CZ/CE4045 Natural Language Processing

Part-of-speech Tagging and HMM

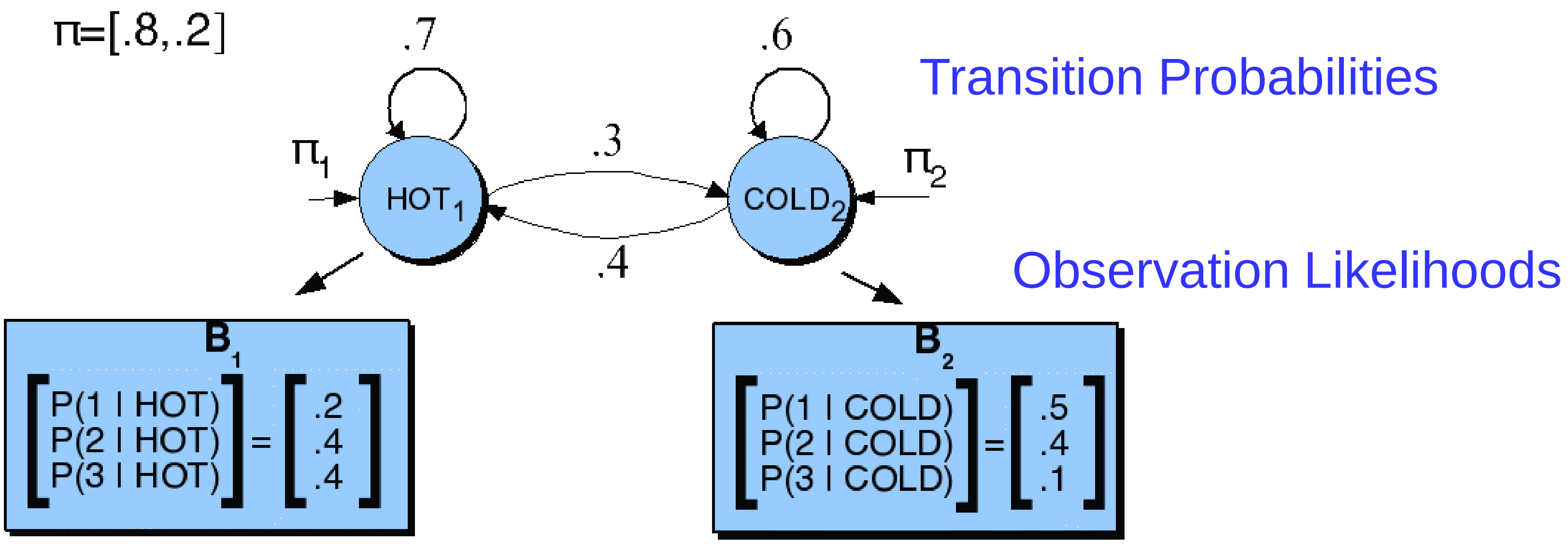
(Chapter 6)

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 2018
- 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

Example of sequence prediction

- Can the number of ice cream eaten be used to **predict** the weather?
  - Ice cream observation sequence: 2,1,3,2,2…
  - Weather Sequence: H,C,H,H,C…



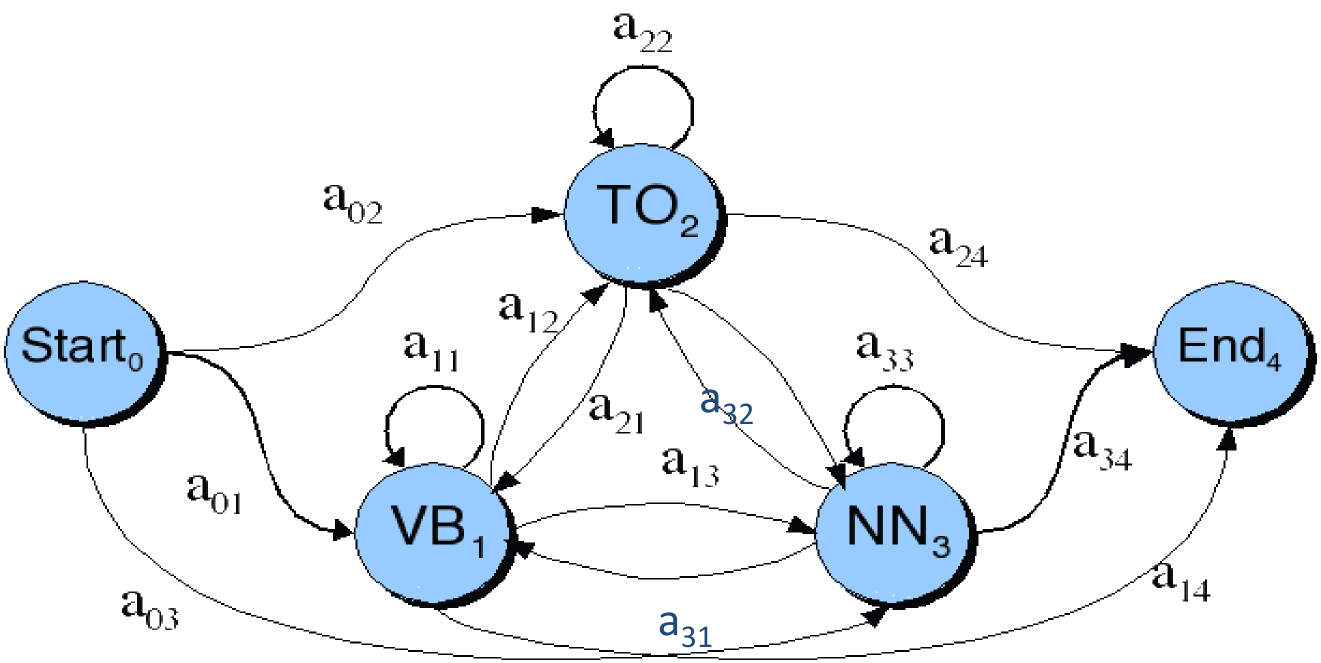
Hidden Markov Models

- What we've described with these two kinds of probabilities is a Hidden Markov Model (HMM)
  - Transition Probabilities
  - Observation Likelihoods

