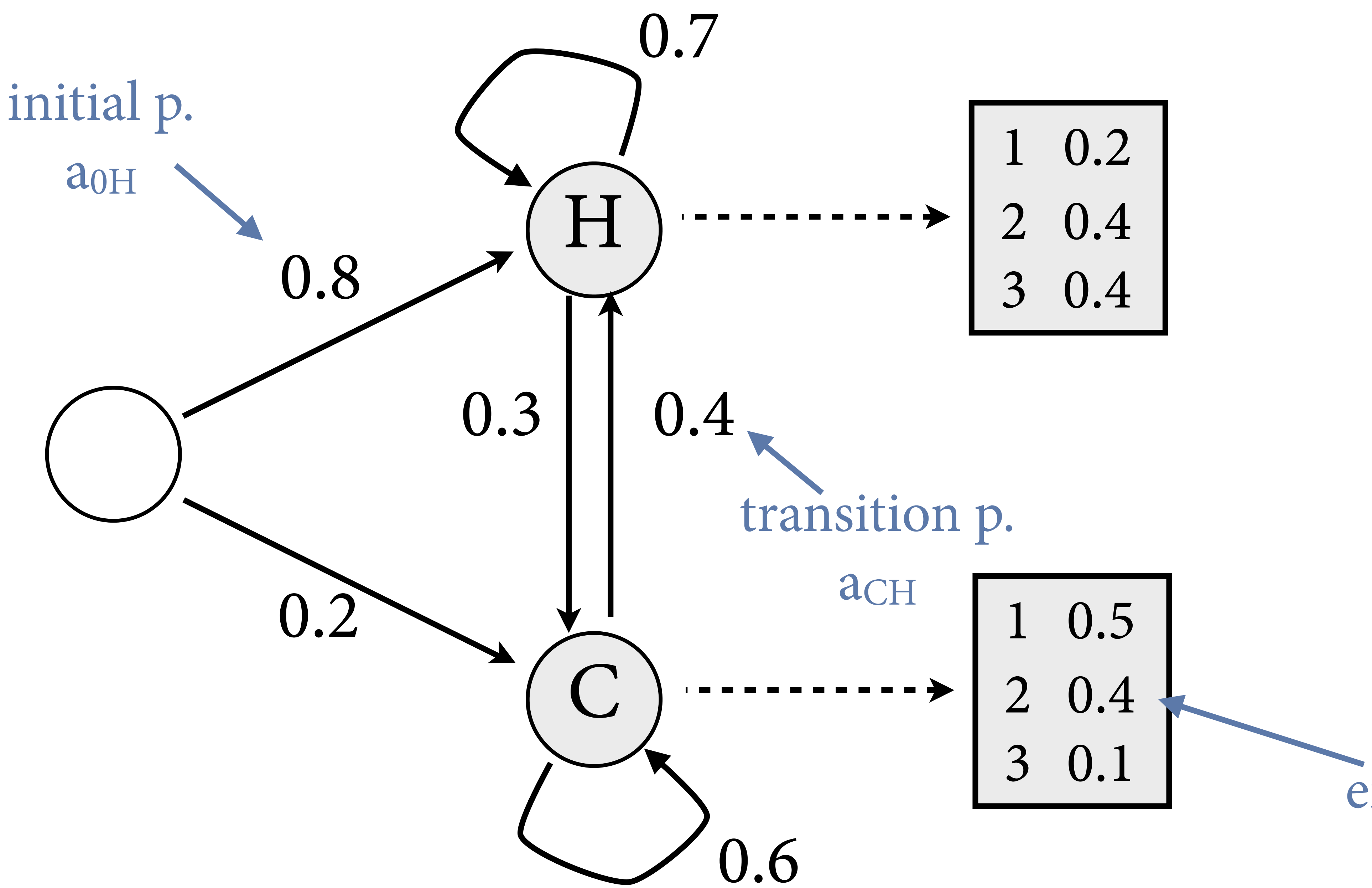
Evaluating and Training HMMs

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

Question 2: Tagging

- Given observations $y_1, \ldots, y_T$, what is the most probable sequence $x_1, \ldots, x_T$ of hidden states?

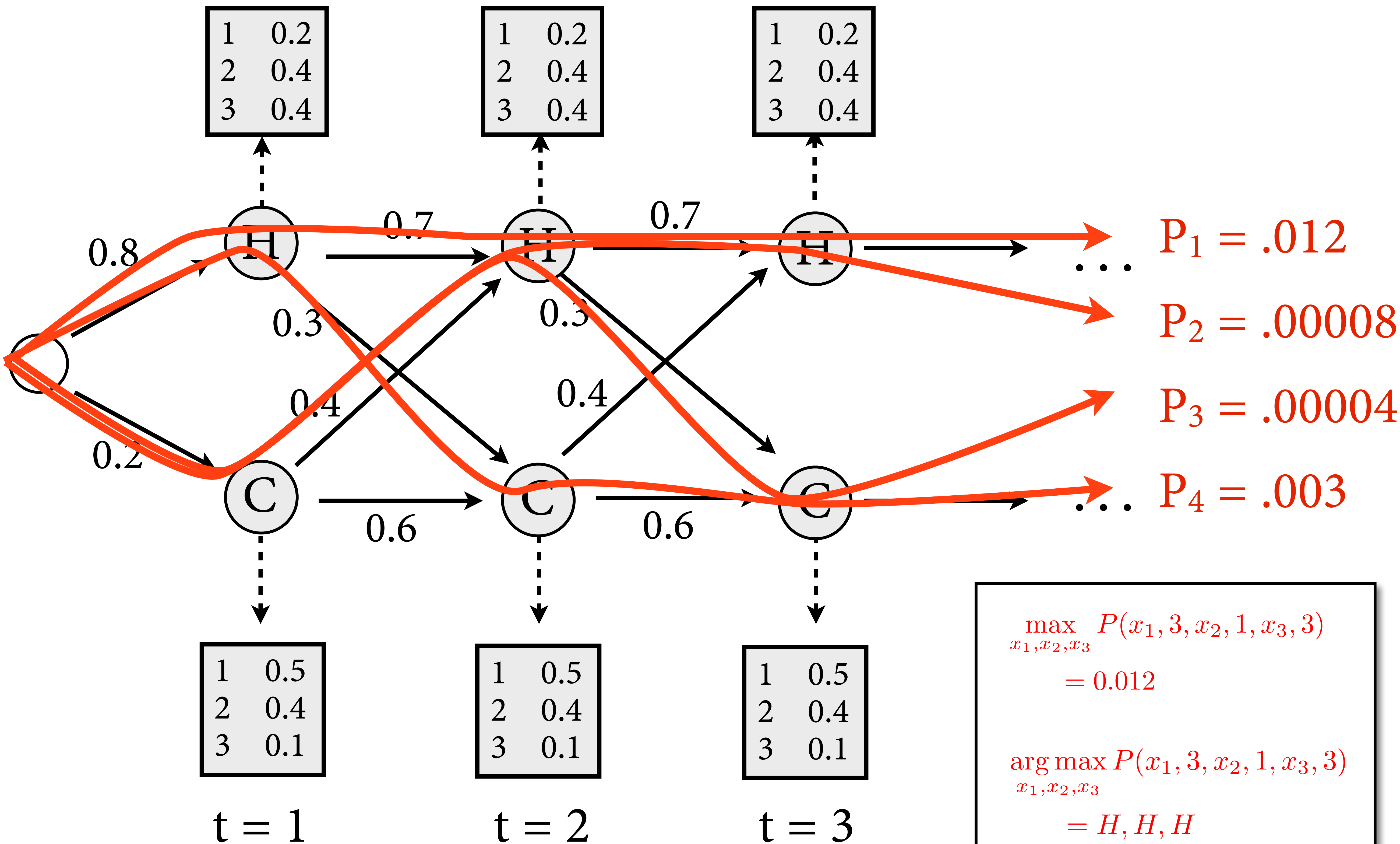
- Maximum probability:

$$\max_{x_1,\ldots,x_T} P(x_1,\ldots,x_T \mid y_1,\ldots,y_T)$$

- We are primarily interested in arg max:

$$\begin{aligned}
&\arg\max_{x_1,\ldots,x_T} P(x_1,\ldots,x_T \mid y_1,\ldots,y_T) \\
&= \arg\max_{x_1,\ldots,x_T} \frac{P(x_1,\ldots,x_T, y_1,\ldots,y_T)}{P(y_1,\ldots,y_T)} \\
&= \arg\max_{x_1,\ldots,x_T} P(x_1,\ldots,x_T, y_1,\ldots,y_T)
\end{aligned}$$

Naive solution



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

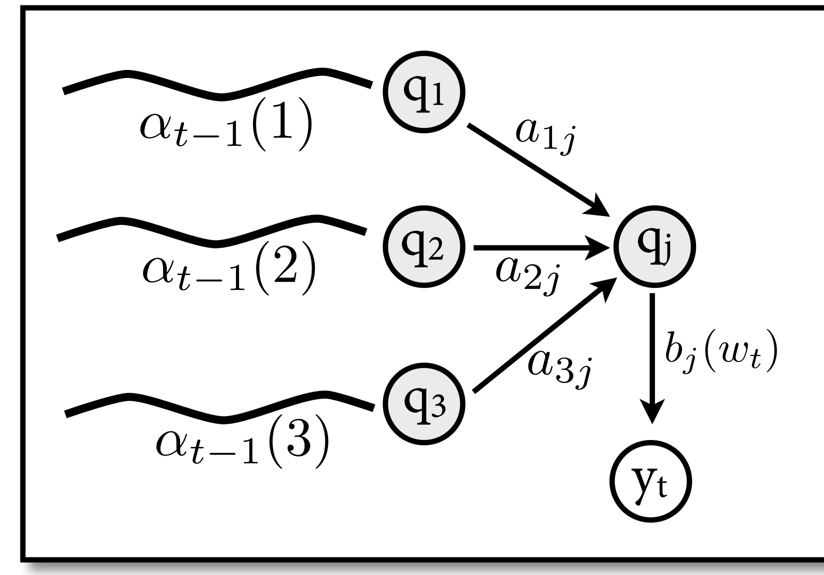
$$\arg\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = H, H, H$$

The Viterbi Algorithm

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

- Base case, t = 1:

$$V_1(j) = b_j(y_1) \cdot a_{0j}$$



- Inductive case, for t = 2, …, T:

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

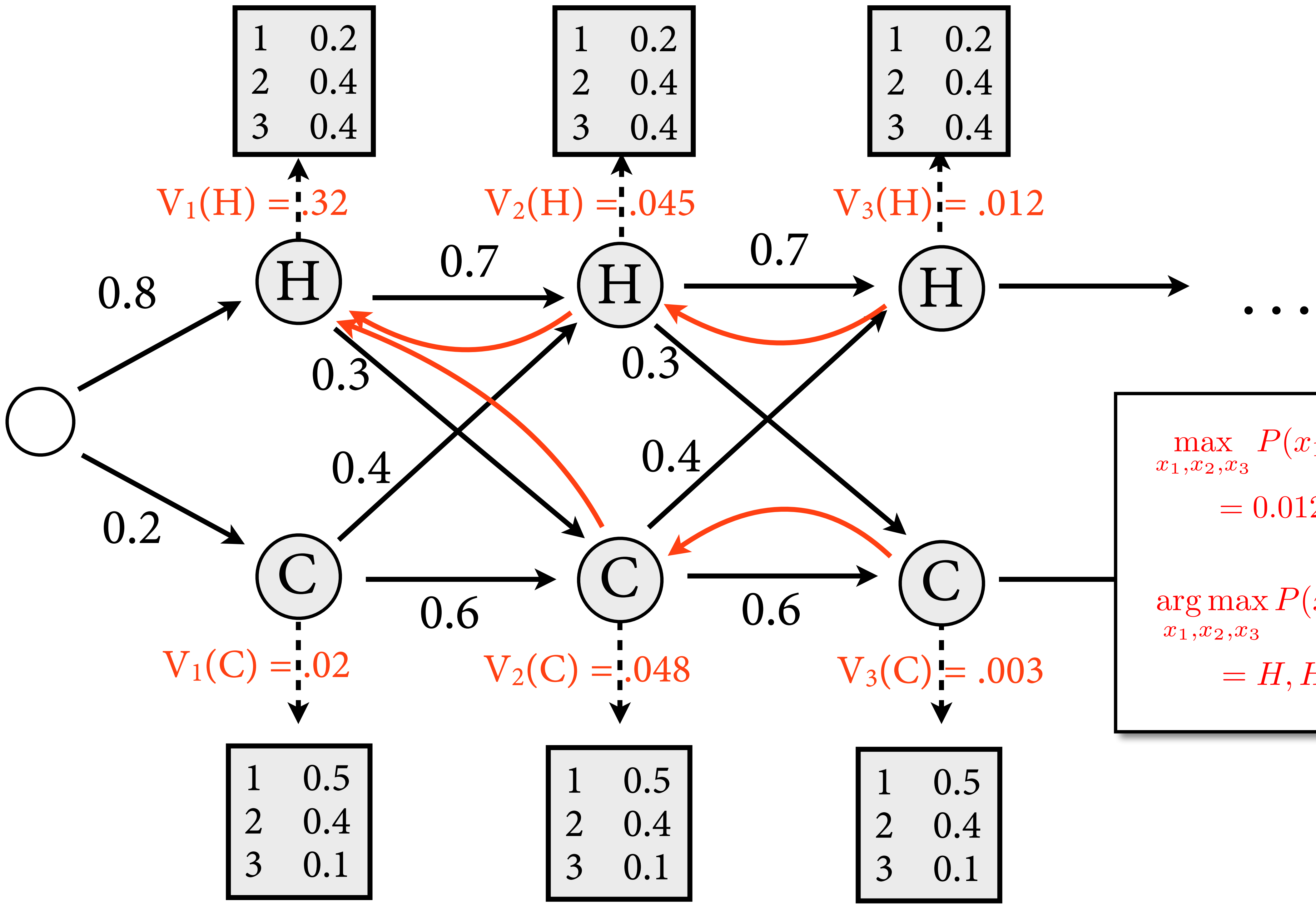
Backpointers

- Once $V_t(j)$ has been computed for all t and j, we need to reconstruct the state sequence y for with the maximum prob P(y | x).
- For each (t,j), remember the i for which maximum was achieved in *backpointer* $bp_t(j)$.
- Use bp to compute best state seq from right to left:

