Attention Mechanisms

Jacob Andreas / MIT 6.804-6.864 / Spring 2020

Admin

HW1 is done! Look out for survey.

HW2b will be released **tonight**.
6.864 students only!

Peer reviews will be assigned on OpenReview **tomorrow**.
Each student will get 2 papers to review.
Plan to spend ~15min / paper.

Recap: recurrent neural networks

Neural networks

$$s = W_2^\top f(W_1^\top x)$$

$$x \qquad\qquad f(W_1^\top x) = h_1 \qquad\qquad W_2^\top h_1 = s$$

$$\begin{array}{r} \textit{cheap} \\ \textit{nasty} \\ \textit{tasty} \\ \textit{bad} \\ \textit{good} \end{array} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \text{input} \xrightarrow{\quad W_1 \quad} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \text{“hidden layer”} \xrightarrow{\quad W_2 \quad} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \text{output}$$

Variable-sized inputs

No ordering information!

$$\begin{array}{rc} & x \\ \begin{array}{r} \textit{cheap} \\ \textit{nasty} \\ \textit{good} \\ \textit{bad} \\ \textit{tasty} \end{array} & \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{array} \text{input}$$

Variable-sized inputs

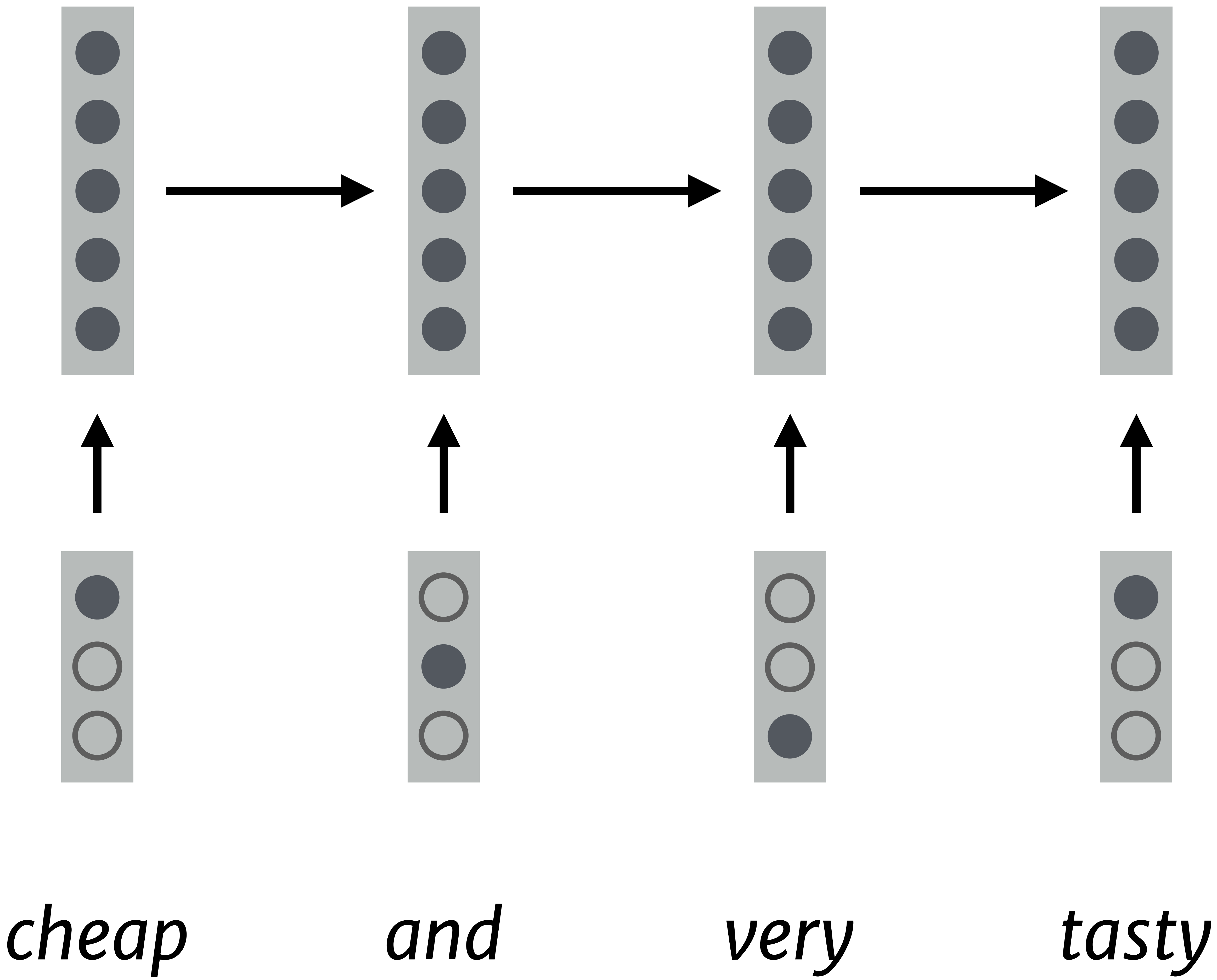
No ordering information!

$$\begin{array}{r c} & x \\ \begin{array}{r} \textit{cheap} \\ \textit{nasty} \\ \textit{good} \\ \textit{bad} \\ \textit{tasty} \end{array} & \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{array} \text{ input}$$

$$\begin{array}{cccc} \textit{cheap} & \textit{and} & \textit{very} & \textit{tasty} \\ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} & \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \end{array}$$

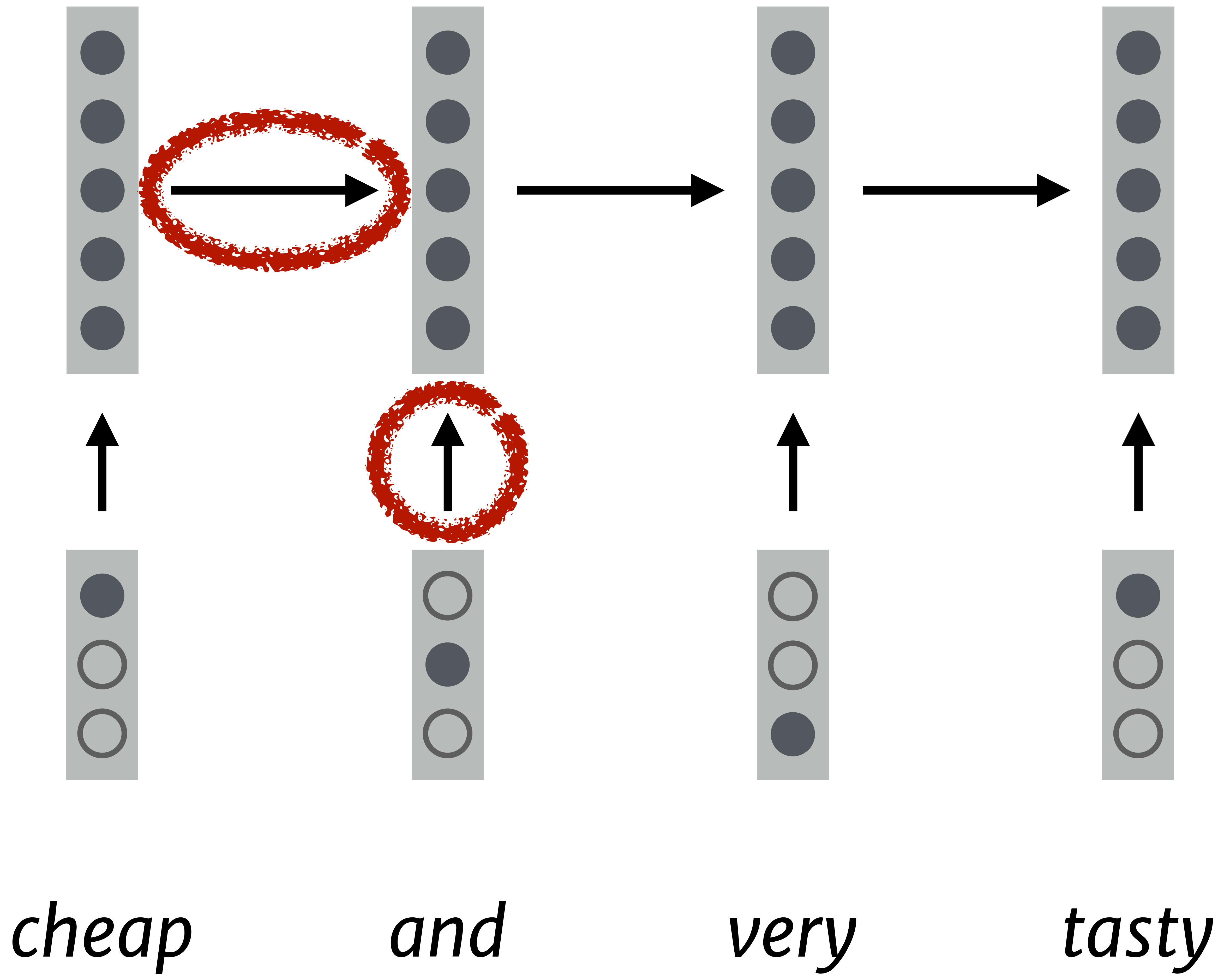
No fixed input dimension!

Recurrent neural networks



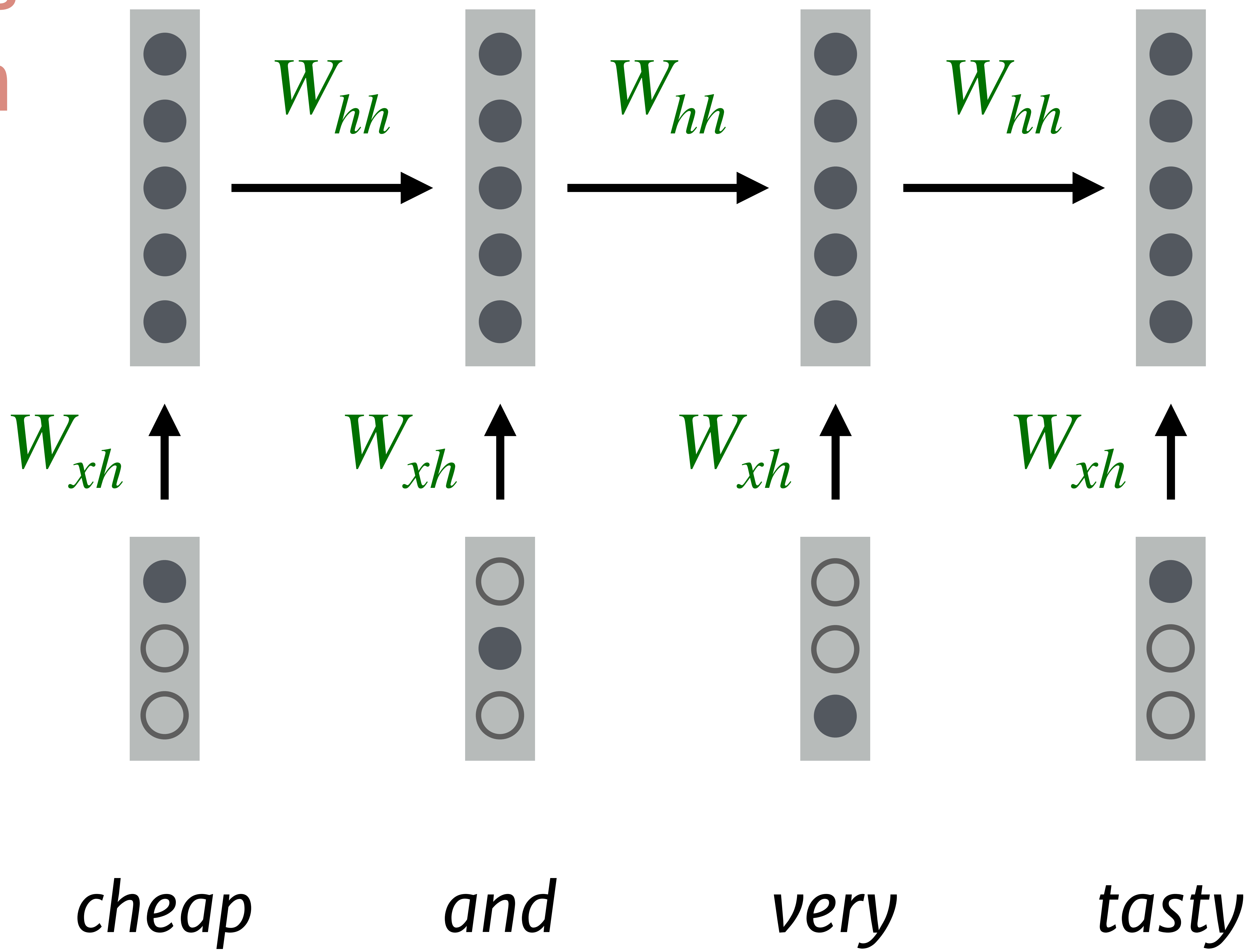
Recurrent neural networks

Hidden states depend on an earlier state **and** an input



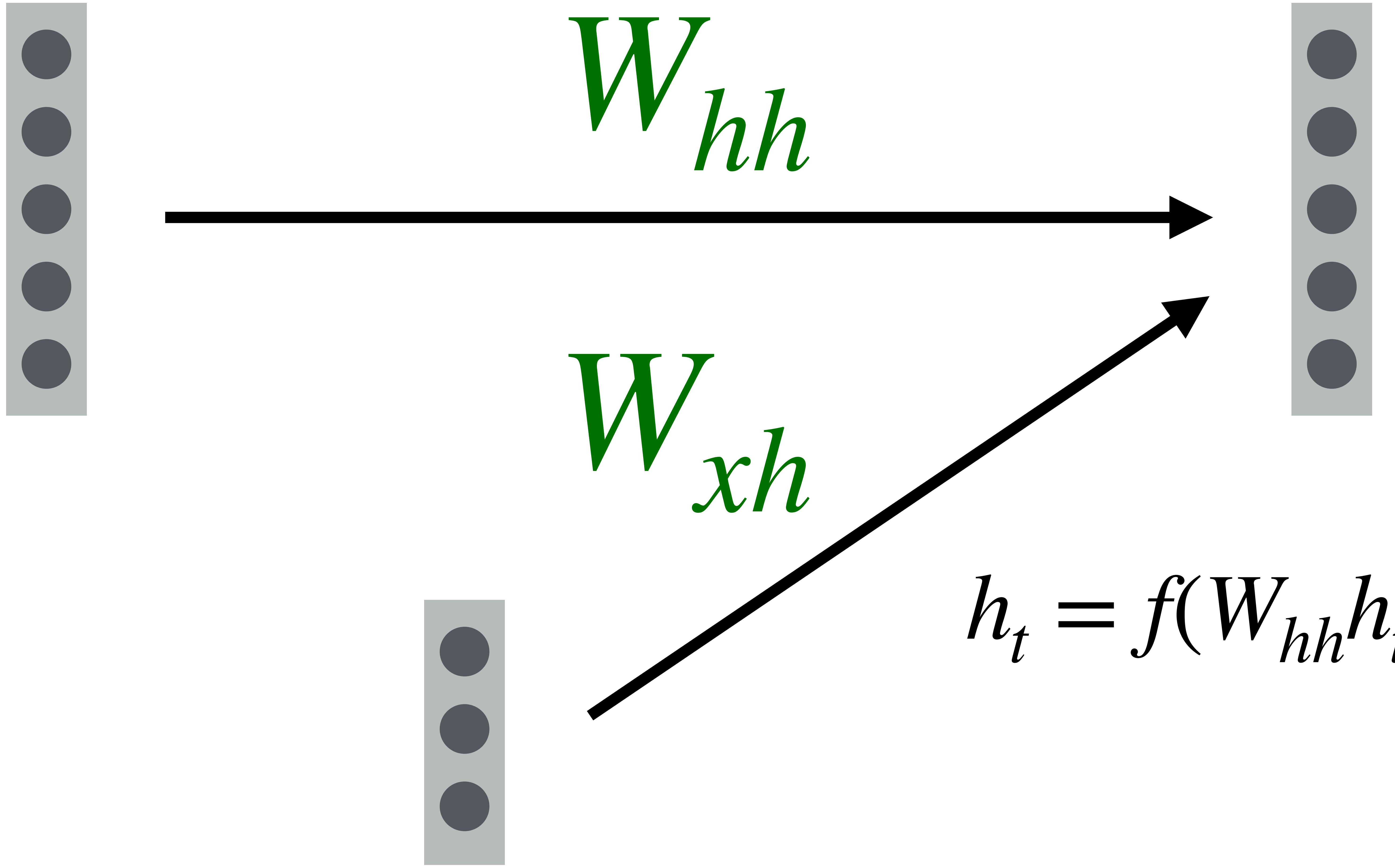
Recurrent neural networks

Hidden states depend on an earlier state **and** an input



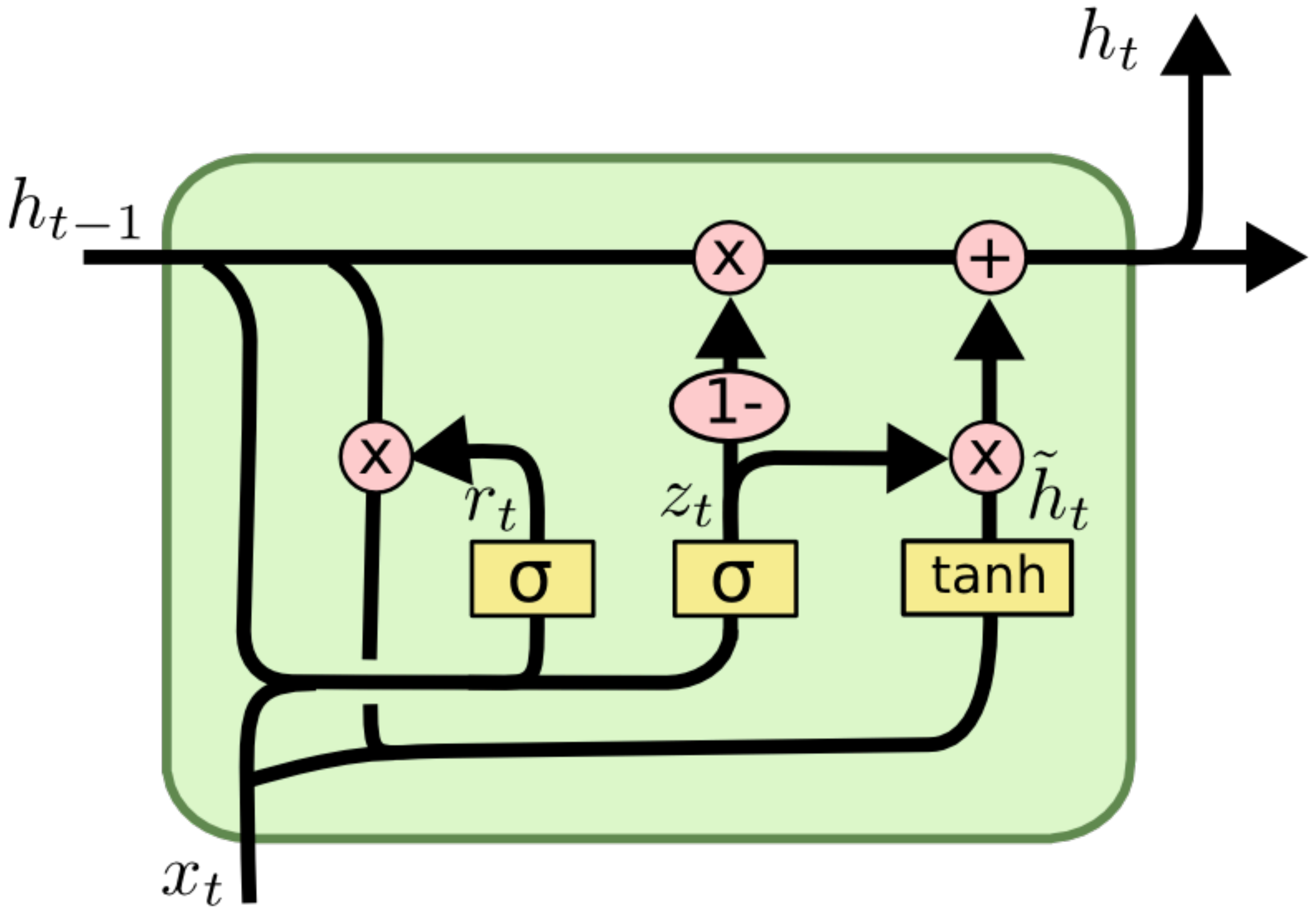
Same weights at every state!

"Vanilla" RNNs



$$h_t = f(W_{hh}h_{t-1} + W_{xh}x_i + b)$$

Gated Recurrent Units



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

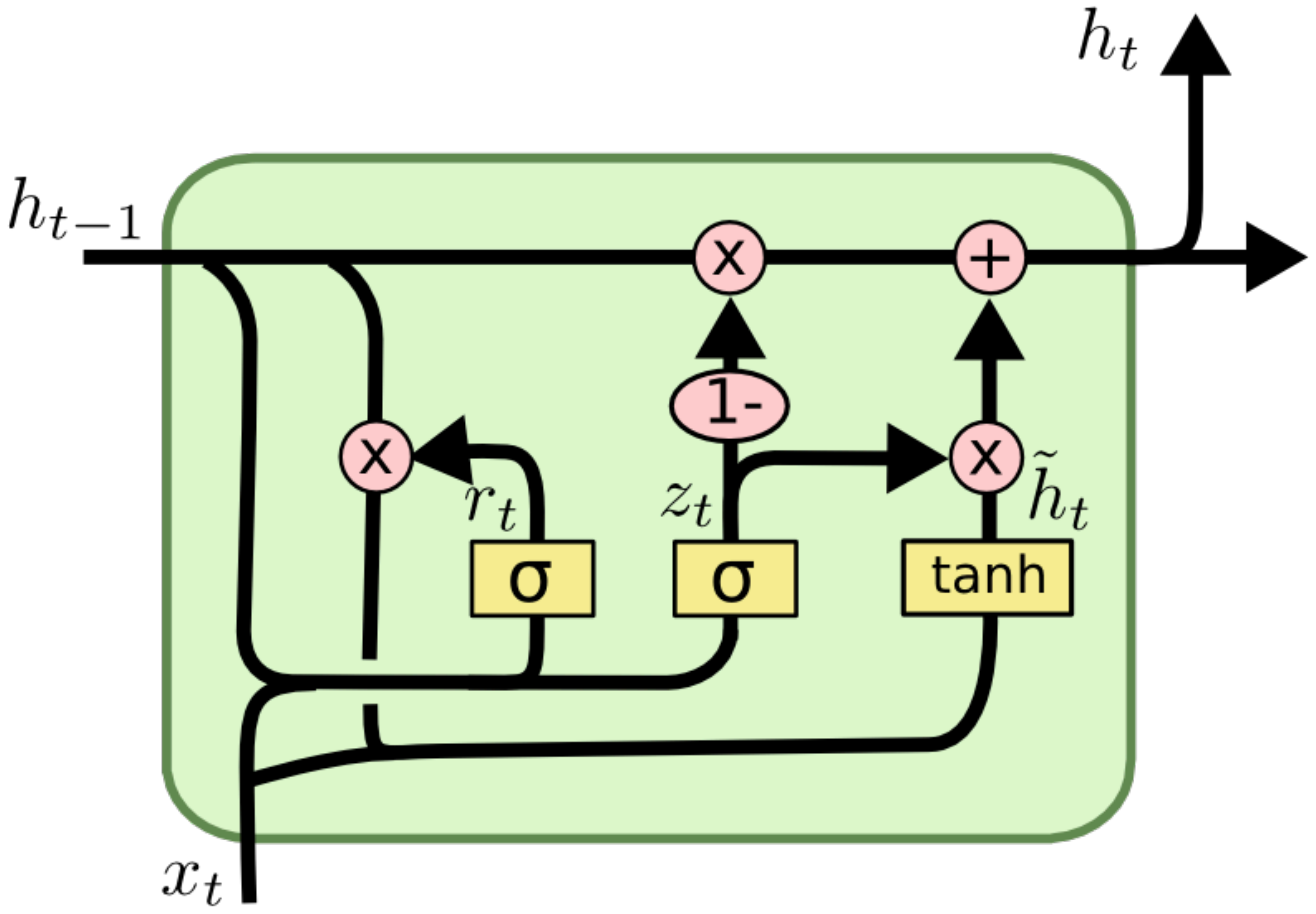
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

[Image: Cristopher Olah]

Gated Recurrent Units



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

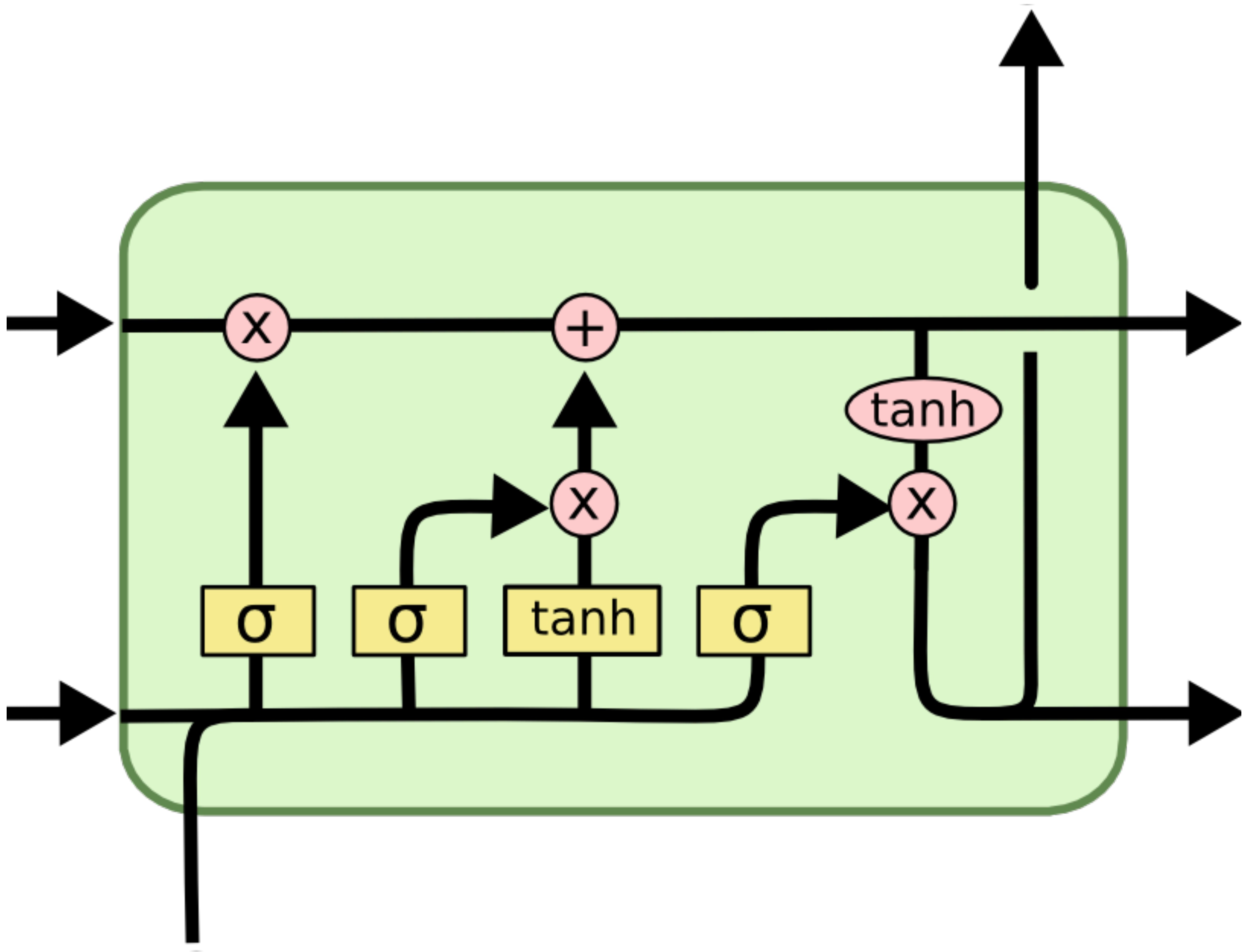
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = \boxed{(1 - z_t) * h_{t-1}} + z_t * \tilde{h}_t$$

