# Today

- Finish word sense disambiguation
- Midterm Review
- The midterm is Thursday during class time. H students and 20 regular students are assigned to **517 Hamilton**.
- Make-up for those with same time midterm is Thurs 6pm. For those with serious problems, second make-up Mon 6pm, but will be harder. RECOMMENDATION: Thurs 6pm if you can.
- Review questions and answer on NLP website.
- On Thursday, I will not be here. I am at a meeting in Maryland. Tas will proctor

# Naïve Bayes Test

- On a corpus of examples of uses of the word line, naïve Bayes achieved about 73% correct
- Good?

# Decision Lists: another popular method

- A case statement....

| Rule | | Sense |
|---|---|---|
| *fish* within window | $\Rightarrow$ | **bass**$^1$ |
| *striped bass* | $\Rightarrow$ | **bass**$^1$ |
| *guitar* within window | $\Rightarrow$ | **bass**$^2$ |
| *bass player* | $\Rightarrow$ | **bass**$^2$ |
| *piano* within window | $\Rightarrow$ | **bass**$^2$ |
| *tenor* within window | $\Rightarrow$ | **bass**$^2$ |
| *sea bass* | $\Rightarrow$ | **bass**$^1$ |
| *play/V bass* | $\Rightarrow$ | **bass**$^2$ |
| *river* within window | $\Rightarrow$ | **bass**$^1$ |
| *violin* within window | $\Rightarrow$ | **bass**$^2$ |
| *salmon* within window | $\Rightarrow$ | **bass**$^1$ |
| *on bass* | $\Rightarrow$ | **bass**$^2$ |
| *bass are* | $\Rightarrow$ | **bass**$^1$ |

# Learning Decision Lists

- Restrict the lists to rules that test a single feature (1-decisionlist rules)
- Evaluate each possible test and rank them based on how well they work.
- Glue the top-N tests together and call that your decision list.

# Yarowsky

- On a binary (homonymy) distinction used the following metric to rank the tests

$$\frac{P(\mathrm{Sense}_1 \mid Feature)}{P(\mathrm{Sense}_2 \mid Feature)}$$

- This gives about 95% on this test…

# WSD Evaluations and baselines

- *In vivo* versus *in vitro* evaluation
- In vitro evaluation is most common now
  - Exact match **accuracy**
    - % of words tagged identically with manual sense tags
  - Usually evaluate using held-out data from same labeled corpus
    - Problems?
    - Why do we do it anyhow?
- Baselines
  - Most frequent sense
  - The Lesk algorithm

# Most Frequent Sense

- Wordnet senses are ordered in frequency order
- So "most frequent sense" in wordnet = "take the first sense"

| Freq | Synset | Gloss |
|---|---|---|
| 338 | $\text{plant}^1$, works, industrial plant | buildings for carrying on industrial labor |
| 207 | $\text{plant}^2$, flora, plant life | a living organism lacking the power of locomotion |
| 2 | $\text{plant}^3$ | something planted secretly for discovery by another |
| 0 | $\text{plant}^4$ | an actor situated in the audience whose acting is rehearsed but seems spontaneous to the audience |

# Ceiling

- Human inter-annotator agreement
  - Compare annotations of two humans
  - On same data
  - Given same tagging guidelines
- Human agreements on all-words corpora with Wordnet style senses
  - 75%-80%

# Problems

- Given these general ML approaches, how many classifiers do I need to perform WSD robustly
  - One for each ambiguous word in the language
- How do you decide what set of tags/ labels/senses to use for a given word?
  - Depends on the application

# WordNet Bass

- Tagging with this set of senses is an impossibly hard task that's probably overkill for any realistic application

1. bass - (the lowest part of the musical range)
2. bass, bass part - (the lowest part in polyphonic music)
3. bass, basso - (an adult male singer with the lowest voice)
4. sea bass, bass - (flesh of lean-fleshed saltwater fish of the family Serranidae)
5. freshwater bass, bass - (any of various North American lean-fleshed freshwater fishes especially of the genus Micropterus)
6. bass, bass voice, basso - (the lowest adult male singing voice)
7. bass - (the member with the lowest range of a family of musical instruments)
8. bass -(nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

# Senseval History

- ACL-SIGLEX workshop (1997)
  - Yarowsky and Resnik paper
- SENSEVAL-I (1998)
  - Lexical Sample for English, French, and Italian
- SENSEVAL-II (Toulouse, 2001)
  - Lexical Sample and All Words
  - Organization: Kilkgarriff (Brighton)
- SENSEVAL-III (2004)
- SENSEVAL-IV -> SEMEVAL (2007)
- SEMEVAL (2010)
- SEMEVAL 2017: http://alt.qcri.org/semeval2017/index.php?id=tasks

SLIDE ADAPTED FROM CHRIS MANNING

# WSD Performance

- Varies widely depending on how difficult the disambiguation task is
- Accuracies of over 90% are commonly reported on some of the classic, often fairly easy, WSD tasks (pike, star, interest)
- Senseval brought careful evaluation of difficult WSD (many senses, different POS)
- Senseval 1: more fine grained senses, wider range of types:
  - Overall: about 75% accuracy
  - Nouns: about 80% accuracy
  - Verbs: about 70% accuracy

# Summary

- Lexical Semantics
  - Homonymy, Polysemy, Synonymy
  - Thematic roles
- Computational resource for lexical semantics
  - WordNet
- Task
  - Word sense disambiguation
- After midterm: semantic parsing, distributional semantics, neural nets

# Requested topics

- POS tagging
- HMM
- Early parsing algorithm

# POS tagging

# 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"
- $\text{Argmax}_x$ f(x) means "the x such that f(x) is maximized"

# Using Bayes Rule

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

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

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

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

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

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

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

# 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:

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

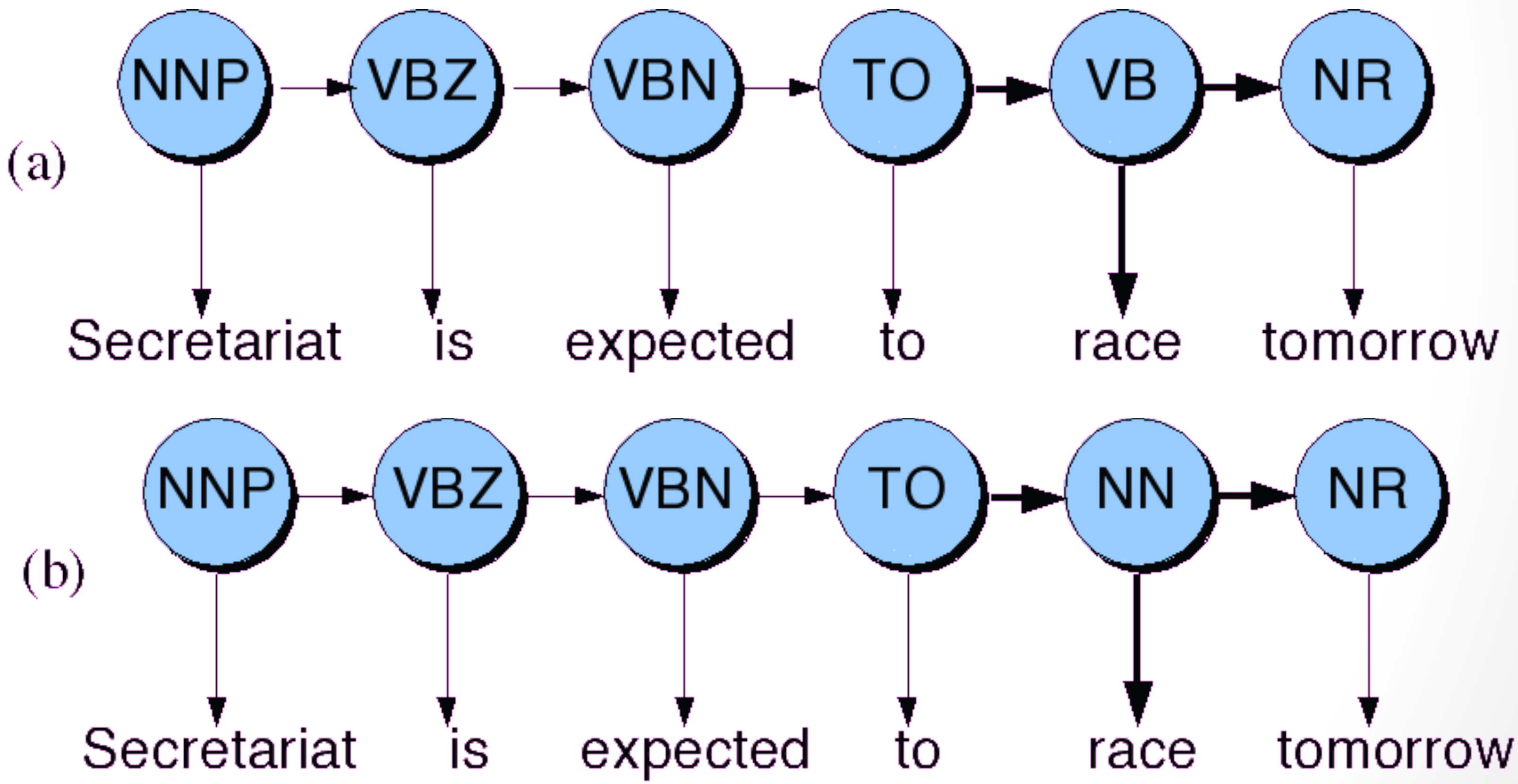
$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

