CHAPTER 7

ANALYSIS, SYNTHESIS / RECOGNITION OF EMOTIONAL BODO SPEECH WITH REFERENCE TO ITS PROSODIC FEATURES USING HIDDEN MARKOV MODEL (HMM)

7.1 Introduction

The aim and objective of the present study is the development of an Emotional Speech Synthesis & Recognition System for the Bodo language. The study also focuses on how emotional content is embedded in the acoustic signal produced by a speaker. The detection and labeling of the emotional content is carried out in such a way so as to maintain the emotion expressed by the speaker. The phonetical perspective is mainly taken into consideration in the present study. Here, the use of HMM is discussed to extract features from speech signals having emotions for its Synthesis and Recognition.

7.2 Use of HMM for Speech Synthesis / Recognition

In the 1970's, there has been successful implementation of the Hidden Markov Model (HMM) for the recognition and synthesis of Speech. State-clustered HMMs were used for the estimation of prosodic parameters for the selection of sub-word units in Speech Synthesizers using concatenative method. The data required by these types of synthesizers was large so as to train the set of decision trees in the HMMs. These were called trainable Speech Synthesizes. In HMM, the HMM state Segmentation is carried out by using the aligned training data from the State Clustered HMMs so as to define speech units for the selection of units [68].

HMM is basically a statistical based approach. Here the modeling of the speech variations is done using automatic statistical learning techniques. But, such statistical techniques have the disadvantage of taking priori modeling assumptions. These are likely to be inaccurate which will hamper the performance of the system. However, in recent times, new approaches have been developed to overcome the drawbacks of the conventional HMM [81]. This new technique is an improvement over the conventional statistical based HMM technique. In spontaneous speech which is highly co-articulated, this newly developed speech model [82] provided a rich structure for the partially observed dynamics (hidden) for the vocal tract resonances. This has the advantage of using unstructured Gaussian mixture components in large numbers to account for the large variation in the observed acoustic data.

7.3 HMMs in Recent Times

HMM technique has been highly used for both Speech Recognition and Synthesis systems. These kinds of systems use parameters of the probabilistic models and similar methods learning the probability distribution. HMMs are trained by optimizing the probability distribution of HMMs. It thus generates a sequence of speech feature vectors and sub-word units like phones. Emotional Speech Recognition / Synthesis is dependent on the estimation of parameters of speech generated from input text using HMMs [82]. The variations in Speech parameters like prosodic and noise variations are normalized away by statistical models that are used for speech recognition / synthesis. This is usually done because it is not considered important for the word and sub-word classification of units and the presence of these features might degrade the system performance. But, in recent times, Speech Recognizer / Synthesizer using

statistical methods like HMM tries to retain these prosody characteristics of speech like **Short Term Energy** (STE), **Zero-Crossing Rate (ZCR)**, **Fo Contour** etc, so as to reproduce the emotional speech. Contextual dependencies between the phone units play an important role in the speech quality of emotional synthesized speech [83]. These are detailed for short-term dependencies amongst the phone units generated by HMMs in emotional speech synthesis. In conventional HMMs, the duration model is used for speech recognition / synthesis which is not very adequate for speech synthesis. This is because the temporal structures of speech are not captured accurately in this model. Hence for Emotional Speech Synthesis using HMM, an improvement of the duration modeling has been suggested [84].

7.4 Hidden Markov Model (HMM): An overview

7.4.1 Structure of HMM

Hidden Markov model (HMM) is basically a Finite State Machine. It changes from state k to state j every time with each step. The state probability output distribution $b_j(o_t)$ is obtained as a vector of continuous observation o_t when it enters a state j for every time period t [85]. An observations sequence, $O = \{o_1,o_2, ...,o_T \}$, where T is the length, is obtained for the sequence of states, q = { q1,q2, ...,qT }. Hence HMM, say λ, which is defined by making use of the transition probabilities which is generated as the Finite State machine from one state k to another state y, a_{ky}. This is done by taking into consideration the probability distribution for state, $b_j(o)$, and the probabilities, π_j for the initial state.

An example of a left-to-right HMM having 3-state is shown in Fig 7.1. This type of models is usually chosen in applications for Emotional Speech Synthesis. In the

present study, HMMs taken are cosidered as left-to-right HMMs. In a left-to-right HMM model , $a_{xx} + a_{xj} = 1$, where a_{xx} is the probability of the HMM remaining in the same state x. The parameter constraints are as follows:

$$\sum_{j=1}^{N} \pi_j = 1 \qquad \text{----------(7.1)}$$

$$\int_{-\infty}^{\infty} b_j(o)do = 1 \qquad \text{----------(7.2)}$$

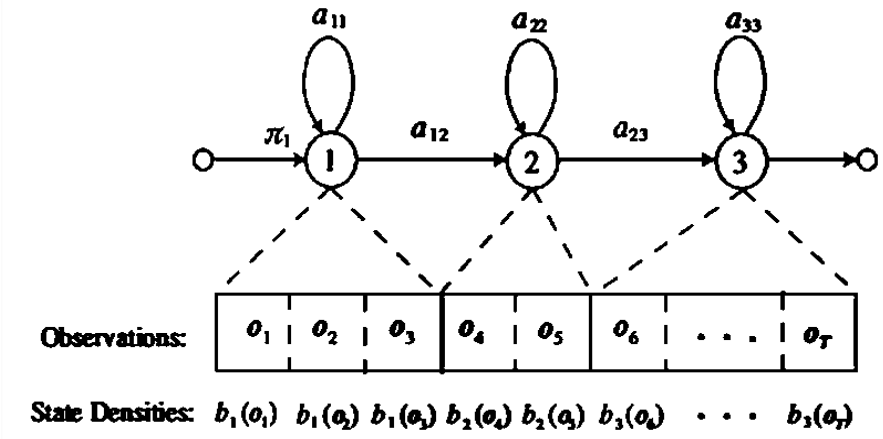


Fig 7.1: A 3 state left-to-right Hidden Markov Model with observation sequences.

7.4.2 Probability Distribution Output

In HMMs having continuous probability distribution, generally the probability distribution, say $b_j(o)$ is modeled by using K-mixture of Gaussian distributions as formulated below:

$$b_j(o) = \sum_{k=1}^{K} r_{jk} \mathcal{N}(o, m_{jk}, U_{jk}) \qquad \text{……(7.3)}$$

$$\mathcal{N}(o, m_{jk}, U_{jk}) = \frac{1}{\sqrt{(2\pi)^L |U_{jk}|}} exp\left(-\frac{1}{2}(o - m_{jk})^\top U_{jk}^{-1}(o - m_{jk})\right), \qquad \text{…….. (7.4)}$$

Where, the notations used are as follows:

m_{jk}, r_{jk} and U_{jk} : the weight of the mixture,

S(S is the dimension of o) : A vector of mean dimension.

and LXL : a covariance matrix (The matrix is made up of k components and state j .

$|U_{jk}|$: represents the determinant of U_{jk}.

Here, the elements from the observation vector which are continuous, i.e O are assumed to be independent. A single Gaussian (K = 1) can be used. The full covariance matrix i.e a **diagonal covariance matrix** is used which is restricted to its diagonal elements. For HMMs having N-state, the mixture weights tends to satisfies the following stochastic constraint [86] as under :

…….(7.5)

$$\begin{cases} \sum_{k=1}^{K} r_{jk} = 1, & 1 \le j \le N \\ r_{jk} \geqslant 0, & 1 \le j \le N, \quad 1 \le k \le K, \end{cases}$$

7.4.3 HMM – Assumptions used

The operations of HMM are done based on the assumptions as per the following conditions [87]:

(i) A state is statistically independent of all other states except the previous state,

(ii) An acoustic observation is statistically independent of all other observations except the state generating it.