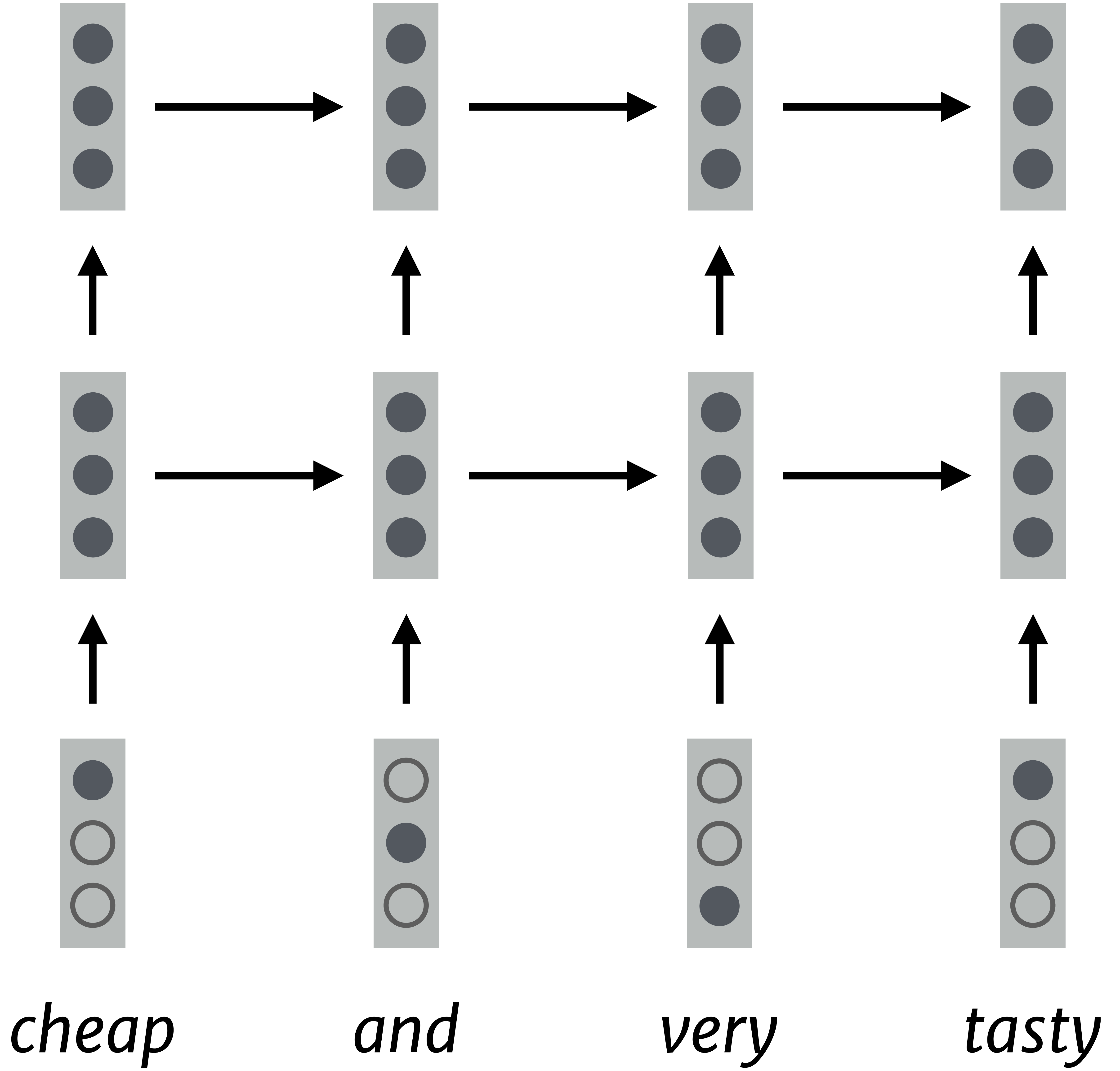
[Image: Cristopher Olah]

Long Short-Term Memory Units

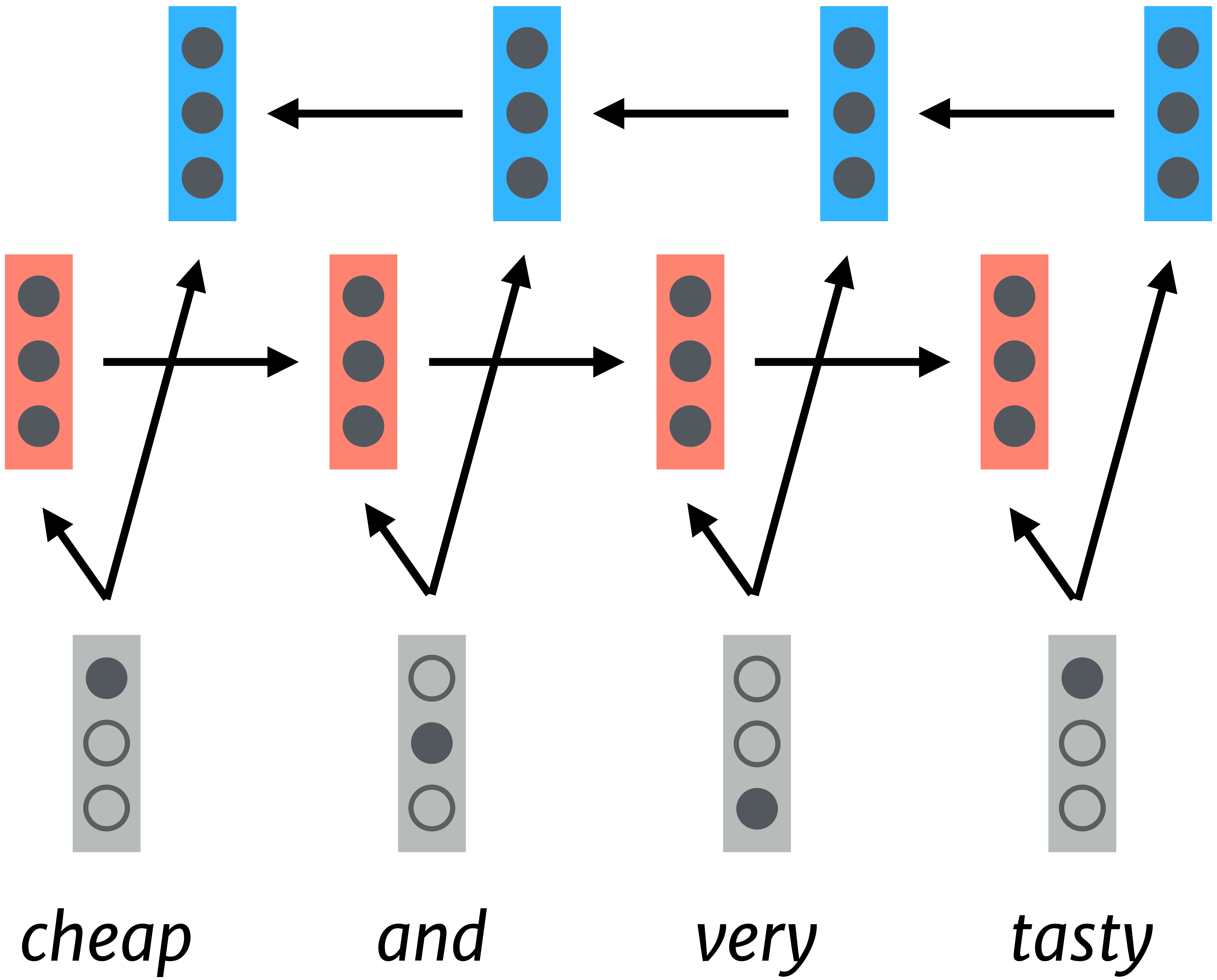


[Image: Cristopher Olah]