$$x_T = \arg\max_q V_T(q)$$

$$x_t = \mathsf{bp}_{t+1}(x_{t+1}) \qquad \text{for t = T-1, …, 1}$$

Viterbi Algorithm: Example



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

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

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

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

Runtime

- Forward and Viterbi have the same runtime, dominated by inductive step:

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

- Compute $N \cdot T$ values for $V_t(j)$. Each computation step requires iteration over N predecessor states.

- Total runtime is $O(N^2 \cdot T)$, i.e.
  - linear in sentence length
  - quadratic in size of tag set

Implementation notes

- Computing $V_t(j)$ requires multiplying many small numbers with each other.
  - Very serious risk of rounding errors.

- Solution: Compute log $V_t(j)$ instead of $V_t(j)$.
  - Replace multiplication by addition:

$$\log(V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)) = \log V_{t-1}(i) + \log a_{ij} + \log b_j(y_t)$$

  - Exploit that log is a monotonic function:

$$\max_i \log V_t(i) = \log \max_i V_t(i)$$

Question 3a: 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:

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

- Obvious ML estimate for emission probabilities:

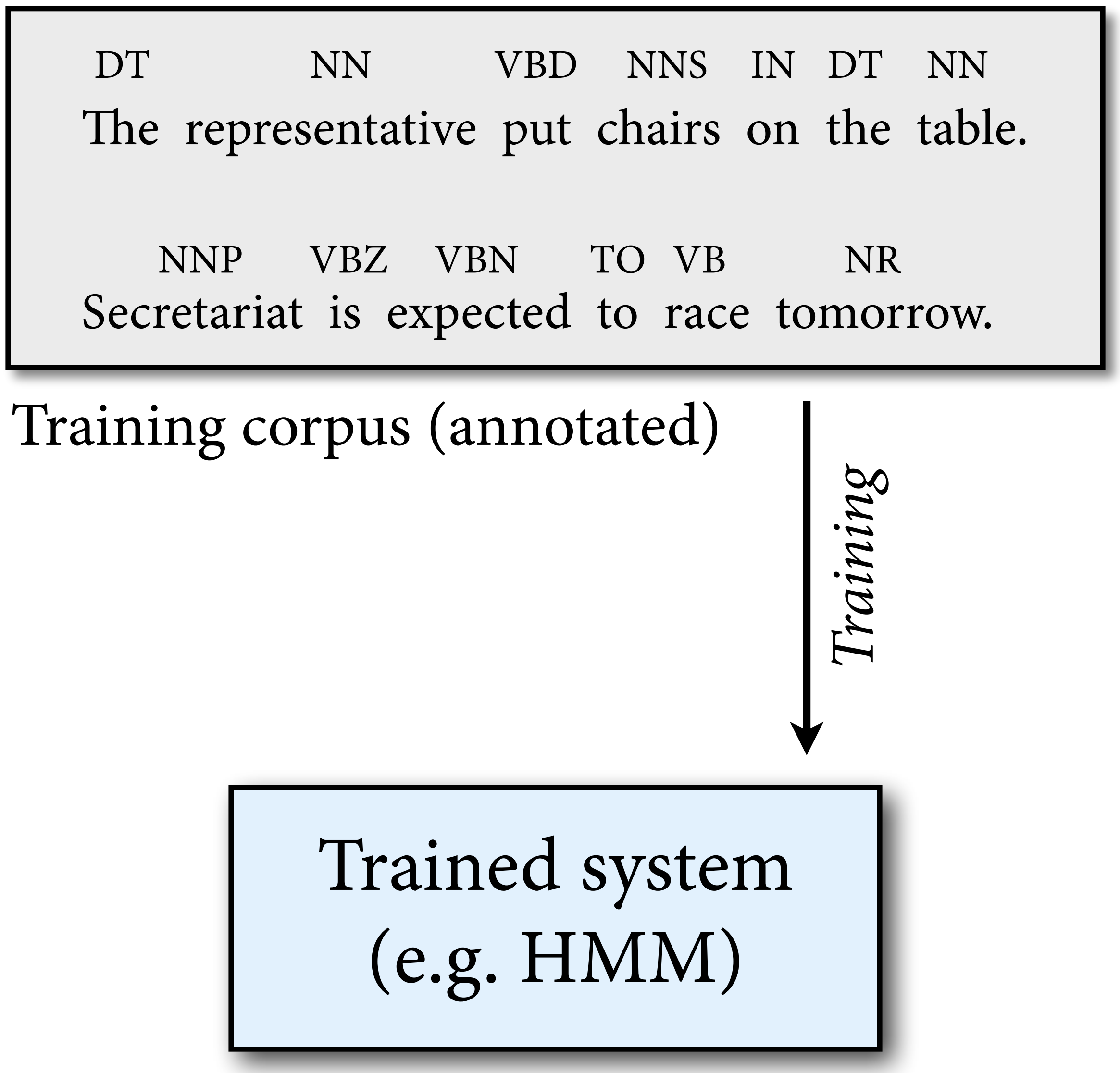
$$b_i(o) = \frac{C(X_t = q_i, Y_t = o)}{C(X_t = q_i)}$$

- Apply smoothing as you would for ordinary n-gram models.

Evaluation

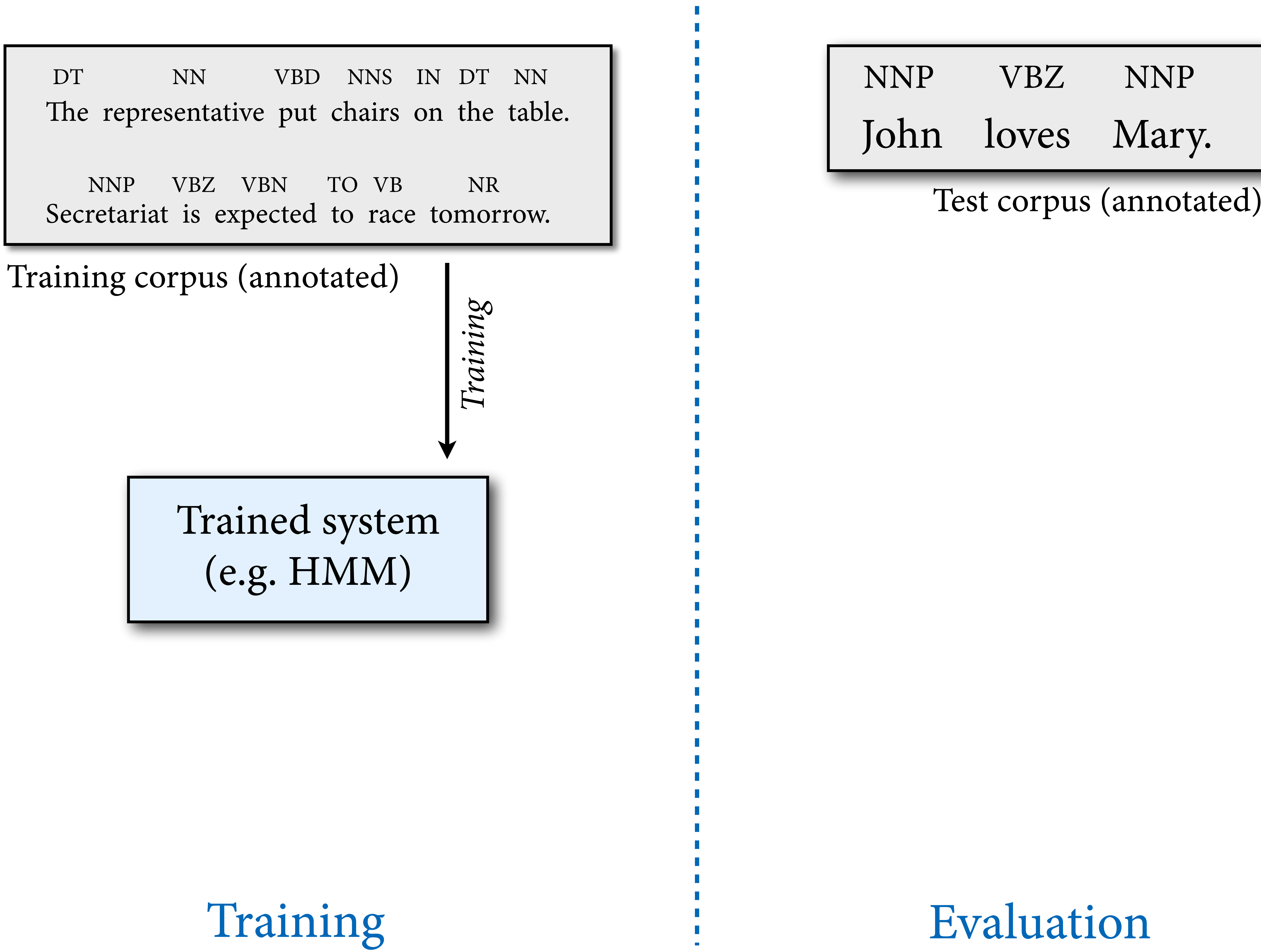
- How do you know how well your tagger works?
- Run it on *test data* and evaluate *accuracy*.
  - Test data: Really important to evaluate on unseen sentences to get a fair picture of how well tagger generalizes.
  - Accuracy: Measure percentage of correctly predicted POS tags.

