A Survey of Discriminative and Connectionist Methods for Speech Processing

Douglas Aberdeen

April 5, 2002

Abstract

Discriminative speech processing techniques attempt to compute the maximum a posterior probability of some speech event, such as a particular phoneme being spoken, given the observed data. Non-discriminative techniques compute the likelihood of the observed data assuming an event. Non-discriminative methods such as simple HMMs (hidden Markov models) achieved success despite their lack of discriminative modelling. This survey will look at enhancements to the HMM model which have improved their discrimination ability and hence their overall performance. This survey also reviews alternative discriminative methods, namely connectionist methods such as ANNs (artificial neural networks) . We will also draw comparisons between discriminative HMMs and connectionist models, showing that connectionist models can be viewed as a generalisation of discriminative HMMs.

1 Introduction

Discriminative methods for speech include using criteria such as MMI (maximum mutual information) and MCE (minimum classification error) during the training of HMMs (hidden Markov models). Connectionist methods bring to mind the use of ANNs (artificial neural networks). These methods are in fact closely related, sharing common solutions for tackling the complex problem of how to design MAP (maximum a posterior) classifiers for speech. For example, the MMI training criterion can be applied to both ANN training and to discriminative HMM training. Alternatively, ANNs can be trained to output the maximum a posterior probability that the input vector is an instance from each output class [30], which is inherently discriminative. Add to this the fact several authors have shown that it is possible to specify an ANN architecture exactly equivalent to discriminative HMM training [8, 34, 56], and discriminative techniques begin to look, in theory, synonymous to connectionist approaches.

ANNs were studied intensively for speech processing in the late 1980s and early 1990s before losing popularity in the face of excellent empirical results from purely HMM approaches. At the current time there seems to be little interest in pure ANN approaches however there is interest in hybrid ANN/HMM approaches [50, 17, 7].

In this survey we briefly present basic approaches to discriminative training techniques for HMMs and ANNs. We also compare these approaches, finding strong similarities between them. Then we look at methods which try to combine the best of both, hybrid HMM/ANN models

trained with discriminative techniques. The message of this survey is that traditional HMM approaches are flawed and that discriminative approaches, particularly those using hybrid approaches may offer significant advantages.

Familiarity is assumed with the basics of both ANNs and HMMs. Many introductory texts can be founds on these topics including [11, 39, 28] for HMMs and [32] for ANNs.

2 The Speech Problem

Speech processing can be thought of as the problem of choosing

$$m^* = \arg\max_m P(m|O_u), \tag{1}$$

where $O_u = \{O_u(1), \ldots, O_u(T_u)\}$ is a time sequence of speech frames associated with utterance u and the m's are the models or classes that categorise the data. The correct model given the data is m^*. The model may represent a phone, word, speaker or some other such unit depending on the processing problem at hand. This is the MAP (maximum a posterior) criteria for selecting the correct model. Using Baye's rule we can transform (1) into

$$m^* = \arg\max_m \frac{P(O_u|m)P(m)}{P(O_u)}. \tag{2}$$

Since all data is assumed equally likely $P(O_u)$ is constant and if we further assume that all models (or classes) are equally likely, then we end up with the ML (maximum likelihood) criteria

$$m^* = \arg\max_m P(O_u|m), \tag{3}$$

which can be interpreted as saying *assuming a model, what is the probability that the given data belongs to it?* By itself this is not a discriminative method since the models do not compete to classify the data. Instead each model is trained individually to maximise the probability that it generates the subset of training data that it was trained on. The ML criteria is the one used in standard Baum-Welch training of HMMs [40]. Once the ML models have been trained we could use (2) to compute the discriminative probability $P(m|O_u)$. However, the individually trained models produce only estimates of $P(O_u|m)$, trained on limited amounts of data, and only to a local maximum. Combined with errors in estimating $P(m)$ — possibly from a different source than the spoken training data — it is preferable to perform training which directly estimates $P(m|O_u)$, or at least trains the likelihood models to not only maximise $P(O_u|m^*)$ but at the same time minimise $P(O_u|m)$ $\forall m \neq m^*$.

Since speech is a signal rather than a static pattern (3) should really be expressed as probabilities of sequences of observations and models

$$\begin{aligned}\mathbf{m}^* &= \arg\max_{\mathbf{m}} P(O_u|\mathbf{m}) \\ &= \arg\max_{\mathbf{m}} P(O_u(1), \ldots, O_u(t)|m_1, m_2, \ldots, m_t).\end{aligned}$$

This expression finds most probable sequence of models up to time t having seen all the data up to that time and assuming there is a model associated with each time step. This is the computation performed by

HMMs where the model at each time step is a state of the HMM. HMMs are based on the assumption that the Markov property holds for speech, which can be phrased as *the most probable model (or state) depends only on the current observation and the previous model*

$$m_t^* = \arg\max_{m_t} P(x_t, |m_t, m_{t-1}),$$

and the probability of the sequence is the sum over all possible model trajectories, where the probability of a trajectory is the product of the probability of each step given only the current model and the previous model

$$P(O_u|\mathbf{m}) = \sum_{\forall m_1,\ldots,m_{T_u}} \prod_{t=1}^{T_u} P(x_t|m_t, m_{t-1}). \tag{4}$$

So now observations are assumed independent and the next model is assumed dependent only on the previous model. It is these assumptions that make HMM training tractable since (4) can be performed with a dynamic-programming like approach [7]. The simplifications of (3) and (4) admit good empirical results while allow real-time processing. Unfortunately the simplifications deliberately make untrue assumptions about speech [40]. This is not just true of HMMs since ANN approaches typically make similar assumptions [6]. However, in ANNs we have the ability to relax these assumptions more readily than we do in HMMs. For example, to incorporate dependence on n previous models instead of just 1, we can add $O(n)$ inputs to the network. To accomplish this with an HMM with M models we need to create $O(M^n)$ individual models. This kind of increase in complexity is seen when HMMs move from modelling context independent phones (61 for the TIMIT corpus) to triphones where around 5000 models are used even after the unlikely or unhelpful triphones are removed [24, 20].

3 Discriminative methods for ANN and HMM training

In this section we briefly describe two popular methods for performing discriminative training which can be applied to both ANNs and HMMs. We roughly follow the notation and structure of [43] which presents both methods in a consistent framework.

