T-61.184
Automatic Speech Recognition: From Theory to Practice

http://www.cis.hut.fi/Opinnot/T-61.184/
November 22, 2004

Prof. Bryan Pellom
Department of Computer Science
Center for Spoken Language Research
University of Colorado

pellom@cslr.colorado.edu

Today

- **Course Review (45 Minutes)**
- **Project Talks (3:15-4:00pm)**
  - Matti Aksela
    "Classifier Combination for Speech Recognition"
  - Teemu Hirsimäki
    "Weighted Finite-State Transducers (WFSTs)"
  - Janne Plykkönen
    "An Implementation of a Token Pass Decoder"
  - Bernhard Leiner
    "Noise Robust Speech Recognition"

Course Feedback

http://www.cs.hut.fi/u/vjs/php/palaute.php

- **It has been my pleasure teaching this course and being here at HUT since August 1st.**
- **Teaching styles between the U.S. and Finland are quite different, I hope this has not been any significant problem and I hope most of all that you have learned something from the course.**
- **Please be sure to fill out the course feedback so that I can improve the course for next time!**

ASR Course Review
"The Highlights"

- Problem Formulation
- Feature Extraction
- Hidden Markov Models
- Acoustic Modeling
- Language Modeling
- Search
- Adaptation

Problem Formulation

Problem Description

- **Given a sequence of observations (evidence) from an audio signal,**

$$\mathrm{O} = o_1 o_2 \cdots o_T$$

- **Determine the underlying word sequence,**

$$\mathrm{W} = w_1 w_2 \cdots w_m$$

- **Number of words (m) unknown, observation sequence is variable length (T)**

Problem Formulation

- **Goal: Minimize the classification error rate**

$$\Downarrow$$

- **Solution: Maximize the Posterior Probability**

$$\hat{W} = \arg\max_{W} P(W \mid O)$$

- **Solution requires optimization over all possible word strings!**

Problem Formulation

- **Using Bayes Rule,**

$$P(\mathrm{W} \mid \mathrm{O}) = \frac{P(\mathrm{O} \mid \mathrm{W}) P(\mathrm{W})}{P(\mathrm{O})}$$

- **Since P(O) does not impact optimization,**

$$\hat{\mathrm{W}} = \arg\max_{\mathrm{W}} P(\mathrm{W} \mid \mathrm{O})$$

$$= \arg\max_{\mathrm{W}} P(\mathrm{O} \mid \mathrm{W}) P(\mathrm{W})$$

Problem Formulation

- **Let's assume words can be represented by a sequence of states, S,**

$$\begin{aligned}\hat{\mathrm{W}} &= \arg\max_{\mathrm{W}} P(\mathrm{O} \mid \mathrm{W})P(\mathrm{W}) \\ &= \arg\max_{\mathrm{W}} \sum_{S} P(\mathrm{O} \mid \mathrm{S})P(S \mid W)P(\mathrm{W})\end{aligned}$$

- **Words → Phonemes → States**
- **States represent smaller pieces of phonemes**

Problem Formulation

- **Optimize:** $\hat{\mathrm{W}} = \arg\max_{\mathrm{W}} \sum_{S} P(\mathrm{O} \mid S) P(S \mid \mathrm{W}) P(\mathrm{W})$

- **Practical Realization,**

O	Observation (feature) sequence
$\mathrm{P(O \mid S)}$	Acoustic Model
$\mathrm{P(S \mid W)}$	Lexicon / Pronunciation Model
$\mathrm{P(W)}$	Language Model

T-61.184

Practical Speech Recognition

- **In practice, we work with log-probabilities,**

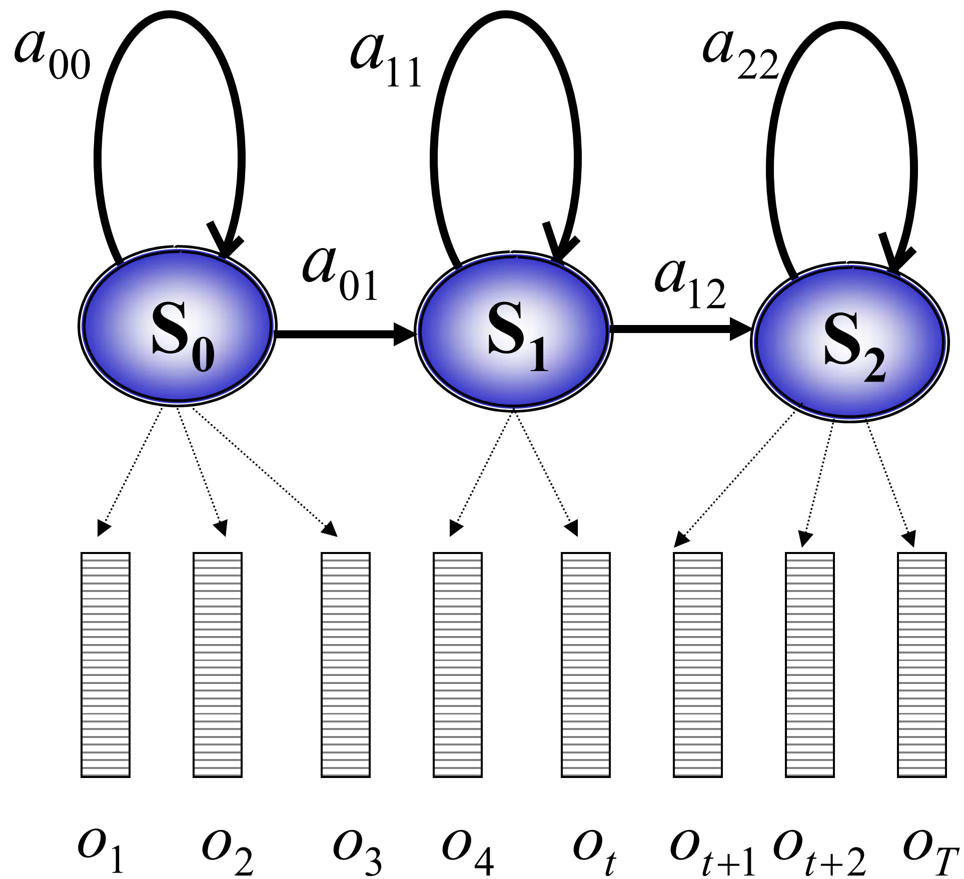
$$\hat{\mathrm{W}} = \arg\max_{\mathrm{W}} \left\{ \log\left(P(\mathrm{O} \mid \mathrm{W}) P(W) \right) \right\}$$

- **Common to scale LM probabilities by a <u>grammar scale factor</u> ("s") and also include a <u>word-transition penalty</u> ("p"):**

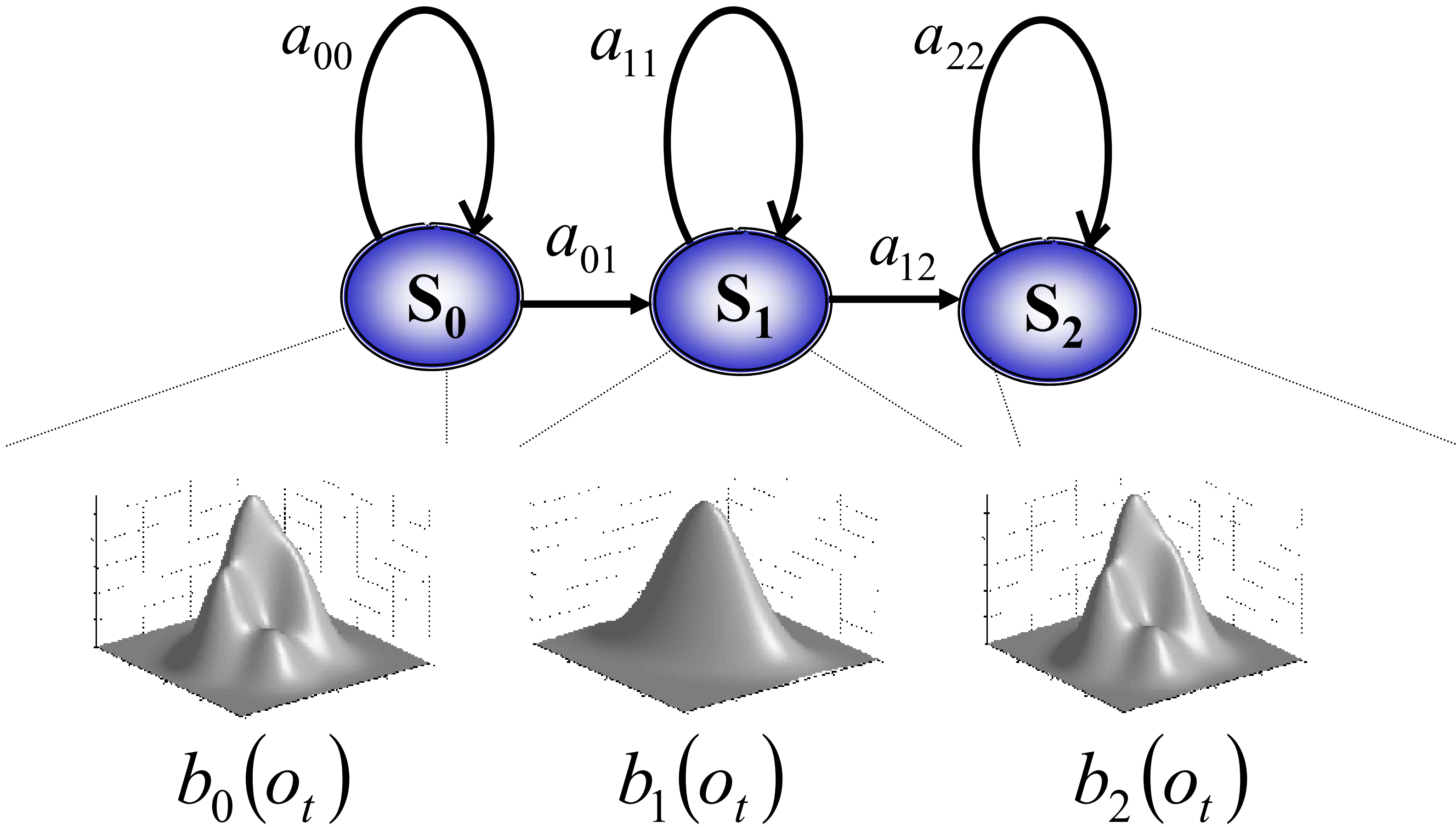
$$\hat{\mathrm{W}} = \arg\max_{W} \left\{ \underbrace{\log\left(P(\mathrm{O} \mid \mathrm{W}) \right)}_{\text{acoustic model}} + \underbrace{\mathrm{s} \cdot \log\left(\mathrm{P}(\mathrm{W}) \right) + \mathrm{p}}_{\text{language model}} \right\}$$