Evaluation on test data

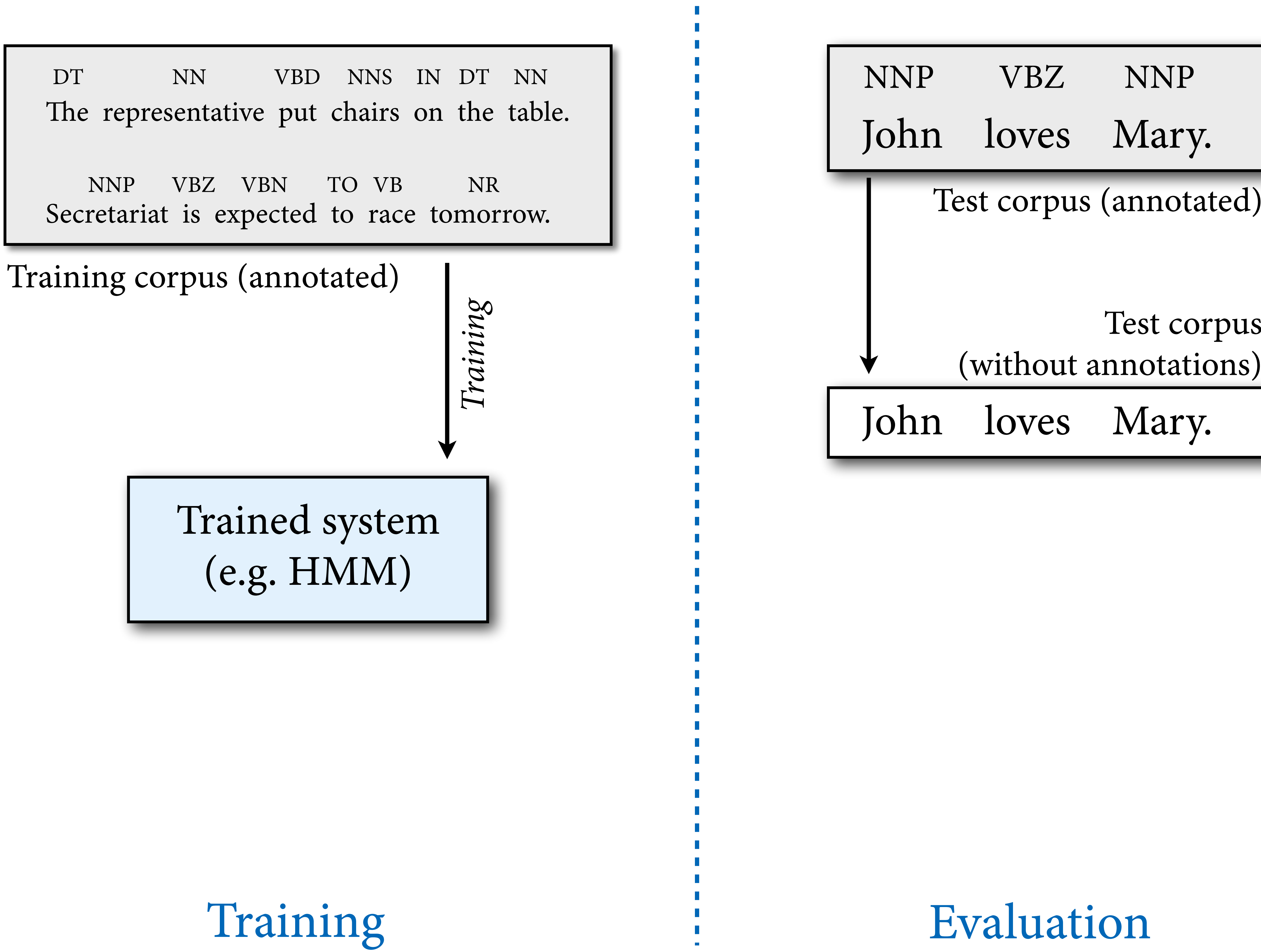


Training

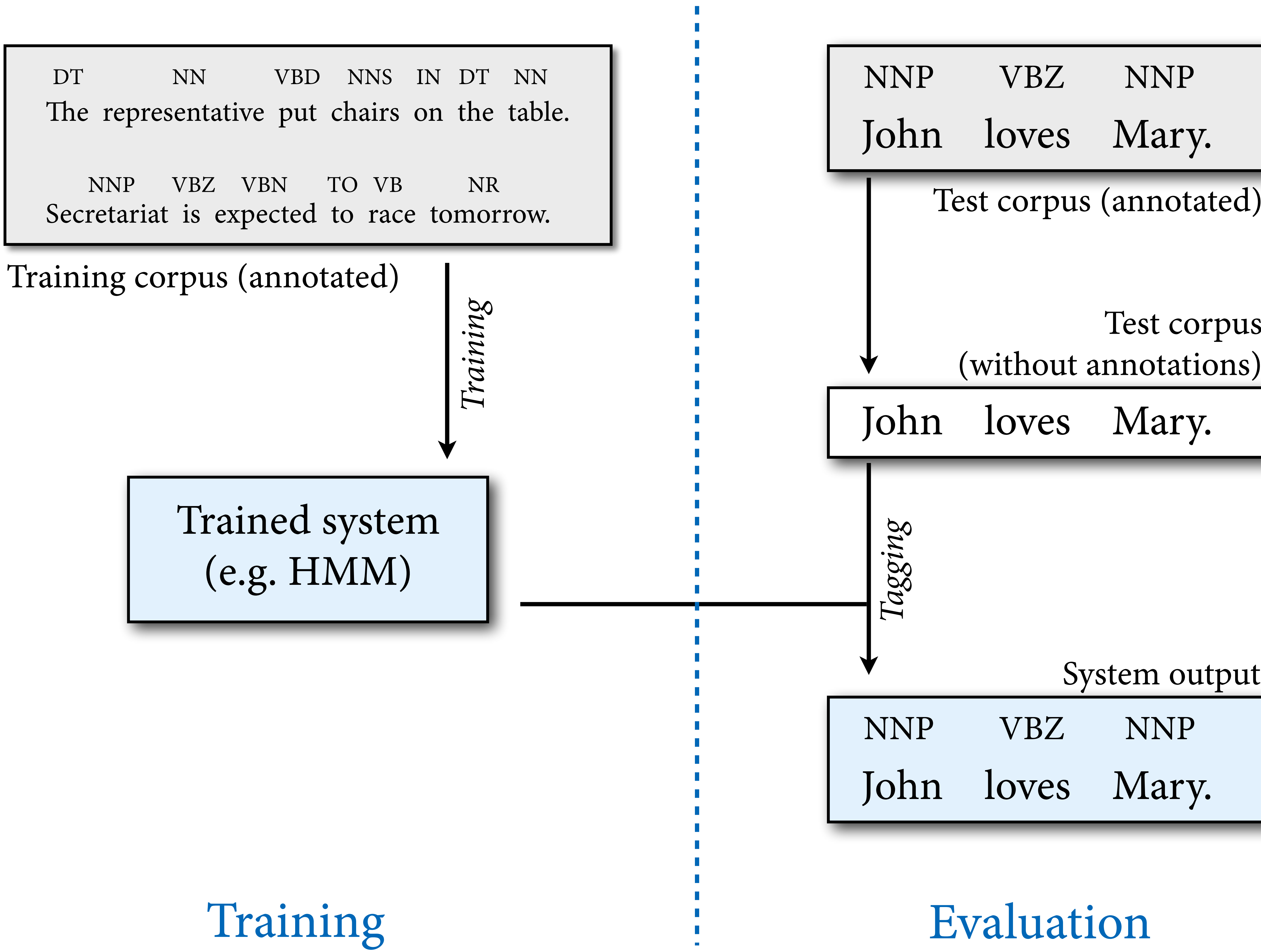
Evaluation on test data



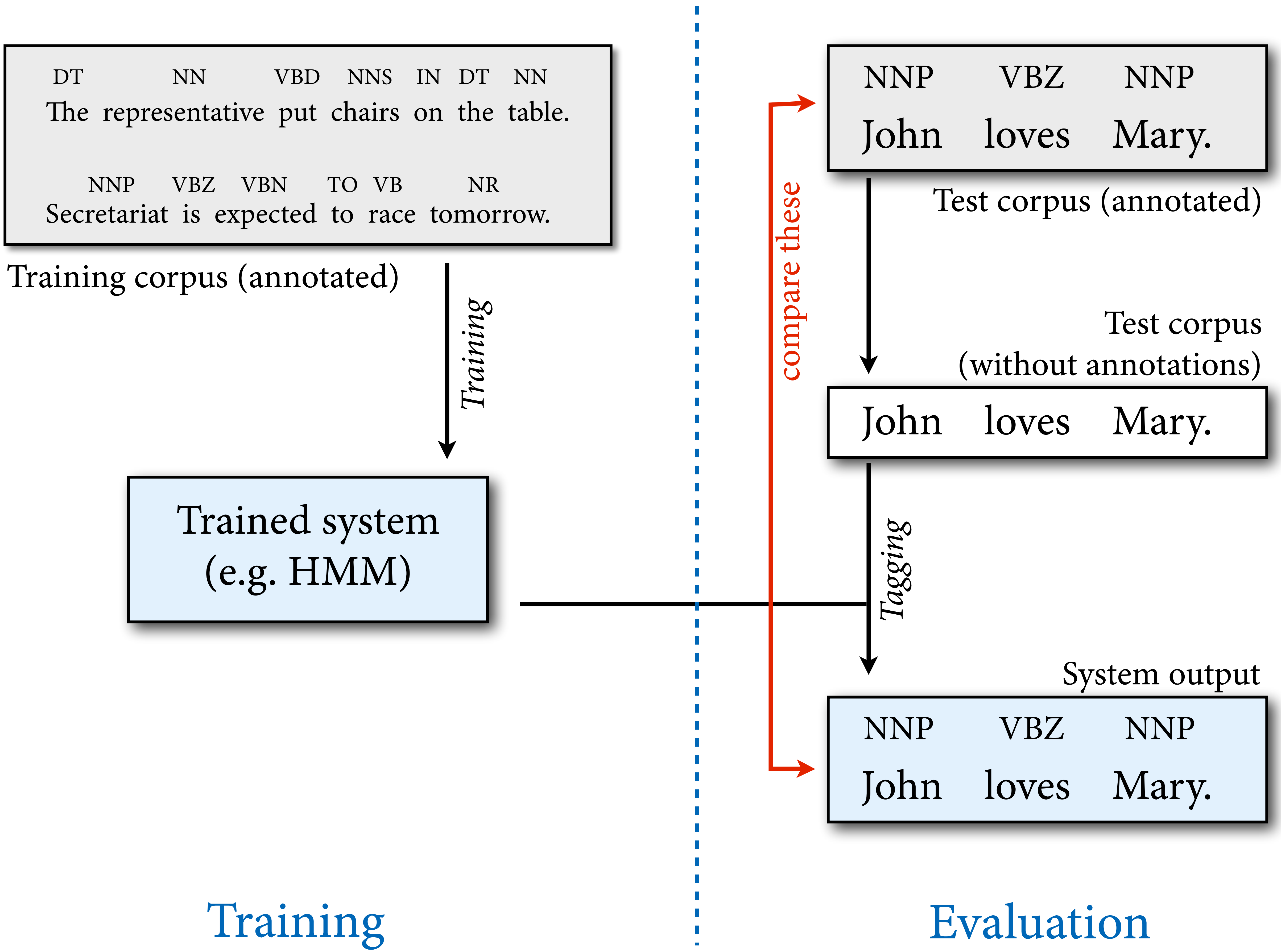
Evaluation on test data



Evaluation on test data



Evaluation on test data



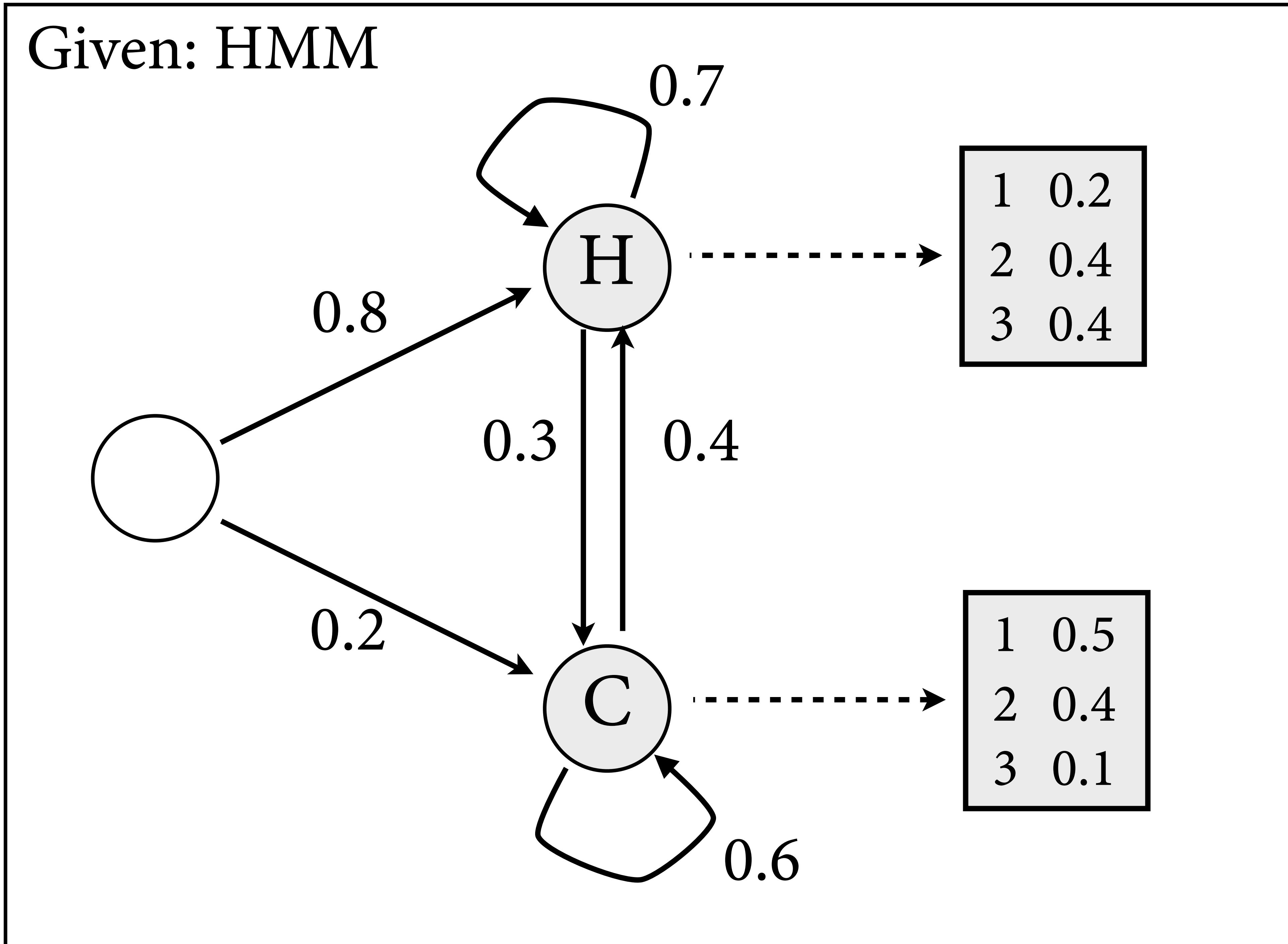
Question 3b: Learning

- 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.

