CS 4705 Hidden Markov Models

Slides adapted from Dan Jurafsky, and James Martin

Announcements and Questions

- HW1: Determine whether unigrams, bigrams, trigrams or some combination of the three works best, experiment with ML parameters (e.g., kernel and C for SVM). Then do feature selection and additional features on the result.
- Keep in mind that you can have lower accuracy without large penalty in points.
- Final exam: tentatively scheduled for 12/21 but will be finalized in Nov by registrar. We will have the exam on the exam date.
- Class electronic policy: no open laptops in class.

POS tagging as a sequence classification task

- We are given a sentence (an “observation” or “sequence of observations”)
  - Secretariat is expected to race tomorrow
- What is the best sequence of tags which corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Choose the tag sequence which is most probable given the observation sequence of n words w1…wn.

Getting to HMM

- Out of all sequences of n tags $t_1 \ldots t_n$ want 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"
- Argmax_x f(x) means "the x such that f(x) is maximized"

Getting to HMM

- This equation is guaranteed to give us the best tag sequence

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

- 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

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

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

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

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

Two kinds of probabilities (1)

- Tag transition probabilities $p(t_i|t_{i-1})$
  - Determiners likely to precede adjs and nouns
    - That/DT flight/NN
    - The/DT yellow/JJ hat/NN
    - So we expect P(NN|DT) and P(JJ|DT) to be high
    - But P(DT|JJ) to be low
  - Compute P(NN|DT) by counting in a labeled corpus:

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

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Two kinds of probabilities (2)

- Word likelihood probabilities $p(w_i|t_i)$
  - VBZ (3sg Pres verb) likely to be “is”
  - Compute P(is|VBZ) by counting in a labeled corpus:

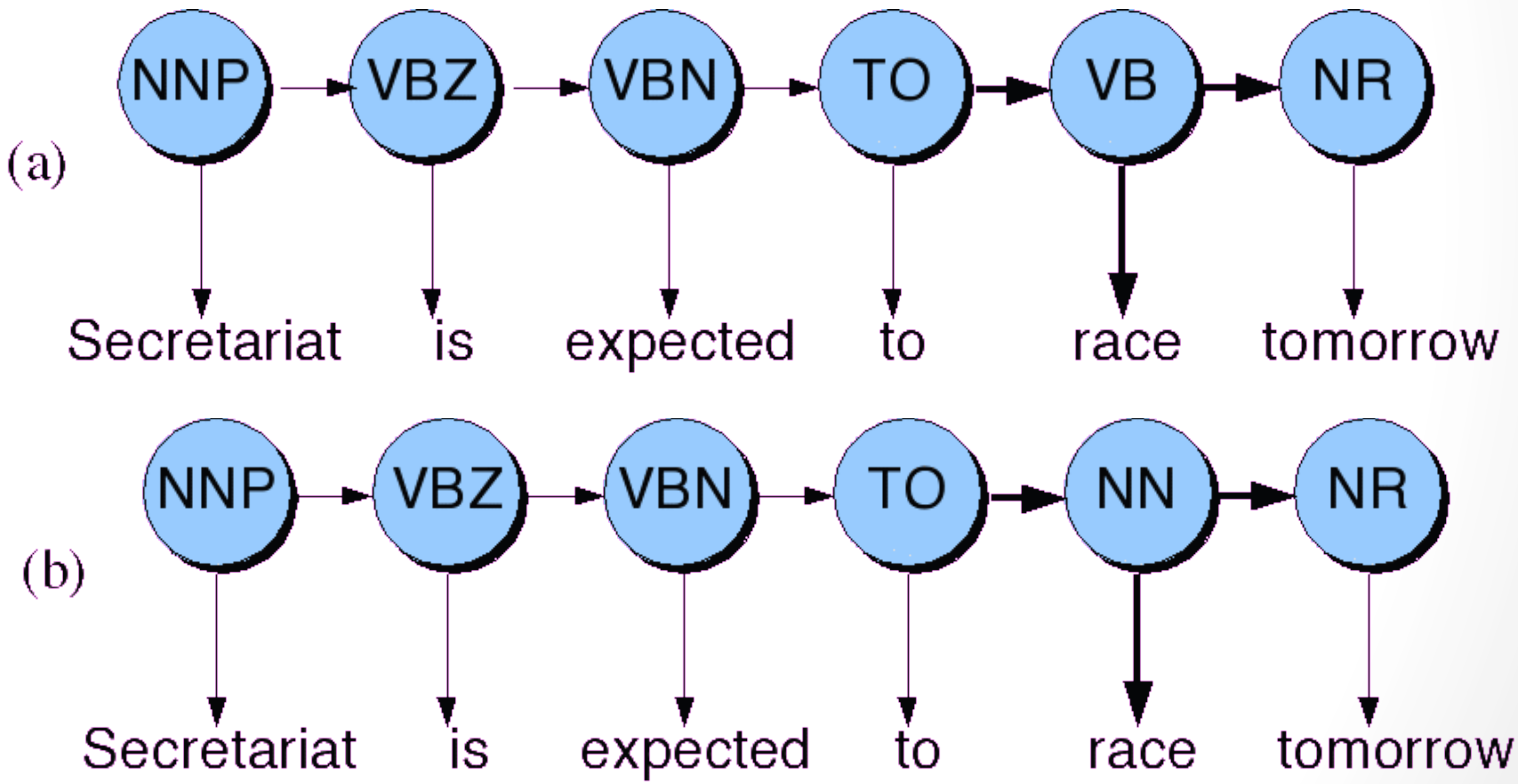
$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

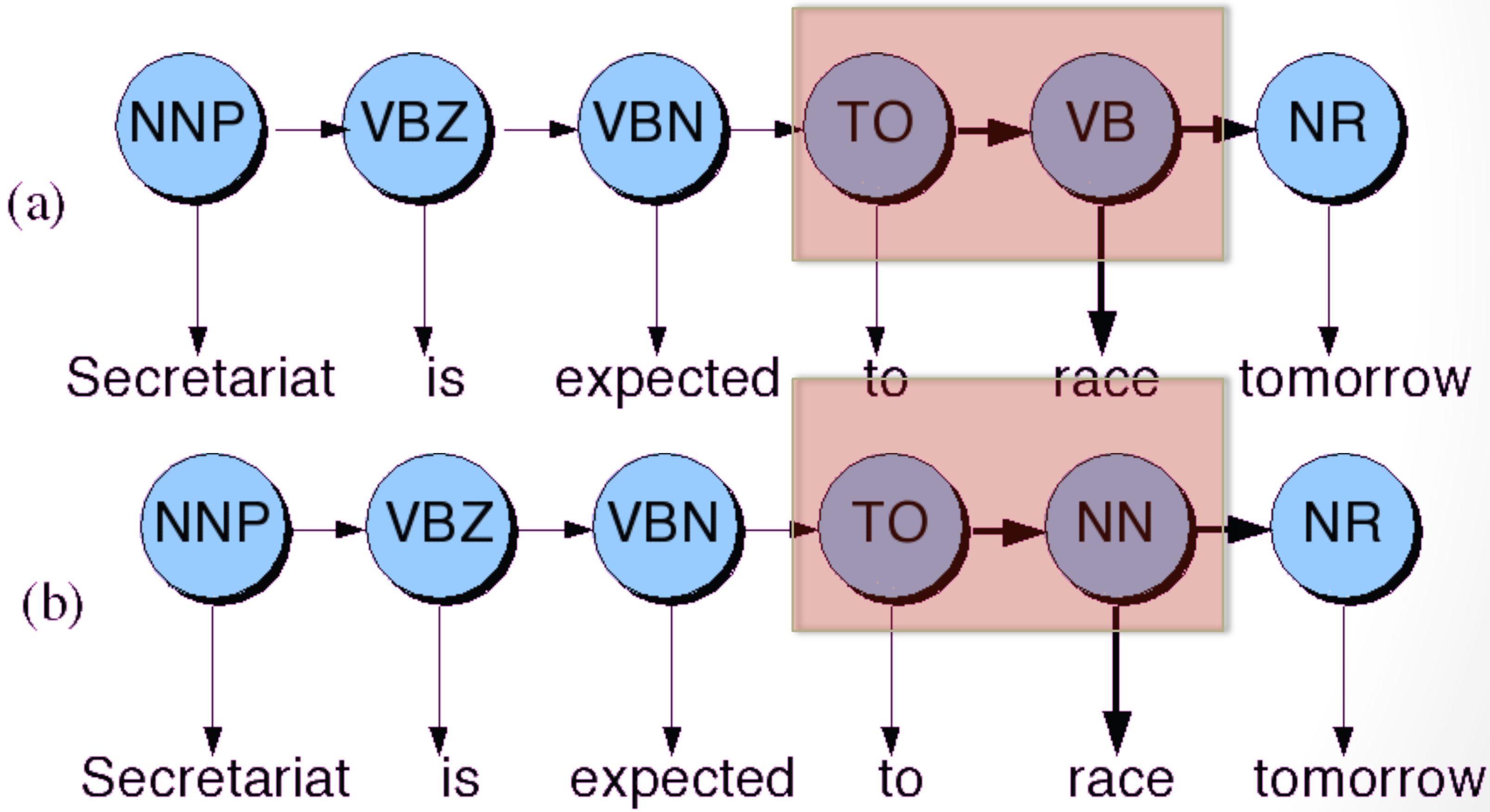
An Example: the verb “race”

- Secretariat/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?

