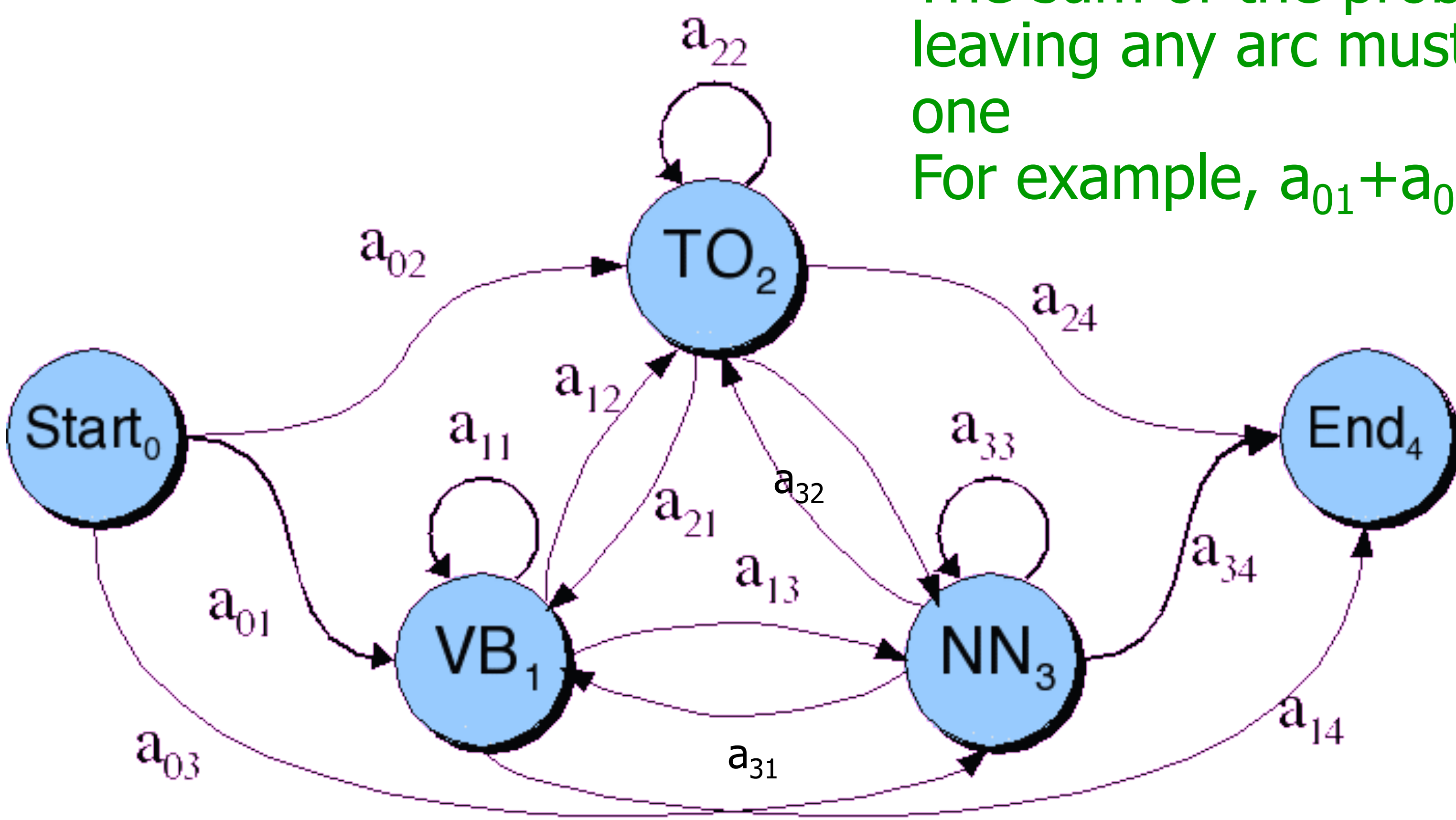
Part-of-speech Tagging and HMM

Hidden Markov Models

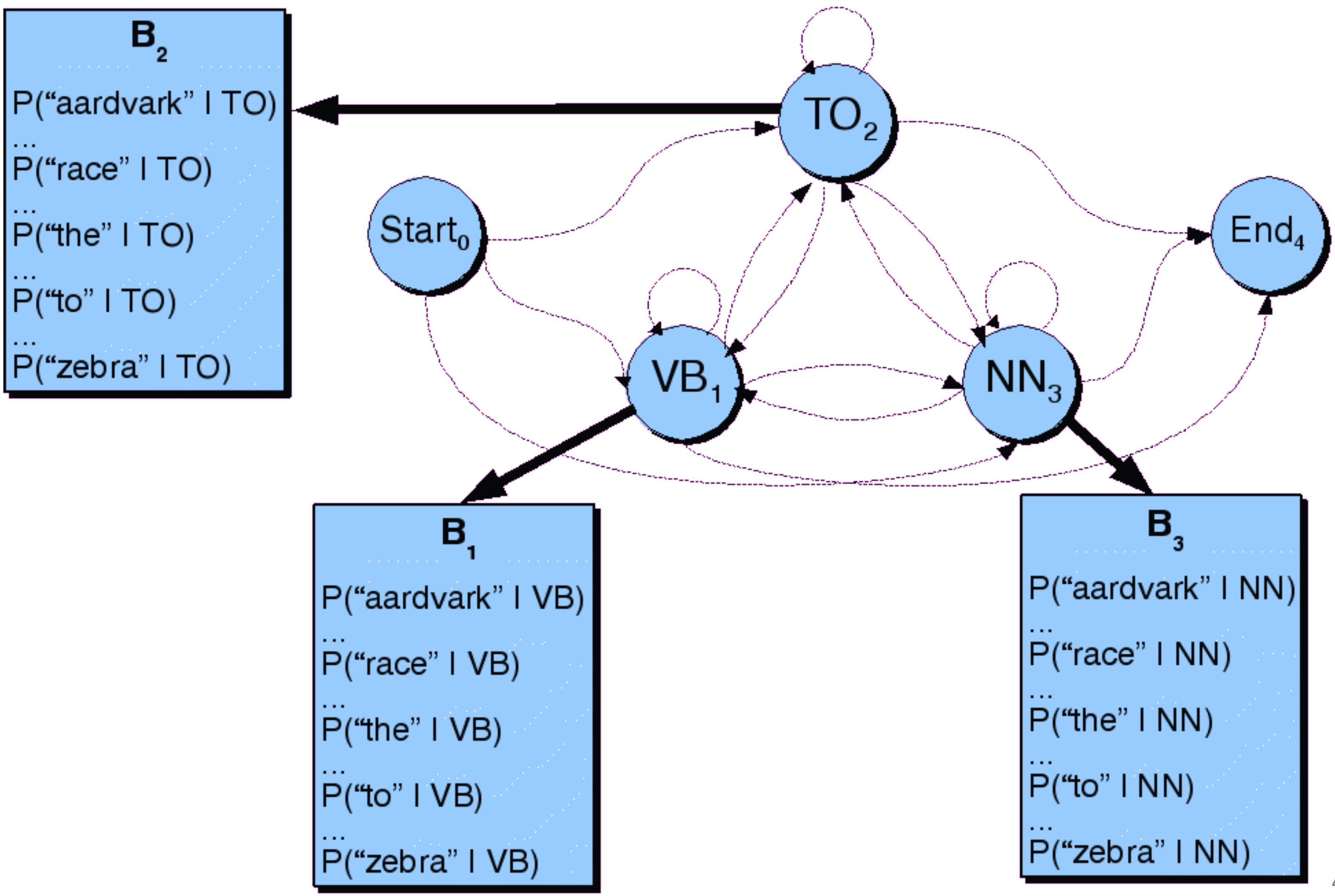
- What we've described with these two kinds of probabilities is a Hidden Markov Model (HMM)
  - Transition Probabilities
  - Observation Likelihoods