Hidden Markov Models (HMMs)

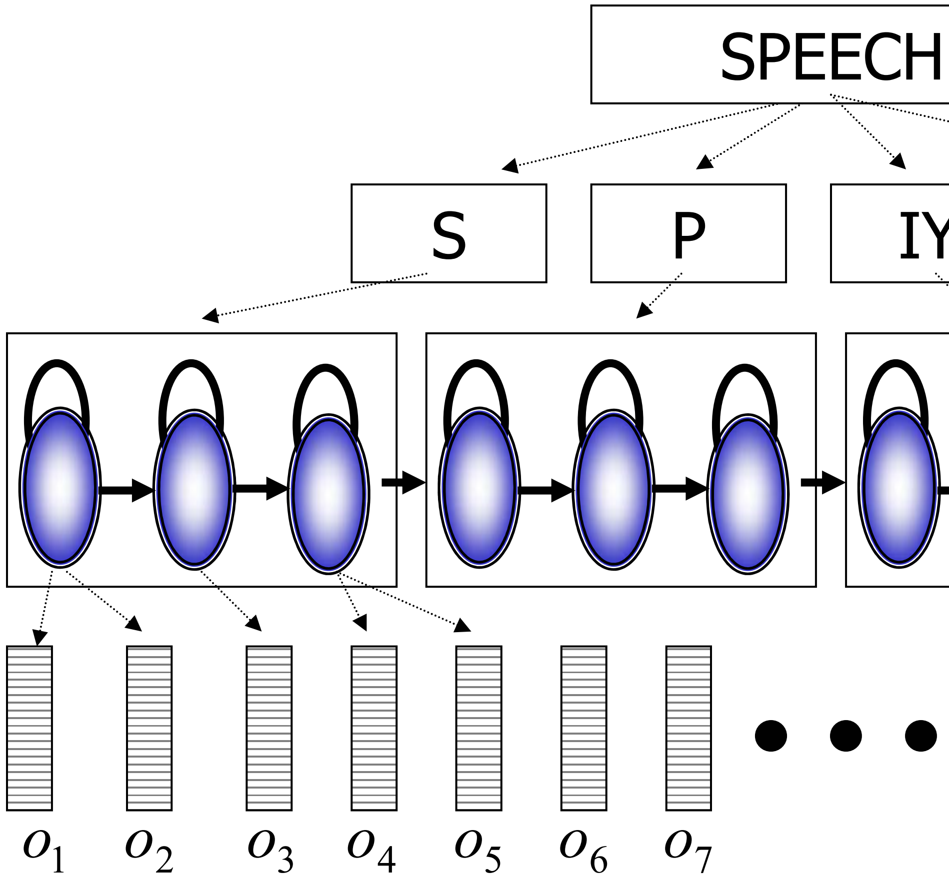
- Observation vectors are assumed to be "generated" by a Markov Model
- HMM: A finite-state machine that at each time t that a state j is entered, an observation is emitted with probability density $b_j(o_t)$
- Transition from state i to state j modeled with probability a_{ij}