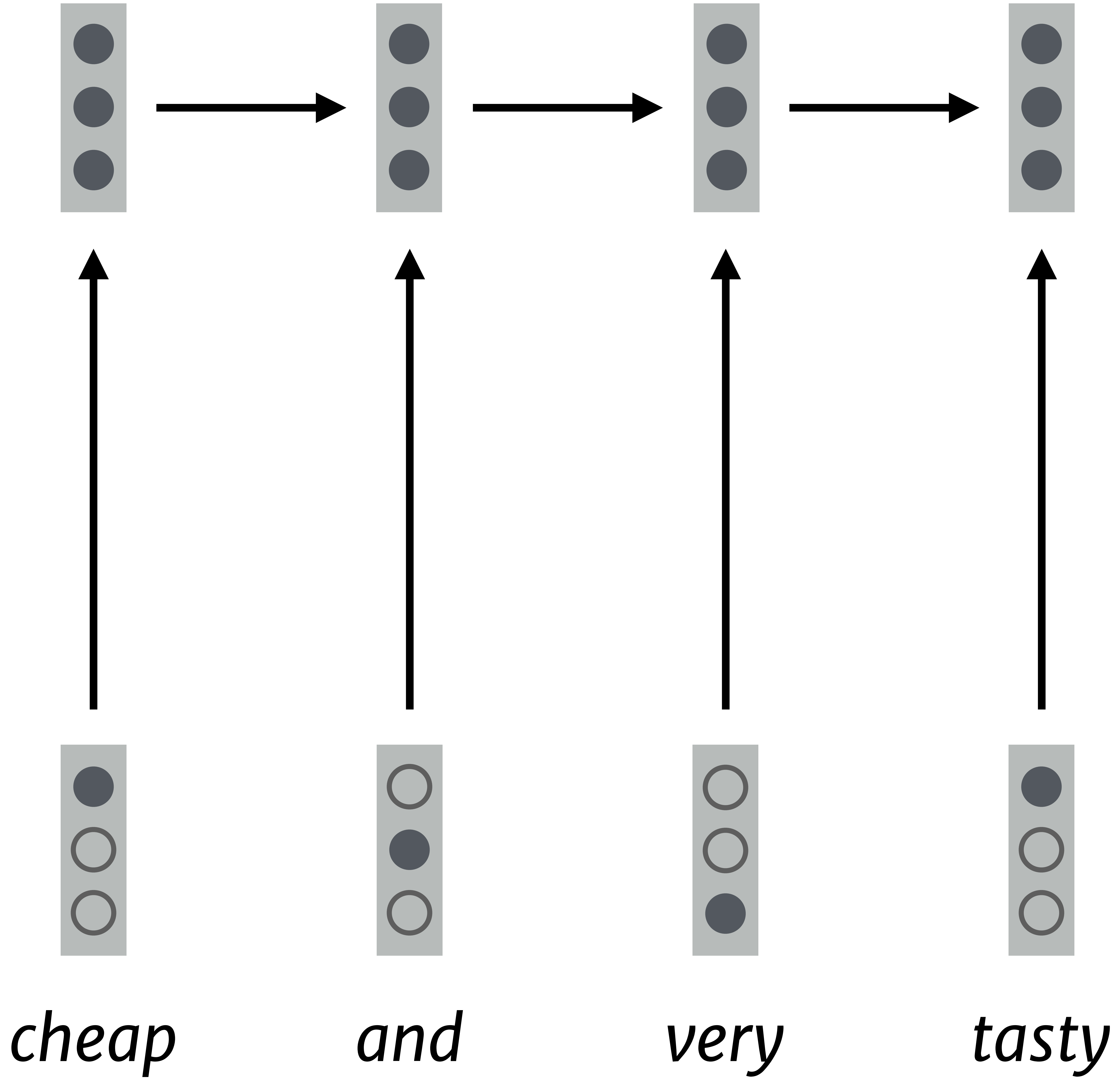
Deeper RNNs



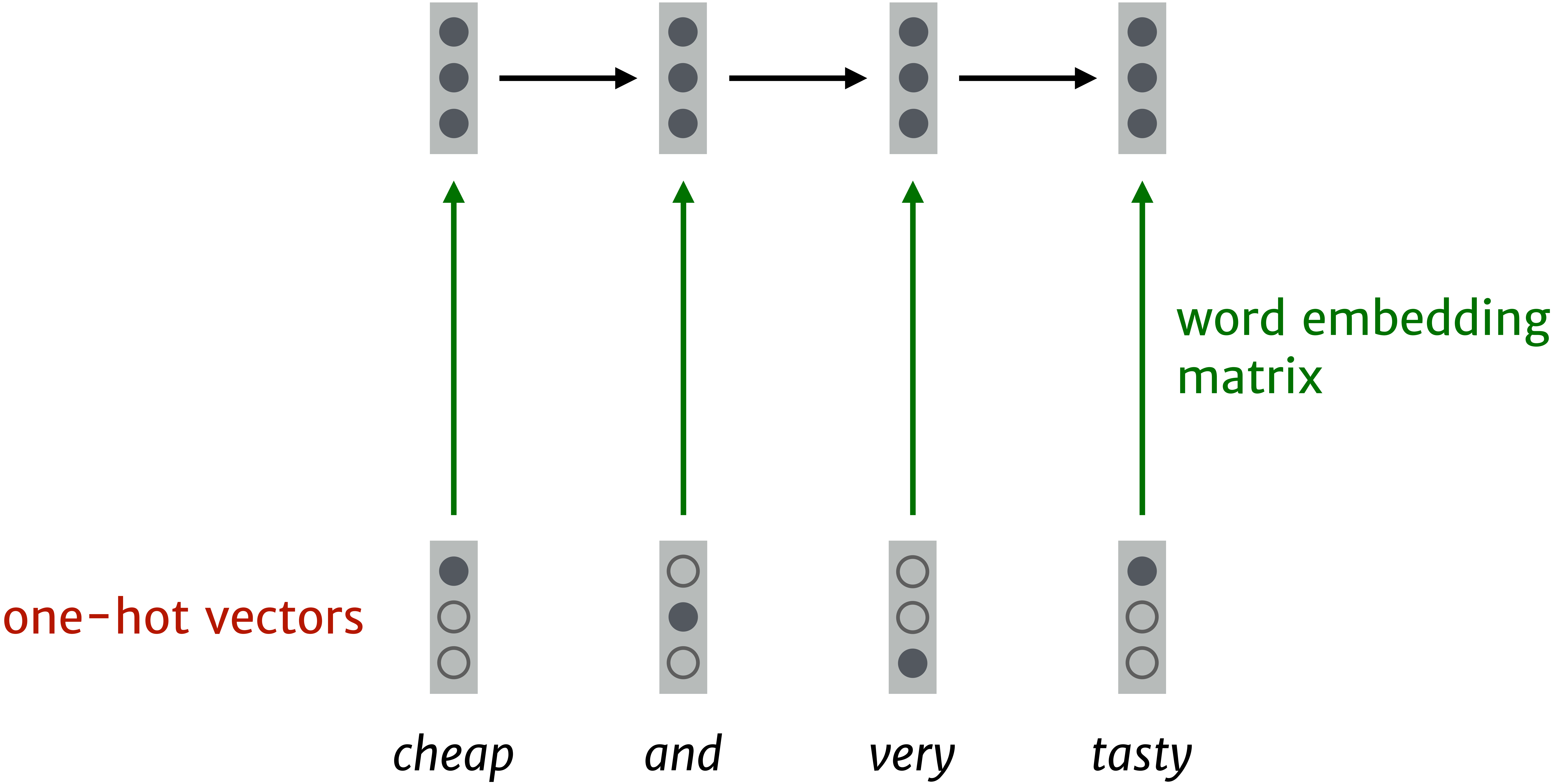
Bidirectional RNNs



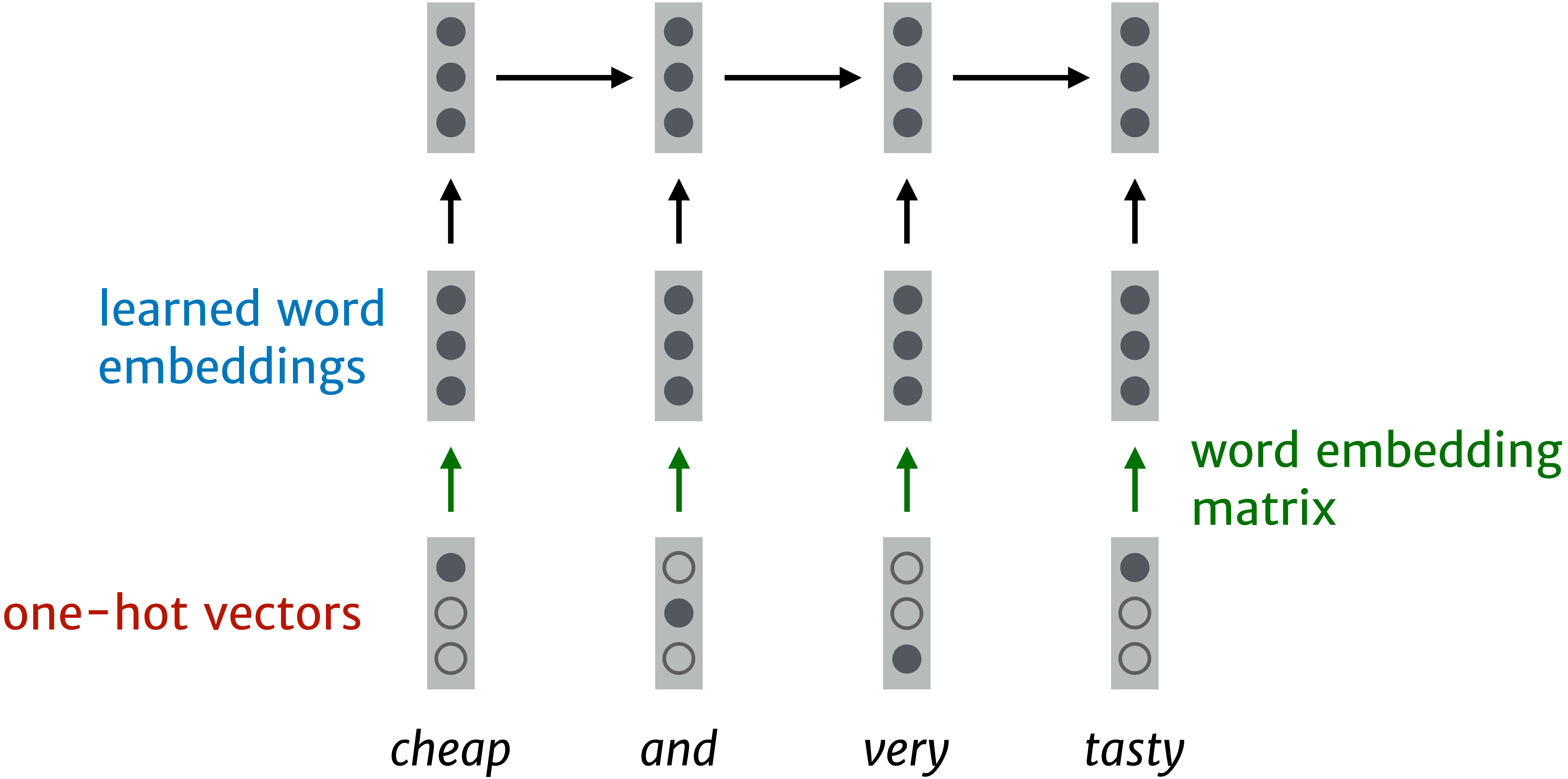
RNNs and word embeddings



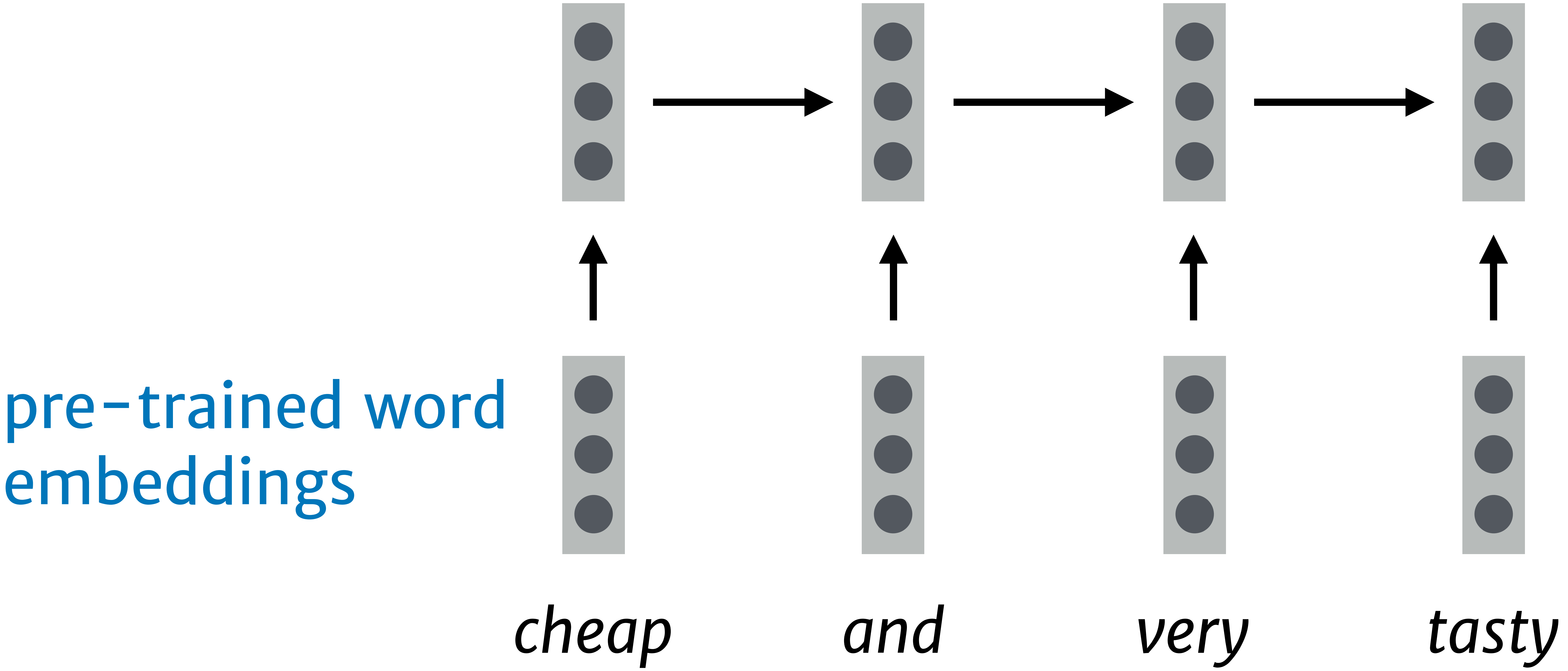
RNNs and word embeddings



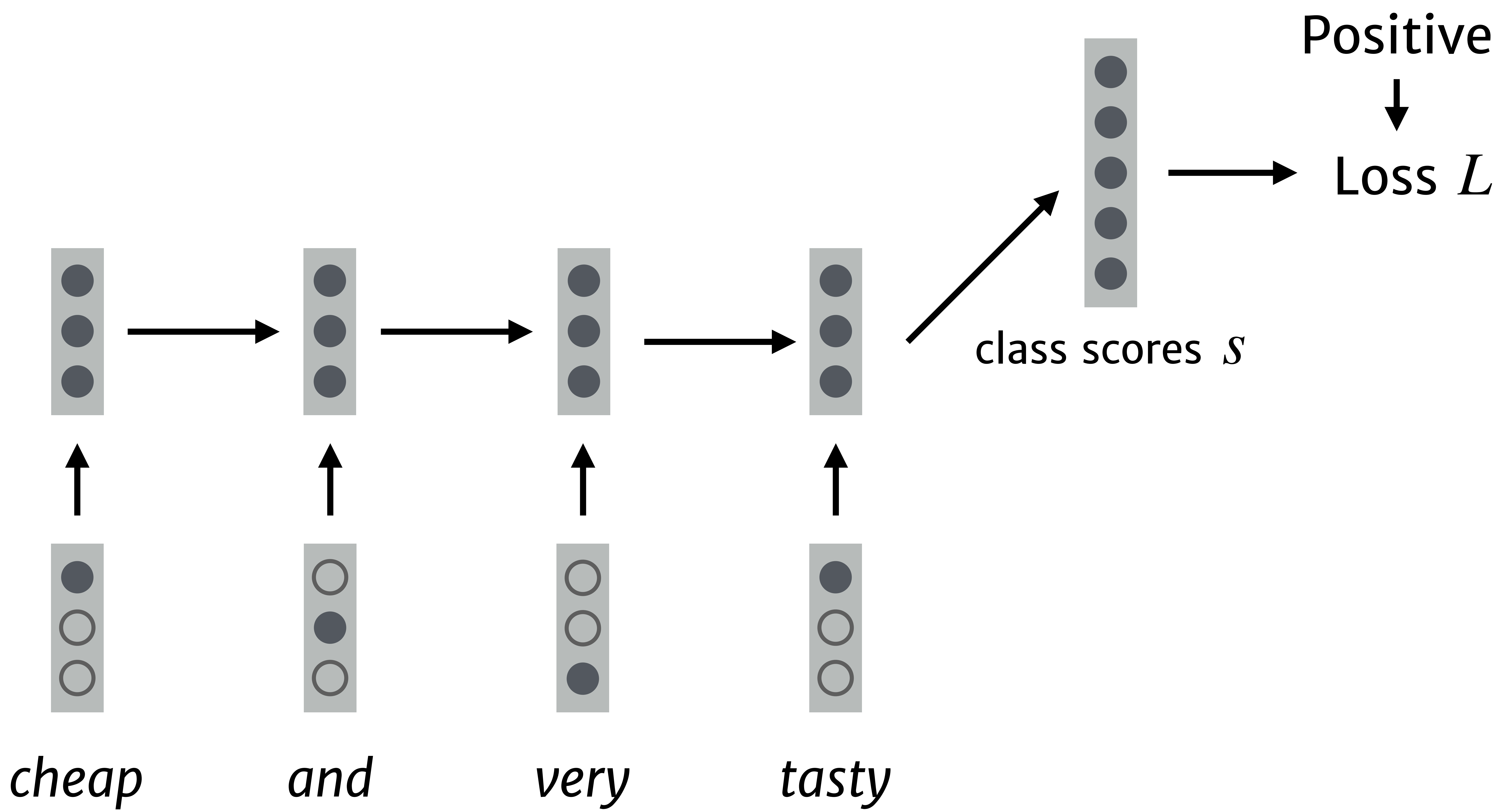
RNNs and word embeddings



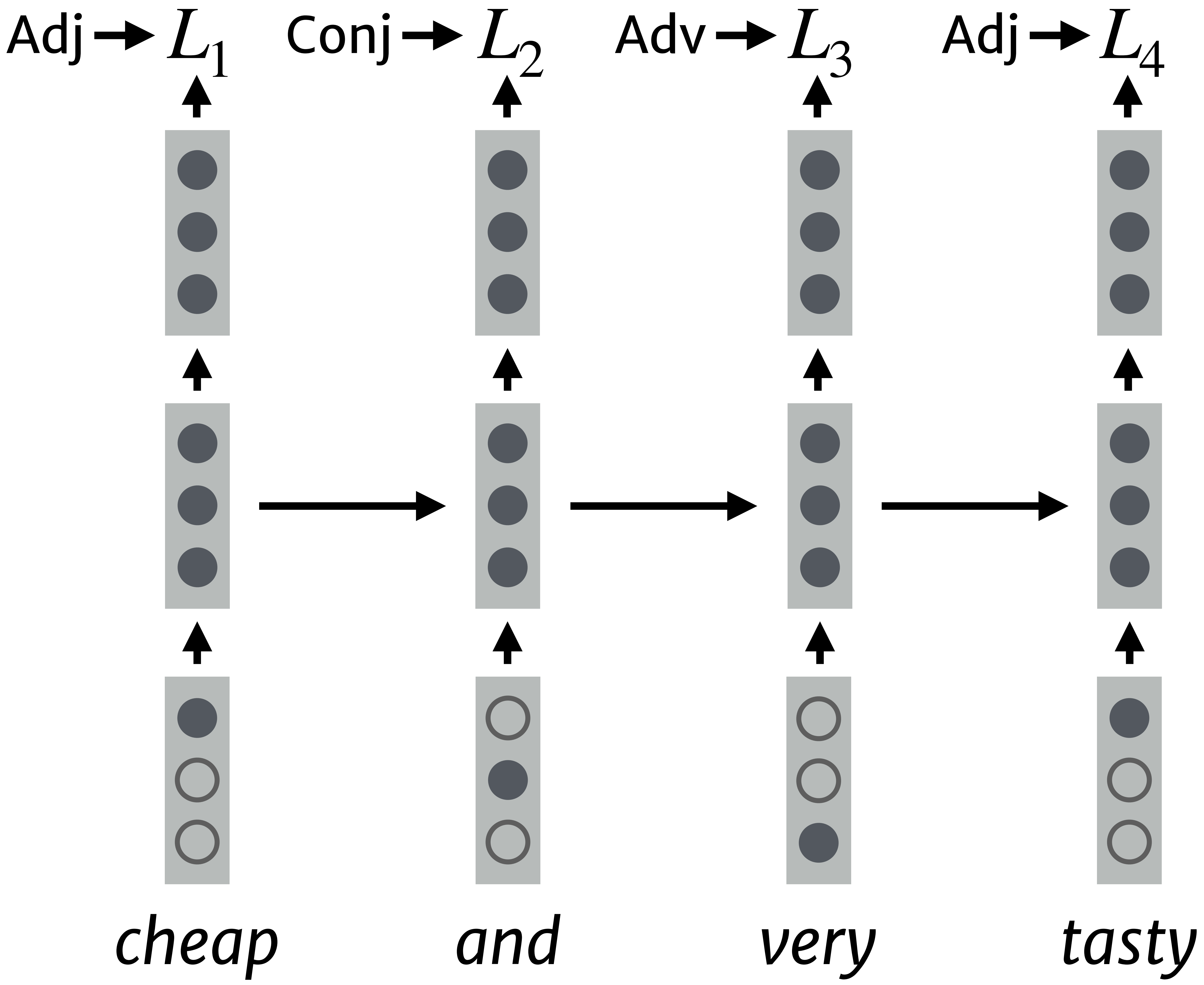
RNNs and word embeddings



Text classification



Sequence labeling

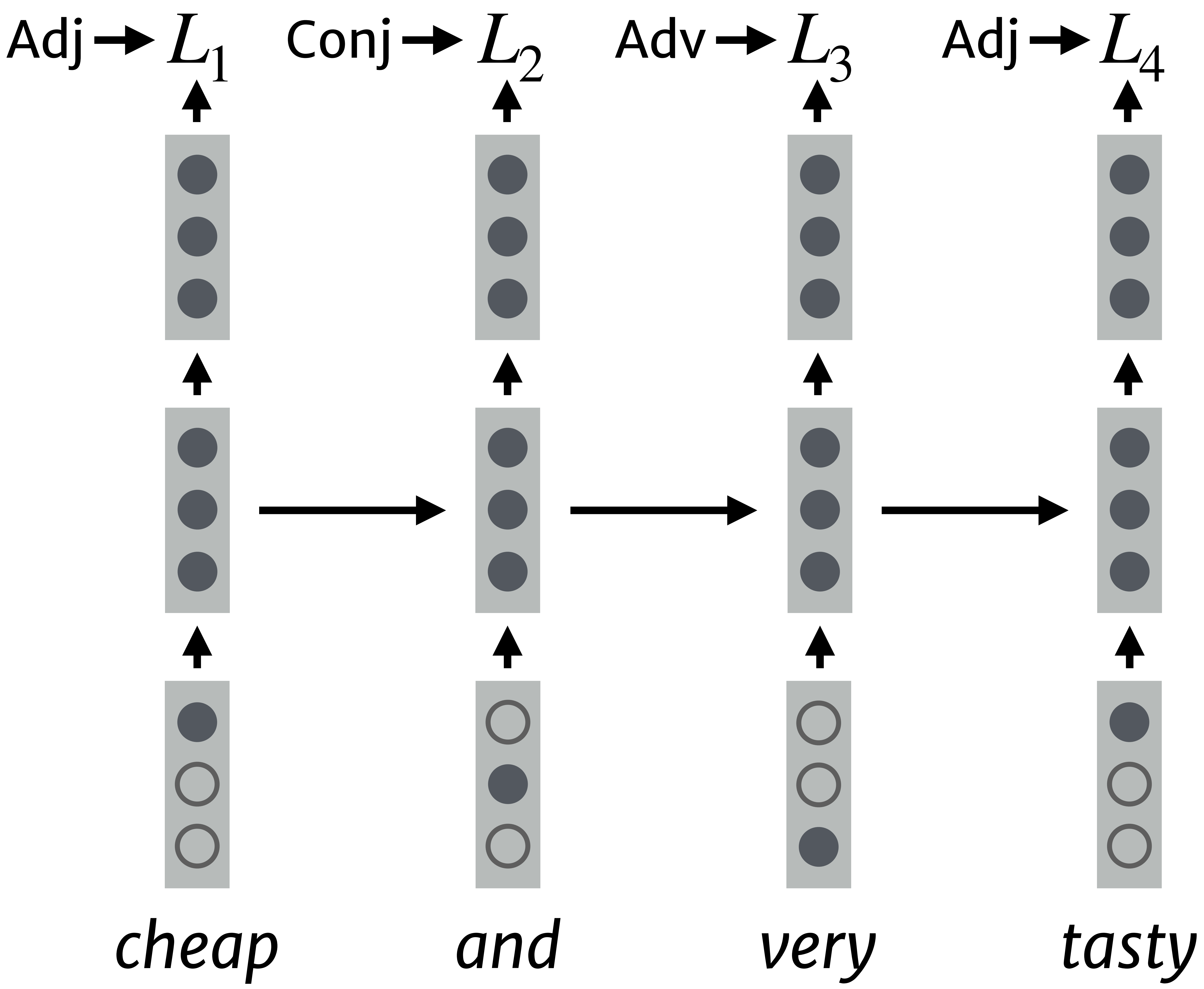


$$L = \sum_t L_t$$

e.g.

$$= -\sum_t \log p(y_t \mid x_{:t})$$

Sequence labeling



$$L = \sum_t L_t$$

e.g.

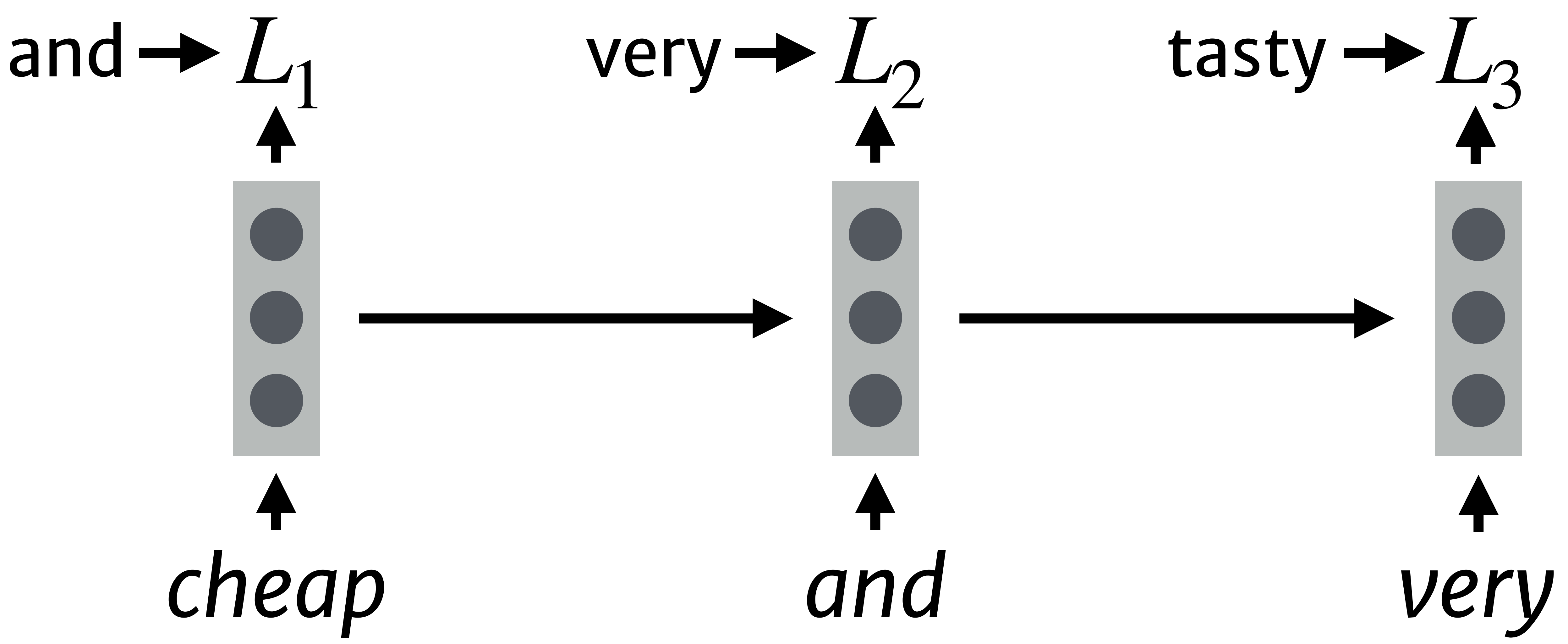
$$= -\sum_t \log p(y_t \mid x_{:t})$$

product of indep. conditionals!

Langauge Modeling

A (unidirectional) can compute $p(y_t \mid x_{:t})$.

Suppose for a sequence x we set $y_t = x_{t+1}$.

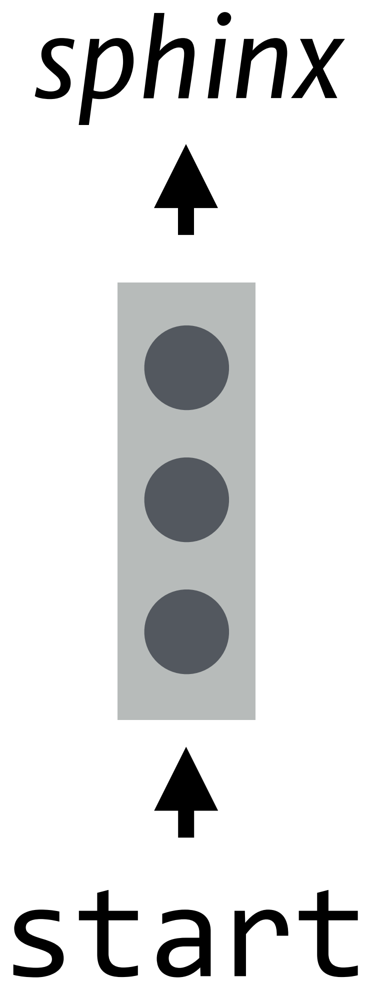


then $\sum_t \log p(x_{t+1} \mid x_{:t}) = p(x)$

Language modeling: sampling

How do we sample from $p(x)$?

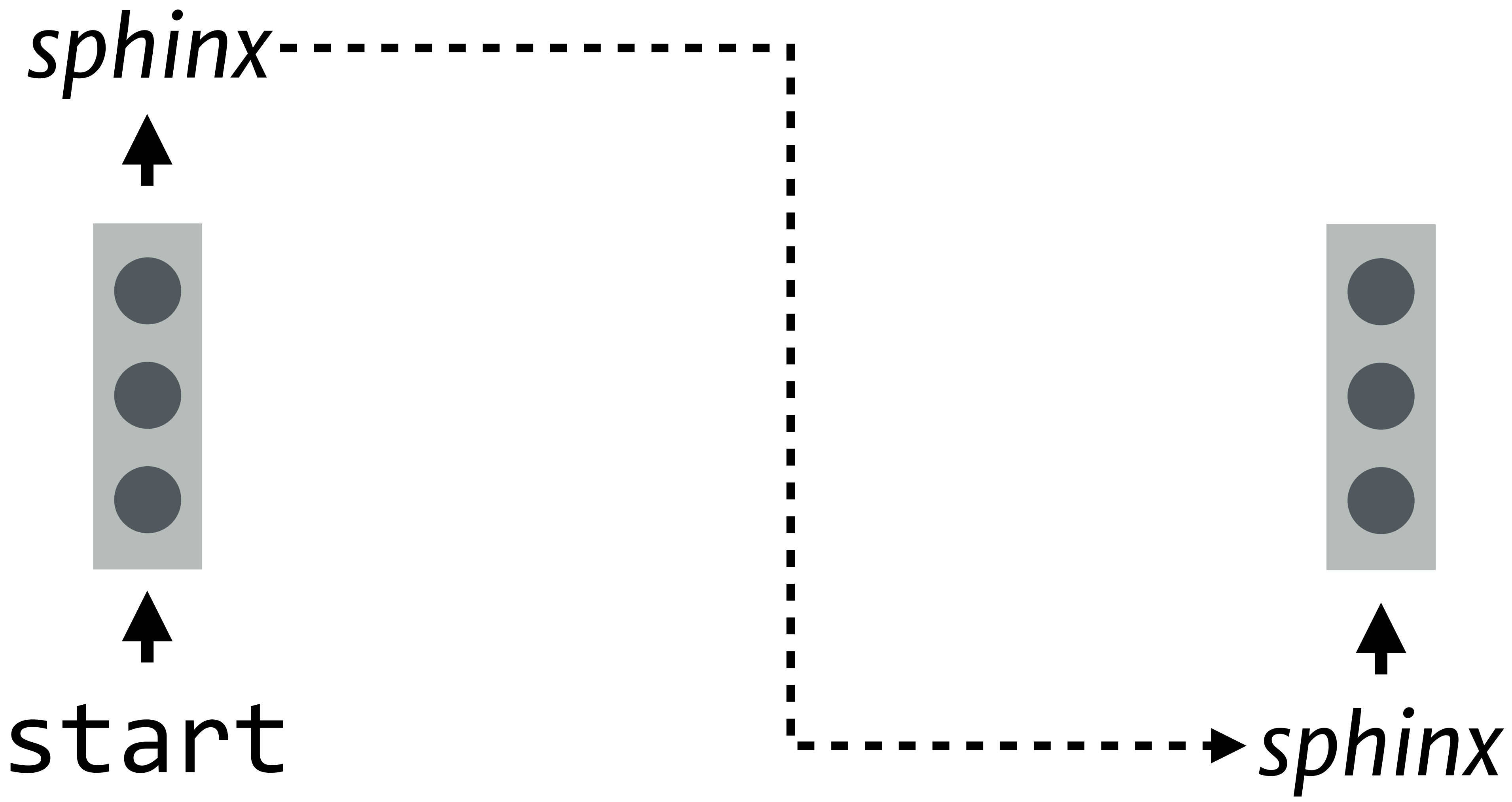
$$x_1 \sim p(x_1 \mid \texttt{start})$$



Language modeling: sampling

How do we sample from $p(x)$?

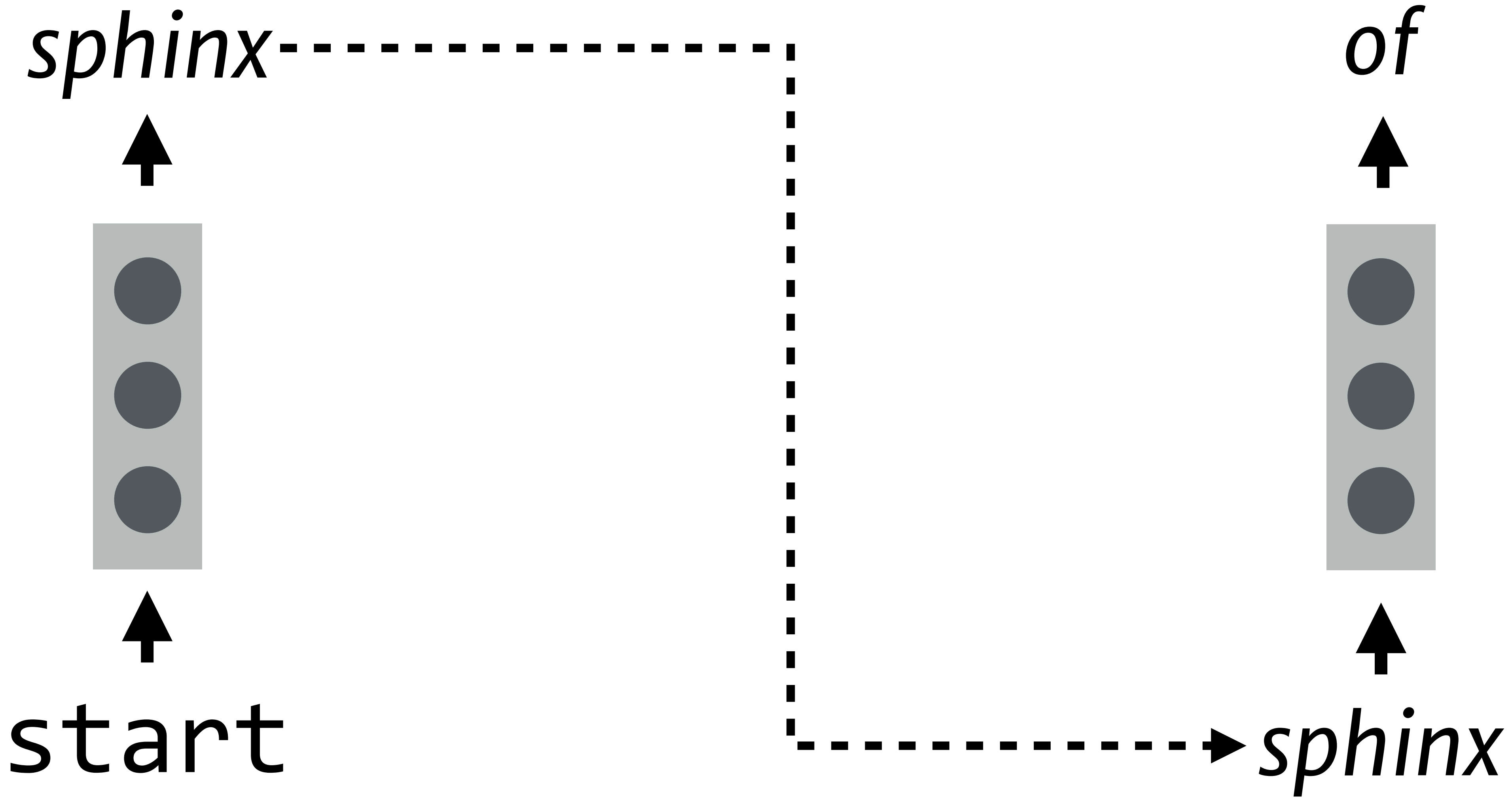
$$x_1 \sim p(x_1 \,|\, \texttt{start})$$



Language modeling: sampling

How do we sample from $p(x)$?

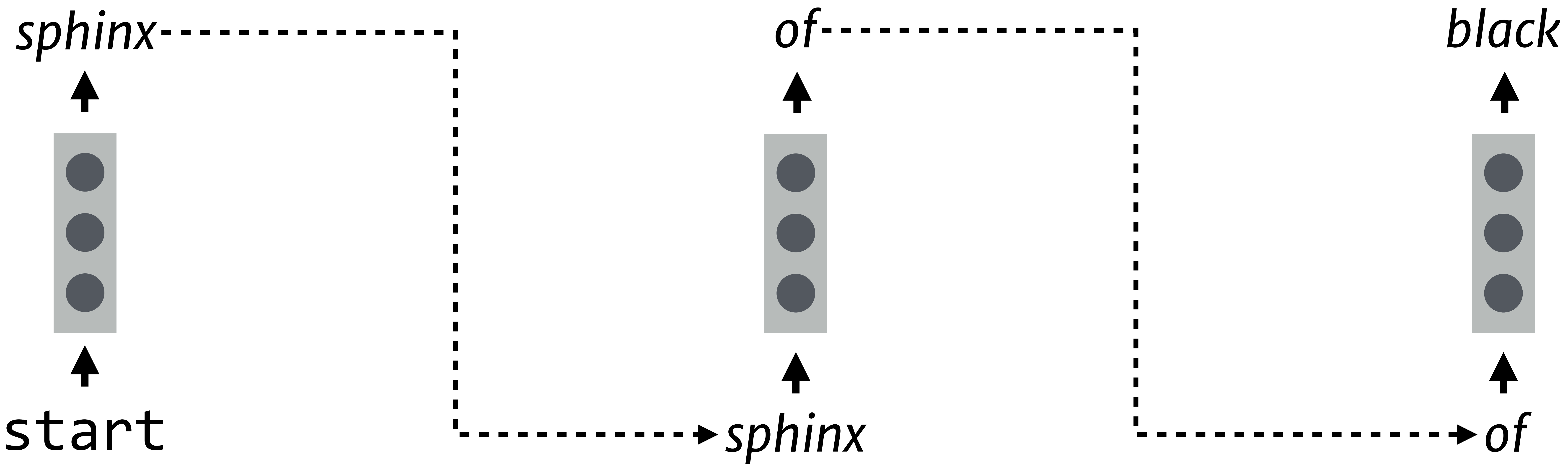
$x_1 \sim p(x_1 \mid \texttt{start})$ $\qquad$ $x_2 \sim p(x_2 \mid \textsf{sphinx})$



Language modeling: sampling

How do we sample from $p(x)$?

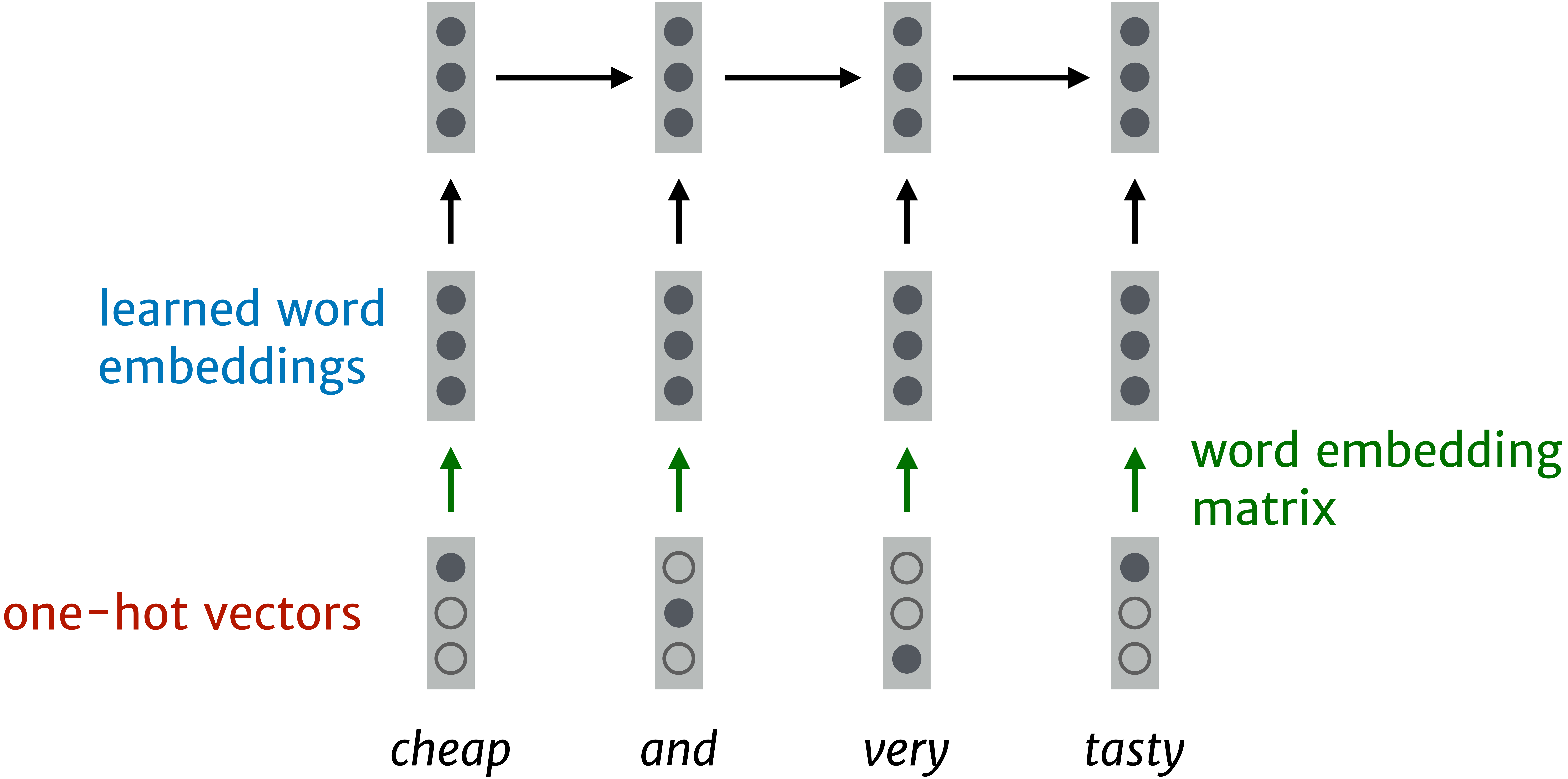
$x_1 \sim p(x_1 \,|\, \texttt{start})$ $\qquad$ $x_2 \sim p(x_2 \,|\, \text{sphinx})$ $\qquad$ $x_2 \sim p(x_2 \,|\, \text{sphinx of})$



SPHINX OF BLACK QUARTZ, JUDGE MY VOW

[Image: egyptianmarketplace.com]