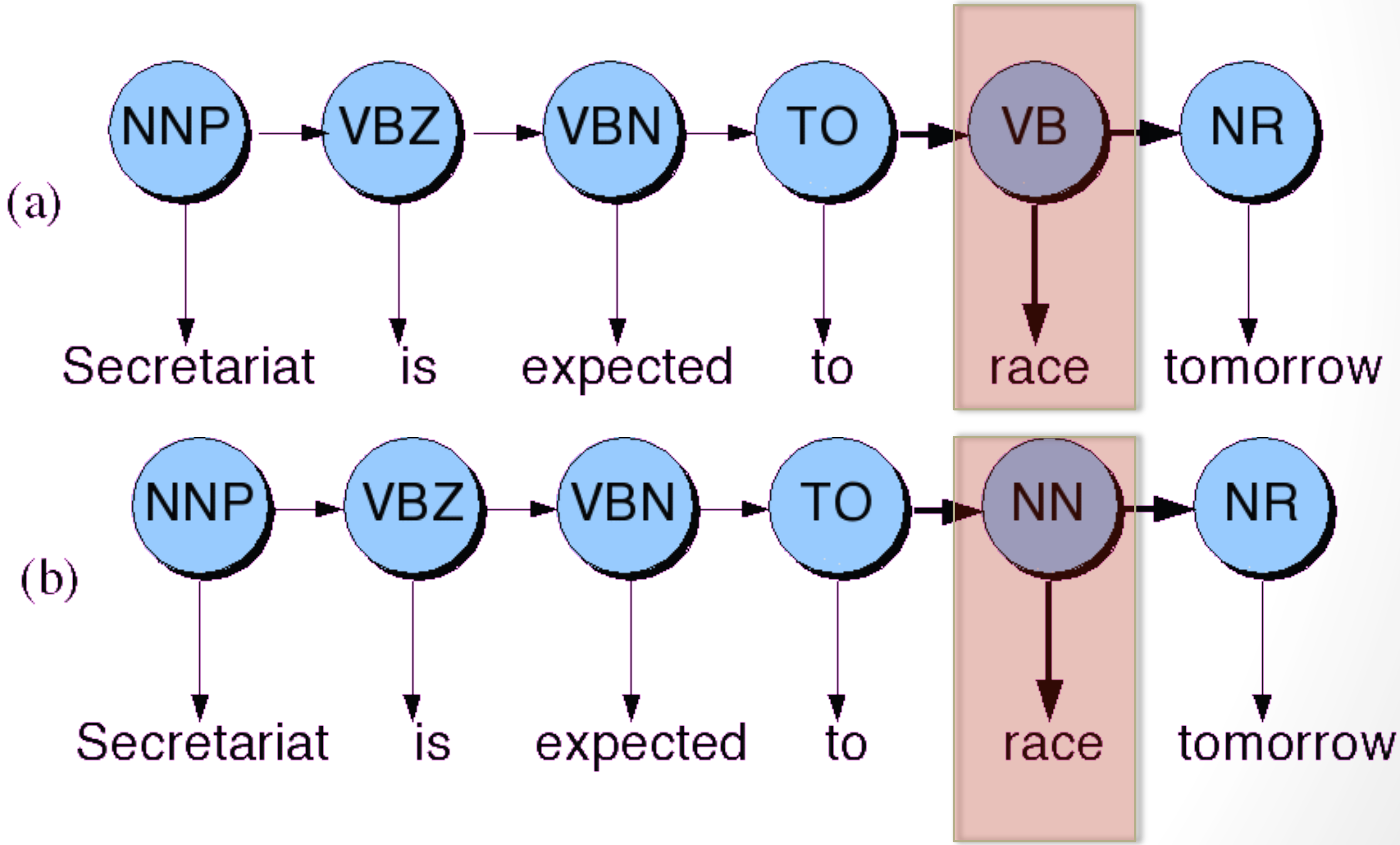
Disambiguating “race”



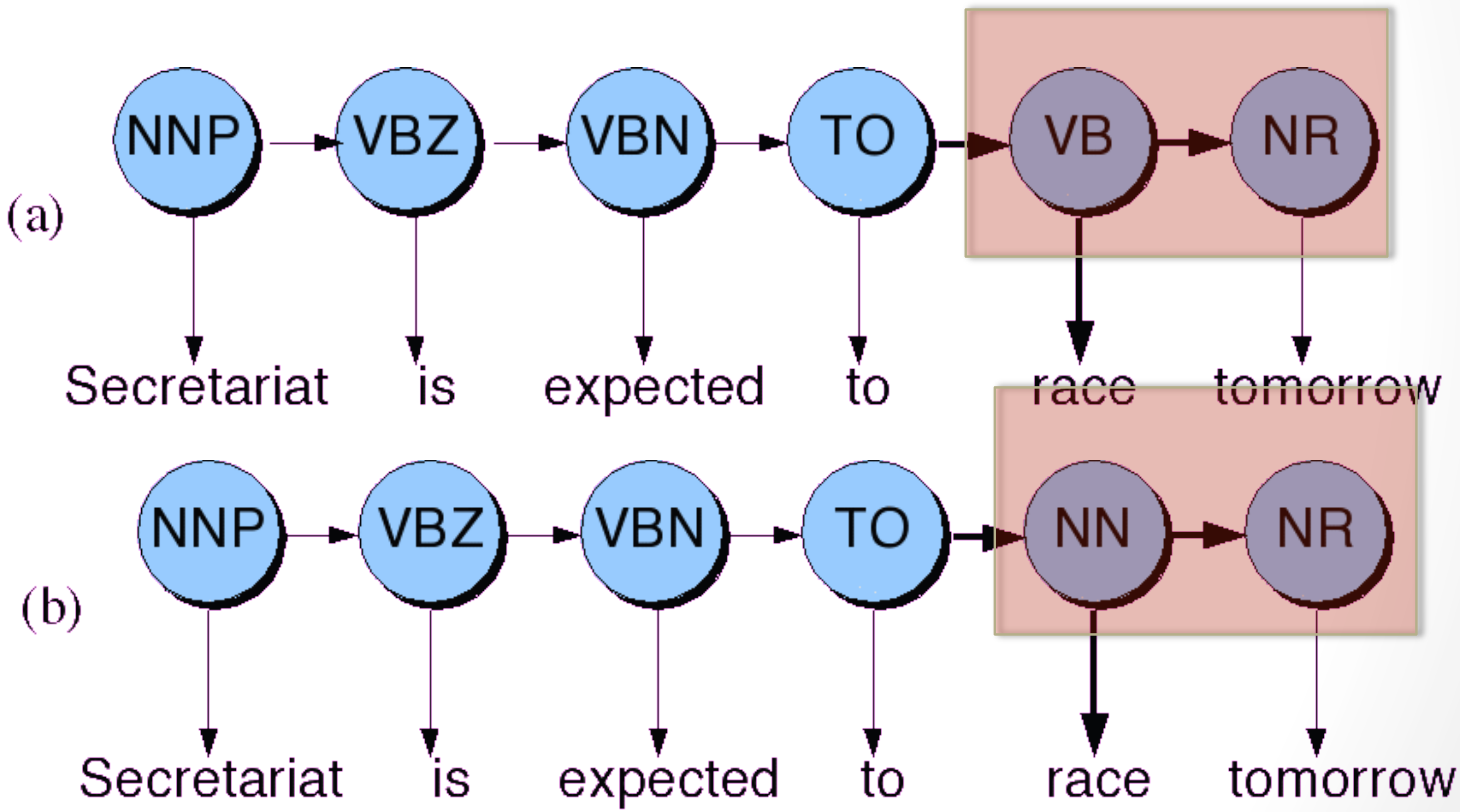
Disambiguating “race”



Disambiguating "race"



Disambiguating "race"

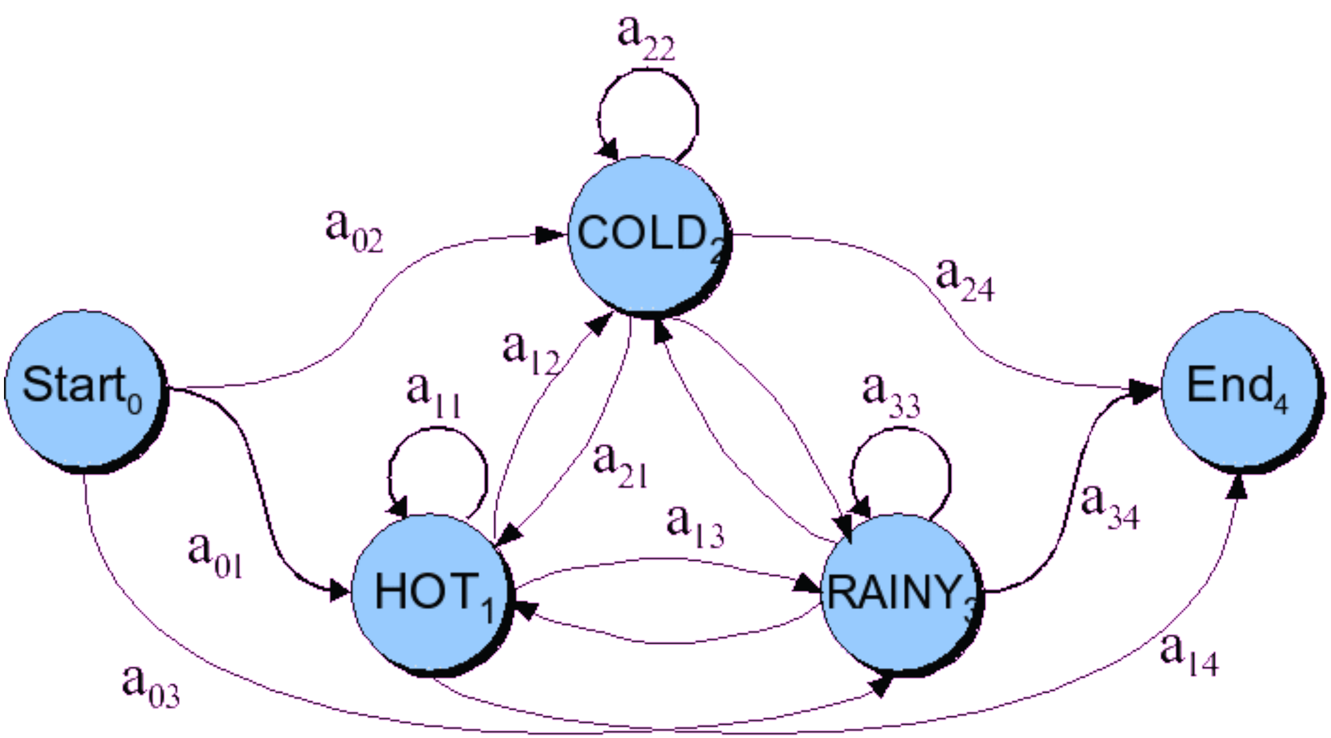


- P(NN|TO) = .00047
- P(VB|TO) = .83
- P(race|NN) = .00057
- P(race|VB) = .00012
- P(NR|VB) = .0027
- P(NR|NN) = .0012
- P(VB|TO)P(NR|VB)P(race|VB) = .00000027
- P(NN|TO)P(NR|NN)P(race|NN)=.00000000032
- So we (correctly) choose the verb reading,