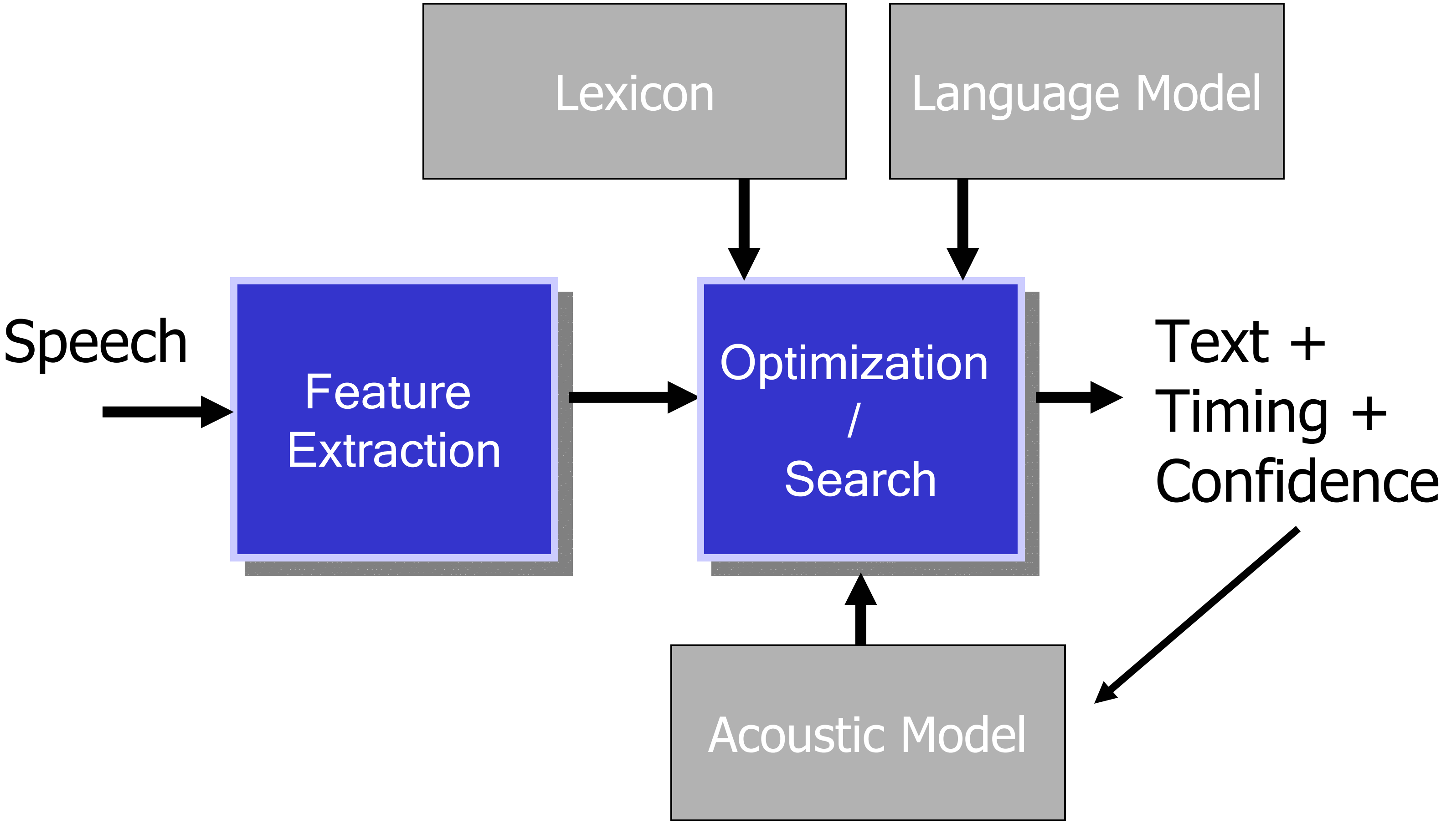
Modeled Probability Distribution



"Beads-on-a-String" HMM Representation



Components of a Speech Recognizer

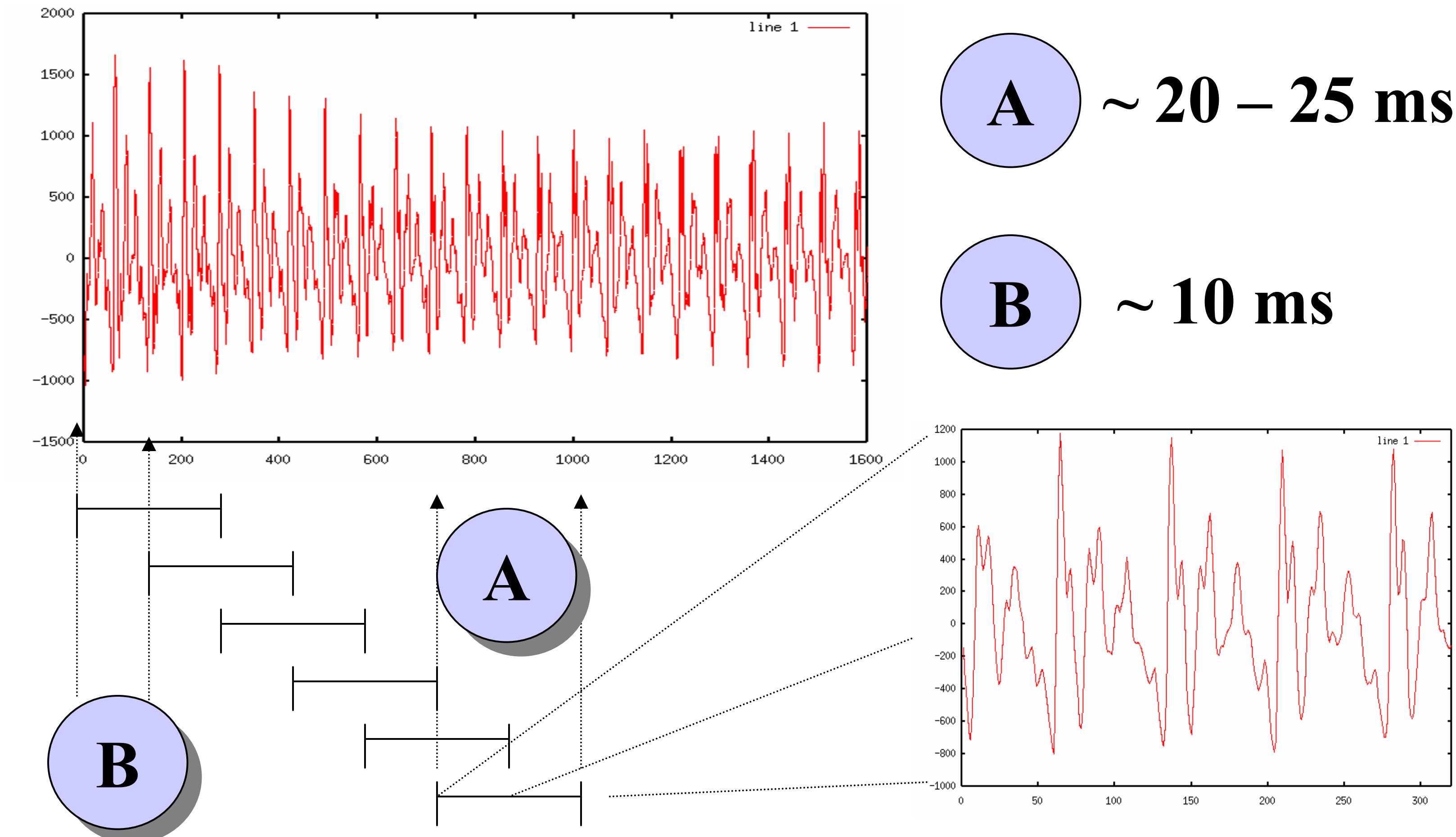


Feature Extraction

Goals of Feature Extraction

- Compactness
- Discrimination Power
- Low Computation Complexity
- Reliable
- Robust

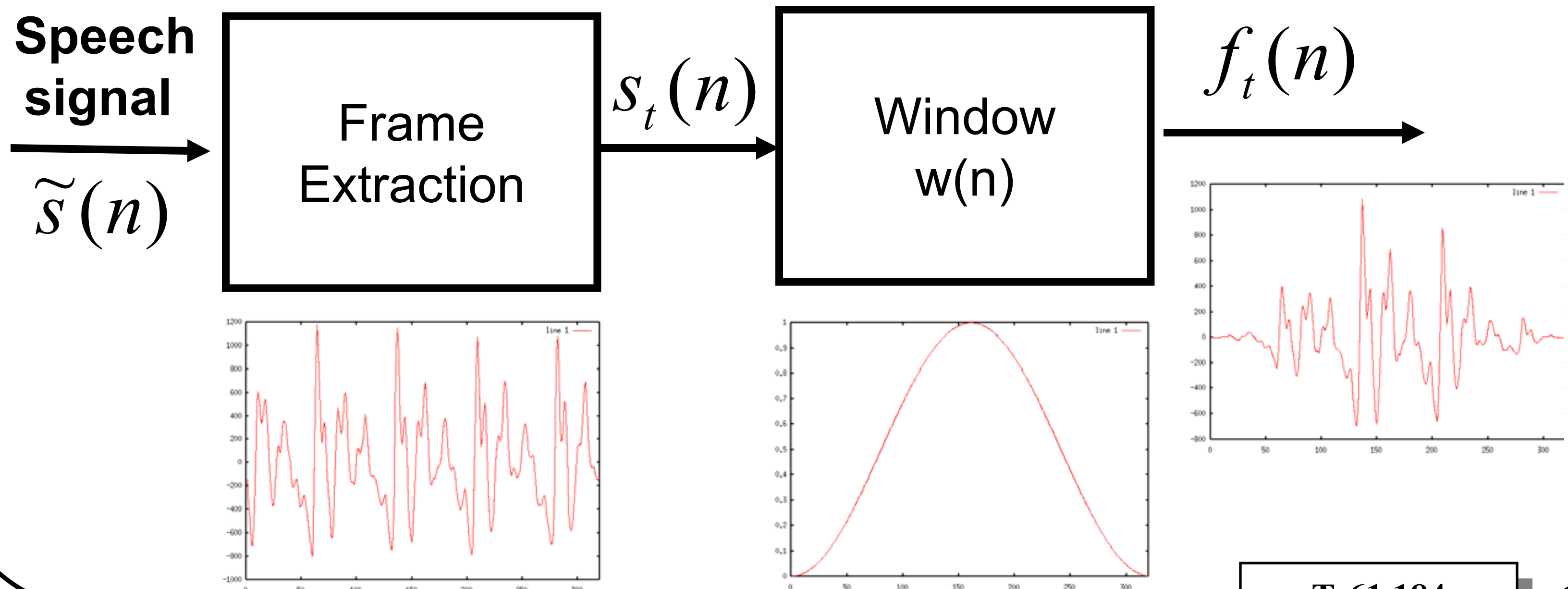
Frame Blocking



Frame Windowing

- **Each frame is multiplied by a smooth window function to minimize spectral discontinuities are the begin/end of each frame,**

$$f_t(n) = s_t(n)w(n)$$

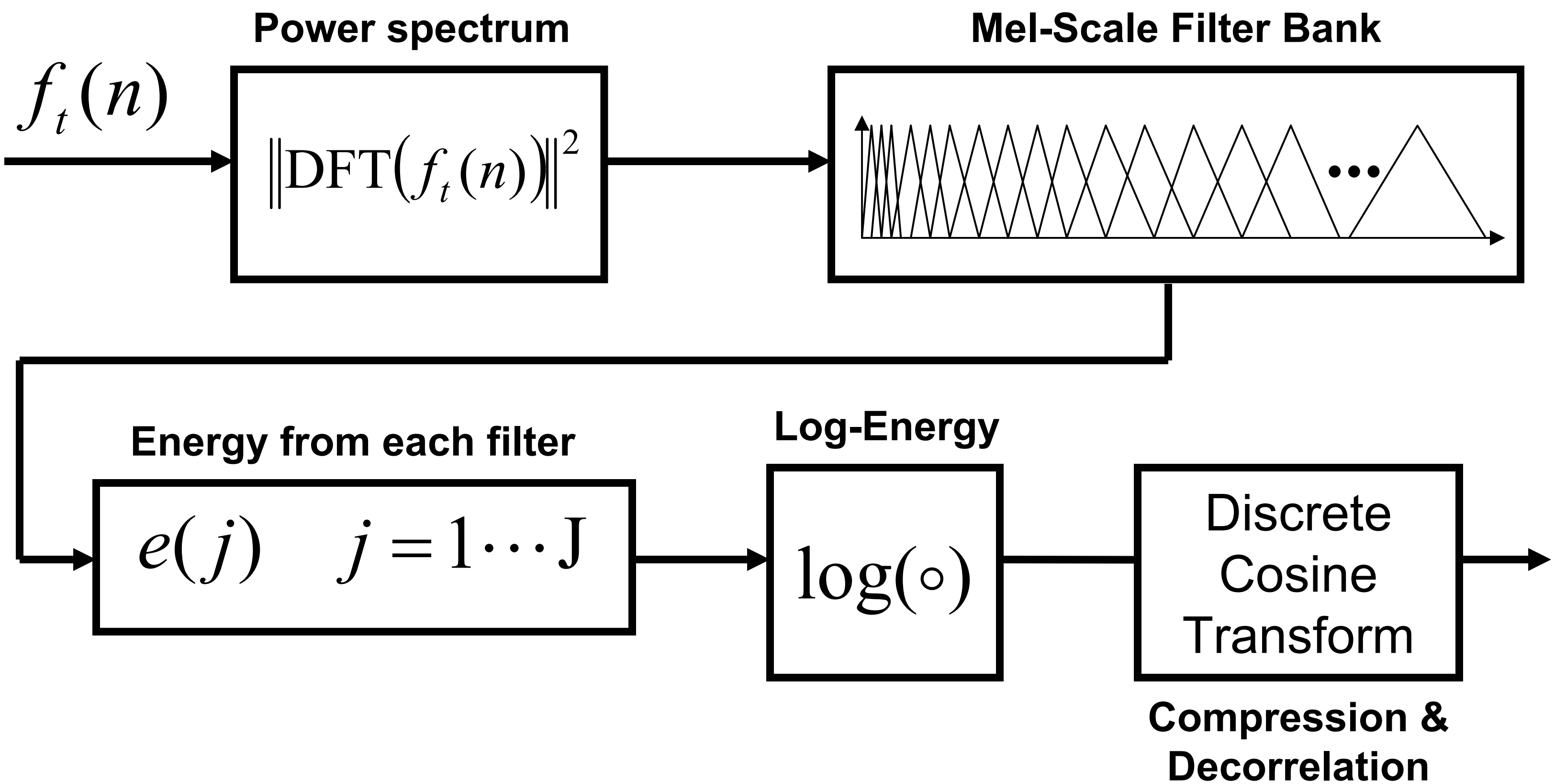


Mel-Frequency Cepstral Coefficients (MFCC)