Given the model, **λ**, the assumption (i) is used for computing of the probability for the state sequence of HMM, **q** = $\{q_1, q_2, ..., q_T\}$,. This is obtained by multiplication of the state transition probabilities as stated below [88]:

$$P(q|\lambda) = \prod_{t=1}^{T} a_{q_{t-1}q_t},$$

……(7.6)

For the given HMM $\boldsymbol{\lambda}$ and state sequence **q,** the initial state probability is represented by $\mathbf{a_{q0q1}}$, or as $\boldsymbol{\pi}_{q1}$. The probability of an observation sequence, O = (o_1,o_2, ...,o_T), for the observation independence assumption (ii), is computed by multiplying the output probabilities of each state as shown below:

$$P(O|q,\lambda) = \prod_{t=1}^{T} b_{q_t}(o_t) \qquad \text{---------- (7.7)}$$

7.4.4 HMMs Duration Modelling

There are no explicit duration models in conventional HMMs. The temporal structure for the continuous observations, say, O can be modeled implicitly by the use of transition probabilities. The exponential probability distribution for each state i, from the model structure, is estimated as follows [89]:

$$p_i(d_i) = a_{ii}^{d_i - 1}(1 - a_{ii}), \qquad \text{--------- (7.8)}$$

Here, $\mathbf{a_{ii}}$ denotes the self-transition probability of the state and $\mathbf{d_i}$ denotes the number of consecutive observations for the state duration in the state i. In all HMMs, some basic unit is associated with the duration models. For example, if the unit under consideration is a phoneme, then it is called a **phoneme-based duration model**. The prior probability for the sequence of all the states q = {q1,q2, ...,qT } is computed as given below[90]:

$$P(q|\lambda, T) = \prod_{i=1}^{N} p_i(d_i), \qquad \text{……(7.9)}$$

Here, the constraint is represented as :

$$\sum_{i=1}^{N} d_i = T, \qquad \text{……(7.10)}$$

Here, the total number of states is N and the total length of the sequence of states is T.

7.4.5 Estimation of Observation Probability in HMMs

7.4.5.1 Optimization Problem in HMMs

Given the model **λ,** one of the most common problem in HMMs is the computation of P(O|**λ**), i.e. the probability of the continuous observation sequence in HMM. This problem is encountered in the decoding section of a speech recognizer. The probability of the observation sequence of every unknown word can be estimated for every word model. The word model, which maximizes the given criterion, is selected. For the model **λ**, the underlying state sequence is hidden, although the continuous observation sequence, O is known. Therefore, assuming that the model is **λ,** the subsequent probability of the observations is estimated by summing all the possible state sequences q as follows [91]:

$$P(\boldsymbol{O}|\lambda) = \sum_{\boldsymbol{q}} P(\boldsymbol{O},\boldsymbol{q}|\lambda), \qquad \text{------- (7.11)}$$

Here, for the model **λ**, the probability that the observation sequence O is generated is P(O,q|**λ**), and q is the state sequence through which it moves. This joint probability [92] is estimated by using the Bayes' theorem and the statistical independence assumptions as given in (7.6) and (7.7). It is stated as follows:

$$P(\boldsymbol{O},\boldsymbol{q}|\lambda) = P(\boldsymbol{O}|\boldsymbol{q},\lambda)P(\boldsymbol{q}|\lambda) = \prod_{t=1}^{T} b_{q_t}(\boldsymbol{o}_t)a_{q_{t-1}q_t} \qquad \text{----- (7.12)}$$

Here P(O| **λ**) can be estimated from the equations (7.11) and (7.12) as follows:

$$P(\boldsymbol{O}|\lambda) = \sum_{\boldsymbol{q}} \prod_{t=1}^{T} b_{q_t}(\boldsymbol{o}_t)a_{q_{t-1}q_t} \qquad \text{------(7.13)}$$

However, the above stated equation is not very practical to use because of high computational demand. The forward-backward algorithm, a recursive algorithm, is found to be an effective method to estimate P(O|**λ**). The probability P(O|λ) can be

estimated by calculating the optimum sequence of states, q*. This in turn, maximizes P(q,O|λ). The Viterbi algorithm [93] can be efficiently used for solving the problem of maximization of P(q,O|λ) and also to compute q*.

7.4.5.2 HMMs and the Viterbi Algorithm

For the model **λ,** given an observation sequence O, the optimum state sequence, q* is computed using the Viterbi algorithm [94] as follows:

$$q^* = \arg\max_{q} P(q, O|\lambda)$$

------- (7.14)

The optimum state sequence is computed by making use of the following recursion :

$$\delta_{t+1}(j) = b_j(O_{t+1}) \max_{1 \leq i \leq N} \{\delta_t(i) a_{ij}\},$$

------ (7.15)

Here the constraint is denoted by :

$$\delta_t(i) = \max_{Q_t} P(q_t = i, O_t|\lambda)$$

----- (7.16)

Thus, along a single path $Q_t = \{q_1, q_2, ..., q_t\}$, this depicts the maximum probability of the partial observations sequence $O_t = \{o_1, o_2, ..., o_t\}$. This means that it finishes at time t in the state i. In the starting state, the initialization is carried out as $\boldsymbol{\delta_0(i)} = 1$. All the other states are assumed to be zero. If the sum total of all the states is taken to be N for the model **λ,** then $1 <= i <= N$. The maximization argument noted for each iteration is given below [95]:

$$\Psi_t(j) = \arg \max_{1 \leq i \leq N} \delta_t(i-1) a_{ij}$$

-------- (7.17)

The probability of the most likely path to be taken at the ending part of the induction process is computed as below:

$$P^* = \max_{1 \leq i \leq N} \delta_t(i)$$

------- (7.18)

By making use of the backtracking method, the best state sequence is estimated which is given as [96]:

$$q_T^* = \arg \max_{1 \leqslant i \leqslant N} \delta_T(i) \qquad \text{------ (7.19)}$$

$$q_t^* = \Psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \cdots, 1 \qquad \text{----- (7.20)}$$

7.4.6 Estimating the Model Parameter of HMM

7.4.6.1 HMM-Optimization Problem

Estimating the parameters of the model, which will best describe the observation sequence, is perhaps the second most important problem of HMM.

This is estimated as below:

$$\lambda^* = \arg\max_{\lambda} P(O|\lambda) = \arg\max_{\lambda} \sum_{q} P(O,q|\lambda) = \arg\max_{\lambda} \sum_{q} P(O|q,\lambda)P(q|\lambda) \qquad \text{------ (7.21)}$$

For the HMM model $\boldsymbol{\lambda}$, for a certain optimization criterion, it is difficult to estimate because both the parameters $\boldsymbol{\lambda}$ as well as q are not known for a certain optimization criterion which will globally maximizes P(O|$\boldsymbol{\lambda}$), like the maximum likelihood. For the HMM model $\boldsymbol{\lambda}$, the parameters are estimated by finding the solution that optimizes P (O|λ). The **Expectation-Maximization** Algorithm (EM) better known as the **Baum-Welch algorithm**, is generally used to find the answer [90] for this problem. In the synthesis/recognition of speech, that part that deals with the training of HMMs is considered the procedure for optimization for a given sequence of words, say Z = {z_1, z_2, ..., z_S}. The equation given in (7.21) can also be written as:

$$\lambda^* = \arg\max_{\lambda} P(O|Z,\lambda)$$

------ (7.22)

The Text analysis procedure is normally used to designate contextual factors for the word sequence Z. It is also used to map them into a sequence of sub-word units which is context dependent, for example sequence of phones etc. All the different context-dependent HMMs are used for each context-dependent unit. Examples of such HMMs are a diphone or a triphone HMM. These context-dependent factors discussed above are generally associated with **accent, stress, part-of-speech** etc. In the training part of HMM, the observation sequences are segmented into a number of states and the phonetic aspect under consideration for the model, are initialized. The initial model may be initially created from a different set of speakers. It can also be obtained from the uniform distribution of each of the words under consideration into states. By making use of the Viterbi algorithm, the best state sequence is found by performing the segmentation [91].

