Probabilistic Tagging

- $W = w_1 \ldots w_n$ - words in the corpus (observed)
- $T = t_1 \ldots t_n$ - the corresponding tags (unknown)

Tagging: Probabilistic View (Generative Model)

Find

$$\begin{aligned}
\hat{T} &= argmax_T P(T|W) \\
&= argmax_T \frac{P(W|T)P(T)}{P(W)} \\
&= argmax_T P(W|T)P(T) \\
&= argmax_T \prod_i P(w_i|w_1 \ldots w_{i-1}, t_1 \ldots t_i)P(t_i|t_1 \ldots t_{i-1})
\end{aligned}$$

Further simplifications

$$\hat{T} = argmax_T \prod_i P(w_i|w_1 \ldots w_{i-1}, t_1 \ldots t_i) P(t_i|t_1 \ldots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag
  $P(w_i|w_1 \ldots w_{i-1}, t_1 \ldots t_i) \approx P(w_i|t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag
  $P(t_i|t_1 \ldots t_{i-1}) \approx P(t_i|t_{i-1})$
- Using these simplifications:
  $\hat{T} = argmax_T \prod_i P(w_i|t_i) P(t_i|t_{i-1})$

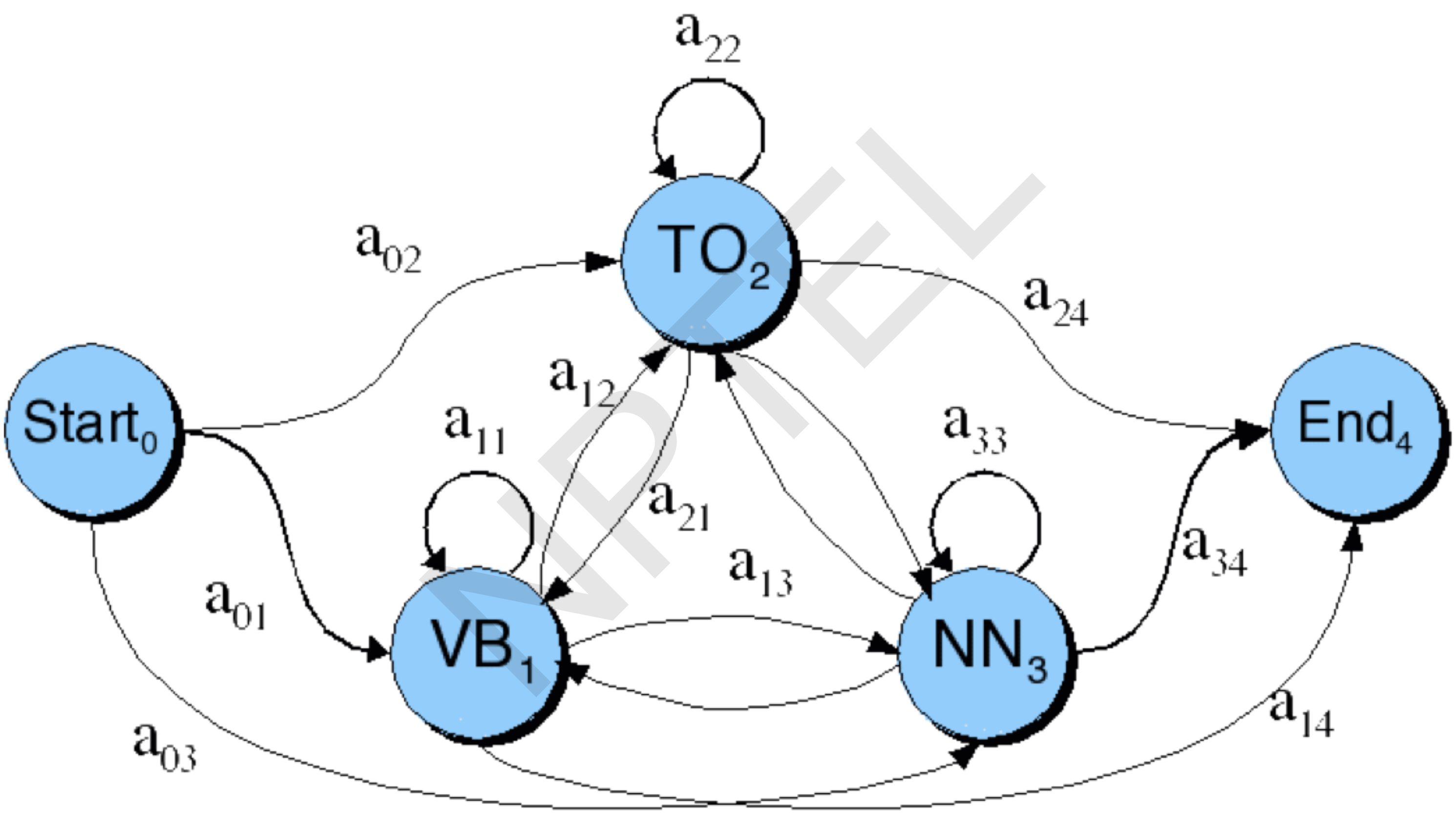
Hidden Markov Models (HMMs)

