Disclaimer

- The material provided in this document is not my original work and is a summary of some one else’s work(s).
- A simple Google search of the title of the document will direct you to the original source of the material.
- I do not guarantee the accuracy, completeness, timeliness, validity, non-omission, merchantability or fitness of the contents of this document for any particular purpose.
- Downloaded from najeebkhan.github.io

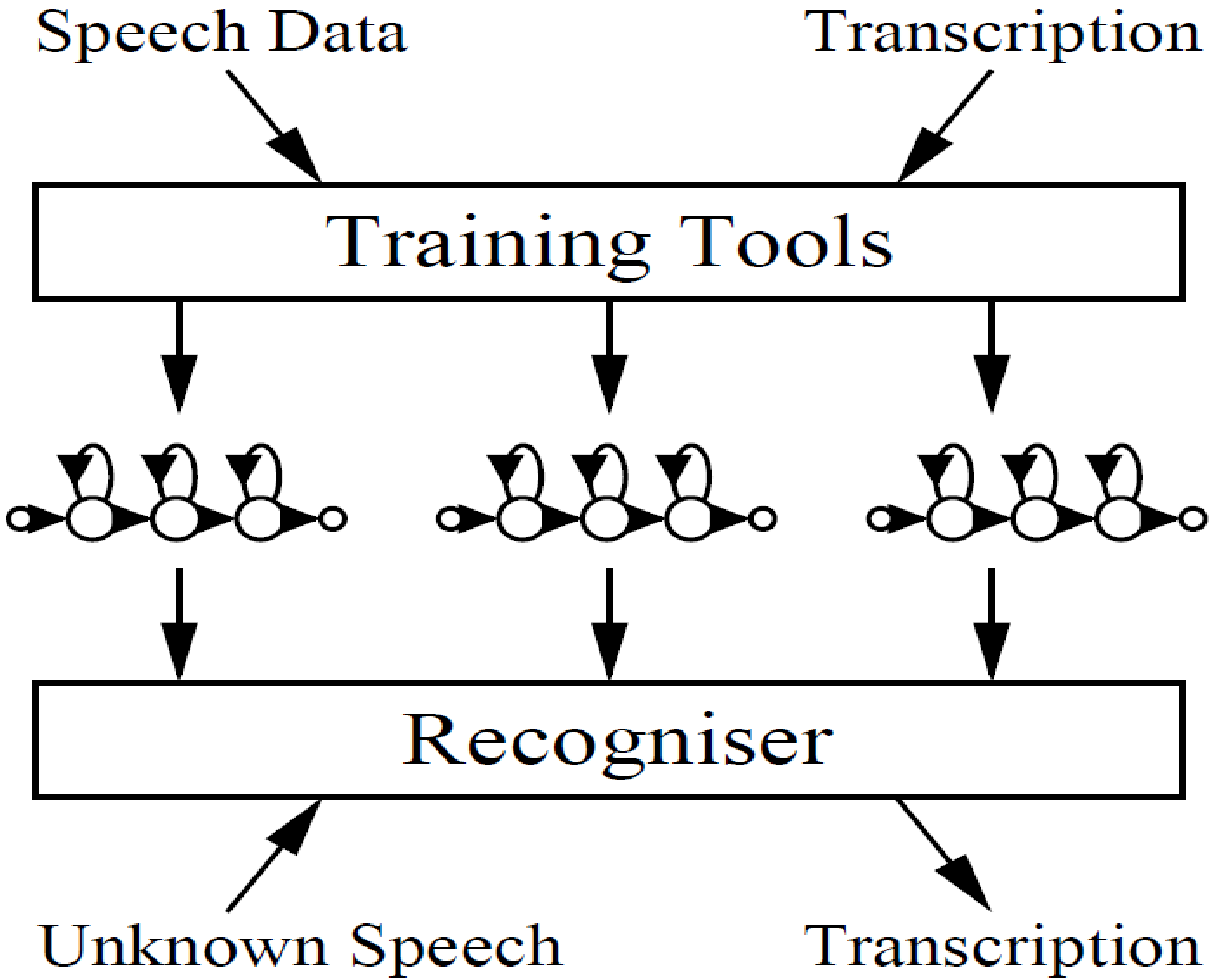
Chapter # 1: The Fundamentals

HMM TOOL KIT HTK

Outline

Introduction

- HTK is a toolkit for building Hidden Markov Models (HMMs)
- HMMs can be used to model any time series and the core of HTK is similarly general-purpose (Designed for speech processing)
- There are two major processing stages involved
  - Estimating the parameters of a set of HMMs using training utterances and their associated transcriptions
  - Unknown utterances are transcribed using the HTK recognition tools

