Hidden Markov Models

Università di Pisa

IP notice: slides from Dan Jurafsky

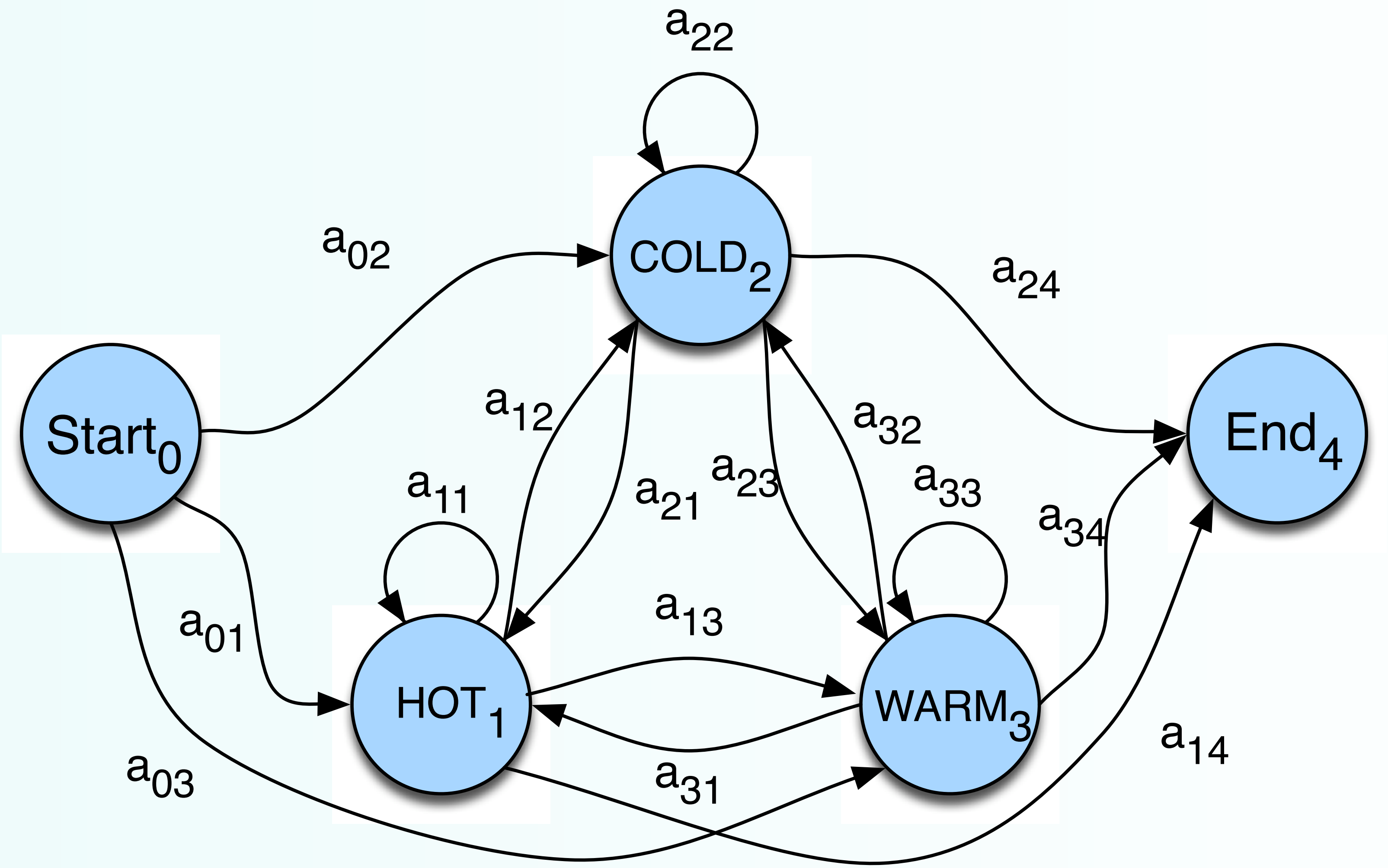
Outline

- Markov Chains
- Hidden Markov Models
- Three Algorithms for HMMs
  - The Forward Algorithm
  - The Viterbi Algorithm
  - The Baum-Welch (EM Algorithm)
- Applications:
  - The Ice Cream Task
  - Part of Speech Tagging

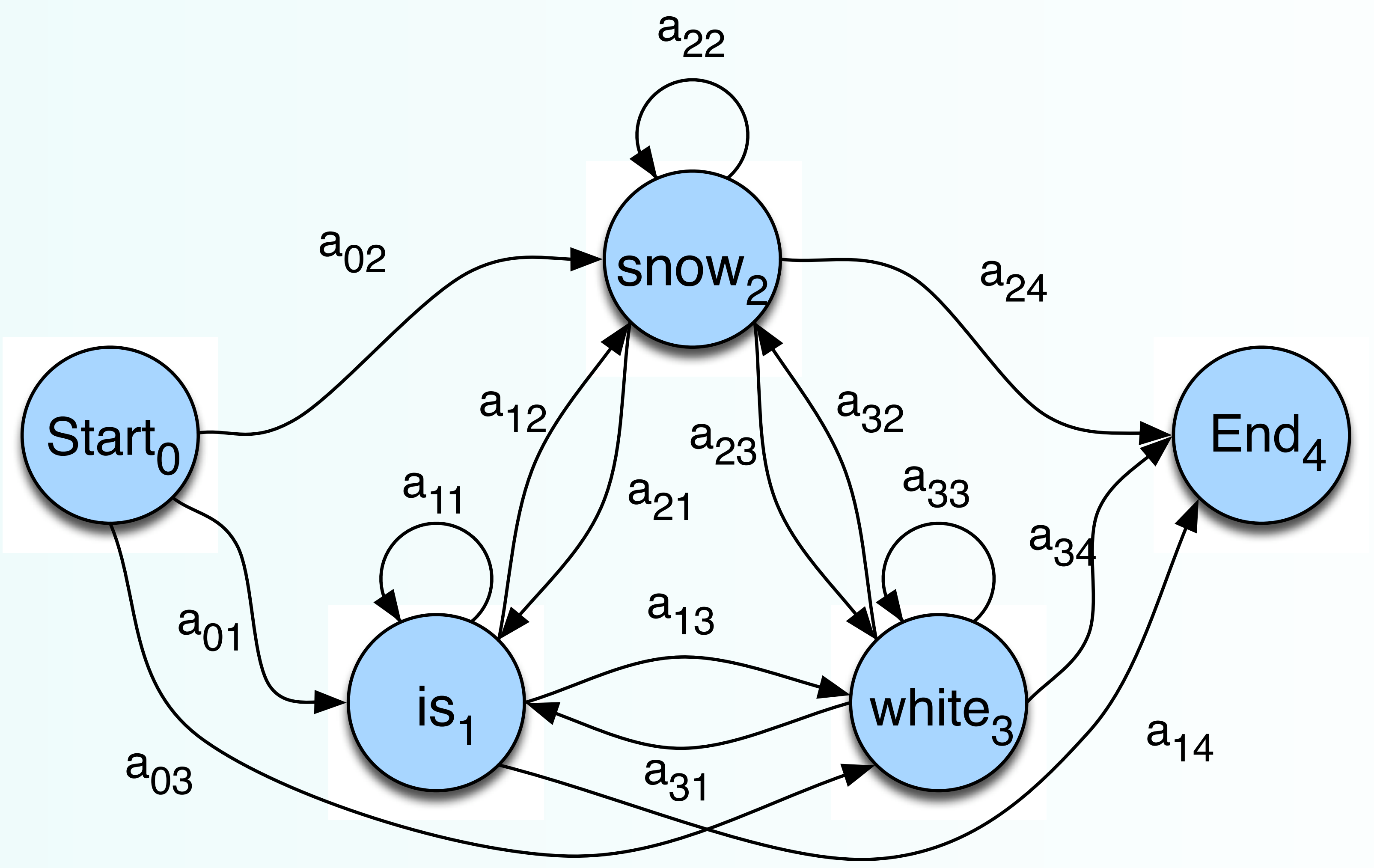
Definitions

- A Markov Chain (or Observable Markov Model)
  - is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event
- A Markov Chain can be represented by a transition diagram, where:
  - Each arc is labeled by a transition probability
  - The sum of the probabilities leaving any arc must sum to one

Markov Chain for weather



Markov Chain for words



Markov Chain: definition

A first-order observable Markov Model (aka Markov Chain) consists in:

- A set of states Q

  $q_1, q_2 \ldots q_N$ sequence of states: 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

$$a_{ij} = P(q_t = j | q_{t-1} = i) \ 1 \leq i, j \leq N$$

$$\sum_{j=1}^{N} a_{ij} = 1 \quad 1 \leq i \leq N$$

- Distinguished start and end states

Markov Chain

Markov Assumption:

- Current state only depends on **previous state**

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

Another representation for start state

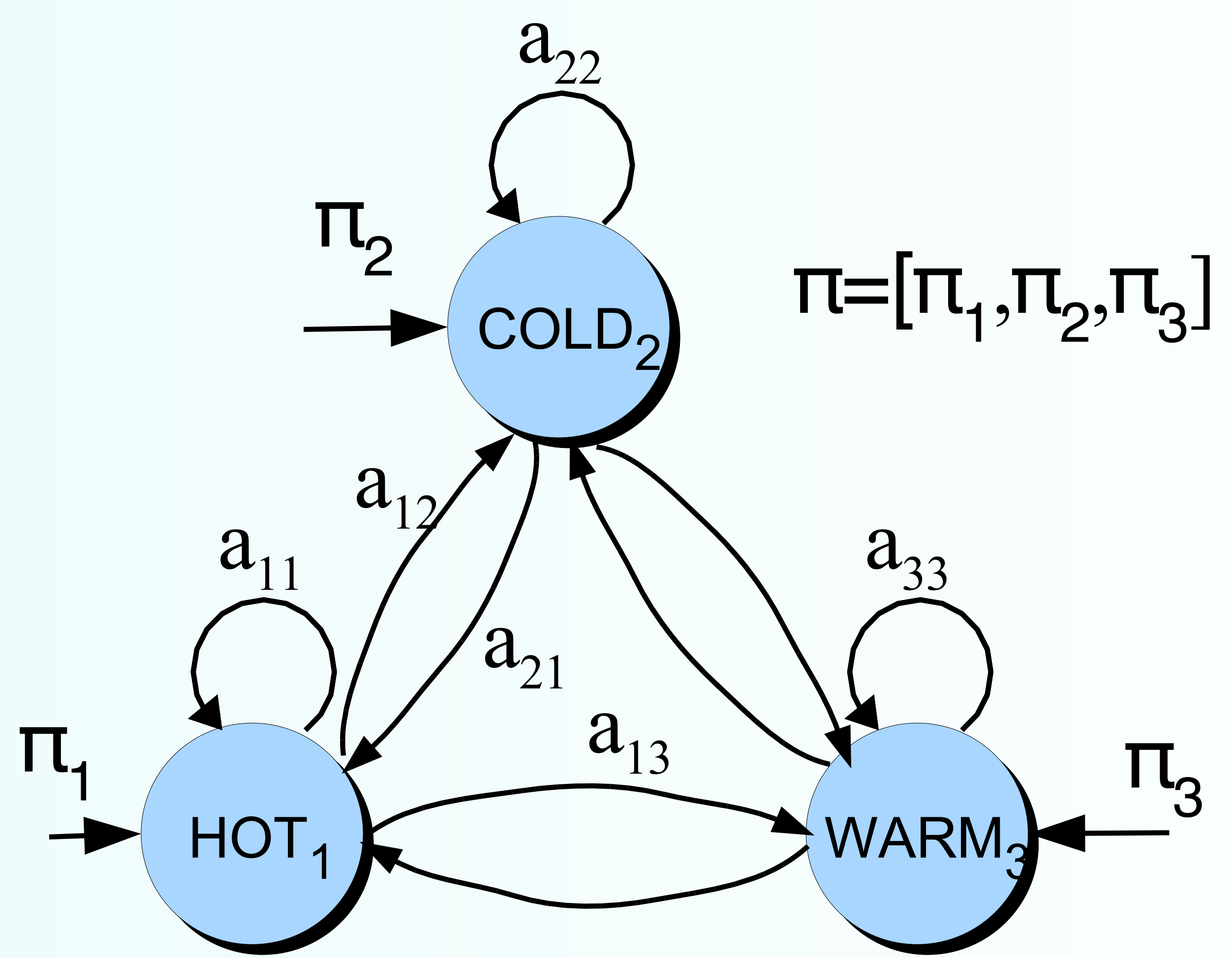
- Instead of start state
- Special initial probability vector π
  - An initial distribution over probability of start states

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

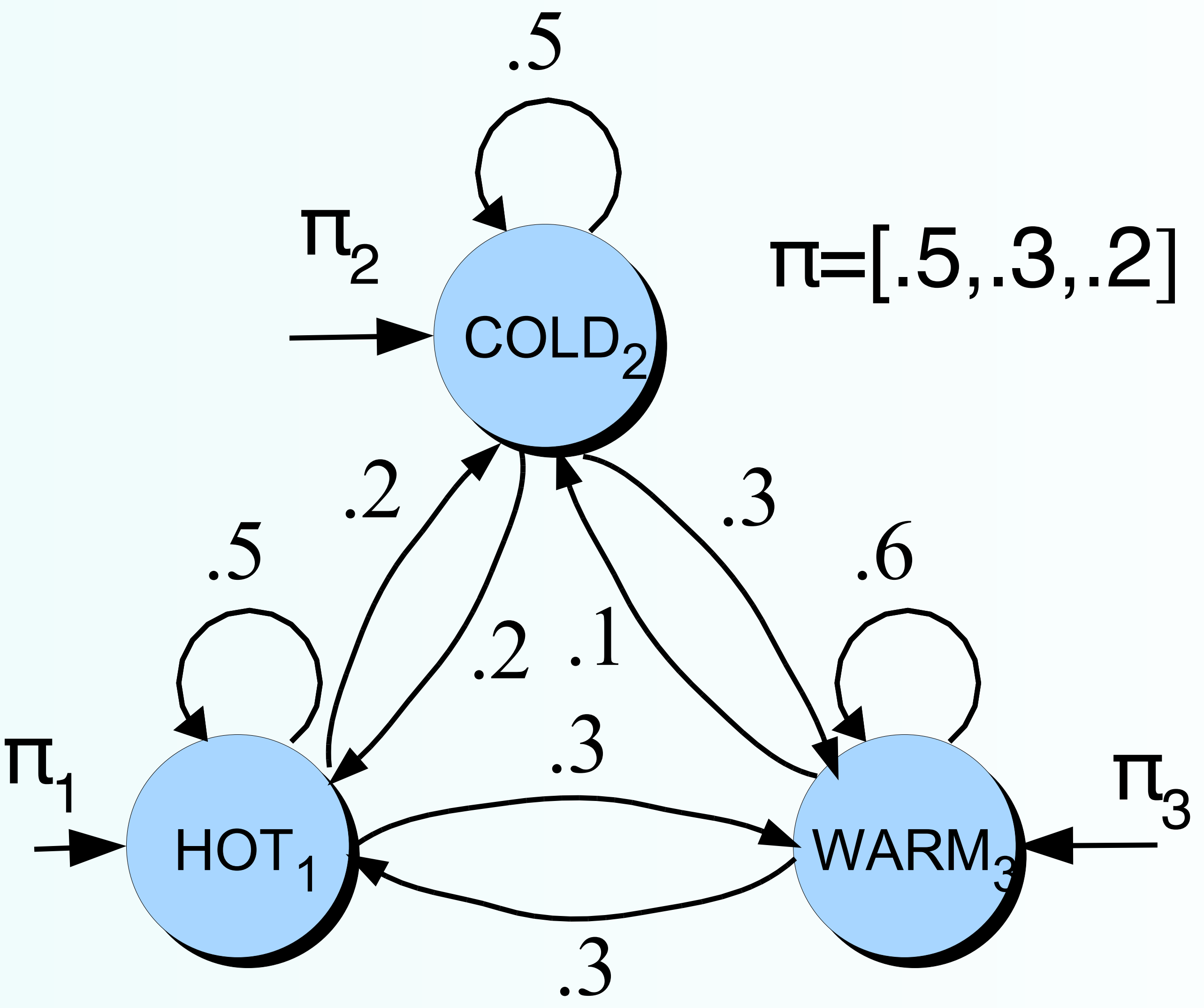
- Constraints:

$$\sum_{j=1}^{N} \pi_j = 1$$

The weather model using π



The weather model: specific example



Markov chain for weather

- What is the probability of 4 consecutive warm days?
- Sequence is warm-warm-warm-warm
- i.e., state sequence is 3-3-3-3

$$P(3, 3, 3, 3) =$$
$$\pi_3 a_{33} a_{33} a_{33} a_{33} = 0.2 \cdot (0.6)^3 = 0.0432$$

How about?

- Hot hot hot hot
- Cold hot cold hot

- What does the difference in these probabilities tell you about the real world weather info encoded in the figure?

Fun with Markov Chains

- Markov Chains "Explained Visually":
  http://setosa.io/ev/markov-chains
- Snakes and Ladders:
  http://datagenetics.com/blog/november12011/
- Candyland:
  http://www.datagenetics.com/blog/december12011/
- Yahtzee:
  http://www.datagenetics.com/blog/january42012/
- Chess pieces returning home and K-pop vs. ska:
  https://www.youtube.com/watch?v=63HHmjlh794

Hidden Markov Models

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states.
  - See **hot** weather: we are in state **hot**
- But in named-entity or part-of-speech tagging (and speech recognition)
  - The output symbols are **words**
  - But the hidden states are something else
    - **Part-of-speech tags**
    - **Named entity 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.

Hidden Markov Model: Definition

$Q = q_1 q_2 \ldots q_N$	a set of N **hidden states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving fronm state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} \forall i$
$O = o_1 o_2 \ldots o_T$	a sequenxe of T **observations**, each one drawn from a vocavulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probabilitu of an observation o_t being generated from a state i
$q_{0,}, q_F$	a special **start state** and an **end state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{1F} \ldots a_{n\mathrm{F}}$ into the end state.

Assumptions

Markov assumption:

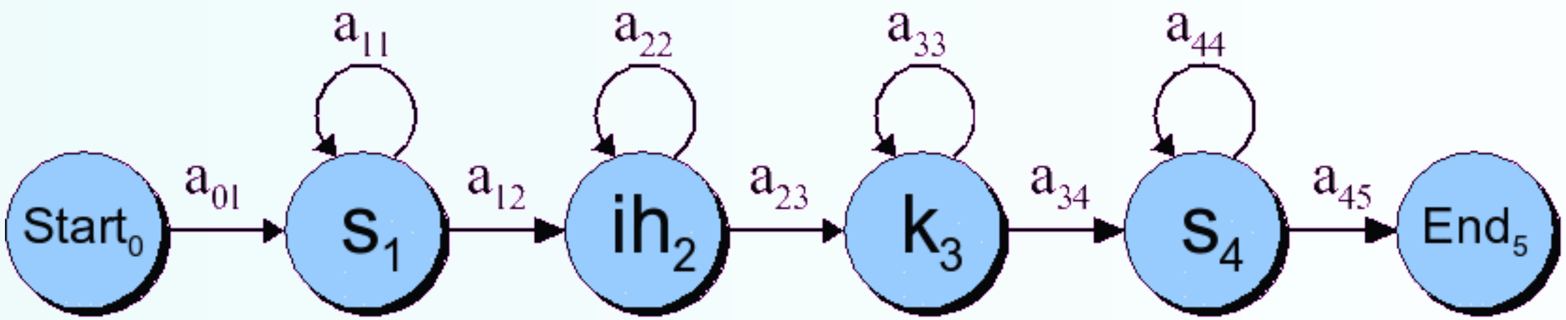
$$P(q_i \mid q_1 \ldots q_{i\text{-}1}) = P(q_i \mid q_{i\text{-}1})$$

