IDIAP RESEARCH REPORT

IDIAP
Martigny - Valais - Suisse

IDIAP HMM/HMM2 System: Theoretical Basis and Software Specifications

Shajith Ikbal [a] Hervé Bourlard [a,b]
Samy Bengio [a] Katrin Weber [a]

IDIAP–RR 01-27

October 1, 2001

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 − 27 − 721 77 11
fax +41 − 27 − 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

a Institut Dalle Molle d'Intelligence Artificielle Perceptive (IDIAP), Martigny, Switzerland
b Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland

IDIAP Research Report 01-27

IDIAP HMM/HMM2 System: Theoretical Basis and Software Specifications

Shajith Ikbal Hervé Bourlard Samy Bengio Katrin Weber

October 1, 2001

Abstract. State-of-the-art Automatic Speech Recognition (ASR) systems make extensive use of Hidden Markov Models (HMMs), characterized by flexible statistical modeling, powerful optimization (training) techniques and efficient recognition algorithms. When allowed by the software implementation, their flexibility can also be fully exploited in research, by testing various topologies, acoustic units, parameterization schemes, etc. Unfortunately, these HMM systems still suffer from an excessive sensitivity to the variability generally observed in real acoustic environments, including speaker, channel and noise characteristics. In an attempt to tackle this problem, IDIAP recently introduced a new form of HMM, referred to as HMM2, exhibiting numerous potential advantages, which could result in improved robustness of current speech recognition systems. HMM2 can be described as a mixture of HMMs where the HMM emission probabilities (usually estimated by Gaussian Mixtures or a neural network) are themselves estimated by state-specific HMMs working along the acoustic features. Among other properties, it is believed that such HMM2 approach could better model the time/frequency speech flow, including better modeling of the feature correlation.
After a brief reminder of the HMM theory, this report first introduces the theoretical basis of HMM2, including their parameterization schemes and the estimation of their parameters through a generalized form of the Expectation-Maximization (EM) training algorithm. It is also the goal of the present report to describe the functionalities and specifications of a new software able to handle, in a flexible way, different forms of HMM and HMM2 topologies and training schemes.

Acknowledgements: This work was supported by the Swiss National Foundation Project (2100-061325.00/1) titled **HMM2: A New Framework for Robust and Adaptive Speech Recognition**.

1 Introduction

The aim of automatic speech recognition (ASR) systems is to extract the linguistic information in the speech signal. Systems which are shown to be the most successful in achieving this task are the ones based on statistical techniques. If $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{T-1}\}$ represents set of feature vectors extracted from the speech signal, statistical techniques formulate the speech recognition as a Maximum A Posteriori (MAP) problem [1] as follows.

$$W^* = \underset{W \in \mathcal{L}}{\text{argmax}} \; P(W|\mathbf{X}) \tag{1}$$

The most likely word sequence W^* from the set of all possible word sequences $\mathcal{L}$, given $\mathbf{X}$, is chosen as the recognition result. The MAP formulation of speech recognition is hard to deal with directly. It is usually converted into a problem based on likelihood estimation using Bayes's equation, as follows.

$$W^* = \underset{W \in \mathcal{L}}{\text{argmax}} \; P(\mathbf{X}, W) \;=\; \underset{W \in \mathcal{L}}{\text{argmax}} \; P(\mathbf{X}|W)P(W) \tag{2}$$

The word sequence W^* which has the highest joint likelihood, along with $\mathbf{X}$, is chosen as the recognition result. In the above equation, $P(\mathbf{X}|W)$, which is the conditional probability of $\mathbf{X}$ given W, is usually referred to as *acoustic model*, and $P(W)$, the prior probability of word sequence W, is referred to as *language model*. In practice, both the acoustic and language models are assumed to fit in some parametric form, say $P_\Theta(.)$ and $P_\Gamma(.)$, with parameter sets Θ and Γ respectively. Then $P_\Theta(\mathbf{X}|W,\Theta)$ and $P_\Gamma(W|\Gamma)$ are used in (2) as estimates for $P(\mathbf{X}|W)$ and $P(W)$. The values of model parameters Θ and Γ are estimated from a training database containing large collection of utterances, with known transcriptions. For this purpose, training database is assumed to be consistent with the test utterances, so that the acoustic and language models learnt would represent the respective distributions of the test utterances. If $\mathcal{E}$ represents the set of all the training utterances along with corresponding transcriptions, ideally the parameters can be estimated according to:

$$\{\Theta^*, \Gamma^*\} = \underset{\Theta \text{ and } \Gamma}{\text{argmax}} \left[\prod_{[\mathbf{X},W] \in \mathcal{E}} P_\Theta(\mathbf{X}|W,\Theta) P_\Gamma(W|\Gamma) \right] \tag{3}$$

But practical constraints do not allow the joint estimation of Θ and Γ. They are usually estimated independently of each other from different training sets, say $\mathcal{E}_a$ and $\mathcal{E}_l$ respectively, yielding:

$$\Theta^* = \underset{\Theta}{\text{argmax}} \left[\prod_{\mathbf{X} \in \mathcal{E}_a} P_\Theta(\mathbf{X}|W,\Theta) \right] \tag{4}$$

$$\Gamma^* = \underset{\Gamma}{\text{argmax}} \left[\prod_{W \in \mathcal{E}_l} P_\Gamma(W|\Gamma) \right] \tag{5}$$

Equation (4) is referred to as Maximum Likelihood (ML) training. State-of-the-art ASR systems use HMMs [1, 2] for acoustic modeling and bigram/trigram probabilities for language modeling. As this report concentrates mainly on the various alternatives for acoustic models, language models will not be discussed.

HMMs used for acoustic modeling $P(\mathbf{X}|W,\Theta)$[1] suffer from an excessive sensitivity to the variabilities in real acoustic environments, including speaker, channel, and noise characteristics. In an attempt to overcome this problem, IDIAP recently introduced a new form of HMM, referred to as HMM2 [3, 4], exhibiting numerous potential advantages that could result in improved robustness. HMM2

[1] From now onwards, $P(\mathbf{X}|W,\Theta)$ will be used in place of $P_\Theta(\mathbf{X}|W,\Theta)$, for convenience.

can be described as a mixture of HMMs where the emission density modeling (which is usually done by Gaussian mixtures or Neural networks) is done by a set of state-specific HMMs working along the acoustic features.

The following sections give a brief theoretical basis of both HMM and HMM2, from the system implementation view point, and describe the functionalities and specifications of a new software able to handle different forms of HMM and HMM2 topologies and training schemes. As will be described in detail latter, using the software, HMM/HMM2 based acoustic models of any topology can be trained. The training of HMM can be done in two different modes, and the training of HMM2 can be done in four different modes.

2 Hidden Markov Model

Hidden Markov Model (HMM) [1, 2] is a statistical tool that can be used for acoustic modeling of the speech signal, $P(\mathbf{X}|W)$. It provides a modular architecture, where $P(\mathbf{X}|W)$ for all the word sequences can be arrived at from independent acoustic models developed for basic linguistic units. The basic linguistic units could be words or subwords like phonemes. When words are used as the basic units, word sequence models are arrived at by simply concatenating the models for words. Whereas, when subwords are used as the basic linguistic units, lexicon is used along with the subword models to first arrive at the word models followed by the word sequence models. Along with an independently developed language model, the word sequence models can be used to compute the combined likelihood $P(\mathbf{X}, W)$, and hence to perform speech recognition.

For acoustic modeling HMM approximates the nonstationary character of speech signal by a piecewise stationarity, assuming the speech generation process as a stochastic finite state automaton. The acoustic model for each basic linguistic unit consists of: a set of stochastic states, with associated emission density functions and fixed transition topology. The speech generation process, while generating sound corresponding to a particular linguistic unit, is assumed to have performed sequential transitions between the states corresponding to that linguistic unit. The sequence of transitions performed is governed by the topology and is hidden to the outside world. The statistics of the speech signal generated while staying at a particular state is characterized by the emission density function assigned to the state. The emission density function basically models the statistics of the acoustic vectors extracted from the speech signal.

Fig. 1(a) and (b) shows two different examples of typical acoustic models for basic linguistic units. In (a) there are 3 states, namely 0_0, 0_1, and 0_2, and in (b) there are two states, 1_0 and 1_1. States I and F corresponds to the nonemiting initial and final states, respectively. Transition topology is specified by the arrows between the states with associated probabilities. The probability of performing transition from state i to state j of linguistic unit u is denoted by symbol a^u_{ij}. The emission density of state i in the linguistic unit u is denoted by symbol $p^{u_i}(.)$. The speech generating process, while generating the sound corresponding to, say the linguistic unit of Fig. 1(a), it is assumed to have gone through the states 0_0, 0_1, and 0_2, as shown in the figure. Entering the state 0_0 at the starting, stays there for a few time steps depending upon the probability a^0_{00}, and emit acoustic vectors every time step based on the density function $p^{0_0}(.)$. Then goes to state 0_1 with probability a^0_{01}, stays there with probability a^0_{11} and emit acoustic vectors based on $p^{0_1}(.)$, and so on.

The likelihood of a vector sequence of length T belonging to the linguistic unit of Fig. 1(a) is the probability of the model for that linguistic unit generating the vector sequence. Each vector would have been generated by one of the states in the model. If the state sequence corresponding to the vector sequence $\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_t, ..., \mathbf{x}_{T-1}\}$ is denoted by $\{q_0, q_1, ..., q_t, ..., q_{T-1}\}$, each q_t can take different values depending upon the topology of the model. For example, if the vector sequence is of length 4, $\{q_0, q_1, q_2, q_3\}$ can take values $\{0_0, 0_0, 0_0, 0_1\}$ or $\{0_0, 0_0, 0_1, 0_1\}$ or $\{0_0, 0_0, 0_1, 0_2\}$ or $\{0_0, 0_1, 0_1, 0_1\}$ or $\{0_0, 0_1, 0_1, 0_2\}$ or $\{0_0, 0_1, 0_2, 0_2\}$. The likelihood of the state sequence $\{0_0, 0_0, 0_1, 0_2\}$ generating the

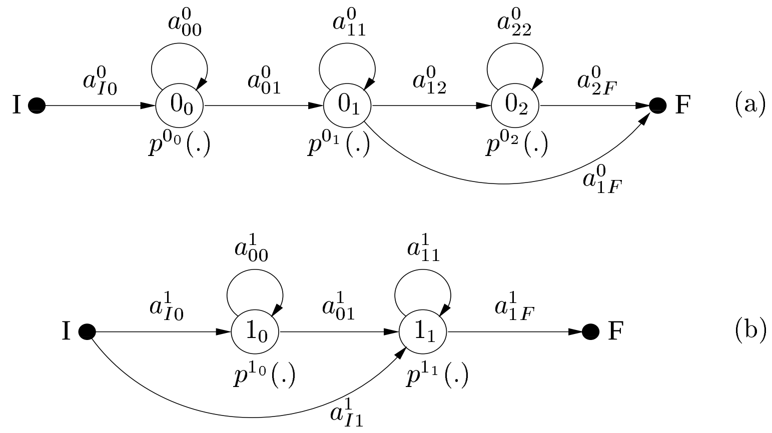

Figure 1: Two typical examples for HMM based acoustic model for phonemes.

vector sequence is $a^0_{I0}p^{0_0}(\mathbf{x}_0)a^0_{00}p^{0_0}(\mathbf{x}_1)a^0_{01}p^{0_1}(\mathbf{x}_2)a^0_{12}p^{0_2}(\mathbf{x}_3)a^0_{2F}$. The over all likelihood of the model generating the vector sequence is the summation of the likelihoods of all possible state sequences generating the vector sequence.

The parameter set Θ characterizing the HMM based acoustic model include,

1. The transition probabilities of all possible transitions between the states, and
2. The parameters of density functions governing the emission probabilities from each state.

If there are U basic linguistic units, denoted by $\{0, 1, 2, ..., u, ..., U-1\}$, with M^u number of states in the model for u^{th} linguistic unit, denoted by $\{u_0, u_1, ..., u_i, ..., u_{M^u-1}\}$, the entire set of parameter is given by:

Initial probabilities of all the states in all the unit models: a^u_{Ii}, $0 \leq i \leq M^u - 1$, $0 \leq u \leq U - 1$.

Transition probabilities of all possible transitions between the states in all the unit models: a^u_{ij}, $0 \leq i \leq M^u - 1$, $0 \leq j \leq M^u - 1$, $0 \leq u \leq U - 1$.

