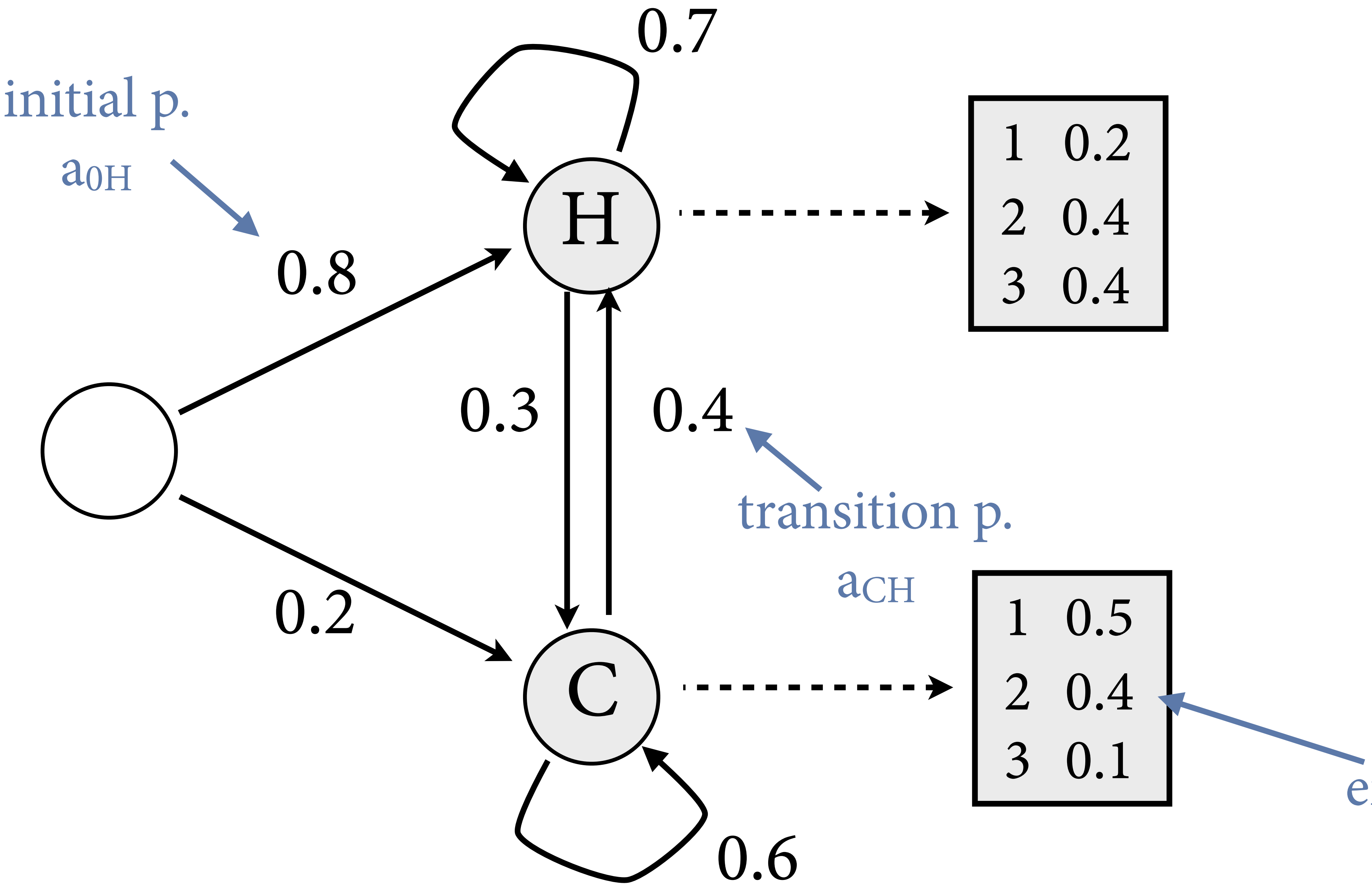
Evaluating and Training HMMs

Computational Linguistics

Alexander Koller

10 November 2017

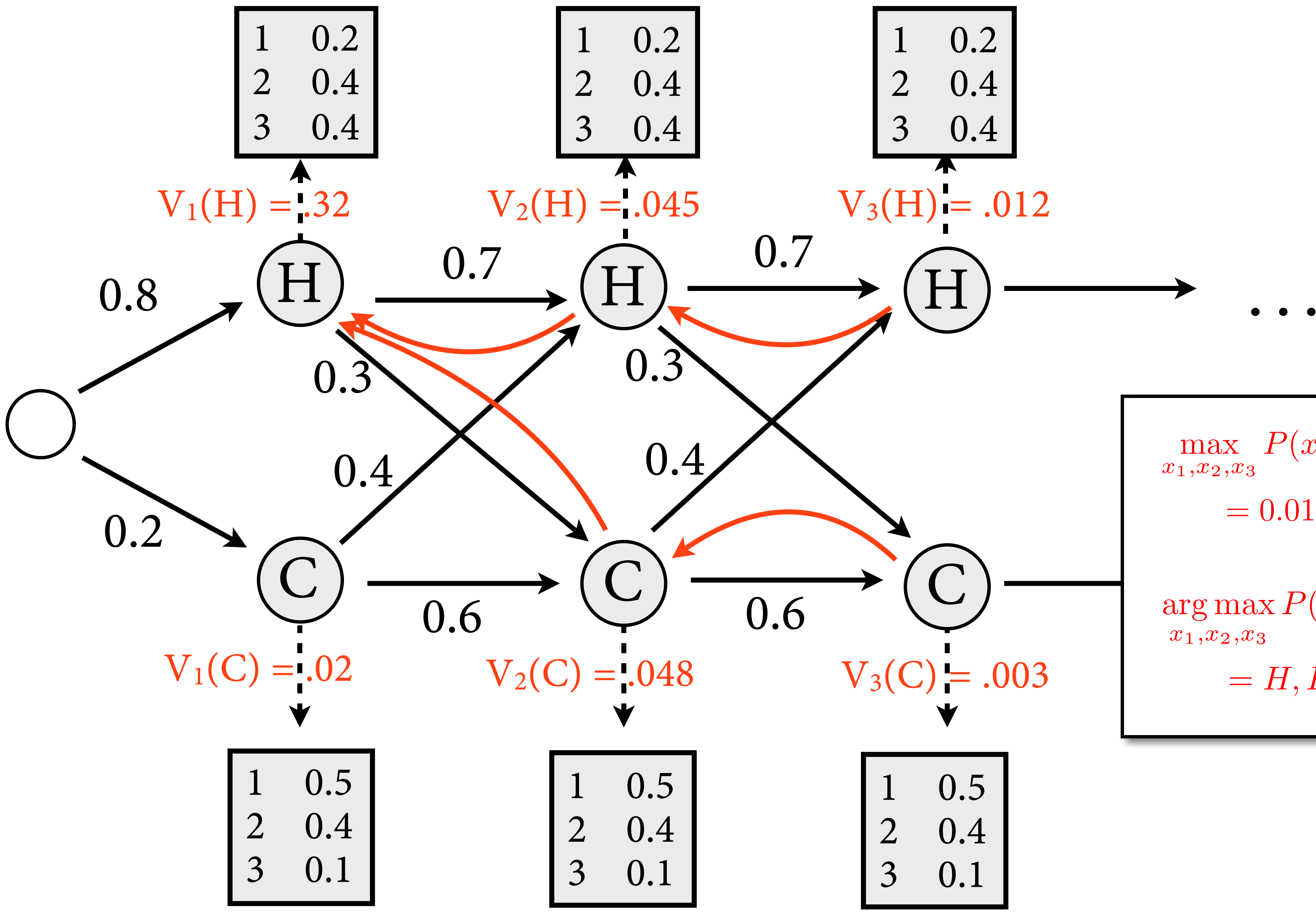
Example HMM: Eisner's Ice Cream



States represent weather on a given day: Hot, Cold

Outputs represent number of ice creams Jason eats that day

Viterbi Algorithm: Example



$$\max_{x_1,x_2,x_3} P(x_1,3,x_2,1,x_3,3) = 0.012$$

$$\arg\max_{x_1,x_2,x_3} P(x_1,3,x_2,1,x_3,3) = H,H,H$$

$$V_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1,\ldots,y_t,x_1,\ldots,x_{t-1},X_t=q_j)$$

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i)\cdot a_{ij}\cdot b_j(y_t)$$

The Forward Algorithm

- Key idea: *Forward probability* $\alpha_t(j)$ that HMM outputs $y_1, \ldots, y_t$ and then ends in $X_t = q_j$.

$$\begin{aligned}\alpha_t(j) &= P(y_1, \ldots, y_t, X_t = q_j) \\ &= \sum_{x_1, \ldots, x_{t-1}} P(y_1, \ldots, y_t, X_1 = x_1, \ldots, X_{t-1} = x_{t-1}, X_t = q_j)\end{aligned}$$

- From this, can compute easily

$$P(y_1, \ldots, y_T) = \sum_{q \in Q} \alpha_T(q)$$

The Forward Algorithm

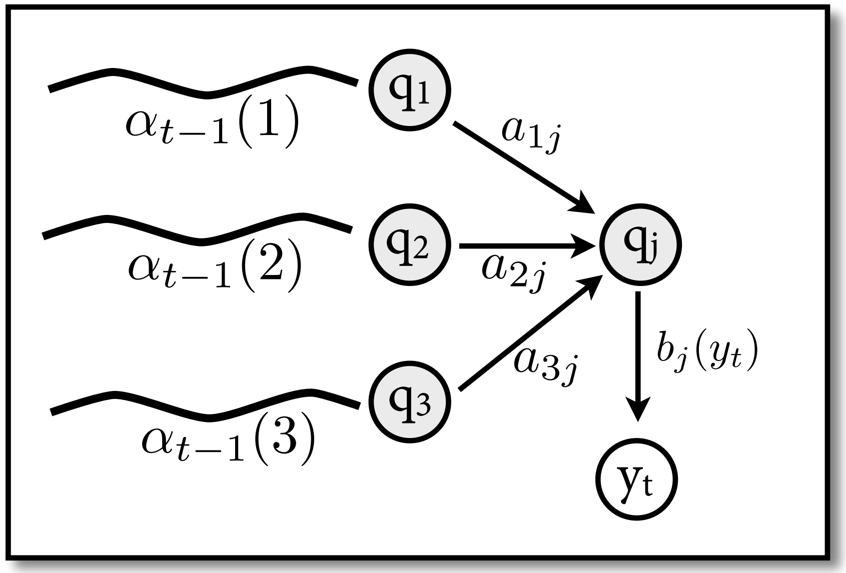
$$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$$

- Base case, t = 1:

$$\alpha_1(j) = P(y_1, X_1 = q_j) = b_j(y_1) \cdot a_{0j}$$

- Inductive case, compute for t = 2, …, T:

$$\begin{aligned} \alpha_t(j) &= P(y_1, \ldots, y_t, X_t = q_j) \\ &= \sum_{i=1}^{N} P(y_1, \ldots, y_{t-1}, X_{t-1} = q_i) \cdot P(X_t = q_j \mid X_{t-1} = q_i) \cdot P(y_t \mid X_t = q_j) \\ &= \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t) \end{aligned}$$



Question 3a: Supervised learning

- Given a set of POS tags and *annotated* training data $(w_1,t_1), \ldots, (w_T,t_T)$, compute parameters for HMM that maximize likelihood of training data.

DT	NN	VBD	NNS	IN	DT	NN
The	representative	put	chairs	on	the	table.

NNP	VBZ	VBN	TO	VB	NR
Secretariat	is	expected	to	race	tomorrow.