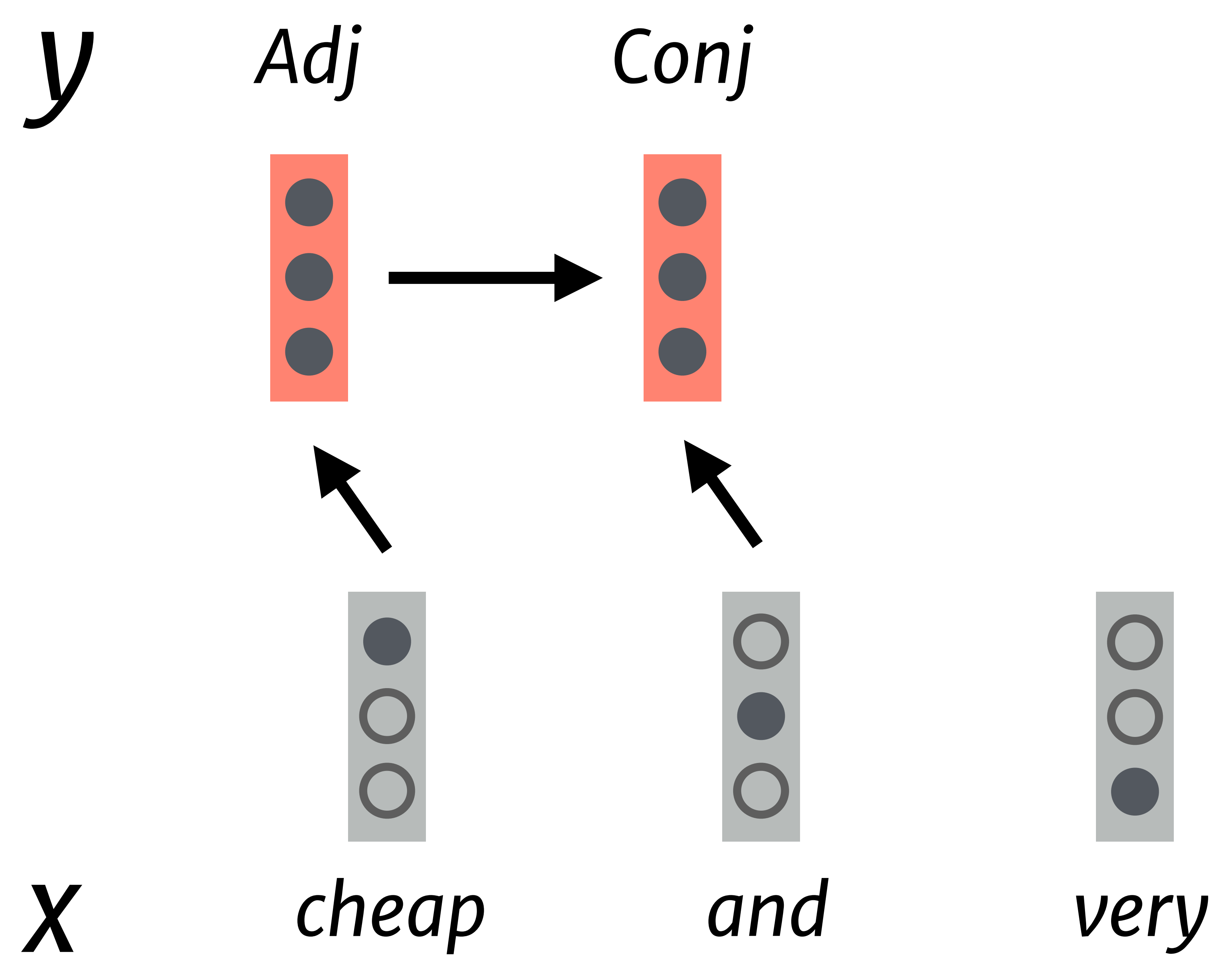
Language modeling as representation learning



RNNs as Markov chains

I can train this network to predict:

$$\log p(y_t \mid x_{:t})$$

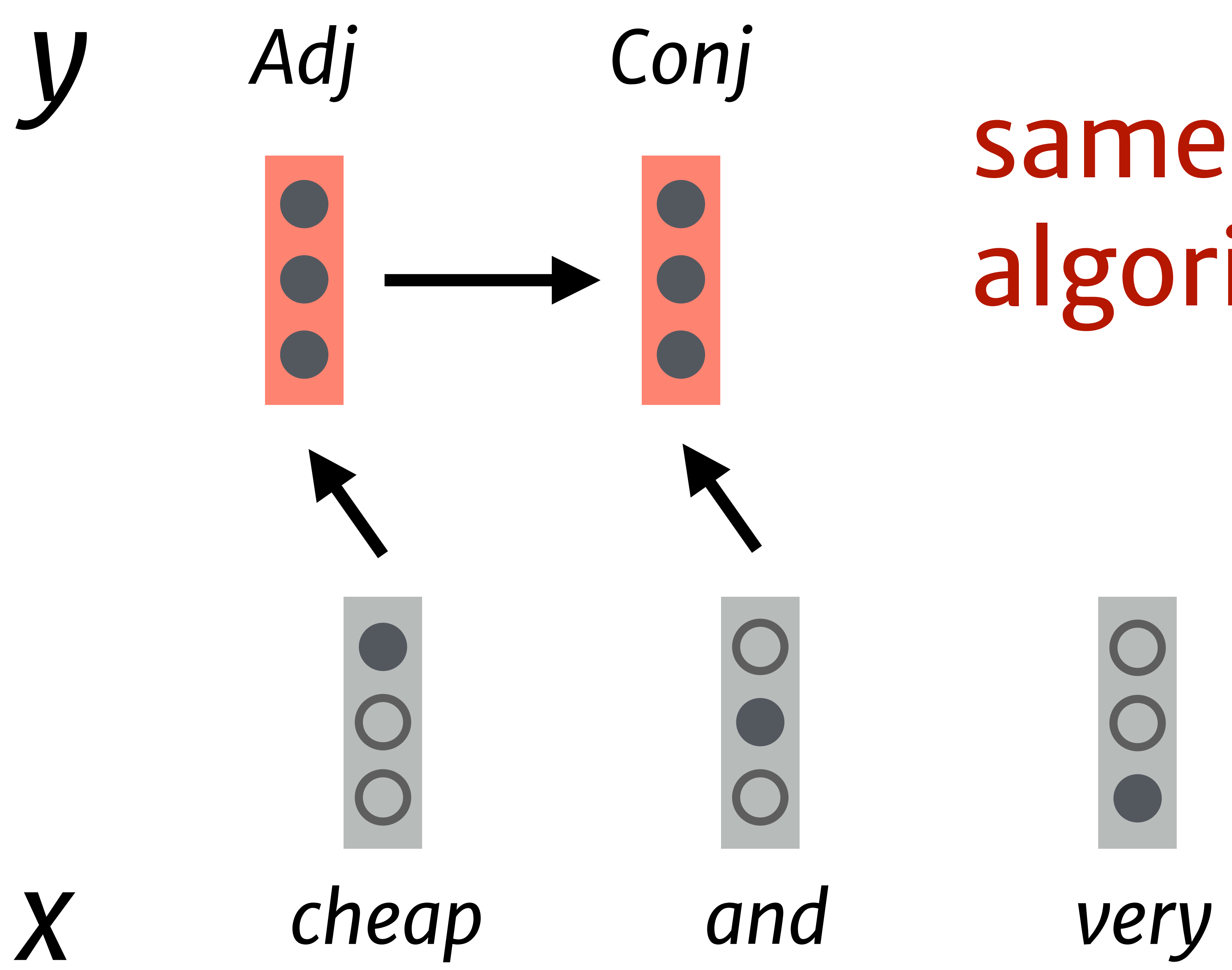


RNNs as Markov chains

I can train this network to predict:

$$\log p(y_t \mid x_{:t}) = p(q_t \mid O_{:t})$$

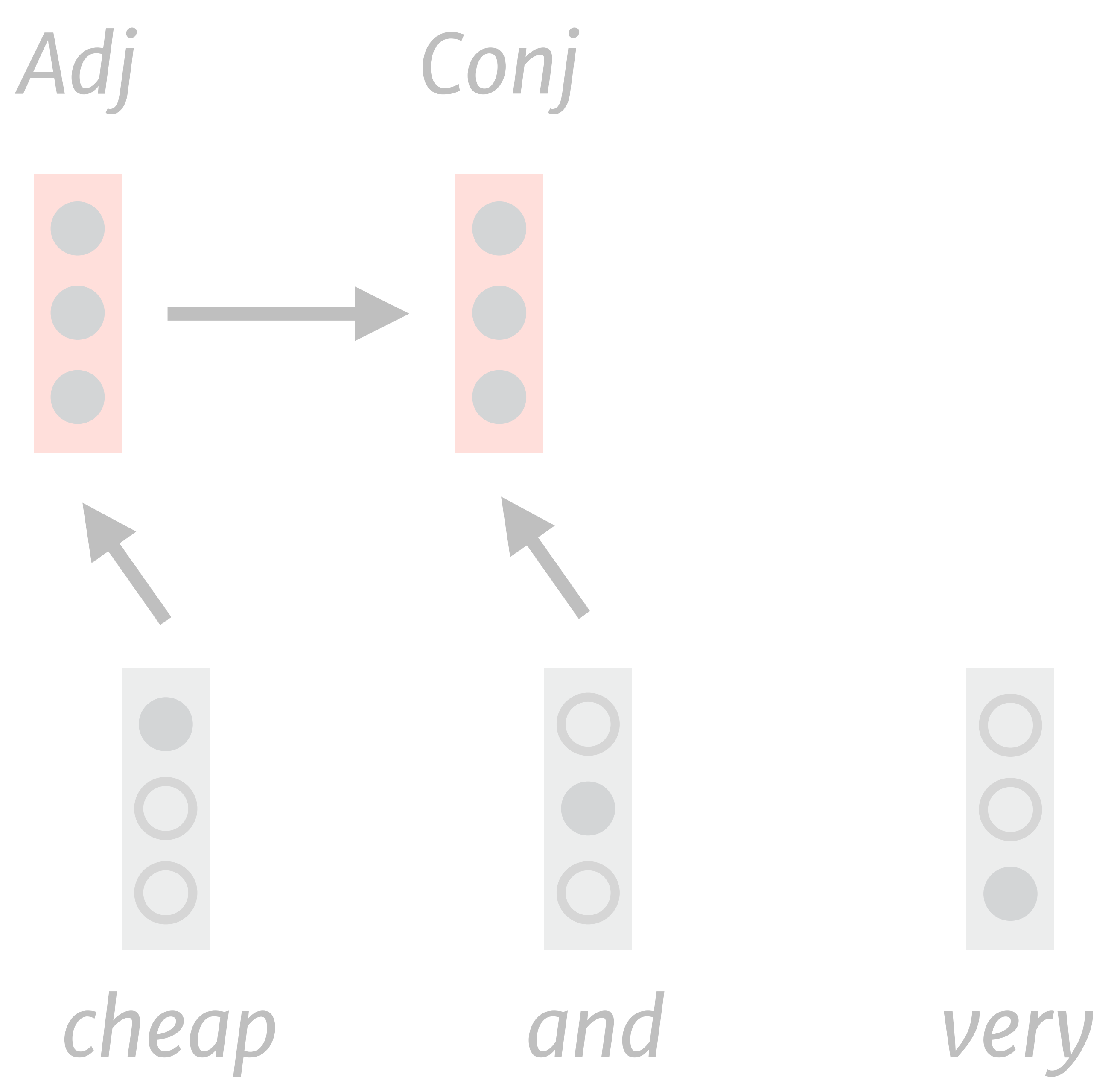
same as forward algorithm!



RNNs as Markov chains

I can train this network to predict:

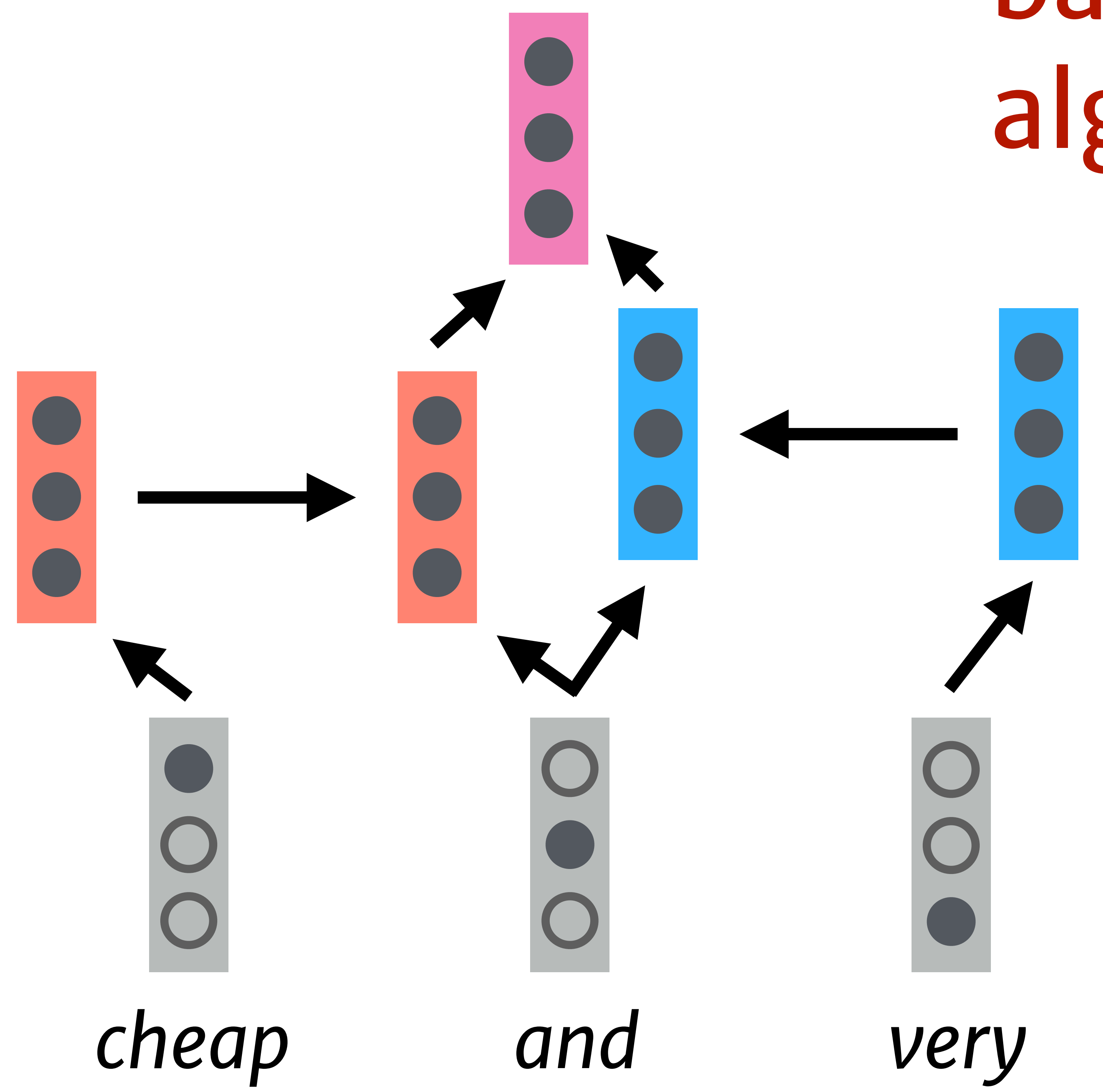
$$\log p(y_t \mid x_{:t}) = p(q_t \mid O_{:t})$$



I can train this network to predict:

$$\log p(y_t \mid x) = p(q_t \mid O)$$

forward–backward algo!



Sequence-to-sequence models

A dataset of math problems

One plus one equals two.

Two times two equals four.

Seven is prime.

One plus two times three equals seven.

A dataset of math problems

One plus one equals two.

Two times two equals four.

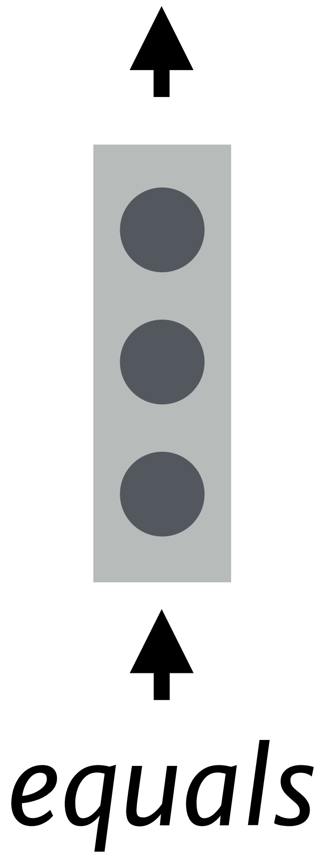
Seven is prime.

One plus two times three equals seven.

Two times three times three equals ???

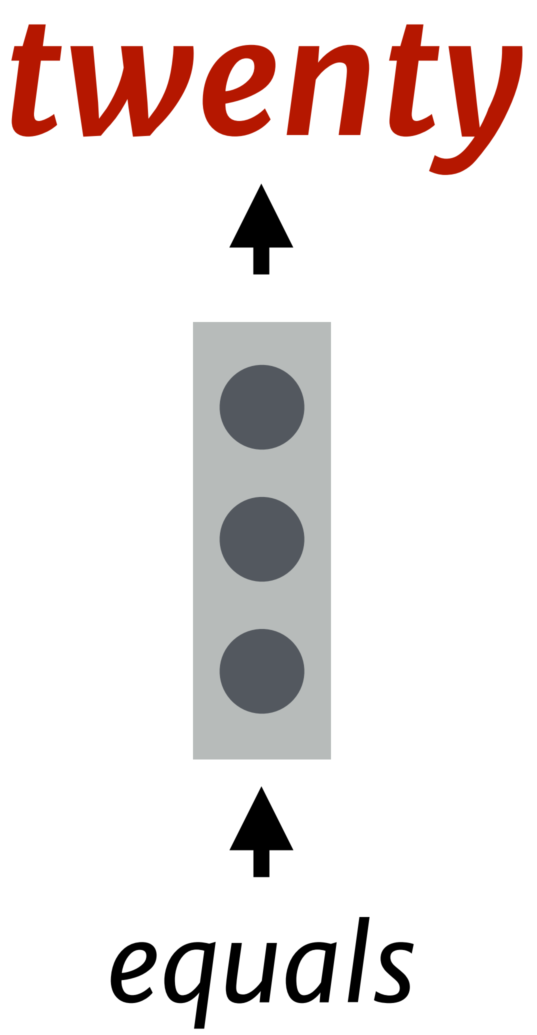
Answering math problems with LMs

$x_1 \sim p(x_1 \mid$ ... times three equals$)$



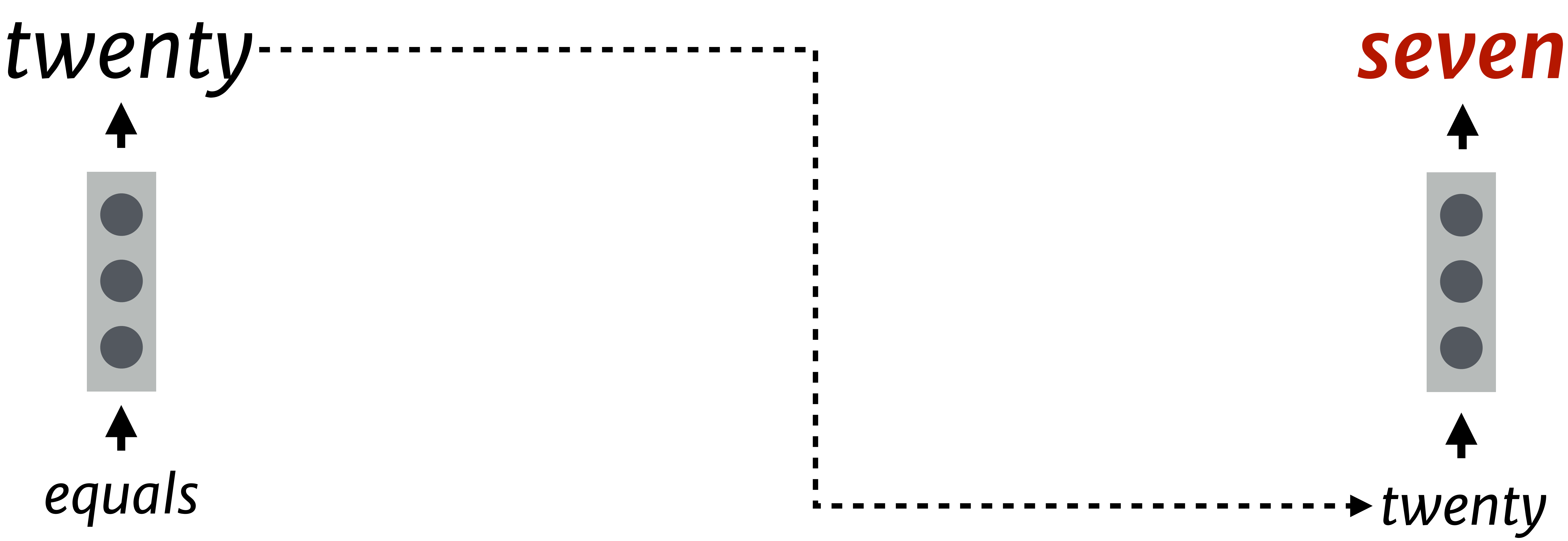
Answering math problems with LMs

$x_1 \sim p(x_1 \mid \ldots$ times three equals$)$



Answering math problems with LMs

$x_1 \sim p(x_1 \mid \ldots \text{ times three equals})$ $\quad$ $x_2 \sim p(x_2 \mid \ldots \text{ equals twenty})$



(don't try this at home)

A dataset of translated sentences

Caecilius est in horto. [SEP] Caecilius is in the garden.

Caecilius in horto sedet. [SEP] Caecilius sits in the garden.

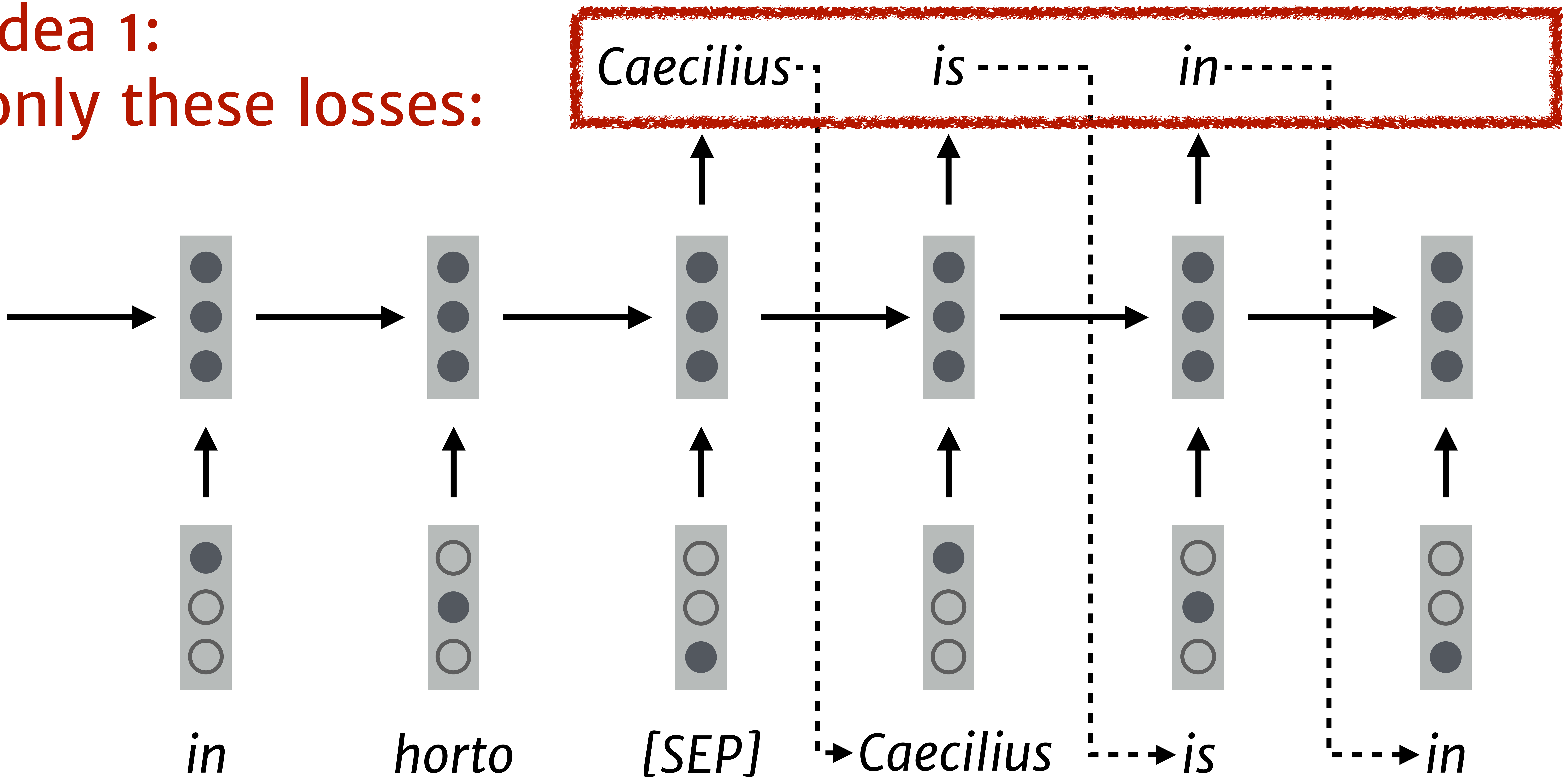
Grumio est in atrio. [SEP] Grumio is in the atrium.

Grumio in atrio laborat. [SEP] ???

(try this at home!)