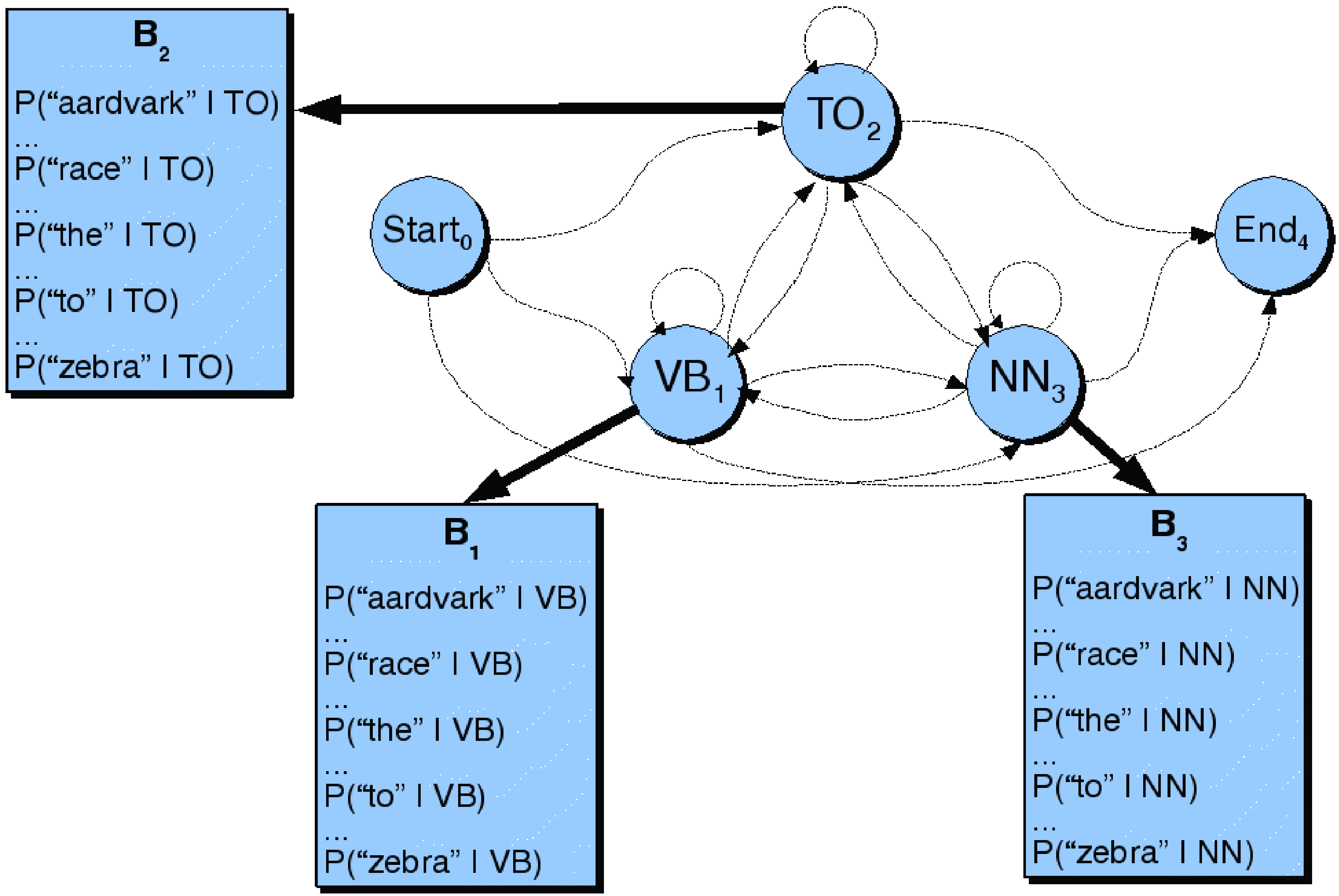
- Formalizing HMM: A **weighted finite-state automaton** where each arc is associated with a probability
  - The probability indicates how likely a path is to be taken

Transition Probabilities

- The sum of the probabilities leaving any arc must sum to one
  - For example, a01+a02+a03 =1

