and words into compound words. Method fspw moves further in this direction by averaging frames to get segment scores, averaging those to get phoneme scores, and averaging those to get word scores. Table 5.6 shows that fspw results are not significantly different from those for fsw.

fspsw: frame/segment/phoneme/syllable/word averaging: Moving closer to the full hierarchical structure, it is interesting to consider averaging across syllables. This is more difficult because the word models do not always give syllable divisions. Instead an algorithm was used to cluster phonemes into syllables. The algorithm is based upon rules by Kreidler (1989) and run as follows.

1. Vowel phonemes (3r|U|u|oU|aU|A|aI|>i|~|**0**|E|ei|I|i:)³ are designated to be proto-syllables. Diphthongs are not divided because they already represent a single phoneme. Adjacent vowels in different phonemes are established as separate syllable nuclei.

2. Zero or one liquids (j|9r|w|l) that occur immediately before proto-syllables are merged in, making those proto-syllables larger.

3. Zero or one (b|d|g|ph|th|kh|tS|dZ|f|S|T|D|v|z|h|d_(|j|m|n) that occur immediately before proto-syllables are merged in next.

4. Zero or one (s) that occur immediately before proto-syllables are merged in next.

³For a definition of the phonemes, please see Table 4.3.

5. Zero or one (S) that occur immediately before (m|n) in proto-syllables are merged in next. These occur in words like Schneider.

6. All unused (b|d|g|ph|th|kh|tS|dZ|s|f|S|T|D|v|z|h|d_(|j|9r|w|l|m|n) that occur immediately after proto-syllables are merged in. At this point all phonemes have been merged into proto-syllables, which can now be called syllables.

7. Occurrences of (9r 1) are split into separate syllables. These occur in words like girl, charles(ton), and carl. This is a dialect-specific issue and could be done with or without a syllable boundary in these contexts. This seemed a good place to start.

This overall algorithm as stated seems to work well with word models from Orator TTS and word models from CMU dictionary, which are used with the Names and PhoneBook corpora respectively. It was spot-tested on a number of words and seemed to have a high accuracy rate. This suggests that it would give a performance indicative of its full value had greater care been taken. The algorithm was not extensively tested.

Table 5.6 shows that fspsw results are about 1% worse than fspw ($\alpha = .1252$) which is not a significant difference. This performance did not seem to justify additional careful study of syllable clustering algorithms at this time.

Conclusions: Table 5.7 shows that for the $-\log(p^r)$ algorithm any type of sub-word averaging is clearly a big win in comparison to fw averaging. This is believed to be due to the presence of insertion-type errors which have been observed during review of impostor segmentations. The review is not dramatically conclusive and is not presented in this thesis but suggests that impostor segmentations often contain short phonemes with very bad scores amid much longer phonemes that are largely correct. By averaging across phonemes each phoneme or segment is treated equally so the longer ones no longer overpower the short ones.

By extension this conjecture would imply that averaging across sub-word units will help if the units are of substantially varying length. (With units of roughly equal length averaging will have no effect.) This seems to be borne out by the good performance of fspw which continues to be unsurpassed among the results yet to be reported in this thesis. It merges a widely varying number of frames into each segment, and merges from one to

three segments into a phoneme.

However it is disappointing that the fspw method with syllables of greatly varying length does not make a further improvement. This could be due to an incorrect approach to identifying syllable boundaries, or an inappropriate choice of test corpora. In any event, the difference is not significant nor is it large.

Based on these conclusions **performance using fspw is presented hereafter for comparison among algorithms.**

The frame/segment/phoneme/word averaging performance of $-\log(p^r)$ ($.1233 \pm .0013$) is better than its frame-to-word averaging performance ($.1639 \pm .0012$). The use of segmental accumulation strategies accounts for this improvement. The performance even surpasses the whole-utterance on-line garbage scoring performance of $g(4,16)$ ($.1554 \pm .0013$), the top whole-word on-line garbage model. It seems possible that segmental accumulation coupled with <garbage>-based normalization might create a further improvement. This is examined in section 5.5.

5.5 On-Line Garbage Improved

The concept here is to normalize each frame score p^r by some identifiable score or group of scores in the frame. This is like comparing to the whole-word garbage score at an estimated rank (section 5.2), but differs in several respects. First, the normalization occurs on a frame by frame basis rather than a whole word (or whole utterance) at a time. The use of frame-based normalization makes it possible to average within segments, which has been shown in section 5.4 to improve performance. Second, the equivalent rank is not computed, but rather by how much the frame score differs from some specified score. Third, the <any> modeled portions of the utterance are not included in the calculation, thus removing any noise they may have been contributing.

Normalization in this way bears a resemblance to acoustic normalization required by Bayes rule: $p(W|A) = p(A|W)p(W)/p(A)$. In this formulation $p(A|W)$ is normally provided by an HMM and is often called a likelihood. $p(W)$ is the (*a priori*) probability of occurrence for word W and is often provided by a language model. The probability

$p(A)$ of the observed acoustics A is often neglected in choosing the best word hypothesis because it is the same for all word hypotheses for that utterance (i.e., the acoustics are the same no matter what words are hypothesized). In theory $p(A)$ can be computed by summing all the $p(A|W)p(W)$ since the total probability is 1.0 by definition. In practice there are too many words W to be considered. If phonemes or sub-phonetic units are used instead of words it becomes possible to sum them all. $p(A)$ might also be estimated (modulo an unknown constant multiplier) by the methods of this section.

Because of restrictions in the training of the ANNs used as recognizers in this thesis (see section 4.1.2), it is possible that the *a posteriori* probabilities generated by the ANN are not fully *a posteriori* at all, but could still benefit from such a normalization as this. If on the other hand they are true *a posteriori* probabilities, the value $p(A)$ estimated by the methods of this section should be approximately constant and will therefore have little or no effect on performance.

5.5.1 Initial Experiments

The first experiments were performed normalizing against scores at median 10, 20, and 50 (ranks 6, 11, and 26 respectively). Low⁴ median values were chosen because they were expected to be more stable, and thus better normalization factors. Part A of Table 5.8 shows that the EER varies across these experiments and that $\log(p^r/g(R6))$ performed the best of the three at $.1138 \pm .0013$.

5.5.2 High Ranks

Because the highest rank seemed to perform better, additional experiments were performed at ranks 1, 2, 3, 4, and 5, to study how performance varies with rank. Part B of Table 5.8 shows that $\log(p^r/g(R2))$ performs the best (nominally) at $.1118 \pm .0013$, but that there is not a statistically significant difference among these normalization alternatives.

⁴Rank 1 is the highest rank.

Table 5.8: Distance Chart for Algorithms in the $\log(p^r/g(\text{low}))$ and $\log(p^r/g(\text{high}))$ Families. Notice that higher ranks seem to produce better performance, but the top ranks all performed about the same. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, frame/segment/phoneme/word averaging, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 72.

Part A: $\log(p^r/g(\text{low}))$ Family

$.1138 \pm .0013, \log(p^r/g(R6))$		
1		$.1178 \pm .0013, \log(p^r/g(R11))$
10	6	$.1292 \pm .0014, \log(p^r/g(R26))$

Part B: $\log(p^r/g(\text{high}))$ Family

.1118±.0013, log(p ^r /g(R2))					
0	.1118±.0014, log(p ^r /g(R3))				
0	0	.1128±.0014, log(p ^r /g(R4))			
0	0	0	.1133±.0013, log(p ^r /g(R5))		
0	0	0	0	.1138±.0013, log(p ^r /g(R6))	
0	0	0	0	0	.1143±.0014, log(p ^r /g(R1))

5.5.3 Averages of High Ranks

Because averaging several numbers tends to reduce variability (e.g., improves the reliability), averaging the top few ranks seemed to promise further performance gains. Experiments were performed averaging ranks (1..2), (1..3), (1..4), and (2..3). Averaging was performed in the logarithm domain (the average of the log-probabilities of the specified ranks was subtracted from $\log(p^r)$). Part A of Table 5.9 shows $\log(p^r/g(R1..4))$ with performance of $.1115 \pm .0013$ emerging as the new nominal leader. The marginal improvement over $\log(p^r/g(R2))$ at $.1118 \pm .0013$ is not significant.

5.5.4 Wider Averages

The $\log(p^r/g(R1..4))$ average gave the most promising results, but the other averages were almost identical. Additional experiments were then performed averaging across ranks (1..10), (1..20), (1..30), (1..40), and (1..50) to assess the usefulness of larger groupings and the effects of lower ranks for computing the normalization factor. Part B of Table 5.9 shows that performance suffers significantly as the lower ranks become involved in the

Table 5.9: Distance Chart for Algorithms in the $\log(p^r/g(\text{few}))$ Family. Notice that ranges of the top ranks performed about the same, but that as lower ranks become involved performance declines. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, frame/segment/phoneme/word averaging, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 72.

Part A: $\log(p^r/g(\text{few}))$ Family

$.1115 \pm .0013, \log(p^r/g(R1..4))$			
0	$.1117 \pm .0015, \log(p^r/g(R2..3))$		
0	0	$.1119 \pm .0015, \log(p^r/g(R1..3))$	
0	0	0	$.1129 \pm .0014, \log(p^r/g(R1..2))$

Part B: $\log(p^r/g(\text{many}))$ Family

$.1137 \pm .0015, \log(p^r/g(R1..10))$			
0	$.1160 \pm .0014, \log(p^r/g(R1..20))$		
2	0	$.1192 \pm .0014, \log(p^r/g(R1..30))$	
4	2	0	$.1221 \pm .0012, \log(p^r/g(R1..40))$
6	4	2	0 $.1250 \pm .0013, \log(p^r/g(R1..50))$

averaging. This suggests that the lower ranks are not as good a standard for comparison as are the upper ones. These experiments substantiate a steady trend with (1..10) being best and (1..50) being worst.

5.5.5 Experimental Details

Motivation: On a frame-by-frame basis the frame probability can be normalized by another score to accentuate how much better or worse it is. The intuition is that ANN performance can be affected by the acoustic quality of the utterance. A noisy utterance can be expected to have worse scores overall than a clean utterance. An unusual speaker can be expected to have worse scores overall than a speaker who is similar to those used in training. It is hypothesized that all important scores will rise or fall together with the acoustic quality of the utterance, and that this fact can be exploited for reliable normalization. If the normalizing score is a consistent baseline (such as the on-line garbage score) then the revised score should indicate improvement over random chance, given the waveform present in that frame.