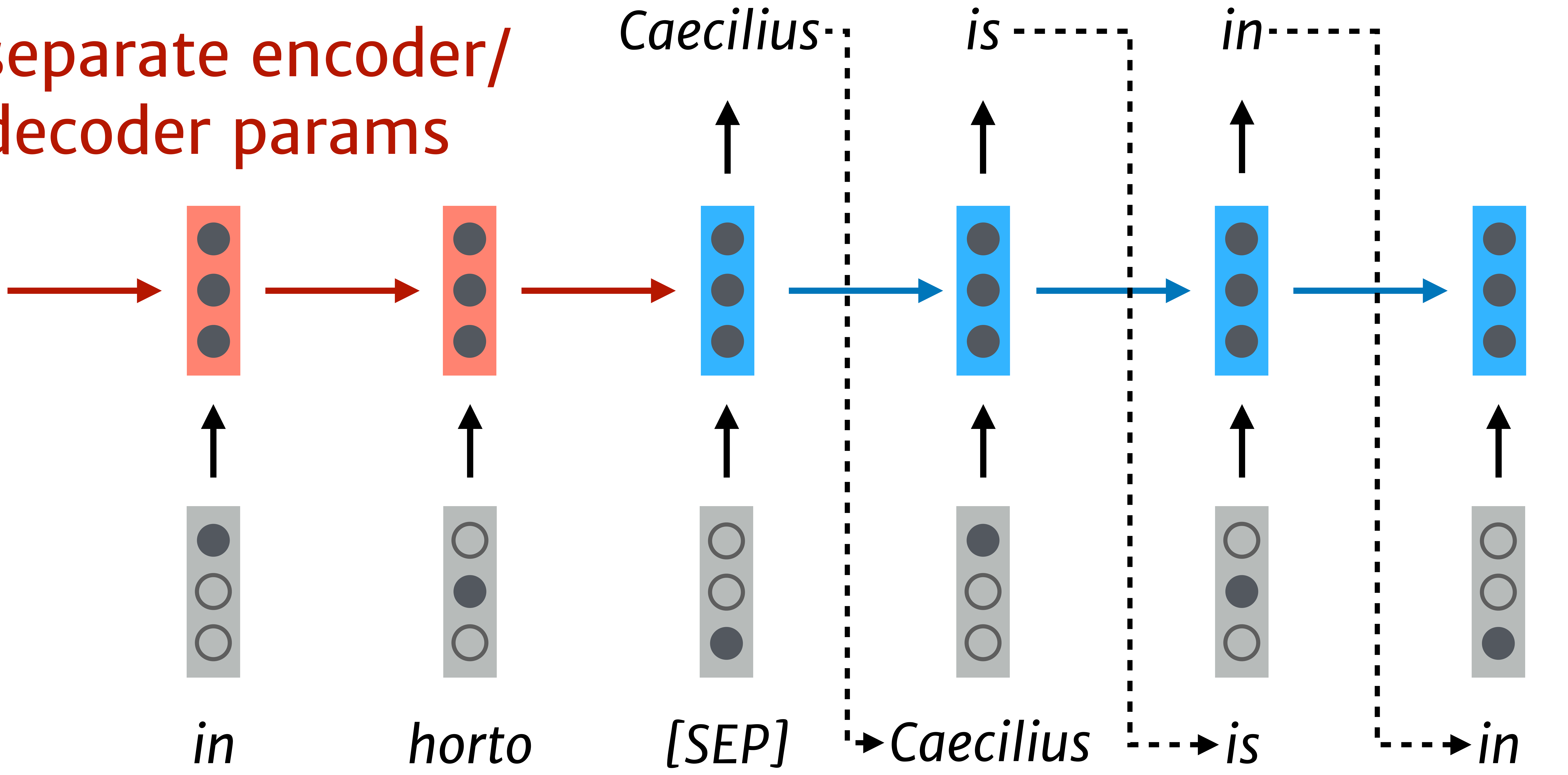
Sequence-**to-sequence** models

Idea 1:
only these losses:

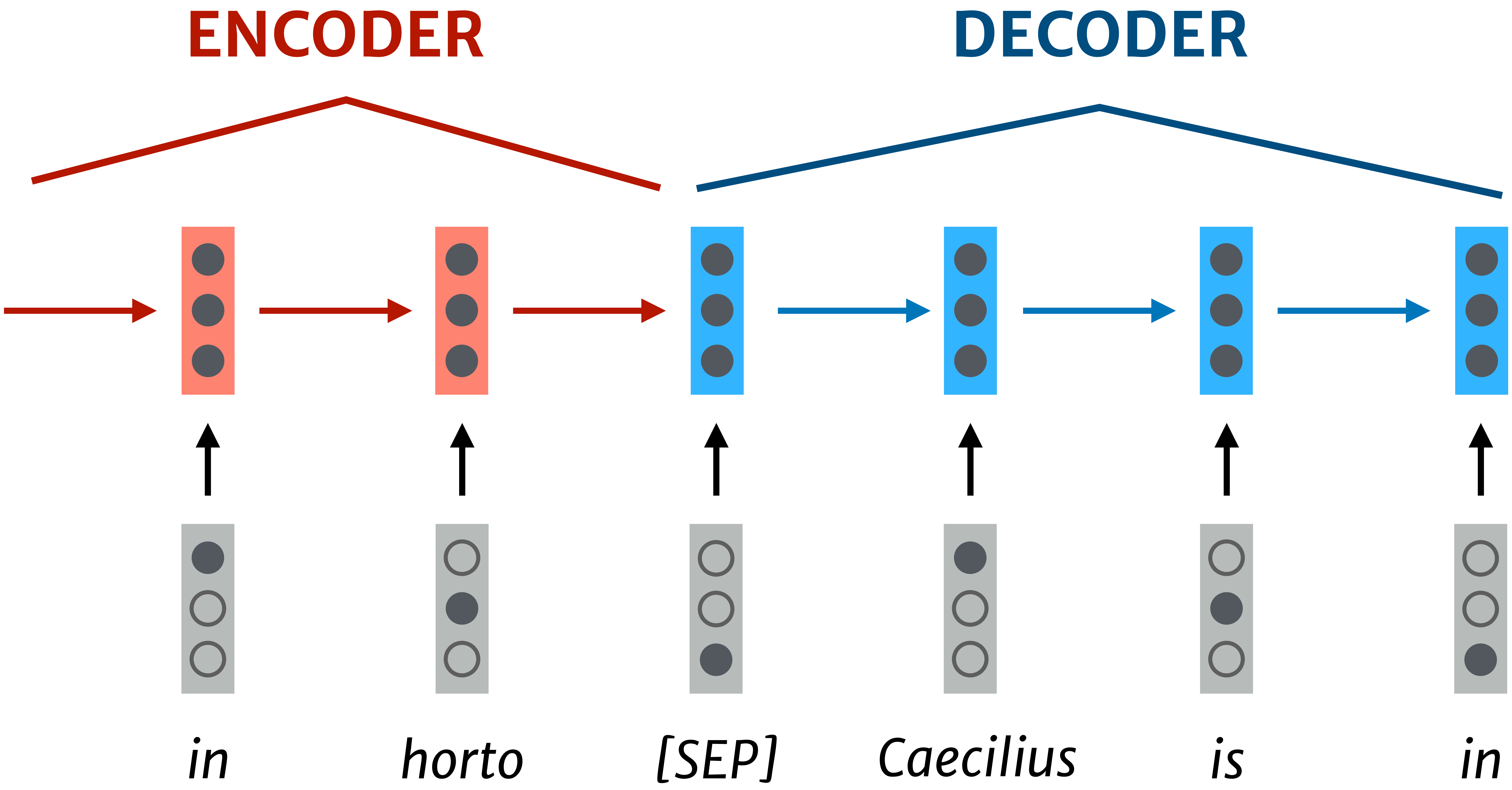


Sequence-**to-sequence** models

Idea 2:
separate encoder/
decoder params

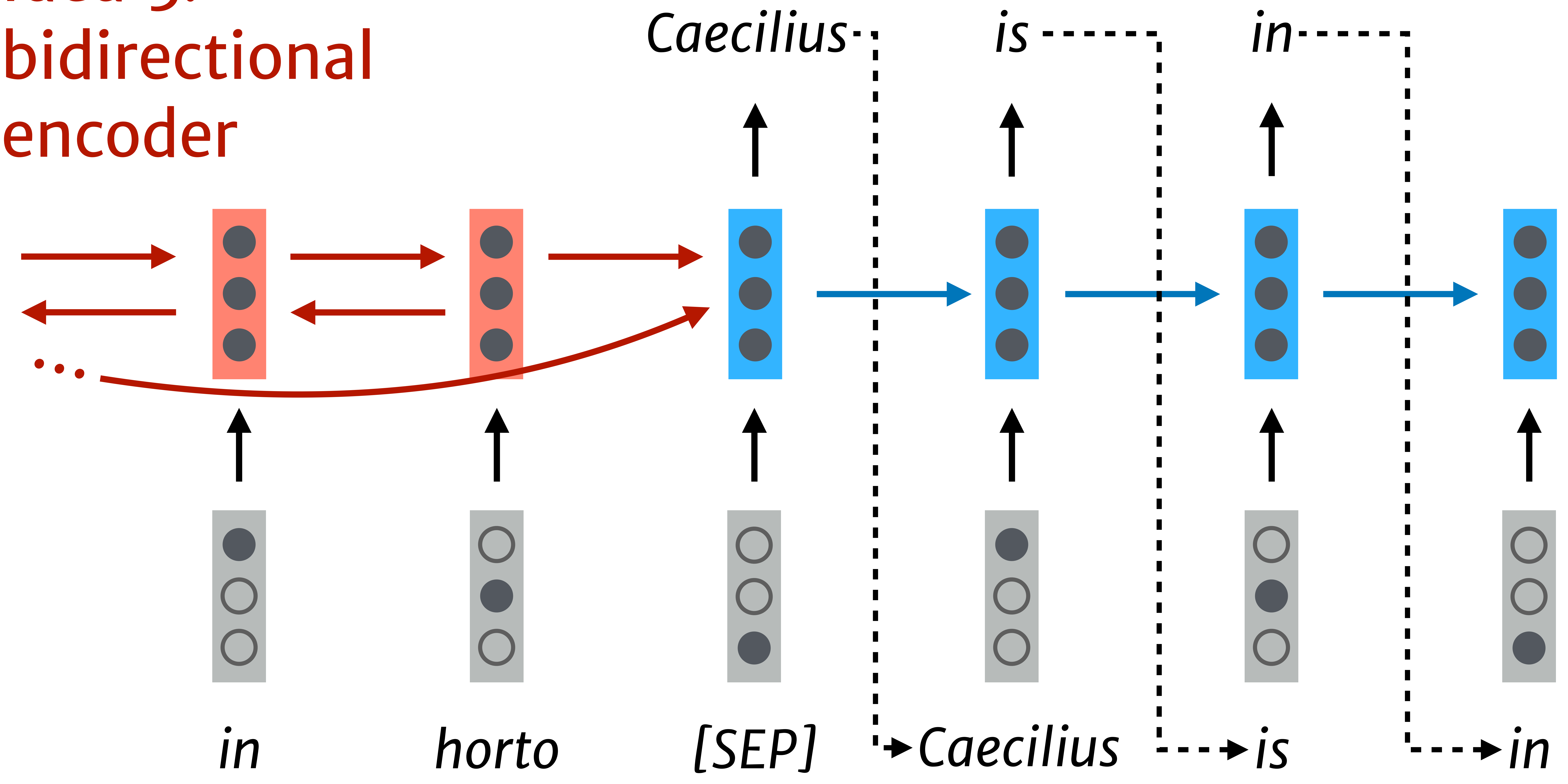


Sequence-**to-sequence** models

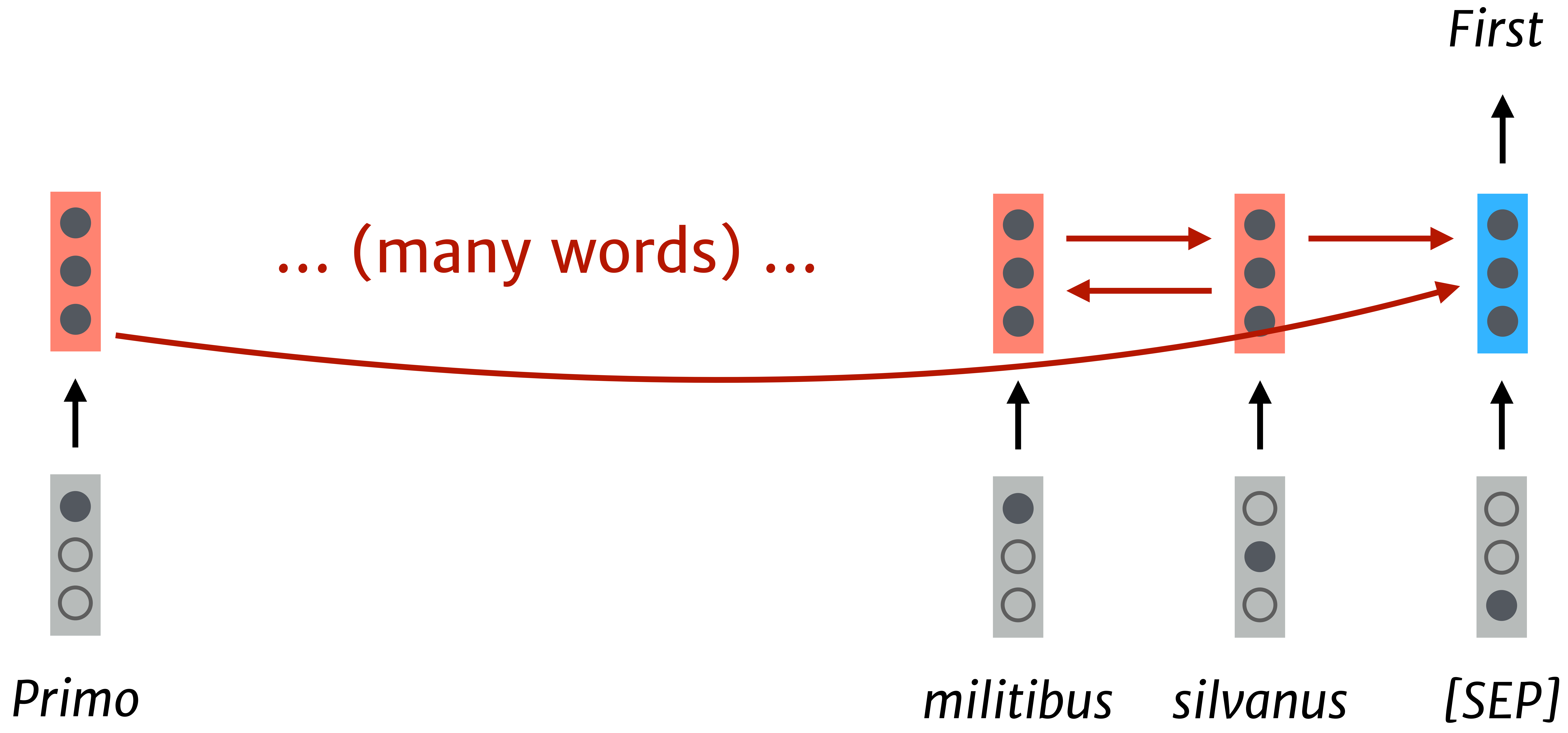


Sequence-**to-sequence** models

Idea 3: bidirectional encoder

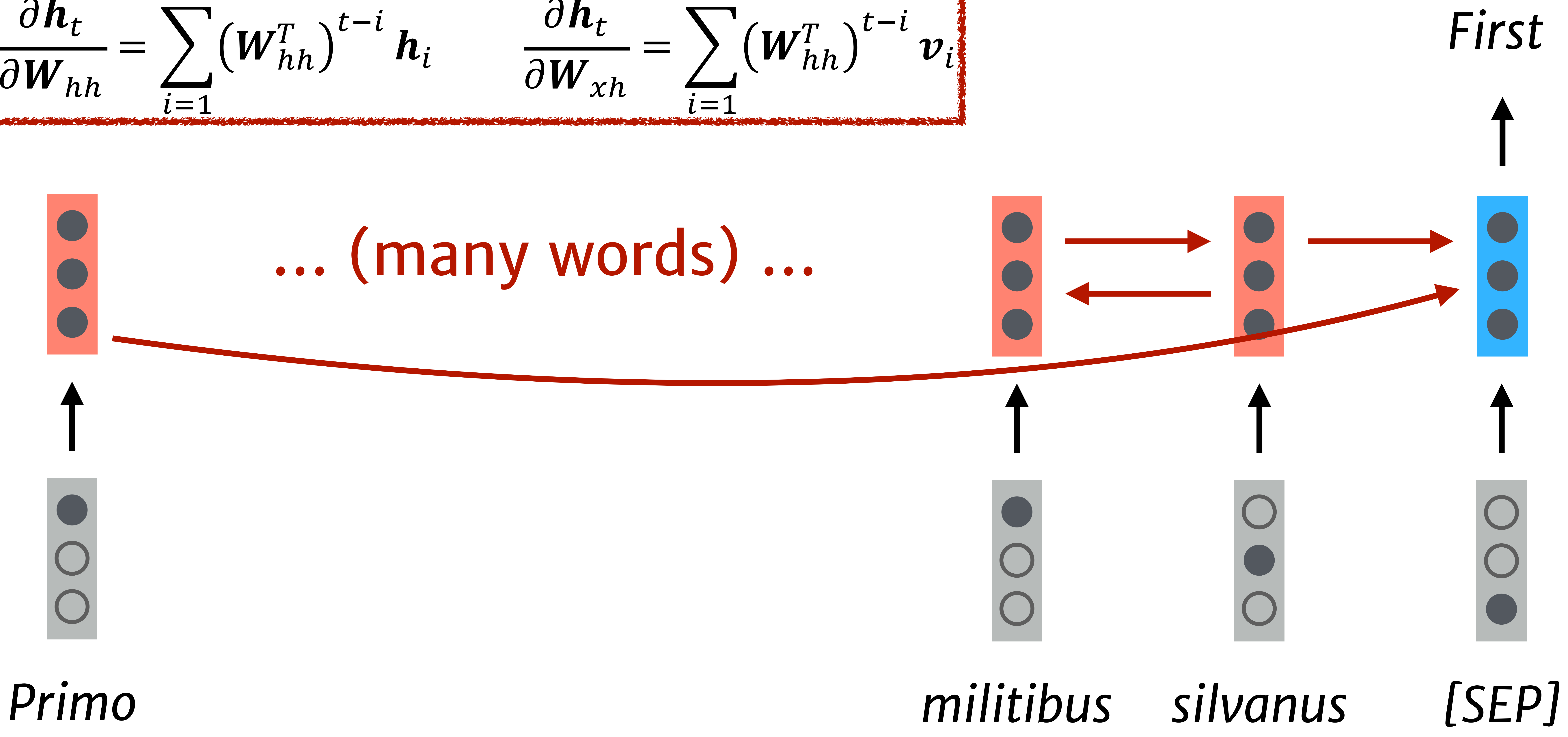


Revenge of the vanishing gradients



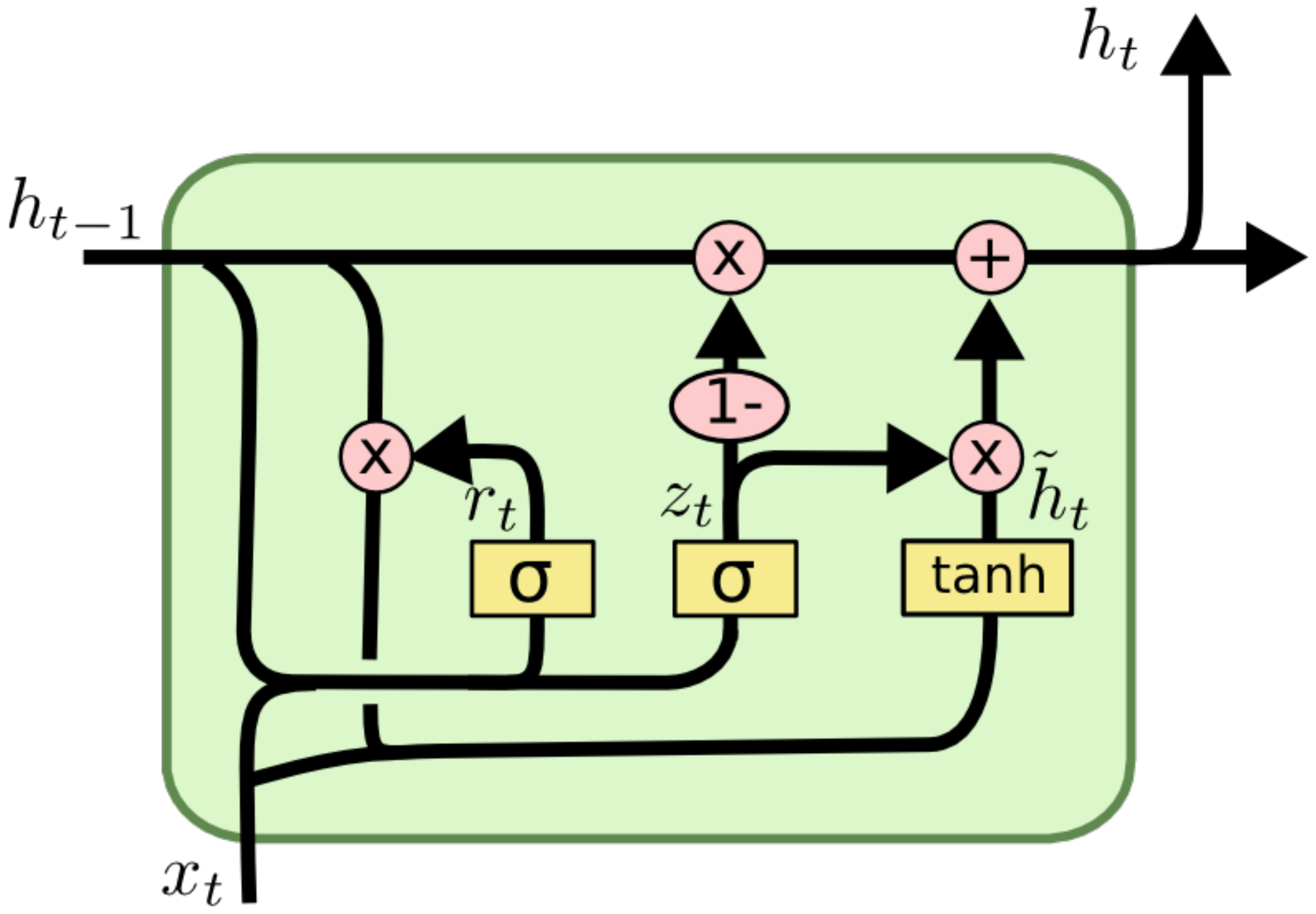
Revenge of the vanishing gradients

$$\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{W}_{hh}} = \sum_{i=1}^{t} \left(\boldsymbol{W}_{hh}^T\right)^{t-i} \boldsymbol{h}_i \qquad \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{W}_{xh}} = \sum_{i=1}^{t} \left(\boldsymbol{W}_{hh}^T\right)^{t-i} \boldsymbol{v}_i$$



Attention mechanisms

Gated Recurrent Units



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

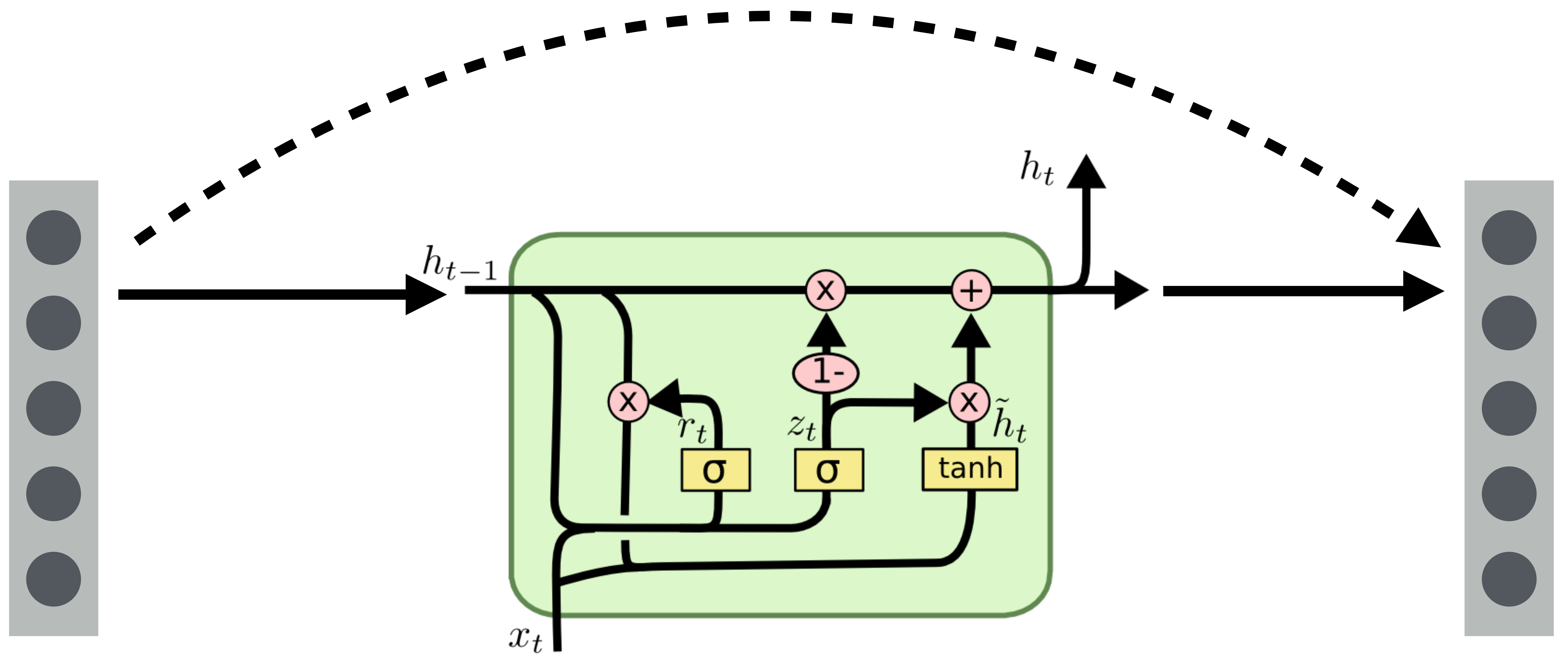
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

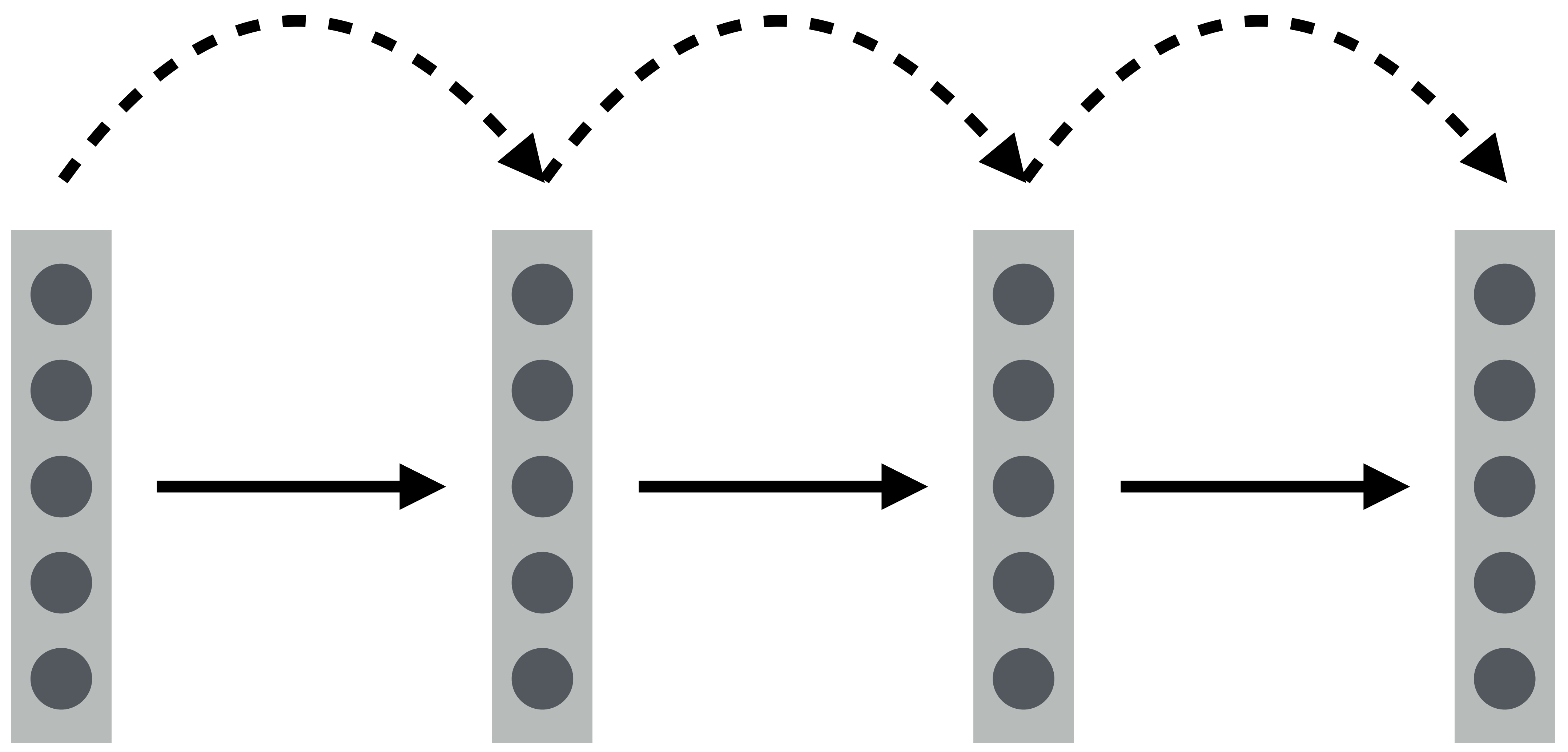
$$h_t = \boxed{(1 - z_t) * h_{t-1}} + z_t * \tilde{h}_t$$