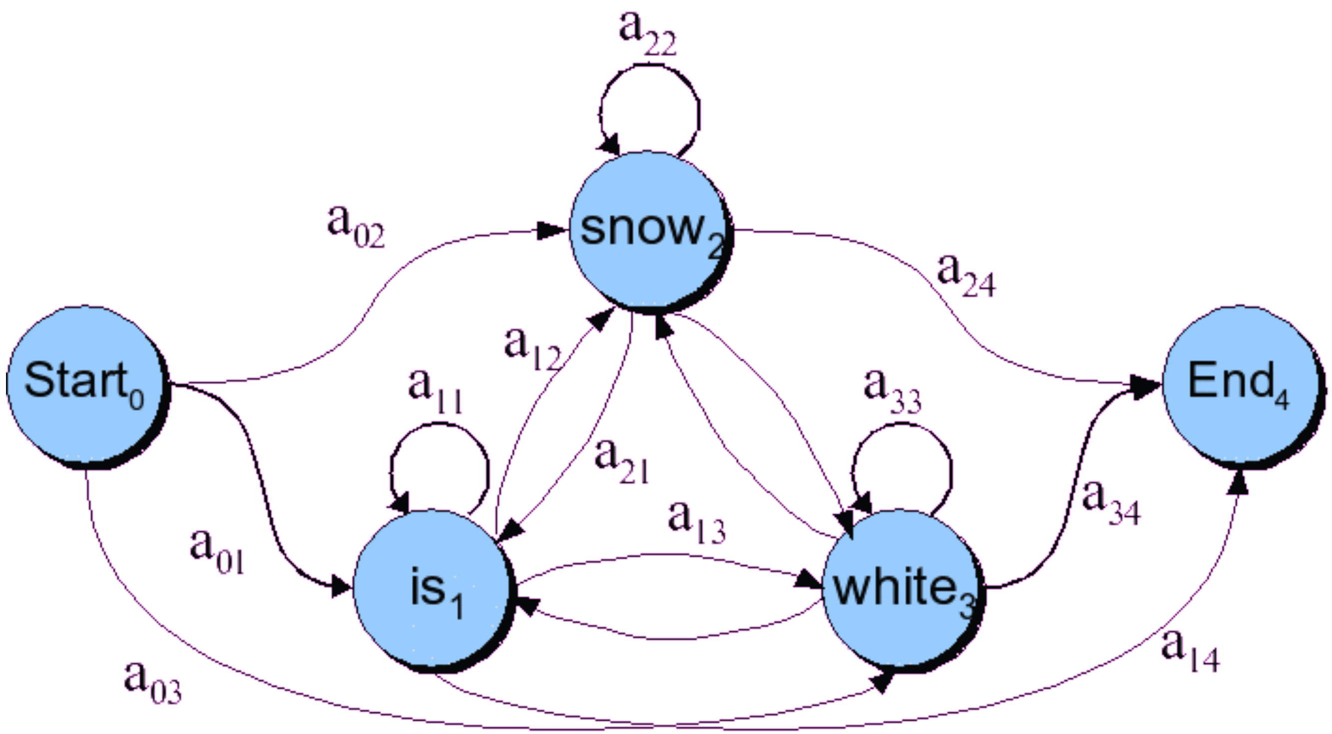
Definitions

- 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
  - the input sequence uniquely determines which states the automaton will go through
- Markov chains can't represent inherently ambiguous problems
  - Assigns 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

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

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

- Distinguished start and end states

Markov chain = "First-order observable Markov Model"

- 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

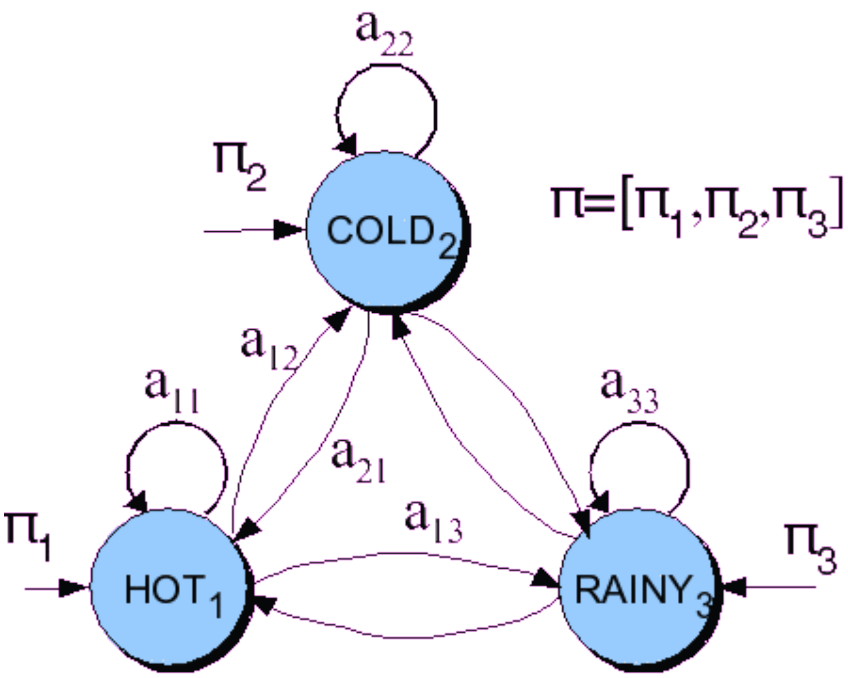
- Instead of start state
- Special initial probability vector π

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

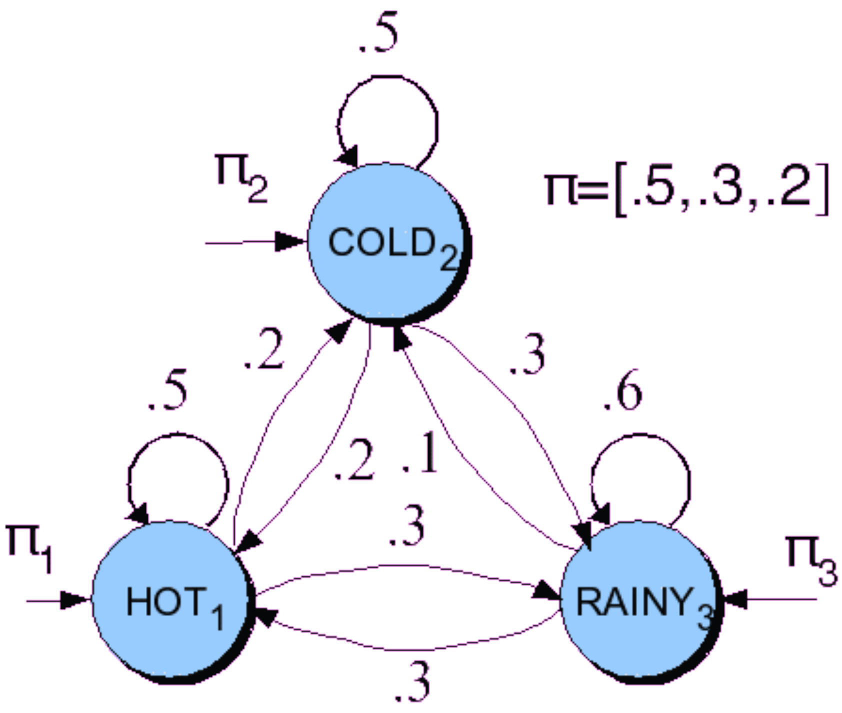
  - An initial distribution over probability of start states
- Constraints:

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

The weather figure using pi



The weather figure: specific example



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 x $(0.6)^3$ = 0.0432

Response

Hidden Markov Models

- We don't observe POS tags
  - We infer them from the words we see

- Observed events

- Hidden events

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 New York, NY for summer of 2007
- But you find Kathy McKeown's diary
- Which lists how many ice-creams Kathy ate every date that summer
- Our job: figure out how hot it was

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**
  - 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.

