HMM Inference

- Decoding: most likely sequence of hidden states
  - Viterbi algorithm
- Evaluation: prob. of observing an obs. sequence
  - Forward Algorithm (very similar to Viterbi)
- Marginal distribution: prob. of a particular state
  - Forward-Backward

Decoding Problem

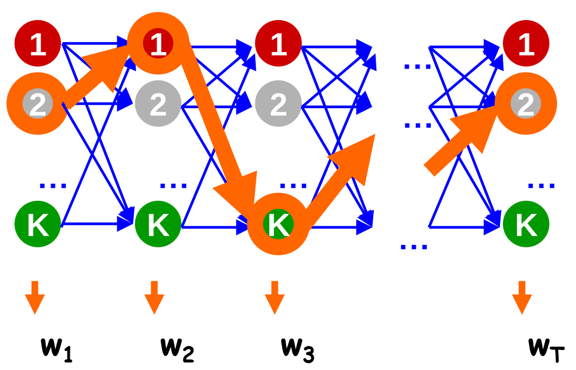
Given $w=w_1 \dots w_T$ and HMM θ, what is "best" parse $y_1 \dots y_T$?

Several possible meanings of 'solution'

1. States which are individually most likely
2. Single best state sequence

We want ***sequence*** $y_1 \dots y_T$,
such that $P(y|w)$ is maximized

$$y^* = \text{argmax}_y \, P(\, y|w \,)$$



Most Likely Sequence

- Problem: find the most likely (Viterbi) sequence under the model
  - Given model parameters, we can score any sequence pair

NNP	VBZ	NN	NNS	CD	NN	.
Fed	raises	interest	rates	0.5	percent	.

P($\mathbf{Y}_{1:T+1}$,$\mathbf{W}_{1:T}$) = q(NNP|<s>,<s>) q(Fed|NNP) P(VBZ|<s>,NNP) P(raises|VBZ) P(NN|NNP,VBZ)…..

  - In principle, we're done – list all possible tag sequences, score each one, pick the best one (the Viterbi state sequence)

2T+1 operations per sequence

NNP VBZ NN NNS CD NN ⇨ logP = -23

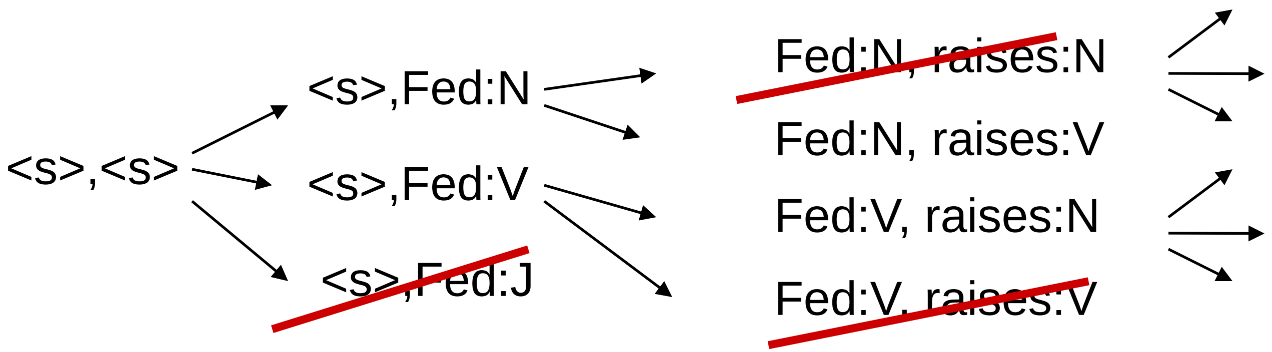
NNP NNS NN NNS CD NN ⇨ logP = -29

NNP VBZ VB NNS CD NN ⇨ logP = -27

$|Y|^T$ tag sequences!

Finding the Best Trajectory

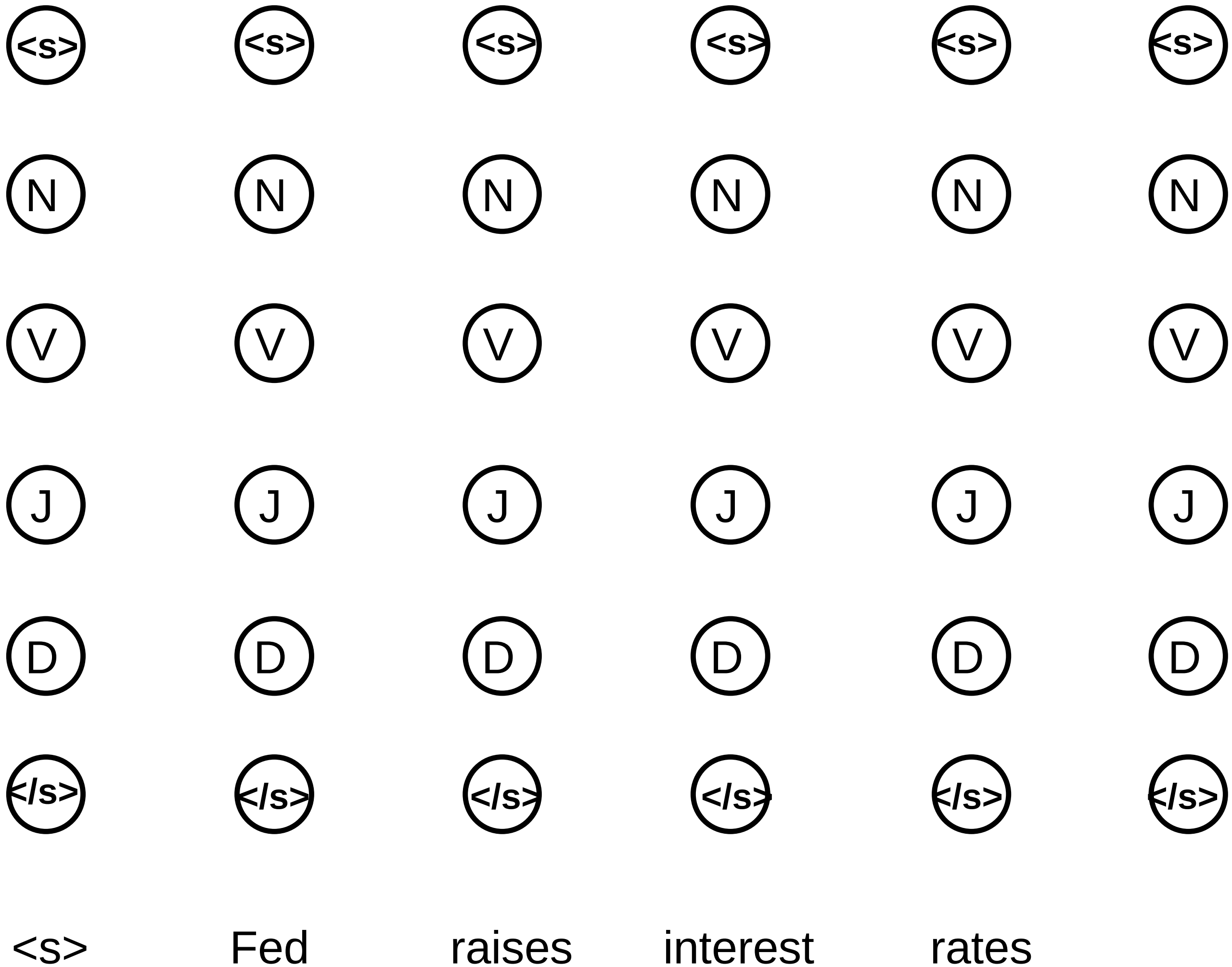
- Brute Force: Too many trajectories (state sequences) to list
- Option 1: Beam Search