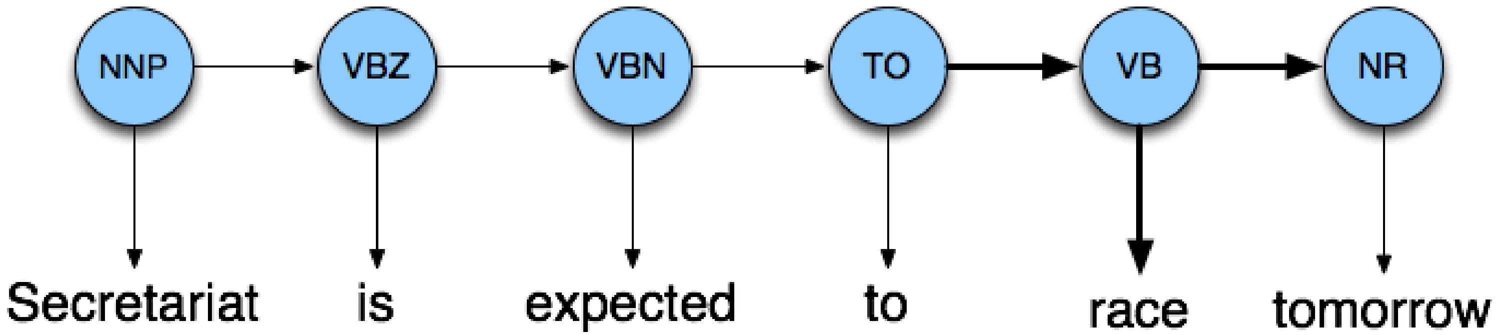


Observation Likelihoods



Hidden Markov Model

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



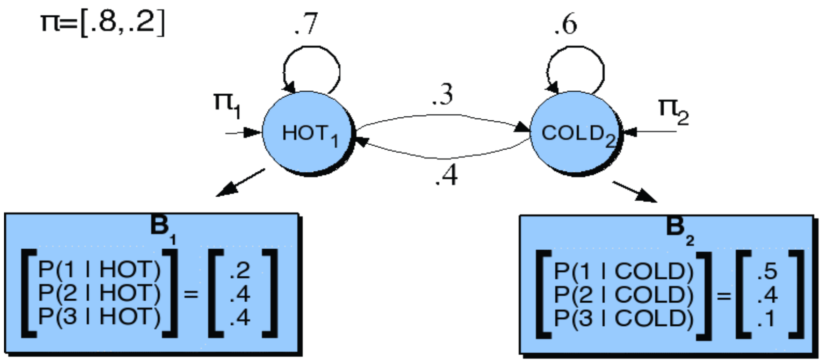
- It has many other applications
  - Named entity recognition, gene prediction, etc

Hidden Markov Models

- States $Q = q_1, q_2 \dots q_N$; and the start and end states $\boldsymbol{q_0, q_F}$
- Observations $O = o_1, o_2 \dots o_T$;
  - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, \dots v_V\}$
  - s_i: the state of the i-th observation;
  - $q_0,\ q_F$ are not associated with observations
- Transition probabilities: Transition probability matrix $A = \{a_{ij}\}$;
  - $a_{ij} = \mathrm{P}(\mathrm{s_t = j | s_{t-1} = i})\ \ 1 \leq i, j \leq N$
- Observation likelihoods: Output probability matrix $B = \{b_i(k)\}$;
  - $b_i(k) = P(X_t = o_k | s_t = i)$
- Special initial probability vector π;
  - $\pi_i = P(s_1 = i)\ 1 \leq i \leq N$

Exercise: Hidden Markov Model for the ice cream task

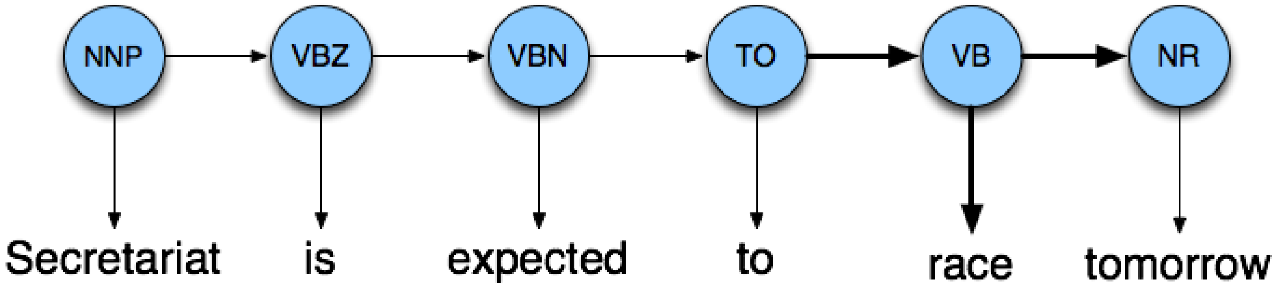
- State the following elements of the HMM for weather prediction based on ice cream and depict an example HMM:
  - States
  - Vocabulary
  - Observations
  - Transition probabilities
  - Observation likelihoods



Recall

Hidden Markov Model

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



$$\begin{aligned}
&p(Secretariat|NNP) * p(NNP|Start) \\
&* p(is|VBZ) * p(VBZ|NNP) \\
&* p(expected|VBN) * p(VBN|VBZ) \\
&* p(to|TO) * p(TO|VBN) \\
&* p(race|VB) * p(VB|TO) \\
&* p(tomorrow|NR) * p(NR|VB)
\end{aligned}$$