- **Davis & Mermelstein (1980)**
- **Computes signal energy from a bank of filters that are linearly spaced at frequencies below 1kHz and logarithmically spaced above 1kHz.**
- **Same and equal spacing of filters along Mel-Scale,**

$$Mel(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

MFCC Block Diagram



Dynamic Cepstral Coefficients

- **Cepstral coefficients do not capture temporal information**
- **Common to compute velocity and acceleration of cepstral coefficients. For example, for delta (velocity) features,**

$$\Delta cep[i][t] = \frac{\sum_{\tau=1}^{D} \tau\left(cep[i][t+\tau] - cep[i][t-\tau]\right)}{2\sum_{\tau=1}^{D} \tau^2}$$

- **D typically 2**

T-61.184

Frame Energy

- **Frame energy is a typical feature used in speech recognition. Frame energy is computed from the windowed frame,**

$$e[t] = \sum_{m} s^2(n)$$

- **Typically a normalized log energy is used. E.g.,**

$$e_{\max} = \arg\max_{t} \{0.1 \cdot \log(e[t])\}$$

$$E[t] = \arg\max\{-5.0\,,\,0.1 \cdot \log(e[t]) - e_{\max} + 1.0\}$$

Final Feature Vector for ASR

- **A single feature vector,**
  - 12 cepstral coefficients (PLP, MFCC, …) + 1 norm energy
  - + 13 delta features
  - + 13 delta-delta
- **100 feature vectors per second**
- **Each vector is 39-dimensional**
- **Characterizes the spectral shape of the signal for each time slice**

Hidden Markov Models

Discrete Symbol Observation HMM

- **A set of N states**

$$S = \{S_0, S_1, \cdots S_N\}$$

- **Transition Probabilities**

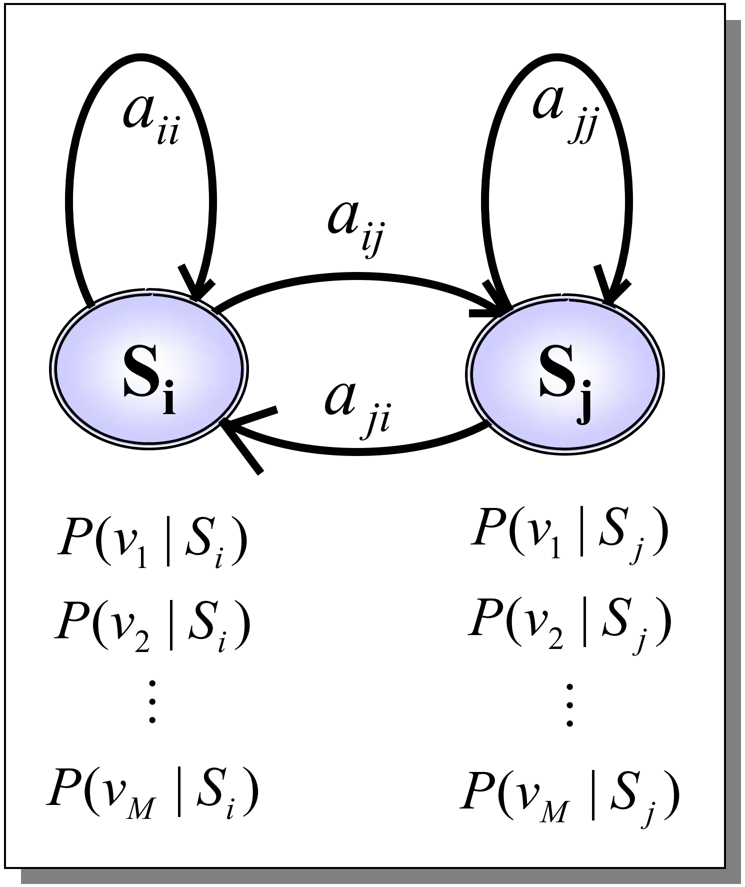
$$a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i)$$

- **A set of M observation symbols**

$$V = \{v_1, v_2, \cdots v_M\}$$

- **Probability Distribution (for state j, symbol k)**

$$b_j(k) = P(o_t = v_k \mid q_t = j)$$



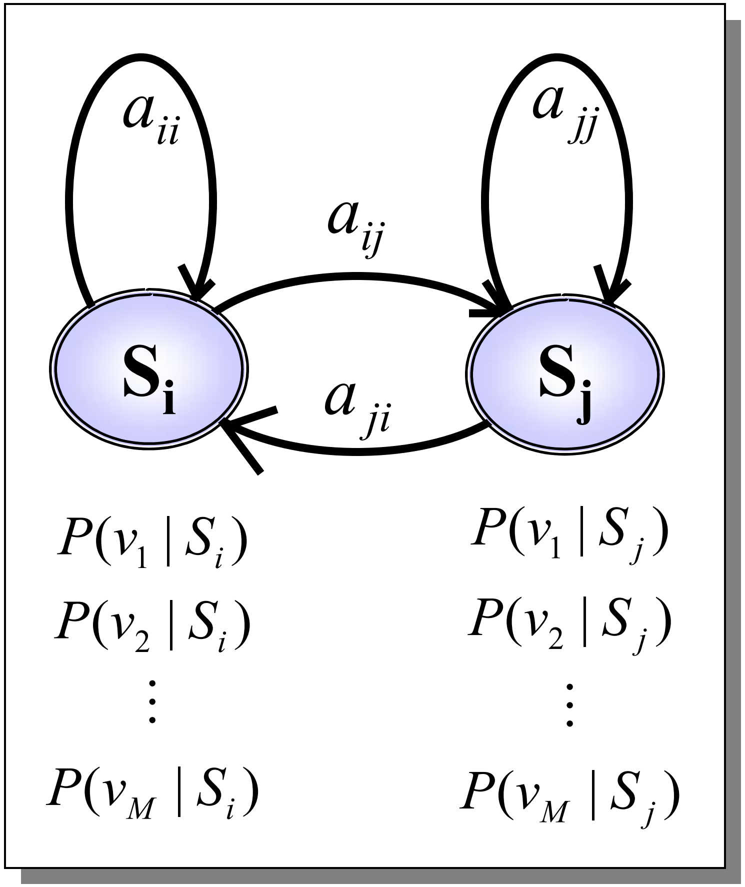
Discrete Symbol Observation HMM

- **Also characterized by:**
  - An initial state distribution

$$\pi = \{\pi_i\} = P(q_1 = i)$$

- **Specification thus requires**
  - 2 model parameters, N and M
  - Specification of M symbols
  - 3 probability measures A,B,π

$$\lambda = (A, B, \pi)$$



Three Interesting HMM Problems

- **Problem 1: Scoring & Evaluation**
  - How to efficiently compute the probability of an observation sequence (O) given a model (λ)? → $P(O|\lambda)$
- **Problem 2: Decoding**
  - Given an observation sequence (O) and a model (λ), how do we determine the corresponding state-sequence (q) that "best explains" how the observations were generated?
- **Problem 3: Training**
  - How to adjust model parameters ($\lambda = \{A,B,\pi\}$) to maximize probability of generating a given observation sequence? → maximize $P(O|\lambda)$.

Problem 2: Decoding

- **Given an observation sequence,**

$$\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \cdots, \mathbf{o}_T\}$$

- **Find the single best sequence of states,**

$$q = \{q_1, q_2, \cdots, q_T\}$$

- **Which maximizes,**

$$P(\mathbf{O}, q \mid \lambda)$$

Viterbi Algorithm

1. **Initialization** $\delta_1(i) = \pi_i b_i(\mathbf{o}_1)$ $\psi_1(i) = 0$

2. **Recursion**

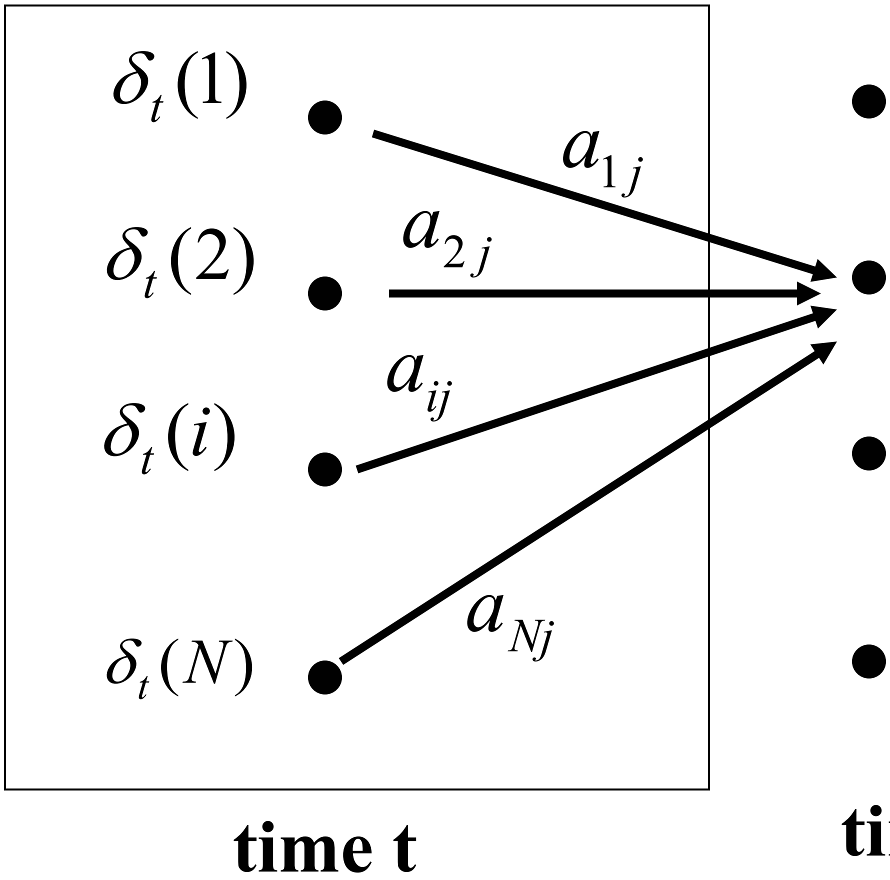
$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(\mathbf{o}_t)$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}]$$

