Precept 4: HMM, MEMM, and Viterbi Algorithm

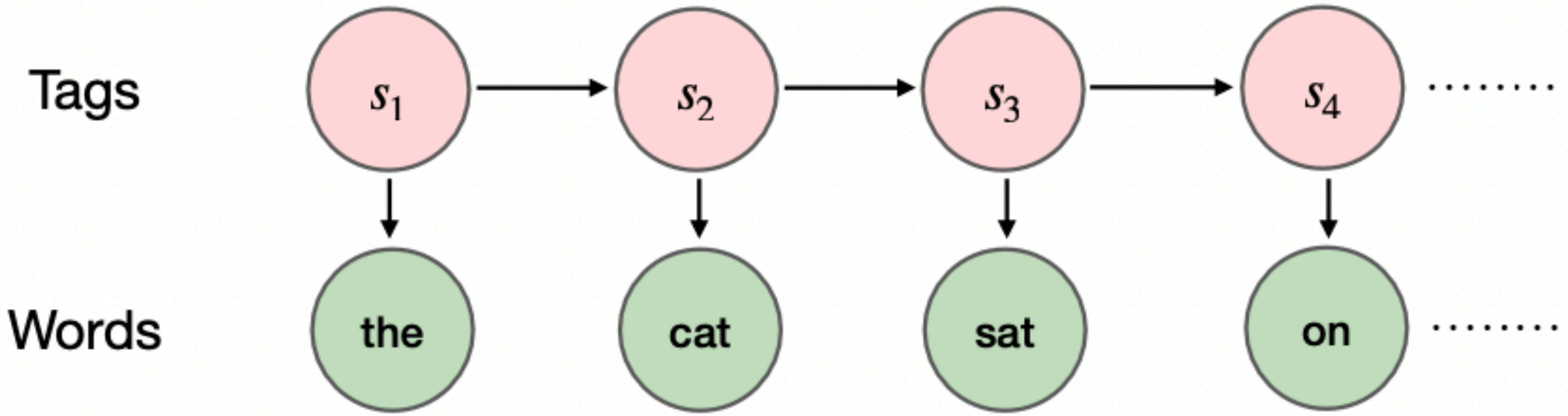
Howard Chen

02/24/2023

Agenda

- HMM
- Viterbi Algorithm
- MEMM

HMM

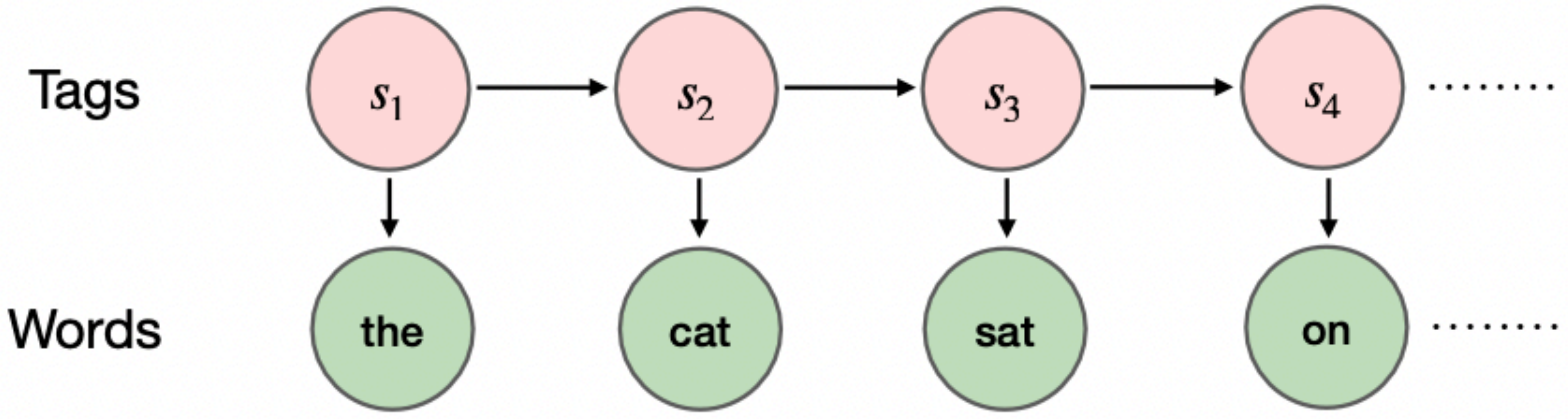


Initial state probability distribution $\pi(s_1)$

Transition probabilities $P(s_{t+1} \mid s_t)$

Emission probabilities $P(o_t \mid s_t)$

HMM



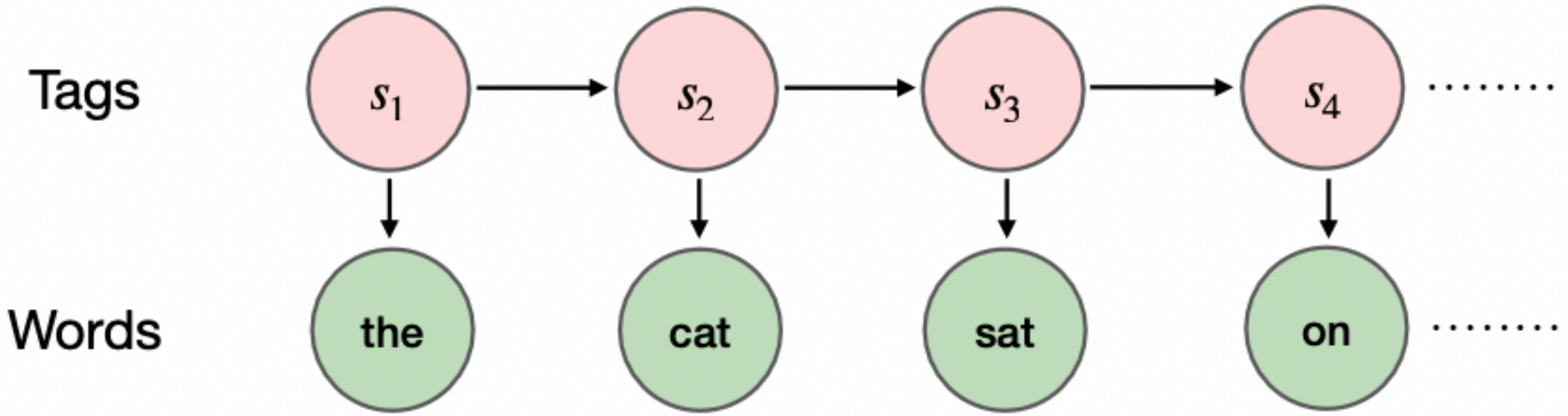
1. Markov assumption:

$$P(s_t \mid s_1, \ldots, s_{t-1}) \approx P(s_t \mid s_{t-1})$$

2. Output independence:

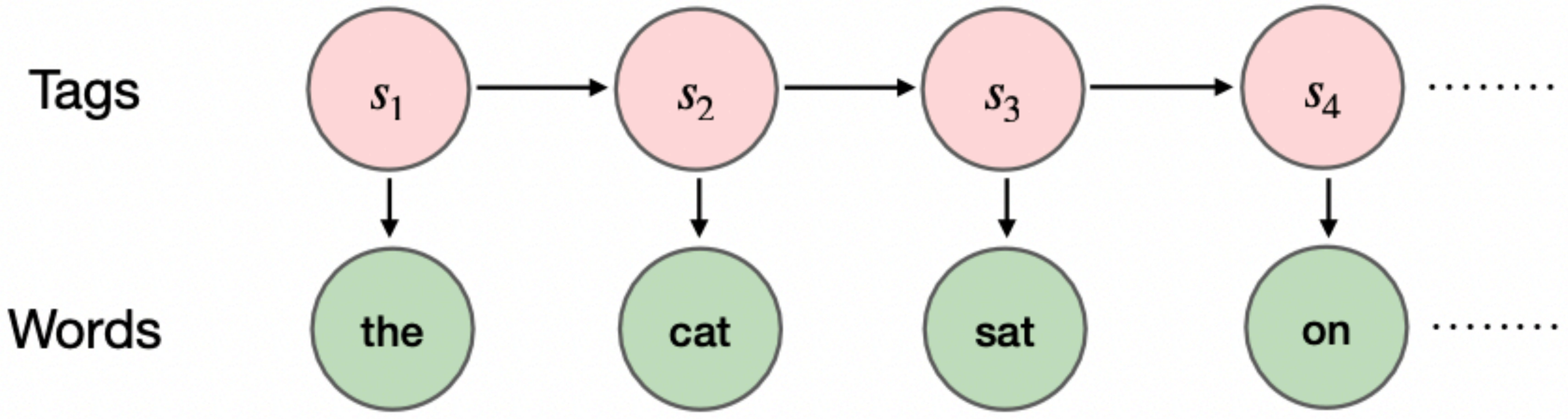
$$P(o_t \mid s_1, \ldots, s_t) \approx P(o_t \mid s_t)$$

HMM



$$P(S, O) = \prod_{i=1}^{n} P(s_i \mid s_{i-1}) P(o_i \mid s_i) \qquad [\, \pi(s_1) = P(s_1 \mid \varnothing)]$$

HMM

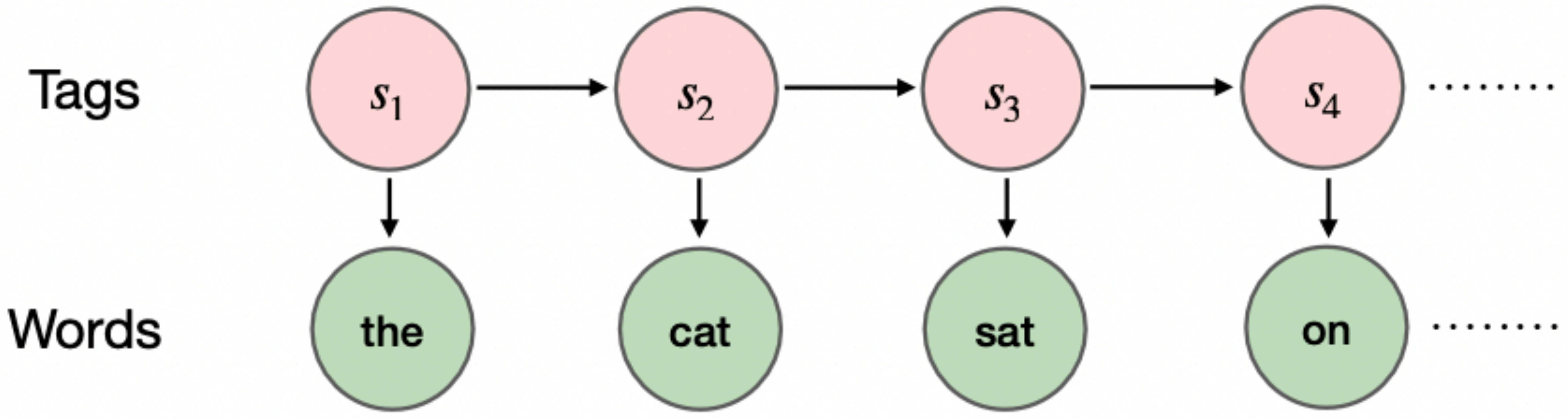


$$\hat{S} = \arg\max_S P(S \mid O) = \arg\max_S \frac{P(O \mid S)P(S)}{P(O)}$$

$$= \arg\max_S P(O \mid S)P(S)$$

$$= \arg\max_{s_1,\ldots,s_n} \prod_{i=1}^{n} P(s_i \mid s_{i-1})P(o_i \mid s_i)$$