Final probabilities of all the states in all the unit models: a^u_{iF}, $0 \leq i \leq M^u - 1$, $0 \leq u \leq U - 1$.

Emission density functions for all the states in all the unit models: $p^{u_i}(.)$, $0 \leq i \leq M^u - 1$, $0 \leq u \leq U - 1$.

Generally, Gaussian Mixture Models (GMMs) or Artificial Neural Networks (ANNs) are used to model the emission density. In the current version of the software discussed here only GMMs are used. GMM is a weighted mixture of several gaussians. It is characterized by the weighting factors, mean vectors, and covariance matrices of all the constituent gaussians. The expression for density function $p(\mathbf{x})$ of GMM is given by,

$$p(\mathbf{x}) = \sum_{k=0}^{K-1} c_k G_k(\mathbf{x}) \tag{6}$$

where K denotes the number of gaussians in the GMM, and c_k denotes the weighting factor for k^{th} gaussian, $G_k(.)$. If μ_k and Σ_k denote respectively the mean vector and covariance matrix of the k^{th}

gaussian, and if D denote the feature vector dimension, expression for $G_k(\mathbf{x})$ is given by,

$$G_k(\mathbf{x}) = \frac{1}{2\pi^{\frac{D}{2}}|\Sigma_k|^{\frac{1}{2}}} \mathrm{e}^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T \Sigma_{\mathbf{k}}{}^{-1}(\mathbf{x}-\mu_k)} \tag{7}$$

Hence the parameters corresponding to the emission density function $p^{u_i}(.)$ are,

$$[c_k^{u_i} \; \mu_k^{u_i} \; \Sigma_k^{u_i}] \; , \; 0 \le k \le K^{u_i} - 1$$

As explained earlier, if the subwords are used as basic linguistic units, lexicon is used to first arrive at the acoustic models for words and then those word models are used to form word sequence models. To illustrate this, let the models given in Fig. 1(a) and (b) represent acoustic models for phonemes ‘a’ and ‘b’, respectively. Let there be two words in the lexicon, namely ‘ab’ and ‘bab’, whose pronunciation models are as shown in Fig. 2(a) and (b), respectively.

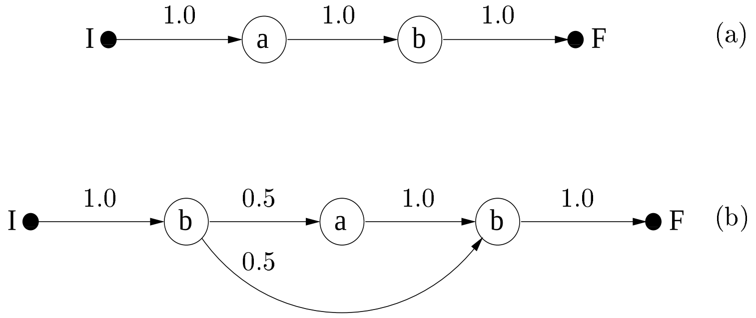

Figure 2: Pronunciation models for two word in the lexicon (a) ‘ab’, and (b) ‘bab’.

Fig. 3 illustrates how the word model for the word ‘ab’ is obtained by merging the phoneme level acoustic models and the lexicon. In the word model there are 5 states $\{w_0, w_1, w_2, w_3, w_4\}$, which correspond to the states $\{0_0, 0_1, 0_2, 1_0, 1_1\}$ of the Fig. 1.

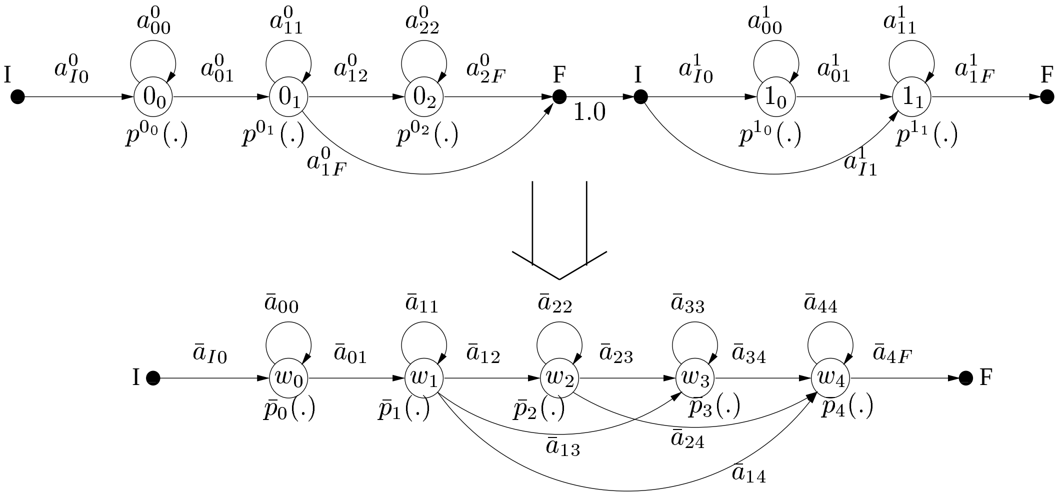

Figure 3: Formation of word level acoustic model for the word ‘ab’ from the the corresponding pronunciation model and the constituent phoneme models.

In the above figure, $\bar{a}_{I0} = a^0_{I0}$, $\bar{a}_{00} = a^0_{00}$, $\bar{a}_{01} = a^0_{01}$, $\bar{a}_{11} = a^0_{00}$, $\bar{a}_{12} = a^0_{12}$, $\bar{a}_{13} = a^0_{1F}a^1_{I0}$, $\bar{a}_{14} = a^0_{1F}a^1_{I1}$, $\bar{a}_{22} = a^0_{22}$, $\bar{a}_{23} = a^0_{2F}a^1_{I0}$, $\bar{a}_{24} = a^0_{2F}a^1_{I1}$, $\bar{a}_{33} = a^1_{00}$, $\bar{a}_{34} = a^1_{01}$, $\bar{a}_{44} = a^1_{11}$, $\bar{a}_{4F} = a^1_{1F}$, $\bar{p}_0(.) = p^{0_0}(.)$, $\bar{p}_1(.) = p^{0_1}(.)$, $\bar{p}_2(.) = p^{0_2}(.)$, $\bar{p}_3(.) = p^{1_0}(.)$, $\bar{p}_4(.) = p^{1_1}(.)$.

Fig. 4 shows word level acoustic model for the word 'bab'. It has 7 states $\{w_0, w_1, w_2, w_3, w_4, w_5, w_6\}$, which correspond to the states $\{1_0, 1_1, 0_0, 0_1, 0_2, 1_0, 1_1\}$ of the Fig. 1.

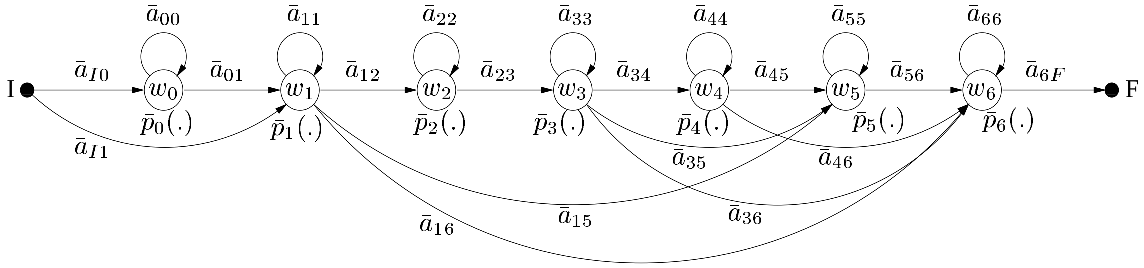

Figure 4: Acoustic model for the word 'bab'.

In the above figure, $\bar{a}_{I0} = a^1_{I0}$, $\bar{a}_{I1} = a^1_{I1}$, $\bar{a}_{00} = a^1_{00}$, $\bar{a}_{01} = a^1_{01}$, $\bar{a}_{11} = a^1_{11}$, $\bar{a}_{12} = a^1_{1F}a^0_{I0}/2$, $\bar{a}_{15} = a^1_{1F}a^1_{I0}/2$, $\bar{a}_{16} = a^1_{1F}a^1_{I1}/2$, $\bar{a}_{22} = a^0_{00}$, $\bar{a}_{23} = a^0_{01}$, $\bar{a}_{33} = a^0_{11}$, $\bar{a}_{34} = a^0_{12}$, $\bar{a}_{35} = a^0_{1F}a^1_{I0}$, $\bar{a}_{36} = a^0_{1F}a^1_{I1}$, $\bar{a}_{44} = a^0_{22}$, $\bar{a}_{45} = a^0_{2F}a^1_{I0}$, $\bar{a}_{46} = a^0_{2F}a^1_{I1}$, $\bar{a}_{55} = a^1_{00}$, $\bar{a}_{56} = a^1_{01}$, $\bar{a}_{66} = a^1_{11}$, $\bar{a}_{6F} = a^1_{1F}$, $\bar{p}_0(.) = p^{1_0}(.)$, $\bar{p}_1(.) = p^{1_1}(.)$, $\bar{p}_2(.) = p^{0_0}(.)$, $\bar{p}_3(.) = p^{0_1}(.)$, $\bar{p}_4(.) = p^{0_2}(.)$, $\bar{p}_5(.) = p^{1_0}(.)$, $\bar{p}_6(.) = p^{1_1}(.)$.

The acoustic models for word sequences are obtained by concatenating the constituent word models. Fig. 5 shows acoustic model for word sequence 'a ab'. In the model, states $\{w_0.w_1, w_2, w_3, w_4, w_5, w_6, w_7\}$ correspond to the states $\{0_0, 0_1, 0_2, 0_0, 0_1, 0_2, 1_0, 1_1\}$ of the Fig. 1.

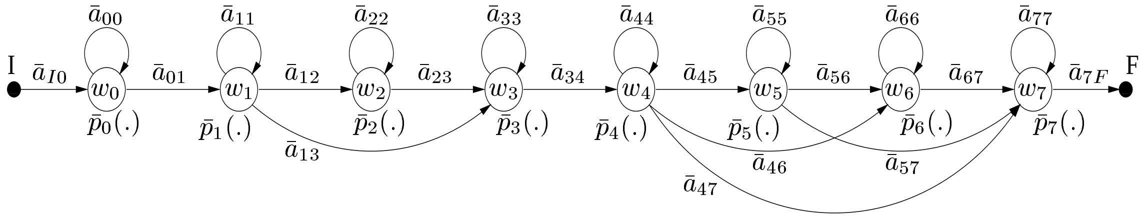

Figure 5: HMM based acoustic model for the word sequence 'a ab'.

In the above figure, $\bar{a}_{I0} = a^0_{I0}$, $\bar{a}_{00} = a^0_{00}$, $\bar{a}_{01} = a^0_{01}$, $\bar{a}_{11} = a^0_{11}$, $\bar{a}_{12} = a^0_{12}$, $\bar{a}_{13} = a^0_{1F}a^0_{I0}$, $\bar{a}_{22} = a^0_{22}$, $\bar{a}_{23} = a^0_{2F}a^0_{I0}$, $\bar{a}_{33} = a^0_{00}$, $\bar{a}_{34} = a^0_{01}$, $\bar{a}_{44} = a^0_{11}$, $\bar{a}_{45} = a^0_{12}$, $\bar{a}_{46} = a^0_{1F}a^1_{I0}$, $\bar{a}_{47} = a^0_{1F}a^1_{I1}$, $\bar{a}_{55} = a^0_{22}$, $\bar{a}_{56} = a^0_{2F}a^1_{I0}$, $\bar{a}_{57} = a^0_{2F}a^1_{I1}$, $\bar{a}_{66} = a^1_{00}$, $\bar{a}_{67} = a^1_{01}$, $\bar{a}_{77} = a^1_{11}$, $\bar{a}_{7F} = a^1_{1F}$, $\bar{p}_0(.) = p^{0_0}(.)$, $\bar{p}_1(.) = p^{0_1}(.)$, $\bar{p}_2(.) = p^{0_2}(.)$, $\bar{p}_3(.) = p^{0_0}(.)$, $\bar{p}_4(.) = p^{0_1}(.)$, $\bar{p}_5(.) = p^{0_2}(.)$, $\bar{p}_6(.) = p^{1_0}(.)$, $\bar{p}_7(.) = p^{1_1}(.)$.

