Precept 4: HMMs, MEMMs, CRFs

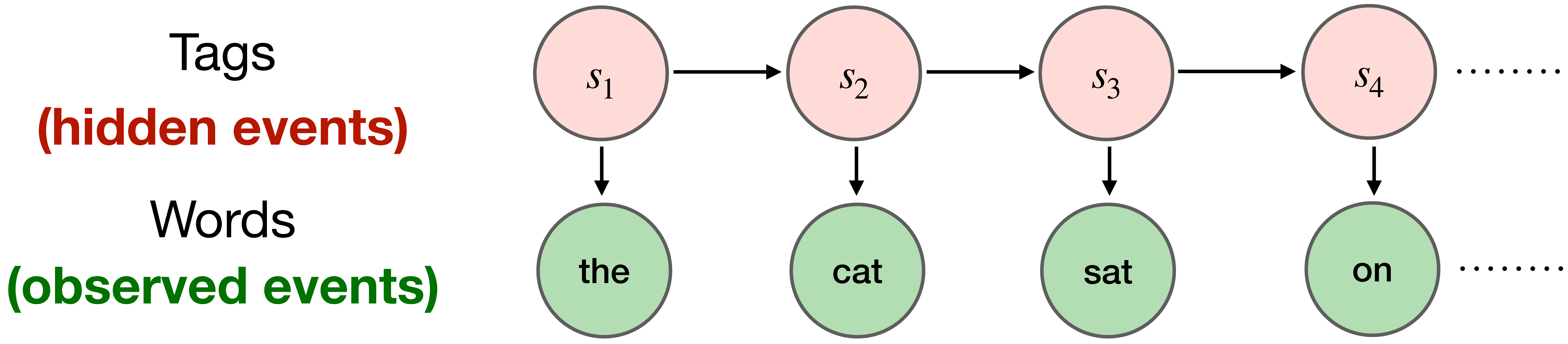
COS 484

Tyler Zhu (Viterbi figures adapted from Howard Chen, lecture)

Today's Plan

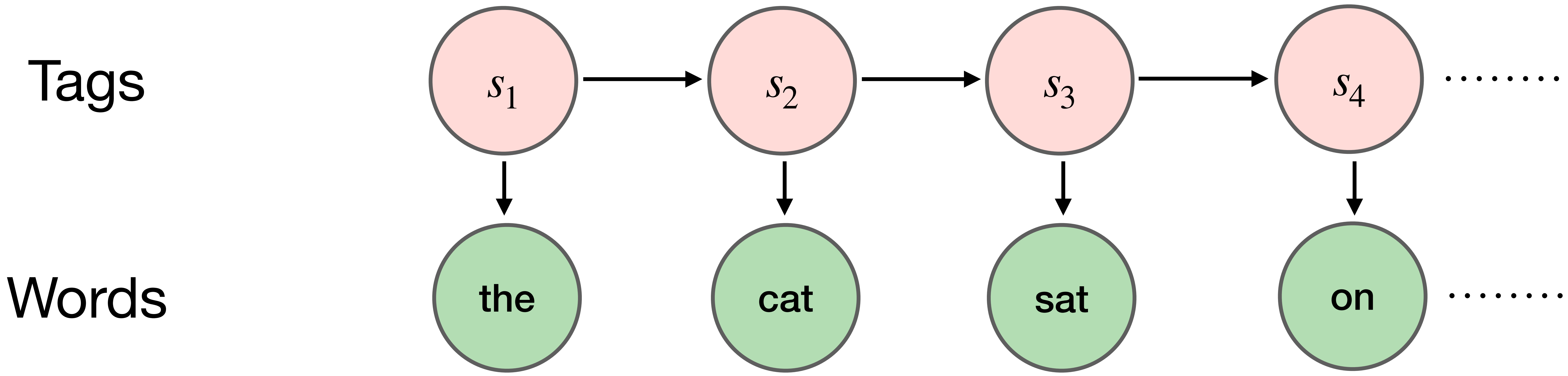
1. Hidden Markov Models (20 min)
2. Maximum Entropy Markov Models (20 min)
3. Conditional Random Fields (5 min)

Hidden Markov Model (HMM)



- We don't normally see sequences of POS tags in text
- However, we do observe the words!
- The HMM allows us to *jointly reason* over both **hidden** and **observed** events.
  - Assume that each position has a tag that generates a word

HMMs: Assumptions



What are the key assumptions?

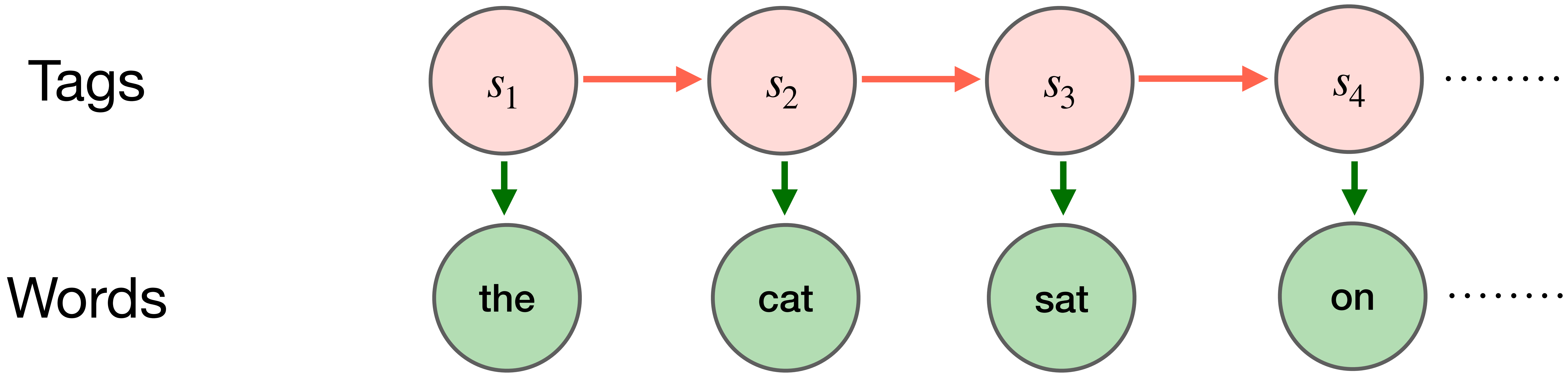
1. Markov assumption:

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

2. Output independence:

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

HMMs: Assumptions



What are the key assumptions?

1. Markov assumption:

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

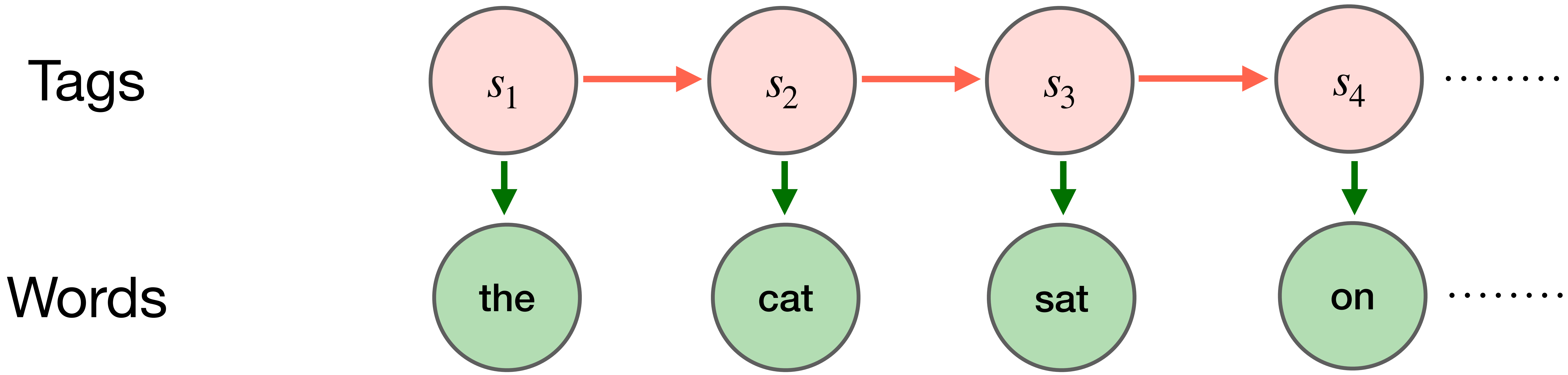
Transition probabilities

2. Output independence:

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

Emission probabilities

HMMs: Training



What do we train and how?

Transition probabilities

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

	DT	NN	VB
∅	0.5	0.3	0.2
DT	0.1	0.5	0.4
NN	0.2	0.3	0.5
VB	0.4	0.3	0.3

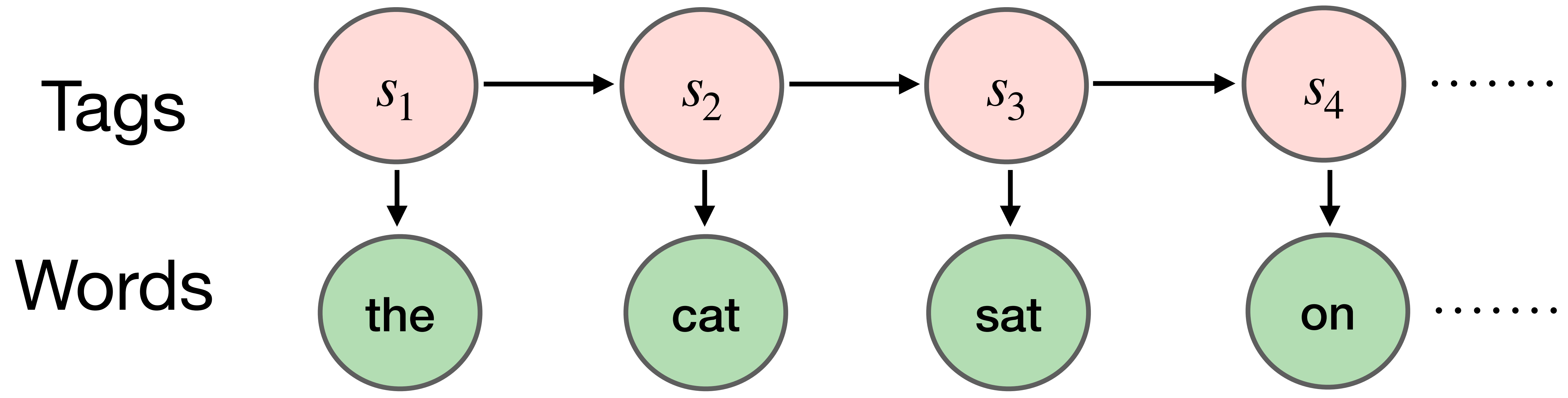
From training data!

Emission probabilities

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

	the	cat	runs
DT	0.4	0.5	0.1
NN	0.5	0.4	0.1
VB	0.2	0.3	0.5

HMMs: Inference



Task: Find the most probable sequence of states $S = s_1, s_2, \ldots, s_n$ given the observations $O = o_1, o_2, \ldots, o_n$

$$\hat{S} = \arg\max_S P(S \mid O) = \arg\max_S \frac{P(O \mid S)P(S)}{P(O)}$$ [Bayes' rule]

$$= \arg\max_S P(O \mid S)P(S)$$ [$P(O)$ doesn't depend on S!]

$$= \arg\max_{s_1, s_2, \ldots s_n} \prod_{i=1}^{n} P(o_i \mid s_i)P(s_i \mid s_{i-1})$$ [Markov assumption]

How can we maximize this?
Search over all state sequences?

[SP23 Midterm] Problem 5: Lie Detection

You are building a lie detector taking as input a stream of recorded behaviors:

- $x_t \in \{a, b\}$, i.e., face touching (a) or blinking (b)

Detector at every moment then predicts one of four labels:

- $y_t \in \{N, U, L, H\}$, i.e., Neutral (N), Unclear (U), Lying (L), and Honest (H)

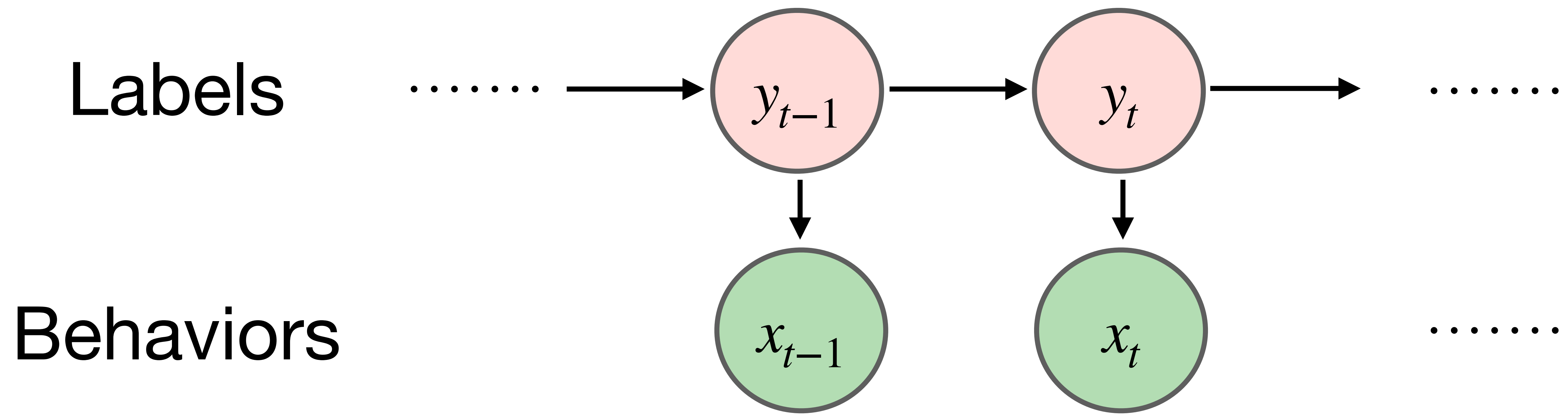
Dataset is triplets (y_{t-1}, y_t, x_t): 9x of (N, L, a), 9x of (U, L, b), 1x of (N, H, b)

- Labels y_t come from body language experts' annotation

Q1: You use the above data to build an HMM. What is $P(x_t = b \,|\, y_{t-1} = N)$?

[SP23 Midterm] Problem 5: Lie Detection

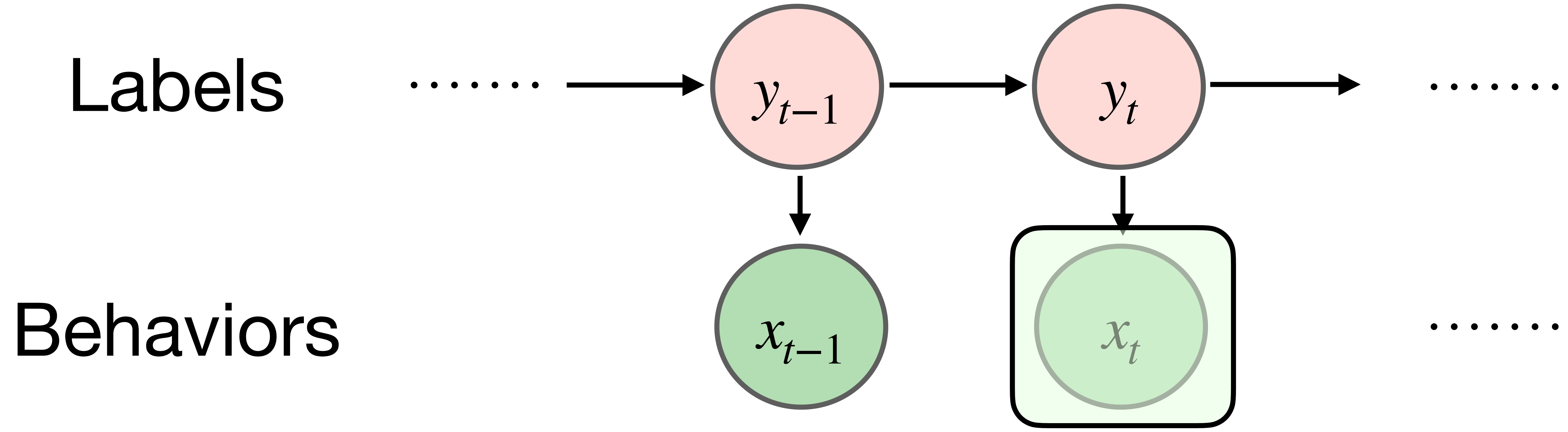
Q1: You use the above data to build an HMM. What is $P(x_t = b \,|\, y_{t-1} = N)$?



Simplified model of the HMM we care about.