HMM



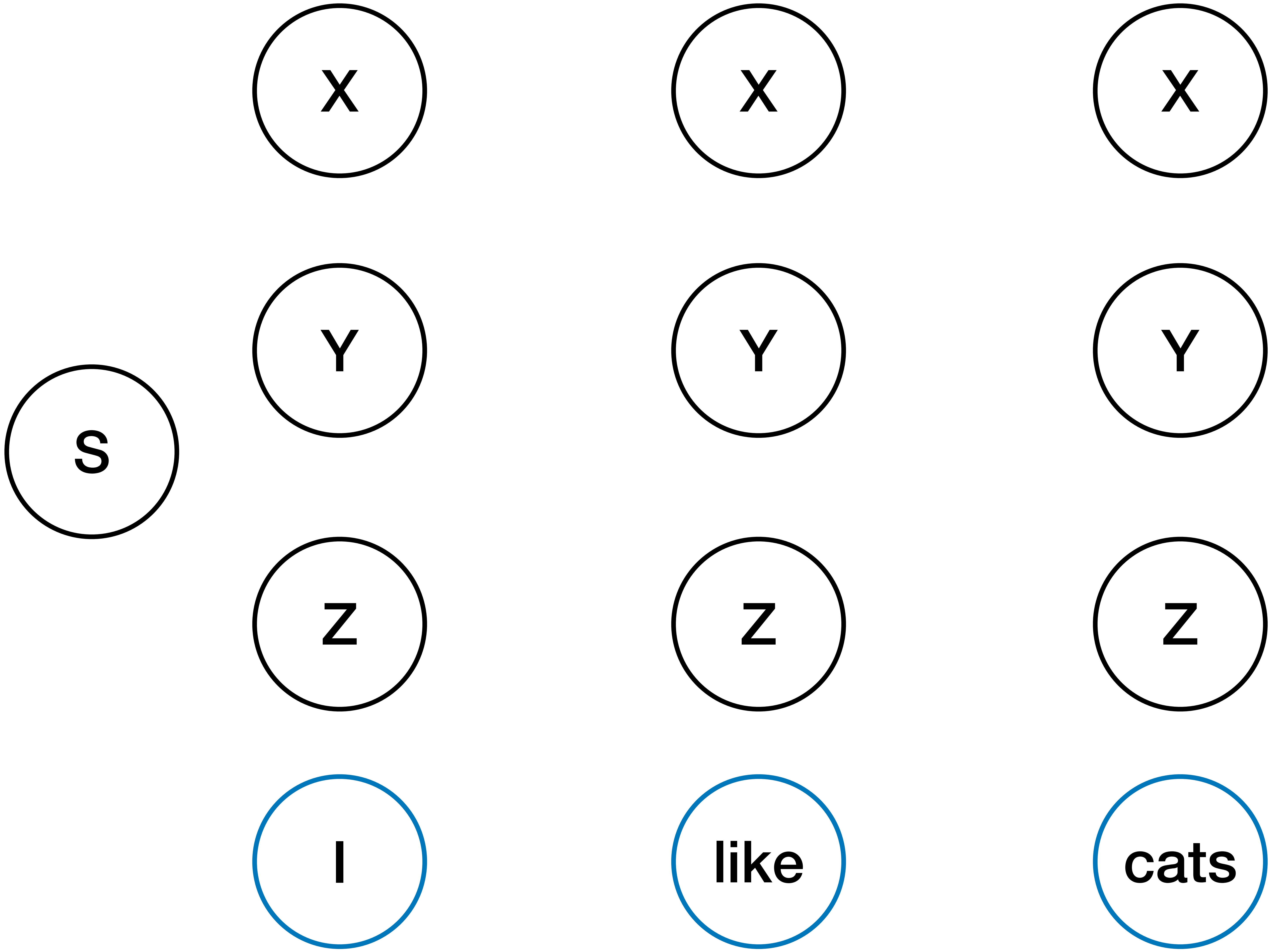
$$\hat{S} = \arg\max_{S} P(S \mid O) = \arg\max_{S} \frac{P(O \mid S)P(S)}{P(O)}$$

$$= \arg\max_{S} P(O \mid S)P(S)$$

$$= \arg\max_{s_1,\ldots,s_n} \prod_{i=1}^{n} P(s_i \mid s_{i-1})P(o_i \mid s_i)$$

