12-13-2003

Network Training for Continuous Speech Recognition

Issac John Alphonso

NETWORK TRAINING FOR CONTINUOUS SPEECH RECOGNITION

By

Issac J. Alphonso

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2003

NETWORK TRAINING FOR CONTINUOUS SPEECH RECOGNITION

By

Issac J. Alphonso

Approved:

Dr. Joseph Picone Professor of Electrical and Computer Engineering (Director of Thesis)	Dr. Georgios Lazarou Assistant Professor of Electrical and Computer Engineering (Committee Member)
Dr. Eric Hansen Assistant Professor of Computer Science and Engineering (Committee Member)	Nicholas Younan Graduate Coordinator of Computer Engineering in the Department of Electrical and Computer Engineering
A. Wayne Bennett Dean of the College of Engineering	

Name: Issac J. Alphonso

Date of Degree: December 13, 2003

Institution: Mississippi State University

Major Field: Computer Engineering

Major Professor: Dr. Joseph Picone

Title of Study: Network Training for Continuous Speech Recognition

Pages in Study: 67

Candidate for Degree of Master of Science

Spoken language processing is one of the oldest and most natural modes of information exchange between humans beings. For centuries, people have tried to develop machines that can understand and produce speech the way humans do so naturally. The biggest problem in our inability to model speech with computer programs and mathematics results from the fact that language is instinctive, whereas, the vocabulary and dialect used in communication are learned. Human beings are genetically equipped with the ability to learn languages, and culture imprints the vocabulary and dialect on each member of society. This thesis examines the role of pattern classification in the recognition of human speech, i.e., machine learning techniques that are currently being applied to the spoken language processing problem.

The primary objective of this thesis is to create a network training paradigm that allows for direct training of multi-path models and alleviates the need for complicated systems and training recipes. A traditional trainer uses an expectation maximization (EM)

based supervised training framework to estimate the parameters of a spoken language processing system. EM-based parameter estimation for speech recognition is performed using several complicated stages of iterative reestimation. These stages typically are prone to human error. The network training paradigm reduces the complexity of the training process while retaining the robustness of the EM-based supervised training framework. The hypothesis of this thesis is that the network training paradigm can achieve comparable recognition performance to a traditional trainer while alleviating the need for complicated systems and training recipes for spoken language processing systems.

DEDICATION

To my mother and father, with love and gratitude.

ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to my major advisor, Dr. Joseph Picone, for his constant inspiration, expert guidance and constructive criticism throughout this work. I have been fortunate to work as a graduate assistant under his guidance. I did learn a lot from his constant monitoring and his passion for perfection. He has been a tremendous influence in my life.

I am deeply indebted to Jonathan Hamaker for his constant support throughout this work. It has been a wonderful experience working with him. His contribution to the experimental setup for this thesis has been immense. His way of explaining concepts in a simple manner and the innumerable discussions that I had with him will always be cherished.

I also extend my thanks to my other friends in ISIP for making my tenure a memorable one. Last but not the least, I would like to thank all my friends who have added so much fun to my life and helped me keep mind off work when needed.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Speech is one of the oldest and most natural means of information exchange between humans beings. For centuries people have tried to develop machines that can understand and produce speech as humans do so naturally [1,2]. The biggest problem in our inability to model speech with computer programs and mathematics results from the fact that language is instinctive, whereas, the vocabulary and dialect used in communication are learned [1,2,3]. Human beings are genetically equipped with the ability to learn languages, and culture imprints the vocabulary and dialect on each member of society [1,4]. The problems mentioned above are some of the open research areas in speech: encoding prior knowledge in the system to mirror the language instinct in humans, dynamically learning to deal with words that are not present in the system's vocabulary, and adapting to the variability in speaker accents and dialects across different gender and age groups. These problems are further compounded by the fact that humans rarely if ever use proper articulation and grammar during conversational speech [2,3,4,5].

There have been numerous approaches aimed at understanding the underlying process involved in the perception and production of speech. These approaches involve disciplines as diverse as pattern classification and signal processing to physiology and linguistics. The interdisciplinary nature of the problem is one thing that makes speech

recognition such a complex and fascinating problem. This thesis examines the role of pattern classification in the recognition of human speech, i.e., machine learning techniques that are currently being applied to the speech recognition problem.

1.1. Statistical Methods

The predominant approach in speech recognition is a statistically-based approach [6,7] in which we choose the most probable word sequence from all word sequences that could have possibly been generated. For example, given a sequence of words $W = w_1, w_2, \ldots, w_N$, if A is the acoustic evidence that is provided to the system to identify this sequence, then the recognition system must choose a word string $\hat{W}$ that maximizes the probability that the word string was spoken. Specifically,

$$\hat{W} = \underset{W}{\operatorname{argmax}}\ p(W|A) \tag{1}$$

where $p(W|A)$ is known as the *a posteriori* probability since it represents the probability of occurrence of a sequence of words after observing the acoustic signal A.

The fundamental concepts involved in the human communication process — perception and production — are still not clearly understood. The lack of understanding of the fundamental concepts adds a degree of uncertainty to the problem. The inherent uncertainty, coupled with a wide variation in the characteristics of speech, makes it difficult to model using an expert based system. The difficulty in using an expert based system is due to the prohibitive costs associated with representing the uncertainty

and variations in speech [2,3,4]. A statistical approach circumvents the need for manually encoding information in favor of a self-organized approach [6,7,9].

1.2. The Maximum Likelihood Approach

The statistically-based approach, described in the previous section, attempts to infer the posterior probability $p(W|A)$ of a word sequence $\hat{W}$ given the acoustic signal A. Hence, speech recognition is a simple matter of finding the most probable word sequence $\hat{W}$ that maximizes $p(W|A)$. This is called the Maximum A Posteriori or MAP criterion. However, finding the most probable word sequence is not an easy task because for any given language there are an infinite number of such word sequences. We can work around this problem by using a slightly weaker criterion known as Maximum Likelihood Estimation (MLE) and applying Bayes rule to find $\hat{W}$ such that [8,10,11],

$$\hat{W} = \underset{W}{\operatorname{argmax}}\ p(A|W)p(W)\ . \qquad (2)$$

The posterior term $p(A|W)$ is the probability that the acoustic signal A was observed if a word sequence W was spoken. The posterior probability is typically determined by an acoustic model in a speech recognition system. The *a priori* term $p(W)$, is the probability associated with the word sequence W. The *a priori* probability is typically given by a language model in a speech recognition system. Hence, a speech

recognition system combines the acoustic and language model probabilities in order to determine the most probable word sequence $\hat{W}$.

A speech recognition system maps the acoustic signal A to a hidden state sequence $Q = q_1, q_2, \ldots, q_T$, i.e., the state sequence is not directly observable but can be observed through another stochastic process [6,7]. Equation 2 can be reformulated as,

$$\hat{W} = \underset{W}{\operatorname{argmax}} \sum_{Q} p(A|W, Q)p(W, Q)$$

$$\hat{W} = \underset{W}{\operatorname{argmax}} \sum_{Q} p(A|Q)p(W, Q) \tag{3}$$

where the last expression makes the assumption that the acoustic signal A is conditionally independent of the word sequence $\hat{W}$ given the hidden state sequence Q. Equation 3 can be approximated as,

$$\hat{W} \approx \underset{W}{\operatorname{argmax}} \max_{Q} p(A|Q)p(W, Q)$$

$$\hat{W} \approx \underset{W}{\operatorname{argmax}} \max_{Q} p(A|Q)p(W)p(Q|W)$$

$$\hat{W} \approx \underset{W}{\operatorname{argmax}} \max_{Q} p(A|Q)p(W)p(Q) \tag{4}$$

where the last expression again makes the assumption that the hidden state sequence Q is conditionally independent of the word sequence $\hat{W}$. The hidden state sequence $p(Q)$ is modelled as a discrete, first-order Markov chain and is given by,

$$p(Q) = p(q_1)\prod_{t=2}^{T} p(q_t|q_{t-1}).$$

The terms $p(A|Q)$ and $p(W)$ in Equation 4 correspond to the acoustic and language modeling part of speech recognition respectively.

1.3. Acoustic Front End

The acoustic front end of a speech recognition system converts the digital representation of an acoustic signal A to a sequence of feature vectors. The main goal of the acoustic front end is to generate a sequence of feature vectors that represents the temporal, spectral and perceptual characteristics of the signal [12,13]. An in-depth review of the role of an acoustic front end in a speech recognition system can be found in [12].

Most front ends today model signal characteristics using mel frequency-scaled cepstrum coefficients [14]. The cepstrum coefficients are derived by transforming the spectrum of the signal into the quefrency domain [14,15]. This is done by taking the discrete cosine transform of the log magnitude of the spectrum. The log magnitude of the spectrum is preprocessed by passing it through a sequence of filter banks before the application of the discrete cosine transform [14,16]. The discrete cosine transform is an orthogonal transformation that decorrelates the spectrum. The orthogonal transformation

is attractive because it validates, to some extent, the assumption made in acoustic modeling that the features are conditionally independent of each other.

The feature generation process typically divides the acoustic signal into 10 msec intervals, commonly referred to as a frame, using an overlapping window approach in which each window accounts for 25 msec of the signal [12,13]. The most popular acoustic front end in use today employs 39 parameters per frame of speech data, and consists of the signal log-energy and 12 mel-spaced cepstral coefficients, plus their first and second-order temporal derivatives. The frame and window lengths are used to track the dynamics of the articulators and control the amount of new data seen per frame.

1.4. Acoustic Modeling

The function of the acoustic model is to compute a posterior probability $p(A|W)$ of the acoustic signal A given the word sequence $\hat{W}$. The goal is to find the best word sequence $\hat{W}$ that matches the input signal A via pattern classification. However, applying pattern classification to the problem is not as simple as one would think, since there are an infinite number of word sequences for any given language. Hence, the word sequences are decomposed into phonemes, the base units of any language. There are typically forty two phonemes used in American English [17]. The number of phonemes, compared to the number of words, makes it possible to construct a reasonable sized corpora that contains enough examples of each phoneme to train the speech recognizer.

The best word sequence $\hat{W}$ that matches the input signal A is determined in two steps. The first step is to map the phonemes, corresponding to each word in $\hat{W}$, to a sequence of HMM states $Q = q_1, q_2, \ldots, q_T$. The mapping gives us the posterior probability $p(A|Q)$ (recall that A is independent of $\hat{W}$ given Q). The second step is to evaluate the posterior probability $p(A|Q)$ as shown below

$$p(A|Q) = \prod_{t=1}^{T} p(X_t|q_t) \tag{5}$$

where the term X_t is the feature vector at time t generated by the acoustic front end corresponding to the acoustic signal A. Note that the last expression makes the assumption that the feature vectors are conditionally independent of each other. The likelihood of the data at a given state $p(X_t|q_t)$ is typically evaluated using a Gaussian mixture model (GMM) within the HMM framework.

1.5. Language Modeling

The goal of a language model is to determine the *a priori* probability associated with the word sequence $\hat{W}$. More specifically, a language model provides an estimate of the probability of any word w_i in the word sequence W, in the context of the words around it [6,18,19,20]. The probability of an arbitrary word w_i in the word sequence W depends on the previously spoken words and is given by,

$$p(W) = \prod_{i=1}^{n} p(w_i | w_1, w_2, \ldots, w_{i-1}) . \quad (6)$$

Naturally, it is unreasonable to assume that the word w_i depends on all previously spoken words in the sequence [6]. Hence, an effective way of modeling a sequence of N words is to use an n^{th} order Markov chain,

$$p(W) = p(W_1^M) = \prod_{i=1}^{M} p(w_i | w_{i-1}, w_{i-2}, \ldots, w_{i-n}) . \quad (7)$$

The probability of a word based on the previous N words is referred to as an N-gram probability. N-gram language models are generated by computing the frequency of all N-word classes in large text corpora [18]. The disadvantage of primarily using the approach just described is that it does not account for unseen word classes. N-gram smoothing [6,17] and back-off language modeling [17] techniques typically account for the occurrence of unseen words. These techniques approximate the probability of higher-order language models using lower-order language models when an unseen word sequence is encountered. An in-depth review of N-gram language models in speech recognition can be found at [6,17].

1.6. Pronunciation Modeling

It is well known that conversational speech tends to have more variability than read speech and words spoken in isolation. This variability can take the form of human and non-human noises, unseen words and alternative pronunciations [21]. When people read a paragraph their pronunciations of the words are more likely to conform to their dictionary

citations. However, in conversational speech people generally use pronunciations that deviate from their dictionary citations. These deviations are due to speaker accent variations and other characteristics of causal speech such as coarticulation and reduction of words [22]. The inability of speech systems to model such alternative pronunciations is part of the reason for the low recognition performance on conversational speech [22,23].

Speech recognition systems in the past have primarily used pronunciations given by the dictionary to model words. The lack of alternative pronunciations results in a set of models that inaccurately represent the underlying observations. The alternative would be to model the words using multiple pronunciations that are commonly used and not otherwise found in the dictionary. However, it has been shown through experiments that any benefit gained by using alternative pronunciations is negated by the additional complexity that is added to the system [23]. The main reason for this is that there is often insufficient data available to train a system, especially when we use context dependent models such as word-internal or cross-word triphones.

There have been several attempts to address the problem of using alternative pronunciations. Initial attempts used phonological rules to model phone substitution and deletion using hand crafted rules by linguists [21]. These rules describe the actions that take place when two specific phones at word boundaries are encountered. Pronunciation dictionaries, which followed, add probabilities to the alternative realizations of each word in the dictionary [22]. The probabilities for the alternative realizations are generated directly by hand or inferred from the data. Decision trees attempted to generate a pronunciation graph with associated probabilities using a bayesian approach [23]. The

decision trees for each phone in the sequence are searched, using linguistically motivated questions, until we reach a leaf node; which, gives us a distribution over the pronunciations. These attempts try to use the pronunciations that reflect the acoustic models and at the same time attempt to generalize to unseen words.

1.7. Thesis Contributions and Organization

The primary objective of this thesis is to create a network training paradigm that allows for direct training of multi-path models and alleviates the need for complicated systems and training recipes. A traditional trainer uses an expectation maximization (EM) based supervised training framework to estimate the parameters of a speech recognition system. EM-based parameter estimation for speech recognition is performed using several complicated stages of iterative reestimation. These stages are prone to human error. The network training paradigm reduces the complexity of the training process while retaining the robustness of the EM-based supervised training framework. The hypothesis of this thesis is that the network training paradigm can achieve comparable recognition performance to a traditional trainer while alleviating the need for complicated systems and training recipes.

