CS114 Lecture 7

HMMs

February 5, 2013

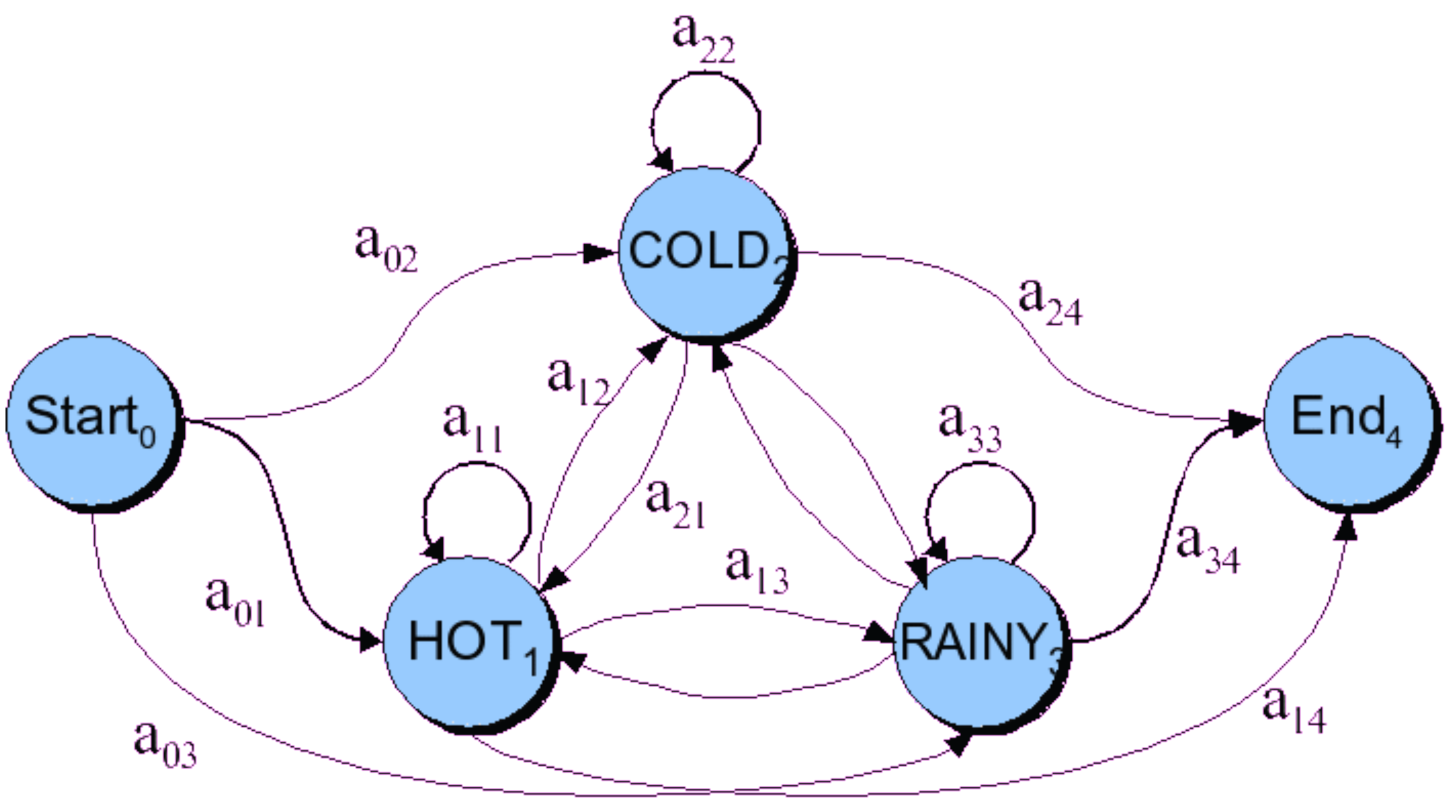
Professor Meteer

Thanks for Jurafsky & Martin & Prof. Pustejovksy for slides

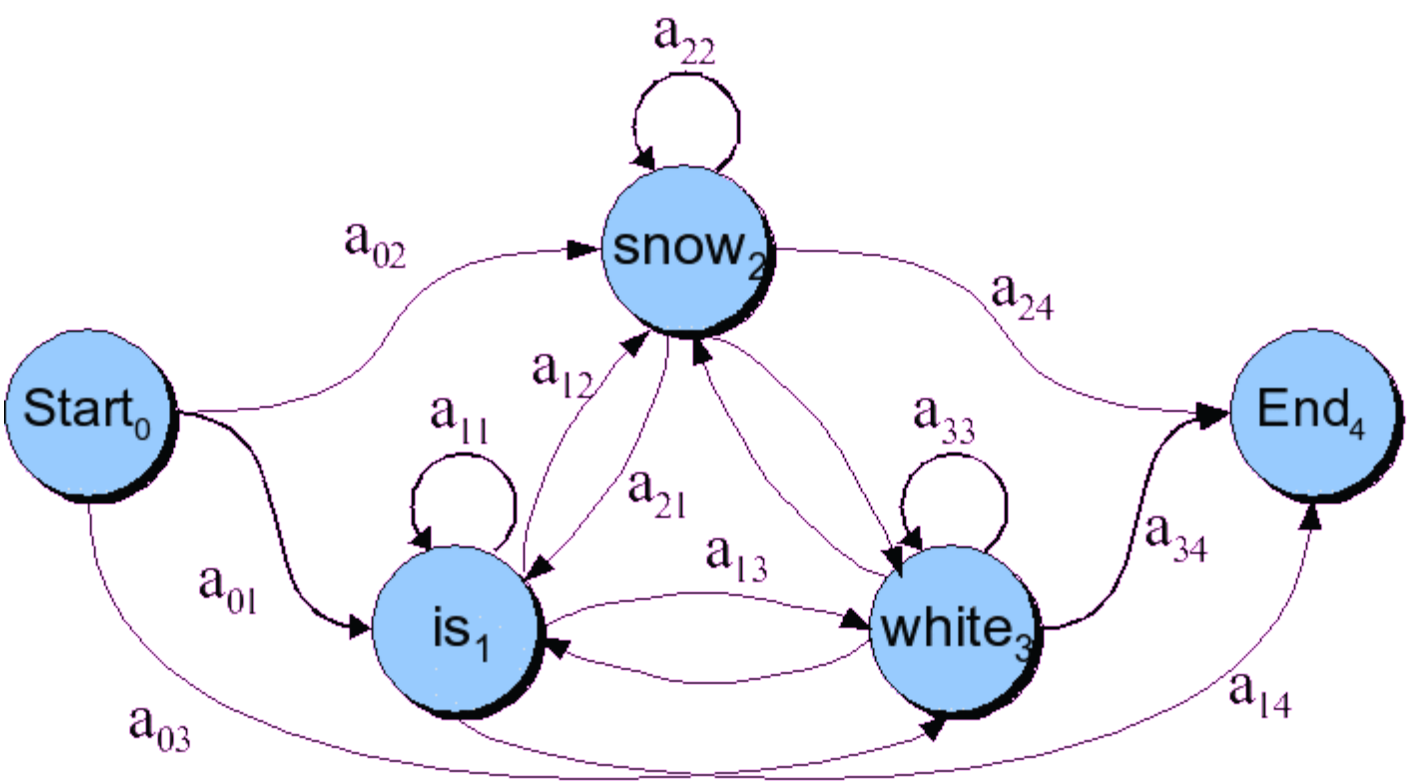
Definitions

- A weighted finite-state automaton adds probabilities to the arcs
  - The sum of the probabilities leaving any arc must sum to one
- A Markov chain is a special case of a WFST in which the input sequence uniquely determines which states the automaton will go through
- Markov chains can't represent inherently ambiguous problems
  - Useful for assigning probabilities to unambiguous sequences

Markov Chain for Weather



Markov Chain for Words



Markov Chain: "First-order observable Markov Model"

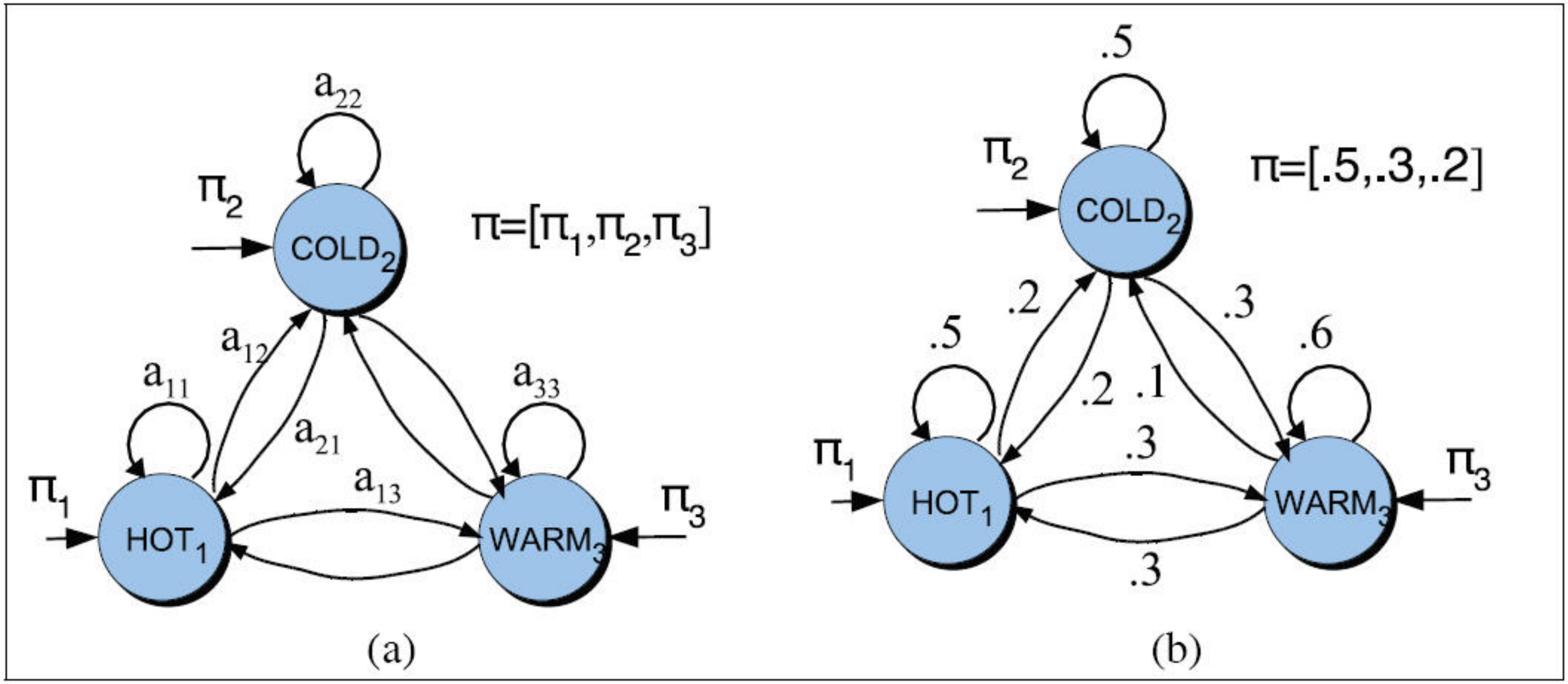
- A set of states
  - $Q = q_1, q_2 \ldots q_{N;}$ the state at time t is q_t
- Transition probabilities:
  - a set of probabilities $A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$.
  - Each a_{ij} represents the probability of transitioning from state i to state j
  - The set of these is the transition probability matrix A
- Current state only depends on previous state

$$P(q_i \mid q_1 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$$

Markov Chain for Weather

- What is the probability of 4 consecutive rainy days?
- Sequence is rainy-rainy-rainy-rainy
- I.e., state sequence is 3-3-3-3
- P(3,3,3,3) =
  - $\pi_1 a_{11} a_{11} a_{11} a_{11} = 0.2 \times (0.6)^3 = 0.0432$

Markov Chain for Weather



Hidden Markov Model

- For Markov chains, the output symbols are the same as the states.
  - See **hot** weather: we're in state **hot**
- But in part-of-speech tagging (and other things)
  - The output symbols are **words**
  - But the hidden states are **part-of-speech tags**
- So we need an extension!
- A Hidden Markov Model is an extension of a Markov chain in which the input symbols are not the same as the states.
- This means we don't know which state we are in.