3. **Termination** $P^* = \max_{1 \le i \le N} \left[\delta_T(i)\right]$ $q_T^* = \arg\max_{1 \le i \le N} \left[\delta_T(i)\right]$

4. **Path Back trace** $q_t^* = \psi_{t+1}(q_{t+1}^*)$

Viterbi Algorithm Illustration



$$\delta_{t+1}(j) = \max_{1 \le i \le N}[\delta_t(i)a_{ij}]b_j(\mathbf{o}_{t+1})$$

Viterbi Algorithm in Log-Domain

1. Initialization $\tilde{\delta}_1(i) = \tilde{\pi}_i + \tilde{b}_i(\mathbf{o}_1) \quad \psi_1(i) = 0$

2. Recursion

$$\delta_t(j) = \max_{1 \le i \le N}[\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}] + \tilde{b}_j(\mathbf{o}_t)$$

$$\psi_t(j) = \arg\max_{1 \le i \le N}[\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}]$$

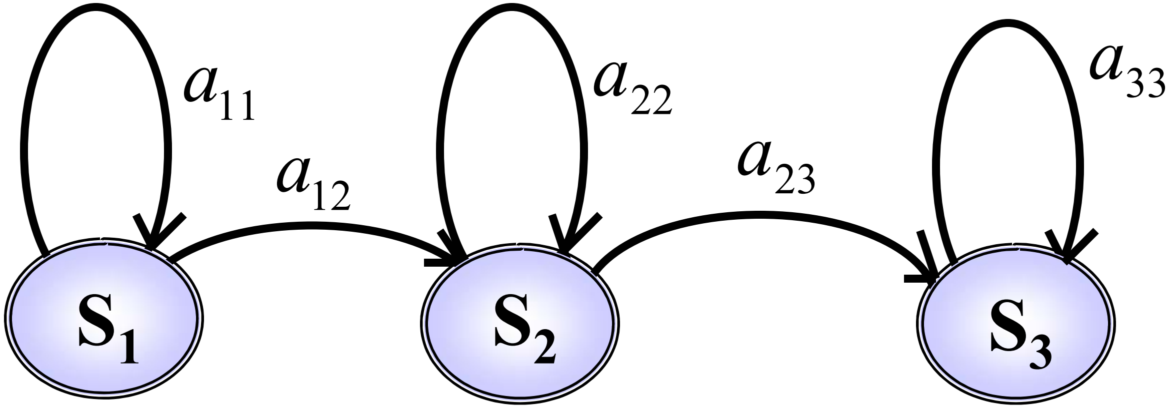
3. Termination $\tilde{P}^* = \max_{1 \le i \le N}\left[\tilde{\delta}_T(i)\right] \quad q_T^* = \arg\max_{1 \le i \le N}\left[\tilde{\delta}_T(i)\right]$

4. Path Back trace $q_t^* = \psi_{t+1}(q_{t+1}^*)$

Acoustic Modeling

Phoneme HMM

- **Let's assume each phoneme is represented by 3 HMM states connected with forward transitions,**



- **S1 models the beginning part of the sound, S2 the middle, and S3 the end-part of the sound unit.**

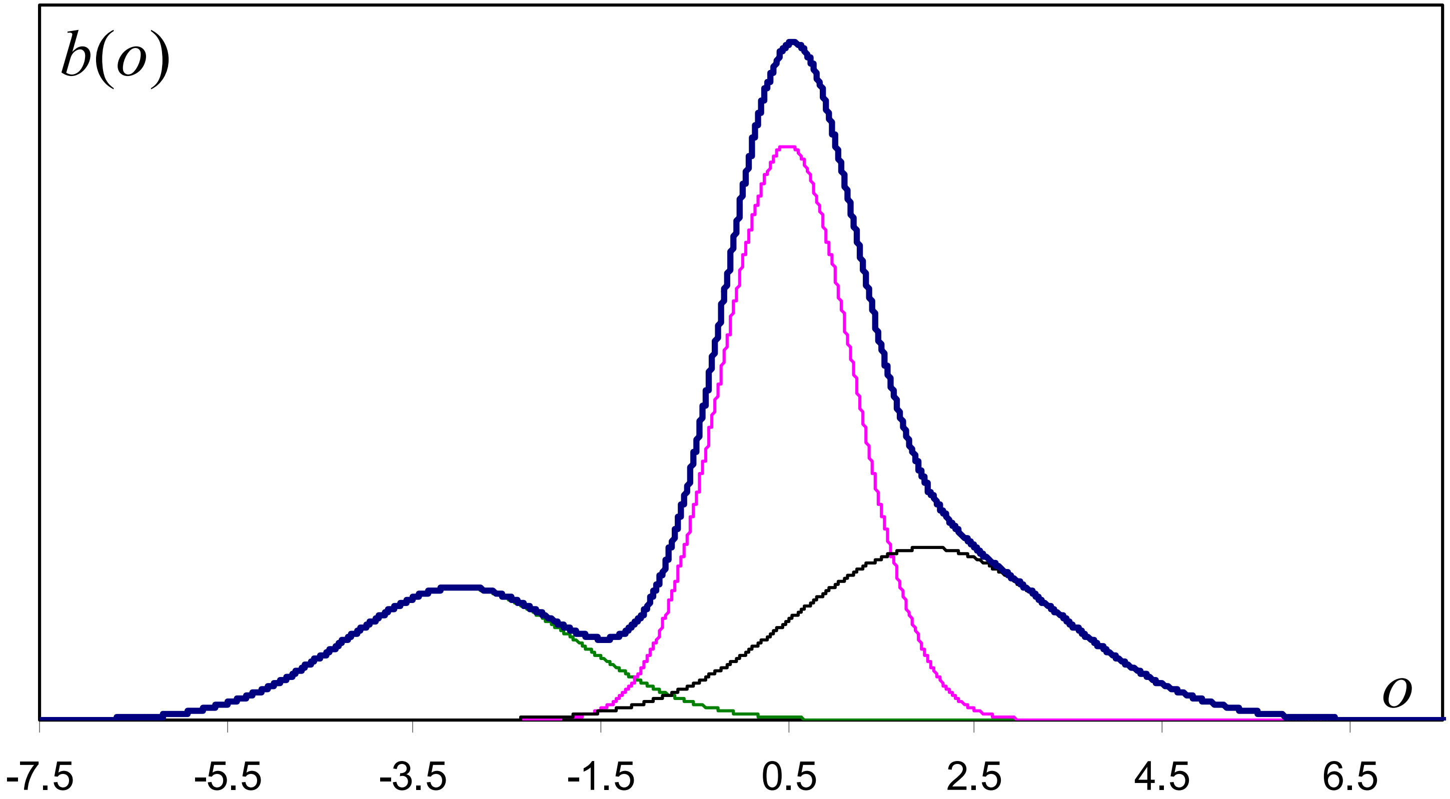
Gaussian Mixture Model

- **Single-state HMM model**
- **Observation probability a sum of M component Gaussians,**

$$b(\mathbf{o_t}) = \sum_{k=1}^{M} w_k b_k(\mathbf{o_t}, \mu_k, \Sigma_k)$$

$$= \sum_{k=1}^{M} \frac{w_k}{(2\pi)^{d/2} \left|\Sigma_k\right|^{1/2}} \exp\left(-\frac{1}{2} (o_t - u_k)' \Sigma_k^{-1} (o_t - u_k) \right)$$

Mixture Gaussian (1-D case)