[SP23 Midterm] Problem 5: Lie Detection

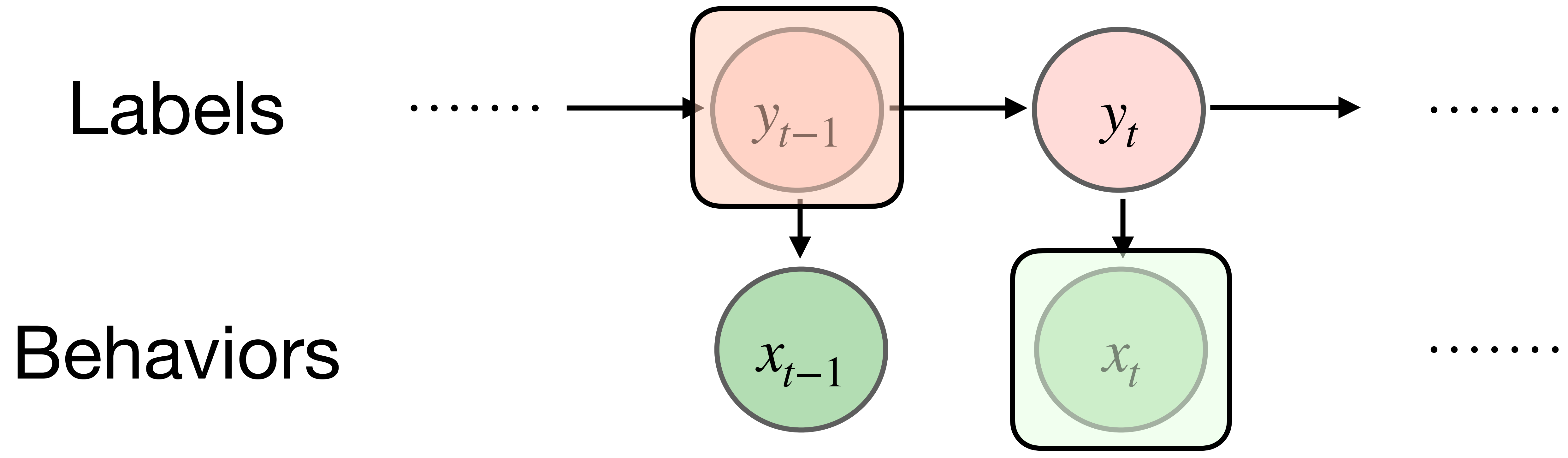
Q1: You use the above data to build an HMM. What is $P(x_t = b \,|\, y_{t-1} = N)$?



Simplified model of the HMM we care about.

[SP23 Midterm] Problem 5: Lie Detection

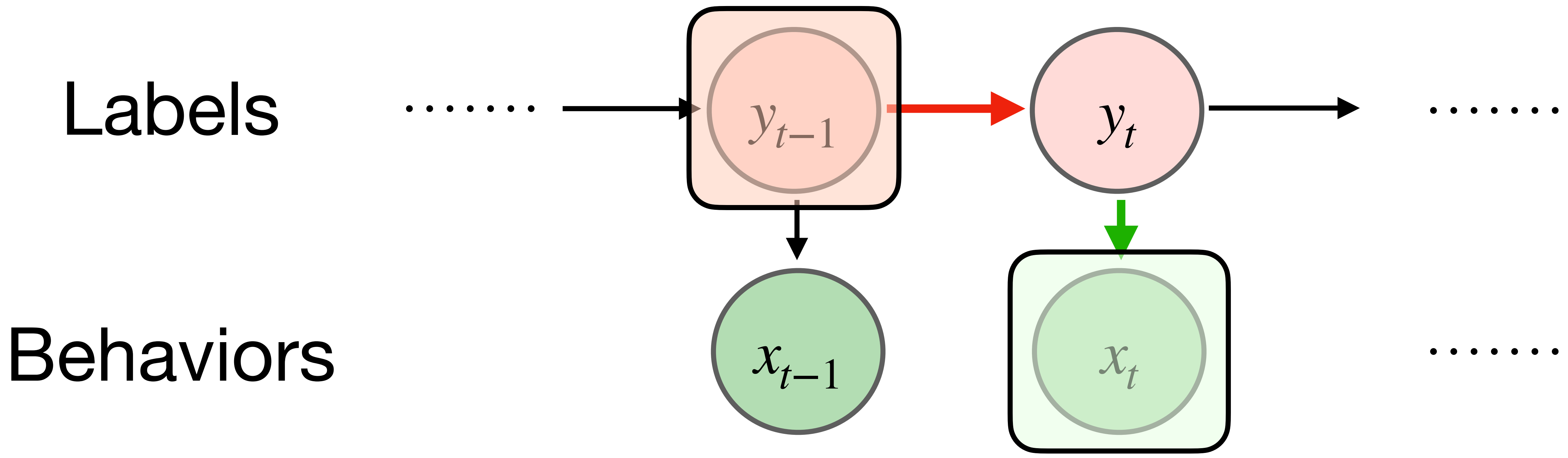
Q1: You use the above data to build an HMM. What is $P(x_t = b \mid y_{t-1} = N)$?



Simplified model of the HMM we care about.

[SP23 Midterm] Problem 5: Lie Detection

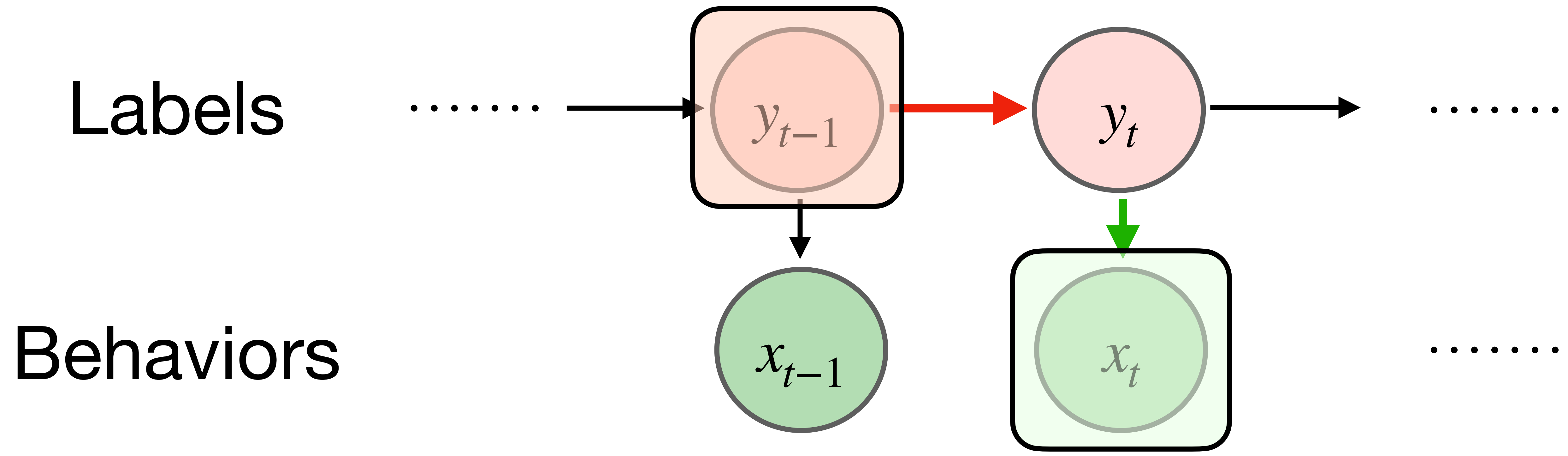
Q1: You use the above data to build an HMM. What is $P(x_t = b \,|\, y_{t-1} = N)$?



Simplified model of the HMM we care about.

[SP23 Midterm] Problem 5: Lie Detection

Q1: You use the above data to build an HMM. What is $P(x_t = b \mid y_{t-1} = N)$?

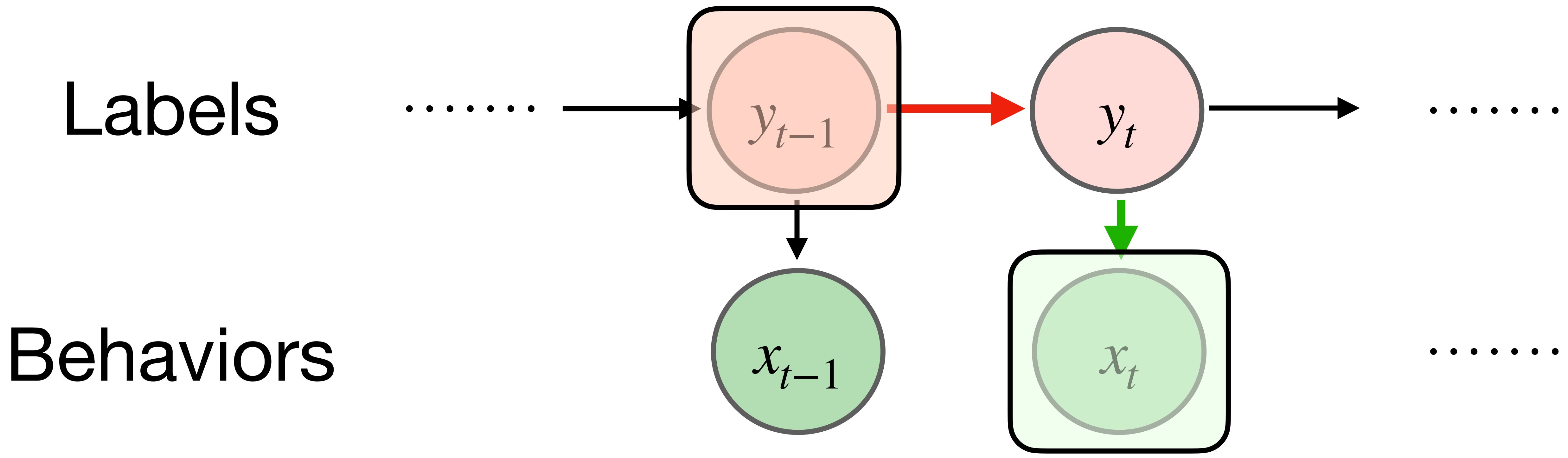


Simplified model of the HMM we care about.

We need to condition over y_t using our emission and transition probabilities.

[SP23 Midterm] Problem 5: Lie Detection

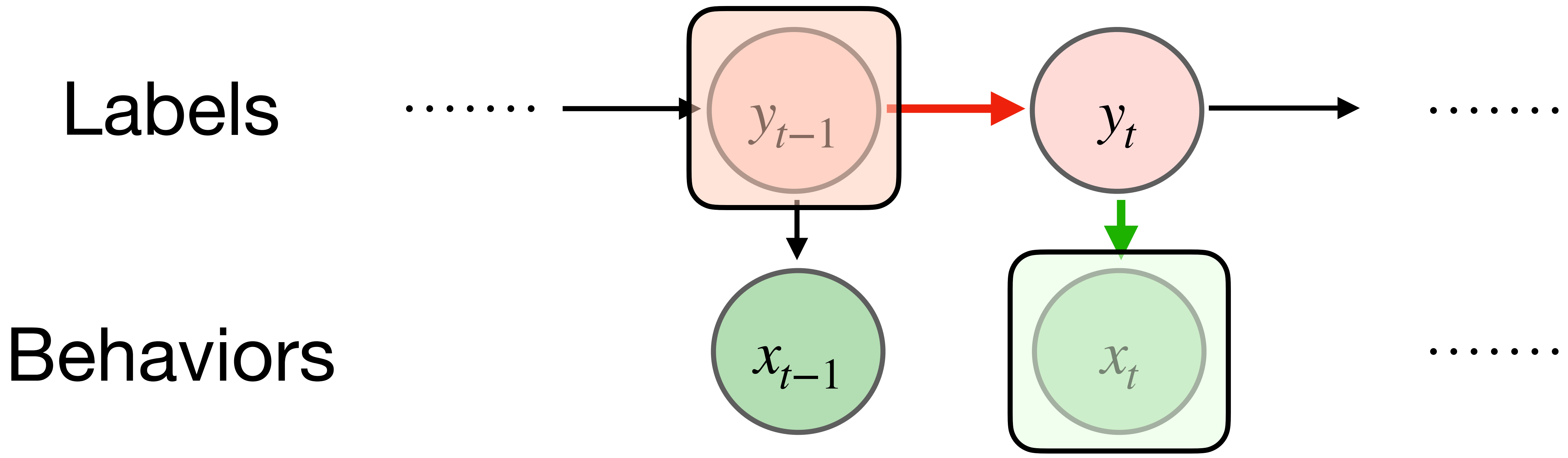
Q1: You use the above data to build an HMM. What is $P(x_t = b \mid y_{t-1} = N)$?



$$P(x_t = b \mid y_{t-1} = N) = \sum_{y_t} P(x_t = b, y_t \mid y_{t-1} = N)$$

[SP23 Midterm] Problem 5: Lie Detection

Q1: You use the above data to build an HMM. What is $P(x_t = b \mid y_{t-1} = N)$?

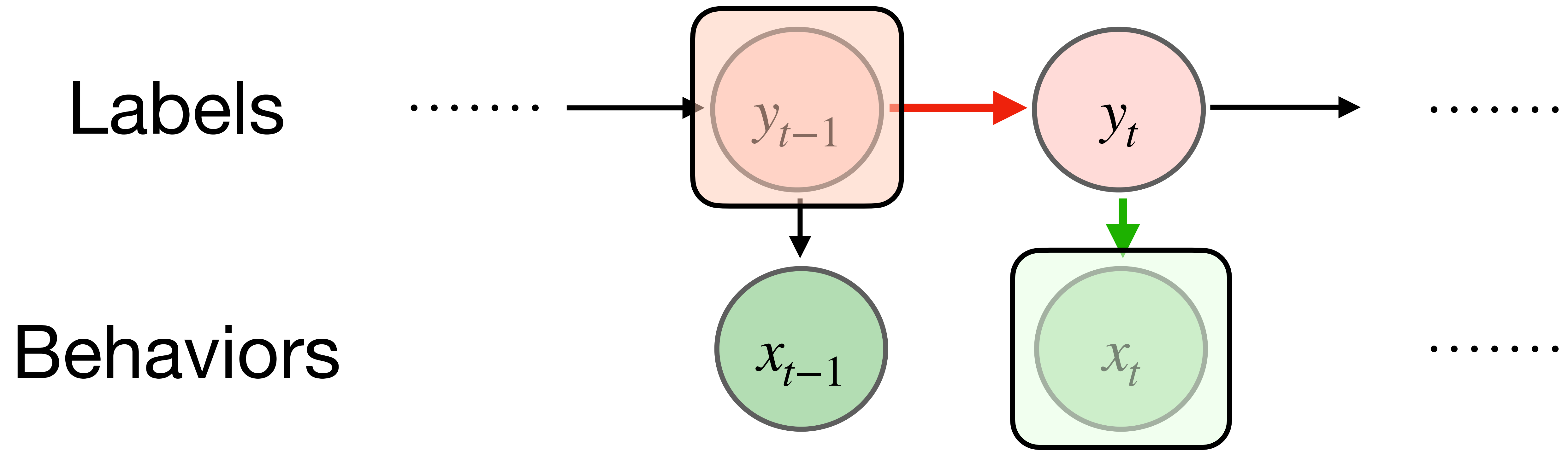


$$P(x_t = b \mid y_{t-1} = N) = \sum_{y_t} P(x_t = b, y_t \mid y_{t-1} = N)$$

$$= \sum_{y_t} \underbrace{P(x_t = b \mid y_t)}_{\text{emission}} \underbrace{P(y_t \mid y_{t-1} = N)}_{\text{transition}}$$