Viterbi Algorithm

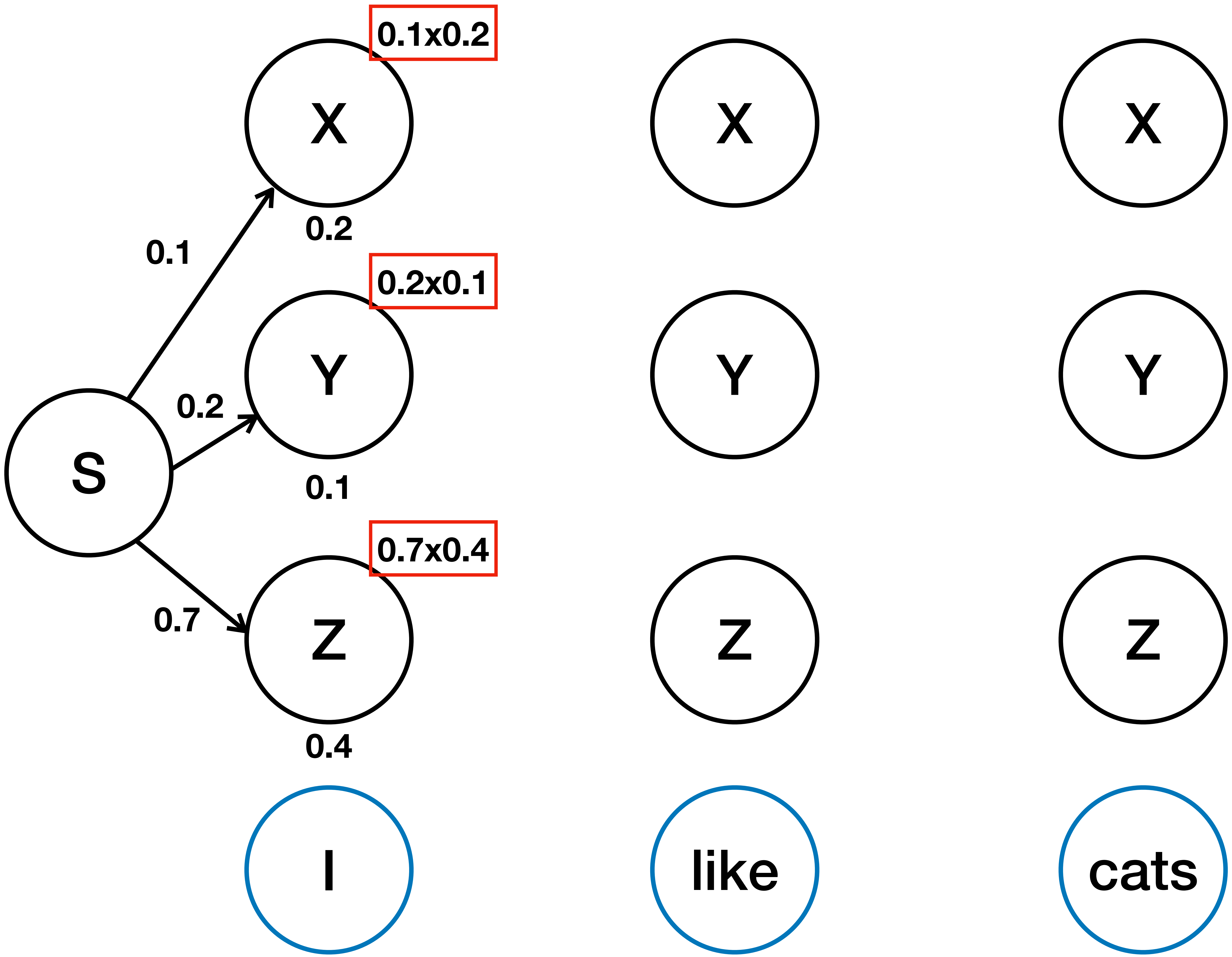
Viterbi Algorithm



	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

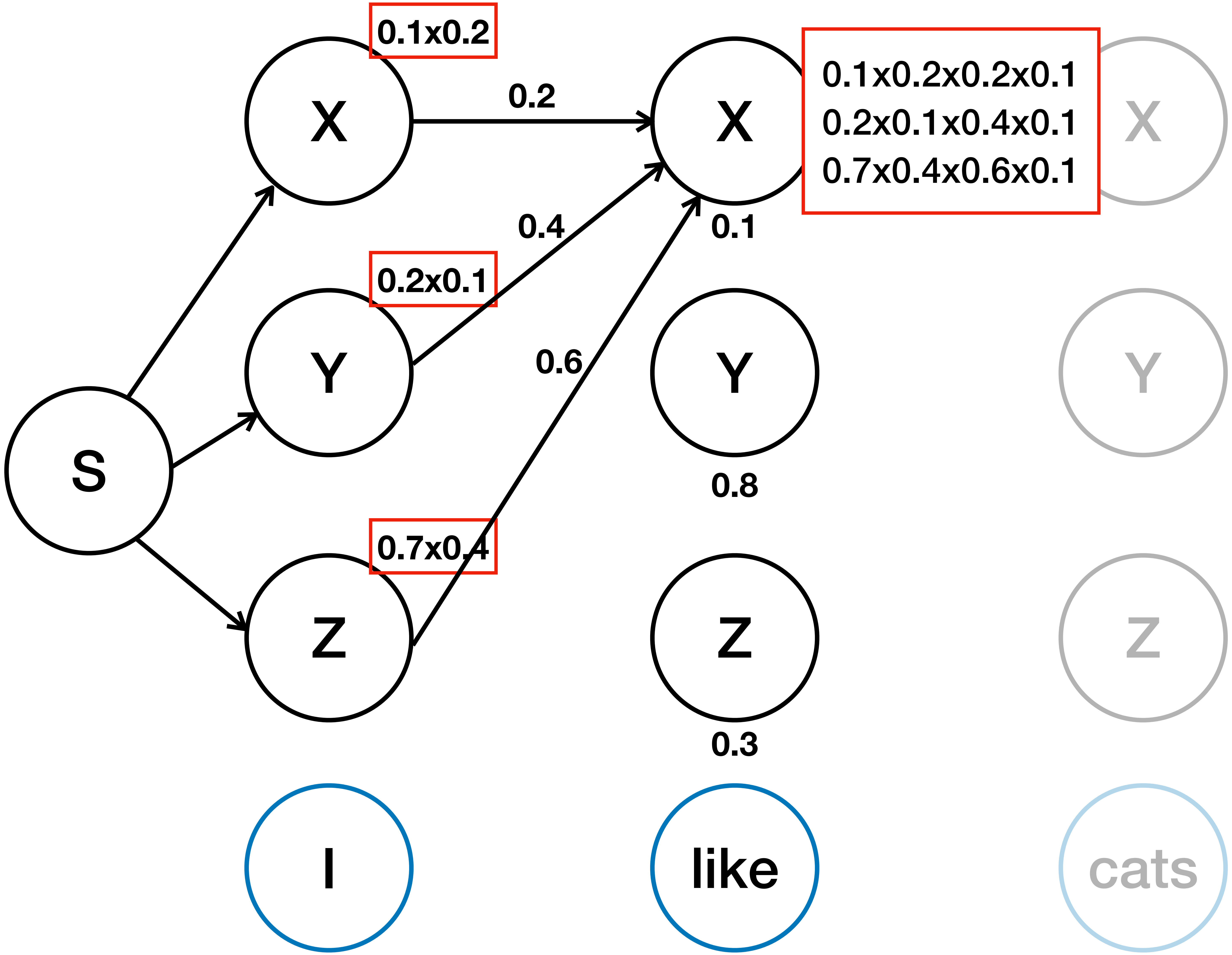
Viterbi Algorithm



	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

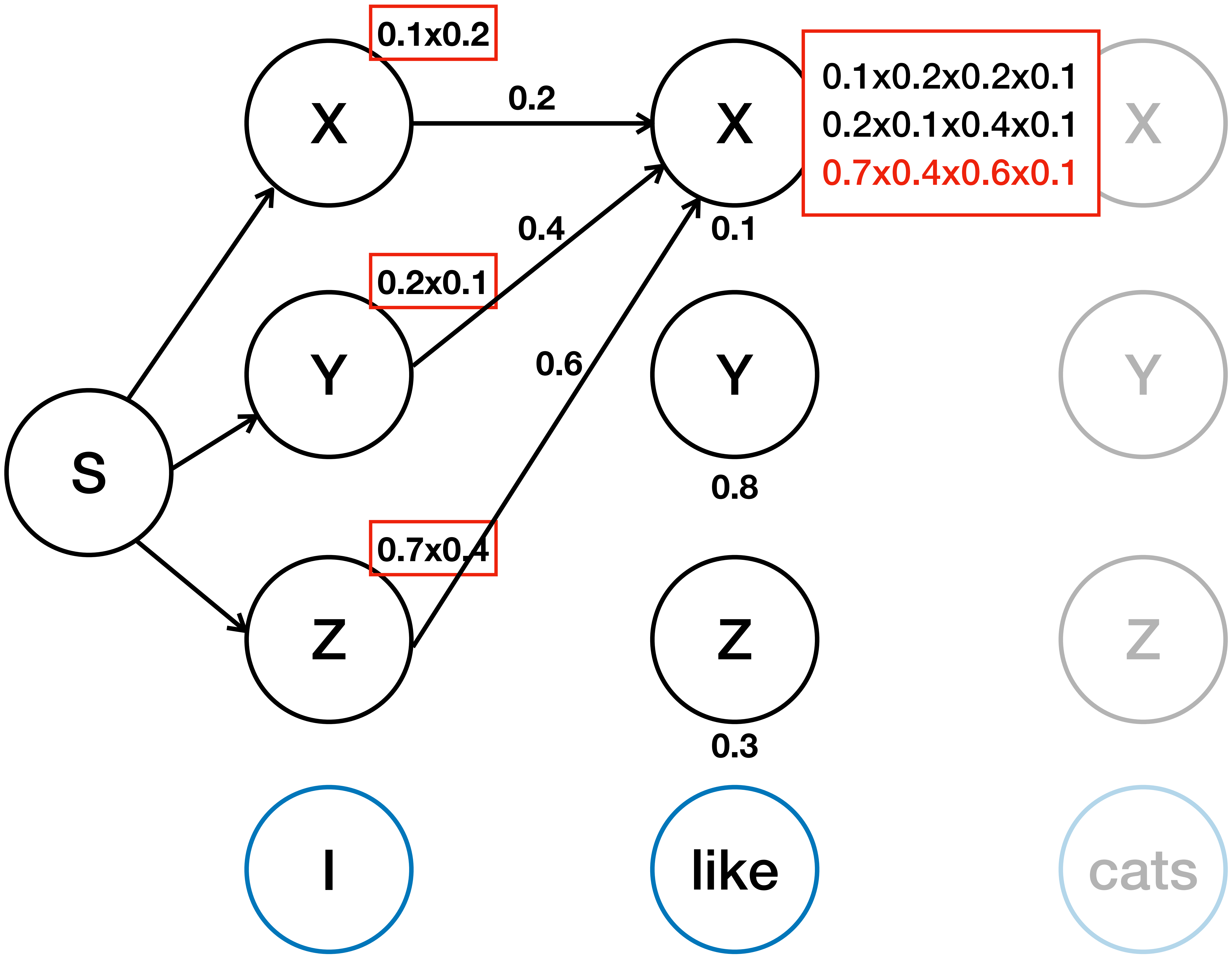
Viterbi Algorithm



	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

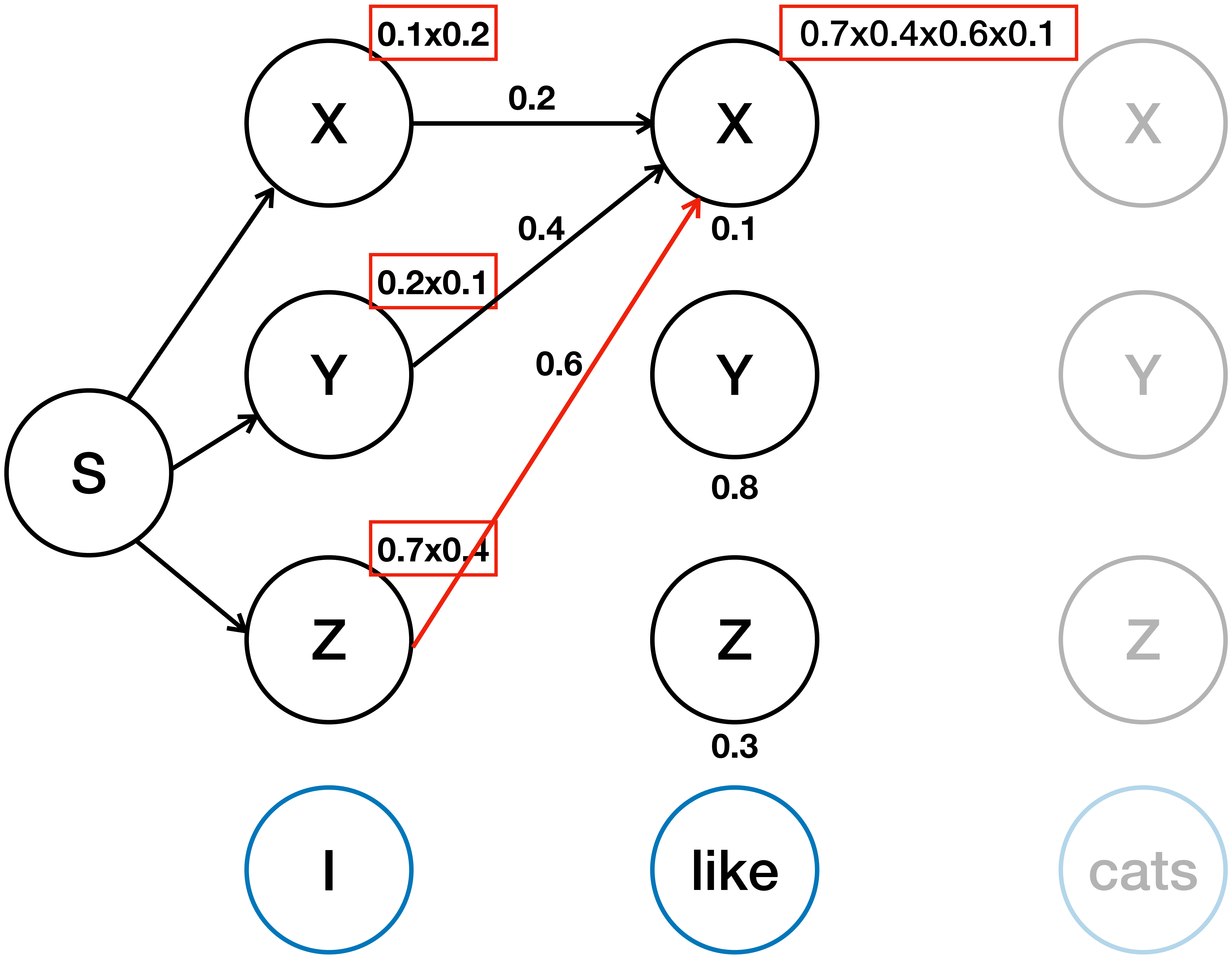
Viterbi Algorithm



	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

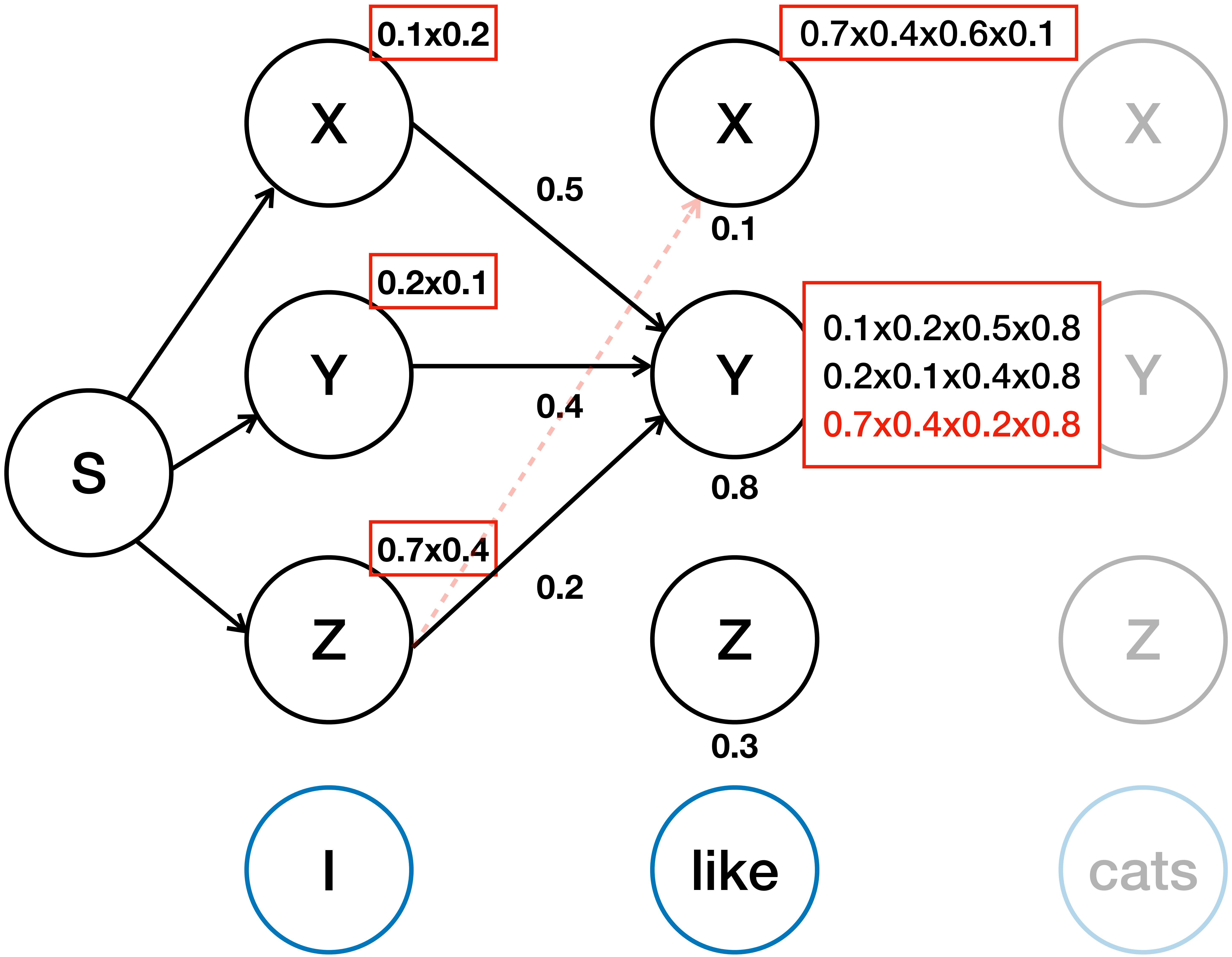