[Image: Cristopher Olah]

Shortcuts

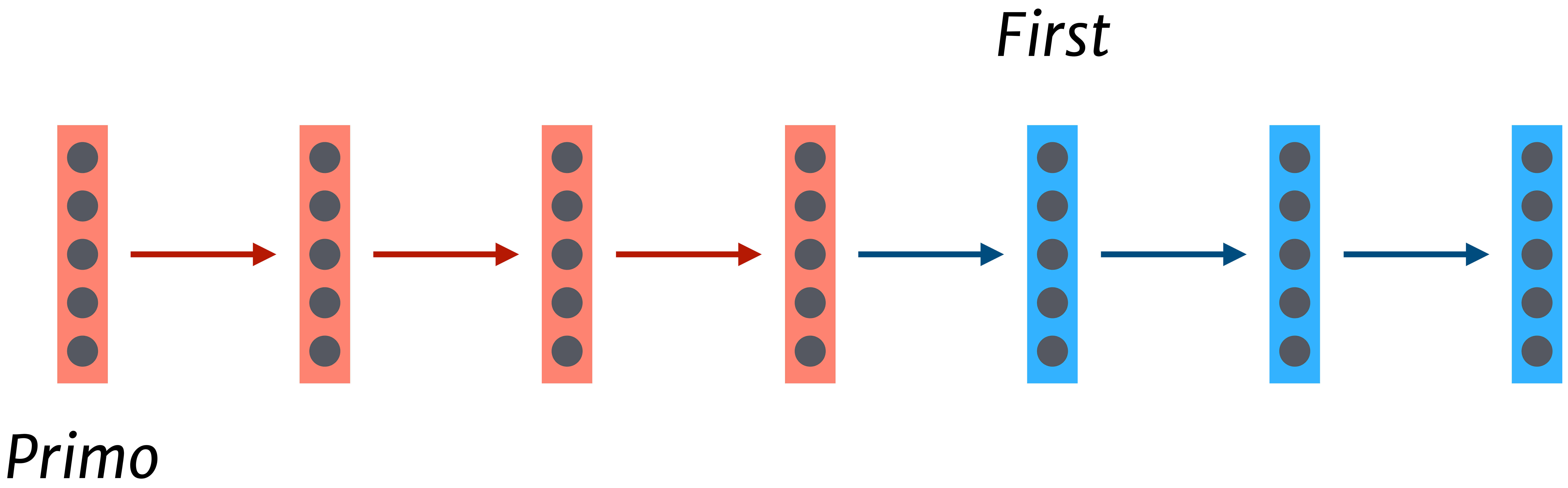


Shortcuts

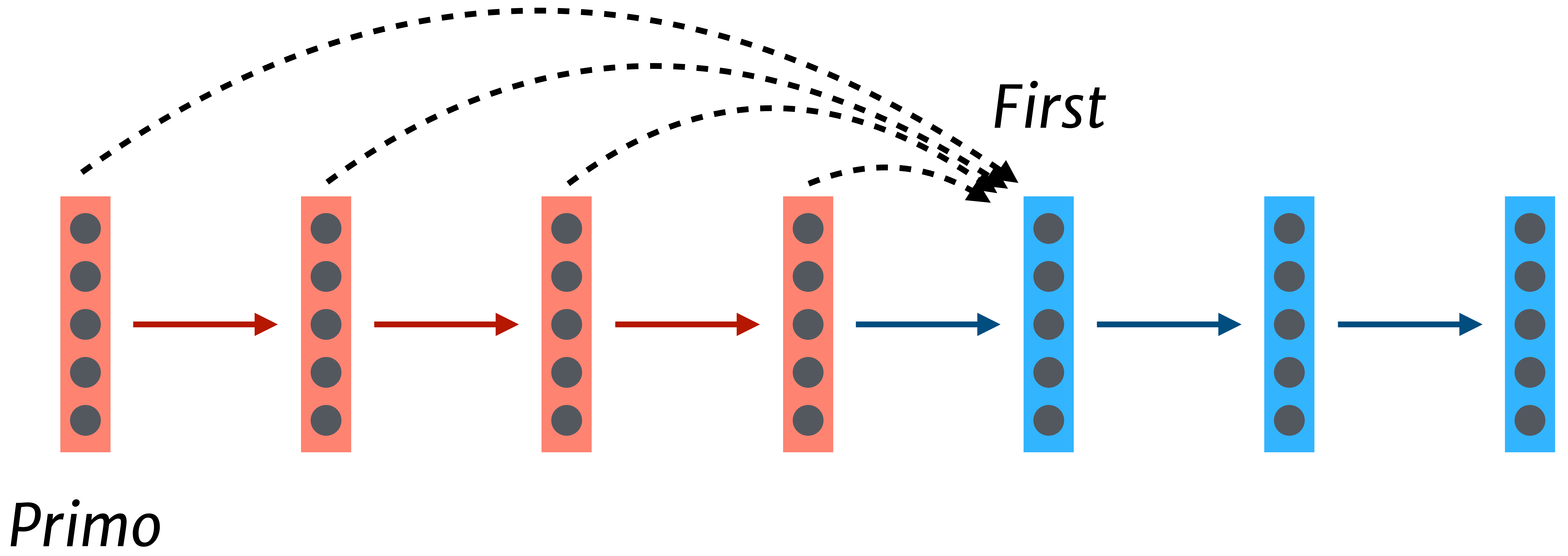


Direct "copying" between hidden states makes it easy to propagate information.

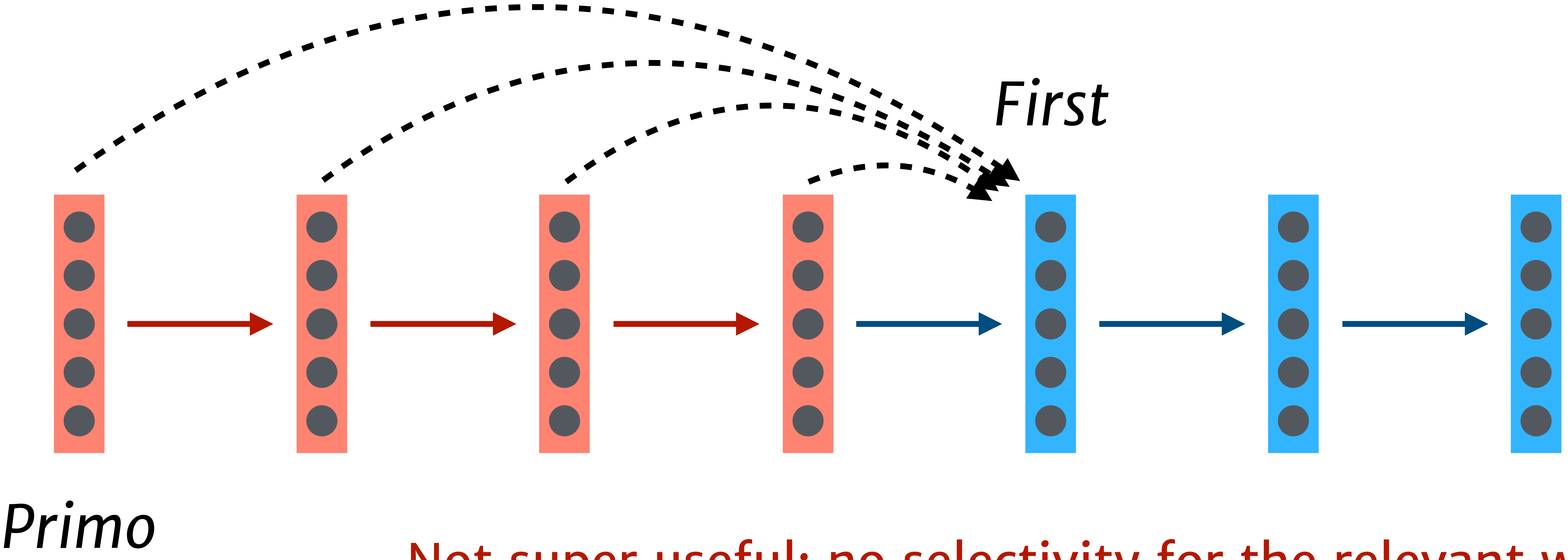
Can we go farther?



Can we go farther?

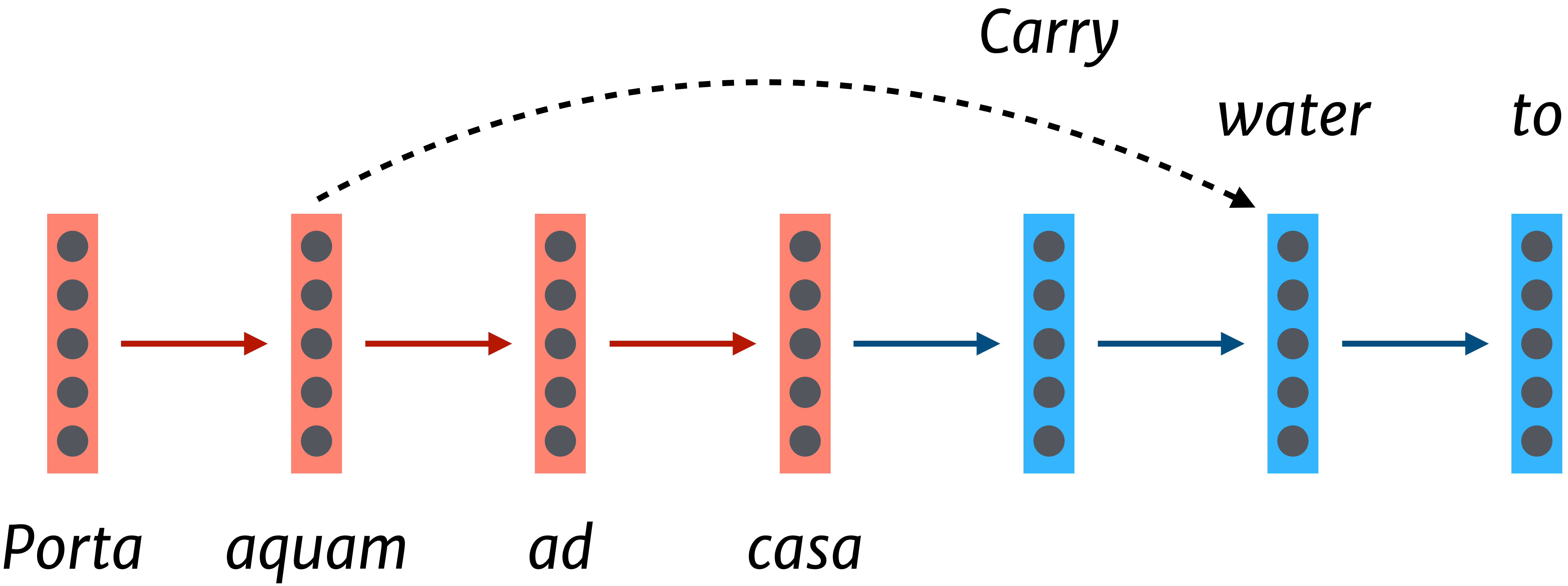


Can we go farther?

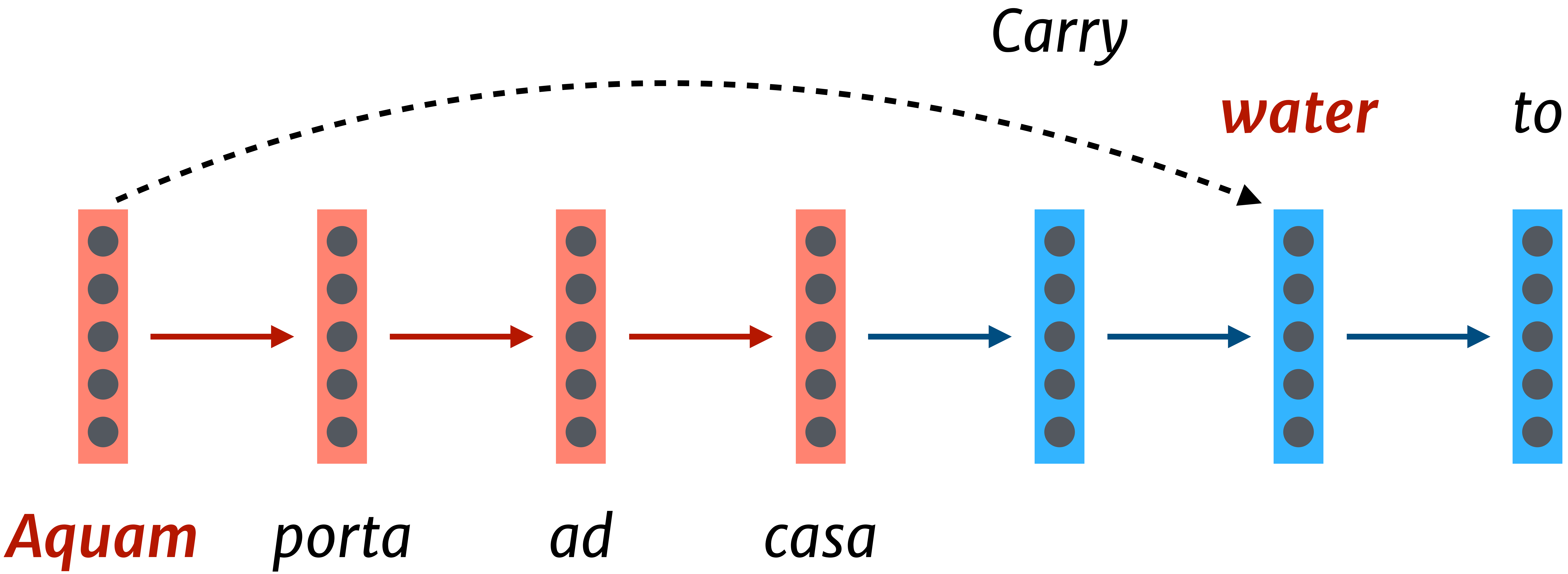


Not super useful: no selectivity for the relevant word (since we don't know which word is relevant when we add connections)

Can we hard-code connections?

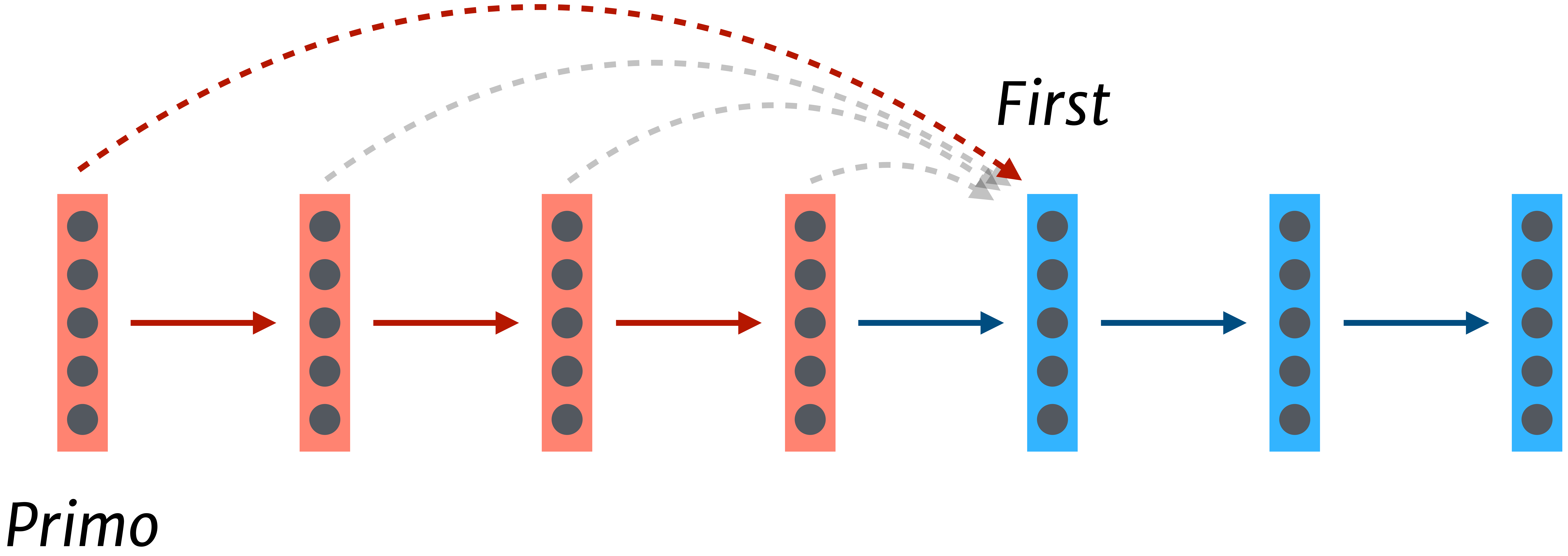


Can we hard-code connections?



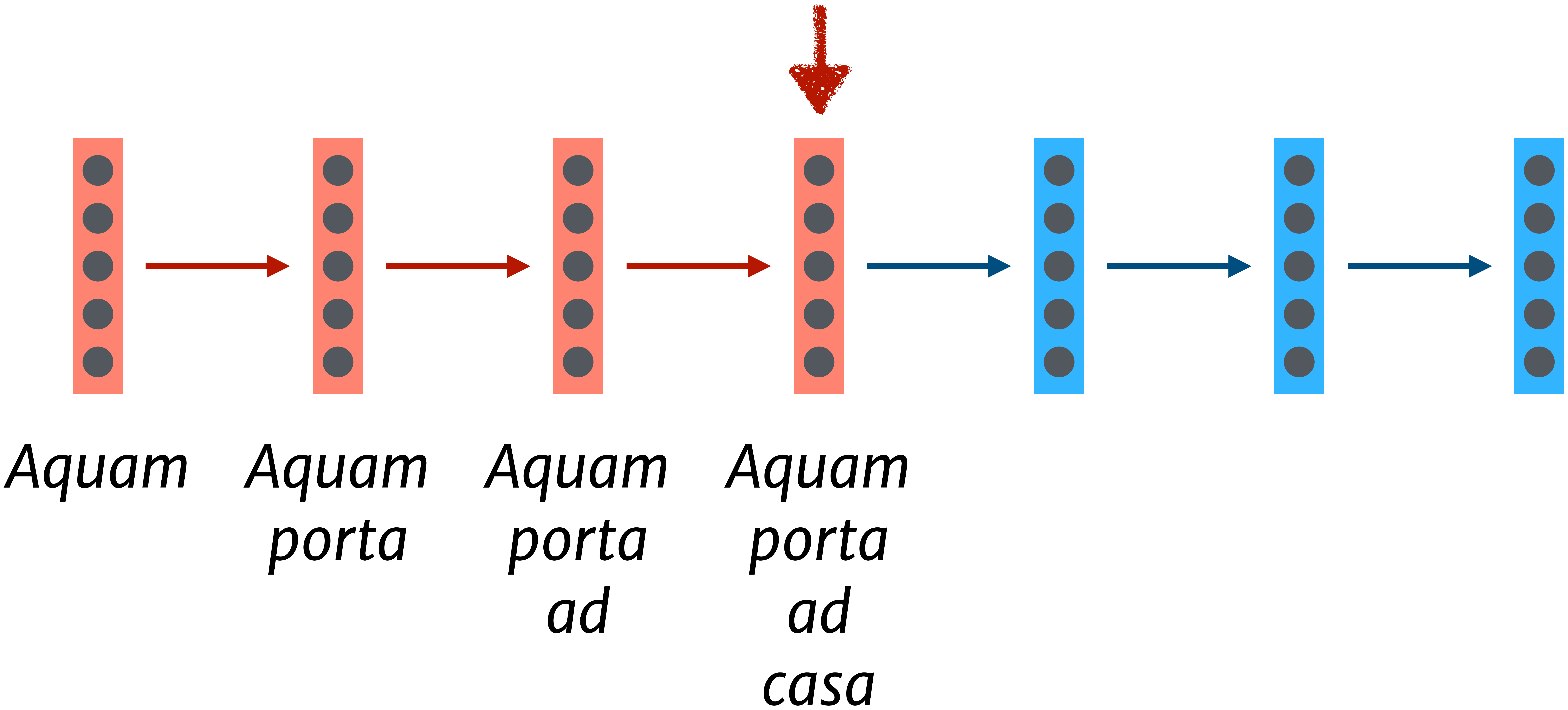
Words aren't one-to-one (and order can change!)

Can we **learn** connections?



Sentence representations

This vector represents the whole sentence!



[Ray Mooney, ca. 2014]

You can't cram the meaning of a whole %&!$# sentence into a single $&!# vector!*

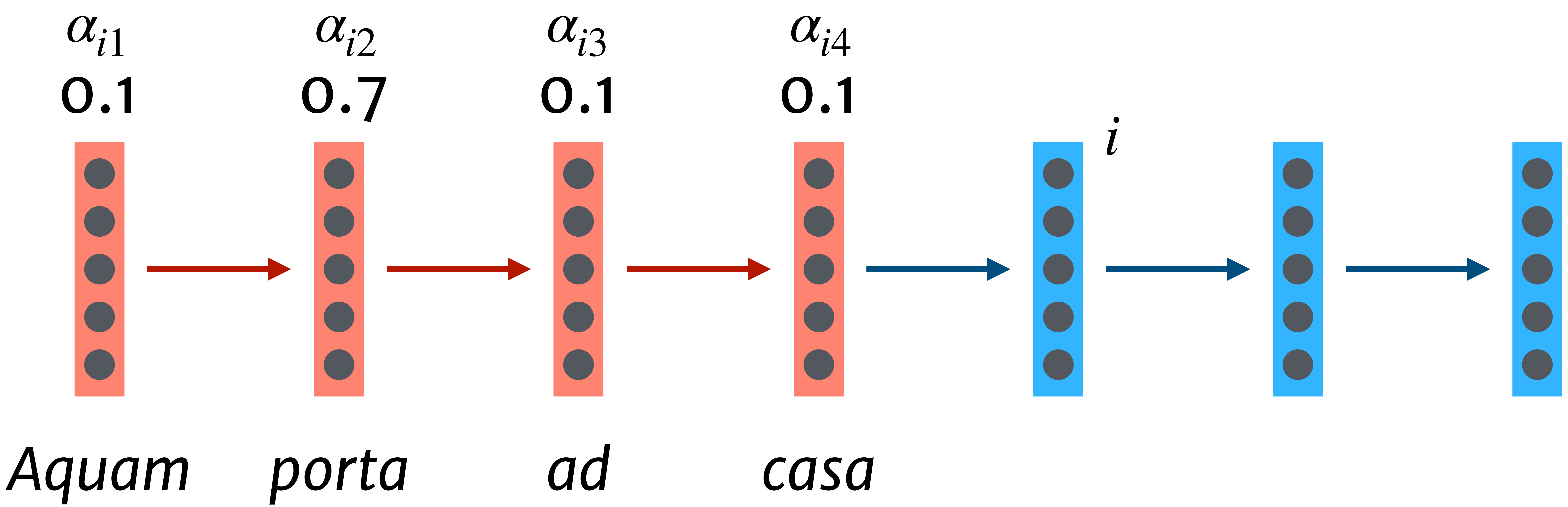
Actually you can!
(But you usually shouldn't.)

[Ray Mooney, ca. 2014]

Attention mechanisms

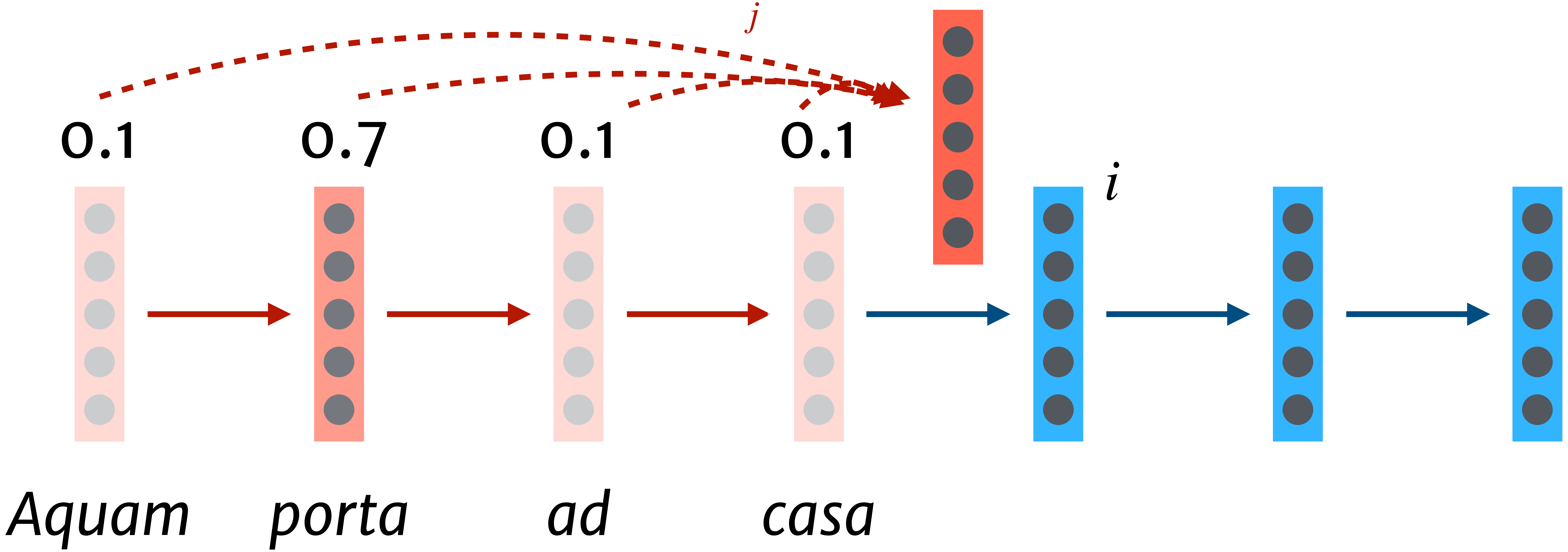
1. When predicting output i, assign a weight α_{ij} to each encoder state h_j

(weights sum to 1)



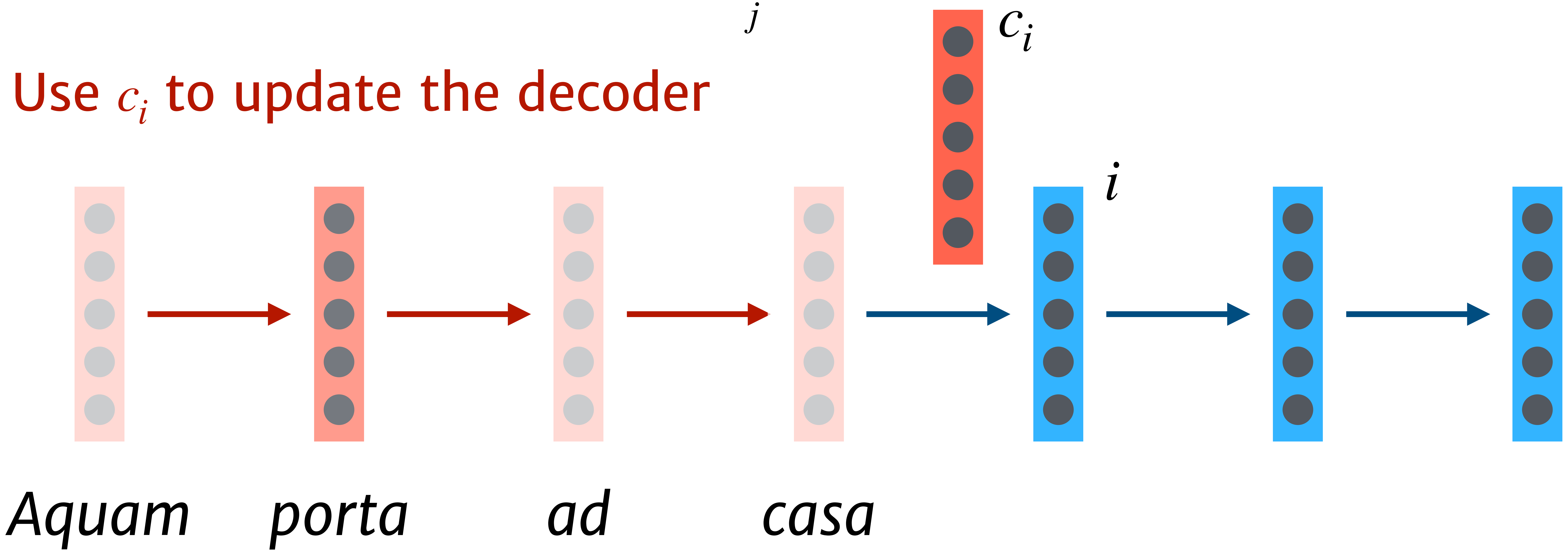
Attention mechanisms

1. When predicting output i, assign a weight α_{ij} to each encoder state h_j
2. Compute a pooled input $c_i = \sum_j \alpha_{ij} h_j$



Attention mechanisms

1. When predicting output i, assign a weight α_{ij} to each encoder state h_j
2. Compute a pooled input $c_i = \sum_j \alpha_{ij} h_j$
3. Use c_i to update the decoder



Design decision: how to compute α_{ij}?