3.1 Maximum Mutual Information

The basic idea of MMI estimation is to maximise the extent to which knowing the data helps us to know which model is correct. An alternative and simpler view is to look at MMI as maximizing the ratio of the correct model to all other models, weighted by the class probabilities.

$$\frac{P(O_u|m^*)}{\sum_{i=1}^{M} P(m_i)P(O_u|m_i)}.$$

MMI estimation methods are discussed and applied in too many papers to enumerate however some of the better descriptions are found in [43, 40, 54]. MMI techniques can be applied to the language modelling phase of

speech systems as well as the low level signal models [33]. In information theory mutual information is defined as

$$I(X,Y) = H(X) - H(X|Y), \text{ where} \tag{5}$$
$$H(X) = -\sum_{x \in X} P(x) \log P(x)$$

which is the entropy of the discrete random variable X. Another interpretation of $H(X)$ is as the expected amount of information in X where the information carried by event $X = x$ is measured as $I(x) = -\log_b P(x)$. If $b = 2$ the information is measured in bits. $I(X,Y)$ tells us to what extent knowing X helps us to know Y. To apply this to speech with the intent of training model m^*, let $Y = m^*$, and take the information expectation over the data in the set $X = \{O_u : m^* = \arg\max_m P(m|O_u)\}$, i.e. all the training data available model m^*. Maximising (5) maximises how much X tells us about the model m^*.

$$\begin{aligned}
I(X,m^*) =& H(X) - H(X|m^*) \\
=& -\sum_{O_u \in X} P(O_u) \log P(O_u) + \sum_{O_u \in X} P(O_u) \log P(O_u|m^*) \\
=& \sum_{O_u \in X} P(O_u) \left[\log P(O_u|m^*) - \log \sum_{i=1}^{M} P(m_i) P(O_u|m_i) \right] \\
=& \sum_{O_u \in X} P(O_u) \log \left(\frac{P(O_u|m^*)}{\sum_{i=1}^{M} P(m_i) P(O_u|m_i)} \right). \qquad (6)
\end{aligned}$$

However the speech technology community generally assumes that all observations are equally probable and define I with respect to a single observation O_u, in which case (6) simplifies to

$$I(O_u, m^*) = \log \left(\frac{P(O_u|m^*)}{\sum_{i=1}^{M} P(m_i) P(O_u|m_i)} \right), \tag{7}$$

and from this form comes the intuition that MMI maximises the ratio of the correct model likelihood to the likelihood of *all* models. Also note that $I(O_u, m^*) + \log(P(m^*))$ gives us $P(m^*|O_u)$, the more desirable MAP criteria. We can also relate the MMI criteria to the idea of minimising cross entropy. Equation (6) can be re-written as

$$-I(O^*, m^*) = \sum_{O_u \in O^*} P(O_u) \log \left(\frac{P(O_u)}{P(O_u|m^*)} \right),$$

which is the calculation for discrete cross entropy [40]. Thus maximising mutual information can be re-cast as minimising cross entropy, which can be thought of as minimising the difference between the distribution of the data, and the data given the model [9].

3.1.1 Gradient Descent for MMI

Suppose we have some parameterised approximator (or possibly an approximator for each model) which computes $P(O_u|m, \theta)$, where θ represent the parameters of the system. By computing the gradient of $-I(O_u, m)$ with respect to θ we can train our approximator to perform speech processing according the MMI criteria. To reduce clutter we label the numerator

of (7) as L_*, the likelihood of the correct model and L_a as the combined likelihood of all models, giving

$$L_* = P(O_u|m^*), \qquad L_a = \sum_{i=1}^{M} P(m_i)P(O_u|m_i).$$

Rewriting (7) to be suitable for minimisation we have

$$\begin{aligned} -I(O_u, m^*) &= \log \frac{L_a}{L_*} \\ I(O_u, m^*) &= \log L_a - \log L_* \\ \frac{-\partial I(O_u, m^*)}{\partial \theta} &= \frac{1}{L_a}\frac{\partial L_a}{\partial \theta} - \frac{1}{L_*}\frac{\partial L_*}{\partial \theta}. \end{aligned}$$

Provided we can compute $\partial L_*/\partial\theta$ and $\partial L_a/\partial\theta$ gradient descent can be used to optimize parameters θ.

How do we apply this to a real system? One approach is to use knowledge of the model priors $P(m_i)$ (or assume they are uniform), and instead of approximating $P(O_u|m_i)$ we approximate the posterior probability $P(m_i|O_u)$ using one large network where each output represents a model. Interpreting network outputs as probabilities is explained in Section 4.1. This approach is used in Alphanets [10, 9, 34] and in several RNNs (Recurrent Neural Networks) [45, 55, 12]. Examples of approximating $P(O_u|m)$ with ANNs are rare since we might expect a single network to share information more efficiently, requiring fewer parameters and consequently require less training data. One example (which uses the MCE described below rather than MMI) is [25], described further in Section 4.3.1.

The difficulty with speech for ANNs is the time varying nature of the signal. The question is how to represent O_u to the network so that it outputs a sequence of model probabilities. ANNs usually assume a static pattern, however speech consists of a possibly continuous stream of data broken down into frames of around 10 ms, each with tens to hundreds of features [23]. The key difference between the various connectionist and hybrid inspired approaches is how they deal with the time varying nature of speech. The natural way HMMs handle time varying signals is a strong reason to prefer them over connectionist methods.

3.1.2 MMI for HMMs

In general it is possible to use gradient ascent for training HMMs though care must be taken to maintain stochastic constraints. For example, the sum of transition probabilities out of a state must sum to one. This can be achieved by mapping parameters in $\mathbb{R}$ to probabilities [34, 8]. In [19] it is pointed out that this method may introduce extra local maxima, which is undesirable since gradient methods only guarantee convergence to one of these local maxima. Alternatively, Lagrange multipliers can be used to perform gradient ascent subject the stochastic constraints [40, 19].

The gradients of the discriminative cost functions described here can be incorporated into an HMM update gradient, or HMM training can be run as normal and then gradient descent on the discriminative objective function can be performed as corrective training [36, 19]. Alternatively

[43, 37, 16] discuss methods which extend the Baum-Welch updates to rational objective functions, which are applicable to the objective functions outlined here.