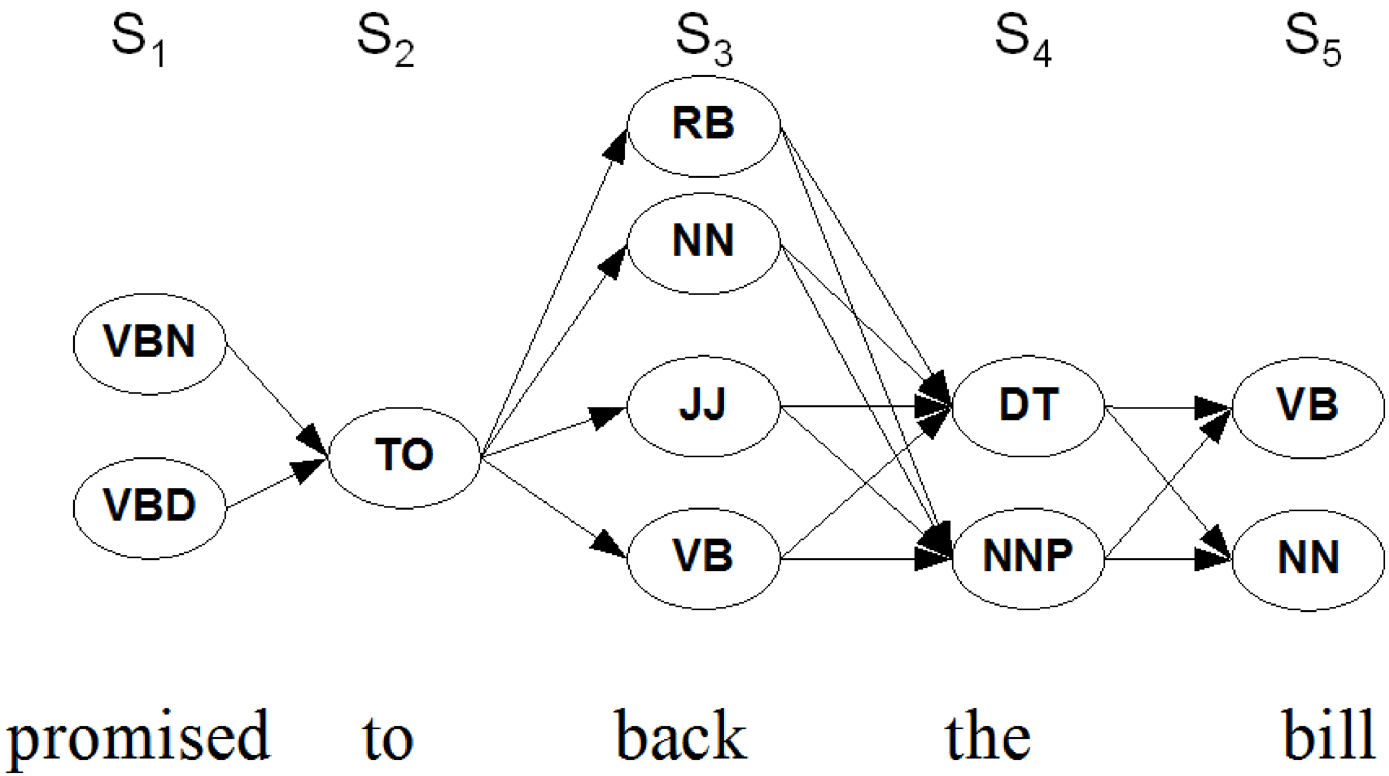
Decoding

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

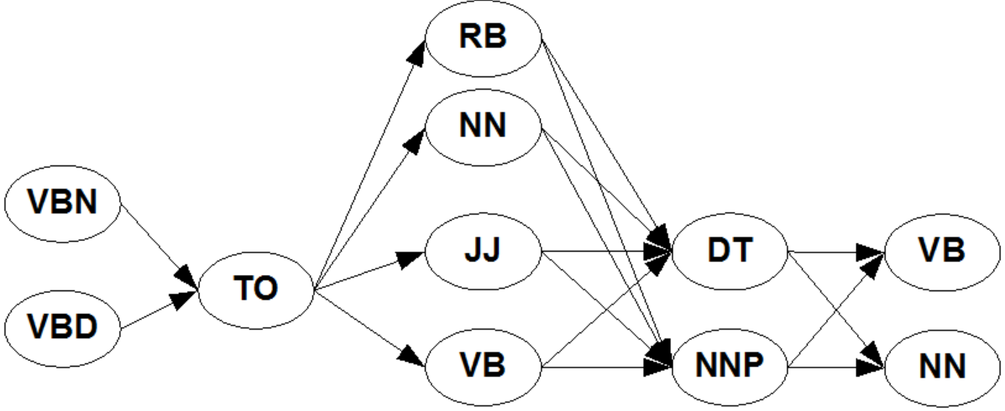
$$\hat{t}_1^n = arg \max_{t_1^n} P(t_1^n | w_1^n)$$

- 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

Example sentence



Enumerate all paths



- VBN TO RB DT VB
- VBN TO RB DT NN
- VBN TO RB NNP VB
- VBN TO RB NNP NN
- VBN TO NN DT VB
- VBN TO NN DT NN
- VBN TO NN NNP VB
- VBN TO NN NNP NN
- VBN TO JJ DT VB
- VBN TO JJ DT NN
- VBN TO JJ NNP VB
- VBN TO JJ NNP NN
- VBN TO VB DT VB
- VBN TO VB DT NN
- VBN TO VB NNP VB
- VBN TO VB NNP NN

