LING439/539 - Statistical NLP
HMM POS tagging

September 20-22 2016

HMM POS tagging

The best sequence of tags

- We want to choose the tag sequence that is most probable give the observation sequence of n word w_1^n $(w_1, w_2, ..., w_n)$.
- In other words, we want out of all sequence of n tags t_1^n the single tag sequence such that $P(t_1^n|w_1^n)$ is highest.

$$\hat{t}_1^n = \underset{t_1^n}{\text{argmax}}\, P(t_1^n|w_1^n) \tag{1}$$

where we want the particular tag sequence t_1^n that maximize the $\hat{t}_1^n$.

The function $\text{argmax}_x\, f(x)$ means "the x such that $f(x)$ is maximized".

Bayes' rule

Bayes' rule:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{2}$$

We don't know how to directly compute $P(t_1^n|w_1^n)$. Therefore,

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

$P(w_1^n)$ doesn't change for each tag sequence: we are always asking about the most likely tag sequence fro the same observation w_1^n, which mush have the same probability $P(w_1^n)$.

Two assumptions

1.

The probability of a word appearing depends only on its own POS tag: that is, it is **independent** of other words and other tags around it:

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i) \tag{3}$$

2.
The probability of a tag appearing is **dependent** only on the previous tag (bigram assumption), rather than the entire tag sequence.

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}) \tag{4}$$

HMM bigram tagger

$$
\begin{aligned}
\hat{t}_1^n &= \operatorname*{argmax}_{t_1^n} P(t_1^n | w_1^n) \\
&= \operatorname*{argmax}_{t_1^n} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)} \\
&= \operatorname*{argmax}_{t_1^n} P(w_1^n | t_1^n) P(t_1^n) \\
&\approx \operatorname*{argmax}_{t_1^n} \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1})
\end{aligned}
$$

Direct count

1.

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

where $C(t_{i-1}, t_i)$ is the number of the bigram $t_{i-1} \quad t_i$.

For example, the probability of getting a noun (NOUN) after a determiner (DET):

$$P(NOUN|DET) = \frac{C(DET, NOUN)}{C(DET)}$$

2.

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

where $C(t_i, w_i)$ is the number of w_i labeled as t_i.

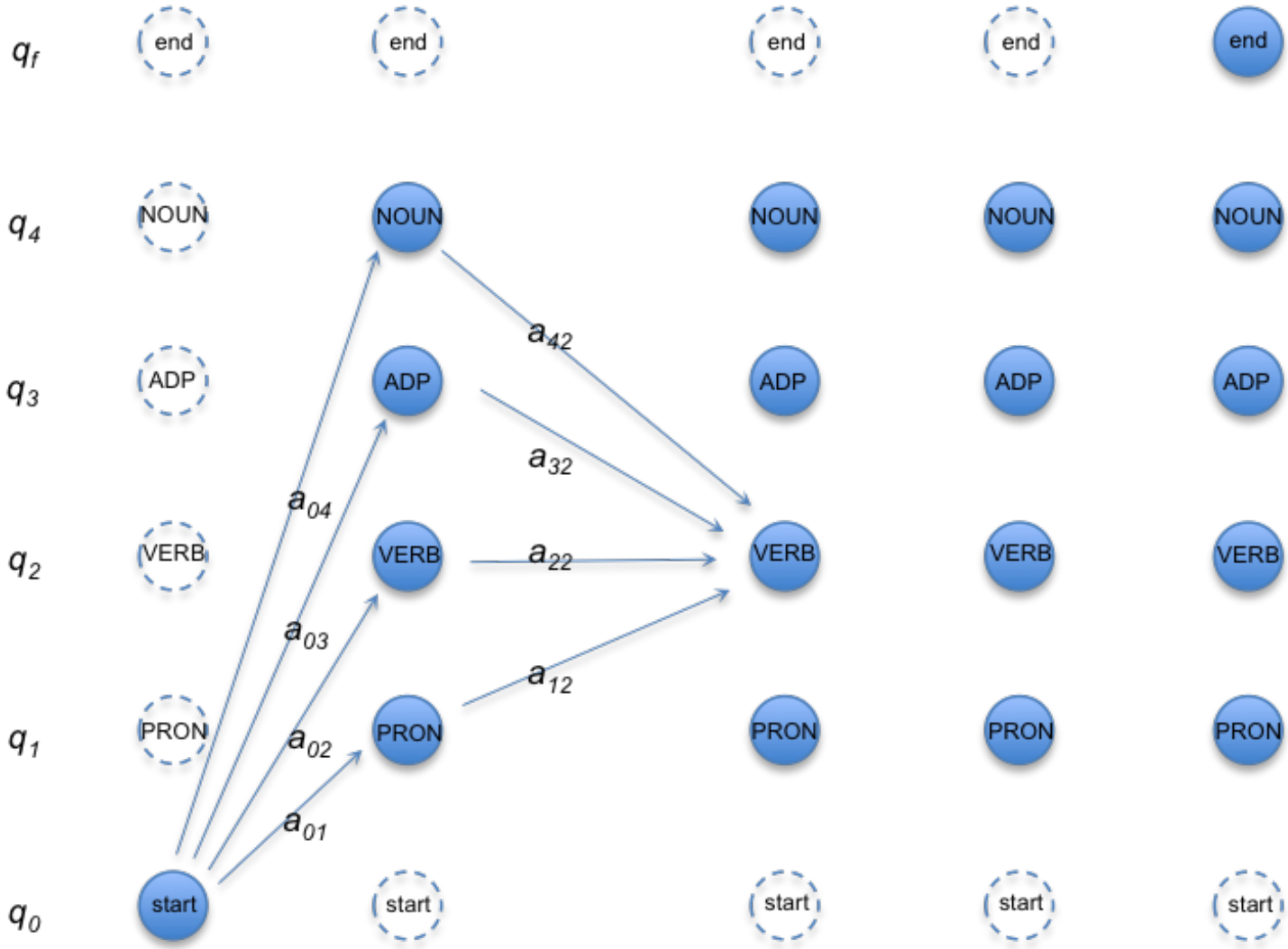
For example, the probability of getting a word *the* labeled as a determiner (DET):

$$P(the|DET) = \frac{C(DET, the)}{C(DET)}$$

HMM

$Q = q_1 q_2 ... q_N$	a set of N **states**
$A = a_{11} a_{12} ... a_{n1} ... anm$	a **transition probability matrix** A, each a_{ij} representing, the probability of moving from state i to state j. $\sum_{j=1}^{n} a_{ij} = 1 \ \ \forall_i$

Transition probabilities

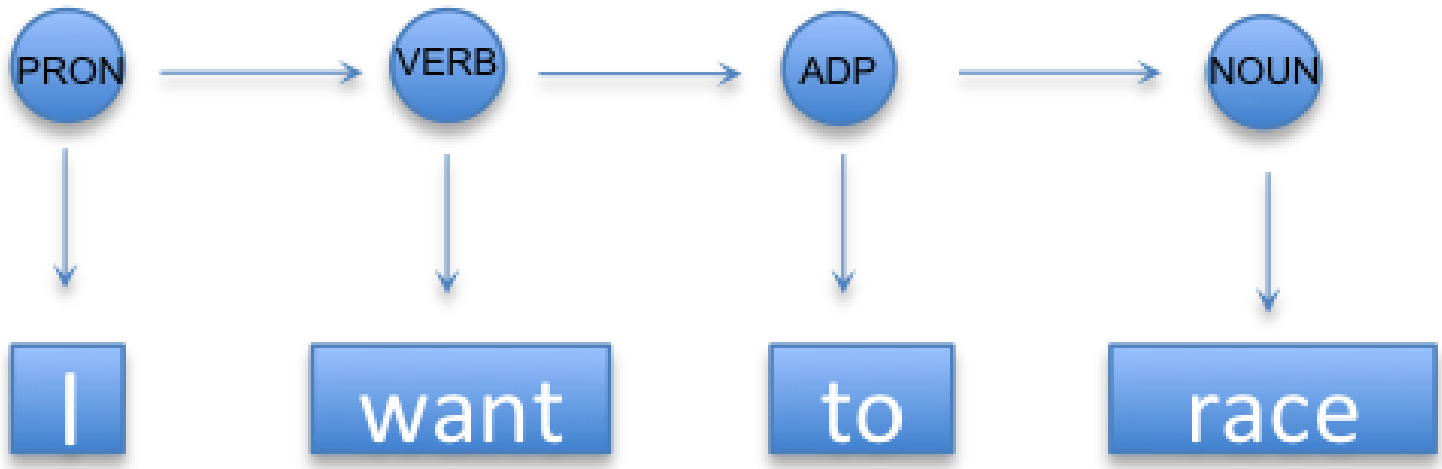


HMM (continued)