All of these methods require the derivative of the cost function with respect to the HMM parameters. The following equations [54] give the gradient of the MMI criterion with respect to the HMM parameters for the state transitions $i \rightarrow j$ of model m, denoted a_{ij}^m and the discrete observation probabilities for symbol $O_u(t)$ in state j, denoted $b_{jO_u(t)}^m$.

$$\begin{aligned} -\frac{\partial I(O_u, m^*)}{\partial a_{ij}^m} &= \left(\frac{1}{L_a} - \frac{\delta_{m,m^*}}{L_*}\right) \sum_{t=1}^{T} \alpha_{t-1}(i) b_{jO_u(t)}^m \beta_t(j) \\ -\frac{\partial I(O_u, m^*)}{\partial b_{jO_u(t))}^m} &= \left(\frac{1}{L_a} - \frac{\delta_{m,m^*}}{L_*}\right) \frac{\alpha_t(j)\beta_t(j)}{b_{jO_u(t)}^m}, \end{aligned} \tag{8}$$

Where $\alpha_t(i)$ is the forward HMM probability of being in state i at time t and $\beta_t(i)$ is the corresponding backwards probability. The sum of these gradients across all the training data will result in the gradient of the negative of (6). The extension of (8) to the case of continuous densities represented by a single Gaussian is given in [10].

3.2 Minimum Classification Error

Minimum Classification Error seeks to minimise exactly what we care about, the empirical error rate. It is introduced and described in a general way in [21] which also compares this criterion to standard error measures such as the mean squared error. A similar measure called Minimum Empirical Error was introduced in [2]. The basic idea is to construct a distance measure between the probability of the correct choice and the probability of all other choices

$$d_*(O_u) = P(O_u|m^*) - \left[\frac{1}{M-1} \sum_{m_i \neq m^*} P(O_u|m_i)^\eta\right]^{\frac{1}{\eta}}. \tag{9}$$

The parameter η can be thought of as adjusting the distance metric used. If $\eta = 1$ we have an L_1 norm and we are simply summing the probabilities of incorrect models. As $\eta \rightarrow \infty$ only the largest incorrect probability has any effect. We then use $-d_*(O_u)$ in a sigmoid to construct a smooth cost function l which can be minimised in order to maximise (9)

$$l(d_*(O_u)) = \frac{1}{1 + \exp(\gamma d_*(O_u))}. \tag{10}$$

By summing (10) over all the training data for each model we achieve an empirical estimate of the probability of misclassification. It is interesting to compare the MCE to MMI. If we set $\eta = 1$ and take the log of both terms in (9) then we have

$$d_*(O_u) = \log\left(\frac{P(O_u|m^*)}{\sum_{m_i \neq m^*} \frac{1}{M-1} P(O_u|m_i)}\right).$$

Which differs from (7) only in whether the correct model m^* is included in the summation and the assumption of uniform priors $P(m_i)$. MCE is also very similar to the idea of distance normalization discussed in [14].

3.2.1 Gradient Descent for MCE

In practice it seems more common to use the log form of (9) [25, 43], which results in the following gradient for $l(d_*(O_u))$ with respect to an arbitrary set of parameters θ

$$\frac{\partial l(d_*(O_u))}{\partial \theta} = \sum_{i=1}^{M} l(d_*(O_u))(1 - l(d_*(O_u)))G_i(O_u)\frac{\partial P(O_u|m_i,\theta)}{\partial \theta} \quad (11)$$

$$G_i(O_u) = \begin{cases} \frac{-1}{P(O_u|m_i)} & \text{if } m_i = m^* \\ \frac{P(O_u|m_i,\theta)}{\sum_{j=1}^{M} P(O_u|m_j,\theta)} & \text{otherwise,} \end{cases}$$

Summing (11) over all the data for each model results in a gradient which minimises the probability of misclassification.

Note that we are subject to the same questions about how to approximate $P(O_u|m,\theta)$ as we were in Section 3.1.1, and also subject to the same solutions. In [21] they take the approach of training a single large network to approximate all the probabilities (see Section 4.1). In [25] a single network is trained for each $P(O_u|m,\theta)$ (see Section 4.3.1).

3.2.2 MCE for HMMs

In [43, 35] a gradient descent version of MCE estimation is used. Denoting the state of HMM m occupied at time t as q_t^m, the gradient for the state-specific observation densities is

$$\frac{\partial l(d_*(O_u))}{\partial P(O_u|j,m)} = l(d_*(O_u))(1 - l(d_*(O_u)))G_i(O_u) \sum_{t:q_t^m=j} \frac{1}{P(O_u(t)|j,m)}, \quad (12)$$

which sums up the gradient contributions for all observations associated with state q_j^m. In the case of discrete symbols we have $P(O_u(t)|j,m) = b^m_{jO_u(t)}$. Equation (12) implies that the optimal state sequence is known, which can be determined using the Viterbi algorithm.

3.3 Results Comparison

Where possible we have provided comparative experimental results for the methods described in this survey, mostly on the TIMIT database [15]. However, due to factors such as varying definitions of accuracy and the varying levels of problem difficulty, the results should not be compared across different sections. A German speech database was used in [43] to compare MMI and MCE training of HMMs. Using the standard MLE criterion they achieved a 59.5% accuracy. Applying MMI improved this to 62.0% and MCE achieved 64.8%. They also reported that MMI training was less stable than MCE, requiring smaller step sizes.

On the TIMIT database with 39 phones [12] demonstrates an RNN (recurrent neural network) system trained with MMI with a frame by frame accuracy of 75.1%. This is compared with the CMU Sphinx [51] HMM system which achieved 73.8%.

On a Cantonese digit test set MCE improved results from 82.9% to 90.0%. This system used a small RNN for each digit. The same system applied to English digits resulted in recognition improving from 92.3% to 93.5%. The disparity of improvement arises from the inherent confusability of Cantonese digits which allows discriminative approaches to work well.

4 Neural Network Speech Processing

Why should we bother with ANNs if HMMs provide a good way to represent speech signals? There are some good reasons to prefer ANNs or Hybrid models, which include:

- ANNs can model arbitrary non-linear transformations of input parameters, including the ability to model arbitrary probability distributions [7]. The most flexible pure HMMs typically assume probability distributions made up of mixtures of Gaussians with a covariance matrix which is 0 except along the diagonal.
- If trained properly, ANNs can directly estimate the discriminative MAP $P(m|O_u)$ criterion (see Section 4.1).
- ANN systems can be 2–5 times faster than traditional techniques for equivalent performance [49].
- A single ANN can be trained to do the same job as multiple HMMs, decreasing the overall number of parameters to be trained, and improving the use of training data [49].
- ANNs can relax the Markov assumption by considering multiple frames of data (past and future) at once [27, 6]. It is difficult to do this with HMMs since it is necessary to minimise the dimensionality of observations and the number of states to allow estimation of the parameters with minimal data. Time derivatives are often incorporated into HMM features to provide context, however the derivatives contain less information than the complete frames.
- ANNs can consider categorical inputs, encoding psycho-acoustic features [38] and features from many sources at once, such as visual cues [7].
- ANNs can model arbitrary state durations, unlike HMMs in which durations follow an exponential model. This is important for normalising the likelihood contributions from short consonants against long vowels and other speech warping phenomenon. HMMs can be modified to model arbitrary durations, however the computational expense seems to outweigh the benefits [40]. Durations can also be modelled in a post-processing phase, but these methods appear ad-hoc, requiring extra weighting terms to be optimized.
- In practice, it appears necessary to cleverly initialize the observation densities used in HMMs to achieve good performance [40, 7]. This is not the case with ANNs.

