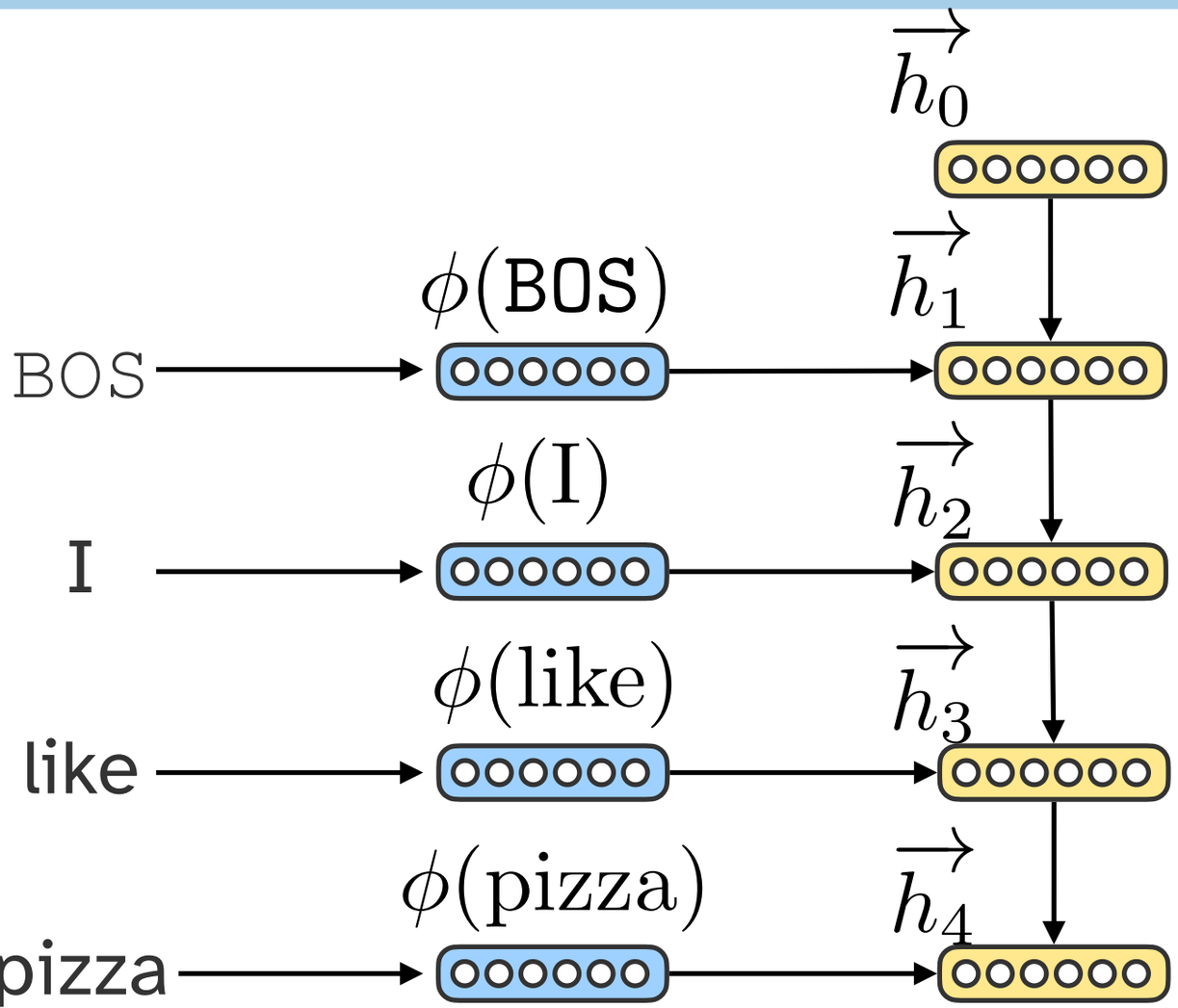
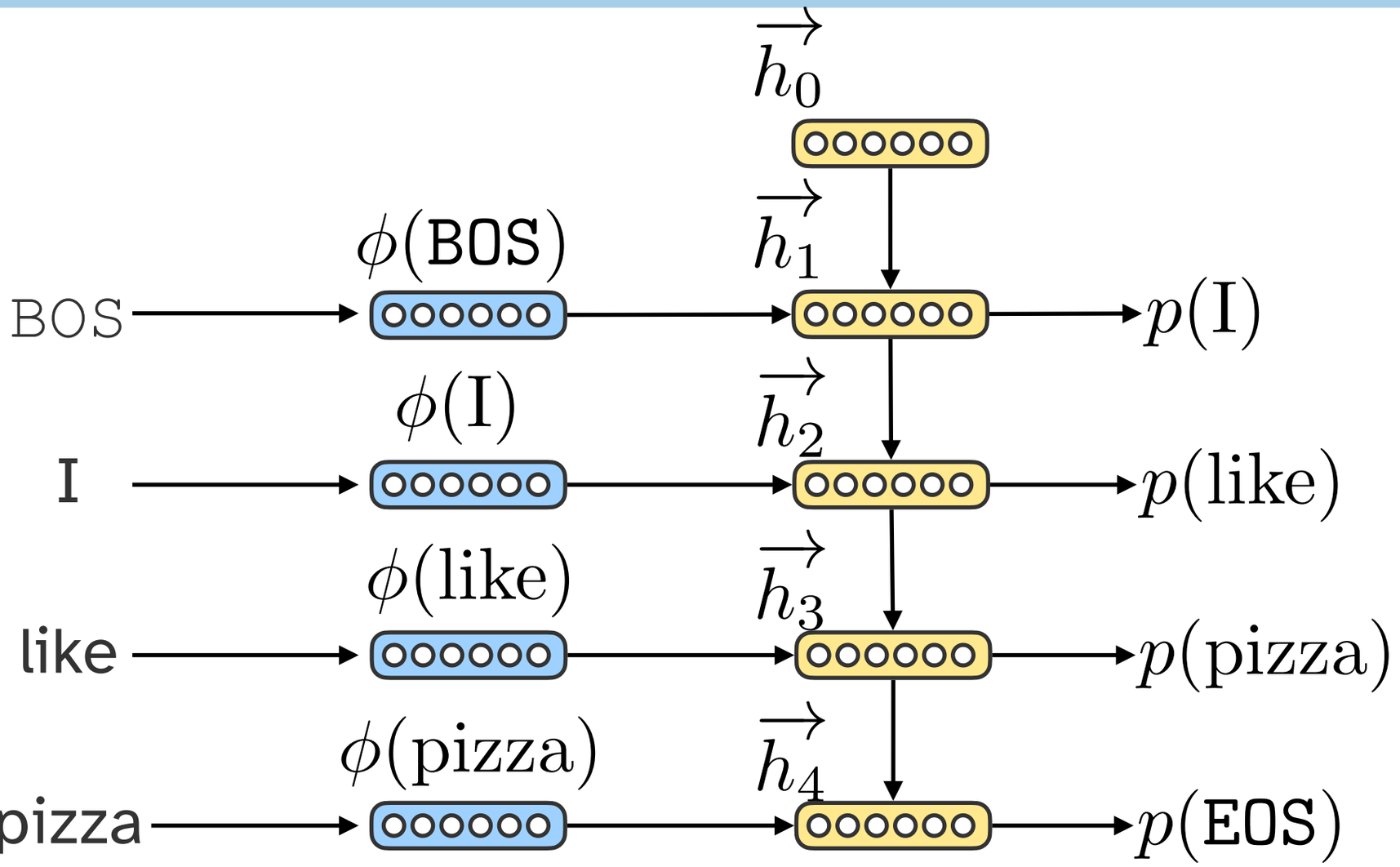
Computing and Using Sequence Embeddings