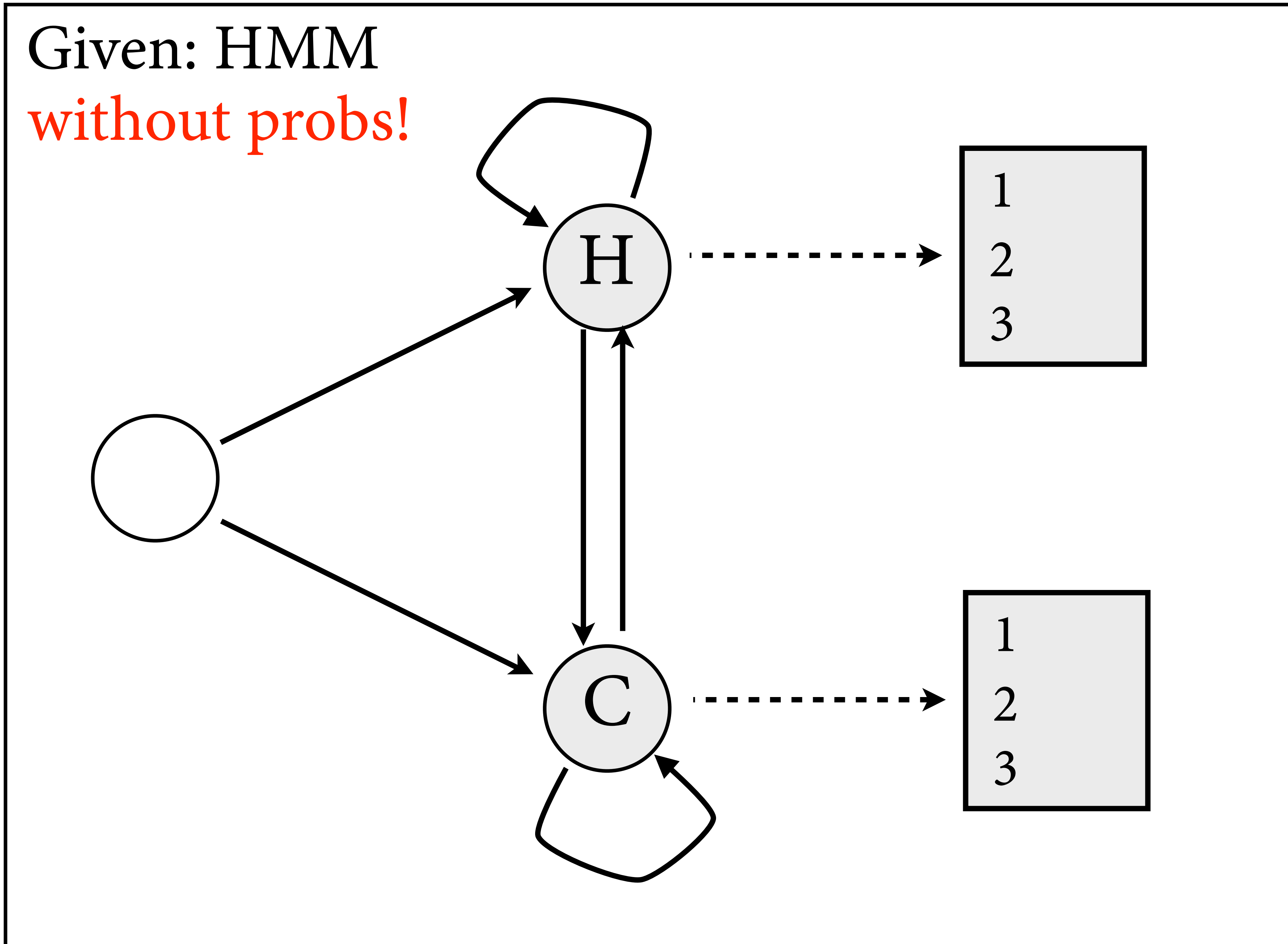
- Useful because annotated data is expensive to obtain, but raw text is really cheap.

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

The setup

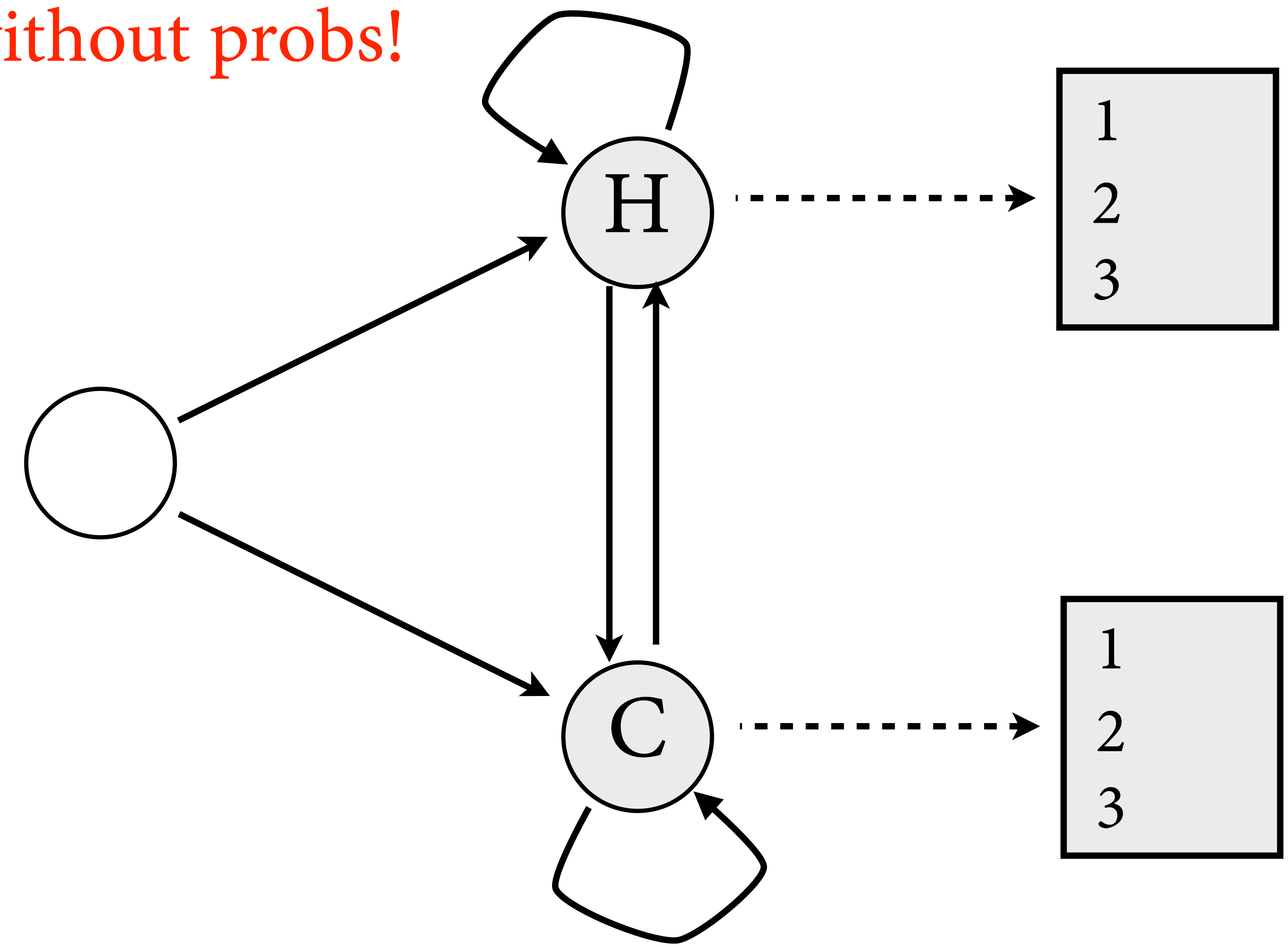


The setup



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.

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

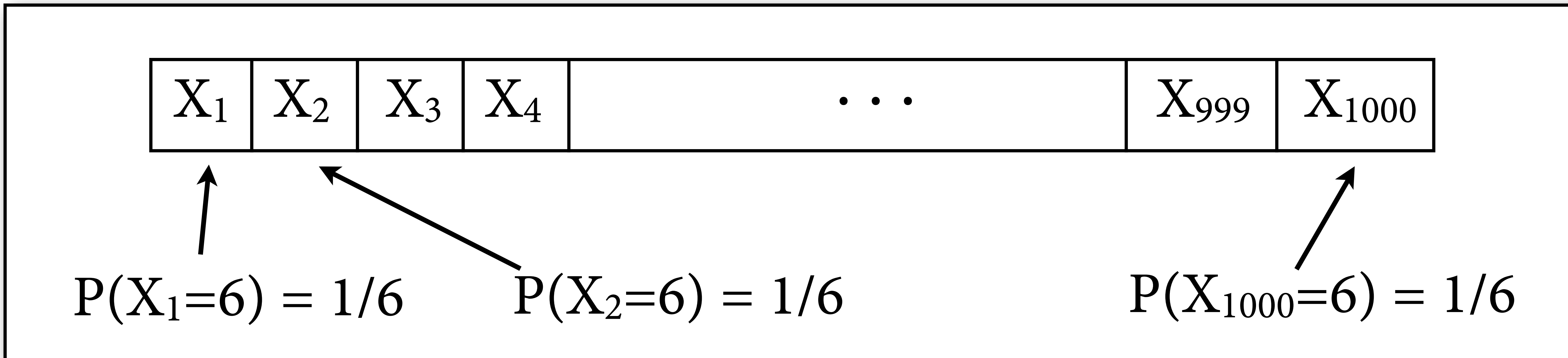
- Idea: replace actual counts by *estimated* counts.

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

- How can we estimate counts?

Expectations

- If I roll a die 1000 times, how many times do I expect to see a 6?



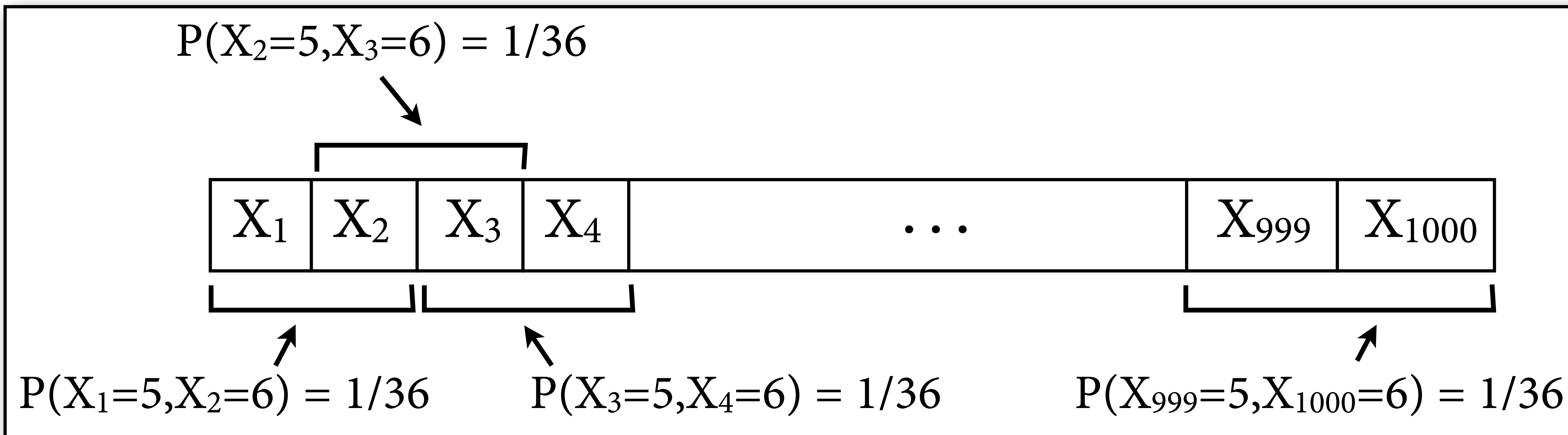
$$E(\delta_6) = \sum_{t=1}^{1000} \sum_{a=1}^{6} P(X_t = a) \cdot \delta_6(a)$$

$$= \sum_{t=1}^{1000} P(X_t = 6) \approx 167$$

Kronecker delta:
$\delta_a(b) = 1$ if $a = b$,
$\delta_a(b) = 0$ if $a \neq b$

Expectations

- If I roll a die 1000 times, how many times do I expect to see a 5 followed by a 6?