This thesis is organized as follows. Chapter 2 describes the theory behind HMM's and the supervised learning process used to estimate the parameters of a speech recognition system. Chapter 3 provides an in-depth look at the network training framework. A comparative analysis of the network training recipe to that of a traditional trainer is also presented. Chapter 4 describes the various experiments that were performed

and how these experiments fit into the framework of this thesis. Preliminary results on various speech corpora are also presented. Chapter 5 summarizes the findings of this thesis and discusses some promising avenues for future work.

CHAPTER II

THEORETICAL FOUNDATIONS

In the previous chapter, we decomposed the speech recognition problem into one of designing a classifier based on the prior probability $p(W)$ and the class-conditional density $p(A|W)$. If we are able to estimate the models using statistical techniques, we could design an optimal classifier to recognize speech. However, in speech recognition applications we rarely if ever have such knowledge regarding the probabilistic structure of the problem. Hence, when faced with such uncertainty, our goal is to reason the best we can using whatever incomplete knowledge we have of the problem.

In speech, the function $p(W)$ is computed by averaging over large text corpora and the class-conditional density $p(A|W)$ is estimated using a hidden Markov model. A hidden Markov model is ideal for problems such as speech recognition due to its ability to simultaneously model temporal and spectral behavior. The speech signal is produced by a

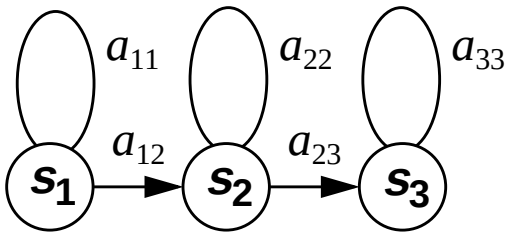

Figure 1. An example of a three-state Markov chain used in speech recognition.

physiological system that is constrained by the normal laws of physics [12,13], and hence states at time t are influenced by states at time $t-1$.

2.1. Markov Processes

A random process $X(t)$ is called a first-order Markov process [24] if the future observations of the process given the present is independent of the past,

$$P[X(t_{k+1}) = x_{k+1} | X(t_k) = x_k, \ldots, X(t_1) = x_1] = P[X(t_{k+1}) = x_{k+1} | X(t_k) = x_k] . \quad (8)$$

If the Markov process $X(t)$ is discrete-valued, then the probability of an observation is given by a probability mass function,

$$P[a < X(t_{k+1}) \leq b = x_{k+1} | X(t_k) = x_k, \ldots, X(t_1) = x_1] = P[a < X(t_{k+1}) \leq b = x_{k+1} | X(t_k) = x_k] . \quad (9)$$

If the Markov process is continuous-valued, then the probability of an observation is given by a probability density function,

$$f_{X(t_{k+1})}[x_{k+1} | X(t_k) = x_k, \ldots, X(t_1) = x_1] = f_{X(t_{k+1})}[x_{k+1} | X(t_k) = x_k] . \quad (10)$$

A discrete-time integer-valued Markov process is called a Markov chain [24]. Markov chains are completely specified by the probability of the initial states,

$$p_j(0) = P[X_0 = j] \quad (11)$$

and their corresponding state transition probabilities,

$$P[X_{n+1} = j | X_n = i] = p_{ij} . \quad (12)$$

An example of a three-state Markov chain can be seen in Figure 1. At any instant of time each state in the Markov chain can transition to either the next state or stay in the same state.

2.2. Hidden Markov Models

A hidden Markov model is a random process that consists of a set of states and their corresponding transition probabilities. Like the Markov chain, a hidden Markov model (HMM) is specified by the initial state probabilities π_j,

$$\pi_j = P[x(0) = j] \tag{13}$$

and the state transition probabilities a_{ij},

$$a_{ij} = P[x(t+1) = j | x(t)] = i \ . \tag{14}$$

Also, since an HMM is a random process each state in the model can be treated as a random variable, with a corresponding probability distribution $b(o)_j$. The probability distribution or state emission probability is the likelihood of state j with respect to the time indexed observation o_t,

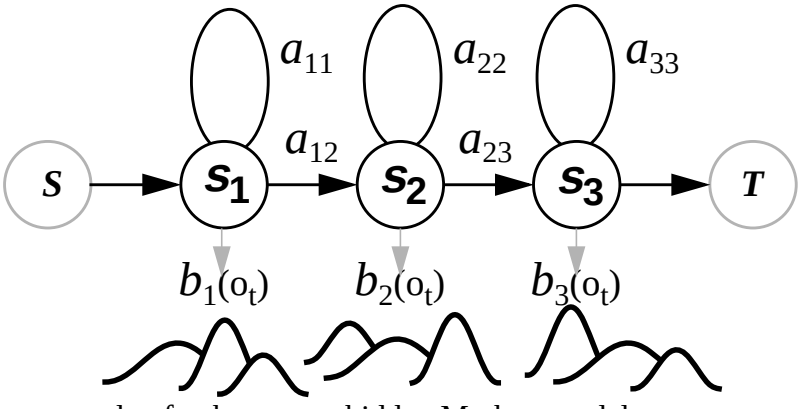

Figure 2. An example of a three-state hidden Markov model.

$$b_j(o_t) = P[o_t | x(t) = j] \ . \tag{15}$$

An HMM is assumed to have homogenous transition probabilities, i.e., the state transition probabilities are assumed to be constant over time [6,17]. Hence, for all states i in the model the following property holds,

$$\sum_{j=2}^{N} a_{ij} = 1.0 \ . \tag{16}$$

The state emission probabilities are typically represented as continuous density distributions [7,25], i.e., the likelihood of each state j has the following property,

$$\int_{o} b_j(o) = 1.0 \ . \tag{17}$$

The parameters of an HMM can be compactly represented as $\lambda = (A, B, \pi)$. Where A is the set of state transition probabilities $\{a_{ij}\}$ for the model and B is the set of symbol observation probabilities $\{b_j\}$ for each state. An example of an HMM is shown in Figure 2. There are some basic assumptions involved in the design of HMM's which are specific to speech processing which include: the model topology, the output probability distribution, the minimum state duration and the nature of dependency within the model.

2.3. HMM Assumptions

The HMM topology typically used in a speech recognition system is a left-to-right topology [7,17] as shown in Figure 2. Most languages can be decomposed into a small set of basic sounds, known as phonemes [6]. Phones are typically what is used to model sounds in an HMM-based speech recognizer. Typically a three-state hidden Markov model

with a dummy start and end node is used to represent individual phones. The dummy start and end nodes are non-emitting states and do not have a probability distribution associated with them.

During model parameter training, speech audio files containing examples of each phone are presented to the HMM. Special care is taken in selecting the training examples so that the audio data is representative of the behavior of the overall population. If this is not the case, then the trained models will not generalize well to unseen data.

The output probability distribution of each state in an HMM is typically modeled by a mixture of multivariate Gaussian probability density functions [25,26]. The motivation for using a Gaussian comes from information theory which tells us that the Gaussian distribution has the highest entropy among all distributions of equal variance [17,27]. If we let w_i represent the weights of the m mixture components of the output distribution, and $\aleph(\underline{\mu}_i, \Sigma_i)$ represent the multivariate Gaussian distribution of dimension d then, for any arbitrary state s, the probability of the feature vector $\underline{x}$ given the state s is given by the following equation,

$$p(\underline{x}|s) = \sum_{i=1}^{m} w_i \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\underline{x}-\underline{\mu}_i)^T \Sigma_i^{-1}(\underline{x}-\underline{\mu}_i)\right] . \quad (18)$$

The duration of time spent in any state of an HMM is determined by an exponential distribution [2,17,25]. This is not very desirable since the probability of visiting a state decreases exponentially as time progresses [17,25]. Alternative implementations of hidden Markov models exist where the state duration probability is

determined by other statistical probability distributions. While these alternate implementations increase the computational overhead of the training process, they do not increase the accuracy of the recognition process by a significant amount [17,26].

HMM's used in speech recognition are typically a first-order approximation of a Markov process. The first-order assumption means that the states at time t are only dependent on states visited at time $t-1$. A first-order approximation is important because it makes computation feasible without any significant loss in recognition accuracy. Higher-order Markov models have been shown to give only marginal improvements in performance while the computational overhead increases by an order of magnitude [7,17].

The class-conditional density $p(A|W)$ is estimated by a supervised learning process. Estimating the class-conditional directly is by no means an easy task. A multivariate Gaussian mixture $\aleph(\underline{\mu}_i, \Sigma_i)$ is typically used to represent the underlying probabilistic structure of the class-conditional density. The parameterization of the class-conditional density allows us to simplify the problem from one of estimating the function $p(A|W)$ to one of estimating the parameters of the mixture [7,17]. The parameter estimation problem in speech is typically addressed using a maximum likelihood estimation procedure [8,28].

2.4. Maximum Likelihood Estimation

The maximum likelihood approach [8] treats the parameters of the model λ as fixed quantities whose values need to be estimated. The parameters are estimated by maximizing the probability of observing the training data, which in our case is the acoustic signal, given the current estimate of the model parameters. The maximum likelihood approach has good convergence properties [7,8,11]. Also, estimation of parameters is more computationally tractable than Bayesian techniques due to the availability of efficient algorithms [7,17].

The goal in the maximum likelihood approach is to maximize the probability $p(A|\lambda)$, i.e., the probability of observing the input signal A given the model λ. If the input signal A is given by the observation sequence $o_1, o_2, \ldots, o_T$, then, the likelihood of the model λ (assuming observations are conditionally independent) can be represented as,

$$p(A|N) = \prod_{i=1}^{T} p(o_i|\lambda) \ . \tag{19}$$

The maximization of the likelihood is normally achieved by maximizing the logarithm of the likelihood [27,28]. Maximizing the log-likelihood is equivalent to maximizing the likelihood since the logarithm is a monotonically increasing function,

$$\ln[p(A|\text{א})] = \sum_{i=1}^{T} \ln[p(o_i|\lambda)] \ . \tag{20}$$

The estimates for λ are obtained by taking the partial derivative with respect to each parameter and setting them to zero. It should be noted that the accuracy of the

estimates computed increases with the number of training samples. Also, the estimates are not guaranteed to represent a global maximum and could, in fact, represent a local maximum [8,28]. In speech processing, the models are perturbed from time to time to help the parameter estimation process avoid getting stuck in a local optimum in the search space.

2.5. Expectation Maximization

The expectation maximization (EM) algorithm represents a general framework that can be used to determine the maximum likelihood estimates of model parameters. EM can also be applied in cases where we have missing features [28,29]. The algorithm iteratively estimates the likelihood of the model parameters given the training data. The EM algorithm uses the estimates to refine the models, following each iteration, until there is no noticeable difference between successive iterations. The algorithm is guaranteed to converge to the maximum-likelihood estimate.

The EM algorithm is based on Jensen's inequality, which can be stated as,

$$\sum_x p(x)\log(p(x)) \geq \sum_x p(x)\log(q(x)). \tag{21}$$

If y is a random variable that represents the observation sequence, and λ represents the parameters of the current model, then the EM algorithm determines the estimates of the model λ' such that the following inequality holds [6],

$$\log p(y|\lambda') - \log p(y|\lambda) \geq 0. \tag{22}$$

The inequality above can be written in the following form, where q is a random variable that depends on y and is generated by the same process that generates y [6],

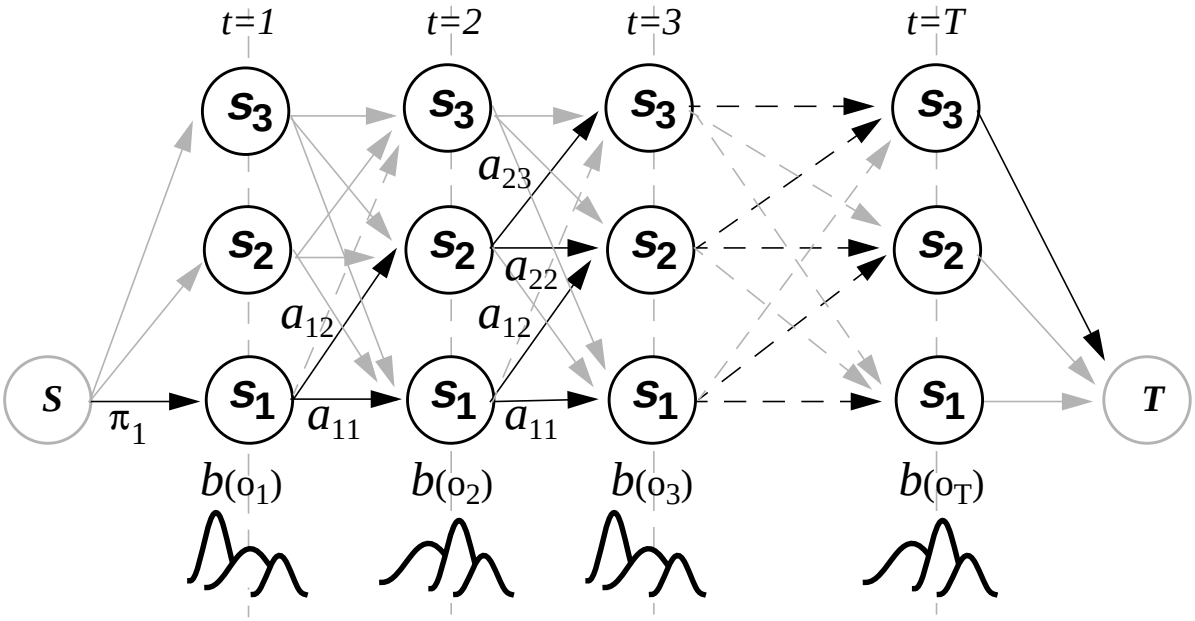

Figure 3. An example of a time evolution of the process that generates the observation sequence for a three state HMM (N=3).

$$\sum_q p(q|y)\log p(y|\lambda') - \sum_q p(q|y)\log p(y|\lambda) \geq 0\,. \tag{23}$$

The random variable q in this case could represent the state sequence or transition probabilities that were used to generate the observation sequence. The right hand side of equation 23 above can be expanded in the following manner [6],

$$\sum_q p(q|y)\log\left[p(y|\lambda')\frac{p(q,y|\lambda')}{p(q,y|\lambda')}\right] - \sum_q p(q|y)\log\left[p(y|\lambda)\frac{p(q,y|\lambda)}{p(q,y|\lambda)}\right] \geq 0 \tag{24}$$

$$\sum_q p(q|y)\log\left[p(y|\lambda')\frac{p(q,y|\lambda')}{p(q|y,\lambda')\cdot p(y|\lambda')}\right] - \sum_q p(q|y)\log\left[p(y|\lambda)\frac{p(q,y|\lambda)}{p(q|(y,\lambda))\cdot p(y|\lambda)}\right] \geq 0 \tag{25}$$

$$\sum_q p(q|y)\log[p(q,y|\lambda')] - \sum_q p(q|y)\log[p(q,y|\lambda)] + \sum_q p(q|y)\log[p(q|y,\lambda)] - \sum_q p(q|y)\log[p(q|y,\lambda')] \geq 0 \quad . \quad (26)$$

Hence, if we drop the last two terms in the equation above and if the difference of the first two terms is positive, then from Jensen's inequality we have the following result [6,27],

$$\sum_q p(q|y)\log[p(y|\lambda')] \geq \sum_q p(q|y)\log[p(y|\lambda)] . \quad (27)$$

The inequality above proves that the EM algorithm finds a maximum likelihood estimate for the model λ' that is either better than or similar to the original model λ. The Baum-Welch algorithm is a computationally efficient implementation of the EM algorithm specific to HMM parameter reestimation in speech recognition [7,8,17].