Definition: The individual frame score f is computed by dividing the raw *a posteriori* probability p^r by a normalizing factor (the n th ranking score or an average of such scores in that same frame). The identities of the normalizing scores are varied across experiments. Specifically the $f = \log(p^r)$ minus the mean of the logarithms of the scores at the normalizing ranks.

Hypothesis 1: Normalizing by a garbage score computed in this manner allows discrimination between correct and incorrect recognitions.

Hypothesis 2: Segment-based averaging is more accurate than whole-word averaging.

Hypothesis 3: Performance varies significantly as a function of the normalizing scores used.

5.5.6 Discussion and Conclusions

Performance varies significantly as a function of the normalizing scores used. Across single-rank algorithms, the top ranks consistently outperform the lower ranks, except that rank 1 is apparently worse than ranks 2 through 6. The cause for this reversal is not understood.

Among rank-range algorithms, those concentrated in the highest ranks consistently perform best. The specific choice of ranks involved does not seem to be very sensitive.

As anticipated the combination of segmental accumulation and <garbage>-based normalization has created a further improvement. The performance of $\log(p^r/g(R1..4))$ (.1115 \pm .0013) is 10% better than the performance of $-\log(p^r)$ (.1233 \pm .0013).

Chapter 6

Vocabulary-Independent Rank-Based Algorithms

This chapter presents new research in the area of rank-based probability estimation. These experiments represent new research in the field of speech recognition. It continues the format of the previous chapter where baseline performance results were developed. It uses the general methodology described in chapter 4.

At the end of this chapter, a section of final results presents the top results from all the experiments that have been reported.

Rank by itself is an indicator of recognition quality. On a frame-by-frame basis, the ANN output used will have some rank R with respect to all ANN outputs p^r in that frame. (Note that R will be used to signal “rank.” This should not be confused with the use of r for “raw” which occurs in the context p^r .)

It is hypothesized that a rank of 1 means the same thing whether the absolute score p^r is 0.6 or 0.2. In particular, rank should be robust to acoustic variations in the incoming speech signal. The intuition is that background noise, microphone quality, and speaker enunciation affect the absolute scores much more than they affect the relative scores or rankings of the phonemes. If this is true then rank will be a better indicator of recognition accuracy than the absolute score is. This section will examine a family of algorithms based solely on the frame-by-frame rank of the phonemes in the word model.

Rank is computed in the most simple and obvious way. The ANN output value p^r is compared to all other values in that frame, and the number of values that are equal or greater becomes the rank. Ranks range from 1 (high) to 544 (low) for the Oct96 recognizer.

Given the rank, it is desirable to convert it back into some form of probability for accumulation, since I have already shown that averaging the logarithms of probabilities (see section 5.3) across segments and then phonemes (see section 5.4) gives a good performance.

The conversion to probability will be done using the *a priori* probabilities of observing those ranks in correct words or in impostors. This allows me to directly model the accuracy of phoneme ranks in the correct-word and out-of-vocabulary settings. The probabilities are trained using a corpus. The value p^r is not used except to determine the rank. Only the rank and the identity of the phoneme are used in computing the frame scores.

6.1 Estimating Probability Three Ways

Three ways are used to formulate probability for these experiments. The most obvious way is the likelihood ratio or odds ($p(\text{true})/p(\text{false})$). Other ways are simple probability ($p(\text{true})$) and cumulative probability.

6.1.1 $p(\text{true})$, $p(\text{false})$

The probability of truth and falsehood are defined differently than they were for $p^n/(1-p^n)$ in section 4.8.3. There the ANN output values were normalized in each frame, and the phoneme used by the word model (p^n) represented truth while the sum of all the rest ($1 - p^n$) represented falsehood.

$p(\text{true})$: Here the probability of truth is defined as the frequency of occurrence of some rank R across a training set of correctly recognized words. If the phoneme x occurs in 1000 frames in that training corpus, and if it has a rank of 1 in 270 of those frames, then $p(\text{rank}=1|\text{truth})$ is .27.

To compile these statistics, all utterances in the training set were used. Each word was recognized using its correct word model. For each frame two things were noted: (a) what is the correct phoneme (ANN output), and (b) what rank does it have.

After performing this forced alignment process for all words in the training set, the results were separated by phoneme (ANN output). Since there are 544 ANN outputs, this resulted in 544 separate lists. Each list gives the ranks that were observed when that

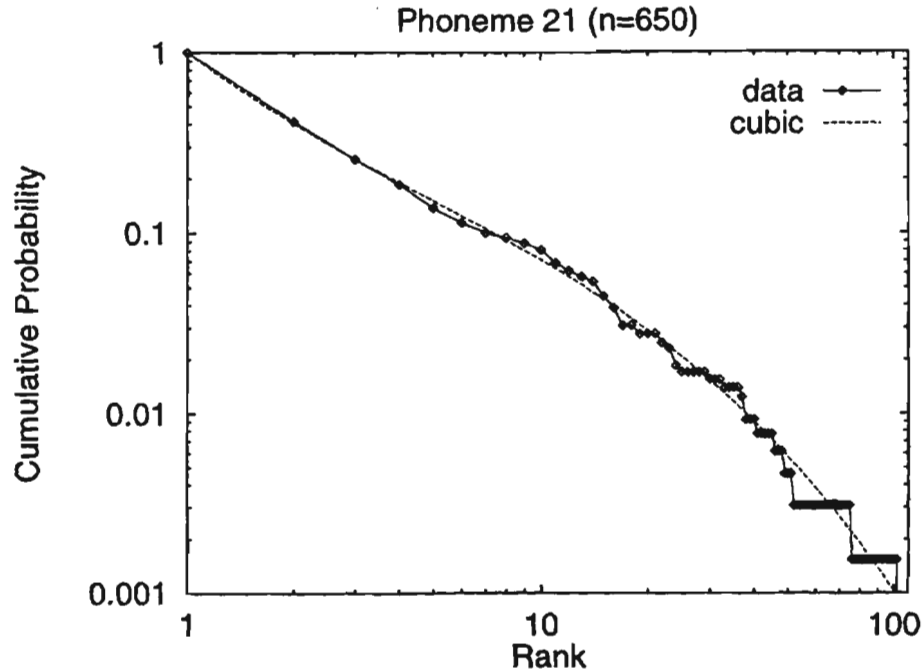


Figure 6.1: Cumulative Probabilities $\Sigma^M(R)$ for Phoneme 21. Notice the close fit for higher ranks. Ranks not shown had zero cumulative frequency. Details: Oct 1996 MFCC-based recognizer, all training set words from OGI Names corpus and NYNEX PhoneBook corpus, word models depending on corpus.

ANN output was actually true. The largest portion of the ranks were 1. Figure 6.1 shows the distribution of these ranks for ANN output 21, where there were 650 frames in the training set that were found to match that ANN output.

Figure 6.2 shows the distribution of these ranks for ANN outputs 3, 23, and 36. Phoneme 36 commonly gives inaccurate rankings even when the phoneme is correct. Phoneme 3 changes from accurate rankings at the high end to inaccurate rankings at the low end. Phoneme 23 gives consistently accurate rankings through the full range shown. This suggests that phoneme 23 is well trained in the ANN, but phoneme 36 is poorly trained.

p(false): The probability of falsehood is estimated across a training set as well. In this case, each utterance was falsely recognized as the best-matching word from a list made of

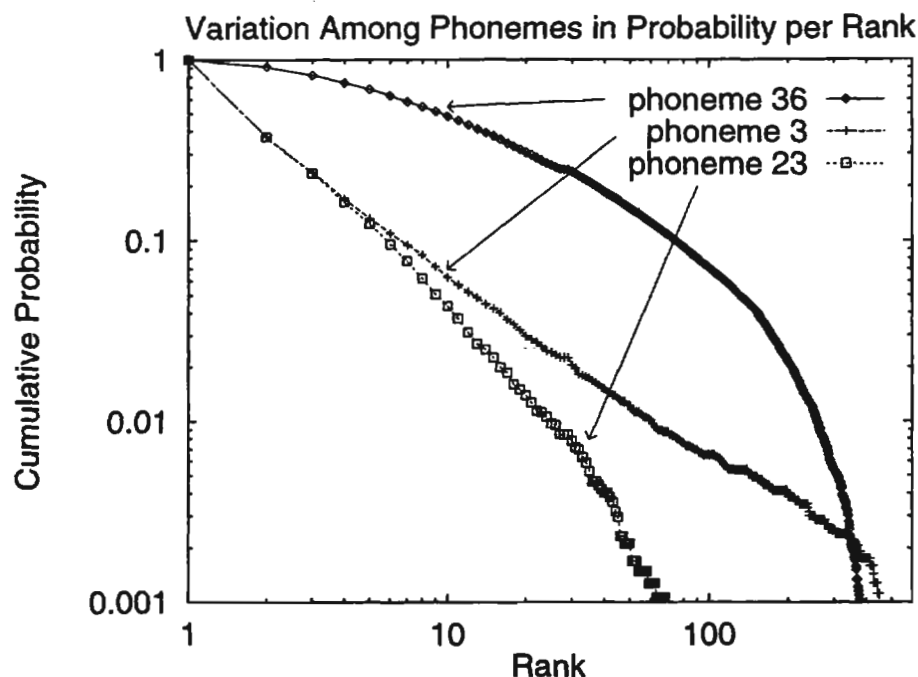


Figure 6.2: Variations Among Phonemes in Cumulative Probability $\Sigma^M(R)$ per Rank. Notice that phoneme 36 commonly gives inaccurate rankings even when the phoneme is correct. Phoneme 3 changes from accurate rankings at the high end to inaccurate rankings at the low end. Phoneme 23 gives consistently accurate rankings through the full range shown. Details: Oct 1996 MFCC-based recognizer, all training set words from OGI Names corpus and NYNEX PhoneBook corpus, word models depending on corpus.

random incorrect words.

Because of the time required to collect this list (many times longer than for the true words) some preliminary experiments were done to determine the best number of wrong words to place in the set from which the impostor would be drawn. Experiments were done with perplexity 2, 20, and 1000. It was seen that between 2 and 20, there was improvement at 20, but between 20 and 1000 the performance was not much different. This could be checked further but due to the time cost of the experiments I concluded based on early results that perplexity 20 would give an adequate indication of the merits of this approach.

