Computational Linguistics

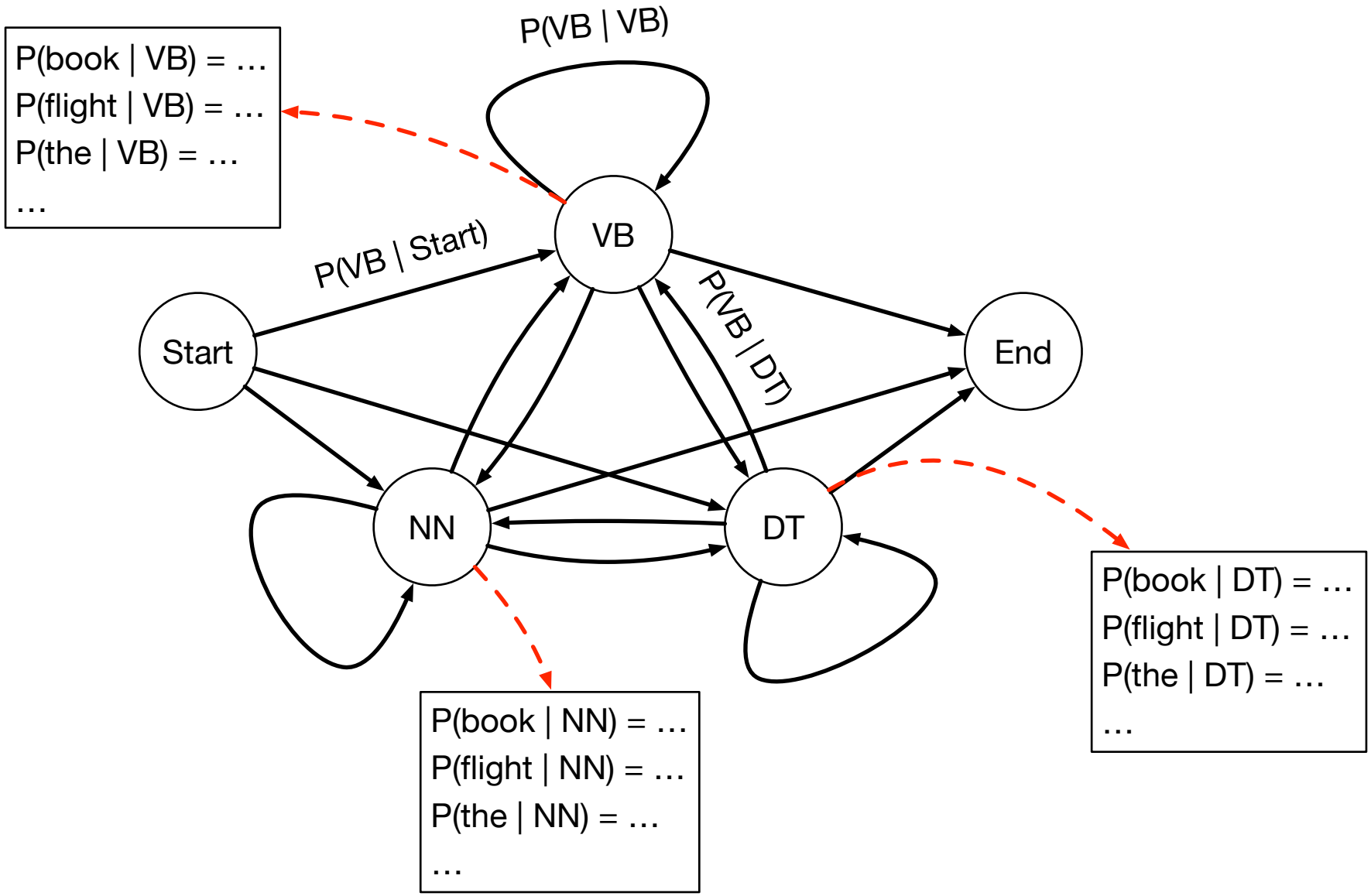
Hidden Markov Models and POS-Tagging

Clayton Greenberg
Stefan Thater

FR 4.7 Allgemeine Linguistik (Computerlinguistik)
Universität des Saarlandes
Summer 2015

[most stuff on whiteboard]

Hidden Markov Model POS-Tagger (Example)



Hidden Markov Model POS-Tagger

- $Q = q_1\ q_2\ \ldots\ q_N$ – set of states
  - q_0, q_F – special start and end states
- $A_{i,j}$ – transition probability matrix
  - $a_{i,j}$ represents the probability of moving from q_i to q_j
- $O = o_1\ o_2\ \ldots\ o_t$ – sequence of observations (words)
- $b_i(o_t)$ – emission probabilities
  - represents the probability that observation o_t has been generated by state q_i