2.6. The Forward Procedure

The motivation for the forward procedure comes from the need to have a computationally efficient way of computing the function $p(\bar{o}|\lambda)$, i.e., the probability of the observation sequence $\bar{o} = o_1 o_2 \ldots o_T$ given the model λ. The probability $p(\bar{o}|\lambda)$ is computed by summing over all possible state sequence $\bar{q} = q_1 q_2 \ldots q_T$ and is given by the following equation [7,17],

$$P(\bar{o}|\lambda) = \sum_{\bar{q}} P(\bar{o}|\bar{q},\lambda) P(\bar{q}|\lambda) \quad . \quad (28)$$

The forward procedure uses a trellis to compute $p(\bar{o}|\lambda)$ in an efficient manner. An example of a trellis can seen in Figure 3. The forward probability $\alpha_t(i)$ is defined as the

probability of being in state i at time t given that you have observed the partial observation sequence $\bar{o} = o_1 o_2 \ldots o_t$. The forward probability can be arrived at inductively by the following equation [7,17],

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1}) \ . \tag{29}$$

Hence, the probability of the observation sequence $\bar{o} = o_1 o_2 \ldots o_T$ is given by,

$$P(\bar{o}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \ . \tag{30}$$

The forward procedure has a computational complexity of $O(N^2)$ as compared to a complexity of $O(N^T)$ for a direct computation [7,17] by enumerating all state sequences. Hence, the forward procedure saves many orders of computations as compared to the direct approach.

2.7. The Backward Procedure

The backward procedure is analogous to the forward procedure in that it is defined as the probability of being in state i at time t given that you will observe the partial observation sequence $\bar{o} = o_{t+1} o_{t+2} \ldots o_T$. The backward probability $\beta_t(i)$ can be arrived at inductively by the following equation [7,17],

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \ . \tag{31}$$

The forward and backward probabilities can be used to find the state occupancy probability $\gamma_t(i)$, i.e., the probability of being in state i at time t,

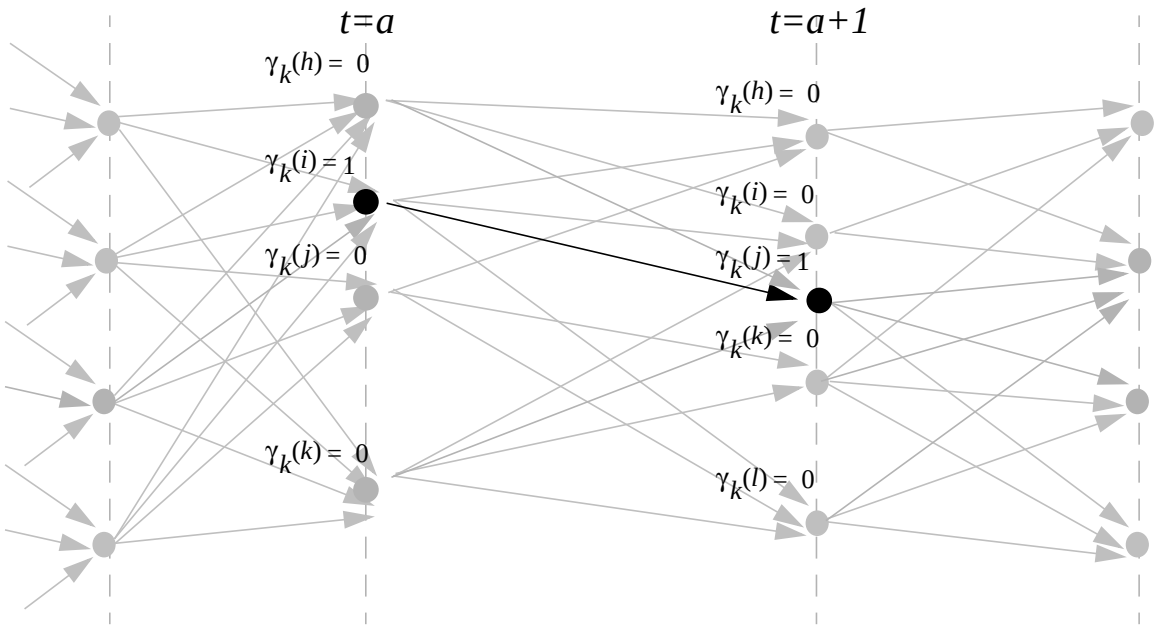

Figure 4. An example of a Viterbi training pass in which at each time instance the HMM can be in only one state.

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} . \tag{32}$$

2.8. Parameter Estimation

The parameter reestimation process requires two intermediate terms to be calculated: (1) the state occupancy probability $\gamma_t(i)$ described in equation 32, and (2) $\xi_t(i, j)$ which is defined as the probability of being in state i at time t and moving to state j at time $t + 1$,

$$\xi_t(i,j) = \frac{\alpha_t(i)\alpha_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)\alpha_{ij}b_j(o_{t+1})\beta_{t+1}(j)} . \quad (33)$$

Using these two terms, we can compute the reestimated state transition probabilities and state observation probabilities [7],

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \quad (34)$$

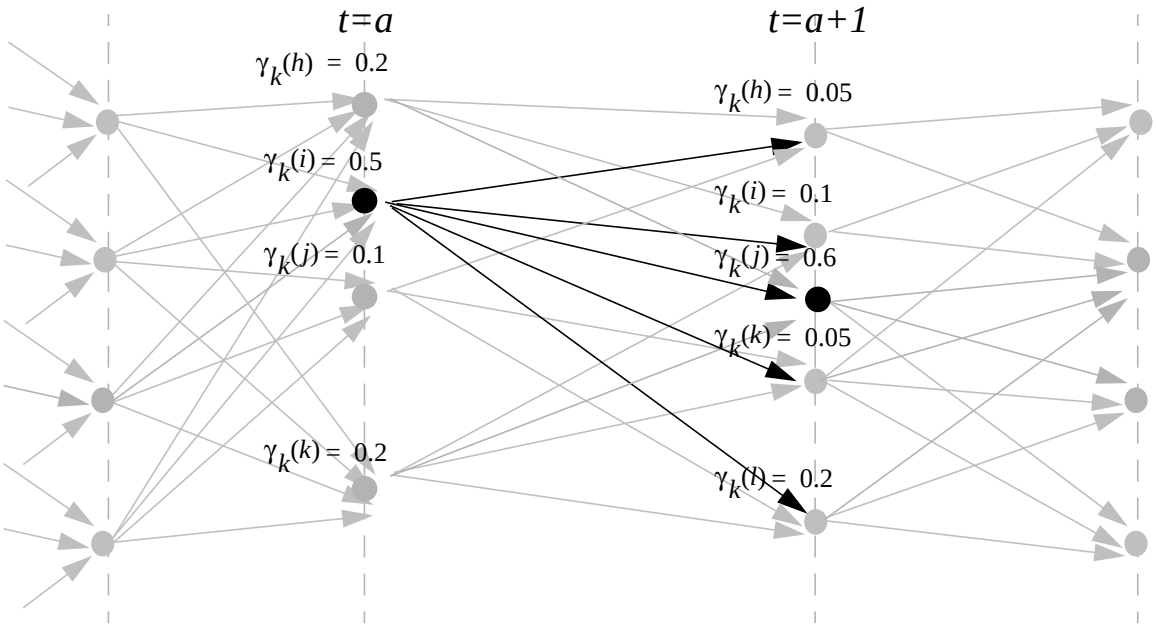

Figure 5. An example of a Baum-Welch training pass in which at each time instance the HMM can be in any of the N state.

$$\hat{b}_j(k) = \frac{\sum_{t=1}^{T-1} \gamma_t(j)'}{\sum_{t=1}^{T-1} \gamma_t(j)} \tag{35}$$

where $\gamma_t(i)'$ represents the summation over all state occupancies in the model λ that relate to a specific symbol or phoneme whose parameters we are trying to estimate.

2.9. Viterbi Training

The Viterbi approach is referred to as a hard decision criteria, i.e., at each time instance in the trellis the HMM can be in one and only one state. The Viterbi procedure tries to find the single best path through the trellis [7,17,30]. The algorithm is similar to the forward procedure with the summation in Equation 28 replaced with a maximization,

$$\alpha_{t+1}(j) = \max_{1 \le i \le N} [\alpha_t(i) a_{ij}] b_j(o_{t+1}) \ . \tag{36}$$

At any time instance the state with the best score is selected, i.e., the state occupancy $\gamma_t(i)$ of the state with the maximum probability is set to one and the others are set to zero. This is why the Viterbi procedure is regarded as a hard decision criteria because we typically use an integer counter to track the number of times a state is visited. The $\gamma_t(i)$ terms are then used to iteratively reestimate the HMM parameters via the EM algorithm. An example of a Viterbi training pass is shown in Figure 4.

2.10. Baum-Welch Training

The Baum-Welch approach is referred to as a soft decision criteria, i.e., at each time instance in the trellis the HMM has some probability of being in any of the N states [7,17]. The Baum-Welch equations guarantee that the sum of the state occupancies across all N states in the trellis for any time instance is one,

$$\sum_{i=1}^{N} \gamma_t(i) = 1.0 \ . \tag{37}$$

At any time instance the state with the best score is assigned the highest probability, i.e., the $\gamma_t(i)$ for that state is assigned the highest probability. However, unlike the Viterbi procedure, the remaining states do have some chance (however small) of being visited. The $\gamma_t(i)$ terms are then used to iteratively reestimate the HMM parameters via the EM algorithm. An example of a Baum-Welch training pass is shown in Figure 5.

The soft decision criteria, which is used in Baum-Welch, can be used to reestimate the parameters of a hierarchical network of HMM's — an approach that is popular in state of the art speech recognition systems [7]. The network parameter reestimation capability can be leveraged for tasks such as language modeling, acoustic unit duration modeling, and pronunciation modeling. Also, the parameter reestimation forms the basis of the network training framework, which is discussed next in Chapter 3.

CHAPTER III

NETWORK TRAINING

In the traditional acoustic model training recipe, a single, most likely, pronunciation is selected for each word. This approach requires the trainer to make a hard decision about which pronunciation is used. It is well known that systems involving soft decisions can provide better performance though these systems may take longer to converge [8,17] during training. In this chapter, we introduce a network training approach that directly trains multi-path models at any level of the speech recognition model hierarchy without the need for complicated systems and training recipes.

The first section of this chapter introduces the network training framework and algorithm. The second section describes the training recipe used in the traditional training framework and compares it to the network training framework. The third section focuses on the differences in duration modeling techniques used by the two systems, i.e., the traditional and network trainer. Finally, the fourth section shifts focus from duration to pronunciation modeling and describes the differences between the two systems.