$O = o_1 o_2 ... o_T$ a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$

$B = b_i(o_t)$ A sequence of **observation likelihoods**, also called emission probabilities, each expression the probability of an observation o_t being generated from a state t

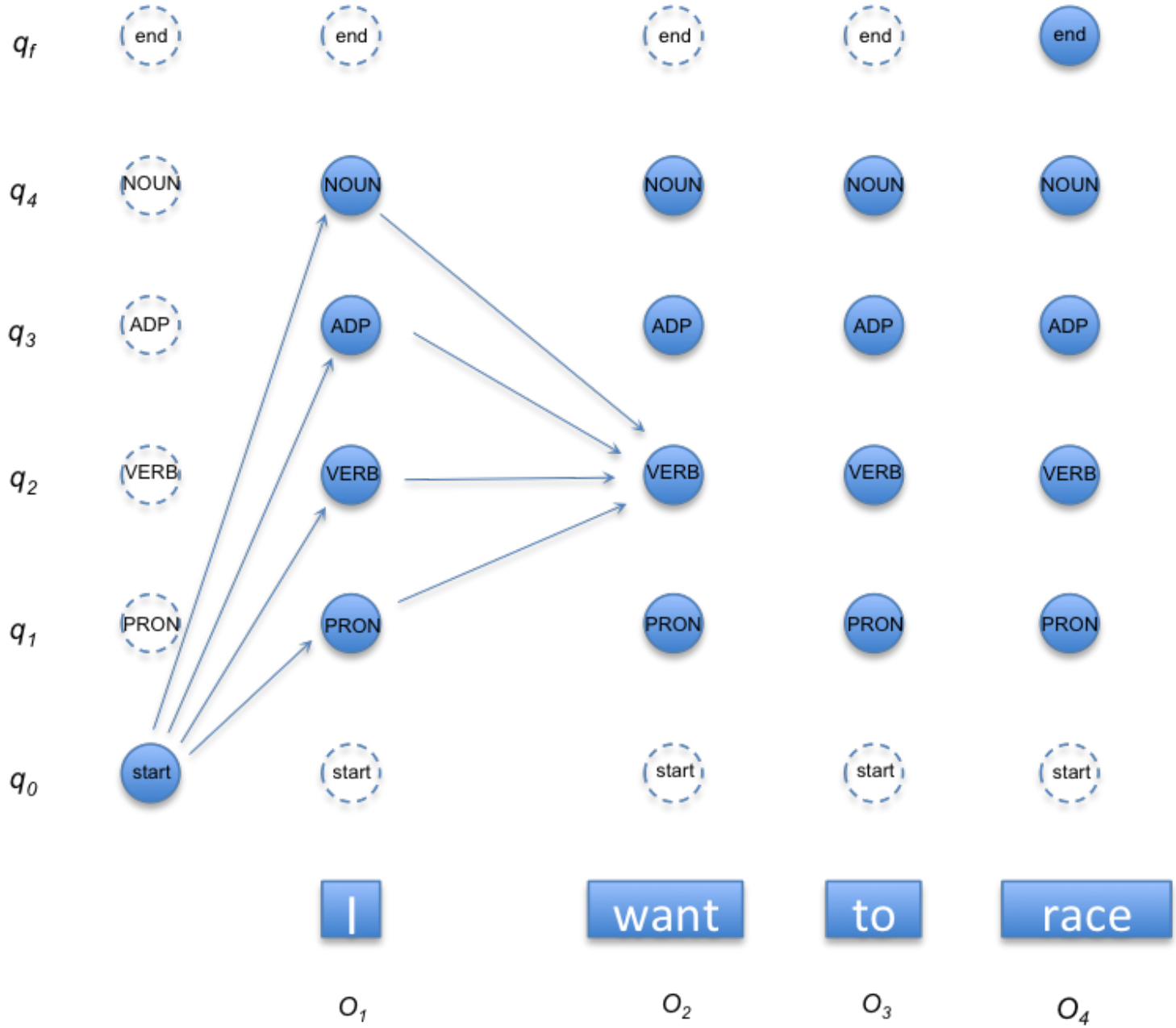
Observation probabilities



HMM (continued)

q_0, q_F a special **start state** and **end (final) state** that are not associated with observation, together with transition probabilities $a_{01}a_{02}...a_{0n}$ out of the start state and $a_{1F}a_{2F}...a_{nF}$ into the final state.