Viterbi Decoding

- Task: Find the sequence of POS-tags that best explains the “observations” (words in the input sequence).
- Input:
  - Hidden Markov Model
  - sequence $o_1 \ldots o_T$ of words
- Output:
  - sequence $q_1 \ldots q_T$ of POS-tags (+ corresponding prob.)
- Algorithm: Dynamic Programming (cf. CYK)

Viterbi Decoding

Matrix: One column per Word, one row per POS-tag

In each cell: compute max. probability + back pointer

	Mary	*had*	*cats*	
	DT	DT	DT	
Start 1.0	NN	NN	NN	End
	VBD	VBD	VBD	

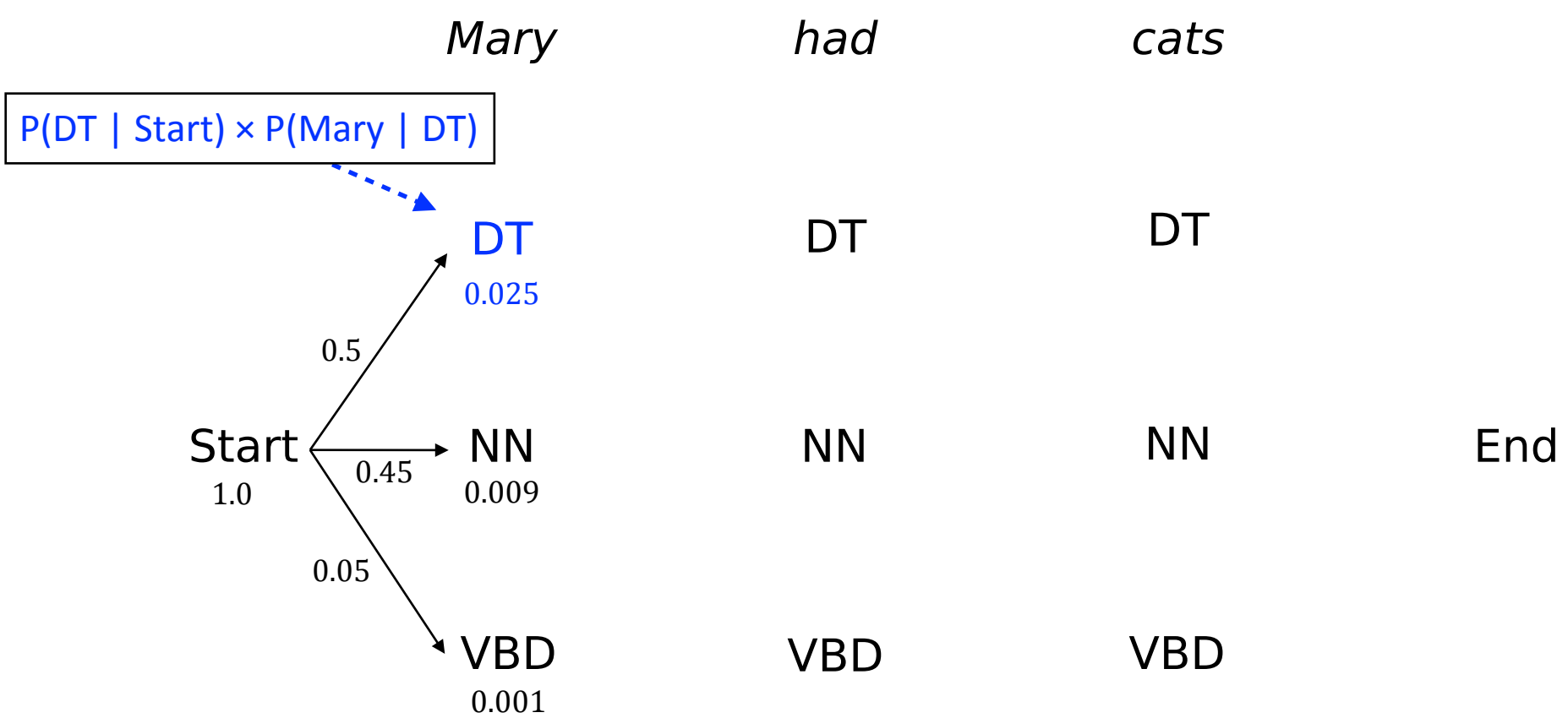
Viterbi Decoding

P(DT | Start) = 0.5
P(NN | Start) = 0.45
P(VBD | Start) = 0.05

P(Mary | DT) = 0.05
P(Mary | NN) = 0.2
P(Mary | VBD) = 0.02

Matrix: One co

In each cell: co



Viterbi Decoding

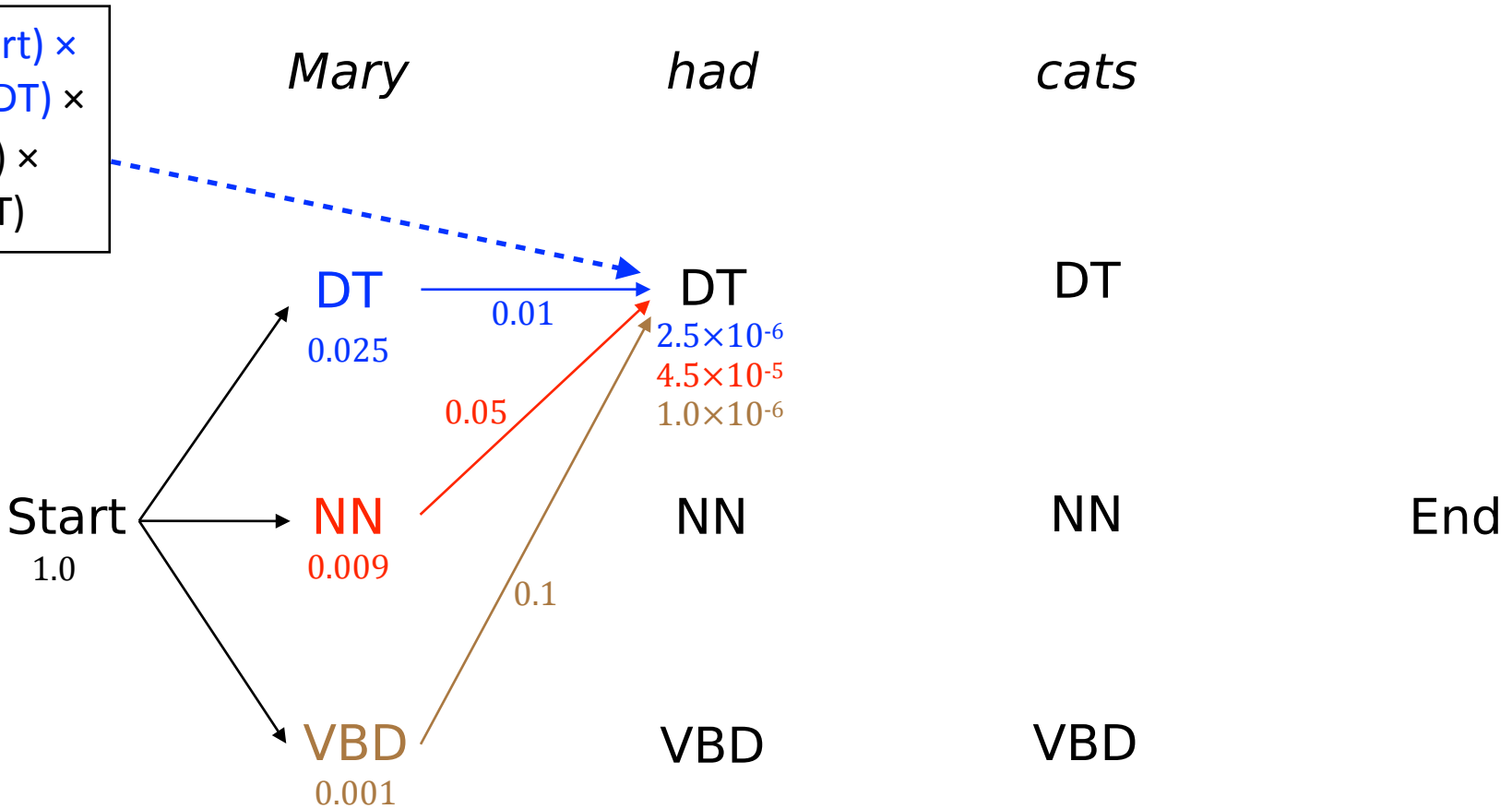
P(DT | DT) = 0.01
P(DT | NN) = 0.05
P(DT | VBD) = 0.1

P(had | DT) = 0.01
P(had | NN) = 0.05
P(had | VBD) = 0.2

Matrix: One co

In each cell: co

P(DT | Start) ×
P(Mary | DT) ×
P(DT | DT) ×
P(had | DT)