3.1. Framework

The network training framework employs maximum likelihood estimation (MLE) within the expectation maximization (EM) framework to reestimate the parameters of the

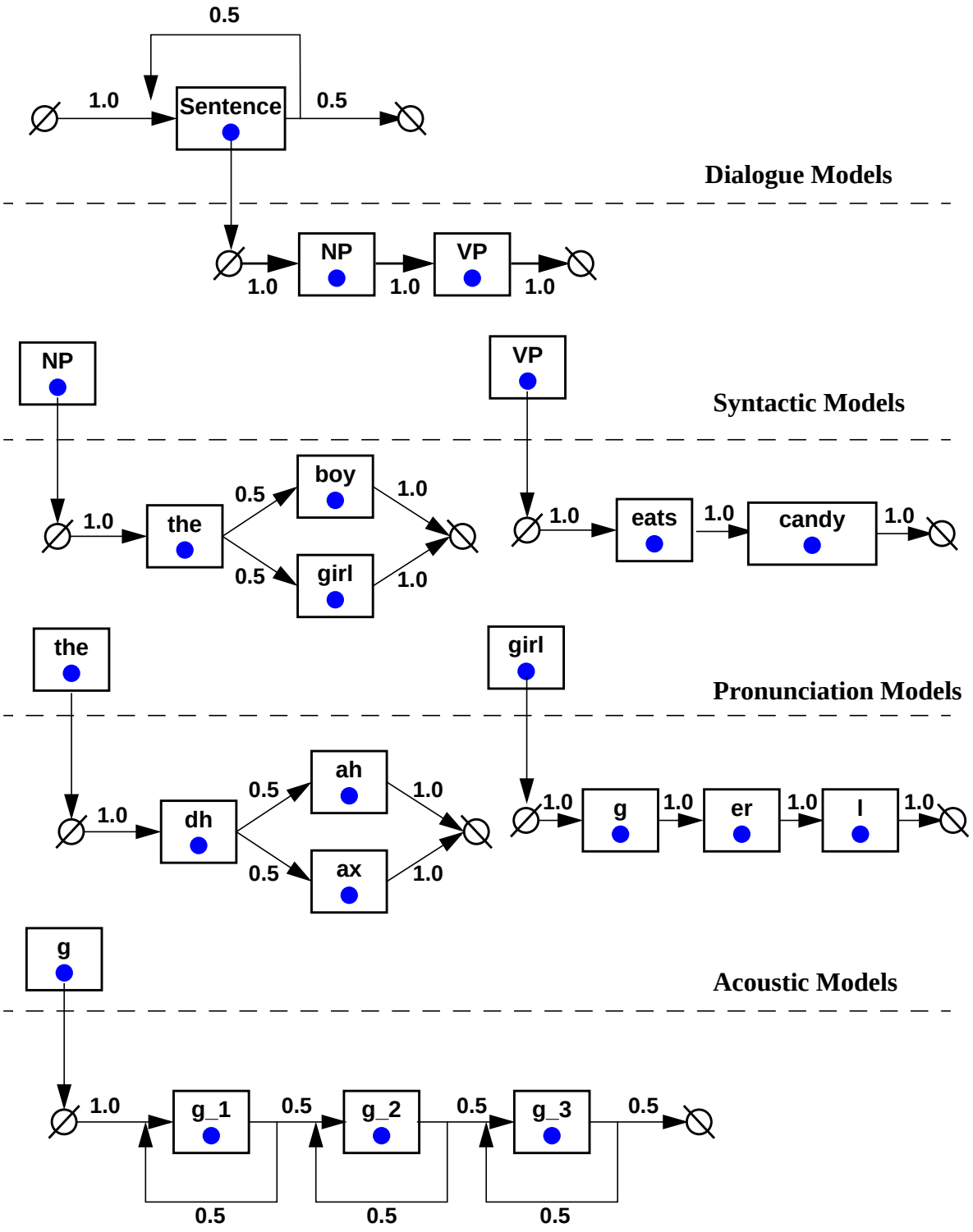

Figure 6. An example of a hierarchical system that contains embedded knowledge sources at each level in the hierarchy.

acoustic models. More specifically, the network training framework uses the Baum-Welch algorithm to reestimate the parameters of the Gaussian mixture models (GMM's). The above description on the surface appears identical to the training paradigm used in a traditional trainer; however, it must be noted that the key difference here lies in the fact that the Baum-Welch reestimation procedure is applied to a hierarchical network, as shown in Figure 6. The ability to train a hierarchical network is what differentiates the network trainer from a more traditional left-to-right HMM trainer.

The reestimation equation for the transition probabilities of the hierarchical network is given by the ratio of $\xi_t(i, j)$ and $\gamma_t(i)$ in Equation 38. The reestimation equations of the GMM's are given by the following set of equations [7,17]

$$\hat{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t(j,k)} \tag{38}$$

$$\hat{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \bullet O_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{39}$$

$$\hat{U}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \bullet (O_t - \mu_{jk})(O_t - \mu_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{40}$$

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}\right]\left[\frac{c_{jk}N(O_t,\mu_{jk},U_{jk})}{\sum_{m=1}^{M}c_{jm}N(O_t,\mu_{jm},U_{jm})}\right] \quad (41)$$

where $\hat{c}_{jk}$ represents the reestimated mixture weights, $\hat{\mu}_{jk}$ represents the reestimated Gaussian mean vector, $\hat{U}_{jk}$ represents the reestimated Gaussian covariance matrix and $\gamma_t(j,k)$ represents the probability of being in state j, mixture k at the time instance t

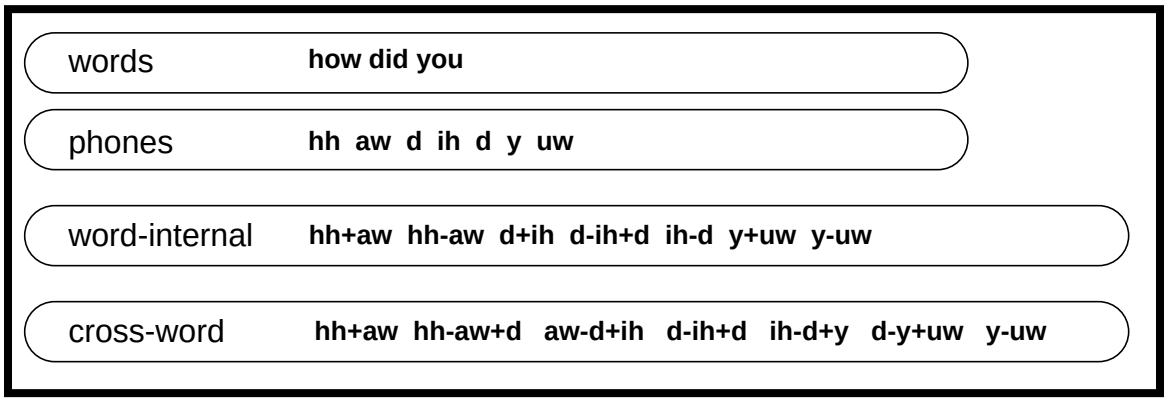

Figure 7. Examples of the different types of transcriptions used in the recognition process. The first two are referred to as context independent, while the last two are context dependent.

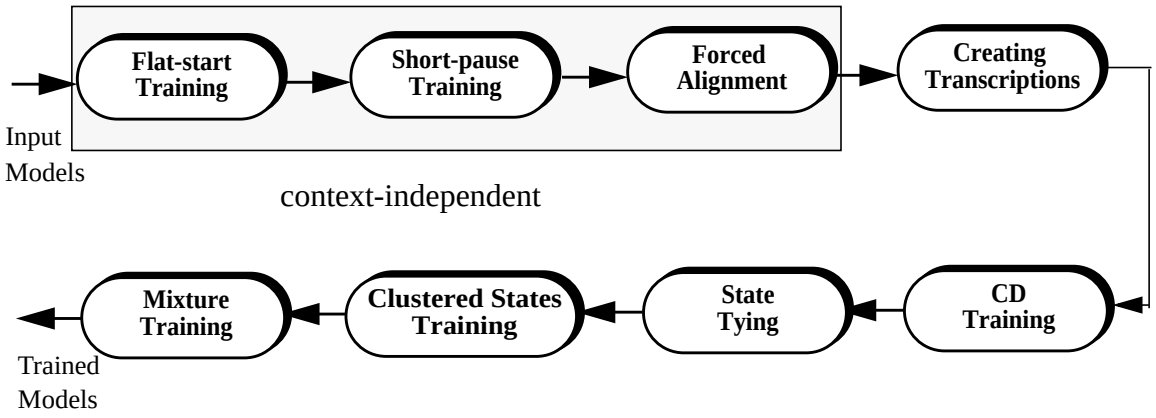

Figure 8. A block diagram representation of the individual steps in a typical training recipe for a traditional speech recognition system.

during the reestimation process. Also, the term $\hat{c}_{jk}$ represents the mixture weight component of the GMM. The reestimated probability is nothing more than the expected number of transitions from symbol j to symbol k, over the expected number of transitions from symbol j. The expectation in this case is computed over time and is computed via a time-expanded search space, i.e., a trellis. Hence, the Baum-Welch algorithm can be generalized to any level of the network hierarchy.

3.2. Training Recipe

During the training process, we provide the system with examples and have it learn the relationships between the labels and their corresponding observations. The labels in this case are the word-level transcriptions, while the observations are the acoustic features generated from the speech signal. The training process is split into two phases: the context-independent phase and the context-dependent phase. In the first phase of training, the phones (the base speech sounds) are assumed to be independent of each other [31]. In the second phase of training, each phone is assumed to be dependent on its neighboring phones [31, 32]. The second phase can take the form of word-internal or cross-word training [32, 33] depending on the desired context as shown in Figure 7. In the traditional training framework, a training recipe is decomposed into eight stages: flat-start, short pause training, forced alignment, transcription creation, context-dependent training, state-tying, clustered-states training and mixture training. A block diagram representation of the eight stages in the training recipe is shown in Figure 8. Flat start, short pause

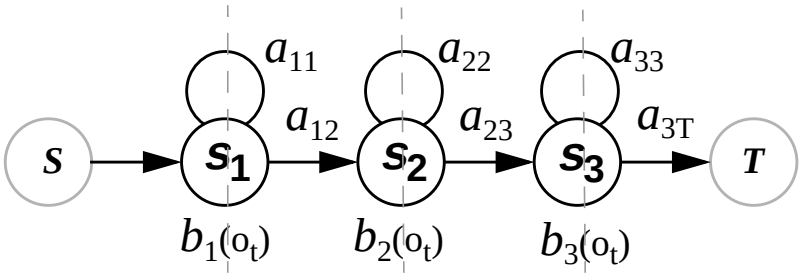

Figure 9. The topology of a three-state left-to-right HMM with self-loops used to model both speech and non-speech sounds.

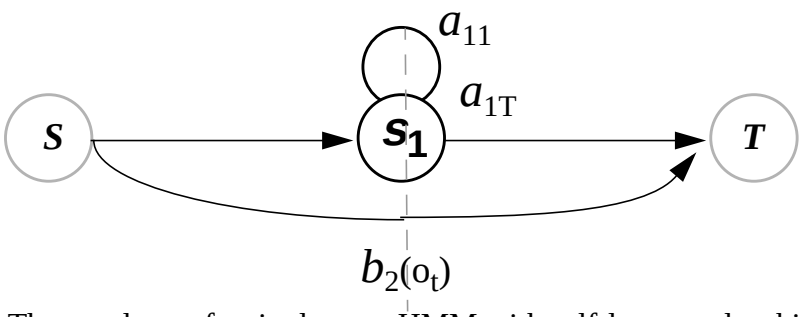

Figure 10. The topology of a single state HMM with self-loops and a skip transition used to model short pauses.

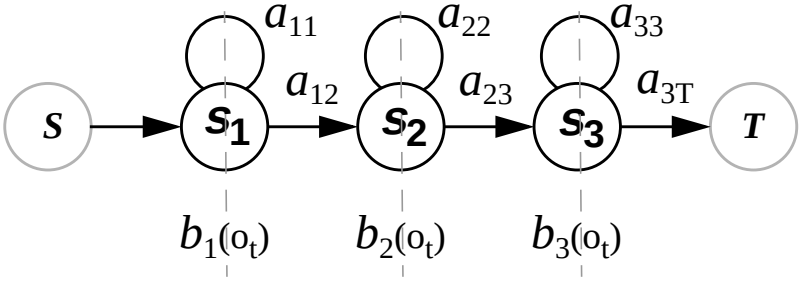

Figure 11. The topology of a silence model used during the flat-start training stage consists of a three-state left-to-right HMM with self-loops.

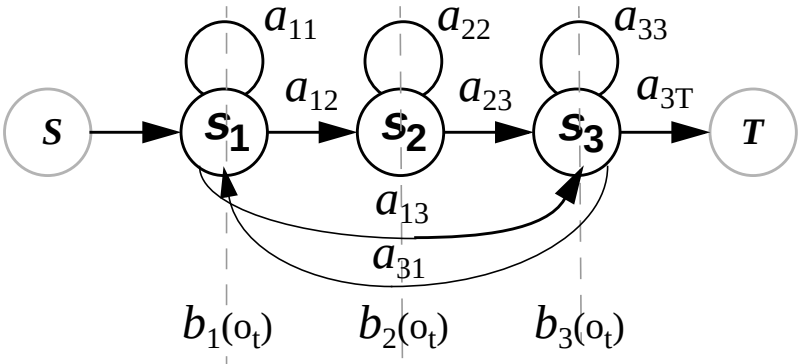

Figure 12. The topology of a silence model used during the short-pause stage of training. This silence model uses a three state left-to-right HMM with self-loops and new transitions are added from the first state to the third state and vice versa.

training, and forced alignment are part of the context-independent phase of training, while the remaining stages make up the context-dependent phase of training.

In the flat-start stage, a silence is appended to the start and end of each transcription. This stage takes advantage of the fact that speech utterances typically are processed in segments separated by silence. During this stage, speech and non-speech models (silence) are estimated using four iterations of Baum-Welch training. The main goal in flat start is to get a good estimate of the segment boundaries, because poor segment boundary estimates can deteriorate performance in the later stages of training. Poor segment estimates can cause an overlap between speech and non-speech model parameters in the early stages of training, which is hard to recover from in later stages of training.

The speech and non-speech sounds are modeled by a three-state left-to-right HMM [25,32] with self-loops, as shown in Figure 9 and Figure 10. In the network training framework, a three-state silence model is forced at the start and end of each transcription, similar to the traditional trainer. However, unlike the traditional trainer, the network trainer does not require a new set of transcriptions for this stage. The main reason for this is that the network training process is automated, i.e., the silence model is automatically inserted at the transcription boundaries. This automation saves resources as we are not required to set up a new set of transcriptions specifically for this stage.

The short-pause stage is an extension of the flat-start stage in which we insert a short-pause symbol between each word in the transcription. In the short-pause stage, the topology of the silence model is modified by adding transitions from the first state to the third state and vice versa, as shown in Figure 11 and Figure 12. The transitions are added

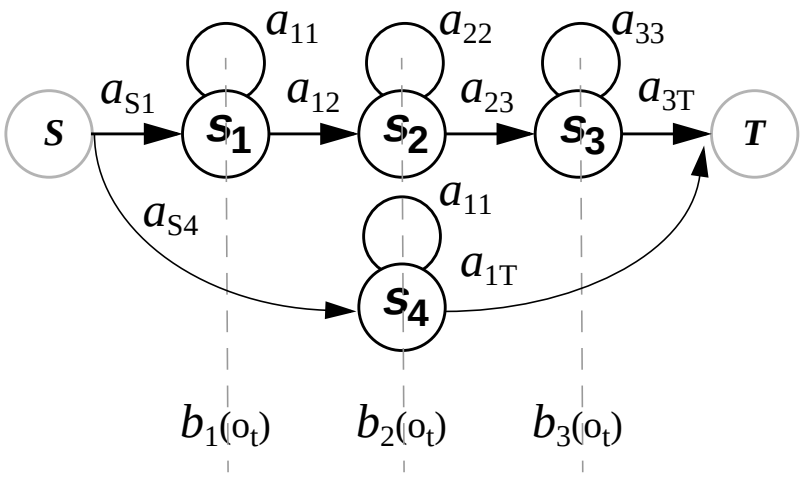

Figure 13. The topology of a multi-path silence model. The model has a path consisting of three states (s_1, s_2, and s_3) that models long durations of silence, and a path consisting of one state (s_4) that is used to model short pauses.