7.4.6.2 Baum-Welch Algorithm

Baum-Welch algorithm [97] is used to maximize the auxiliary function of current model **λ'** and the new **λ**. It is used for the estimation of the HMM parameters as:

$$A(\lambda',\lambda) = \sum_{q} P(O,q|\lambda') \log P(O,q|\lambda)$$

------(7.23)

A(**λ'**, **λ**) is said to maximized over to **λ** to improve **λ'.** It is maximized as the increase in the likelihood for the model **λ** i.e P(O,q|**λ**). The parameters assumes for the model are as follows:

π_i : the initial i-th state probability,

a_{ij} : the transition probabilities, and

$\mathbf{r_{jk}}$, $\mathbf{m_{jk}}$ and $\mathbf{U_{jk}}$: the coefficients from the mixture density function of equation(7.4). Thus, in a model $\boldsymbol{\lambda}$, the sequence of hidden states P(O,q| $\boldsymbol{\lambda}$) can be obtained from equation (7.12) as follows [98] :

$$\log P(\boldsymbol{O},\boldsymbol{q}|\boldsymbol{\lambda}) = \log \pi_{q_0} + \sum_{t=1}^{T} \log a_{q_{t-1}q_t} + \sum_{t=1}^{T} \log b_{q_t}(o_t) \qquad \text{-----(7.24)}$$

The formulas to calculate the HMM parameters can be obtained by using the equations (7.23) and (7.24). Thus the formulae for the initial and the transition probabilities can be re-estimated as follows:

$$\pi_i = \frac{\alpha_0(i)\beta_0(i)}{\sum_{j=1} N \alpha_T(j)} \qquad \text{------(7.25)}$$

$$a_{ij} = \frac{\sum_{t=1}^{T} \alpha_{t-1}(i) a_{ij} b_j(o_t) \beta_t(j)}{\sum_{t=1}^{T} \alpha_{t-1}(i) \beta_{t-1}(i)}, \qquad \text{------(7.26)}$$

Here, $\boldsymbol{\alpha}_t$(i) : probability of occurance of partial observation sequence at time t for 1 to t.

and $\boldsymbol{\beta}_t$(i) : probability of the partial observation from t to T,

i.e.

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i|\boldsymbol{\lambda}) \qquad \text{-----(7.27)}$$

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T | q_t = i, \boldsymbol{\lambda}) \qquad \text{-----(7.28)}$$

The **forward-backward algorithm** is used for recursive computation of the forward and the backward probabilities.

Here, $\boldsymbol{\alpha}_t$(i)- forward probability and

$\boldsymbol{\beta}_t$(i)- backward probability.

Therefore,

$$\alpha_1(i) = \pi_i b_i(o_1), \qquad 1 \leqslant i \leqslant N \qquad \text{------ (7.29)}$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), \qquad 1 \leqslant t \leqslant T-1$$

---- (7.30)

$$1 \leqslant j \leqslant N$$

----- (7.31)

$$\beta_T(i) = 1, \qquad 1 \leqslant i \leqslant N$$

----- (7.32)

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \qquad t = T-1, T-2, \ldots, 1$$

--- (7.33)

$$1 \leqslant i \leqslant N$$

----- -(7.34)

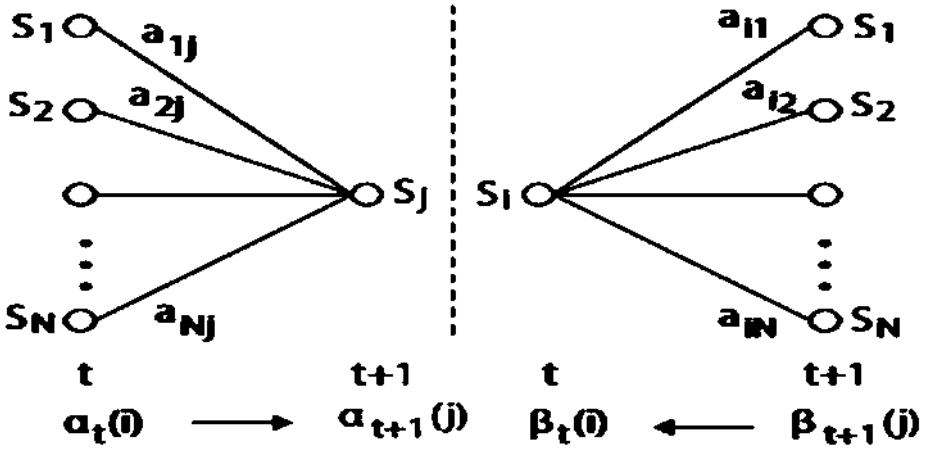


Fig 7.2: Diagram showing estimating forward, $\alpha_t(i)$ and backward, $\beta_t(i)$ probabilities

Fig (7.2) depicts the calculation of $\alpha_{t+1}(i)$ as well as $\beta_t(i)$. Hence, the probability of attaining a particular state S_j in the time (t +1) is obtained by the summation of $\alpha_t(i)a_{ij}$.

This is true for all possible states, N i.e S_i at time t, with 1<=i<=N. Hence, forward probability $\alpha_{t+1}(j)$ is obtained by multiplying the probability of the observation with this sum at state j. The forward probability i.e $\alpha_{t+1}(j)$ is also computed for all the states of j for each time t.

In a similar manner, the backward probability, β_t (i) is also computed by obtaining the sum of all the products for transition probability for all the states.

The output probability distribution parameters is computed from the equations (7.7) and (7.4) by making use of the forward and backward variables. The re-estimation formulae of maximum likelihood for these parameters are [98]:

$$r_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{K} \gamma_t(j,k)} \quad \text{---- (7.35)}$$

$$\boldsymbol{m}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \quad \text{----- (7.36)}$$

$$\boldsymbol{U}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (\boldsymbol{o}_t - \boldsymbol{m}_{jk})(\boldsymbol{o}_t - \boldsymbol{m}_{jk})^{\top}}{\sum_{t=1}^{T} \gamma_t(j,k)}, \quad \text{---- (7.37)}$$

Here k : is the mixture factor of $P(o_t)$ &

γ_t(j, k) : probability of factor k existing in state j at given time t

observation sequence : $o_{t+1}, o_{t+2}, \ldots, o_T$

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \right] \left[\frac{r_{jk} \mathcal{N}(\boldsymbol{o}_t, \boldsymbol{m}_{jk}, \boldsymbol{U}_{jk})}{\sum_{k=1}^{K} \mathcal{N}(\boldsymbol{o}_t, \boldsymbol{m}_{jk}, \boldsymbol{U}_{jk})} \right] \quad \text{---- (7.38)}$$