EECS 183/283a: Natural Language Processing

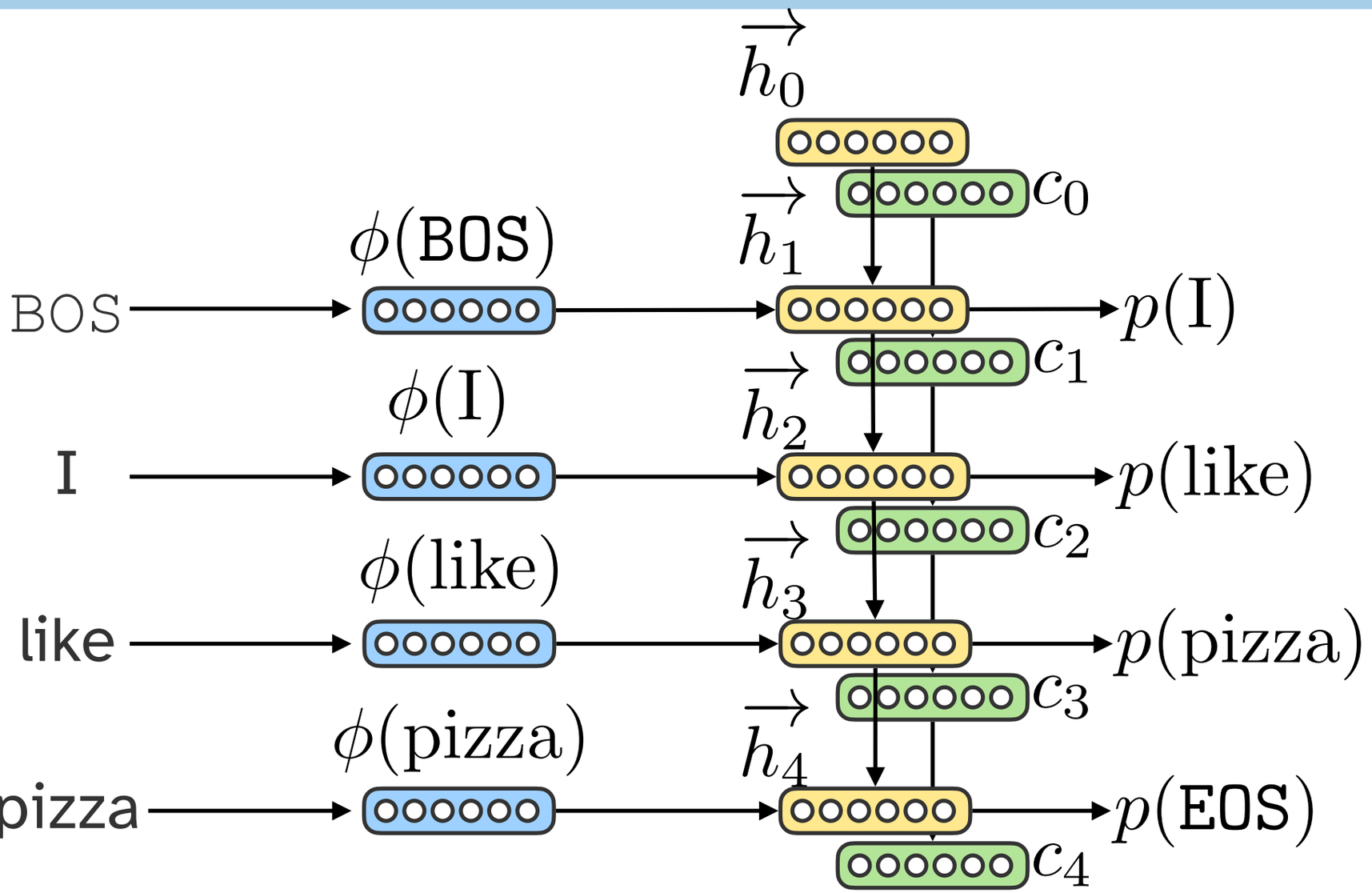
Recurrent Neural Networks



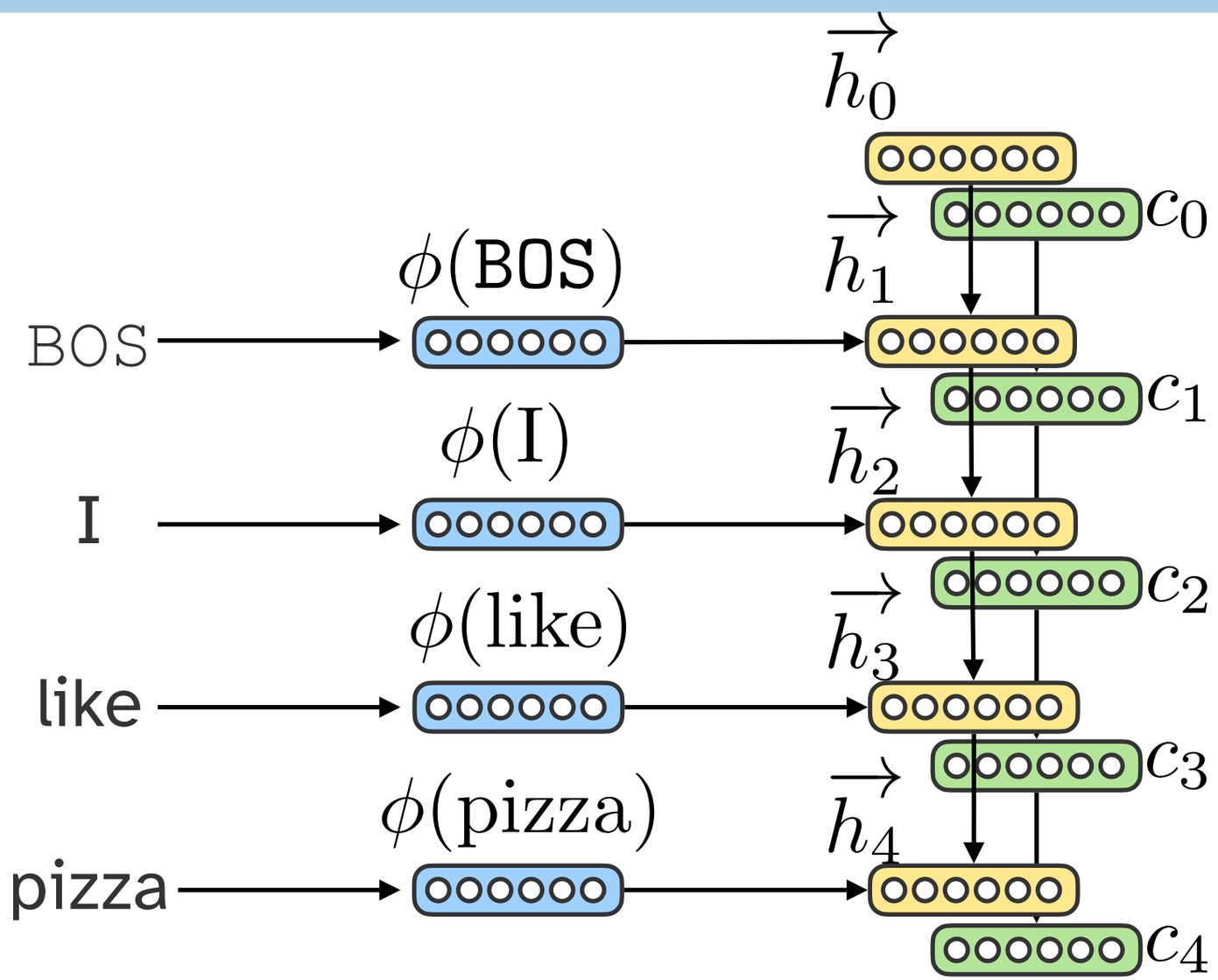
Recurrent Neural Networks



Recurrent Neural Networks (with LSTM cells)



Recurrent Neural Networks (with LSTM cells)