# An Example: the verb "race"

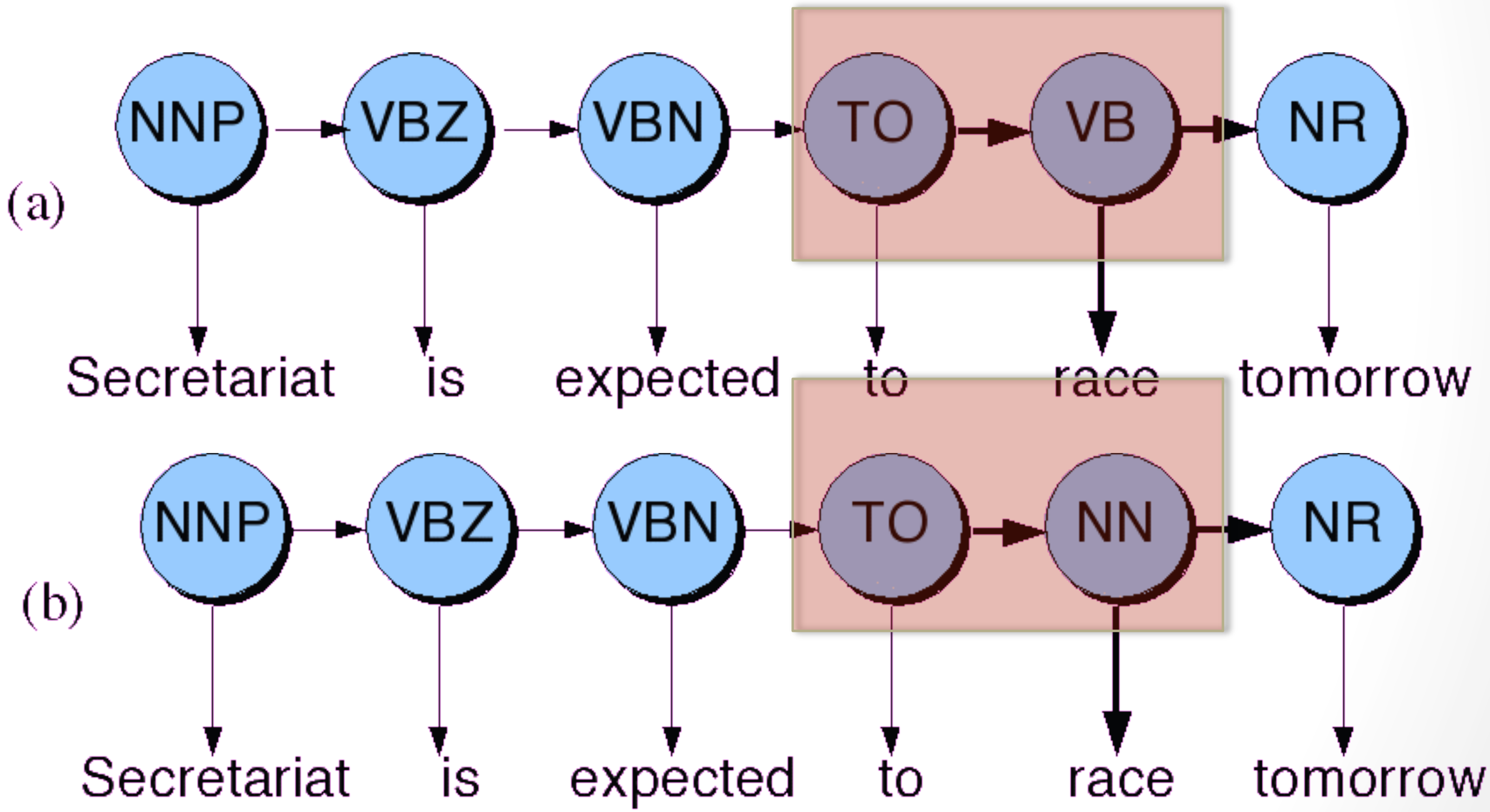
- Secretariat/NNP is/VBZ expected/VBN to/TO **race**/VB tomorrow/NR
- People/NNS continue/VB to/TO inquire/VB the/DT reason/NN for/IN the/DT **race**/NN for/IN outer/JJ space/NN

- How do we pick the right tag?

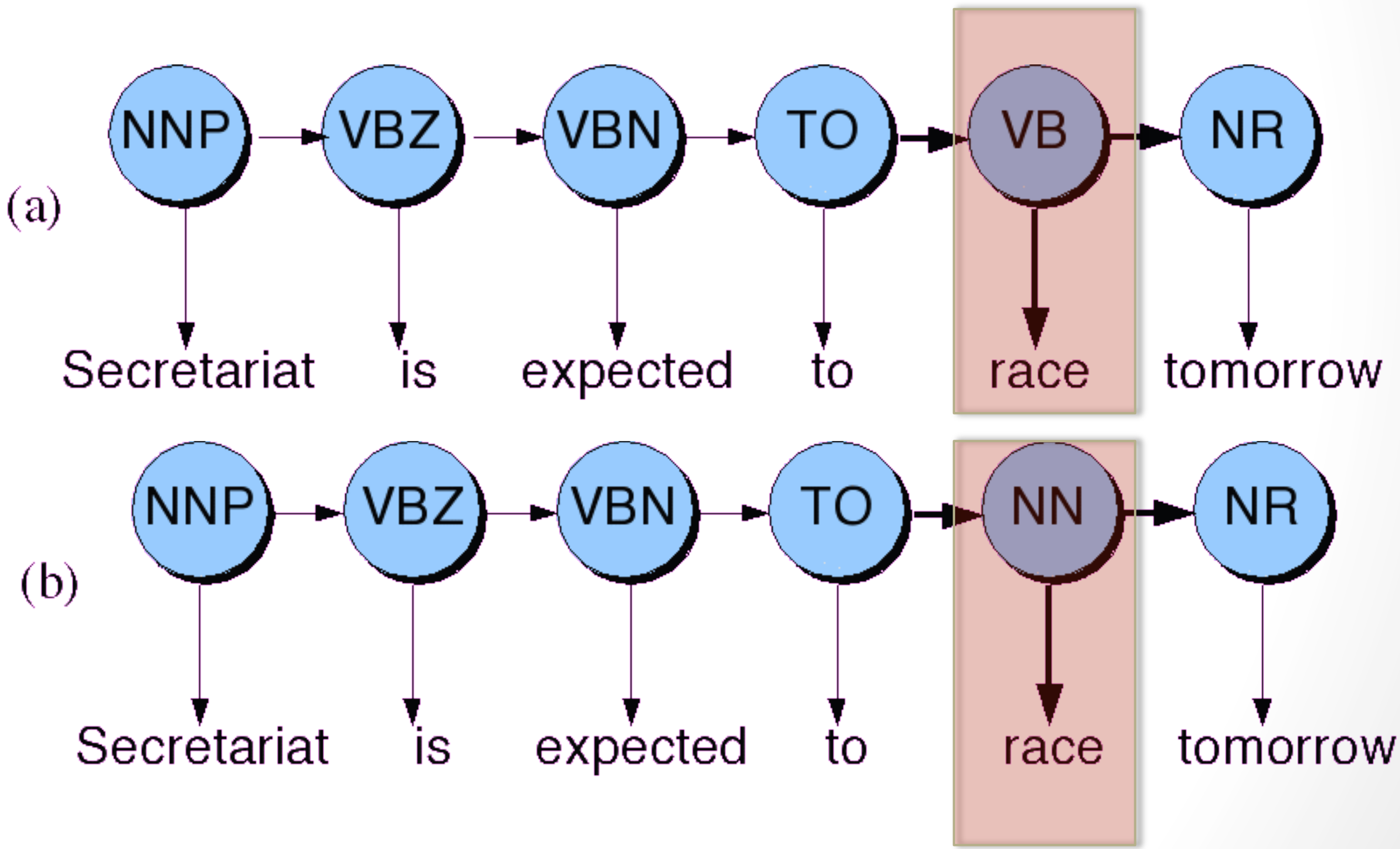
# Disambiguating "race"