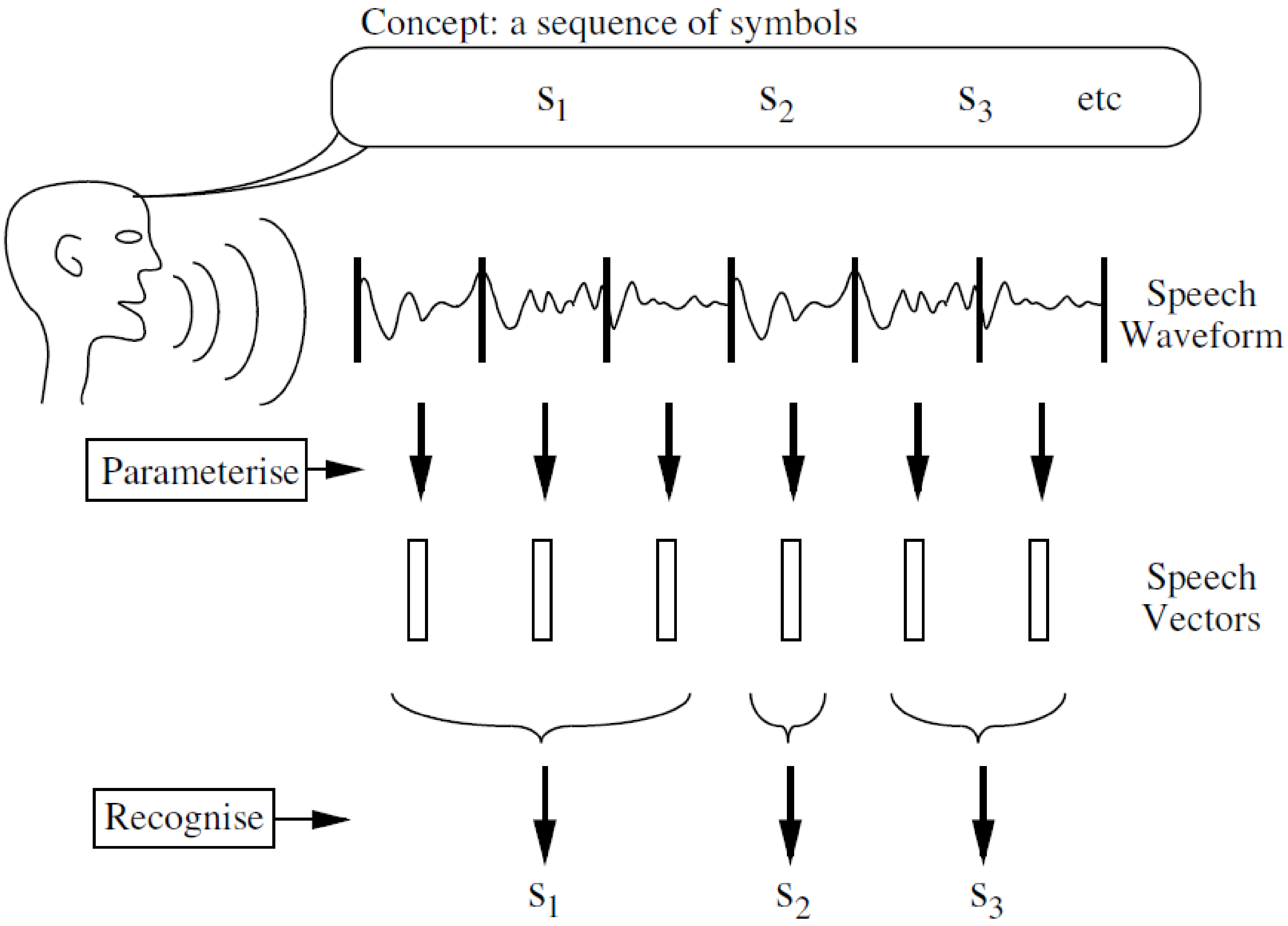

Speech Data
Transcription
Training Tools
Recogniser
Unknown Speech
Transcription

General Principles of HMMs

- Speech recognition systems generally assume that the speech signal is a realization of some message encoded as a sequence of one or more symbols
- To effect the reverse operation of recognizing the underlying symbol sequence given a spoken utterance, the continuous speech waveform is first converted to a sequence of equally spaced discrete parameter vectors

General Principles of HMMs

- Typical parametric representations in common use are smoothed spectra or linear prediction coefficients
- The role of the recognizer is to effect a mapping between sequences of speech vectors and the wanted underlying symbol sequences
- Two problems make this very difficult
  - The mapping from symbols to speech is not one-to-one since different underlying symbols can give rise to similar speech sounds
  - The boundaries between symbols cannot be identified explicitly from the speech waveform


Concept: a sequence of symbols
s_1
s_2
s_3
etc
Speech Waveform
Parameterise
Speech Vectors
Recognise
s_1
s_2
s_3

Isolated Word Recognition

- Let each spoken word be represented by a sequence of speech vectors or observations O

$$O = o_1, o_2, \ldots, o_T$$

- The isolated word recognition problem can then be regarded as that of computing

$$\operatorname{argmax}\left\{P\left(w_i \mid O\right)\right\}$$

$$P\left(w_i \mid O\right) = \frac{P\left(O \mid w_i\right)P\left(w_i\right)}{P\left(O\right)}$$

Isolated Word Recognition

- Given the dimensionality of the observation sequence O, the direct estimation of the joint conditional probability $P(o_1, o_2, \ldots |w_i)$ from examples of spoken words is not practicable
- If a parametric model of word production such as a Markov model is assumed, estimating the class conditional observation densities $P(O|w_i)$ is replaced by the much simpler problem of estimating the Markov model parameters
- In HMM based speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each word is generated by a Markov model