Output-independence assumption

$$P(o_t \mid O_1^{t-1}, q_1^t) = P(o_t \mid q_t)$$

Example: HMM for speech

- Observed outputs are **phones** (speech sound)
- Hidden states are **phonemes** (unit of sound)
- HMM for the word "six":



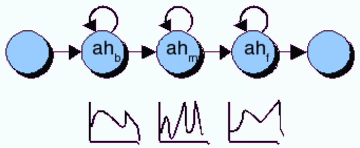
- Loopbacks present because
  - a phone is ~100 milliseconds long
  - An observation of speech every 10 ms
  - So each phone repeats ~10 times (simplifying greatly)

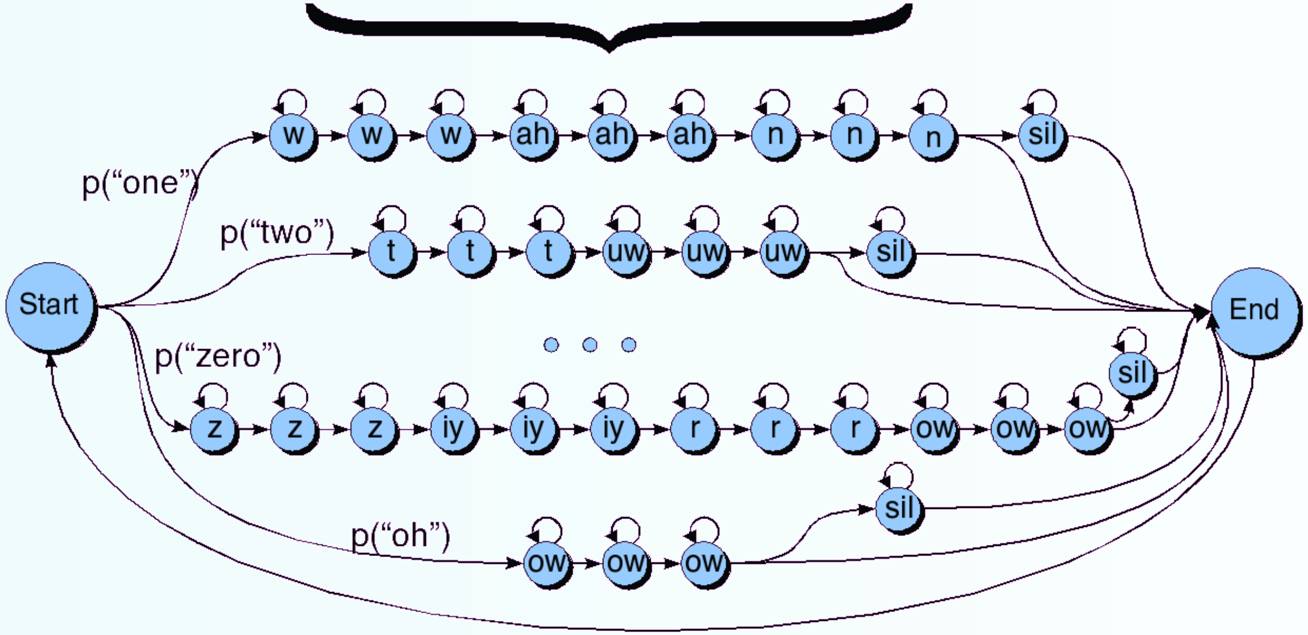
HMM for Speech: Recognizing Digits

Lexicon

one	w ah n
two	t uw
three	th r iy
four	f ao r
five	f ay v
six	s ih k s
seven	s eh v ax n
eight	ey t
nine	n ay n
zero	z iy r ow
oh	ow

Phone HMM





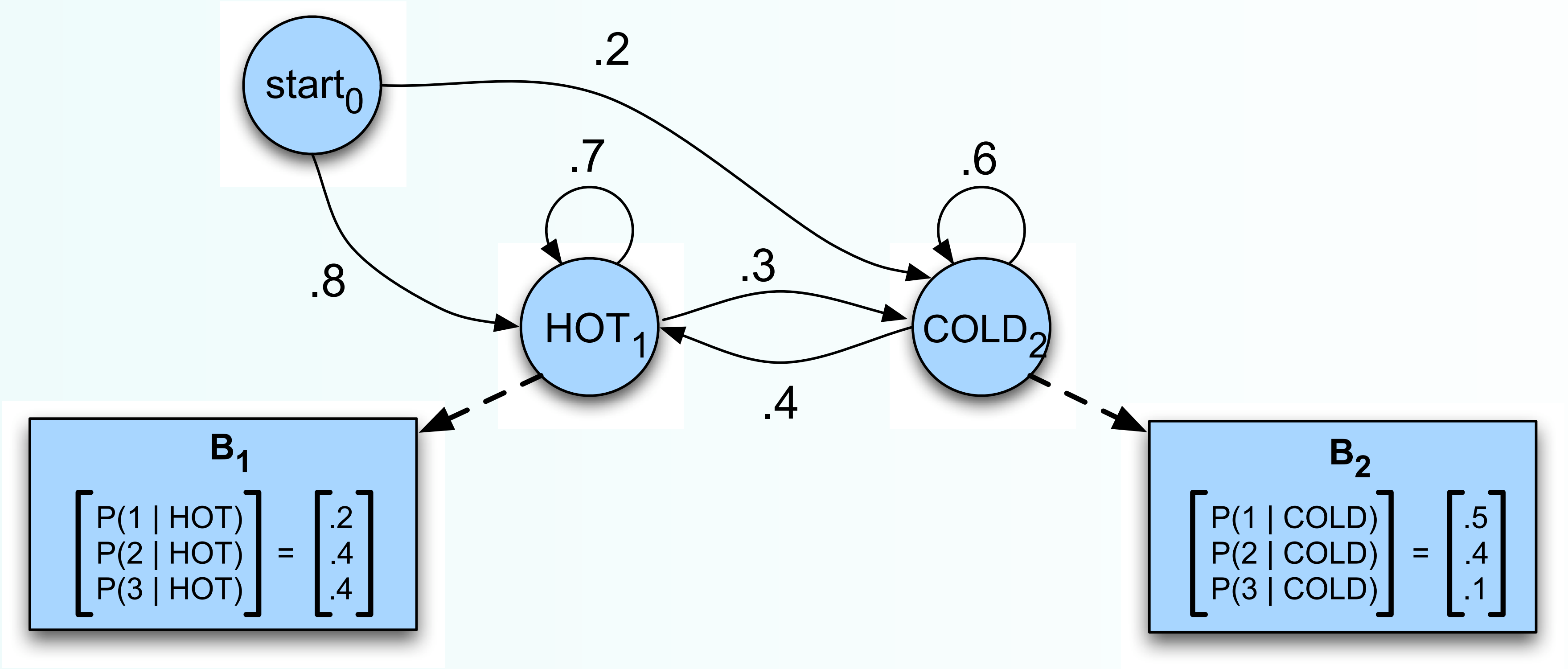
HMM for Ice Cream

- You are a climatologist in the year 2799
- Studying global warming
- You can’t find any records of the weather in Baltimore, MD for summer of 2008
- 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

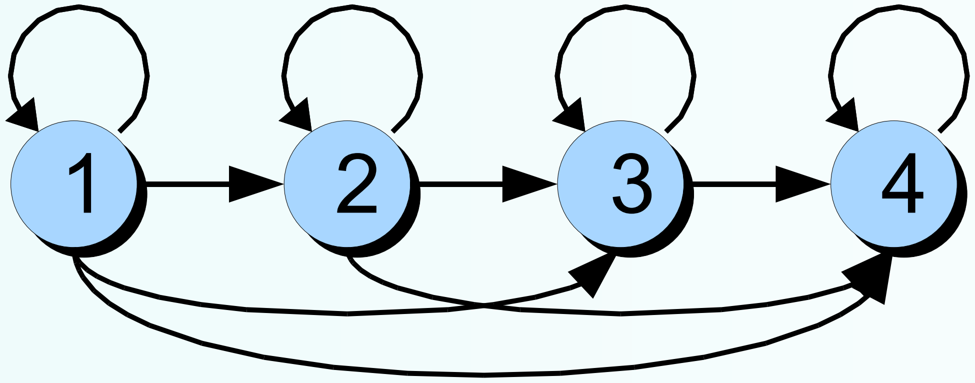
Eisner task

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

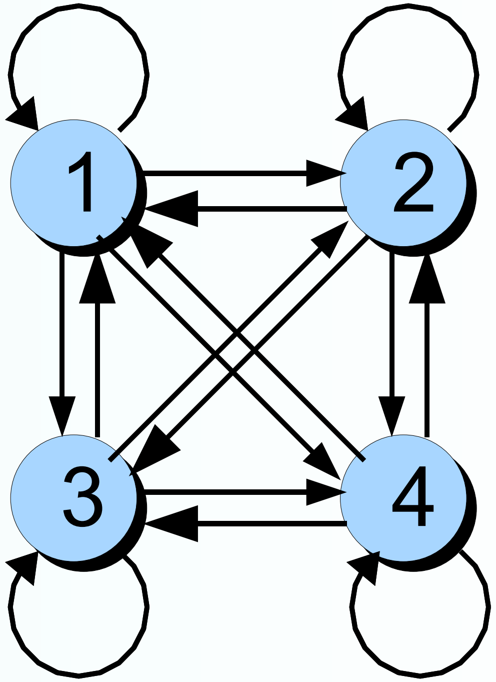
HMM for ice cream



Different types of HMM structure



Bakis = left-to-right



Ergodic = fully-connected

The Three Basic Problems for HMMs

Problem 1 (**Evaluation**): Given the observation sequence $O = (o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A, B)$, how do we efficiently compute $P(O|\ \Phi)$, the probability of the observation sequence, given the model