1. When predicting output i, assign a weight α_{ij} to each encoder state h_j

$$e_{ij} = \tanh(W[h_i, h_j])$$

[Bahdanau 2014]

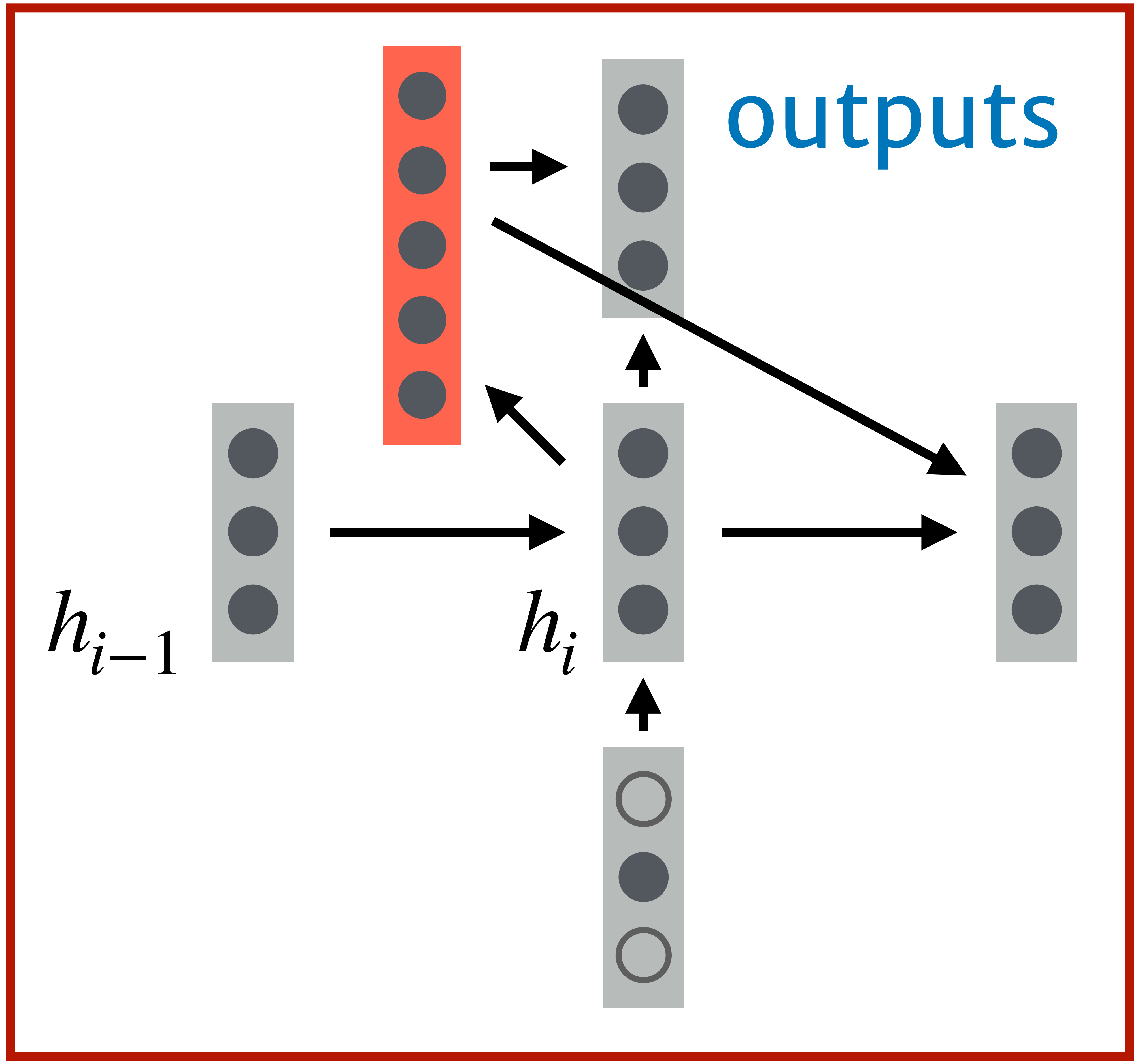
$$\boxed{e_{ij} = h_i^\top W h_j}$$

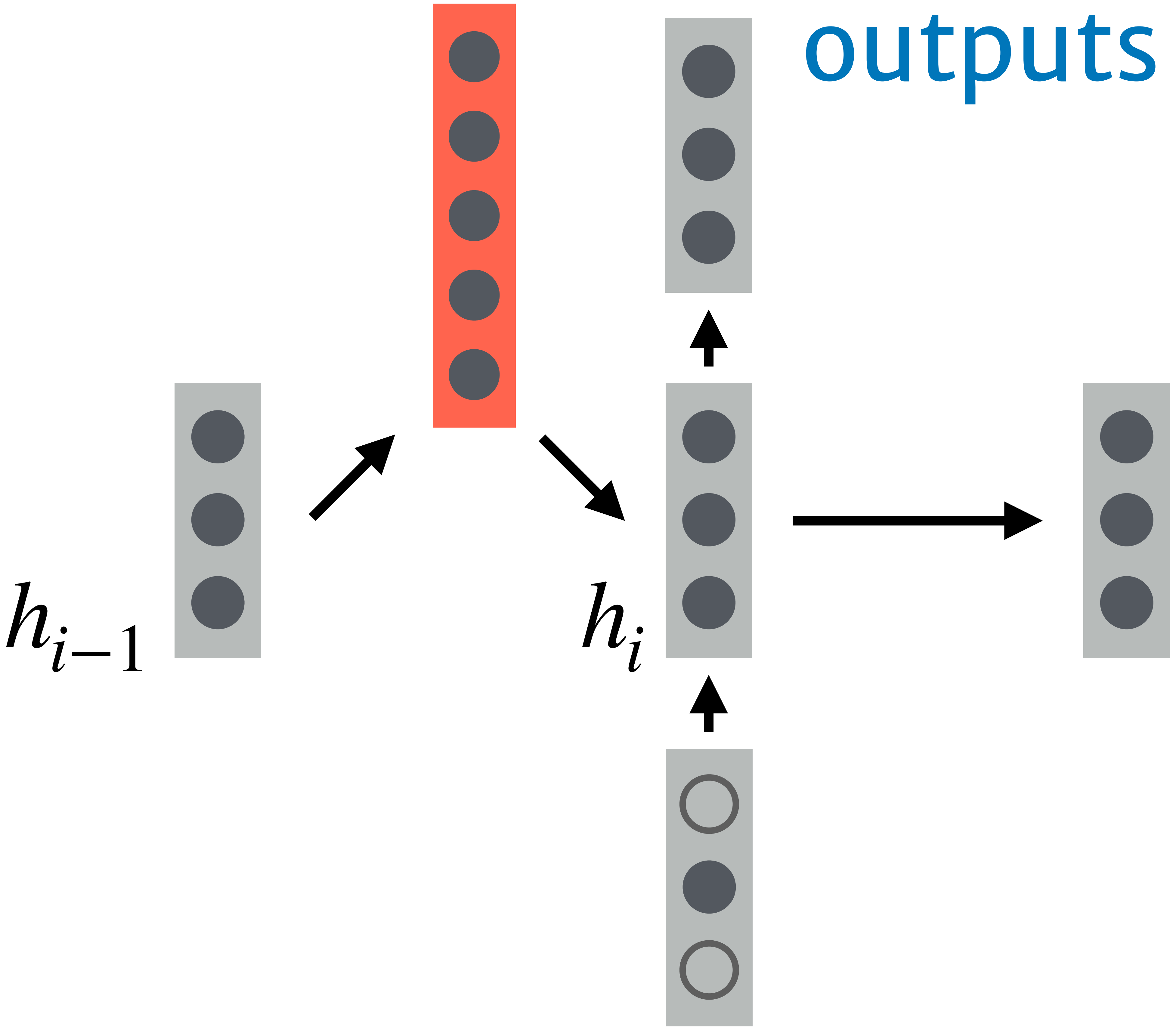
[Luong 2015]

$$\alpha_{i:} = \mathrm{softmax}(e_{i:})$$

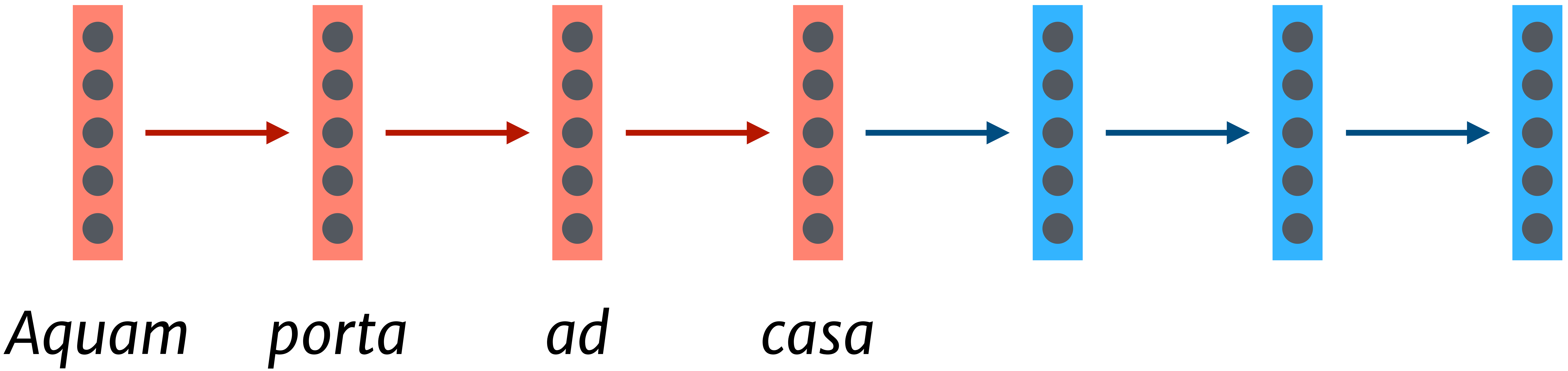
Design decision: how to use c_i?

3. Use c_i to update the decoder



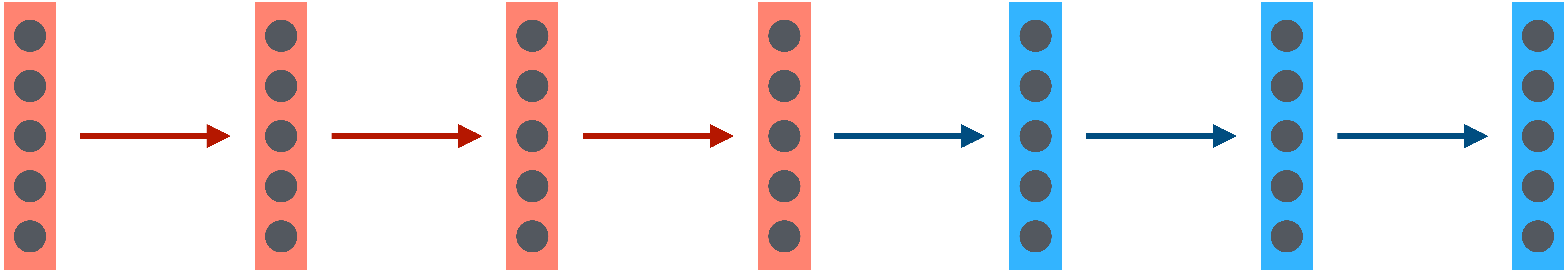


Why does this work?



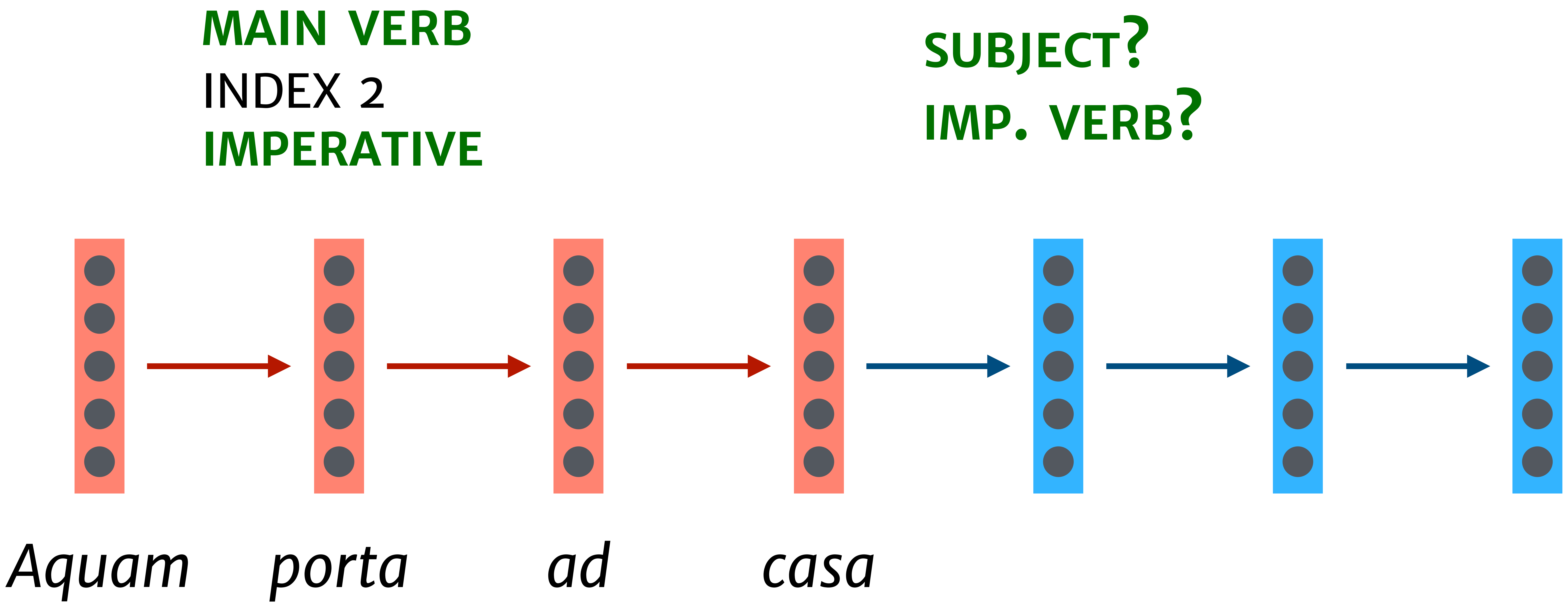
Why does this work?

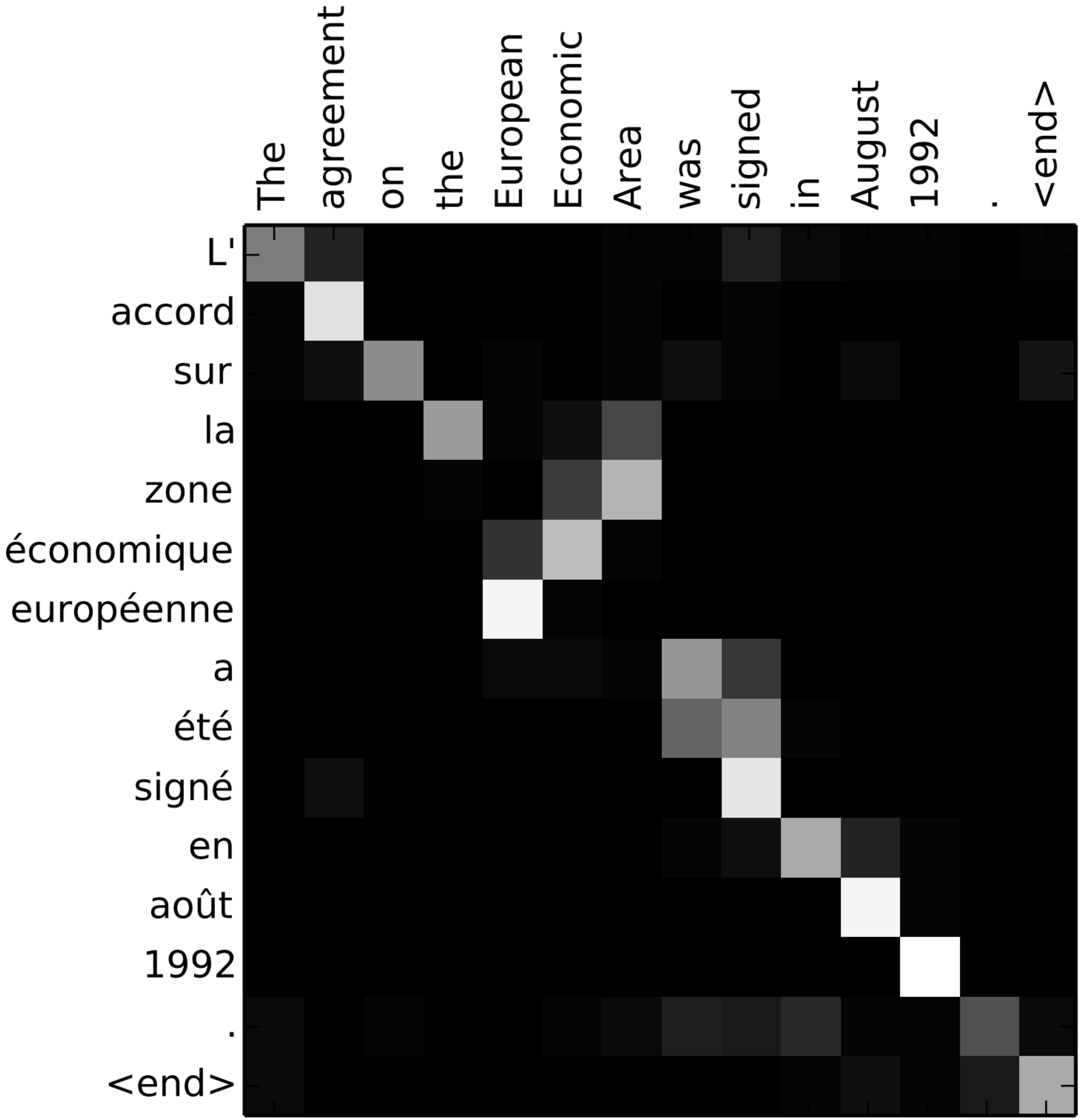
MAIN VERB
INDEX 2
IMPERATIVE



Aquam *porta* *ad* *casa*

Why does this work?





[Example from Greg Durrett]

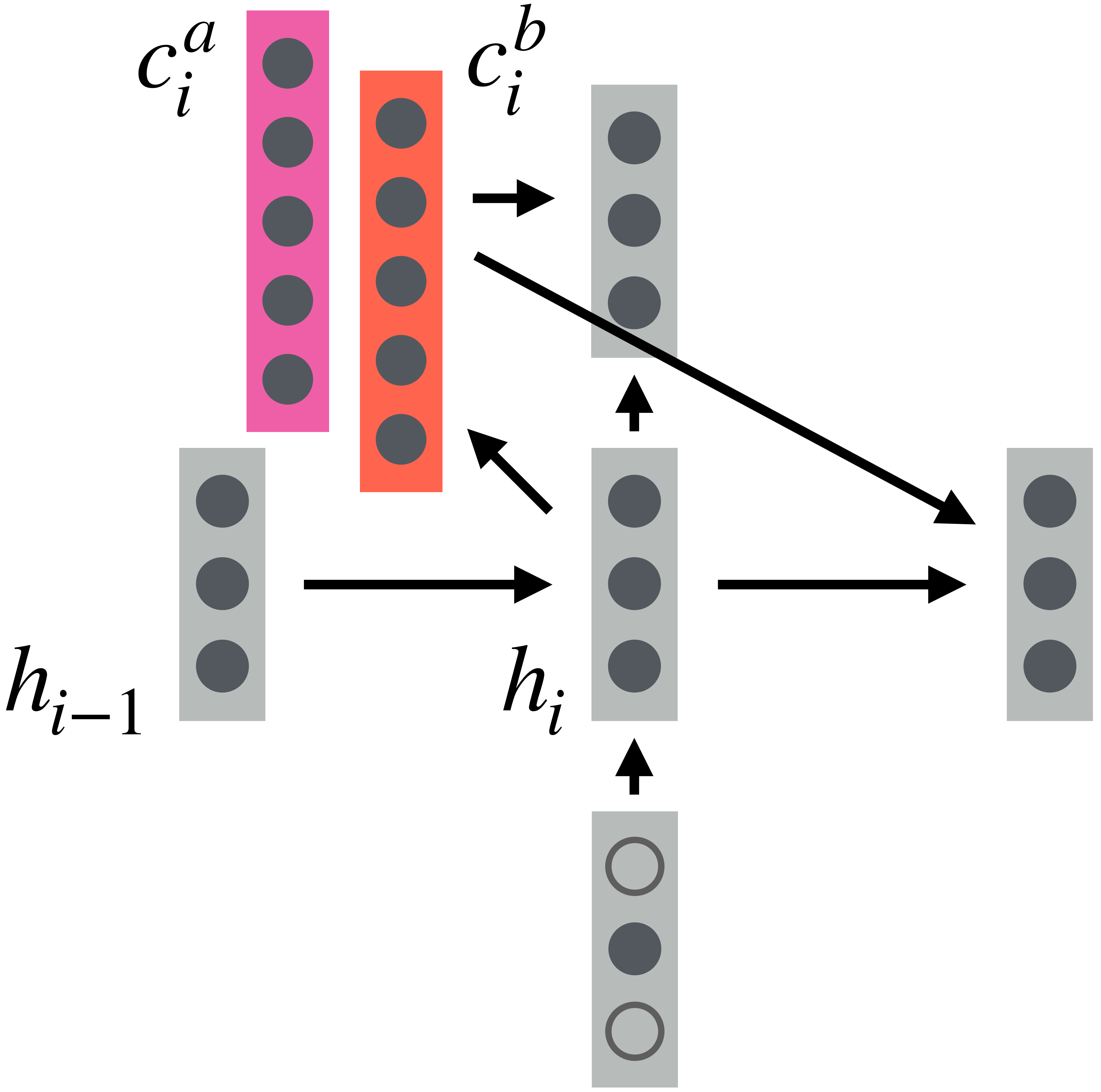
Multi-headed attention

Look two places at once!

$$e_{ij}^{a} = h_i^\top W_a h_j$$

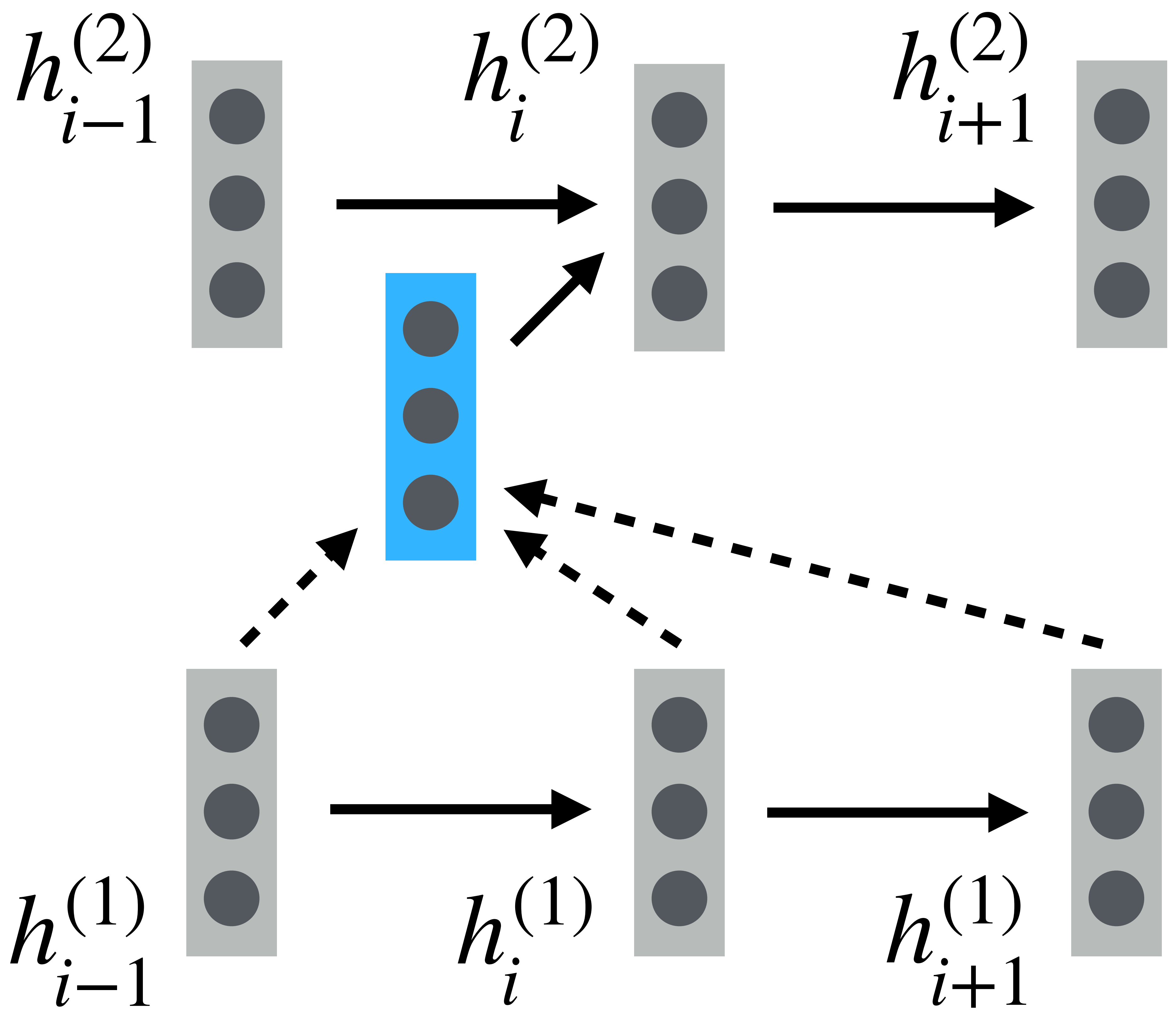
$$e_{ij}^{b} = h_i^\top W_b h_j$$

etc.

