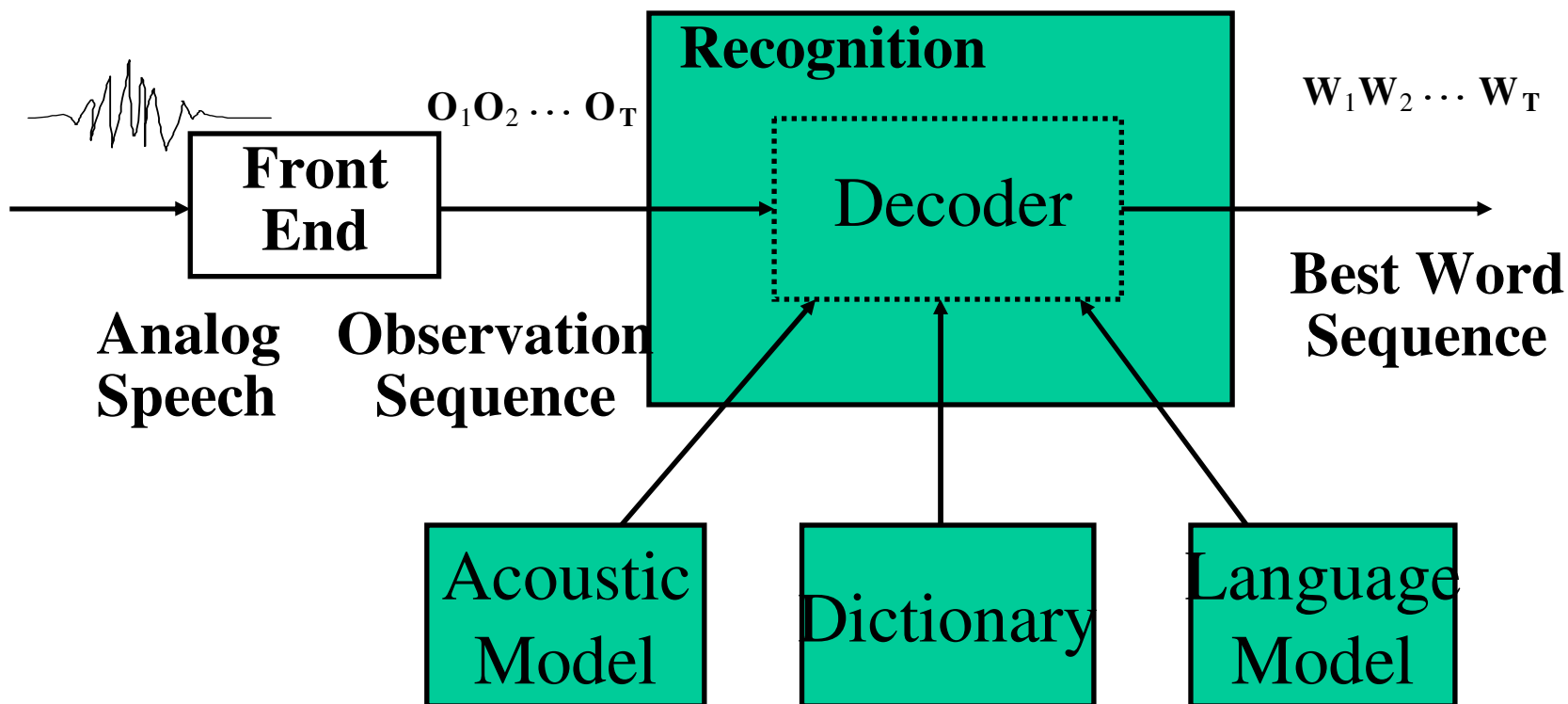


Hidden Markov Model (I)

1. Oct. 2008

Speech Recognition (System Components)

- Recognizer Components:



Speech Recognition

- Goal:
 - Given acoustic data $A = a_1, a_2, \dots, a_k$
 - Find word sequence $W = w_1, w_2, \dots, w_n$
 - Such that $P(W | A)$ is maximized