$$E(\delta_{5\to6}) = \sum_{t=1}^{999}\sum_{a=1}^{6}\sum_{b=1}^{6} P(X_t = a, X_{t+1} = b)\cdot\delta_{5\to6}(a \to b)$$

$$= \sum_{t=1}^{999} P(X_t = 5, X_{t+1} = 6) \approx 28$$

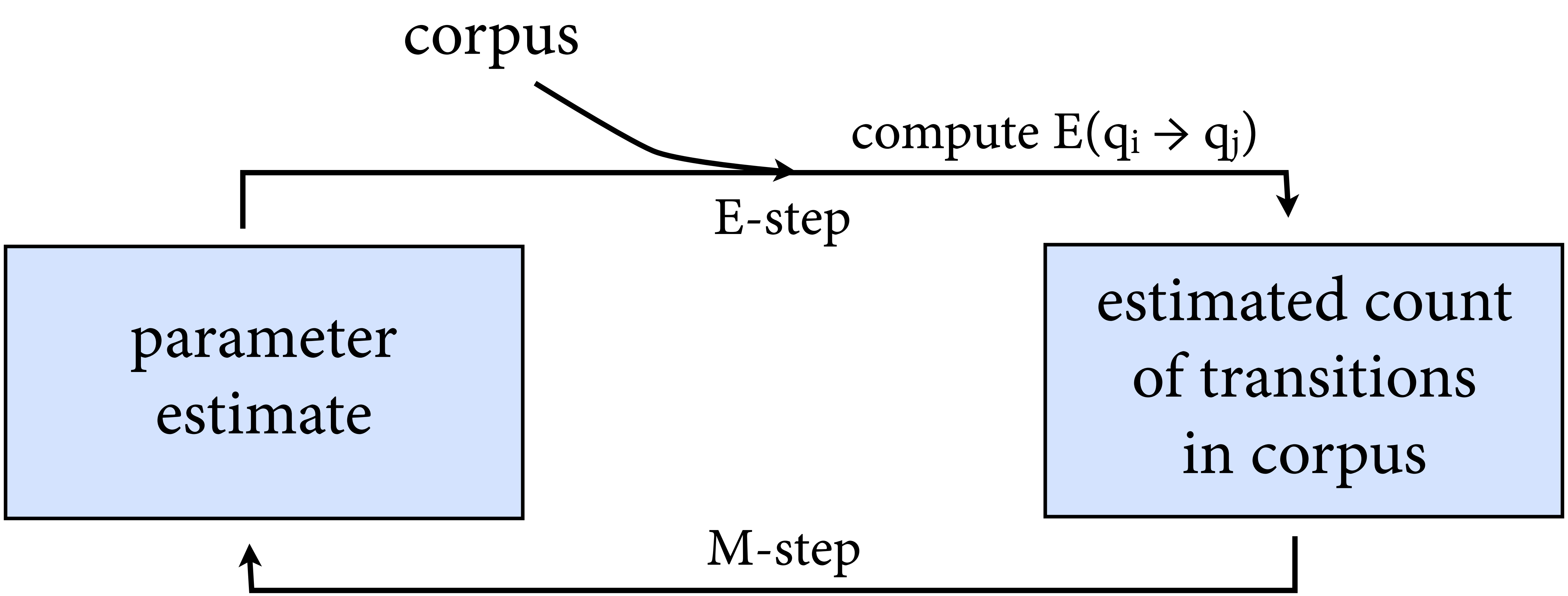
Expectations as fake counts

- Suppose we have
  - estimate of parameters; say of transition probs
  - observations y of length M
- How many times do we expect transition $q_i \rightarrow q_j$?

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

- Use expected value $E(q_i \rightarrow q_j)$ with respect to old parameter values as approximation of true counts $C(q_i \rightarrow q_j)$.

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

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

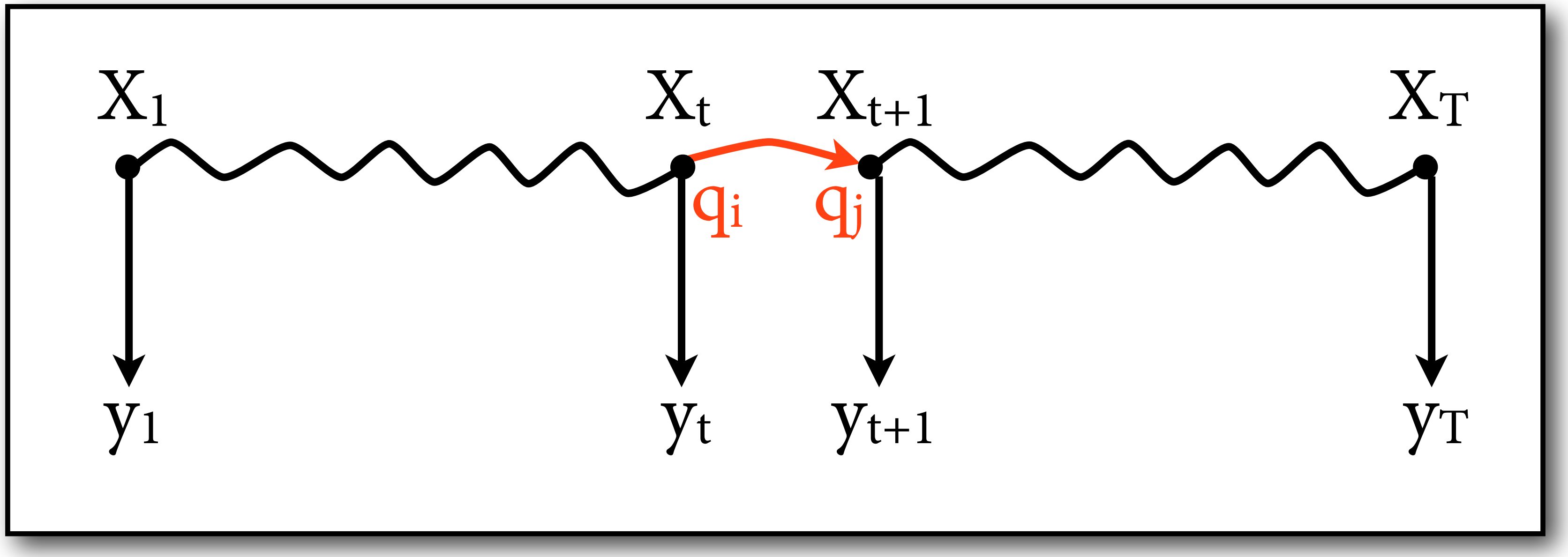
- How can we compute $\hat{P}$ efficiently?
  Challenge: It is conditioned on y.

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

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

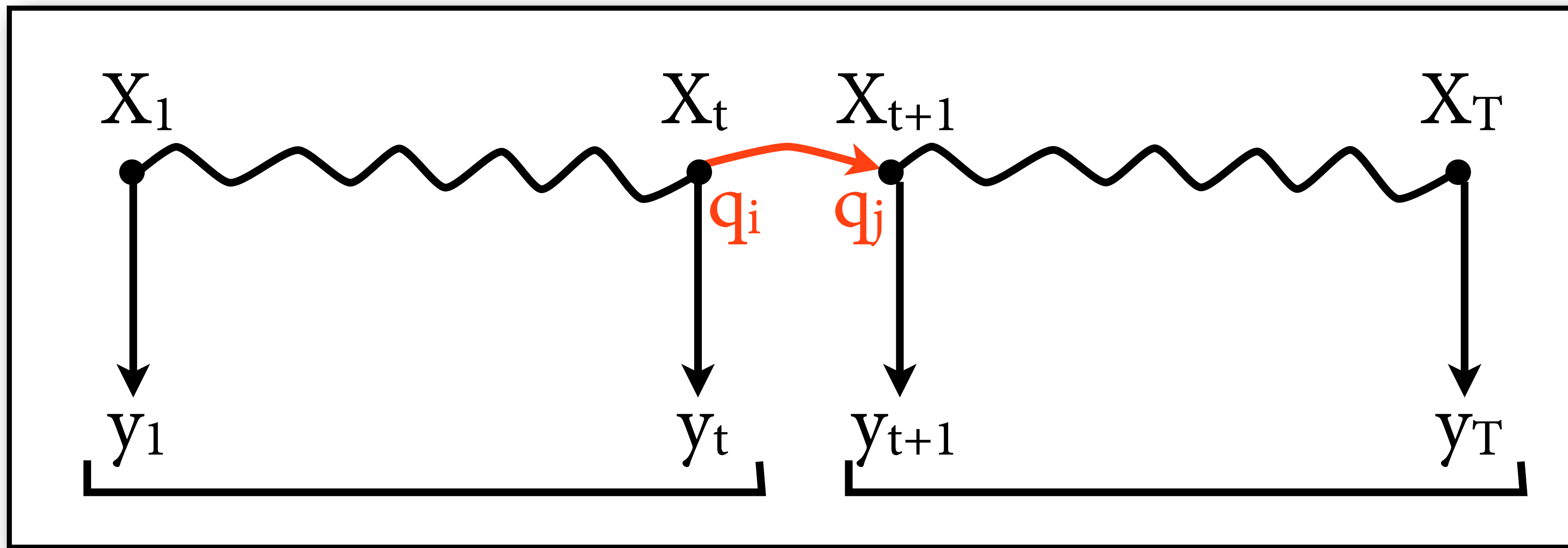
- Do it in two steps:
  - compute $\xi'_t(i,j) = \hat{P}(X_t = q_i, X_{t+1} = q_j, y)$
  - compute P(y)

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



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

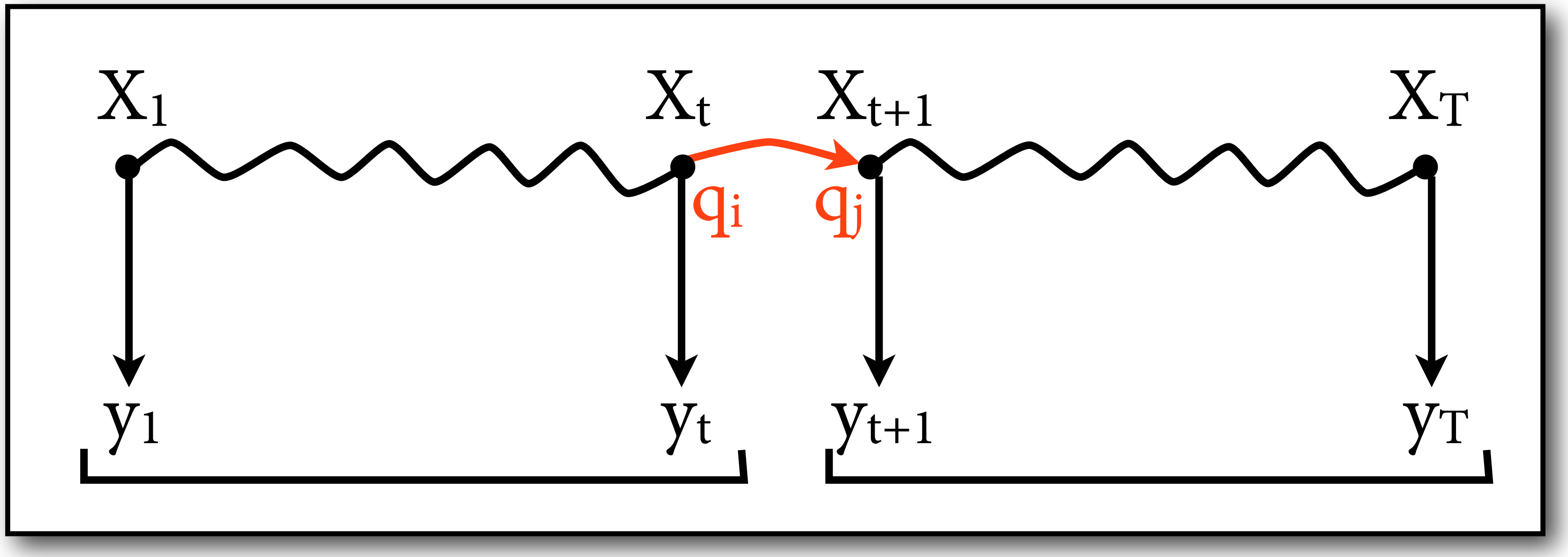
$$\xi'_t(i,j) = \hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$