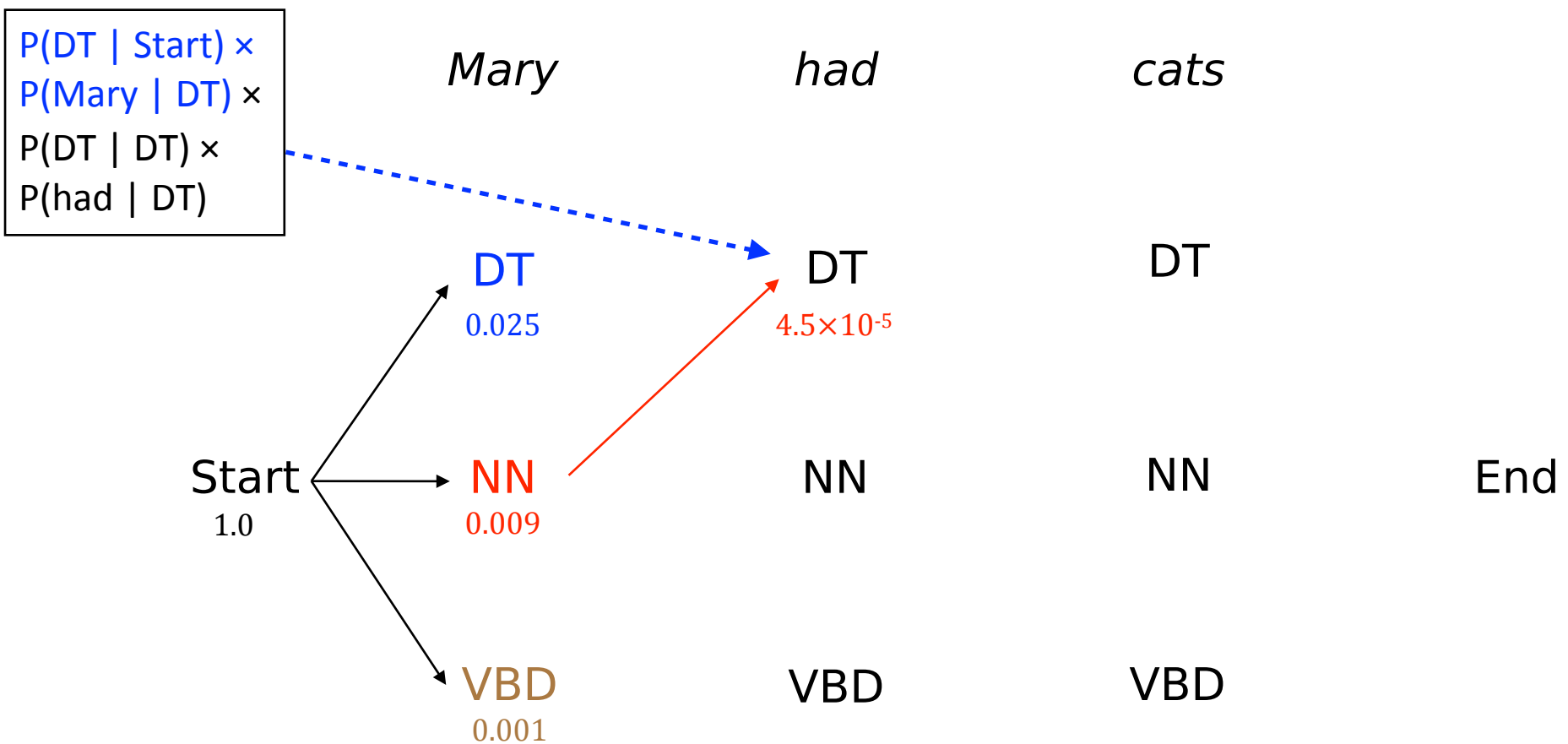
Viterbi Decoding

P(DT | DT) = 0.01
P(DT | NN) = 0.05
P(DT | VBD) = 0.1

P(had | DT) = 0.01
P(had | NN) = 0.05
P(had | VBD) = 0.2

Matrix: One co