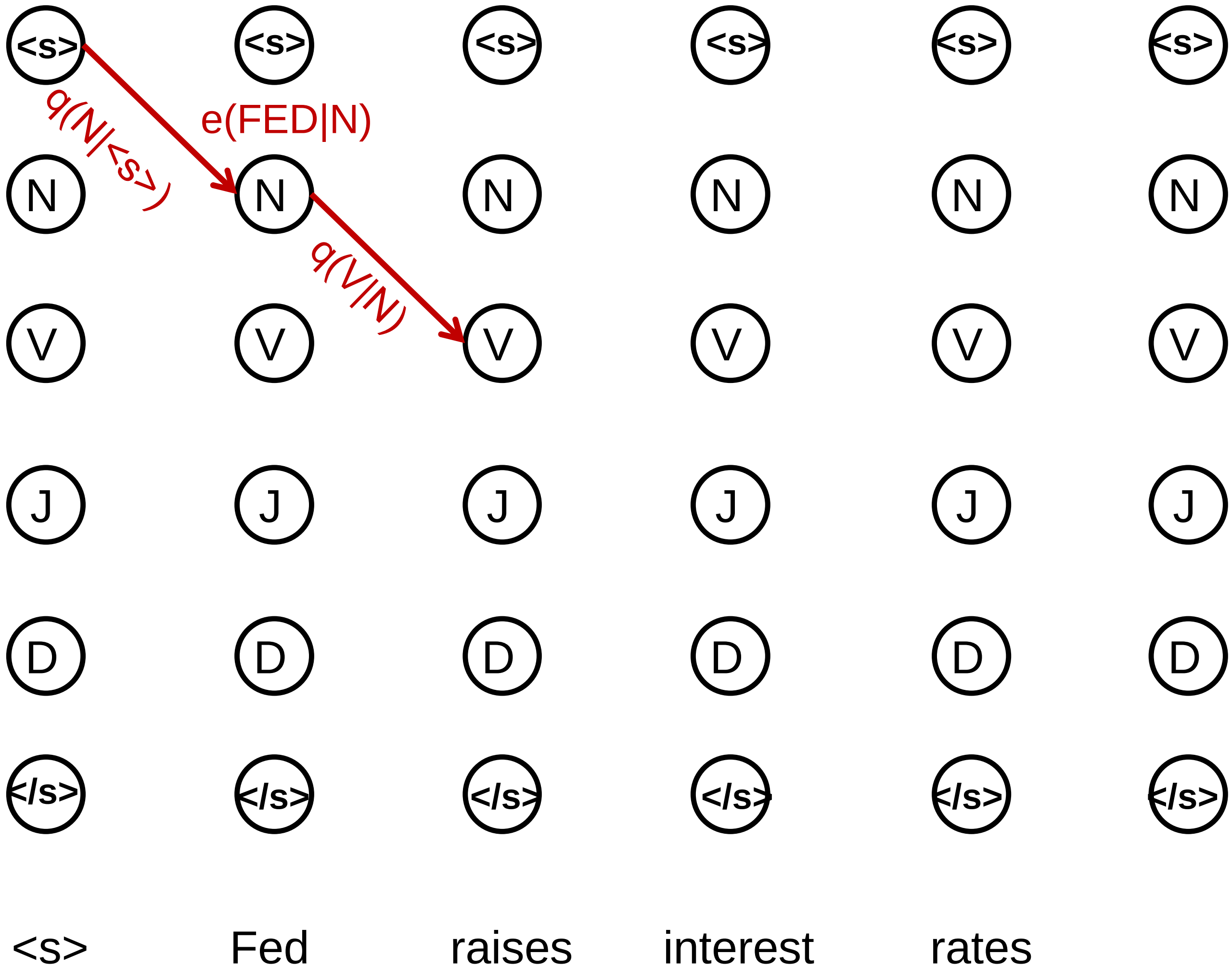
  - A beam is a set of partial hypotheses
  - Start with just the single empty trajectory
  - At each derivation step:
    - Consider all continuations of previous hypotheses
    - Discard most, keep top k

- Beam search works ok in practice
  - … but sometimes you want the optimal answer
  - … and there's often a better option than naïve beams

State Lattice / Trellis (Bigram HMM)



State Lattice / Trellis (Bigram HMM)



Dynamic Programming (Bigram)