- Formalizing HMM: A weighted finite-state automaton where each arc is associated with a probability
  - The probability indicates how likely a path is to be taken

Transition Probabilities



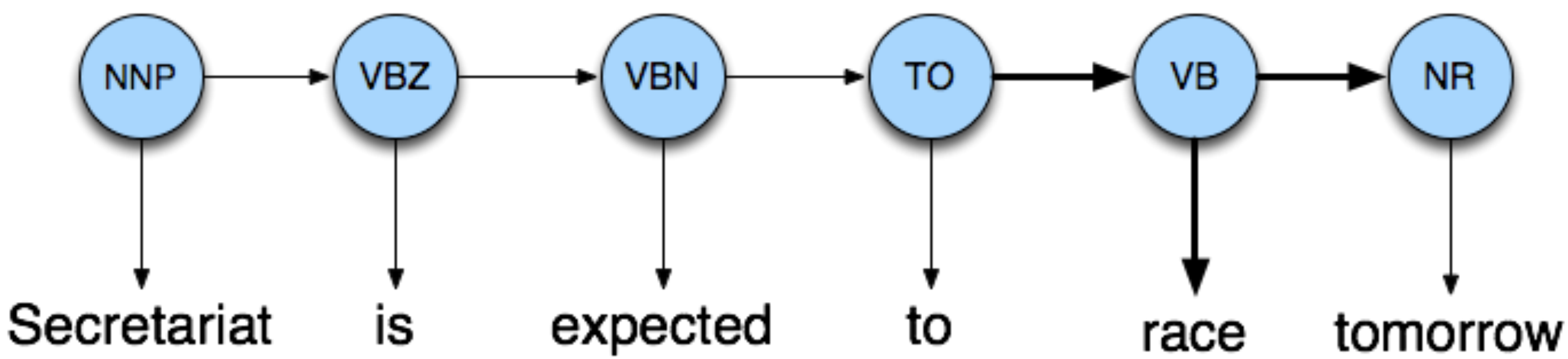
The sum of the probabilities leaving any arc must sum to one
For example, $a_{01}+a_{02}+a_{03} = 1$