7.5 Data preparation for the Proposed Emotional Recognition / Synthesis System

Data preparation is the primary stage for an Emotional Speech Recognizer / Synthesizer using HMM. Emotional Speech data is required for the training, testing and generating the speech by HMM. For building the system, emotional speech is recorded from the Bodo scripts that are prepared. Scripts are necessary for prompting for the Bodo sentences i.e they act as the test data. Also, the scripts will act as references with which the recognizer / Synthesizer's performance will be evaluated. The task grammar

used as a random generator is an efficient way to create them. To initiate the HMM training process, the training data is prepared in conjunction with a pronunciation dictionary to generate the initial level phonetic transcriptions. Hence, good phonetic coverage and variation has been added to the data needed for the application. The training data is used as the proposed system is developed. The test data gives the reference transcriptions with the help of which the performance of the proposed emotional recognition / synthesis system is measured. The proposed model is shown as below:

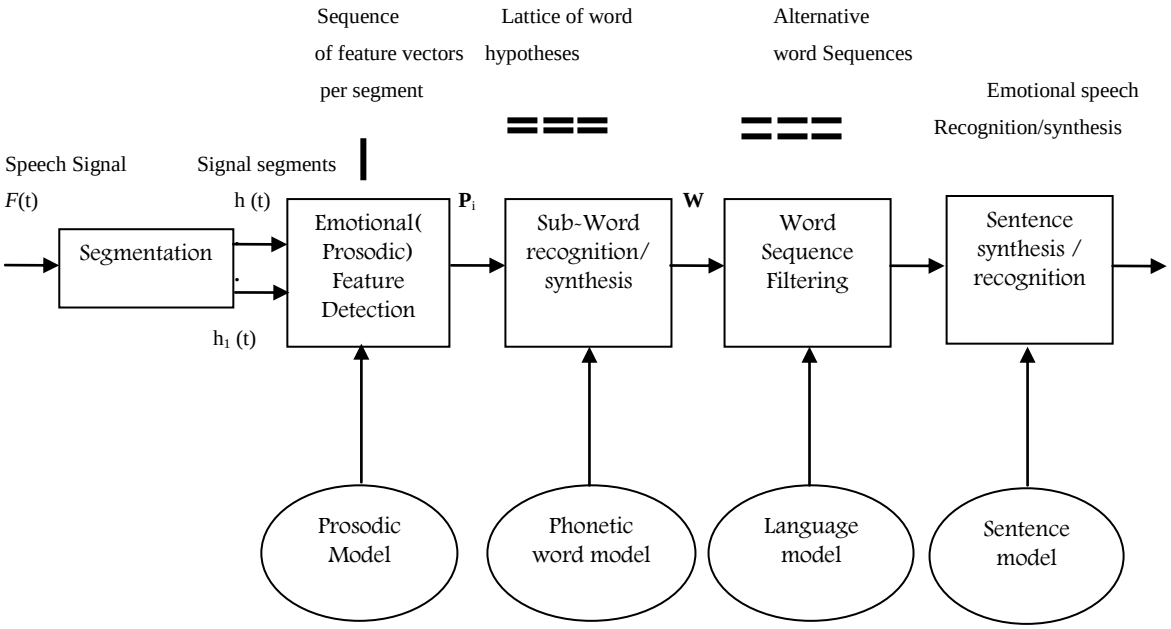


Fig 7.3: Proposed Model for Emotion Speech Synthesis / Recognition for the Bodo language.

7.5.1 Preparing the task Grammar

The constraints to be applied on the input data of the recognizer / synthesizer are defined by the task grammar. For example- the input data has to be a string of digits if the application is telephone handling system. In the present study, 500 sentences with 3

different emotions are considered. The grammar in use is defined in Bracket-Normal i.e BN-form. It is as follows: **$**variable denote a phrase between the subsequent = symbol and the semicolon. Here | stands for a logical or. Brackets are used to do the usual grouping function. Square brackets are used to denote optional. An example is given below for the following three sentences:

(i) सा-सा नाइ (ii)जेव-जेव खां (iii) जेव-जेव नु

Task Grammer: $words= सा | सा | नाइ || जेव | जेव | खां || जेव | जेव | नु ||
(SENTENCE-START[**$words**]SENTENCE-END)
$ phonemes = सा | सा | ना इ || जेव | जेव | खां || जेव | जेव | नु
(SENTENCE-START[**$phoneme**]SENTENCE-END)

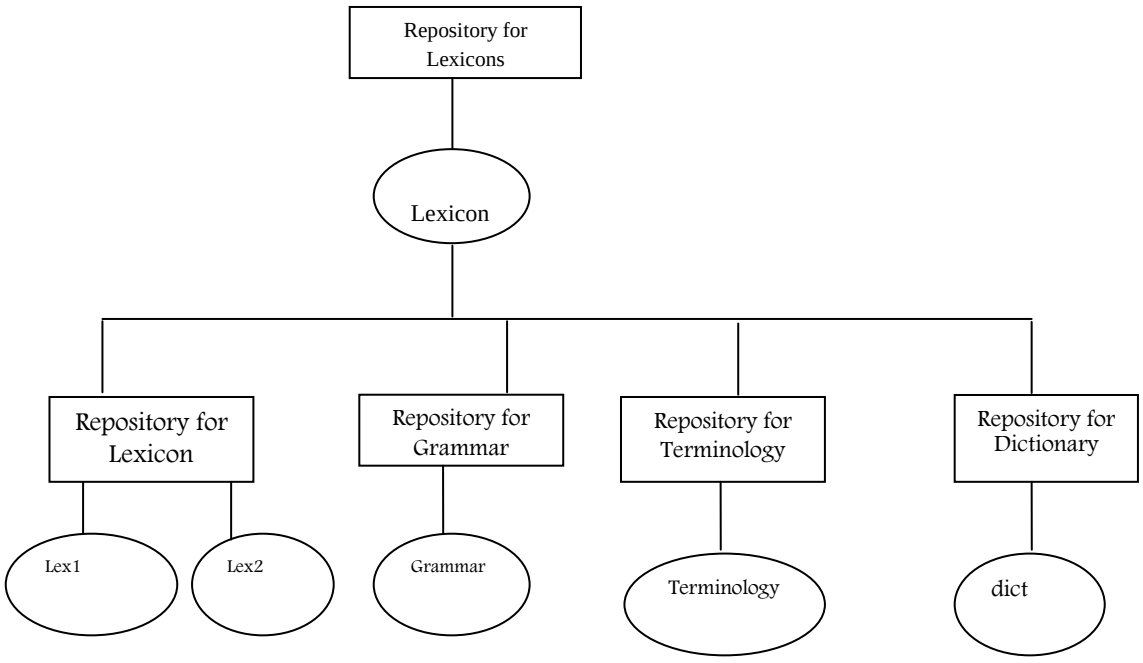


Fig 7.4: High Level Representation of task Grammar for voice Word-Phoneme network for Bodo language.