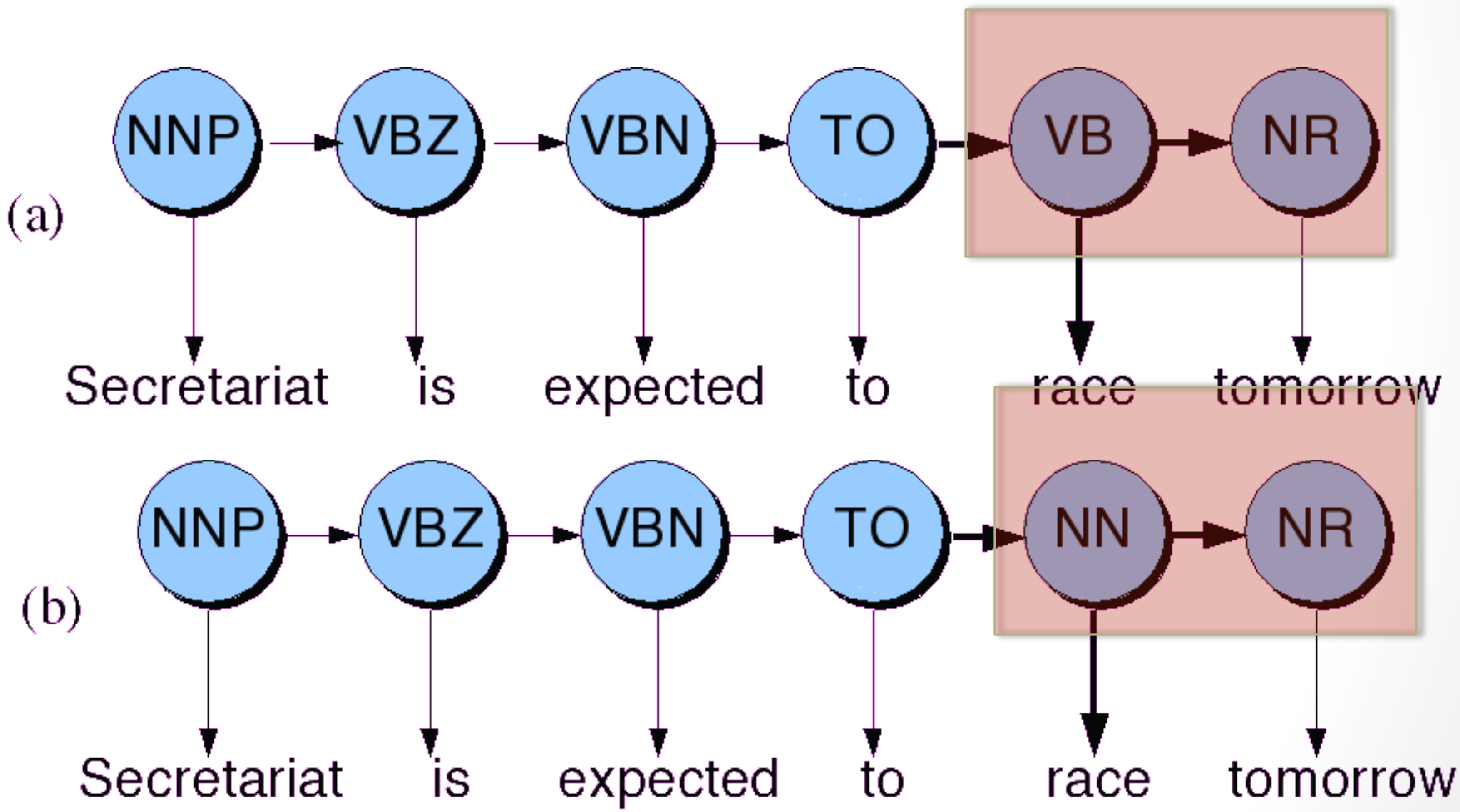
# Disambiguating "race"



# 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,

# Problem

- Observation likelihood: “Promise”

  Count (#promise, VB)/#all verbs

- I promise to back the bill (N).
  I promise to back the bill (V)

# HMMS

# Hidden Markov Models

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

- Observed events

- Hidden events

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

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

- 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 $\pi$

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

# Hidden Markov Models

- Some constraints

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

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

$$\sum_{k=1}^{M} b_i(k) = 1$$

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

# Assumptions

- **Markov assumption:**

- **Output-independence assumption**

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

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

# Three fundamental Problems for HMMs

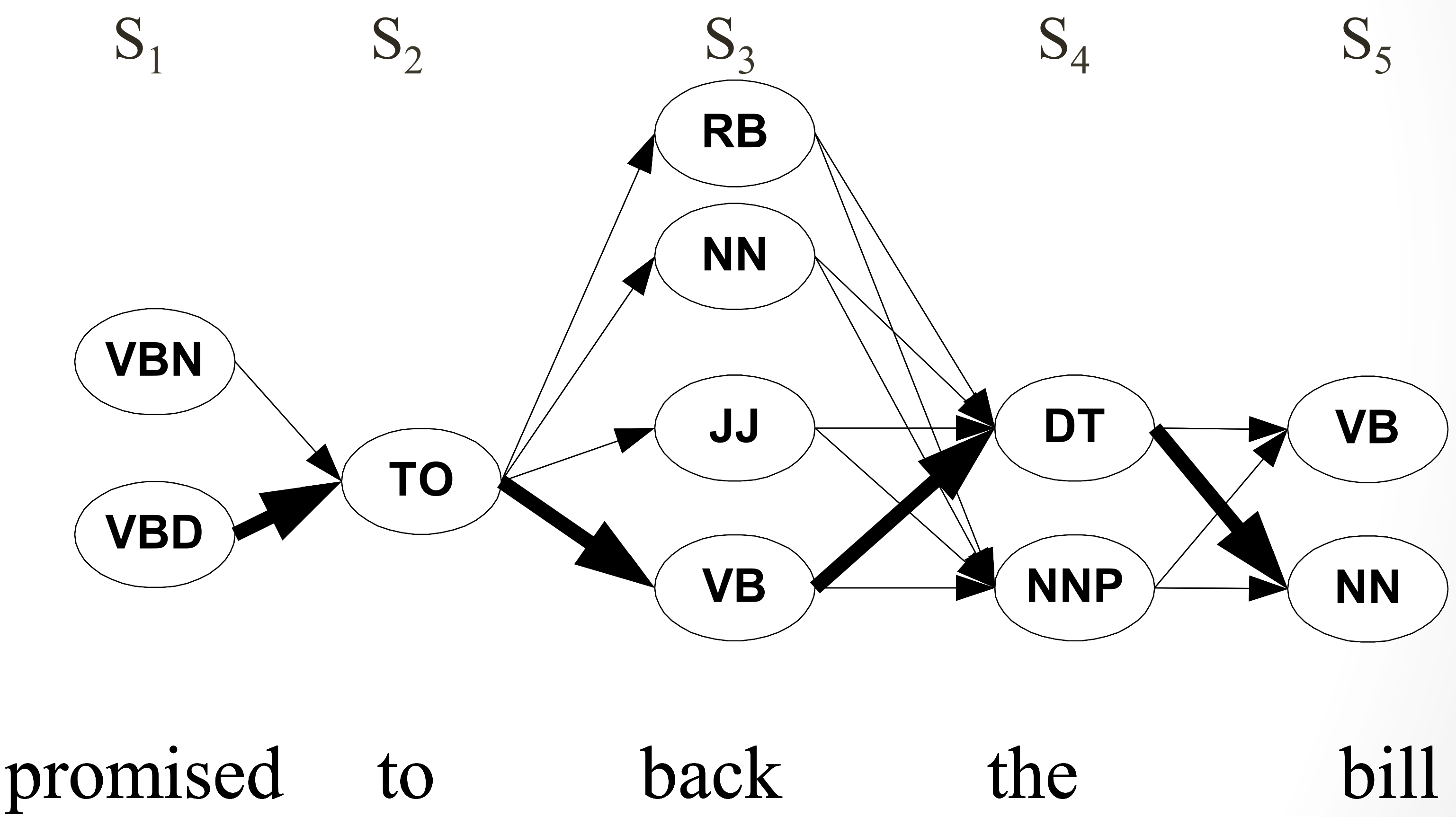
- ***Likelihood***: Given an HMM $\lambda = (A,B)$ and an observation sequence O, determine the likelihood $P(O, \lambda)$.
- ***Decoding***: Given an observation sequence O and an HMM $\lambda = (A,B)$, discover the best hidden state sequence Q.
- ***Learning***: Given an observation sequence O and 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?*

