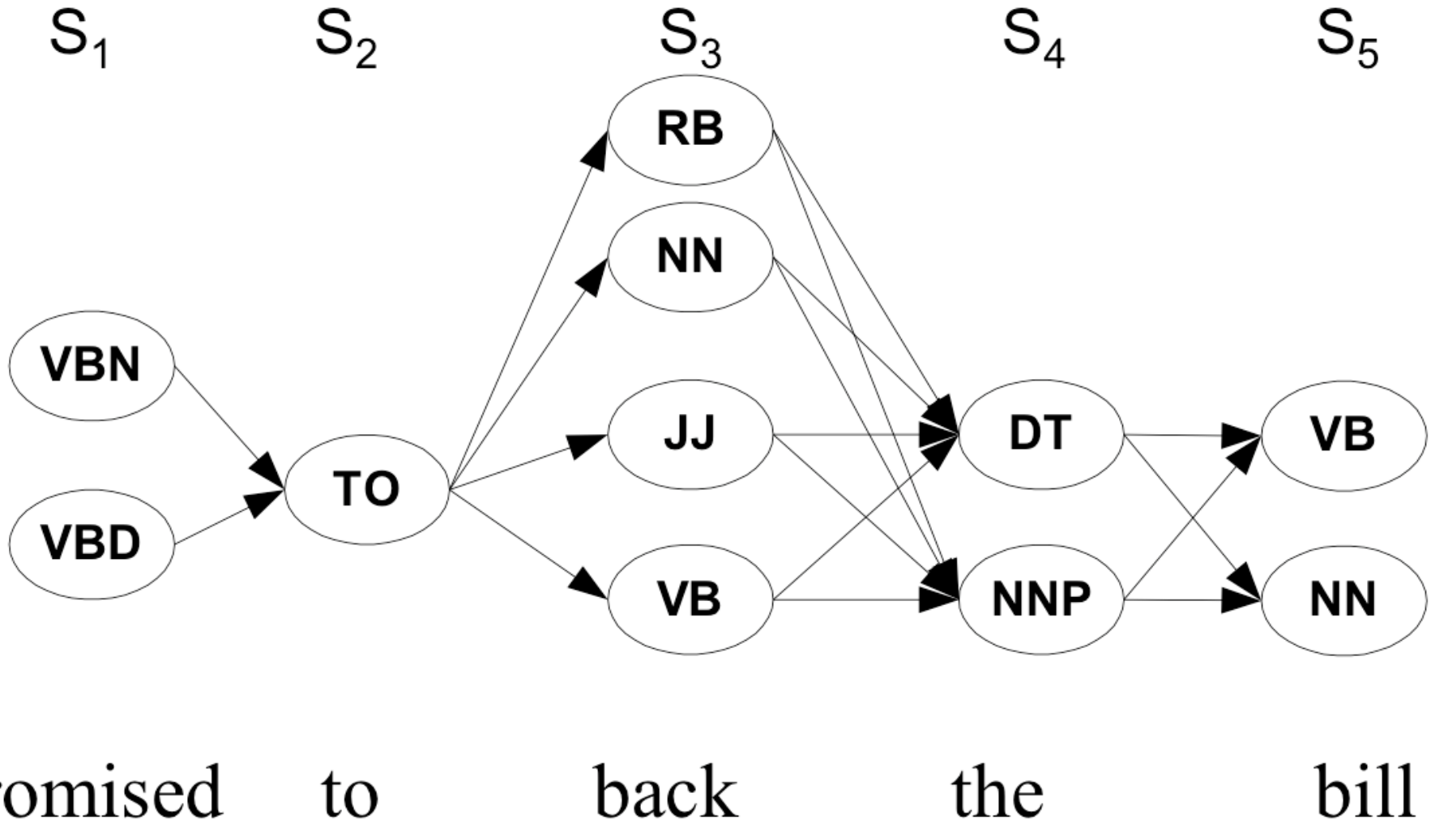
Viterbi algorithm for HMMs

Given a word sequence, how to compute the most probable sequence of POS tags efficiently?