Given, above is the high-level representation of a task grammar is shown in Fig 7.4. A word network has to be defined because the HTK recognizer actually requires it for its functioning. Thus, a low level notation which is called the HTK **Standard**

Lattice Format (SLF) is built up. In this lattice, every instance of the words and every transition of the words has been explicitly listed. Thus, by using the HParse tool, the word network can be created from the grammar automatically by assuming that the above grammar is contained in the file **pran** as below:

HParse – A –D -T 1 pran.txt wordnet.slf

An equivalent word network is created in the file wordnet, as shown in fig below:

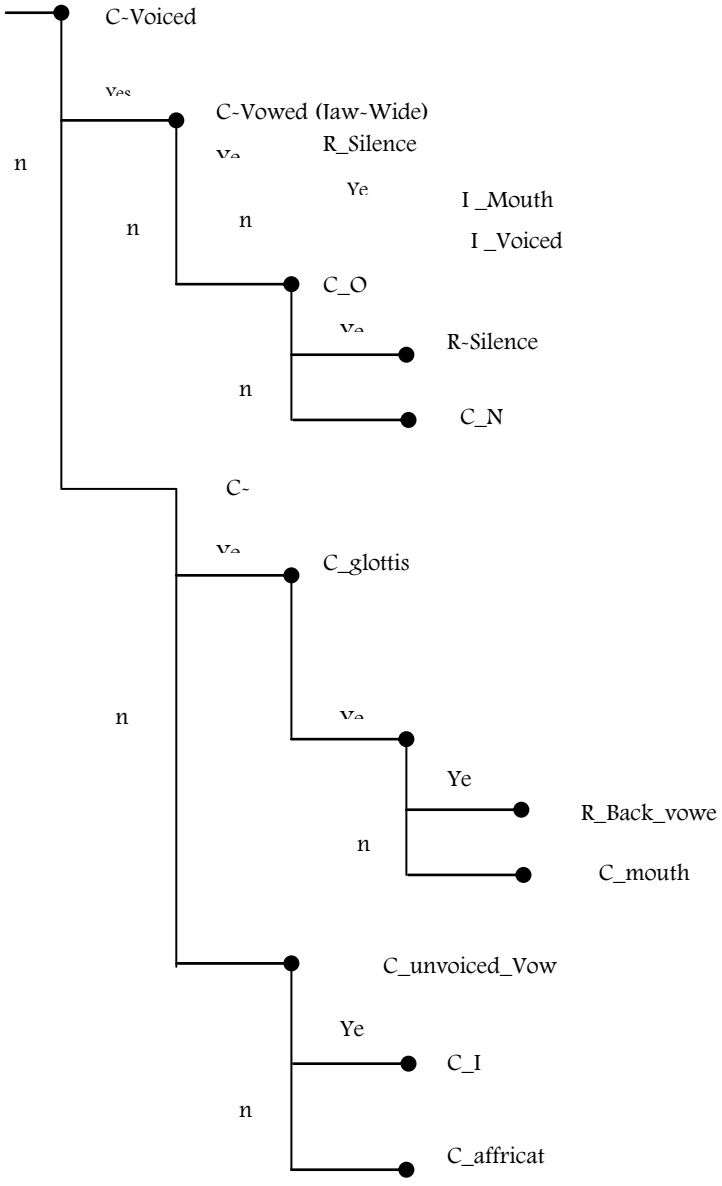


Fig 7.5: Diagram showing creation of a word lattice

Thereafter, random sentences can be generated with the help of the above created lattice by the use of HTK tool HSGen, which are later used for training and testing purposes.

7.5.2 Pronunciation Dictionary

A large pronunciation dictionary is required for training the HMM network. The pronunciation dictionary is associated with words that are used in the task grammar and the acoustic models. It may be comprised of sub word (phonetic, syllabic etc.) units. The present research work provides a voice operating interface. Hence, our motive is to costruct a dictionary for a large vocabulary for the Recognition/ Synthesis system for Bodo language.

7.5.3 Procedure for Data Recording and labeling

HTK tool HSLab was used for training and testing the data. This toolkit is capable of simultaneously recording and labeling the data. HMM is estimated for all the words after the training and testing the data process is over. For training and testing the emotion recognizer / synthesizer on the selected domain, 100 phonetically i.e emotionally rich Bodo sentences were automatically generated using the grammar with HTK's HSGen. Since phoneme duration was the requirement of the toolkit for training the sentences, the differences in timing of the pronunciation of the sentences used for training is important. Then the toolkit learns to recognize/synthesize the words in a sentence with different kind of emotions through fitting of the word transcriptions on to the training set. These transcriptions are also used for the training of the same sentences. This is because there might be variation between different speakers relative to the

transcriptions. The training corpus consisted of 500 sentences. These were recorded and labeled using the HTK tool HSLab.

After the preparation and labeling of the training sentences was done, a test corpus of Bodo language was also prepared in the same way as the training corpus. The difference that was noticed in the pronunciations between speakers was categorized as articulation variation with respect to different emotions. This problem of phonetic changes was encountered by using isolated whole word / sentence models with different emotions and also using many different sentences for a speaker independent system. Variation in articulation is another problem for speech emotion recognition / synthesis. However, if there was is variation in articulation, speech recognition will become an instance based learning problem.

7.5.4 Phonetic Transcription of Data

In the procedure of the training of HMMs, every file of the data used for training must have an associated phone-level transcription. Since there is lack of hand labeled data for bootstrapping a set of models, a flat-start scheme is used instead. For this, two sets of phone transcriptions are needed. Initially, the set used will have no short-pause (sp) between the words. However, once reasonable phone models have been created sp will be inserted between words. This will take care of any pause that was introduced by the speaker. The start point used for both sets of phone transcription is an orthographic transcription that is in HTK label format. This can be generated using a scripting language or a text editor. The script that has been provided in the HTK Tutorial directory is **prompts2 mlf**. The effect is to be able to convert the prompt utterances.

The word.mlf thus generated is as follows

#! MLF !#

“data/train/as02.wav” बियो गांसे बिजाब फरायो

SIL

बियो

SIL

गांसे

SIL

बिजाब

SIL

फरायो

“data/train/as03.wav”

Since the main objective of the present research is to create an isolated phoneme / word / sentence emotion recognizer/synthesizer, a file called **source1.mlf** was created which was associated with each recorded and labeled speech data with a word by using the command as

HLed-1 ‘/data/train/’-d dict –I phones1.mlf mkphones1.led words.mlf

#!MLF!#

“data/train/as02.wav” बियो गांसे बिजाब फरायो

बि

यो

SIL

गां

से

SIL

बि

जाब

SIL

फ

रा

यो

SIL

“data/train/as03.wav”

For obtaining the transcriptions of the model, an HTK edit script called ‘**mkphones1.led**’ is generated using the following commands:

EX

IS sil sil

DE sp

Here the commands are as follows:

(i) expand EX : replaces each word in word.mlf by its corresponding pronunciation in the dictionary file dict.

(ii) IS: inserts a silence sil at the start and end of each utterance and

(iii) delete DE : deletes all unwanted short-pause sp labels in the transcription labels.

7.5.5 Data Coding for Synthesis / Recognition

The tools used for speech recognition / synthesis cannot process directly on speech waveforms. These waveforms have to be presented in a more compact and efficient manner. This process is called the “**acoustical analysis**”. Here, the signal is segmented in successive frames (the length of each frame is typically chosen to be between 20ms and 40ms), and they are overlapping with one another. Each frame is then multiplied by some windowing function like Hamming function. The spectral properties of the frame are given a compact representation in the form of a vector of acoustical coefficients which is extracted from each of the windowed frame.