Markov trellis



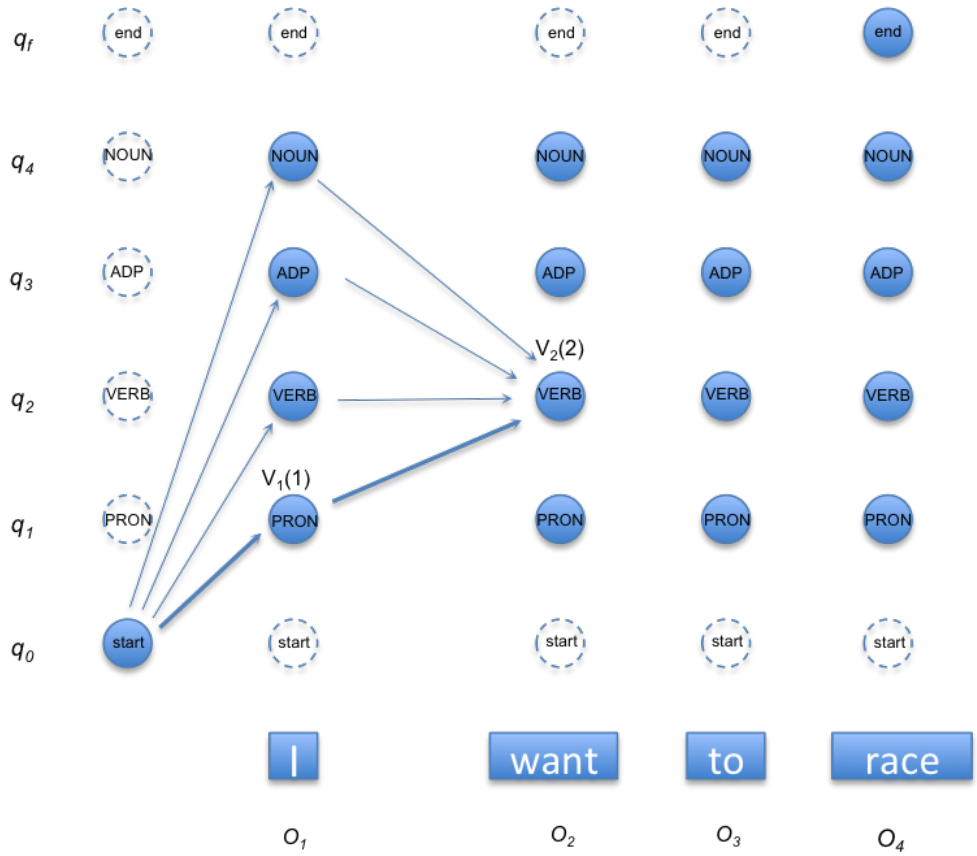
The most likely tag sequence

Transition probabilities

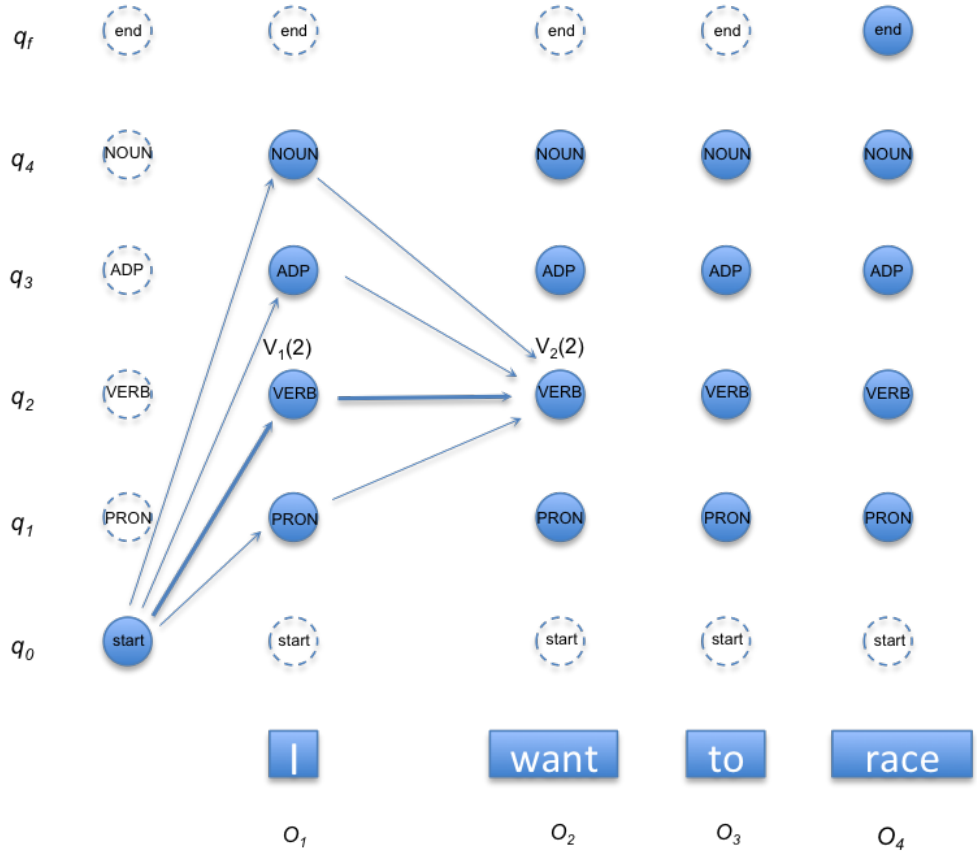
	VERB	**ADP**	**NOUN**	**PRON**
<s>	.019	.0043	.041	.067
VERB	.0038	.035	.047	.007
ADP	.83	0	.00047	0
NOUN	.004	.016	.087	.0045
PRON	.23	.00079	.0012	.00014

Observation probabilities

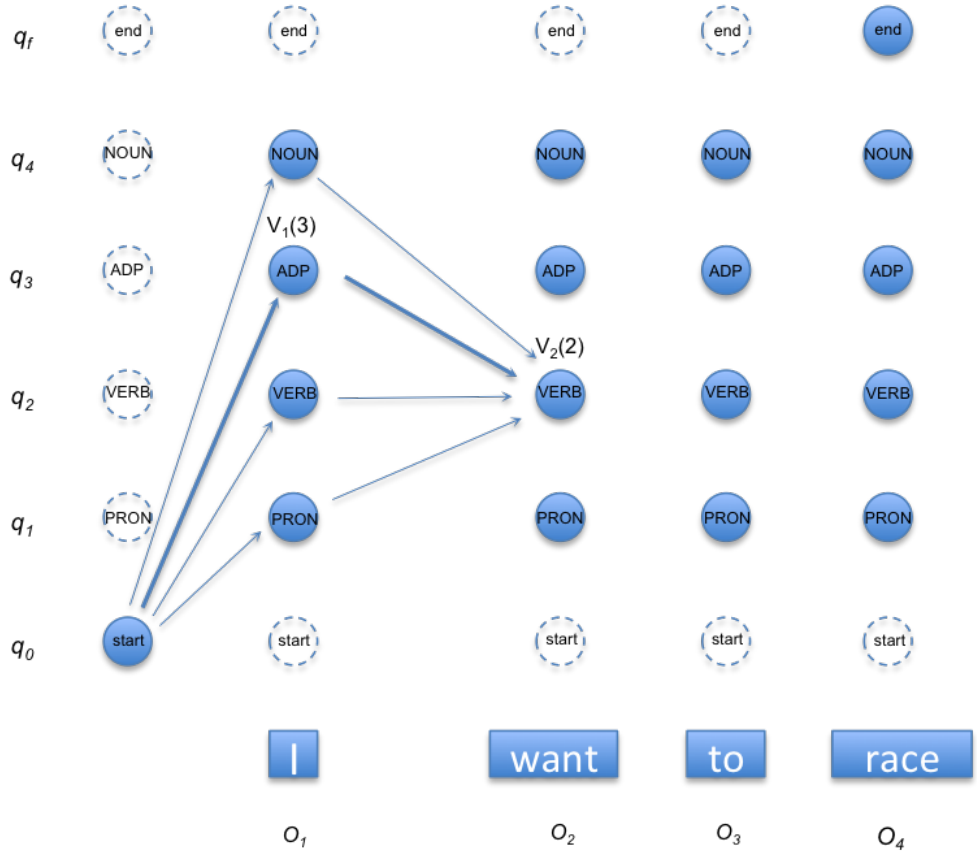
	I	**want**	**to**	**race**
VERB	0	.0093	0	.00012
ADP	0	0	.99	0
NOUN	0	.000054	0	.00057
PRON	.37	0	0	0



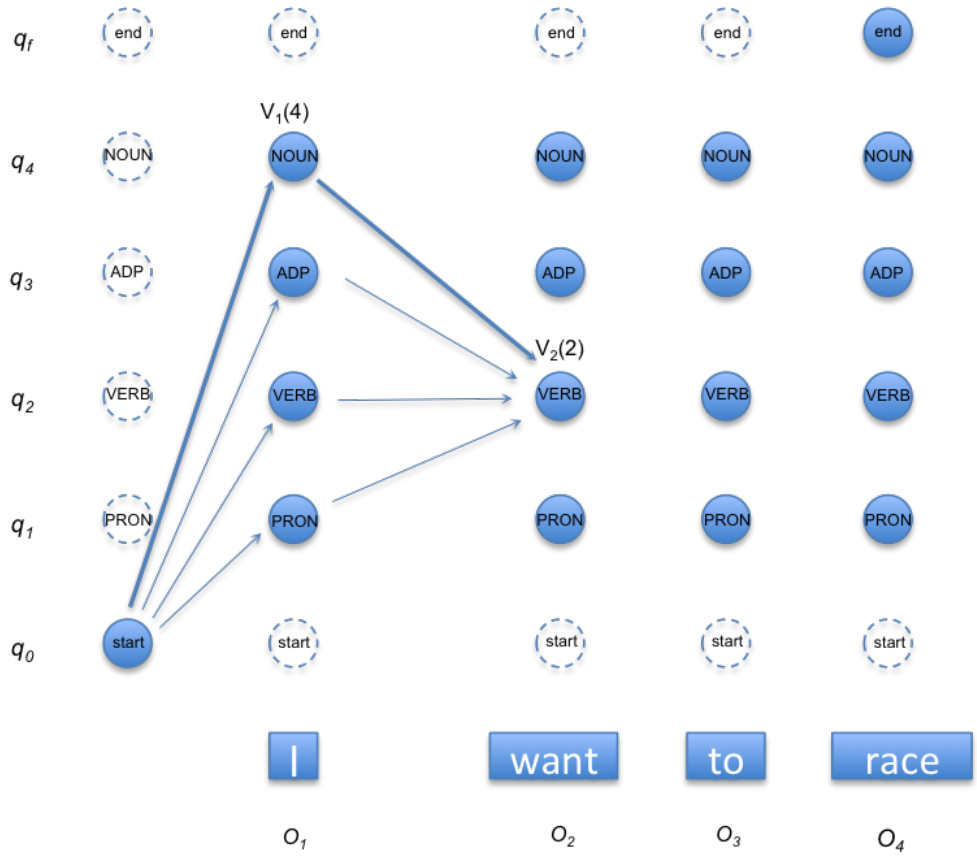
$$P(PRON| < s >) \times P(< s >)$$
$$v_1(1) = P(PRON| < s >) \times P(< s >) \times P(I|PRON)$$
$$v_1(1) \times P(VERB|PRON)$$



$$P(VERB| < s >) \times P(< s >)$$
$$v_1(2) = P(VERB| < s >) \times P(< s >) \times P(I|VERB)$$
$$v_1(2) \times P(VERB|VERB)$$



$$P(ADP| < s >) \times P(< s >)$$
$$v_1(3) = P(ADP| < s >) \times P(< s >) \times P(I|ADP)$$
$$v_1(3) \times P(VERB|ADP)$$



$$P(NOUN| < s >) \times P(< s >)$$
$$v_1(4) = P(NOUN| < s >) \times P(< s >) \times P(I|NOUN)$$
$$v_1(4) \times P(VERB|NOUN)$$

HMM in detail

Likelihood	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Decoding	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence O.
Learning	Given an observation sequence O and the set of states in the HMM, learn the HMM parameter A and B.