Observation Likelihoods



Hidden Markov Model

- in part-of-speech tagging
  - The input symbols are **words**
  - But the hidden states are **part-of-speech tags**



- It has many other applications
  - Named entity recognition, gene prediction, etc

Hidden Markov Models

- States $Q = q_1, q_2 \ldots q_{N;}$ Tags $q_0, q_{F;}$ start and end states
- Observations $O = o_1, o_2 \ldots o_{T;}$ Words
  - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, \ldots v_V\}$
  - s_i: the state of the ith observation
- Transition probabilities
  - Transition probability matrix $A = \{a_{ij}\}$

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

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

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

- Special initial probability vector π

$$\pi_i = P(s_1 = i) \quad 1 \le i \le N$$

q_0, q_F not associated with observation
Start state $a_{01} \ldots a_{0N}$
a_{01} a_{02} a_{03}
End state $a_{1F} \ldots a_{NF}$
a_{14} a_{24} a_{34}

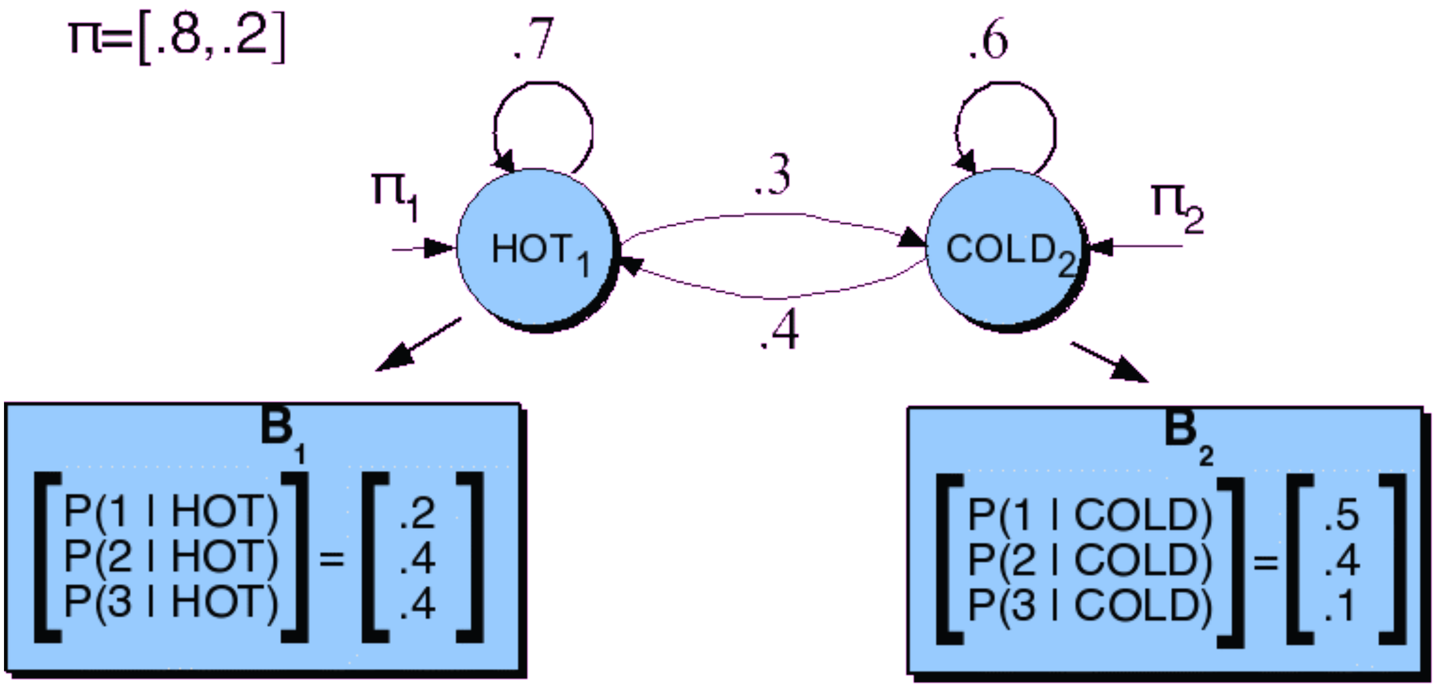
HMM for Ice Cream

- You are a climatologist in the year 2799 studying global warming
- You can’t find any records of the weather in Singapore for summer of 2012
- But you find your grandma’s diary which lists how many ice-creams she ate every date that summer
- Your job: figure out whether each day was cold/hot