A configuration file, for example, **config.Txt** was created to specify the nature of the audio data (like format, sample rate, etc) and feature extraction parameters (like type of feature, pre-emphasis, window length etc) to HTK. It was as follows:

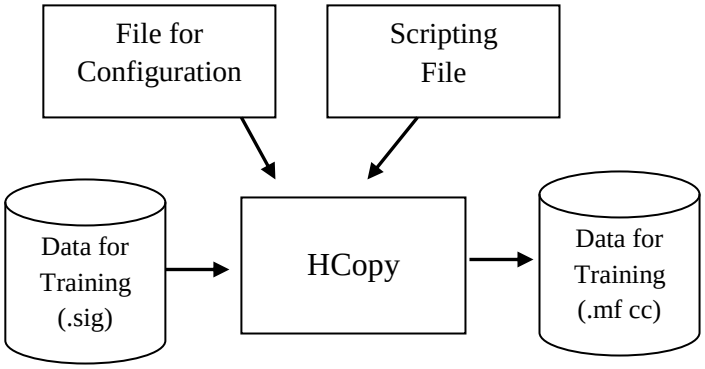


Fig 7.6: Diagram showing training of data for HMM

#coding parameters

SOURCEKIND = waveform

SOURCEFORMAT = HTK

SOURCERATE = 625

TARGETKIND = MFCC_0_D_A

TARGETRATE = 100000.0

SAVECOMPRESSED = T

SAVEWITHCRC = T

WINDOWSIZE = 250000.0

USEHAMMING = T

PREEMCOEF = 0.97

NUMCHANS = 26

CEPLIFTER = 22

NUMCEPS = 12

ENROMALISE = F

In the configuration file for all the frame:

- NUMCEPS is initialized as 12, i.e the first twelve Mel Frequency Cepstral Coefficients
- The total energy of the frame is proportional to C0 i.e the first null Mel coefficients
- The 1st order derivatives i.e 1st prosody in [c0, c1….c12] is estimated by the thirteen "Delta coefficients".
- The 2nd order derivatives i.e 2nd prosody in [c0, c1….c12] is estimated by the thirteen "Acceleration coefficients".

Thus, a total of 39 vector coefficients are taken out from all the frames.

The conversion from the original waveform to a series of acoustical vectors is done with the HCopy HTK tool:

HCopy – A –D –C Config. Txt-S train.scp

Config.txt is a configuration file setting the parameters of the acoustical coefficient extraction.

Train.scp specifies the name and location of each waveform to process, along with the name and location of the target coefficient files.

To run Hcopy a list of each source file and its corresponding output file is needed. The first few lines look like:

Data/train/as01.wav data/mfc/as01.mfc

Data/train/as02.wav data/mfc/as02.mfc

Data/train/as03.wav data/mfc/as03.mfc

Data/train/sil10.wav data/mfc/sil10.mfc

One line for each file in the training set. This file tells HTK to extract features from each audio file in the first column and save them to the corresponding feature file in the second column.

7.5.6 Prototype Model for HMM

The first step in HMM training is to define a prototype model. The purpose of the prototype is to define a model topology on which all the other models can be based. In HTK a HMM is a description file e.g.

~o

< VectSize > 39 < MFCC_0_D_A >

~h “b”

<BeginHMM>

<NumState> 6

<State>2<Mean>39

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

< Variance > 39

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

<State> 3

<Mean>39

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

<Variance> 39

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

<State>4

<Mean> 39

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

<Variance> 39

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

<State> 5

<Mean>39

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

<Variance> 39

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

<Trans P>6

0.0 0.4 0.3 0.3 0.0 0.0

0.0 0.0 0.4 0.3 0.3 0.0

0.0 0.0 0.0 0.4 0.3 0.3

0.0 0.0 0.0 0.0 0.5 0.5

0.0 0.5 0.5 0.0 0.0 0.0

0.0 0.0 00. 0.0 0.0 0.0

<EndHMM>

~ o <Vec Size>39 <MFCC_O_D_A>

- It gives the size of the vector coefficients i.e 39 coefficients.

<Begin HMM> (……) < EndHMM >

- It denotes the description of every HMM Models.

<NumStates>6

- It generates the total number of states, including two non-emitting that is utilized in HMM.

<State> 2

- The observation function for state 2 is shown here. For the present study, a single – Gaussian observation function with diagonal matrices is chosen. Such functions are explained entirely by the use of a mean vector and a variance vector [99].

<Mean> 39

0.0 0.0 (…..) 0.0 (x 39)

- For the current observation function, it yields the values of mean vector which has 39 dimension observation spaces. At the beginning, value of all the elements is considered to be zero. The HMM " prototype" is given here.

  <Variance> 39

  1.0 (……) 1.0 (1 x 39)

- The estimate of variance vector for the present observation function is given. The value for all the elements is initialized at 1.

  <TransP>

- This results in a (6 x 6) transition matrix. Here, the probability of reaching state j from state i is given by a_{ij} and state "Null" signify that transition will not take place. The rest of the values in the matrix are initialized in such a way that the row summation is one. When the HMMs are being trained, these values are changed. These prototype models are built for every utterance in the various emotional states. Models for each of the events were also constructed.

In a given HMM, the HTK tool **HCompV** will filter a set a data files, calculate the global mean and variance and then set all of the Gaussians to have the same mean and variance. Therefore, after assuming that a list of all the training files has been stored in train.scp, the following command is used.

HCompV – C config –f 0.01 –m –s train.scp –M hmm1 proto

This will create a new version of proto in the directory hmm1 in which the zero means and unit variances above have been replaced by the global speech means and variances.

Given this new prototype model stored in the directory hmm1, a Master Macro File (MMF) called hmmdefs containing a copy for each of the required HMM monophones is constructed by manually copying the prototype and re-labeling it for each of the required monophone (including "Sil"). The format of an MMF is same to that of an MLF. This is because it serves the same purpose i.e avoiding a large number of individual HMM definition files.

7.5.7 HMM TRAIING

The entire recognition / synthesis system is designed using the HTK toolkit. Here the commands used are as follows:

i. (HInit ---- -used to initialize the data,
ii. HCompV --- used to model the unseen factors of data,
iii. HRest ---- used for re-estimation of the training models,
iv. HERest---- used for embedded re-estimation of the HMM to build the entire system.
v. HVite ---- used for recognition purpose once building is done and
vi. HResults ---- used to check the performance of output.

7.5.7.1 Data Initialization

The data used for the training purpose was initialized using HTK tool HInit as given below:

HInit –A –D –T 1 –S train.scp –M model/hmm1 – H hmm file –l label – L label dir name of hmm where:

Name of **hmm** is the name of the HMM to initialize. **Hmm** file is a description file containing the prototype of the HMM called name of **hmm** (here: hmm as, **hmm Bodo**, e.t.c.) **trin.scp** gives the complete list of the .mfcc files forming the training corpus (corresponding to the training corpus (here: data/train/lab/). Label indicated which labeled segment must be used within the training corpus (here: as, Bodo, etc.. model/hmm1 is the name of the directory which is created beforehand) where the result of initialized HMM description will be sent as output. This procedure is to be repeated for each model. The HMM file-output by using **HInit** has the same name as the input prototype. E.g. **Hinit –A –D –T 1 –S train.scp –M Model/hmm0 – H hmm 1.txt -1 as –L data/train.** This process has been repeated for all the models. The models to the training data was initialized using the HTK tool **HCompV** as follows.

HCompV –C config.txt –f 0.01 –m –S train.scp –M hmm1 proto.txt

However, HCompV was not used for initializing the models as it was already initialized with HInit). **HCompV** generates a file called **vFloors**. It contains the global variance vector multiplied by the factor 0.01. The values stored in "**variance floor macro**"(varFloorl) are utilized later during the training process as the floor values for the vectors of estimated variance. This results in the generation of two files – (i) proto and (ii) vFloors - in the directory hmm1. These files were used in the following way:- When an error occurs at the point which rearranges the order of the parts of the MFCC 0 D A label as MFCC D A 0. This was however corrected. The first three lines of the proto were cut and pasted into vFloors. This was then saved as macros.