Elements of an HMM model

- A set of states (here: the tags)
- An output alphabet (here: words)
- Initial state (here: beginning of sentence)
- State transition probabilities (here $p(t_n|t_{n-1})$)
- Symbol emission probabilities (here $p(w_i|t_i)$)

Graphical Representation

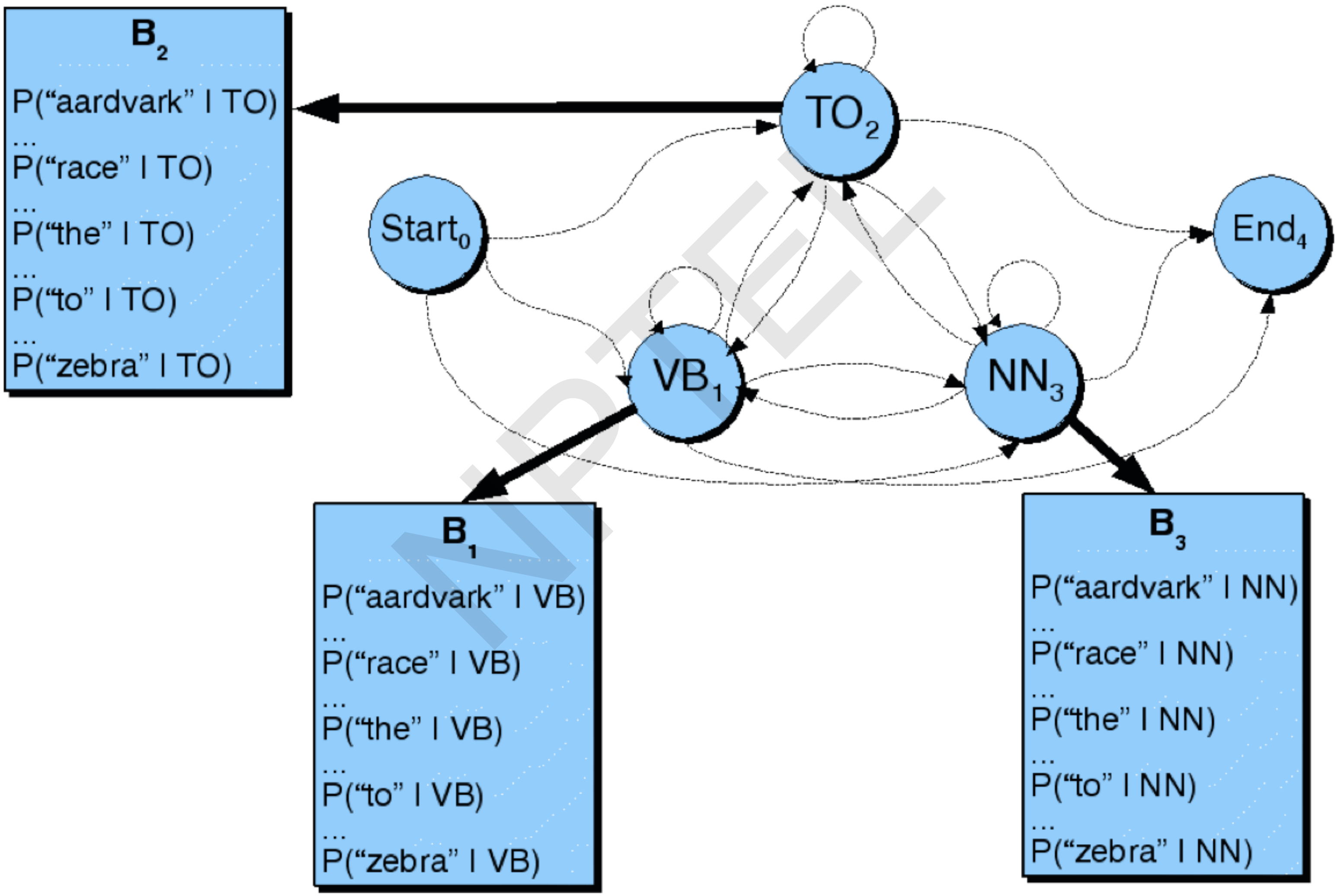
When tagging a sentence, we are walking through the state graph:



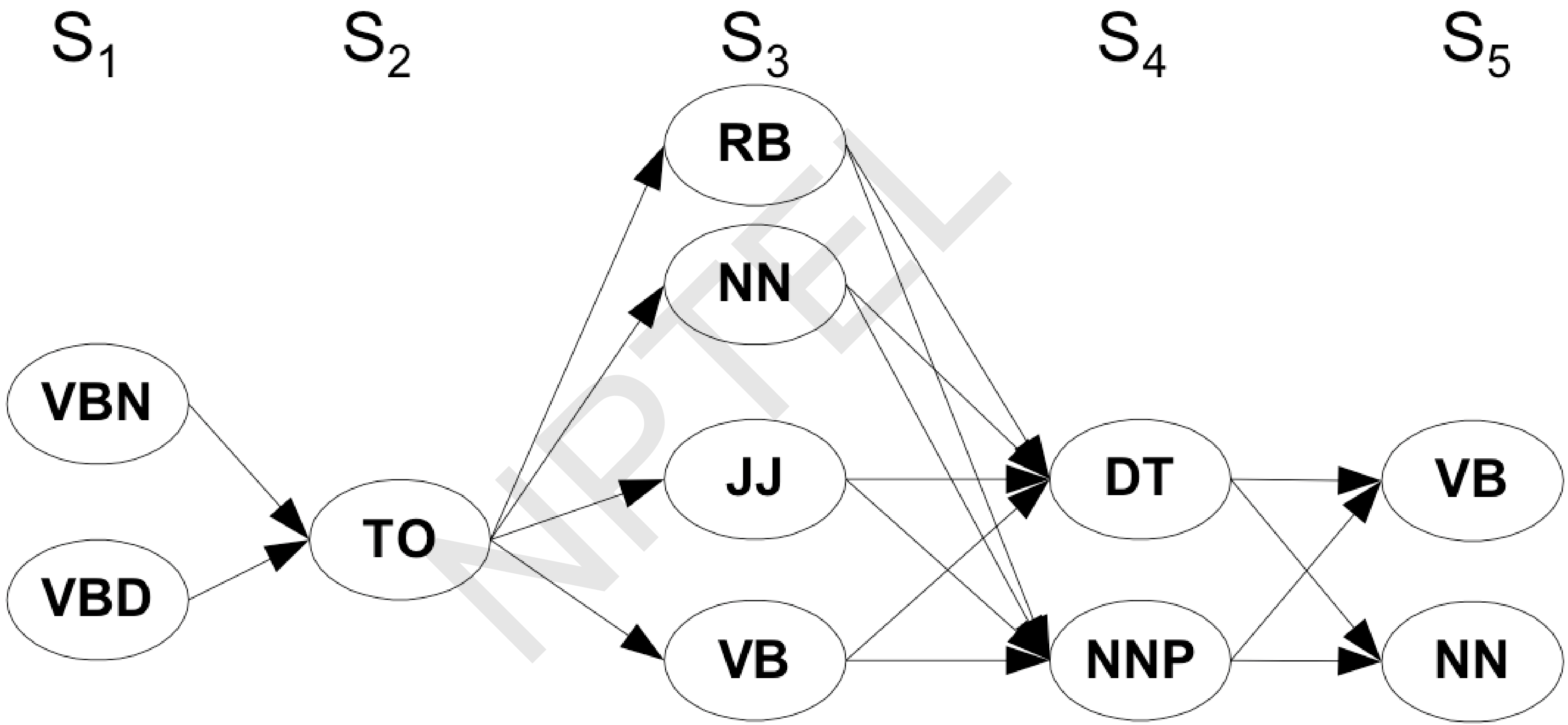
Edges are labeled with the state transition probabilities: $p(t_n|t_{n-1})$