Likelihood: forward algorithm

Computing likelihood

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

For HMMs, each hidden state produces only a single observation $\rightarrow$ The sequence of hidden states and the sequence of observation have the same length.
For particular hidden state sequence $Q = q_1, q_2, ..., q_T$ and an observation sequence $O = o_1, o_2, ..., o_T$, the **likelihood of the observation sequence** is

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i) \tag{5}$$

Likelihood of the observation sequence, $P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$:

$$\begin{aligned} & P(\text{I want to race}|\text{PRON VERB ADP VERB}) \\ = \; & P(\text{I}|\text{PRON}) \times P(\text{want}|\text{VERB}) \times P(\text{to}|\text{ADP}) \times \\ & P(\text{race}|\text{VERB}) \end{aligned}$$

Problem: we don't actually know what the hidden state (POS label) sequence is.

Solution: we need to compute the probability of observation events by summing over all possible POS sequence, weighted by their probability.

$$P(O) = \sum_{Q} P(O, Q) \tag{6}$$

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

$$
\begin{aligned}
& P(\text{I want to race, PRON VERB ADP VERB}) \\
= \; & P(\text{I}|\text{PRON}) \times P(\text{want}|\text{VERB}) \times P(\text{to}|\text{ADP}) \times \\
& P(\text{race}|\text{VERB}) \\
& \times P(\text{PRON}|\text{Start}) \times P(\text{VERB}|\text{PRON}) \times P(\text{ADP}|\text{VERB}) \times \\
& P(\text{VERB}|\text{ADP}) \times P(\text{End}|\text{VERB})
\end{aligned}
$$

$$P(O) = \sum_Q P(O, Q) = \sum_Q P(O|Q)P(Q)$$

$P(\text{I want to race}) =$
$P(\text{I want to race, PRON PRON PRON PRON}) +$
$P(\text{I want to race, PRON PRON PRON VERB}) +$
$P(\text{I want to race, PRON PRON VERB PRON}) +$
$P(\text{I want to race, PRON VERB PRON PRON}) + ...$

For an HMM with N hidden states and an observation sequence T observations, there are ??? possible hidden sequences.

$$P(O) = \sum_Q P(O, Q) = \sum_Q P(O|Q)P(Q)$$

$P(\text{I want to race}) =$
$P(\text{I want to race, PRON PRON PRON PRON}) +$
$P(\text{I want to race, PRON PRON PRON VERB}) +$
$P(\text{I want to race, PRON PRON VERB PRON}) +$
$P(\text{I want to race, PRON VERB PRON PRON}) + ...$

For an HMM with N hidden states and an observation sequence T observations, there are N^T possible hidden sequences.

Forward algorithm

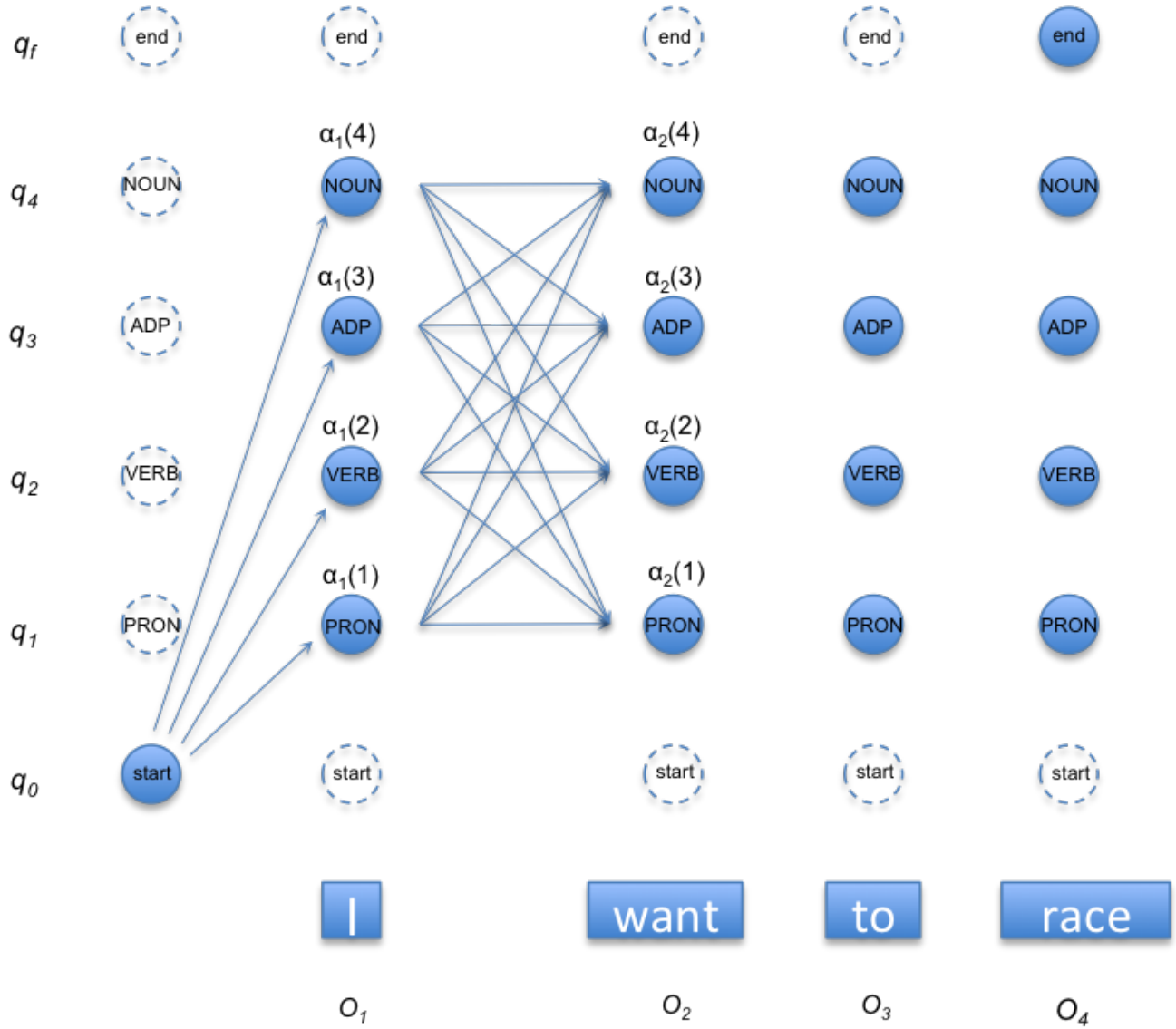
Instead of exploring N^T possible hidden sequences, we use an efficient algorithm called the **forward algorithm** with $O(N^2T)$.

$\alpha_t(j)$ represents the probability of being in state j after seeing the first t observation., given the automata λ.