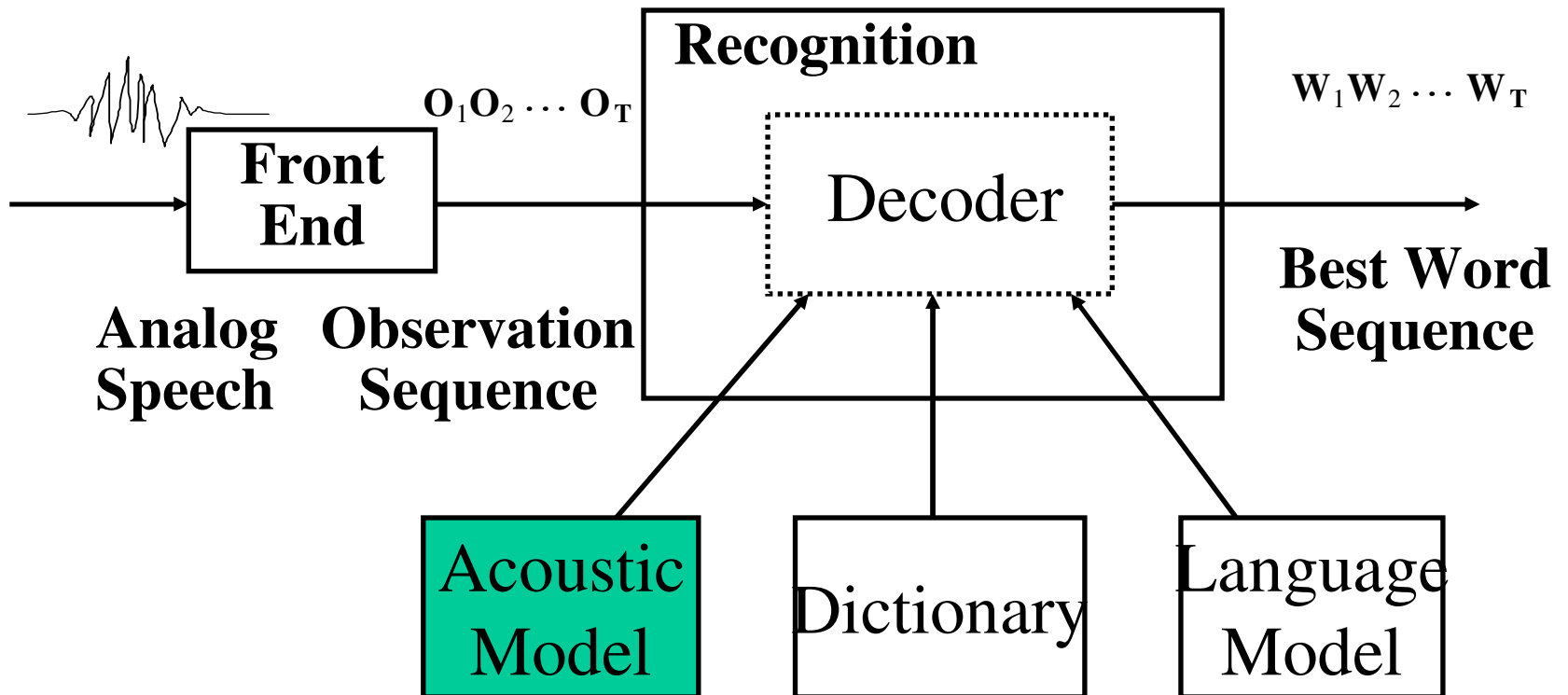
Bayes Rule:

$$P(W | A) = \frac{\overset{\text{acoustic model (HMMs)}}{P(A | W)} \cdot \overset{\text{language model}}{P(W)}}{P(A)}$$

$P(A)$ is a constant for a complete sentence

Speech Recognition (Components)

- Recognizer Components:



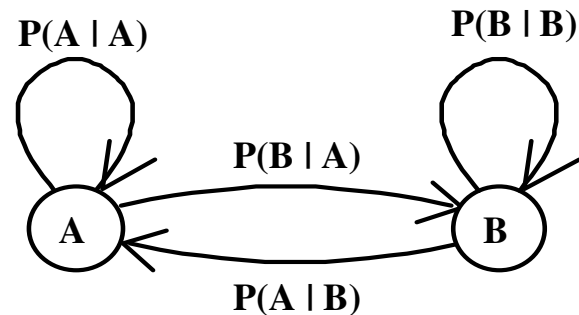
Markov Models

Elements:

States :

$$\mathbf{S} = \{S_0, S_1, \dots, S_N\}$$

Transition probabilities : $P(q_t = S_i \mid q_{t-1} = S_j)$



Markov Assumption:

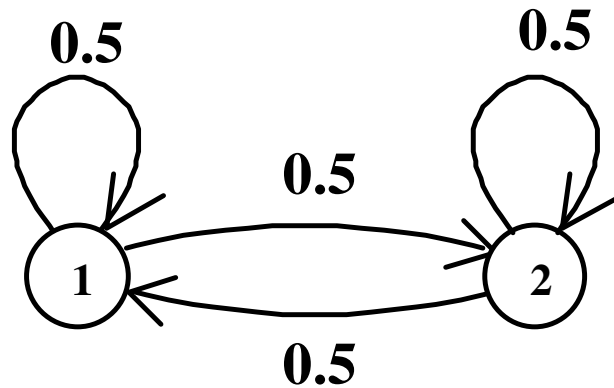
Transition probability depends only on current state

$$P(q_t = S_i \mid q_{t-1} = S_j, q_{t-2} = S_k, \dots) = P(q_t = S_i \mid q_{t-1} = S_j) = a_{ji}$$

$$a_{ji} \geq 0 \quad \forall j, i \quad \sum_{i=0}^N a_{ji} = 1 \quad \forall j$$

Single Fair Coin

- Outcome head corresponds to state 1, tail to state 2
- Observation sequence uniquely defines state sequence



$$P(H) = 1.0$$

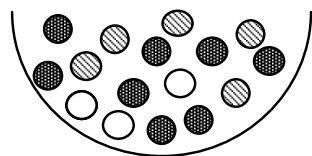
$$P(H) = 0.0$$

$$P(T) = 0.0$$

$$P(T) = 1.0$$

Discrete Observation HMM

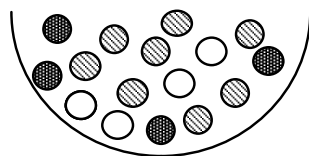
- Observation sequence: RRRBYBBYYY ... R
- not unique to state sequence



$$P(R) = 0.31$$

$$P(B) = 0.50$$

$$P(Y) = 0.19$$

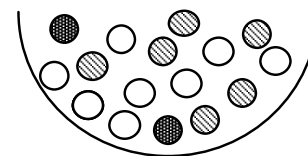


$$P(R) = 0.50$$

$$P(B) = 0.25$$

$$P(Y) = 0.25$$

...



$$P(R) = 0.38$$

$$P(B) = 0.12$$

$$P(Y) = 0.50$$

Hidden Markov Models

- Elements:

- States

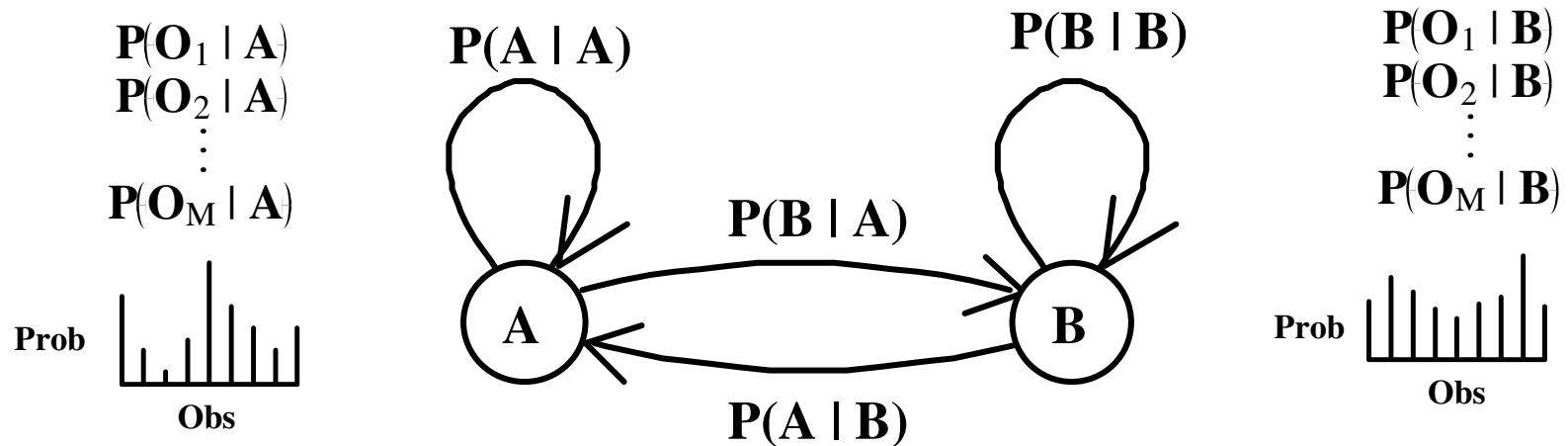
$$S = \{S_0, S_1, \dots, S_N\}$$

- Transition probabilities

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

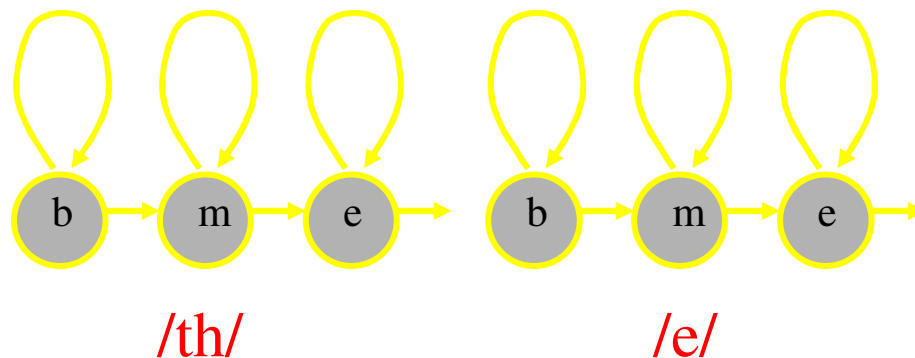
- Output prob distributions
(at state j for symbol k)

$$P(y_t = O_k \mid q_t = S_j) = b_j(k)$$



Acoustic Modeling

- Hidden Markov Models:
 - Words, Phonemes, States
 - Observation Probability: $P(\text{feature vector} \mid \text{state})$



HMM Problems And Solutions

- Evaluation:
 - Problem - Compute Probability of observation sequence given a model
 - Solution - **Forward Algorithm** and **Viterbi Algorithm**
- Decoding:
 - Problem - Find state sequence which maximizes probability of observation sequence
 - Solution - **Viterbi Algorithm**
- Training:
 - Problem - Adjust model parameters to maximize probability of observed sequences
 - Solution - **Forward-Backward Algorithm**

Evaluation

Probability of observation sequence

given HMM model λ is : $\mathbf{O} = O_1 O_2 \dots O_T$

$$\mathbf{P}(\mathbf{O} | \lambda) = \sum_{\forall \mathbf{Q}} \mathbf{P}(\mathbf{O}, \mathbf{Q} | \lambda)$$

where $\mathbf{Q} = q_0 q_1 \dots q_T$ is a sequence of states

$$= \sum_{\forall q_0, \dots, q_T} a_{q_0 q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \dots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