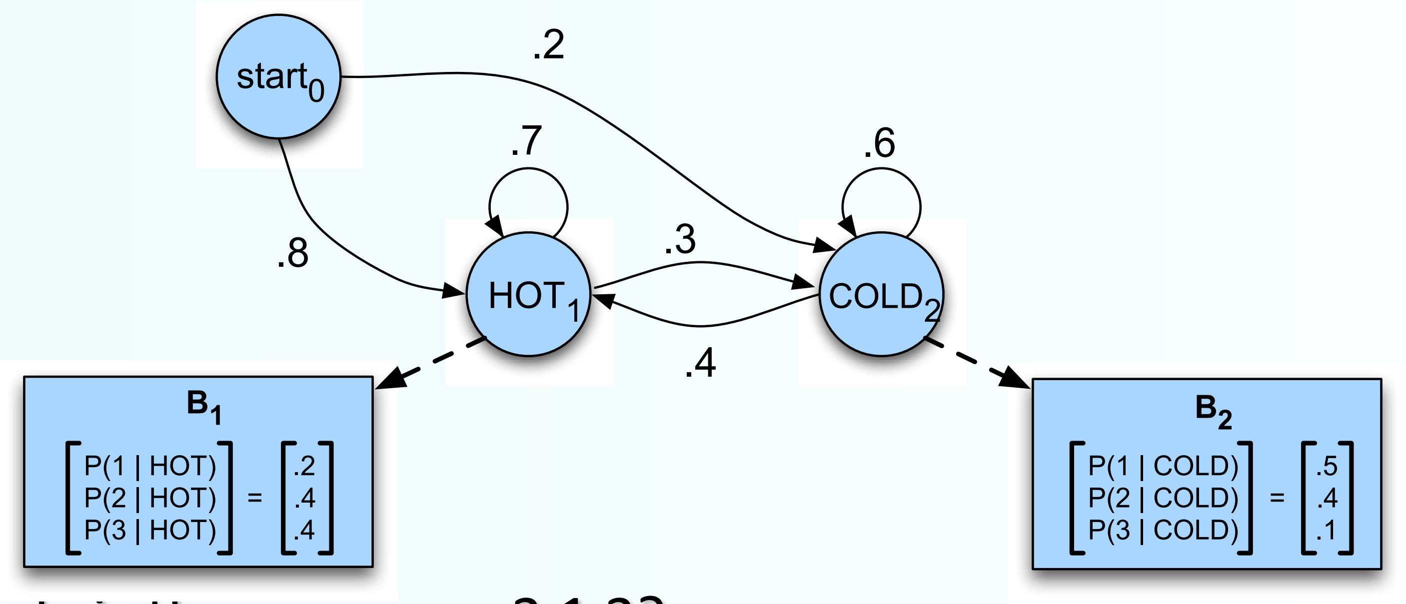
Problem 2 (**Decoding**): Given the observation sequence $O = (o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A, B)$, how do we choose a corresponding state sequence $Q = (q_1 q_2 \ldots q_T)$ that is optimal in some sense (i.e., best explains the observations)

Problem 3 (**Learning**): How do we adjust the model parameters $\Phi = (A,B)$ to maximize $P(O|\ \Phi)$?

Problem 1: computing the observation likelihood

Computing Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O, \lambda)$.

Given the following HMM:



How likely is the sequence 3 1 3?

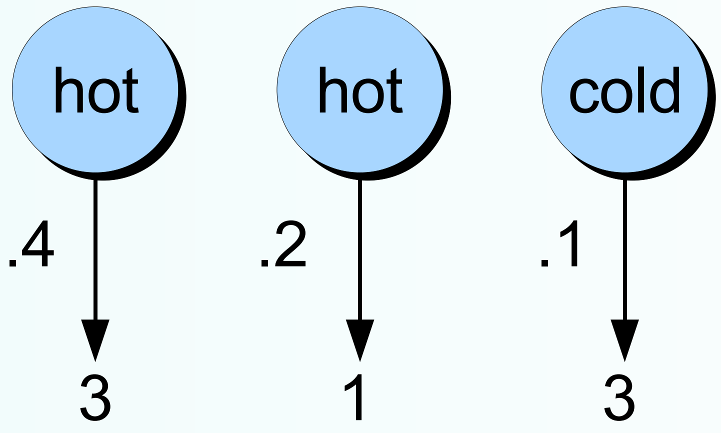
How to compute likelihood

- For a Markov chain, we just follow the states 3 1 3 and multiply the probabilities
- But for an HMM, we don't know what the states are!
- So let's start with a simpler situation.
- Computing the observation likelihood for a **given** hidden state sequence
  - Suppose we knew the weather and wanted to predict how much ice cream Jason would eat.
  - i.e. $P(3\ 1\ 3 \mid \mathrm{H\ H\ C})$

Computing likelihood of 3 1 3 given hidden state sequence

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$$

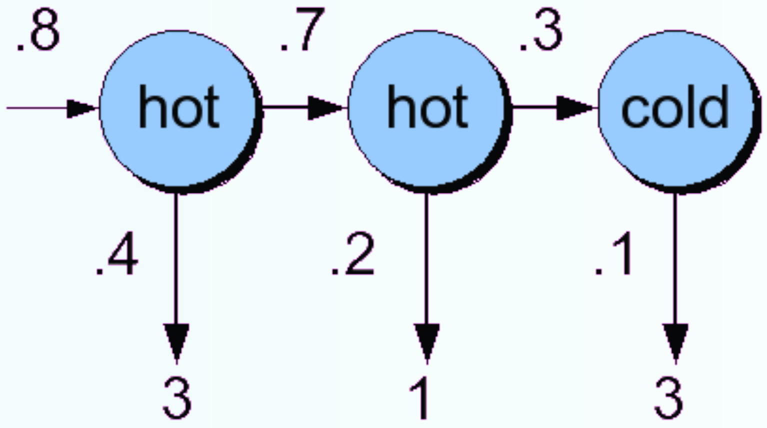
$$P(3\ 1\ 3|\text{hot hot cold}) \;=\; P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})$$



Computing joint probability of observation and state sequence

$$P(O,Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{n} P(o_i|q_i) \times \prod_{i=1}^{n} P(q_i|q_{i-1})$$

$$\begin{aligned} P(3\ 1\ 3, \text{hot hot cold}) &= P(\text{hot}|\text{start}) \times P(\text{hot}|\text{hot}) \times P(\text{cold}|\text{hot}) \\ &\times P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold}) \end{aligned}$$



Computing total likelihood of 3 1 3

- We would need to sum over
  - Hot hot cold
  - Hot hot hot
  - Hot cold hot
  - ….

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

- How many possible hidden state sequences are there for this sequence?

$$P(3\ 1\ 3) = P(3\ 1\ 3, \text{cold cold cold}) + P(3\ 1\ 3, \text{cold cold hot}) + P(3\ 1\ 3, \text{hot hot cold}) + \ldots$$

- How about in general for an HMM with N hidden states and a sequence of T observations?

  N^T

- So we can't just do separate computation for each hidden state sequence.

Instead: the Forward algorithm

- A kind of **dynamic programming** algorithm
  - Just like Minimum Edit Distance
  - Uses a table to store intermediate values
- Idea:
  - Compute the likelihood of the observation sequence
  - By summing over all possible hidden state sequences
  - But doing this efficiently
    - By folding all the sequences into a single **trellis**

The forward algorithm

- The goal of the forward algorithm is to compute

$$P(o_1, o_2 \ldots o_T, q_T = q_F \mid \lambda)$$

- We'll do this by recursion

The forward algorithm

- Each cell of the forward algorithm trellis $\alpha_t(j)$
  - Represents the probability of being in state j
  - After seeing the first t observations
  - Given the automaton
- Each cell thus expresses the following probability

$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j \mid \lambda)$$

The Forward Recursion

1. Initialization:

$$\alpha_1(j) \;=\; a_{0j} b_j(o_1) \;\; 1 \leq j \leq N$$

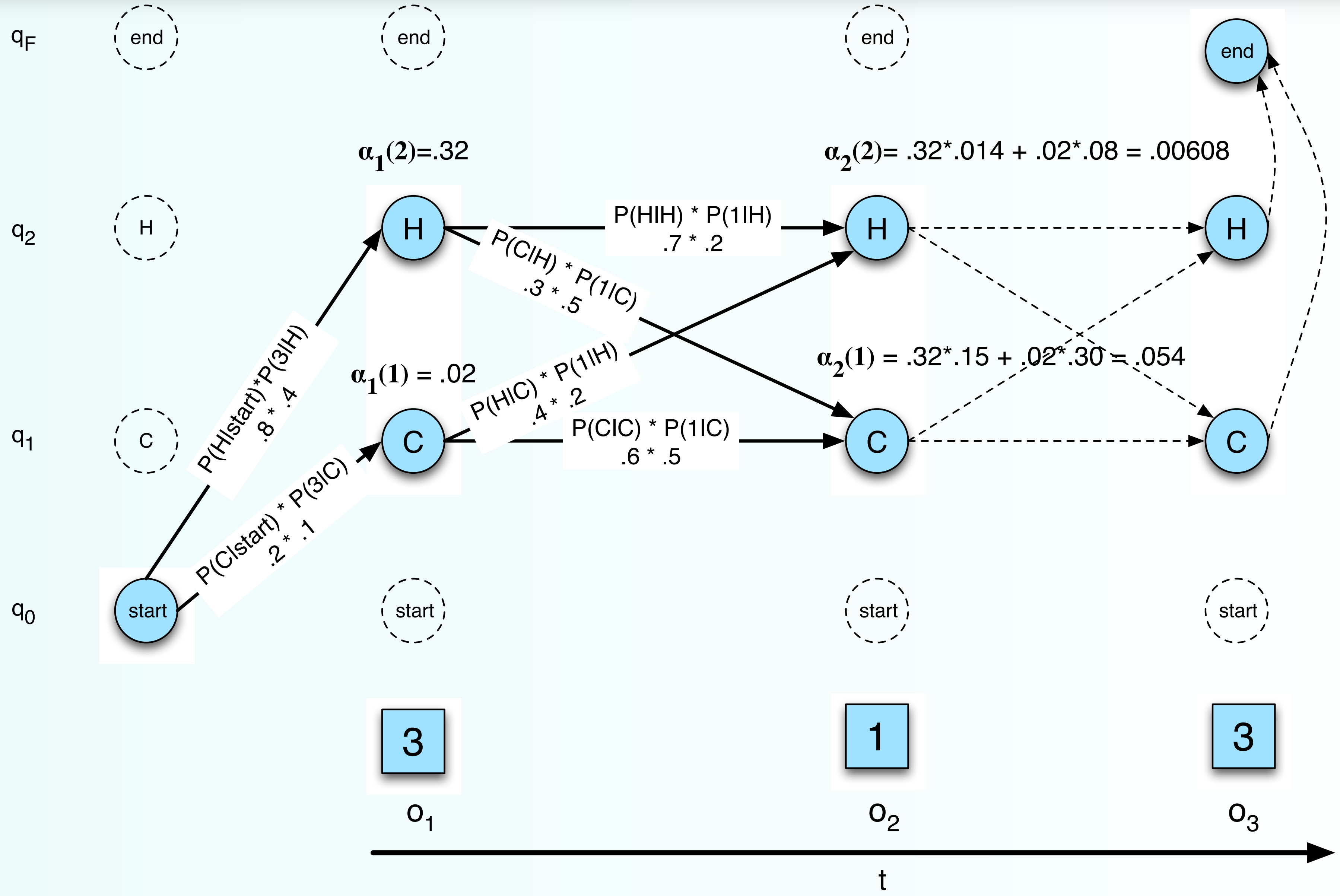
2. Recursion (since states 0 and F are non-emitting):