HMM for Ice Cream

- You are a climatologist in the year 2799
- Studying global warming
- You can't find any records of the weather in Baltimore, MA for summer of 2007
- But you find Jason Eisner's diary

- Which lists how many ice-creams Jason ate every date that summer
- Our job: figure out how hot it was

Hidden Markov Models

- States $Q = q_1, q_2 \ldots q_{N;}$
- Observations $O = o_1, o_2 \ldots o_{N;}$
  - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, \ldots v_V\}$
- Transition probabilities
  - Transition probability matrix $A = \{a_{ij}\}$

$$a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \le i, j \le N$$

- Observation likelihoods
  - Output probability matrix $B = \{b_i(k)\}$

$$b_i(k) = P(X_t = o_k \mid q_t = i)$$

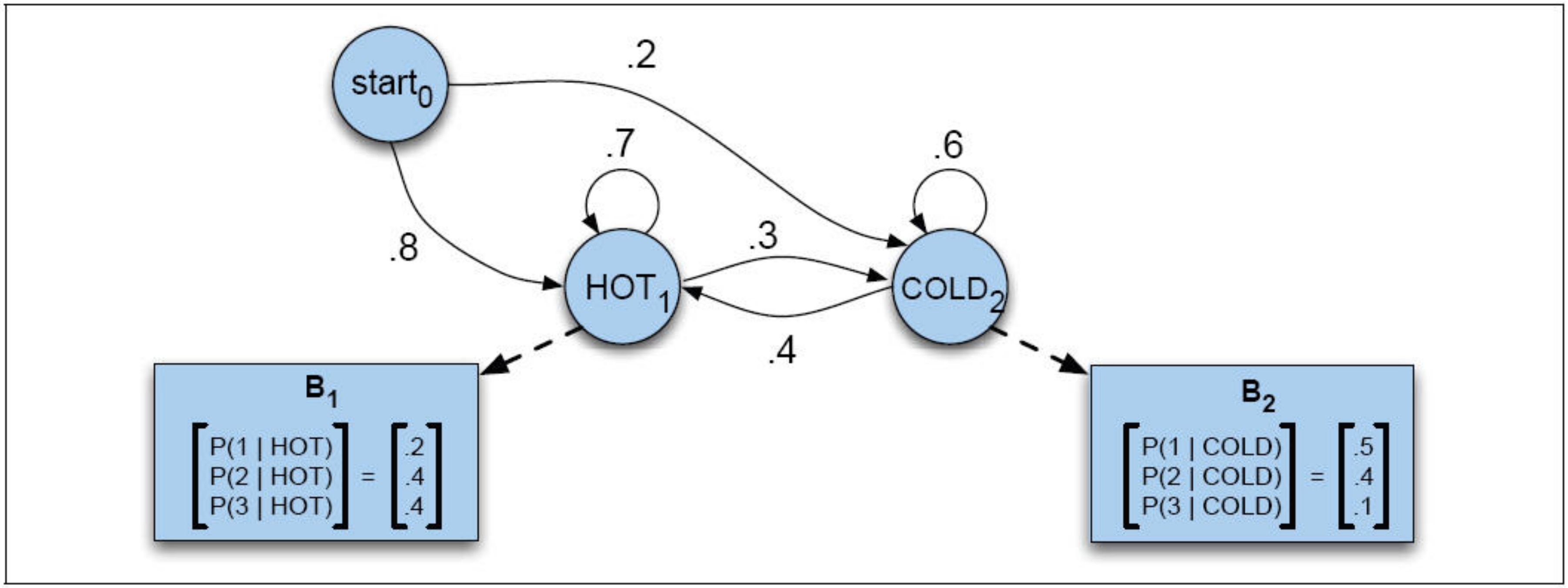
- Special initial probability vector π

$$\pi_i = P(q_1 = i) \quad 1 \le i \le N$$

Eisner Task

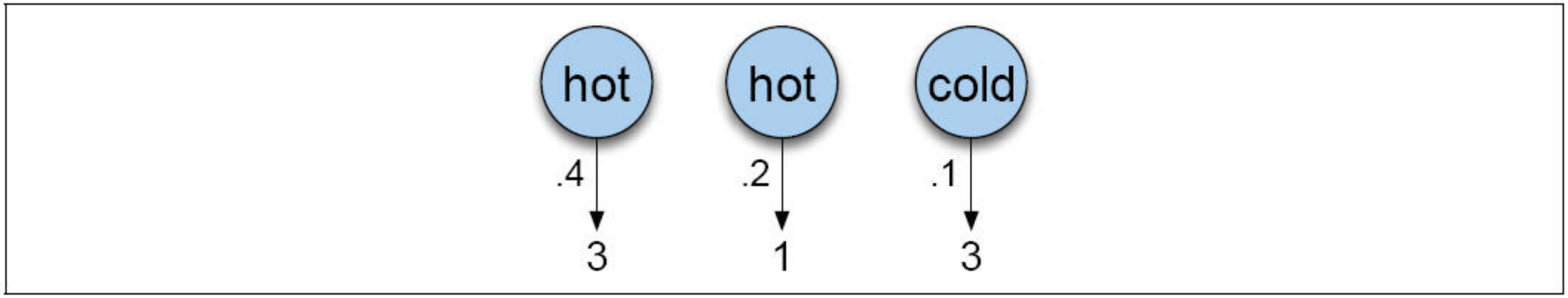
- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3…
- Produce:
  - Weather Sequence: H,C,H,H,H,C…