Viterbi Algorithm



	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

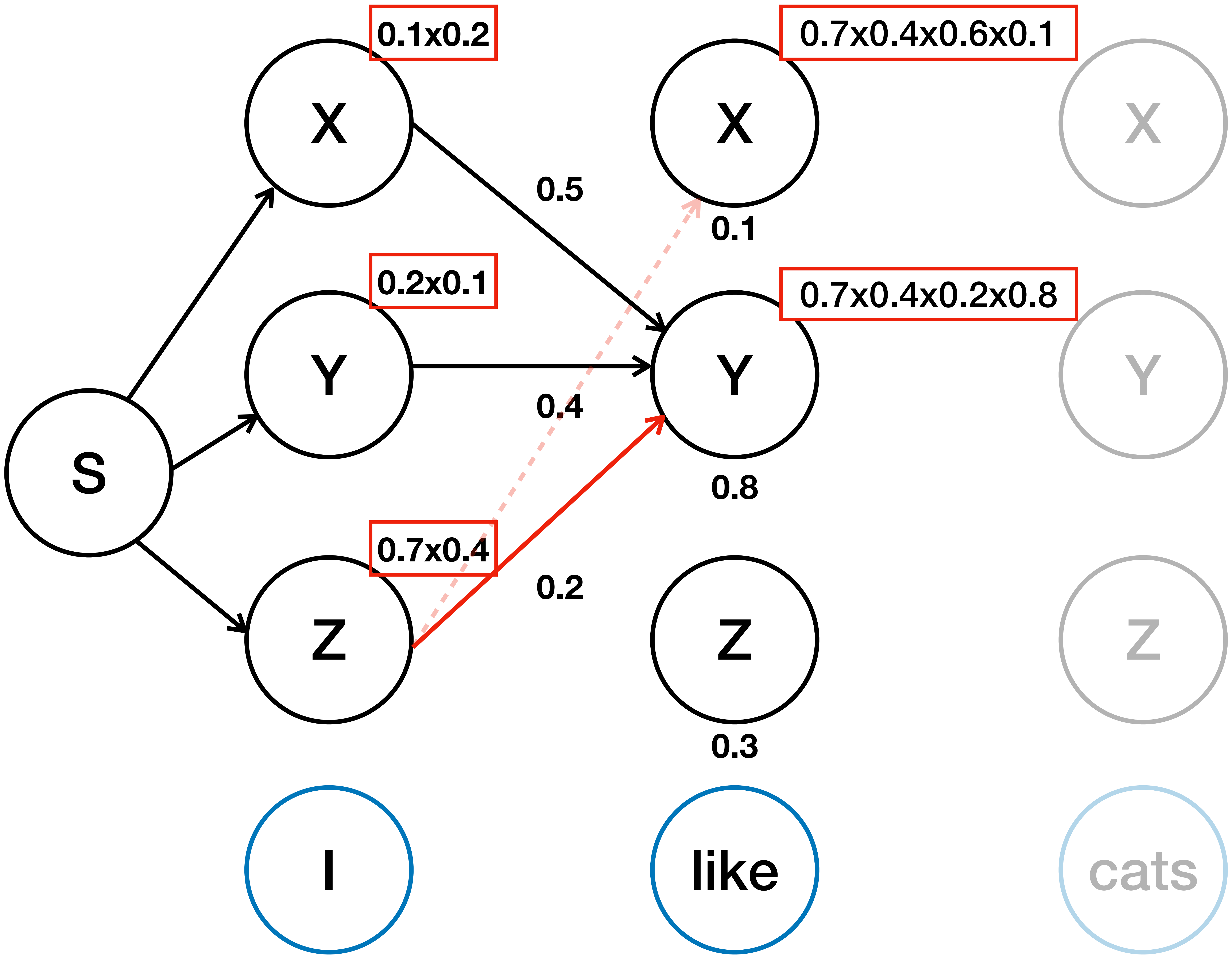
Viterbi Algorithm



	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

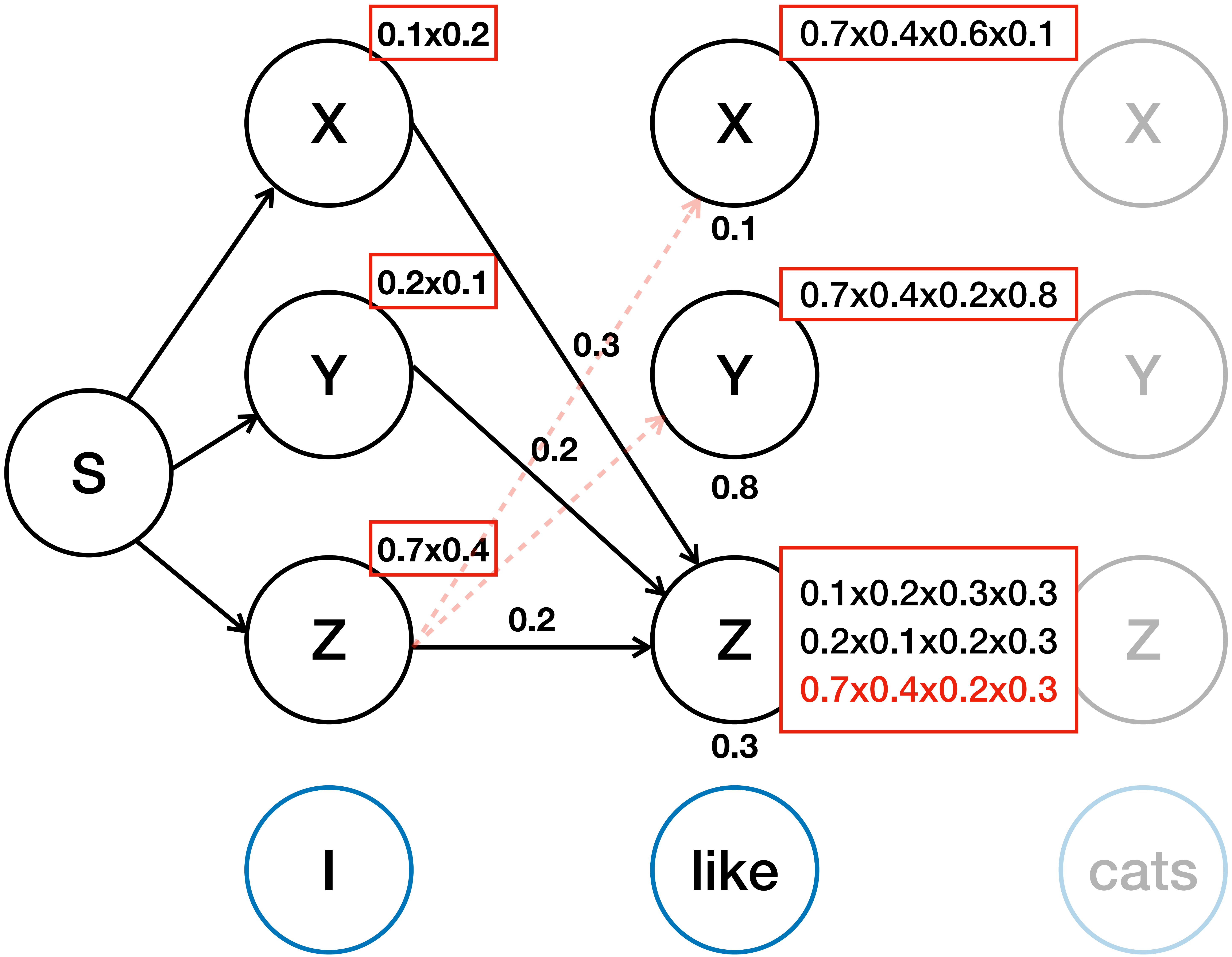
Viterbi Algorithm



	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

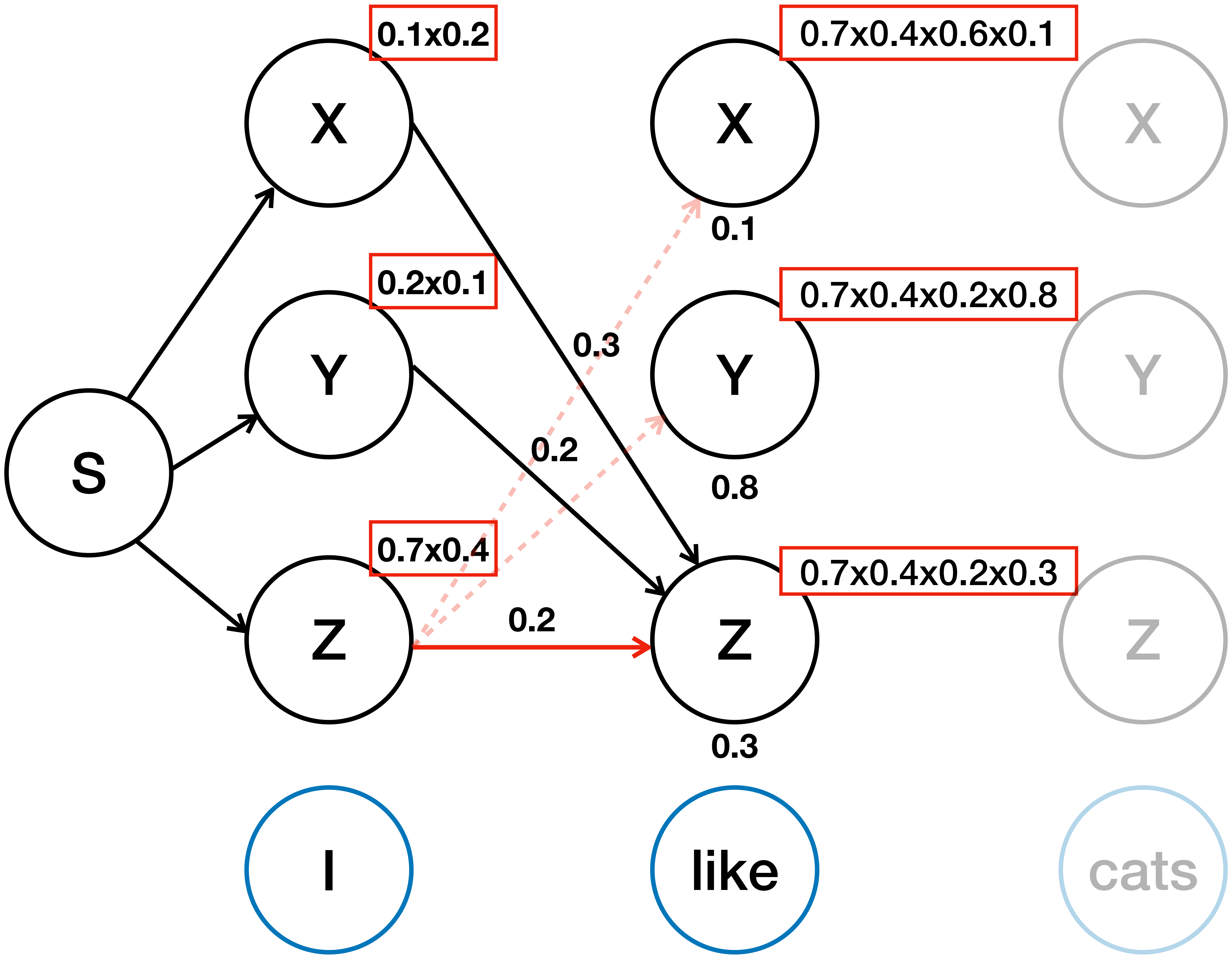
Viterbi Algorithm



	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

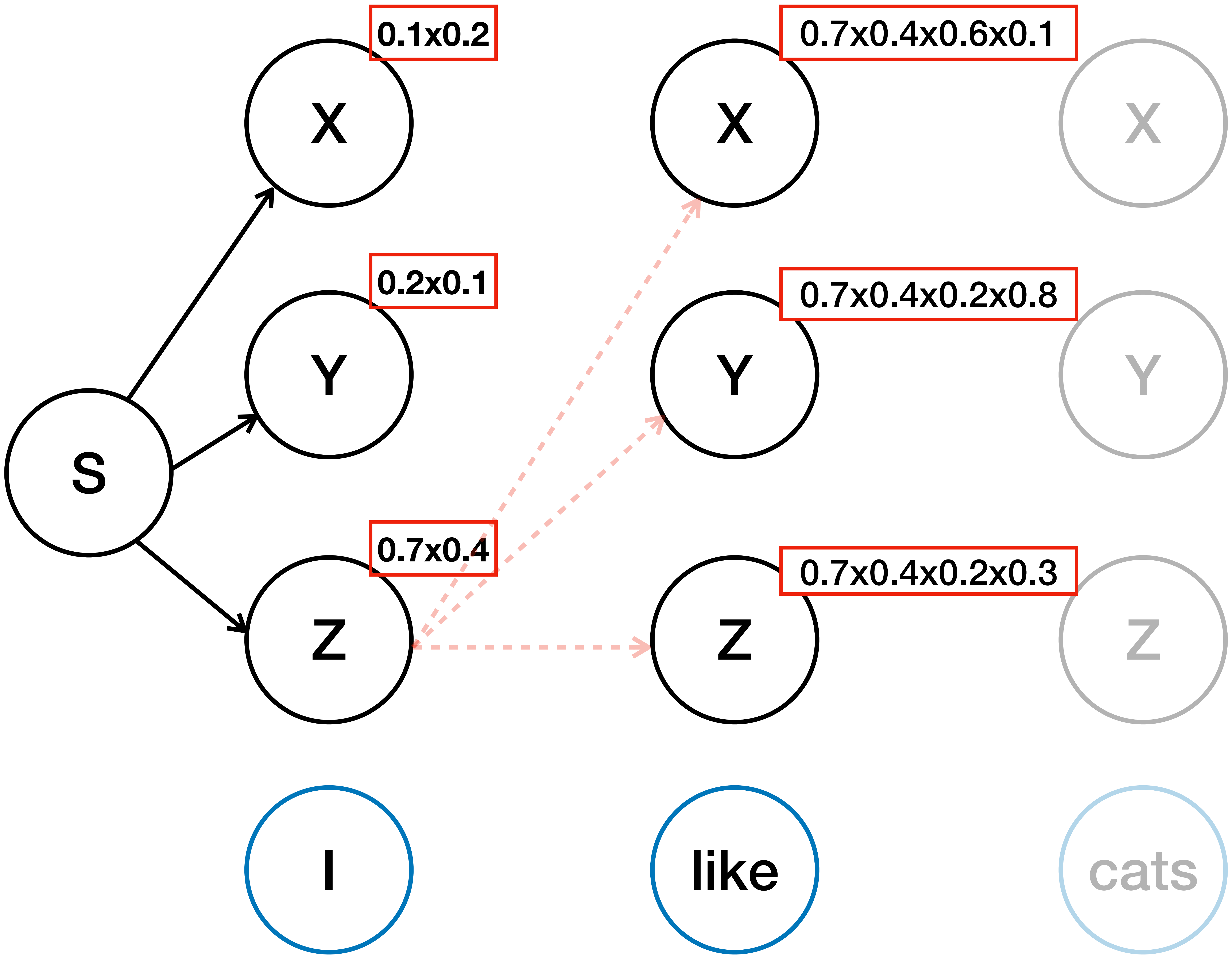
Viterbi Algorithm



	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