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

3. Termination:

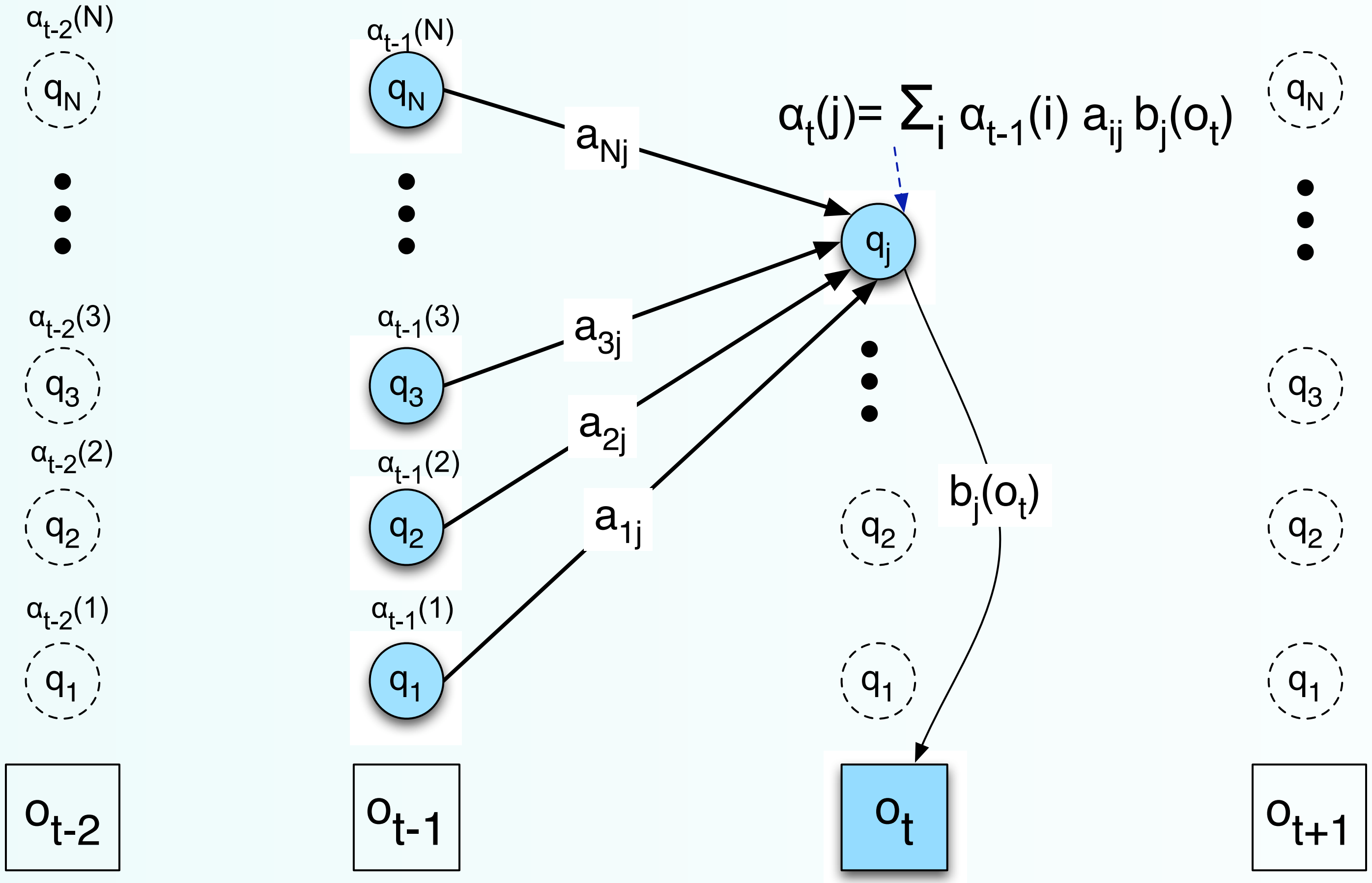
$$P(O|\lambda) = \alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$$

The Forward Trellis



We update each cell

$\alpha_{t-1}(i)$	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



The 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**

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

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

return $forward[q_F,T]$

Decoding

- Given an observation sequence
  - 3 1 3
- and an HMM
- The task of the **decoder**
  - To find the best **hidden** state sequence
- Given the observation sequence $O = (o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A,B)$, how do we choose a corresponding state sequence $Q=(q_1 q_2 \ldots q_T)$ that is optimal in some sense (i.e., best explains the observations)

Decoding

- One possibility:
  - For each hidden state sequence Q
    - HHH, HHC, HCH,
  - Compute $P(O|Q)$
  - Pick the highest one
- Why not?

  N^T
- Instead:
  - The Viterbi algorithm
  - Is again a **dynamic programming** algorithm
  - Uses a similar trellis to the Forward algorithm

Viterbi intuition

- We want to compute the joint probability of the observation sequence together with the best state sequence

$$\max_{q_0,q_1,\ldots,q_T} P(q_0,q_1,\ldots,q_T,o_1,o_2,\ldots,o_T,q_T=q_F \mid \lambda)$$

$$v_t(j) = \max_{q_0,q_1,\ldots,q_{t-1}} P(q_0,q_1\ldots q_{t-1},o_1,o_2\ldots o_t,q_t=j|\lambda)$$

$$v_t(j) \;=\; \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

Viterbi Recursion

1. **Initialization:**

$$v_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$$
$$bt_1(j) = 0$$

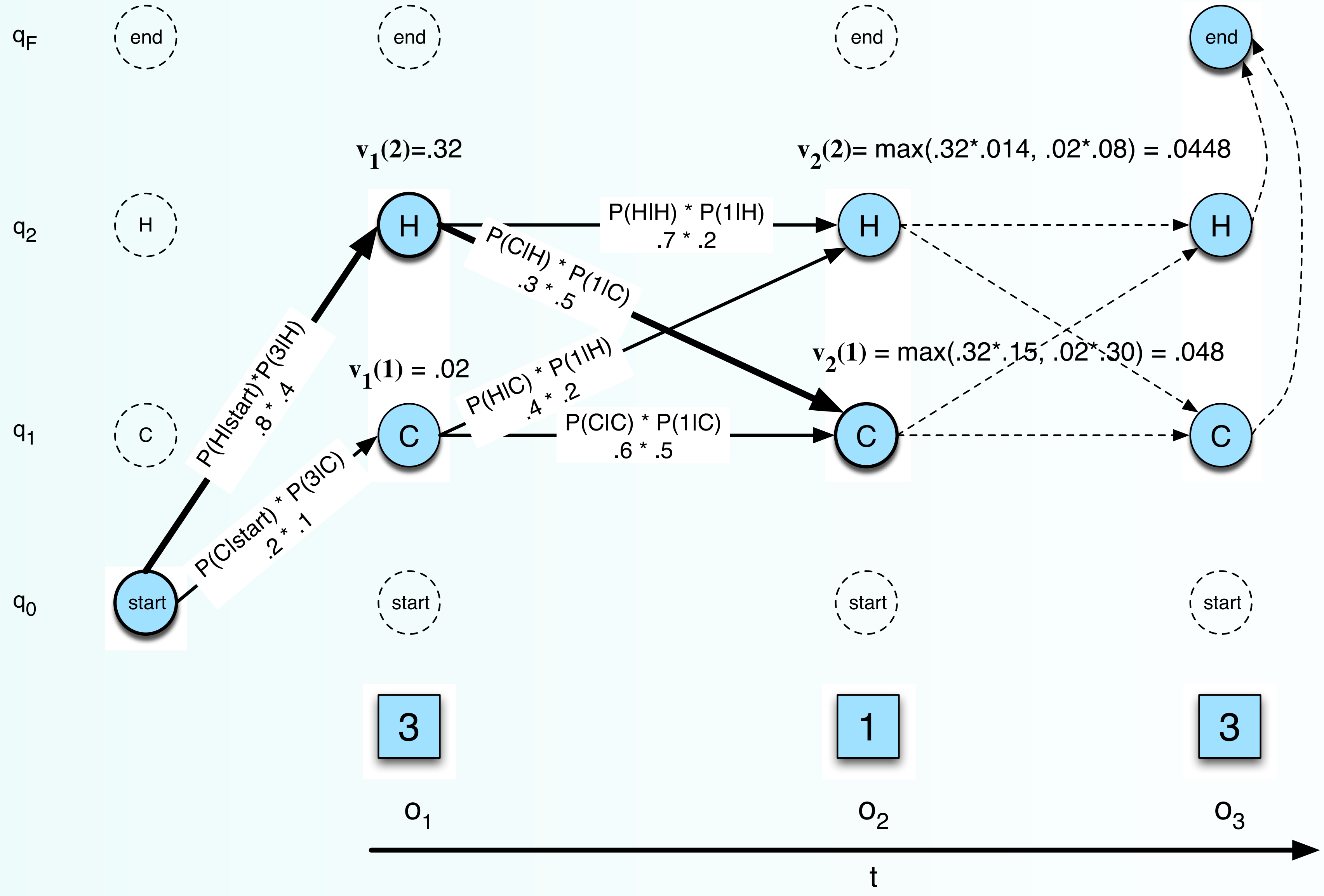
2. **Recursion** (recall that states 0 and q_F are non-emitting):