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 $A = \{a_{ij}\}$

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

- 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

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

Hidden Markov Models

- Some constraints

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

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

$$\sum_{k=1}^{M} b_i(k) = 1 \qquad\qquad \sum_{j=1}^{N} \pi_j = 1$$

Assumptions

- **Markov assumption:**

- **Output-independence assumption**

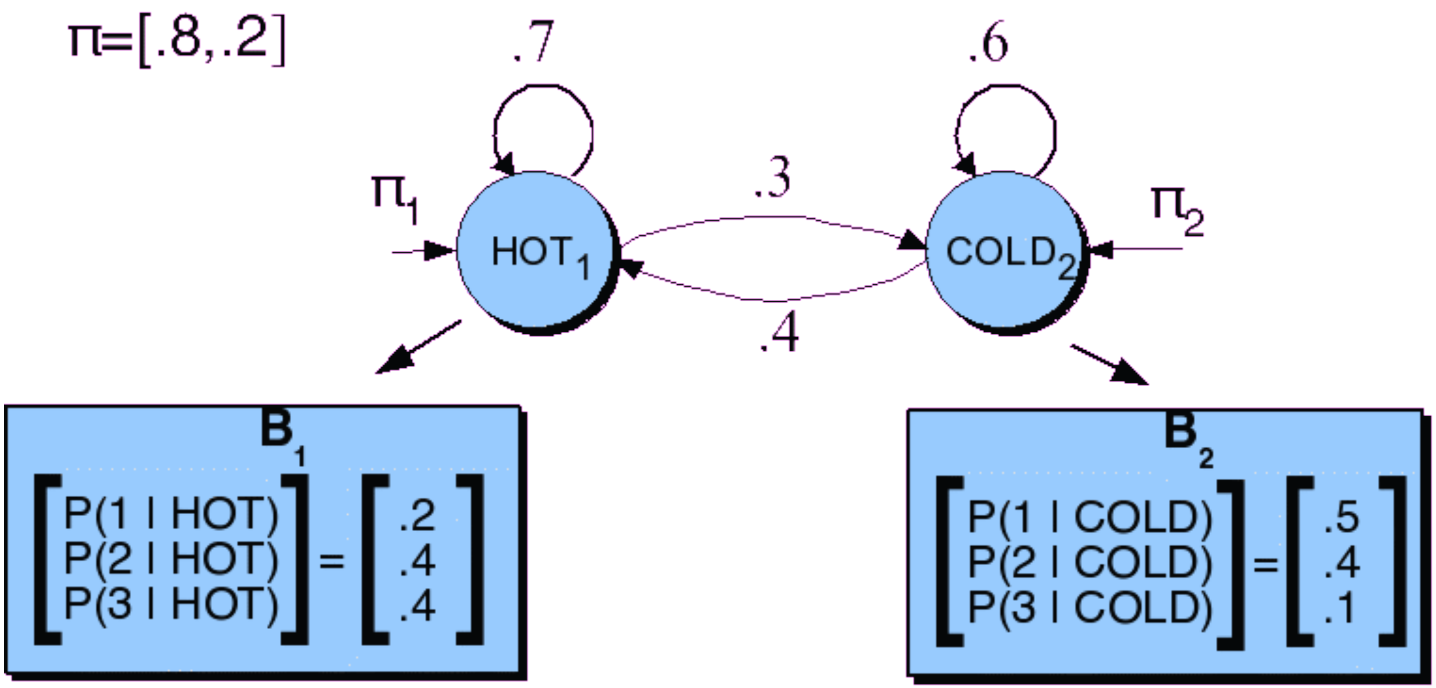
$$P(q_i \mid q_1 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$$