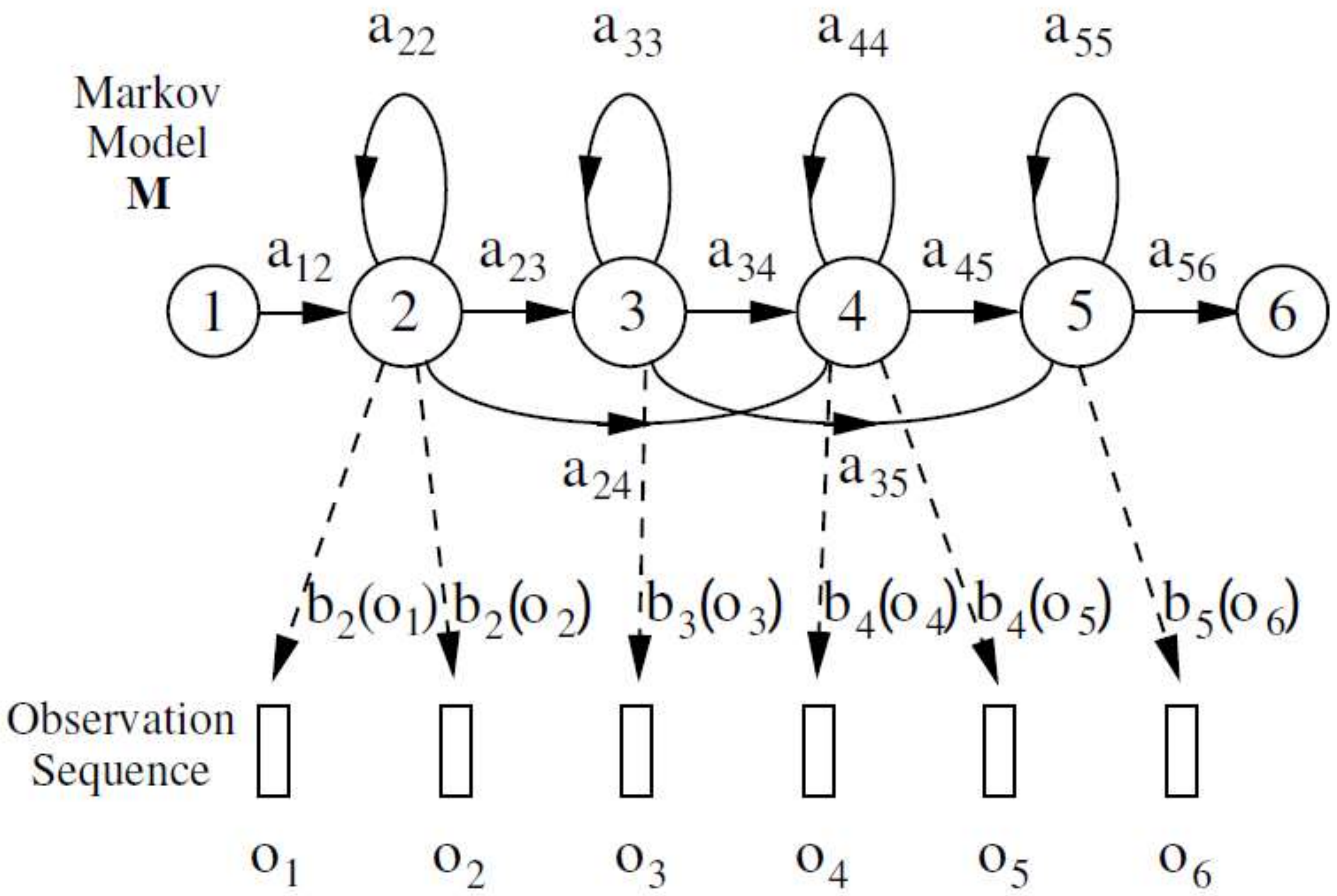


Fig. 1.3 The Markov Generation Model

Isolated Word Recognition

- A Markov model is a finite state machine which changes state once every time unit and each time t that a state j is entered, a speech vector o_t is generated from the probability density $b_j(o_t)$
- The transition from state i to state j is also probabilistic and is governed by the discrete probability a_{ij}

Isolated Word Recognition

- The joint probability that O is generated by the model M moving through the state sequence X is calculated simply as the product of the transition probabilities and the output probabilities

$$P(\boldsymbol{O}, X|M) = a_{12}b_2(\boldsymbol{o}_1)a_{22}b_2(\boldsymbol{o}_2)a_{23}b_3(\boldsymbol{o}_3)\ldots$$

- In practice, only the observation sequence O is known and the underlying state sequence X is hidden

Isolated Word Recognition

- Given that X is unknown, the required likelihood is computed by summing over all possible state sequences X = x(1), x(2), x(3), …, x(T)

$$P(\boldsymbol{O}|M) = \sum_X a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(\boldsymbol{o}_t) a_{x(t)x(t+1)}$$