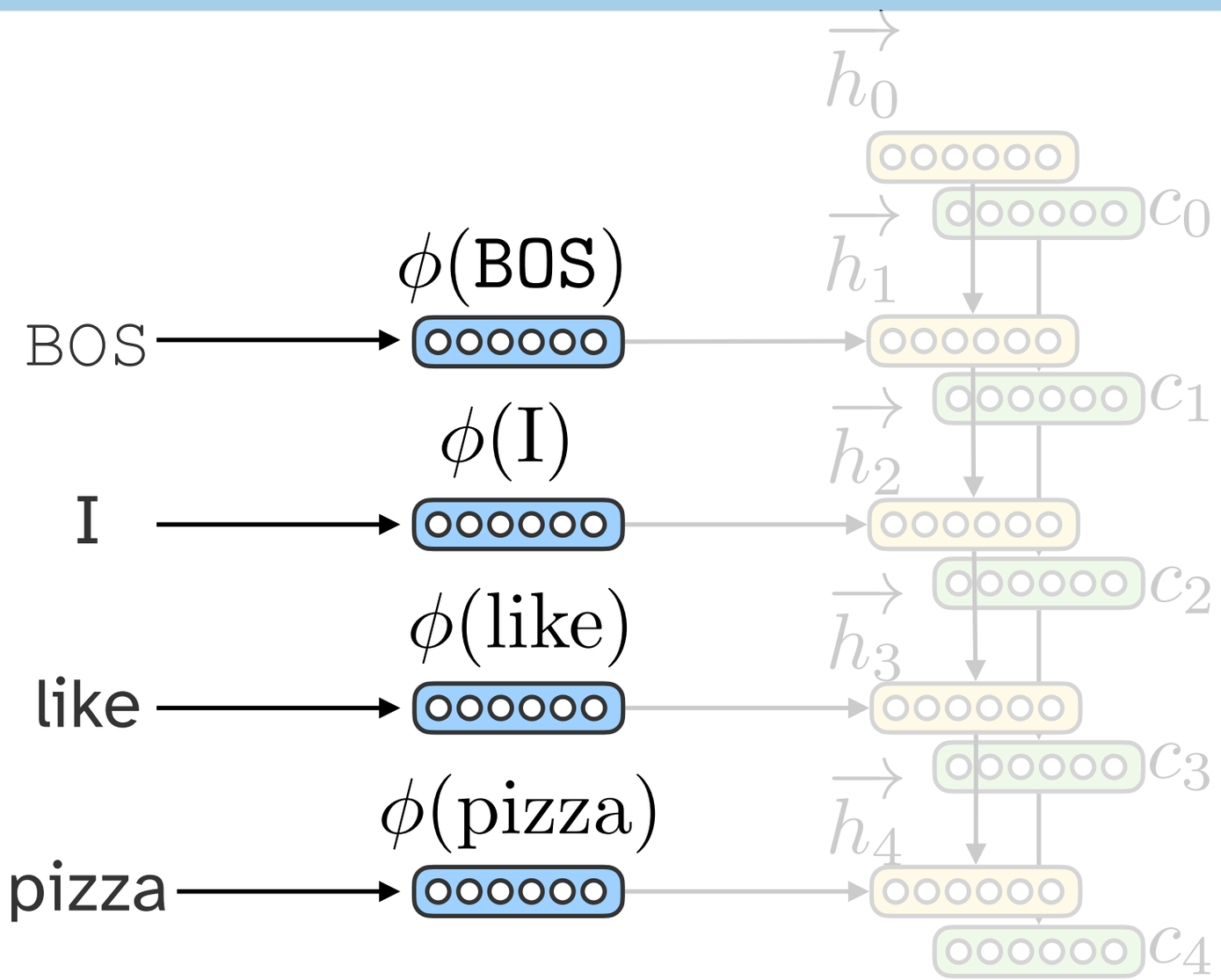


Could we get

$\phi(\overline{x}) = \phi(\mathrm{I\ like\ pizza})$?

If so, what would it be useful for?