[SP23 Midterm] Problem 5: Lie Detection

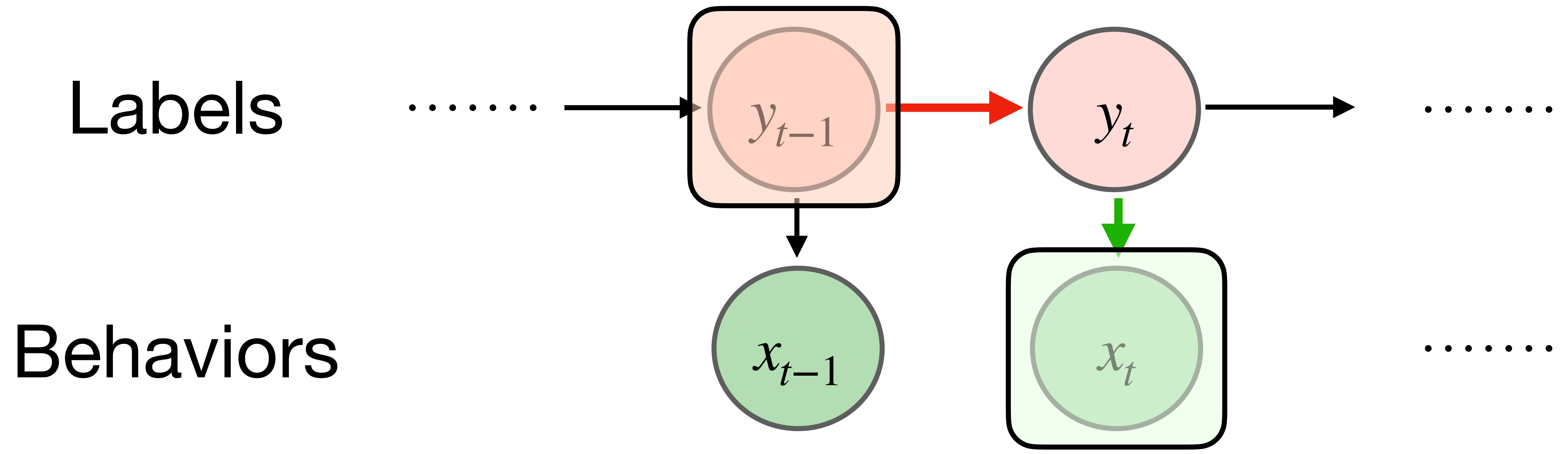
Q1: You use the above data to build an HMM. What is $P(x_t = b \mid y_{t-1} = N)$?



$$
\begin{aligned}
P(x_t = b \mid y_{t-1} = N) &= \sum_{y_t} P(x_t = b, y_t \mid y_{t-1} = N) \\
&= \sum_{y_t} P(x_t = b \mid y_t) P(y_t \mid y_{t-1} = N) \\
&= P(b \mid L) P(L \mid N) + P(b \mid H) P(H \mid N)
\end{aligned}
$$

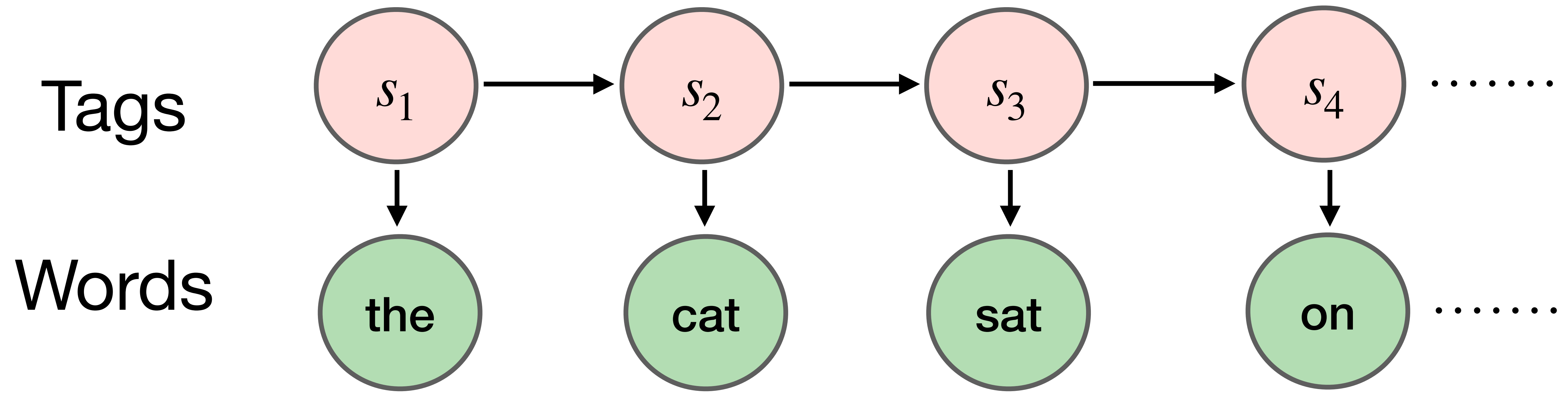
[SP23 Midterm] Problem 5: Lie Detection

Q1: You use the above data to build an HMM. What is $P(x_t = b \mid y_{t-1} = N)$?



$$
\begin{aligned}
P(x_t = b \mid y_{t-1} = N) &= \sum_{y_t} P(x_t = b, y_t \mid y_{t-1} = N) \\
&= \sum_{y_t} P(x_t = b \mid y_t) P(y_t \mid y_{t-1} = N) \\
&= P(b \mid L)P(L \mid N) + P(b \mid H)P(H \mid N) \\
&= \frac{1}{2} \times \frac{9}{10} + 1 \times \frac{1}{10} = \frac{11}{20}
\end{aligned}
$$

HMMs: Efficient Inference

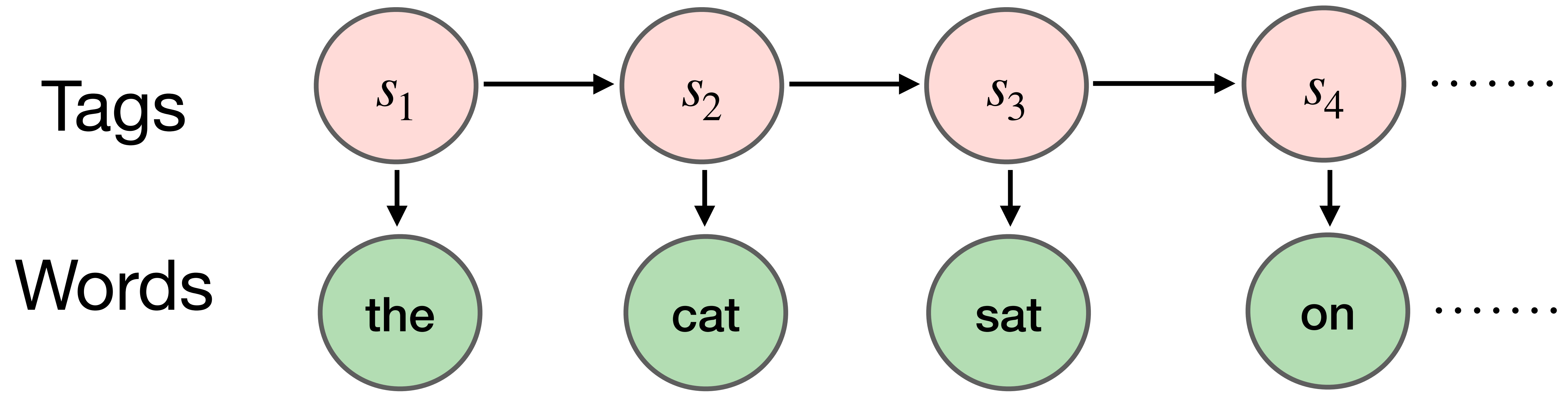


Task: Find the most probable sequence of states $S = s_1, s_2, \ldots, s_n$ given the observations $O = o_1, o_2, \ldots, o_n$

$$\hat{S} = \arg\max_{S} P(S \mid O) = \arg\max_{s_1, s_2, \ldots s_n} \prod_{i=1}^{n} P(o_i \mid s_i) P(s_i \mid s_{i-1})$$

How can we maximize this?
Search over all state sequences?

HMMs: Efficient Inference



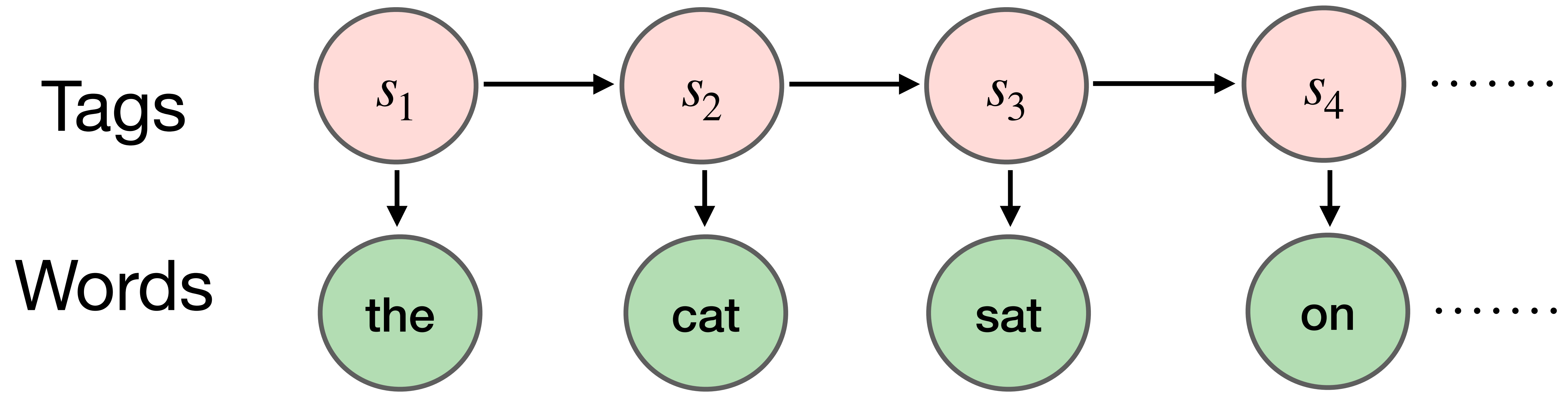
Task: Find the most probable sequence of states $S = s_1, s_2, \ldots, s_n$ given the observations $O = o_1, o_2, \ldots, o_n$

$$\hat{S} = \arg\max_{S} P(S \mid O) = \arg \max_{s_1, s_2, \ldots s_n} \prod_{i=1}^{n} P(o_i \mid s_i) P(s_i \mid s_{i-1})$$

Do what we just did for the example, but for every possible state!

- Viterbi is simply realizing that we only need to do this one-step calculation
- Try all possible explanations s_i for o_i to find the most likely one
- Multiply by (highest) $P(s_{i-1})$, i.e. score, so “most likely” is over all observations

HMMs: Efficient Inference

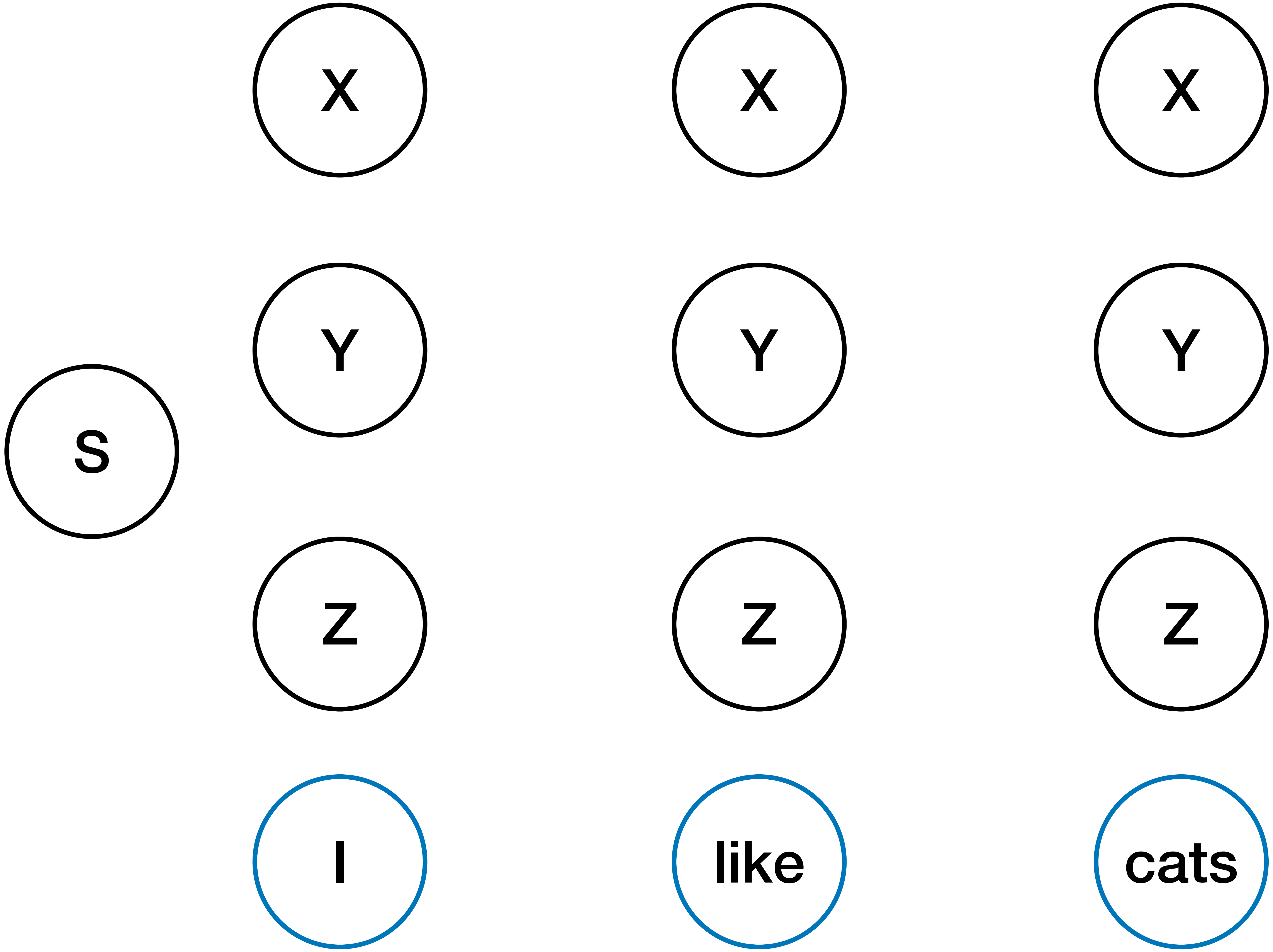


Task: Find the most probable sequence of states $S = s_1, s_2, \ldots, s_n$ given the observations $O = o_1, o_2, \ldots, o_n$

$$\hat{S} = \arg\max_{S} P(S \mid O) = \arg\max_{s_1, s_2, \ldots s_n} \prod_{i=1}^{n} P(o_i \mid s_i) P(s_i \mid s_{i-1})$$

Thanks to Markov, we only need to "look-back" one time step to decode!
Makes dynamic programming possible, so we only need to keep the "best" for each previous time step to decode the next (instead of more history).