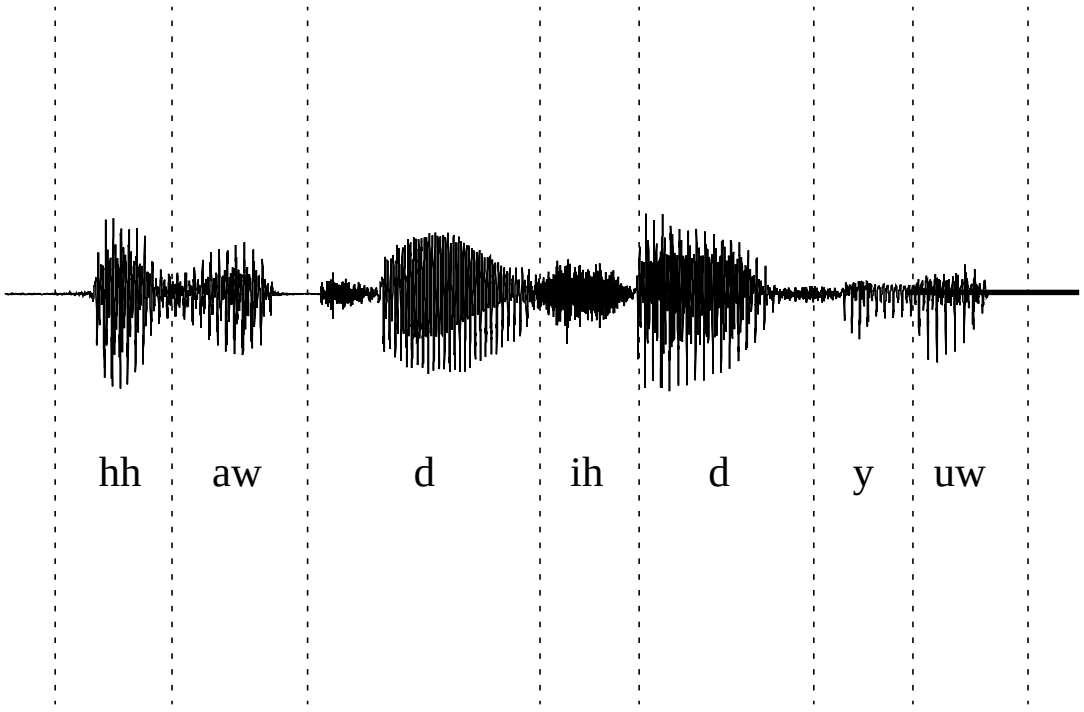

Figure 14. An example of how the transcription “how did you” is aligned to the speech signal. The forced-alignment stage selects the most likely pronunciation for each word in the transcription and aligns the pronunciation to the speech signal.

to take into account impulsive noise in the speech signal. An impulse or a noise spike can cause the system to leave the center state in the silence model prematurely [25,32,34]. The transitions give the system a chance to recover or return to the center state. The short-pause model consists of a single state HMM with a self-loop and a skip transition, as shown previously in Figure 10. The single state is tied to the center state of the silence model.

The main goal in the short-pause stage is to model silence between words. During this stage of training, the models are reestimated using four iterations of Baum-Welch. In the network training framework, a multi-path silence model, shown in Figure 13, is inserted between each word of the transcription. The network trainer does not require a new set of transcriptions for this stage because the entire process is automated, similar to flat start. The ability to use a multi-path silence model also saves resources since we are again not required to use a new set of transcriptions for this stage of training.

During the forced-alignment stage, the phone sequence corresponding to each transcription is determined by aligning the transcription to the speech data, as shown in Figure 14. In the forced-alignment stage, each word is defined by the set of pronunciations available to it in the lexicon. Each pronunciation in the lexicon has two variants — one has a silence appended to it, the other has short pause appended to it. By aligning the transcriptions we achieve two goals: we determine the most likely pronunciation (phone sequence) for each word and we determine the duration of the silence model (silence or short-pause) used between words [25,32,35,36]. The network trainer does not require the forced-alignment stage since it employs word networks [37]. The word networks

inherently allow multiple pronunciations for each word in the lexicon. The ability to use word networks saves resources since we are not required to set up a new set of transcriptions for this stage of training.

The transcription creation stage is an extension of the forced-alignment stage. Using the phonetic alignments of the word transcriptions we create new phonetic transcriptions [25,32,36]. The new phonetic transcriptions take the form of either word-internal or cross-word transcriptions depending on the desired context, as previously shown in Figure 9. This stage marks the beginning of the context-dependent phase, and the context-dependent phone transcriptions generated here are used in the following training stages. Note that the traditional trainer always uses phone transcriptions during training. The network trainer on the other hand uses word transcriptions, which is why the transcription creation stage is unnecessary.

In the context-dependent training stage, each phone in the transcription is modeled using the context of the surrounding phones. In the case of triphones, each phone is modeled using a context of the phone to its left and the phone to its right, as previously shown in Figure 9. Each triphone is associated with the HMM corresponding to its center phone. The context-dependent transcriptions, which are used during training, are the outcome of the previous stage. The models in this stage are updated using four iterations of Baum-Welch reestimation. The network trainer, which uses word transcriptions, generates the context-dependent phones dynamically during the training process. Dynamically generating the context-dependent phones adds additional overhead to the

real-time rate of the trainer. However, this overhead is acceptable given that it simplifies the training recipe.

The state-tying stage draws on domain knowledge to cluster similar context-dependent phones together [38,39]. The clustering process ties the probability distributions of the context-dependent models together using linguistic rules. This is a very important step because the training data does not contain sufficient examples of all context-dependent phones to yield robust models [36,40]. Note that the state-tying stage only ties the observation probability distributions and not the state transition probabilities. The state-tying stage is similar for both the traditional trainer and the network trainer.

In the clustered-states training stage, the clustered models are reestimated using four iterations of Baum-Welch. The main reason behind this is that after the state-tying stage the probability distributions of the tied models tend to be very peaky, and four iterations of Baum-Welch are intended to smooth them [36,40]. The clustered-states training stage is similar for both the traditional trainer and the network trainer.

The mixture training stage splits the probability distributions of the models, i.e., the mean is shifted one standard deviation in either direction and the variance is kept the same [36,41]. The theory behind using mixtures is to enable the system to better model the underlying characteristics of the speech signal such as speaker and channel variations. The mixture splitting process takes place in multiples of two, i.e., two, four, eight, sixteen, etc. After each split the models are reestimated using four iterations of Baum-Welch. The mixture training stage is similar for both the traditional trainer and the network trainer.

3.3. Duration Modeling

In the traditional training paradigm, a forced-alignment stage is used to determine the duration of the silence model used between words. The reason this is done is because a GMM, which is used to represent the underlying probability distribution, has an exponentially decreasing likelihood of staying in the same state over time [7,17]. Hence, a short-pause model — a single state silence model with a self-loop and a skip transition — cannot be used to model longer silence durations between words.

In the network training paradigm, a forced alignment is not necessary. This is because the multi-path model — which is inserted between each word in the transcription — provides the option of either a long path (three-state path) or a short path (one-state path) through the silence model. The system is given the opportunity to select the most likely path through the silence model. This is similar to the forced-alignment stage in the traditional training framework.

Using a multi-path silence model, however, has its disadvantages. The advantage of using a multi-path silence model is that we don't need to generate new transcriptions. In the traditional training paradigm, new transcriptions are generated three times in the context-independent stage alone. Alleviating the need to generate new transcriptions for each stage reduces the complexity of the training process, which in turn simplifies the user interface for this process[1]. The drawback of using a multi-path silence model is the added confusion introduced in training, since the system must learn to discriminate between the

1. These intermediate transcriptions are stored in separate files. These files often are corrupted or incorrectly matched with their corresponding audio files. This results in failed experiments and creates a great deal of confusion for the novice user.

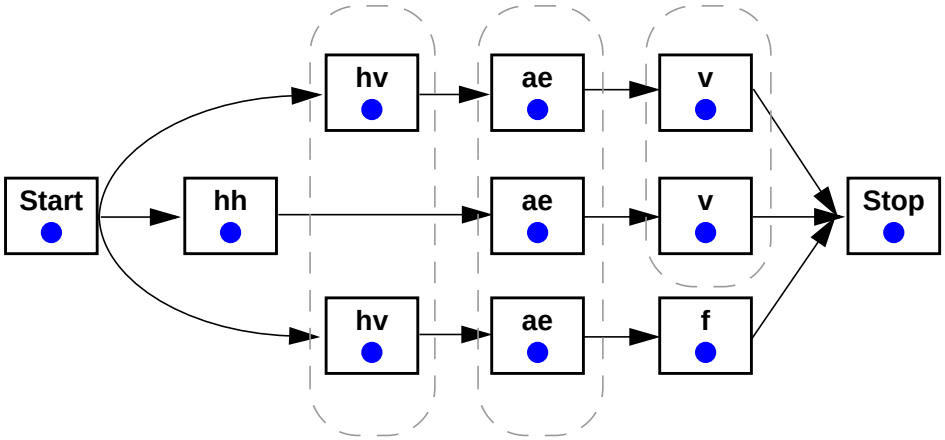

Figure 15. The word “have” has three different pronunciations that share phone models.

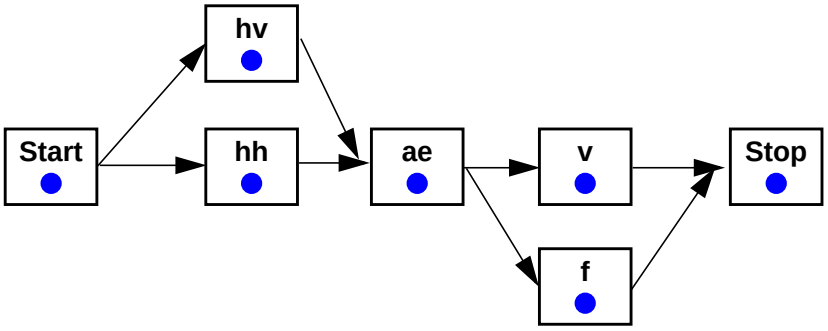

Figure 16. A multi-path pronunciation model for the word “have.”

two path options (e.g., short vs. long). In the next chapter, we will see that this ambiguity must be carefully moderated during training to avoid divergence of the model.

3.4. Pronunciation Modeling

In the traditional training paradigm, a single pronunciation is used for all words in the vocabulary during training. The lack of sufficient training data, which is needed to cover all possible variants in the pronunciations, negates any advantage gained by using multiple pronunciations due to the increased complexity added to the system [23]. Basic

training works by using a single canonical pronunciation for each word during the flat-start and short-pause stages. The most likely pronunciation (phone sequence) for each word is selected during the forced-alignment stage.

In the network training paradigm, we employ word networks for modeling the pronunciation variants [37]. A word network consists of a series of unique paths representing variants of the canonical pronunciation. Examples of such networks are given in Figures 15 and 16. While the canonical pronunciation is obtained from the dictionary, variants of the canonical pronunciation are obtained from various sources which include text-to-phone systems and pronunciation dictionaries. In Figure 15, the word “have” has three different canonical pronunciations and the common phones in each pronunciation share emission probabilities. In Figure 16, the word “have” is realized using a multi-path pronunciation model. Such networks allow us to generalize to pronunciations not encountered in the training corpus. Word networks allow us to skip the forced-alignment stage and simplify the training recipe.

The network trainer can be used to directly infer the pronunciation probabilities of the word network from the data. The pronunciation probabilities are estimated by applying the Baum-Welch algorithm to the hierarchical network, as previously described in Figure 6. The transition probabilities for each word network, corresponding to each word in the lexicon, are estimated in a manner similar to how the HMM transition probabilities are estimated. Hence, reestimation of the pronunciation probabilities fits nicely within the Baum-Welch framework.

Despite its advantages, word networks experience the same problem of sparse training data, i.e., despite the large volume of training data, many cross-word triphones have insufficient examples to yield robust models during reestimation. The problem occurs when reestimating the output probability distributions and the pronunciation probabilities of the word networks. The sparsity of training data leads to poorly estimated models [23, 42], which in turn leads to poor recognition performance.

3.5. Recipe Comparison

Previously, we gave a brief description of the different stages of the traditional training recipe, and we provided details on how the network training recipe differs from the traditional training recipe. In Table 1, we show a side-by-side comparison of the two training recipes. The table shows the different stages of training and the number of passes of Baum-Welch reestimation for each stage.

Table 1. A detailed comparison of the different stages in the training recipe for both the network trainer and the traditional trainer.

Context	Training Stage	Traditional Trainer	Network Trainer
CI Training	Initialize	Yes	Yes
	Flat-start	4 passes	4 passes (using a fixed silence model at transcription boundaries)
	Short-pause	4 passes	9 passes (using an optional multi-path silence model between words and a fixed silence model at transcription ends)
	Forced-alignment	5 passes	

Table 1. A detailed comparison of the different stages in the training recipe for both the network trainer and the traditional trainer.

Context	Training Stage	Traditional Trainer	Network Trainer
CD Training	Create phone transcriptions	Yes	No
	CD training	4 passes	4 passes
	State-tying	Yes	Yes
	Clustered states training	4 passes	4 passes
	2-mixture training	4 passes	4 passes
	4-mixture training	4 passes	4 passes
	8-mixture training	4 passes	4 passes
	16-mixture training	4 passes	4 passes

The major differences in the two training recipes occur before the context-dependent training phase (CD training). In the context-independent phase (CI training) both training recipes require the models to be initialized and flat-started. During the flat-start stage both training recipes use a fixed three-state silence model at the transcription ends. The network training recipe does not require the short-pause, forced-alignment and transcription creation stages. However, the network trainer does include nine passes of reestimation using an optional multi-path silence model between words and a fixed silence model at transcription ends.

In the next chapter, we will validate the claims made in this chapter by showing experimental evidence on both clean and noisy data sets. These experiments will compare the network training and the traditional training frameworks on the same tasks.