Given a vector sequence $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{T-1}\}$, $P(\mathbf{X}|W, \Theta)$ for a particular word sequence W is the likelihood of the acoustic model for that word sequence generating the vector sequence. If $Q = \{q_0, q_1, ..., q_{T-1}\}$ represents the state sequence that has generated the vector sequence, then Q can take several values depending upon the topology of the model. Likelihood of one such state sequence generating the vector sequence is given by,

$$P(\mathbf{X}, Q|W, \Theta) = p(q_0|I)p(\mathbf{x}_0|q_0) \prod_{t=1}^{T-1} p(q_t|q_{t-1})p(\mathbf{x}_t|q_t) \tag{8}$$

where $p(q_t|q_{t-1})$ is the probability of performing transition from state q_{t-1} at time $t-1$ to the state q_t at time t. $p(q_0|I)$ is the probability of performing transition from nonemiting initial state I to the state q_0 at time 0. $p(\mathbf{x}_t|q_t)$ is the probability of state q_t generating the vector $\mathbf{x}_t$, at time t. If $\mathcal{C}$ represents the set of all possible state sequences for the model, the likelihood of the overall model generating the vector sequence $\mathbf{X}$ is,

$$P(\mathbf{X}|W,\Theta) = \sum_{Q\in\mathcal{C}} P(\mathbf{X},Q|W,\Theta) \tag{9}$$

The above equation gives the exact formula for the computation of the likelihood $P(\mathbf{X}|W,\Theta)$. Sometimes, it is also approximated as the likelihood of best state sequence generating the vector sequence as follows,

$$P(\mathbf{X}|W,\Theta) = \max_{Q\in\mathcal{C}} P(\mathbf{X},Q|W,\Theta) \tag{10}$$

This is called Viterbi approximation.

2.1 Training

The values of the parameter set Θ are found from a training database containing large collection of utterances, with known transcription. The training procedure could be supervised or unsupervised. In the software developed a popular unsupervised learning procedure called Expectation-Maximization (EM) algorithm [5, 6, 7] is used. EM algorithm is an iterative procedure where, in each iteration, new values of the model parameters are found from the old values, so that the overall likelihood of the training data is increased. From the transcriptions available we can form the word sequence models for all the training utterances, using the parameter set. If there are E utterances in the training data, denoted by $\mathcal{E} = \{\mathbf{X}^0, \mathbf{X}^1, ..., \mathbf{X}^{E-1}\}$, with corresponding acoustic models denoted by $\{W^0, W^1, ..., W^{E-1}\}$, the new values for the parameter set Θ^{new} are found from the old values Θ^{old} such that,

$$\prod_{e=0}^{E-1} P(\mathbf{X}^e|W^e,\Theta^{\text{new}}) \geq \prod_{e=0}^{E-1} P(\mathbf{X}^e|W^e,\Theta^{\text{old}}) \tag{11}$$

If the likelihood to be maximized is the complete likelihood as can be computed using the Eqn. 9, the training is said to be done in *Baum-Welch* (BW) mode. On the other hand, if the likelihood maximized is the likelihood of the best path, as given in Eqn. 10, training is said to be done in *Viterbi* mode.

Before explaining the EM algorithm, let us define few notations as follows.

T^e, length of the acoustic vector sequence $\mathbf{X}^e$.

N^e, number of states in model W^e, with states denoted by $\{w_0^e, w_1^e, ..., w_i^e, ..., w_{N^e-1}^e\}$. Note that each state, w_i^e, in the word sequence model corresponds to some state u_j in one of the basic linguistic unit models.

$\bar{a}_{Ii}^e$, initial probability of the i^{th} state in e^{th} model W^e.

$\bar{a}_{ij}^e$, probability of performing transition from the i^{th} state to j^{th} state in e^{th} model.

$\bar{a}_{iF}^e$, final probability of the i^{th} state in e^{th} model.

$\bar{p}_i^e(.) = p^{w_i^e}(.)$, emission probability of the i^{th} state in e^{th} model.

$Q^e = \{q_0, q_1, ..., q_t, ..., q_{T^e-1}\}$, state sequence corresponding to the vector sequence $\mathbf{X}^e$ for the e^{th} model W^e.

EM performs two steps in each iteration, namely E and M steps. During the E step, estimates of posterior distribution of some hidden variables are found from the old values of the parameter set. During M step, those estimates are used to find the new values for parameters so that the overall likelihood of the training data is increased. The hidden variables include:

1. Probability of the i^{th} temporal state at time t, for the e^{th} model and utterance, $p(q_t = i|\mathbf{X}^e, W^e)$, denoted by $\gamma^e(i,t)$.
2. Probability of performing transition from the i^{th} state at time $t-1$ to the j^{th} state at time t, in the e^{th} model for the e^{th} utterance, $p(q_t = i|q_{t-1} = j, \mathbf{X}^e, W^e)$, denoted by $\varepsilon^e(j,i,t)$.
3. Probability of mixture component k in the GMM of i^{th} state in model W^e, emitting the vector at time t, $p(mixturecomponent = k|q_t = i, \mathbf{x}_t^e, W^e)$, denoted by $\chi^e(k,i,t)$.

The BW and Viterbi modes of training differ only in the way γ and ε are computed.

2.1.1 E - step

Computation of γ and ε:

In the BW mode of training, γ and ε are computed from two variables, namely α and β, which are defined as follows.

$$\alpha^e(i,t) = p(\mathbf{x}_0^e, \mathbf{x}_1^e, ..., \mathbf{x}_t^e, q_t = i|W^e)$$

$$\beta^e(i,t) = p(\mathbf{x}_{t+1}^e, \mathbf{x}_{t+2}^e, ..., \mathbf{x}_{T-1}^e|q_t = i, W^e)$$

α and β for the e^{th} utterance are computed as follows.

α's for time 0 are initialized as,

$$\alpha^e(i,0) = p(q_0 = i|I, W^e)p(\mathbf{x}_0^e|q_0 = i, W^e), \ 0 \le i \le N^e - 1$$

where, $p(q_0 = i|I, W^e)$ is the initial probability of the i^{th} state in e^{th} model, which is nothing but $\bar{a}_{Ii}^e$, and $p(\mathbf{x}_0^e|q_0 = i, W^e)$ is the likelihood of the i^{th} state in e^{th} model emitting the 0^{th} vector of e^{th} utterance, which is equal to $p^{w_i^e}(\mathbf{x}_0^e)$. Hence, the above equation in terms of the parameters of HMM becomes,

$$\alpha^e(i,0) = \bar{a}_{Ii}^e p^{w_i^e}(\mathbf{x}_0^e)$$

α's for $t > 0$ are found using recursion,

$$\alpha^e(i,t) = \sum_{j=0}^{N^e-1} \alpha^e(j,t-1)p(q_t = i|q_{t-1} = j, W^e)p(\mathbf{x}_t^e|q_t = i, W^e), \ 0 \le i \le N^e - 1, \ 0 < t \le T^e - 1$$

where, $p(q_t = i|q_{t-1} = j, W^e)$ is the probability of being in j^{th} state at time $t-1$ and in i^{th} state at time t for e^{th} model, which is equal to $\bar{a}_{ji}^e$, and $p(\mathbf{x}_t^e|q_t = i, W^e)$ is the likelihood of the i^{th} state in e^{th} model emitting the t^{th} vector of e^{th} utterance, which is equal to $p^{w_i^e}(\mathbf{x}_t^e)$. The above equation in terms of the parameters of HMM is,

$$\alpha^e(i,t) = \sum_{j=0}^{N^e-1} \alpha^e(j,t-1)\bar{a}_{ji}^e p^{w_i^e}(\mathbf{x}_t^e)$$

β's for time T^e-1 are initialized as,

$$\beta^e(i,T^e-1) = p(F|q_{T^e-1}=i,W^e),\ 0 \le i \le N^e-1$$

where, $p(F|q_{T^e-1}=i)$ is the final state probability of the i^{th} state in e^{th} model. In terms of HMM parameters,

$$\beta^e(i,T^e-1) = \bar{a}^e_{iF}$$

β's for $t < T^e-1$ are found using recursion,

$$\beta^e(i,t) = \sum_{j=0}^{N^e-1} p(q_{t+1}=j|q_t=i,W^e)p(\mathbf{x}^e_{t+1}|q_{t+1}=j,W^e)\beta^e(j,t+1),\ 0 \le i \le N^e-1,\ 0 \le t < T^e-1$$

In terms of HMM parameters,

$$\beta^e(i,t) = \sum_{j=0}^{N^e-1} \bar{a}^e_{ij} p^{w^e_j}(\mathbf{x}^e_{t+1})\beta^e(j,t+1)$$

γ's and ε's are computed from the α's and β's using the formulae given below.

$$\gamma^e(i,t) = \frac{\alpha^e(i,t)\beta^e(i,t)}{L^e},\ \ 0 \le i \le N^e-1,\ \ 0 \le t \le T^e-1 \tag{12}$$

$$\varepsilon^e(j,i,t) = \frac{\alpha^e(i,t-1)p(q_t=j|q_{t-1}=i,W^e)p(\mathbf{x}^e_t|q_t=j,W^e)\beta^e(j,t)}{L^e},$$
$$0 \le j \le N^e-1,\ \ 0 \le i \le N^e-1,\ \ 0 < t \le T^e-1 \tag{13}$$

where,

$$L^e = P(\mathbf{X}^e|W^e) = p(\mathbf{x}^e_0,\mathbf{x}^e_1,...,\mathbf{x}^e_{T^e-1}|W^e) = \sum_{i=0}^{N^e-1} \alpha^e(i,t)\beta^e(i,t)$$

In terms of HMM parameters, the Eqn. 13 becomes,

$$\varepsilon^e(j,i,t) = \frac{\alpha^e(i,t-1)\bar{a}^e_{ij}p^{w^e_j}(\mathbf{x}^e_t)\beta^e(j,t)}{L^e}$$

In Viterbi mode of training, γ and ε are found from best state sequence Q^e_{best} as follows: If for the e^{th} utterance and model, $\mathcal{C}^e$ represents the set of all possible state sequences, the best state sequence is found as,

$$Q^e_{\text{best}} = \underset{Q\,\in\,\mathcal{C}^e}{\text{argmax}}\ P(\mathbf{X}^e,Q|W^e,\Theta_{\text{old}})$$

From Q^e_{best}, γ's and ε's are found as follows:

$$\gamma^e(i,t) = \left| \begin{array}{ll} 1\,, & \text{if in } Q^e_{\text{best}},\ q_t=i \\ 0\,, & \text{otherwise.} \end{array} \right. \quad ,\ 0 \le i \le N^e-1,\ 0 \le t \le T^e-1 \tag{14}$$

$$\varepsilon^e(j,i,t) = \left| \begin{array}{ll} 1\,, & \text{if in } Q^e_{\text{best}},\ q_{t-1}=i \text{ and } q_t=j \\ 0\,, & \text{otherwise.} \end{array} \right. \quad ,\ 0 \le j \le N^e-1,\ 0 \le i \le N^e-1,\ 0 < t \le T^e-1 \tag{15}$$

Computation of χ:

χ's are found using formula given as follows.

$$\chi^e(k,i,t) = \frac{c_k^{w_i^e} G_k^{w_i^e}(\mathbf{x}_t^e)}{\sum_{l=0}^{K^{w_i^e}-1} c_l^{w_i^e} G_l^{w_i^e}(\mathbf{x}_t^e)} \ , 0 \le k \le K^{w_i^e}-1, 0 \le i \le N^e-1, 0 \le t \le T^e-1 \tag{16}$$

where $K^{w_i^e}$ denotes the number of mixture components in the GMM of i^{th} state in e^{th} model. $c_k^{w_i^e}$ and $G_k^{w_i^e}(.)$ represent, respectively, the weighting factor and Gaussian density function for the k^{th} mixture component in i^{th} state GMM of e^{th} model.

2.1.2 M - step

The new values of the parameter set are found from γ's, ε's, and χ's computed for all the training utterances. Let us define a variable z as follows:

$$z_{i,j}^{u,e} = \left| \begin{array}{l} 1\ , \ \text{if } u_i = w_j^e \\ 0\ , \ \text{otherwise.} \end{array} \right.$$

Expressions for the new values of parameters are given as follows.