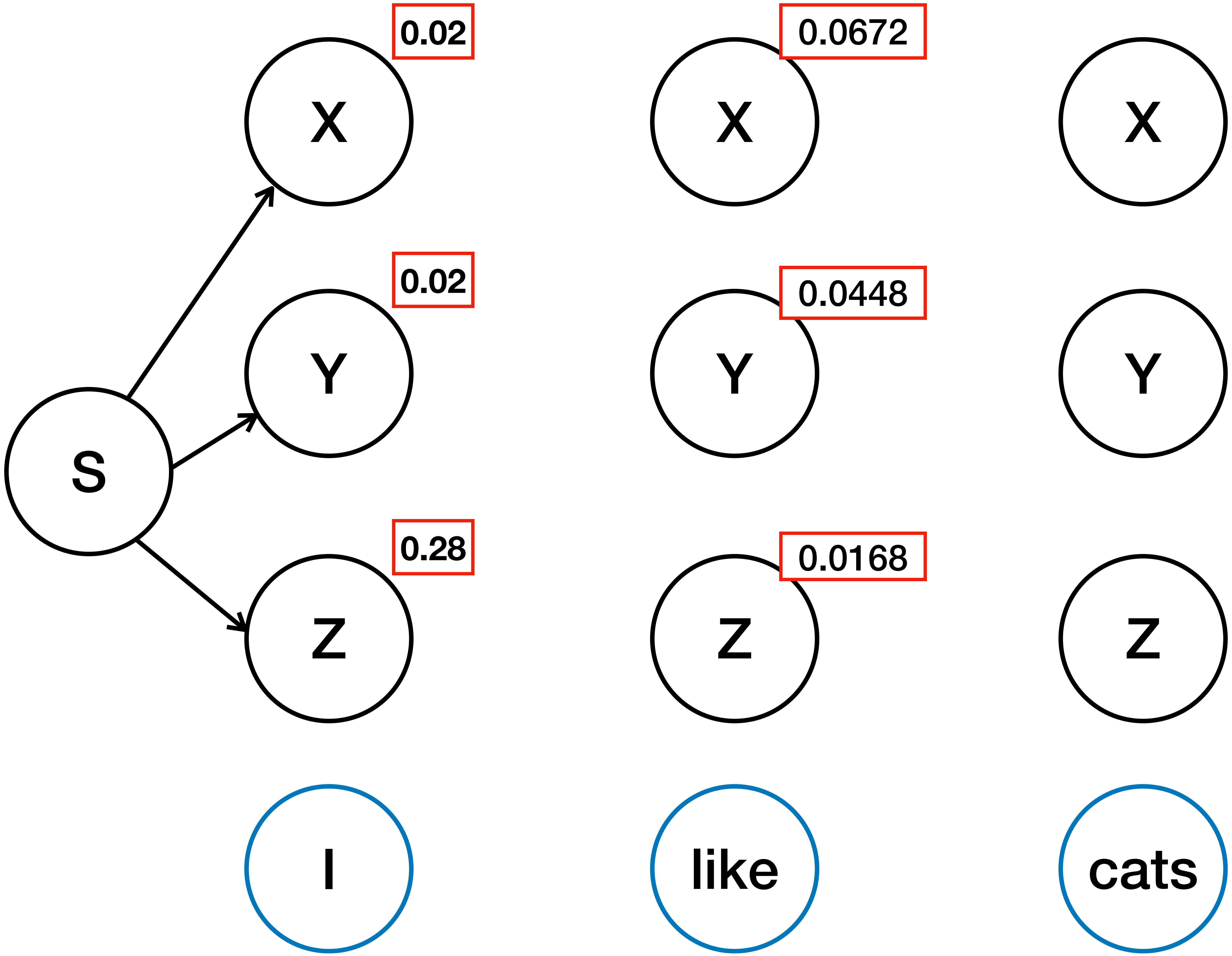
Viterbi Intuition



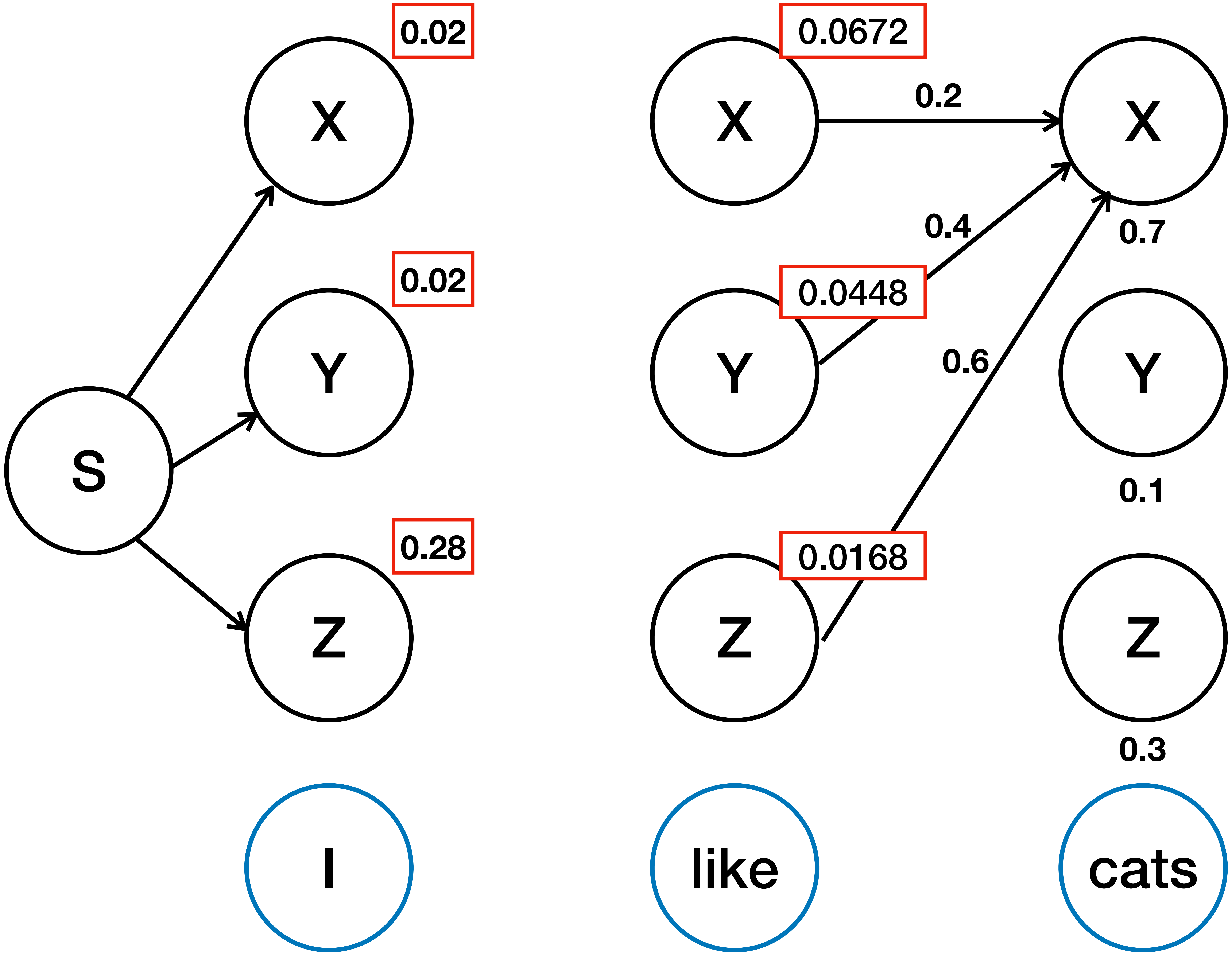
	X	Y	Z
S	0.1	0.2	0.7
X	0.2	0.5	0.3
Y	0.4	0.4	0.2
Z	0.6	0.2	0.2

	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 Intuition

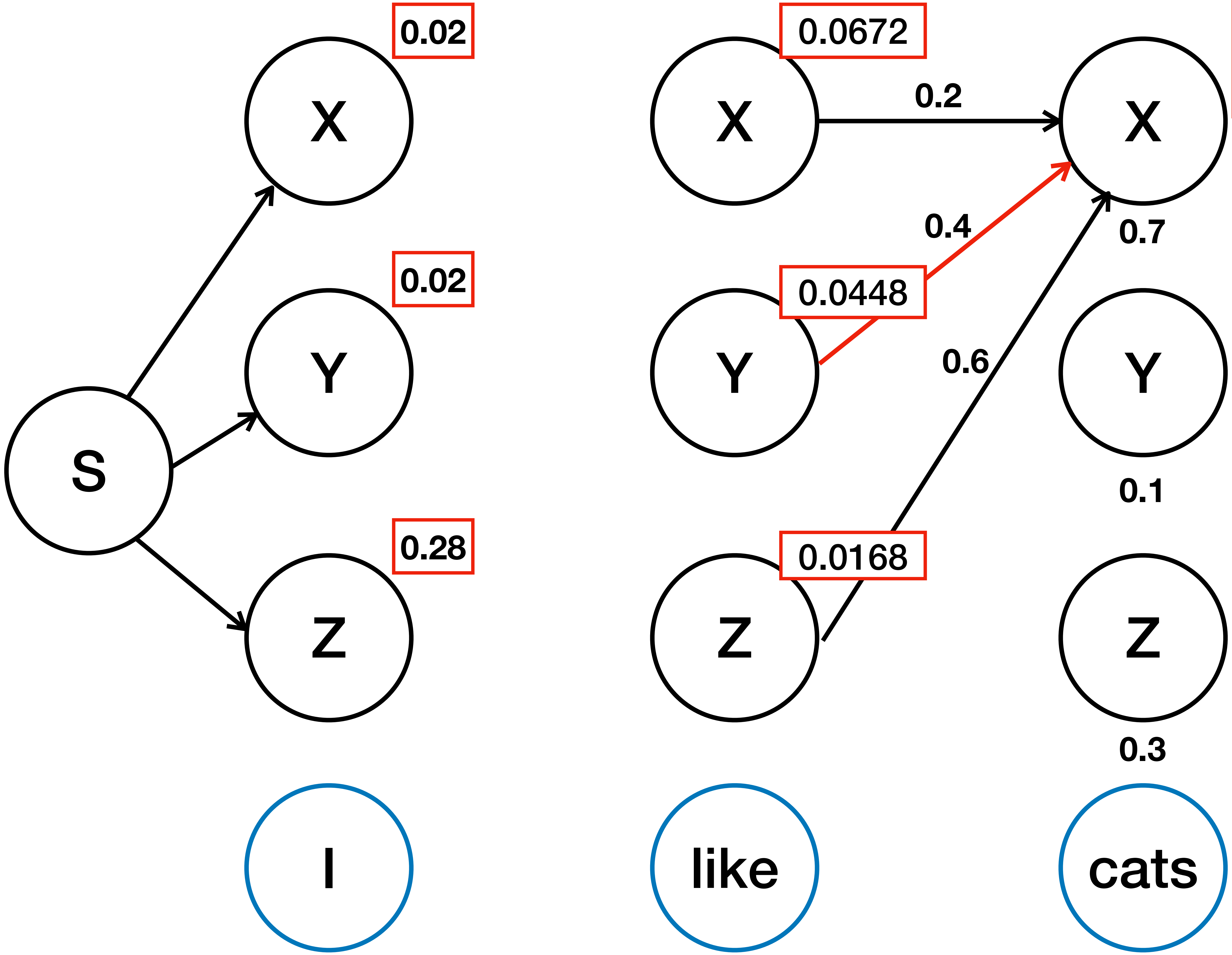


Viterbi Intuition



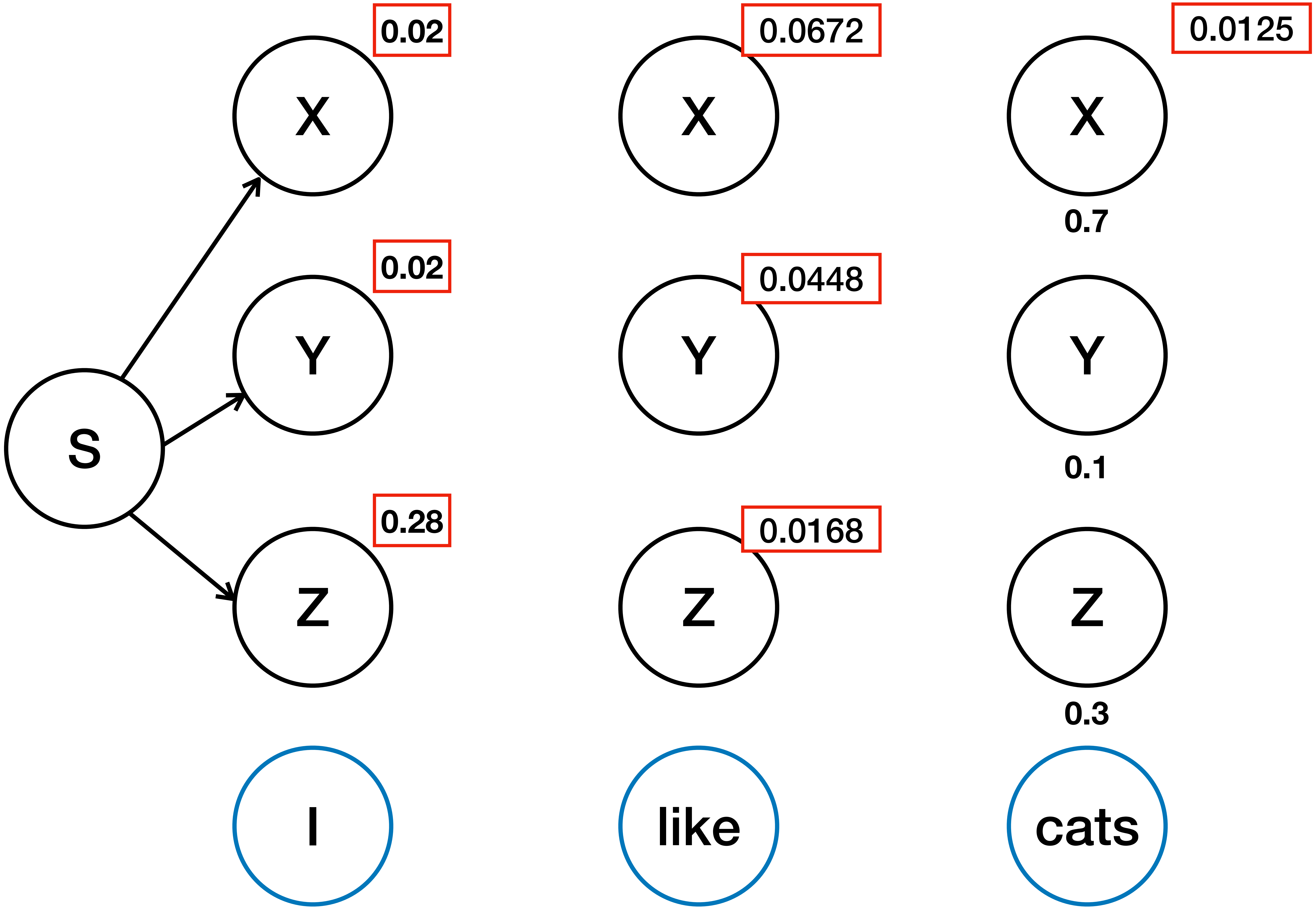
0.0672x0.2x0.7 = 0.0023
0.0448x0.4x0.7 = 0.0125
0.0168x0.6x0.7 = 0.0070