CHAPTER IV

EXPERIMENTS AND ANALYSIS

The primary objective of this thesis is to create a network training paradigm that allows for direct training of multi-path models and alleviates the need for complicated systems and training recipes. To prove the above hypothesis, experiments were conducted on three corpora representing industry-standard tasks: (1) speaker independent continuous digit recognition on data collected in studio-quality recording conditions, (2) spoken letter and number recognition on data collected over long distance telephone lines, and (3) read sentences from a command and control application collected in studio-quality recording conditions. The experiments described in this chapter compare the performance of speech recognition systems that have been trained using both the network training recipe and the traditional training recipe.

The first section of this chapter describes the corpora used to prove the above hypothesis. The second section discusses the network topology used in network training with special emphasis given to optional silence training. The third section presents experimental results on a digit recognition task. The fourth section presents experimental results on a spoken letter and number recognition task. Finally, the fifth section presents experimental results on a read sentence corpus.

4.1. An Overview of the Corpora

The performance of a speech recognition system can vary depending on the vocabulary size and the quality of the speech recordings. Hence, the corpora used to run experiments and verify a hypothesis are extremely important. This thesis makes use of three corpora: TIDigits [43], OGI Alphadigits [44] and Resource Management [47].

The TIDigits corpus was collected by Texas Instruments (TI) in 1983 to establish a common baseline for performance on connected word recognition tasks. The corpora has a vocabulary of eleven words. This includes numbers from ‘zero’ through ‘nine’ and ‘oh’ (an alternate pronunciation for ‘zero’). The recording conditions consisted of speech collected in a studio quality recording environment. The corpora consists 326 speakers (111 men, 114 women and 101 children).

The TIDigits corpus was initially selected because of its small vocabulary size and studio-quality recording environment. The corpus is a good base condition to test our hypothesis because we can initially ignore issues such as channel noise and sparse training data and focus on the network training framework.

The OGI Alphadigits corpus was collected by the Oregon Graduate Institute (OGI). The data was collected using the CSLU T1 digital data collection system — a digital interface into the public telephone network. The sampling rate was 8 kHz and the files were stored in an 8-bit mu-law format. The vocabulary includes all letters in the English alphabet (e.g., ‘a’) and digits (‘zero’ to ‘nine’ including ‘oh’). The recording conditions included a variety of telephone handsets and long-distance telephone

limes, and hence represents a moderately noisy recording environment. The corpus consists of 2,983 speakers (1,419 men, 1,533 women and 30 children).

The OGI Alphadigits corpus was selected because it is a much harder acoustic modeling problem than TIDigits**.** In addition to the noisy recording conditions, the corpus contains what are known as minimal pairs [44], i.e., words such as “p” and “b” which differ only in one linguistic feature. Since any of the 37 words in the lexicon can follow any other word, a language model cannot be used to help disambiguate hypotheses. Good performance on this task requires good acoustic modeling, which is the focus of HMM training. Finally, state of the art performance on this task is a word error rate of about 10% [45,46], which is sufficiently high to observe differences in acoustic modeling technology and to measure statistically significant differences in performance.

The Resource Management corpus [47] was collected by the Defense Advanced Research Projects Agency (DARPA). The corpus is a collection of recordings of spoken sentences pertaining to a naval resource management task. The recording conditions consisted of speech collected in a low background noise environment using a Sennheiser HMD 414 headset microphone. The corpus consists of 80 speakers, each reading two “dialect” sentences plus 40 sentences from the Resource Management text corpus.

The Resource Management corpus was selected because of its medium-sized vocabulary (1000 words) and clean recording conditions. The corpus was specifically designed for the purpose of evaluating new algorithms and training concepts on a continuous speaker independent recognition tasks. The corpus uses a language model and covers all phonemes in the English language, unlike the previous corpora. Resource

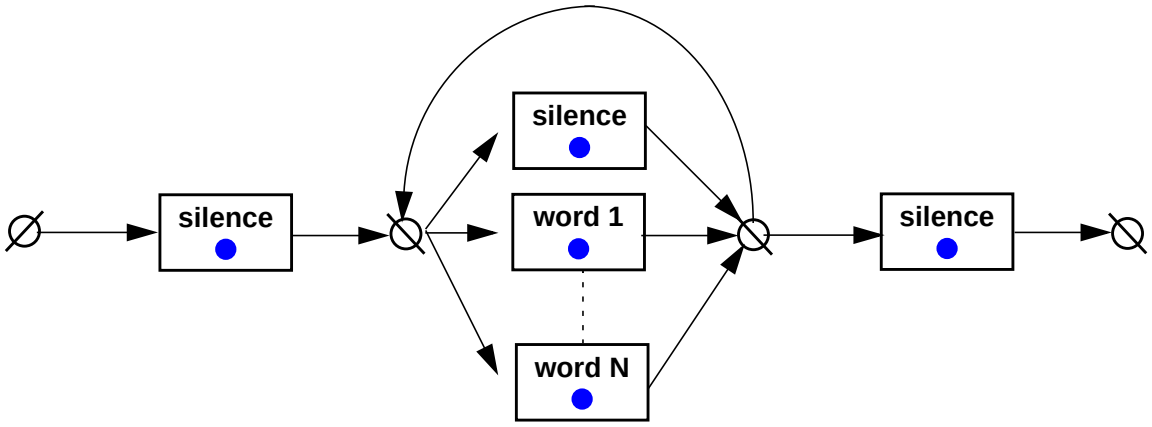

Figure 17. An example of a language model employed by the network trainer.

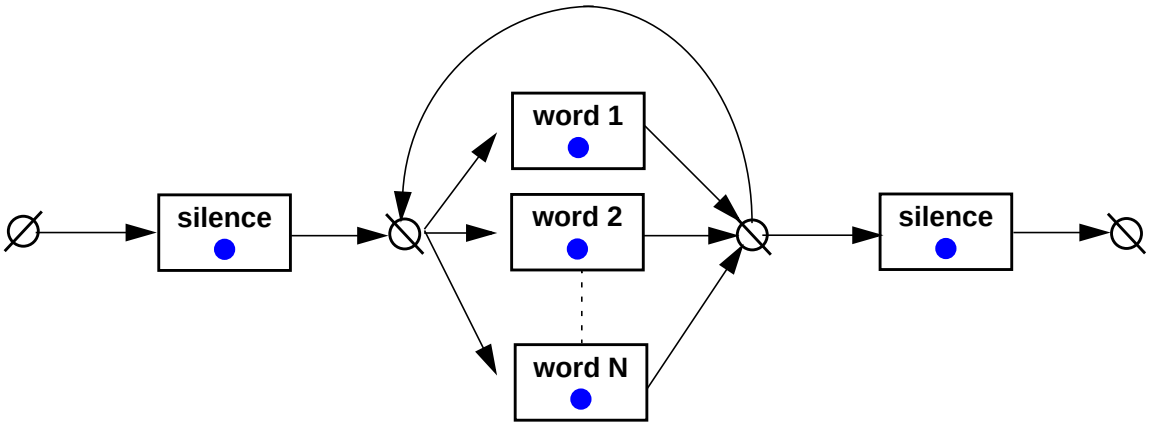

Figure 18. A example of a language model employed the traditional trainer.

4.2. Silence Model Training

A good silence model is very important in speech recognition since non-speech segments of the signal are mapped to this model. A good silence model prevents overlap between speech and non-speech segment boundaries. If these boundaries are not properly estimated during training, poor performance will be observed. The network trainer uses an optional silence between words in the transcription during the training process, as shown

Management is also small enough that we don't have to get involved in all the computational issues involved with large vocabulary tasks.

The results presented in this chapter primarily use context-independent models for training as well as recognition. Although the context-dependent stage is a direct extension of the context-independent stage, and requires no changes in the training recipe, an efficient tree-based decoder is needed to decode cross-word models. An efficient tree-based decoder is currently under development in a related project but was not available at the time this research was performed.

in Figure 17. The optional silence used by the network trainer is different from that used by a traditional trainer, as shown in Figure 18. The main difference lies in the fact that the network trainer allows the silence to be optional, at the transcription level, whereas, the traditional trainer forces a silence between words, at the phonetic level.

In the previous chapter, some disadvantages of using an optional multi-path silence model during network training were briefly described. One disadvantage of using an optional silence model is that speech signals typically have a definitive segment of silence at signal boundaries. Hence, using an optional silence at transcription boundaries only adds confusion during the training process. One way to avoid this ambiguity is to use a fixed silence at transcriptions boundaries. Hence, we fix the silence at the transcription bounds and make it optional between words. The experimental results in Table 2 show

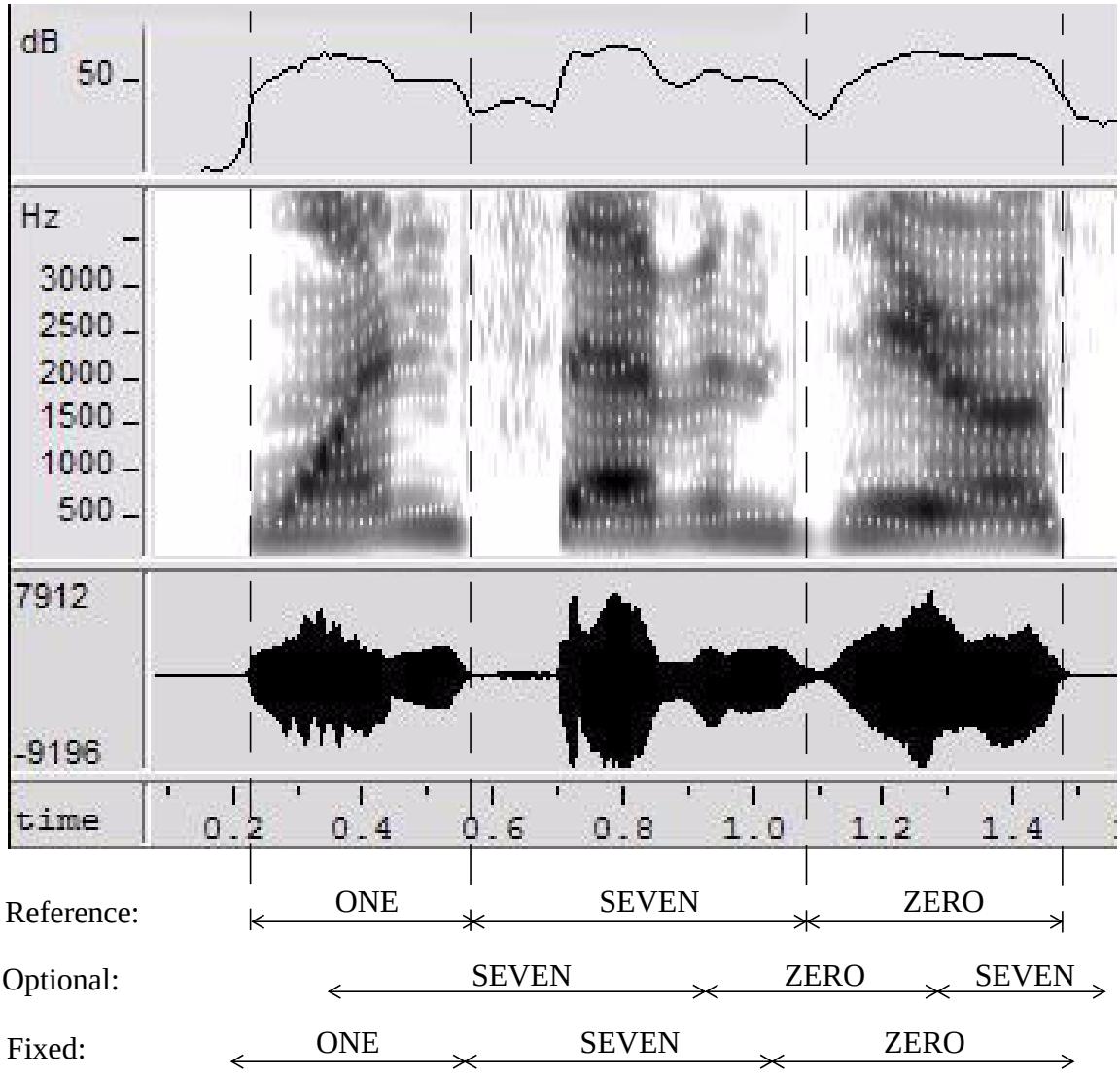

Figure 19. A comparison of time alignments using optional and fixed silence models.

how the recognition performance varies for a fixed versus an optional silence at transcription boundaries. The experiments were conducted on the TIDigits Corpus using word models.

The first row of Table 2 represents a condition in which silence is optional between words and at the beginning and end of an utterance. The second row represents an

experimental condition in which silence is still optional between words, but is required at the beginning and end of an utterance. The high substitution rate in the first row of Table 2 shows that there is a high degree of confusion in the models, which indicates that the models are not learning how to represent segment boundaries. An analysis of the time alignments generated by the two systems reveals that the system with an optional silence at transcription boundaries does not properly learn the segment boundaries.

Table 2. Variations in recognition performance for a fixed versus an optional silence at transcription boundaries.

System	Training Iterations	WER	Insertion Rate	Deletion Rate	Substitution Rate
Optional	4	45.3%	1.6%	31.1%	12.5%
Fixed	4	2.6%	0.4%	0.7%	1.5%

For example, in Figure 19, the word “one” follows the initial silence in the fixed model hypothesis (where silence is required at utterance ends). However, the word “seven” follows the initial silence in the optional model hypothesis (where silence is optional at utterance ends), which is the second word in the fixed silence hypothesis. It should be noted that the fixed silence hypothesis matches the reference transcription.