- Decoding: $$\vec{y}^* = \arg\max_{\vec{y}} P(\vec{y} \mid \vec{w}) = \arg\max_{\vec{y}} P(\vec{w}, \vec{y})$$
$$= \arg\max_{\vec{y}} \prod_{t=1}^{T+1} q(y_t \mid y_{t-1}) \prod_{t=1}^{T} e(w_t \mid y_t)$$
- First consider how to compute max
- Define $$\delta_i(y_i) = \max_{y[1:i-1]} P(y_{[1..i]}, w_{[1..i]})$$
  - probability of ***most likely*** state sequence ending with tag y_i, given observations $w_1, \ldots, w_i$

$$\delta_i(y_i) = \max_{y[1:i-1]} e(w_i \mid y_i) q(y_i \mid y_{i-1}) P(y_{[1..i-1]}, w_{[1..i-1]})$$
$$= e(w_i \mid y_i) \max_{y_{i-1}} q(y_i \mid y_{i-1}) \max_{y[1:i-2]} P(y_{[1..i-1]}, w_{[1..i-1]})$$
$$= e(w_i \mid y_i) \max_{y_{i-1}} q(y_i \mid y_{i-1}) \delta_{i-1}(y_{i-1})$$

Viterbi Algorithm for Bigram HMMs

- Input: $w_1,\ldots,w_T$, model parameters q() and e()
- Initialize: $\delta_0(<s>) = 1$
- For k=1 to T do
  - For (y') in all possible tagset

$$\delta_i(y') = e(w_i \mid y') \max_{y} q(y' \mid y)\delta_{i-1}(y)$$

- Return

$$\max_{y'} q(</s> \mid y')\delta_T(y')$$

returns only the optimal value

keep backpointers

Viterbi Algorithm for Bigram HMMs

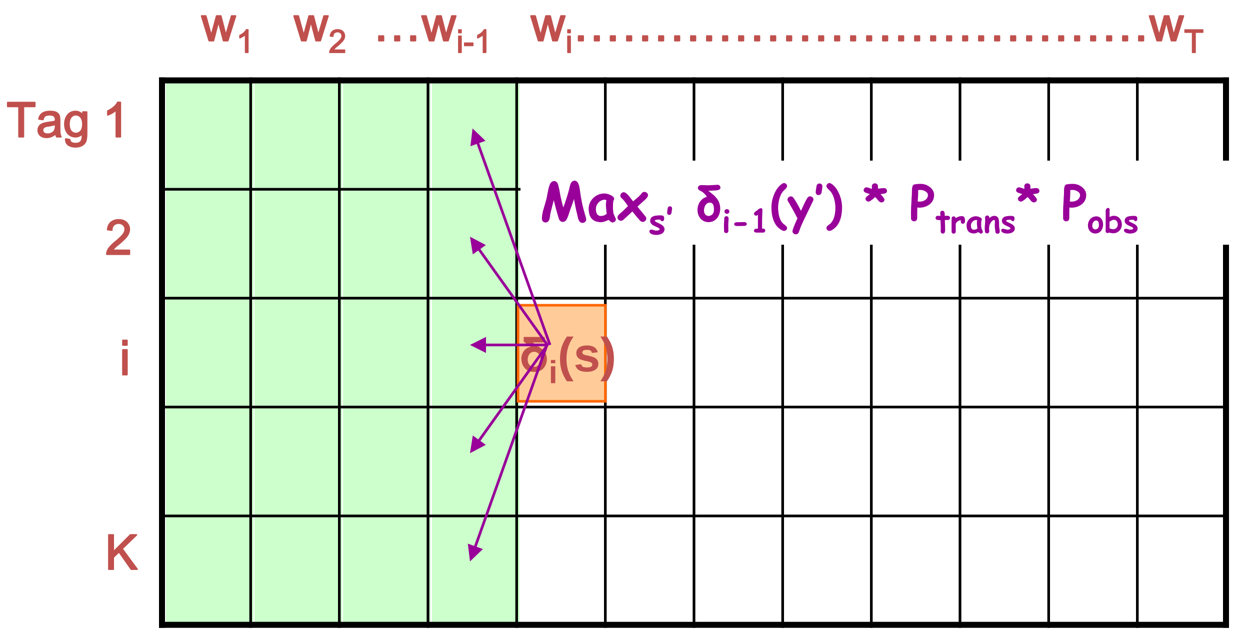
- Input: $w_1,\ldots,w_T$, model parameters q() and e()
- Initialize: $\delta_0(\text{<s>},\text{<s>}) = 1$
- For k=1 to T do
  - For (y') in all possible tagset

$$\delta_i(y') = e(w_i \mid y') \max_{y} q(y' \mid y)\delta_{i-1}(y)$$
$$bp_i(y') = e(w_i \mid y') \arg\max_{y} q(y' \mid y)\delta_{i-1}(y)$$

- Set $y_T = \arg\max_{y'} q(</s> \mid y')\delta_T(y')$
- For k=T-1 to 1 do
  - Set $y_k = bp_k(y_{k+1})$
- Return y[1..T]