# 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.

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

- An extension of a path from state i at time t-1 is computed by multiplying:

| | |
|---|---|
| $v_{t-1}(i)$ | 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.

| | 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.

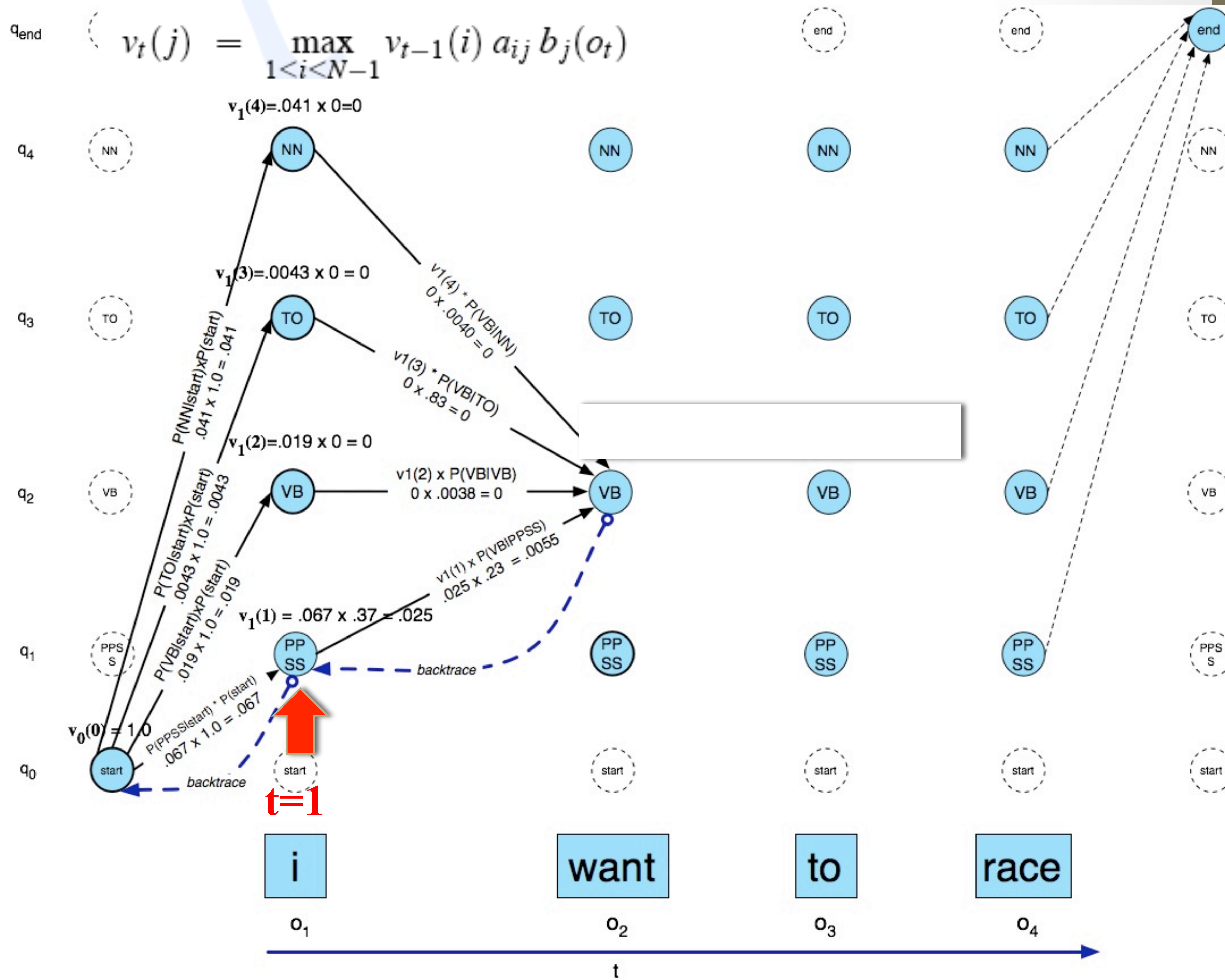
| | I | want | to | race |
|---|---|---|---|---|
| **VB** | 0 | .0093 | 0 | .00012 |
| **TO** | 0 | 0 | .99 | 0 |
| **NN** | 0 | .000054 | 0 | .00057 |
| **PPSS** | .37 | 0 | 0 | 0 |