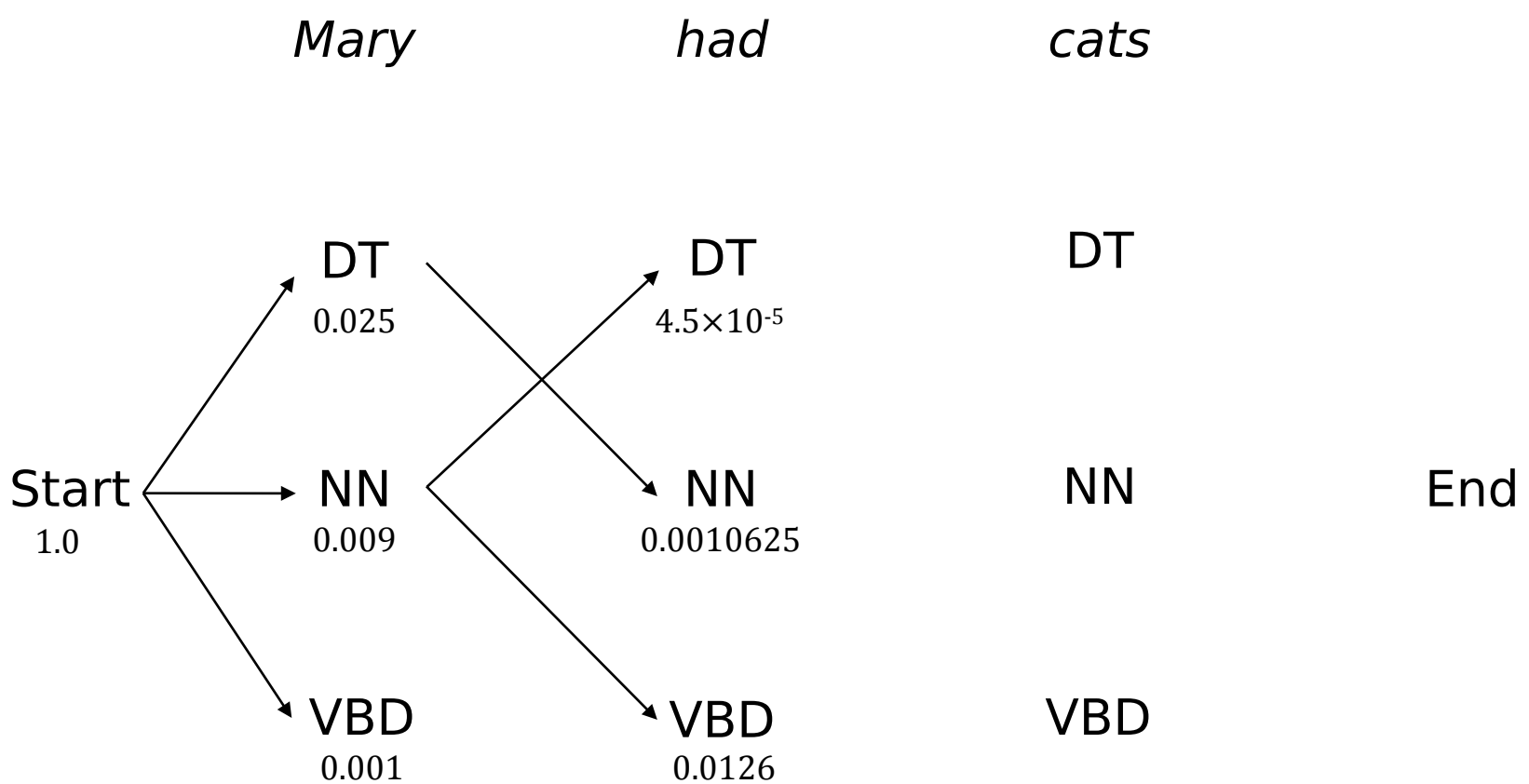
In each cell: co



Viterbi Decoding

Matrix: One column per Word, one row per POS-tag

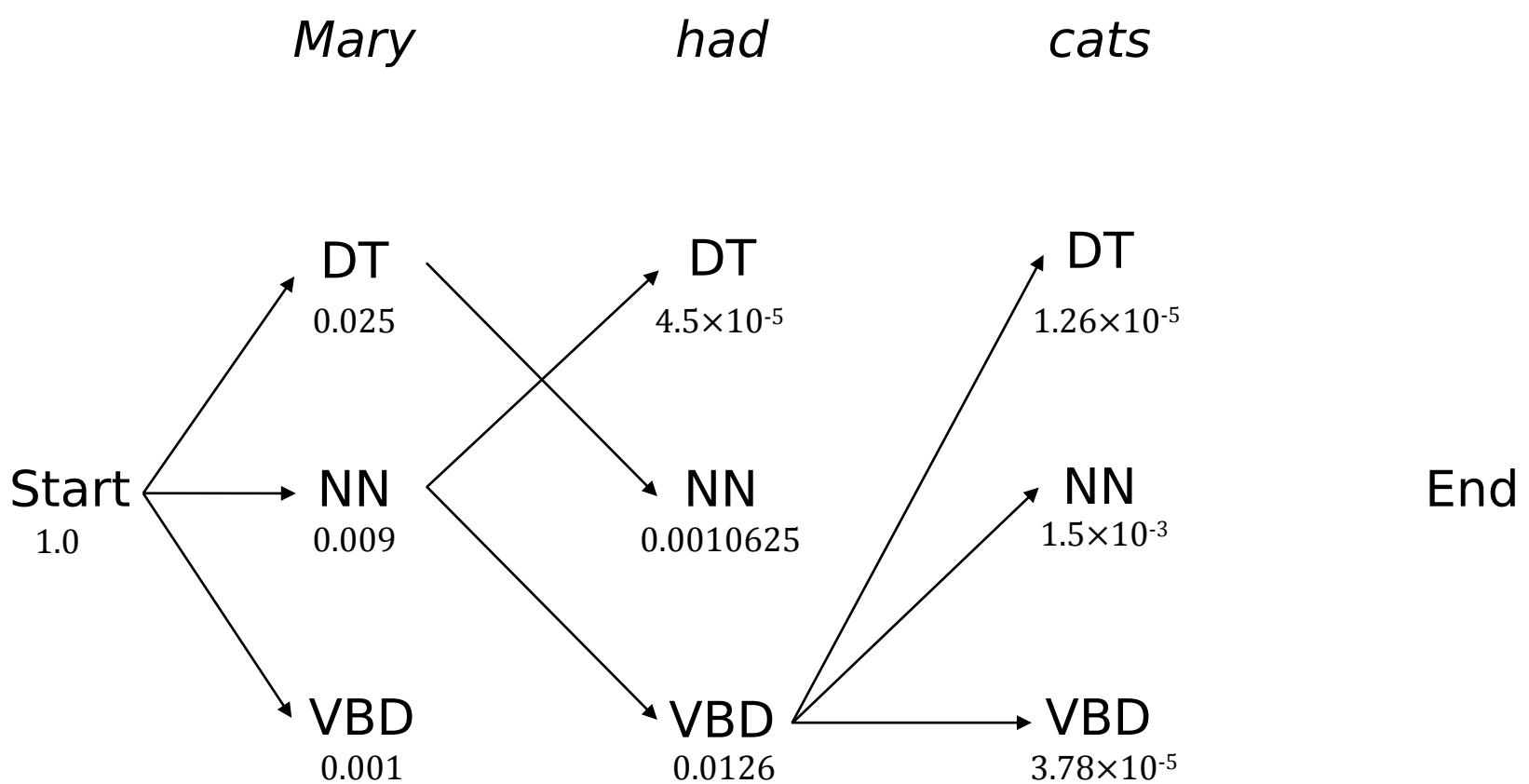
In each cell: compute max. probability + back pointer