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

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

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

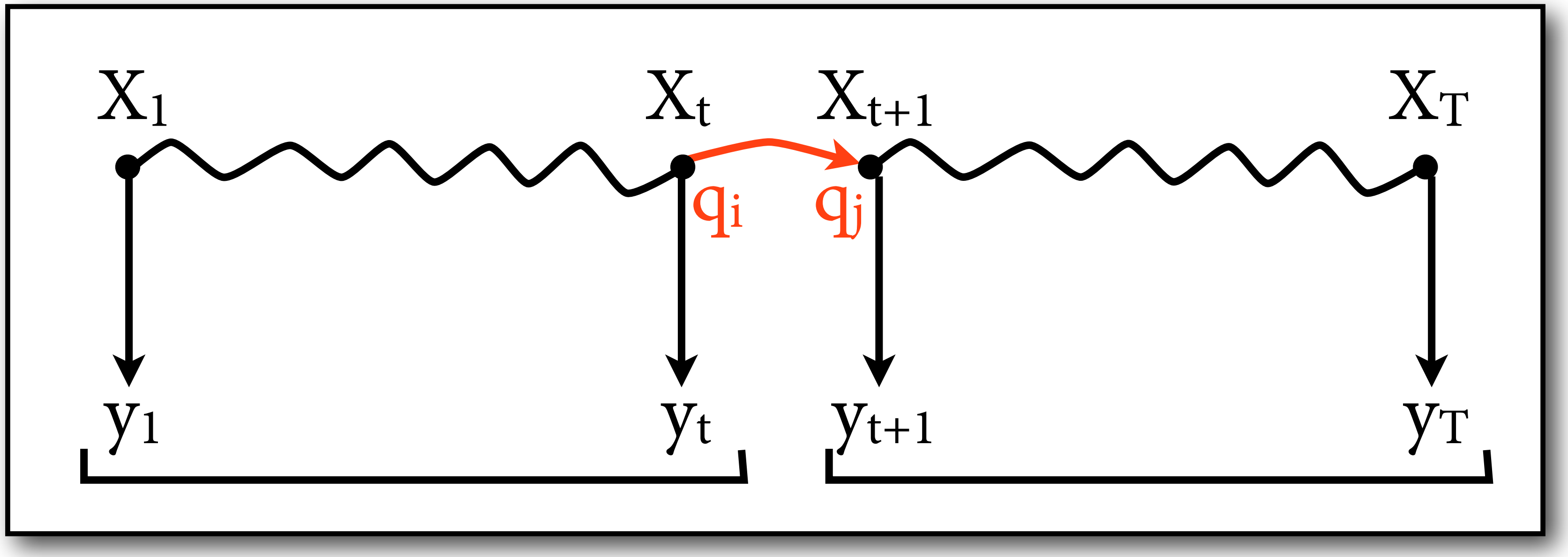


$$\hat{P}(X_t = q_i, X_{t+1} = q_j, y)$$

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

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

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



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

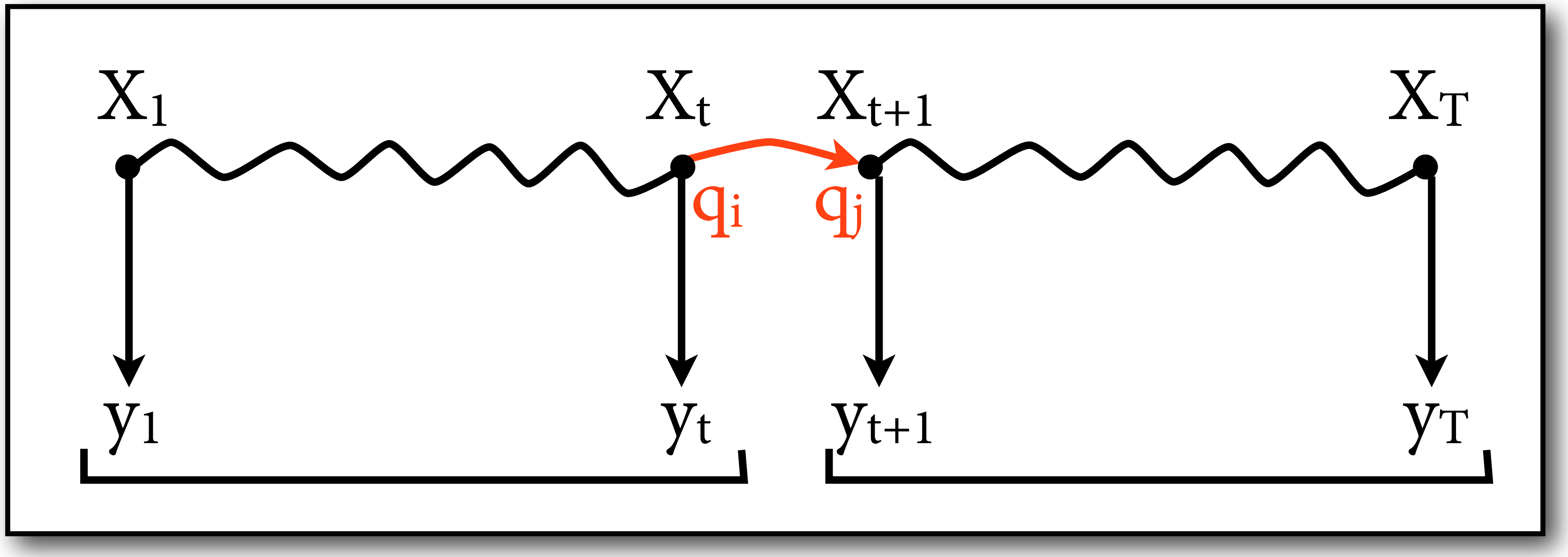
$$= \hat{P}(y_1, \ldots, y_t, X_t = q_i) \cdot \hat{P}(y_{t+1}, X_{t+1} = q_j \mid X_t = q_i) \cdot \hat{P}(y_{t+2}, \ldots, y_T \mid X_{t+1} = q_j)$$

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

forward prob (last time):

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

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



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

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

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

forward prob (last time):

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

backward prob (today):

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

Backward probabilities

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

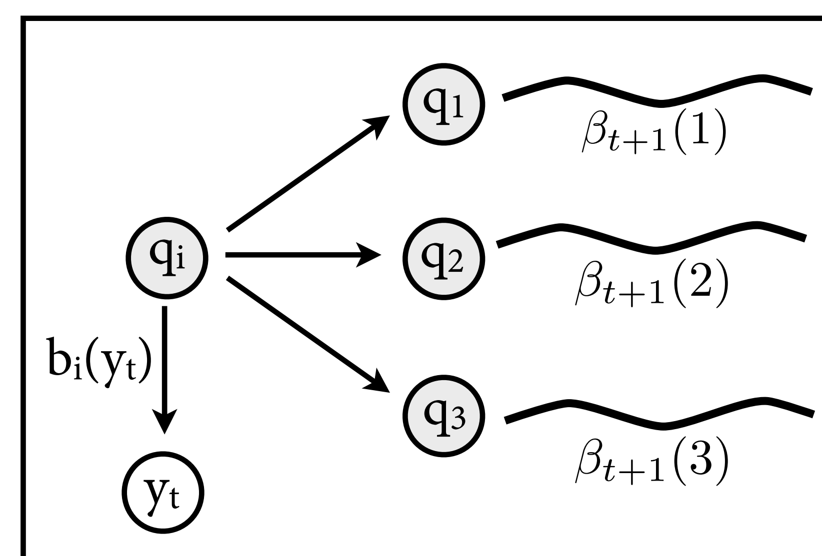
- Base case, t = T:

  $\beta_T(i) = 1$ 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.

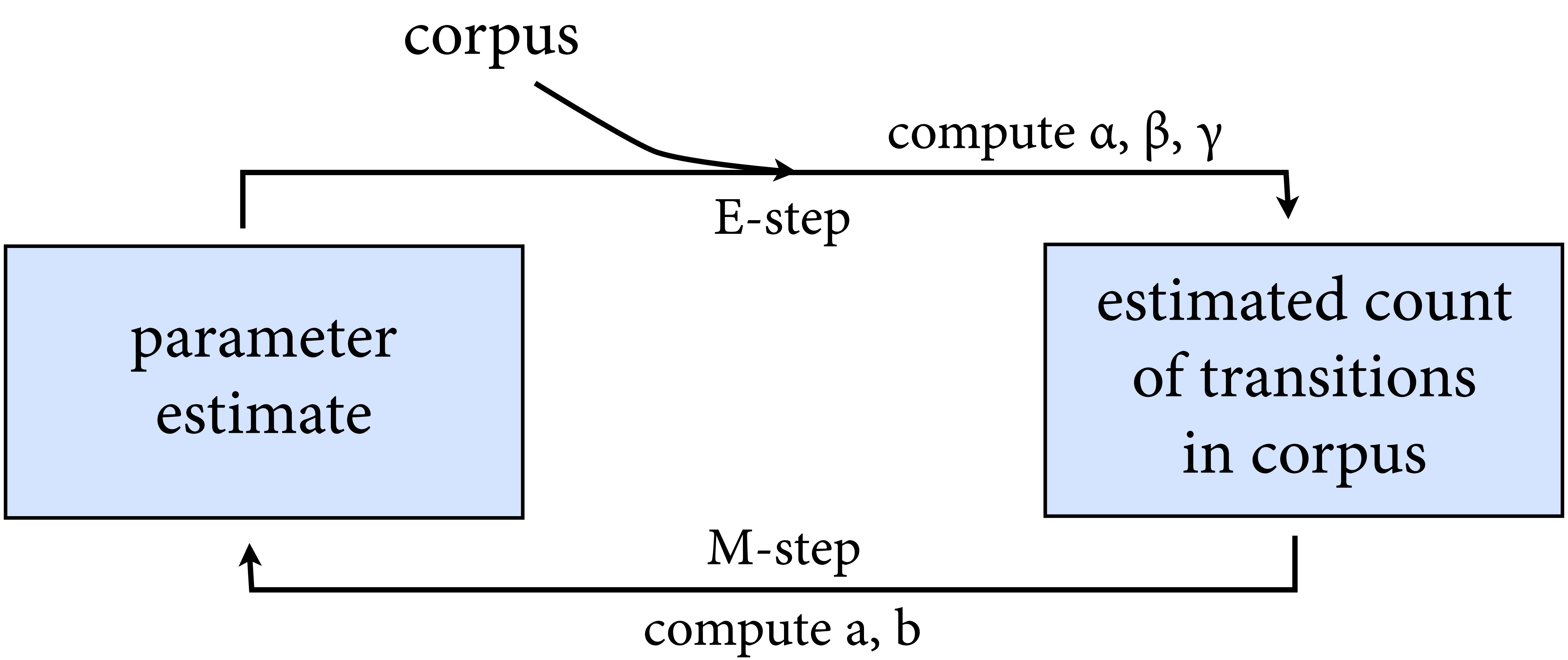
- Here's what it looks like for emission prob:

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

$$b_j(o) \approx \left(\underbrace{\sum_{\substack{t=1 \\ y_t=o}}^{T} \gamma_t(j)}_{\text{estimated count of o emitted in state } q_j} \right) \Big/ \underbrace{\sum_{t=1}^{T} \gamma_t(j)}_{\text{estimated count of state } q_j}$$