For each impostor word, the frames were examined individually. The designated

phoneme (ANN output) was either correct or not. This was determined by comparison to the alignment of the correct word model for that utterance. If the impostor and the correct word model specified the same phoneme, then the phoneme was correct, even though it appeared in the impostor word. This occurred in the word pair “foil” versus “coil,” where the majority of the frames in the impostor were actually correct.

After the correct frames were eliminated, the remaining (incorrect) frames were again examined. For the phoneme used in the impostor, the rank was computed and added to a list for that phoneme. This resulted in 544 lists of ranks, where each rank was an actual observed rank for the phoneme in question, in an impostor word, and was not the true correct phoneme for that frame.

Choice of Impostors An important question remains unanswered. That is the question of how best to select impostors. Ideally the impostors would follow the same distribution what would occur naturally when an out-of-vocabulary utterance is given to a randomly chosen task-specific recognizer. Unfortunately it is not known how to create task-specific recognizers at random, and how to create the kinds of OOV utterances that such recognizers might encounter. Even the selection of a random set of tasks for which task-specific recognizers could be constructed seems intractable.

Therefore as a first approximation the utterances were chosen at random and applied to a task-independent vocabulary also chosen at random. However, this cannot be more than a first approximation due to the problems with correctly characterizing impostor recognitions.

Summary: The probability of falsehood is estimated across a training set of impostors at perplexity 20. Other perplexities were examined but the results do not seem to be particularly sensitive to this choice. The choice of impostors remains an important and unsettled issue.

If the phoneme x occurs in 1000 frames in that training corpus, and if it has a rank of 1 in 80 of those frames, then $p(\text{rank}=1|\text{falsehood})$ is .08.

In any frame where the impostor phoneme is the same as the true phoneme, the

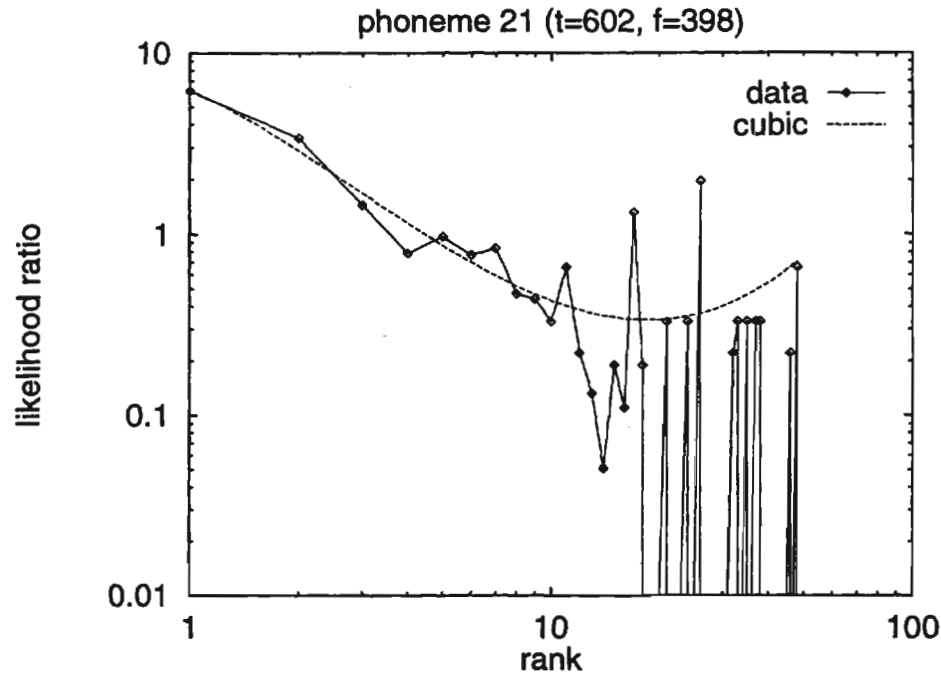


Figure 6.3: Likelihood Ratios $\ell^P(R)$ for Phoneme 21. Notice the poor fit for lower ranks. Ranks not shown had zero frequency. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, all training set words from NYNEX PhoneBook corpus, word models depending on corpus.

impostor is ignored. This helps prevent foil/coil problems, where the true word is “foil,” the impostor is “coil,” and the “oil” frames should not be counted as both true and impostor. Instead they are counted only as true.

6.1.2 Cubic Polynomial Smoothing

Few trues occur at low ranks. For that matter few falses occur at low ranks either. Smoothing is critical to estimate reasonable probabilities in the low-rank tail of these distributions. For each ANN output a separate probability curve was fitted, using a cubic polynomial taking the logarithm of rank as the independent variable and returning the logarithm of the probability. Examples are shown in figures 6.3 and 6.1.

6.1.3 Likelihood Ratio

Likelihood Ratio is denoted by $\ell^P(R)$. (The P indicates PhoneBook training.) It identifies a set of 544 cubic polynomials trained to estimate the logarithm of the likelihood ratio of the PhoneBook corpus training set given the logarithm of the rank.

The likelihood ratio in the above case would be $\frac{.27}{.08}$, which combines with the prior likelihood $\frac{p(t)}{p(f)}$ to yield the likelihood given the observed rank. The typical assumption is that truth and falsehood are equally likely so $\frac{p(t)}{p(f)} = 1$ and it cancels out of the equation leaving just $\frac{.27}{.08}$ as the likelihood given the observed rank.

Figure 6.3 illustrates the fit between data observed and the cubic polynomial. For most of the 544 phonemes the fit was better and n was larger but the tail of righthand the curve still came up. Much more data may be required to get a reliable distribution.

6.1.4 Simple Probability

Simple Probability is denoted by $S^P(R)$ (for PhoneBook training) or $S^M(R)$ (for Mixed training). The simple true probability in the above case would be .27. The probability of falsehood does not enter into the calculation. This is expected to be less accurate than the likelihood ratio, but given the fundamental problems with generation of impostors, simple probability is an interesting alternative worth examining.

6.1.5 Cumulative Probability

This is denoted by $\Sigma^P(R)$ (for PhoneBook training) or $\Sigma^M(R)$ (for Mixed training). Each identifies a set of 544 cubic polynomials trained to estimate the logarithm of the cumulative probability of the training corpus set given the logarithm of the rank.

The cumulative true probability is perhaps the most interesting alternative. It takes into account the belief that higher rank implies a better match. This seems obvious, but it is not used in either the likelihood ratio formulation nor in the simple probability formulation. In the cumulative formulation, probability is the sum of the simple probability at that rank and at all lower (worse) ranks. Thus by definition the cumulative probability of truth at rank 1 is always 1.0. In the above case, the cumulative probability at rank 2

Table 6.1: Distance Chart for Algorithms in the $f^P(R)$ Family. Notice that cumulative is nominally better but only by an insignificant margin. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, frame/segment/phoneme/word averaging, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 72.

		$.1195 \pm .0014, \text{Mean}(\Sigma^P(R))$
0		$.1225 \pm .0014, \text{Mean}(\ell^P(R))$
1	0	$.1230 \pm .0014, \text{Mean}(S^P(R))$

would be $1.0 - .27 = .73$. Figure 6.1 illustrates the fit between data observed and the cubic polynomial. For most of the 544 phonemes the fit was better and n was larger.

6.1.6 Estimating Simple Probability

To get the simple (non-cumulative) proportion of scores at a certain rank a “delta cumulative” approach is convenient. Because of sparse data in the lower ranks, and the convenience of having the cumulative curve already fitted, the probability at any rank R is estimated as the cumulative probability at that rank minus the cumulative probability at rank $(R + 1)$. This is exactly the original probability at that rank, but smoothed to adjust for sparsity of data.

6.2 Probability Training Corpus Selection

It is not immediately clear which approach should yield the best performance. The frame scores play together in complicated ways. A variety of experiments will be performed to try to create some intuition about the relative behaviors. The first experiment tests to see which of these probability formulations is best, or whether they are not distinguishable. The probabilities are trained using PhoneBook. Table 6.1 shows that cumulative is better by an insignificant margin. $\text{Mean}(\ell^P(R))$ was eliminated from consideration at this point because the approach is very computationally expensive, requiring 500 times more impostors to train compared to the cumulative approach, and not yielding any improvement in performance. Therefore, results were generated for $\ell^P(R)$ but not $\ell^M(R)$.

Table 6.2: Distance Chart for Algorithms in the $f^M(R)$ Family. Notice that Mixed training is significantly better than PhoneBook training. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, frame/segment/phoneme/word averaging, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 72.

$.1144 \pm .0014, \text{Mean}(\Sigma^M(R))$				
0	$.1171 \pm .0013, \text{Mean}(S^M(R))$			
2	0	$.1195 \pm .0014, \text{Mean}(\Sigma^P(R))$		
4	2	0	$.1225 \pm .0014, \text{Mean}(\ell^P(R))$	
4	2	1	0	$.1230 \pm .0014, \text{Mean}(S^P(R))$

Table 6.3: Distance Chart for Algorithms in the $f(R)$ Family. Notice that Mixed training still appears to be better than the PhoneBook training, although the results are not as significant. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, NYNEX PhoneBook corpus, frame/segment/phoneme/word averaging, word models from CMU dictionary, 16000 trials, final test set, equal error rates. For more explanation see page 72.

$.0587 \pm .0015, \text{Mean}(\Sigma^M(R))$				
0	$.0595 \pm .0012, \text{Mean}(\Sigma^P(R))$			
0	0	$.0617 \pm .0014, \text{Mean}(S^M(R))$		
3	2	1	$.0658 \pm .0015, \text{Mean}(S^P(R))$	
3	3	1	0	$.0659 \pm .0013, \text{Mean}(\ell^P(R))$

The second experiment tests whether using NYNEX PhoneBook corpus is better, or whether equal mix of OGI Names corpus and NYNEX PhoneBook corpus is better. Table 6.2 shows that Mixed provides significantly better training for both $\Sigma(R)$ ($\alpha=.0089$) and $S(R)$ ($\alpha=.0024$). This indicates that “more data is better.” However, it also raises a question on whether this result is due to testing with the Mixed corpus.