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$
$$bt_t(j) = \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

3. **Termination:**

$$\text{The best score:} \quad P* = v_t(q_F) = \max_{i=1}^{N} v_T(i) * a_{i,F}$$
$$\text{The start of backtrace:} \quad q_T* = bt_T(q_F) = \operatorname*{argmax}_{i=1}^{N} v_T(i) * a_{i,F}$$

The Viterbi trellis



Viterbi intuition

- Process observation sequence left to right
- Filling out the trellis
- Each cell:

$$v_t(j) = \max_{q_0, q_1, \ldots, q_{t-1}} P(q_0, q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t, q_t = j | \lambda)$$

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

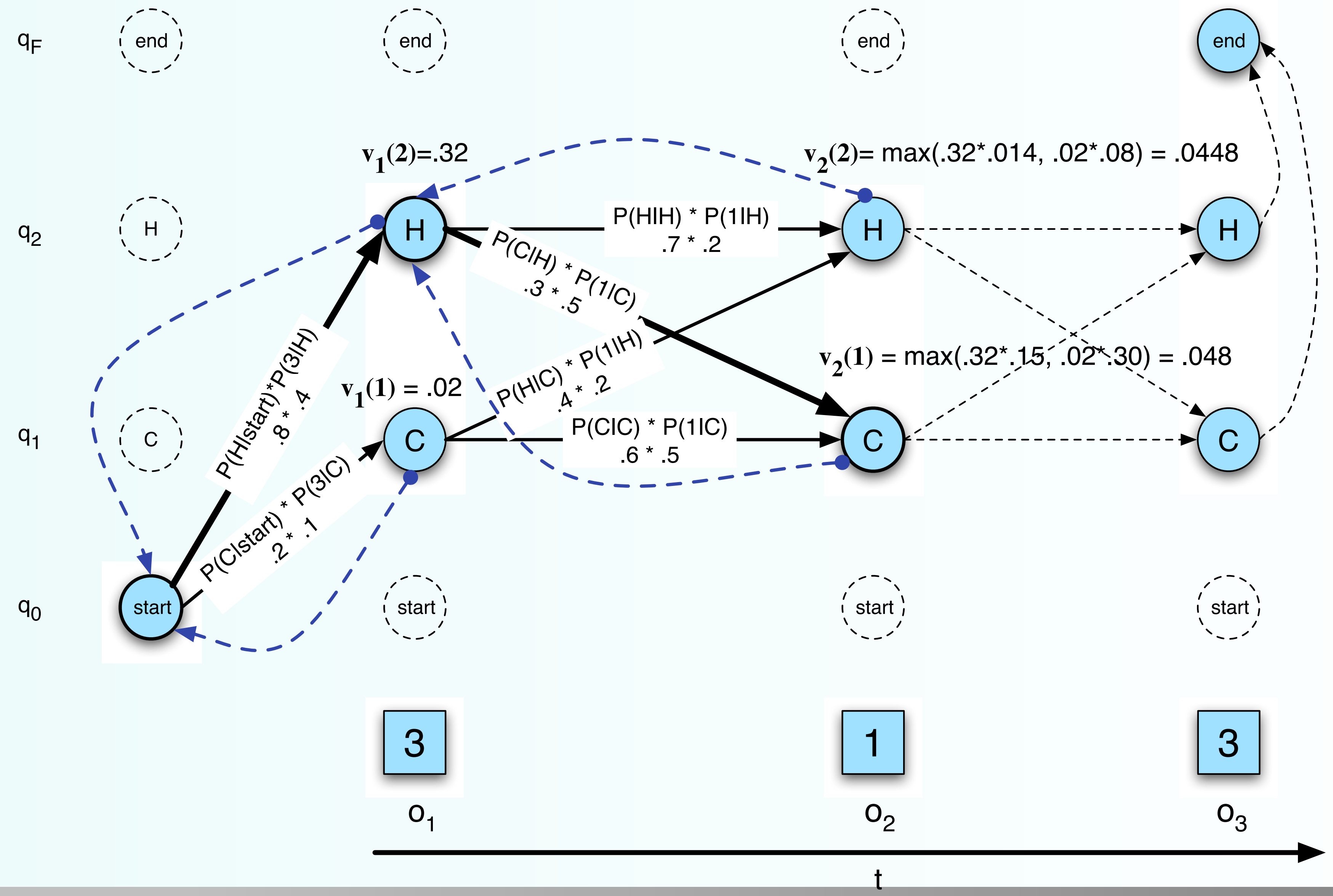
$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

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 backtrace



Training a HMM

- Forward-backward or Baum-Welch algorithm (Expectation Maximization)
- Backward probability (prob. of observations from t+1 to T)

$\beta t(i) = P(o_{t+1}, o_{t+2}\ldots o_T \mid q_t = i, \lambda)$

$\beta_T(i) = a_{i,F} \quad 1 \leq i \leq N$

$$\beta_t(i) = \sum_{i=1}^{N} a_{ij}\beta_j(o_{t+1})\,\beta_{t+1}(j), \qquad 1 \leq i \leq N, 1 \leq t \leq T$$

$$P(O|\lambda) = \alpha_T(q_F) = \beta_1(0) = \sum_{j=1}^{N} a_{0j} b_j(o_1)\beta_1$$

Baum-Welch Algorithm

function FORWARD-BACKWARD(*observations* of len *T*, *output vocabulary V*, *hidden state set Q*) **returns** *HMM=(A,B)*

initialize *A* and *B*

iterate until convergence

E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} \ \forall t, j \qquad \xi_t(i,j) = \frac{\alpha_t(j) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(N)} \ \forall t, i, j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i,j)} \qquad \hat{b}_j(v_k) = \frac{\sum_{t=1\ s.t.\ o_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return *A*, *B*

HMM for Part of Speech Tagging

Part of speech tagging

- 8 (ish) traditional English parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc.
  - This idea has been around for over 2000 years (Dionysius Thrax of Alexandria, c. 100 B.C.)
  - Called: parts-of-speech, lexical category, word classes, morphological classes, lexical tags, POS
  - We'll use POS most frequently
  - Assuming that you know what these are

POS examples

N	noun	*chair, bandwidth, pacing*
V	verb	*study, debate, munch*
ADJ	adj	*purple, tall, ridiculous*
ADV	adverb	*unfortunately, slowly,*
P	preposition	*of, by, to*
PRO	pronoun	*I, me, mine*
DET	determiner	*the, a, that, those*

POS Tagging example

Word	Tag
the	**DET**
koala	**NOUN**
put	**VERB**
the	**DET**
keys	**NOUN**
on	**PREP**
the	**DET**
table	**NOUN**

POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = ADJ
  - On my *back* = NOUN
  - Win the voters *back* = ADV
  - Promised to *back* the bill = VERB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

These examples from Dekang Lin

POS tagging as a Sequence Classification task

- We are given a sentence (an "observation" or "sequence of observations")

  Secretariat is expected to race tomorrow

  She promised to back the bill
- What is the best sequence of tags which corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words $w_1 \ldots w_n$.

Problem Formulation

- We want, out of all sequences of n tags $t_1 \ldots t_n$ the single tag sequence such that $P(t_1 \ldots t_n | w_1 \ldots w_n)$ is highest.

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

- Hat ^ means "our estimate of the best one"
- $\operatorname{argmax}_x f(x)$ means "the x such that $f(x)$ is maximized"
- How to make it operational? How to compute this value?
- Intuition of Bayesian classification:

  Use Bayes rule to transform into a set of other probabilities that are easier to compute

Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

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

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

Likelihood and prior

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n | t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^{n} P(w_i | t_i)$$

Naïve Bayes assumption

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1})$$

Markov assumption

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n | t_1^n) P(t_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1})$$

Two kinds of probabilities (1)

- Tag transition probabilities $P(t_i|t_{i\text{-}1})$
  - Determiners likely to precede adjectives and nouns

    That/DET flight/NOUN

    The/DET yellow/JADJ hat/NOUN

    So we expect P(NOUN|DET) and P(ADJ|DET) to be high

    But P(DET|ADJ) to be low
  - Compute P(NOUN|DET) by counting in a labeled corpus:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NOUN|DET) = \frac{C(DET, NOUN)}{C(DET)} = \frac{56{,}509}{116{,}454} = 0.49$$

Two kinds of probabilities (2)

- Word likelihood probabilities $P(w_i|t_i)$
  - VERB likely to be "is"
  - Compute $P(\text{is}|\text{VERB})$ by counting in a labeled corpus:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VERB) = \frac{C(VERB, is)}{C(VERB)} = \frac{10{,}073}{21.627} = 0.47$$