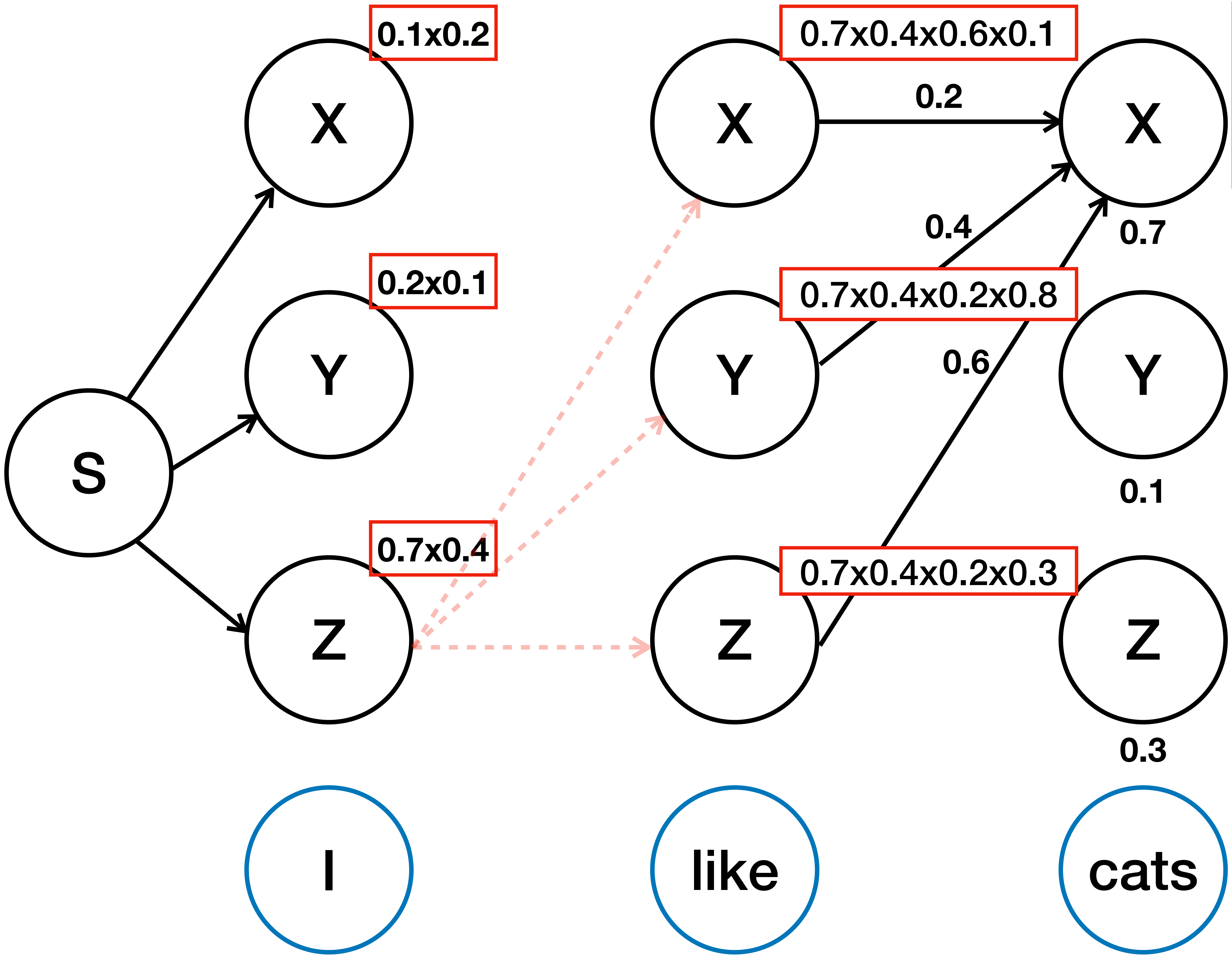
Viterbi Algorithm



	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

Viterbi Algorithm

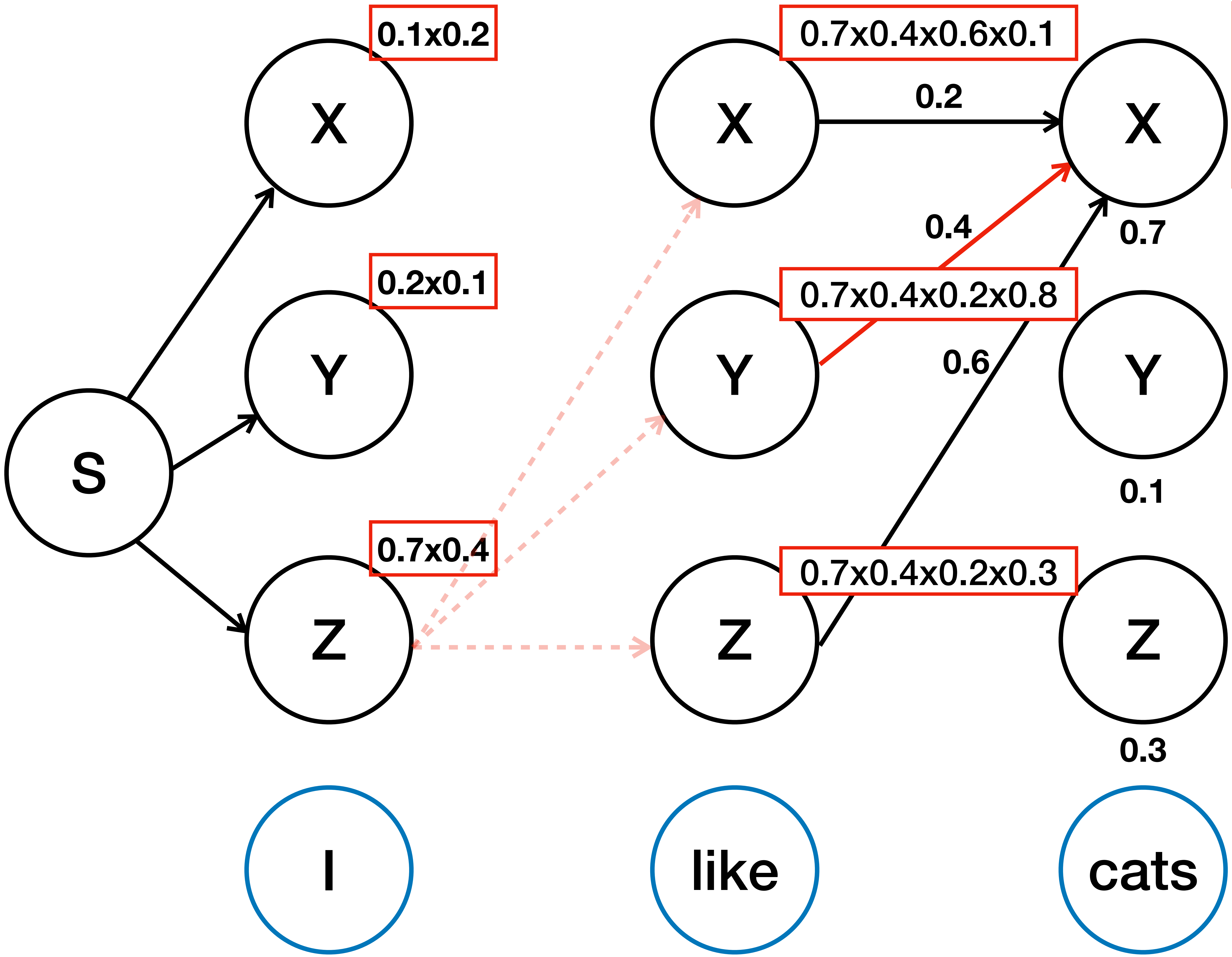


0.7x0.4x0.6x0.1x0.2x0.7 = 0.0023
0.7x0.4x0.2x0.8x0.4x0.7 = 0.0125
0.7x0.4x0.2x0.3x0.6x0.7 = 0.0070

		Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

Viterbi Algorithm

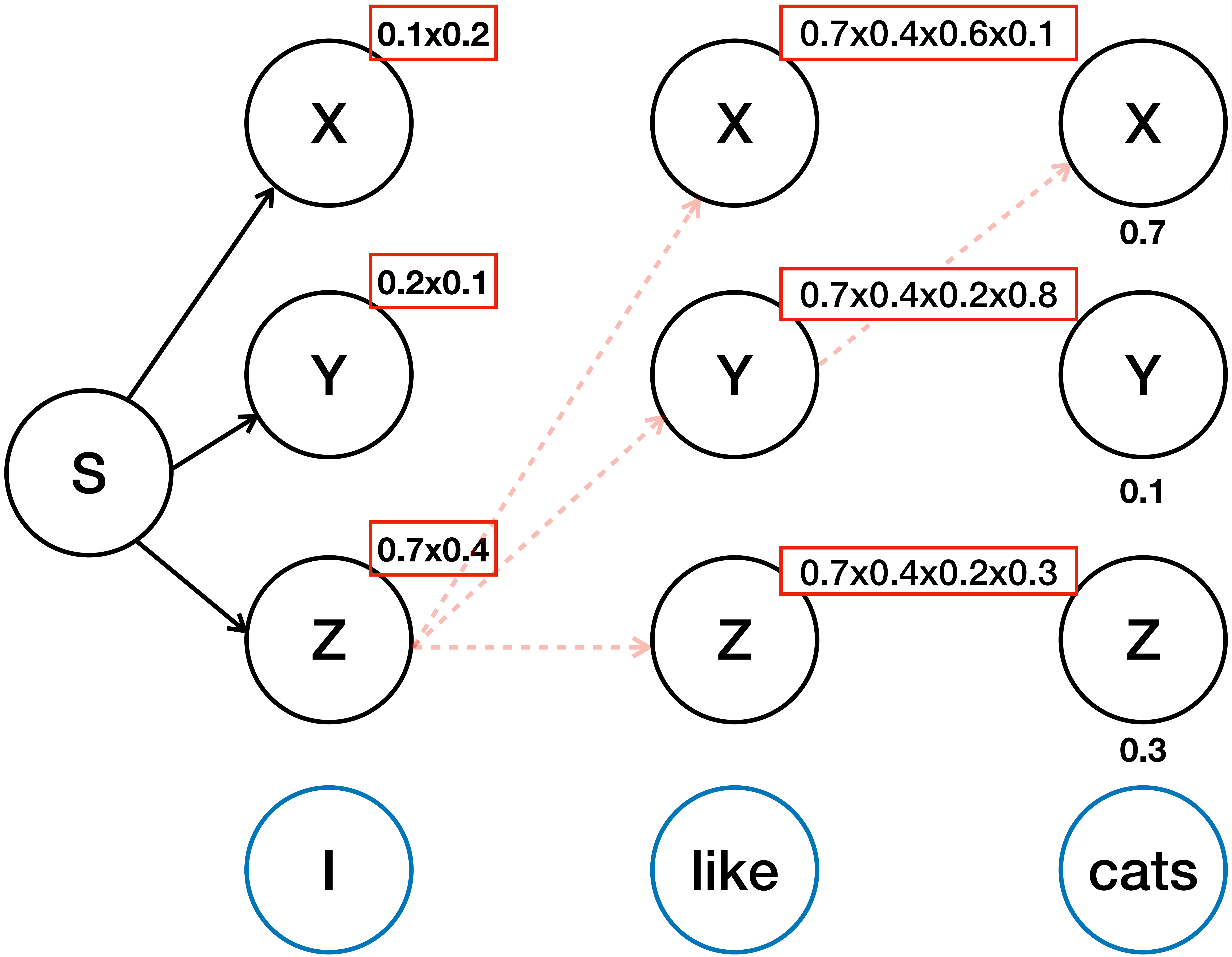


0.7x0.4x0.6x0.1x0.2x0.7 = 0.0023
0.7x0.4x0.2x0.8x0.4x0.7 = 0.0125
0.7x0.4x0.2x0.3x0.6x0.7 = 0.0070

		Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

Viterbi Algorithm

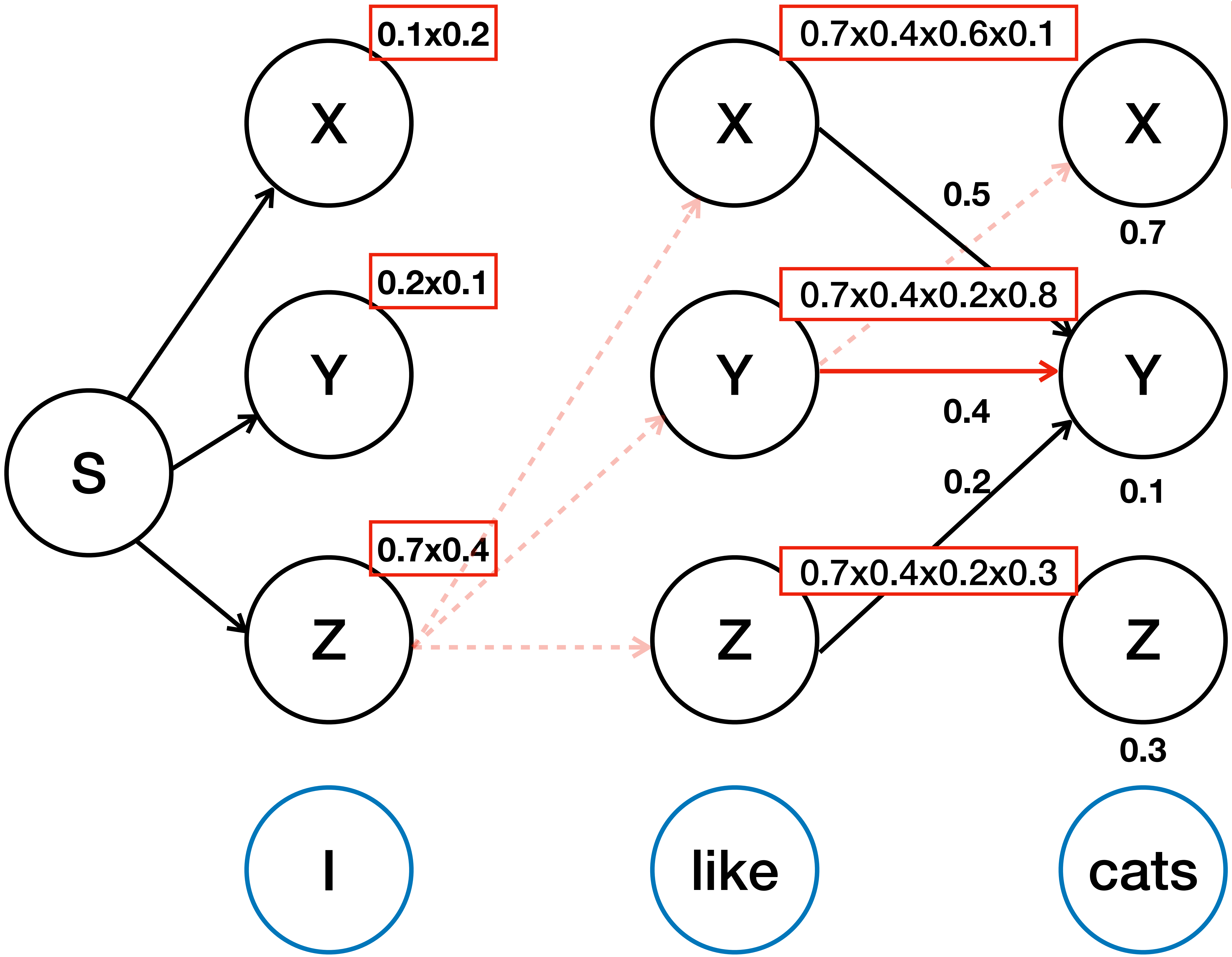


0.7x0.4x0.6x0.1x0.2x0.7 = 0.0023
0.7x0.4x0.2x0.8x0.4x0.7 = 0.0125
0.7x0.4x0.2x0.3x0.6x0.7 = 0.0070

		Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

Viterbi Algorithm



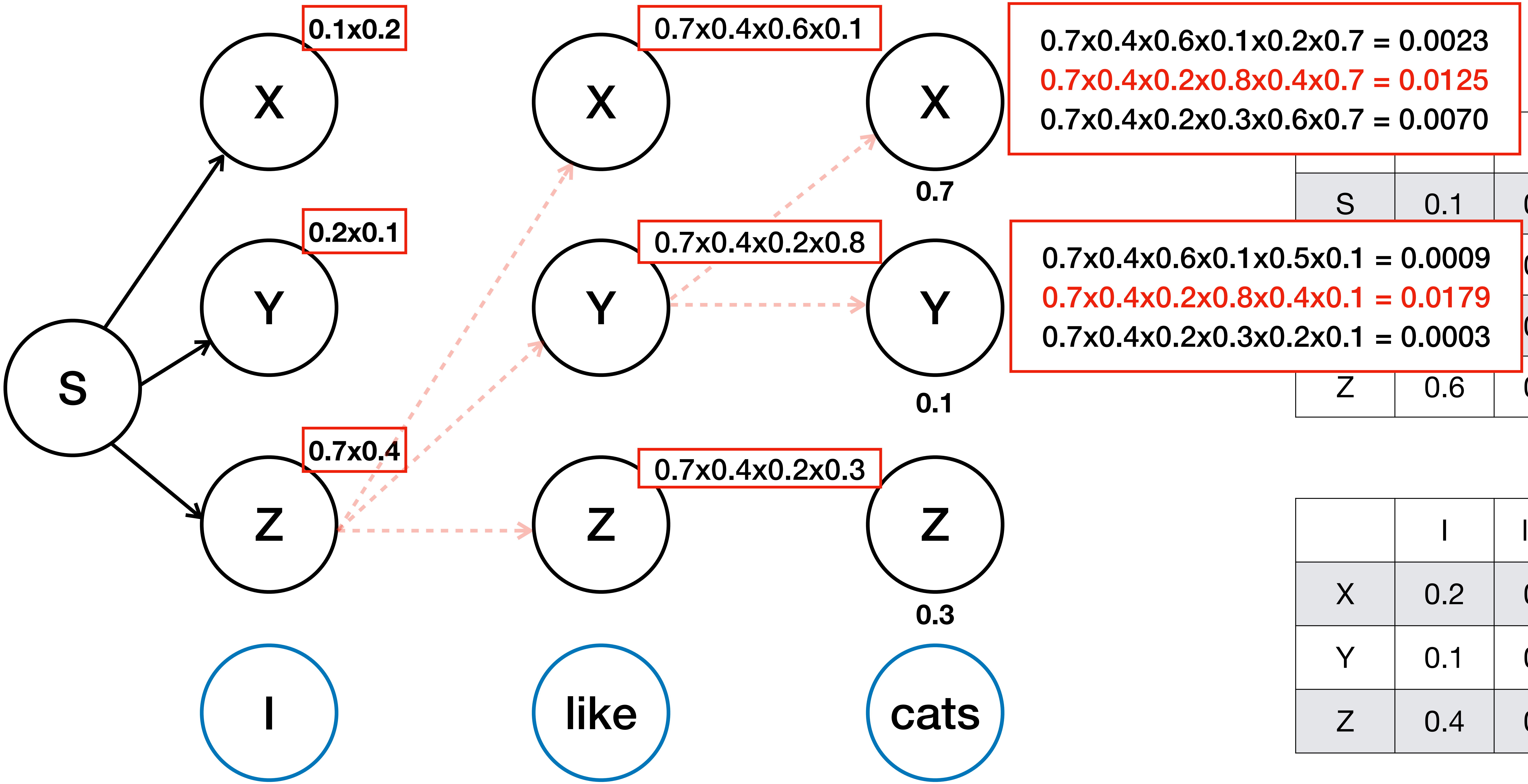
0.7x0.4x0.6x0.1x0.2x0.7 = 0.0023
0.7x0.4x0.2x0.8x0.4x0.7 = 0.0125
0.7x0.4x0.2x0.3x0.6x0.7 = 0.0070

0.7x0.4x0.6x0.1x0.5x0.1 = 0.0009
0.7x0.4x0.2x0.8x0.4x0.1 = 0.0179
0.7x0.4x0.2x0.3x0.2x0.1 = 0.0003

[illegible]	[illegible]	Y	Z
S	0.1	0.2	0.7
[illegible]	[illegible]	0.5	0.3
[illegible]	[illegible]	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

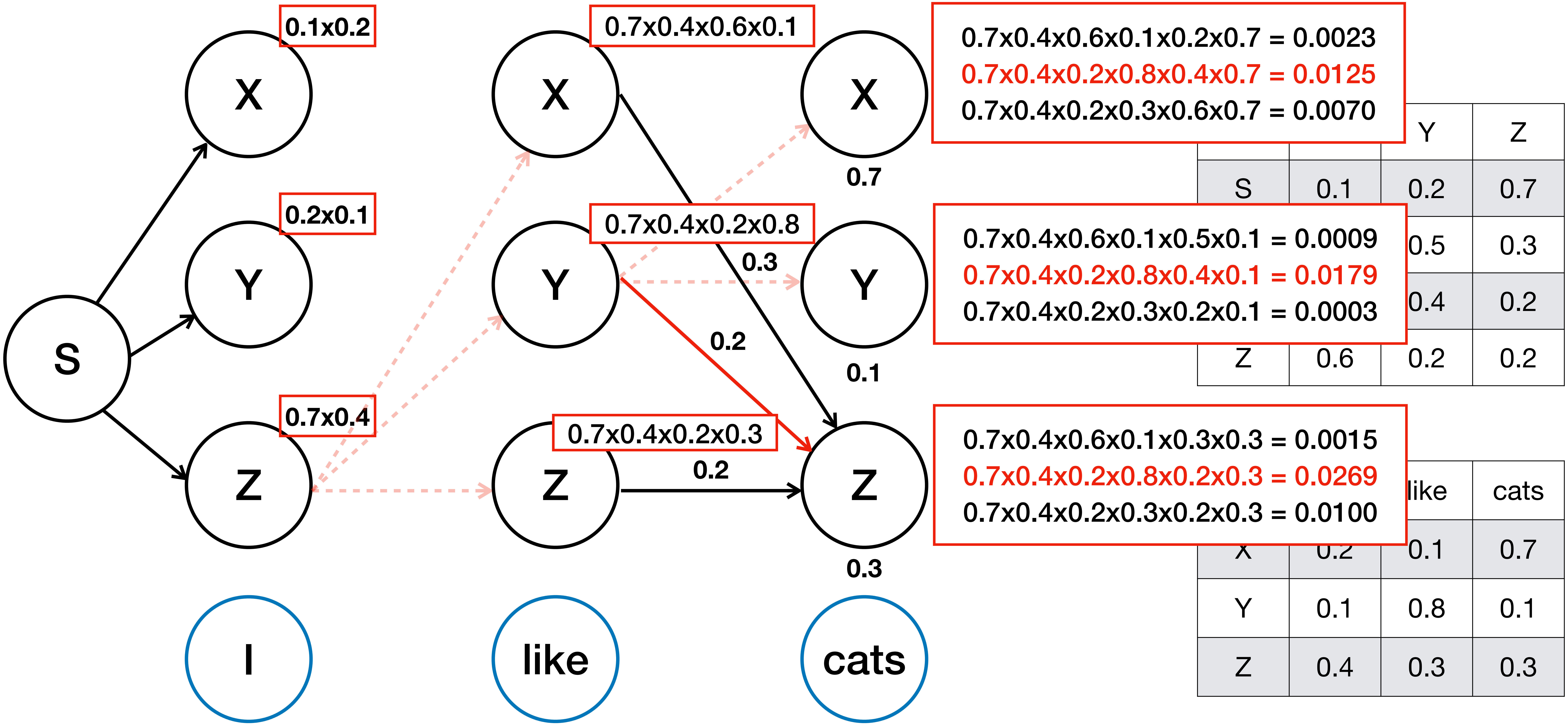
Viterbi Algorithm



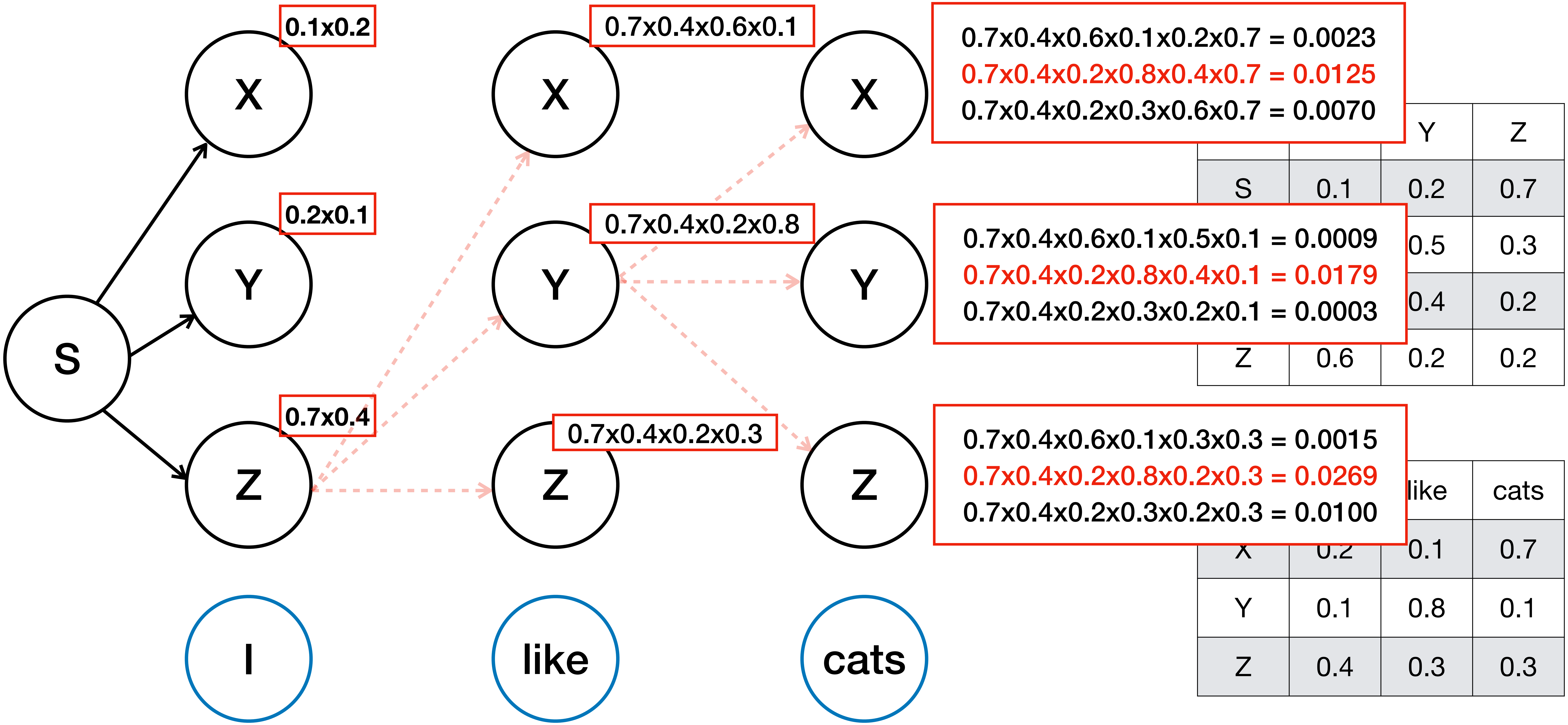
	[illegible]	Y	Z
S	0.1	0.2	0.7
[illegible]	[illegible]	0.5	0.3
[illegible]	[illegible]	0.4	0.2
Z	0.6	0.2	0.2