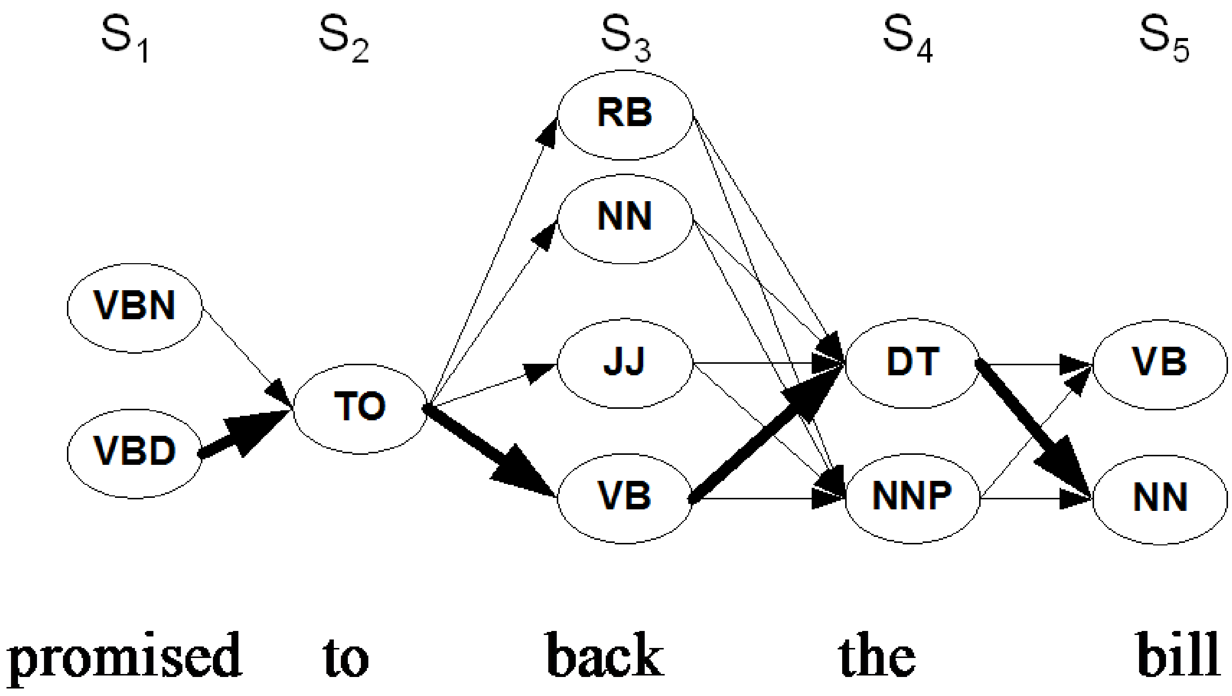
- VBD TO RB DT VB
- VBD TO RB DT NN
- VBD TO RB NNP VB
- VBD TO RB NNP NN
- VBD TO NN DT VB
- VBD TO NN DT NN
- VBD TO NN NNP VB
- VBD TO NN NNP NN
- VBD TO JJ DT VB
- VBD TO JJ DT NN
- VBD TO JJ NNP VB
- VBD TO JJ NNP NN
- VBD TO VB DT VB
- VBD TO VB DT NN
- VBD TO VB NNP VB
- VBD TO VB NNP NN

Estimate the probabilities of all paths

- $p(VBN| < s >) * p(promised|VBN) * p(TO|VBN) * p(to|TO) * p(RB|TO) * p(back|RB) * p(DT|RB) * p(the|DT) * p(VB|DT) * p(bill|VB)$
- $p(VBN| < s >) * p(promised|VBN) * p(TO|VBN) * p(to|TO) * p(RB|TO) * p(back|RB) * p(DT|RB) * p(the|DT) * p(NN|DT) * p(bill|NN)$
- $p(VBN| < s >) * p(promised|VBN) * p(TO|VBN) * p(to|TO) * p(RB|TO) * p(back|RB) * p(NNP|RB) * p(the|NNP) * p(VB|NNP) * p(bill|VB)$
- $p(VBN| < s >) * p(promised|VBN) * p(TO|VBN) * p(to|TO) * p(RB|TO) * p(back|RB) * p(NNP|RB) * p(the|NNP) * p(NN|NNP) * p(bill|NN)$
- $\ldots$

- $p(VBD| < s >) * p(promised|VBD) * p(TO|VBD) * p(to|TO) * p(RB|TO) * p(back|RB) * p(DT|RB) * p(the|DT) * p(VB|DT) * p(bill|VB)$
- $p(VBD| < s >) * p(promised|VBD) * p(TO|VBD) * p(to|TO) * p(RB|TO) * p(back|RB) * p(DT|RB) * p(the|DT) * p(NN|DT) * p(bill|NN)$
- $p(VBD| < s >) * p(promised|VBD) * p(TO|VBD) * p(to|TO) * p(RB|TO) * p(back|RB) * p(NNP|RB) * p(the|NNP) * p(VB|NNP) * p(bill|VB)$
- $p(VBD| < s >) * p(promised|VBD) * p(TO|VBD) * p(to|TO) * p(RB|TO) * p(back|RB) * p(NNP|RB) * p(the|NNP) * p(NN|NNP) * p(bill|NN)$
- $\ldots$

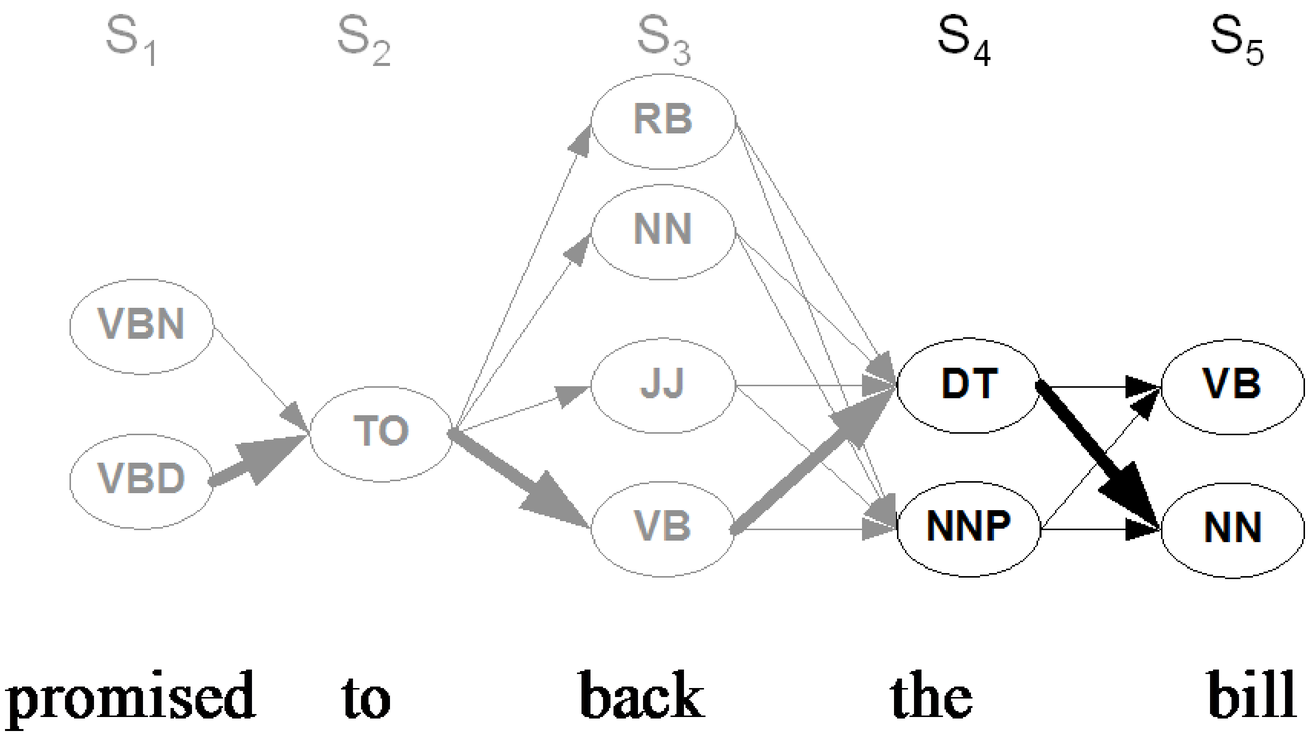
The best choice?