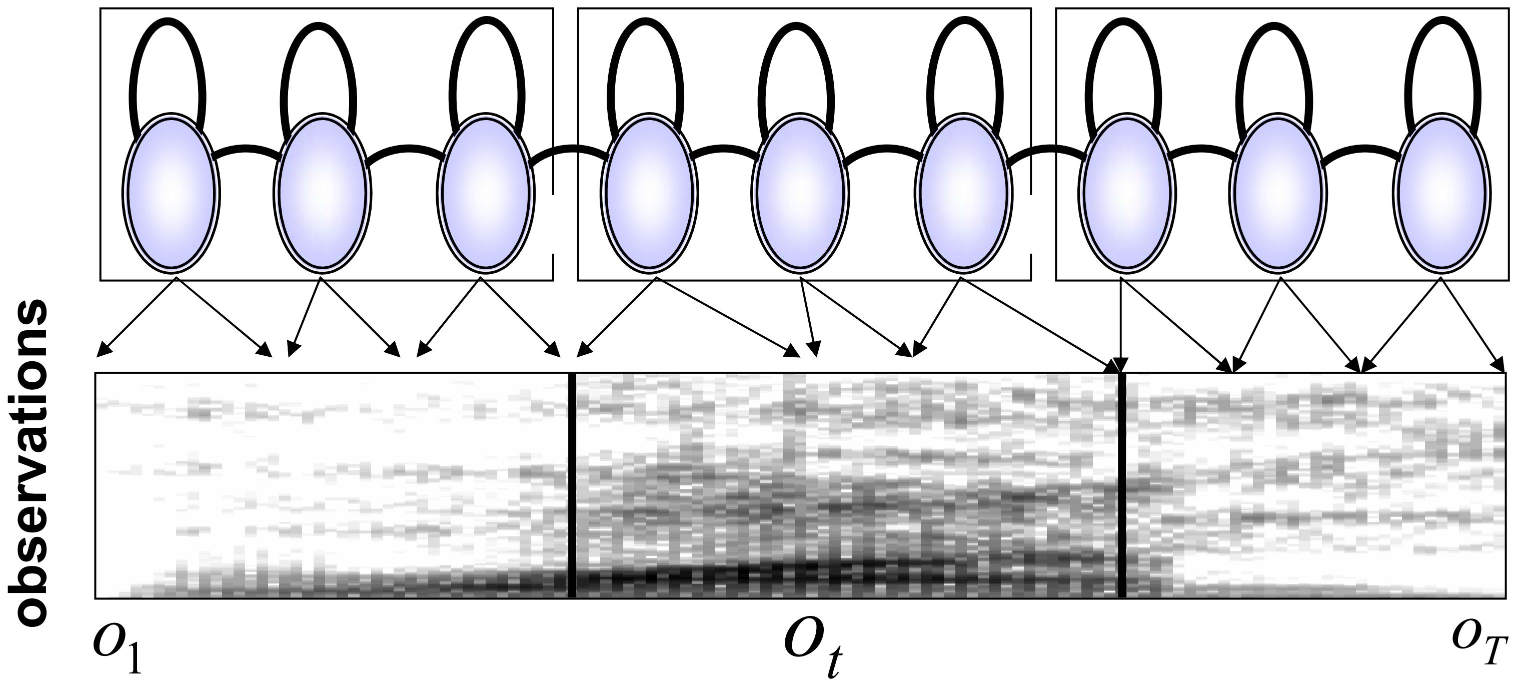


Example: 3 mixtures used to model underlying random process of 3 Gaussians

T-61.184

Viterbi Training

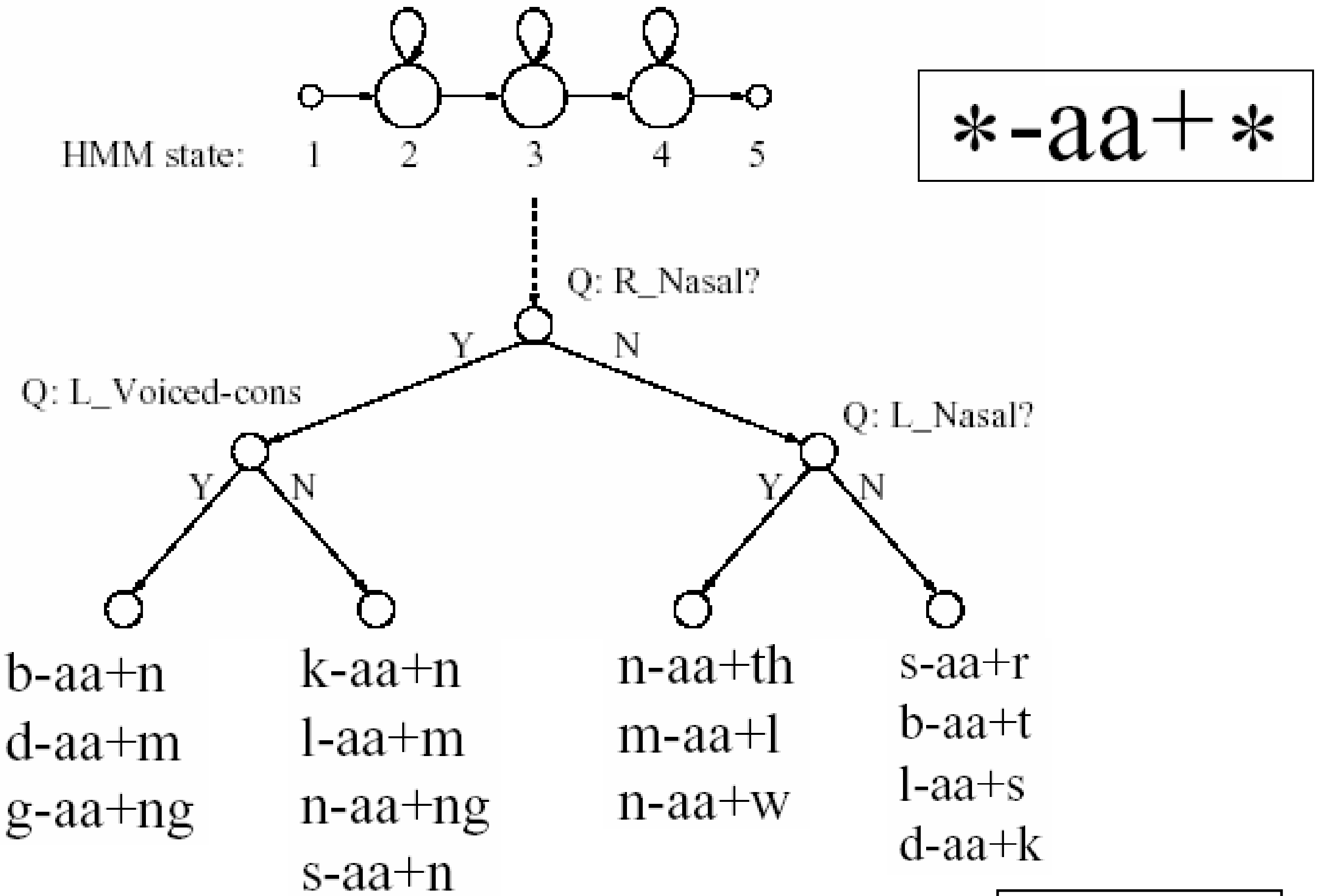
- **Given an utterance, we can construct the composite HMM from the phone units and use the Viterbi algorithm to find the best state-sequence (assignment of feature-vectors to HMM states):**



Describing Context-Dependent Phonetic Models

- **Monophone:**
  - A single model used to represent phoneme in all contexts.
- **Biphone:**
  - Each model represents a particular left or right context.
  - Left-context biphone notation: (a-b)
  - Right-context biphone notation: (b+c)
- **Triphone:**
  - Each model represents a particular left & right context.
  - (a-b+c) refers to phoneme “b” with “a” preceding and “c” immediately following.

Decision-Tree State Clustering (one tree built for each state-position)



Example Splitting Questions

$silence	SIL br ls lg ga
$aspiration	HH
$dental	DH TH
$l_w	L W
$s_sh	S SH
$s_z_sh_zh	S Z SH ZH
$affricate	CH TS JH
$nasal	M N NG
$schwa	AX IX AXR
$voiced_fric	DH Z ZH V
$voiceless_fric	TH S SH F

"Is the Left-Context an "L" or "W"?
"Is the Right-Context an "L" or "W"?

Language Modeling

Regular Expression Grammar

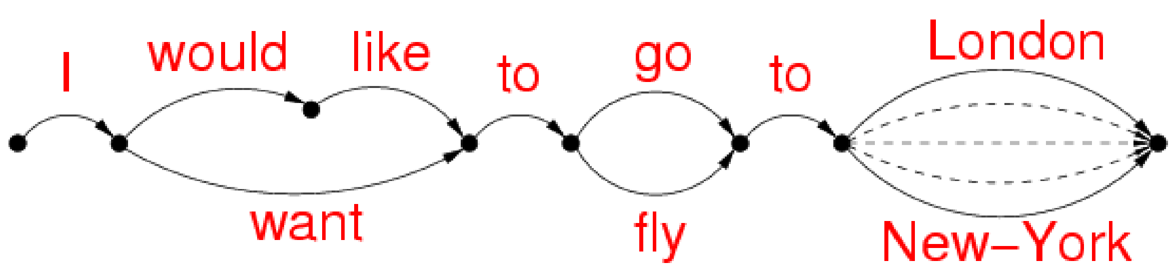
Grammar:

$$\langle\text{sentence}_1\rangle = \text{I} \left\{ \begin{array}{c} \text{would like} \\ \text{want} \end{array} \right\} \text{to} \left\{ \begin{array}{c} \text{go} \\ \text{fly} \end{array} \right\} \text{to} \ \langle\text{airport}\rangle$$

$$\langle\text{sentence}_2\rangle = \cdots$$

$$\langle\text{airport}\rangle = \{ \text{London}, \ \text{New–York}, \ldots \}$$

Finite–state representation:



N-gram Language Models

... the united states of ???