Maximum likelihood training

- Estimate bigram model for state sequence:

$$a_{ij} = \frac{C(X_t = q_i, X_{t+1} = q_j)}{C(X_t = q_i)} \qquad a_{0j} = \frac{\#\text{ sentences with } X_1 = q_j}{\#\text{ sentences}}$$

- ML estimate for emission probabilities:

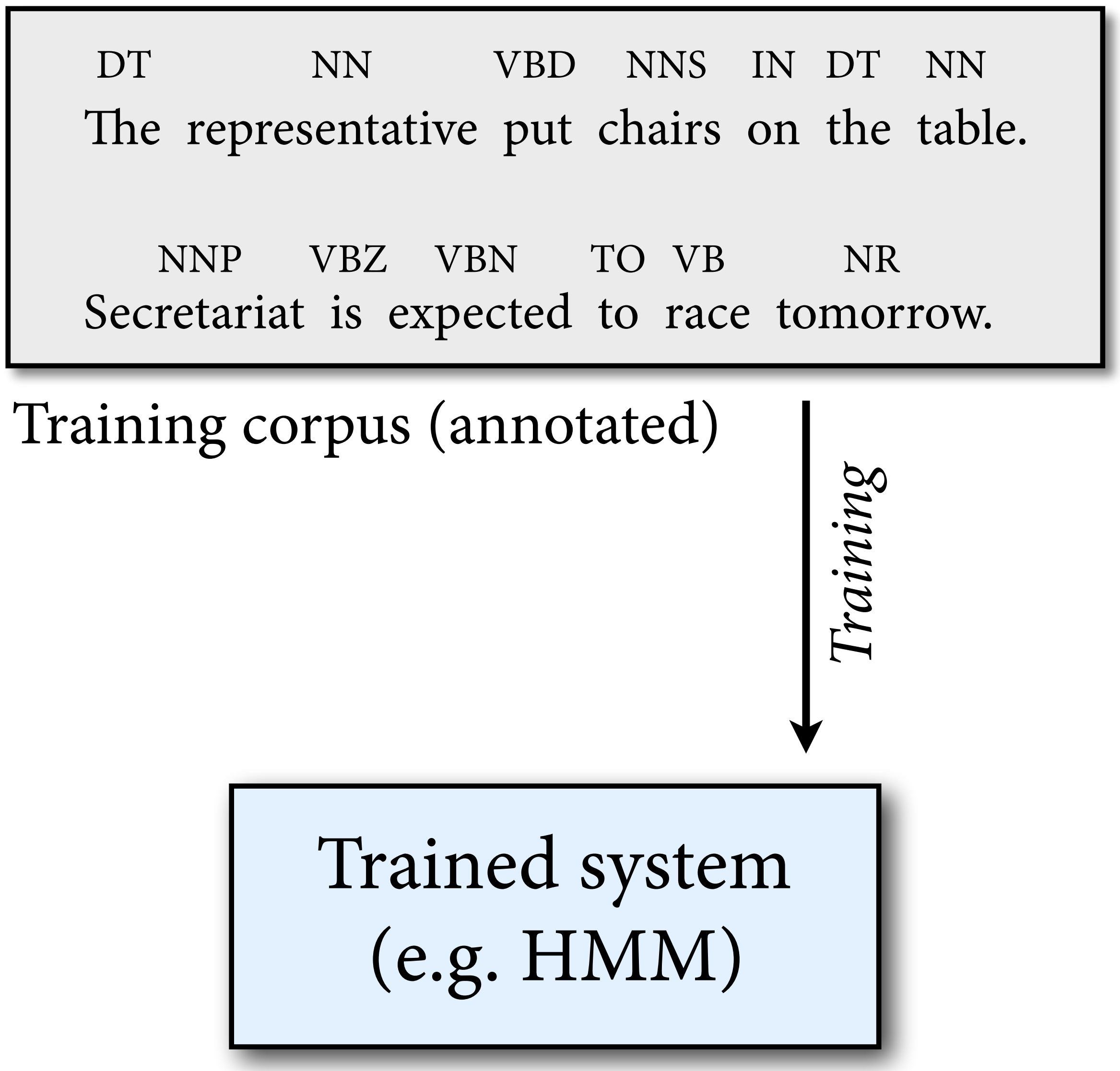
$$b_i(o) = \frac{C(X_t = q_i, Y_t = o)}{C(X_t = q_i)}$$

- Apply smoothing as you would for ordinary n-gram models (increase all counts C by one).

Evaluation

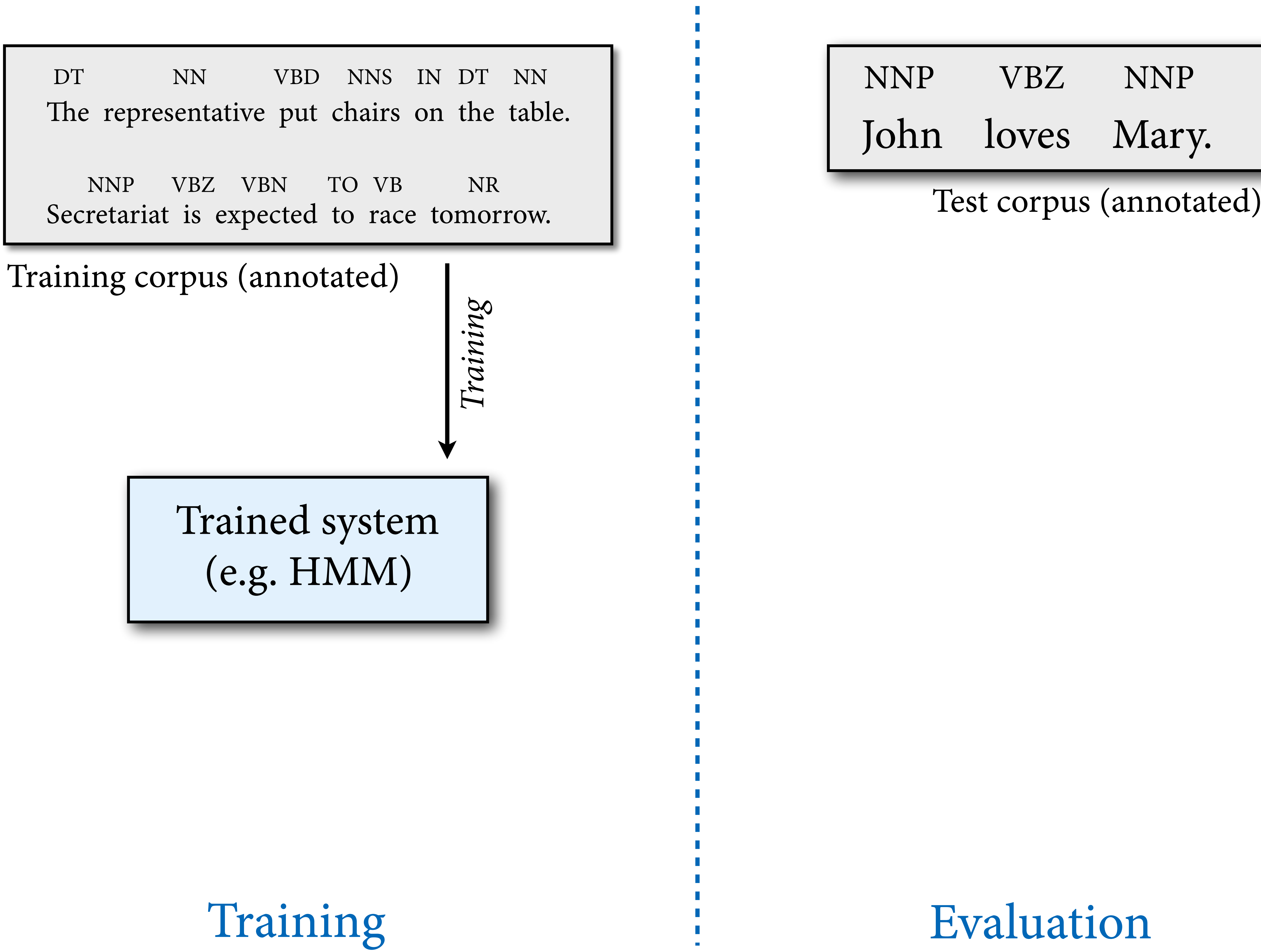
- How do you know how well your tagger works?
- Run it on *test data* and evaluate *accuracy*.
  - Test data: Really important to evaluate on unseen sentences to get a fair picture of how well tagger generalizes.
  - Accuracy: Measure percentage of correctly predicted POS tags.