HMM for Ice Cream



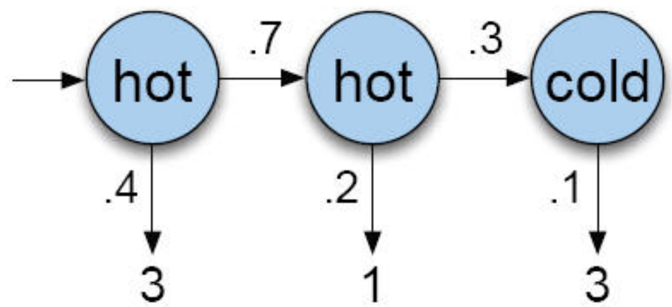
Observation Probability

Probability of events 3 - 1 – 3 given hidden states Hot Hot Cold



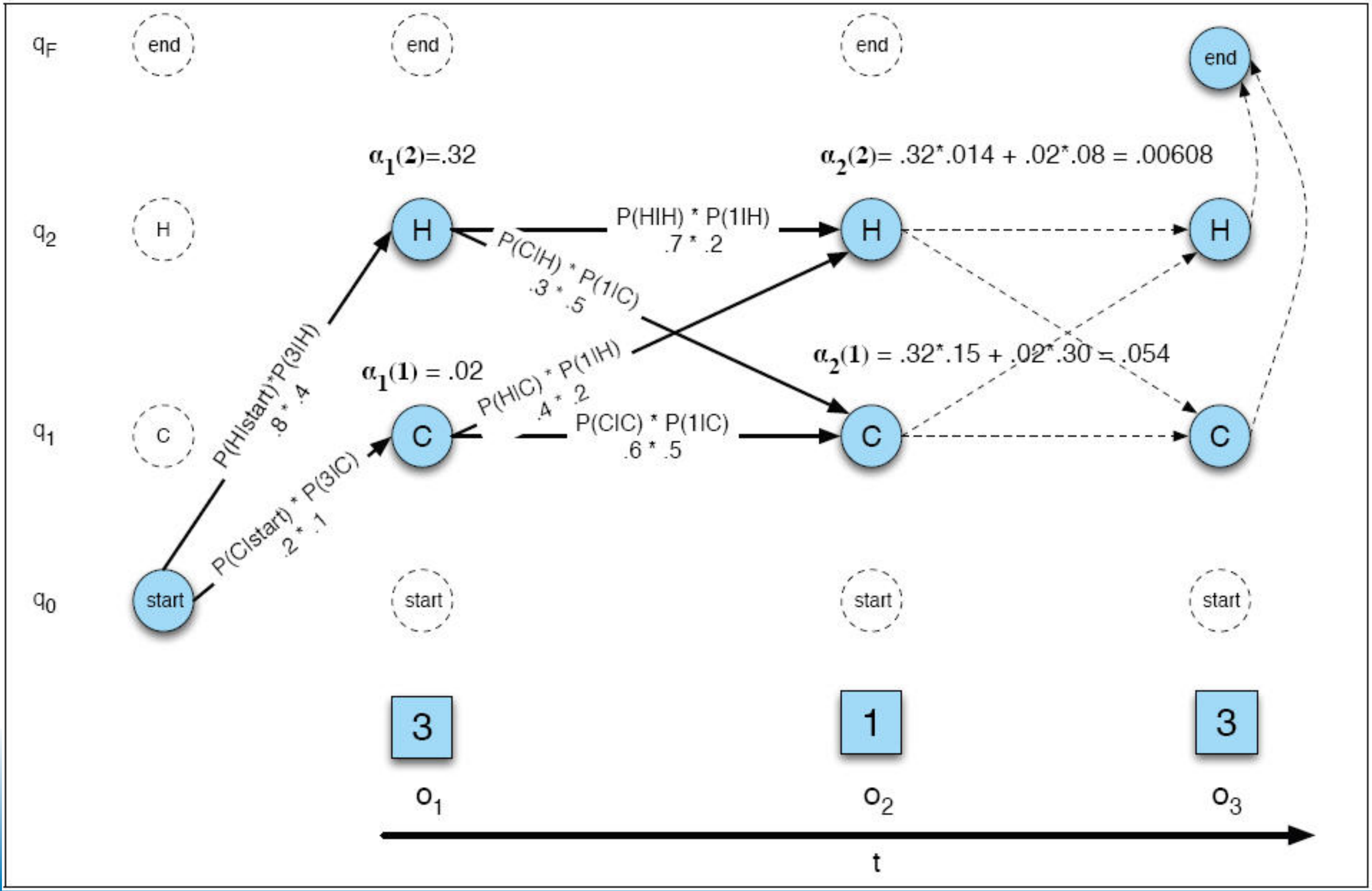
Joint probability

The computation of the joint probability of the ice cream events 3 – 1 – 3 and the hidden state sequence Hot Hot Cold



To find the most likely you would have to compute the probability for every sequence of hidden states. Too slow!

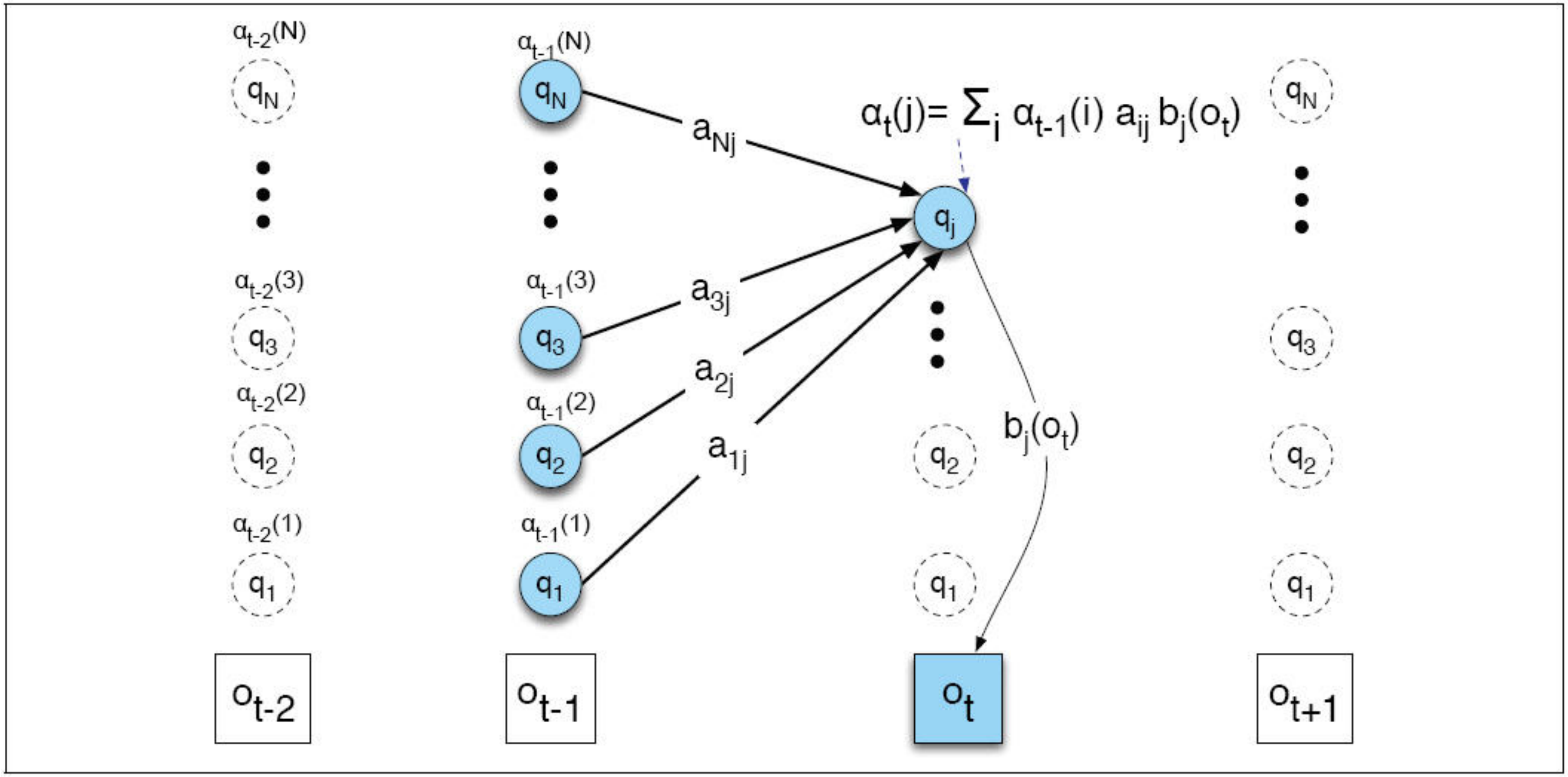
Dynamic Programming: Forward Algorithm



3 Factors

$\alpha_{i-1}(i)$	The previous forward path probability from the previous time step
a_{ij}	The transition probability from previous state q_i to current state q_j
$b_j(o_t)$	The state observation likelihood of the observation symbol o_t given the current state *j*