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

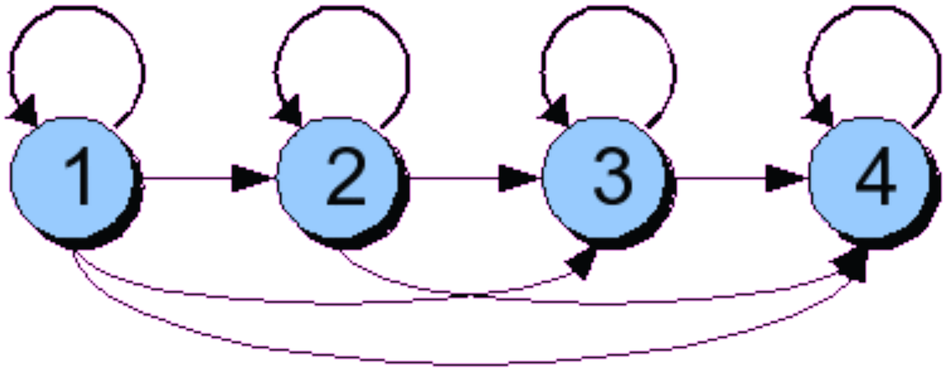
McKeown task

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

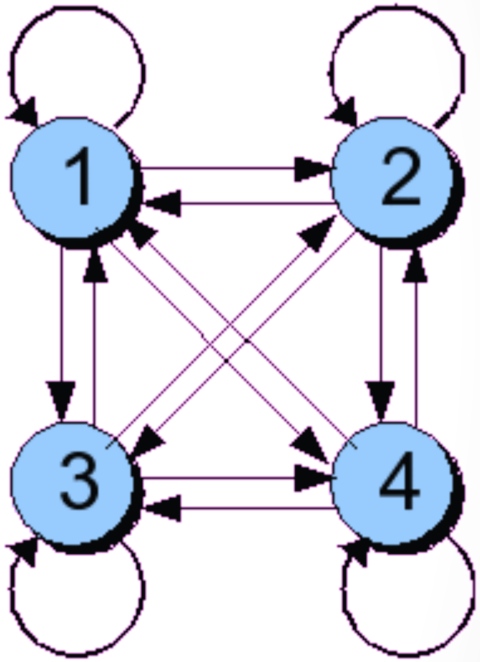
HMM for ice cream



Different types of HMM structure

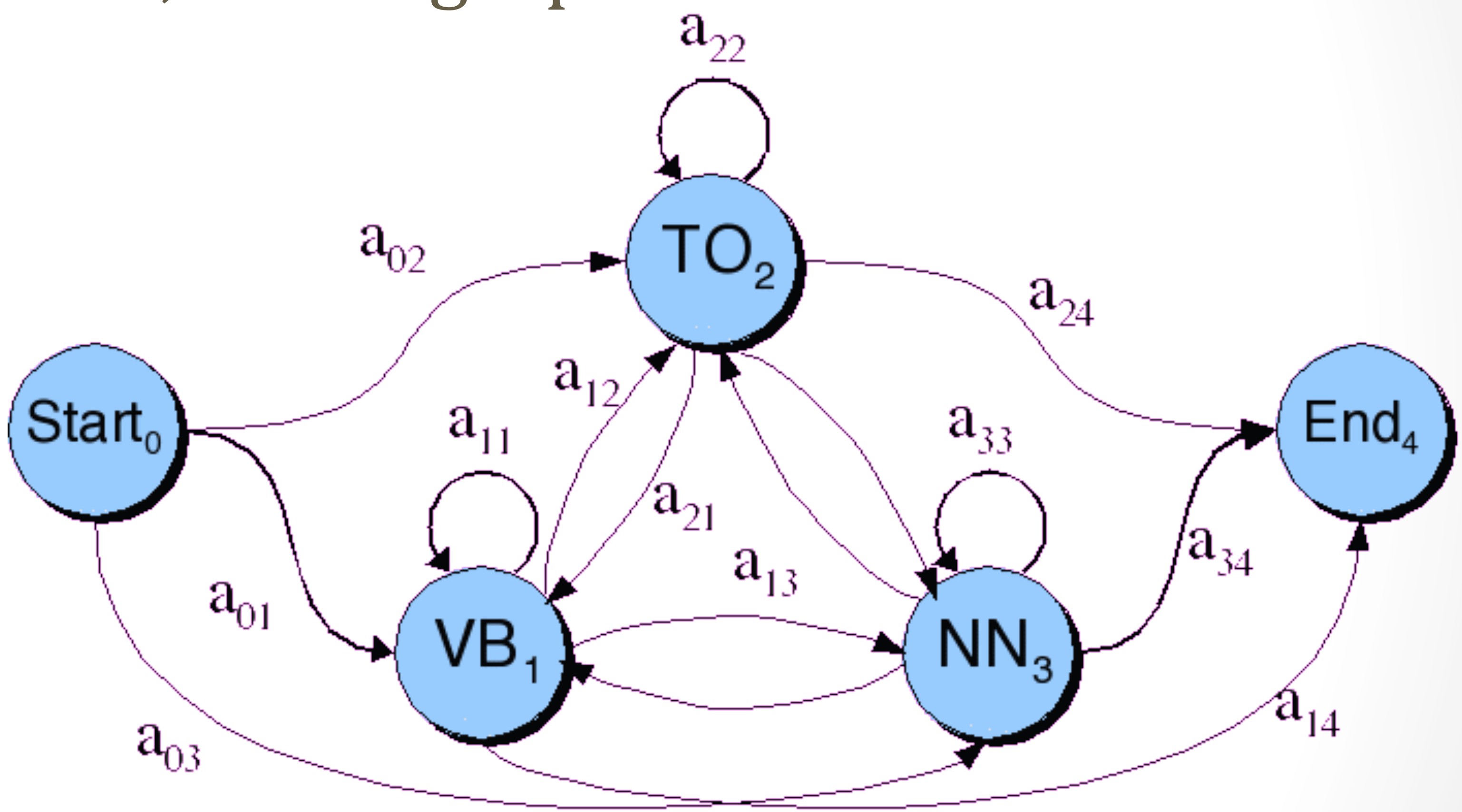


Bakis = left-to-right

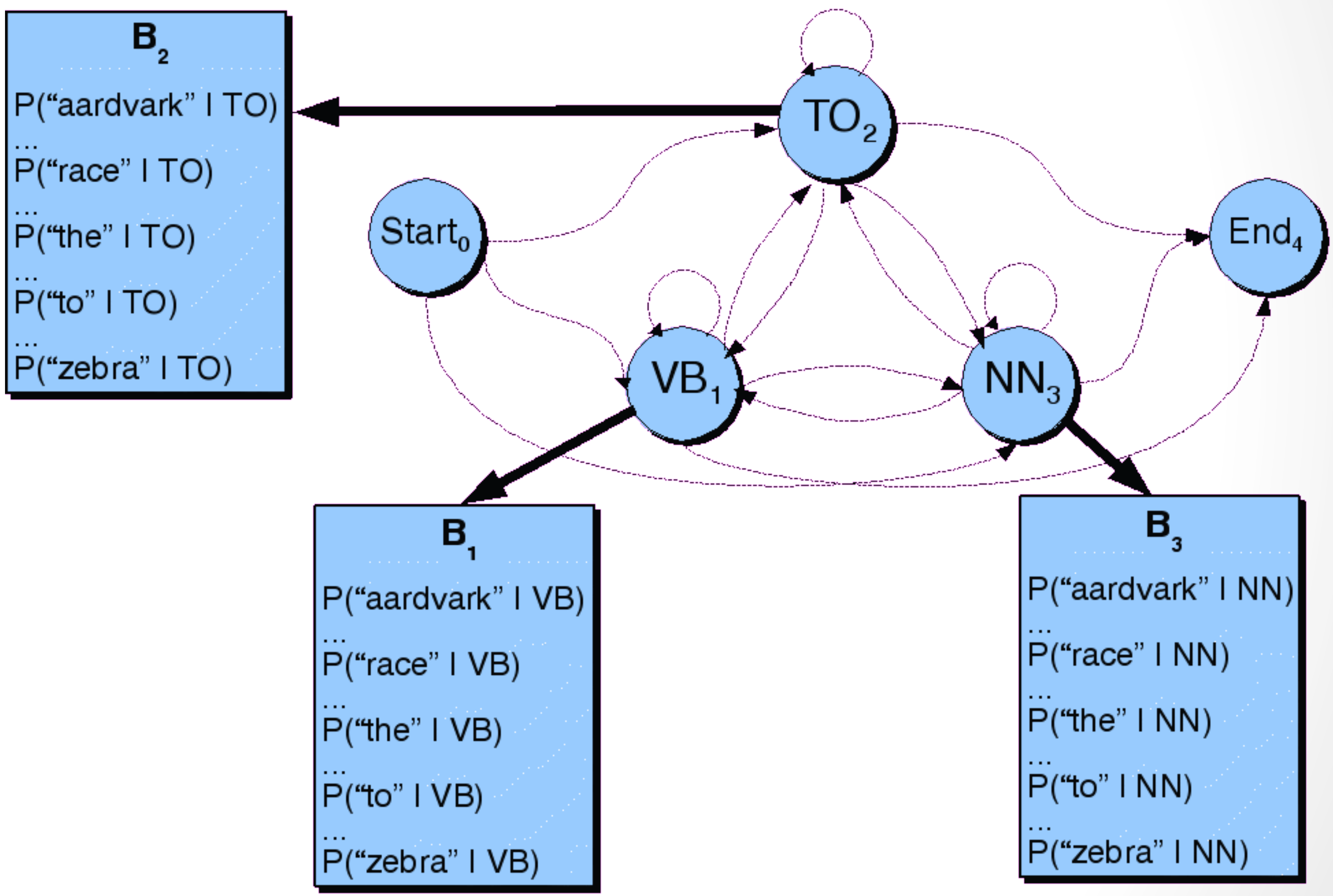


Ergodic =
fully-connected

Transitions between the hidden states of HMM, showing A probs



B observation likelihoods for POS HMM



Three fundamental Problems for HMMs

- ***Likelihood***: Given an HMM λ = (A,B) and an observation sequence O, determine the likelihood P(O, λ).
- ***Decoding***: Given an observation sequence O and an HMM λ = (A,B), discover the best hidden state sequence Q.
- ***Learning***: Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

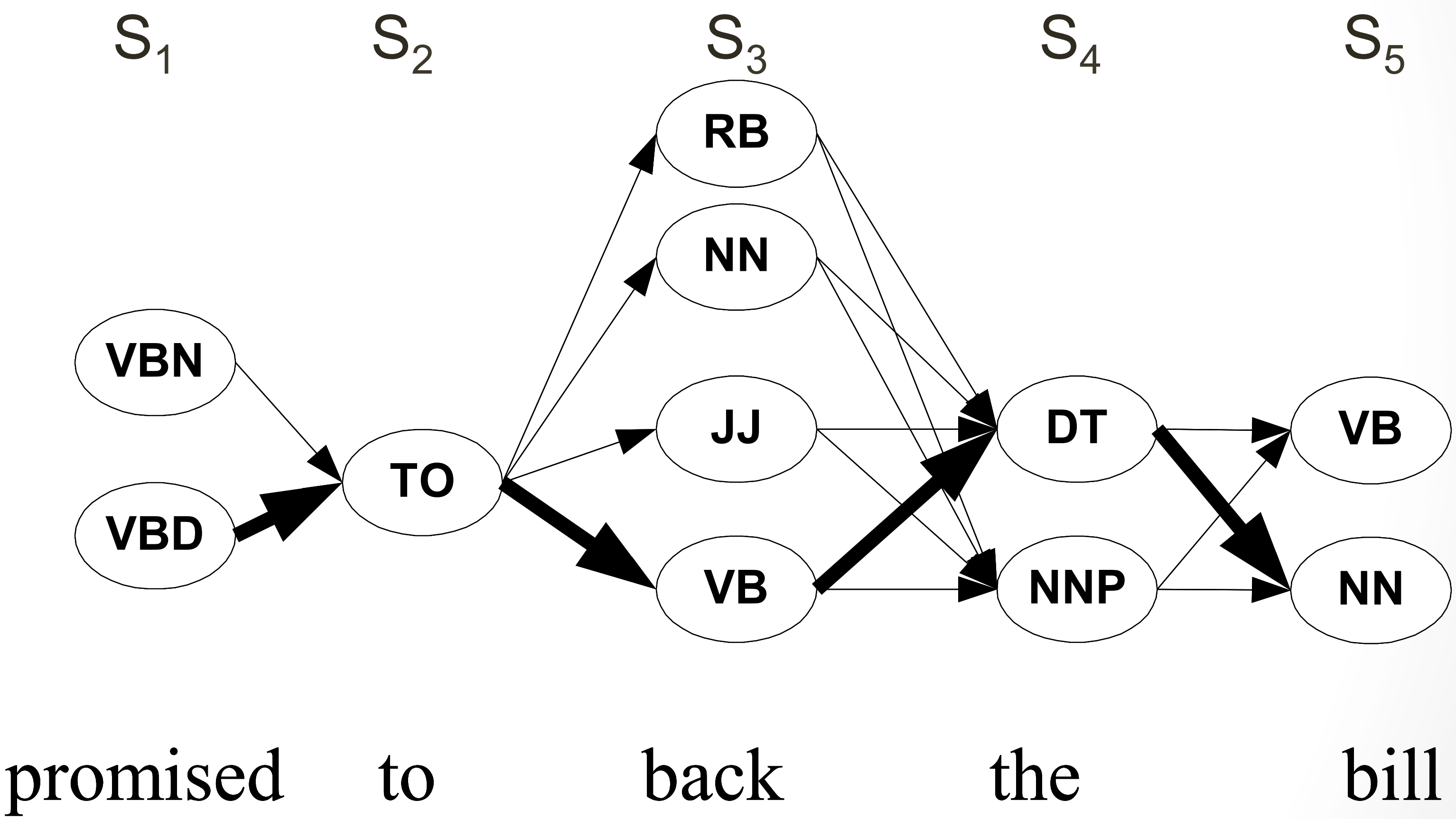
  What kind of data would we need to learn the HMM parameters?

Response

Decoding

- The best hidden sequence
    - Weather sequence in the ice cream task
    - POS sequence given an input sentence
- We could use argmax over the probability of each possible hidden state sequence
  - *Why not?*
- Viterbi algorithm
  - Dynamic programming algorithm
  - Uses a dynamic programming trellis
    - Each trellis cell represents, $v_t(j)$, represents the probability that the HMM is in state j after seeing the first t observations and passing through the most likely state sequence

Viterbi intuition: we are looking for the best 'path'



Slide from Dekang Lin

Intuition

- The value in each cell is computed by taking the MAX over all paths that lead to this cell.
- An extension of a path from state i at time t-1 is computed by multiplying:

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

$v_{t-1}(i)$	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

The Viterbi Algorithm

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

num-states ← NUM-OF-STATES(*state-graph*)
Create a path probability matrix *viterbi[num-states+2,T+2]*
viterbi[0,0] ← 1.0
for each time step t **from** 1 **to** T **do**
  for each state s **from** 1 **to** *num-states* **do**
    $viterbi[s,t] \leftarrow \max_{1 \leq s' \leq num\text{-}states} viterbi[s', t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{1 \leq s' \leq num\text{-}states} viterbi[s', t-1] * a_{s',s}$
Backtrace from highest probability state in final column of *viterbi[]* and return path

The A matrix for the POS HMM

	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

Figure 4.15 Tag transition probabilities (the a array, $p(t_i|t_{i-1})$ computed from the 87-tag Brown corpus without smoothing. The rows are labeled with the conditioning event; thus $P(PPSS|VB)$ is .0070. The symbol <s> is the start-of-sentence symbol.

What is P(VB|TO)? What is P(NN|TO)? Why does this make sense?

What is P(TO|VB)? What is P(TO|NN)? Why does this make sense?