Evaluation on test data

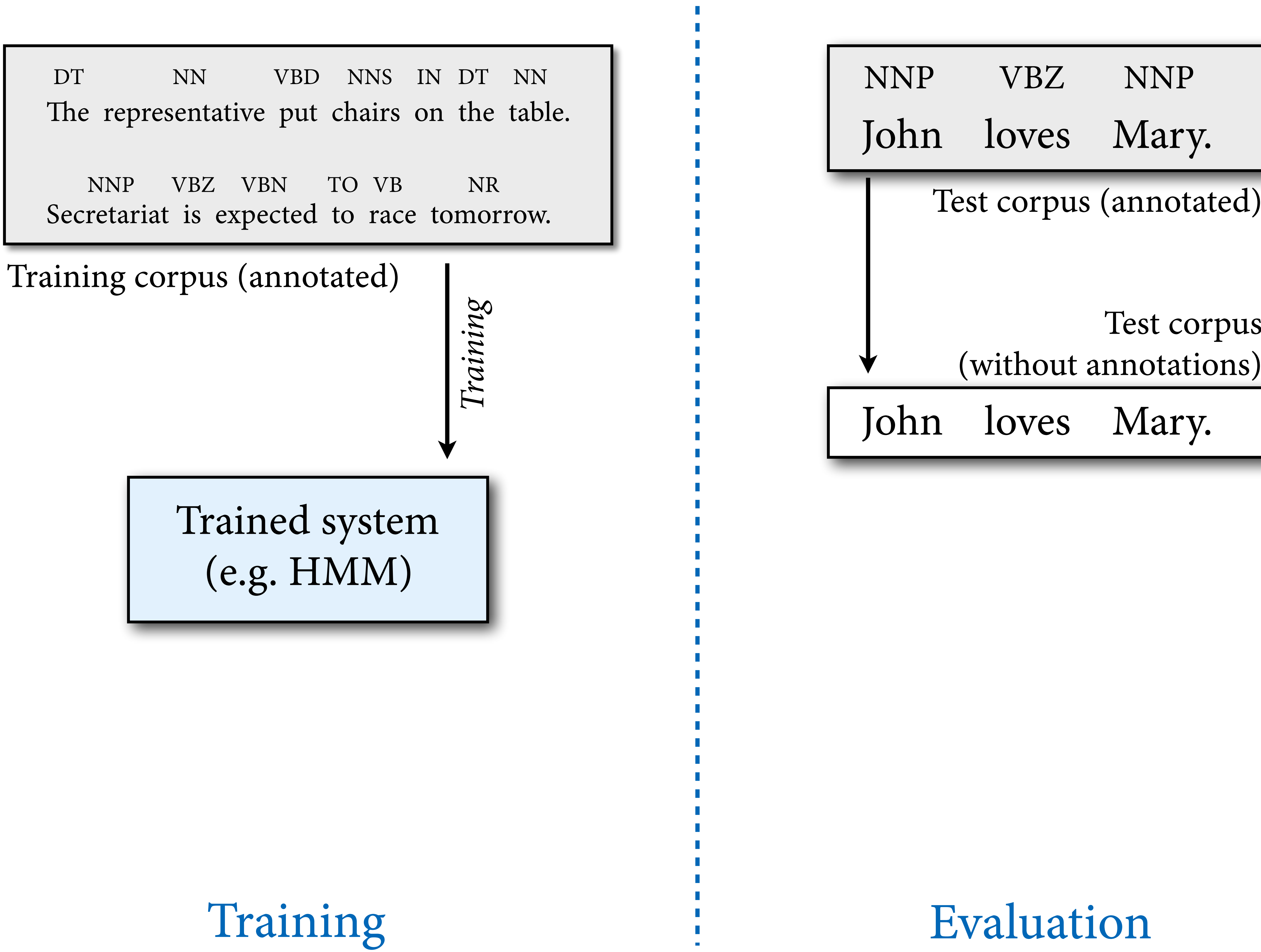


Training

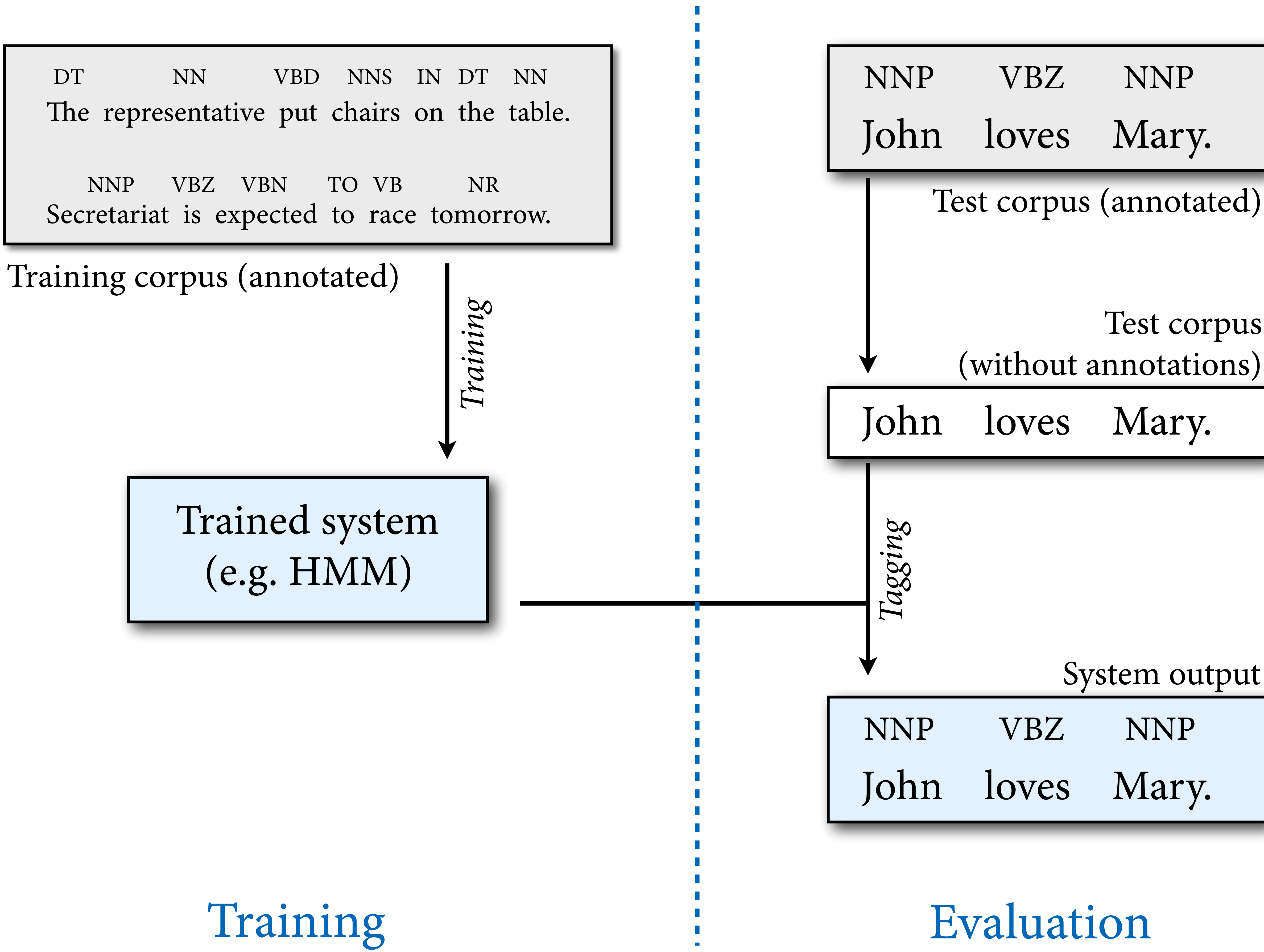
Evaluation on test data



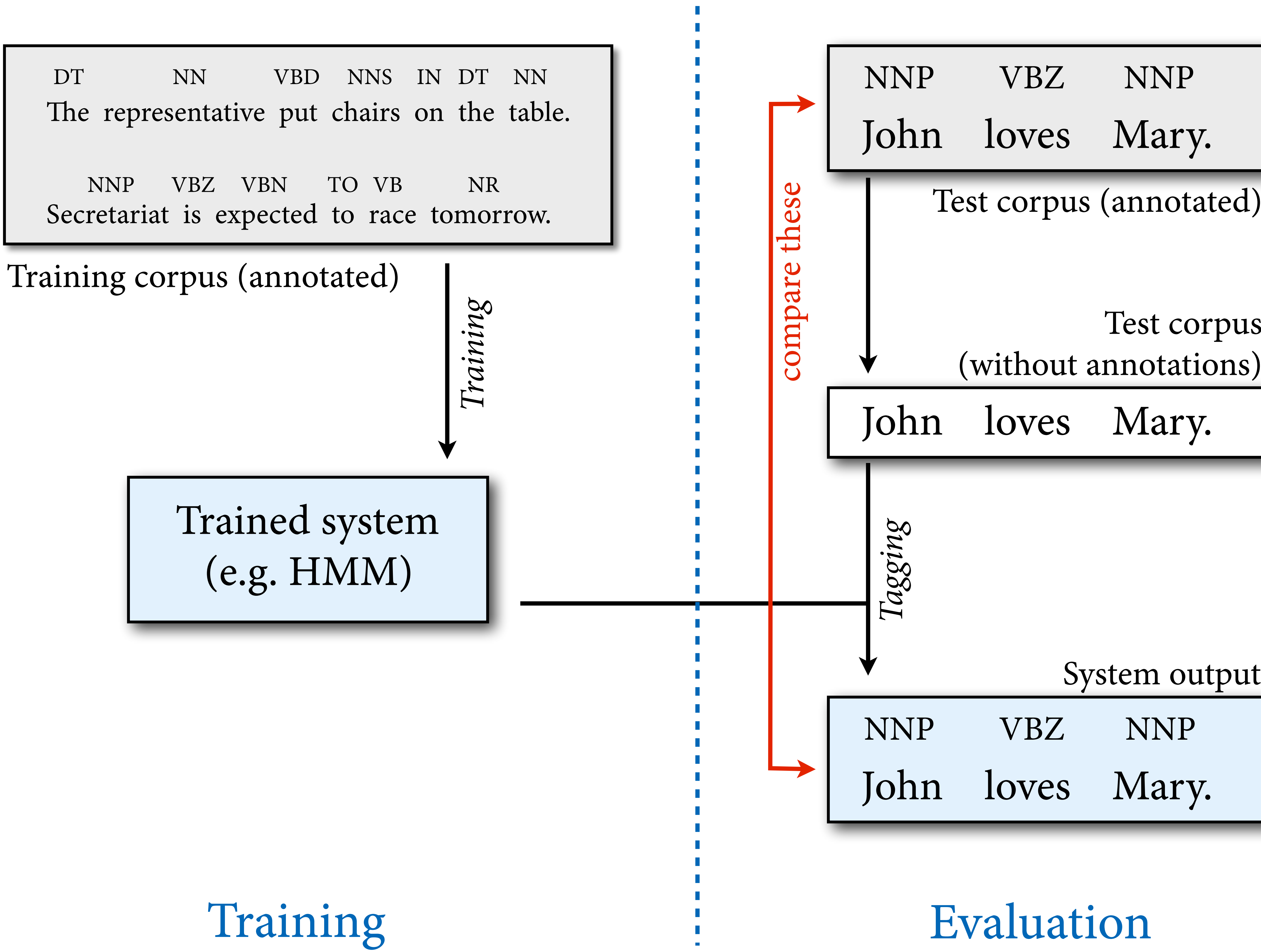
Evaluation on test data



Evaluation on test data



Evaluation on test data



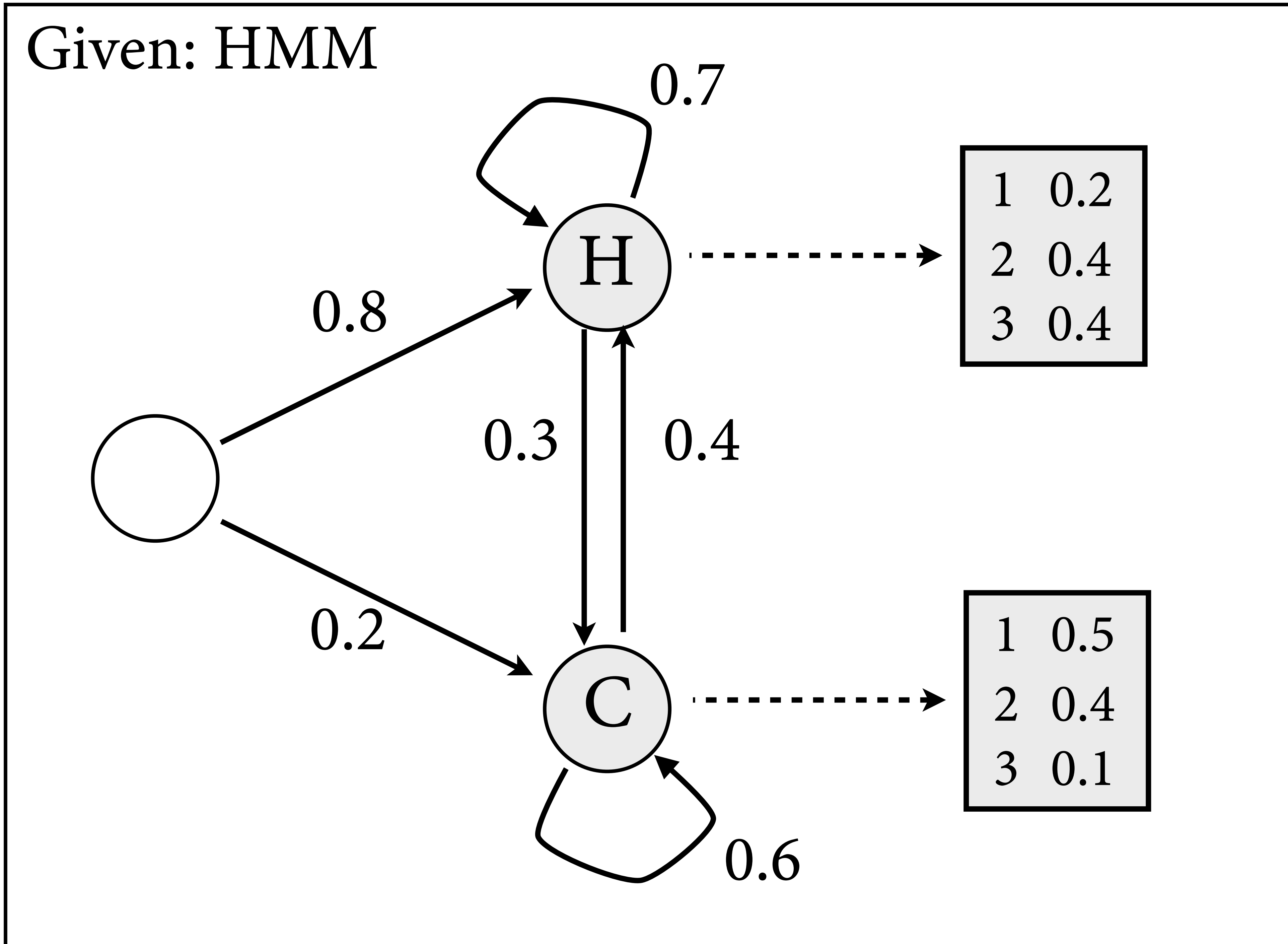
Question 3b: Unsupervised learning

- Given a set of POS tags and *unannotated* training data $w_1, \ldots, w_T$, compute parameters for HMM that maximize likelihood of training data.

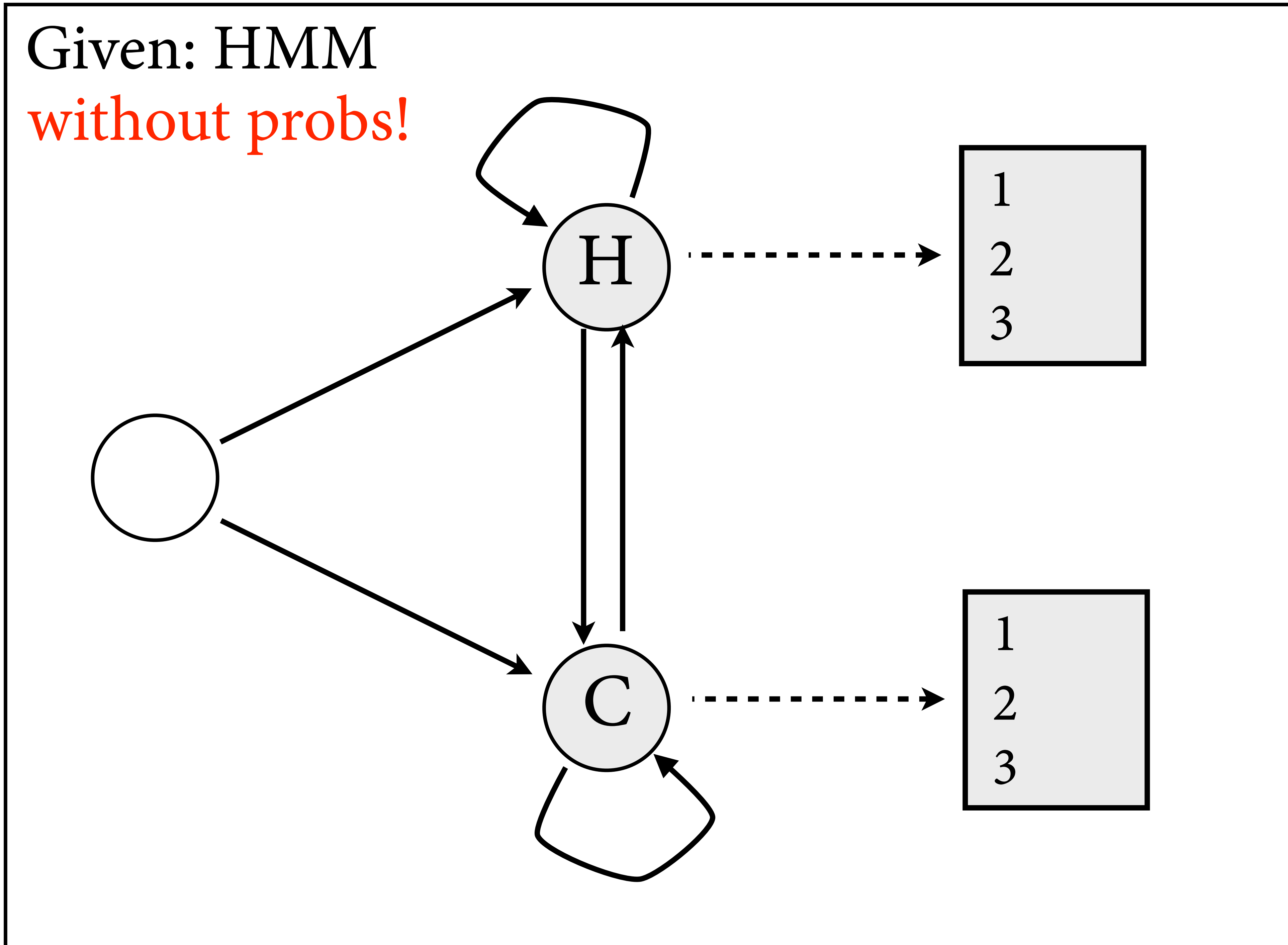
- Useful because annotated data is expensive to obtain, but raw text is really cheap.