Emission model parameters:

$$c_{ik}^u = \frac{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z_{i,j}^{u,e}\gamma^e(j,t)\chi^e(k,j,t)\right]}{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z_{i,j}^{u,e}\gamma^e(j,t)\right]} \ ,$$
$$0 \le k \le K^{u_i}-1,\ \ 0 \le i \le M^u-1,\ \ 0 \le u \le U-1 \tag{17}$$

$$\mu_{ik}^u = \frac{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z_{i,j}^{u,e}\gamma^e(j,t)\chi^e(k,j,t)\mathbf{x}_t^e\right]}{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z_{i,j}^{u,e}\gamma^e(j,t)\chi^e(k,j,t)\right]} \ ,$$
$$0 \le k \le K^{u_i}-1,\ \ 0 \le i \le M^u-1,\ \ 0 \le u \le U-1 \tag{18}$$

$$\Sigma_{ik}^u = \frac{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z_{i,j}^{u,e}\gamma^e(j,t)\chi^e(k,j,t)(\mathbf{x}_t^e-\mu_k^{u_i})^T(\mathbf{x}_t^e-\mu_k^{u_i})\right]}{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z_{i,j}^{u,e}\gamma^e(j,t)\chi^e(k,j,t)\right]} \ ,$$
$$0 \le k \le K^{u_i}-1,\ \ 0 \le i \le M^u-1,\ \ 0 \le u \le U-1 \tag{19}$$

If the covariance matrix is assumed to be a diagonal matrix, a simplified form of (19) to find the components of diagonal covariance matrix is given by,

$$\sigma_{ik}^u(d) = \frac{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z_{i,j}^{u,e}\gamma^e(j,t)\chi^e(k,j,t)(x_t^e(d)-\mu_k^{u_i}(d))^2\right]}{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z_{i,j}^{u,e}\gamma^e(j,t)\chi^e(k,j,t)\right]} \ ,$$
$$0 \le d \le D-1,\ \ 0 \le k \le K^{u_i}-1,\ \ 0 \le i \le M^u-1,\ \ 0 \le u \le U-1 \tag{20}$$

where D denotes the dimension of the acoustic vectors.

Transition parameters:

Let us define three more variables as follows:

$z^e_{i,j}$, whose value becomes 1, if while forming the e^{th} model, the probability of transition $\bar{a}^e_{ij}$, between the states w^e_i and w^e_j, is obtained from a single transition probability in the phoneme model. 0 if it is obtained from more than one probabilities. For example, in the Fig. 5, $z^e_{12} = 1$ and $z^e_{13} = 0$.

$z^{u,e}_{I,i,j}$, whose value becomes 1, if while forming the e^{th} model, transition between the states w^e_i and w^e_j is obtained from more than one transitions in the phoneme models, including the transition through initial state I of the u^{th} phoneme model. 0 otherwise.

$z^{u,e}_{i,j,F}$, whose value becomes 1, if while forming the e^{th} model, transition between the states w^e_i and w^e_j is obtained from more than one transitions in the phoneme models, including the transition through final state F of the u^{th} phoneme model. 0 otherwise.

Expressions for the transition parameters are given as follows:

$$a^u_{ij} = \frac{\sum_{e=0}^{E-1}\sum_{t=1}^{T^e-1}\sum_{b=0}^{N^e-1}\sum_{a=0}^{N^e-1}\left[z^{u,e}_{i,a} z^{u,e}_{j,b} z^e_{a,b}\varepsilon^e(b,a,t)\right]}{\sum_{e=0}^{E-1}\sum_{t=1}^{T^e-1}\sum_{b=0}^{N^e-1}\sum_{a=0}^{N^e-1}\left[z^{u,e}_{i,a}\varepsilon^e(b,a,t)\right]} ,$$

$$0 \le i \le M^u - 1,\ \ 0 \le j \le M^u - 1,\ \ 0 \le u \le U - 1 \qquad (21)$$

$$a^u_{Ii} = \frac{\sum_{e=0}^{E-1}\sum_{t=1}^{T^e-1}\sum_{b=0}^{N^e-1}\sum_{a=0}^{N^e-1}\left[z^{u,e}_{I,a,b} z^{u,e}_{j,b}\varepsilon^e(b,a,t)\right]}{\sum_{e=0}^{E-1}\sum_{t=1}^{T^e-1}\sum_{b=0}^{N^e-1}\sum_{a=0}^{N^e-1}\left[z^{u,e}_{I,a,b}\varepsilon^e(b,a,t)\right]} + \sum_{e=0}^{E-1}\sum_{a=0}^{N^e-1}\left[z^{u,e}_{i,a}\gamma^e(i,0)\right] ,$$

$$0 \le i \le M^u - 1,\ \ 0 \le u \le U - 1 \qquad (22)$$

$$a^u_{iF} = \frac{\sum_{e=0}^{E-1}\sum_{t=1}^{T^e-1}\sum_{b=0}^{N^e-1}\sum_{a=0}^{N^e-1}\left[z^e_{i,a} z^{u,,e}_{a,b,F}\varepsilon^e(b,a,t)\right]}{\sum_{e=0}^{E-1}\sum_{t=1}^{T^e-1}\sum_{b=0}^{N^e-1}\sum_{a=0}^{N^e-1}\left[z^e_{i,a}\varepsilon^e(b,a,t)\right]} ,$$

$$0 \le i \le M^u - 1,\ \ 0 \le u \le U - 1 \qquad (23)$$

2.2 Initialization of the parameters

The model parameters need to be initialized at the start of the EM, so that those initial values could be used as the old parameter values, Θ^{old}, for the first iteration. One way of initializing is *random initialization*, which assigns random values (may be between certain range) to the parameters. But random initialization can lead to convergence to a poor local minimum, and result in very poor performance. This could be avoided by guessing the initial values such that, in the parameter space, the initial values lie as close as possible to the optimal values. One method by which we can make a better guess for the initial values is *initialization by linear segmentation*, which assumes an imaginary iteration, in Viterbi mode, before the first iteration that has resulted in Q^e_{best} as follows: If $n = \frac{T^e}{N^e}$, $Q^e_{\text{best}} = \{q_0 = w^e_0, q_1 = w^e_0, ..., q_{n-1} = w^e_0, q_n = w^e_1, q_{n+1} = w^e_1, ..., q_{2n-1} = w^e_1, ..., q_{T^e-1} = w^e_{N^e-1}\}$. From Q^e_{best} for all the utterances, set of vectors associated with each state of the unit models can be extracted. The set of vectors corresponding to a given state is then clustered into number of groups equal to the number of gaussians in the GMM for that state, using K-Means clustering algorithm[2]. For example, vectors of i^{th} state in e^{th} unit model are clustered into K^{u_i} groups. Then mean vector and covariance matrix computed from each cluster are used as the mean and covariance for the corresponding gaussian.

All the transition parameters are initialized to equal values, satisfying the constraint, summation of all the probabilities should be equal to 1. For example, if there are n possible transitions from a particular state, then the probabilities of all the transitions starting from that state are initialized to value $\frac{1}{n}$.

2.3 Recognition

In recognition, the task is to identify the spoken sequence of words corresponding to the test utterance. If $\mathcal{L}$ represents the set of all possible word sequences that can be formed from the lexicon, acoustic models for all those word sequences can be developed from the acoustic models for basic linguistic units and lexicon. Using those models, spoken word sequence W^* corresponding to the vector sequence $\mathbf{X}$, extracted from the test utterance, can be found, along with an independently developed language model, using (2). If there is no language model assumed, i.e., if all the word sequences are assumed to be equi-probable, (2) simplifies to

$$W^* = \underset{W \in \mathcal{L}}{\operatorname{argmax}} \; P(\mathbf{X}|W, \Theta) \tag{24}$$

$P(\mathbf{X}|W, \Theta)$ for all the word sequences can be computed using (9) or (10). When (9) is used the recognition is said to be done in *Baum-Welch* (BW) mode, whereas when (10) is used recognition is said to be done in *Viterbi* mode. BW mode of recognition is nearly impossible to achieve in practice, as we need to form the models for all possible word sequences explicitly and compute $P(\mathbf{X}|W)$ for each of them. Hence, BW mode of recognition is never used.

The recognition software implemented here used the algorithm called *One Stage Dynamic Programming* [8, 9]. It virtually realizes all possible word sequence models from the models for all the words in the lexicon and computes $P(\mathbf{X}|W)$ using a dynamic programming procedure. A single word sequence model is formed by concatenating the models for all the words in the lexicon. The word sequence model is provided with additional transitions so that the initial state I of any word model can be reached from the final state F of any word model. Finding Q_{best} corresponding to the vector sequence $\mathbf{X}$ in this model is equivalent to finding out the spoken word sequence from all possible word sequences.

3 HMM2

HMM2 [3, 4] is the same as the classical HMM except for the emission density modeling. It integrates standard HMM (which will be referred to as *temporal HMM* here after) with state-dependent feature-based HMMs (referred to as *internal HMMs* or *feature HMMs*) for the purpose of emission density modeling. As depicted in Fig. 6, each time state is assigned an internal HMM, which treats each

[2]K-Means is an iterative algorithm to cluster the vectors in an unsupervised manner, which is explained as follows: Suppose there are N vectors which are to be clustered into K groups. To start with, K random vectors are picked up from the N vectors, to serve as mean vectors for each one of the clusters. Then in each iteration following steps are done repeatedly until convergence.

1. Each one of the N vectors are assigned to one of the clusters whose mean vector is closest to it, in terms of equilidean distance.
2. The mean vector for each cluster is updated as the mean of vectors assigned to that cluster.

acoustic vector as a fixed length sequence. Each vector is converted into a set of vectors, called internal vectors, and internal HMMs model them like the temporal HMM models the acoustic vector sequence. Each internal HMM has a set of states called internal states. The probability of emission of an acoustic vector $\mathbf{x}_t$ by an internal HMM is the likelihood of the internal states corresponding to it generating the internal vectors derived from that acoustic vector. Emission model in each internal state, which we call internal emission model, could be GMMs or ANNs.

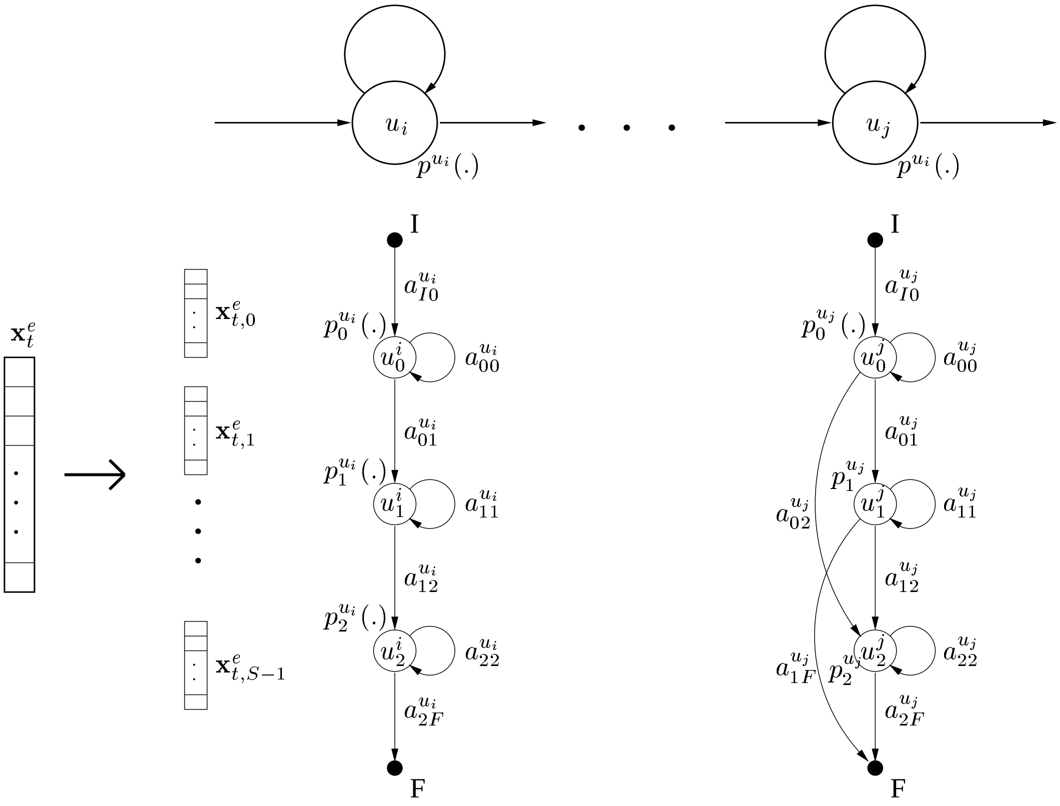

Figure 6: Illustration of HMM2.