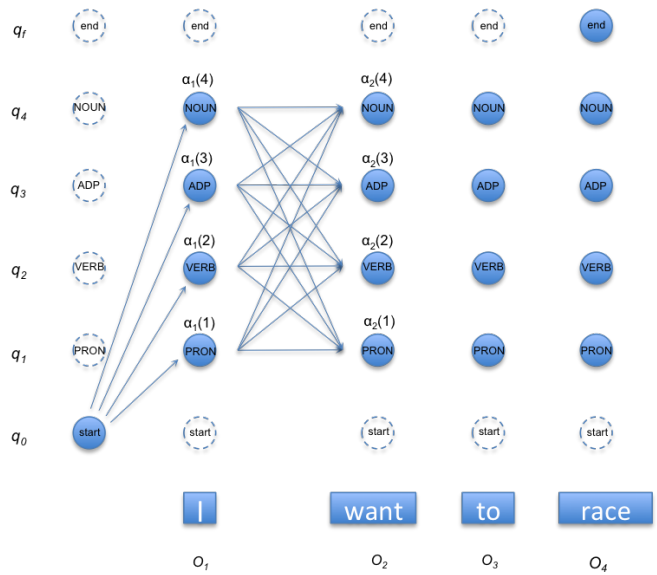
$\alpha_t(j) = P(o_1, o_2, ...o_t, q_t = j|\lambda)$
where $q_t = j$ is the probability that the tth state in the sequence of state is state j.

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) \tag{7}$$

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


q_f
q_4
q_3
q_2
q_1
q_0
end
NOUN
ADP
VERB
PRON
start
$\alpha_1(4)$
$\alpha_1(3)$
$\alpha_1(2)$
$\alpha_1(1)$
$\alpha_2(4)$
$\alpha_2(3)$
$\alpha_2(2)$
$\alpha_2(1)$
I
want
to
race
O_1
O_2
O_3
O_4

$$
\begin{aligned}
\alpha_1(1) = &\quad P(\text{PRON}|\text{Start}) \times P(\text{I}|\text{PRON}) \\
\alpha_1(2) = &\quad P(\text{VERB}|\text{Start}) \times P(\text{I}|\text{VERB}) \\
\alpha_1(3) = &\quad P(\text{ADP}|\text{Start}) \times P(\text{I}|\text{ADP}) \\
\alpha_1(2) = &\quad P(\text{NOUN}|\text{Start}) \times P(\text{I}|\text{NOUN})
\end{aligned}
$$



$$
\begin{aligned}
\alpha_2(1) = \quad & \alpha_1(1) \times P(\text{PRON}|\text{PRON}) \times P(\text{want}|\text{PRON}) \\
& + \alpha_1(2) \times P(\text{PRON}|\text{VERB}) \times P(\text{want}|\text{PRON}) \\
& + \alpha_1(3) \times P(\text{PRON}|\text{ADP}) \times P(\text{want}|\text{PRON}) \\
& + \alpha_1(4) \times P(\text{PRON}|\text{NOUN}) \times P(\text{want}|\text{PRON}) \\
= \quad & \sum_{i=1}^{4} \alpha_1(i) \times a_{i2} \times P(\text{want}|\text{PRON})
\end{aligned}
$$

$$
\alpha_2(2) = \quad ...
$$

$$
\alpha_2(3) = \quad ...
$$

$$
\alpha_2(4) = \quad ...
$$

1. initialization:
   - $\alpha_1(j) = a_{0j} b_j(o_1) \qquad 1 \leq j \leq N$
2. recursion:
   - $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) \qquad 1 \leq j \leq N, 1 < t \leq T,$
3. termination:
   - $\alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$
   - $\alpha_T(q_F) \rightarrow P(O|\lambda)$, likelihood.

Forward algorithm

function FORWARD(T, N) **return** *forward-probability*
T: *observation* of length T
N: *state-graph* of length N
  create a probability matrix $foward[N+2, T]$
  for each state s **from** 1 **to** N **do**
    $forward[s, 1] \leftarrow a_{0,s} \times b_s(o_1)$
  for each time step t **from** 2 **to** T **do**
    for each state s **from** 1 **to** N **do**

$$forward[s, t] \leftarrow \sum_{s'=1}^{N} forward[s', t-1] \times a_{s',s} \times b_s(o_1)$$

$$forward[q_F, T] \leftarrow \sum_{s=1}^{N} forward[s, T] \times a_{s,q_F}$$

return $forward[q_F, T]$

Decoding: Viterbi algorithm

For HMM POS tagging, the task of decoding is to find the best hidden POS label sequence.

Decoding

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

Viterbi algorithm have become standard terms for the application of dynamic programming algorithms to maximization problems involving probabilities.

Dynamic programming is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions – ideally, using a memory-based data structure.

[Souce: Wikipedia, accessed on September 20, 2016]

$$v_t(j) = \max_{q_0, q_1, \ldots, q_{t-1}} P(q_0, q_1, ..., q_{t-1}, o_1, o_2, ..., o_t, q_t = j|\lambda) \quad (8)$$

Note that we represent **the most probable path** by taking the maximum over all possible previous state sequence $\max_{q_0, q_1, \ldots, q_{t-1}}$.

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \quad (9)$$

likelihood

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

decoding

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

$v_{t-1}(i)$	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

1. initialization:
   - $v_1(j) = a_{0j} b_j(o_1) \qquad 1 \leq j \leq N$
   - $bt_1(j) = 0$
2. recursion:
   - $\alpha_t(j) = \sum_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \qquad 1 \leq j \leq N, 1 < t \leq T,$
   - $bt_t(j) = \underset{i=1}{\overset{N}{\operatorname{argmax}}}\, v_{t-1}(i) a_{ij} b_j(o_t) \qquad 1 \leq j \leq N, 1 < t \leq T,$
3. termination:
   - The best score: $v_T(q_F) = \max_{i=1}^{N} v_T(i) a_{iF}$
   - The start of backtrace: $bt_T(q_F) = \underset{i=1}{\overset{N}{\operatorname{argmax}}}\, v_T(i) a_{iF}$

Viterbi algorithm

function VITERBI(T, N) **return** *best-path*
T: *observation* of length T
N: *state-graph* of length N

%% initialization step
....
%% recursion step
....
%% termination step
....

Viterbi algorithm (continued)

function VITERBI(T, N) **return** *best-path*
T: *observation* of length T
N: *state-graph* of length N

create a path probability matrix $viterbi[N+2, T]$
%% initialization step
for each state s **from** 1 **to** N **do**
$viterbi[s, 1] \leftarrow a_{0,s} \times b_s(o_1)$
$backpointer[s, 1] \leftarrow 0$

Viterbi algorithm (continued)

function VITERBI(T, N) **return** *best-path*

%% initialization step

....

%% recursion step

for each time step t **from** 2 **to** T **do**

for each state s **from** 1 **to** N **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] \times a_{s',s} \times b_s(o_1)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] \times a_{s',s}$$

Viterbi algorithm (continued)

function VITERBI(T, N) **return** *best-path*

%% initialization step

....