Not practical since the number of paths $O(N^T)$

is where N = number of states in model

and T = number of observations in sequence

The Forward Algorithm

$$\alpha_t(j) = P(O_1 O_2 \cdots O_t, q_t = S_j \mid \lambda)$$

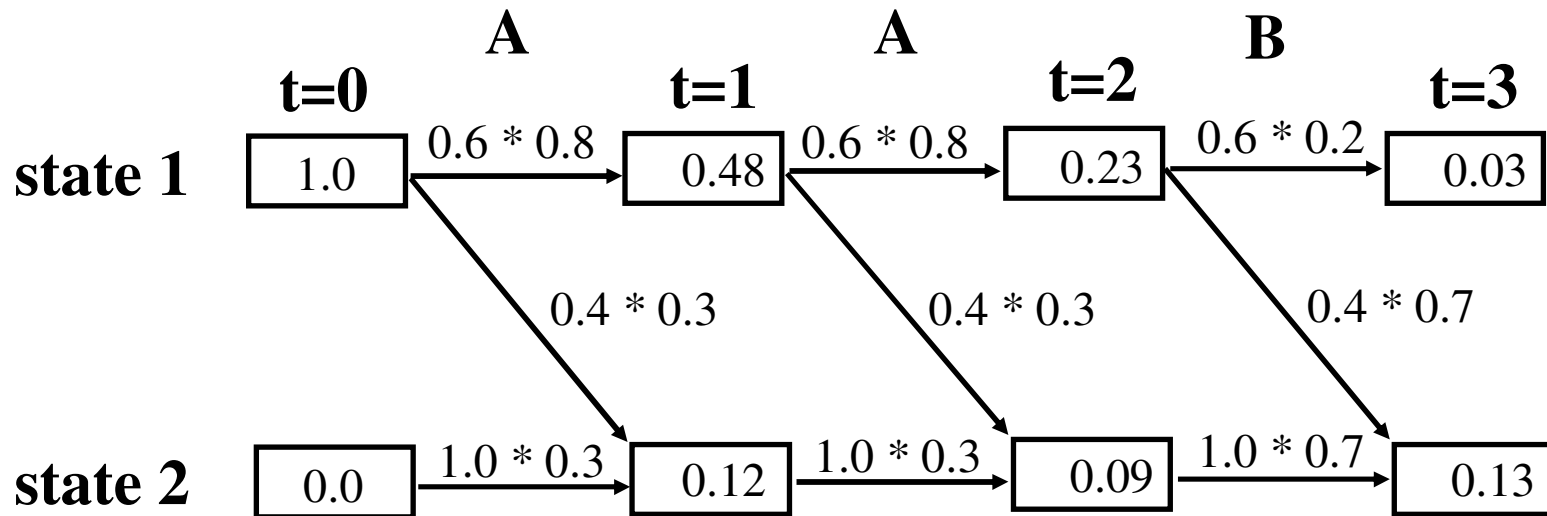
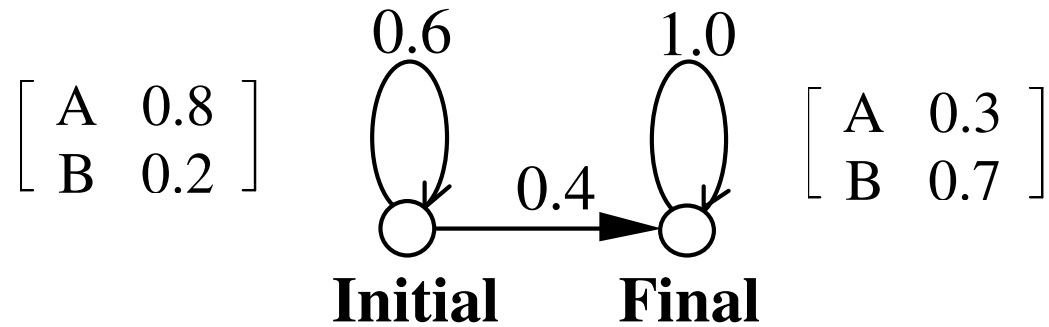
Compute α recursively:

$$\alpha_0(j) = \begin{cases} \mathbf{1} & \text{if } j \text{ is start state} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\alpha_t(j) = \left[\sum_{i=0}^N \alpha_{t-1}(i) a_{ij} \right] b_j(O_t) \quad \mathbf{t} > \mathbf{0}$$