Forward Algorithm Computation



Forward Algorithm

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward[N+2,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$forward[s,1] \leftarrow a_{0,s} * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

$$forward[q_F,T] \leftarrow \sum_{s=1}^{N} forward[s,T] * a_{s,q_F}$$; termination step

return $forward[q_F,T]$

Factors in the Viterbi Algorithm

$v_{t-1}(i)$	The previous Viterbi path probability from the previous time step
a_{ij}	The transition probability from previous state q_i to current state q_j
$b_j(o_t)$	The stat observation likelihood of the observation symbol o_t given the current state j

3 Factors

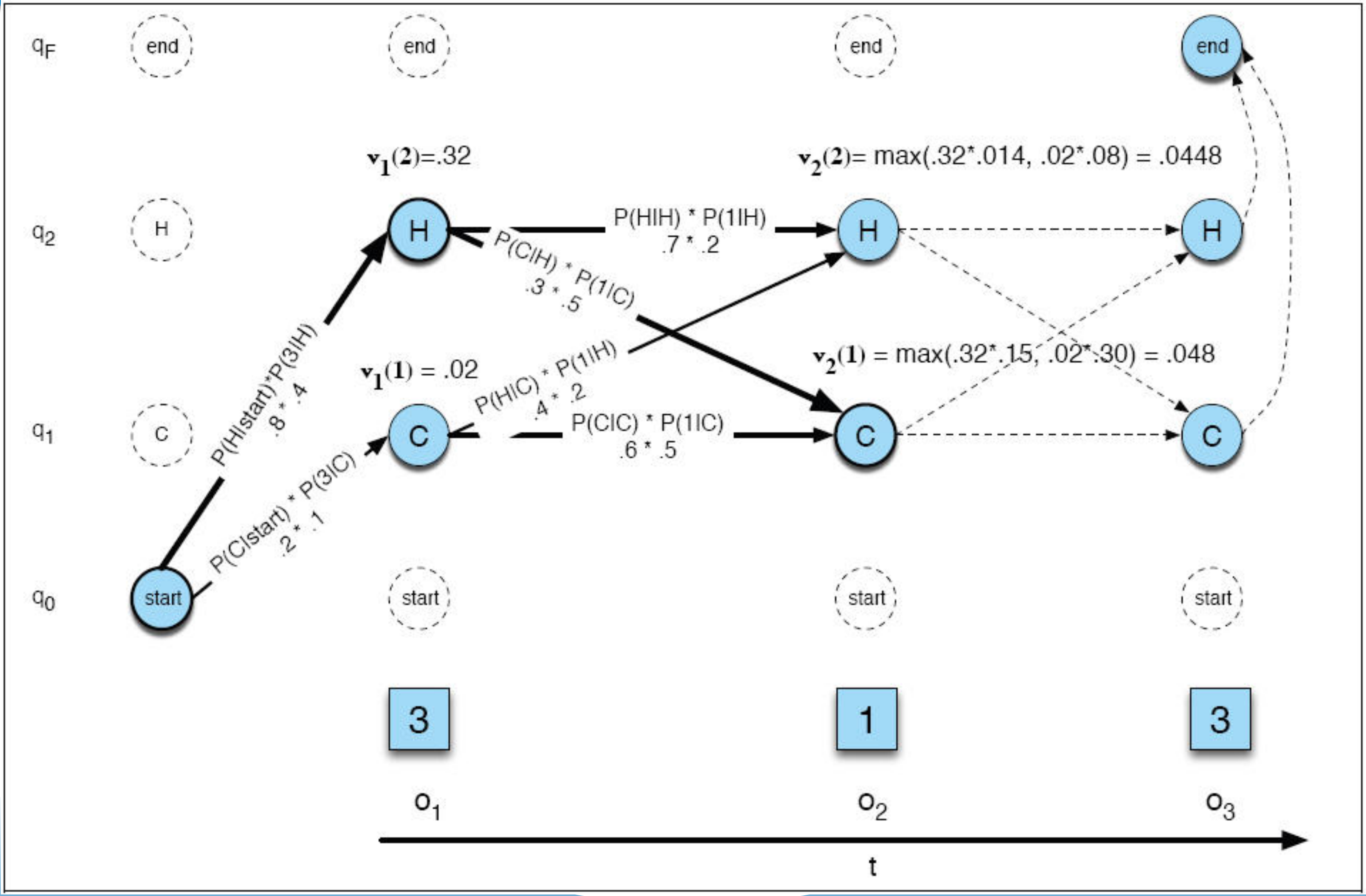
$\alpha_{i-1}(i)$	The previous forward path probability from the previous time step
a_{ij}	The transition probability from previous state q_i to current state q_j
$b_j(o_t)$	The stat observation likelihood of the observation symbol o_t given the current state j