%% recursion step

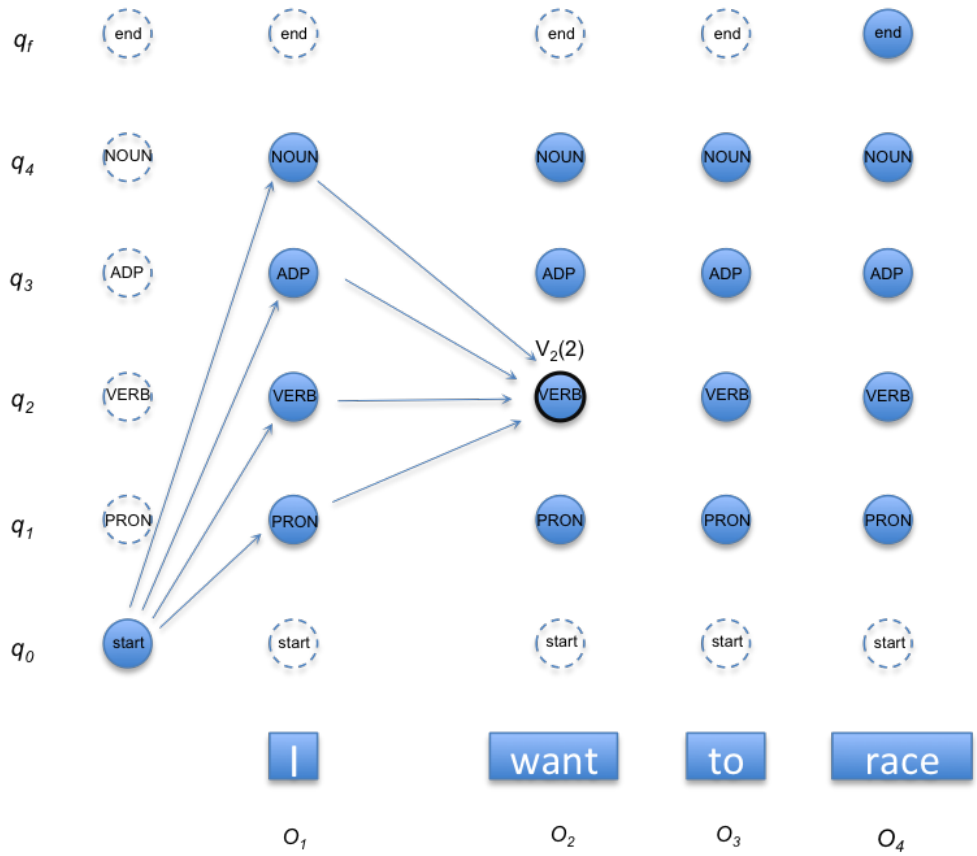
....

%% termination step

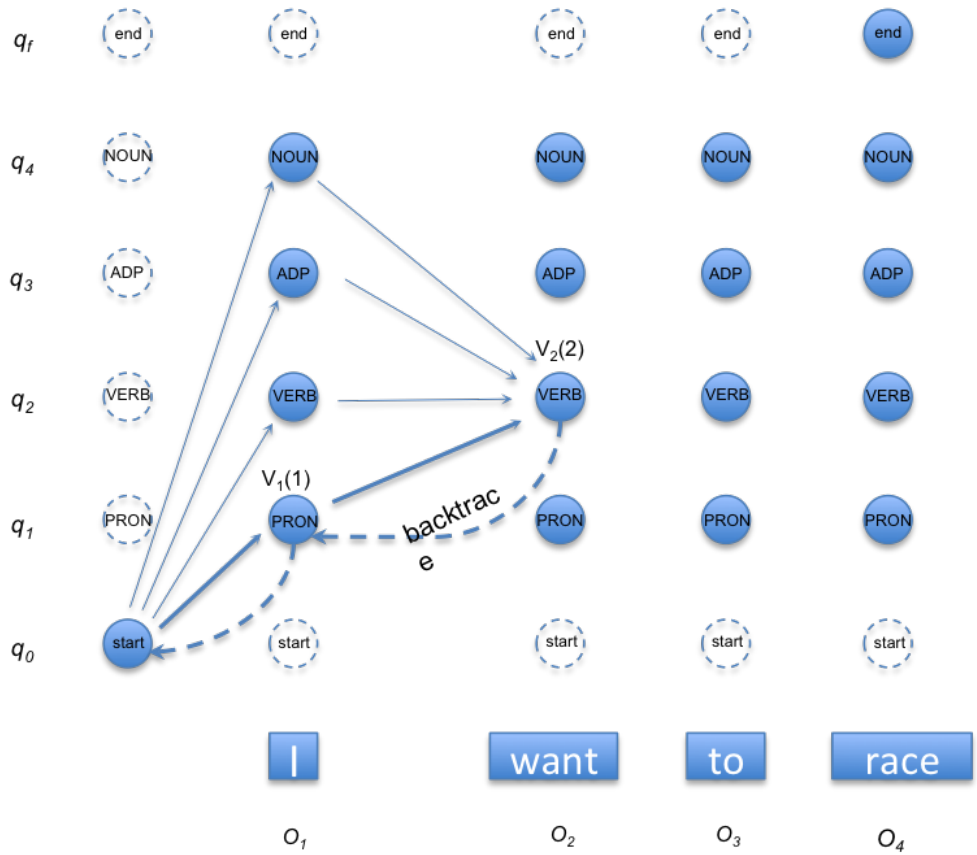
$viterbi[q_F, T] \leftarrow \max_{s=1}^{N} viterbi[s, T] \times a_{s,q_F}$

$backpointer[q_F, T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s, T] \times a_{s,q_F}$

return the backtrace path by following backpointers to state back in time from $backpointer[q_F, T]$



$$v_2(2) = \max(v_1(1) \times a_{12}, v_1(2) \times a_{22}, v_1(3) \times a_{32}, v_1(4) \times a_{42}) \times P(want|VERB)$$



$$backpointer[s,t] \leftarrow$$
$$\text{argmax}(v_1(1) \times a_{12}, v_1(2) \times a_{22}, v_1(3) \times a_{32}, v_1(4) \times a_{42})$$

HMM Training: learning the parameters of the HMM

Learning

Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameter A and B.

The standard algorithm for HMM training is the **forward-backward**, or **Baum-Welch** algorithm, a special case of the **Expectation-Maximization** (EM) algorithm.

The EM algorithm will let us train both the transition probabilities A and the emission (observation) probabilities B of the HMM.

Maximum likelihood estimate of the probability a_{ij}

$$a_{ij} = \frac{C(i \to j)}{\sum_{q \in Q} C(i \to q)} \tag{10}$$

$$=$$

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})} \tag{11}$$

$$a_{ij} = \frac{C(i \rightarrow j)}{\sum_{q \in Q} C(i \rightarrow q)} \tag{12}$$

However, for an HMM, we **cannot compute** these count directly from an observation sequence since we don't know which path of states was taken through the machine for a given input.

1. iteratively estimate the counts
   - start with an estimate for the transition and observation probabilities and then use these estimated probabilities to derive better and better probabilities
2. get estimated probabilities by computing the forward probability for an observation and then dividing that probability mass among all the different paths that contributed to this forward probability.

Backward algorithm

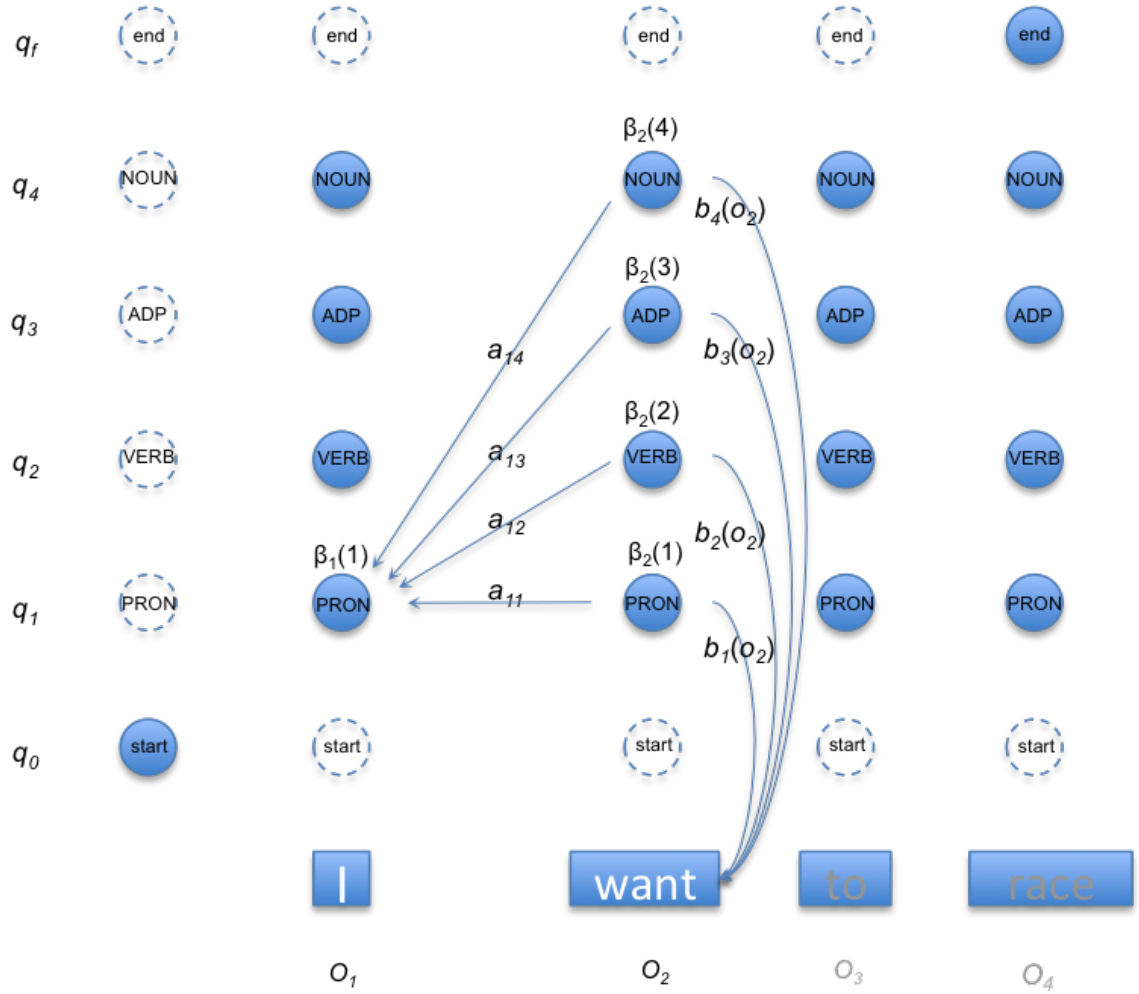
The backward probability β is the probability of seeing the observation from time $t+1$ to the end, given that we are in state i at time t (and given the automaton λ).