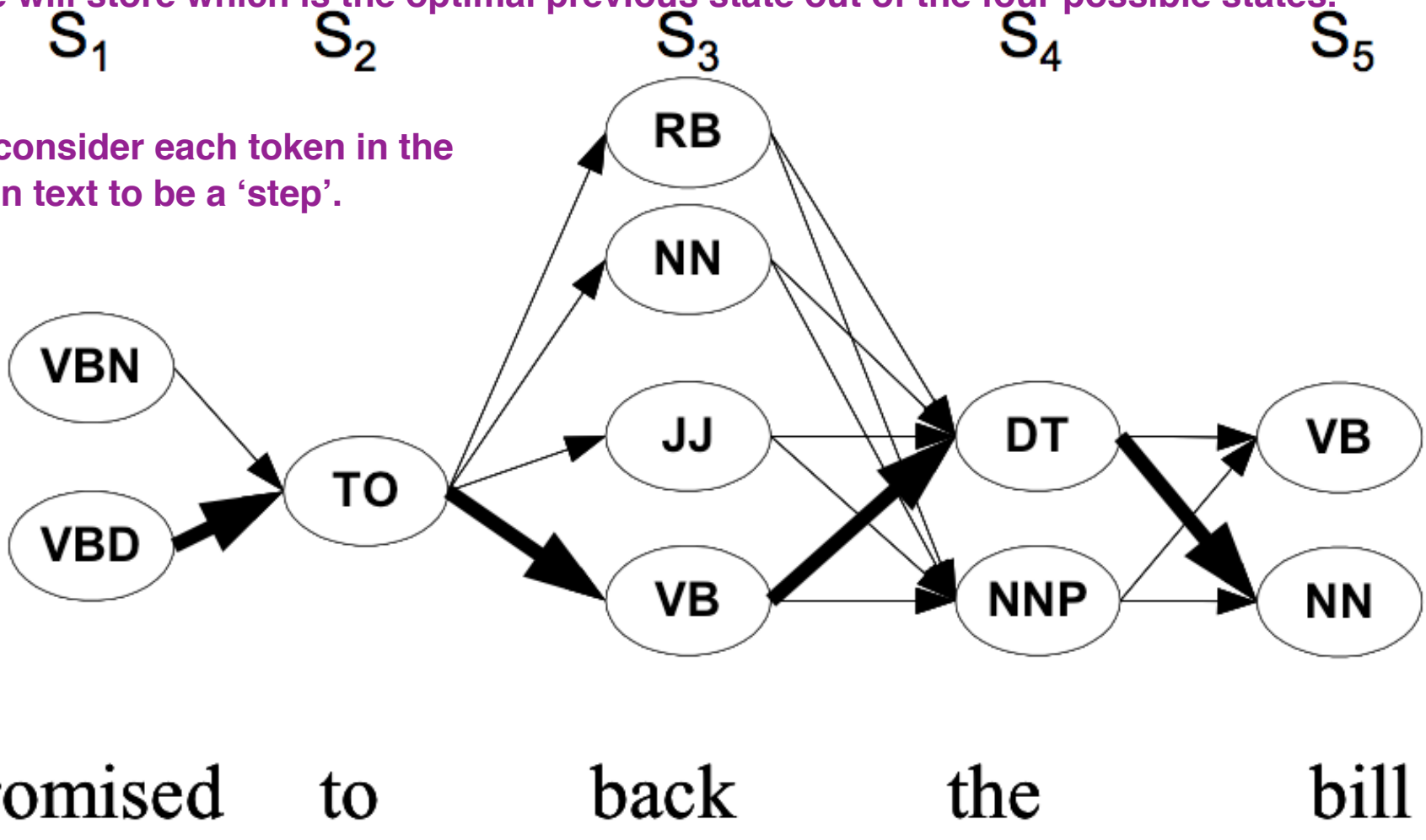
Walking through the states: best path

Enumerating all possible tag sequences and computing probabilities — inefficient



Walking through the states: best path

A Dynamic Programming approach where we will first compute the shorter best paths, and then the longer best paths.
For each state, we will store the optimal predecessor (previous state). E.g., for state DT, we will store which is the optimal previous state out of the four possible states.



Finding the best path: Viterbi Algorithm

Intuition

Optimal path for each state can be recorded. We need

- Cheapest cost to state j at step s: $\delta_j(s)$
- Backtrace from that state to best predecessor $\psi_j(s)$

Specifically in our context, cheapest cost ~ highest probability

e.g., on previous slide, the best predecessor of DT was VB

Finding the best path: Viterbi Algorithm

Intuition

Optimal path for each state can be recorded. We need

- Cheapest cost to state j at step s: $\delta_j(s)$
- Backtrace from that state to best predecessor $\psi_j(s)$

Computing these values

- $\delta_j(s+1) = max_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$

**We have computed the cheapest cost for every state i at step s.
Now we want to compute the cheapest cost for state j at step s+1. Along with the state transition probability (i to j), we also need to consider the probability of the token at step (s+1) being generated.**