	I	like	cats
X	0.2	0.1	0.7
Y	0.1	0.8	0.1
Z	0.4	0.3	0.3

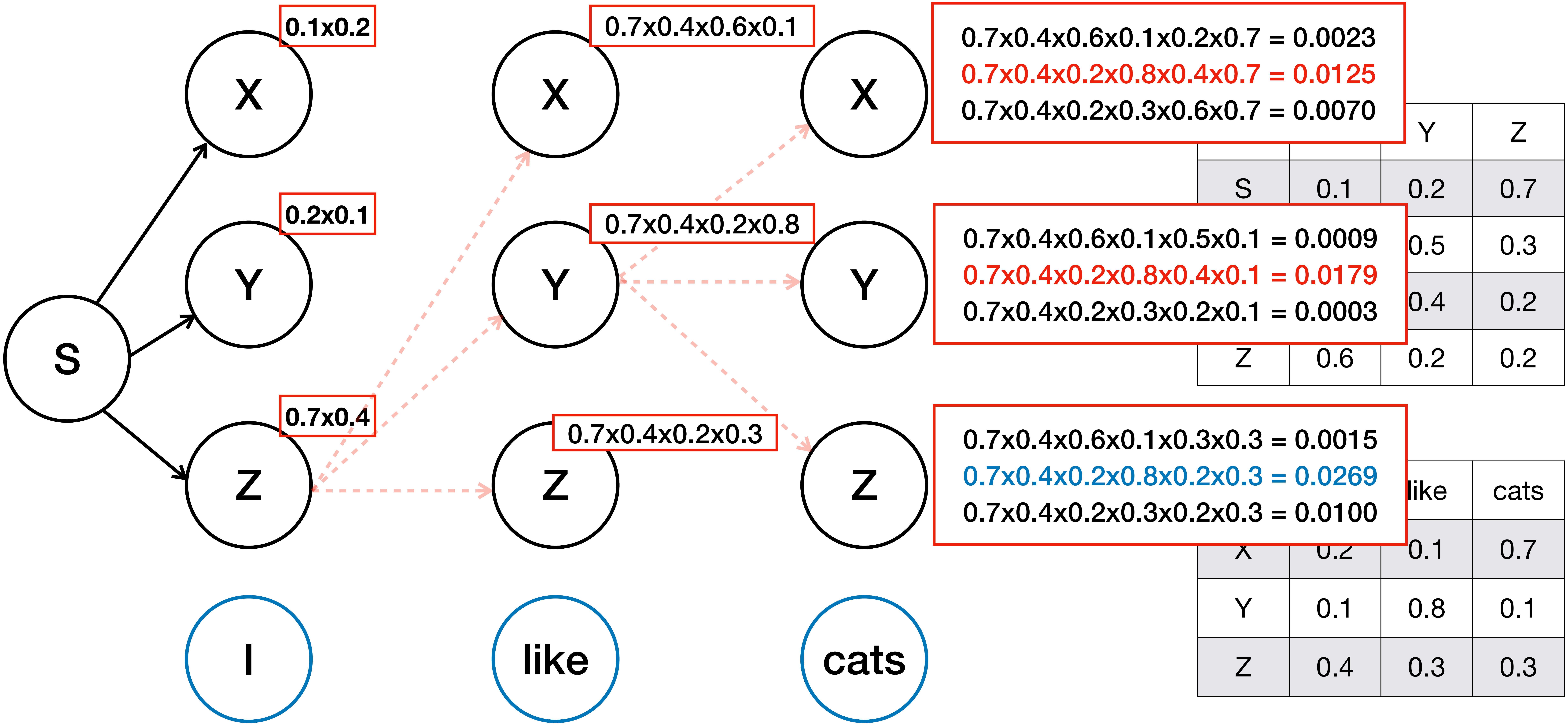
Viterbi Algorithm



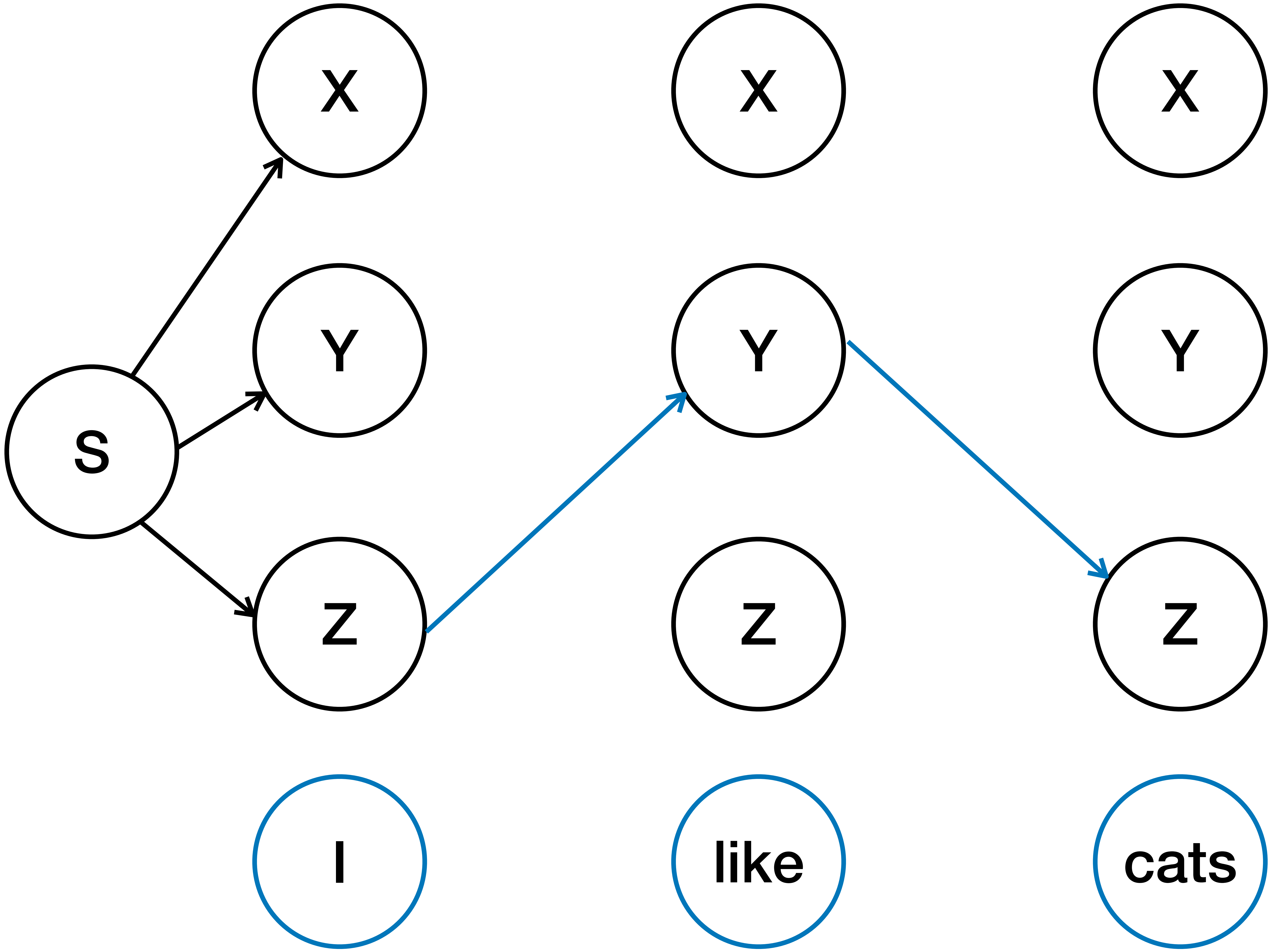
Viterbi Algorithm



Viterbi Algorithm



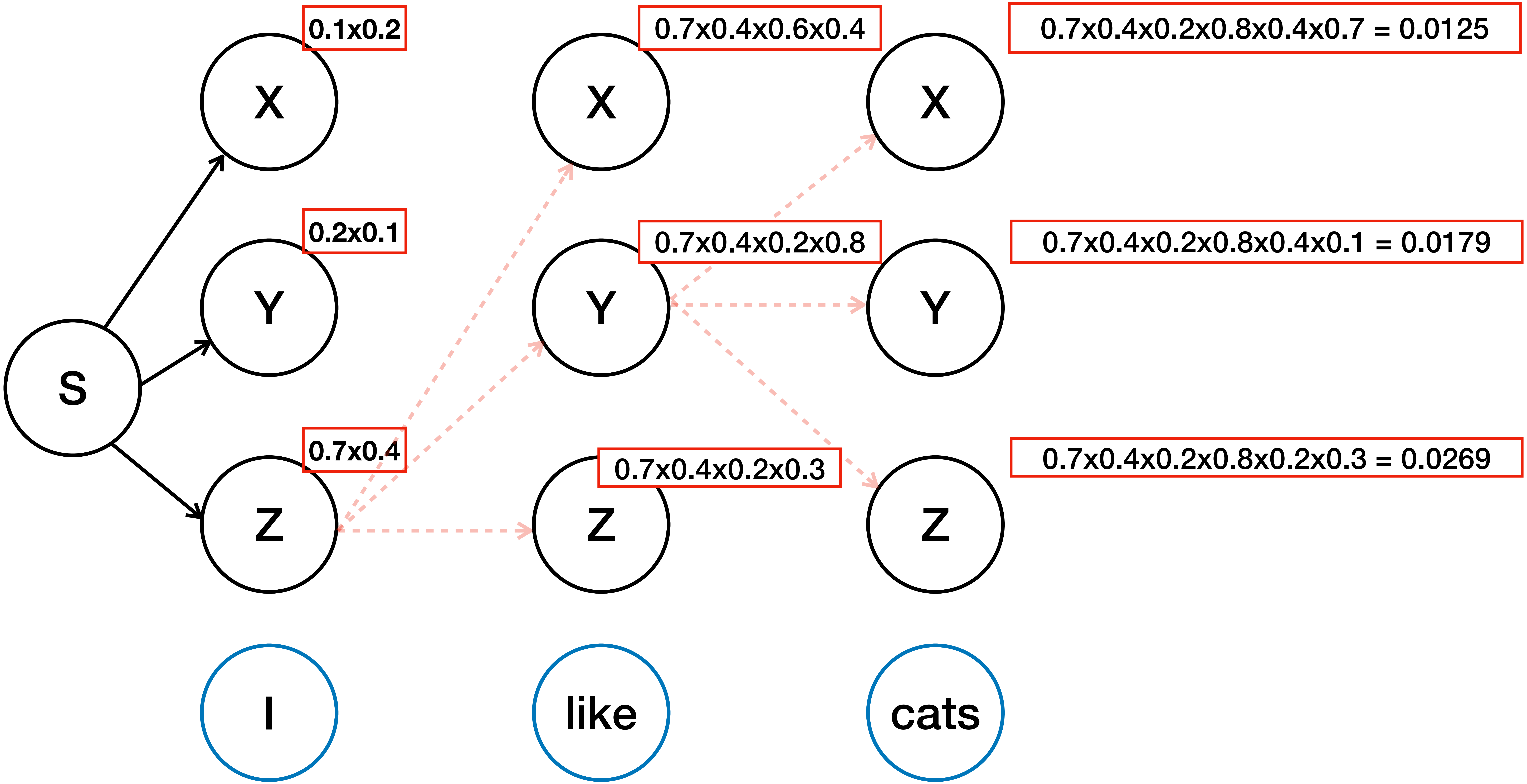
Viterbi Algorithm



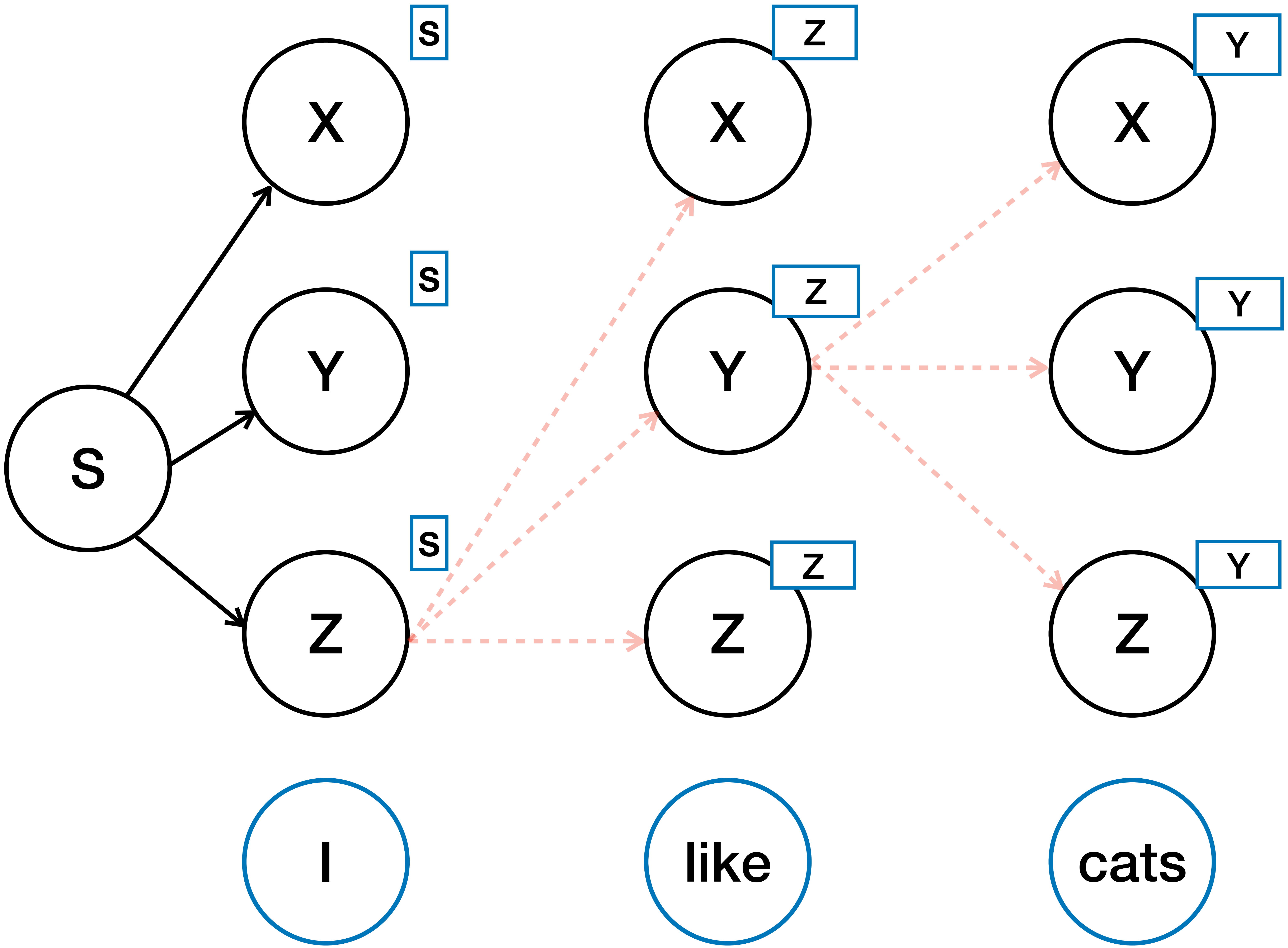
The final tags should be: <Z, Y, Z>

How do we know the path?
Answer: use a backtracking matrix

Viterbi Algorithm



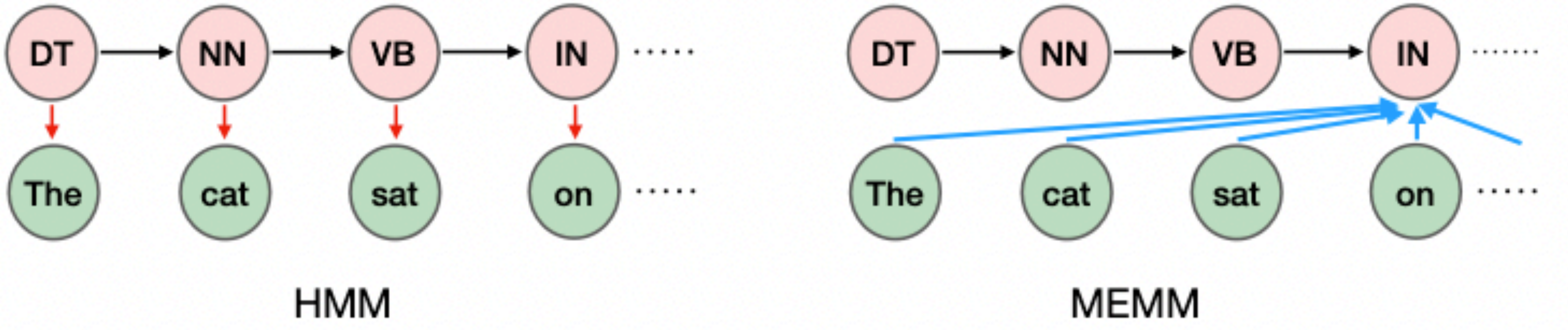
Viterbi Algorithm



The backtracking matrix keeps track of the best node from the previous step.

MEMM

MEMM



$$P(S \mid O) = \prod_{i=1}^{n} P(s_i \mid s_{i-1}, s_{i-2}, \ldots, s_1, O)$$

$$= \prod_{i=1}^{n} P(s_i \mid s_{i-1}, O)$$

$$O = \langle o_1, o_2, \ldots, o_n \rangle$$

Markov assumption:
Bigram MEMM

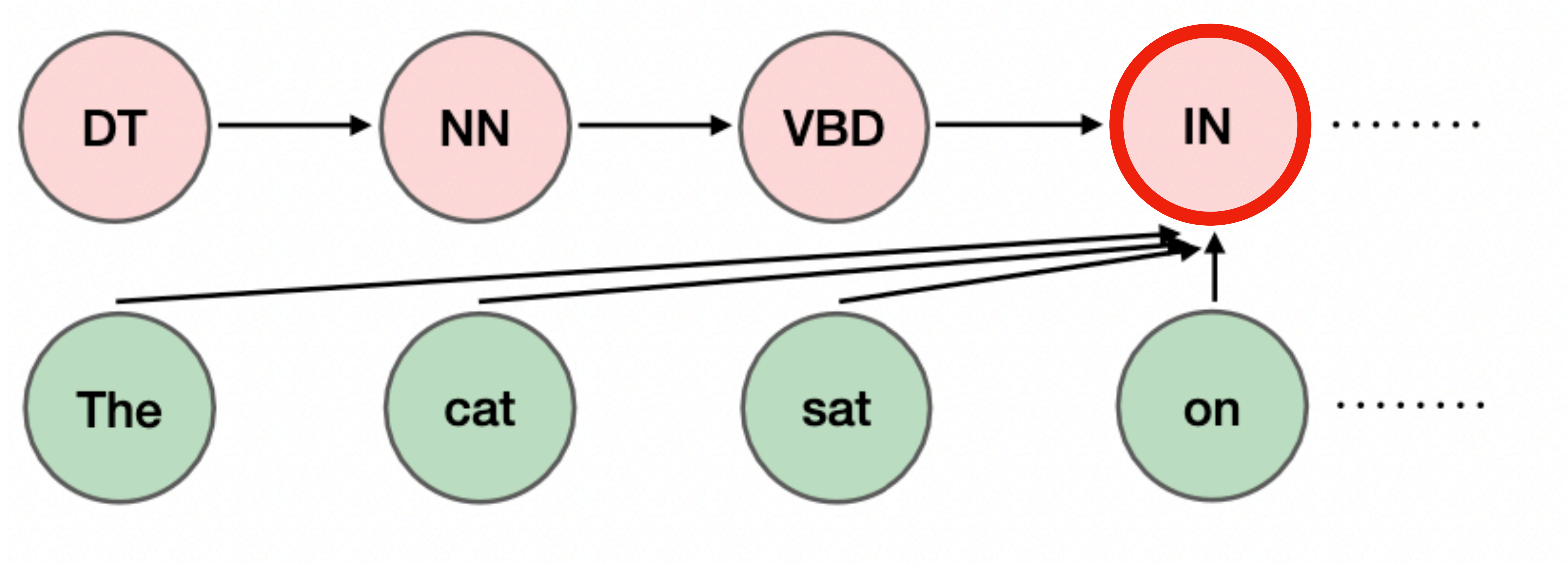