The representative put chairs on the table.
Secretariat is expected to race today.

The setup

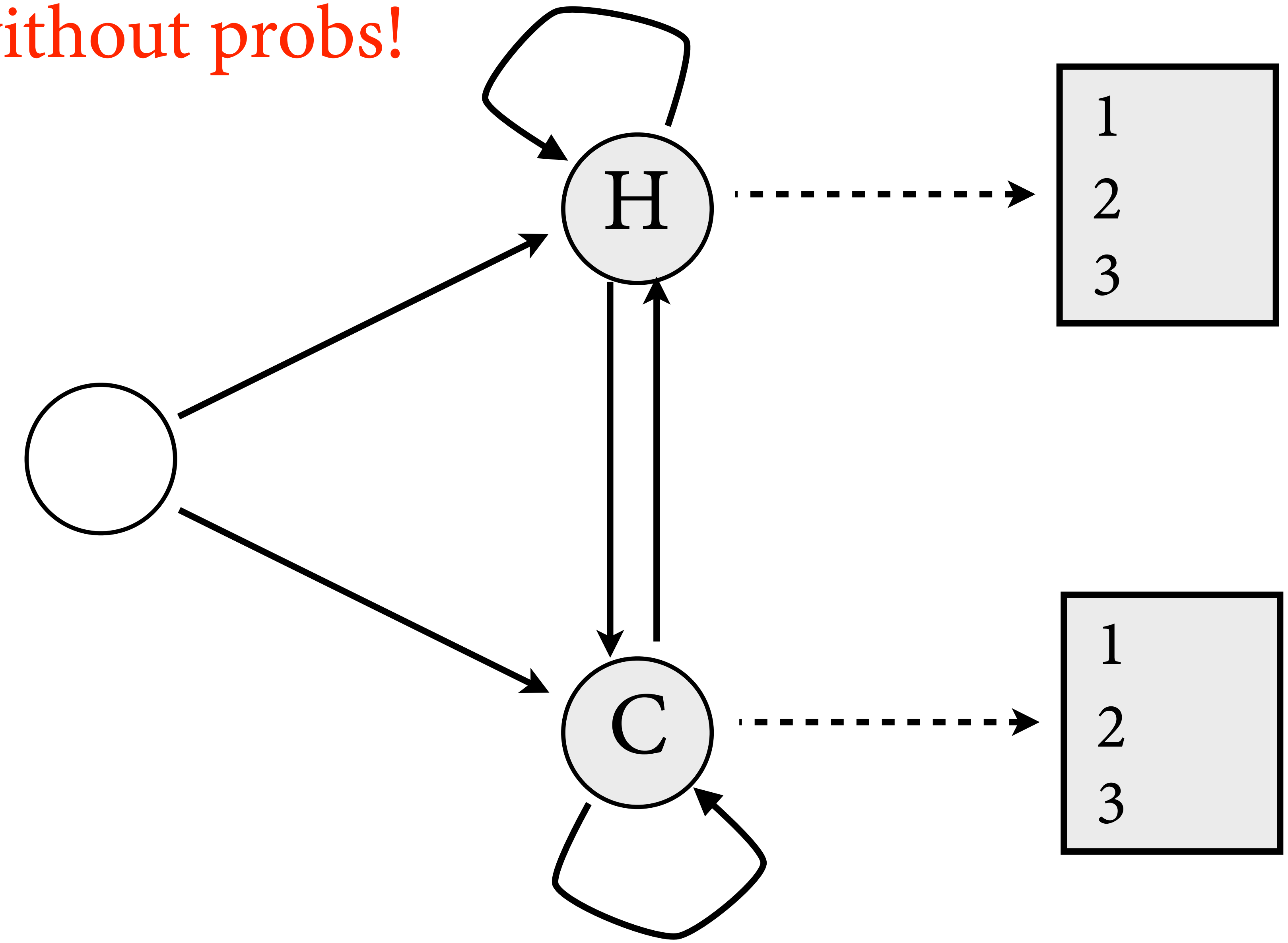


The setup



The setup

Given: HMM
without probs!



Observations: 2, 3, 3, 2, 3, 2, 3, 2, 2, 3, 1, 3, 3, …

The setup

- If we had counts of state transitions in corpus, we could simply use ML estimation.

$$a_{ij} = \frac{C(q_i \to q_j)}{C(q_i \to \bullet)}$$

- Idea: replace actual counts by *estimated* counts.

$$a_{ij} \approx \frac{\hat{C}(q_i \to q_j)}{\hat{C}(q_i \to \bullet)}$$

- How can we estimate counts?

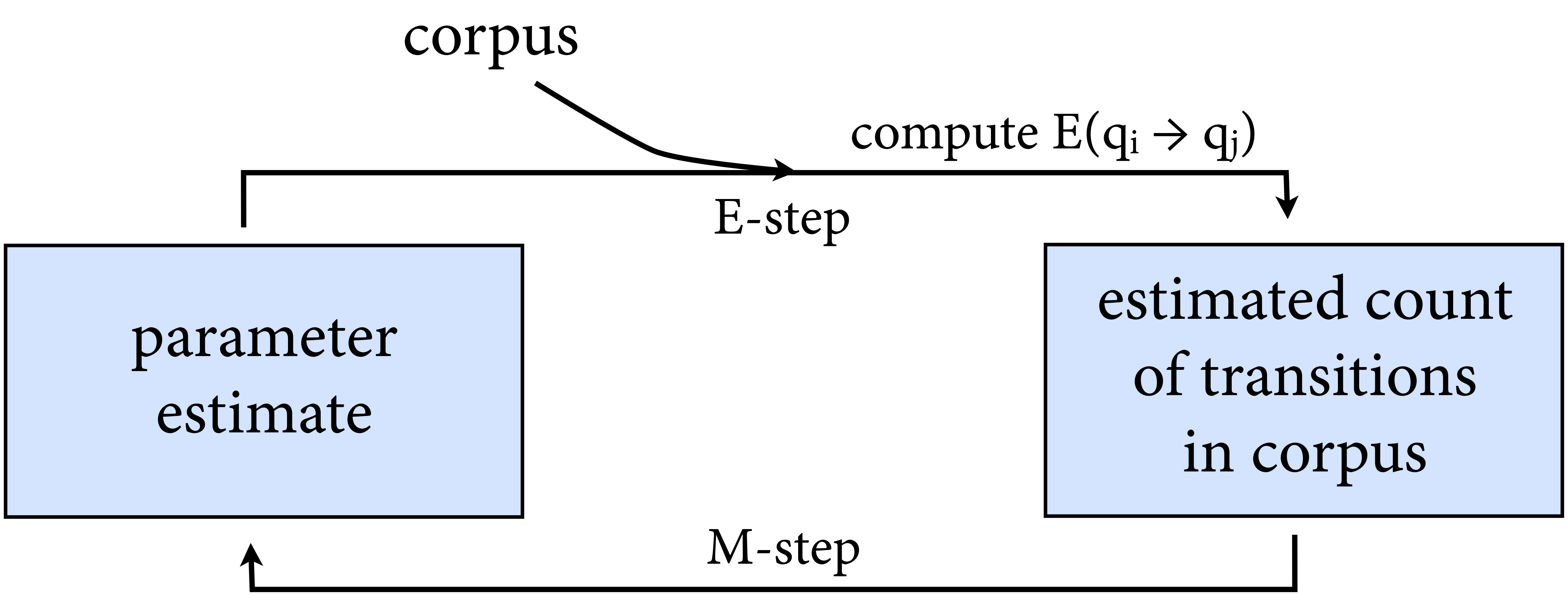
Estimated counts

Observations y	3	1	3		C(H → H)		P(x | y)
Hidden states x	H	H	H		2	*	0.408
	H	H	C	+	1	*	0.034
	H	C	H	+	0	*	0.272
		…					
	C	C	C	+	0	*	0.136
					0.864		

$$\hat{C}(q_i \to q_j) = E(q_i \to q_j) = \sum_x \hat{P}(x \mid y) \cdot C(q_i \to q_j \text{ in } x)$$

$$= \sum_{t=1}^{M-1} \hat{P}(X_t = q_i, X_{t+1} = q_j \mid y)$$

Expectation Maximization



$$a_{ij} \approx \frac{E(q_i \to q_j)}{E(q_i \to \bullet)} = \frac{E(q_i \to q_j)}{\sum_j E(q_i \to q_j)}$$

Plan for computing E

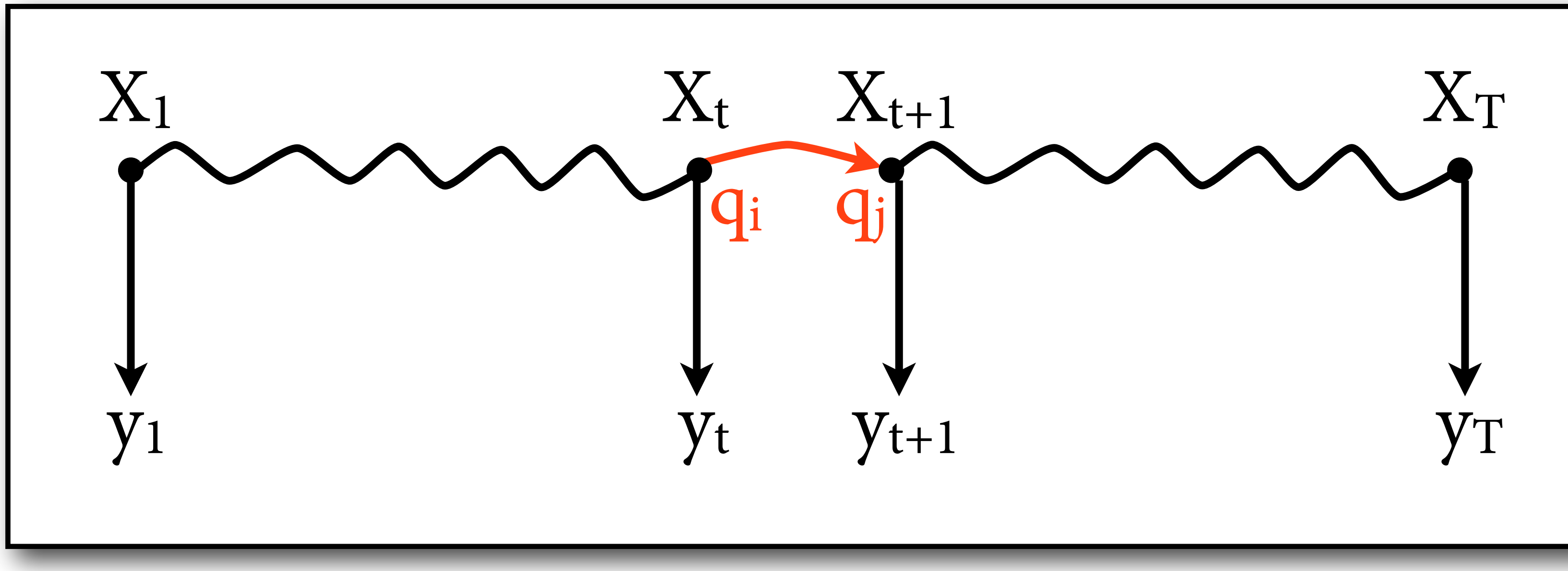
$$E(q_i \to q_j) = \sum_{t=1}^{M-1} \hat{P}(X_t = q_i, X_{t+1} = q_j \mid y)$$

- How can we compute $\hat{P}$ efficiently?
  Challenge: It is conditioned on y.