Finding the best path: Viterbi Algorithm

Intuition

Optimal path for each state can be recorded. We need

- Cheapest cost to state j at step s: $\delta_j(s)$
- Backtrace from that state to best predecessor $\psi_j(s)$

Computing these values

- $\delta_j(s+1) = max_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$
- $\psi_j(s+1) = argmax_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$

Finding the best path: Viterbi Algorithm

Intuition

Optimal path for each state can be recorded. We need

- Cheapest cost to state j at step s: $\delta_j(s)$
- Backtrace from that state to best predecessor $\psi_j(s)$

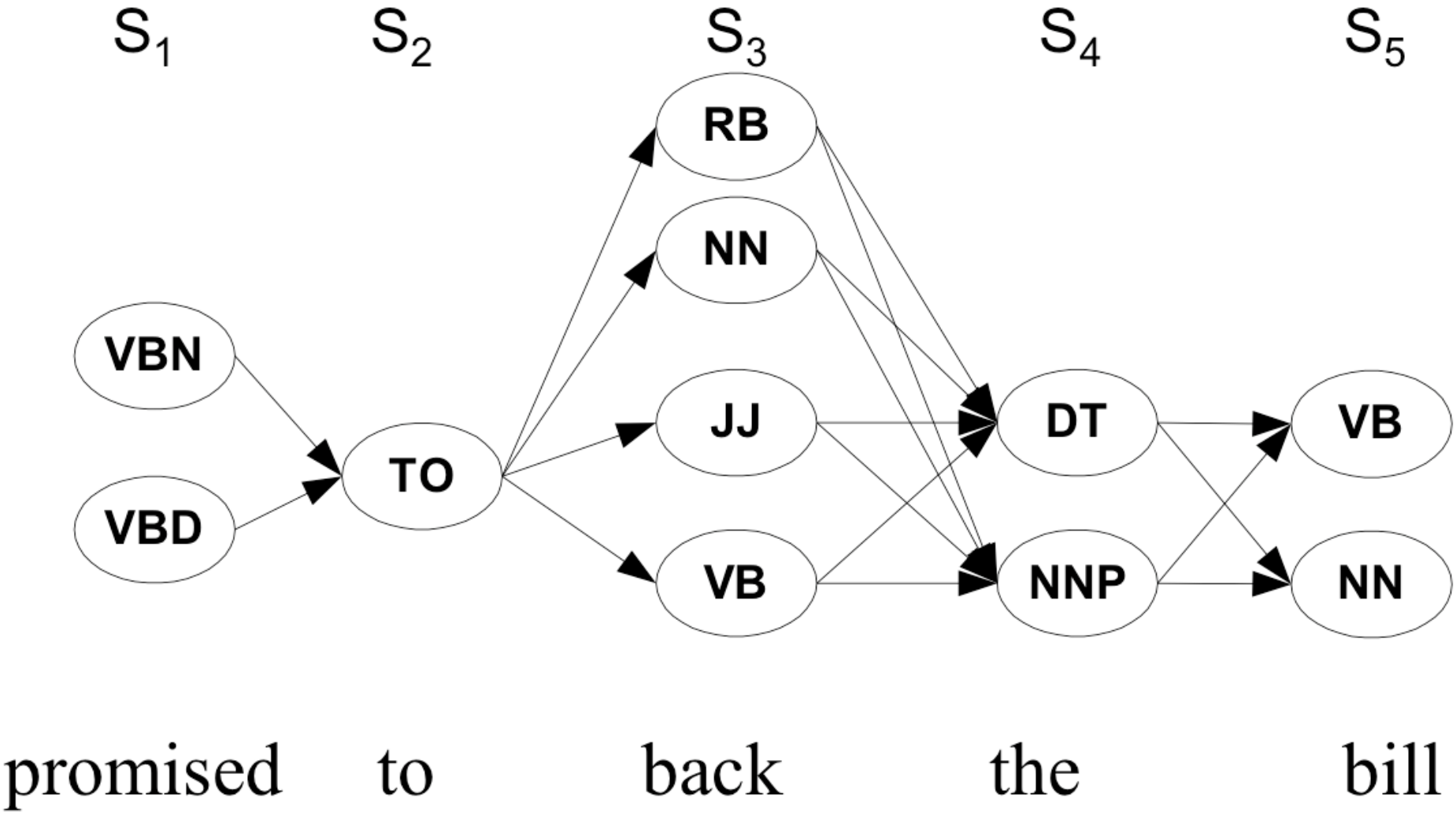
Computing these values

- $\delta_j(s+1) = max_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$
- $\psi_j(s+1) = argmax_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$