$P(\text{states} \mid \text{the united}) = \ldots$

$P(\text{of} \mid \text{united states}) = \ldots$

$P(\text{America} \mid \text{states of}) = \ldots$

$P(\text{Belgium} \mid \text{states of}) = \ldots$

Obtaining N-gram Probabilities

- **Maximum likelihood estimates of word probabilities are based on counting frequency of occurrence of word sequences from a training set of text data:**

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}, w_n)}{C(w_{n-1})}$$

$$P(w_n \mid w_{n-2}, w_{n-1}) = \frac{C(w_{n-2}, w_{n-1}, w_n)}{C(w_{n-2}, w_{n-1})}$$

Making N-grams work for Speech Recognition

- **Raw probabilities estimates from N-grams can lead to 0 probability events (as we just saw)**
- **For a vocabulary of 20,000 words, there are 400 million possible bigrams. Given a corpus of 10 million training words, there will be MANY unseen events**
- **Methods for addressing this problem:**
  - ❑ Smoothing
  - ❑ Discounting
  - ❑ Backing-off
  - ❑ Interpolation

Katz (1987) Back-off Language Model

- **Uses Good-Turing Smoothing. "Back-off" to lower-order n-grams,**

$$P_{Katz}(w_n \mid w_{n-2,} w_{n-1}) =$$

$$\begin{cases} \dfrac{C^*(w_{n-2,} w_{n-1}, w_n)}{C(w_{n-2,} w_{n-1})} & \text{if } C(w_{n-2,} w_{n-1}, w_n) > 0 \\ \alpha(w_{n-2,} w_{n-1}) \cdot P_{Katz}(w_n \mid w_{n-1}) & \text{otherwise} \end{cases}$$

- **α (back-off weight) is calculated so probabilities sum to 1**

Computing a Probability from the Back-off (N=3)-gram Model

$$P(w_n \mid w_{n-2,} w_{n-1}) =$$

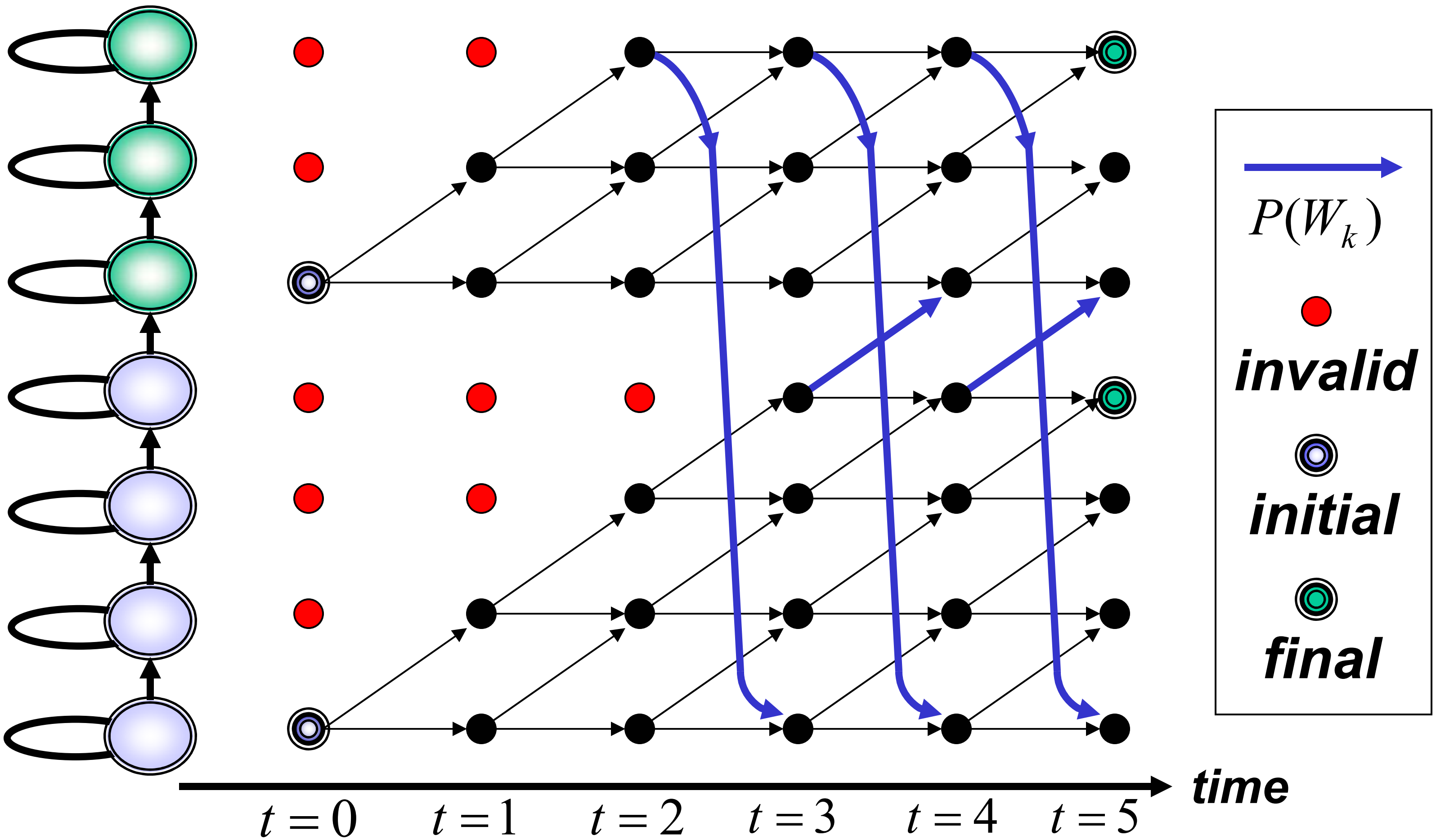
$$\left\{ \begin{array}{ll} P(w_n \mid w_{n-2,} w_{n-1}) & \text{if trigram exists, else,} \\ \alpha(w_{n\text{-}2}, w_{n-1}) P(w_n \mid w_{n-1}) & \text{if bigram } (w_{n\text{-}2}, w_{n-1}), \\ P(w_n \mid w_{n-1}) & \end{array} \right\}$$

$$P(w_n \mid w_{n-1}) =$$

$$\left\{ \begin{array}{ll} P(w_n \mid w_{n-1}) & \text{if bigram exists,} \\ \alpha(w_{n\text{-}1}) P(w_n) & \text{otherwise} \end{array} \right\}$$

Search

Connected-Word Viterbi Search



T-61.184

The Token Passing Model

- Proposed by Young et al. (1989)
- Provides a conceptually appealing framework for connected word speech recognition search
- Allows for arbitrarily complex networks to be constructed and searched
- Efficiently allows n-gram language models to be applied during search

Illustration of WLR Generation

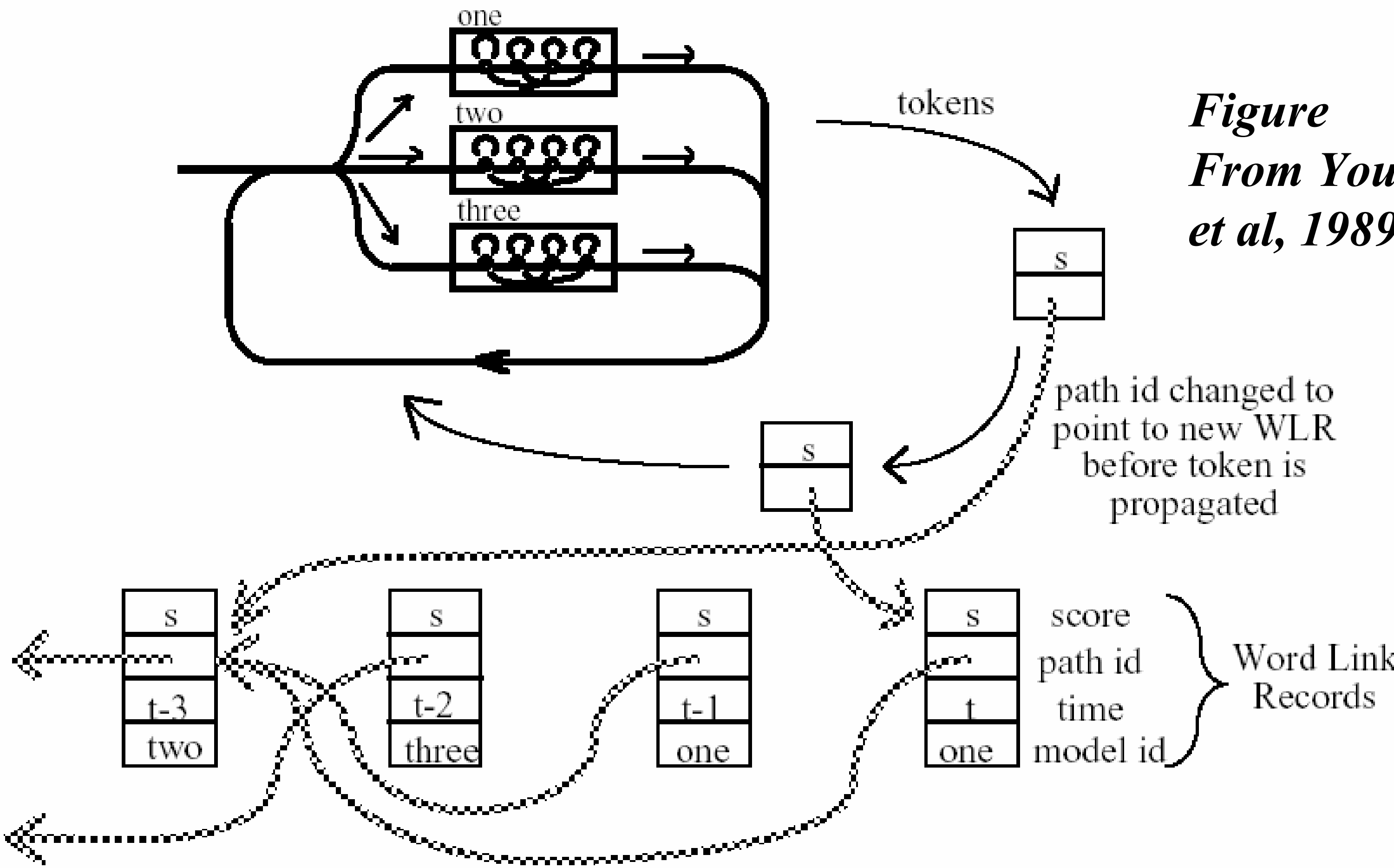


Figure From Young et al, 1989.