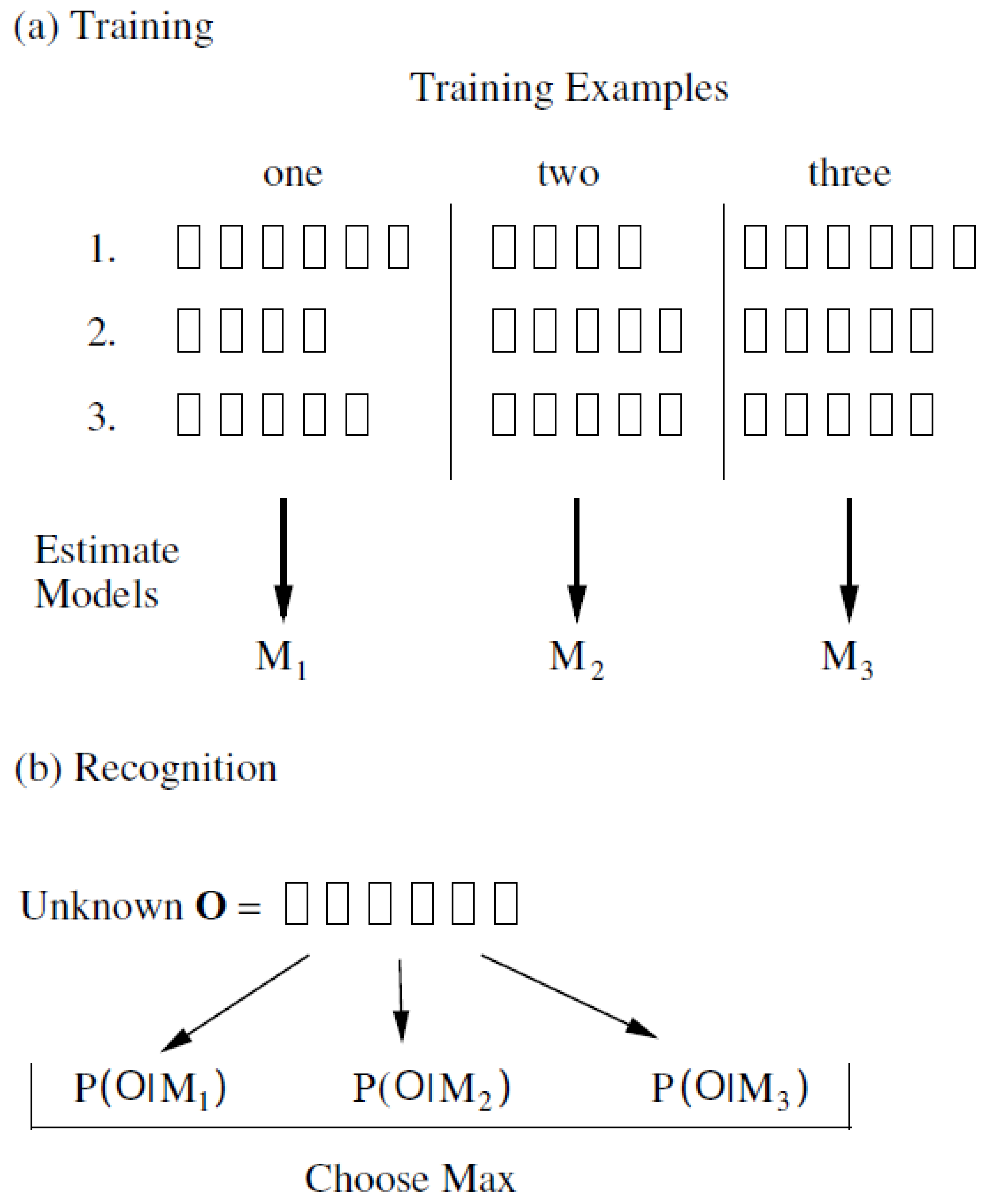
- Alternatively, the likelihood can be approximated by only considering the most likely state sequence

$$\hat{P}(\boldsymbol{O}|M) = \max_X \left\{ a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(\boldsymbol{o}_t) a_{x(t)x(t+1)} \right\}$$

- The direct computation of these equations is not tractable, however simple recursive procedures exist which allow both quantities to be calculated very efficiently

Isolated Word Recognition

- Given a set of models M_i corresponding to words w_i $P(\boldsymbol{O}|w_i) = P(\boldsymbol{O}|M_i)$
- Given a set of training examples corresponding to a particular model, the parameters of that model can be determined automatically by a robust and efficient re-estimation procedure
- Given a sufficient number of representative examples of each word can be collected then a HMM can be constructed which implicitly models all of the many sources of variability inherent in real speech


(a) Training
Training Examples
one
two
three
1.
2.
3.
Estimate Models
M_1
M_2
M_3
(b) Recognition
Unknown **O** =
$P(O|M_1)$
$P(O|M_2)$
$P(O|M_3)$
Choose Max

Output Probability Specification

- HTK is designed primarily for modelling continuous parameters using continuous density multivariate output distributions
- HTK represents output distributions by Gaussian Mixture Densities
- HTK allows each observation vector at time t to be split into a number of S independent data streams o_{st}

$$b_j(\boldsymbol{o}_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_s} c_{jsm} \mathcal{N}(\boldsymbol{o}_{st}; \boldsymbol{\mu}_{jsm}, \boldsymbol{\Sigma}_{jsm}) \right]^{\gamma_s}$$

$$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}-\boldsymbol{\mu})}$$

Baum-Welch Re-Estimation

- To determine the parameters of a HMM it is first necessary to make a rough guess at what they might be
- Once this is done, more accurate parameters can be found by applying the so-called Baum-Welch re-estimation formulae
- Mixture components can be considered to be a special form of sub-state in which the transition probabilities are the mixture weights