- We compute $\xi_t(i,j) = \hat{P}(X_t = q_i, X_{t+1} = q_j \mid y)$
  $$= \frac{\hat{P}(X_t = q_i, X_{t+1} = q_j, y)}{\hat{P}(y)}$$

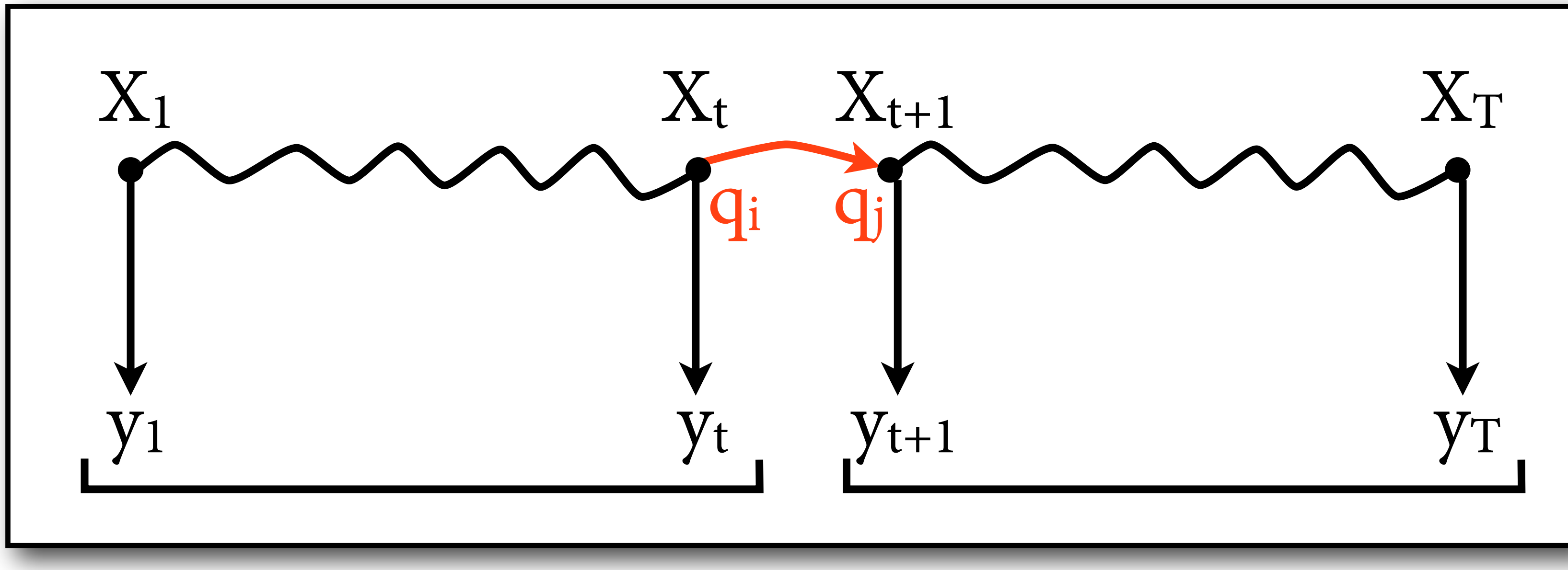
- Do it in two steps:
  - compute $\xi'_t(i,j) = \hat{P}(X_t = q_i, X_{t+1} = q_j, y)$
  - compute P(y)

$$\xi'_t(i,j) = \hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$



$$\hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$

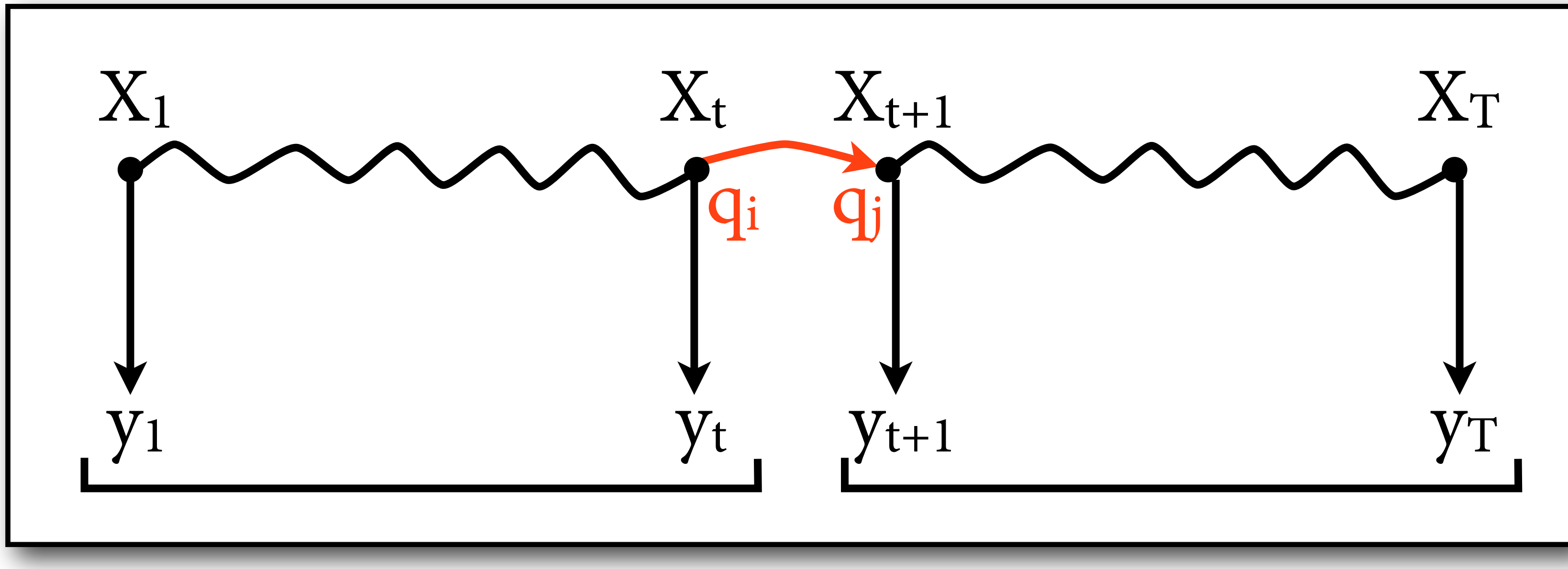
$$\xi'_t(i,j) = \hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$



$$\hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$

$$= \hat{P}(y_1, \ldots, y_t, X_t = q_i) \cdot \hat{P}(y_{t+1}, X_{t+1} = q_j \mid y_1, \ldots, y_t, X_t = q_i)$$
$$\cdot \hat{P}(y_{t+2}, \ldots, y_T \mid y_1, \ldots, y_{t+1}, X_t = q_i, X_{t+1} = q_j)$$

$$\xi'_t(i,j) = \hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$

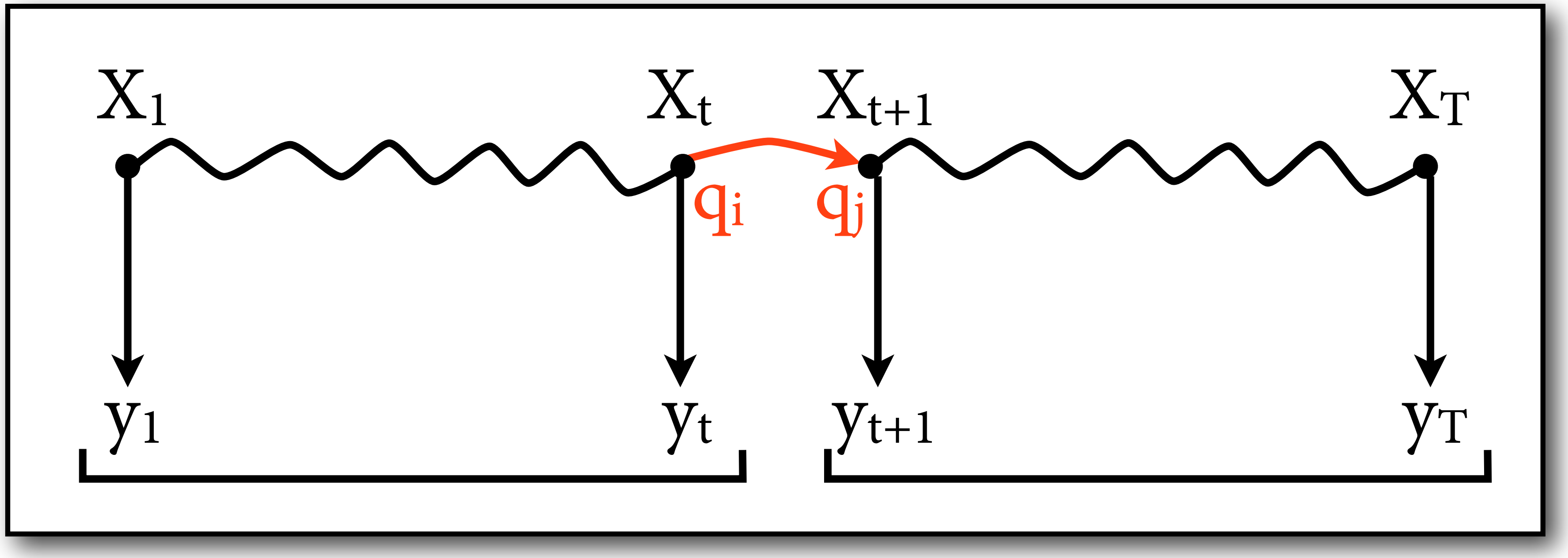


$$\hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$

$$= \hat{P}(y_1, \ldots, y_t, X_t = q_i) \cdot \hat{P}(y_{t+1}, X_{t+1} = q_j \mid y_1, \ldots, y_t, X_t = q_i)$$
$$\cdot \hat{P}(y_{t+2}, \ldots, y_T \mid y_1, \ldots, y_{t+1}, X_t = q_i, X_{t+1} = q_j)$$

$$= \hat{P}(y_1, \ldots, y_t, X_t = q_i) \cdot \hat{P}(y_{t+1}, X_{t+1} = q_j \mid X_t = q_i) \cdot \hat{P}(y_{t+2}, \ldots, y_T \mid X_{t+1} = q_j)$$

$$\xi'_t(i,j) = \hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$



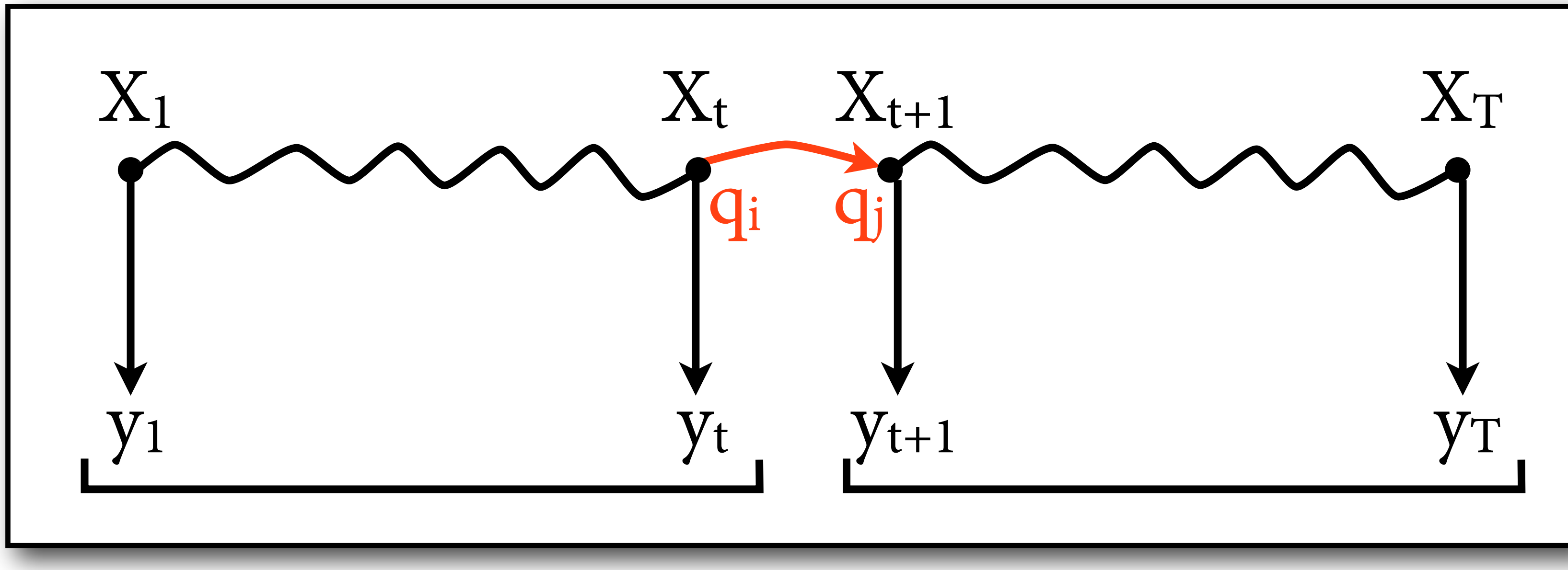
$$\hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$

$$= \hat{P}(y_1, \ldots, y_t, X_t = q_i) \cdot \hat{P}(y_{t+1}, X_{t+1} = q_j \mid y_1, \ldots, y_t, X_t = q_i)$$
$$\cdot \hat{P}(y_{t+2}, \ldots, y_T \mid y_1, \ldots, y_{t+1}, X_t = q_i, X_{t+1} = q_j)$$

$$= \hat{P}(y_1, \ldots, y_t, X_t = q_i) \cdot \hat{P}(y_{t+1}, X_{t+1} = q_j \mid X_t = q_i) \cdot \hat{P}(y_{t+2}, \ldots, y_T \mid X_{t+1} = q_j)$$

$$\cdot a_{ij} \cdot b_j(w_{t+1}) \cdot$$

$$\xi'_t(i,j) = \hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$



$$\hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$

$$= \hat{P}(y_1, \ldots, y_t, X_t = q_i) \cdot \hat{P}(y_{t+1}, X_{t+1} = q_j \mid y_1, \ldots, y_t, X_t = q_i)$$
$$\cdot \hat{P}(y_{t+2}, \ldots, y_T \mid y_1, \ldots, y_{t+1}, X_t = q_i, X_{t+1} = q_j)$$