The third experiment tests whether these results hold up when tested against the PhoneBook corpus. That is, when the probabilities are trained on corpus x do they simply perform better on corpus x ? Table 6.3 shows that Mixed training still appears to be better than the PhoneBook training, although the results are not as significant. It is still reasonable to believe that Mixed training is better. The $\text{Mean}(\ell^P(R))$ turns in a

particularly poor showing on this set, which does not bode well for its long-term abilities.

6.3 Weighted Alternatives to Mean Accumulation

Up to this point averaging has been done in the ordinary way, with perhaps a change to the logarithmic domain to get a geometric mean. The geometric averaging has been shown to contribute to performance for the averaging of probabilities.

Review of the actual ranks obtained on a segment by segment basis showed that at the beginning and end of correct segments the ranks tended to be poor, but in the middle of each segment the ranks were high. This indicates that the ANN transitions are still a problem as the processing moves from segment to segment.

This section of experiments looks at several alternative ways to perform averaging. It is motivated by examination of the actual probabilities that make up the scores for trues and impostors. Based on visual observation it was hypothesized that impostors have a higher proportion of bad frame scores. To test this hypothesis three alternate forms of averaging were created. For each of these forms of averaging the raw probabilities are first sorted within the segment, and are then weighted according to their position in the sorted sequence. Better scores appear first and are weighted more lightly. Worse scores appear last and are weighted more heavily. Following are the weighting schemes used.

6.3.1 Mean Averaging

In mean averaging the weights are constant. For n frames, each is weighted by 1. The sum is divided by the sum of the weights (n). This is common, ordinary averaging. It is the baseline for comparison with the experimental forms of averaging.

The mean average score is roughly equal to one of the middle score in the set.

6.3.2 Triangular Averaging

In triangular averaging the weights increase by one for each additional item, starting from a base of zero. For n frames, the best is weighted by 1, the next by 2, then 3, and so on to the last which is weighted by n . The sum is divided by the sum of the weights ($\frac{n(n-1)}{2}$).

By doing a triangular form of averaging, the worse scores have much more effect on the final segment score than do the better scores. The idea is that better scores are simply expected and should not be rewarded, but the worse scores are a violation of expectations and should be penalized.

The triangular average score is roughly equal to one of the worse scores in the set.

6.3.3 Trapezoidal Averaging

In trapezoidal averaging the weights increase by one for each additional item, starting from a base of n . For n frames, the best is weighted by $n + 1$, the next by $n + 2$, and so on to the last which is weighted by $2n$. The sum is divided by the sum of the weights.

By doing trapezoidal averaging the good scores make a larger contribution to the average than under triangular averaging. Trapezoidal is a compromise between triangular and mean averaging.

The trapezoidal average score is roughly half way between the mean average score and the triangular score.

6.3.4 Parabolic Averaging

In parabolic averaging the difference between weights increases by one for each additional item. For n frames, the best is weighted by 1, the next by 2, then 4, then 7, then 11, and so on. The n th is weighted by $\frac{1}{2}x^2 - \frac{1}{2}x + 1$. The sum is divided by the sum of the weights.

By doing parabolic averaging the bad scores make a larger contribution than under triangular averaging. This is yet more extreme than triangular averaging, and approximates selecting the next-to-worst score as the representative for the entire segment.

6.3.5 Usage

Mean, triangular, trapezoidal, and parabolic forms of averaging are evaluated in the next two sections. In section 6.4 the rank numbers (1, 2, 3, ...) are averaged before converting to the probability domain. In section 6.5 the ranks are converted to probabilities first and then the averages are computed.

Table 6.4: Distance Chart for Algorithms in the $f(av(R))$ Family. Notice that more exotic averaging (trapezoidal, triangular, parabolic) has not improved performance. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, frame/segment/phoneme/word averaging, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 72.

$.1164 \pm .0013, \Sigma^M(\text{Mean}(R))$			
0	$.1165 \pm .0014, \Sigma^M(\text{Trap}(R))$		
0	0	$.1175 \pm .0014, \Sigma^M(\text{Tri}(R))$	
0	0	0	$.1182 \pm .0014, \Sigma^M(\text{Para}(R))$

6.4 Averaging Ranks

In this experiment the ranks themselves were averaged before computing the probability. For those cubic polynomials that are largely straight across the range of ranks involved, this will be the same as averaging the logarithms of the probabilities. In other cases it will make a difference. This experiment is motivated by the visual observations made while examining the phoneme ranks for trues and impostors. Those observations seemed to indicate that incorrect alignments had more bad scores than correct alignments did. I hypothesized that averaging the ranks would perform differently from averaging the probabilities (discussed in section 6.5).

Table 6.4 shows that exotic averaging (trapezoidal, triangular, parabolic) has not improved performance. In fact, as the weighting becomes more extreme the performance appears to drop more. Thus mean with the least weighting difference performs best, and parabolic with the most weighting difference performs worst. Unfortunately there is not enough accuracy in the numbers to draw solid conclusions. Therefore this observation is preliminary.

6.5 Averaging Probabilities

In this experiment the logarithms of probabilities were averaged. For those cubic polynomials that are largely straight across the range of ranks involved, this will be the same

Table 6.5: Distance Chart for Algorithms in the $av(f(R))$ Family. Table 6.4 results are included for comparison. Notice that computing probabilities before averaging seems to improve performance. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, frame/segment/phoneme/word averaging, word models depending on corpus, 32000 trials, final test set, equal error rates. For more explanation see page 72.

$.1144 \pm .0014, \text{Mean}(\Sigma^M(R))$					
0	$.1146 \pm .0014, \text{Trap}(\Sigma^M(R))$				
0	0	$.1149 \pm .0014, \text{Tri}(\Sigma^M(R))$			
0	0	0	$.1164 \pm .0013, \Sigma^M(\text{Mean}(R))$		
0	0	0	0	$.1165 \pm .0014, \Sigma^M(\text{Trap}(R))$	
0	0	0	0	0	$.1175 \pm .0014, \Sigma^M(\text{Tri}(R))$
1	1	0	0	0	$.1182 \pm .0014, \Sigma^M(\text{Para}(R))$

as averaging the ranks as reported in section 6.4. In other cases it will make a difference. I hypothesized that averaging the ranks would perform differently from averaging the probabilities (discussed in section 6.5).

Table 6.5 shows that computing log-probabilities before averaging seems to improve performance, and the exotic averaging (trapezoidal and triangular) still seem to fall behind the simple mean average. However, the differences between all these results are too weak to be conclusive.

6.6 Conclusions

It can be seen that $\text{Mean}(\Sigma^M(R))$ ($.1144 \pm .0014$) has good performance. In fact the performance is not significantly different from $\log(p^r/g(R1..4))$ ($.1115 \pm .0013$). It is interesting to see that rank alone is able to achieve this quality of result.

Exotic forms of averaging do not seem to improve performance. Averaging ranks rather than probabilities does not improve performance. Further, averaging that weights each item equally appears to perform better than averaging that emphasizes items with lower scores. Mean averaging appears to be best.

Cumulative probabilities show promise in comparison to simple probabilities and likelihood ratios, but the results are not conclusive. A larger number of trials is required to see whether these apparent differences are real.

6.7 Vocabulary-Independent Final Results

Table 6.6 presents the top results from all the experiments that have been reported in the previous several chapters on vocabulary-independent experiments. They are shown using three separate evaluation sets: Names, PhoneBook, and Mixed. To be concise only the top performers are presented, using the fspw and fw accumulation strategies and impostors at perplexity 20. Preliminary results using the development test set show that the algorithms best at this perplexity are also best at other perplexities. Final results were limited to perplexity 20 only.

Table 6.6: Distance Chart for the Top Algorithms. Details: impostors at perplexity 20, Oct 1996 MFCC-based recognizer, final test set, equal error rates. For more explanation see page 72.

NYNEX PhoneBook corpus

word models from CMU dictionary, 16000 trials

.0587±.0015, Mean($\Sigma^M(R)$) fspw							
0	.0591±.0015, log($p^r/g(R1..4)$) fspw						
3	3	.0664±.0015, -log(p^r) fspw					
13	13	7	.0803±.0016, log($p^r/g(R1..4)$) fw				
18	18	14	3	.0893±.0016, Mean($\Sigma^M(R)$) fw			
19	19	14	4	0	.0910±.0017, g(4,16) fw		
21	21	17	8	2	1	.0967±.0018, -log(p^r) fw	
37	36	35	32	29	28	26	.1524±.0017, p^r fspw
36	36	35	34	32	32	31	.2216±.0021, p^r fw

equal mix of OGI Names corpus and NYNEX PhoneBook corpus

word models depending on corpus, 32000 trials

.1115±.0013, log($p^r/g(R1..4)$) fspw							
0	.1144±.0014, Mean($\Sigma^M(R)$) fspw						
7	5	.1233±.0013, -log(p^r) fspw					
21	19	13	.1414±.0013, log($p^r/g(R1..4)$) fw				
24	22	18	4	.1503±.0015, Mean($\Sigma^M(R)$) fw			
27	26	22	10	2	.1554±.0013, g(4,16) fw		
30	29	26	17	9	5	.1639±.0012, -log(p^r) fw	
40	39	38	34	31	30	28	.2097±.0014, p^r fspw
49	48	48	46	43	44	43	.2755±.0014, p^r fw

OGI Names corpus

word models from Orator TTS, 16000 trials

.1648±.0023, log($p^r/g(R1..4)$) fspw							
0	.1661±.0019, Mean($\Sigma^M(R)$) fspw						
4	4	.1791±.0022, -log(p^r) fspw					
17	18	11	.2051±.0021, log($p^r/g(R1..4)$) fw				
18	19	13	1	.2117±.0025, Mean($\Sigma^M(R)$) fw			
21	22	17	4	1	.2176±.0022, g(4,16) fw		
24	26	21	10	6	3	.2294±.0022, -log(p^r) fw	
30	31	28	22	19	18	13	.2614±.0025, p^r fspw
34	34	32	29	27	27	25	.3200±.0023, p^r fw

Chapter 7

Confidence

Rejection by raw thresholds may be a completely adequate solution for many situations in automatic speech recognition. But “tuning” to find the right setting can be difficult. It can depend on the makeup and size of the impostor vocabulary, as well as the cost analysis of making different types of errors. Vocabulary independence and integration with higher processes such as a dialogue manager further increases the difficulty of using raw thresholds. Confidence provides a uniform approach to these issues.