Viterbi Intuition

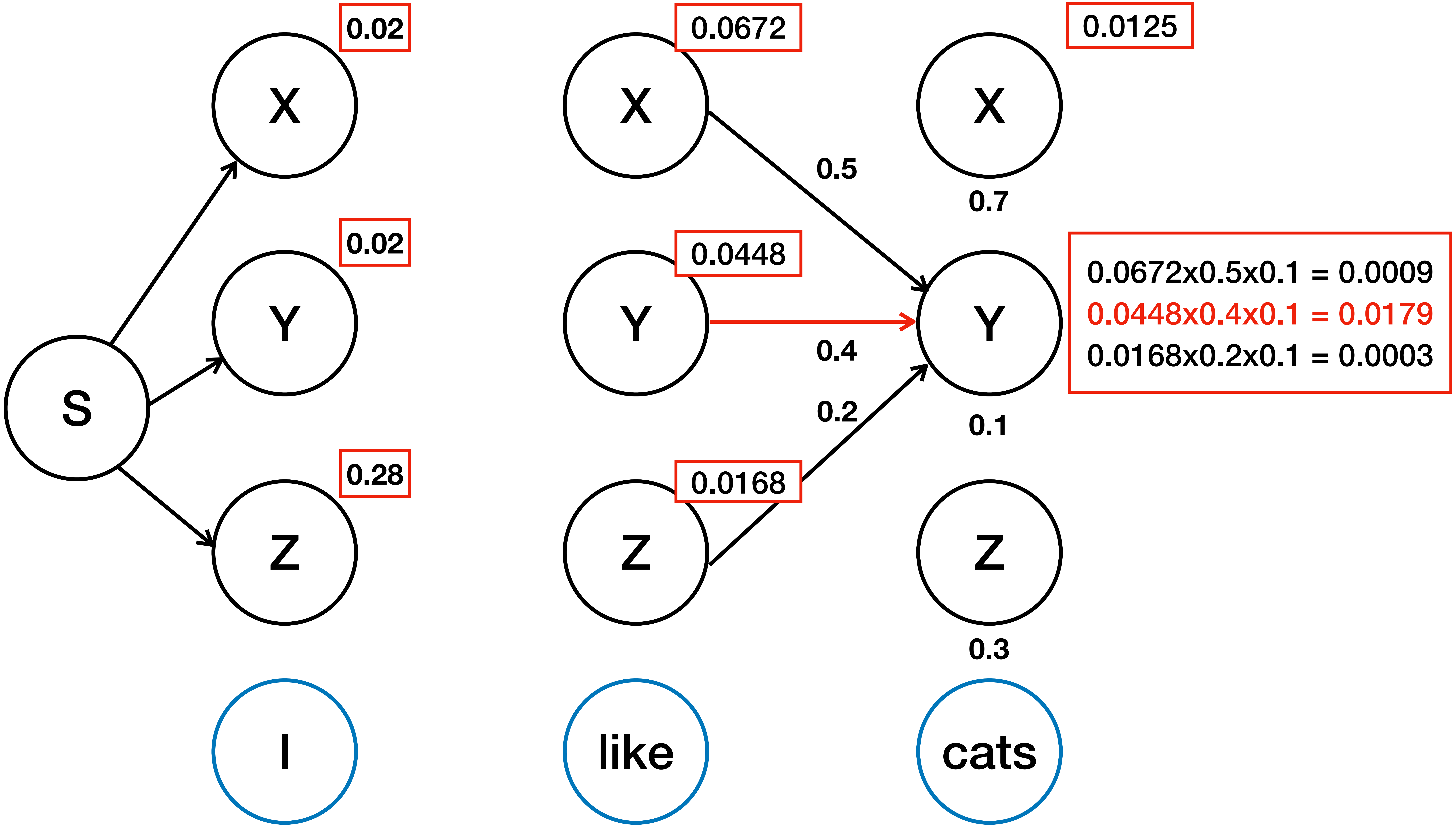


0.0672x0.2x0.7 = 0.0023
0.0448x0.4x0.7 = 0.0125
0.0168x0.6x0.7 = 0.0070

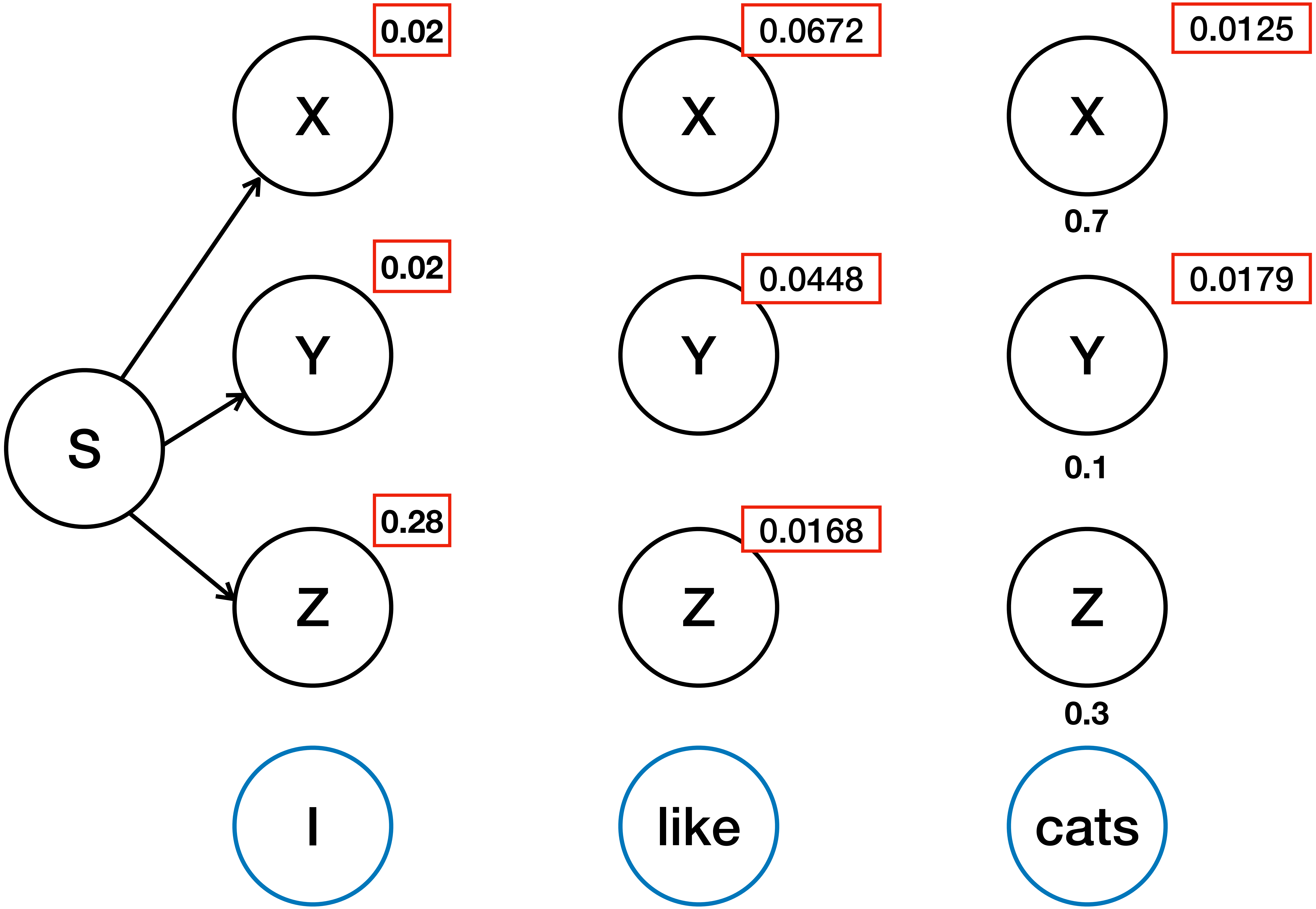
Viterbi Intuition



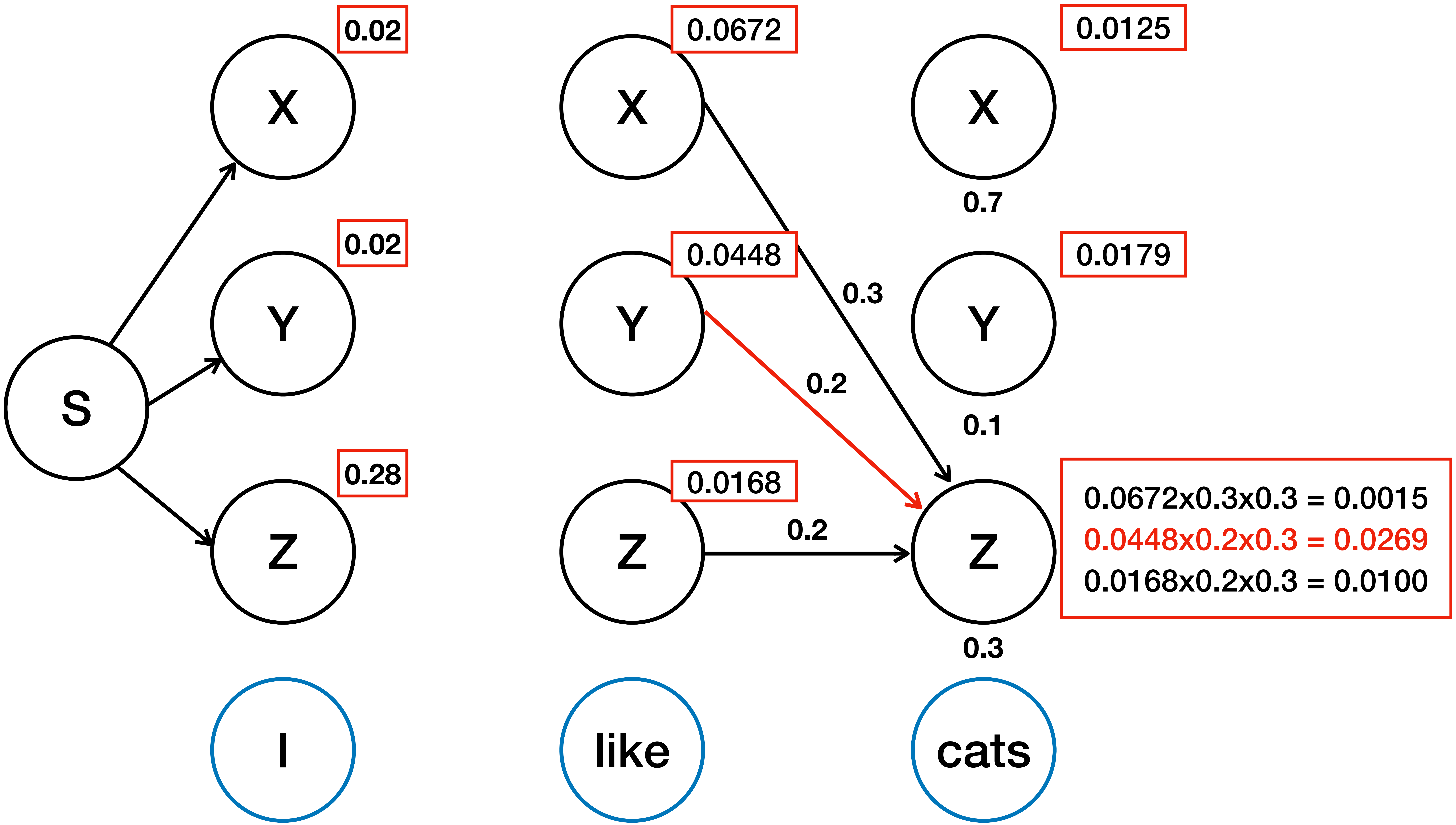
Viterbi Intuition



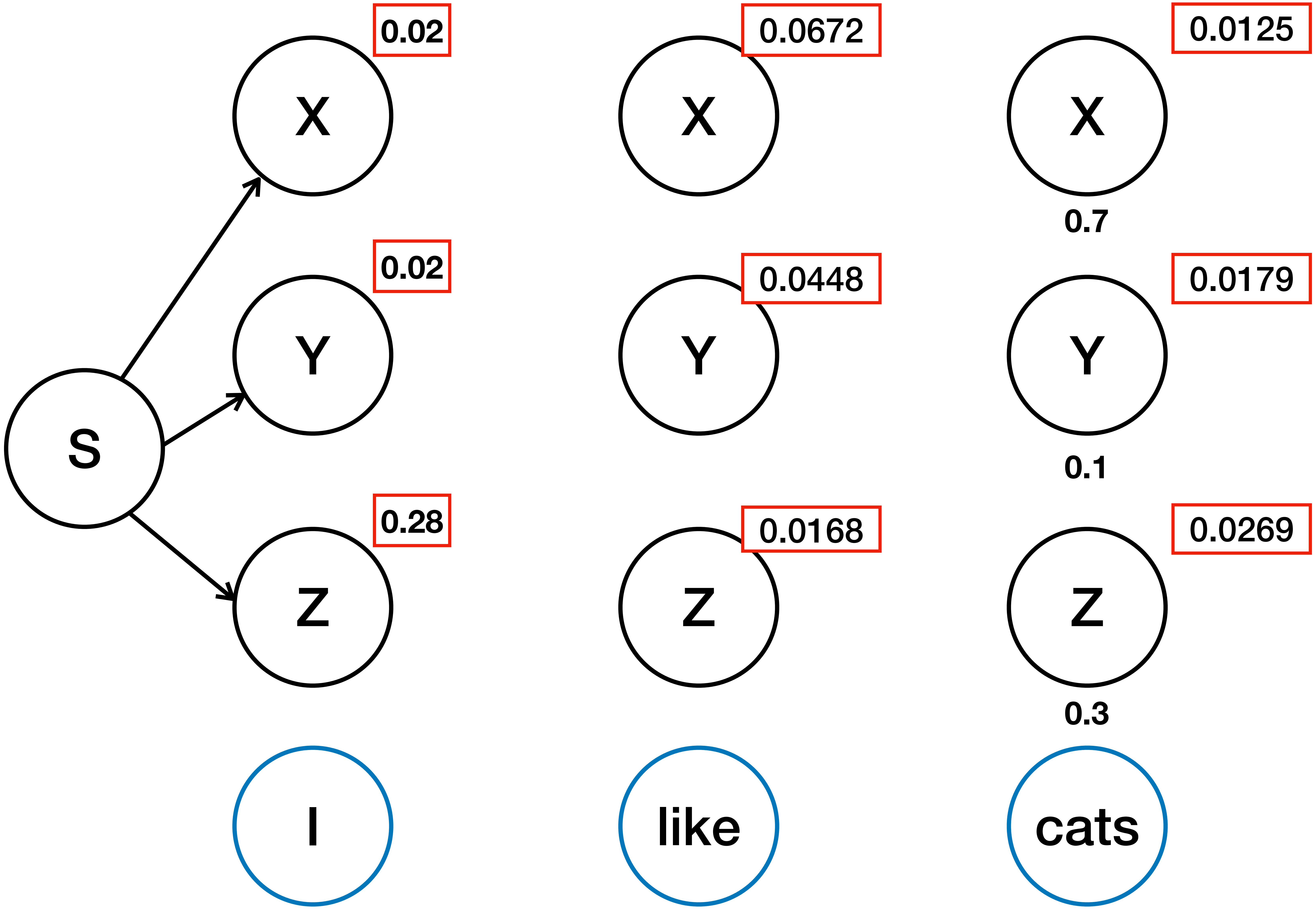
Viterbi Intuition



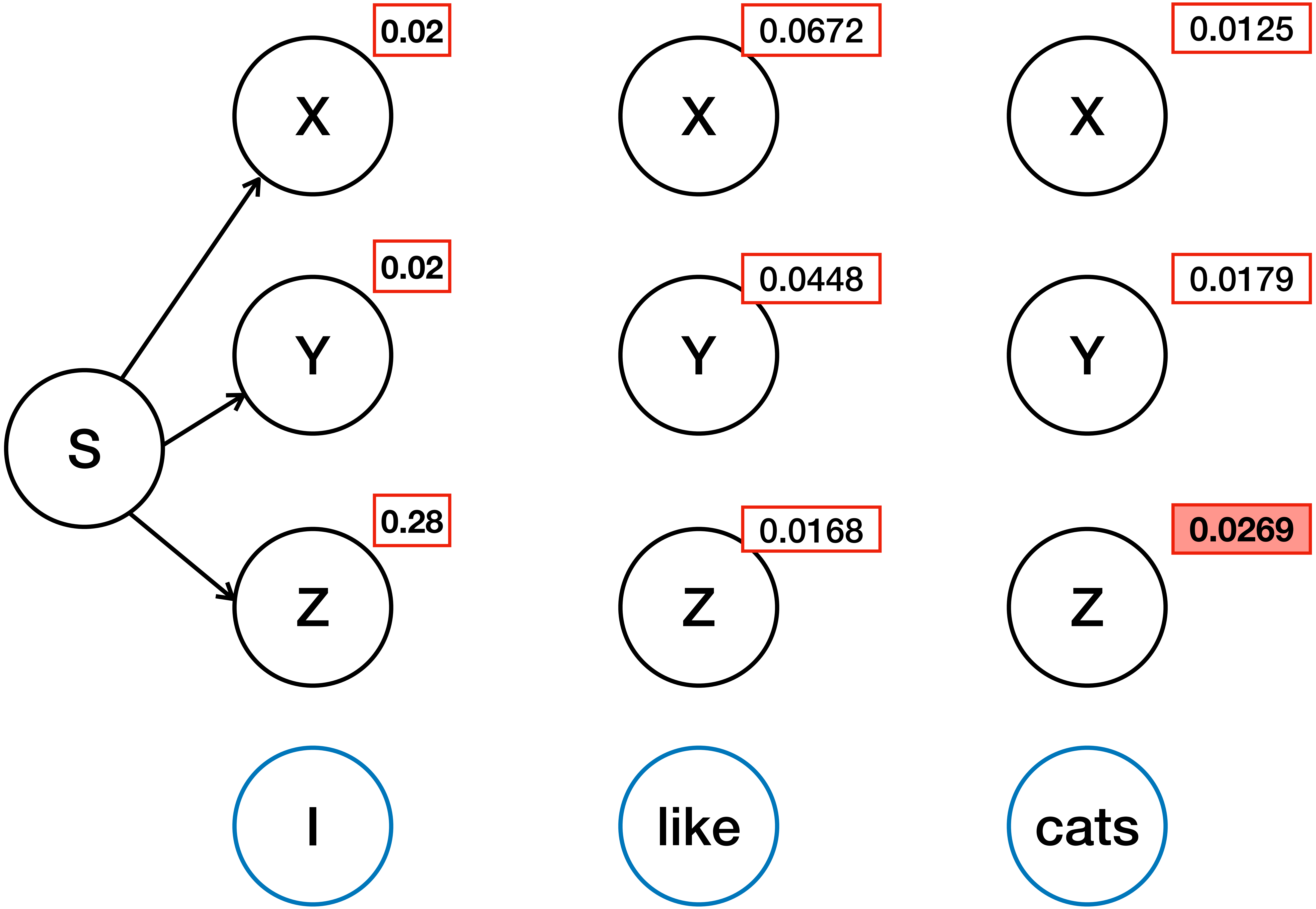
Viterbi Intuition



Viterbi Intuition



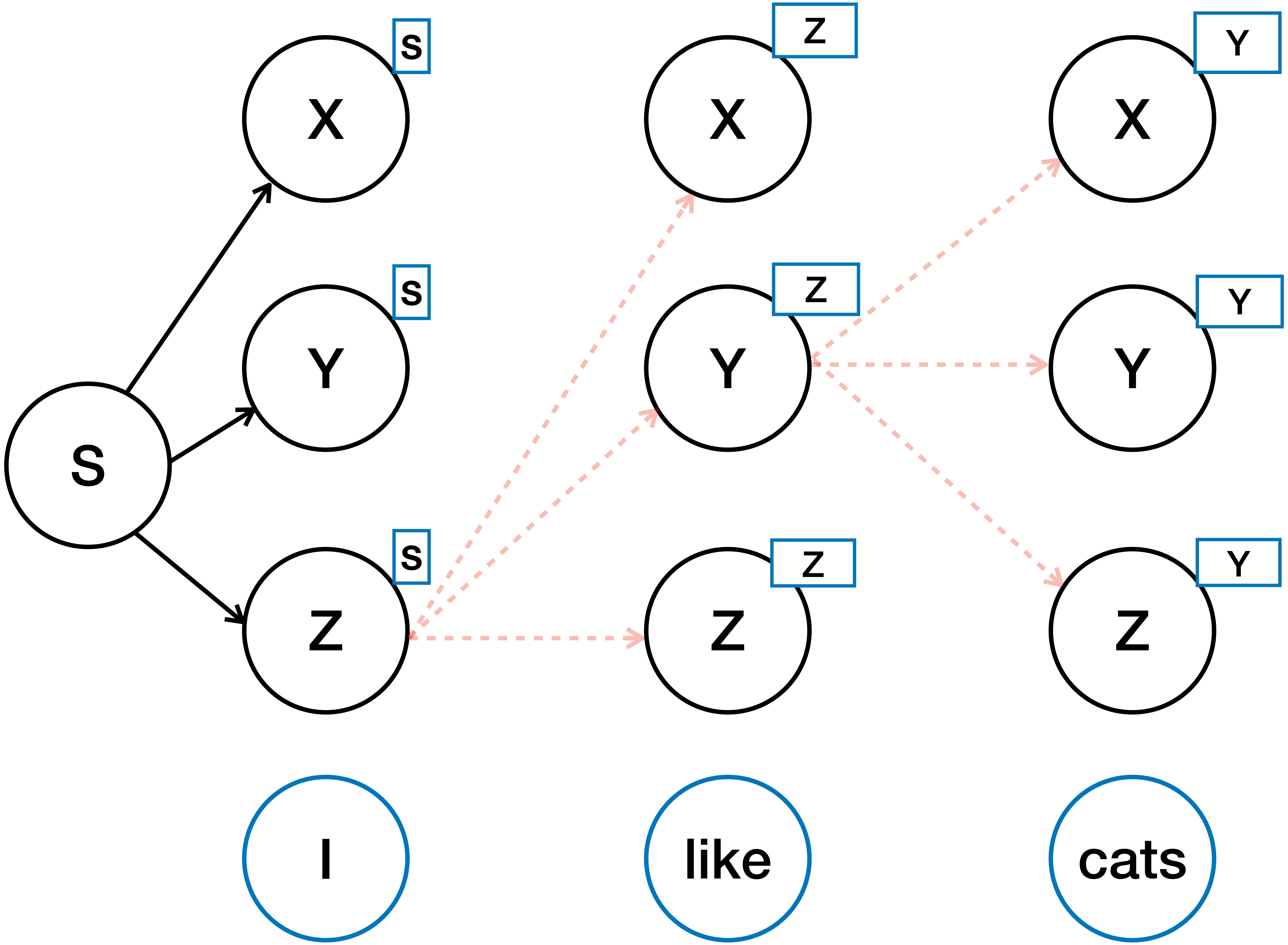
Viterbi Intuition



0.0269 is the maximum likelihood estimate our HMM predicts for “I like cats”

But what states gives rise to this sentence?

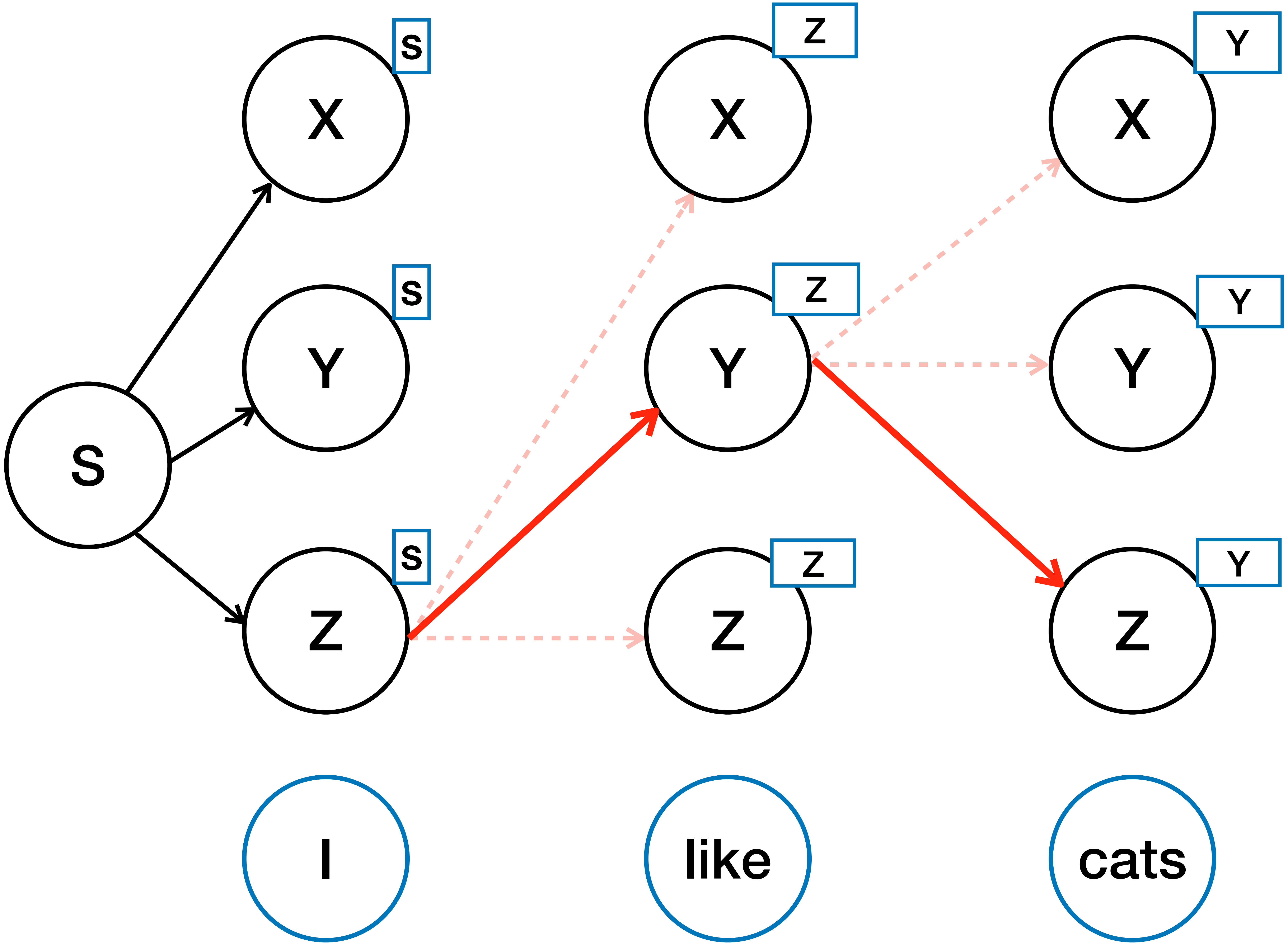
Viterbi Intuition: Backtracking



This is where the backtracking matrix comes in handy!

- Keep track of best node from previous step

Viterbi Intuition: Backtracking



Final tags are <Z, Y, Z>

This is where the backtracking matrix comes in handy!

- Keep track of best node from previous step

Viterbi Understanding Check

How does Viterbi on a trigram HMM change? What about a 4-gram HMM?

Viterbi Understanding Check

How does Viterbi on a trigram HMM change? What about a 4-gram HMM?

Key: Without Markov, we just need to look **further back** to calculate our likelihood!

- HMM extended to trigram, 4-gram etc: $P(S, O) = \prod_{i=1}^{n} P(s_i \mid s_{i-1}, s_{i-2}) P(o_i \mid s_i)$
- MLE estimate: $P(s_i \mid s_{i-1}, s_{i-2}) = \dfrac{\text{Count}(s_i, s_{i-1}, s_{i-2})}{\text{Count}(s_{i-1}, s_{i-2})}$
- Viterbi:

$$M[i, j, k] = \max_{r} M[i-1, k, r]\ P(s_j \mid s_k, s_r)\ P(o_i \mid s_j) \quad 1 \leq j, k, r \leq K \ \ 1 \leq i \leq n$$

  - most probable sequence of states ending with state *j* at time *i*, and state *k* at *i-1*
  - Time complexity $= O(nK^3)$

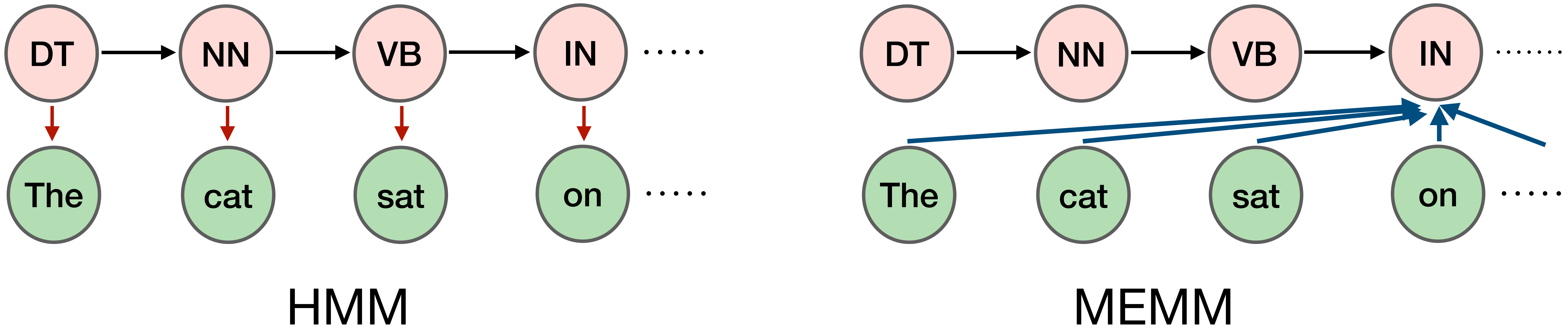
Maximum Entropy Markov Models

Generative vs. Discriminative