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

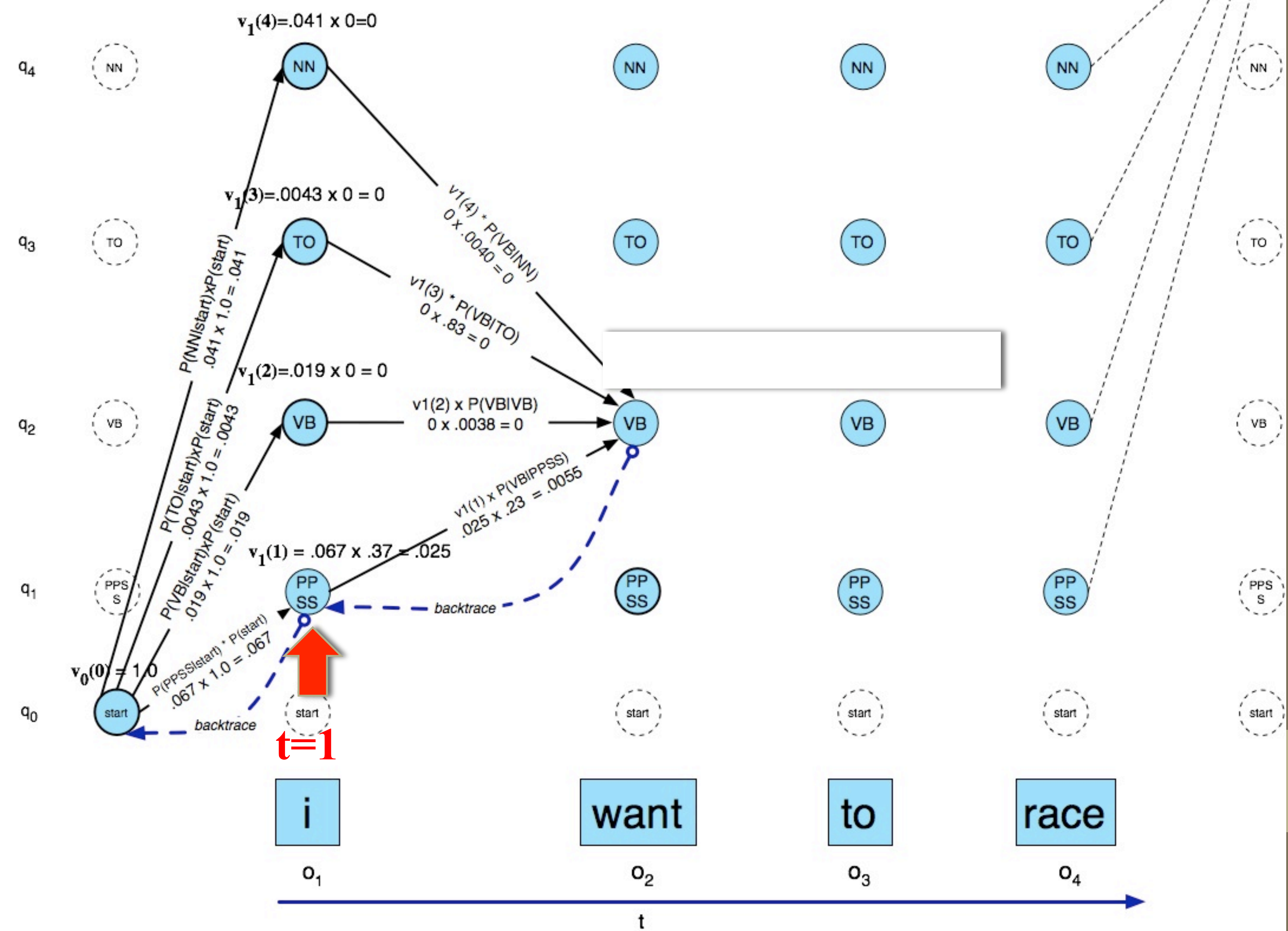
# Problem

- I want to race  (possible states: PPS VB TO NN)

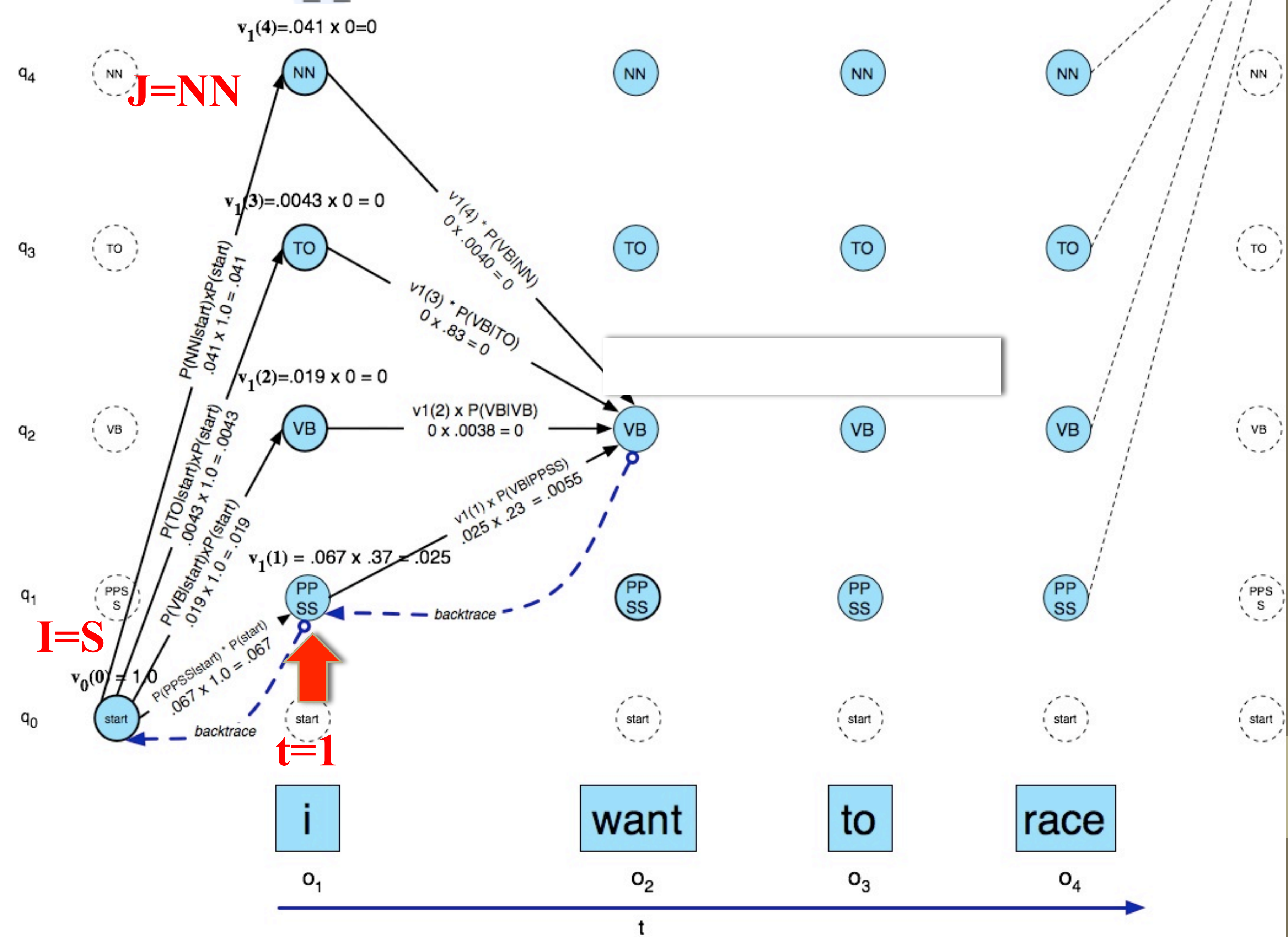
$$v_t(j) = \max_{1<i<N-1} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$