Viterbi Algorithm

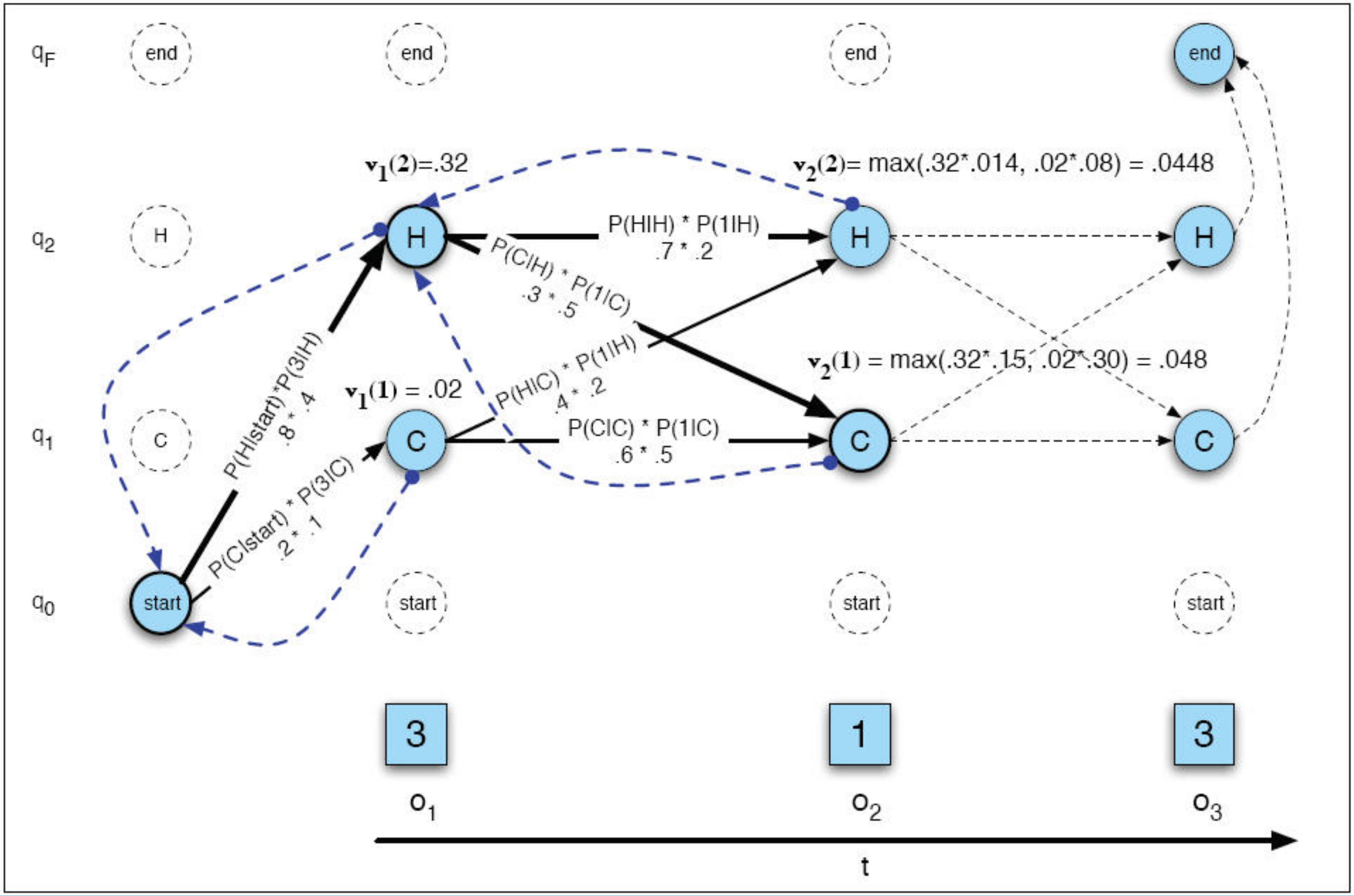
function VITERBI(*observations* of len T, *state-graph* of len N) **returns** *best-path*

create a path probability matrix *viterbi[N+2,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
$backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**
$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$$
$$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$$; termination step
$$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