HMM2, which includes classical mixture Gaussian HMM as a particular case, is believed to have several potential advantages over classical HMM as follows:

1. Internal HMM which is nothing but a mixture of GMMs can be shown to be a more flexible version of GMM. Topology of the internal HMM decides the forms of the emission density function which it could realize. This factor allows us to have better control over the various aspects of modeling based on the topology. For example,

   (a) If we use more internal states than the number of internal vectors, internal state transitions could possibly encode the correlation information across the acoustic vectors.

   (b) On the other hand, if we use fewer internal states, then the internal HMM will try to segment the feature vectors into subbands of similar characteristics.

   (c) It can be shown that, keeping the number of internal states same, introducing a transition between two internal states simply introduces few more gaussians, sharing the existing mean and covariance values for the new gaussians, if GMM is used as internal emission model.

2. In the same way as the temporal HMM performs nonlinear time warping and integration, internal HMM can perform nonlinear frequency warping and integration while operating on the spectral domain.

3. HMM2 has a potential to extract formant structure information [10], which is often considered as important to robust speech recognition.

Estimation of emission probability using internal HMM is done as follows: If the acoustic vector $\mathbf{x}_t$ is converted into an internal vector sequence of length S, $\{\mathbf{x}_{t,0}, \mathbf{x}_{t,1}, ..., \mathbf{x}_{t,s}, ..., \mathbf{x}_{t,S-1}\}$, the internal state sequence, denoted by $R = \{r_0, r_1, ..., r_s, ..., r_{S-1}\}$, can take several values depending upon the topology of the internal HMM. If $\mathcal{D}$ represents the set of all possible internal state sequences, the probability of internal HMM generating the internal vector sequence is given by,

$$p(\mathbf{x}_t) = \sum_{R \in \mathcal{D}} p(\mathbf{x}_t, R) \tag{25}$$

where,

$$p(\mathbf{x}_t, R) = p(r_0|I) p(x_{t,0}|r_0) \prod_{s=1}^{S-1} p(r_s|r_{s-1}) p(\mathbf{x}_{t,s}|r_s)$$

Similar to the temporal HMM, Viterbi approximation of the Eqn. 25 is given by,

$$p(\mathbf{x}_t) = \max_{R \in \mathcal{D}} p(\mathbf{x}_t, R) \tag{26}$$

Retaining the same notation as used for the classical HMM, and in addition, if there are M^{u_i} internal states, denoted by $\{u_0^i, u_1^i, ..., u_l^i, ..., u_{M^{u_i}-1}^i\}$, in the internal HMM of i^{th} state in the model for u^{th} basic linguistic unit, the entire parameter set of HMM2 is given as follows.

Initial probabilities of all the time states in all the unit models: $a_{Ii}^u,\ 0 \leq i \leq M^u - 1,\ 0 \leq u \leq U - 1$.

Probabilities of transitions between all the time states in all the unit models: $a_{ij}^u,\ 0 \leq i \leq M^u - 1,\ 0 \leq j \leq M^u - 1,\ 0 \leq u \leq U - 1$.

Final probabilities of all the time states in all the unit models: $a_{iF}^u,\ 0 \leq i \leq M^u - 1,\ 0 \leq u \leq U - 1$.

Initial probabilities of all the internal states in all the unit models: $a_{Il}^{u_i},\ 0 \leq l \leq M^{u_i} - 1,\ 0 \leq i \leq M^u - 1,\ 0 \leq u \leq U - 1$.

Probabilities of transitions between all the internal states in all the unit models: $a_{lm}^{u_i},\ 0 \leq l \leq M^{u_i} - 1,\ 0 \leq m \leq M^{u_i} - 1,\ 0 \leq i \leq M^u - 1,\ 0 \leq u \leq U - 1$.

Final probabilities of all the internal states in all the unit models: $a_{lF}^{u_i},\ 0 \leq l \leq M^{u_i} - 1,\ 0 \leq i \leq M^u - 1,\ 0 \leq u \leq U - 1$.

Emission density function for all the internal states in all the unit models: $p_l^{u_i}(.),\ 0 \leq l \leq M^{u_i},\ 0 \leq i \leq M^u - 1,\ 0 \leq u \leq U - 1$.

In the software developed, GMMs are used for the emission density modeling of the internal states. Hence, the parameters of $p_l^{u_i}(.)$ are,

$$[c_{lk}^{u_i} \quad \mu_{lk}^{u_i} \quad \Sigma_{lk}^{u_i}], \quad 0 \leq k \leq K_l^{u_i}$$

3.1 Training

The parameters of HMM2 can be jointly estimated using an EM algorithm, as given in [3]. EM for HMM2 differs from that of the classical HMM in the estimation of the emission model parameters. As in HMM2, HMM exists at temporal and frequency levels, it can be trained in 4 different modes:

1. Both temporal and internal HMMs in BW mode.
2. Temporal HMM in BW mode and internal HMM in Viterbi mode.
3. Temporal HMM in Viterbi mode and internal HMM in BW mode.
4. Both temporal and internal HMMs in Viterbi mode.

Retaining the same notation as used for the EM of the conventional HMM, few additional notations are given as follows.

S, length of the internal vector sequence obtained from the acoustic vectors.

$\{\mathbf{x}^e_{t,0}, \mathbf{x}^e_{t,1}, ..., \mathbf{x}^e_{t,s}, ..., \mathbf{x}^e_{t,S-1}\}$, internal vector sequence obtained from the acoustic vector $\mathbf{x}^e_t$.

$\mathcal{R}^{w^e_i} = \{r_0, r_1, ..., r_s, ..., r_{S-1}\}$, internal state sequence corresponding to the internal vector sequence of acoustic vector $\mathbf{x}^e_t$, for the internal HMM of i^{th} state in W^e.

EM algorithm performs 2 steps: E and M steps. During the E step, estimates of distribution of some hidden variables are found from the old parameter values, Θ^{old}. During M step, those estimates are used to find out the new values for parameters so that the over all likelihood of the training data is increased. The hidden variables estimated during the E step include,

1. Probability of i^{th} temporal state at time t, for the e^{th} model and utterance, $p(q_t = i|\mathbf{X}^e, W^e)$, denoted by $\gamma^e(i,t)$.
2. Probability of performing transition from the i^{th} temporal state at time $t-1$ to the j^{th} temporal state at t, in the e^{th} model for the e^{th} utterance, $p(q_t = j|q_{t-1} = i, \mathbf{X}^e, W^e)$, denoted by $\varepsilon^e(j,i,t)$.
3. Probability of l^{th} internal state at internal vector index s of the acoustic vector $\mathbf{x}^e_t$, for the i^{th} temporal state of e^{th} model at time t, $p(r_s = l|\mathbf{x}^e_t, q_t = i, W^e)$, denoted by $\chi^e(i,l,s,t)$.
4. Probability of performing transition from the l^{th} internal state at internal vector index $s-1$, to the m^{th} internal state at s, for the acoustic vector $\mathbf{x}^e_t$ and i^{th} temporal state of e^{th} model at time t, $p(r_s = m|r_{s-1} = l, \mathbf{x}^e_t, q_t = i, W^e)$, denoted by $\varepsilon^e(i,m,l,s,t)$.
5. Probability of k^{th} mixture component in the GMM of l^{th} internal state in i^{th} temporal state of e^{th} model emitting the vector $\mathbf{x}^e_{t,s}$ at internal vector index s, $p(mixturecomponent = k|r_s = l, \mathbf{x}^e_{t,s}, q_t = i, W^e)$, denoted by $\chi^e(i,k,l,s,t)$.

The E and M steps are explained as follows.

3.1.1 E - step

γ and ε for temporal HMM are computed in a manner similar to that of the computation γ and ε for classical HMM, as explained in Section 2.1.1. Computation of γ, ε, and χ for internal HMM are explained as follows.

Computation of internal γ and ε:

In BW mode, internal γ and ε are computed from two variables, namely internal α and β, which are defined as follows.

$$\alpha^e(i,l,s,t) = p(\mathbf{x}^e_{t,0}, \mathbf{x}^e_{t,1}, ..., \mathbf{x}^e_{t,s}, r_s = l | q_t = i, W^e)$$

$$\beta^e(i,l,s,t) = p(\mathbf{x}^e_{t,s+1}, \mathbf{x}^e_{t,s+2}, ..., \mathbf{x}^e_{t,S-1} | r_s = l, q_t = i, W^e)$$

Internal α and β for e^{th} utterance are computed as follows.

internal α's for internal vector index 0 are initialized as,

$$\alpha^e(i,l,0,t) = p(r_0 = l | I, q_t = i, W^e) p(\mathbf{x}^e_{t,0} | r_0 = l, q_t = i, W^e),$$

$$0 \le l \le M^{w^e_i} - 1,\ \ 0 \le i \le N^e - 1,\ \ 0 \le t \le T^e - 1$$

In terms of HMM2 parameters,

$$\alpha^e(i,l,0,t) = a^{w^e_i}_{Il} p^{w^e_i}_l(\mathbf{x}^e_{t,0})$$

internal α's for internal vector index > 0 are found using recursion,

$$\alpha^e(i,l,s,t) = \sum_{m=0}^{M^{w^e_i}-1} \alpha^e(i,m,s-1,t) p(r_s = l | r_{s-1} = m, q_t = i, W^e) p(\mathbf{x}^e_{t,s} | r_s = l, q_t = i, W^e),$$

$$0 \le l \le M^{w^e_i} - 1,\ \ 0 < s \le S - 1,\ \ 0 \le i \le N^e - 1,\ \ 0 \le t \le T^e - 1$$

In terms of HMM2 parameters,

$$\alpha^e(i,l,s,t) = \sum_{m=0}^{M^{w^e_i}-1} \alpha^e(i,m,s-1,t) a^{w^e_i}_{ml} p^{w^e_i}_l(\mathbf{x}^e_{t,s})$$

internal β's for internal vector index $S-1$ are initialized as,

$$\beta^e(i,l,S-1,t) = p(F | r_{S-1} = l, q_t = i, W^e), \quad 0 \le l \le M^{w^e_i} - 1,\ \ 0 \le i \le N^e - 1,\ \ 0 \le t \le T^e - 1$$

In terms of HMM2 parameters,

$$\beta^e(i,l,S-1,t) = a^{w^e_i}_{lF}$$

internal β's for internal vector index $< S-1$ are found using recursion,

$$\beta^e(i,l,s,t) = \sum_{m=0}^{M^{w^e_i}-1} p(r_{s+1} = m | r_s = l, q_t = i, W^e) p(\mathbf{x}^e_{t,s+1} | r_{s+1} = m, q_t = i, W^e) \beta^e(i,m,s+1,t),$$

$$0 \le l \le M^{w^e_i} - 1,\ \ 0 \le s < S - 1,\ \ 0 \le i \le N^e - 1,\ \ 0 \le t \le T^e - 1$$

In terms of HMM2 parameters,

$$\beta^e(i,l,s,t) = \sum_{m=0}^{M^{w^e_i}-1} a^{w^e_i}_{lm} p^{w^e_i}_m(\mathbf{x}^e_{t,s+1}) \beta^e(i,m,s+1,t)$$

Internal γ and ε are computed from internal α and β using the formulae given below.

$$\gamma^e(i,l,s,t) = \frac{\alpha^e(i,l,s,t)\beta^e(i,l,s,t)}{L^e_{it}} \ ,$$
$$0 \le l \le M^{w^e_i} - 1,\ \ 0 \le s \le S-1,\ \ 0 \le i \le N^e - 1,\ \ 0 \le t \le T^e - 1 \qquad (27)$$

$$\varepsilon^e(i,m,l,s,t) = \frac{\alpha^e(i,l,s-1,t)p(r_s = m|r_{s-1} = l, q_t = i, W^e)p(\mathbf{x}^e_{t,s}|r_s = m, q_t = i, W^e)\beta^e(i,m,s,t)}{L^e_{it}} \ ,$$
$$0 \le m \le M^{w^e_i} - 1,\ \ 0 \le l \le M^{w^e_i} - 1,\ \ 0 < s \le S-1,\ \ 0 \le i \le N^e - 1,\ \ 0 \le t \le T^e - 1 \qquad (28)$$

where L^e_{it} is the emission probability given as follows.