Task

- Given : Observations O
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3…
  - Special initial probability vector
- Produce: States S
  - Weather Sequence: H,C,H,H,H,C…

HMM for Ice Cream

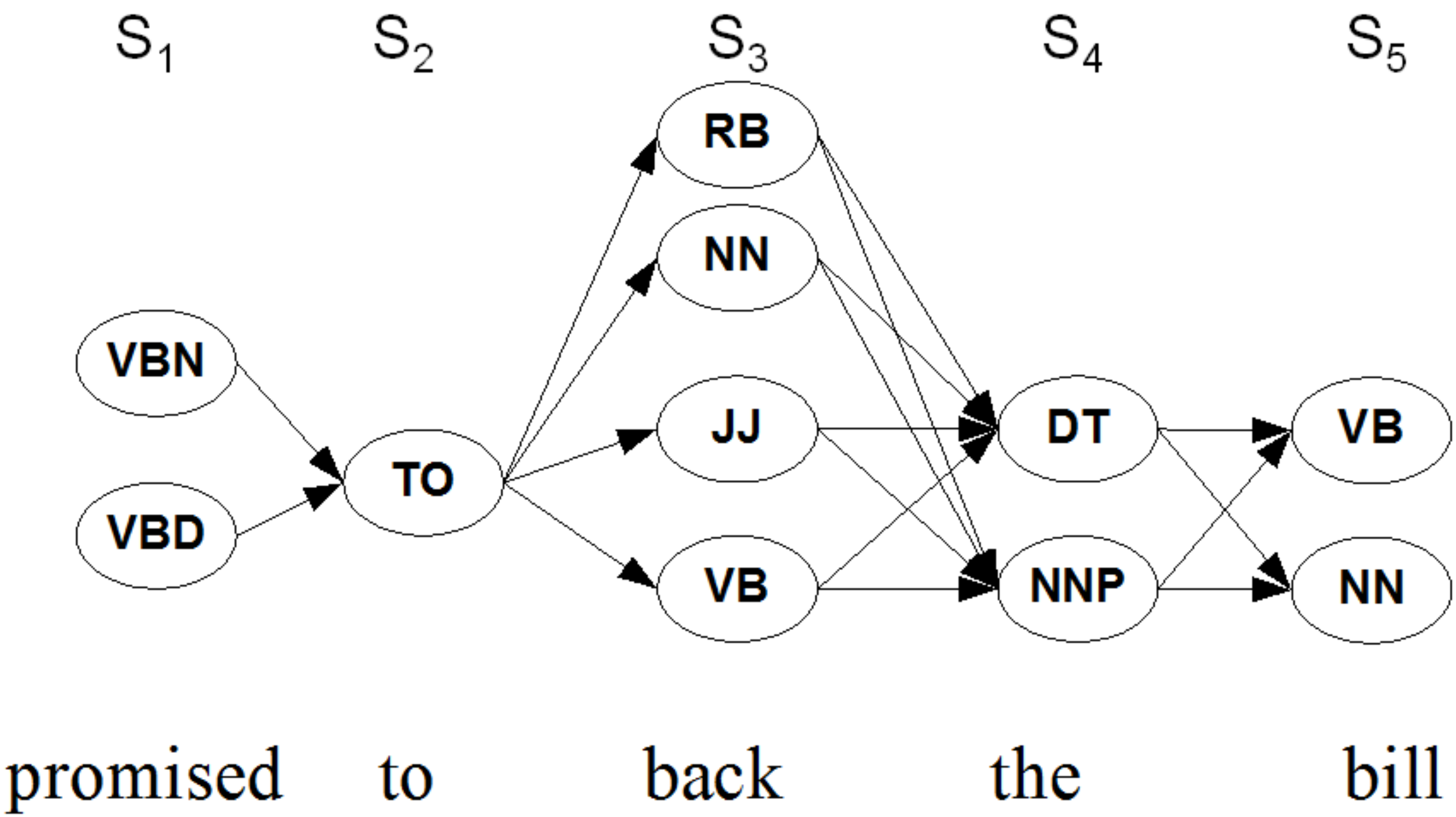


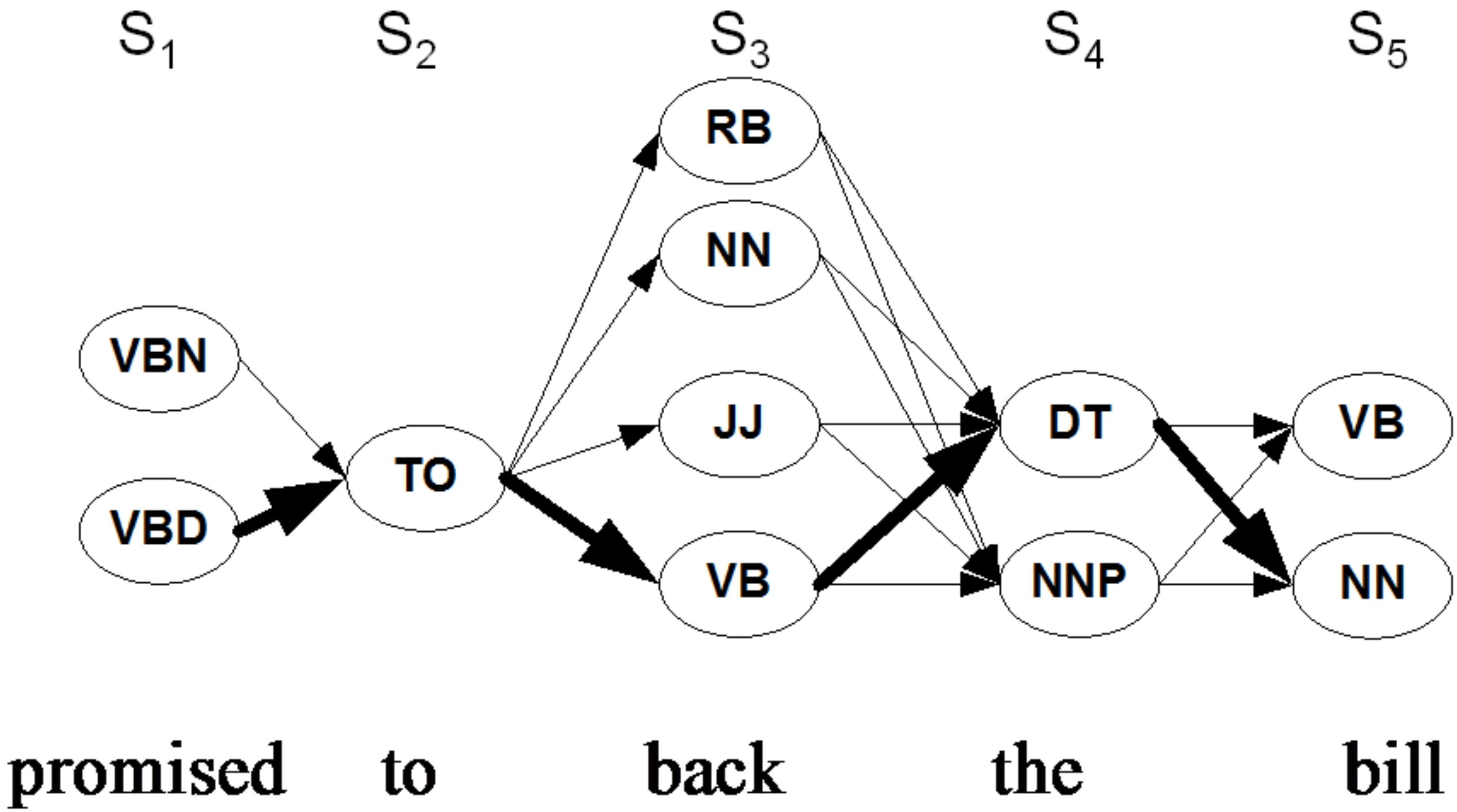
Decoding

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

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

- Determine sequences of variables, given sequence of observations
- We could just enumerate all paths given the input and use the model to assign probabilities to each.
  - Not a good idea. 1 2 -- HH, HC,CC,CH
  - N^T : N (number of states) T (size of sequence)
  - Luckily dynamic programming helps us here


S1
S2
S3
S4
S5
RB
NN
VBN
TO
JJ
DT
VB
VBD
VB
NNP
NN
promised
to
back
the
bill


S_1 S_2 S_3 S_4 S_5
RB
NN
VBN
JJ
DT
VB
TO
VBD
VB
NNP
NN
promised to back the bill

Intuition

- You’re interested in the shortest distance from here to Woodland
- Consider a possible location on the way to Woodland, say Jurong.
- Find the shortest distance among all the possible ways to get to Jurong
- Afterwards, we do not need to remember all the ways to Jurong, as finding the shortest distance from Jurong to Woodland

Intuition

- Consider a state sequence (tag sequence) that ends at state i with a particular tag T.
- The probability of that tag sequence can be broken into two parts
  - The probability of the BEST tag sequence up through i-1
  - Multiplied by the transition probability from the tag at the end of the i-1 sequence to T.
  - And the observation probability of the word given tag T.