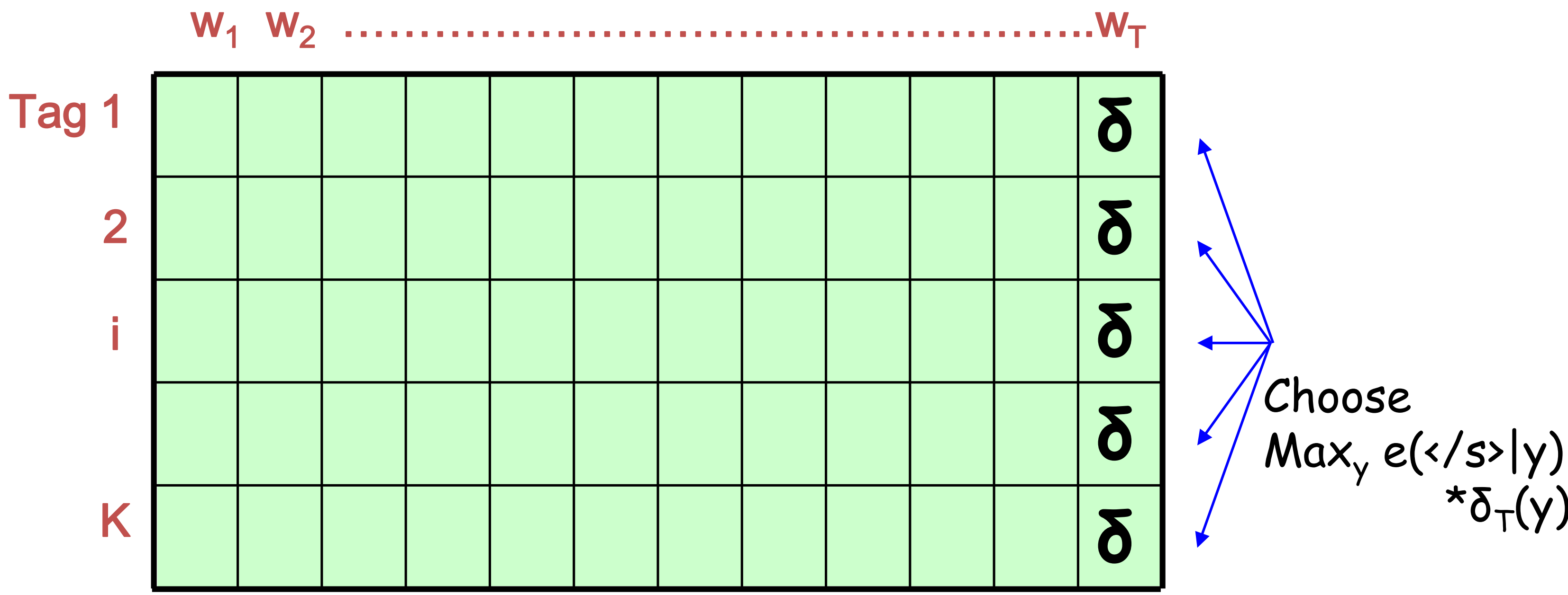
Time: $O(|Y|^2T)$
Space: $O(|Y|T)$

Viterbi Algorithm for Bigram HMMs

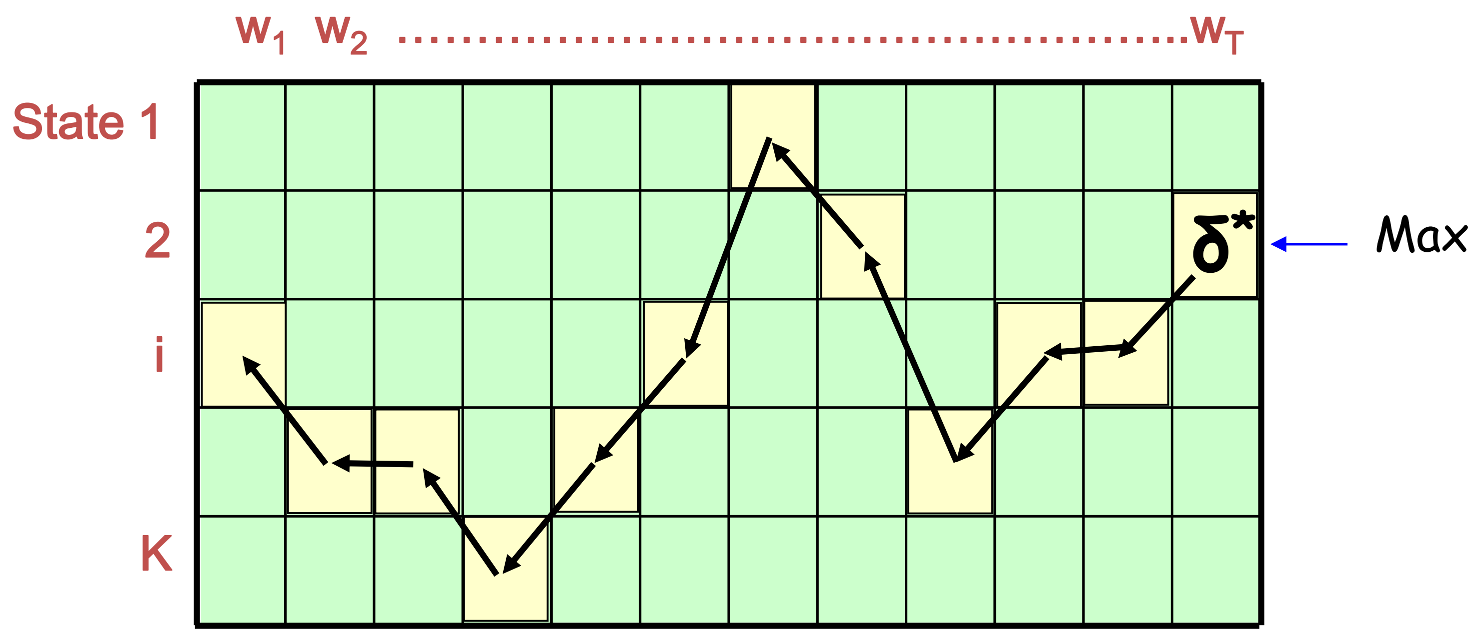


Remember: $\delta_i(y)$ = probability of most likely tag seq ending with y at time i