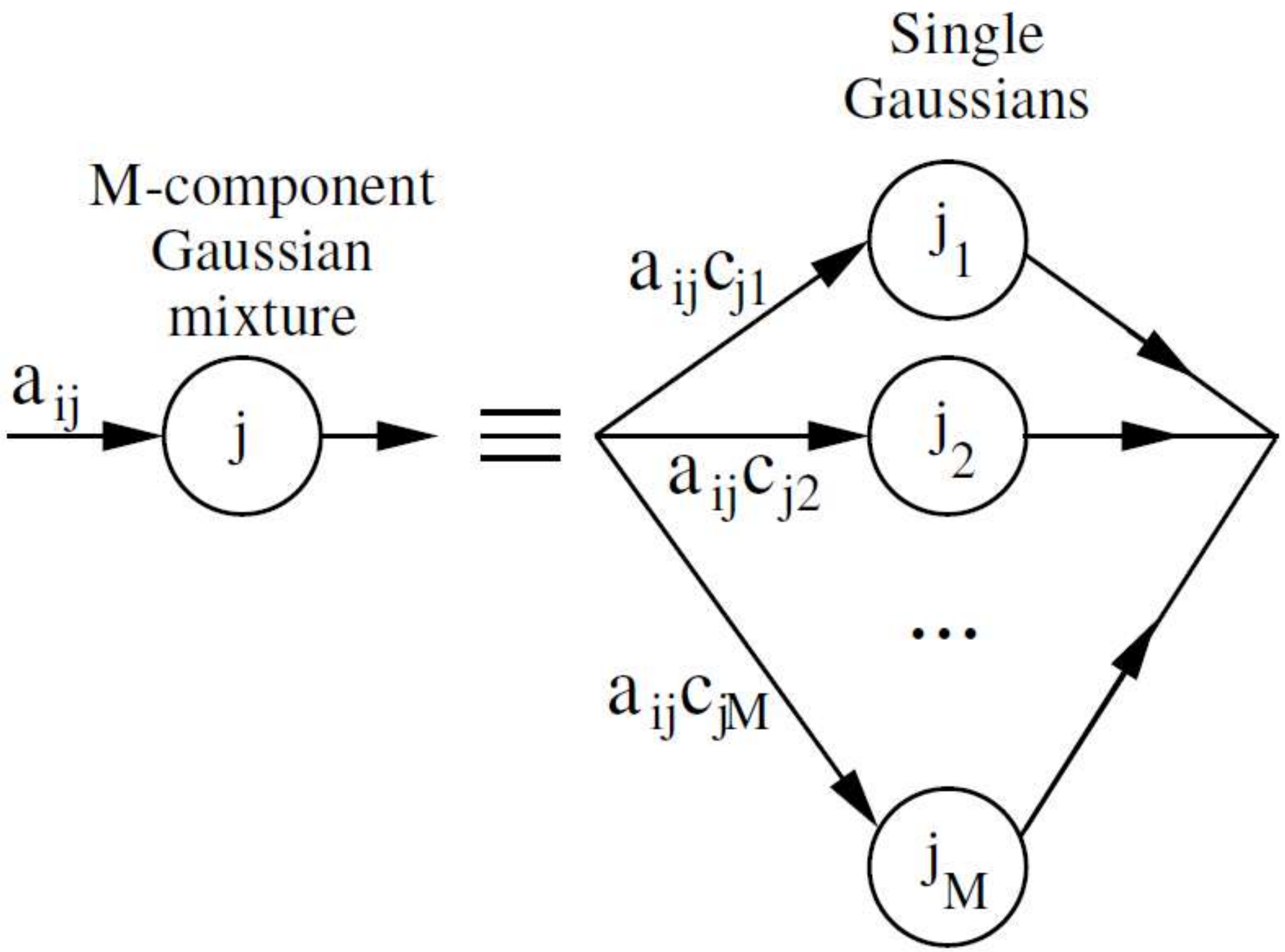


Fig. 1.5 Representing a Mixture

Baum-Welch Re-Estimation

- The essential problem is to estimate the means and variances of a HMM in which each state output distribution is a single component Gaussian
- If there was just one state j in the HMM, this parameter estimation would be easy

$$\hat{\boldsymbol{\mu}}_j = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{o}_t$$

$$\hat{\boldsymbol{\Sigma}}_j = \frac{1}{T}\sum_{t=1}^{T} (\boldsymbol{o}_t - \boldsymbol{\mu}_j)(\boldsymbol{o}_t - \boldsymbol{\mu}_j)'$$

Baum-Welch Re-Estimation

- In practice, of course, there are multiple states and there is no direct assignment of observation vectors to individual states because the underlying state sequence is unknown
- HInit first divides the training observation vectors equally amongst the model states and then calculates initial values for the mean and variance of each state

Baum-Welch Re-Estimation

- It then finds the maximum likelihood state sequence using the Viterbi algorithm, reassigns the observation vectors to states and then recalculates mean and variance to get better initial values
- This process is repeated until the estimates do not change

Baum-Welch Re-Estimation

- Instead of assigning each observation vector to a specific state as in the above approximation, each observation is assigned to every state in proportion to the probability of the model being in that state when the vector was observed
- if $L_j(t)$ denotes the probability of being in state j at time t then