Let T': the tag at the end of the i-1 sequence

W: the word at state i

Viterbi[T, i] = Viterbi[T', i-1] * p(T|T') * p(W|T)

$v_t[i]$ $a_{T',T}$ $b_T(W)$

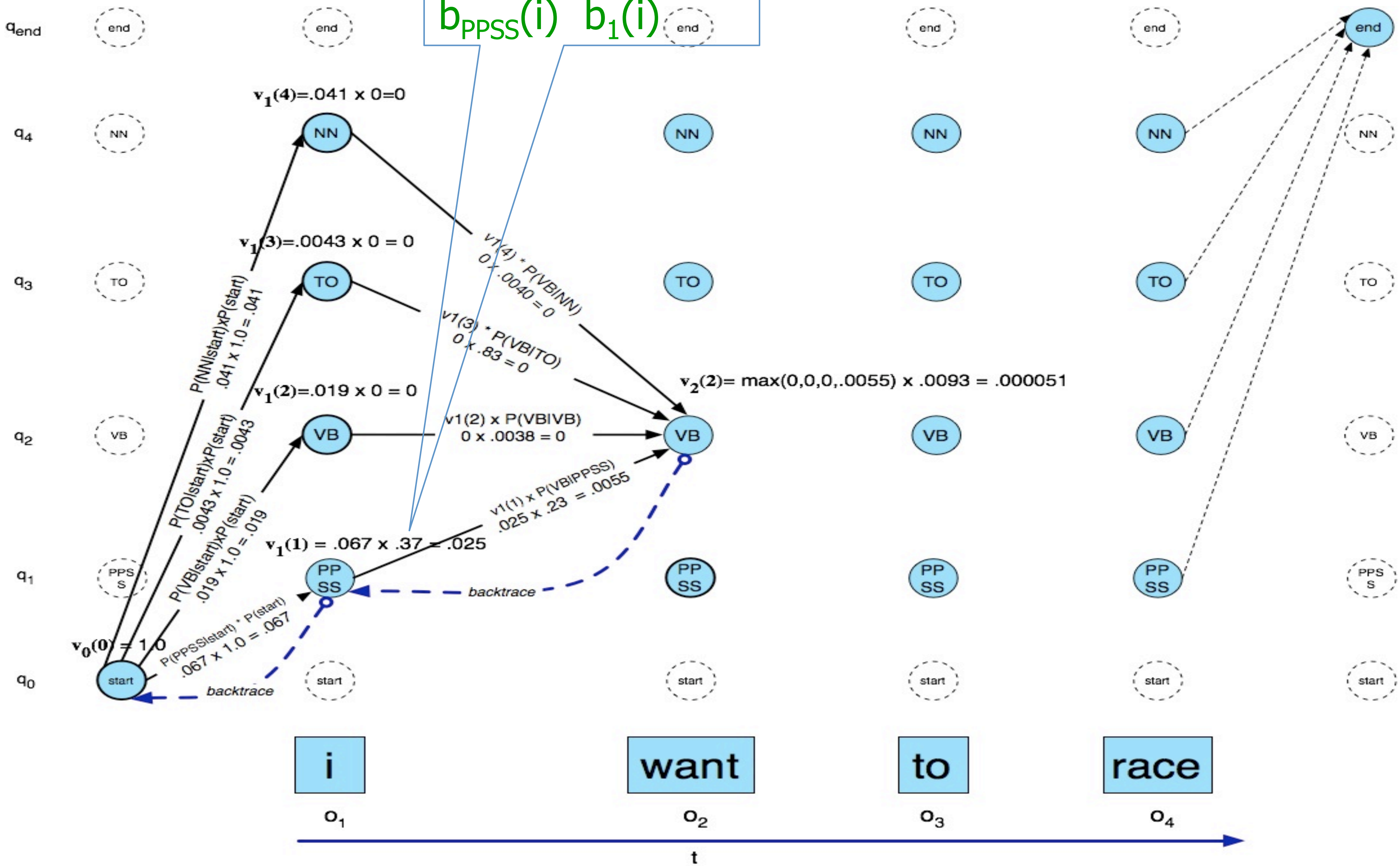
Main Idea

- We also have a matrix.
  - Each column– a time (observation)
  - Each row – a state
- Variable $v_t[i]$ the Viterbi path probability at time t for state i
  - Time t corresponds to observation
  - For each cell $v_t[i]$, we compute the probability of the best path to the cell

Viterbi Example

Variable $v_t[i]$ the Viterbi path probability at time t for state i

.37 : p(i| PPSS)
$b_{PPSS}(i)$ $b_1(i)$



The Viterbi Algorithm

function VITERBI($\boxed{\textit{observations} \text{ of len } T, \textit{state-graph} \text{ of len } N}$) **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 $backpointer[q_F,T]$

T: word N: tag

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).

summary

- HMM
  - Transition Probabilities
  - Observation Likelihoods
- Decoding
  - Viterbi
- Next
  - Evaluation
  - Assigning probabilities to inputs
    - Forward
  - Finding optimal parameters for a model

Evaluation

- So once you have your 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...

Error Analysis

- Look at a confusion matrix

Returned by tagger

correct

	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)

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.