Terminating Viterbi



Terminating Viterbi



How did we compute δ^*? $\quad Max_{s'}\ \delta_{T-1}(y') * P_{trans} * P_{obs}$

Now Backchain to Find Final Sequence

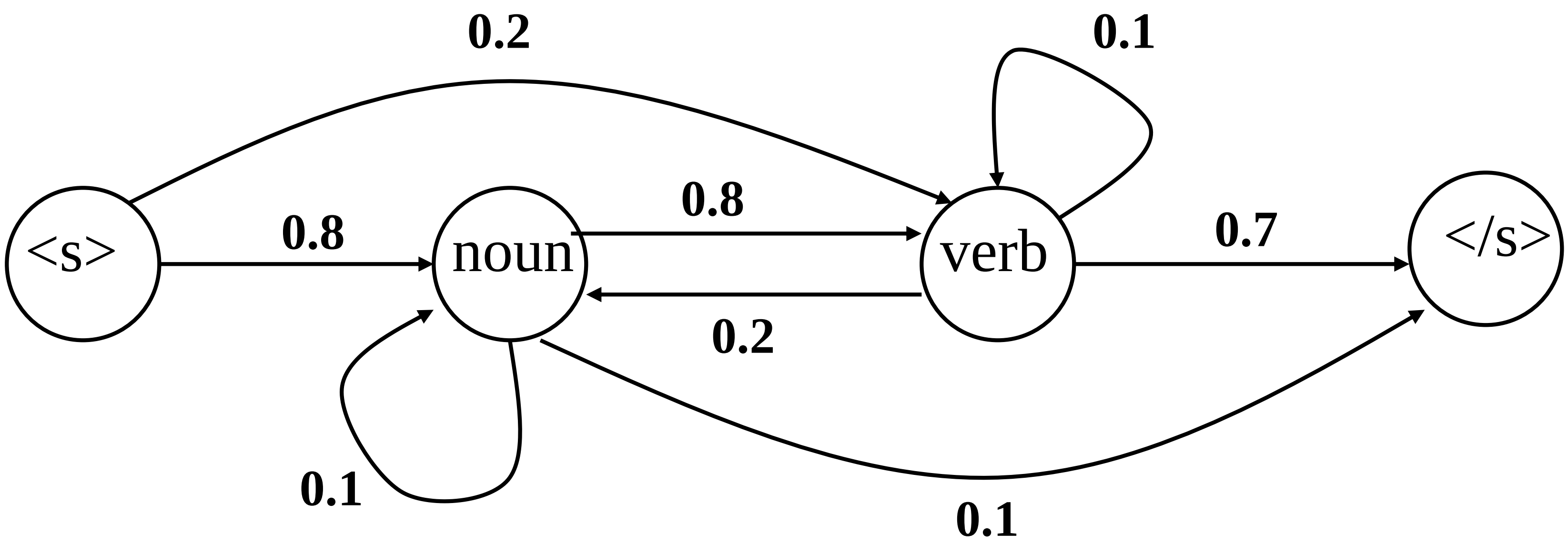
Time: $O(|Y|^2 T)$

Space: $O(|Y| T)$ ← Linear in length of sequence

Example

Fish sleep.

Example: Bigram HMM

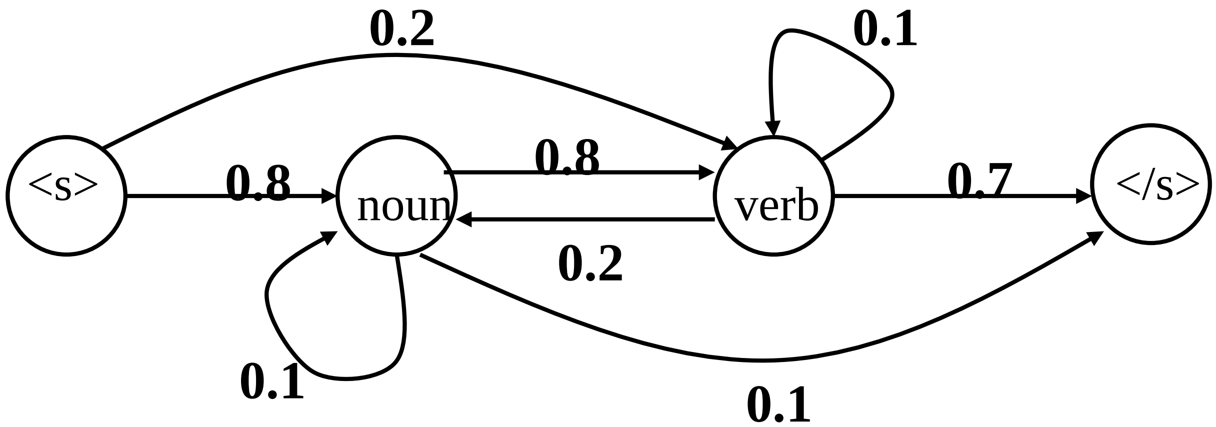


Data

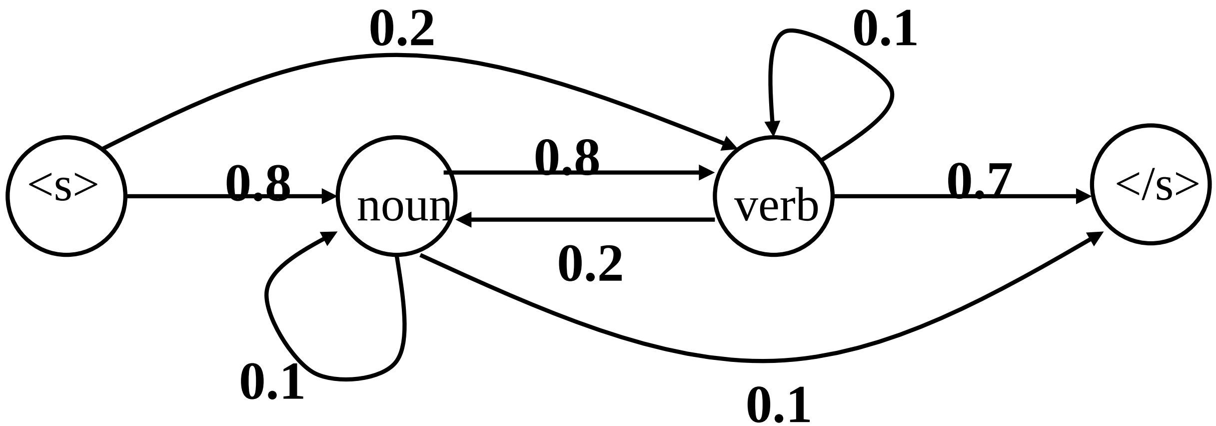
- A two-word language: "fish" and "sleep"
- Suppose in our training corpus,
  - "fish" appears 8 times as a noun and 5 times as a verb
  - "sleep" appears twice as a noun and 5 times as a verb
- Emission probabilities:
  - Noun
    - P(fish | noun) : 0.8
    - P(sleep | noun) : 0.2
  - Verb
    - P(fish | verb) : 0.5
    - P(sleep | verb) : 0.5