The disadvantages of ANNs include:

- The lack of a principled way to convert a sequence of observations into an optimal sequence speech units, i.e. there is a need to include some form of search for the globally optimal sequence of units given the local estimates of matches from an ANN. This is the function usually achieved by the Viterbi search in HMMs. Hybrid techniques are a way to avoid this problem.
- ANN systems using gradient methods are 10–20 times slower to train than HMMs using Baum-Welch training since they are restricted to small steps in parameter space [49]. Conjugate gradient [13] and line searches can speed up gradient ascent training. Approximate gradient ascent algorithms such as RPROP can also be used [49].

- Speech ANNs have roughly 10 thousand to 2 million parameters [49], requiring a large amount of data and cross-validation to avoid over-fitting.
- To date, state of the art HMMs with sophisticated tied and interpolated distributions and thousands of context dependent models, have roughly 25% better error rate than the best ANN/Hybrid systems when sufficient training data is available [7]. However the ANN/hybrid systems tend to be much simpler.

The rest of this section looks at different ways to contrive ANNs capable of handling speech data.

4.1 Big, Dumb Neural Networks

Ignoring for the moment the problem of time dependence in speech it is possible to view an ANN as performing a series of static probability estimation tasks. The input to the network is a frame of speech plus future and past frames of speech to provide context. Each output gives an estimate to the probability of a particular model given the input.

Consider neural network outputs $y_1, \ldots, y_M \in \mathbb{R}$; how do we interpret these outputs as probabilities. More specifically, how would we construct a network to compute the posterior probabilities $P(m_i|O_u)$? A standard method for doing this is to use a softmax distribution at the output [45, 31]. Assume that the network is learning to estimate the MAP probability $P(m_i|O_u)$ then for each possible model $m_1, \ldots m_M$ we define

$$P(m_i|O_u) \triangleq \frac{\exp(y_i)}{\sum_{j=1}^{M} \exp(y_j)}.$$

Given an arbitrary cost function J, such as $-I(X, m^*)$, we compute

$$\frac{\partial J}{\partial \theta_j} = \frac{\partial J}{\partial P(m_i|O_u)} \sum_{k=1}^{M} \frac{\partial P(m_i|O_u)}{y_k} \frac{\partial y_k}{\partial \theta_j}. \tag{13}$$

The gradient of the softmax distribution with respect to the network outputs is

$$\frac{\partial P(m_i|O_u)}{\partial y_k} = P(m_i|O_u)(\delta_{i,k} - P(m_k|O_u)) \tag{14}$$

which can be interpreted as driving the difference between the desired probability $\delta_{i,k}$ and the actual output probability to zero. Once the gradient of the cost function with respect to the outputs is known it's straight forward to use back propagation to compute $\frac{\partial y_k}{\partial \theta}$.

For example, consider the following simple cost function for the observation $O_u(t)$

$$J = -logP(m_t^*|O_u(t), \theta) \tag{15}$$
$$\frac{\partial J}{\partial P(m_t^*|O_u(t), \theta)} = -\frac{1}{P(m_t^*|O_u(t), \theta)}.$$

This is the equation for the Normalized-Likelihood cost function [44]. It simply measures the log probability of utterance O_u assuming we know (or can estimate) the correct model m_t^*. Minimising this quantity will maximise the posterior probability of the correct model given the observation. Minimising the sum of (15) over time will maximise the log likelihood of

the correct sequence of models. Substituting (15) and (14) into (13), we obtain

$$\frac{\partial J}{\partial \theta_j} = -\sum_{k=1}^{M} (\delta_{m_t^*, m_k} - P(m_k|O_u(t), \theta)) \frac{\partial y_k}{\partial \theta}. \tag{16}$$

In the simple case where y_k represents the kth output of an ANN with linear output nodes and w_{hk} is the weight from hidden node h to output k, (16) simplifies to

$$\frac{\partial J}{\partial w_{hk}} = -(\delta_{m_t^*, m_k} - P(m_k|O_u(t), \theta)) w_{hk}.$$

4.1.1 Alternative Cost Functions for MAP Estimation

Provided there is sufficient training data and the ANN is sufficiently complex to represent $P(m_i|O_u(t))$, minimising (15) will result in an ANN that estimates $P(m_i|O_u(t))$. This is proved in [44], which also proves that the same is true of the mean square error cost function and the cross entropy function

$$J = -\sum_{i=1}^{M} d_i \log P(m_i|O_u(t), \theta) + (1 - d_i) \log(1 - P(m_i|O_u(t), \theta))$$

$$\frac{\partial J}{\partial \theta_j} = -\sum_{i=1}^{M} \frac{d_i - P(m_i|O_u(t), \theta)}{P(m_i|O_u(t))(1 - P(m_i|O_u(t)))} \frac{\partial P(m_i|O_u(t))}{\theta_j},$$

where $d_i = 1$ if $m_i = m_t^*$ and 0 otherwise. Cross entropy has been popular for speech recognition applications, for example [55, 12].

Experimental comparisons of these three cost functions showed that they all produce similar results if enough training data is available. The cross entropy cost and normalised likelihood weight cost converged faster than MSE, and the normalised likelihood resulted in marginally better estimation accuracy in regions of low probability [44].

These cost function differ from those covered in Section 3 because they estimate posterior probabilities of models rather than likelihoods $P(O_u(t)|m)$. Despite this, these training techniques can be used to estimate scaled likelihoods simply by dividing the outputs by the prior probability of the models. This is discussed further in Section 5.1,

4.1.2 Implementation Issues

In the case where unlabelled data is available, it is sufficient to train the system using as much labelled data as is available, then use the resultant classifier to label the unlabelled data, using this new larger labelled set to train a new classifier. This process is repeated, with each classifier bootstrapping off the labelling of the previous classifier, until no improvement is gained [7].