3 Problems

- Given this framework there are 3 problems that we can pose to an HMM
  - Given an observation sequence and a model, what is the most likely state sequence?
  - Given an observation sequence, what is the probability of that sequence given a model?
  - Given an observation sequence, infer the best model parameters for model

Problem

- Most probable state sequence given a model and an observation sequence

Decoding: Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, ..., o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$.

  - Typically used in tagging problems, where the tags correspond to hidden states
  - Viterbi solves problem

Problem

- The probability of a sequence given a model.. P(seq|model).

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

  - Forward algorithm

[redacted]

- [redacted]

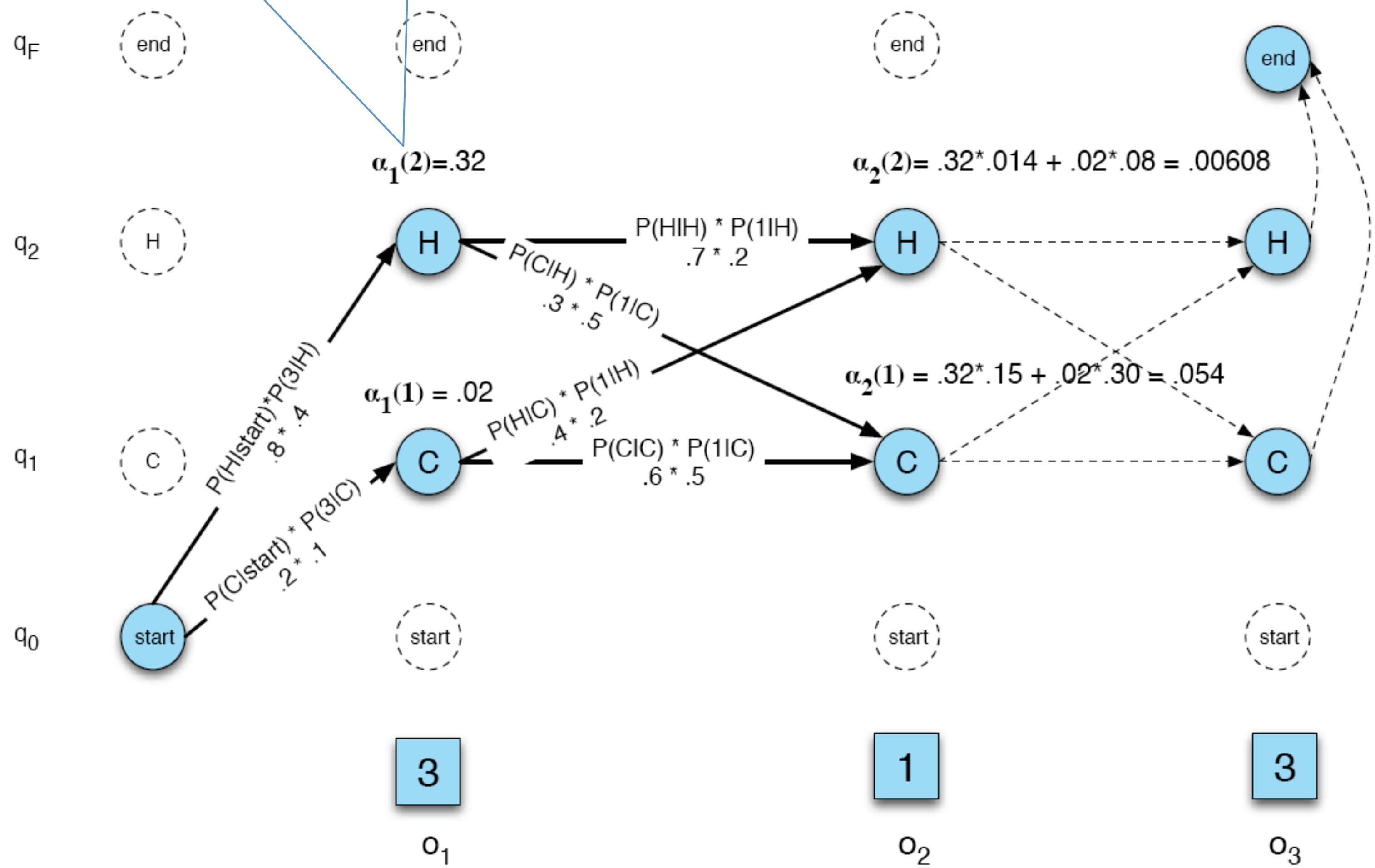
  [redacted]

[redacted]