Viterbi Probabilities

	0	1	2	3
start				
verb				
noun				
end				

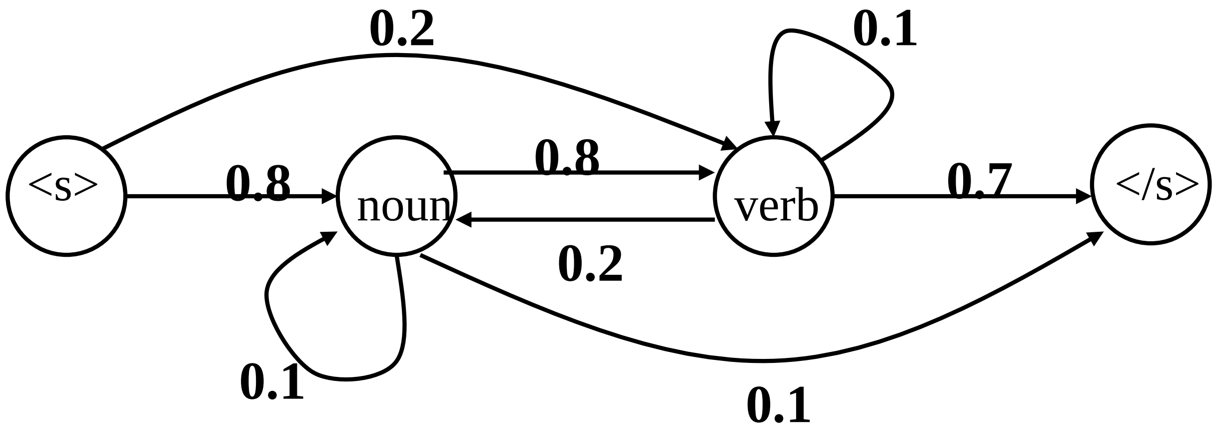


	0	1	2	3
start	1			
verb	0			
noun	0			
end	0			



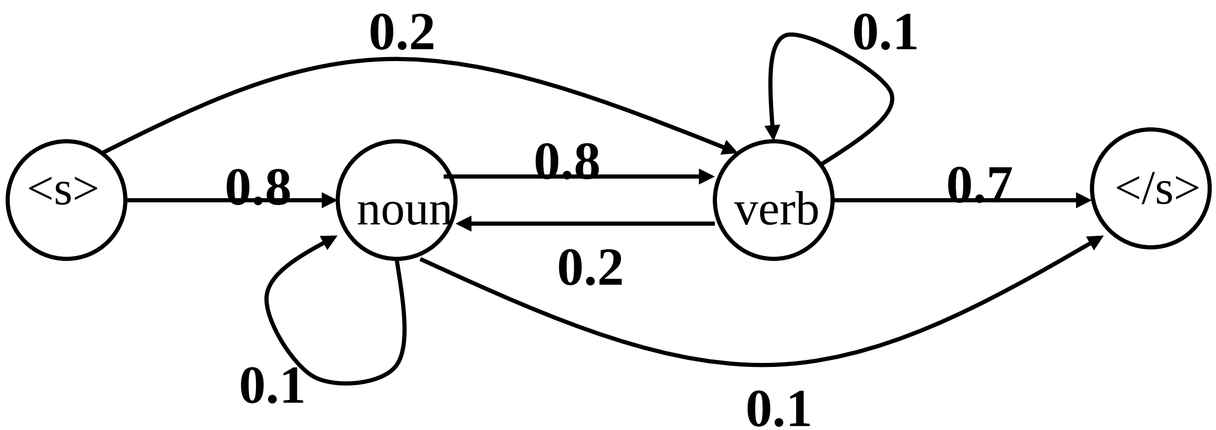
Token 1: fish

	0	1	2	3
start	1	0		
verb	0	.2 * .5		
noun	0	.8 * .8		
end	0	0		



Token 1: fish

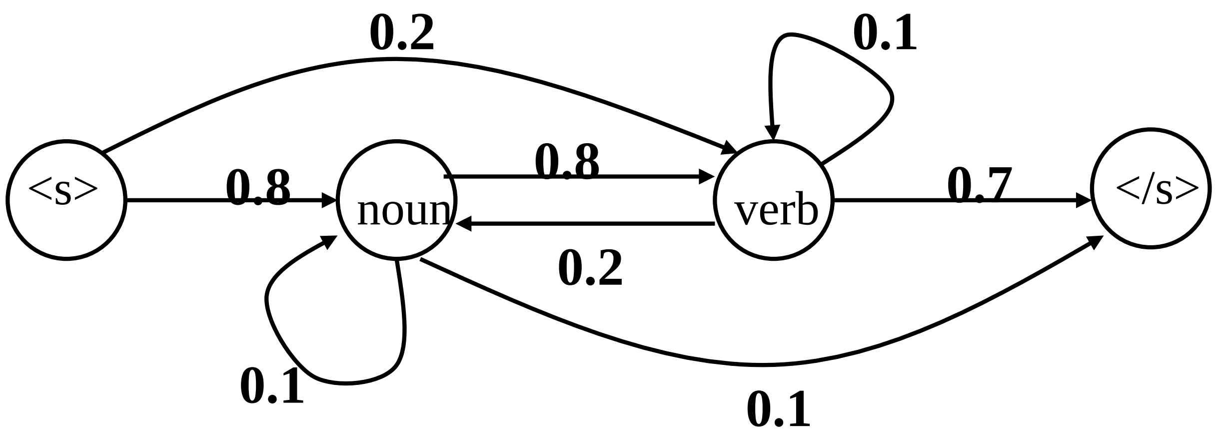
	0	1	2	3
start	1	0		
verb	0	.1		
noun	0	.64		
end	0	0		



Token 2: sleep

(if ‘fish’ is verb)

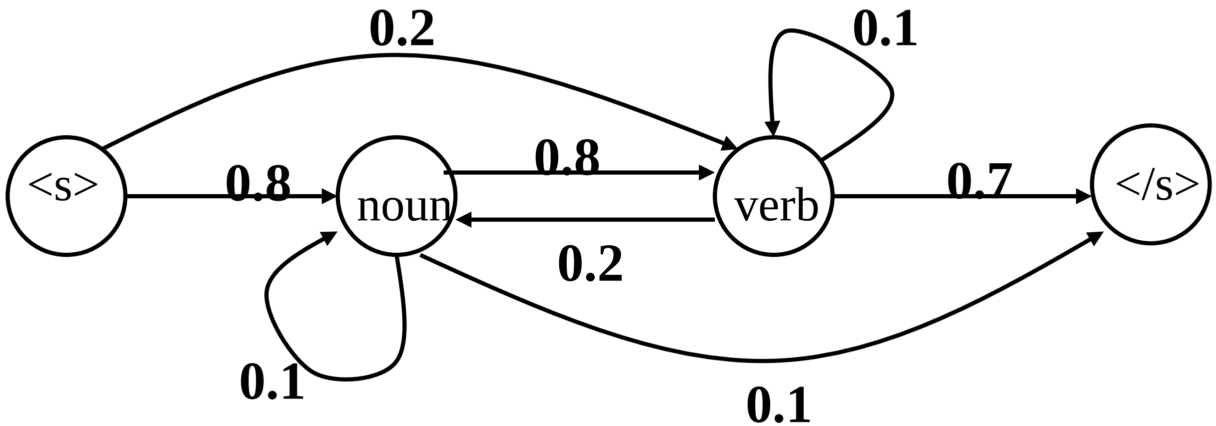
	0	1	2	3
start	1	0	0	
verb	0	.1	.1*.1*.5	
noun	0	.64	.1*.2*.2	
end	0	0	-	