Graphical Representation

At each state we emit a word: $P(w_n|t_n)$

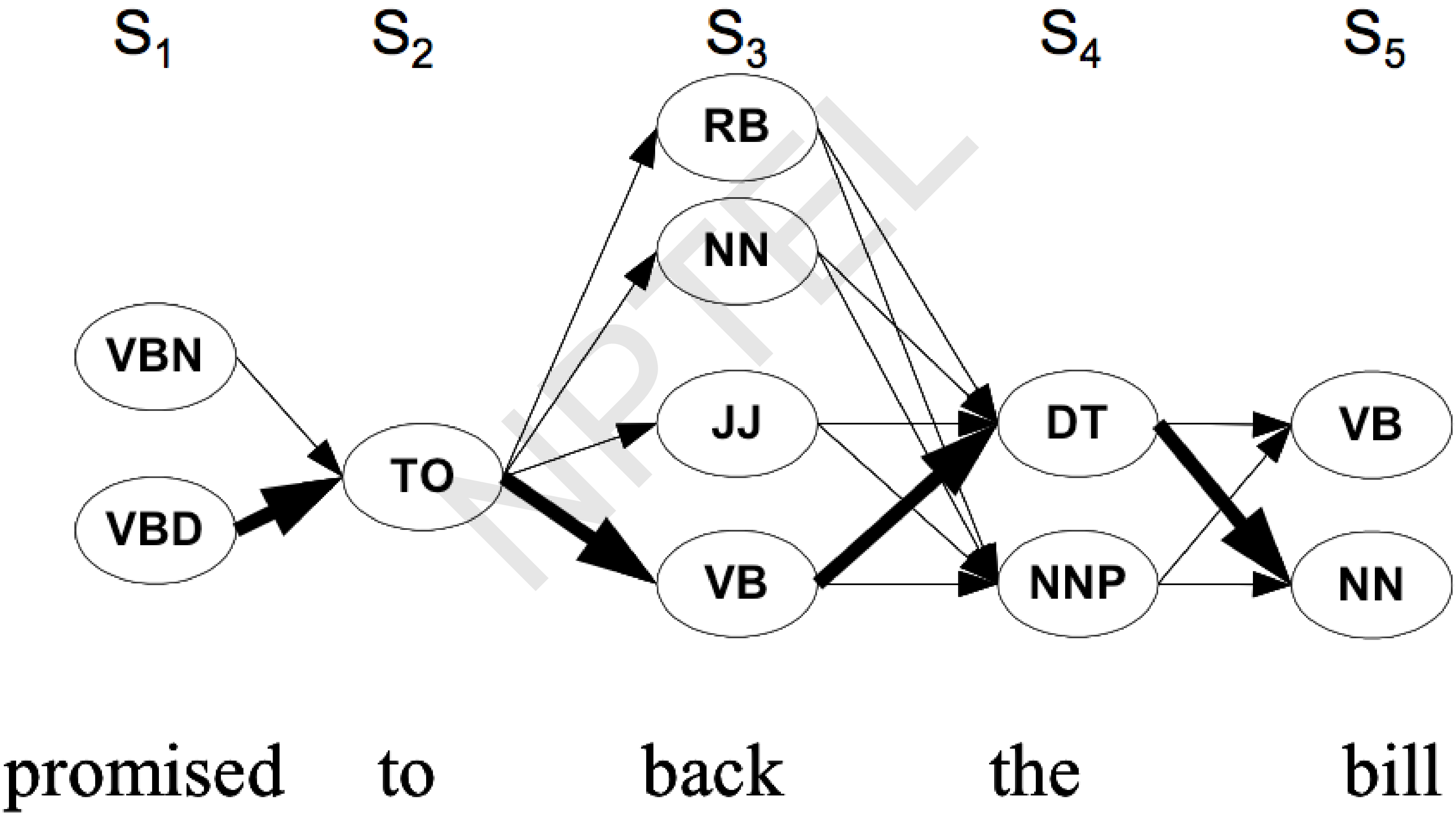


Walking through the states: best path

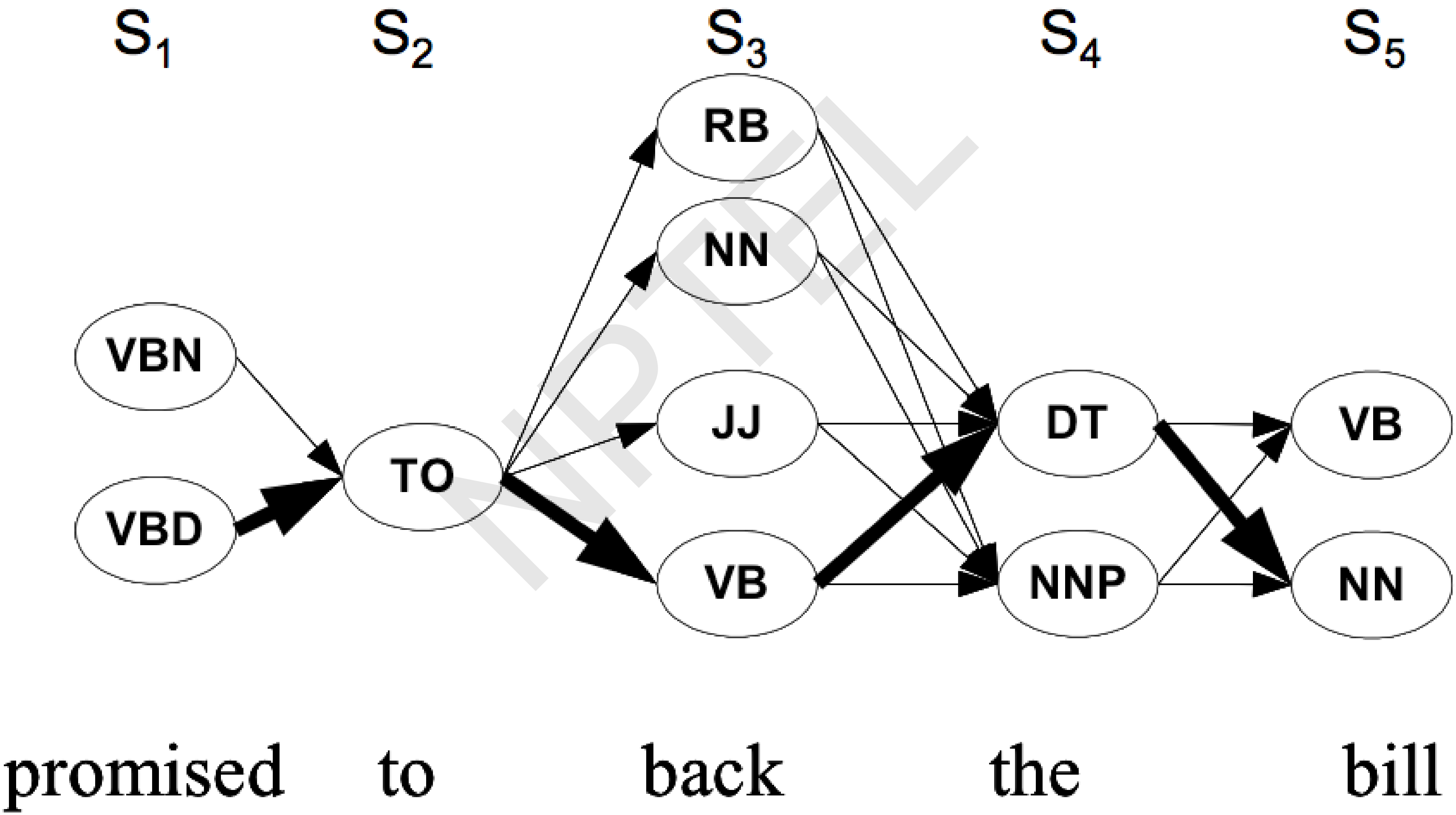


promised to back the bill

Walking through the states: best path



Walking through the states: best path



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

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.