In practice, networks which classify the 61 phone TIMIT database have several hundred inputs, including frames for context, 500–4000 hidden units, and 61 outputs, requiring in the order of 10^6 parameters [31]. Training such networks provides interesting challenges [1]. Once such a network has been trained some form of search is needed to compute the most likely phone sequence.

Figure 1: A Time-Delay Neural Network with 7 inputs and 2 frames of memory, 5 hidden nodes with 3 frames of memory and 3 outputs with 4 frames of memory.

4.2 Time-Delay Neural Networks

TDNNs (Time-Delay Neural Networks) were one of the earliest attempts to modify ANNs to cope with sequences of inputs. The output of each node of a TDNN is the same as a standard ANN, the weighted sum of its inputs, but integrated over T_l frames. Each layer l may integrate over a different time period. Thus each node in layer l must have a local shift register to store the weighted sum of the last T_l inputs. If the input to a node at the current time is $\mathbf{x}_t$, then the output is

$$y_t = \sum_{s=0}^{T_l - 1} c_s f(\mathbf{x}_{t-s}),$$

where c_s is an optional weighting term for each past frame. Figure 1 illustrates this idea. TDNNs can be thought of as integrating evidence for or against a class over a finite period of time. They are trained using a modified form of error back propagation. Good results were obtained for classifying plosive consonants using TDNNs compared to standard ANNs [53]. They have also been used to approximately determine phone labels to use as discrete HMM symbols in [29]. This system recognised Dutch digits, discriminating between 21 phonemes. Results improved from 90% to 93% over a HMM with 200 discrete symbols. A drawback of TDNNs is the fixed amount of memory for each node. This is somewhat rectified in [26] where TDNNs are extended to automatically adapt the value of T_l. TDNNs are further reviewed in [27, 17, 9].

4.3 Recurrent Neural Networks

RNNs avoid the main problem of TDNNs by allowing all previous inputs to effect the current output. Any traditional network architecture can be classed as an RNNs if it involves feedback from the output back into the inputs and/or hidden units [17]. In a typical application the inputs are augmented with a real-valued state vector output by the network at the previous time step. This implies that the outputs are augmented to provide the next state given the current input $O_u(t)$ and state $\mathbf{x}_t$. RNNs draw theoretical justification from their similarity to the feedback methods used in linear state-based control systems [12].

The most common training method is *Back-Propagation Through Time*, which unfolds the network N times and propagates the errors at the outputs $\mathbf{y}_t$ and state $\mathbf{x}_t$ back through each time step (Figure 2). The initial

Figure 2: An RNN unfolded to N times for training.

state can be set to an arbitrary fixed value. The state error at time N is zero, since the final state has no effect on the classification being made. Unfortunately this method limits the amount of past context the network can be trained to consider to N frames. More complex methods exist avoid this limitation [45, 47].

In [45] an RNN was used to classify the 61 phone TIMIT database. The network was trained using the cross-entropy criterion as described in Section 4.1. This allowed the 61 outputs to be interpreted as $P(m_i|O_u(1), O_u(2), \ldots, O_u(t))$ and fed into a Viterbi decoder. Thus this application is actually a hybrid approach. The state dimension was 176, with 47,400 parameters, trained with Back-Propagation Through Time using the fast RPROP of error descent algorithm [49]. Results show that 72.8% of phones were correctly identified by the RNN compared to 74.4% for a mono-phone HMM system trained with the MMI criterion. At the time the best results were from an HMM based system [52] with 76.7%. These slightly lower results for the RNN suggest why HMMs have dominated the speech community's efforts. RNNs are further reviewed in [27, 9, 12, 17, 50].

4.3.1 Modeling Likelihoods with RNNs

If we wish to approximate the prior probabilities $P(O_u|m_i)$ we could use a separate small ANN for each model and either have a single output giving the probability of O_u. This is the approach in [25] where one RNN is trained for each of 11 Cantonese digits. The number of outputs is approximately the number of identifiable acoustic units, or states, that exist in the digit. At each time step the current state in each network is $q_j = \arg\max_y y_i$. A network has a high probability of being the correct model for the observations if the state index j increases monotonically to the final state along the duration of the utterance.

4.3.2 Bi-Directional Neural Networks

As they have been formulated above RNNs cannot take into account future frames of data in computing $P(m|O_u)$. It is natural to expect that we need future context as well as past context to optimally identify a unit of speech. A simple way to provide future context is to train the network to delay its decision on frame t until frame $t+c$ where c is the number of future frames to consider. An alternative is to extend RNNs to allow all frames, past and present, to be considered. This architecture is called the the Bi-Directional RNN [48, 49, 50]. BRNNs have two sets of state vectors, one for the forward time direction and one for the reverse time direction. At time t separate hidden layers compute the next forward and backward state vectors, while the output layer estimates $P(m_i|O_u(1), O_u(2), \ldots, O_u(T_u))$ based on $O_u(t)$, the forward state computed at $t-1$ and the backward state computed at $t+1$. For real-time recognition some window of speech

data needs to be considered if utterances are longer than a few tens of frames.

A subset of the TIMIT 61 phone database was trained using BRNNs in [48] and compared to RNNs using delayed decisions of up to 4 frames. The best RNN actually had 0 delay with an accuracy of 51.2% using 8 state variables and 1518 parameters. The BRNN had an accuracy of 55.3% using 8 state variables in each direction and a total of 2182 parameters. The poor performance of the delayed RNNs in this experiment is not consistent with other results and may be due to the use of only 30 sentences as training data and the extra difficulty in training for delayed decisions.

4.4 Representing HMMs as ANNs

In [56] the similarity between HMMs and RNNs is emphasized. We do not usually attempt to interpret the value of an RNN state vector, but we might conceive of a network trained to produce a state vector equivalent to the forward state probability used in the Baum-Welch training procedure for HMMs, i.e. $\mathbf{x}_t = \alpha_t$. The key idea is to note that the recursive forward probability computation for a single HMMs can be written as

$$\begin{aligned} \alpha_t(j) &= \sum_i \alpha_{t-1}(i) a_{ij} b_{jO_u(t)} \\ &= \sum_i \alpha_{t-1}(i) w_{ij}(t), \end{aligned} \tag{17}$$

which looks like the computation performed in a linear node of an ANN except that the weights are dependent on the current observation.