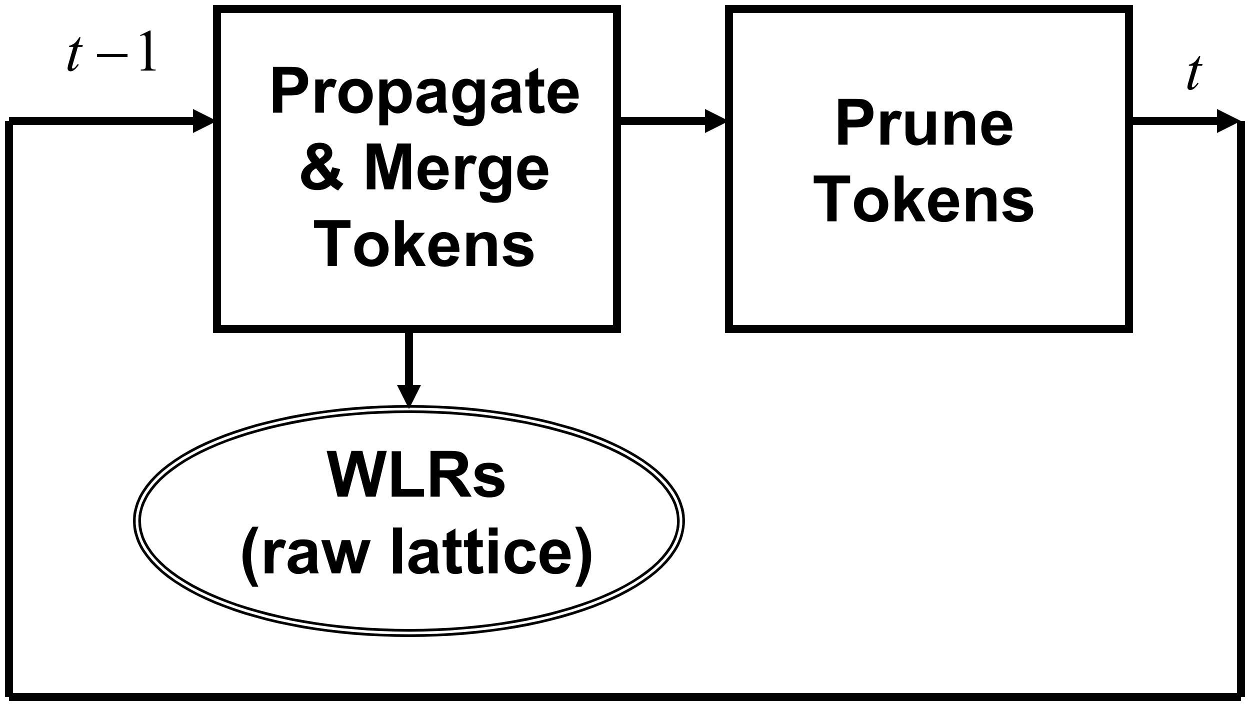
Beam Pruning for Token Passing

- **Find token with maximum partial path log-score, "s" at time "t".**
- **Prune away tokens that have score less than a threshold, e.g.,**

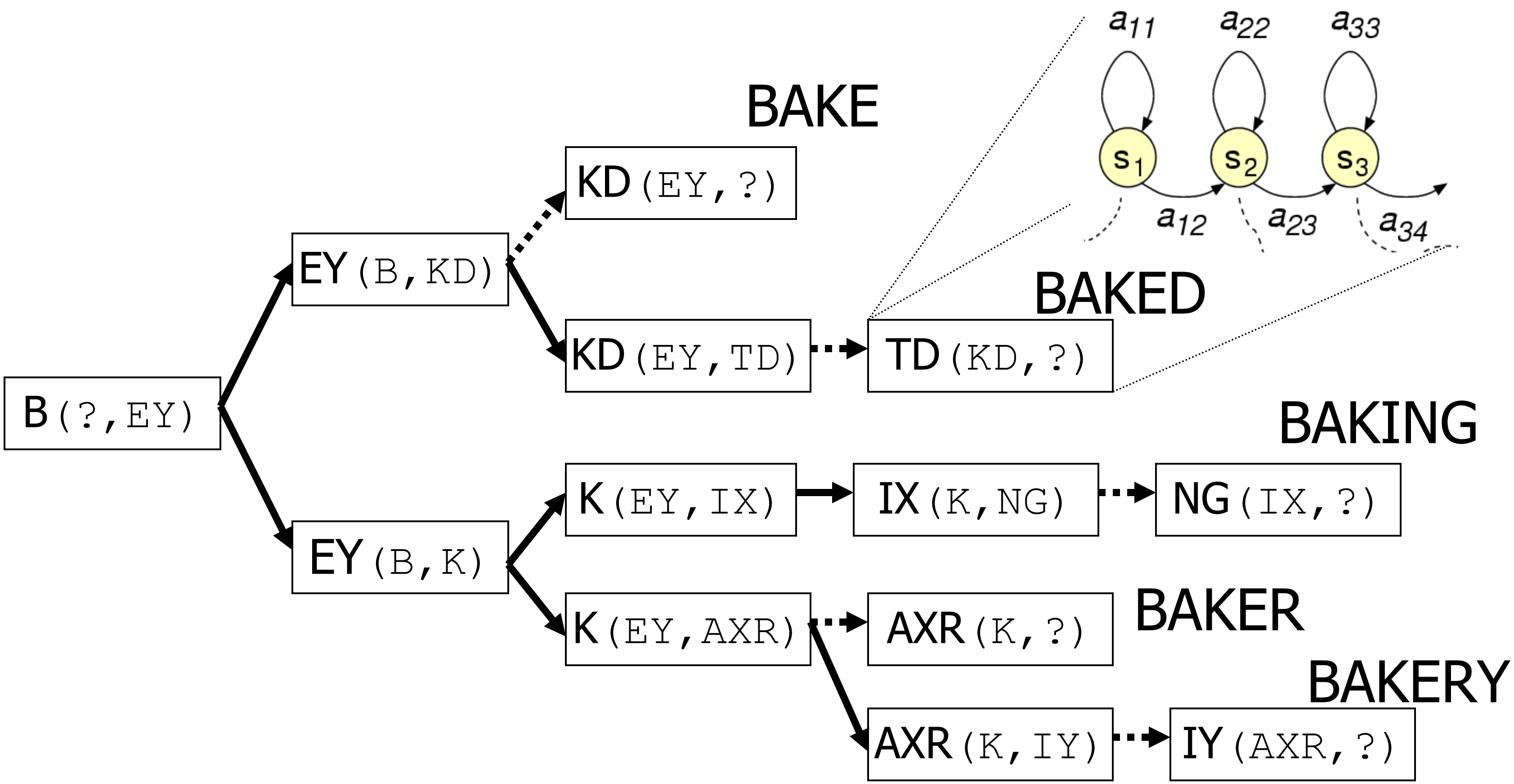
$$\text{prune} \quad \text{if} \quad s < (s_{\max} - BW)$$

- **BW is preset "beam width"**
- **BW > 0**

Typical Token Passing Search Loop



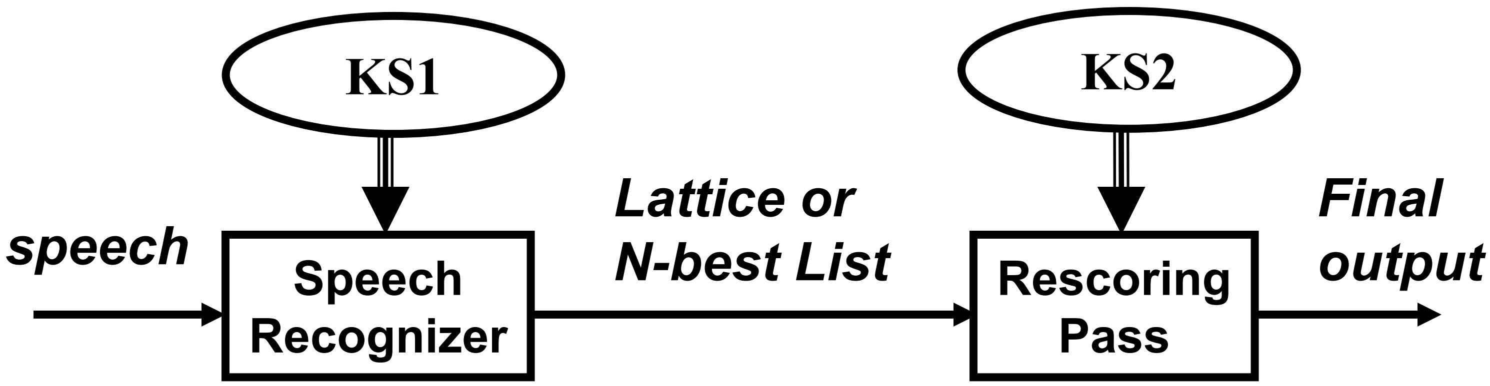
Efficient Search Representations



** Figure adapted from Huang et al., Spoken Language Processing, Prentice Hall*

Multi-Pass Search

- Step 1: Use Knowledge Source (KS) #1 to generate a reduced hypothesis space
- Step 2: Rescore resulting hypothesis space with Knowledge Source #2.



Robust Acoustic Modeling & Adaptation

Robust Acoustic Modeling

- **Robust Front-end Processing**
  - Remove noise from speech
  - Design features to be as noise, channel robust as possible
- **Feature Compensation**
  - Reduce the observed mismatch between the extracted features and estimated model parameters
- **Acoustic Model Compensation**
  - Modify acoustic model parameters to closer match the observed test environment

Robust Acoustic Modeling