Intuition

- You’re interested in the shortest distance from NTU to Woodland
- Consider a possible location on the way to Woodland, say Jurong.
  - We can work out the shortest distance among all the possible ways from Jurong to Woodland
    - **Jurong → Woodland**
  - Then we only need to work out shortest path from NTU to Jurong
    - **NTU → Jurong**

Back to our example

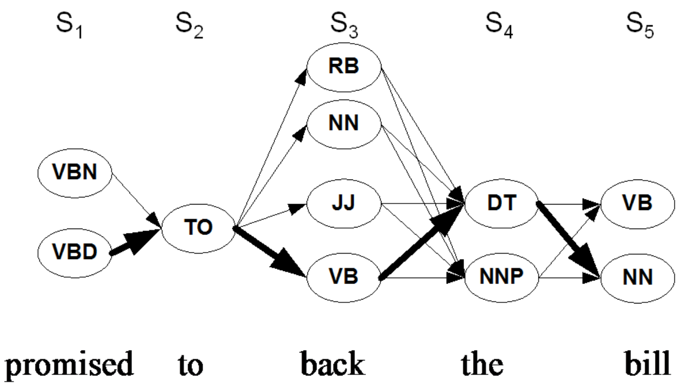


Intuition

- Consider a state sequence (tag sequence) that ends at time t with a particular tag i.
- The probability of that tag sequence can be broken into two parts
  - The probability of the BEST tag sequence up through $t-1$
  - Multiplied **by the transition probability** from the tag at the end of the $t-1$ sequence to i.
  - And the observation probability of the word given tag i.
- Let j be the tag at the end of the $t-1$ sequence, and W be the word at time t

$$\underbrace{Viterbi[i,t]}_{v_t[i]} = Viterbi[j,t-1] \times \underbrace{p(i|j)}_{a_{j,i}} \times \underbrace{p(W|i)}_{b_i(W)}$$

Consider paths ending with bill:NN



From S4, we have two paths P1, P2 to reach NN

- $p_1 = v_4[DT] * p(NN|DT) * p(bill|NN)$
- $p_2 = v_4[NNP] * p(NN|NNP) * p(bill|NN)$
- $v_5[NN] = \max(p_1, p_2)$
- $v_4[DT] = \max(v_3[RB] * p(DT|RB) * p(the|DT),$
  $v_3[NN] * p(DT|NN) * p(the|DT),$
  $v_3[JJ] * p(DT|JJ) * p(the|DT),$
  $v_3[VB] * p(DT|VB) * p(the|DT))$
  $= \max(v_3[RB] * p(DT|RB), v_3[NN] * p(DT|NN), v_3[JJ] * p(DT|JJ), v_3[VB] * p(DT|VB)) * p(the|DT)$