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



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



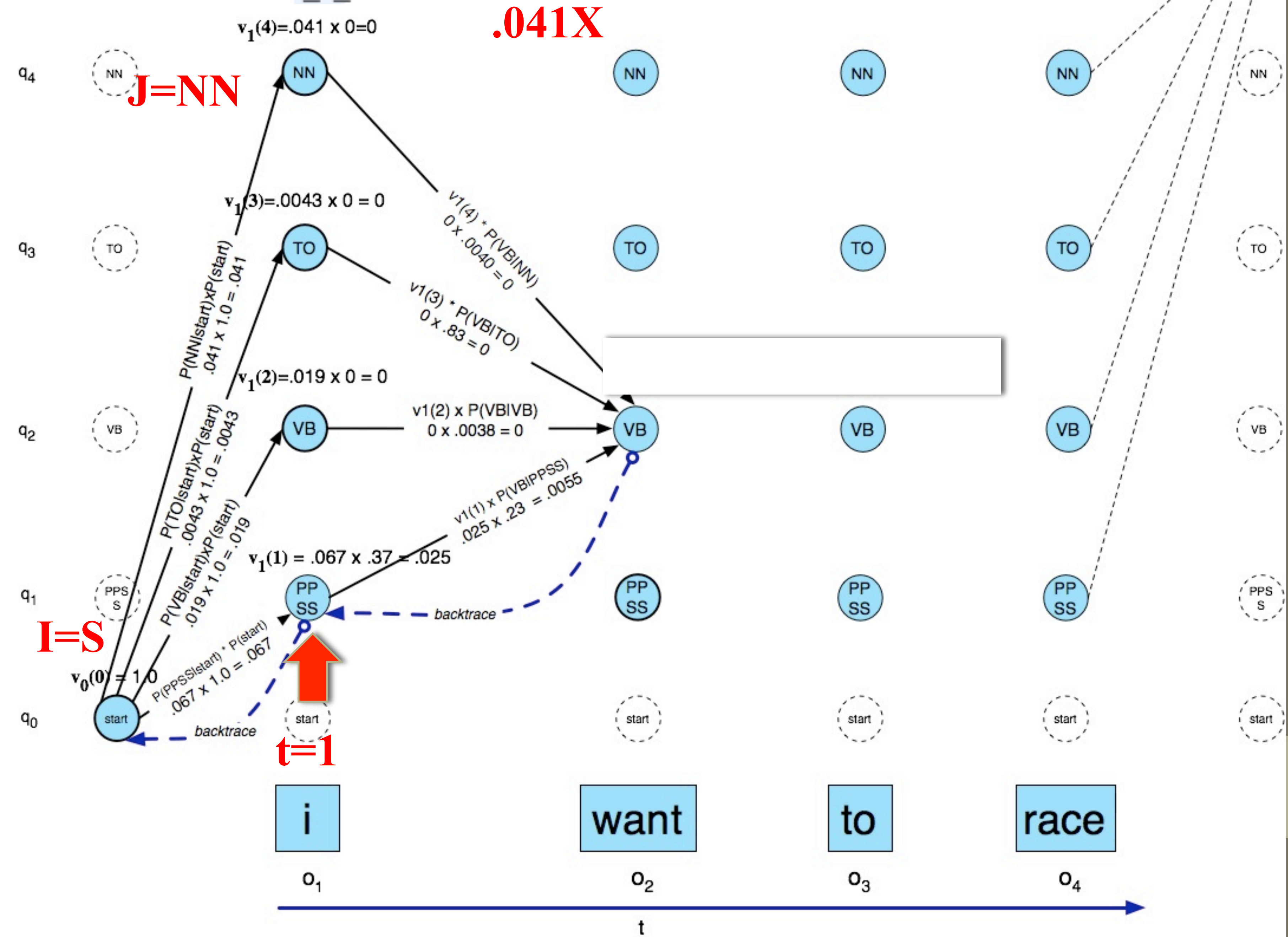
# The A matrix for the POS HMM

| | 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.

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

X



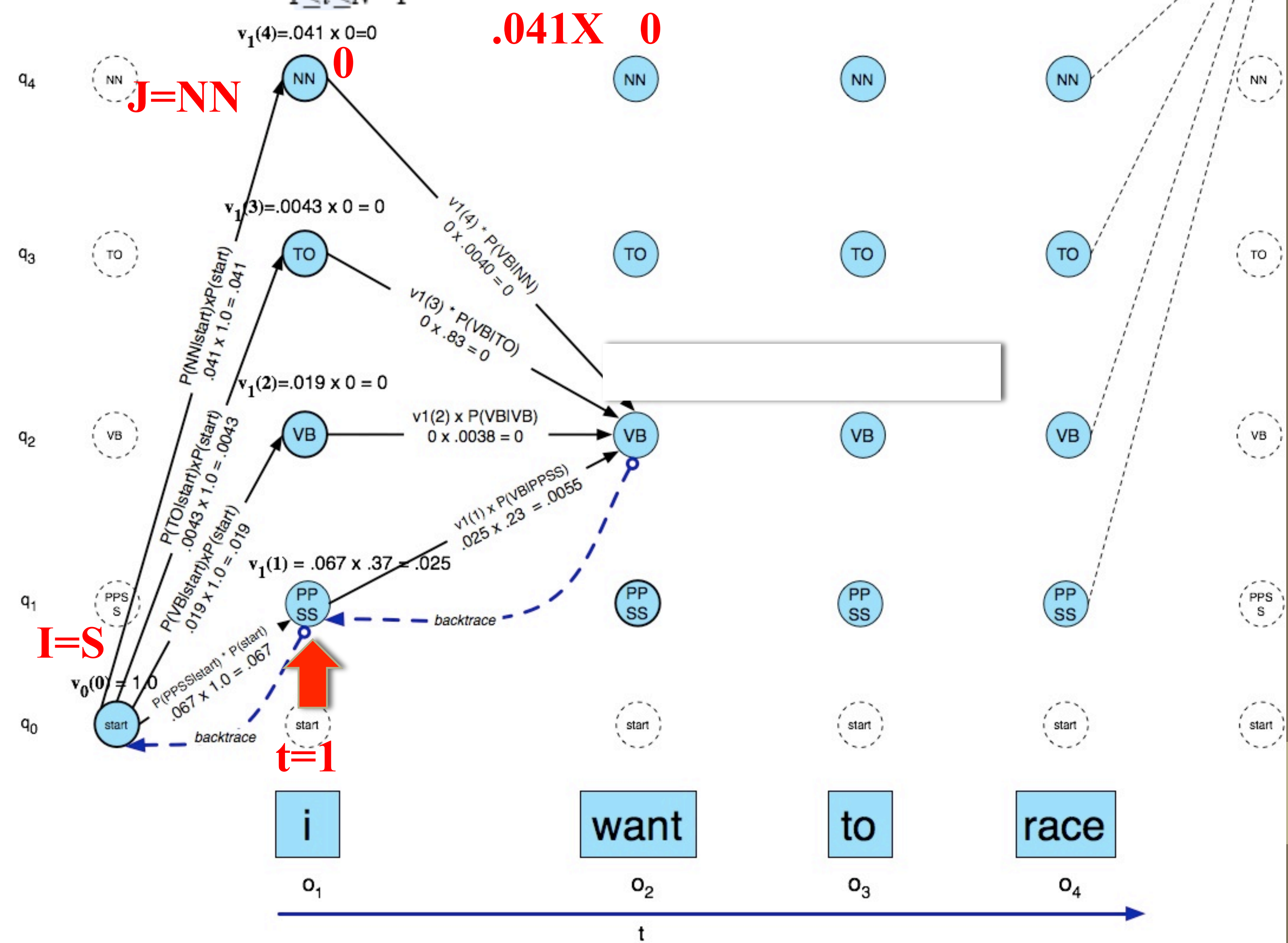
# The B matrix for the POS HMM

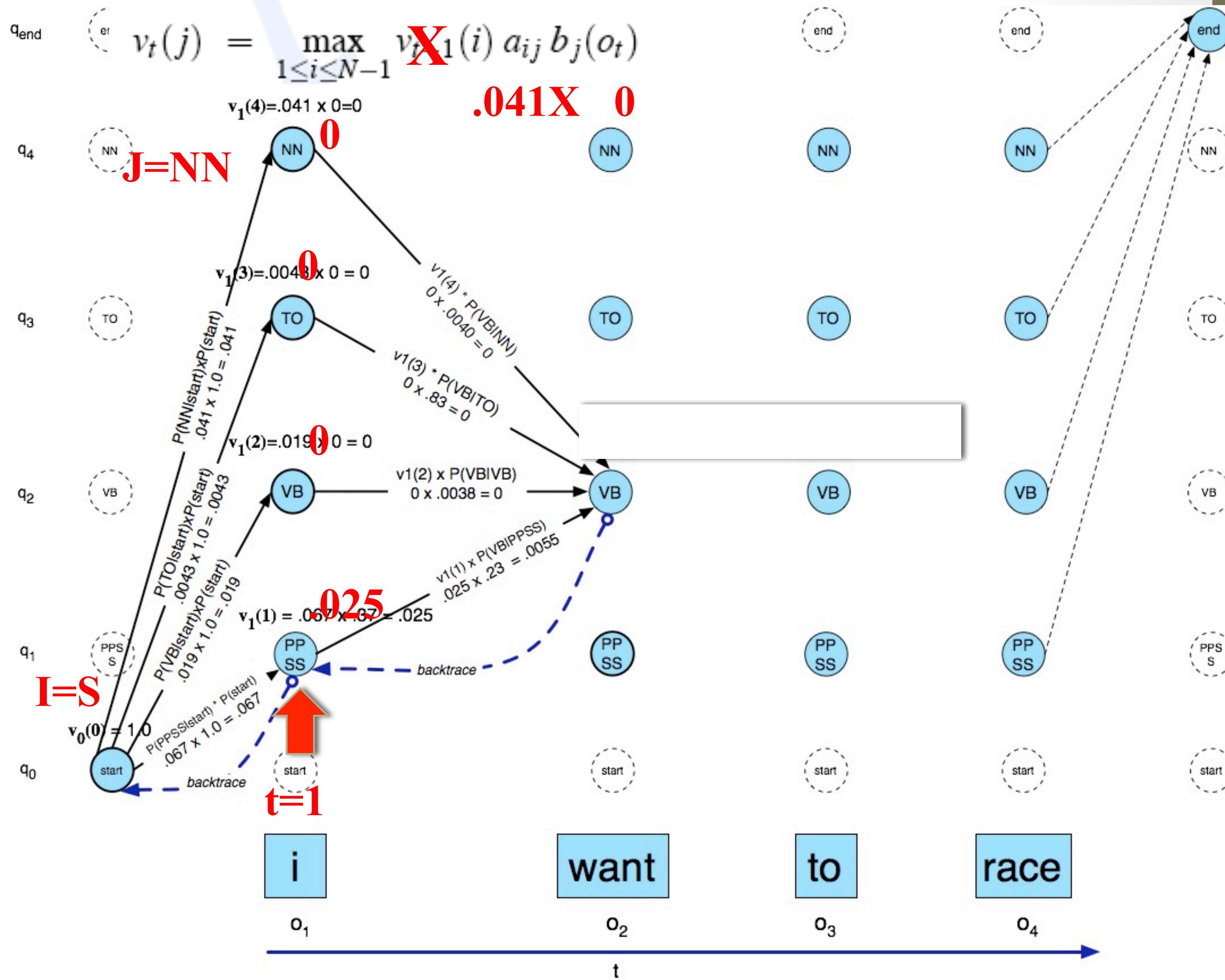
| | I | want | to | race |
|---|---|---|---|---|
| **VB** | 0 | .0093 | 0 | .00012 |
| **TO** | 0 | 0 | .99 | 0 |
| **NN** | 0 | .000054 | 0 | .00057 |
| **PPSS** | .37 | 0 | 0 | 0 |