$$L^e_{it} = p(\mathbf{x}^e_t|q_t = i, W^e) = p^{w^e_i}(\mathbf{x}^e_t) = \sum_{l=0}^{M^{w^e_i}-1} \alpha^e(i,l,s,t)\beta^e(i,l,s,t)$$

In terms of HMM2 parameters, (28) becomes,

$$\varepsilon^e(i,m,l,s,t) = \frac{\alpha^e(i,l,s-1,t)a^{w^e_i}_{lm}p^{w^e_i}_m(\mathbf{x}^e_{t,s})\beta^e(i,m,s,t)}{L^e_{it}}$$

In Viterbi mode of training, internal γ and ε are found from the best internal state sequence $R^{e,i,t}_{\text{best}}$ as follows: If for the vector $\mathbf{x}^e_t$ and internal HMM in i^{th} state of e^{th} model, $\mathcal{D}^{e,i,t}$ represents the set of all possible internal state sequences, the best internal state sequence is found as,

$$R^{e,i,t}_{\text{best}} = \underset{R \in \mathcal{D}^{e,i,t}}{\text{argmax}} \ p(\mathbf{x}^e_t, R|q_t = i, W^e)$$

From $R^{e,i,t}_{\text{best}}$, internal γ and ε are found as follows.

$$\gamma^e(i,l,s,t) = \left| \begin{array}{ll} 1\ , & \text{if in } R^{e,i,t}_{\text{best}},\ r_s = l \\ 0\ , & \text{otherwise.} \end{array} \right.$$
$$0 \le l \le M^{w^e_i} - 1,\ \ 0 \le s \le S-1,\ \ 0 \le i \le N^e - 1,\ \ 0 \le t \le T^e - 1 \qquad (29)$$

$$\varepsilon^e(i,m,l,s,t) = \left| \begin{array}{ll} 1\ , & \text{if in } R^{e,i,t}_{\text{best}},\ r_{s-1} = l \text{ and } r_s = m \\ 0\ , & \text{otherwise.} \end{array} \right.$$
$$0 \le l \le M^{w^e_i} - 1,\ \ 0 \le m \le M^{w^e_i} - 1,\ \ 0 < s \le S-1,\ \ 0 \le i \le N^e - 1,\ \ 0 \le t \le T^e - 1 \qquad (30)$$

Computation of internal χ:

Equation given below is used for the computation of internal χ.

$$\chi^e(i,k,l,s,t) = \frac{c^{w^e_i}_{lk} G^{w^e_i}_{lk}(\mathbf{x}^e_{t,s})}{\sum_{n=0}^{K^{w^e_i}_l - 1} c^{w^e_i}_{ln} G^{w^e_i}_{ln}(\mathbf{x}^e_{t,s})} \ ,$$
$$0 \le k \le K^{w^e_i}_l - 1,\ \ 0 \le l \le M^{w^e_i} - 1,\ \ 0 \le s \le S-1,\ \ 0 \le i \le N^e - 1,\ \ 0 \le t \le T^e - 1 \qquad (31)$$

where $K^{w^e_i}_l$ denotes the number of mixture components in the GMM of l^{th} internal state in i^{th} temporal state of e^{th} model. $c^{w^e_i}_{lk}$ and $G^{w^e_i}_{lk}(.)$ represent, respectively, the weighting factor and Gaussian density function for the k^{th} mixture component of GMM in l^{th} internal state of i^{th} temporal state in e^{th} model.

3.1.2 M - step

a^u_{Ii}, a^u_{ij}, and a^u_{iF} are computed respectively using (22), (21), and (23), as given in Section 2.1.2. Expressions for $a^{u_i}_{Il}$, $a^{u_i}_{lm}$, $a^{u_i}_{lF}$, $c^{u_i}_{lk}$, $\mu^{u_i}_{lk}$, and $\Sigma^{u_i}_{lk}$ are given below.

$$c^{u_i}_{lk} = \frac{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z^{u,e}_{i,j}\gamma^e(j,t)\sum_{s=0}^{S-1}\gamma^e(j,l,s,t)\chi^e(j,k,l,s,t)\right]}{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z^{u,e}_{i,j}\gamma^e(j,t)\sum_{s=0}^{S-1}\gamma^e(j,l,s,t)\right]},$$

$$0 \le k \le K^{u_i}_l - 1,\ 0 \le l \le M^{u_i} - 1,\ 0 \le i \le M^u - 1,\ 0 \le u \le U - 1 \tag{32}$$

$$\mu^{u_i}_{lk} = \frac{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z^{u,e}_{i,j}\gamma^e(j,t)\sum_{s=0}^{S-1}\gamma^e(j,l,s,t)\chi^e(j,k,l,s,t)\mathbf{x}^e_{t,s}\right]}{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z^{u,e}_{i,j}\gamma^e(j,t)\sum_{s=0}^{S-1}\gamma^e(j,l,s,t)\chi^e(j,k,l,s,t)\right]},$$

$$0 \le k \le K^{u_i}_l - 1,\ 0 \le l \le M^{u_i} - 1,\ 0 \le i \le M^u - 1,\ 0 \le u \le U - 1 \tag{33}$$

$$\Sigma^{u_i}_{lk} = \frac{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z^{u,e}_{i,j}\gamma^e(j,t)\sum_{s=0}^{S-1}\gamma^e(j,l,s,t)\chi^e(j,k,l,s,t)(\mathbf{x}^e_{t,s}-\mu^{u_i}_{lk})(\mathbf{x}^e_{t,s}-\mu^{u_i}_{lk})^T\right]}{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z^{u,e}_{i,j}\gamma^e(j,t)\sum_{s=0}^{S-1}\gamma^e(j,l,s,t)\chi^e(j,k,l,s,t)\right]},$$

$$0 \le k \le K^{u_i}_l - 1,\ 0 \le l \le M^{u_i} - 1,\ 0 \le i \le M^u - 1,\ 0 \le u \le U - 1 \tag{34}$$

$$a^{u_i}_{lm} = \frac{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z^{u,e}_{i,j}\gamma^e(j,t)\sum_{s=1}^{S-1}\varepsilon^e(j,m,l,s,t)\right]}{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z^{u,e}_{i,j}\gamma^e(j,t)\sum_{s=1}^{S-1}\sum_{m=0}^{M^{u_i}-1}\varepsilon^e(j,m,l,s,t)\right]},$$

$$0 \le l \le M^{u_i} - 1,\ 0 \le m \le M^{u_i} - 1,\ 0 \le i \le M^u - 1,\ 0 \le u \le U - 1 \tag{35}$$

$$a^{u_i}_{Il} = \frac{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z^{u,e}_{i,j}\gamma^e(j,t)\gamma^e(j,l,0,t)\right]}{\sum_{e=0}^{E-1}\sum_{t=0}^{T^e-1}\sum_{j=0}^{N^e-1}\left[z^{u,e}_{i,j}\gamma^e(j,t)\right]},$$

$$0 \le l \le M^{u_i} - 1,\ 0 \le i \le M^u - 1,\ 0 \le u \le U - 1 \tag{36}$$

3.2 Initialization of HMM2 parameters

Similar to the classical HMM, HMM2 parameters can also be initialized by *random initialization* or *initialization by linear segmentation*. In the initialization by linear segmentation procedure, having collected the set of vectors for each temporal state, using the procedure as explained in Section 2.2, the computation of emission parameters is done as follows: Internal vector sequence corresponding to each acoustic vector belonging to each temporal state is assumed to have been Viterbi aligned with the corresponding internal HMM and have resulted in $R^{u,i,t,e}_{\text{best}}$ as follows. If $n = \frac{S}{M^{u_i}}$, $R^{u,i,t,e}_{\text{best}} = \{r_0 = u^i_0, r_1 = u^i_0, ..., r_{n-1} = u^i_0, r_n = u^i_1, ..., r_{2n-1} = u^i_1, ..., r_{S-1} = u^i_{M^{u_i}-1}\}$. From $R^{u,i,t,e}_{\text{best}}$ for all possible cases, set of internal vectors corresponding to each internal state in each unit model is found out. The set of internal vectors corresponding to a given internal state is then clustered into number of groups equal to the number of gaussians in GMM for that internal state, using K-Means clustering algorithm. For example, internal vectors of l^{th} internal state in i^{th} temporal state of e^{th} model are clustered into $K^{u_i}_l$ groups. Then mean vector and covariance matrix computed from each cluster are used as the mean and covariance for the corresponding gaussian.

3.3 Recognition

Recognition using HMM2 is performed in a similar way as recognition using classical HMM, which is explained in the Section 2.3. The only difference is the emission probability computation, which is done using internal HMM instead of GMMs. As emission probability using internal HMM can be computed in BW or Viterbi mode, recognition using HMM2 can be performed in two different modes.

4 Software Description

The software has been developed mainly for carrying out research in robust speech recognition. Implementation of the software provides enough flexibility to test various HMM/HMM2 topologies, for various parameterization and training schemes. By giving all the inputs required to the program in appropriate format, experiments can be conducted on large databases and performance can be measured. The whole software is written in C language. The main functionalities of the software are given in Section 4.1. A detailed coverage of specifications required to use the software is given in Section 4.2.

4.1 Functionalities

The software described here can accomodate both the conventional HMM and HMM2. The discrimination between the two is made in the specification of the emission model. If GMM is used as the emission model, the software implements an HMM. Whereas, if internal HMM is used, the software implements an HMM2. It is also possible to use a combination of both HMM and HMM2 systems, by specifying GMM as emission model for a few temporal states and internal HMM as emission model for the others. The system provides a modular way of developing HMM/HMM2 based acoustic models for word sequences, from the independent HMM/HMM2 based acoustic models defined and developed for basic linguistic units. The basic linguistic units could be words or subwords like phonemes.

The software entertain the following functionalities:

1. Training of the conventional HMM can be done using EM algorithm, as explained in Section 2.1, in one of the following two modes.
   (a) BW mode.
   (b) Viterbi mode.
2. Training of HMM2 can be done using EM algorithm, as explained in Section 3.1, in one of the following modes.
   (a) BW for both temporal and internal HMMs.
   (b) BW for temporal HMM and Viterbi for internal HMM.
   (c) Viterbi for temporal HMM and BW for internal HMM.
   (d) Viterbi for both temporal and internal HMMs.
3. Recognition with conventional HMM can be done in Viterbi mode using an efficient algorithm called One stage dynamic programming, as explained in Section 2.3.
   - Not possible to use language model with the present version.
4. Recognition with HMM2 can be done in Viterbi mode using an efficient algorithm called One stage dynamic programming, as explained in Section 3.3.
   - Emission probabilities using internal HMM can be computed in BW or Viterbi mode.

- Not possible to use language models with the present version.

5. Initialization of the HMM and HMM2 parameters, can be done through one of the following procedures[3].
   (a) Initialization by linear segmentation - explained in Sections 2.2 and 3.2.
   (b) File initialization - using the previously stored values of parameters, in case of retraining.
   (c) Random initialization - explained in Sections 2.2 and 3.2.
6. Highly flexible topology specification procedure allows specification of any kind of topology for both the temporal HMM and internal HMM (provided they can be trained!).
7. Highly flexible pronunciation model specification procedure for the lexicon.
8. For the emission modeling of classical HMM and the internal emission modeling of HMM2, only GMMs can be used with the current version of the software.
9. For GMMs, only diagonal covariance matrix can be used with the current version of the software.
10. The software can also be used for tasks other than speech recognition, like character recognition in image processing.

4.2 Specification

4.2.1 Input format

All the inputs required for the execution of the program are sent through command line arguments and a few files.

Command line arguments:

A list of inputs to be passed through command line arguments, during training, is given below:

1. Train mode for temporal HMM (BW or Viterbi).
2. Train mode for internal HMM (BW or Viterbi).
3. Name of the file containing the list of all the basic linguistic units and the topology specifications of their acoustic models.
4. Name of the file containing the list of all the utterances to be used for training.
5. Name of the file containing the information about the feature vectors (both the acoustic and internal vectors).
6. Initialization procedure to be used for initializing all the parameters. (Initialization by linear segmentation or File initialization or Random initialization).
7. Convergence criteria to be used for stopping the training.
8. Directory, including the complete path,
   (a) of the location of all the training data.
   (b) of location of transcription files for all the training data.
   (c) where all the outputs generated by the program (model parameter values) should be stored.

[3] In addition to the various initialization procedures listed above, initialization of the parameters can also be done from the time alignment information given along with the database. Current version of software do not include that.

(d) where all the intermediate files generated by the program (which can be deleted after the completion of the program execution) should be stored.