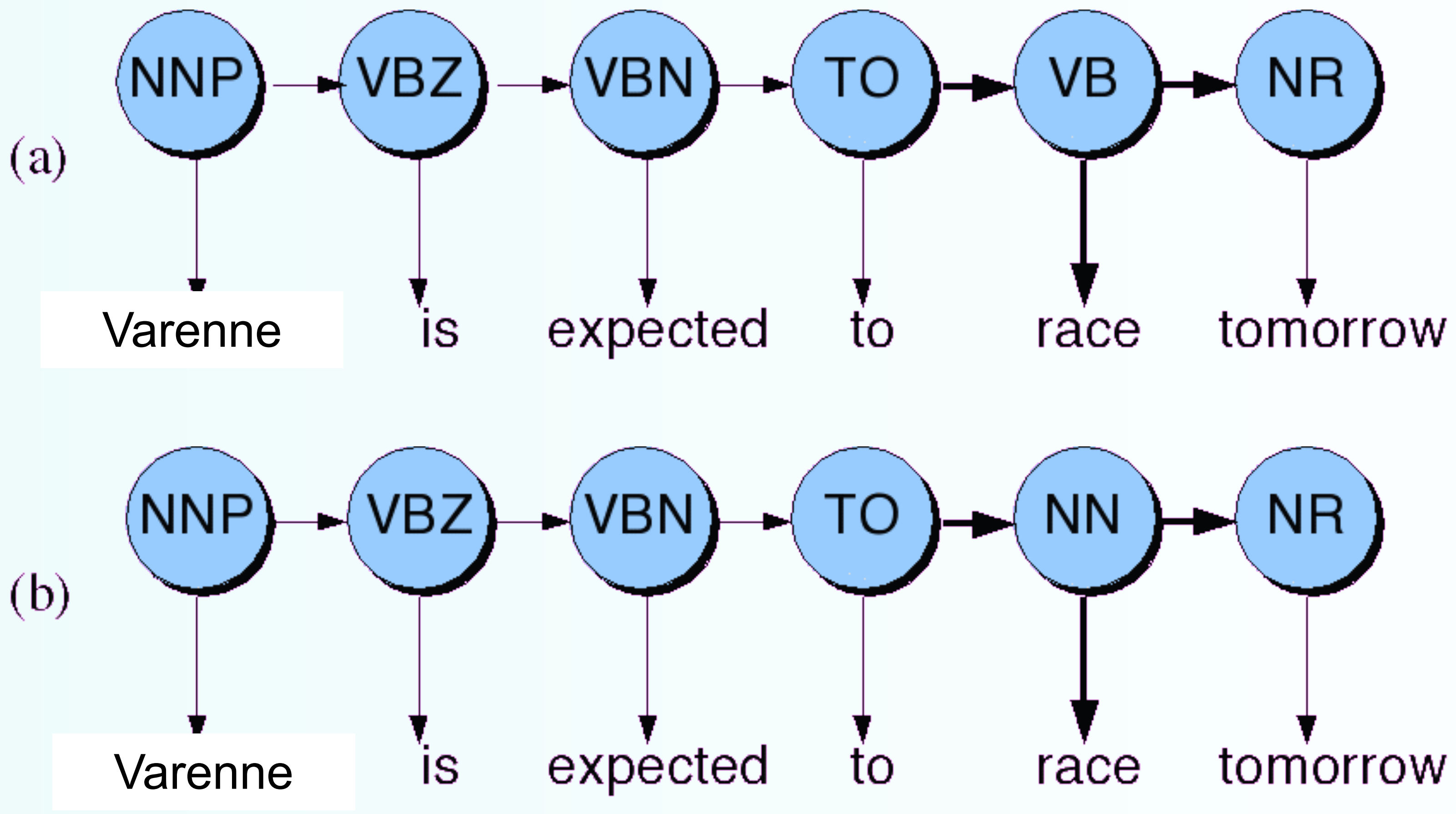
An Example: the word "race"

Varenne/NNP is/VBZ expected/VBN to/TO **race**/VB tomorrow/NR

People/NNS continue/VB to/TO inquire/VB the/DT reason/NN for/IN the/DT **race**/NN for/IN outer/JJ space/NN

- How do we pick the right tag?

Disambiguating "race"



ML Estimation

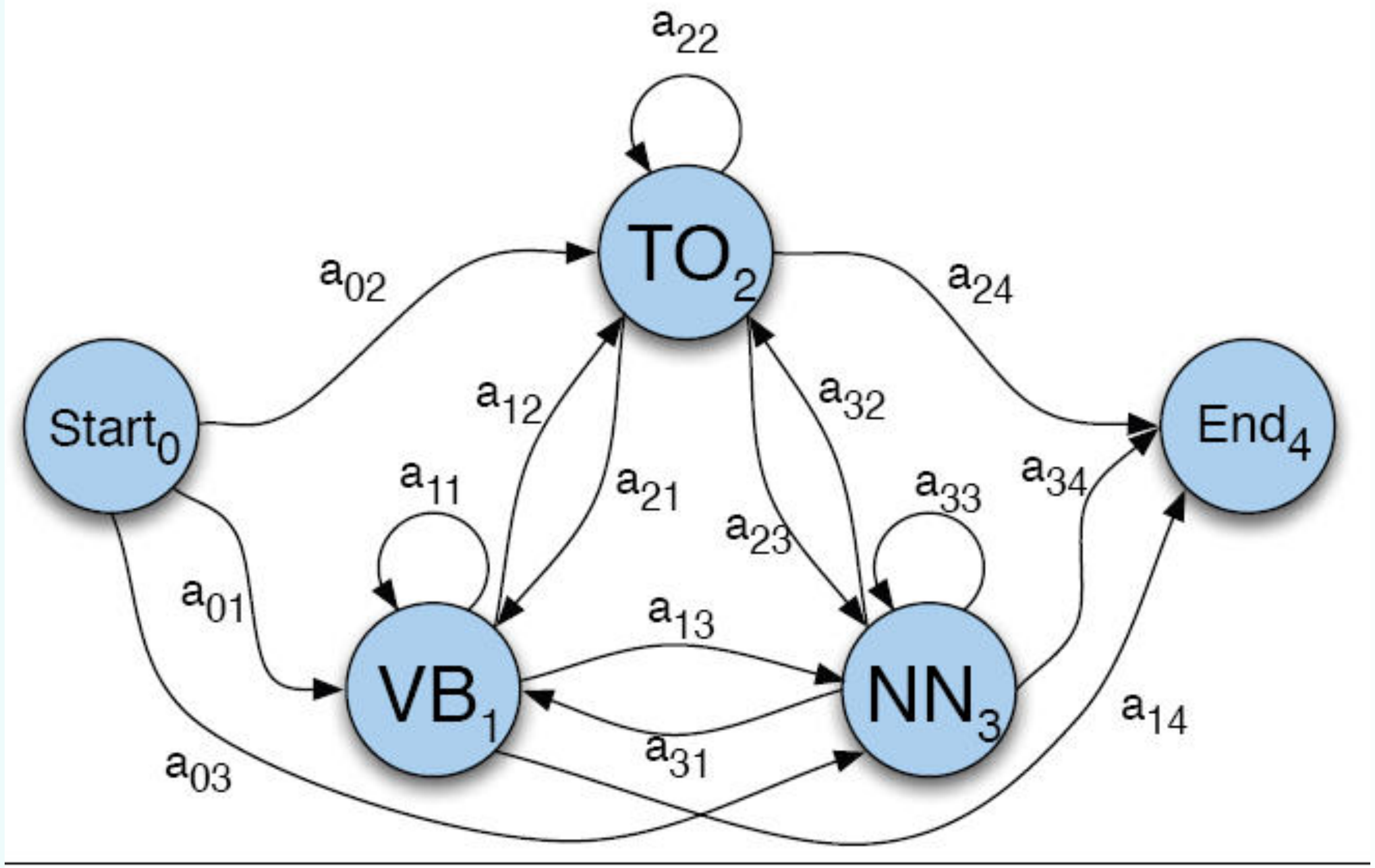
$P(\mathrm{NN}\|\mathrm{TO}) = .00047$ $P(\mathrm{VB}\|\mathrm{TO}) = 0.83$	Transition prob
$P(\mathrm{race}\|\mathrm{NN}) = 0.00057$ $P(\mathrm{race}\|\mathrm{VB}) = 0.00012$	Emission prob
$P(\mathrm{NR}\|\mathrm{VB}) = 0.0027$ $P(\mathrm{NR}\|\mathrm{NN}) = 0.0012$	Transition prob

$$P(\mathrm{VB}|\mathrm{TO})P(\mathrm{race}|\mathrm{VB})P(\mathrm{NR}|\mathrm{VB}) = .00000027$$
$$P(\mathrm{NN}|\mathrm{TO})P(\mathrm{race}|\mathrm{NN})P(\mathrm{NR}|\mathrm{NN}) =.00000000032$$