Viterbi Trellis



Viterbi Trellis with Backtrace

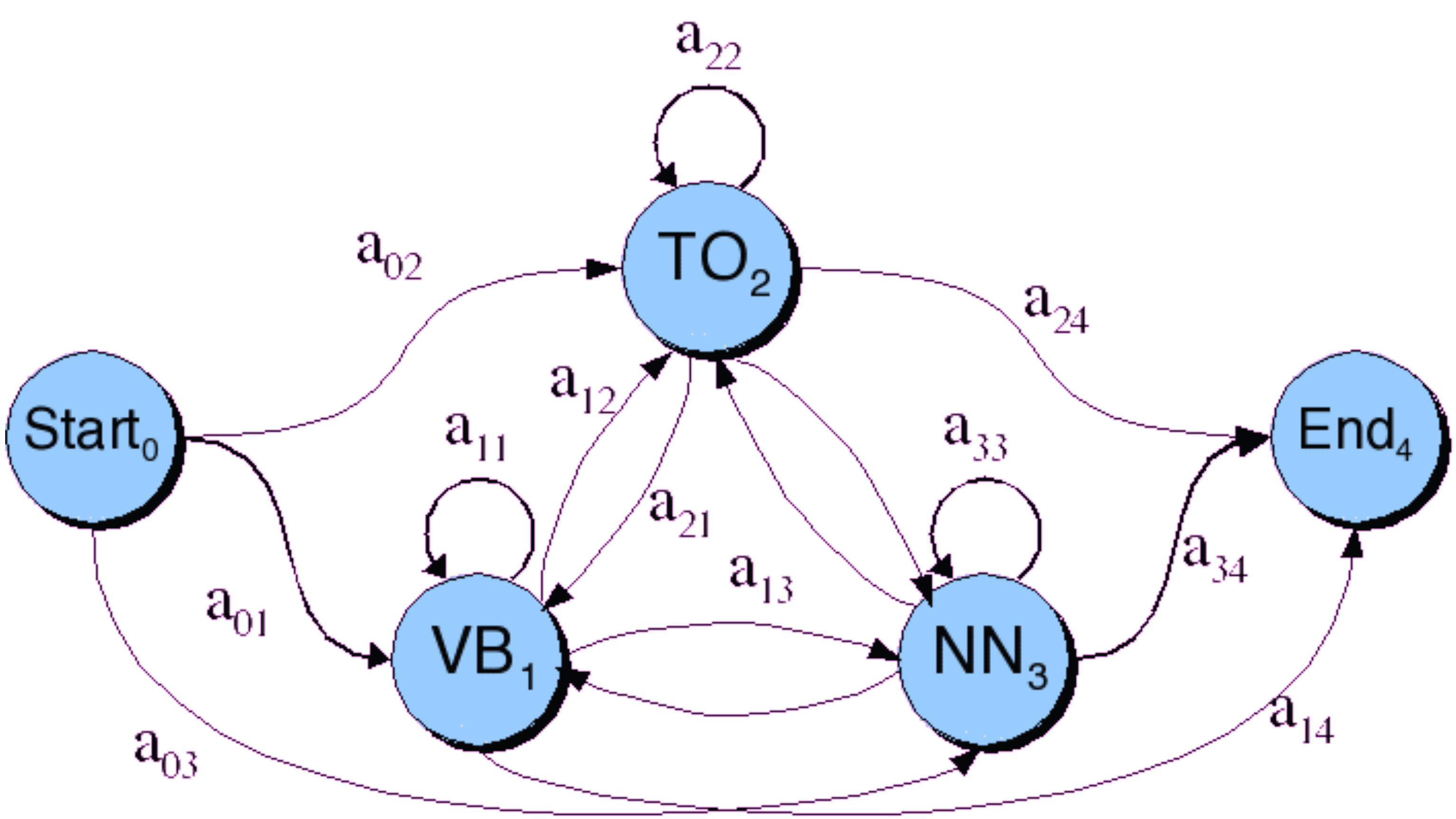


The Three Basic Problems for HMMs

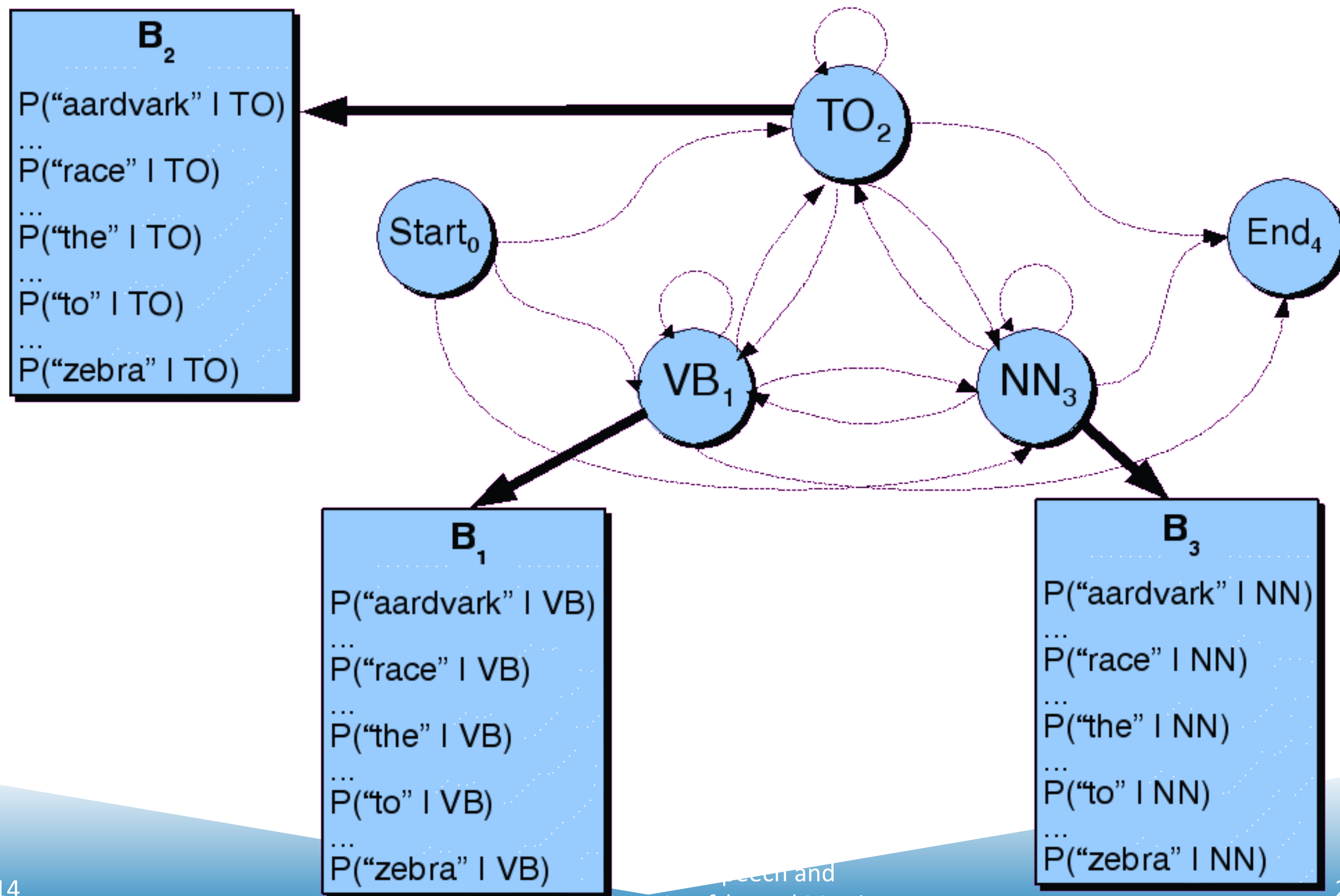
Jack Ferguson at IDA in the 1960s

- Problem 1 (**Evaluation**):
  - Given the observation sequence $O=(o_1o_2...o_T)$, and an HMM model $\Phi = (A,B)$, how do we efficiently compute $P(O|\Phi)$, the probability of the observation sequence, given the model
- Problem 2 (**Decoding**):
  - Given the observation sequence $O=(o_1o_2...o_T)$, and an HMM model $\Phi = (A,B)$, how do we choose a corresponding state sequence $Q=(q_1q_2...q_T)$ that is optimal in some sense (i.e., best explains the observations)
- Problem 3 (**Learning**):
  - How do we adjust the model parameters $\Phi = (A,B)$ to maximize $P(O|\Phi)$?