$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_{t=1}^{T} L_j(t)\boldsymbol{o}_t}{\sum_{t=1}^{T} L_j(t)}$$

$$\hat{\boldsymbol{\Sigma}}_j = \frac{\sum_{t=1}^{T} L_j(t)(\boldsymbol{o}_t - \boldsymbol{\mu}_j)(\boldsymbol{o}_t - \boldsymbol{\mu}_j)'}{\sum_{t=1}^{T} L_j(t)}$$

Baum-Welch Re-Estimation

- The probability of state occupation $L_j(t)$ is calculated efficiently by the forward-backward algorithm

Forward Algorithm

- $\alpha_j(t)$ is the joint probability of observing the first t speech vectors and being in state j at time t

$$\alpha_j(t) = P(\boldsymbol{o}_1, \ldots, \boldsymbol{o}_t, x(t) = j | M)$$

- This forward probability can be efficiently calculated by the following recursion

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij} \right] b_j(\boldsymbol{o}_t)$$

- With the initial and final conditions

$$\alpha_1(1) = 1 \qquad \alpha_j(1) = a_{1j} b_j(\boldsymbol{o}_1)$$

$$\alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T) a_{iN}$$

- The calculation of the forward probability also yields the total likelihood $P(O_j|M)$.

$$P(\boldsymbol{O}|M) = \alpha_N(T)$$

Backward Algorithm

- $\beta_j(t)$ is given by

$$\beta_j(t) = P(\boldsymbol{o}_{t+1}, \ldots, \boldsymbol{o}_T | x(t) = j, M)$$

- This Backward probability can be efficiently calculated by the following recursion

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_j(t+1)$$

- With the initial and final conditions

$$\beta_i(T) = a_{iN} \qquad \beta_1(1) = \sum_{j=2}^{N-1} a_{1j} b_j(\boldsymbol{o}_1) \beta_j(1)$$

Baum-Welch Re-Estimation

- Probability of state occupation can be determined by taking the product of the two probabilities

$$\alpha_j(t)\beta_j(t) = P(\mathbf{O}, x(t) = j|M)$$

$$\begin{aligned} L_j(t) &= P(x(t) = j|\mathbf{O}, M) \\ &= \frac{P(\mathbf{O}, x(t) = j|M)}{P(\mathbf{O}|M)} \\ &= \frac{1}{P}\alpha_j(t)\beta_j(t) \end{aligned}$$

Baum-Welch Re-Estimation

- Baum-Welch Re-Estimation can be summarized as follows
  - For every parameter vector/matrix requiring re-estimation, allocate storage for the numerator and denominator summations of the Baum-Welch re-estimation formulae. These storage locations are referred to as accumulators
  - Calculate the forward and backward probabilities for all states j and times t
  - For each state j and time t, use the probability $L_j(t)$ and the current observation vector o_t to update the accumulators for that state
  - Use the final accumulator values to calculate new parameter values
  - If the value of $P = P(O_j|M)$ for this iteration is not higher than the value at the previous iteration then stop, otherwise repeat

Recognition and Viterbi Decoding

- The efficient recursive algorithm for computing the forward probability also yielded as a by-product the total likelihood P(O|M)
- This algorithm could also be used to find the model which yields the maximum value of P(O|M_i), and hence, it could be used for recognition
- In practice, however, it is preferable to base recognition on the maximum likelihood state sequence since this generalizes easily to the continuous speech
- This likelihood is computed using essentially the same algorithm as the forward probability calculation except that the summation is replaced by a maximum operation

Viterbi Algorithm

- Let $\Phi_j(t)$ represent the maximum likelihood of observing speech vectors o_1 to o_t and being in state j at time t
- This partial likelihood can be computed efficiently using the following recursion

$$\phi_j(t) = \max_i \{\phi_i(t-1)a_{ij}\} b_j(\boldsymbol{o}_t)$$

- With the initial and final conditions

$$\phi_1(1) = 1 \qquad \phi_j(1) = a_{1j} b_j(\boldsymbol{o}_1)$$

$$\phi_N(T) = \max_i \{\phi_i(T) a_{iN}\}$$

- In log domain

$$\psi_j(t) = \max_i \{\psi_i(t-1) + log(a_{ij})\} + log(b_j(\boldsymbol{o}_t))$$

Continuous Speech Recognition

- The extension to continuous speech simply involves connecting HMMs together in sequence
- Each model in the sequence corresponds directly to the assumed underlying symbol
- In HTK the use of HInit and HRest for initializing sub-word models is regarded as a bootstrap operation
- The main training phase involves the use of a tool called HERest which does embedded training

Embedded Training

- Embedded training uses the same Baum-Welch procedure as for the isolated case but rather than training each model individually all models are trained in parallel. ET involves the following steps
  - Get the next training utterance
  - Construct a composite HMM by joining in sequence the HMMs corresponding to the symbol transcription of the training utterance
  - Calculate the forward and backward probabilities for the composite HMM (considering non-emitting states)
  - Compute the probabilities of state occupation at each time frame
  - Repeat until all training utterances have been processed
  - Use the accumulators to calculate new parameter estimates for all of the HMMs