This chapter completes the discussion of rejection by developing an actual confidence score that can for example guide higher-level decisions about dialogue processing.

7.1 Continuous Versus Discrete

The final use of any confidence and rejection calculation is probably a discrete decision to do one thing or do something else. It seems useful to view “confidence” as a continuum of scores with some designated threshold such that computed scores on one side are “good enough” (accepted) for some purpose, and on the other side they are “not good enough” (rejected). What information should be returned from a confidence and rejection calculation? Is a confidence measure necessary?

7.1.1 Accept, Verify, or Try Again

One approach to confidence and rejection is to set two thresholds. The best-scoring recognitions are automatically accepted. The worst-scoring recognitions are automatically rejected. In a voice response system, rejection would generally cause the prompt to be

repeated. The middle-scoring recognitions might be verified by a dialogue such as, “Did you say (the thing recognized)?”

At any given threshold there is some proportion of correct recognitions that will be rejected, and some proportion of incorrect recognitions that will be accepted. Depending upon the application, there may be different penalties for different system errors. For example, if the question is, “Did you say ‘Delete all files’?”, one might wish to err on the side of caution and only accept a “yes” that is clearly a “yes.” But if the response seems to be “no,” one might wish to generously accept it, possibly requiring a frustrated user to repeat the command. To err on the side of caution, it is clear that the making of an accept-verify-reject decision requires task-specific information.

The scope of this thesis is to develop general techniques that are applicable in a broad variety of settings. Therefore the verification (or “confidence”) component is designed to report the probability of some specified answer. Other components can be constructed as needed to respond to that assessment.

7.2 True Confidence

The goal is to create a confidence measure that can be used by other processes in a straightforward way. One obvious definition for confidence is the posterior probability that a given recognition event is correct. Because probabilities are equivalent to likelihood ratios,¹ and because prior probabilities and task-specific cost information may not be known, confidence will instead be presented as a likelihood ratio ℓ . Specifically $\ell(\text{score}) = \frac{p(\text{score}|\text{true})}{p(\text{score}|\text{impostor})}$. These probabilities can be estimated from a training set.

Notice that prior probabilities for words and larger constructs are not available for the analysis of this thesis. The availability of prior probabilities is limited to the occurrence of phonemes and strings of phonemes in English. Task-based or vocabulary-based prior probabilities together with updating information from the acoustic reliability scores (confidence measures) created here can be used to generate posterior probabilities. Such use is illustrated in the collect-call example of this thesis, but a detailed consideration of

¹For probability p , the likelihood ratio ℓ is $\ell = \frac{p}{1-p}$. Similarly $p = \frac{\ell}{1+\ell}$.

posterior probabilities is beyond the scope of this thesis.

7.2.1 Estimating $p(\text{Impostor})$

Impostors are generated from random lists of incorrect words. As more words are used in each list, the chance of getting a better impostor is improved. With a list of one word, that word becomes the impostor. With the addition of a second word, there is a 50% chance the second word will have a better score than the first word did, and will thus become the new impostor. When the n th word is added to the list, there is a 1-in- n chance that the new word will become the impostor. Thus perplexity is logically related to the goodness of fit for the impostor.

Figure 7.1 shows log-frequency histograms at various perplexities. Notice that the histograms are roughly parabolic, indicating normalcy in the underlying distribution. Notice also the even spacing of the parabolas, suggesting that the impostor curve is a simple function of the logarithm of the perplexity. Figure 7.2 shows the same histograms on a linear-frequency scale. Notice the spacing and goodness of fit. The distributions of scores for impostors are found to be roughly normally distributed. By taking the logarithm of the histogram, a normal distribution becomes a parabola open downward and can be fitted using ordinary statistical methods.

It can be seen that the top of the distribution for “impostor 2” is at a raw score of about -4. The top of “impostor 10” is at a raw score of about -3. The top of “impostor 50” is at a raw score of about -2. From this it is conjectured that each multiplying of the perplexity for the impostor moves the peak of the distribution by one point. It must be stressed that this is only a first approximation. Using this rule, “impostor 250” would be expected to peak at -1, “impostor 1250” would be expected to peak at 0, and “impostor 2795” ($1250 \times 5^{0.5}$) would be expected to peak at 0.5. Recall that these curves and datapoints are specific to the Oct96 recognizer, and would be different for other recognizers.

From this information $\log(p(\text{score}|\text{impostor}, \text{perplexity}))$ can be estimated for any score and perplexity.

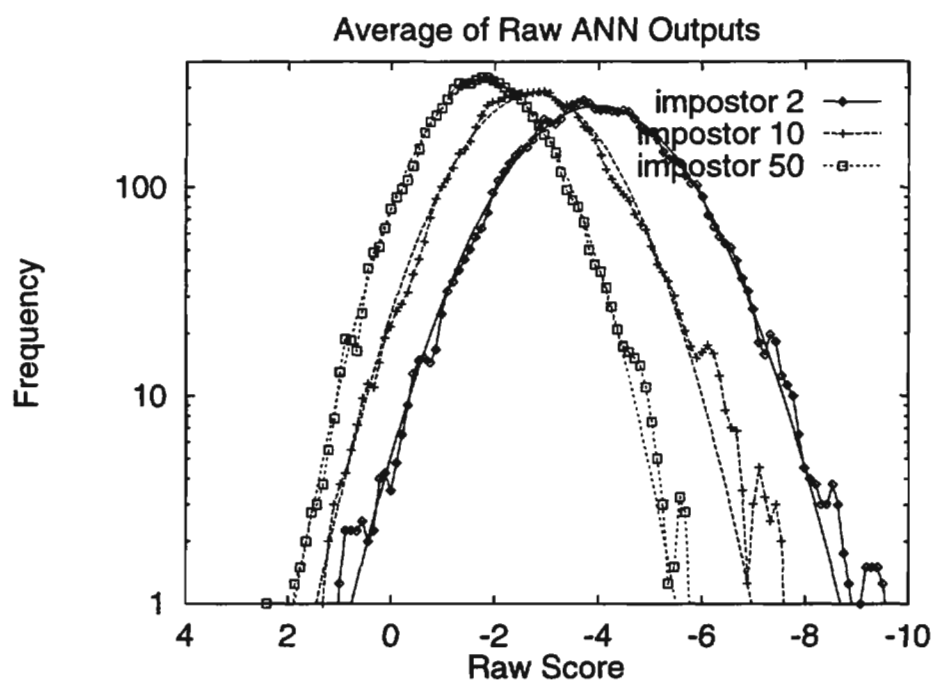


Figure 7.1: Log-Scale Histograms at Various Perplexities. Notice that the histograms are roughly parabolic, indicating normalcy in the underlying distribution. Notice also the even spacing of the parabolas. Details: Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, development test set, probability normalized by rank 6, frame/segment/phoneme/word averaging, 8000 trials, word models depending on corpus.

7.2.2 Estimating $p(\text{True})$

Perplexity does not play a role in true scores. Figure 7.3 confirms this. It shows the histograms for six different perplexities (2, 3, 5, 10, 20, 50) for the same dataset. Notice that the six histograms are nearly identical.

The histograms for true scores appear to peak at a raw score of about 0.5.

The histograms of true values are not as normal as the histograms for the impostors. It is not clear what the underlying distribution might be. The Weibull distribution has been suggested but not evaluated. However the variation from normal is only pronounced for positive raw scores, at which point the impostor scores become insignificant. Therefore although fitting by a normal distribution does affect the final likelihood ratio, all likelihoods

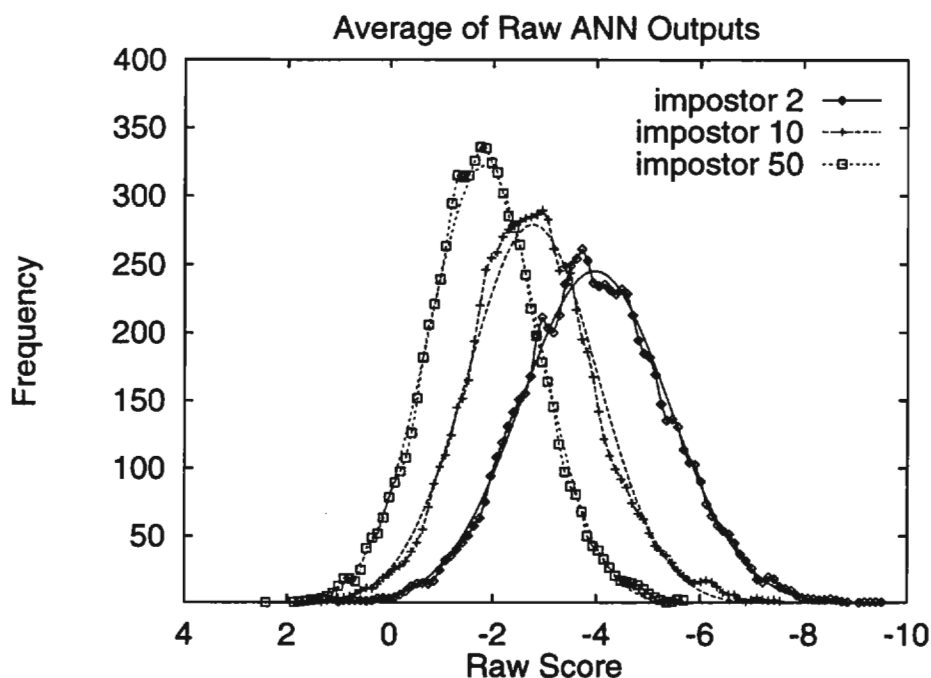


Figure 7.2: Histograms at Various Perplexities. Notice the spacing and goodness of fit. Details: Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, development test set, probability normalized by rank 6, frame/segment/phoneme/word averaging, 8000 trials, word models depending on corpus.

are quite large, so it does not substantially affect the final probability since large likelihoods all yield probabilities near 1.0. (Doubling the likelihood hardly affects the probability.) The final goal is a probability rather than a likelihood.