Viterbi Decoding

Matrix: One column per Word, one row per POS-tag

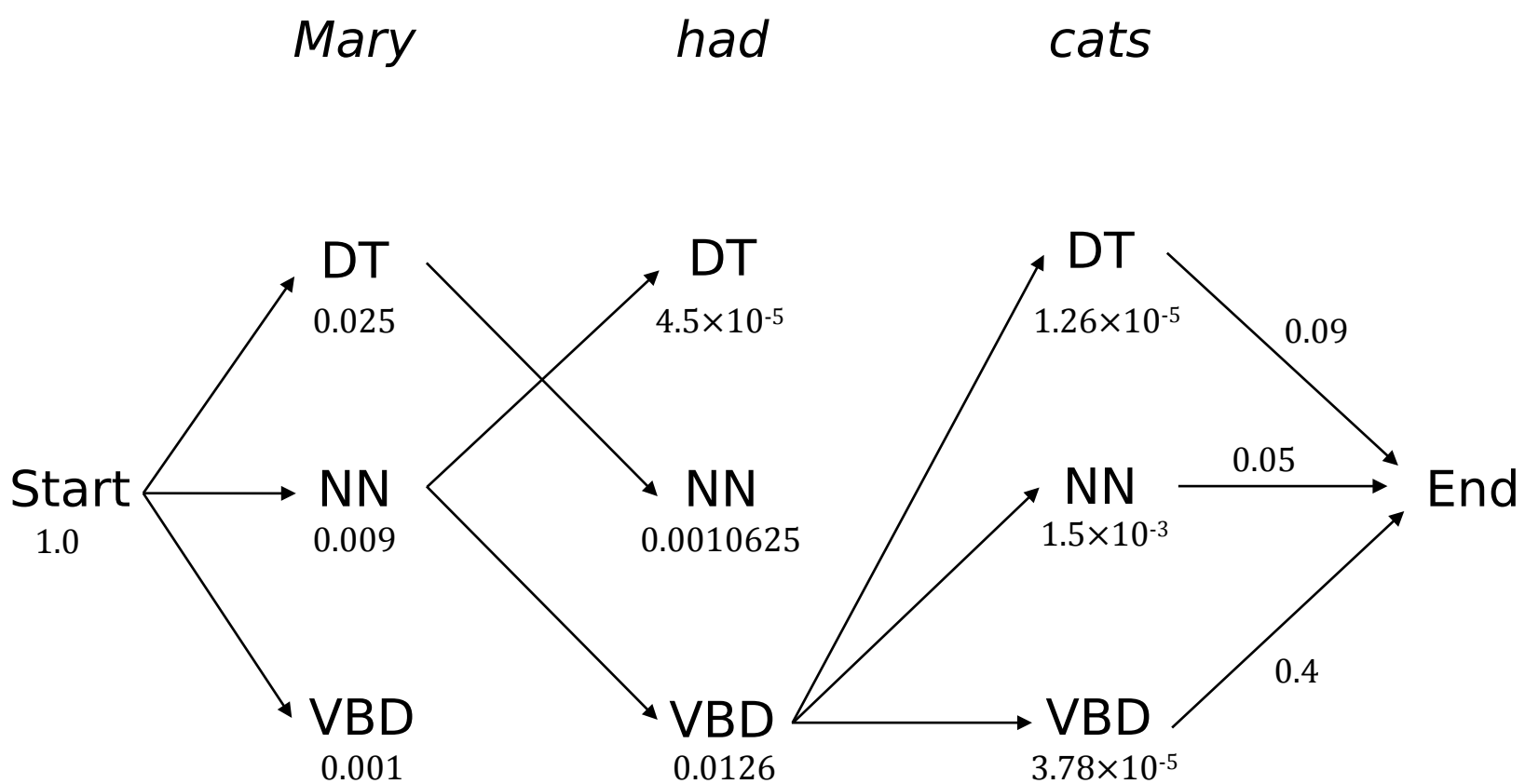
In each cell: compute max. probability + back pointer