Continuous Speech Recognition

- Whereas the extensions needed to the Baum-Welch procedure for training sub-word models are relatively minor, the corresponding extensions to the Viterbi algorithm are more substantial
- Token Passing Model

Token Passing Model

- In HTK, an alternative formulation of the Viterbi algorithm is used called the Token Passing Model
- Imagine each state j of a HMM at time t holds a single moveable token which contains, amongst other information, the partial log probability $\Phi_j(t)$
- This token then represents a partial match between the observation sequence o_1 to o_t and the model subject to the constraint that the model is in state j at time t

Token Passing Model

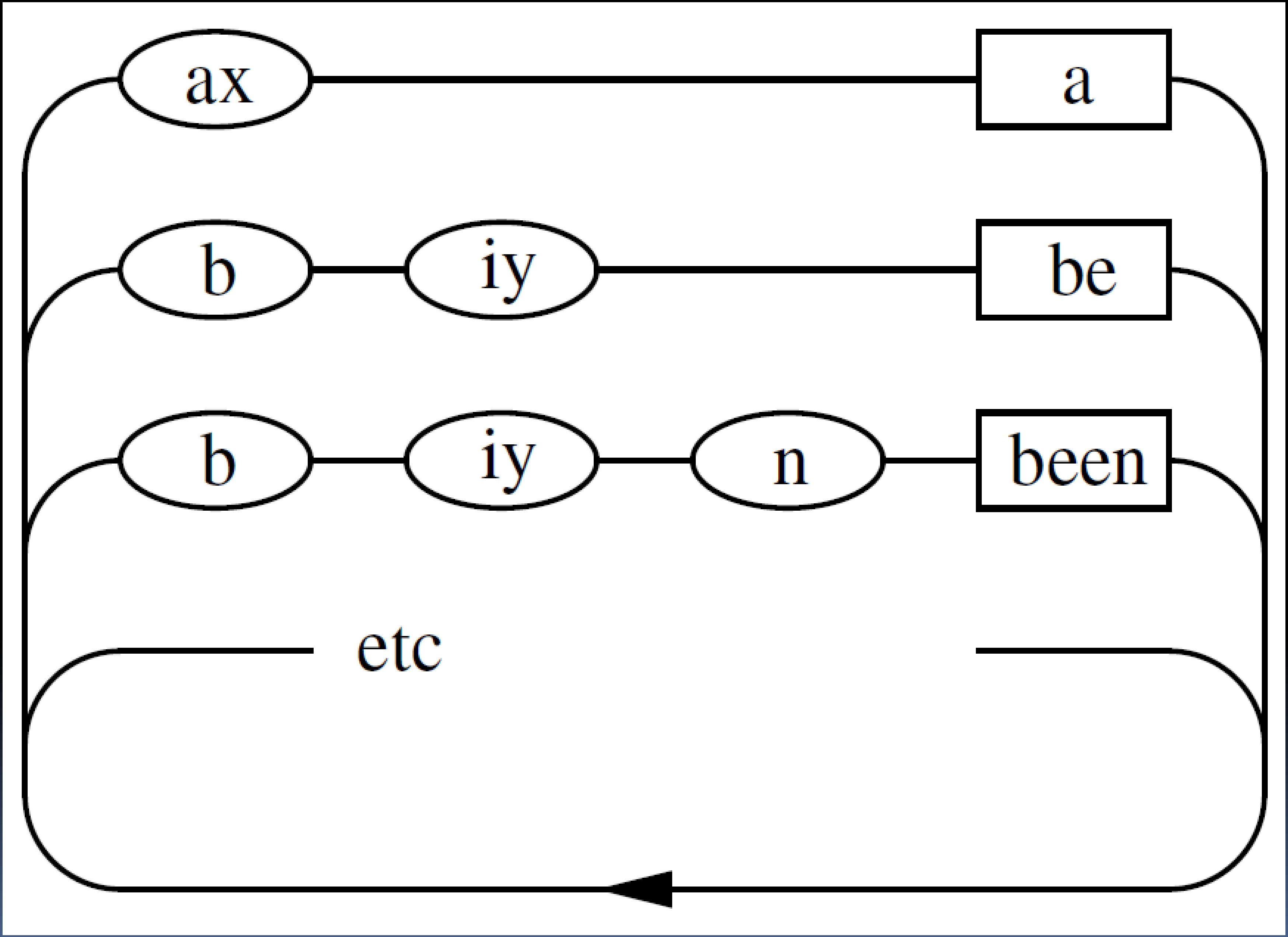
- The path extension algorithm is then replaced by the equivalent token passing algorithm which is executed at each time frame t
- The key steps in this algorithm are as follows
  - Pass a copy of every token in state i to all connecting states j, incrementing the log probability of the copy by $\log[a_{ij}] + \log[b_j(o(t))]$
  - Examine the tokens in every state and discard all but the token with the highest probability

Token Passing Model

- To deal with the non-emitting states
  - Tokens in entry states are assumed to represent paths extended to time t - δt
  - Tokens in exit states are assumed to represent paths extended to time t + δt