4.5 Alphanets

The concepts of the previous section are extended by [8, 10], resulting in Alphanets, and the work of [34]. Both reach the conclusion that HMMs can be cast exactly as an RNN if we allow multiplication and division units as well as the standard summation units. The first two factors in (17) can be computed with a feed forward pass where the inputs are α_t and the weights are the stochastic matrix elements a_{ij}. The observation probabilities $b_{jO(t)}$ can be estimated with another feed forward network where the inputs are $O_u(t)$ and there is an output for the probability given each state. The outputs are multiplied for each state, and then normalised to give α_{t+1}. The output of the network is the log sum of the α_{t+1}s. A possible network is shown in Figure 3.

Applications based on the Alphanets methodology appear to assume one HMM state per acoustic model, using one network to represent one large HMM. They use discriminative forms of HMM training to update the parameters such as those discussed in Section 3.1.2. Alphanets are not a new technique for speech processing, rather the importance of Alphanets is in providing a new view of existing HMM techniques and unifying the idea of discriminative and connectionist approaches to speech [9].

4.6 Other uses of Neural Networks for Speech

In this section we have focused on ANN methods which attempt to determine the probability of a model (such as a phone) given the observations O_u. However ANNs can be used in speech processing in many other ways. Some of these include:

Figure 3: An RNN implementation of an HMM.

- *Phone recognition* – Alternative methods of performing phone recognition include the use of Kanerva models, Classification & Regression Trees and other ANNs with novel processing units [12].
- *Vector Quantization* – Learning Vector Quantizers based on Self-Organizing Feature maps and other ANN approaches can be used to process observations to generate symbols for discrete HMMs [22, 3].
- *Pre-processing* – ANNs can perform arbitrary non-linear transformations of the input. This can perform tasks such as removing noise, or adapting to a new speaker [18, 42, 57].
- *Hierarchical Mixtures of Experts* – Various expert classifiers including those discussed already can be combined through the use of a hierarchy of gating networks [49, 41] trained with the EM algorithm [46].
- *Predictive Networks* – ANNs can be used to predict extra features. For example, they can be trained as autoregressive models given previous observations and the current state [7].
- *Language Modelling* – ANNs can be used to estimate the probabilities of sequences of phones, used for re-scoring N-best lists of phone sequences [7].

5 Hybrid Speech Processing

In this section we present two of the most common hybrid approaches. The first is a broad approach, allowing many theoretically justified variations, and is the subject of active research. The second describes a method for globally optimising both the ANN and HMM components of hybrid systems.

5.1 Estimating Observation Densities with ANNs

Each state i of an HMM is associated with a probability distribution over the observations $P(O_u(t)|i)$. Section 4 noted that an advantage of ANNs over HMMs is their ability to model an arbitrary distribution, non-linear in the inputs. A large body of work including [7, 5, 6, 31] is devoted to

Figure 4: Using an ANN to generate HMM observation likelihoods.

this idea. Essentially the techniques of Section 4.1 are applied to estimate observation likelihoods $p(O_u(t)|i)$, and those likelihoods are used in the HMM procedures in place of $b_{iO_u(t)}$, such as in (17).

However, the ANN methods of Section 4.1 estimate posterior $P(m|O_u(t))$ rather than the likelihoods used by the HMM search. It is easy to convert from a posterior to a *scaled likelihood* by assuming $P(O_u(t))$ is constant and using Baye's rule

$$p(O_u(t)|m) \propto \frac{P(m|O_u(t))}{P(m)}.$$

This simply amounts to dividing the network output probabilities by the model priors estimated from the training data. This quantity can be used in (17) to compute the model likelihood. In this context a model m and an HMM state i are synomous, since the ANN estimates a phone probability and the HMM consists of one phone per state. Figure 4 illustrates the process. One phone per state limits the applicability of the algorithm though in principle there is no reason why ANNs could not be used to estimate probabilities for multi-state phones. Once the ANN has been trained, an HMM training procedure can be used to estimate the state transition probabilities a_{ij}. On an 152 speaker subset of the TIMIT database with 64 phones, the technique described above achieved a frame by frame accuracy of 54.8% with 351 inputs and 1024 hidden nodes [6]. Each frame had 39 inputs and ± 4 frames of context were provided for a total of 351 NN inputs. Single Gaussian per phone density estimates achieved 43.3%.

5.1.1 MAP Estimation with ANN/HMM Hybrids

It may seem counter-intuitive to estimate posterior probabilities, just to turn them into less informative scaled likelihoods. Let us return again to the idea of estimating state probabilities for HMMs rather than model probabilities. By estimating state *transition* probabilities $P(i_t|i_{t-1}, O_u(t))$ instead of just occupancy probabilities $P(i_t|O_u(t))$ we can avoid computing the scaled likelihood. The probability of the model is the product of

$P(j|i, O_u(t))$ for each state i_t in the sequence, summed across all possible state sequences. Taking into account that HMM m may model extra language information not modelled by the ANN transition probabilities, we have [7]

$$P(m|O_u, \theta) = P(m) \sum_{\forall i_1, \ldots, i_T} \left[\prod_{t=1}^{T} P(i_t|i_{t-1}, O_u(t), \theta) \frac{P(i_t|i_{t-1}, m)}{P(i_t|i_{t-1})} \right]$$

and the Viterbi approximation replaces the sum with a maximization at each step. $P(i_t|i_{t-1}, m)$ is the state transition probability according to the model. $P(i_t|i_{t-1})$ is the state transition probability estimated from the training data by counting transitions. The dependency on θ emphasizes the dependency on the ANN parameters. If the HMM model simply models the transitions in the data we have $P(i_t|i_{t-1}, m) = P(i_t|i_{t-1})$ and the last factor vanishes. Otherwise, $P(i_t|i_{t-1}, m)$ allows us to encode useful knowledge that may not be evident in the training data, for example, the task may involve a restricted set of words, altering the distribution of phones.

5.2 Global Optimization of ANN/HMM Hybrids

An alternative ANN/HMM approach taken by [4] views the ANN as mapping a high dimensionality set of frame data into a small set of continuous observations to be input to an HMM that estimates observation probabilities using a mixture of Gaussians. Used in this way the ANNs are performing regression rather than classification. Multiple networks can be used to pre-process the data in different ways, each concentrating on various hard to distinguish features. For example, one network is trained to produce observations particularly useful for difficult plosive classification, while another network may produce broadly useful features.