Viterbi Decoding

Matrix: One column per Word, one row per POS-tag

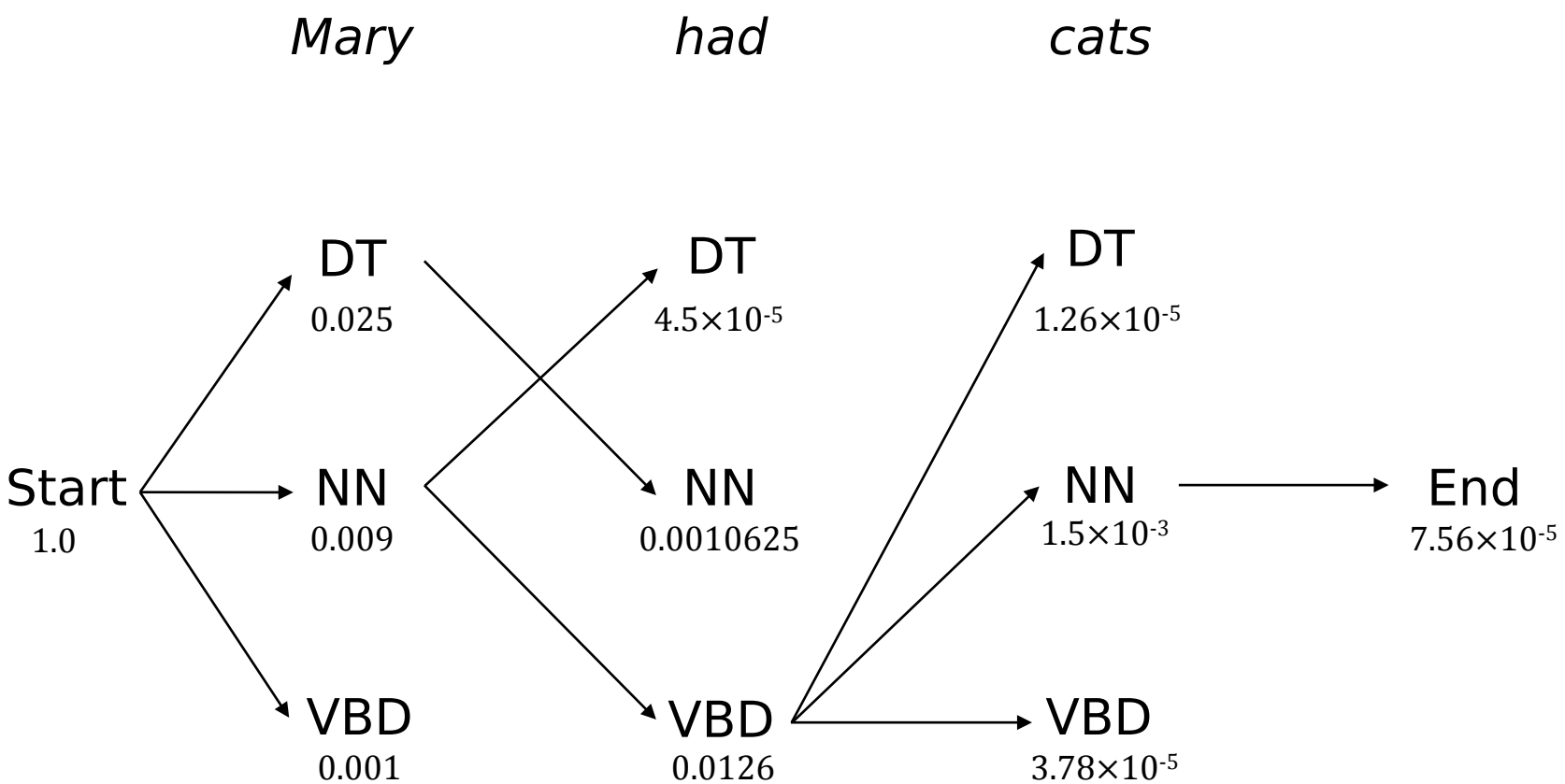
In each cell: compute max. probability + back pointer