A list of inputs to be passed through command line arguments, during recognition, is given below:

1. Emission probability computation mode for internal HMM (BW or Viterbi).
2. Name of the file containing list of all the basic linguistic units and topology specifications of all their acoustic models.
3. Name of the file containing list of all the test utterances to be recognized.
4. Name of the file containing information about the feature vectors (both the acoustic and internal vectors).
5. Name of the file containing list of words in the lexicon.
6. Directory, including the complete path,
   (a) of location of all the test utterances.
   (b) of location of transcription files for all the word in the lexicon.
   (c) from where all the model parameter values should be read.
   (d) where the recognition result is to be written.
   (e) where all the intermediate files generated by the program (which can be deleted after the completion of the program execution) are to be written.

Input files:
List of all the files through which some inputs are passed to the program is as follows.

1. *Train utterances list file*, containing names of all the training utterances to be used for training.
2. *Test utterances list file*, containing names of all the test utterances that are to be recognized.
3. *Lexicon file*, containing a list of all the words in the lexicon.
4. *Topology file*, containing list of all the basic linguistic units and the topology specifications of their acoustic models.
5. *Transcription files*, containing transcription, in terms of the basic linguistic units defined in the *topology file*, for
   (a) All the training utterances.
   (b) All the words in the lexicon.

The format in which various inputs are to be stored in the files are as follows: The first 3 files, namely the *train utterances list file*, *test utterances list file*, and *lexicon file*, contain the list of names. Each name should be separated by a space or a new-line character. The format of *topology file* is given in Appendix A. The format of *transcription files* is given in Appendix B.

4.2.2 Algorithm

Steps performed by the program during training are as follows:

1. Read all the inputs passed through the *command line arguments*.
2. Read the list of all the basic linguistic units from *topology file*.
3. Read the topology specifications for all the linguistic units, from *topology file*, and build HMM/HMM2 based acoustic models for each of them.
4. Read the list of utterances to be used for training, from *train utterances list file*.
5. Extract feature vectors from all the training utterances according to the feature vector specification.
6. Read transcription of all the training utterances, from *transcription files*.
7. Build word sequence acoustic models for all the training utterances, from the transcriptions and the acoustic models for all the basic linguistic units, using the procedure explained in Section 2.
8. Initialize the parameters of acoustic models of all the basic linguistic units, based on the initialization procedure specified. Various initialization procedures are explained in Sections 2.2 and 3.2.
9. Compute the parameters of word sequence acoustic models from the parameters of acoustic models of basic linguistic units, as explained in Section 2.
10. Perform E step, as explained in Sections 2.1 and 3.1.
11. Perform M step, as explained in Sections 2.1 and 3.1.
12. Store the new values of the parameters of acoustic models for all the basic linguistic units.
13. Stop the program if converged, else repeat the steps from step 9 to 13.

Steps performed by the program during recognition are as follows:

1. Read all the inputs passed through the *command line arguments*.
2. Read the list of all the basic linguistic units, from *topology file*.
3. Read the topology specifications for all the linguistic units, from *topology file*, and build HMM/HMM2 based acoustic models for each of them.
4. Read the values of parameters of acoustic models for all the basic linguistic units.
5. Read the list of words, from *lexicon file*.
6. Read the transcription of all the words in the lexicon, from *transcription files*.
7. Build word models for all the words in the lexicon, from the transcriptions and acoustic models of basic linguistic units, using procedure as explained in Section 2.
8. Build word sequence model, from word models of step 7, using procedure explained in Sections 2.3 and 3.3, for performing One Stage Dynamic Programming.
9. Read the list of utterances to be recognized, from *text utterances list file*.
10. Extract feature vectors from each test utterance according to the feature vector specification.

11. Perform One Stage Dynamic Programming for all the test utterances, as explained in Sections 2.3 and 3.3, to recognize the corresponding word sequence spoken.
12. Compute overall performance (Word Recognition Rate) from the recognized and true word sequences for all the test utterances.

4.2.3 Data handling

The parameters of acoustic models for basic linguistic units are handled in the program as structures explained as follows: For each linguistic unit a structure with elements given below is defined.

unit_model_parameters –>	*number_of_states*
	transition_topology[]
	emission_gmm[]
	emission_internalHMM[]

where *number_of_states* specifies the number of temporal states in the unit model. For u^{th} linguistic unit *number_of_states* will be equal to M^u. *transition_topology*, *emission_gmm*, and *emission_internalHMM* are array of structures, specifying the transition and emission topologies for all the temporal states. The size of these arrays are equal to *number_of_states*. *transition_topology*[i] specifies all the transitions starting from i^{th} temporal state. If the system is conventional HMM, *emission_model*[i] specifies the parameters of GMM for i^{th} temporal state. If the system is HMM2, *emission_internalHMM*[i] specifies the topology of internal HMM for the i^{th} temporal state. The elements of structure *transition_topology* are given as follows.

transition_topology[i] –>	*number_of_transitions*
	state_index[]
	probability_of_transitions[]

where *number_of_transitions* specifies the number of transitions starting from temporal state i. *state_index* and *probability_of_transition* are, respectively, arrays of integers and floats. The sizes of both the arrays are equal to *number_of_transitions*. *state_index*[j] specifies index of the temporal state to which j^{th} transition from i^{th} temporal state occurs. The probability of that transition is specified in *probability_of_transition*[j]. The elements of the structure *emission_gmm* are given as follows.

emission_gmm[i] –>	*number_of_mixtures*
	weight[]
	mean[][]
	diagonal_covariance[][]

where *number_of_mixtures* specifies the number of mixture components in the GMM for i^{th} temporal state, K^{u_i}. *weight*, *mean*, and *diagonal_covariance* specify respectively the weighting factor, mean vector, and diagonal covariance matrix for each component in the gaussian. The elements of structure *emission_internalHMM* are given as follows.

emission_internalHMM[i] –>	*number_of_internal_states*
	initial_probability[]
	transition_topology[]
	emission_gmm[]

where, *number_of_internal_states* specifies the number of internal states in the internal HMM for i^{th} temporal state, M^{U_i}. *initial_probability* specifies initial probabilities of all the internal states in i^{th} temporal state. *transition_topology* and *emission_gmm* are structures, defined early, specify the transition topology and emission topology of all the internal states in i^{th} temporal state.

The parameters of acoustic models for words or word sequences are handled in the program as structures explained as follows: Model for word sequence or word is specified as an array of structures, with each array element assigned to one temporal state in the model. The elements of structure used are as follows.

word_model_parameters –> | *unit_index*
| *state_index*
| *initial_probability*
| *number_of_transitions*
| *state_index*[]
| *probability_of_transition*[]
| *final_state_probability*

where *unit_index* specifies the index of the linguistic unit model, u, from which the state has been taken, while forming the word or word sequence model. *state_index* specifies the index of the temporal state, i, within the unit model. *initial_probability* specifies the initial probability of the word sequence state corresponding to the structure. *number_of_transitions* specifies the number of transitions from the present state to the other states within the word sequence model. *state_index* and *probability_of_transition* are, respectively, arrays of integers and floats, with size equal to *number_of_transitions*. *state_index*[j] specifies the index of the state to which j^{th} transition from the present state occurs, within the word sequence model. *probability_of_transition*[j] specifies the probability of that transition. *final_state_probability* specifies the probability of transition from the present state to the non-emitting final state within the word sequence model.

4.2.4 Specific implementation details

Specific cases of software implementation details which require special attention are as follows:

Underflow handling: Computation of α and β in the E-step of HMM and HMM2 training (see Sections 2.1.1 and 3.1.1 involve multiplications of several probability values. Since all the probability values are in the range from 0.0 to 1.0, multiplication of several probabilities may result in values less than the minimum value that can be handled by float or double, a condition referred to as underflow. To avoid this and keep the values inside the operating range, variables called scaled α and β, denoted by $\tilde{\alpha}$ and $\tilde{\beta}$, are used instead of normal α and β [2]. Equations related to the computation of $\tilde{\alpha}$ and $\tilde{\beta}$, and the computation of γ and ε from $\tilde{\alpha}$ and $\tilde{\beta}$ are given as follows:

$$\alpha(i,0) = \bar{a}_{Ii}\bar{p}_i(\mathbf{x}_0)\ ,\ \ 0 \le i \le N-1$$

$$\hat{\alpha}(i,0) = \alpha(i,0)\ ,\ \ 0 \le i \le N-1$$

$$c(0) = \frac{1}{\sum_{j=0}^{N-1}\hat{\alpha}(j,0)}$$

$$\tilde{\alpha}(i,0) = c(0)\hat{\alpha}(i,0)\ ,\ \ 0 \le i \le N-1$$

$$\hat{\alpha}(i,t) = \sum_{j=0}^{N-1}\tilde{\alpha}(i,t-1)\bar{a}_{ij}\bar{p}_i(\mathbf{x}_t)\ ,\ \ 0 \le i \le N-1,\ \ 0 < t \le T-1$$

$$c(t) = \frac{1}{\sum_{j=0}^{N-1}\hat{\alpha}(j,t)}\ ,\ \ 0 < t \le T-1$$

$$\tilde{\alpha}(i,t) = c(t)\hat{\alpha}(i,t)\ ,\ \ 0 \le i \le N-1,\ \ 0 < t \le T-1$$

$$\beta(i,T-1) = \bar{a}_{iF}\ ,\ \ 0 \le i \le N-1$$

$$\hat{\beta}(i,T-1) = \beta(i,T-1)\ ,\ \ 0 \le i \le N-1$$

$$\tilde{\beta}(i,T-1) = c(T-1)\hat{\beta}(i,T-1)\ ,\ \ 0 \le\le N-1$$

$$\hat{\beta}(i,t) = \sum_{j=0}^{N-1} \bar{a}_{ij}\bar{p}_j(\mathbf{x}_{t+1})\tilde{\beta}(j,t+1)\ ,\ \ 0 \le i \le N-1,\ \ 0 \le t < T-1$$

$$\tilde{\beta}(i,t) = c(t)\hat{\beta}(i,t)\ ,\ \ 0 \le i \le N-1,\ \ 0 \le t < T-1$$

$$\gamma(i,t) = \frac{\tilde{\alpha}(i,t)\tilde{\beta}(i,t)}{c(t)}\ ,\ \ 0 \le i \le N-1,\ \ 0 \le t \le T-1$$

$$\varepsilon(j,i,t) = \tilde{\alpha}(i,t-1)\bar{a}_{ij}\bar{p}_j(\mathbf{x}_t)\tilde{\beta}(j,t)\ ,\ \ 0 \le i \le N-1,\ \ 0 \le j \le N-1,\ \ 0 < t \le T-1$$

Internal α and β are also handled in a similar manner using internal $\tilde{\alpha}$ and $\tilde{\beta}$.

Memory optimization:

During training, it will be very expensive, memorywise, to keep all the feature vectors and word sequence models of all the training utterances in RAM. To avoid this they are stored in separate files in system disk, in a directory mentioned as *cache directory* through command line arguments. In the program a single feature vector array and a single word sequence model structure of dimension equal to the maximum required, are used. The feature vectors and word sequence model parameters of various training utterances are read into these variables from the corresponding files as and when required.

Log optimization:

If the training is done in Viterbi mode and if GMM with mixture size 1 is used for emission or internal emission, all the probability values could be dealt in logarithmic domain alone. In such case the computation involved in exponential and logarithmic operations, required for conversion between logarithmic and real values of probabilities and vice versa, can be avoided completely. A flag called *log_optimization_flag* passed to the program through command line arguments is used for this purpose.

All the variables used for storing probability values are defined two times, one for storing the real probability values and the other for storing the logarithms of probabilities. This is done because program requires logarithmic values of probabilities instead of the real values at most of the places. Single logarithmic computation and storage will save the unnecessary computation involved in repeated calculation of logarithm for same probability value at different places, as logarithm is a computationally expensive operation.

4.3 Results of Some Preliminary Experiments

Results of few preliminary experiments conducted on *Numbers 95* database[4] using the new software are given below. 3233 utterances from the training set were used for training and 1206 utterances

[4]Speech database containing speaker-independent free formant numbers spoken over telephone.

from the development set were used for testing. Initialization of the parameters were done through linear segmentation procedure. A floor value of 1.0 x 10^{-3} was used for all the variance values in the parameter set, to aviod log zero operations.

Table 1 gives performance comparison between the HMM and HMM2 systems implemented using new software. Though the performance of HMM2 system is not competitive with the classical HMM system, these results show the good behavior of the software.