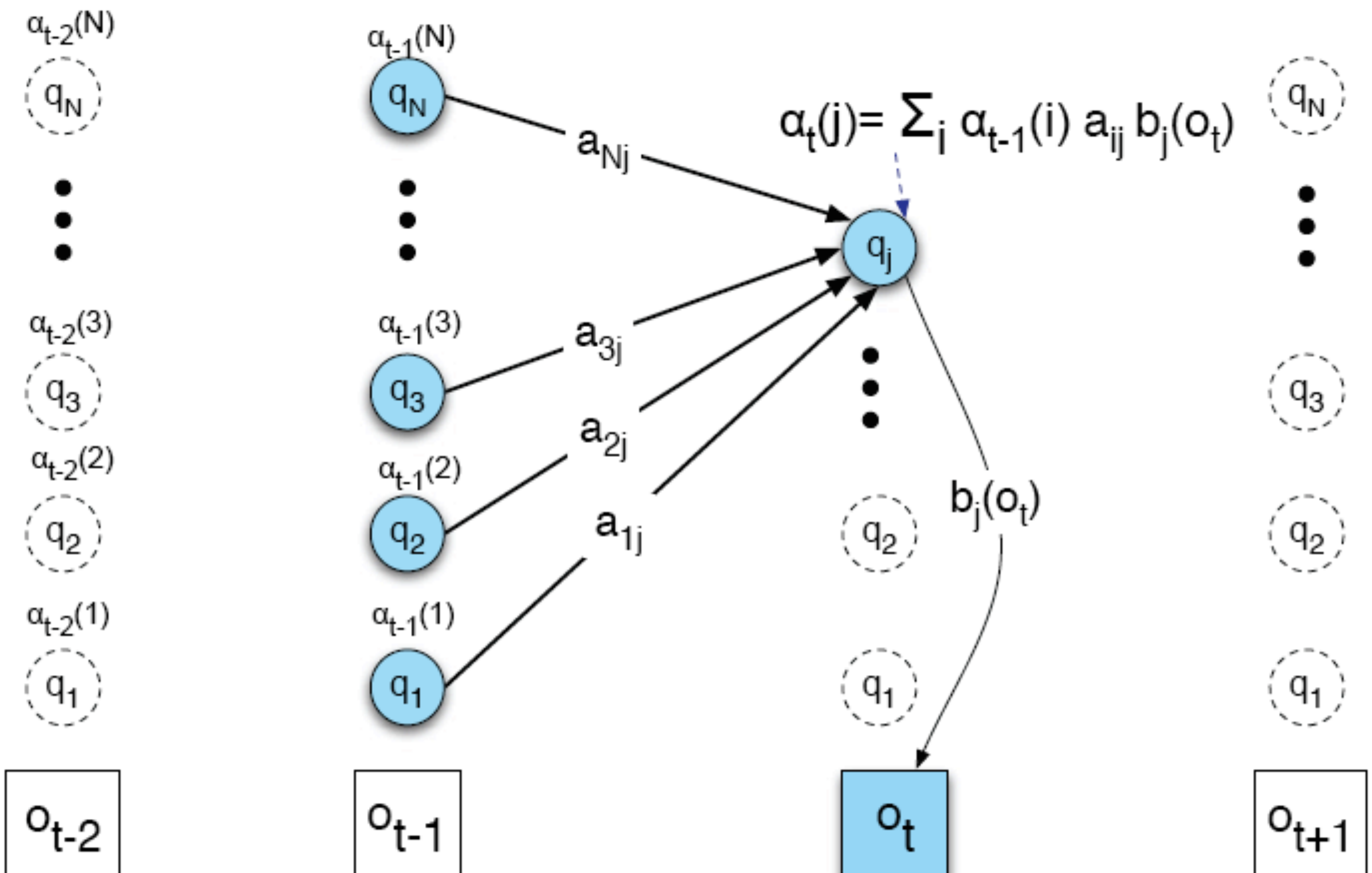
[redacted]

forward[H,3 icecream]
forward [2, 1]

Variable $a_t[i]$ the forward path probability at time t for state i



[redacted]



$\alpha_{t-2}(N)$
$\alpha_{t-1}(N)$
$\alpha_t(j)= \Sigma_i\ \alpha_{t-1}(i)\ a_{ij}\ b_j(o_t)$
q_N
a_{Nj}
q_j
$\alpha_{t-2}(3)$
$\alpha_{t-1}(3)$
a_{3j}
q_3
$\alpha_{t-2}(2)$
$\alpha_{t-1}(2)$
a_{2j}
q_2
$b_j(o_t)$
a_{1j}
$\alpha_{t-2}(1)$
$\alpha_{t-1}(1)$
q_1
o_{t-2}
o_{t-1}
o_t
o_{t+1}

[redacted]

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 $forward[q_F,T]$

Problem

- Infer the best model parameters, given a model and an observation sequence...
  - That is, fill in the A and B tables with the right numbers...
    - The numbers that make the observation sequence most likely
  - Useful for getting an HMM without having to hire annotators...
  - Baum-Welch (or forward-backward): Expectation-Maximization (EM) (Section 6.5-6.8)

Summary

- HMM model- two probabilities
- Viterbi algorithm
- Evaluation
- Three problems in HMM model