**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.

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

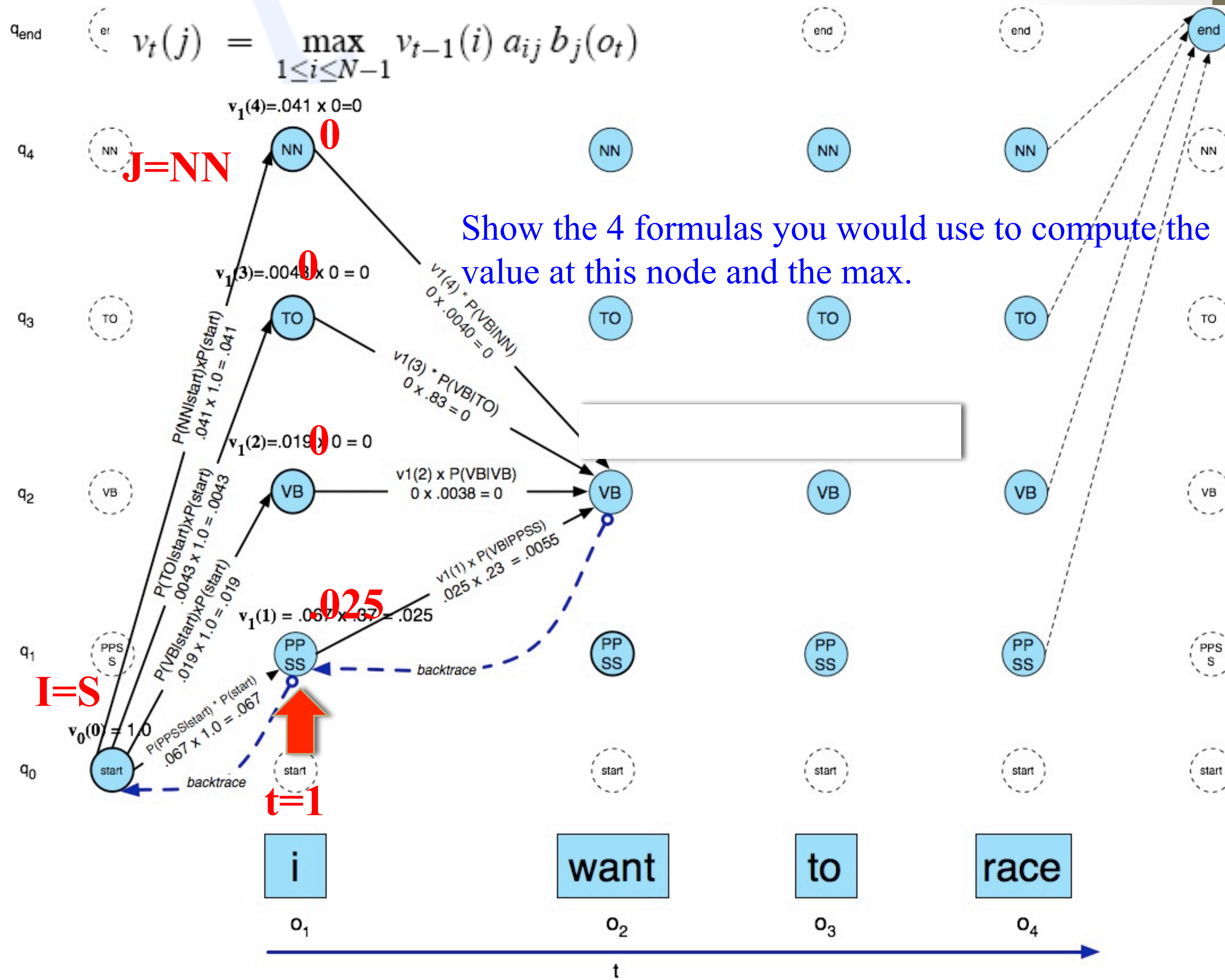




$v_t(j) = \max_{1 \leq i \leq N-1} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$
X
.041X 0
J=NN
I=S
t=1
v1(4)=.041 x 0=0
0
v1(3)=.0043 x 0 = 0
0
v1(2)=.019 x 0 = 0
0
v1(1) = .067 x .37 = .025
.025
v0(0) = 1.0
P(NN|start)xP(start) .041 x 1.0 = .041
P(TO|start)xP(start) .0043 x 1.0 = .0043
P(VB|start)xP(start) .019 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
q0
q1
q2
q3
q4
qend
start
PPSS
VB
TO
NN
end
i
want
to
race
o1
o2
o3
o4
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.



# Early Algorithm

# Earley Algorithm

- March through chart left-to-right.
- At each step, apply 1 of 3 operators
  - Predictor
    - Create new states representing top-down expectations
  - Scanner
    - Match word predictions (rule with word after dot) to words
  - Completer
    - When a state is complete, see what rules were looking for that completed constituent

# Predictor

- Given a state
  - With a non-terminal to right of dot (not a part-of-speech category)
  - Create a new state for each expansion of the non-terminal
  - Place these new states into same chart entry as generated state, beginning and ending where generating state ends.
  - So predictor looking at
    - S -> . VP [0,0]
  - results in
    - VP -> . Verb [0,0]
    - VP -> . Verb NP [0,0]

# Scanner

- Given a state
  - With a non-terminal to right of dot that is a part-of-speech category
  - If the next word in the input matches this POS
  - Create a new state with dot moved over the non-terminal
  - So scanner looking at VP -> . Verb NP [0,0]
  - If the next word, "book", can be a verb, add new state:
    - VP -> Verb . NP [0,1]
  - Add this state to chart entry following current one
  - Note: Earley algorithm uses top-down input to disambiguate POS! Only POS predicted by some state can get added to chart!

# Completer

- Applied to a state when its dot has reached right end of role.
- Parser has discovered a category over some span of input.
- Find and advance all previous states that were looking for this category
  - copy state, move dot, insert in current chart entry
- Given:
  - NP -> Det Nominal . [1,3]
  - VP -> Verb. NP [0,1]
- Add
  - VP -> Verb NP . [0,3]

# How do we know we are done?

- Find an S state in the final column that spans from 0 to n+1 and is complete.