Token 2: sleep

(if ‘fish’ is verb)

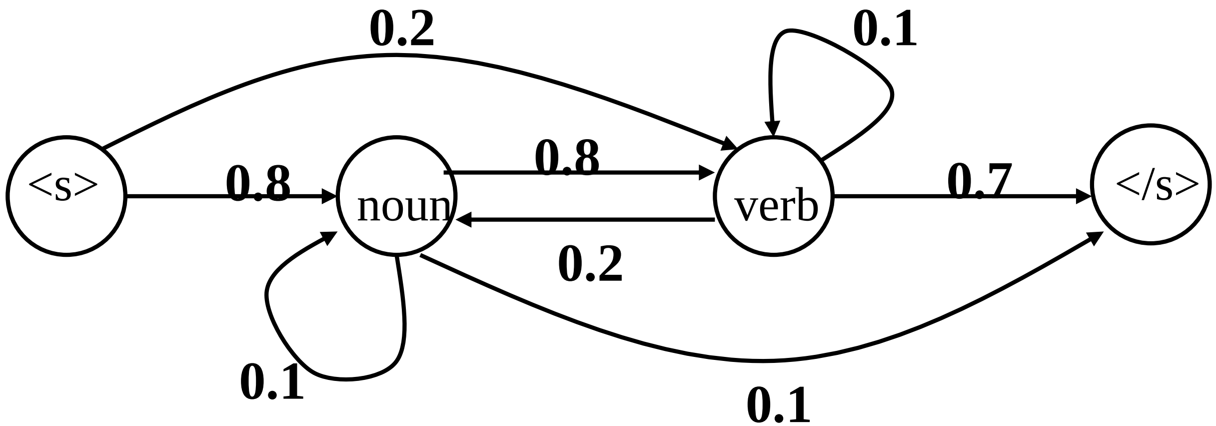
	0	1	2	3
start	1	0	0	
verb	0	.1	.005	
noun	0	.64	.004	
end	0	0	-	



Token 2: sleep

(if ‘fish’ is a noun)

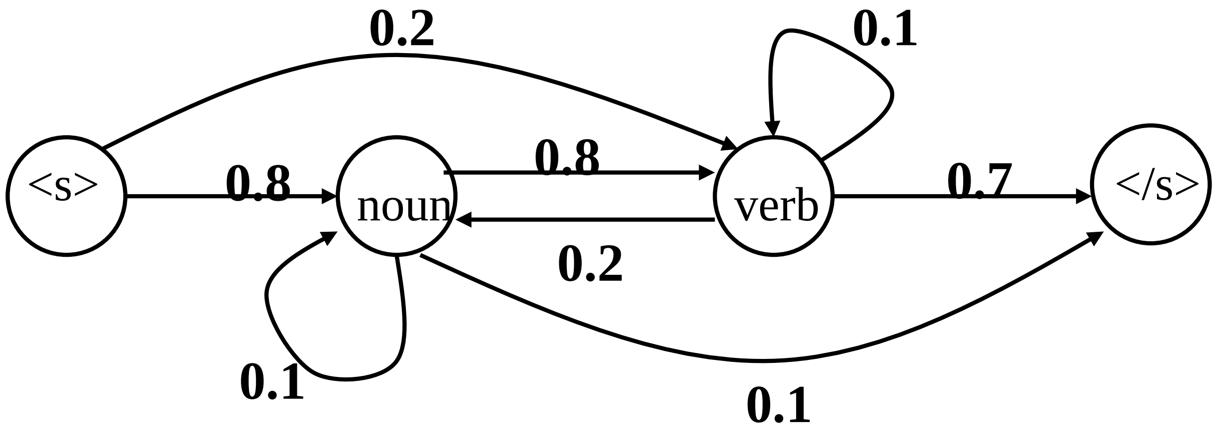
	0	1	2	3
start	1	0	0	
verb	0	.1	.005 .64*.8*.5	
noun	0	.64	.004 .64*.1*.2	
end	0	0	-	



Token 2: sleep

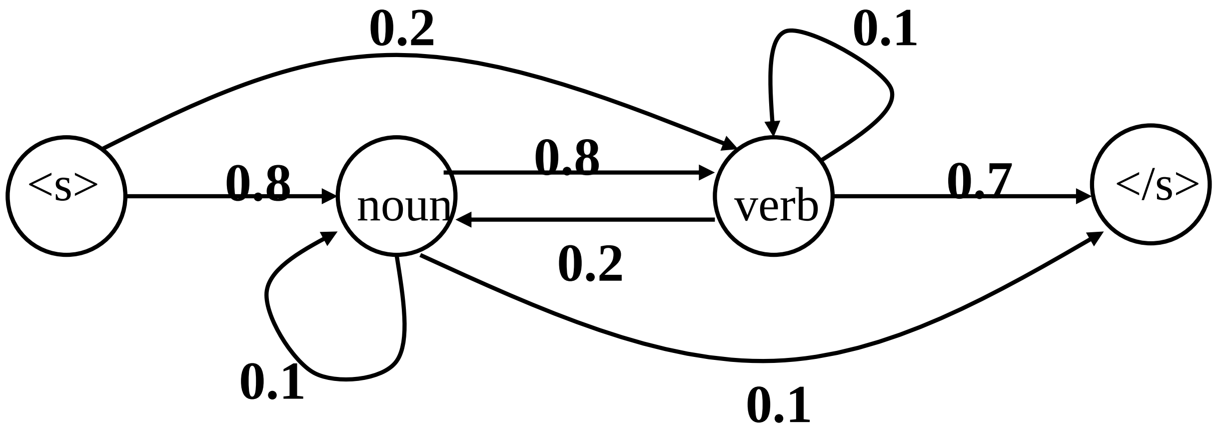
(if ‘fish’ is a noun)