If the conjecture in the previous section about impostors holds, then when the perplexity is about 2795, the score of the impostor is likely to be as good as the score of the true word, and it will become impossible to discriminate them based on this present technology. It will then be necessary to use other sources of knowledge, such as language models, or a better recognizer. However as the accuracy of the recognizer improves, this limit of 2795 should also increase because the true scores should become better while the impostor scores should become worse, thus increasing the separation between the $p(\text{true})$ and $p(\text{false})$ alternatives.

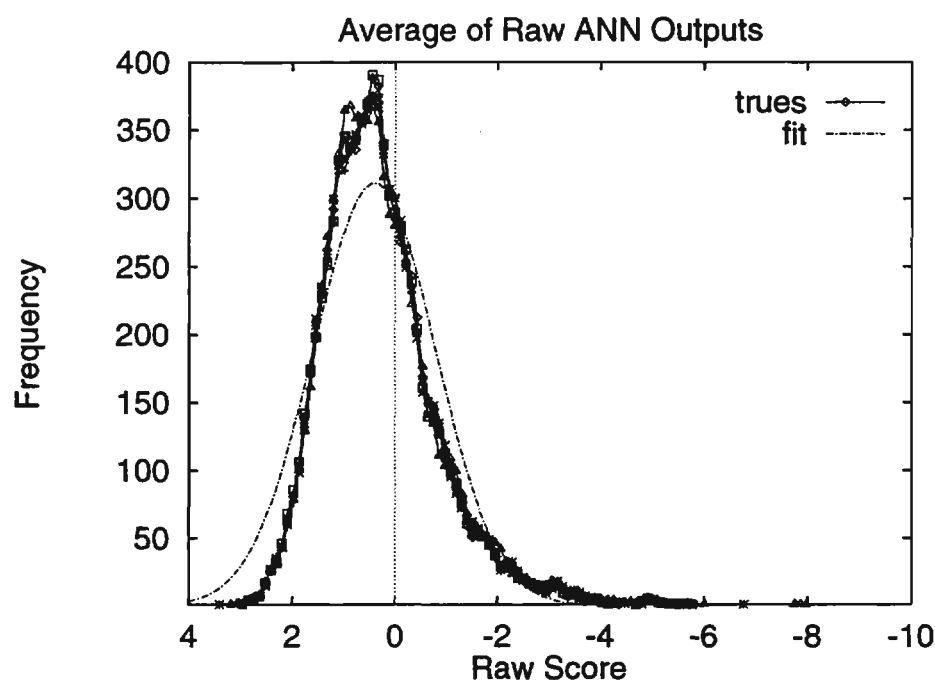


Figure 7.3: True Histograms at Various Perplexities. Notice that the histograms are nearly identical, and that they are only slightly skewed away from the fitted normal curve. Details: Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, development test set, probability normalized by rank 6, frame/segment/phoneme/word averaging, 8000 trials, word models depending on corpus.

7.2.3 Estimating the Likelihood Ratio

Given $p(\text{score}|\text{impostor}, \text{perplexity})$ and $p(\text{score}|\text{true})$ the likelihood ratio is immediate. Figure 7.4 shows probabilities derived from likelihood ratios for several perplexities, based on the fitted curves. Outside the displayed range of 4 to -10 the components of the likelihood ratio are so small as to produce surprising effects, such as $p(\text{true})$ overtaking $p(\text{impostor})$ for raw scores around -30. Such raw scores would be rare indeed in practice.

7.3 Application to a Real-World Problem

As in the case of the “collect call” system mentioned on page 1 it would also be useful to know what decision should be made. The likelihood ratio can be converted to a probability

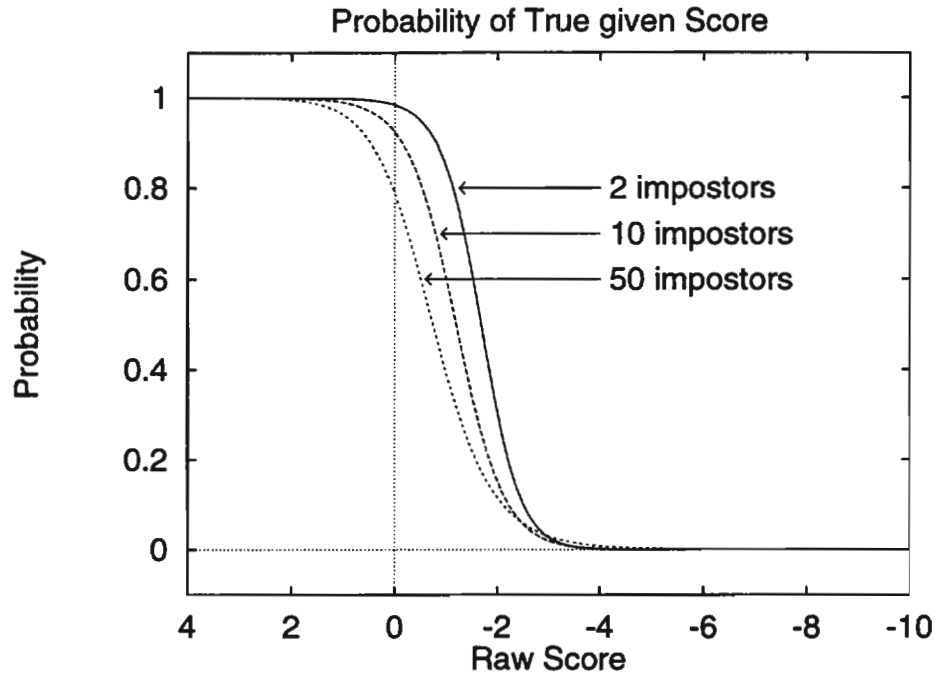


Figure 7.4: Probabilities from Likelihood Ratios. Details: Oct 1996 MFCC-based recognizer, equal mix of OGI Names corpus and NYNEX PhoneBook corpus, development test set, probability normalized by rank 6, frame/segment/phoneme/word averaging, 8000 trials, word models depending on corpus.

so the system can report that there is, for instance, 95% certainty that the answer is “yes.”

When the likelihood ratio is combined with the prior probabilities of true and impostor, the result can be used to derive the final confidence or probability of truth. For example, if the likelihood ratio is 35 to 1 and the overall probability of a true recognition is 0.8, then the updated likelihood becomes $35(\frac{0.8}{1-0.8}) = 140$. The final probability of truth is $\frac{140}{1+140} = .9929$.

The cost of a decision can be computed in a straightforward manner also. There are four parameters: the cost of accepting the truth (at), the cost of rejecting the truth (Type I error) (rt), the cost of accepting a falsehood (Type II error) (af), and the cost of rejecting a falsehood (rf). When the probability of truth is t , the expected value of accepting (a) or rejecting (r) the decision are:

$$a = at * t + af * (1 - t) \quad (7.1)$$

$$r = rt * t + rf * (1 - t) \quad (7.2)$$

Thus it is shown that an accurate measure of confidence expressed as a probability or as a likelihood ratio provides a uniform approach to decision making and rejection under a variety of possible conditions.

Chapter 8

Conclusions

Several forms of utterance verification were presented. The majority of the research is concerned with vocabulary-independent confidence and rejection. Vocabulary independence means that the words in the vocabulary can be supplied after the algorithms are developed; the algorithms do not depend on any particular choice of vocabulary words.

8.1 General Conclusions

It was shown (section 5.3) that frame scores which are probabilities can be averaged to advantage if they are first converted to the logarithmic domain. This same result should apply to likelihoods as well. Averaging in the linear probability domain was shown to work less well. Averaging in the log domain is believed to work better because the underlying scores resemble probabilities, and the combination of them is best done by multiplication in the linear domain or by addition (or averaging) in the log domain.

It was shown (section 5.4) that hierarchical averaging works. Frame scores can be averaged across segments (frames with the same ANN output identity) to make segment scores, and those can be averaged across phonemes and then words to make word scores. Figure 5.6 illustrates the improved separation of true scores from impostors using this scheme. This is expected because of an intuition that human perception occurs in a hierarchical manner at these same levels, and that normalization for time may need to occur on a routine basis.

It was shown (section 5.5) that normalizing the ANN outputs by an average of the top several scores in each frame gives an improved separation of true scores from impostors,

as compared to not doing this normalization. This resulted in a “best score” among all algorithms tested. Normalizing using lower-ranked ANN outputs was shown to worsen performance. The intuition here is that higher-ranked phonemes make up the actual alternate hypothesis for a recognition. Lower-ranked phonemes are irrelevant. A favorable comparison between the aligned (putative) phoneme and these top-ranked alternatives is the thing that is of key importance.

It was shown (section 6) that throwing away ANN scores and using just the corresponding ranks also results in a “best score” among all algorithms tested. The intuition here is that rankings should be more robust to noise and other disturbances. When all scores go up or down with changes in acoustic quality of the utterance, the rankings should remain stable. This is perhaps just a different way of looking at the comparison between the aligned phoneme and top-ranked alternatives, but in this case it is the number of top-ranked alternatives that is important, rather than the ratio of their scores.

Weighted averaging schemes (triangular, trapezoidal, and parabolic) were examined in section 6.3 and found to give no additional discriminative benefit. It was expected that some form of weighted averaging would yield better performance. The failure in finding such an improvement may simply indicate that within a segment (phoneme state) the ranks vary in a random manner that is not biased by whether or not the alignment is true, at least with respect to impostor recognitions.

It was shown (section 7.2.1) that perplexity of the impostor set plays an important role in computing the impostor probability used in the likelihood ratio. This is because the impostors are generated from random lists of incorrect words. As more words are used in each list, the chance of getting a better impostor is improved. This also suggests that a given recognizer has some maximum perplexity limit beyond which out-of-vocabulary utterances cannot be discriminated on the basis of acoustics alone.

8.2 Noteworthy Points

Bootstrap parameter estimation techniques (section 4.11.8) were utilized to assess the strength of performance differences. The earlier use of these techniques would have prevented some incorrect conclusions that wrongly guided some decisions and conclusions in early phases of the research. Had these mistakes been corrected earlier, the research would have been completed sooner.

It was shown that likelihood ratios (odds) and probabilities can be estimated from raw scores (section 7.2.3) and that these can be used to solve typical business problems in a principled way.