Best final state is $argmax_{1 \leq i \leq N} \delta_i(|S|)$, we can backtrack from there

Walking through the states: best path

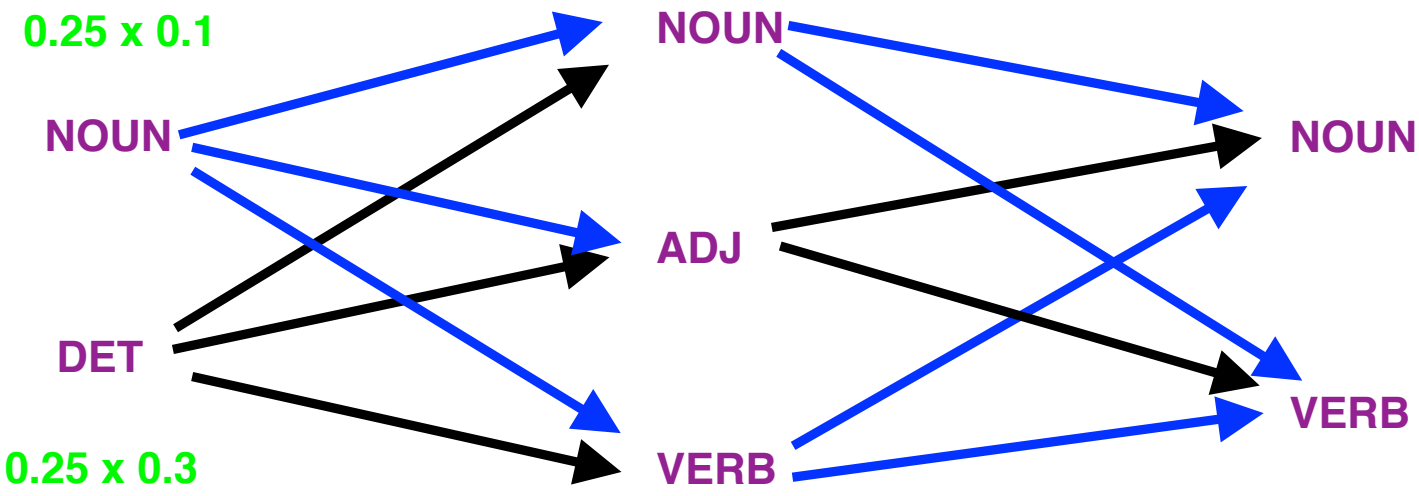
At the 5th step, we will know which of the two states (VB or NN) gives the cheapest cost / higher probability. We can backtrack from that state, and get the optimal sequence of states (POS tags).



Practice Question

- Suppose you want to use a HMM tagger to tag the phrase, “the light book”, where we have the following probabilities:
- P(the|Det) = 0.3, P(the|Noun) = 0.1, P(light|Noun) = 0.003, P(light|Adj) = 0.002, P(light|Verb) = 0.06, P(book|Noun) = 0.003, P(book|Verb) = 0.01
- P(Verb|Det) = 0.00001, P(Noun|Det) = 0.5, P(Adj|Det) = 0.3, P(Noun|Noun) =0.2, P(Adj|Noun) = 0.002, P(Noun|Adj) = 0.2, P(Noun|Verb) = 0.3, P(Verb|Noun) = 0.3, P(Verb|Adj) = 0.001, P(Verb|Verb) = 0.1
- Work out in details the steps of the Viterbi algorithm. You can use a Table to show the steps. Assume all other conditional probabilities, not mentioned to be zero. Also, assume that all tags have the same probabilities to appear in the beginning of a sentence.

Assume there are only 4 tags, so probability of each tag starting the sentence is 0.25.



the	light	book
s1	s2	s3

Learning the Parameters

Two Scenarios

- A labeled dataset is available, with the POS category of individual words in a corpus
- Only the corpus is available, but not labeled with the POS categories

We need the following probabilities:
- probability of each POS tag starting a sentence
- transition probability of one POS tag to another tag
- probability of a word being generated from a POS tag

These probabilities are the parameters of the HMM.

Learning the Parameters

Two Scenarios

- A labeled dataset is available, with the POS category of individual words in a corpus
- Only the corpus is available, but not labeled with the POS categories

Methods for these scenarios

- For the first scenario, parameters can be directly estimated using maximum likelihood estimate from the labeled dataset **already discussed**
- For the second scenario, *Baum-Welch Algorithm* is used to estimate the parameters of the hidden markov model.