Type	Emission Topology	Word Recognition Rate, % acc.
HMM	GMM with mixture size 2, diagonal covariance matrix	86.8%
HMM2	Top-down internal HMM with 39 internal states, GMM with mixture size 2 for internal emission	81.4%

Table 1: Performance comparison between the HMM and HMM2 systems implemented using new software. Feature vector used is 39 dimensional MFCC (13 coefficients + 13 delta coefficients + 13 delta delta coefficients). 27 phonemes are used as basic linguistic units, and 3 state L-R temporal HMM is used for each phoneme model.

Table 2 gives performance comparison between HMM systems implemented using the new software and HTK[5]. These results show that the new software is as good as HTK. In the experiment conducted with HTK, initialization of the parameters was done using the time alignment information given along with the training data set. Variable variance floor values[6] were used for all the variance values in the parameters.

Software used	Word Recognition Rate, % acc.
IDIAP HMM/HMM2 system	89.9 %
HTK	90.7%

Table 2: Performance comparison between HMM systems implemented using the new software and HTK. Feature vector used is 39 dimensional MFCC (13 coefficients + 13 delta coefficients + 13 delta delta coefficients). 27 phonemes are used as basic linguistic units, 3 state L-R temporal HMM is used for each phoneme model, and GMM with 10 mixture components and diagonal covariance matrix is used for emission density modeling in each temporal state.

5 Conclusion

In this paper, a theoretical basis of classical HMM and a new form of HMM, referred to as HMM2, is presented. HMM2 has been introduced as an alternative to HMM, in order to handle the distortions in the speech signal, introduced by the variability generally observed in real acoustic environments like speaker, channel, and noise characteristics. Various advantages of HMM2 that could result in improved robustness of the speech recognition system is also discussed. In the end, various functionalities and specifications of a new software developed to handle, in a flexible manner, different forms of HMM and HMM2 topologies and training schemes is given. Results of some preliminary experiments conducted is also given. Though the performance of HMM2 system is not competitive with that of the classical HMM system, the results show the good behavior of the software.

[5] HTK is a toolkit for building HMM models, extensively used in speech recognition applications.
[6] Refer to HTK reference manual.

References

[1] Bourlard H. and Morgan N. (1993). "Connectionist Speech Recognition: A Hybrid Approach". *The Kluwer International Series in Engineering and Computer Science*, Kluwer Academic Publishers, Boston, USA. Vol. 247, 1993.

[2] Rabiner L. and Juang B. H. (1993). "Fundamentals of Speech Recognition". *PTR Prentice-Hall, Inc.*, Englewood Cliffs, NJ, USA. 1993

[3] Bengio S., Bourlard H., and Weber K. (2000). "An EM Algorithm for HMMs with Emission Distributions Represented by HMMs". *IDIAP Research Report*, IDIAP, Martigny, Switzerland. IDIAP-RR 00-11, May 2000. ftp://ftp.idiap.ch/pub/reports/2000/rr00-11.ps.gz.

[4] Weber K., Bengio S., and Bourlard H. (2000). "HMM2 - A Novel Approach to HMM Emission Probability Estimation". *in Proc. of ICSLP.* Vol. 3, 147-150. Oct. 2000. ftp://ftp.idiap.ch/pub/reports/2000/rr00-30.ps.gz.

[5] Baum L. E., Petrie T., Soules G., and Weiss N. (1970). "A Maximization Technique Occuring in the Statistical Analysis of Probabilistic Functions of Markov Chains". *Ann. Math. Statistic.* Vol. 41, 164-171, 1970.

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

[7] Bilmes J. A. (1998). "A Gentle Tutorial of the EM algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models". *Report*, ICSI, Berkeley, CA, USA. TR-97-021, Apr. 1998.

[8] Bourlard H., Kamp Y., Ney H., and Wellekens C. J. (1985). "Speaker-Dependent Connected Speech Recognition via Dynamic Programming and Statistical Methods". in *Speech and Speaker Recognition* (editors, Schroeder M. R., and Gottingen) *Karger*, Basel. No. 12, 115-148, 1985.

[9] Ney H. (1984). "The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition". IEEE Trans. on ASSP. ASSP-32, 263-271, 1984.

[10] Weber K., Bengio S., and Bourlard H. (2001). "HMM2 - Extraction of Formant Structures and Their Use for Robust ASR". *Eurospeech*, Aalborg, Denmark. 607-610, Sep. 2001.

Appendix A

The format in which various inputs in *topology file* should be written is given as follows:

n_basic_linguistic_units U
{list of basic linguistic units}
transition_topology_similarity_flag *value*
{transition topology specification}
emission_similarity_flag *value*
{emission specification}

where 'n_basic_linguistic_units' specifies the number of basic linguistic units to be used, U. The *value* for 'transition_topology_similarity_flag' specifies whether the transition topology for all the linguistic units are same or not. If $value = 1$, they are all same. If $value = 0$, they are not same. The value for 'emission_similarity_flag' tells whether the emission model for all the time states are same or not. If $value = 1$, they are all same. If $value = 0$, they are not same. The list of basic linguistic units is specified in {list of basic linguistic units}. The transition and emission topologies of those linguistic units are specified respectively in {transition topology specification} and {emission specification}. {list of basic linguistic units} is given as follows.

0	*symbol of* 0^{th} *linguistic unit*
1	*symbol of* 1^{st} *linguistic unit*
:	:
u	*symbol of* u^{th} *linguistic unit*
:	:
$U-1$	*symbol of* $(U-1)^{th}$ *linguistic unit*

If the transition topologies of all the linguistic units are same, then {transition topology specification} becomes,

{transition topology specification for single linguistic unit}

If they are not same, then {transition topology specification} specifies transition topologies of all the unit models as,

0	{transition topology specification for single linguistic unit}
1	{transition topology specification for single linguistic unit}
:	:
u	{transition topology specification for single linguistic unit}
:	:
$U-1$	{transition topology specification for single linguistic unit}

where, u denotes the u^{th} linguistic unit. {transition topology specification for single linguistic unit} is given as follows.

n_states M^u
from -1 n_to_states *number*
    $tostate[0]$ $tostate[1]$ ... $tostate[number-1]$
from 0 n_to_states *number*
    $tostate[0]$ $tostate[1]$ ... $tostate[number-1]$
from 1 n_to_states *number*
    $tostate[0]$ $tostate[1]$ ... $tostate[number-1]$
    :
from $M^u - 1$ n_to_states *number*
    $tostate[0]$ $tostate[1]$ ... $tostate[number-1]$

where 'n_states' specifies the number of temporal states in the corresponding linguistic unit. For u^{th} linguistic unit it is equal to M^u. 'from i' specifies all the transitions starting from i^{th} state. *number* for 'n_to_states' specifies the total number of transitions starting from i^{th} state, with termination state indices specified by $tostate[j]$. State indices -1 and M^u stands for initial and final states respectively.

If the emission topologies for all the states in all the unit models are same, then the {emission specification} becomes,

> {emission specification for single state}

If they are not same, then the {emission specification} specifies emission topologies of all the states in all the unit models as,

$$\begin{array}{|ccl}
0 & 0 & \{\text{emission specification for single state}\} \\
0 & 1 & \{\text{emission specification for single state}\} \\
\vdots & \vdots & \quad\vdots \\
0 & M^0 - 1 & \{\text{emission specification for single state}\} \\
\vdots & \vdots & \quad\vdots \\
u & 0 & \{\text{emission specification for single state}\} \\
\vdots & \vdots & \quad\vdots \\
u & u_i & \{\text{emission specification for single state}\} \\
\vdots & \vdots & \quad\vdots \\
u & M^u - 1 & \{\text{emission specification for single state}\} \\
\vdots & \vdots & \quad\vdots \\
U-1 & 0 & \{\text{emission specification for single state}\} \\
\vdots & \vdots & \quad\vdots \\
U-1 & M^{U-1} - 1 & \{\text{emission specification for single state}\}
\end{array}$$

where u denotes the u^{th} linguistic unit and u_i denotes the i^{th} state in the u^{th} model. {emission specification for single state} is given as follows.

> emission_model_flag *value*
> {emission model specification}

The *value* for the 'emission_model_flag' specifies whether the emission model is GMM or Internal HMM. If the *value*= 0, emission model is GMM. It it is 1, emission model is Internal HMM. If it is GMM, {emission model specification} becomes,

> {GMM specification}

If it is Internal HMM, {emission model specification} becomes,

> {Internal HMM specification }

where {GMM specification} is as follows.

> n_mixtures K^{u_i}
> covariance_flag *value*

where 'n_mixtures' specifies the number of mixture components in the GMM. For i^{th} state in u^{th} model it becomes K^{u_i}. The *value* for 'covariance_flag' specifies whether the covariance matrix of GMM is diagonal or not. With the current version of the software it is possible to use only diagonal covariance matrix. If the *value*= 0, covariance matrix is a diagonal matrix. If Internal HMM is used for emission modeling, {Internal HMM specification} is as follows.

> n_internal_states M^{u_i}
> {Internal HMM transition topology}
> internal_emission_model_similarity_flag *value*
> {internal emission specification}

where 'n_internal_states' specifies the number of internal states in the internal HMM. For internal HMM of i^{th} state in u^{th} unit model this becomes M^{u_i}. The *value* for 'internal_emission_model_similarity_flag' specifies whether the internal emission model for all the internal states are same or not. If *value*$= 1$, they are all same. If *value*$= 0$, they are not same. {internal HMM transition topology} and {internal emission specification} specifies respectively the transition and emission topologies of internal HMM. {Internal HMM transition topology} is given as follows.

> from -1 n_to_states *number*
> $\quad tostate[0]\;\; tostate[1]\; ...\; tostate[number-1]$
> from 0 n_to_states *number*
> $\quad tostate[0]\;\; tostate[1]\; ...\; tostate[number-1]$
> $\vdots$
> from l n_to_states *number*
> $\quad tostate[0]\;\; tostate[1]\; ...\; tostate[number-1]$
> $\vdots$
> from $M^{u_i}-1$ n_to_states *number*
> $\quad tostate[0]\;\; tostate[1]\; ...\; tostate[number-1]$

where 'from l' specifies all the transition starting from internal state l of internal HMM. *number* for 'n_to_states' specifies the total number of transition starting from l^{mboxth} internal state, with termination internal state indices specified by $tostate[m]$.

If internal emission topology for all the internal states are same, {internal emission specification} becomes,

> {internal emission specification for single internal state}

If they are not same, {internal emission specification} specifies the internal emission topologies for all the internal states as follows.

> 0 {internal emission specification for single internal state}
> 1 {internal emission specification for single internal state}
> : :
> l {internal emission specification for single internal state}
> : :
> M^{u_i} {internal emission specification for single internal state}

where l denote the l^{th} internal state. {internal emission specification for single internal state} is given as follows.

> internal_emission_model_flag *value*
> {internal emission model specification}

where *value* for 'internal_emission_model_flag' specifies the internal emission model, which can be only GMM with the current version of the software. $value = 0$ specifies GMM. In that case {internal emission model specification} becomes {GMM specification}, which is given already.

Appendix B

Transcription files are used to specify the pronounciation models for all the words. The format of specification is as follows:

$$
\begin{array}{|l}
n\_linguistic\_units \\
unit\_symbol[0] \;\; unit\_symbol[1] \; ... \; unit\_symbol[n\_linguistic\_units - 1] \\
-1 \;\; n\_to\_units \\
\qquad\quad to\_unit[0] \;\; to\_unit[1] \; ... \; to\_unit[n\_to\_states - 1] \\
\qquad\qquad\qquad \vdots \\
i \;\; n\_to\_units \\
\qquad\quad to\_unit[0] \;\; to\_unit[1] \; ... \; to\_unit[n\_to\_states - 1] \\
\qquad\qquad\qquad \vdots \\
n\_linguistic\_units \;\; n\_to\_units \\
\qquad\quad to\_unit[0] \;\; to\_unit[1] \; ... \; to\_unit[n\_to\_states - 1]
\end{array}
$$

where $n\_linguistic\_units$ specifies the number of basic linguistic units present in the word. $unit\_symbol[i]$ specifies the symbol corresponding to the i^{th} linguistic unit of word, which is defined in *topology file*. i specifies all the transitions of the i^{th} state. n_to_states specifies the number of transitions starting from i^{th} state. $to\_unit[j]$ specifies index of the unit to which j^{th} transition occurs.