8.3 Future Work

Combining Best Methods: The best two methods were the new rank-based approach and the procedure of normalizing scores by the average score of the top several phonemes in a frame. The intuition for combining the approaches is that I suspect they use different information and in combination the performance would improve compared to either one alone. The combination would first normalize ANN outputs according to their performance on a training set. This is similar to the rank-based approach in that each of the 544 segment types has its own performance accuracy that varies by rank or by absolute score. The differences in performance are probably due to inadequate training of the ANN (insufficient examples or poor clustering of examples). After this normalization a second normalization would occur based on the average of the top-scoring phonemes in the frame.

Utterance Length: It would be interesting to take into account the length of an utterance in computing the probability of an impostor utterance. If an utterance is long enough there is a high probability of finding a strong impostor by wordspotting.

Vocabulary Confusability: It would be interesting to take into account the confusability of the vocabulary when computing the probability of an impostor.

Hierarchical Accumulation: It would be interesting to look further into the segmental accumulation of frame scores. Why is it that frame/segment/phoneme/word averaging seems to perform better than frame/segment/phoneme/syllable/word averaging? The intuition is that frame/segment/phoneme/syllable/word averaging should give the better performance. Is the failure due to poor modeling of syllable boundaries? Is human perception seated in some other construct like the syllable but not equal to it?

Un-Pipelining: Pipelined recognition using a lookahead of no more than about 150 msec was used throughout. Pipelining sacrifices some accuracy in exchange for faster recognition. It would be interesting to trade back some lookahead for additional accuracy if the recognition is of low confidence. In particular, the entire utterance (or relevant portion) could be used to initialize the filters, and then could be reused in recognition. This can be justified on the basis that humans may use short-term memory to re-parse an utterance that was not initially understood.

Phonological Rules: It would be interesting to use phonological rules (as in Oshika, Zue, Weeks, Neu, and Aurbach 1975) to modify the standardized pronunciation from a Text-to-Speech system so that it more properly represents the variety in pronunciations to be expected. Using improved word models one might hypothesize a more accurate match between the correct word model and the utterance waveform. On the other hand, it may be true that the increased perplexity due to allowing phonological variation will also allow incorrect word models to match better. It is to be hoped that the net effect would improve recognition and confidence measurement.

Word Boundaries: Word modeling was done with the assumption that the recognized word was surrounded by silence. This is certainly not true for most embedded utterances. There are several ways that this might be improved. One would be to discount the scores coming from the first and last segment of the word model, since these would be the segments affected by bad assumptions about context. Another more difficult way would be to extend the context by identifying part or all of the surrounding utterance.

Bibliography

Austin, S., J. I. Makhoul, R. M. Schwartz, and G. Zavaliagos (1991). Continuous Speech Recognition Using Segmental Neural Nets. In *DARPA Speech and Natural Language Workshop*, Pacific Grove, CA, pp. 249–252. Defense Advanced Research Projects Agency Information Science and Technology Office.

Austin, S., G. Zavaliagos, J. I. Makhoul, and R. M. Schwartz (1992). Speech recognition using segmental neural nets. In *Proceedings of the Seventeenth International Conference on Acoustics, Speech, and Signal Processing (ICASSP-92)*, San Francisco, Volume 1, pp. 625–628. Institute of Electrical and Electronic Engineers.

Barnard, E., R. A. Cole, M. Fanty, and P. Vermeulen (1995). Real-world speech recognition with neural networks. In *Applications and Science of artificial neural networks*, Volume 2492, pp. 524–537. SPIE.

Boite, J.-M., H. Bourlard, B. D'hoore, and M. Haesen (1993). New Approach towards Keyword Spotting. In *Proceedings of the Third European Conference on Speech Communication and Technology (EUROSPEECH'93)*, Berlin, Volume 2, pp. 1273–1276. European Speech Communication Association.

Bourlard, H., B. D'hoore, and J.-M. Boite (1994). Optimizing Recognition and Rejection Performance in Wordspotting Systems. In *Proceedings of the Nineteenth International Conference on Acoustics, Speech, and Signal Processing (ICASSP-94)*, Adelaide, Australia, Volume 1, pp. 373–376. Institute of Electrical and Electronic Engineers.

Bourlard, H. and C. J. Wellekens (1989). Links between Markov Models and Multilayer Perceptrons. In *Advances in Neural Information Processing Systems 1*, Denver, pp. 502–510.

Chase, L. L., R. Rosenfeld, and W. H. Ward (1994). Error-Responsive Modifications to Speech Recognizers: Negative N-grams. In *Proceedings of the Third International Conference on Spoken Language Processing (ICSLP 94)*, Yokohama, Japan, Volume 2, pp. 827–830. Acoustical Society of Japan.

- Cole, R. A., M. Fanty, M. Noel, and T. Lander (1994). Telephone speech corpus development at CSLU. In *Proceedings of the Third International Conference on Spoken Language Processing (ICSLP 94)*, Yokohama, Japan, Volume 4, pp. 1815–1818. Acoustical Society of Japan.
- Cole, R. A., M. Noel, D. C. Burnett, M. Fanty, T. Lander, B. Oshika, and S. Sutton (1994). Corpus development activities at the Center for Spoken Language Understanding. In *ARPA Human Language Technology Workshop*, pp. 34–39. Advanced Research Projects Agency.
- Cole, R. A., D. G. Novick, M. Fanty, P. Vermeulen, S. Sutton, D. C. Burnett, and J. Schalkwyk (1994). A prototype voice-response questionnaire for the U.S. census. In *Proceedings of the Third International Conference on Spoken Language Processing (ICSLP 94)*, Yokohama, Japan, Volume 1, pp. 683–686. Acoustical Society of Japan.
- Colton, D., M. Fanty, and R. A. Cole (1995). Utterance Verification Improves Closed-Set Recognition and Out-of-Vocabulary Rejection. In *Proceedings of the Fourth European Conference on Speech Communication and Technology (EUROSPEECH'95)*, Madrid, Volume 2, pp. 1067–1070. European Speech Communication Association.
- Cox, S. J. and R. C. Rose (1996). Confidence Measures for the SWITCHBOARD database. In *Proceedings of the 21st International Conference on Acoustics, Speech, and Signal Processing (ICASSP-96)*, Atlanta, Volume 1, pp. 511–514. Institute of Electrical and Electronic Engineers.
- Deller, John R., J., J. G. Proakis, and J. H. L. Hansen (1993). *Discrete-Time Processing of Speech Signals*. Macmillan.
- Duda, R. O., P. E. Hart, and N. J. Nilsson (1976). Subjective Bayesian methods for rule-based inference systems. In *Proceedings National Computer Conference*, Volume 15. AFIPS.
- Efron, B. and R. J. Tibshirani (1993). *An Introduction to the Bootstrap*. New York, London: Chapman and Hall.
- Fanty, M., R. A. Cole, and K. Roginski (1992). English Alphabet Recognition with Telephone Speech. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 4*, Denver. Morgan Kaufmann.
- Fetter, P., F. Dandurand, and P. Regel-Brietzmann (1996). Word Graph Rescoring Using Confidence Measures. In *Proceedings of the Fourth International Conference on*

Spoken Language Processing (ICSLP 96), Philadelphia, Volume 1, pp. 10–13. Alfred I duPont Institute.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. New York: Academic Press.

Gillick, L., Y. Ito, and J. Young (1997). A Probabilistic Approach to Confidence Estimation and Evaluation. In *Proceedings of the 22nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP-97)*, Munich, Volume 2, pp. 879–882. Institute of Electrical and Electronic Engineers.

Hampshire, II, J. B. and B. A. Pearlmutter (1990). Equivalence Proofs for Multi-Layer Perceptron Classifiers and the Bayesian Discriminant Function. In D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton (Eds.), *Proceedings of the 1990 Connectionist Models Summer School*. Morgan Kaufmann.

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

Hetherington, I. L. (1995). *A Characterization of the Problem of New, Out-of-Vocabulary Words in Continuous-Speech Recognition and Understanding*. Massachusetts Institute of Technology.

Hon, H.-W. (1992). *Vocabulary Independent Speech Recognition: The VOCIND System*. Pittsburgh, PA 15213 USA: School of Computer Science, Carnegie Mellon University.

Hon, H.-W. and K.-F. Lee (1990). On vocabulary-independent speech modeling. In *Proceedings of the Fifteenth International Conference on Acoustics, Speech, and Signal Processing (ICASSP-90)*, Albuquerque, pp. 725–728. Institute of Electrical and Electronic Engineers.

Jelinek, F. (1981). Self-Organized Continuous Speech Recognition. In J.-P. Haton (Ed.), *Automatic Speech Analysis and Recognition*, Bonas, France, pp. 231–238. D. Reidel (Kluwer).

Kreidler, C. W. (1989). *The pronunciation of English : a course book in phonology*. New York: Basil Blackwell.

Lleida-Solano, E. and R. C. Rose (1996a). Efficient Decoding and Training Procedures for Utterance Verification in Continuous Speech Recognition. In *Proceedings of the 21st International Conference on Acoustics, Speech, and Signal Processing (ICASSP-96)*, Atlanta, Volume 1, pp. 507–510. Institute of Electrical and Electronic Engineers.