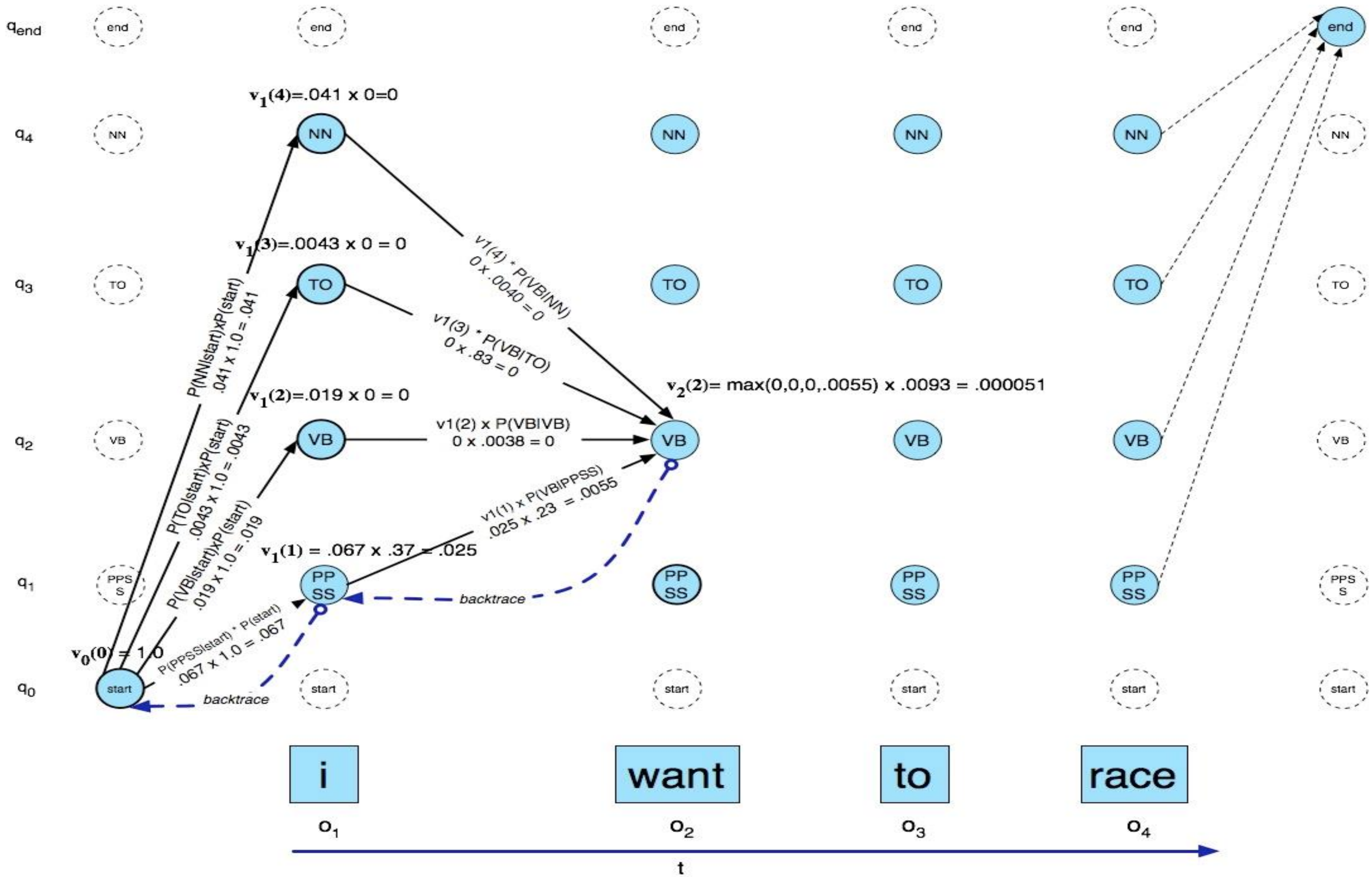
Main Idea

- We also have a matrix.
  - Each column– a time ‘t’ (observation)
  - Each row – a state ‘i’
  - For each cell $v_t[i]$, we compute the probability of the **best** path to the cell

- the Viterbi path probability at time t for state i
  - there are $|Q|$ number of paths from $t-1$ to $v_t[i]$
  - if we know the best path to each cell in $t-1$ (v_{t-1} $[j]$)

$$\arg\max_j v_{t-1}[j] \times P(i|j) \times P(s_t|i)$$

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



Viterbi algorithm: Example

- $v_2[NN] = \max(v_1[NN] * p(NN|NN), v_1[TO] * p(NN|TO), v_1[VB] * p(NN|VB), v_1[PPSS] * p(NN|PPSS)) * p(want|NN)$
- $= \max(0 * 0.087, 0 * 0.00047, 0 * 0.047, 0.025 * 0.0012) * 0.000054$

	VB	**TO**	**NN**	**PPSS**
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

	I	**want**	**to**	**race**
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

The Viterbi Algorithm

T: word N: tag

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 $backpointer[q_F,T]$

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 tags

	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 parameters for model (Skip; Section 6.5-6.8)