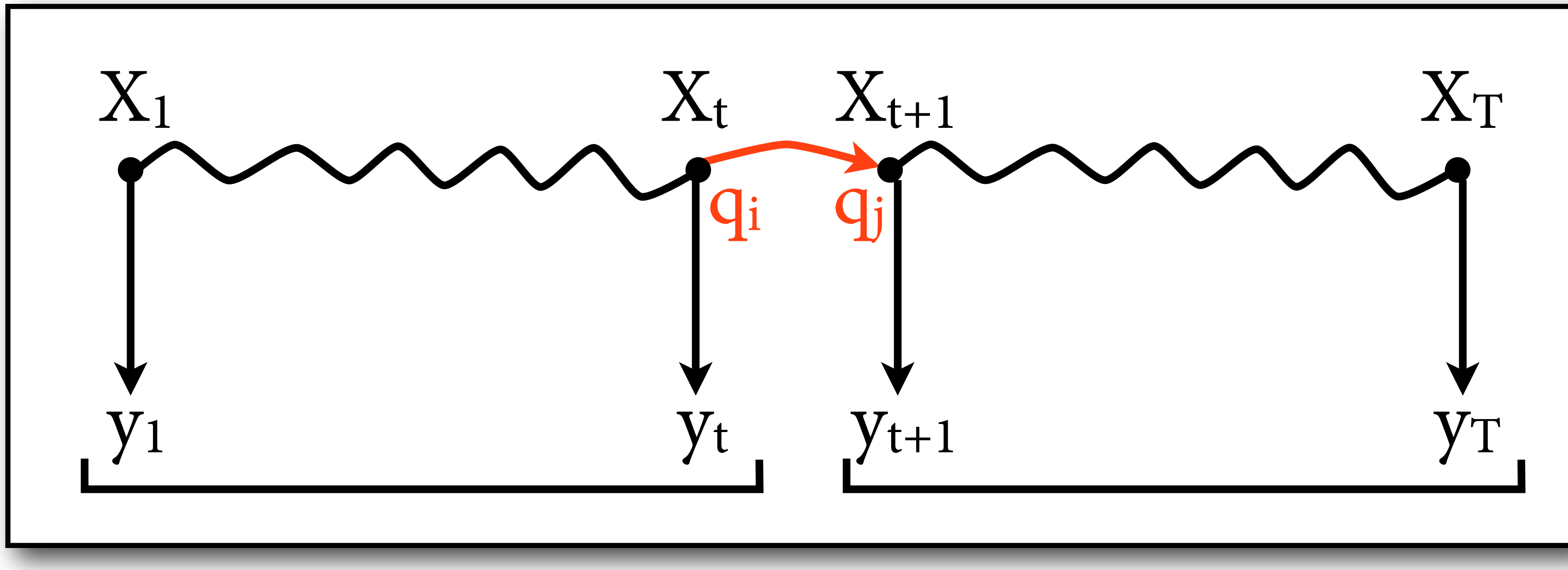
$$= \hat{P}(y_1, \ldots, y_t, X_t = q_i) \cdot \hat{P}(y_{t+1}, X_{t+1} = q_j \mid X_t = q_i) \cdot \hat{P}(y_{t+2}, \ldots, y_T \mid X_{t+1} = q_j)$$

$$= \alpha_t(i) \quad \cdot a_{ij} \cdot b_j(w_{t+1}) \cdot$$

forward prob:

$$\alpha_t(i) = P(y_1, \ldots, y_t, X_t = q_i)$$

$$\xi'_t(i,j) = \hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$



$$\hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$

$$= \hat{P}(y_1, \dots, y_t, X_t = q_i) \cdot \hat{P}(y_{t+1}, X_{t+1} = q_j \mid y_1, \dots, y_t, X_t = q_i)$$
$$\cdot \hat{P}(y_{t+2}, \dots, y_T \mid y_1, \dots, y_{t+1}, X_t = q_i, X_{t+1} = q_j)$$

$$= \hat{P}(y_1, \dots, y_t, X_t = q_i) \cdot \hat{P}(y_{t+1}, X_{t+1} = q_j \mid X_t = q_i) \cdot \hat{P}(y_{t+2}, \dots, y_T \mid X_{t+1} = q_j)$$

$$= \alpha_t(i) \cdot a_{ij} \cdot b_j(w_{t+1}) \cdot \beta_{t+1}(j)$$

forward prob:

$$\alpha_t(i) = P(y_1, \dots, y_t, X_t = q_i)$$

backward prob:

$$\beta_t(i) = P(y_{t+1}, \dots, y_t \mid X_t = q_i)$$

Backward probabilities

$$\beta_t(i) = P(y_{t+1}, \dots, y_t \mid X_t = q_i)$$

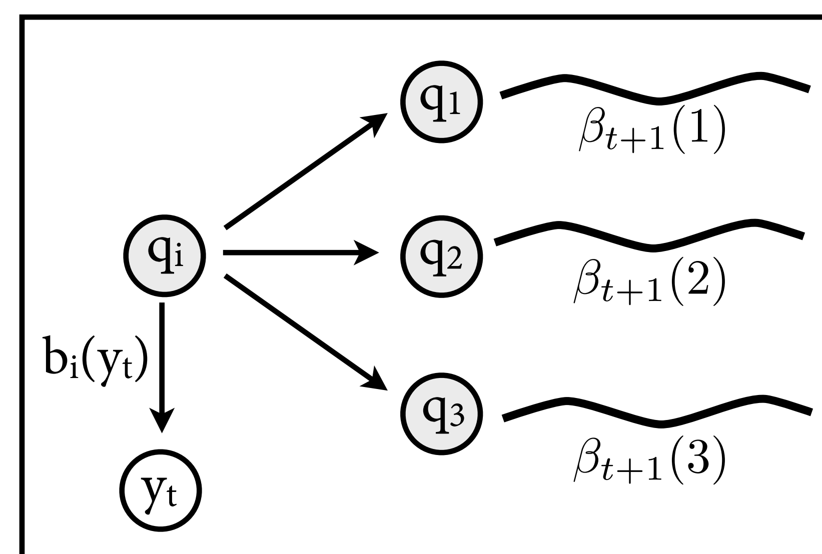
- Base case, t = T:

  $\beta_T(i) = 1$ for all i *

- Inductive case, compute for t = T-1, …, 1:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \cdot b_j(y_{t+1}) \cdot \beta_{t+1}(j)$$

- Exact mirror image of forward.



*) this is different in J&M because of q_F

Putting it all together

- Compute estimated transition counts for all i, j, t:

$$\xi_t(i,j) = \frac{\xi'_t(i,j)}{\hat{P}(y)} = \frac{\alpha_t(i) \cdot a_{ij} \cdot b_j(y_{t+1}) \cdot \beta_{t+1}(j)}{\sum_q \alpha_T(q)}$$

- Compute overall estimated transition counts:

$$E(q_i \to q_j) = \sum_{t=1}^{T-1} \xi_t(i,j)$$

- Revised estimate of transition probabilities:

$$a_{ij} \approx \frac{E(q_i \to q_j)}{E(q_i \to \bullet)}$$

The other parameters

- Revise initial and emission probabilities using estimated counts, in completely analogous way.

- Here's what it looks like for emission prob:

$$\gamma_t(j) = P(X_t = q_j \mid y) = \frac{\hat{P}(X_t = q_j, y)}{\hat{P}(y)} = \frac{\alpha_t(j) \cdot \beta_t(j)}{\hat{P}(y)}$$

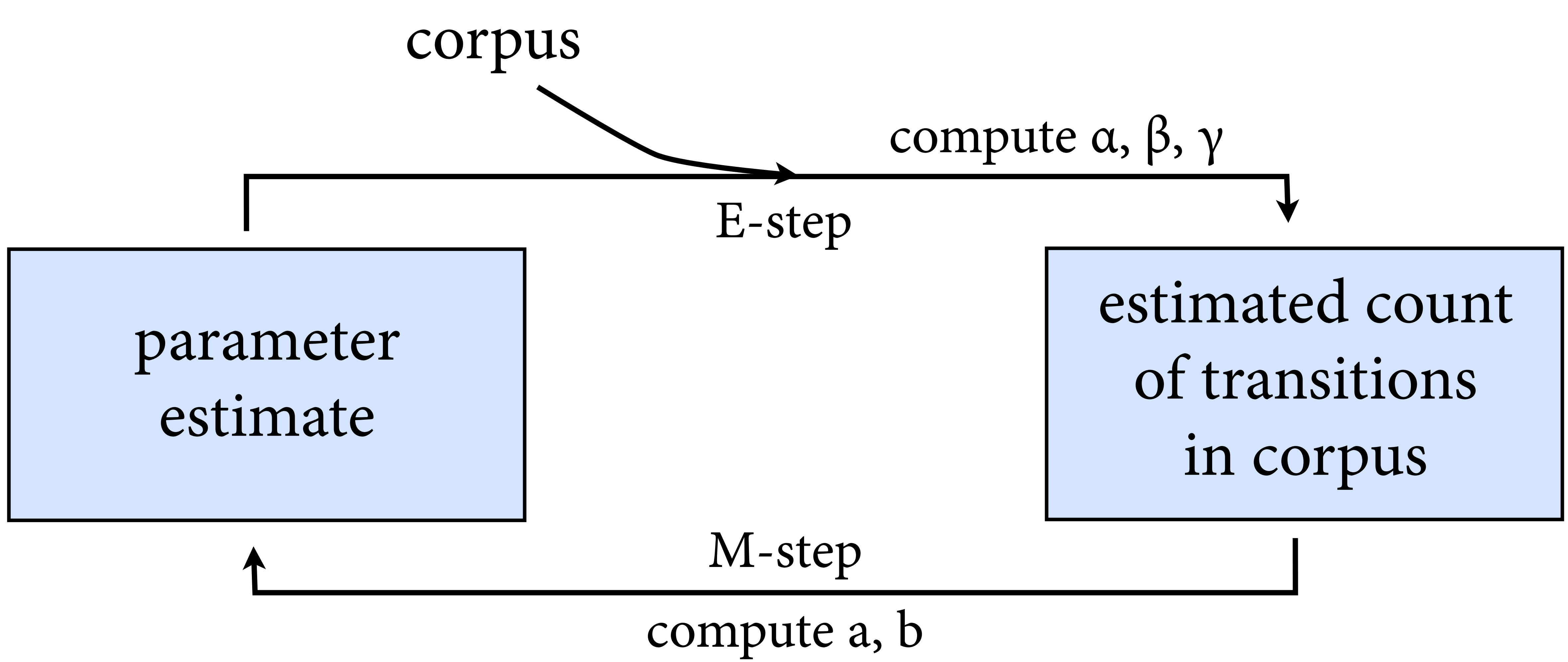
$$b_j(o) \approx \left(\sum_{\substack{t=1 \\ y_t = o}}^{T} \gamma_t(j) \right) \Big/ \sum_{t=1}^{T} \gamma_t(j)$$