- If that's the case you're done.
  - S –> $\alpha \cdot$ [0,n+1]

# Earley

- More specifically…

1. Predict all the states you can upfront
2. Read a word
   1. Extend states based on matches
   2. Add new predictions
   3. Go to 2
3. Look at N+1 to see if you have a winner

# Example

- Book that flight
- We should find… an S from 0 to 3 that is a completed state…

# CFG for Fragment of English

| | |
|---|---|
| S → NP VP | VP → V |
| S → Aux NP VP | PP -> Prep NP |
| NP → Det Nom | N → old \| dog \| footsteps \| young \| flight |
| NP →PropN | V → dog \| include \| prefer \| book |
| Nom -> Adj Nom | Aux → does |
| Nom → N | Prep →from \| to \| on \| of |
| Nom → N Nom | PropN → Bush \| McCain \| Obama |
| Nom → Nom PP | Det → that \| this \| a\| the |
| VP → V NP | Adj -> old \| green \| red |

| | |
|---|---|
| S → NP VP, *S -> VP* | VP → V |
| S → Aux NP VP | PP -> Prep NP |
| NP → Det Nom | N → old \| dog \| footsteps \| young \| *flight* |
| NP →PropN, NP -> Pro | V → dog \| include \| prefer \| *book* |
| | Aux → does |
| Nom → N | Prep →from \| to \| on \| of |
| Nom → N Nom | PropN → Bush \| McCain \| Obama |
| Nom → Nom PP | Det → *that* \| this \| a\| the |
| VP → V NP, VP -> V NP PP, VP -> V PP, VP -> VP PP | Adj -> old \| green \| red |

| | |
|---|---|
| S → NP VP, *S -> VP* | VP → V |
| S → Aux NP VP | PP -> Prep NP |
| NP → Det Nom | N → old \| dog \| footsteps \| young \| *flight* |
| NP →PropN, NP -> Pro | V → dog \| include \| prefer \| *book* |
| | Aux → does |
| Nom → N | Prep →from \| to \| on \| of |
| Nom → N Nom | PropN → Bush \| McCain \| Obama |
| Nom → Nom PP | Det → *that* \| this \| a\| the |
| VP → V NP, VP -> V NP PP, VP -> V PP, VP -> VP PP | Adj -> old \| green \| red |

# Example

| | | | | |
|---|---|---|---|---|
| Chart[0] | S0 | $\gamma \rightarrow \bullet S$ | [0,0] | Dummy start state |
| | S1 | $S \rightarrow \bullet NP\ VP$ | [0,0] | Predictor |
| | S2 | $S \rightarrow \bullet Aux\ NP\ VP$ | [0,0] | Predictor |
| | S3 | $S \rightarrow \bullet VP$ | [0,0] | Predictor |
| | S4 | $NP \rightarrow \bullet Pronoun$ | [0,0] | Predictor |
| | S5 | $NP \rightarrow \bullet Proper\text{-}Noun$ | [0,0] | Predictor |
| | S6 | $NP \rightarrow \bullet Det\ Nominal$ | [0,0] | Predictor |
| | S7 | $VP \rightarrow \bullet Verb$ | [0,0] | Predictor |
| | S8 | $VP \rightarrow \bullet Verb\ NP$ | [0,0] | Predictor |
| | S9 | $VP \rightarrow \bullet Verb\ NP\ PP$ | [0,0] | Predictor |
| | S10 | $VP \rightarrow \bullet Verb\ PP$ | [0,0] | Predictor |
| | S11 | $VP \rightarrow \bullet VP\ PP$ | [0,0] | Predictor |

# Example

| | | | | |
|---|---|---|---|---|
| Chart[1] | S12 | $Verb \rightarrow book \bullet$ | [0,1] | Scanner |
| | S13 | $VP \rightarrow Verb \bullet$ | [0,1] | Completer |
| | S14 | $VP \rightarrow Verb \bullet NP$ | [0,1] | Completer |
| | S15 | $VP \rightarrow Verb \bullet NP\ PP$ | [0,0] | Completer |
| | S16 | $VP \rightarrow Verb \bullet PP$ | [0,0] | Predictor |
| | S17 | $S \rightarrow VP \bullet$ | [0,1] | Completer |
| | S18 | $VP \rightarrow VP \bullet PP$ | [0,1] | Completer |
| | S19 | $NP \rightarrow \bullet Pronoun$ | [1,1] | Predictor |
| | S20 | $NP \rightarrow \bullet Proper\text{-}Noun$ | [1,1] | Predictor |
| | S21 | $NP \rightarrow \bullet Det\ Nominal$ | [1,1] | Predictor |
| | S22 | $PP \rightarrow \bullet Prep\ NP$ | [1,1] | Predictor |

# Example

| | | | | |
|---|---|---|---|---|
| Chart[1] | S12 | $Verb \rightarrow book \bullet$ | [0,1] | Scanner |
| | S13 | $VP \rightarrow Verb \bullet$ | [0,1] | Completer |
| | S14 | $VP \rightarrow Verb \bullet NP$ | [0,1] | Completer |
| | S15 | $VP \rightarrow Verb \bullet NP\ PP$ | [0,0] | Completer |
| | S16 | $VP \rightarrow Verb \bullet PP$ | [0,0] | Predictor |
| | S17 | $S \rightarrow VP \bullet$ | [0,1] | Completer |
| | S18 | $VP \rightarrow VP \bullet PP$ | [0,1] | Completer |
| | S19 | $NP \rightarrow \bullet Pronoun$ | [1,1] | Predictor |
| | S20 | $NP \rightarrow \bullet Proper\text{-}Noun$ | [1,1] | Predictor |
| | S21 | $NP \rightarrow \bullet Det\ Nominal$ | [1,1] | Predictor |
| | S22 | $PP \rightarrow \bullet Prep\ NP$ | [1,1] | Predictor |

# Example

| | | | | | |
|---|---|---|---|---|---|
| Chart[2] | S23 | $Det \rightarrow that \bullet$ | [1,2] | Scanner |
| | S24 | $NP \rightarrow Det \bullet Nominal$ | [1,2] | Completer |
| | S25 | $Nominal \rightarrow \bullet Noun$ | [2,2] | Predictor |
| | S26 | $Nominal \rightarrow \bullet Nominal\ Noun$ | [2,2] | Predictor |
| | S27 | $Nominal \rightarrow \bullet Nominal\ PP$ | [2,2] | Predictor |
| Chart[3] | S28 | $Noun \rightarrow flight \bullet$ | [2,3] | Scanner |
| | S29 | $Nominal \rightarrow Noun \bullet$ | [2,3] | Completer |
| | S30 | $NP \rightarrow Det\ Nominal \bullet$ | [1,3] | Completer |
| | S31 | $Nominal \rightarrow Nominal \bullet Noun$ | [2,3] | Completer |
| | S32 | $Nominal \rightarrow Nominal \bullet PP$ | [2,3] | Completer |
| | S33 | $VP \rightarrow Verb\ NP \bullet$ | [0,3] | Completer |
| | S34 | $VP \rightarrow Verb\ NP \bullet PP$ | [0,3] | Completer |
| | S35 | $PP \rightarrow \bullet Prep\ NP$ | [3,3] | Predictor |
| | S36 | $S \rightarrow VP \bullet$ | [0,3] | Completer |

# Details

- What kind of algorithms did we just describe
  - Not parsers – recognizers
    - The presence of an S state with the right attributes in the right place indicates a successful recognition.
    - But no parse tree… no parser
    - That's how we solve (not) an exponential problem in polynomial time

# Converting Earley from Recognizer to Parser

- With the addition of a few pointers we have a parser
- Augment the “Completer” to point to where we came from.

# Augmenting the chart with structural information

Chart[1]

| | | | | | |
|---|---|---|---|---|---|
| S8 | $Verb \rightarrow book\bullet$ | [0,1] | Scanner | |
| S9 | $VP \rightarrow Verb\bullet$ | [0,1] | Completer | S8 |
| S10 | $S \rightarrow VP\bullet$ | [0,1] | Completer | S9 |
| S11 | $VP \rightarrow Verb \bullet NP$ | [0,1] | Completer | S8 |
| S12 | $NP \rightarrow \bullet\, Det\ NOMINAL$ | [1,1] | Predictor | |
| S13 | $NP \rightarrow \bullet\, Proper\text{-}Noun$ | [1,1] | Predictor | |

Chart[2]

| | | |
|---|---|---|
| $Det \rightarrow that\bullet$ | [1,2] | Scanner |
| $NP \rightarrow Det\bullet NOMINAL$ | [1,2] | Completer |
| $NOMINAL \rightarrow \bullet\, Noun$ | [2,2] | Predictor |
| $NOMINAL \rightarrow \bullet\, Noun\ NOMINAL$ | [2,2] | Predictor |