7.5.7.2 Training the HMM

The HTK tool HRest was used for estimating the optimal values for the parameters of HMM like transition probabilities, mean and variance vectors for each of the observation function.

The following command was used to generate re-estimated values:

HRest -A - D -T 1 -S train12.scp -M model/hmml -H vFloors -H model/hmmO/hm˜m 1.txt -11 as -L data/train as1.

train12.scp gives the complete list of the .mfc files which form the training corpus and is storeded in the directory data/train/mfc. **Model/hmml** is the output directory that indicates the index of the current iteration. vFloors is the file which contains the variance floor macro obtained with HCompV. **hmm 1.txt** and is the description file of the HMM called as1. It is stored in a directory whose name indicates the index of the last iteration (here model/hmm0). -1 as is an option that indicates the label that is used within the training data.

Data/train is the directory where the label files (.lab) corresponding to the training corpus as1 is the name of the HMM for trainig. This procedure is repeated many times for each of the HMMs which is to be trained. Each time, the HRest iterations (i.e. iterations within the current re-estimation iteration) is displayed on the screen, it indicates the convergence through the change measure. As soon as this measure does not fall/decrease (in absolute value) from one HRest iteration to another, it's time to stop the process. In the present study, 3 re-estimation iterations are used. The final word HMMs are then repectively hmm3/hmm 1, hmm3/hmm 0, and hmm3/hmm sil etc. A file called hmmdefs.txt

was created by grouping all the HMM's into one file. After each iteration an error occurred. It rearranges the order of the parts of the MFCC_0_D_A label as MFCC _ D _ A _0. This was then consequently corrected with every iteration. The HMM re-estimation iterations is shown with the help of the following figure.

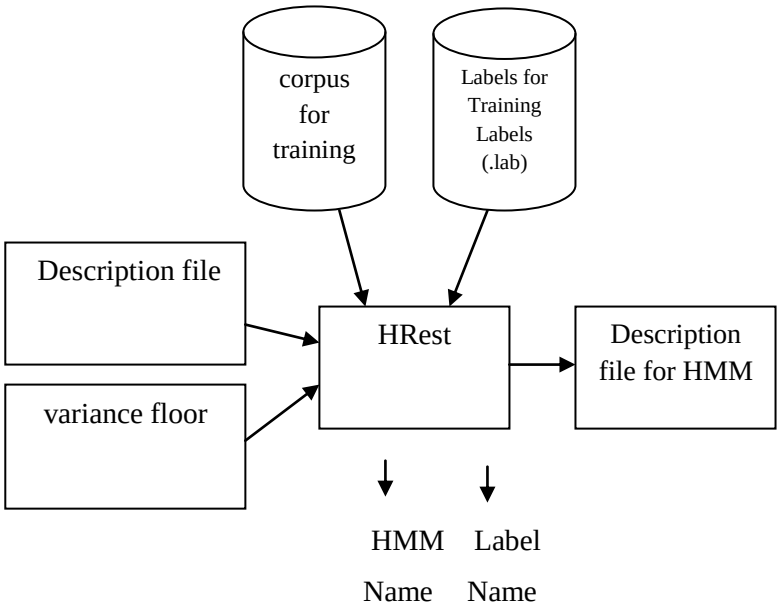


Fig 7.7: Representation of re-estimation iteration for HMM

7.5.7.3 Embedded Training of HMM

HERest is used to operate on HMMs with initial parameter values which are estimated by HInit/HRest. HERest can work with discrete and tied-mixture HMMs, multiple mixture Gaussians, multiple data streams, parameter tying within and between models, and full or diagonal covariance matrices

For using HERest, it is however necessary to construct a file containing a list of all HMMs in the model set with the name of each model being written on a separate

line. The model names in this list must correspond to the labels used in the transcriptions. Also, there must be a corresponding model for each distinct transcription label. HERest is used by a command line of the form

HERest -S trainlist1 -I labs -H dirl/hmacs -M dir2 hmmlist

where hmmlist1 contains the list of HMMs. HEREST, on startup will automatically load the HMM master macro file (MMF) .It will then search for the definition of each HMM which are listed in the hmmlist. If any HMM name is not found in the master macro file, it generally attempts to open a file by the same name as in the current directory. However, in large sub-word systems, all of the HMM defined are stored in MMFs. Similarly, all the transcriptions that are required will be stored in one or more Master Label Files (MLFs). Here, in this example, they are stored in the one single Master Label File called **labs**.

After all the MMFs and MLFs have been loaded, each file in the trainlist is processed by HEREST. It thus accumulates the required statistics. After it is completed, output to the directory dir2 is an updated MMF. However, If a second iteration is necessary, then HEREST is again invoked. It reads in the MMF from dir2 and outputs it to a new file to dir3, and so on.

7.6 Synthesis / Recognition process

The recognizer / synthesizer is now almost complete and therefore its performance can be analyzed. The recognition / synthesis network and dictionary have been constructed. The test data has also been recorded. Thus, the recognizer / synthesizer can now start functioning.

By using the HTK tool, HCopy, the input speech signal (speech with prosodic features) is first of all transformed into a series of "**acoustical vectors**" (here MFCs). This is done in a similar way as what was done for the training data. The result was stored in a file (for acoustical observation) and was known as **test.scp**.

The input observation was then processed by using a Viterbi algorithm and compared with the recognizer's Markov models by using the HTK tool HVite:

HVite -A -D -T 1 -H model/hmm3/hmmdefs.txt recout.mlf -w wdnet diet hmmlist.txt -S test.scp.

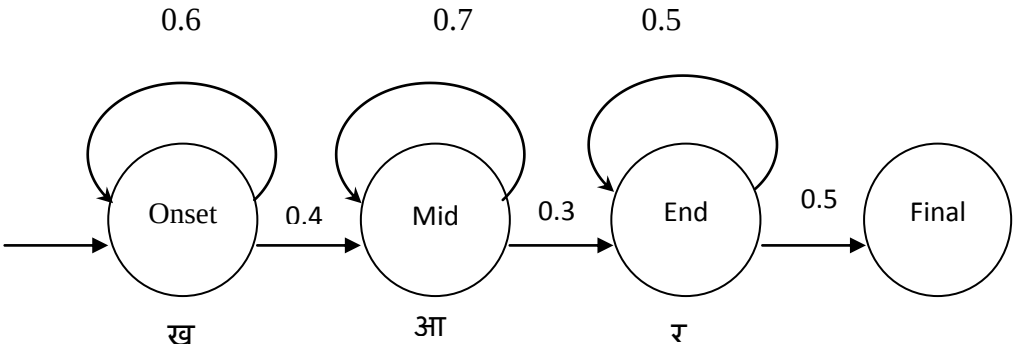
Here hmmdefs.txt is a file containing the definition of the HMMs. The -H option can be repeated to list the different HMM definition files. In this example: **-H model/hmm3/hmm 0.txt -H model/hmm3/hmm 1.txt** etc. However it is simpler to collect all the definitions in a single file called the Master Macro File (particularly when the number of model is more than 3). In the present study, this file was generated by copying each definition one after the other in a particular single file, without the repetition of the header information.