	0	1	2	3
start	1	0	0	
verb	0	.1	.005 .256	
noun	0	.64	.004 .0128	
end	0	0	-	



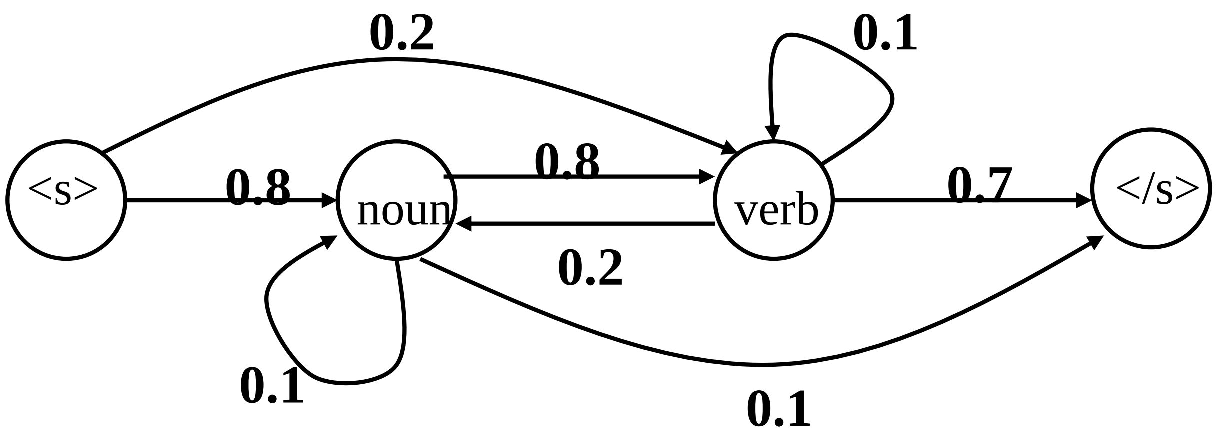
Token 2: sleep
take maximum,
set back pointers

	0	1	2	3
start	1	0	0	
verb	0	.1	~~.005~~ .256	
noun	0	.64	~~.004~~ .0128	
end	0	0	-	



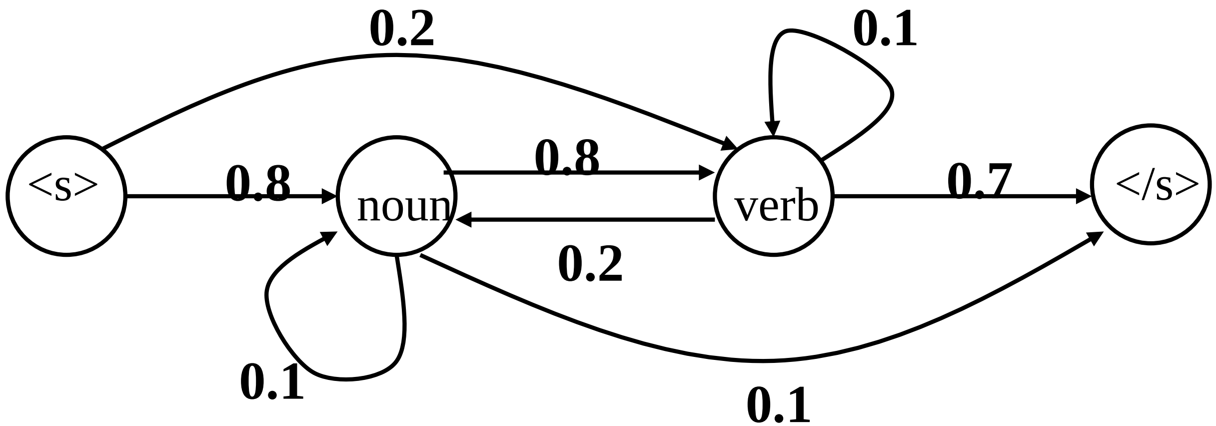
Token 2: sleep
take maximum,
set back pointers

	0	1	2	3
start	1	0	0	
verb	0	.1	.256	
noun	0	.64	.0128	
end	0	0	-	



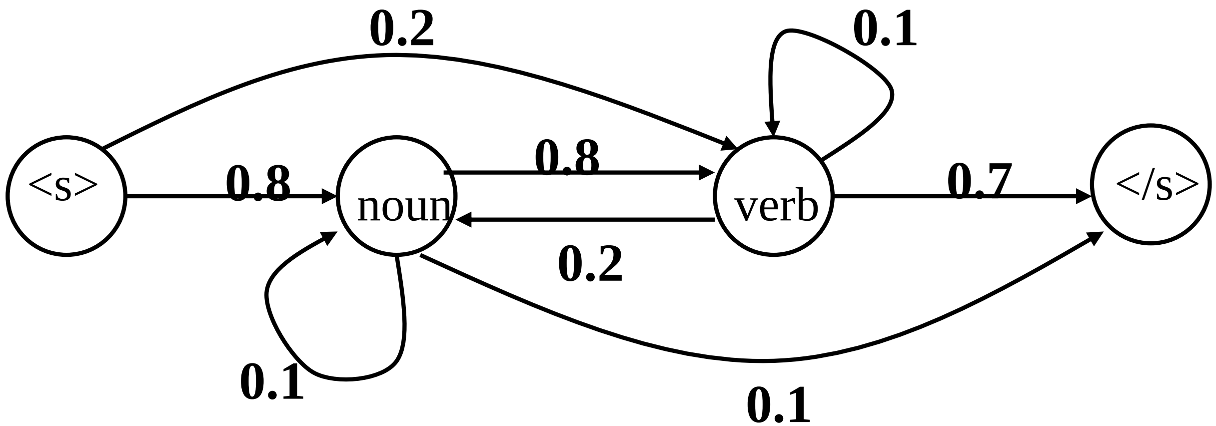
Token 3: end

	0	1	2	3
start	1	0	0	0
verb	0	.1	.256	-
noun	0	.64	.0128	-
end	0	0	-	.256*.7 .0128*.1



Token 3: end
take maximum,
set back pointers

	0	1	2	3
start	1	0	0	0
verb	0	.1	.256	-
noun	0	.64	.0128	-
end	0	0	-	.256*.7 ~~.0128*.1~~



Decode:
fish = noun
sleep = verb

	0	1	2	3
start	1	0	0	0
verb	0	.1	.256	-
noun	0	.64	.0128	-
end	0	0	-	.256*.7

State Lattice / Trellis (Trigram HMM)