estimated count of o emitted in state q_j

estimated count of state q_j

Forward-Backward Algorithm

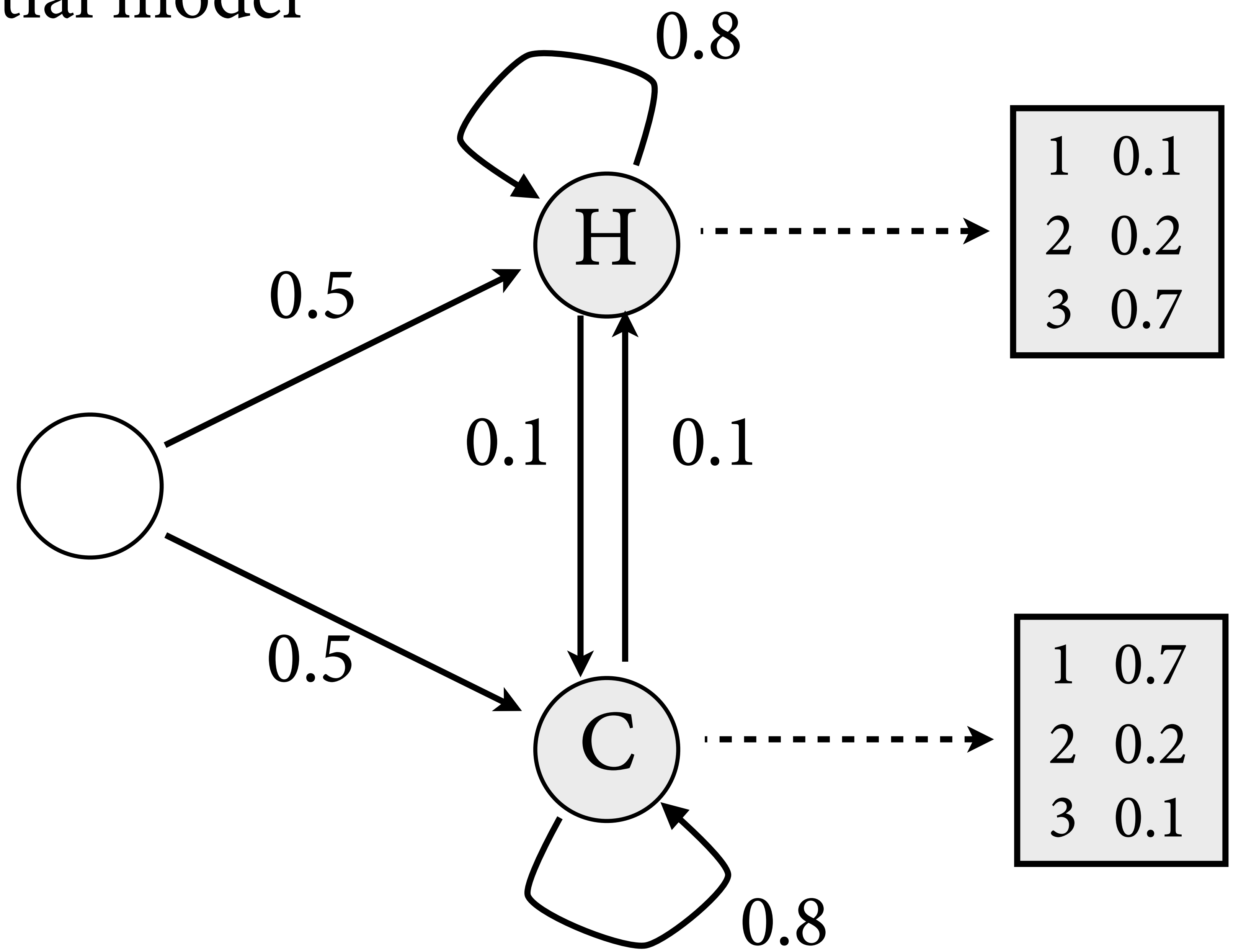
Initialization: start with some estimation of parameters.



Continue computation until parameters don’t change much.

Example

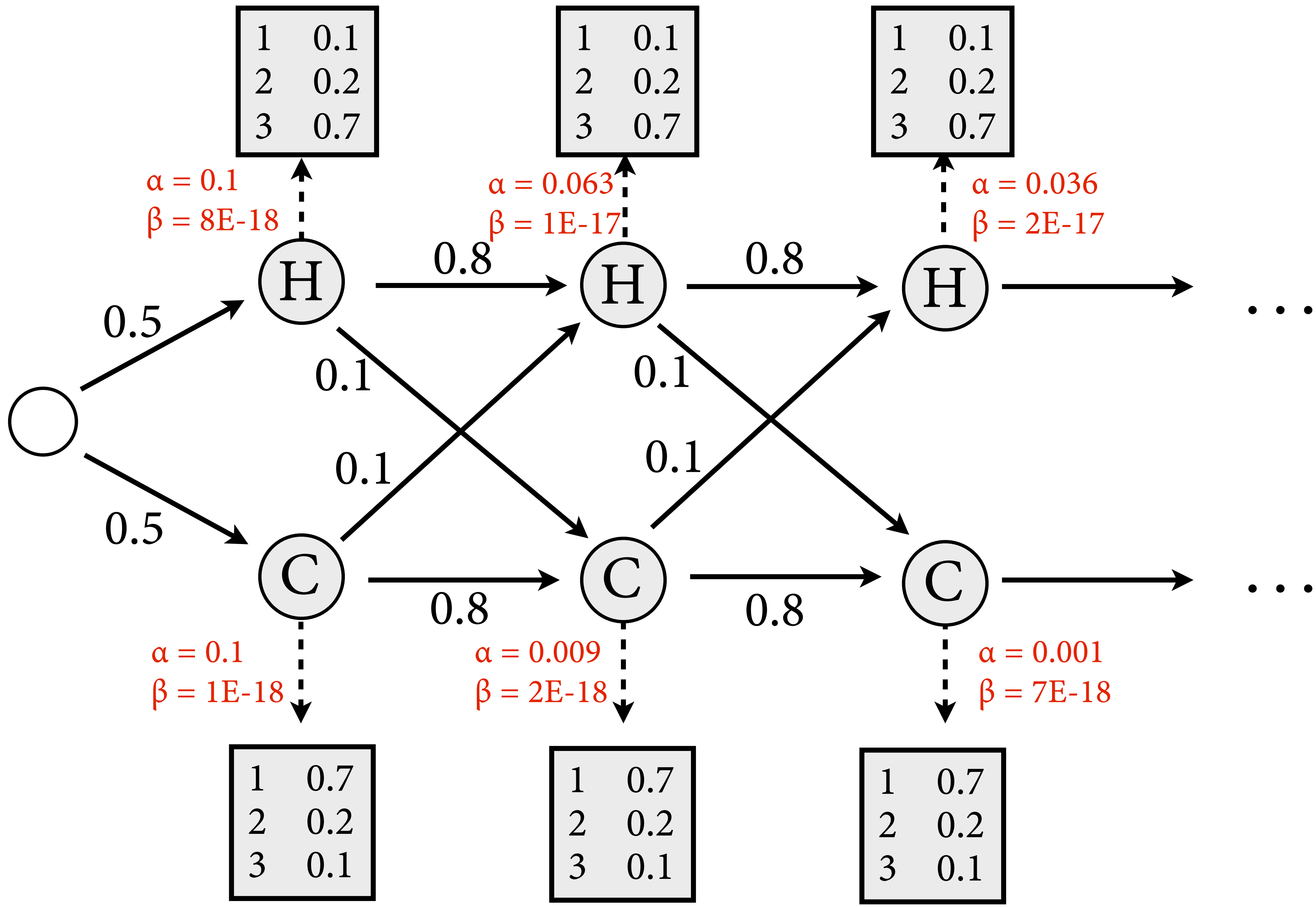
Step 1: initial model



Probs do not sum to one because Eisner's HMMs have q_F.

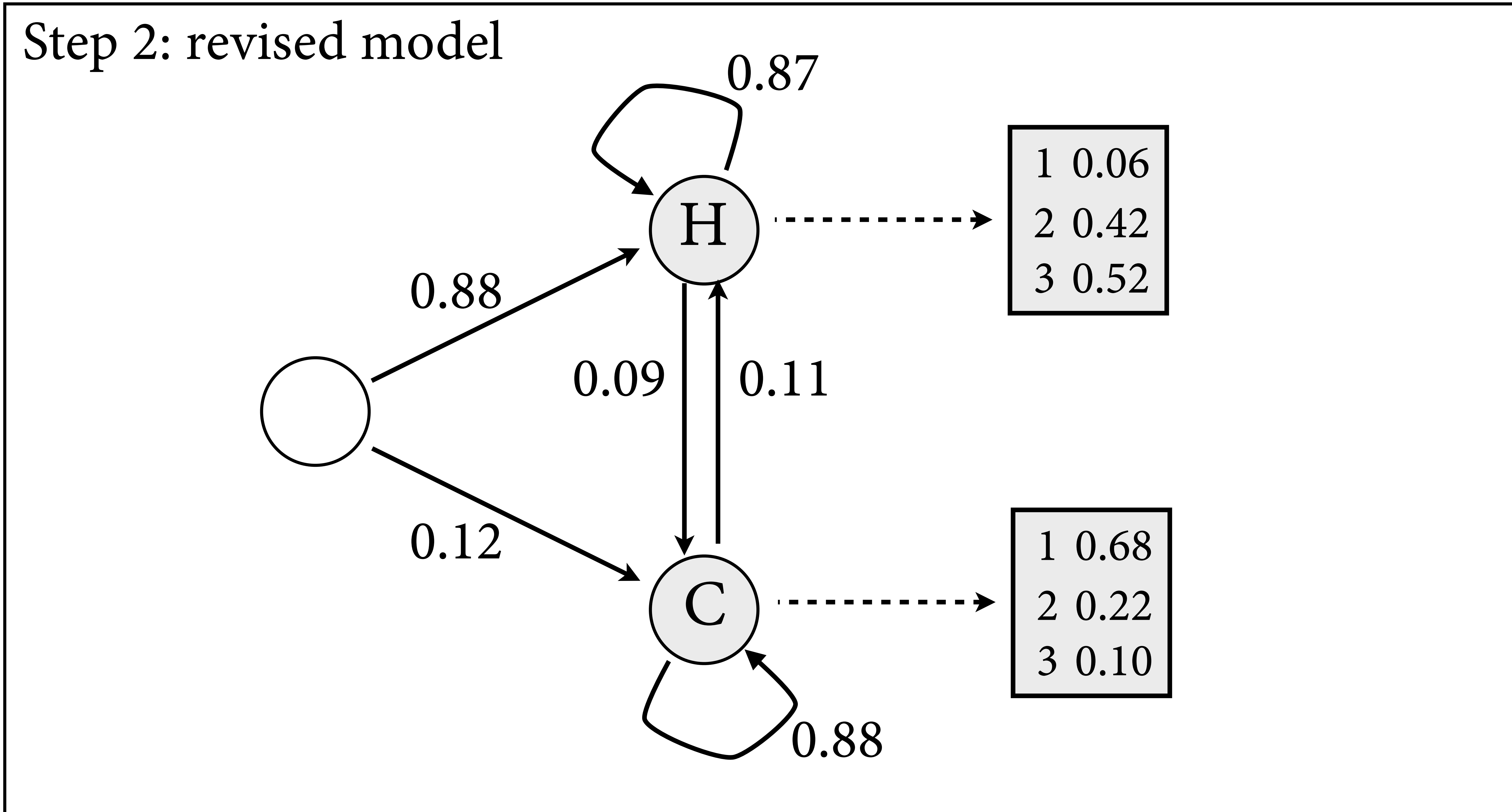
2, 3, 3, 2, 3, 2, 3, 2, 2, 3, 1, 3, 3, 1, 1, 1, 2, 1, 1, 1, 3, 1, 2, 1, 1, 1, 2, 3, 3, 2, 3, 2, 2

E-Step



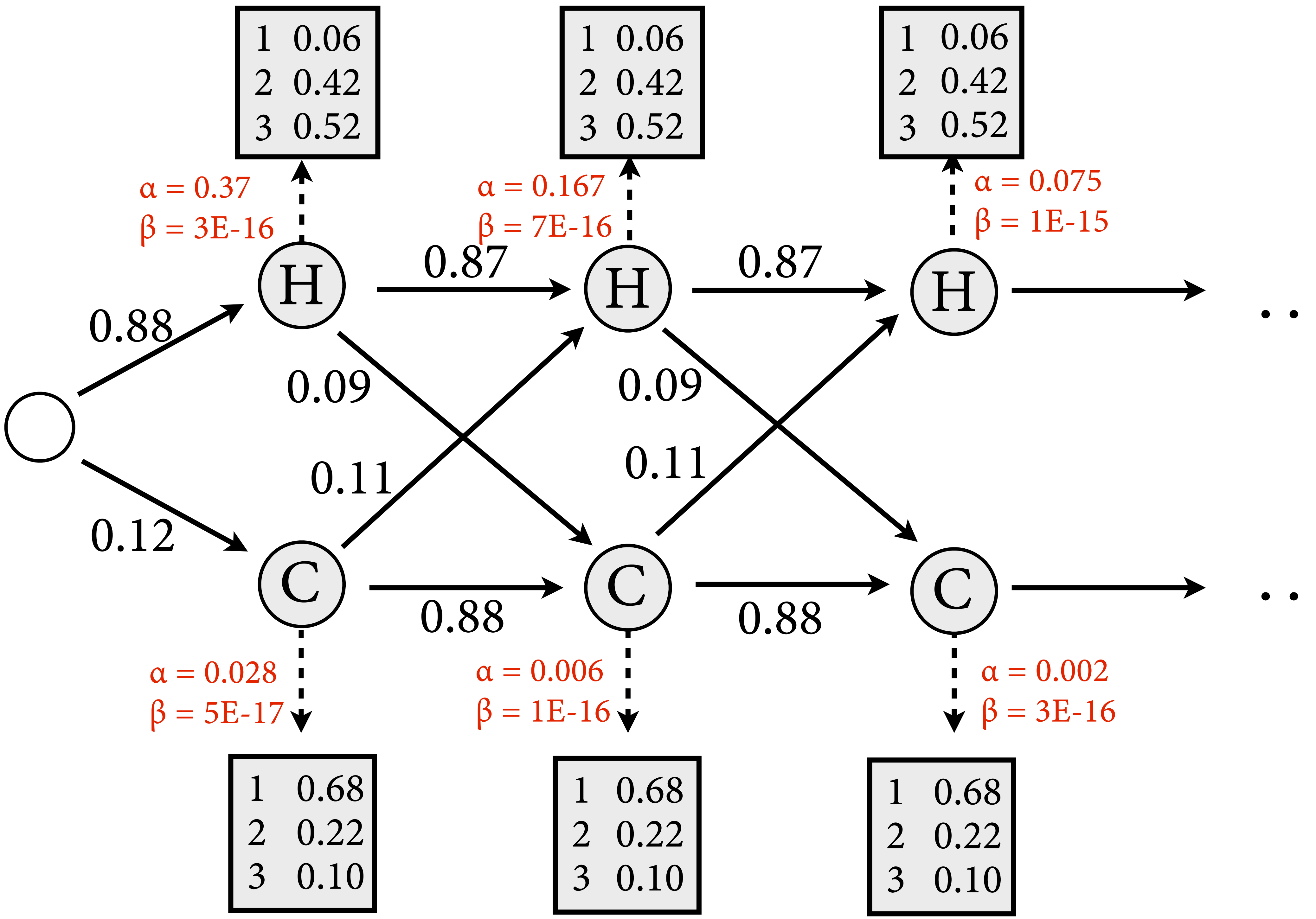
1 0.1
2 0.2
3 0.7
α = 0.1
β = 8E-18
α = 0.063
β = 1E-17
α = 0.036
β = 2E-17
0.5
0.5
H
C
0.8
0.1
0.1
0.8
H
C
0.8
0.1
0.1
0.8
H
C
. . .
. . .
α = 0.1
β = 1E-18
α = 0.009
β = 2E-18
α = 0.001
β = 7E-18
1 0.7
2 0.2
3 0.1