When we discussed the network training topology, previously shown in Figure 17, we did not justify why we use two silence models. We use a three-state silence model at transcription boundaries and a multi-path silence model between words. In order to understand the reason for the two silence models we need to look at the experimental results in Table 3, which compare the traditional trainer, shown in the first row, to the

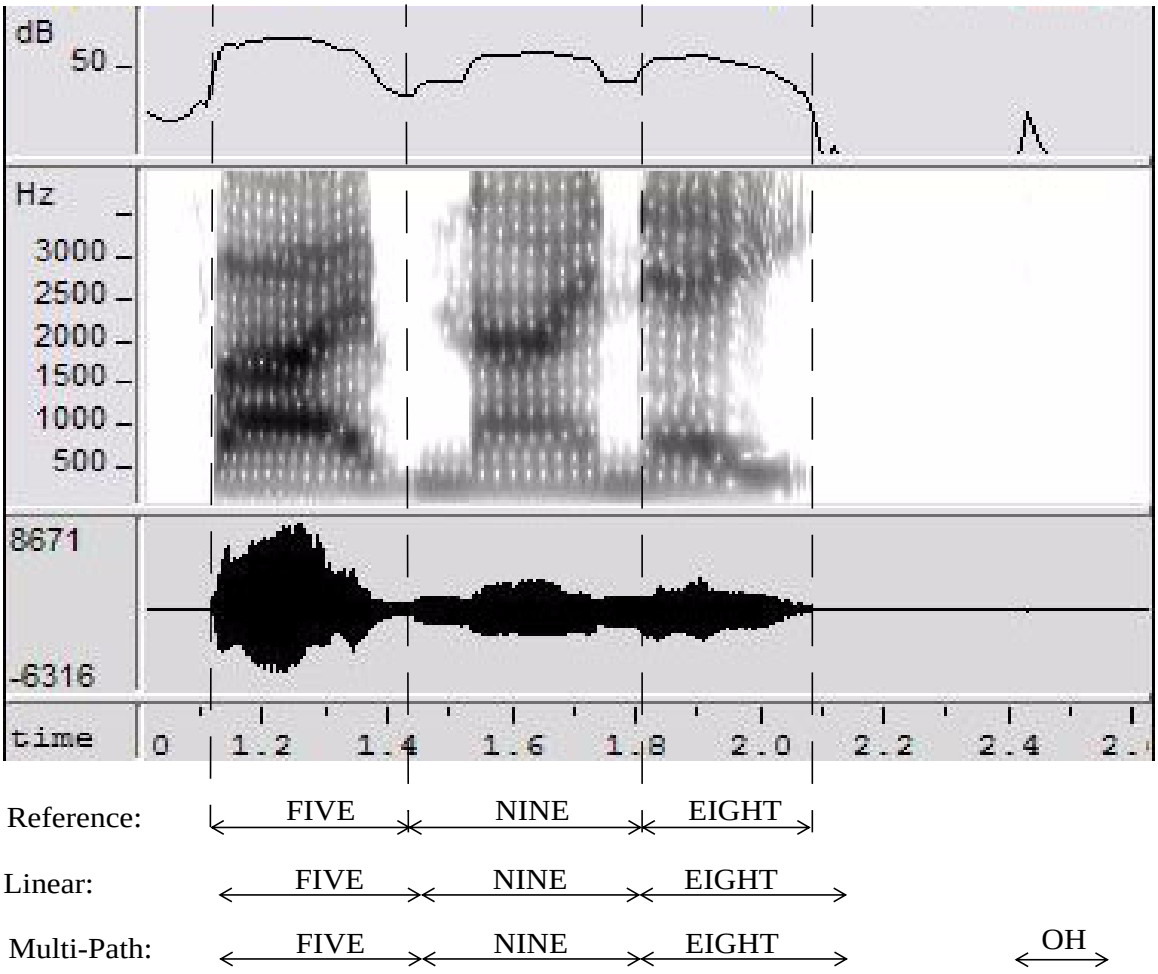

Figure 20. A comparison of time alignments using linear and multi-path silence models.

Hypothesis State Alignments:

start frame = 212	**stop frame = 213**	**state = S_1**
start frame = 213	**stop frame = 237**	**state = S_2**
start frame = 237	**stop frame = 241**	**state = S_3**

Reference State Alignments:

start frame = 212	**stop frame = 248**	**state = S_1**
start frame = 248	**stop frame = 270**	**state = S_2**
start frame = 270	**stop frame = 271**	**state = S_3**

Figure 21. The state-alignments for the utterance shown in Figure 20. The alignments focus on the tail end of the signal, i.e., the part following the word “eight”.

network trainer, shown in the second row. The experiments were conducted on the TIDigits corpus using phone models.

Table 3. A comparison of recognition performance for systems trained using a traditional trainer and the proposed network trainer.

System	Training Iterations	WER	Insertion Rate	Deletion Rate	Substitution Rate
Traditional	8	9.9%	4.2%	0.5%	5.2%
Network	8	10.5%	4.6%	0.4%	5.5%

The results show that the network trainer gives 0.6% degradation in WER compared to the traditional trainer. In order to analyze the 0.6% performance degradation in the network trainer, an utterance was selected which the traditional trainer recognized correctly but the network trainer recognized incorrectly. This utterance contained the word “oh”. The word “oh” was chosen because it had the highest number of word insertion errors in both systems. An analysis of the time alignments generated by the two systems, shown in Figure 20, does not reveal anything interesting except for the fact that the network trainer using the multi-path silence model inserts the word “oh” towards the end of the utterance. This seems to suggest that the multi-path silence model parameters have not been robustly estimated by the network trainer.

Also, an analysis of the state alignments produced by the network trainer reveals the observations shown in Figure 21. The state-level alignments in Figure 21 focus on the silence immediately following the word “eight”, as shown in Figure 20. The first two

columns list the start and stop frames respectively, i.e., the time interval spent in the state specified in the third column. A frame in this case represents 10 msec. Notice that when aligned using the reference transcription, the system spends as much time in state S_1 as it does in the entire silence model when aligned with the hypothesis. This suggest that the network trainer is having problems learning when to take the 3-state path versus when to take the 1-state path in the multi-path silence model. Hence, additional supervision is required for the silence model at the transcription bounds since speech signals have longer silence segments at the signal boundaries. This is an artifact of the way we excise the signal during data collection, and the way in which we run experiments.

4.3. Experiments on Digit Recognition

This section presents experimental results on the TIDigits corpora using context-independent phone models. The experimental results in Table 4 represent models that were reestimated using the traditional and network trainer respectively. The recognition experiments use a word insertion penalty of -90 (which was found to be optimal via a development test set). The recognition experiment also uses open beams, i.e., there is no pruning, and the experiments use a loop-grammar language model (any word can follow any other word).

Table 4. A comparison of the recognition results for the different stages of the traditional training recipe (first three rows) and the network training recipe (last two rows) respectively using context-independent phone models.

Stage	WER	Insertion Rate	Deletion Rate	Substitution Rate
Flat-start	8.7%	0.3%	2.7%	5.7%

Table 4. A comparison of the recognition results for the different stages of the traditional training recipe (first three rows) and the network training recipe (last two rows) respectively using context-independent phone models.

Stage	WER	Insertion Rate	Deletion Rate	Substitution Rate
Short-pause	8.2%	0.1%	2.7%	5.4%
Forced Alignment	7.7%	0.1%	2.5%	5.0%
Flat-start	8.7%	0.3%	2.6%	5.7%

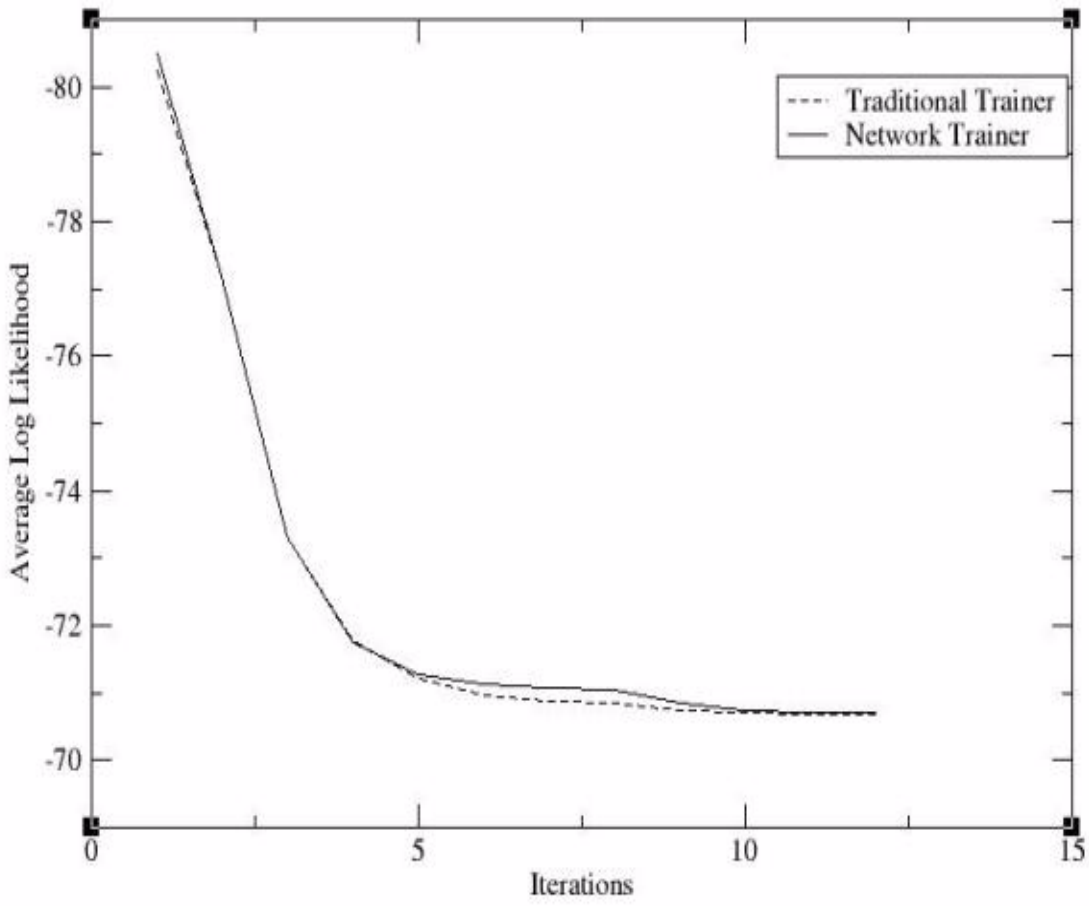

Figure 22. A comparison of the average log likelihood per iteration of training for both the network trainer and the traditional trainer on the TIDigits corpus.

Table 4. A comparison of the recognition results for the different stages of the traditional training recipe (first three rows) and the network training recipe (last two rows) respectively using context-independent phone models.

Stage	WER	Insertion Rate	Deletion Rate	Substitution Rate
CI	7.6%	0.1%	2.4%	5.0%

The experimental results in Table 4 show that the network trainer gives comparable performance to the traditional trainer on the TIDigits corpus using context-independent phone models. The experimental results show that the network trainer converges in word error rate to the traditional trainer. The substitution rate, which is a measure of the inherent confusion in the models, also indicates that the models reestimated by the network trainer are similar to the models reestimated by the traditional trainer.

Figure 22 shows the average log likelihood per iteration for both the network trainer and the traditional trainer. The plot in Figure 22 shows us that although the models reestimated by the network trainer start out a little worse, they eventually converge, in likelihood, to the models reestimated by the traditional trainer. Hence, the network trainer converges in both word error rate and likelihood to the traditional trainer on the TIDigits corpus using a simpler training recipe.

4.4. Experiments on Spoken Letter and Number Recognition

This section presents experimental results on the OGI Alphadigits corpora using context-independent phone models. The experimental results in Table 5 represent models that were reestimated using the traditional trainer and network trainer. The recognition experiments use a word insertion penalty of -90 (which was found to be optimal via a

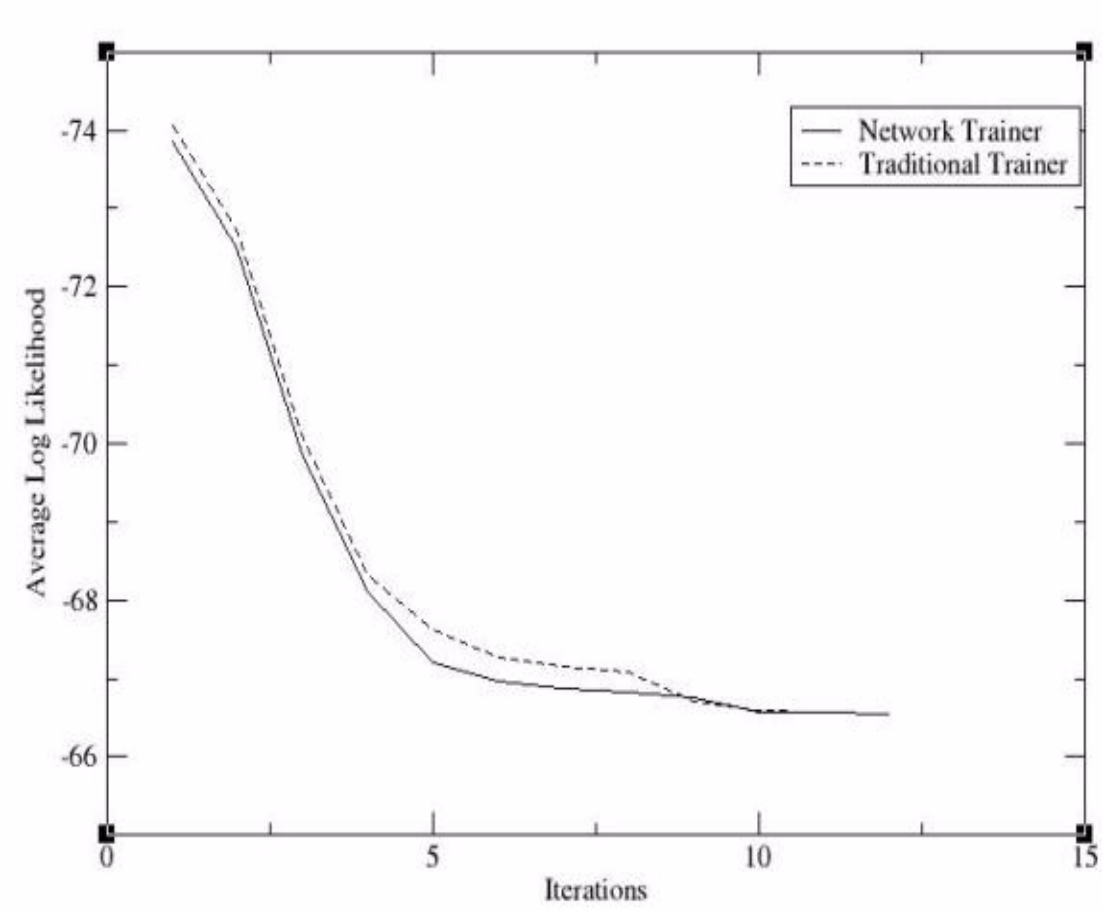

Figure 23. A comparison of the average log likelihood per iteration of training for both the network trainer and the traditional trainer on the OGI Alphadigits corpus.

development test set). The recognition experiment also use open beams, i.e., there is no pruning, and the experiments use a loop-grammar language model.