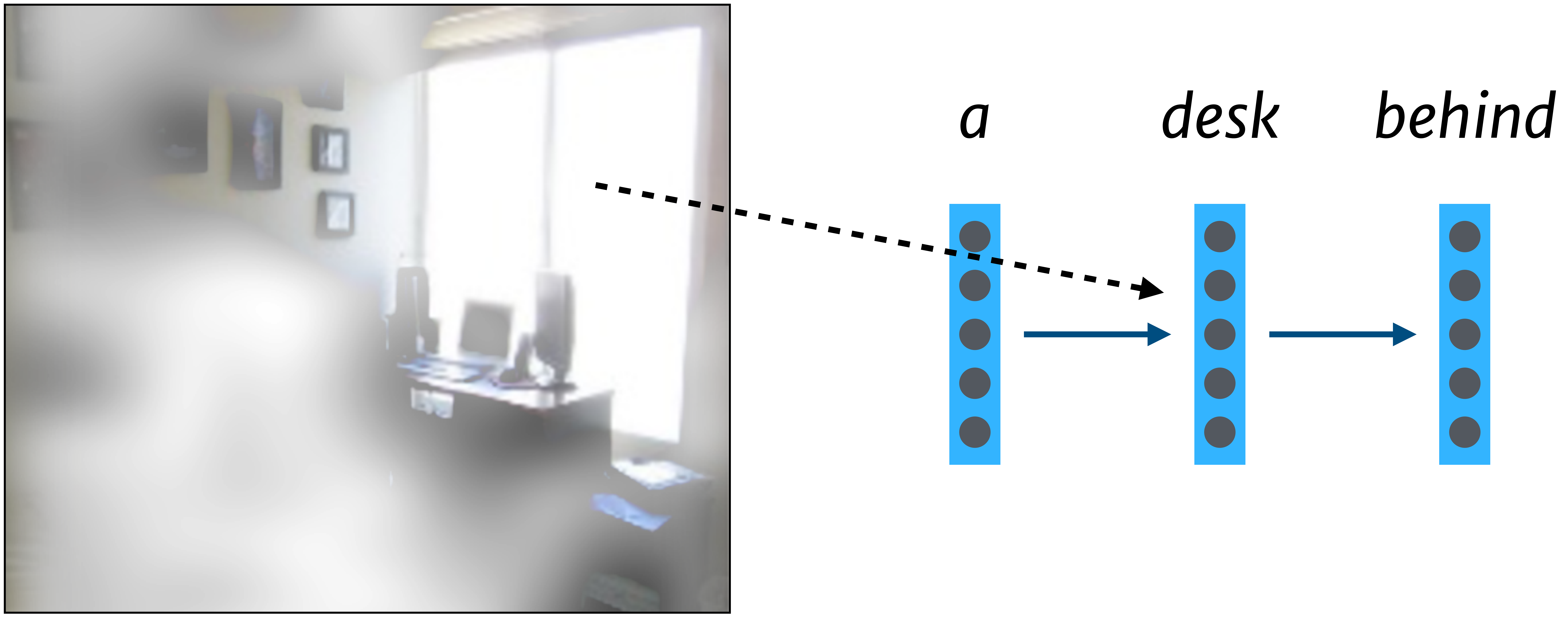


Self-attention

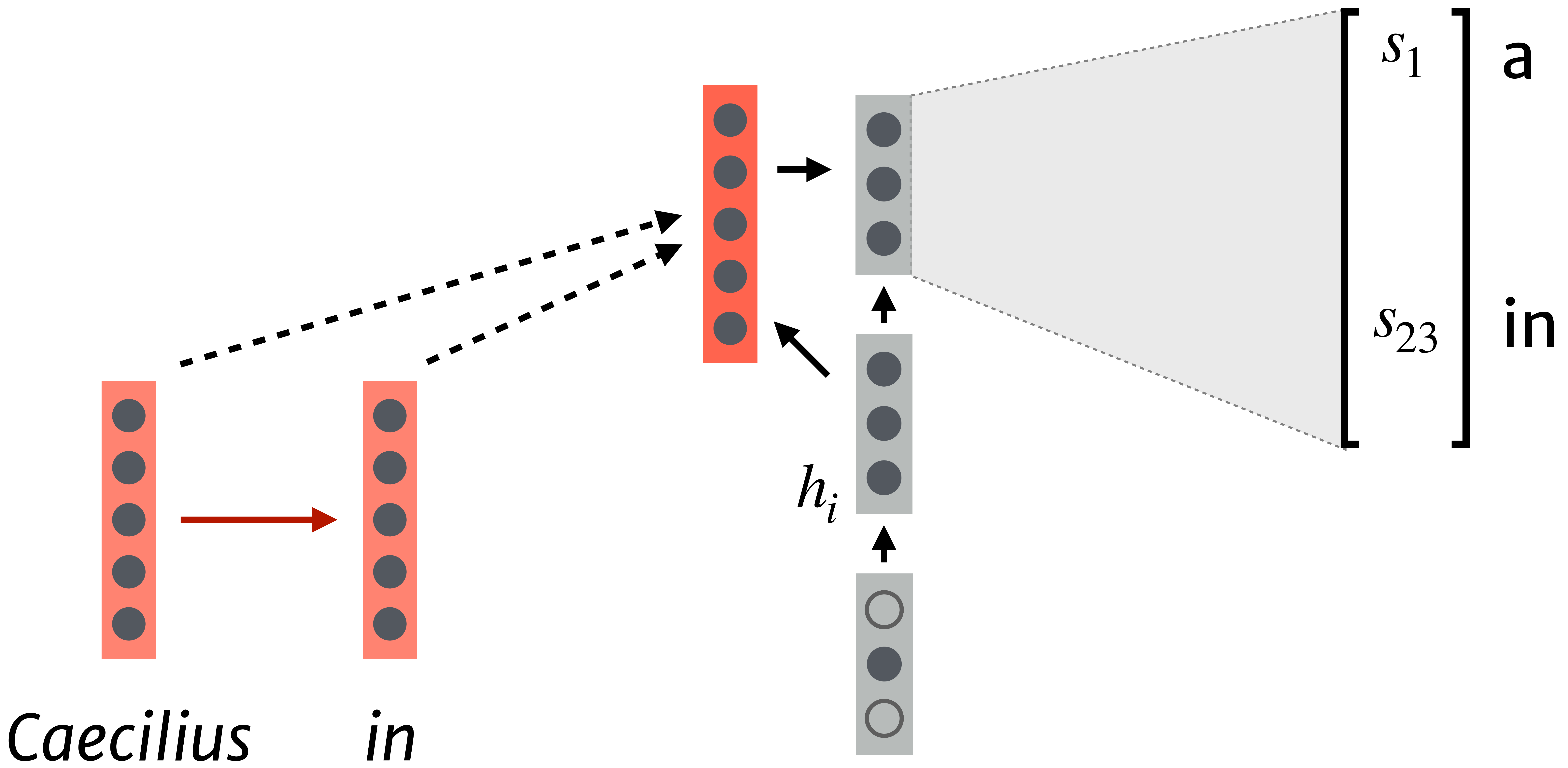
Attention to lower RNN layers (instead of decoder → encoder)



Non-textual attention

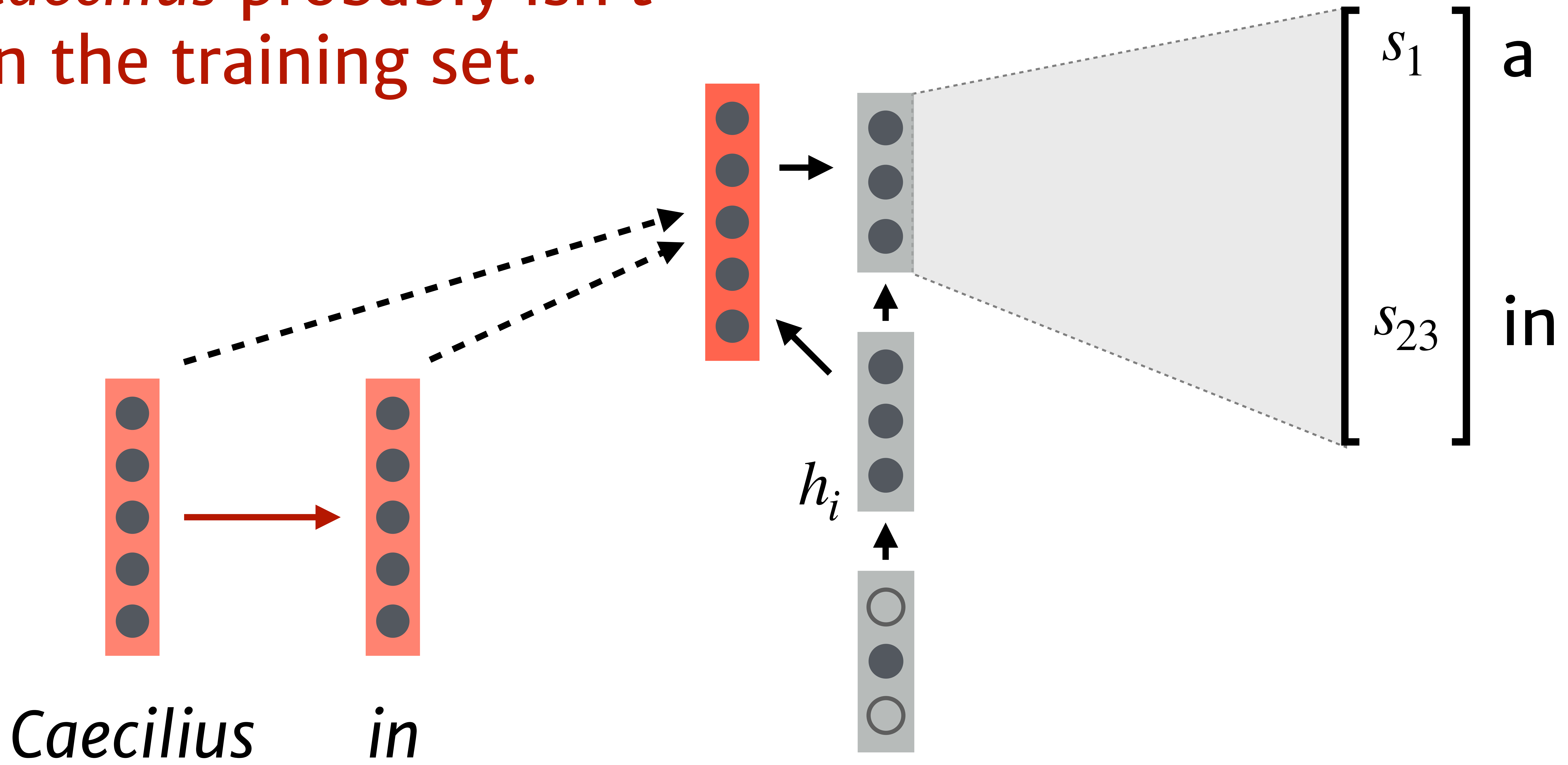


Copying



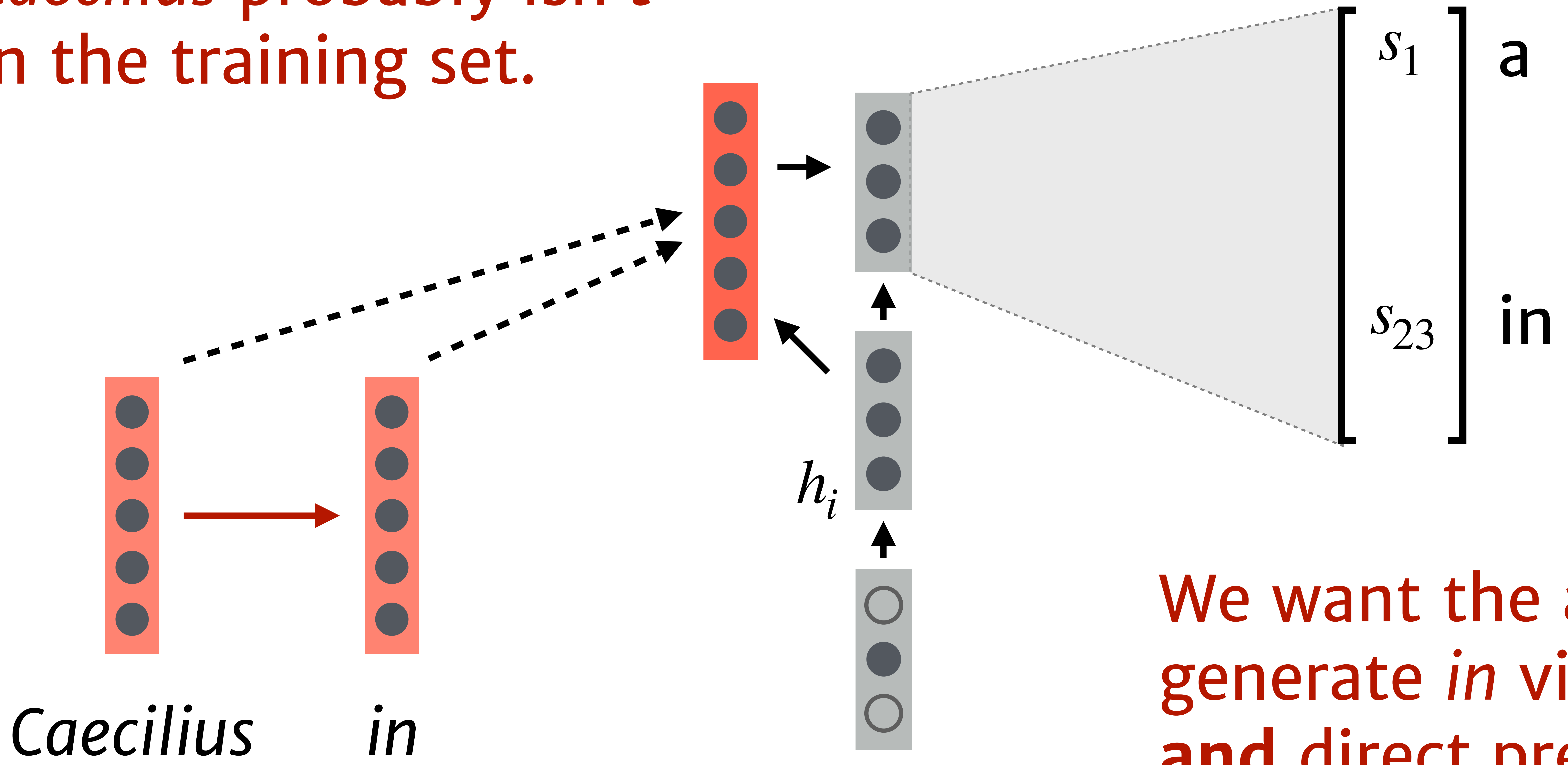
Copying

Caecilius probably isn't in the training set.



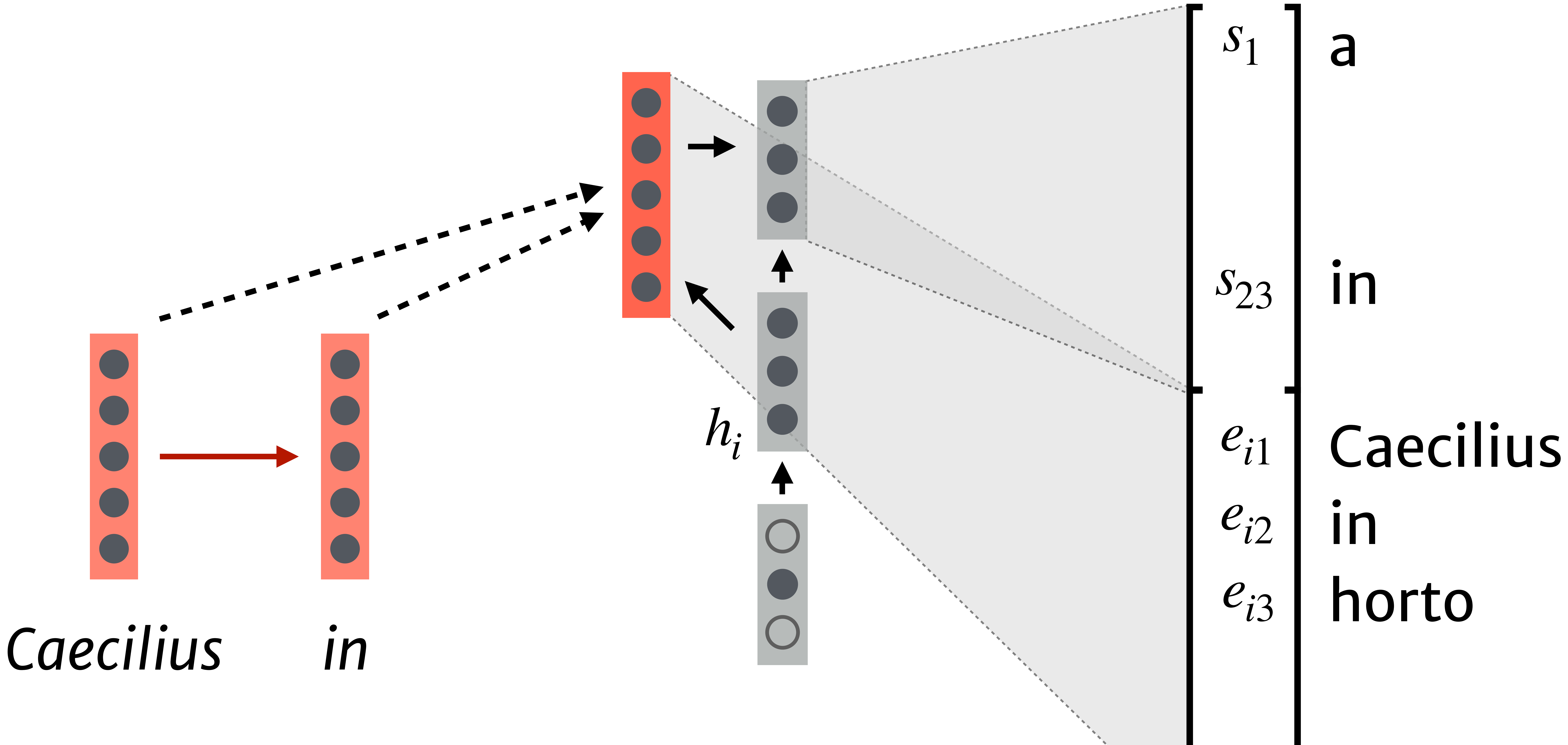
Copying

Caecilius probably isn't in the training set.



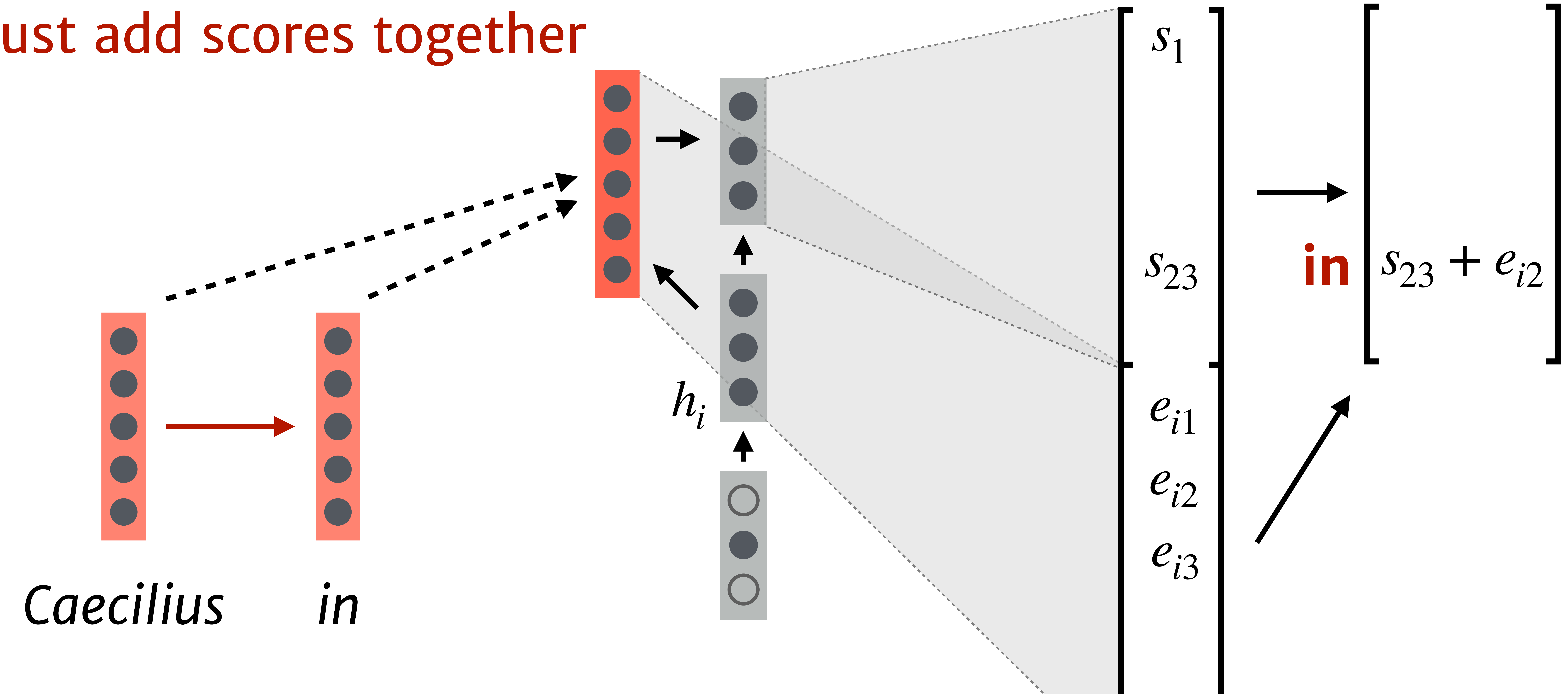
We want the ability to generate *in* via copying **and** direct prediction.

Copying



Copying

In is double-counted:
just add scores together



Hard attention

1. When predicting output i, assign a weight α_{ij} to each encoder state h_j

$$e_{ij} = \tanh(W[h_i, h_j])$$

[Bahdanau 2014]

$$e_{ij} = h_i^\top W h_j$$

[Luong 2015]

attention $\alpha_i = \operatorname{argmax}(e_{i:})$

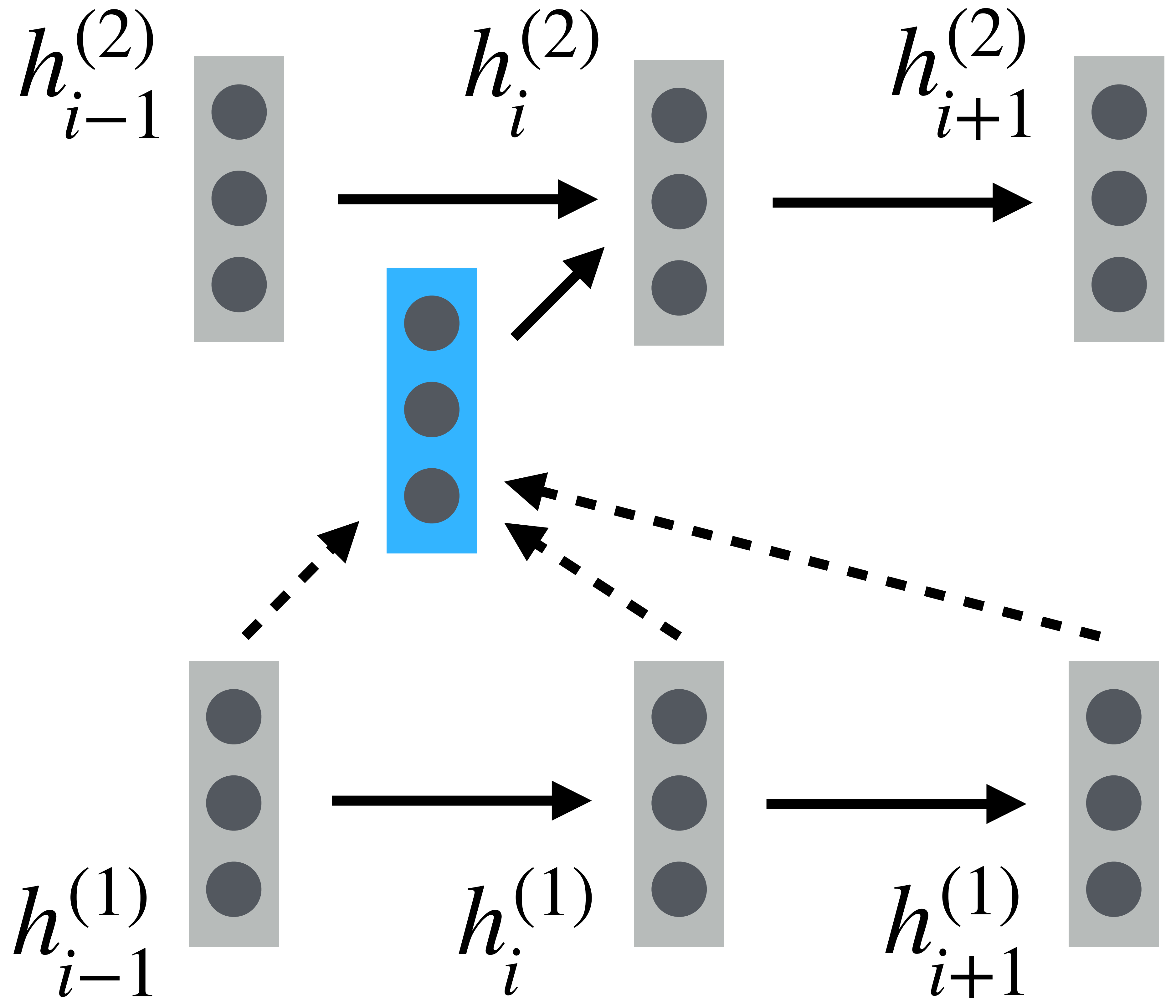
context repr $c_i = h_{\alpha_i}$

nondifferentiable!

but sometimes better generalization

(now you know how to build anything)

Self-attention revisited



Next class: transformers