- HMM is a *generative* model : we compute probability $P(S, O)$
- Can we model $P(s_1, ..., s_n \mid o_1, ..., o_n)$ directly?

	Generative	**Discriminative**
Text classification	Naive Bayes: $P(c)\, P(d \mid c)$	Logistic Regression: $P(c \mid d)$
Sequence prediction	HMM: $P(s_1, \ldots s_n)\, P(o_1, \ldots o_n \mid s_1, \ldots s_n)$	MEMM: $P(s_1, \ldots s_n \mid o_1, \ldots o_n)$

MEMM Basics



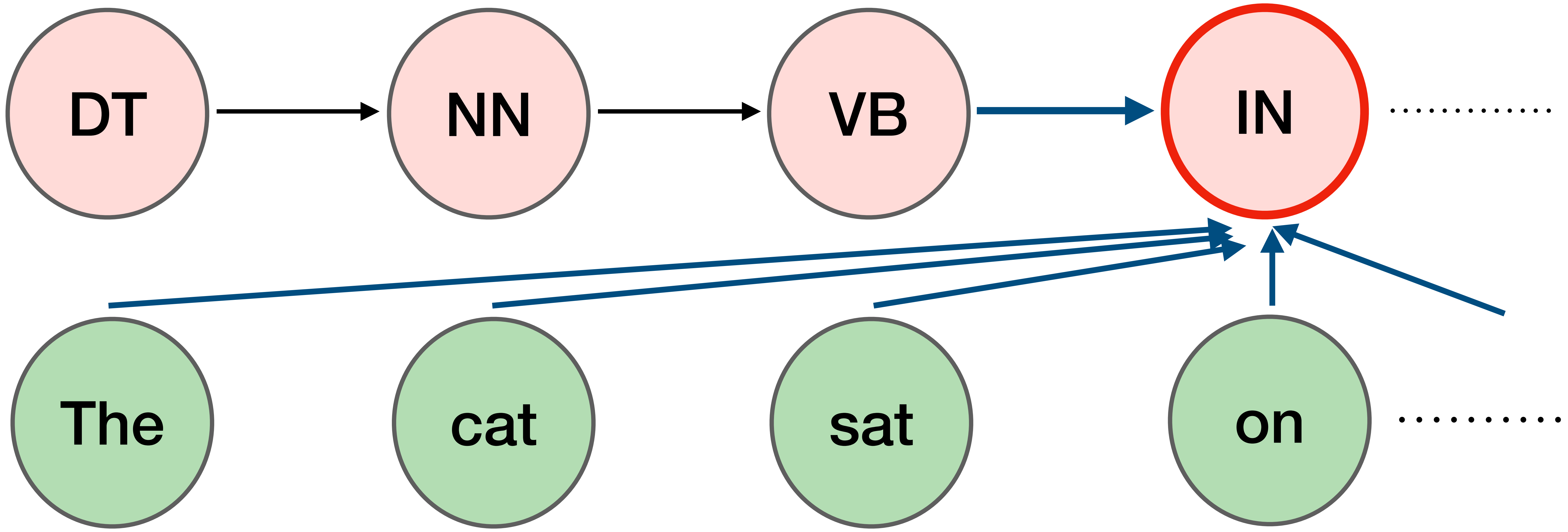
$$P(S \mid O) = \prod_{i=1}^{n} P(s_i \mid s_{i-1}, s_{i-2}, \ldots, s_1, O)$$

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

Markov assumption:
Bigram MEMM

Instead of learning how to model observations given states,
directly learn to predict states given observations.

MEMM Basics



- To predict the red node, the bigram MEMM conditions on the “prior tag” (VB) and the observations in the window (The, cat, sat, on)

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

(Bigram) MEMM Formulation

- To make the equivalence b/w MEMMs & logistic regression clearer, we will depart slightly from the lecture notation
- Our primary objective: $P(S \mid O) = \prod_{i=1}^{n} P(s_i \mid s_{i-1}, s_{i-2}, \ldots, s_1, O) = \prod_{i=1}^{n} P(s_i \mid s_{i-1}, O)$

<u>Logistic Regression</u>

Input: documents d

Features: $\vec{\mathbf{x}} = \mathbf{f}(d)$ (#words, patterns, …)

Output: $z_i = \mathbf{w}^{(i)} \cdot \vec{\mathbf{x}} \quad (\mathbf{w}^{(i)} \in \mathbb{R}^d)$

$P(y = i \mid d) = \text{softmax}(\vec{\mathbf{z}})_i$

$\propto \exp(\mathbf{w}^{(i)} \cdot \mathbf{f}(d))$

weights features

<u>MEMM</u>

Input: state s_{i-1}, observations O, position i

Features: $\mathbf{f}(s_{i-1}, O, i)$ (1 if ⟨man, the⟩ else 0)

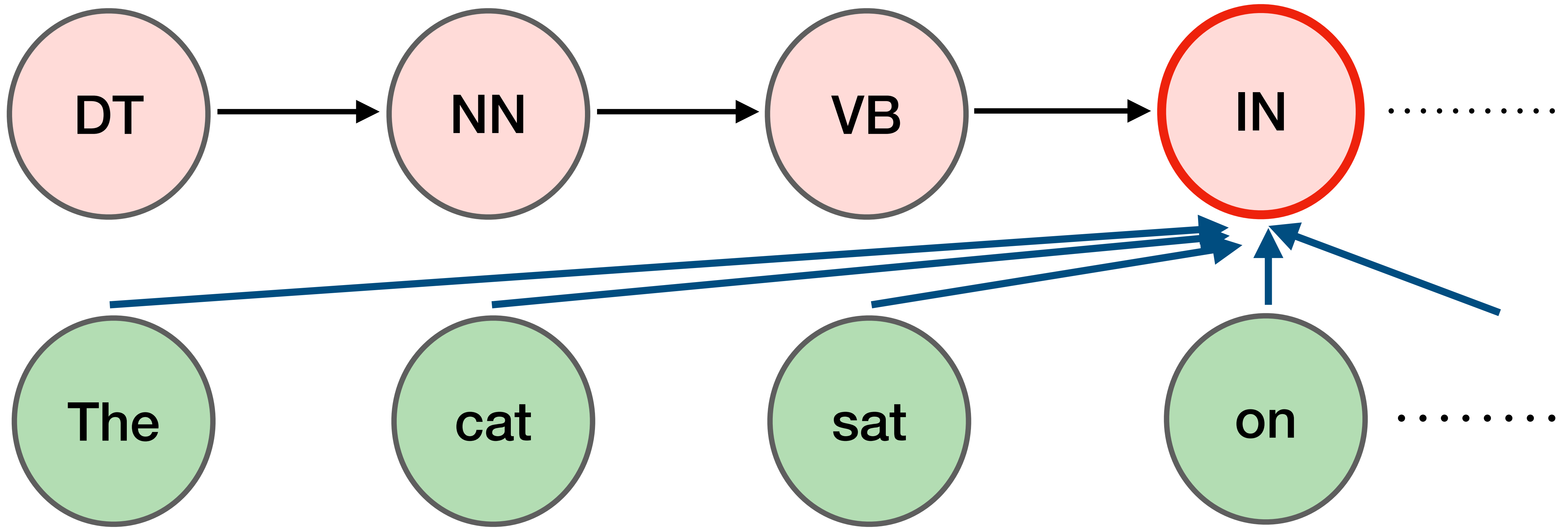
Output: $z_s = \mathbf{w}^{(s)} \cdot \mathbf{f}(s_{i-1}, O, i) \quad (\mathbf{w}^{(s)} \in \mathbb{R}^d)$