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ...o_T | q_t = i, \lambda) \tag{13}$$

It is computed inductively in a similar manner to the forward algorithm.

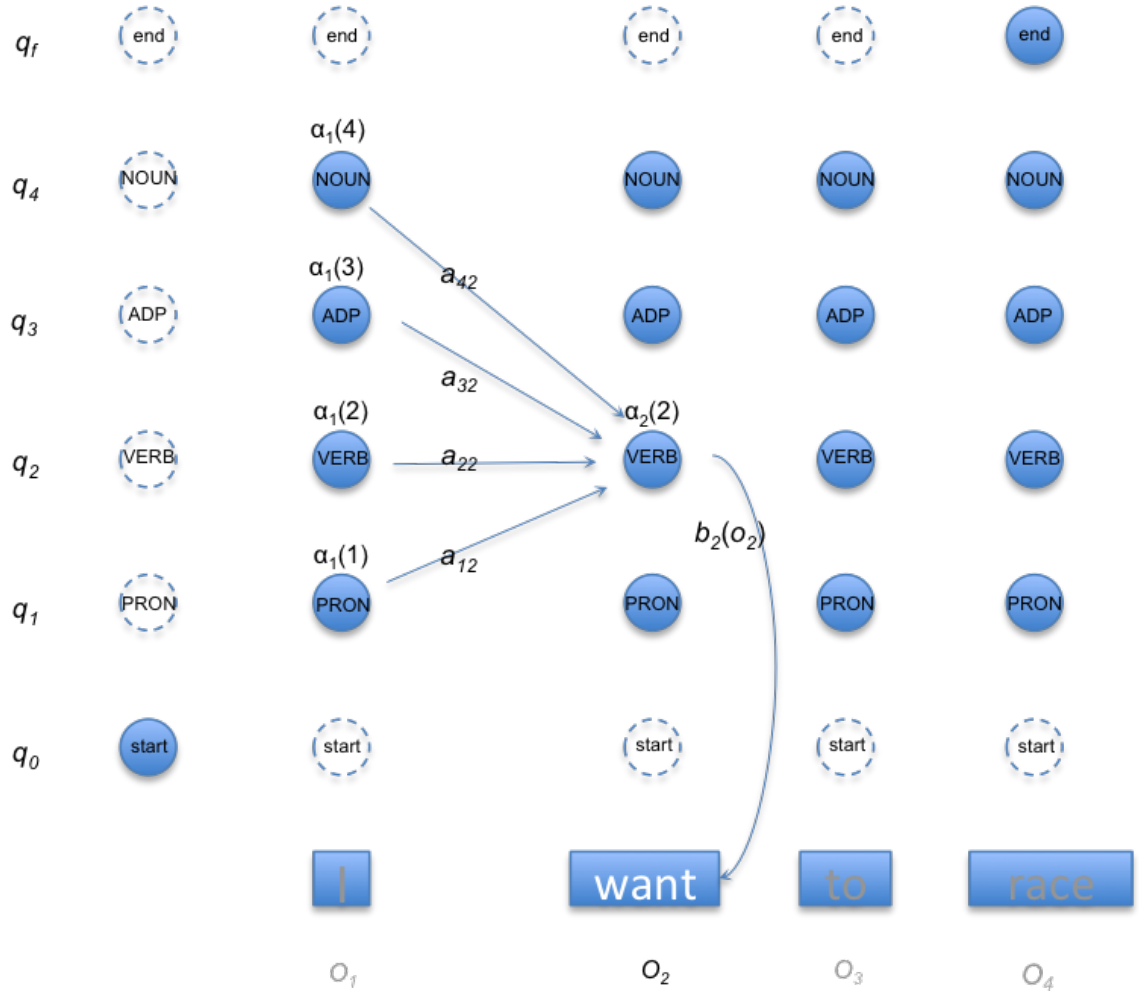
1. initialization:
   - $\beta_T(i) = a_{i,F}, \qquad 1 \leq i \leq N$
2. recursion:
   - $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_t + 1)\beta_{t+1}(j), \qquad 1 \leq i \leq N,\ 1 \leq t < T$
3. termination:
   - $\beta_1(q_0) = \sum_{j=1}^{N} a_{0j} b_j(o1)\beta_1(j), \qquad 1 \leq i \leq N,\ 1 \leq t < T$

Backward



$$\beta_1(1) = \sum_{j=1}^{4} \beta_2(j) a_{1j} b_j(o_2) \quad \rightarrow \quad \beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1})$$

Forward



$$\alpha_2(2) = \sum_{i=1}^{4} \alpha_1 a_{i2} b_2(o_2) \quad \rightarrow \quad \alpha_t(i) = \sum_{i=1}^{N} \alpha_{t-1} a_{ij} b_j(o_t)$$

Transition probability

$$a_{ij} = \frac{C(i \to j)}{\sum_{q \in Q} C(i \to q)} \tag{14}$$

However, we don't know the actual path for the HMM.

$$\hat{a}_{ij} = \frac{\text{expected number of transition from state } i \text{ to state } j}{\text{expected number of transition from state } i} \tag{15}$$

Assume we had some estimate of the probability that a given transition $i \rightarrow j$ was taken at a particular point in time t in the observation sequence.

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j | O, \lambda) \tag{16}$$

Recall $P(X|Y,Z) = \frac{P(X,Y|Z)}{P(Y|Z)}$

$$\xi_t(i,j) = \frac{P(q_t = i, q_{t+1} = j, O|\lambda)}{P(O|\lambda)} \tag{17}$$