- Lleida-Solano, E. and R. C. Rose (1996b). Likelihood Ratio Decoding and Confidence Measures for Continuous Speech Recognition. In *Proceedings of the Fourth International Conference on Spoken Language Processing (ICSLP 96)*, Philadelphia, Volume 1, pp. 478–481. Alfred I duPont Institute.
- Mathan, L. and L. Miclet (1991). Rejection of extraneous input in speech recognition applications, using multi-layer perceptrons and the trace of HMMs. In *Proceedings of the Sixteenth International Conference on Acoustics, Speech, and Signal Processing (ICASSP-91)*, Toronto, pp. 93–96. Institute of Electrical and Electronic Engineers.
- Oshika, B. T., V. W. Zue, R. V. Weeks, H. Neu, and J. Aurbach (1975). The Role of Phonological Rules in Speech Understanding Research. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23(1), 104–112.
- Pearl, J. (1990). Bayesian Decision Methods. In G. Shafer and J. Pearl (Eds.), *Readings in Uncertain Reasoning*, pp. 345–352. Morgan Kaufmann.
- Pitrelli, J. F., C. Fong, and H. C. Leung (1995). PhoneBook Final Report. In *PhoneBook Corpus (CD-ROM)*, Philadelphia. Linguistic Data Consortium, University of Pennsylvania.
- Pitrelli, J. F., C. Fong, S. H. Wong, J. R. Spitz, and H. C. Leung (1995). PhoneBook: A Phonetically-Rich Isolated-Word Telephone-Speech Database. In *Proceedings of the Twentieth International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, Detroit, Volume 1, pp. 101–104. Institute of Electrical and Electronic Engineers.
- Rabiner, L. R. and B.-H. Juang (1993). *Fundamentals of Speech Recognition*. Prentice Hall.
- Richard, M. D. and R. P. Lippmann (1991). Neural Network Classifiers Estimate Bayesian a posteriori Probabilities. *Neural Computation*, 3, 461–483.
- Rivlin, Z., M. Cohen, V. Abrash, and T. Chung (1996). A Phone-Dependent Confidence Measure for Utterance Rejection. In *Proceedings of the 21st International Conference on Acoustics, Speech, and Signal Processing (ICASSP-96)*, Atlanta, Volume 1, pp. 515–517. Institute of Electrical and Electronic Engineers.
- Schaaf, T. and T. Kemp (1997). Confidence Measures for Spontaneous Speech Recognition. In *Proceedings of the 22nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP-97)*, Munich, Volume 2, pp. 875–878. Institute of Electrical and Electronic Engineers.

Setlur, A. R., R. A. Sukkar, and J. Jacob (1996). Correcting Recognition Errors via Discriminative Utterance Verification. In *Proceedings of the Fourth International Conference on Spoken Language Processing (ICSLP 96)*, Philadelphia, Volume 2, pp. 602–605. Alfred I duPont Institute.

Spence, J. T., J. W. Cotton, B. J. Underwood, and C. P. Duncan (1992). *Elementary Statistics*. Prentice-Hall.

Sukkar, R. A., A. R. Setlur, M. G. Rahim, and C.-H. Lee (1996). Utterance Verification of Keyword Strings Using Word-Based Minimum Verification Error (WB-MVE) Training. In *Proceedings of the 21st International Conference on Acoustics, Speech, and Signal Processing (ICASSP-96)*, Atlanta, Volume 1, pp. 518–521. Institute of Electrical and Electronic Engineers.

Van Trees, H. L. (1967). *Detection, Estimation, and Modulation Theory*. John Wiley.

Weintraub, M., F. Beaufays, Z. Rivlin, Y. Konig, and A. Stolcke (1997). Neural-Network Based Measures of Confidence for Word Recognition. In *Proceedings of the 22nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP-97)*, Munich, Volume 2, pp. 879–882. Institute of Electrical and Electronic Engineers.

Young, S. R. (1994). Detecting Misrecognitions and Out-of-Vocabulary Words. In *Proceedings of the Nineteenth International Conference on Acoustics, Speech, and Signal Processing (ICASSP-94)*, Adelaide, Australia, Volume 2, pp. 21–24. Institute of Electrical and Electronic Engineers.

Young, S. R. and W. H. Ward (1993). Recognition Confidence Measures for Spontaneous Spoken Dialog. In *Proceedings of the Third European Conference on Speech Communication and Technology (EUROSPEECH'93)*, Berlin, Volume 2, pp. 1177–1179. European Speech Communication Association.

Index

- α error, 65
- β error, 65
- p^n , 25, 70
- $p^n/(1 - p^n)$, 25, 71
- p^r , 25, 49

- A-stabil, 35
- active vocabulary, 5, 21, 78
- allophone, 20
- alpha error, 24, 65
- ANN, 11, 12, 19, 49
- any model, 22, 63, 64, 97
- ARPA, 26, 29
- ASR, 19

- Barnard, Etienne, iv
- Bayes rule, 26, 32, 105
- Bellcore, 61
- best, 21
- beta error, 24, 65
- bootstrap, 28, 83
- Bowcutt, Jean, 146
- Brigham Young University, 146

- Cambridge UK, 61
- CDL, 20, 46
- CDR, 21, 46

- Center for Spoken Language Understanding, 58
- cepstrum, 10, 27
- CI, 20, 46
- Clayton, Lois, 146
- closed-set rejection, 22, 74
- CMU dictionary, 61
- CMU, 63
- Cole, Ronald A., iii, iv
- Colton, Benjamin, iv
- Colton, Daniel, iv
- Colton, Isaac, iv
- Colton, Jean, iv, 146
- Colton, Joseph, iv
- Colton, Larissa, iv
- Colton, Lawrence B., iv, 146
- Colton, Lois, iv, 146
- Colton, Stacia, iv
- comp.speech, 26, 61, 63
- confidence, 1, 24, 125, 126
- confirm, 3, 23
- context-free grammar, 64
- corpus, 28
- CSLU, 26, 58
- CSLUrp, 26, 68, 89
- CSLUsh, 26, 89

- DARPA, 26, 29
DEC, 26, 61
DECtalk TTS, 61
deny, 3, 23
Digital Equipment Corporation, 61
DoD, 26, 146
DTFT, 27

EER, 24, 67, 81
equal error rate, 24, 67
EUROSPEECH, 29

Fanty, Mark, iii, iv
FAQ, 27, 61
female, 2
figure of merit, 24, 82
filler model, 22, 63, 64
FOM, 24, 81, 82
forced alignment, 78
formant, 10
forward/backward algorithm, 47
fpw, 25, 100
frame, 4, 19, 76
frame size, 10
frequency domain, 27
fspw, 25, 103
fspw, 25, 102
fsw, 25, 101
fw, 25, 100

 $g(0,10)$, 95
 $g(0,10,20\dots)$, 93
 $g(0,2,4,6\dots)$, 92
 $g(0,2,4,8\dots)$, 93
 $g(0,4,16)$, 94
 $g(4,16)$, 95
garbage model, 22
gender corpus, 2
geometric averaging, 25, 96
grammars, 21, 63
Griffin College, 146

Hermansky, Hynek, iv
hidden nodes, 13
HMM, 19, 45
Hosom, John-Paul, 7

ICASSP, 29
ICSLP, 29
impostor, 23, 78, 127
impostor score, 23
in-vocabulary, 23, 80
IV, 23, 80

JANKA, 35

knot points, 25, 91

likelihood ratio, 25, 71
logit, 96
loglikelihood, 96
LPC, 10

male, 2
May96, 48

- mfcc, 10, 27, 46
- Microsoft, 146
- minimum verification error, 66
- Moby dictionary, 61
- Monte Carlo, 83
- MVE, 24, 66, 81

- Names corpus, 57
- Names, 57
- National Science Foundation, v, 146
- no, 14
- NSF, v, 26, 146
- NYNEX, 26, 58

- Oct96, 46
- odds, 25, 71
- OGI, 26, 58
- OOV, 23, 80
- open-set rejection, 23, 74
- Orator TTS, 61
- Orator, 63
- Oregon Graduate Institute, 58
- out-of-vocabulary, 23, 80

- Pavel, Misha, iv
- Pavel, Misha (Michael), iii
- perplexity, 22, 60, 80, 127, 134
- phone halves, 20, 46
- phone state, 20, 31
- phone thirds, 20, 46
- PhoneBook, 58
- phoneme, 3, 20
- phonetic units, 19, 46
- PLP, 10, 28, 49
- pruning, 22, 78
- Pumping Lemma, 146
- putative, 21, 36
- putative recognition, 11

- regular expression, 64
- rejection, 1, 22
- RMS, 12
- ROC curve, 24, 81
- Rsynth TTS, 61

- segmental neural network, 31
- SNN, 31
- speaker, 7
- spectrogram, 10
- step size, 10
- Systems Application Engineering, 146

- talker, 7
- Texas Instruments, v, 146
- time domain, 27
- toolkit, 26, 89
- total verification error, 24, 65
- trial, 28
- true recognition, 23
- true score, 23
- TTS, 26, 61
- TVE, 24, 65
- Type I error, 24, 65
- Type II error, 24, 65

U S West, v

utterance, 19

VERBMOBIL, 35

verifier, 3, 23

Viterbi algorithm, 16

Viterbi score, 2, 76

Viterbi search, 11, 14, 15, 61

Ward, Wayne H., iii

wavefile, 19, 55

waveform, 8

word model, 21

wordspotting, 21, 63, 64

Worldbet, 22, 62

yes, 13, 14

Biographical Note

Larry Don Colton was born on Chanute Air Force Base, near Urbana-Champaign, Illinois, on January 30, 1954. His parents are Lawrence Boyd Colton and the former Emma Jean Bowcutt.

Starting in 1971, Don attended Brigham Young University, in Provo, Utah, on a tuition scholarship. He interrupted his schooling for two years to serve as a missionary in South Korea. He then returned to BYU, where he married in 1975 the former Lois Ann Clayton, and earned in 1976 a Bachelor of Science degree with High Honors, with a major in Mathematics and a minor in Computer Science. In 1978 he earned a Master of Business Administration degree, also from BYU. Don then worked as a computer programmer at Texas Instruments (Lubbock), a computer consultant (programmer) at Systems Application Engineering (Boston), and as a programmer, marketer, and product support manager at Microsoft (Redmond).

From 1988 to 1993, Don was Assistant Professor at Griffin College (Seattle), a small business college, where he chaired the computer programming department and served on the academic standards committee. In 1991 Don determined to return to school himself, to earn a PhD in Computer Science, specializing in spoken language translation. In 1992 he received Honorable Mentions in the National Science Foundation Graduate Research Fellowship program, and also in the Department of Defense Graduate Research Fellowship program. During 1992 and 1993 Don attended part-time at the University of Washington, and published a short paper on the Pumping Lemma for Context-Free Languages. In 1993 he accepted admission to the Oregon Graduate Institute of Science and Technology, and successfully challenged the written portion of the PhD Qualifying Examination. In 1994 he was awarded three-year Graduate Research Fellowships from both the National Science Foundation (which he accepted) and the Department of Defense (which he declined). In

1997 he completed his PhD in spoken language understanding, with particular emphasis on rejection of out-of-vocabulary recognitions. While a student at OGI he also taught several graduate courses.

Beginning in August 1997 Don returns to teaching and research as Assistant Professor of Computer Science at Brigham Young University Hawaii campus in Laie, Hawaii. Areas of interest include speech recognition, language understanding, machine translation, and most of all teaching.