$P(s_i = s \mid s_{i-1}, O) = \text{softmax}(\vec{\mathbf{z}})_s$

$\propto \exp(\mathbf{w}^{(s)} \cdot \mathbf{f}(s_{i-1}, O, i))$

weights features

MEMM Formulation

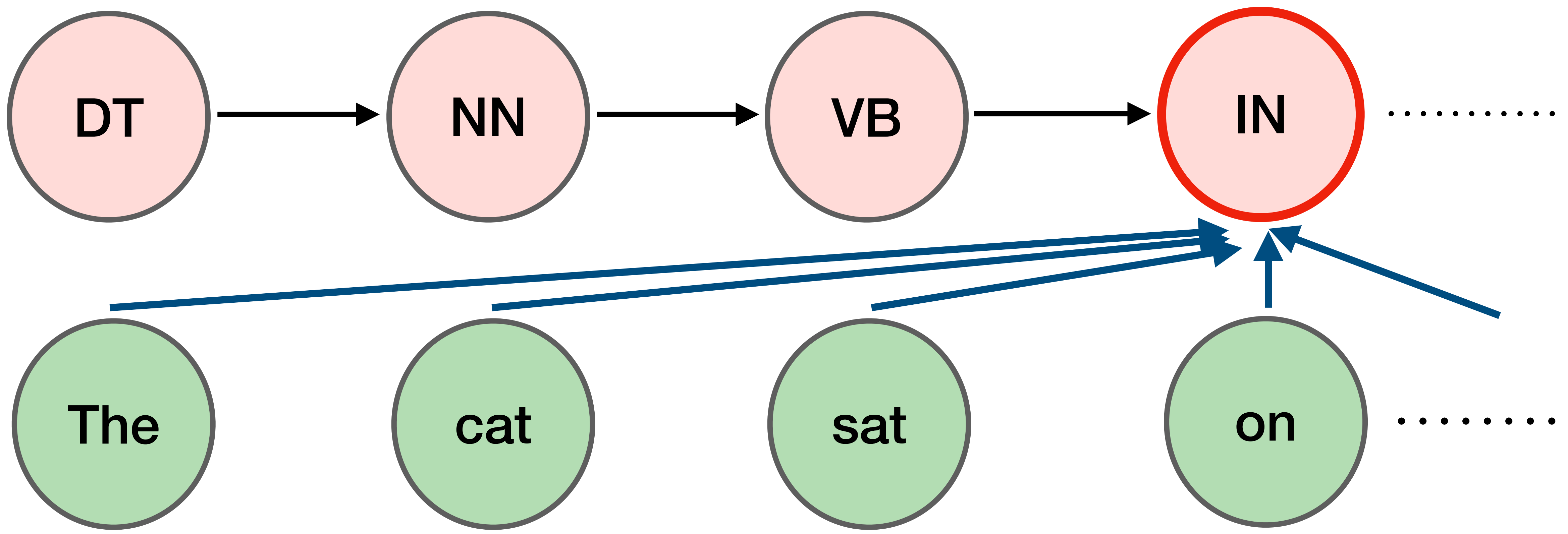


Examples of *binary* features and potential weights:

- $(o_{i-2} = \text{animal}, s_{i-1} = \text{VB})$: $w_{\text{IN}} = 3, w_{\text{VB}} = -1, w_{\text{DT}} = 0, w_{\text{NN}} = 1$
- (tri-gram) $(s_{i-2} = \text{NN}, s_{i-1} = \text{VB})$: $w_{\text{IN}} = 4, \ldots$

$\mathbf{f}$ would look like $[1,0,0,1,\ldots]$, and $\mathbf{w}_{\text{IN}} = [3,0,1,4,\ldots]$

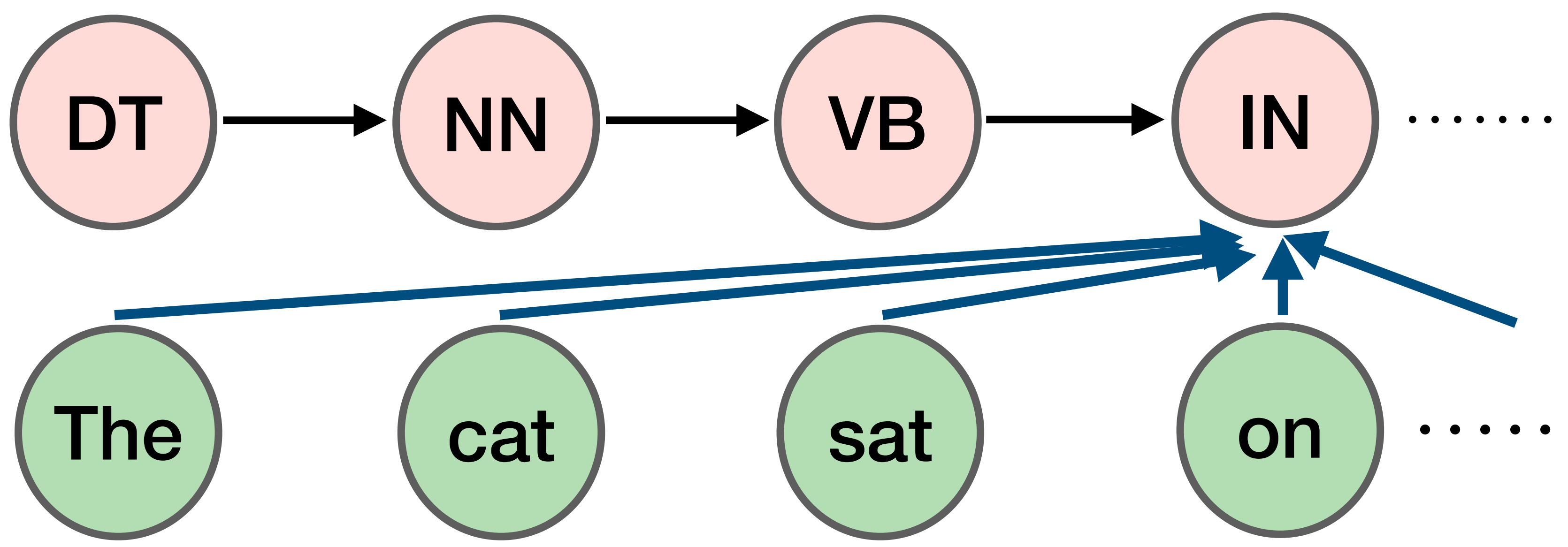
MEMM Formulation



Can also define generic feature *templates*:

- $\langle s_i, o_{i-2} \rangle$, $\langle s_i, o_{i-1} \rangle$, $\langle s_i, o_i \rangle$, $\langle s_i, o_{i+1} \rangle$, $\langle s_i, o_{i+2} \rangle$
- $\langle s_i, s_{i-1} \rangle$, $\langle s_i, s_{i-1}, s_{i-2} \rangle$

MEMM Basics



MEMM

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

Markov assumption:
Bigram MEMM

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

weights features

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

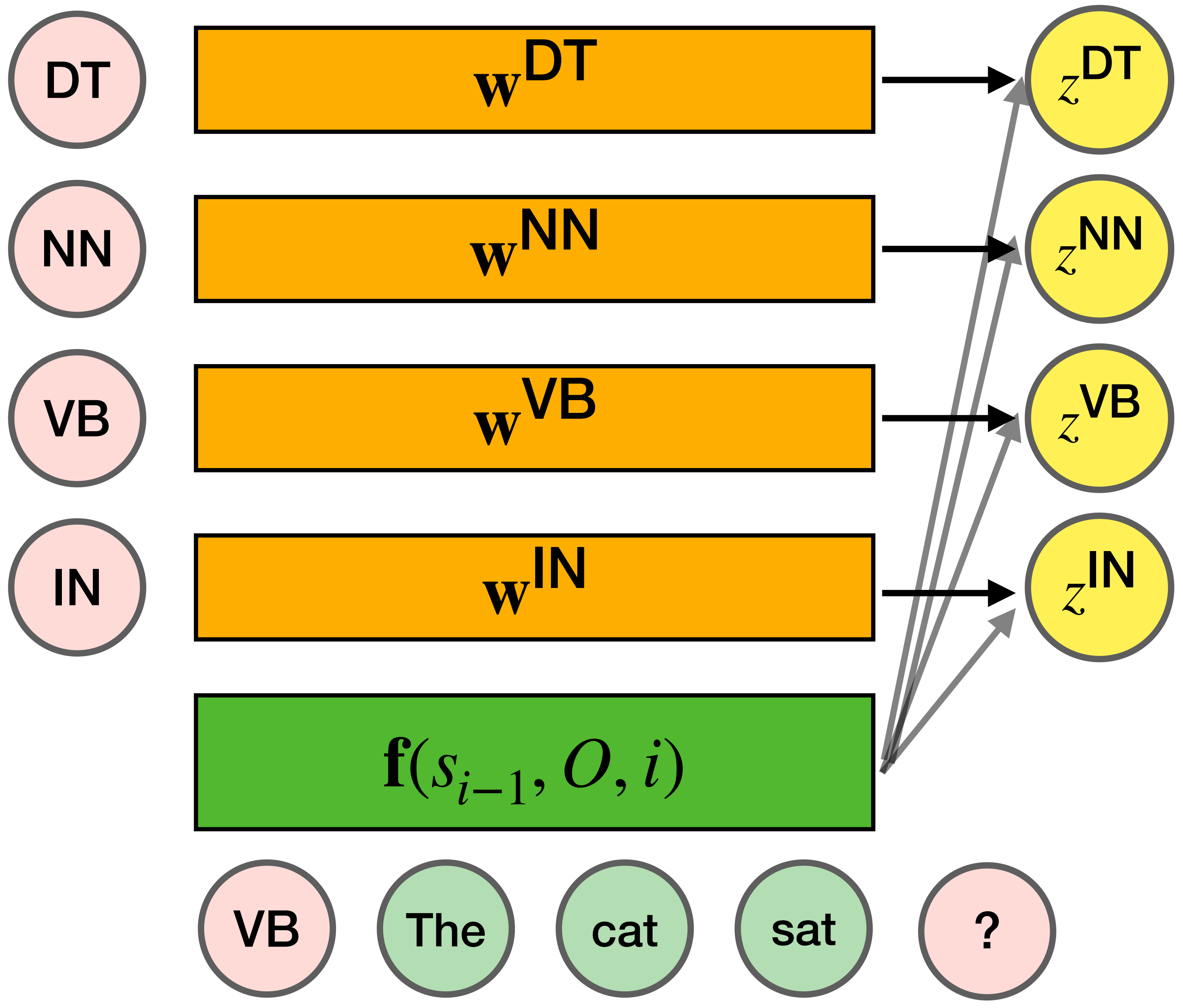
$$P(s_i = s \,|\, s_{i-1}, O) = \frac{\exp(\mathbf{w}^{(s)} \cdot \mathbf{f}(s_{i-1}, O, i))}{\sum_{s'=1}^{K} \exp(\mathbf{w}^{(s')} \cdot \mathbf{f}(s_{i-1}, O, i))}$$

Quick Aside: Two Equivalent Formulations

This formulation is identical to one in lecture; it's just different in featurization.

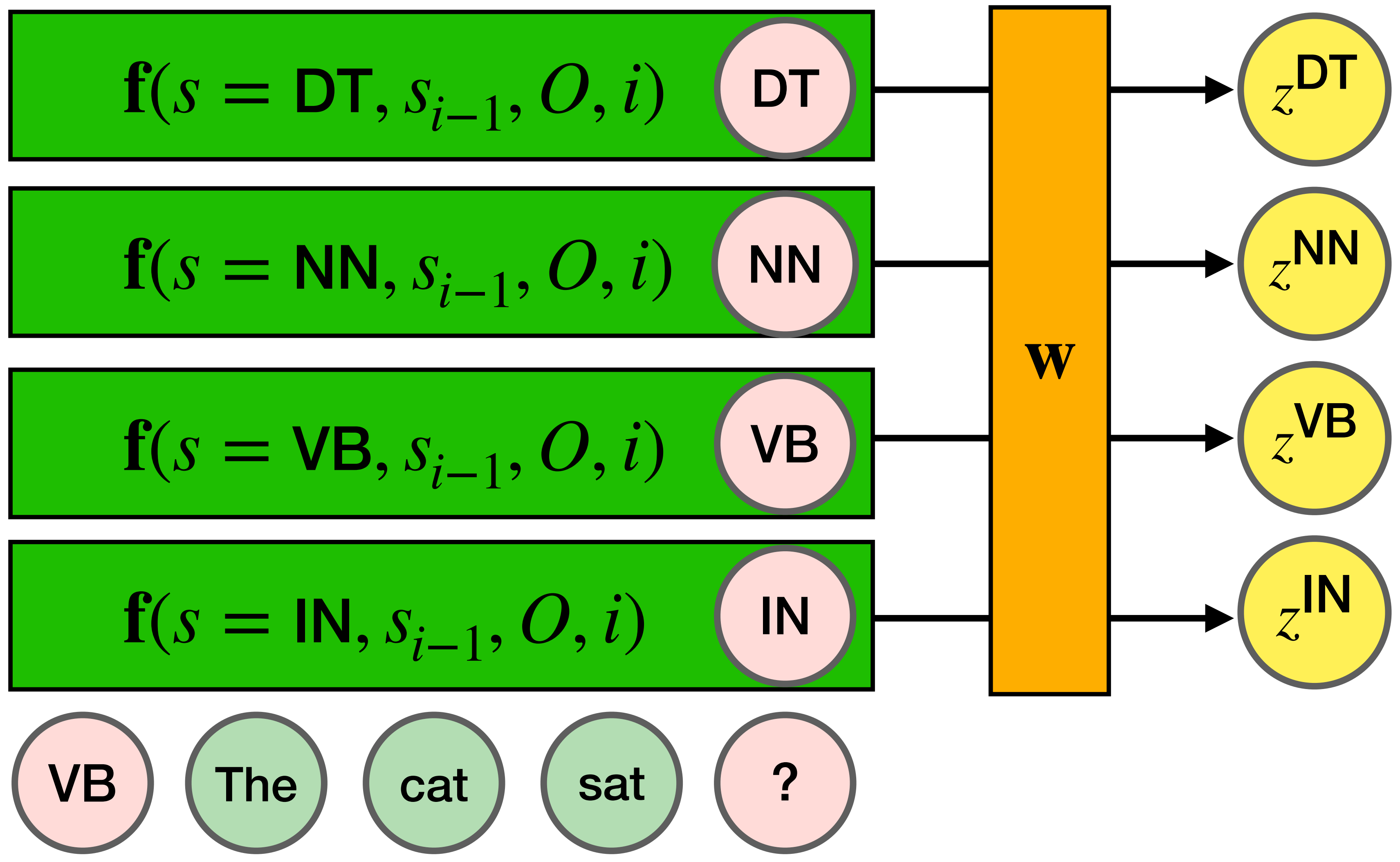
$$P(s_i = s \mid s_{i-1}, O) \propto \exp(\mathbf{w}^{(s)} \cdot \mathbf{f}(s_{i-1}, O, i))$$