$\xi_t(i, j)$

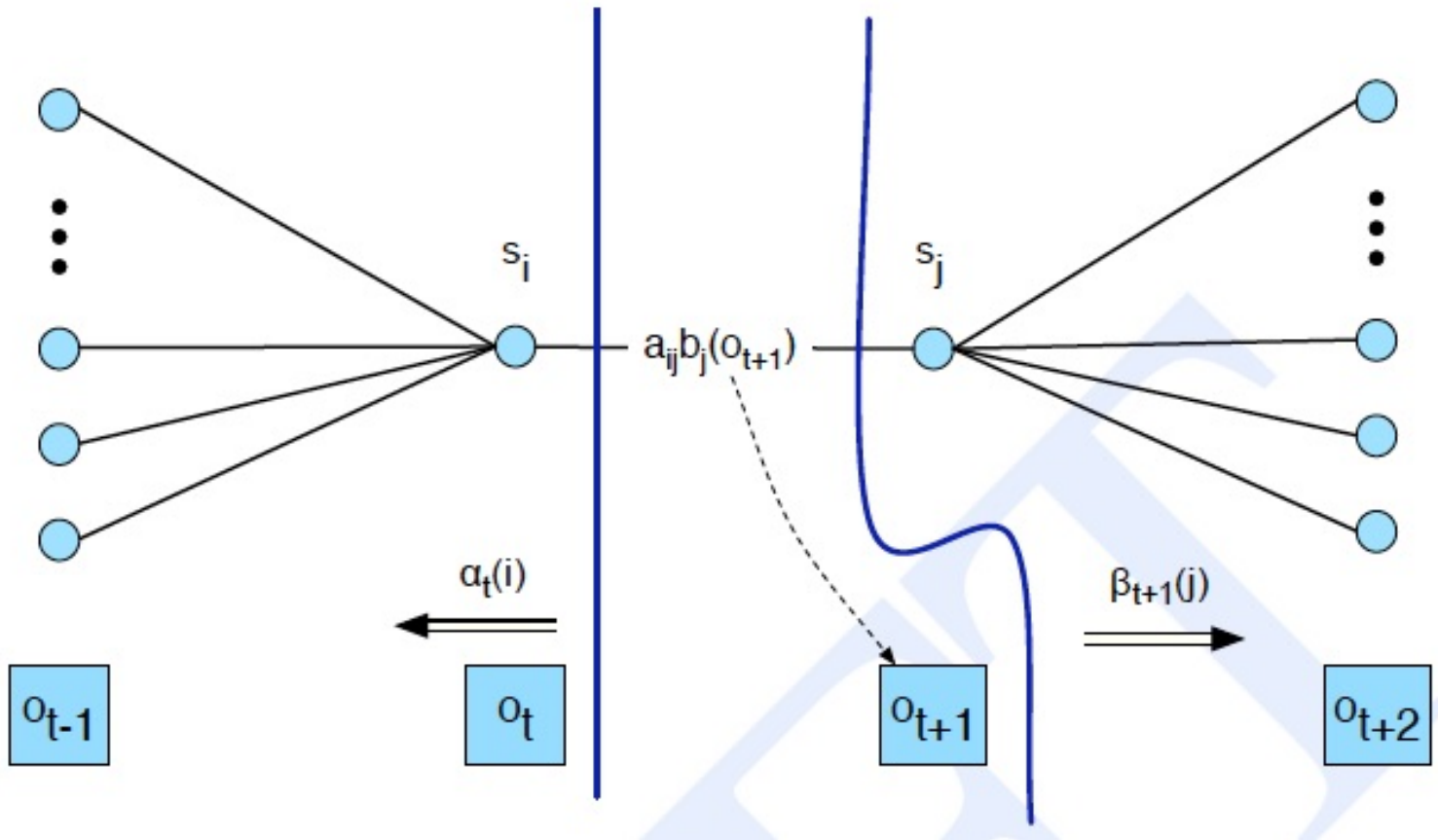
nominator the joint probability of being in state i at time t and state j at time $t + 1$

- To produce $P(q_t = i, q_{t+1} = j, O|\lambda)$, we combine α and β probabilities, the transition probability a_{ij}, and the observation probability $b_j(o_{t+1})$ (Rabiner, 1989).
- $\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)$

denominator likelihood

- $P(O|\lambda) = \alpha_T(q_F) = \beta_T(q_0)$

Joint probability of being in state i at time t and state j at time $t+1$



$$\begin{aligned}\xi_t(i,j) &= \frac{P(q_t = i, q_{t+1} = j, O|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)}\end{aligned}$$

To expected number of transitions from state i to state j is then the sum over all t of ξ.

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)} \tag{18}$$

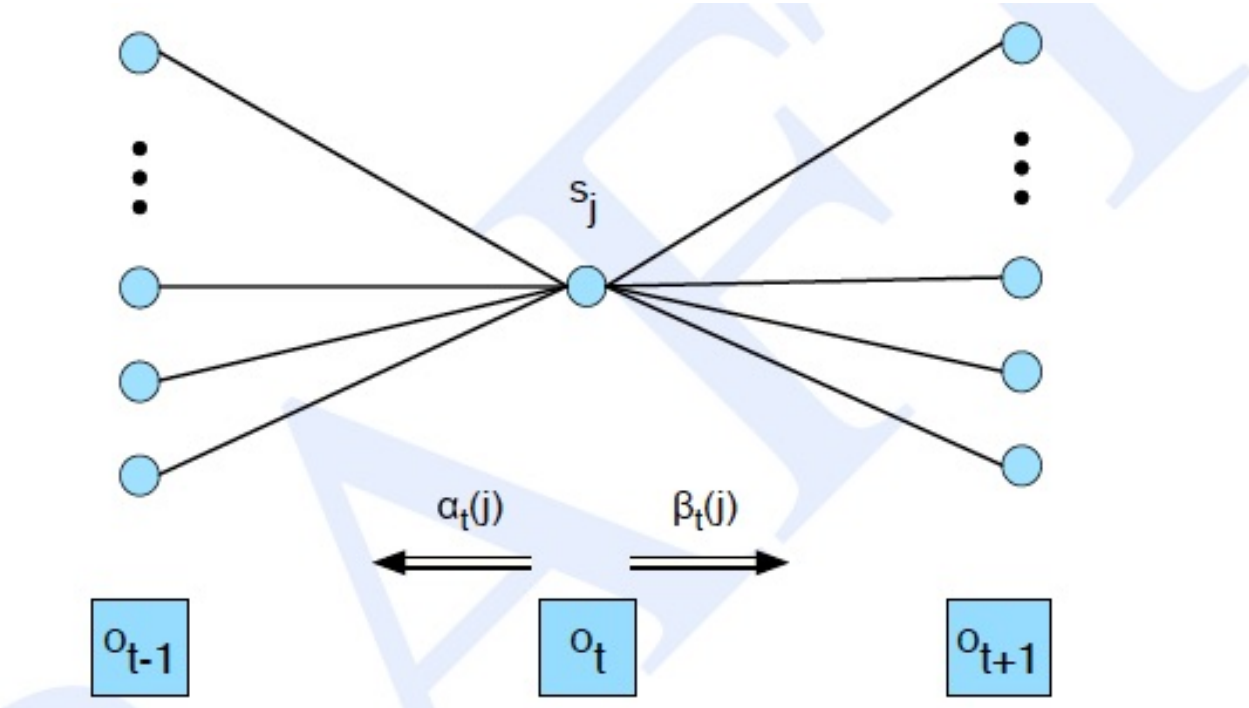
Recompute observation probability:

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} \tag{19}$$

We will need to know the probability of being in state j at time t, which we will call $\gamma_t(j)$:

$$\gamma_t(j) = P(q_t = j|O, \lambda) \tag{20}$$

$$\gamma_t(j) = \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)} \tag{21}$$



The computation $\gamma_t(t)$, the probability of being state j at time t. Note that γ is a degenerate case of ξ (Rabiner, 1989).

$$\gamma_t(j) = \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)}$$

For the numerator, we sum $\gamma_t(j)$ for all time steps t in which the observation o_t is the symbol v_k that we are interested in. For the denominator, we sum $\gamma_t(j)$ over all time steps t.

$$\hat{b}_j(v_k) = \frac{\sum_{t=1, O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{22}$$

We now have ways to *re-estimate* the transition A and observation B probabilities from an observation sequence O, assuming that we already have a previous estimate A and B.

The forward-backward algorithm has two steps: the expectation step (E-step) and the maximization step (M-step):

1. In the E-step, we compute the expected state occupancy count γ and the expected state transition count ξ from the earlier A and B probabilities.
2. In the M-step, we use γ and ξ to recompute A and B probabilities.

Forward-backward algorithm

function FORWARD-BACKWARD(T, V, Q) **return** *HMM=(A,B)*
T: *observation* of length
V: *output vocabulary*
Q: *hidden state set*

initialize A and B
iterate until convergence
E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)}, \quad \forall_{t \text{ and } j}$$

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)}, \quad \forall_{t,\ i, \text{ and } j}$$

M-step

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

return A, B