$$P(s_i = s \mid s_{i-1}, O) \propto \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, s_{i-1}, O, i))$$

weights

features

Important: you can define features over entire word sequence O!

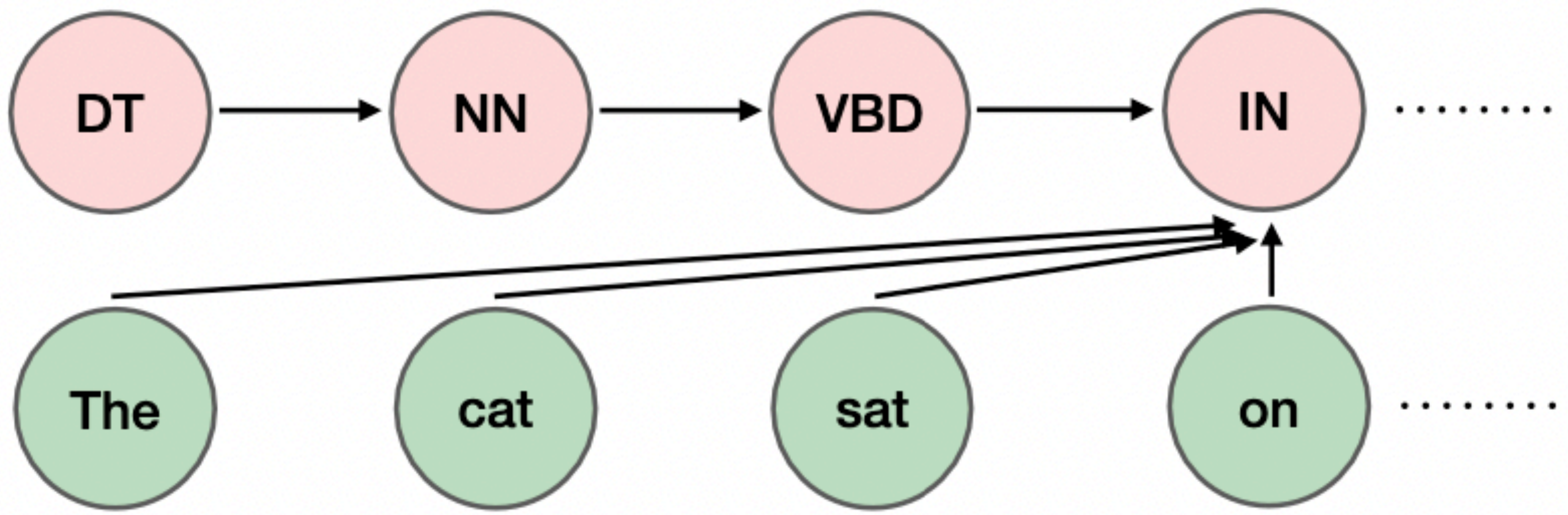
MEMM



- To predict the red node, the 4-gram MEMM conditions on the “prior tags” (DT, NN, VBD, IN) and the observations in the window (The, cat, sat, on)

- Prior tags and observations will be transformed into features (some sort of vector representation)

MEMM



We can design feature templates:

o_{i-2} = animal & s_{i-1} = VBD

s_{i-2} = NN & s_{i-1} = VBD

s_{i-3} = NNP

For predicting the IN tag position, the feature vector would be [1, 1, 0]. In practice, the final feature vector might be more complicated than this — the prior tags might be represented as one-hot vectors in addition to the template feature vectors.