$$P(O \mid \lambda) = \alpha_T(S_N) \quad \text{Computation is } O(N^2 T)$$

Forward Trellis



The Backward Algorithm

$$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T, q_t = S_i | \lambda)$$

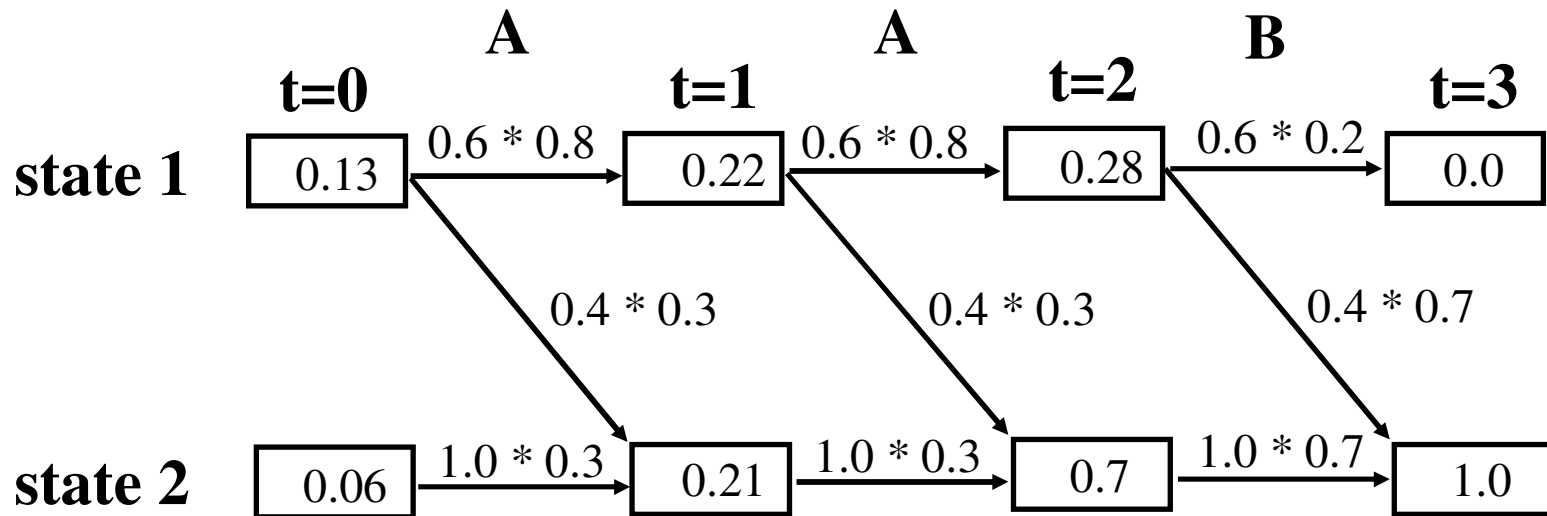
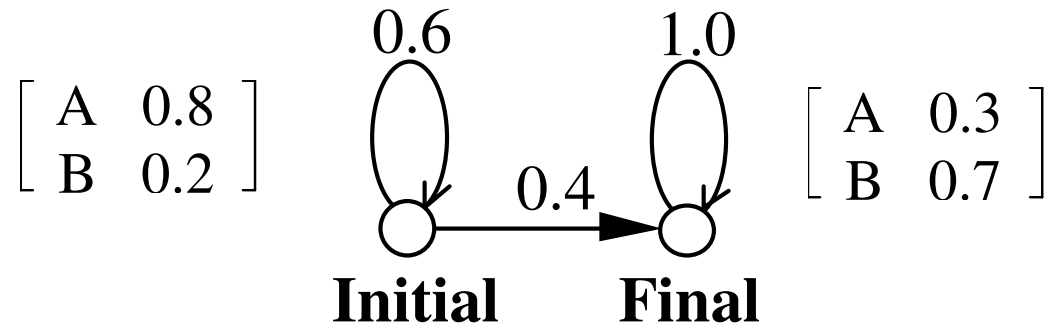
Compute α recursively:

$$\beta_T(i) = \begin{cases} \mathbf{1} & \text{if } i \text{ is end state} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\beta_t(i) = \sum_{j=0}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad \mathbf{t} < \mathbf{T}$$

$$P(O | \lambda) = \beta_0(S_0) = \alpha_T(S_N) \quad \text{Computation is } O(N^2 T)$$

Backward Trellis



Decoding

The Viterbi Algorithm:

- Find the state sequence **Q** which maximizes **P(O, Q | λ)**
- Similar to Forward Algorithm except **MAX** instead of **SUM**

$$VP_t(i) = \text{MAX}_{q_0, \dots, q_{t-1}} P(O_1 O_2 \dots O_t, q_t=i | \lambda)$$

Recursive Computation:

$$VP_t(j) = \text{MAX}_{i=0, \dots, N} VP_{t-1}(i) a_{ij} b_j(O_t) \quad t > 0$$

$$P(O, Q | \lambda) = VP_T(S_N)$$

Save each maximum for backtrace at end

Viterbi Trellis