Parameters: $\mathbf{w}^{(s)} \in \mathbb{R}^d$ for $s = 1,2,\ldots,K$



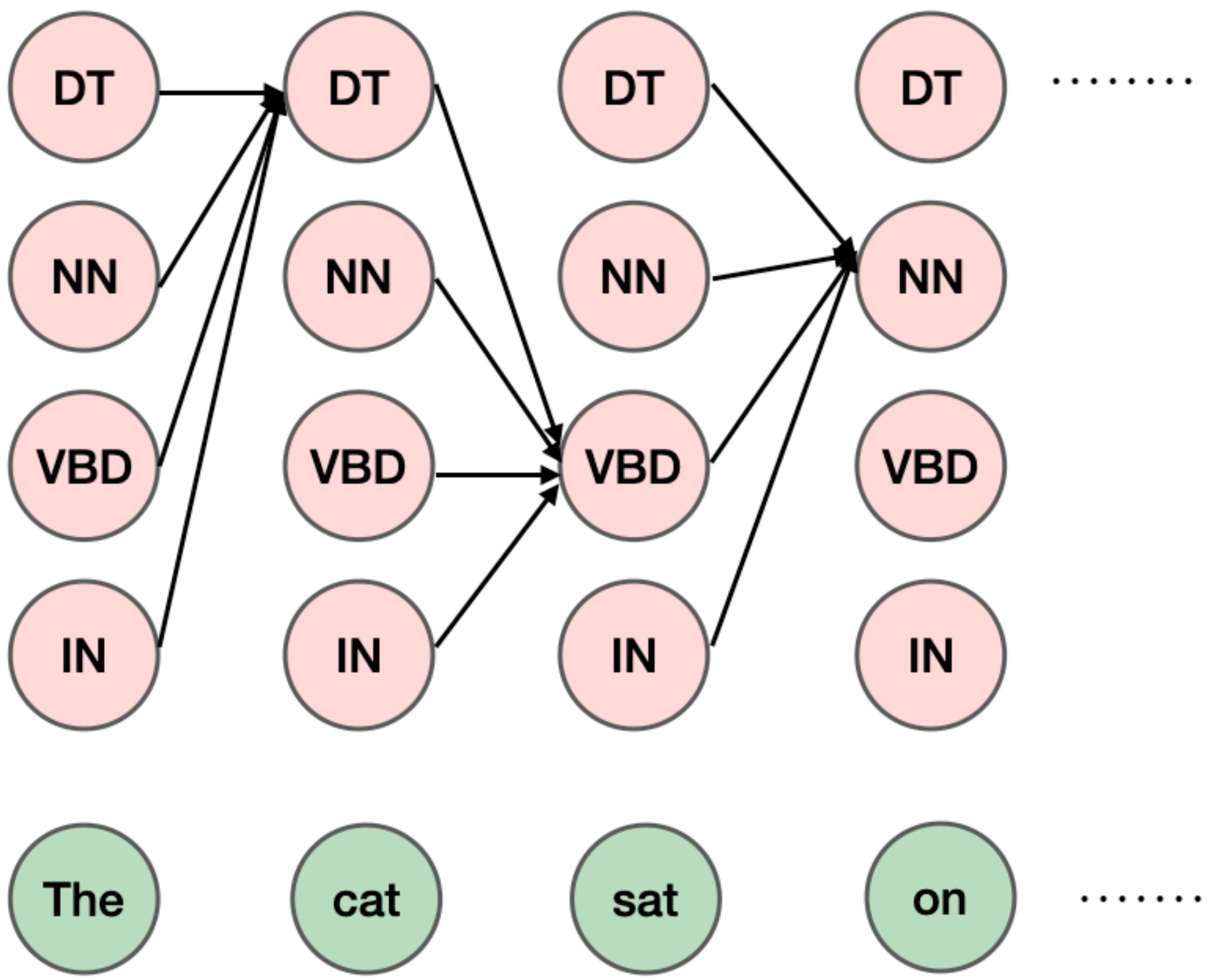
$$P(s_i = s \mid s_{i-1}, O) \propto \exp(\mathbf{w} \cdot \mathbf{f}(s = s_i, s_{i-1}, O, i))$$

Parameters: $\mathbf{w} \in \mathbb{R}^d$, same for all features



$$f_{100}(h, y) = \begin{cases} 1 & \text{if } x_i \text{ is base and } y = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

Viterbi Decoding for MEMMs



$M[i, j]$ stores joint probability of most probable sequence of states ending with state j at time i

$$M[i,j] = \max_k M[i-1,k] \boxed{P(s_i = j \mid s_{i-1} = k, O)} \quad 1 \leq k \leq K \quad 1 \leq i \leq n$$

Backward: Pick $\max_k M[n,k]$ and backtrack using B

MEMMs vs. HMMs

- HMM models the joint $P(S, O)$ while MEMM models the required prediction $P(S \mid O)$
- MEMM has more expressivity
  - accounts for dependencies between neighboring states and **entire observation sequence**
  - allows for **more flexible features**
- HMM may hold an advantage if the dataset is small

[SP23 Midterm] Problem 5: Lie Detection

You are building a lie detector taking as input a stream of recorded behaviors:

- $x_t \in \{a, b\}$, i.e., face touching (a) or blinking (b)

Detector at every moment then predicts one of four labels:

- $y_t \in \{N, U, L, H\}$, i.e., Neutral (N), Unclear (U), Lying (L), and Honest (H)

Dataset is triplets (y_{t-1}, y_t, x_t): 9x of (N, L, a), 9x of (U, L, b), 1x of (N, H, b)

- Labels y_t come from body language experts' annotation

Q1: You use the above data to build an HMM. What is $P(x_t = b \,|\, y_{t-1} = N)$?

[SP23 Midterm] Problem 5: Lie Detection

(3) (4 points) Now assume you design the system with an MEMM model that predicts the label y_t:

$$P(y_t = \hat{y}_t \mid y_{t-1} = N, x_t = b) = \frac{\exp\Big(\mathbf{w} \cdot \mathbf{f}(y_t = \hat{y}_t, y_{t-1} = N, x_t = b)\Big)}{\sum_{y_t'} \exp\Big(\mathbf{w} \cdot \mathbf{f}(y_t = y_t', y_{t-1} = N, x_t = b)\Big)},$$

where $\mathbf{f}(\cdot)$ is defined by the feature templates: $\langle y_{t-1}, y_t \rangle$, $\langle y_t, x_t \rangle$, and $\langle y_{t-1}, y_t, x_t \rangle$. Assume that, after the model is trained, the learned weight vector $\mathbf{w}$ is:

$$w_i = \begin{cases} 0.2, & \text{for } \mathbb{1}[y_{t-1} = N, y_t = L] \\ 0.1, & \text{for } \mathbb{1}[y_{t-1} = N, y_t = H] \\ 0.6, & \text{for } \mathbb{1}[y_t = L, x_t = b] \\ 0.5, & \text{for } \mathbb{1}[y_t = H, x_t = b] \\ 0.7, & \text{for } \mathbb{1}[y_{t-1} = U, y_t = L, x_t = b] \\ 0.3, & \text{for } \mathbb{1}[y_{t-1} = N, y_t = H, x_t = b] \\ 0, & \text{otherwise} \end{cases}$$

where w_i is the element in $\mathbf{w}$. Which tag will this model predict at time step t if we know $y_{t-1} = N$ and $x_t = b$? With the model predicting $\hat{y}_t$, does this triplet $(y_{t-1} = N, y_t = \hat{y}_t, x_t = b)$ have a higher count than all other triplets $(y_{t-1} = N, y_t = y_t', x_t = b)$ in the past annotated recordings?

Labels are (N, H, L, U)

[SP23 Midterm] Problem 5: Lie Detection

(a) Just add up each of the z_i for each label; unnormalized is good enough

- $\tilde{P}(y_t = H \mid y_{t-1} = N, x_t = b) =$ 0.1 + 0.5 + 0.3 = 0.9
- $\tilde{P}(y_t = L \mid y_{t-1} = N, x_t = b) =$ 0.2 + 0.6 = 0.8

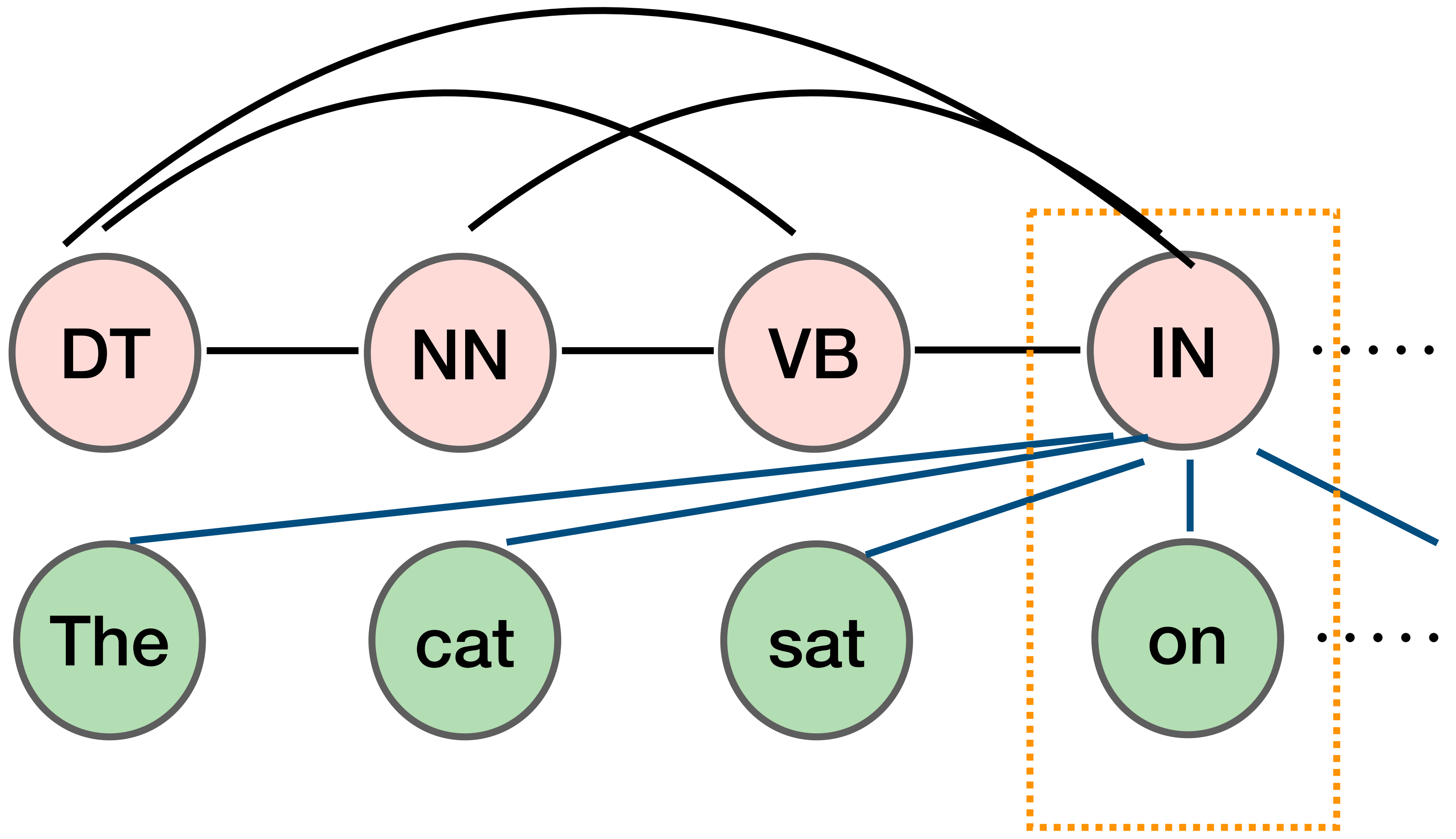
Therefore $\hat{y}_t = H$ is most likely.

(b) Yes. The 0.3 on $(y_{t-1} = N, y_t = H, x_t = b)$ shows that.

$$w_i = \begin{cases} 0.2, & \text{for } \mathbb{1}[y_{t-1} = N, y_t = L] \\ 0.1, & \text{for } \mathbb{1}[y_{t-1} = N, y_t = H] \\ 0.6, & \text{for } \mathbb{1}[y_t = L, x_t = b] \\ 0.5, & \text{for } \mathbb{1}[y_t = H, x_t = b] \\ 0.7, & \text{for } \mathbb{1}[y_{t-1} = U, y_t = L, x_t = b] \\ 0.3, & \text{for } \mathbb{1}[y_{t-1} = N, y_t = H, x_t = b] \\ 0, & \text{otherwise} \end{cases}$$

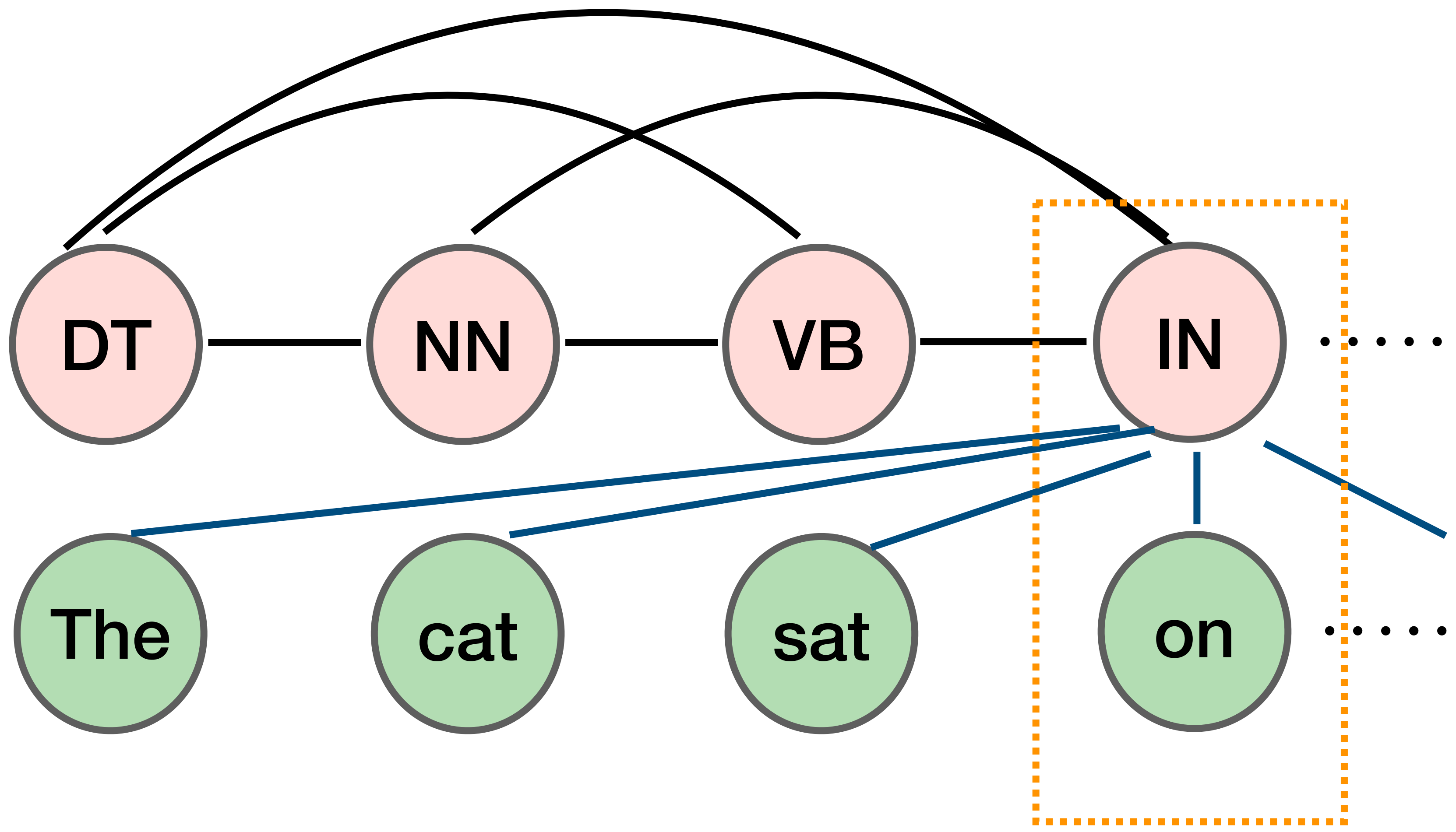
Conditional Random Field

- Model $P(s_1, \ldots, s_n \mid o_1, \ldots, o_n)$ directly
- No Markov assumption
  - Map entire sequence of states S and observations O to a **global** feature vector
    - Normalize over entire sequences
- Generalization of MEMMs



$$P(S \mid O) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(S, O))}{\sum_{S'} \exp(\mathbf{w} \cdot \mathbf{f}(S', O))} = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(S, O))}{Z(O)}$$

Features



$$P(S \mid O) = \frac{\exp(\sum_{k=1}^{m} w_k \cdot F_k(S, O))}{\sum_{S'} \exp(\sum_{k=1}^{m} w_k \cdot F_k(S', O))}$$

- Each F_k in $\mathbf{f}$ is a **global** feature function
- Can be computed as a combination of local

  features: $$F_k = \sum_{i=1}^{n} f_k(s_{i-1}, s_i, O, i)$$

- Each local feature only depends on previous and current states

$$\mathbb{1}\{x_i = \textit{the},\ y_i = \text{DET}\}$$
$$\mathbb{1}\{y_i = \text{PROPN},\ x_{i+1} = \textit{Street},\ y_{i-1} = \text{NUM}\}$$
$$\mathbb{1}\{y_i = \text{VERB},\ y_{i-1} = \text{AUX}\}$$

CRFs vs. MEMMs

- MEMM models the required prediction $P(S \mid O)$ using the Markov assumption, while the CRF does not
- CRF uses global features while MEMM features are localized
- Feature design is flexible in both models
- CRF is computationally more complex