Token Passing Model

- Token Passing Model extends very simply to the continuous speech case
- Suppose that the allowed sequence of HMMs is defined by a finite state network


ax
a
b
iy
be
b
iy
n
been
etc

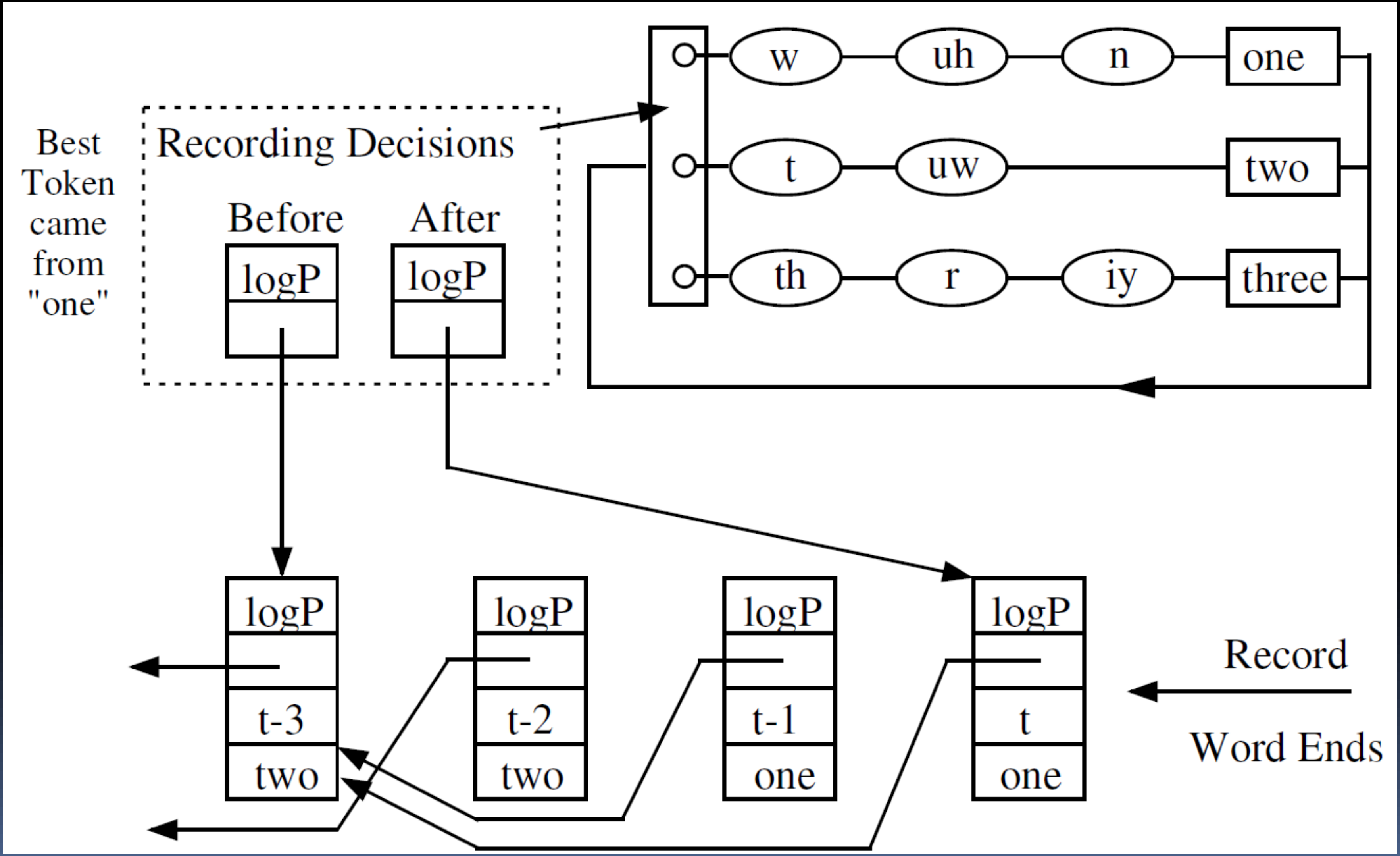
Token Passing Model

- Each word is defined as a sequence of phoneme based HMMs and all of the words are placed in a loop
- The oval boxes denote HMM instances and the square boxes denote word-end nodes
- This composite network is essentially just a single large HMM
- When the best token reaches the end of the speech, the route it took through the network must be known in order to recover the recognized sequence of models

Token Passing Model

- Every token carries a pointer called a *word end link*
- A record called a Word Link Record is generated in which is stored the identity of the word from which the token has just emerged and the current value of the token's link
- The token's actual link is then replaced by a pointer to the newly created WLR
- Once all of the unknown speech has been processed, the WLRs attached to the link of the best matching token can be traced back to give the best matching sequence of words

Token Passing Model



Token Passing Model

- If required, the same principle can be used to record decisions at the model and state level
- More than just the best token at each word boundary can be saved
- The algorithms are embedded in the library modules HNet and HRec and they may be invoked using the recognizer tool called HVite
- They provide single and multiple-token passing recognition, single-best output, lattice output, N-best lists, support for cross-word context-dependency, lattice rescoring and forced alignment

Speaker Adaptation

- The trained systems can be improved upon by customizing the HMMs to the characteristics of a particular speaker
- HTK provides the tools HERest and HVite to perform adaptation using a small amount of enrollment or adaptation data
- HERest performs offline supervised adaptation
- HVite recognizes the adaptation data and uses the generated transcriptions to perform the adaptation

Thank You