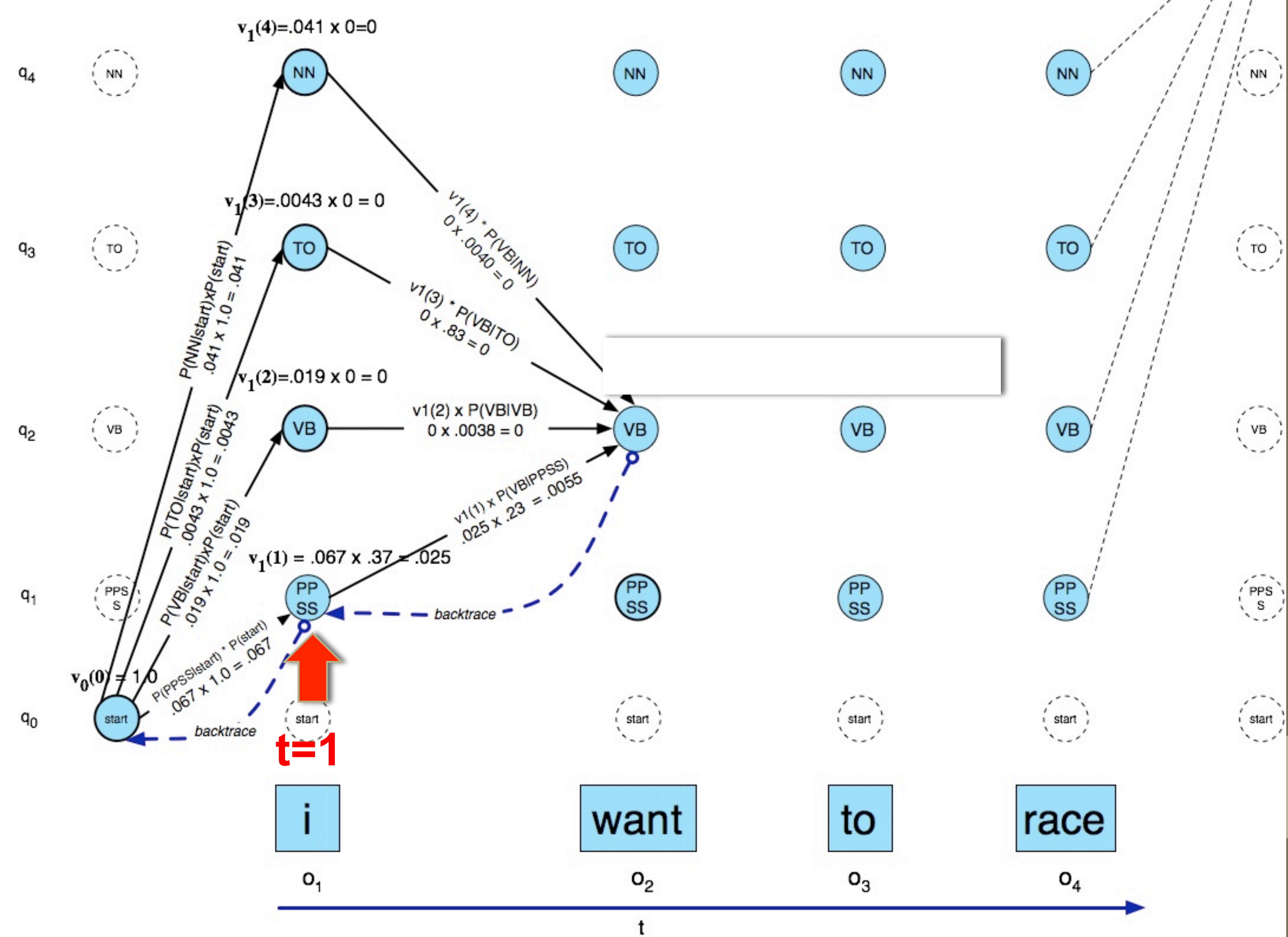
The B matrix for the POS HMM

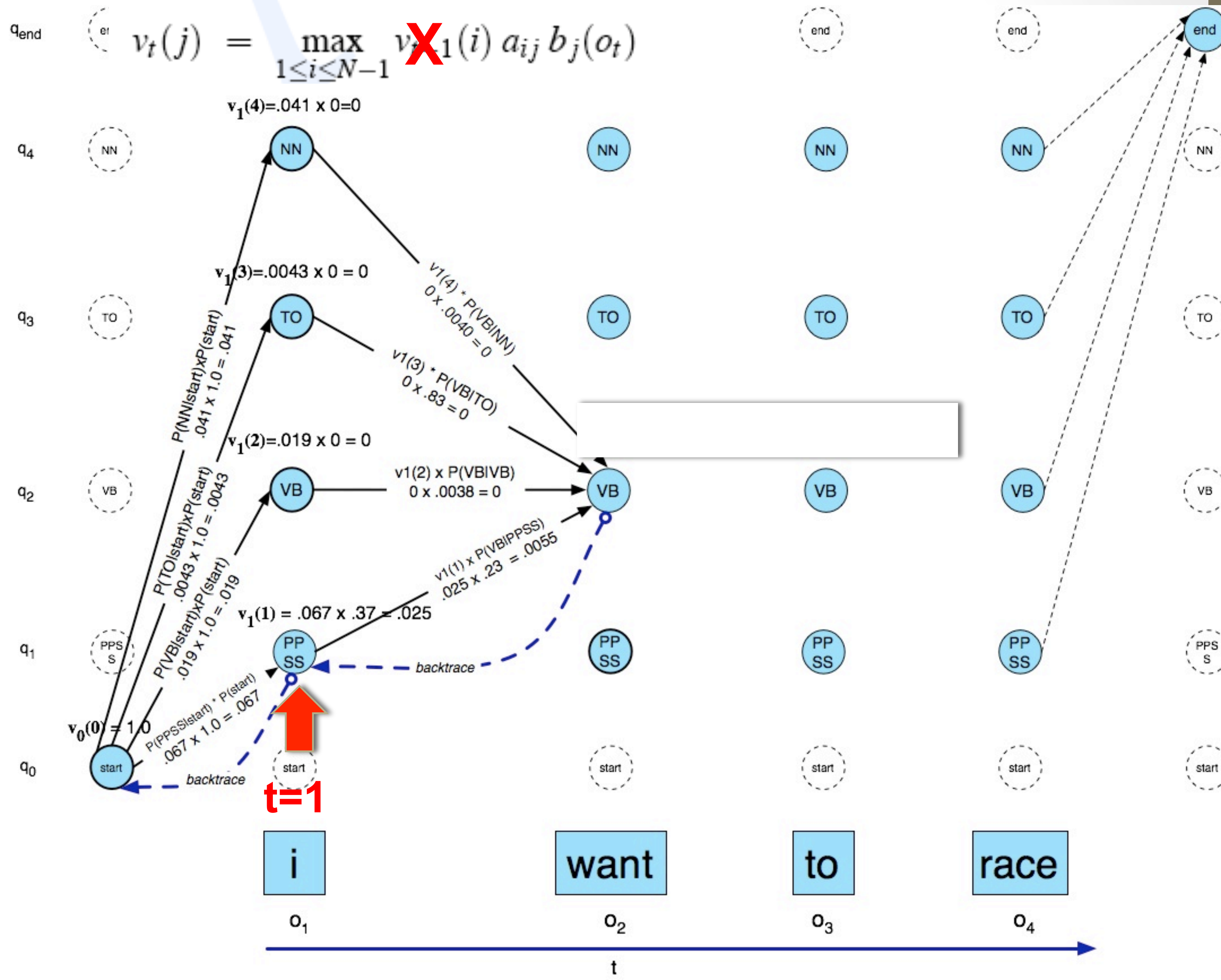
	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

Figure 4.16 Observation likelihoods (the *b* array) computed from the 87-tag Brown corpus without smoothing.

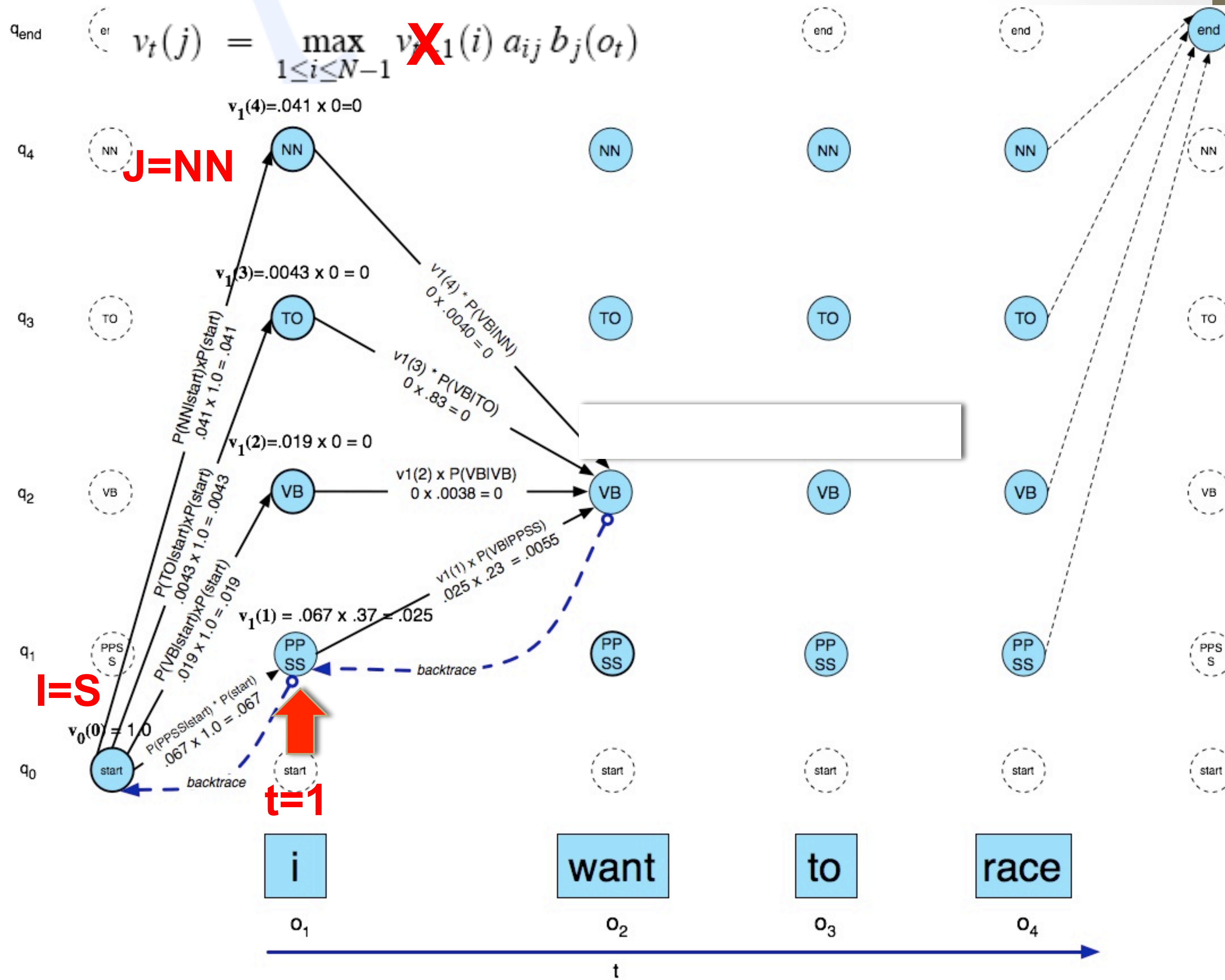
Look at P(want|VB) and P(want|NN). Give an explanation for the difference in the probabilities.