The clever aspect of this structure is that the gradients of the HMM parameters, computed as in Section 3.1.2, can be propagated back into the ANNs, simultaneously maximising the discriminative powers of the HMMs and the ANNs. This requires computing $\partial b_{jO_u(t)}/\partial y_k$, the derivative of the observation probability at time t for state j with respect to the kth output of the combined networks. Having derived this quantity back-propagation can be used to derive the gradients of the network weights. The same globally trained ANNs are used to provide observations for all states of all HMMs.

This method was evaluated using the TIMIT database with 7214 triphone models. Observation features were calculated with 3 networks: a recurrent network for broad features using 12 inputs, a recurrent network for plosive features using 74 inputs and a linear network to combine the results of the first two. The result was 8 continuous observations for the HMMs. The networks use a total of 23,578 weights. The first two networks were pre-trained to perform recognition tasks. Each HMM had 14 states and 3 distributions, tied to the transitions between the states. Results are given in terms of the segmentation accuracy [1] rather than the frame by frame match. Global optimization boosted accuracy from 81% to 86%. A hybrid based on the ideas in Section 5.1 achieved 74%.

[1] Accuracy is defined as 100% - % deletions - % substitutions - % insertions.

6 Summary

This survey has described the popular MMI and MCE criteria for discriminative speech processing systems. We showed how either criteria can be applied to HMMs, ANNs, or indeed some arbitrary parameterised probability estimator. Connectionist approaches to probability estimation were reviewed, including static multi-layer perceptrons, TDNNs, RNNs and Bi-Directional RNNs. The close link between HMM approaches and connectionist approaches was also explored by showing how ANN architectures such as Alphanets perform the same calculation as the forward probability calculation used in HMMs. The advantages of ANNs for speech processing compared to HMMs were also listed, noting that while ANNs have many theoretical advantages over HMMs, taking advantage of them is difficult, requiring the training of very large ANNs. Finally we described the hybrid techniques which use ANNs to estimate phone likelihoods for HMM time alignment, and techniques which allow the global optimization of one form of HMM/ANN hybrid.

References

[1] D. Aberdeen, J. Baxter, and R. Edwards. 92¢ /MFlop/s, Ultra-Large-Scale Neural-Network training on a PIII cluster. In *Proceedings of Super Computing 2000*, Dallas, TX., November 2000. SC2000 CDROM.

[2] A.Ljolje, Y.Ephraim, and L.R.Rabiner. Estimation of hidden markov model parameters by minimizing empirical error rate. In *Proceedings ICASSP 1990*, volume 2, pages 709–712. IEEE Signal Processing Society, IEEE, April 1990.

[3] T. R. Anderson. Auditory models with Kohonen SOFM and LVQ for speaker independent phoneme recognition. In *IEEE Internation Conference on Neural Networks*, volume VII, pages 4466–4469, Orlando, Florida, June 1994. IEEE.

[4] Y. Bengio, R. D. Mori, G. Flammia, and R. Kompe. Global optimization of a neural network-hidden markov model hybrid. *IEEE Transactions on Neural Networks*, 3(2):252–259, March 1992.

[5] H. Bourlard, Y. Konig, and N. Morgan. Remap : Recursive estimation and maximization of a posteriori probabilities in connectionist speech recognition. In *Proc. EUROSPEECH '95*, Madrid, September 1995.

[6] H. A. Bourlard and N. Morgan. *Connectionist Speech Recognition* A Hybrid Approach. Kluwer Academic Publishers, 1 edition, 1994.

[7] H. A. Bourlard and N. Morgan. *Hybrid HMM/ANN Systems for Speech Recognition: Ovierview and New Research Directions* , volume 1387 of Lec of *Lecture Notes in Artificial Intelligence* , pages 389–417. Springer-Verlag, 1998.

[8] J. S. Bridle. Alpha-nets: A recurrent 'neural' network architecture with a hidden markov model interpretation. *Speech Communication*, 9:83–92, February 1990.

[9] J. S. Bridle. *Spech Recognition and Understanding. Recent Advances* , chapter Neural Networks or Hidden Markov Models for Automatic Speech Recognition: Is there a Choice?, pages 225–236. Number F75

in NATO ASI. Springer-Verlag Berlin, Royal Signals and Radar Est., 1992.

[10] J. S. Bridle and L. Dodd. An alphanet approach to optimising input transformations for continuous speech recognitio. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, number 5.7, pages 277–280, Speech Research Unit, Royal Signals and Radar Establishment, 1991. IEEE.

[11] D.B.Paul. Speech recognition using hidden markov models. *The Lincoln Laboratory Journal*, 1:41–62, 1990.

[12] F. Fallside. *Speech Recognition and Understanding. Recent Advances*, chapter Neural Networks for Continuous Speech Recognition, pages 237–257. Number F75 in NATO ASI. Berlin: Springer-Verlag, Cambridge University, 1992.

[13] T. L. Fine. *Feedforward Neural Network Methodology*. Springer, New York, 1999.

[14] S. Furui. An overview of speaker recognition technology. In *ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, pages 1–9. ESCA, ESCA, 1994.

[15] J. S. Garofolo et al. TIMIT acoustic-phonetic continuous speech corpus, 1993. `http://www.ldc.upenn.edu/Catalog/LDC93S1.html`.

[16] P. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo. A generalization of the baum algorithm to rational objective functions. In *ICASSP-89*, 1989.

[17] J.-P. Haton. Connectionist and hybrid models for automatic speech recognition. In K.M.Ponting, editor, *Proceedings of the NATO Advanced Study Institute on Computational Models of Speech Pattern Processing*, number F169 in NATO ASI, LORIA/Université Henri Poincaré, France, July 1997. Springer-Verlag Berlin.

[18] X. D. Huang, K. F. Lee, and A. Waibel. Connectionist speaker normalization and its applications to speech recognition. In *IEEE Workshop for Neural Networks for Signal Processing*, New Jersey, October 1991. IEEE.

[19] Q. Huo and C. Chan. The gradient projection method for the training of hidden Markov models. *Speech Communication*, 13:307–313, May 1993.

[20] M.-Y. Hwang and X. Huang. Shared distribution hidden Markov models for speech recognition. *IEEE Trans Speech and Audio Processing*, 1(4):414–420, 1993.

[21] B.-H. Juang and S. Katagiri. Discrimintative learning for minimum error classification. *IEEE Transactions on Signal Pocessing*, 40(12):3043 – 3054, December 1992.

[22] M. Kurimo. Training mixture density HMMs with SOM and LVQ. *Computer Speech and Language*, 11(4):321–343, October 1997. `citeseer.nj.nec.com/kurimo97training.html`.