Forward-Backward Algorithm

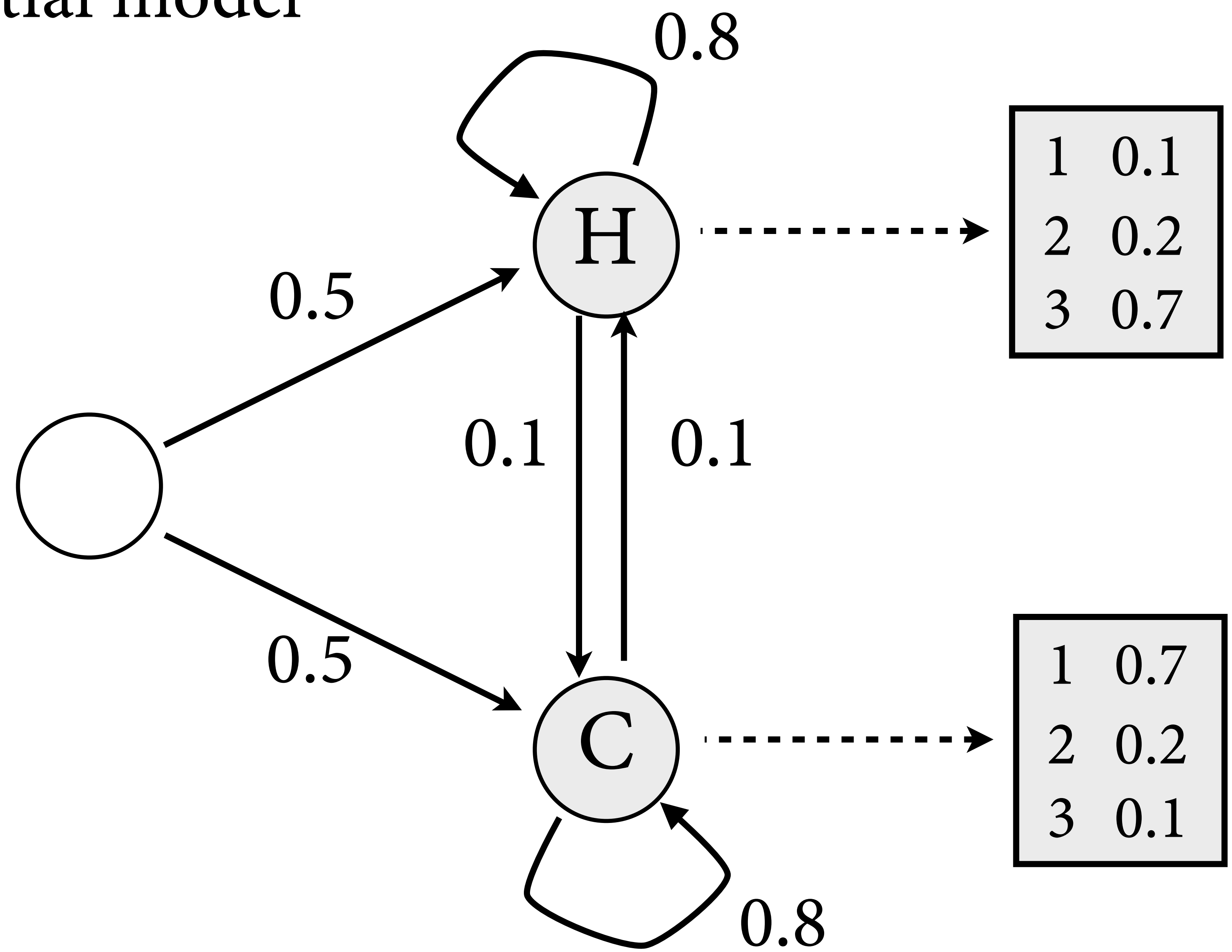
Initialization: start with some estimation of parameters.



Continue computation until parameters don't change much.

Example

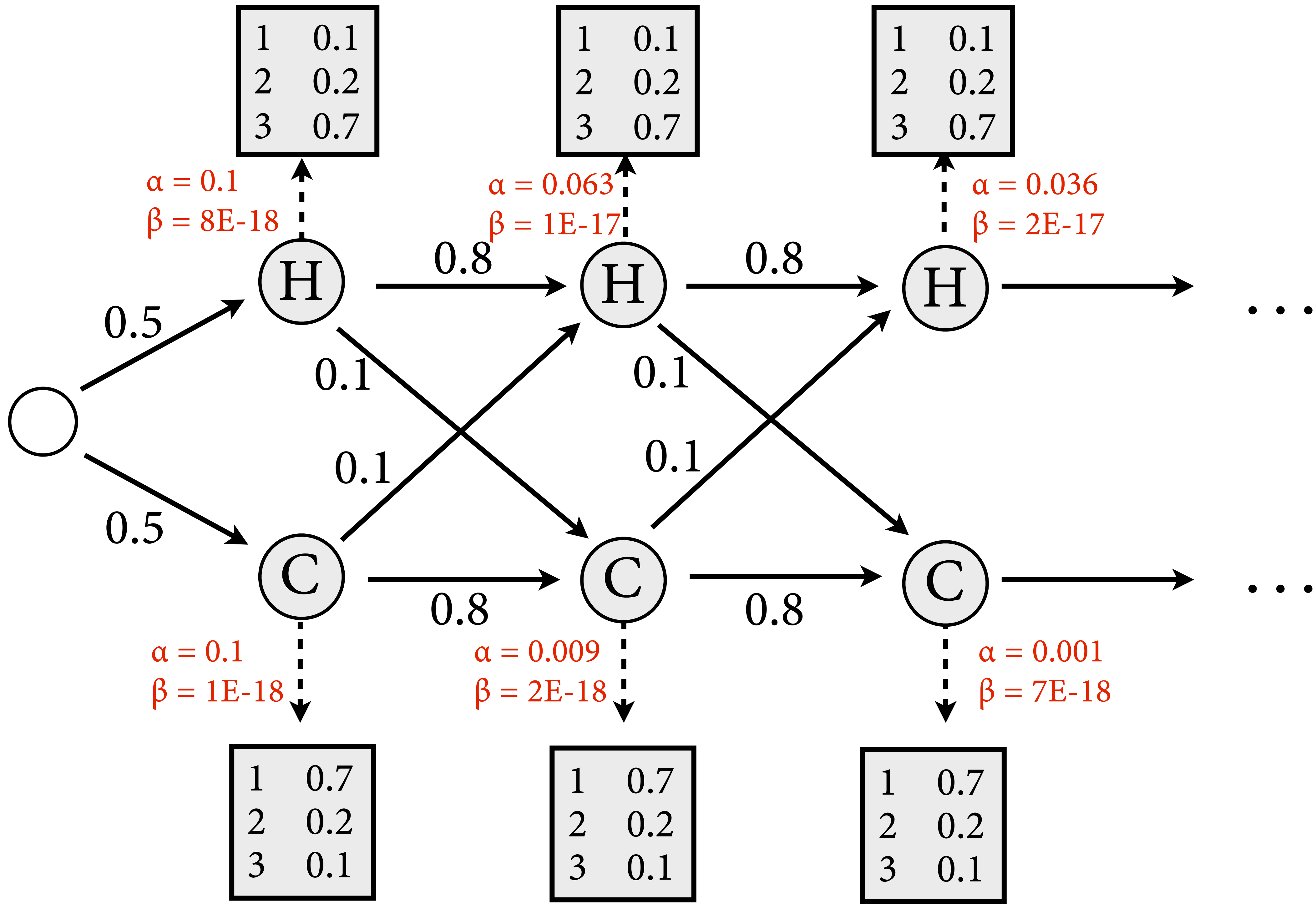
Step 1: initial model



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

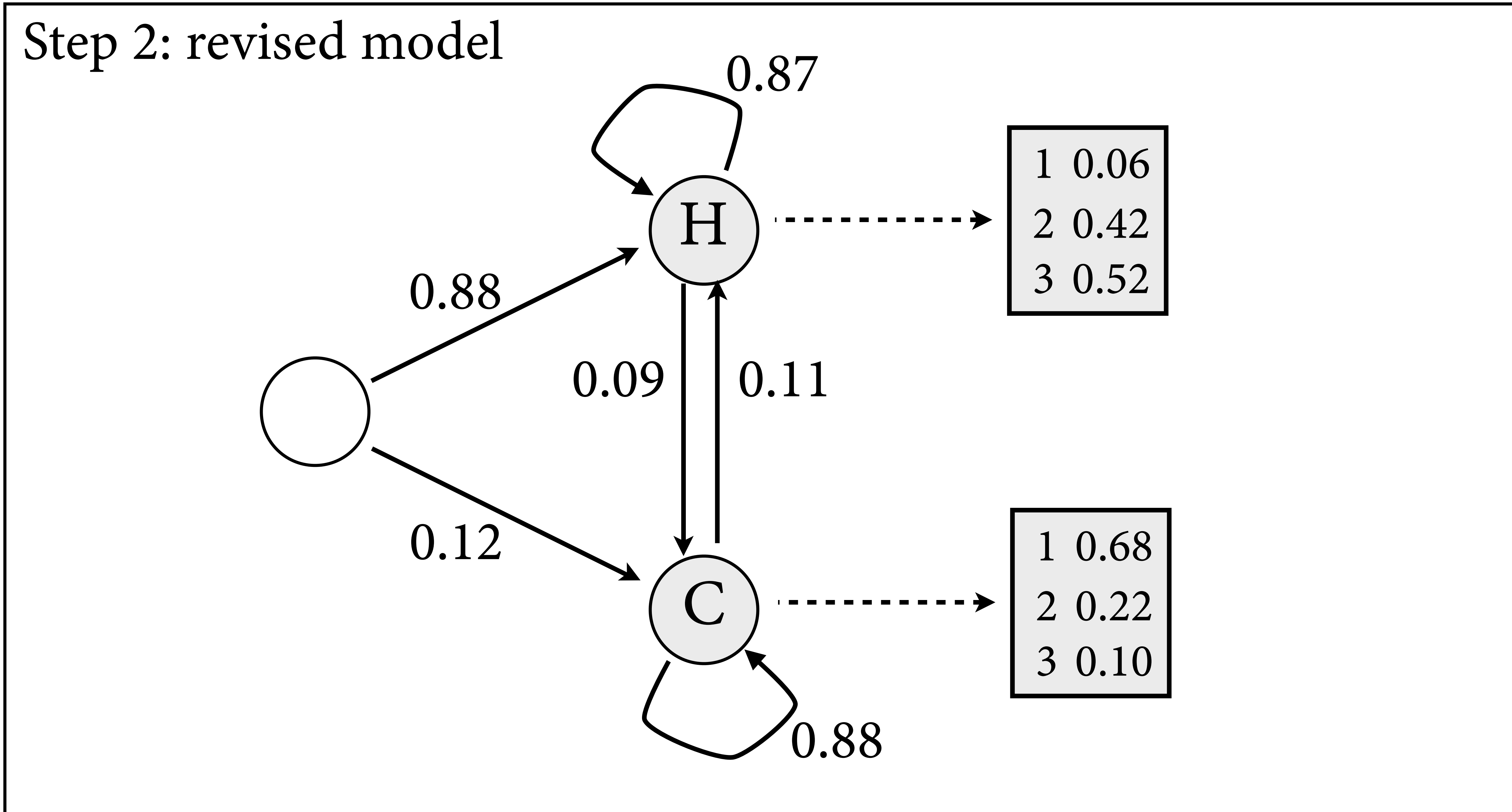
2, 3, 3, 2, 3, 2, 3, 2, 2, 3, 1, 3, 3, 1, 1, 1, 2, 1, 1, 1, 3, 1, 2, 1, 1, 1, 2, 3, 3, 2, 3, 2, 2