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




$v_t(j) = \max_{1 \le i \le N-1} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$
q_{end}
q_4
q_3
q_2
q_1
q_0
$v_1(4)$=.041 x 0=0
$v_1(3)$=.0043 x 0 = 0
$v_1(2)$=.019 x 0 = 0
$v_1(1)$ = .067 x .37 = .025
$v_0(0)$ = 1.0
P(NN|start)xP(start) .041 x 1.0 = .041
P(TO|start)xP(start) .0043 x 1.0 = .0043
P(VB|start)xP(start) .019 x 1.0 = .019
P(PPSS|start) * P(start) .067 x 1.0 = .067
v1(4) * P(VB|NN) 0 x .0040 = 0
v1(3) * P(VB|TO) 0 x .83 = 0
v1(2) x P(VB|VB) 0 x .0038 = 0
v1(1) x P(VB|PPSS) .025 x .23 = .0055
backtrace
backtrace
t=1
NN
TO
VB
PPSS
start
end
i
want
to
race
o_1
o_2
o_3
o_4
t

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



The A matrix for the POS HMM

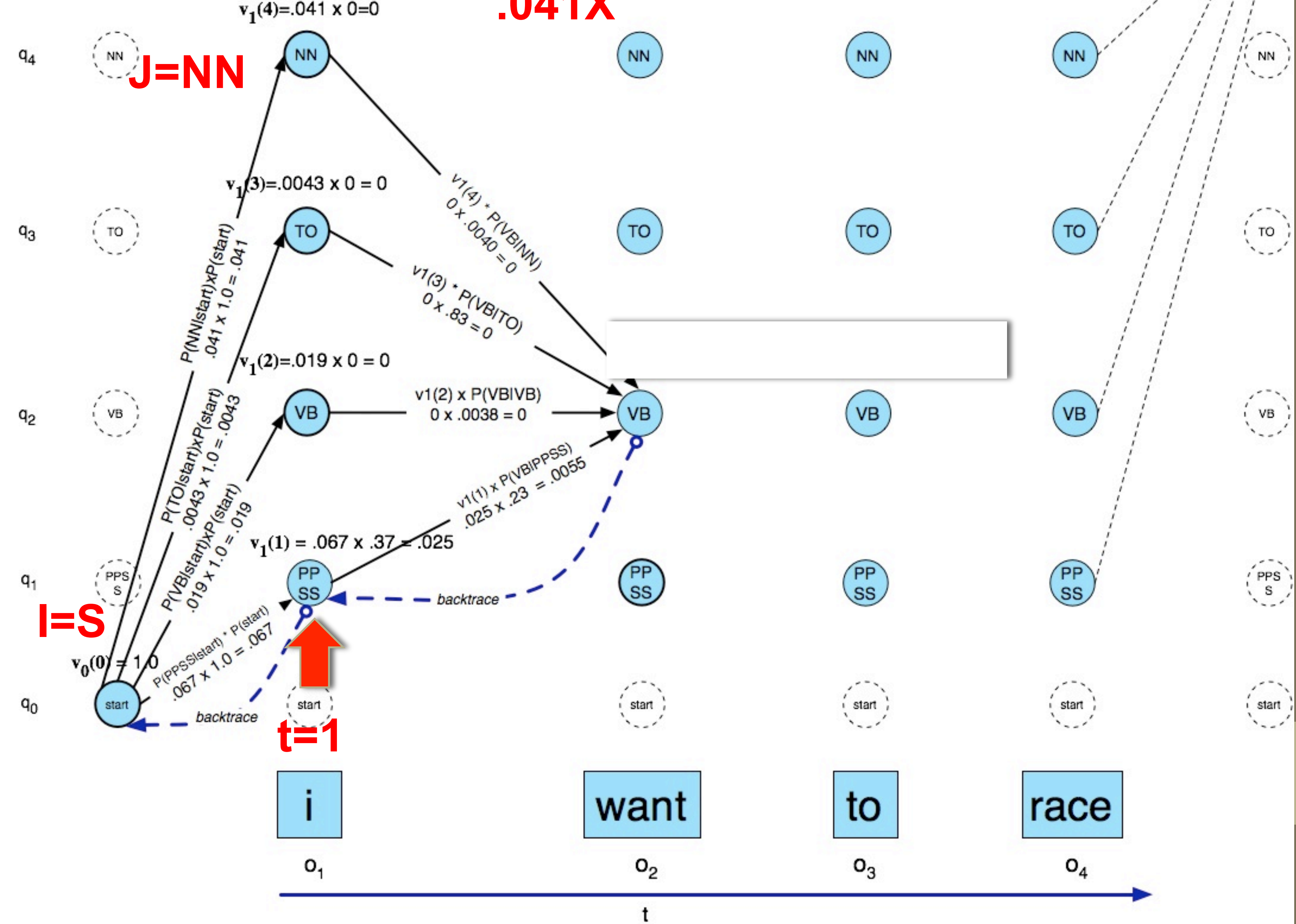
	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

Figure 4.15 Tag transition probabilities (the *a* array, $p(t_i|t_{i-1})$ computed from the 87-tag Brown corpus without smoothing. The rows are labeled with the conditioning event; thus $P(PPSS|VB)$ is .0070. The symbol <s> is the start-of-sentence symbol.