Transition Probabilities



Observation Likelihoods



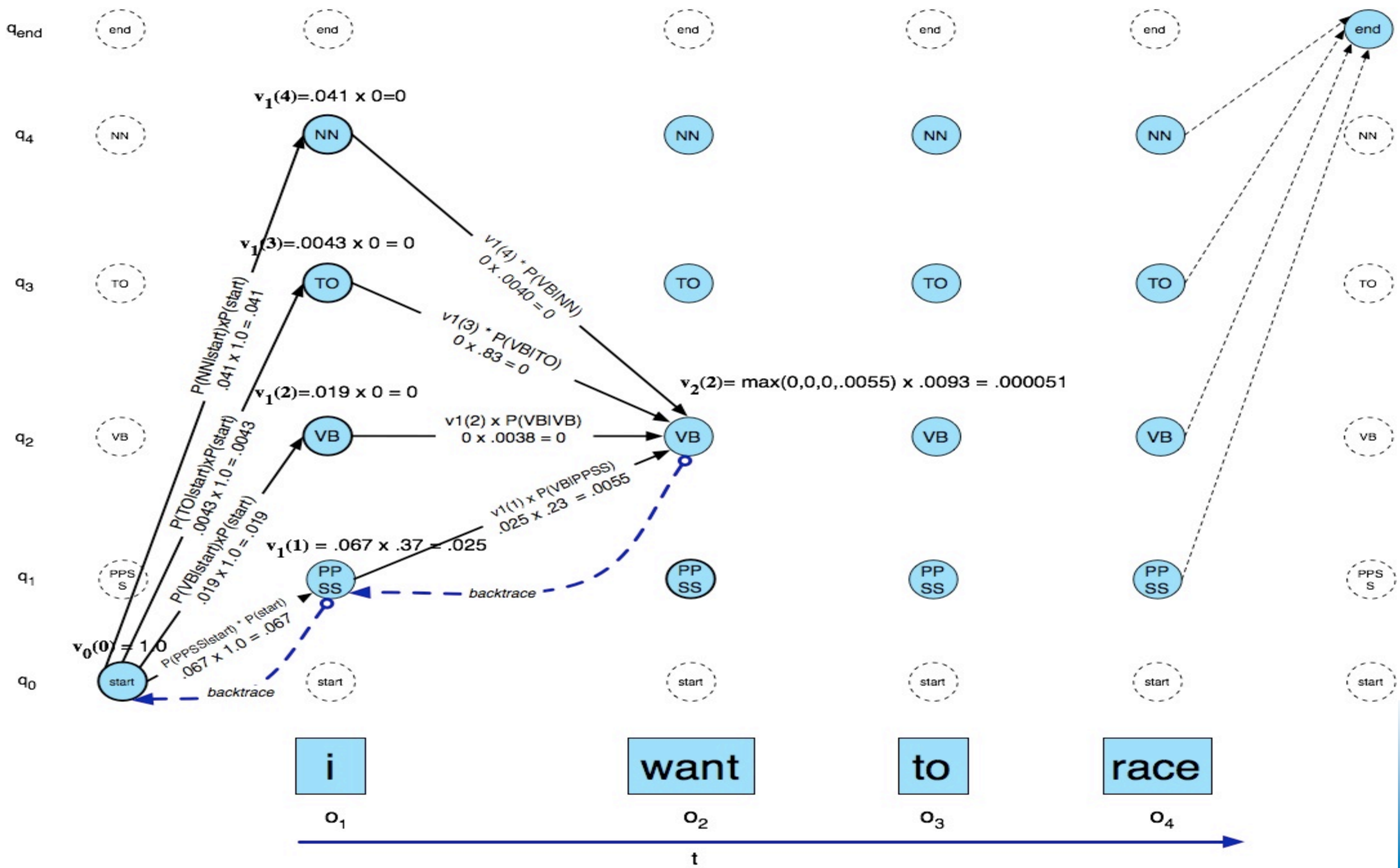
Decoding

- Ok, now we have a complete model that can give us what we need. Recall that we need to get

$$\hat{t}_1^n = \operatorname*{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

- We could just enumerate all paths given the input and use the model to assign probabilities to each.
  - Not a good idea.
  - Luckily dynamic programming (last seen in Ch. 3 with minimum edit distance) helps us here

Viterbi Example



Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states
- Sweep through the array in one pass filling the columns left to right using our transition probs and observations probs
- Dynamic programming key is that we need only store the MAX prob path to each cell, (not all paths).

Unknown Words:
Integrating features into the model

- Unknown words are a problem in open text
- Features of the word can help
  - Inflectional endings (e.g. –ing)
  - Derivational endings (e.g. –ly)
  - Hyphenation
  - Capitalization (+initial+capitalized,-initial+capitalized…)
- Instead of word emit probability use
  - $p^*(w_j \mid t_i = p(\text{unknown-word} \mid t_i) *$
    - » $p(\text{Capital-feature} \mid t_i)$
    - » $P(\text{endings/hypenations} \mid t_i)$

Results using features

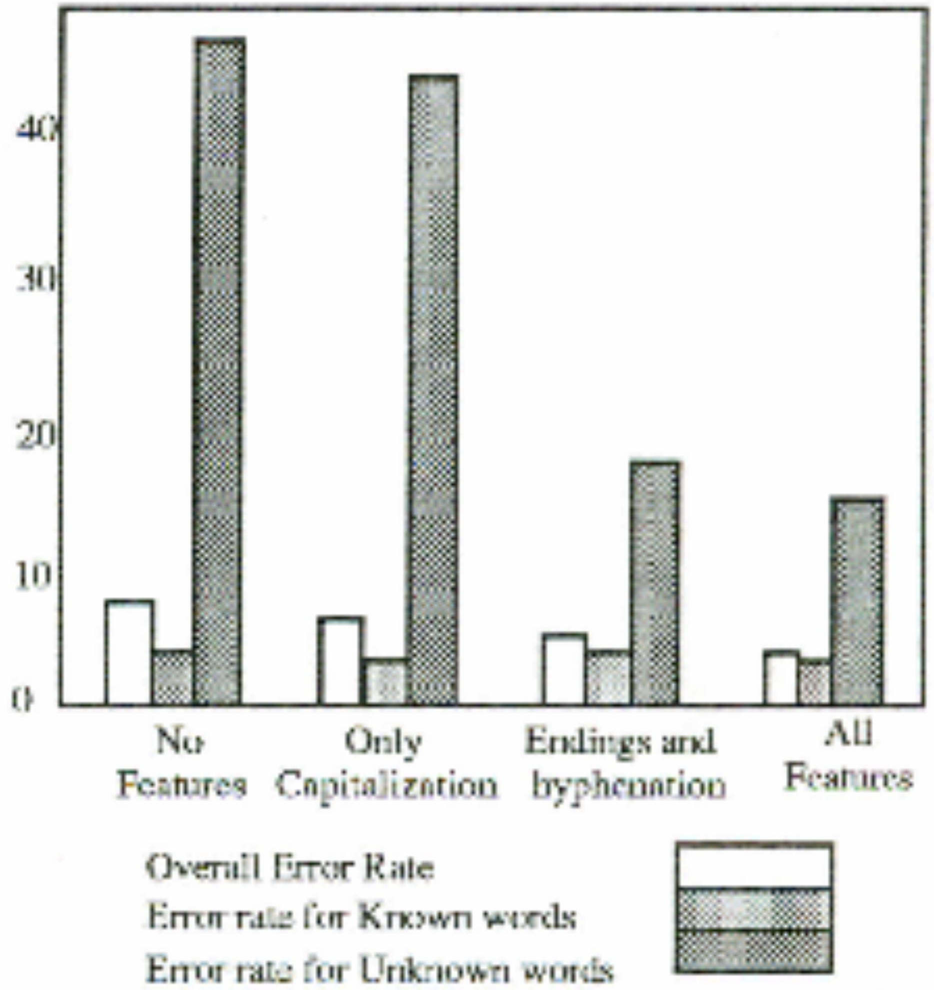


Figure 1: Decreasing error rate with use of word features

Evaluation

- So once you have you POS tagger running how do you evaluate it?
  - Overall error rate with respect to a gold-standard test set.
  - Error rates on particular tags
  - Error rates on particular words
  - Tag confusions...

Evaluation

- The result is compared with a manually coded "Gold Standard"
  – Typically accuracy reaches 96-97%
  – This may be compared with result for a baseline tagger (one that uses no context).
- Important: 100% is impossible even for human annotators.

Error Analysis

-

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	—	.2			.7		
JJ	.2	—	**3.3**	2.1	1.7	.2	**2.7**
NN		**8.7**	—				.2
NNP	.2	**3.3**	**4.1**	—	.2		
RB	**2.2**	2.0	.5		—		
VBD		.3	.5			—	**4.4**
VBN		**2.8**				**2.6**	—

- See what errors are causing problems
  - Noun (NN) vs ProperNoun (NNP) vs Adj (JJ)
  - Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)