E-Step



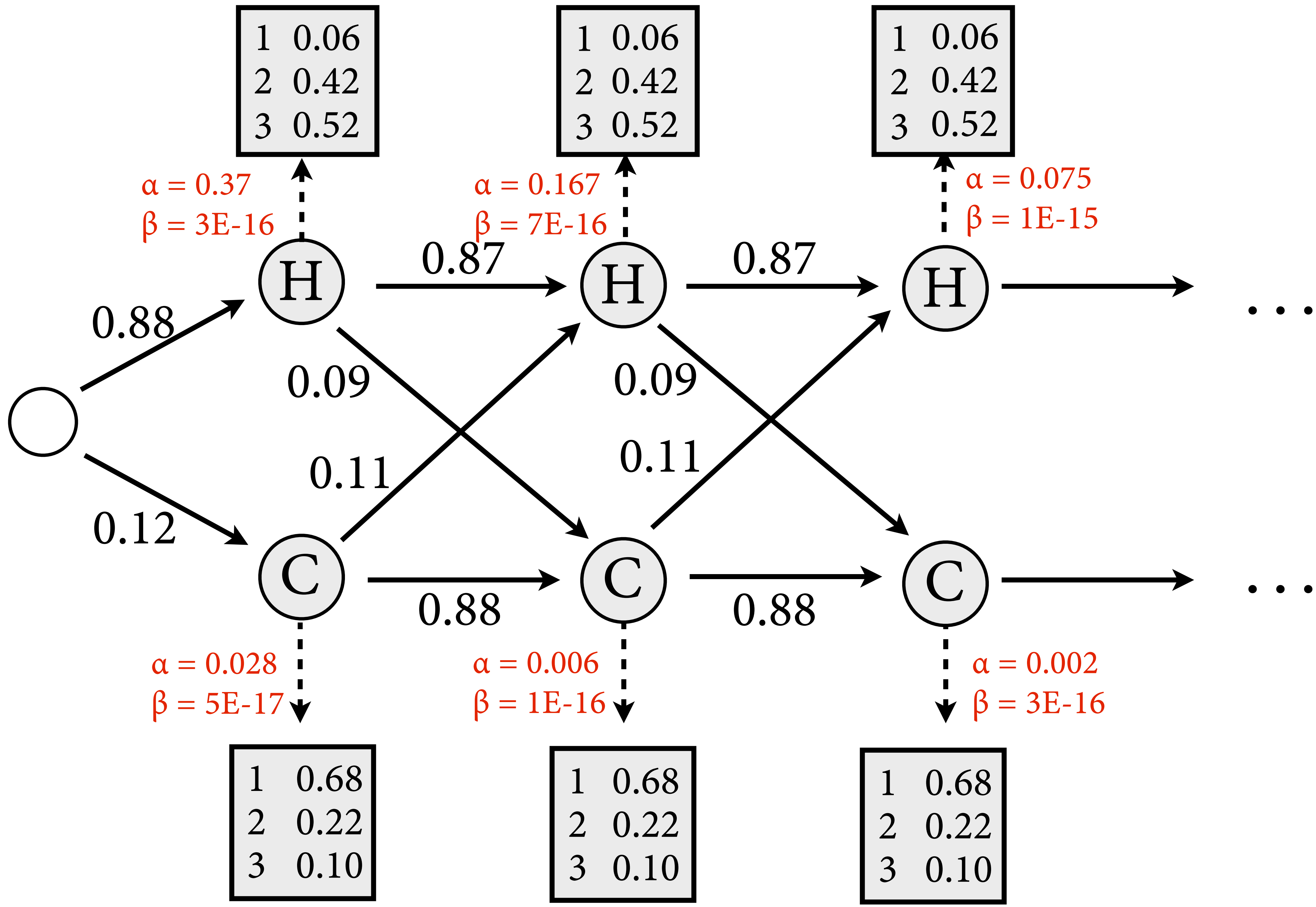
1 0.1
2 0.2
3 0.7
α = 0.1
β = 8E-18
α = 0.063
β = 1E-17
α = 0.036
β = 2E-17
0.5
0.5
H
C
0.8
0.1
0.1
0.8
α = 0.1
β = 1E-18
α = 0.009
β = 2E-18
α = 0.001
β = 7E-18
1 0.7
2 0.2
3 0.1

M-Step



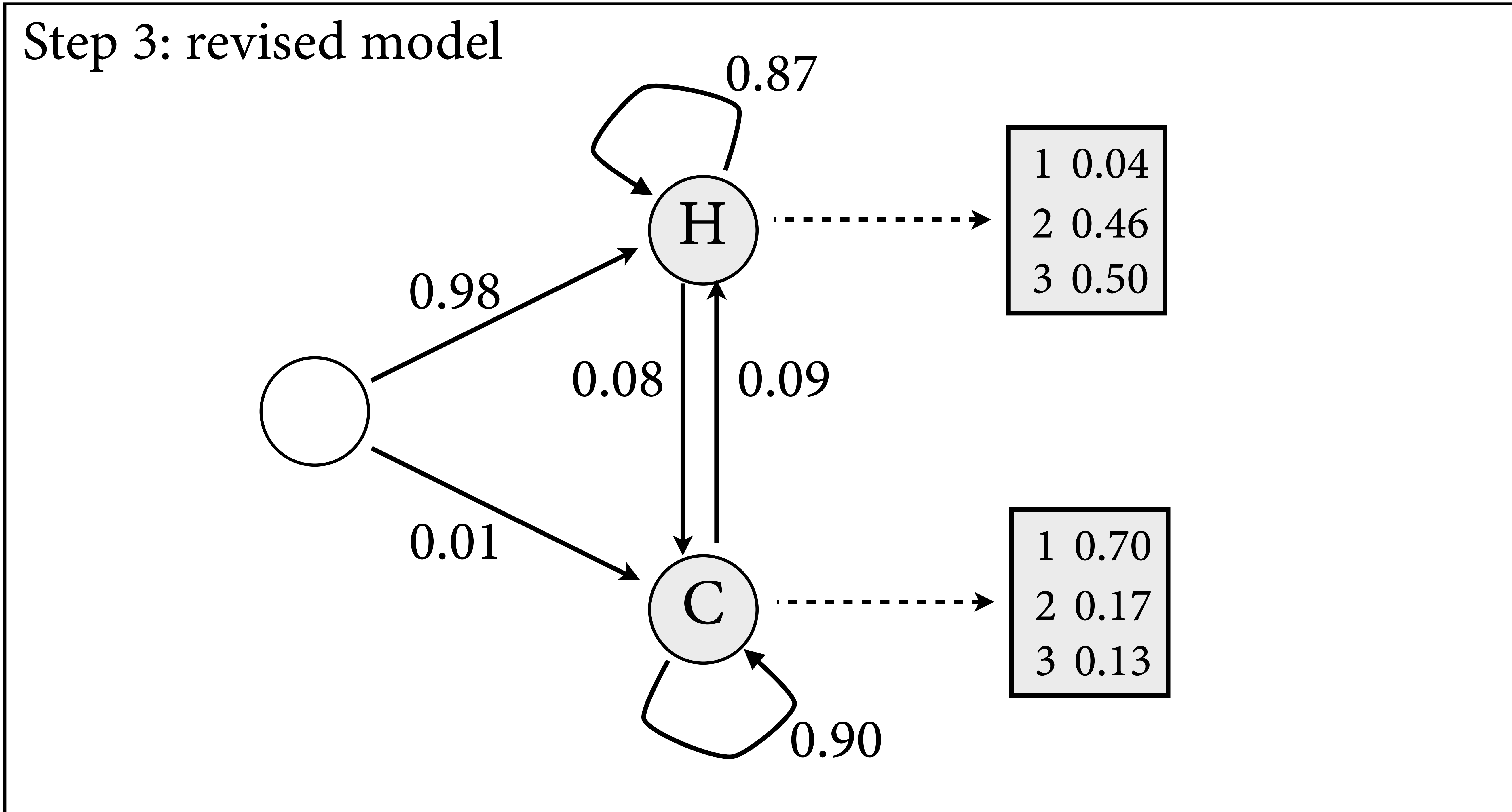
2, 3, 3, 2, 3, 2, 3, 2, 2, 3, 1, 3, 3, 1, 1, 1, 2, 1, 1, 1, 3, 1, 2, 1, 1, 1, 2, 3, 3, 2, 3, 2, 2

E-Step



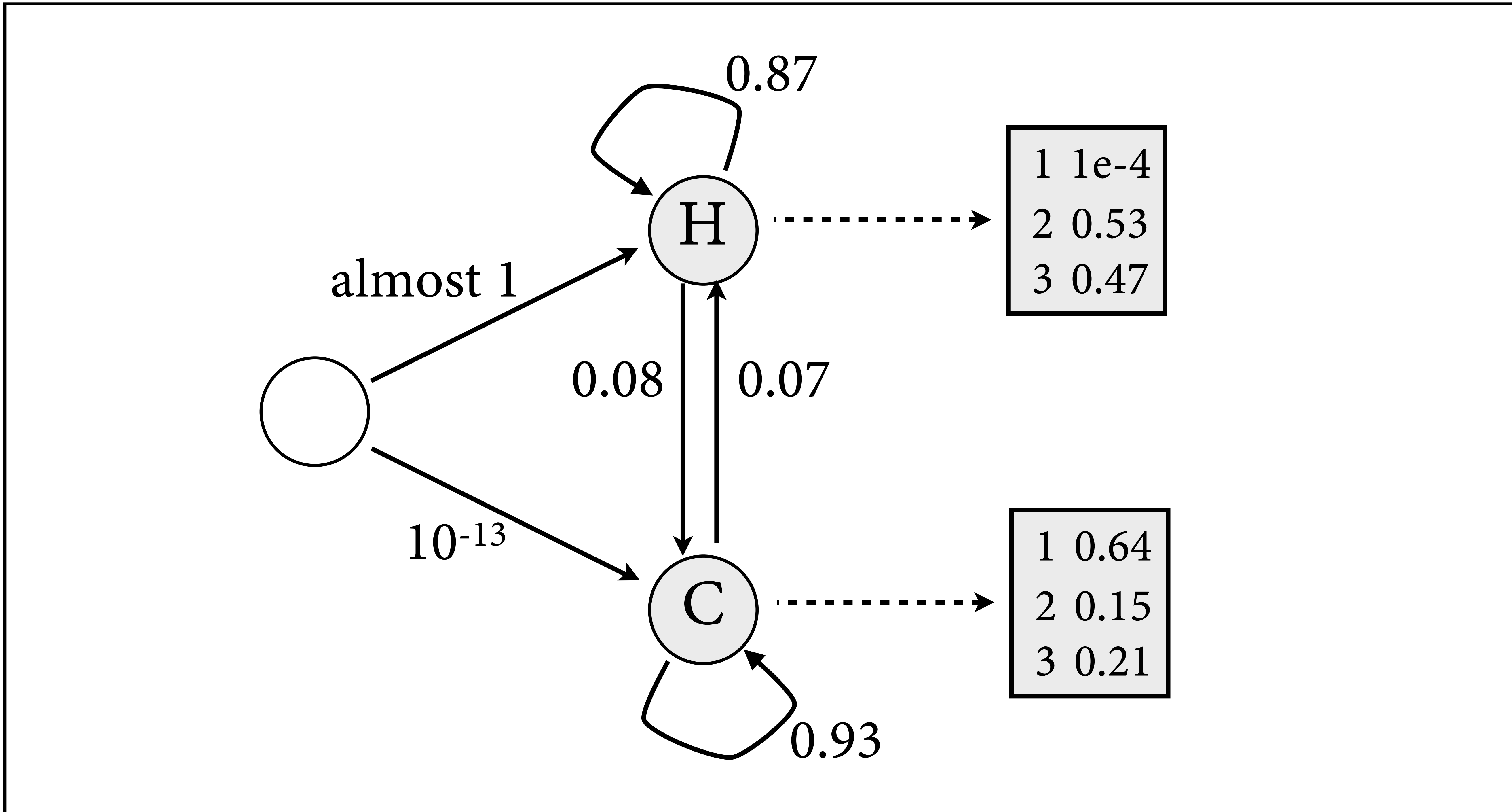
1 0.06
2 0.42
3 0.52
α = 0.37
β = 3E-16
α = 0.167
β = 7E-16
α = 0.075
β = 1E-15
H
0.87
0.88
0.09
0.11
0.12
C
α = 0.028
β = 5E-17
α = 0.006
β = 1E-16
α = 0.002
β = 3E-16
1 0.68
2 0.22
3 0.10

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

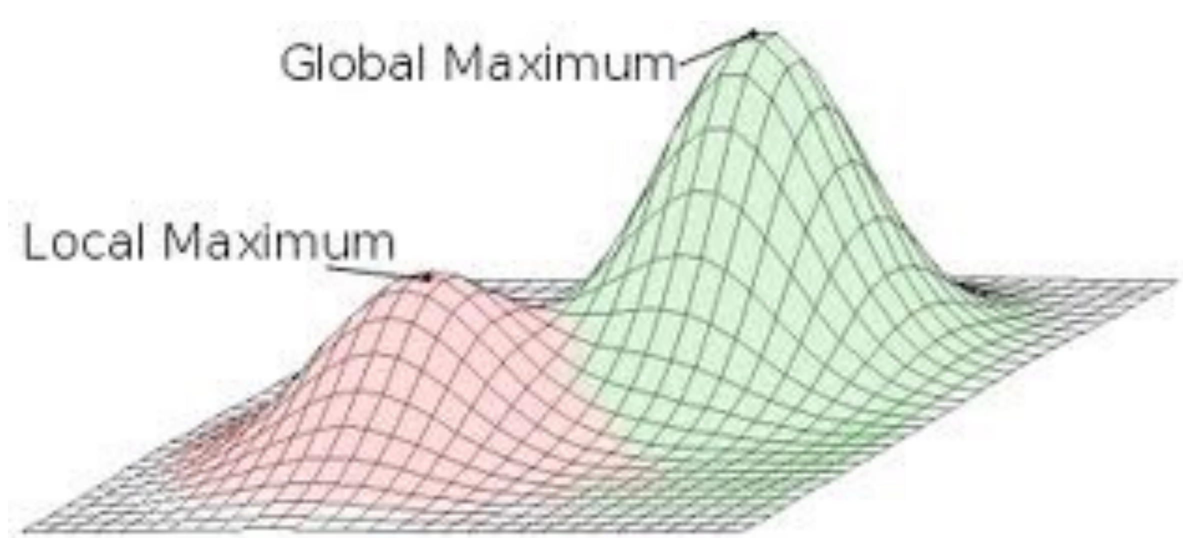
- Forward-backward algorithm also called *Baum-Welch Algorithm* after inventors.

- 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. Not really useful in practice for HMM estimation.



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