Table 5. These experiments show results, using monophone models, for models that were trained on the OGI Alphadigit corpus using the traditional trainer (first three rows) and the network trainer (last two rows) respectively.

Stage	WER	Insertion Rate	Deletion Rate	Substitution Rate
Flat-start	45.7%	2.1%	9.0%	34.6%

Table 5. These experiments show results, using monophone models, for models that were trained on the OGI Alphadigit corpus using the traditional trainer (first three rows) and the network trainer (last two rows) respectively.

Stage	WER	Insertion Rate	Deletion Rate	Substitution Rate
Short-pause	41.0%	1.2%	4.9%	35.0%
Forced Alignment	38.0%	0.8%	3.0%	34.2%
Flat-start	46.7%	2.5%	7.1%	37.2%
CI	35.3%	0.8%	2.2%	32.4%

The experimental results in Table 5 show that the network trainer gives a slight improvement in performance over the traditional trainer on the OGI Alphadigits corpus using context-independent phone models. The experimental results show that the network trainer converges in word error rate (with a 2.7% improvement) to the traditional trainer. The substitution rate also indicates that the models reestimated by the network trainer are similar to the models reestimated by the traditional trainer.

Figure 23 shows the average log likelihood per iteration for both the network trainer and the traditional trainer. The plot in Figure 23 shows us that the models reestimated by the network trainer converges, in likelihood, to the models reestimated by the traditional trainer. Hence, the network trainer converges in both word error rate and likelihood to the traditional trainer on the OGI Alphadigits corpus using a simpler training recipe.

4.5. Experiments on Read Sentence Recognition

This section presents experimental results on the Resource Management corpus using context-independent phone models. The experimental results in Table 6 represent

Figure 24. A comparison of the average log likelihood per iteration of training for both the network trainer and the traditional trainer on the Resource Management corpus.

models that were reestimated using the traditional trainer. The recognition experiments use a word insertion penalty of -90 (which was found to be optimal via a development test set). The recognition experiments also use a MAPMI pruning threshold of 10,000, a maximum word-end pruning threshold of 150 and word, phone and state level beam pruning thresholds of 250, 250, and 300 respectively. Furthermore, the recognition experiments use a standard bigram language model with a perplexity less than 60. A language model scale factor of 7.0 was used.

Table 6. A comparison of the recognition results for the different stages of the traditional training recipe (first three rows) and the network training recipe (last two rows) using context-independent phone models.

Stage	WER	Insertion Rate	Deletion Rate	Substitution Rate
Flat-start	28.6%	2.3%	7.1%	19.2%
Short-pause	26.5%	2.1%	7.0%	17.5%
Forced Alignment	25.7%	1.9%	6.7%	17.1%
Flat-start	29.5%	2.7%	7.1%	19.7%
CI	27.5%	2.6%	7.1%	17.9%

The experimental results in Table 6 show that the network trainer gives comparable performance to the traditional trainer on the Resource Management corpus using context-independent phone models. It should be noted that the 1.8% degradation in performance is not significant, and the experimental results in Table 6 were obtained using a much simpler training recipe than the traditional trainer. We use the matched pairs sentence-segment word error (MAPSSWE) test with a 0.1% confidence in order to

determine statistical significance [48]. The MAPSSWE test is the most powerful of the statistical tests used by the National Institute for Standards and Technology (NIST) for evaluating continuous speech processing tasks. The experimental results show that the network trainer converges in word error rate to the traditional trainer. The substitution rate indicates that the models reestimated by the network trainer are similar to the models reestimated by the traditional trainer.

Figure 24 shows the average log likelihood per iteration for both the network trainer and the traditional trainer. The plot in Figure 24 shows us that the models reestimated by the network trainer converges, in likelihood, to the models reestimated by the traditional trainer. Hence, the network trainer converges in both word error rate and likelihood to the traditional trainer on the Resource Management corpus using a simpler training recipe.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

The previous chapter of this thesis analyzed the effects of the network training recipe on the reestimation process. The network training recipe was discussed in detail in Chapters 3 and 4, and a step-by-step comparison with the traditional training recipe was provided in Chapter 3. Experiments performed on different corpora suggest that the network trainer gives better or comparable performance to the traditional trainer. This is primarily due to the fact that the network trainer uses a soft decision criteria, i.e., it does not force the trainer to learn a fixed solution during the reestimation process. The network trainer let's the data decide which solution is most likely during reestimation, while, giving the other solutions a chance as well (be it a very small chance).

5.1. Thesis Contribution

This thesis has explored the effectiveness of a novel training recipe in the reestimation process for speech processing. The effectiveness of the training recipe was demonstrated by analyzing the performance of the speech recognizer on three different corpora: TIDigits, OGI Alphadigits and Resource Management. For TIDigits, at a 7.6% WER, the performance of the network trainer was better by 0.1%. Also, for OGI Alphadigits, at a 35.3% WER, the performance of the network trainer was better by

approximately 2.7%. Finally, for Resource Management, at a 27.5% WER, the performance of the network trainer degraded slightly by about 1.8%. However, the degradation was shown to be insignificant using the NIST standard MAPSSWE test.

The work presented in this thesis also shows that the network trainer allows for multi-path model reestimation while simultaneously reducing the need for complicated systems and training recipes. This was done by using an optional multi-path silence model which is automatically inserted between words in the transcription. The network trainer alleviates the need for a forced-alignment stage in training by using a soft decision criteria during reestimation.

5.2. Future Work

The results presented in the previous chapter were obtained using single mixture monophone models. The context-dependent stage is a direct extension of the context-independent stage, which requires no changes in the training recipe. However, an efficient tree-based decoder is needed to decode the cross-word models. An efficient tree-based decoded is currently under development and recognition results using the cross-word models are planned.

In the previous chapter none of the corpora mentioned used multiple pronunciations. In order to fully test the power of the network training framework we will need to run it on larger corpora like Switchboard [49]. The ability of the network trainer to model multiple pronunciations, without modifying the training recipe, gives it a big edge over the traditional trainer. Again due to time constraints and issues related to the

efficiency of the system experiments using the Switchboard corpora could not be performed. The Switchboard are planned once these issues mentioned above are resolved.

REFERENCES

[1] S. Pinker, *The Language Instinct*, Harper Collins, New York City, New York, USA, 1994.

[2] N. Deshmukh, A. Ganapathiraju, J. Picone, “Hierarchical Search for Large Vocabulary Conversational Speech Recognition,” *IEEE Signal Processing Magazine*, vol. 1, no. 5, pp. 84-107, September 1999.

[3] K. F. Lee, “Context-dependent Phonetic Hidden Markov Models for Speaker-independent Continuous Speech Recognition,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-38, pp. 599-609, April 1990.

[4] J. W. Butzberger, et. al., “Spontaneous speech effects in large vocabulary speech recognition applications,” *Proceedings of DARPA Speech and Natural Language Workshop*, Morgan Kaufmann, pp. 339-343, 1992.

[5] E. Shriberg, “Disfluencies in SWITCHBOARD,” *Proceedings of the International Conference on Spoken Language Processing*, vol. Addendum, pp. 11-14, Philadelphia, Pennsylvania, October 1996.

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

[7] L. Rabiner, B. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Upper Saddle River, New Jersey, USA, 1993.

[8] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, John Wiley & Sons, New York City, New York, USA, 2001.

[9] S. B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-28, no. 4, pp. 357-366, August 1980.

[10] K. Fukunaga, Introduction to Statistical Pattern Recognition. Academic Press, New York, New York, USA, 1990.

[11] A. P. Dempster, N. M. Laird and D. B. Rubin, “Maximum Likelihood Estimation from Incomplete Data,” *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1-38, 1977.

[12] J. Picone, “Signal Modeling Techniques in Speech Recognition,” *Proceedings of the IEEE*, Vol. 81, No. 9, pp. 1215-1247, September 1993.

[13] J.R. Deller, J.G. Proakis and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan Publishing, New York, USA, 1993.

[14] S. Furui, “Cepstral Analysis Technique for Automatic Speaker Verification,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 2, pp. 254-272, 1981.

[15] J. G. Proakis, D. G. Manolakis, *Digital Signal Processing - Principles, Algorithms and Applications*, Prentice Hall, New Jersey, 1996.

[16] M. J. Hunt, "Spectral Signal Processing for ASR", *Proceedings of the 1999 Workshop on Automatic Speech Recognition and Understanding*, Keystone, Colorado, USA, December 1999.

[17] X. Huang, A. Acero, H. Hon, *Spoken Language Processing*, Prentice Hall, Upper Saddle River, New Jersey, USA, 2001.

[18] M. Jardino, “Multilingual Stochastic N-gram Class Language Models,” *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 161-163, May 1996.

[19] R. Kuhn, R. D. Mori, “A Cache Based Natural Language Model for Speech Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, pp. 570-583, 1992.

[20] H. Ney, U. Essen and R. Kneser, “On Structuring Probabilistic Dependencies in Stochastic Language Modeling,” *Computer Speech and Language*, Vol. 8, No. 1, pp. 1-38, 1994.

[21] E. Giachin, A. E. Rosenberg and C. Lee, “Word Juncture Modeling using Phonological Rules for HMM-based Continuous Speech Recognition,” submitted to *Computer, Speech and Language*, May 1991.

[22] E. Edie, “Automatic Modeling of Pronunciation Variants,” *Proceedings of Eurospeech*, pp. 451-454, Budapest, Hungary, September 1999.

[23] B. Byrne et. al., “WS97 Pronunciation Modeling for Conversational Speech Recognition Final Report,” P*roceedings of the 1997 LVCSR Summer Research Workshop*, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, Maryland, USA, December 1997.

[24] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineers*, Addison-Wesley, Reading, Massachusetts, USA, 1994.

[25] N. Deshmukh, A. Ganapathiraju, J. Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition," *IEEE Signal Processing Magazine*, vol. 1, no. 5, pp. 84-107, September 1999.

[26] K. F. Lee, *Large Vocabulary Speaker Independent Continuous Speech Recognition*, Ph. D. Thesis, Carnegie Mellon University, Pittsburgh, USA, 1988.

[27] T. M. Cover, J. A. Thomas, *Elements of Information Theory,* John Wiley & Sons, New York City, New York, USA, 1991.

[28] A. P. Dempster, N. M. Laird, D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society,* vol. 39, pp. 1-38, 1977.

[29] C. F. J. Wu, "On the Convergence Properties of the EM Algorithm," *The Annals of Statistics*, vol. 11, no. 1, pp. 95-103, 1983.

[30] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260-269, April 1967.

[31] K. F. Lee, "Context-dependent Phonetic Hidden Markov Models for Speaker-independent Continuous Speech Recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing,* vol. ASSP-38, pp. 599-609, April 1990.

[32] J. Picone, "Continuous Speech Recognition Using Hidden Markov Models," *IEEE Acoustics, Speech and Signal Processing Magazine,* vol. 7, no. 3, pp. 26-41, July 1990.

[33] B. H. Juang, L. R. Rabiner, "Issues in Using Hidden Markov Models for Speech Recognition," *Advances in Signal Processing*, pp. 509-554, New York, New York, 1992.

[34] R. Cole, J. Mariani, H. Uszkoreit, G. B. Varile, A. Zaenen, A. Zampolli, and V. Zue, eds., *Survey of the State of the Art in Human Language Technology,* Chapter 9, Cambridge University Press, Cambridge, Massachusetts, USA, March 1998.

[35] H. Murveit, P. Monaco, V. Digilakis, J. Butzburger, “Techniques to Achieve an accurate real-time, large vocabulary speech recognition system,” *Proceedings of the ARPA Human Language Technology Workshop*, pp. 368-373, Austin, Texas, USA, March 1995.

[36] P. Woodland, et. al., HTK Version 1.5: User, Reference and Programmer Manuals, Cambridge University Engineering Department and Entropic Research Laboratories Inc., 1995.

[37] C. Wooters and A. Stolcke, “Multiple-Pronunciation Lexical Modeling in a Speaker Independent Speech Understanding System,” submitted to the *International Conference of Spoken Language Processing*, Yokohama, Japan, September, 1994.

[38] P. Arabie, L. J. Hubert, G. D. Soete, *Clustering and Classification*, World Scientific, River Edge, New Jersey, 1998.

[39] A. K. Jain, R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[40] S. J. Young, P. C. Woodland, “State Clustering in HMM-based Continuous Speech Recognition,” *Computer Speech and Language*, vol. 8, no. 4, pp. 369-384, 1993.

[41] V. Digilakis, P. Monaco, H. Murveit, “Genomes: Generalized Mixture Tying in Continuous HMM-based Speech Recognition,” *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 4, pp. 281-289, 1996.

[42] M. Saraclar, H. Nock and S. Khudanpur, “Pronunciation Modeling by Sharing Gaussian Densities across Phonetic Models,” *Computer Speech and Language*, Vol. 14, No. 2, pp. 137-160, 2000.

[43] R. Leonard, “A Database for Speaker-Independent Digit Recognition,” *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, vol. 9, pp. 328-331, March 1984.

[44] R. Cole, et. al., “Alphadigit Corpus,” http://www.cse.ogi.edu/CSLU/corpora/alphadigit, Center for Spoken Language Understanding, Oregon Graduate Institute, Oregon, USA 1997.

[45] J. Hamaker, et. al., “A Proposal for a Standard Partitioning of the OGI AlphaDigit Corpus,” available at http://isip.msstate.edu/projects/lvcsr/recognition_task/alphadigits/data ogi_alphadigits/trans_eval.text.

[46] K. F. Lee, *Large Vocabulary Speaker Independent Continuous Speech Recognition*, Ph. D. Thesis, Carnegie Mellon University, Pittsburgh, USA, 1998.

[47] P. Price, W. M. Fisher, J. Bernstein and D. S. Pallett, "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 1, pp. 651-654, May 1998.

[48] L. Gillick, S. J. Cox, "Some Statistical Issues in the Comparison of Speech Recognition Algorithms," *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 532-535, May 1989.

[49] J. J. Godfrey, et. al., "SWITCHBOARD: Telephone Speech Corpus for Research and Development," Proceedings of International Conference on Acoustics, Speech, and Signal Processing, vol. 1, pp. 517-520, San Francisco, California, March 1992.