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

X

.041X



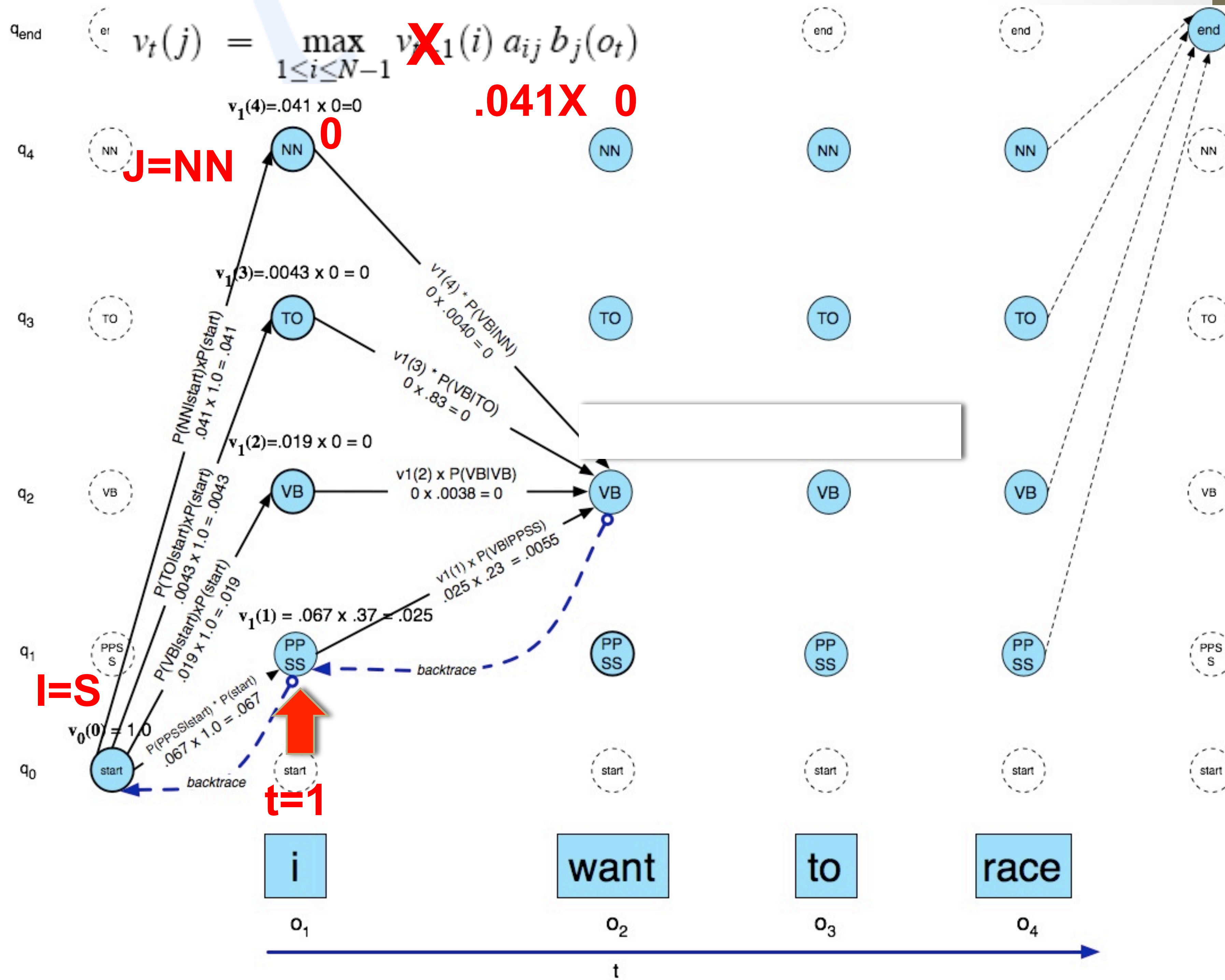
The B matrix for the POS HMM

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

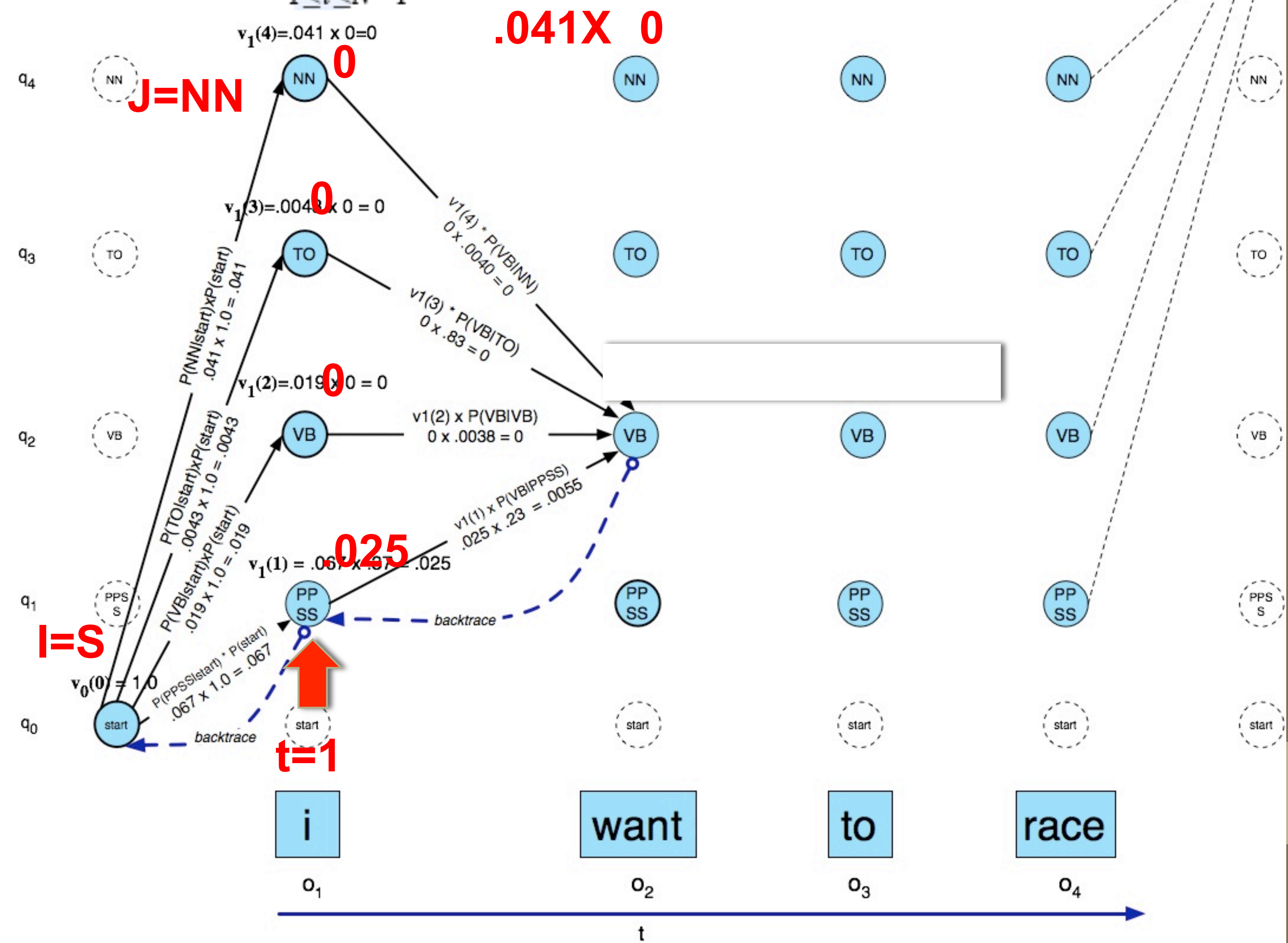
Figure 4.16 Observation likelihoods (the *b* array) computed from the 87-tag Brown corpus without smoothing.

Look at P(want|VB) and P(want|NN). Give an explanation for the difference in the probabilities.

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

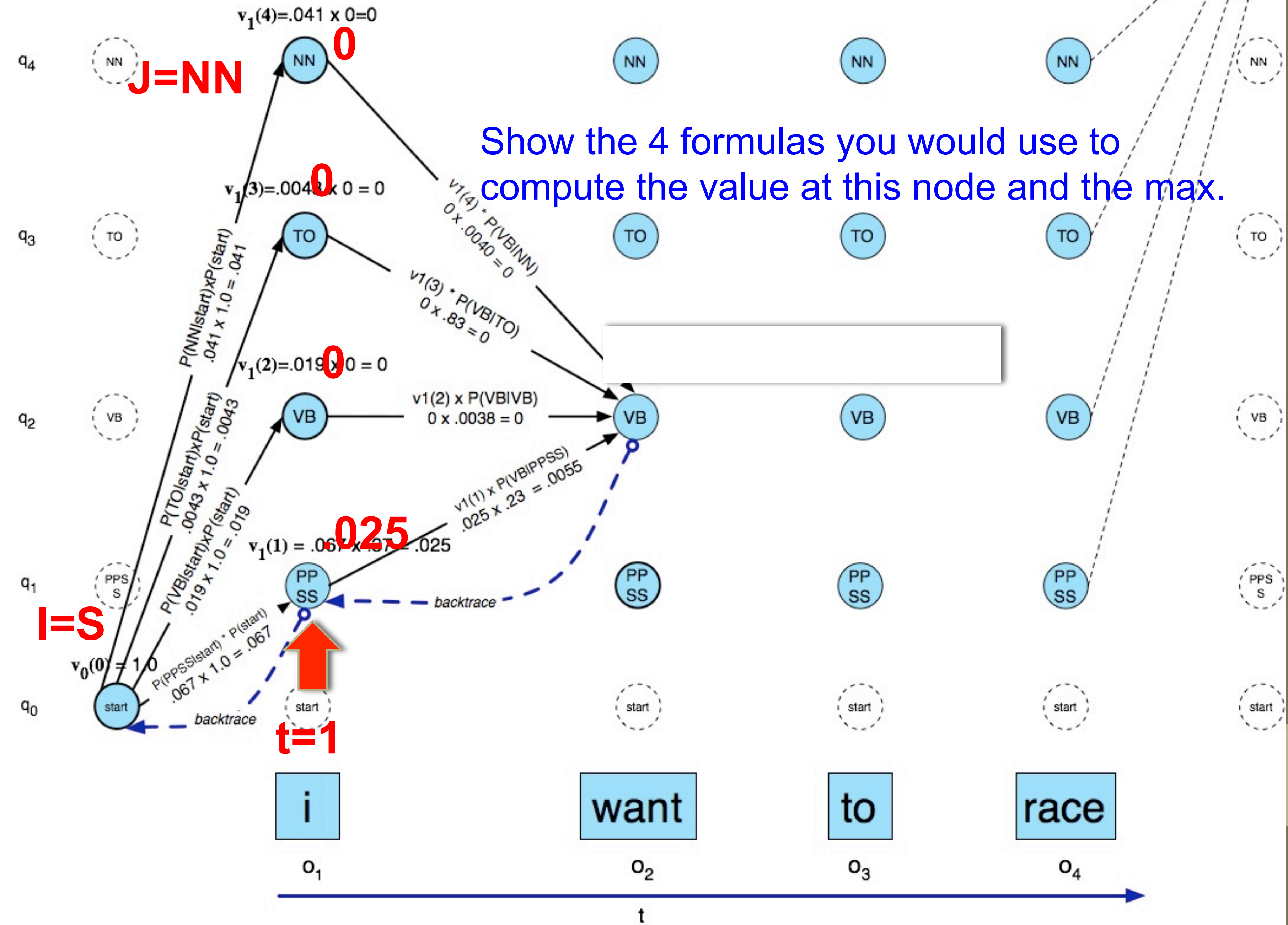


$$v_t(j) = \max_{1 \le i \le N-1} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$



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

Show the 4 formulas you would use to compute the value at this node and the max.



Computing the Likelihood of an observation

- Forward algorithm
- Exactly like the viterbi algorithm, except
  - To compute the probability of a state, sum the probabilities from each path

Error Analysis: ESSENTIAL!!!

- Look at a confusion matrix

	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 (NN) vs Adj (JJ)
  - Adverb (RB) vs Prep (IN) vs Noun (NN)
  - Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)