M-Step

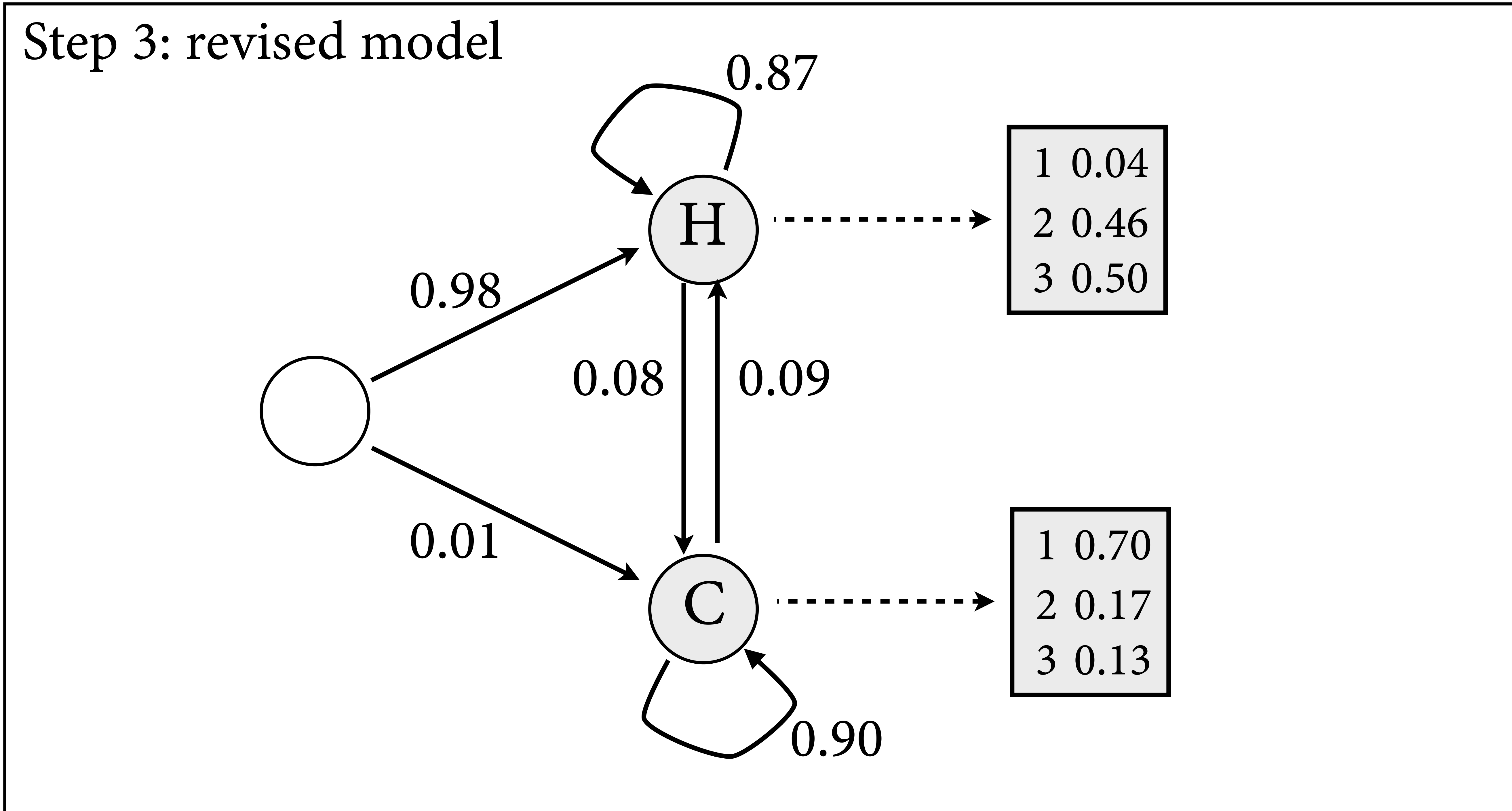


2, 3, 3, 2, 3, 2, 3, 2, 2, 3, 1, 3, 3, 1, 1, 1, 2, 1, 1, 1, 3, 1, 2, 1, 1, 1, 2, 3, 3, 2, 3, 2, 2

E-Step

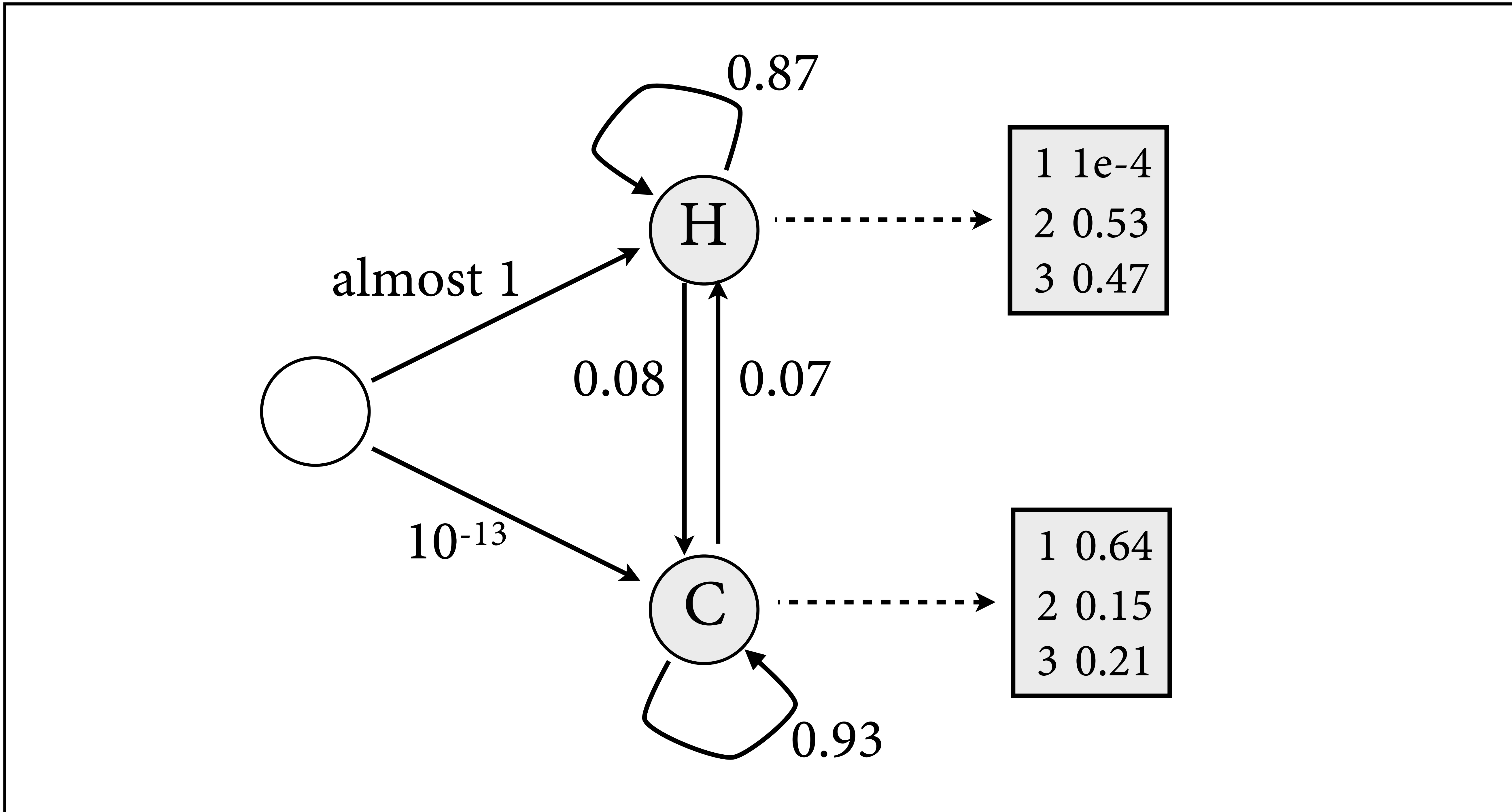

1 0.06
2 0.42
3 0.52
1 0.06
2 0.42
3 0.52
1 0.06
2 0.42
3 0.52
α = 0.37
β = 3E-16
α = 0.167
β = 7E-16
α = 0.075
β = 1E-15
H
H
H
0.87
0.87
0.88
0.09
0.09
0.11
0.11
0.12
C
C
C
0.88
0.88
. . .
. . .
α = 0.028
β = 5E-17
α = 0.006
β = 1E-16
α = 0.002
β = 3E-16
1 0.68
2 0.22
3 0.10
1 0.68
2 0.22
3 0.10
1 0.68
2 0.22
3 0.10

M-Step



2, 3, 3, 2, 3, 2, 3, 2, 2, 3, 1, 3, 3, 1, 1, 1, 2, 1, 1, 1, 3, 1, 2, 1, 1, 1, 2, 3, 3, 2, 3, 2, 2

Result after 10 iterations



2, 3, 3, 2, 3, 2, 3, 2, 2, 3, 1, 3, 3, 1, 1, 1, 2, 1, 1, 1, 3, 1, 2, 1, 1, 1, 2, 3, 3, 2, 3, 2, 2

Some remarks

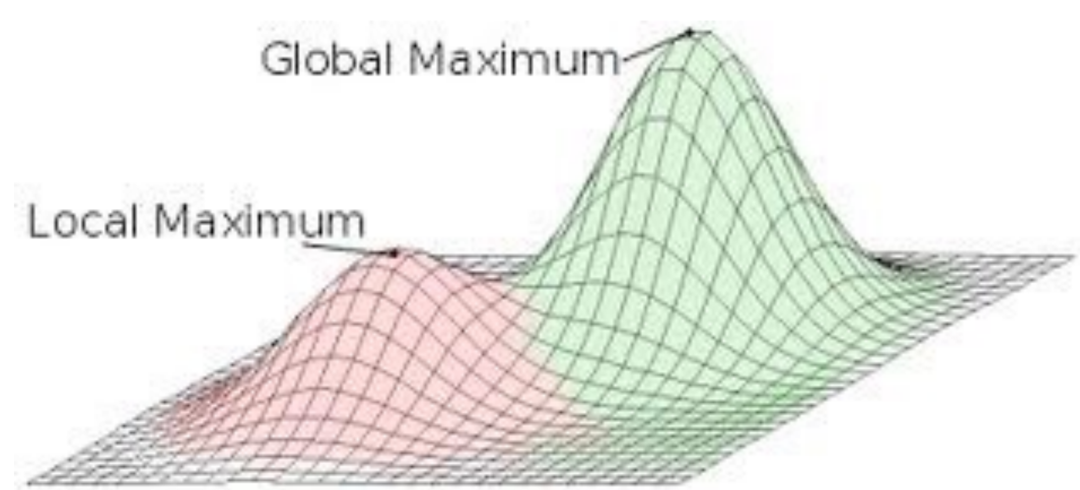
- Forward-backward algorithm also called *Baum-Welch Algorithm* after inventors.

- Special case of the *expectation maximization* algorithm:
  - E-Step: Compute expected values of relevant counts based on current parameter estimate.
  - M-Step: Adjust model based on estimated counts.

- Runtime of each iteration is $O(N^2 T)$.
  Most of the time goes into E-step.

Some remarks

- EM algorithm is guaranteed to improve likelihood of corpus in each iteration.
- However, can run into *local maxima:* would have to go through worse model to find globally best one.
- Extremely sensitive to initial parameter estimate. Only useful in practice if HMM structure very strongly constrained (e.g. speech recognition).



Conclusion

- Evaluate tagger on *accuracy* on *unseen* data.
- Training algorithms for HMM estimation:
  - *supervised* training from annotated data: maximum likelihood
  - *unsupervised* training from unannotated data: forward-backward