Viterbi Decoding

Matrix: One column per Word, one row per POS-tag

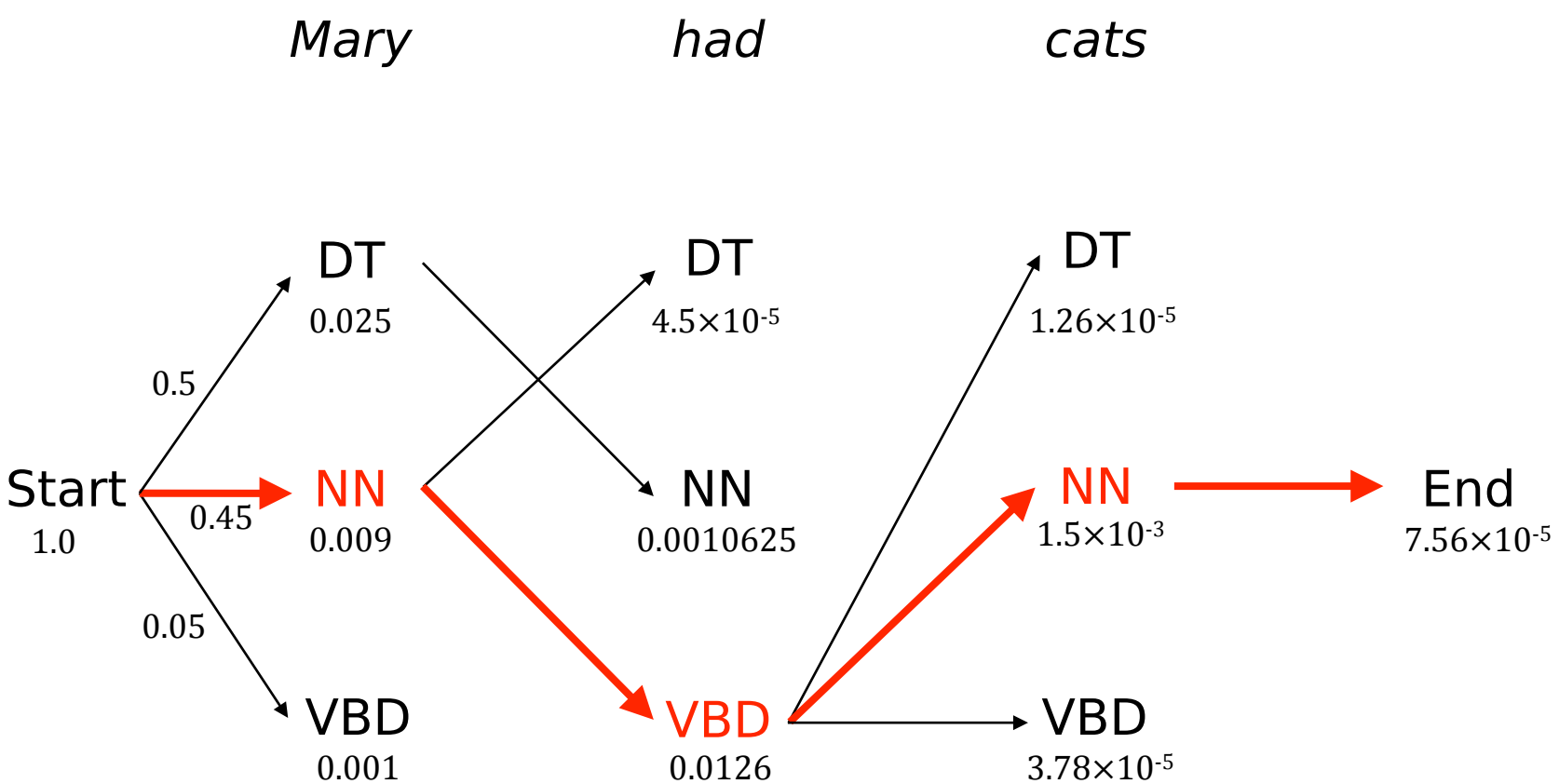
In each cell: compute max. probability + back pointer



Viterbi Decoding

Matrix: One column per Word, one row per POS-tag

In each cell: compute max. probability + back pointer



(Jurafsky & Martin, p. 181)

Pseudo code

function **decode_viterbi**(observations of len T, state-graph of len N):

create a path probability matrix viterbi[N+2,T]

for each state s **from** 1 **to** N **do** # initialization, states = tags

viterbi[s,1] ← $a_{0,s} * b_s(o_1)$

backpointer[s,1] ← 0

for each t **from** 2 **to** T:

for each state s **from** 1 **to** N **do**:

viterbi[s,t] ← $\max_{s'}$ viterbi[s', t−1] $* a_{s',s} * b_s(o_t)$

backpointer[s,t] ← $\arg\max_{s'}$ viterbi[s', t−1] $* a_{s',s} * b_s(o_t)$

viterbi[q_F,T] ← $\max_{s}$ viterbi[s, T]$* a_{s,qF}$ # termination step

backpointer[q_F,T] ← $\arg\max_{s}$ viterbi[s, T]$* a_{s,qF}$

return the backtrace path by following backpointers starting at backpointer[q_F,T]

References

- Jurafsky & Martin: Speech and Language Processing. Pearson. Chapters 5 and 6.