So we (correctly) choose the **verb** reading

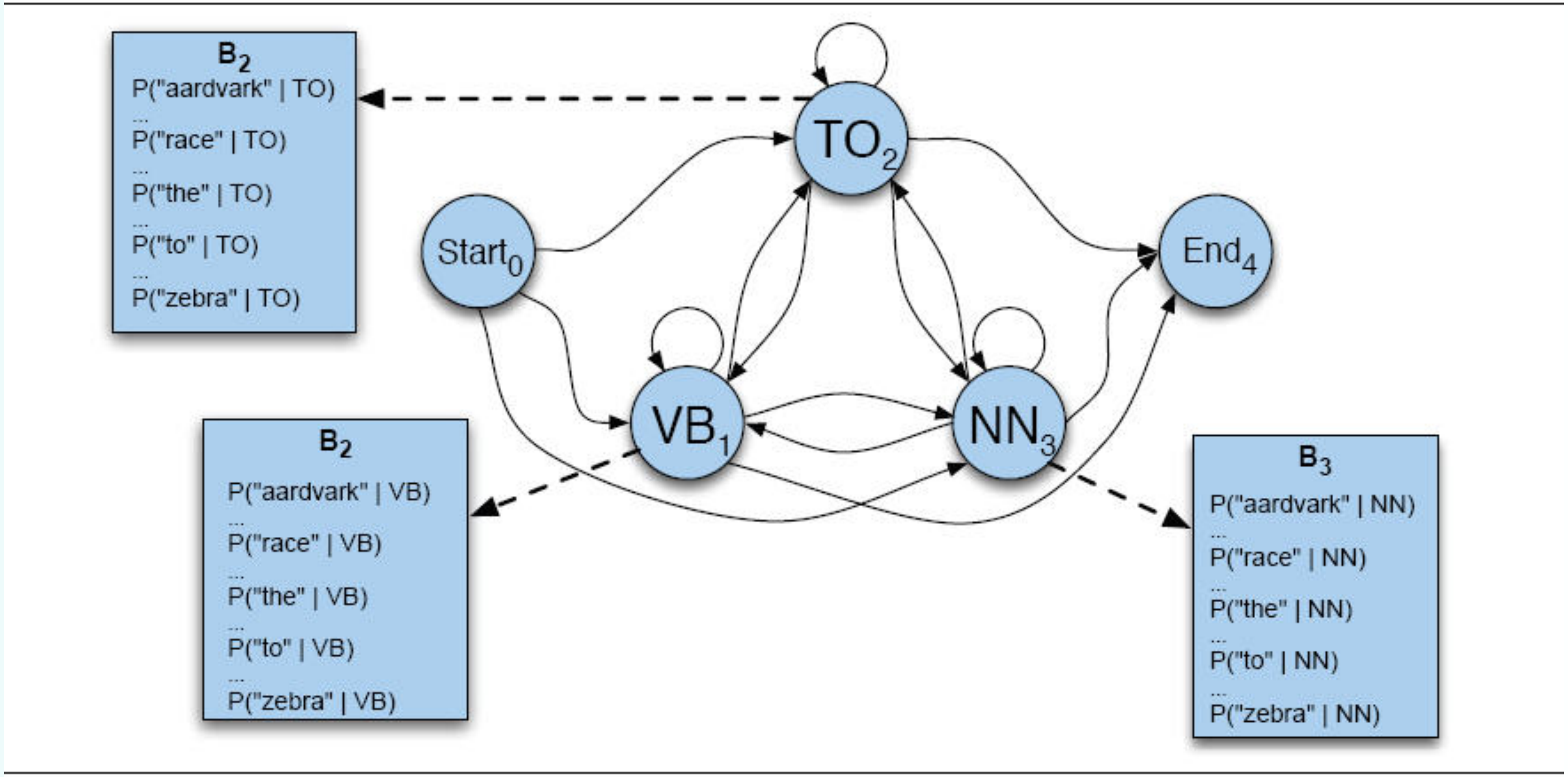
HMM for PoS tagging

Transitions probabilities A between the hidden states: tags



B observation likelihoods for POS HMM

- **Emission probabilities** B: words



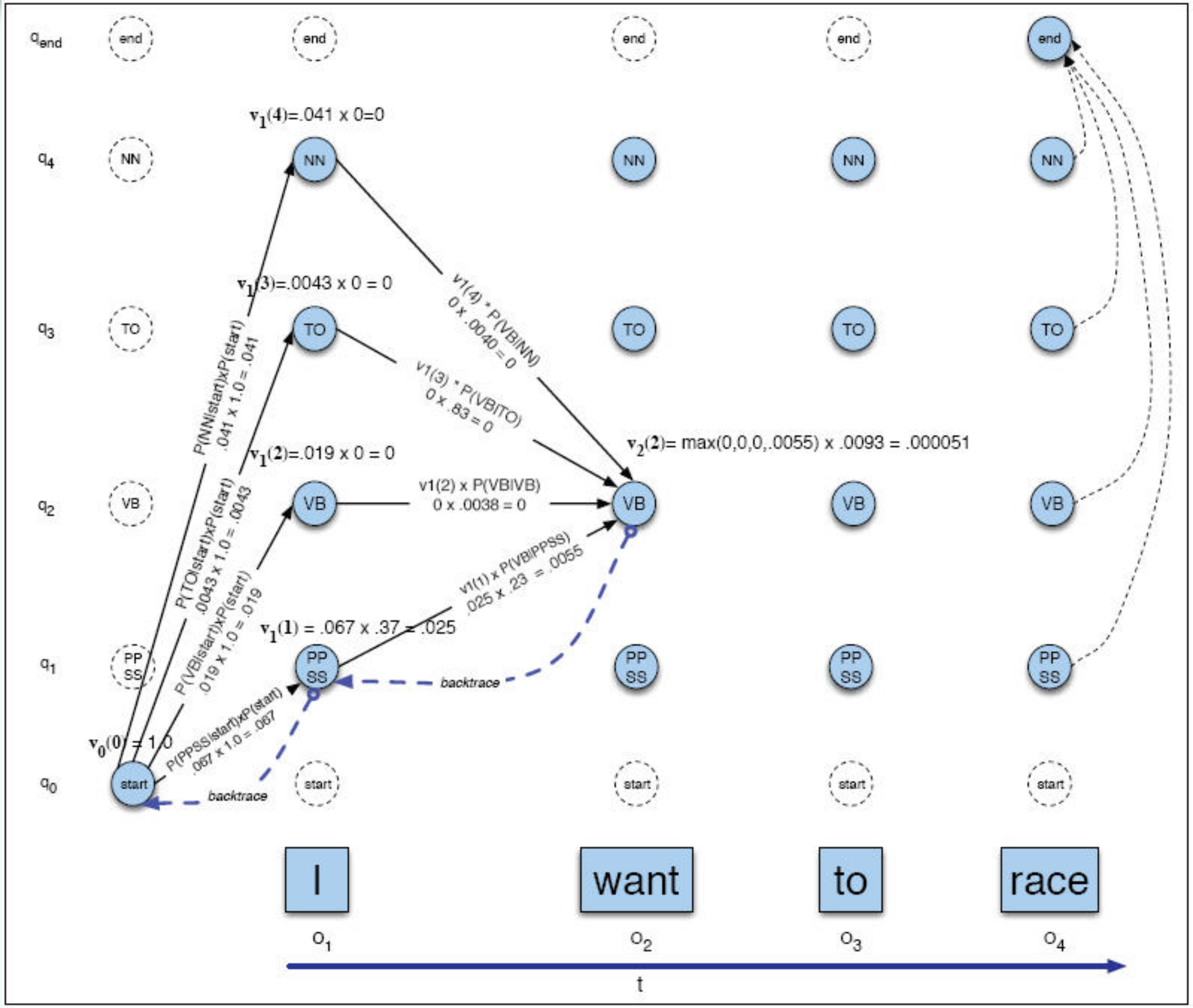
The A matrix for the POS HMM

	VB	TO	NN	PPSS
<s>	0.019	0.0043	0.041	0.0076
VB	0.0038	0.035	0.047	0.007
TO	0.83	0	0.00047	0
NN	0.004	0.016	0.087	0.0045
PPSS	0.23	0.0008	0.012	0.0001

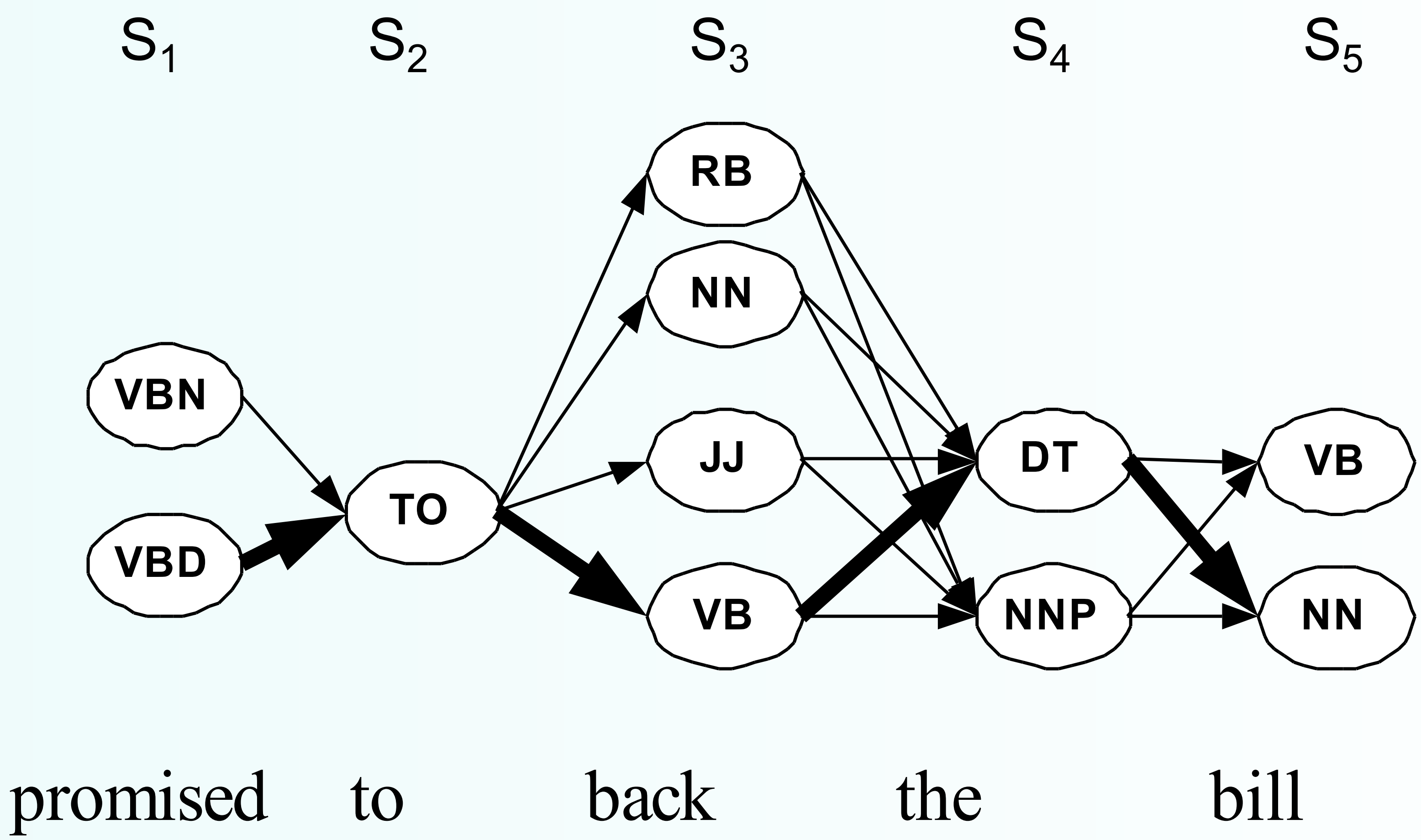
The B matrix for the POS HMM

	I	want	to	race
VB	0	0.093	0	0.0012
TO	0	0	0.99	0
NN	0	0.0005	0	0.0057
PPSS	0.37	0	0	0

Viterbi example



Viterbi intuition: looking for the best 'path'



Slide from Dekang Lin

Outline

- Markov Chains
- Hidden Markov Models
- Three Algorithms for HMMs
  - The Forward Algorithm
  - The Viterbi Algorithm
  - The Baum-Welch (EM Algorithm)
- Applications:
  - The Ice Cream Task
  - Part of Speech Tagging