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

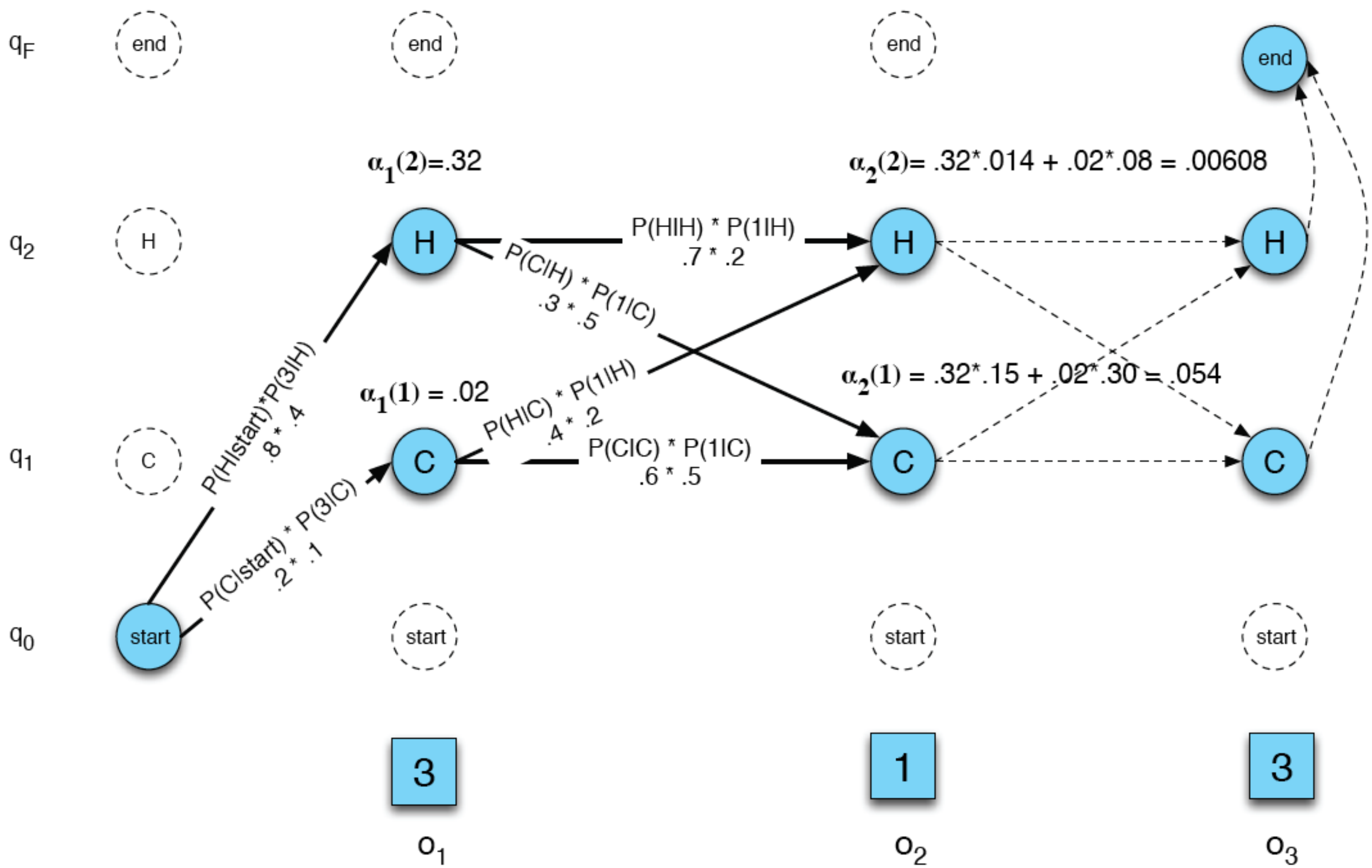
Forward

- Efficiently computes the probability of an observed sequence given a model
  - $P(sequence|model)$

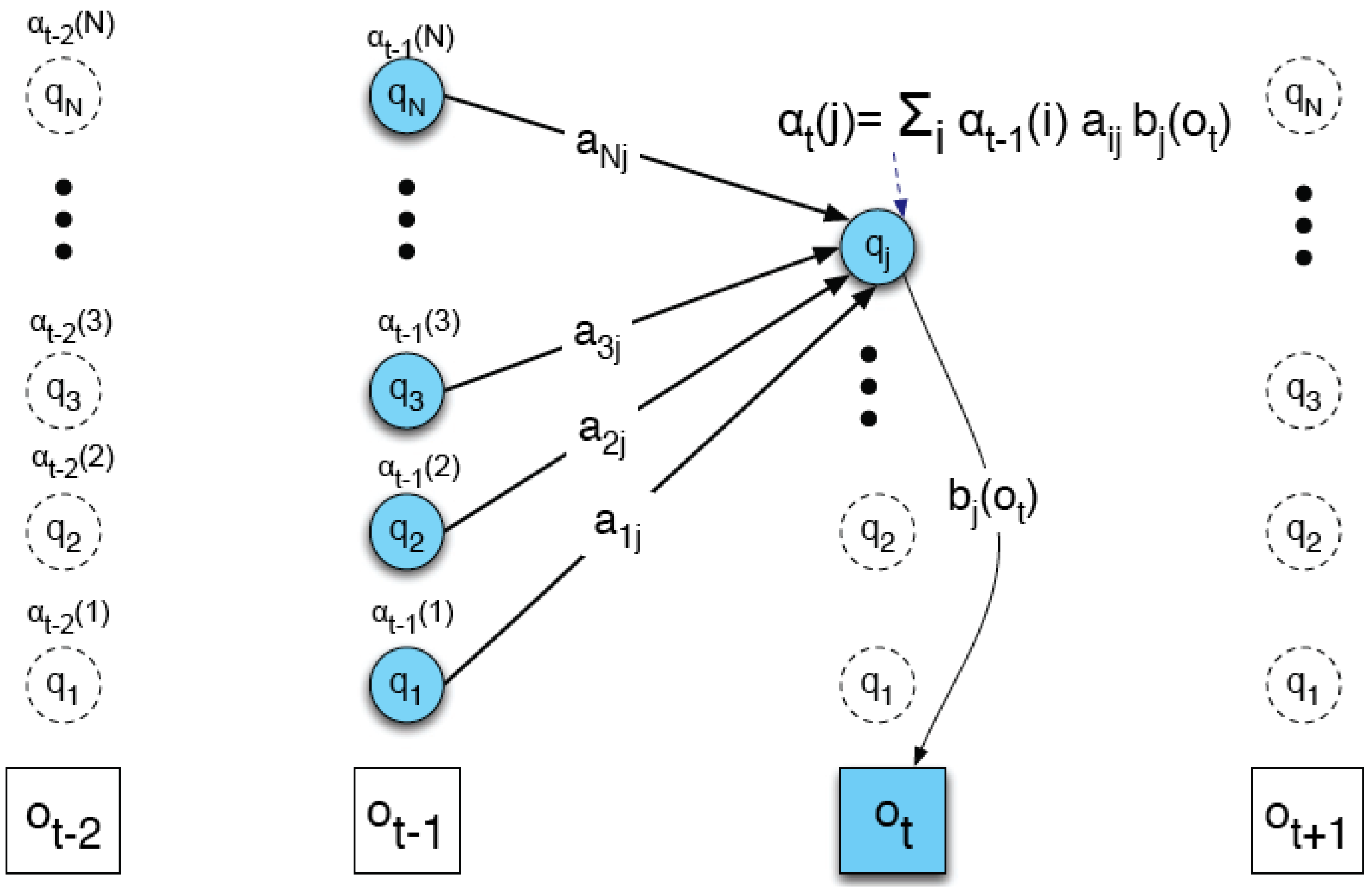
- Nearly identical to Viterbi;
  - replace the MAX with a **SUM**

Ice Cream Example

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



Forward algorithm: SUM



Forward

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 $forward[q_F, T]$

Summary

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