[23] K.-F. Lee. *Large-Vocabulary Speaker-Independant Continuous Speech Recognition: The SPHINX System*. PhD thesis, Computer Science Department, Carnegie-Mellon University, April 1988.

[24] K.-F. Lee. *Automatic Speech Recognition, The Development of the SPHINX System*. Kluwer international series in engineering and computer science. SECS 62. Kluwer Academic Publishers, 1989.

[25] T. Lee, P. Ching, and L. Chan. An RNN based speech recognition system with discriminative training. In *Proceedings of EuroSpeech-95*, volume 3, pages 1667–70, 1995.

[26] D.-T. Lin. *The Adaptive Time-Delay Neural Network: Characterization and Applications to Pattern Recognition, Prediction and Signal Processing*. PhD thesis, Institute for Systems Research, University of Maryland, 1994. `http://www.isr.umd.edu/TechReports/ISR/1994/PhD_94-12/PhD_94-12.phtml`.

[27] R. P. Lippmann. Review of neural networks for speech recognition. *Neural Computation*, 1(1):1–38, 1989.

[28] L.R.Rabiner and B.H.Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, pages 4–16, January 1986.

[29] W. Ma and D. V. Compernolle. TDNN labeling for a HMM recognizer. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, number 8.3, pages 421–424, Albuquerque, New Mexico, April 1990. IEEE.

[30] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[31] N. Morgan. Big dumb neural nets: A working brute force approach to speech recognition. In *IEEE Internation Conference on Neural Networks*, volume VII, pages 4462–4465, Orlando, Florida, June 1994. IEEE.

[32] B. Müller, J. Reinhardt, and M. Strickland. *Neural networks : an introduction*. Physics of neural networks. Springer-Verlag, New York, 2nd edition, 1995.

[33] H. Ney. The use of the maximum likelihood criterion in language modelling. In K.M.Ponting, editor, *Proceedings of the NATO Advanced Study Institute on Computational Models of Speech Pattern Processing*, number F169 in NATO ASI, RWTH Aachen – University of Technology, July 1997. Springer-Verlag Berlin.

[34] L. T. Niles and H. F. Silverman. Combining hidden Markov models and neural network classifiers. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, number 8.2, pages 417–420, Albuquerque, New Mexico, April 1990.

[35] A. Nogueiras-Rodriíguez, J. B. M. no, and E. Monte. An adaptive gradient-search based algorithm for discriminative training of HMMs. In *Proceedings of ICLSP'98*. Universitat Politècnica de Catalunya, Causal Productions, 1998.

[36] Y. Normandin. *Hidden markov models, maximum mutual information estimation and the speech recognition problem*. PhD thesis, McGill University, 1991.

[37] Y. Normandin and R. Cardin. *Developments in High-Performance Connected Digit Recognition*, volume F.75 of *NATO ASI Series*, pages 89–94. Springer–Verlag, Berlin, 1992.

[38] L. C. Pols. Psycho-acoustics and speech perception. In K.M.Ponting, editor, *Proceedings of the NATO Advanced Study Institute on Computational Models of Speech Pattern Processing*, number F169 in NATO ASI, IFOTT, University of Amsterdam, July 1997. Springer-Verlag Berlin.

[39] A. B. Poritz. Hidden markov models: A guided tour. In *ICASSP '88*, pages 7–13. Morgan Kaufmann, 1988.

[40] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech processing. In *Proceedings of the IEEE*, volume 77. IEEE, February 1989.

[41] S. Ran and J. B. Millar. Phoneme discrimination using hierarchically organised connectionist networks. In *Proceedings of Second Australian Conference of Neural Networks*, pages 279–282, Sydney, Australia, 4–6 February 1991.

[42] S. Ran and J. B. Millar. Two schemes of phonetic feature extraction using artificial neural networks. In *Proceedings of Eurospeech'93*, pages 1607–1610, Berlin, Germany, 1993.

[43] W. Reichl and G. Ruske. Discriminative training for continuous speech recognition. In *Europ. Conf. on Speech Communication and Technology*, volume 1, pages 537–540, September 1995.

[44] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate bayesian *a posteriori* probabilities. *Neural Computation*, 3(4):461–483, Winter 1991.

[45] T. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(3):298–305, 1994.

[46] S. Russel. The em algorithm. Technical Report Machine learning, CS281, University of Alberta, Spring 1998. `http://www.cs.ualberta.ca/~jcheng/AI/Summer98.htm`.

[47] J. H. Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 2(4):234–242, 1992.

[48] M. Schuster. Bi-directional recurrent neural networks for speech recognition. Technical report, 1996.

[49] M. Schuster. *Encyclopedia of Electrical and Electronics Engineering*, chapter Neural networks for speech processing. John Wiley & Sons, June 1998.

[50] M. Schuster. *On supervised learning from sequential data with applications for speech recognition*. PhD thesis, Graduate School of Information Science, Nara Institute of Science and Technology, February 1999.

[51] K. Seymore, S. Chen, S.-J. Doh, M. E. E. Gouvea, B. Raj, M. Ravishankar, R. Rosenfeld, M. Siegler, R. S. ane, and E. Thayer. The 1997 CMU sphinx-3 english broadcast news transcription system. In *Proceedings of the 1998 DARPA Speech Recognition Workshop*. DARPA, 1998.

[52] University of Cambridge. *Hidden Markov Model Toolkit V3*, 2001. `http://htk.eng.cam.ac.uk/`.

[53] A. Waibel. Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, 1(1):39–46, 1989.

[54] N. D. Warakagoda. A hybrid ANN-HMM ASR system with NN based adaptive preprocessing. Master's thesis, Institutt for Teleteknikk, Transmisjonsteknikk, May 1996. `http://jedlik.phy.bme.hu/~gerjanos/HMM/hoved.html`.

[55] W. Wei and S. Vuuren. Improved neural network training of inter-word context units for connected digit recognition. In *ICASSP'98*, pages 497–500, Seattle, WA, May 1998. IEEE.

[56] S. J. Young. Competitive training in hidden Markov models. In *Proc. International Conference on Acoustics, Speech and Signal Processing* , number 13.1, pages 681–684, Albuquerque, New Mexico, April 1990. IEEE.

[57] D. Zhang and J. B. Millar. Digit-specific feature extraction for multi-speaker isolated digit recognition using neural networks. In *Proceedings of 5th Australian International Conference on Speech Science and Technology*, pages 522–527, Brisbane, Australia, 6–8 December 1994.