7.7 Transition Diagram and Transition Matrix for some Bodo Words using HMM

Standard Transition Diagram (STD) is a convenient way to describe the behavior of a system that is time dependent. One important rule is that – The behavior of the system in any state must be same irrespective of the path taken to arrive at the state. Moreover, the system behavior can be described by more than one Standard Transition Diagram .A Standard Transition Diagram (STD) has:

(i) an observable mode of the behavior of a system can be shown by a state.
(ii) an Standard Transition Diagram at any particular time can have only one state.

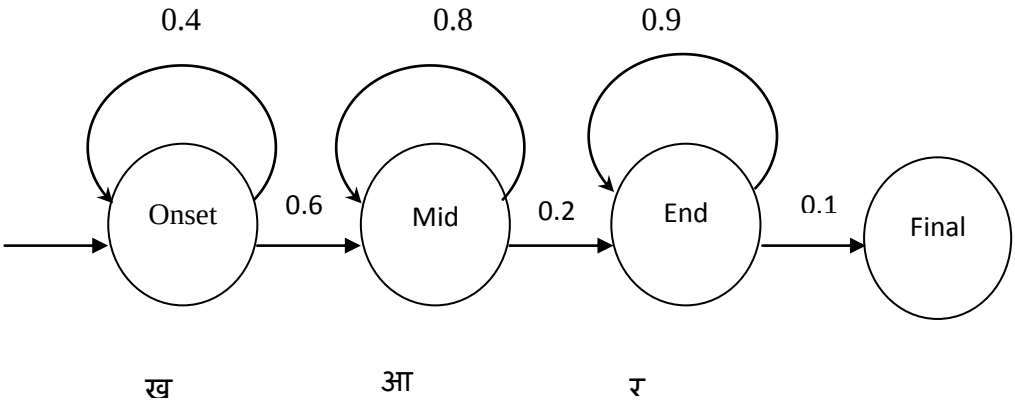
A probability matrix is called a **transition matrix** and is used to show the transition of a Markov chain. The following is the example of a transition diagram along with its state transition matrix for the Bodo word "/खार/(K^har)"(to Run).



Output probabilities for the phone HMM:

Onset:	Mid:	End:
C1:05	C3:0.2	C4:0.3
C2:0.2	C4:0.4	C6:0.5
C3:0.6	C5:0.4	C7:0.1

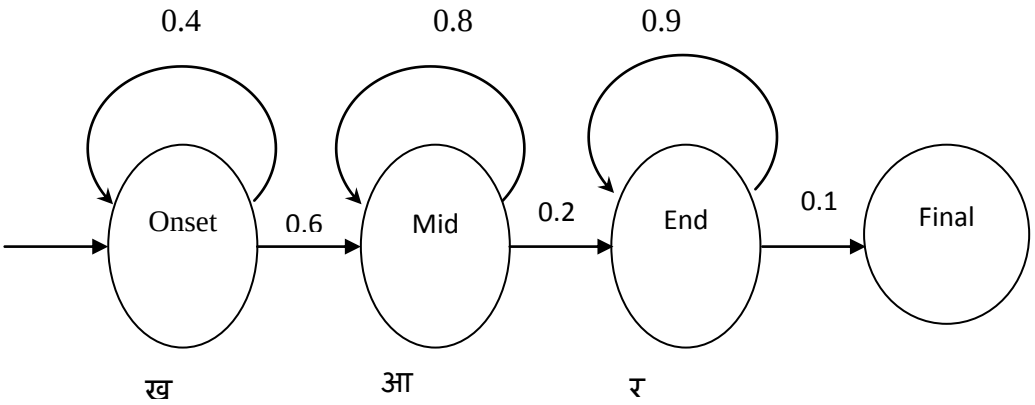
Fig 7.8: An example of Transition diagram and Transition matrix for Bodo word /खार/(K^har) in Normal Mood.



Output probabilities for the phone HMM:

Onset:	Mid:	End:
C1:02	C3:0.2	C4:0.2
C2:0.3	C4:0.6	C6:0.1
C3:0.4	C5:0.2	C7:0.9

Fig 7.9: An example of Transition diagram and Transition matrix for Bodo word /खार/(K^har) in Surprise Mood.



Output probabilities for the phone HMM:

Onset:	Mid:	End:
C1:01	C3:0.6	C4:0.1
C2:0.2	C4:0.2	C6:0.5
C3:0.8	C5:02	C7:0.4

Fig 7.10: An example of Transition diagram and Transition matrix for Bodo word /खार/(K^har) in Angry mode.

7.8 HMM Performance Analysis

To perform the performance analysis, two .mlf files are used as follows:

i) The first file called "**ref1.mlf**' contains the correct transcriptions of the whole test corpus. This is mainly the transcriptions obtained by hand-labelling,

ii)The second file called "**recout1.mlt**" contains the recognised transcriptions of the whole test corpus. It contained the hypothesised transcriptions yielded by the recogniser by using HVite.

HVite -A -D -T 1 -H model/hmm3/hmmdefs.txt recout.mlf -w wdnet diet hmmlist.txt -S test.scp.

The performance measures using HTK performance evaluation tool, HResults, will yield the comparison between the reference transcription (**ref1.mlf**) and the recognition hypothesis (**recout1.mlf**) for each data. It is used as:

HResults -A -D -T 1 -e ??? sil -I ref1.mlf \labellist1.txt recout1.mlf > results1.txt.

Here **results1.txt** stores the output performance statistics and **labellist1.txt** lists all the labels appearing in the transcription files. The option " **-e ??? sil**" indicates that the sil labels will not be considered when the performance statistics is computed.

Here, the recognition rate for sentences is found to be 90%. The word recognition rate stands at 92.12% and the syllable recognition rate stands at =90.18%.

Sentence: Correct=90.00 [H=42, S=4, N=500]

Word: Correct=93.33 Acc=93.33 D=6 S=2, I=0, N=150

Syllable: Correct=90.20 Acc=90.20 D=8 S=10, I=0, N=230

Here, the total number of test data i.e N=500 and the number of test data correctly recognized is 42, i.e H=42 and S=4, i.e the number of substitution errors. From the results it can be concluded that 450 numbers of sentences were recognized of the total 500 sentences, which is equivalent to **90.00%.** The number of sentences which was substituted by other sentences was 4. In the word recognition process, the number of deletion errors was 60, i.e D=60, substitution errors, S=20 and insertion errors, I=0. The total number of words that made up the test data was N=150 of which total of 140 words were correctly recognized. This realized to **93.33%** recognition for words. Here, since the insertion error is zero, it leads to the fact that the accuracy figure, i.e Acc = 93.33%.

The results for syllable are realizable for connected recognition systems. However, there were 10 substitution errors (S), 80 deletion errors (D), and 0 insertion errors (I). N=230 gives the total number of syllables making the test data and of these 216 were correctly recognized leading to a **90.20%** recognition. The accuracy figure of 90.20% is found to be the same as the percentage correct.

These results lead to the fact that the training of the synthesis/recognition system was successful to a great extent. The developed system is also found to be speaker independent.

7.9 Results and Discussion

The evaluation of the Bodo Emotion Recognizer / Synthesizer's performance must be measured on some data corpus that is different from the training corpus. A new separate test corpus with the Bodo language is created. This is done in a similar process as was done previously with the training corpus. The test corpus is made up of 500 recorded sentences and labeled data. These are then later transformed into MFC. To test that the system is speaker independent, some speakers who participated in building the testing corpus was not allowed to participate in building the training corpus.