Sequence Embeddings



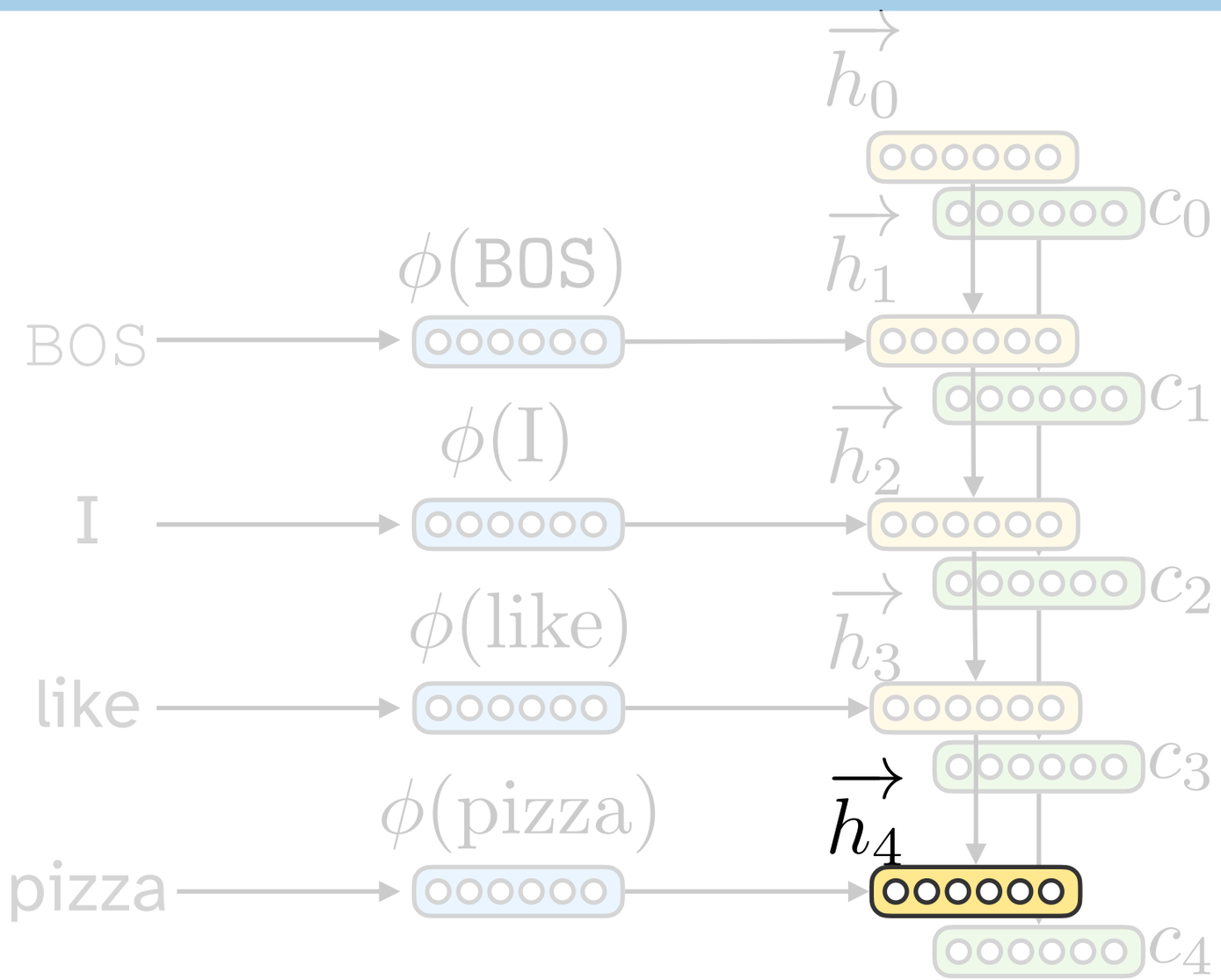
$$\phi(\overline{x}) = \phi(\text{I like pizza})$$

$$= \frac{1}{|\overline{x}|} \sum_{i=1}^{|\overline{x}|} \phi(x_i)$$

Option 0: average word embeddings

But we lose all notions of order

Sequence Embeddings

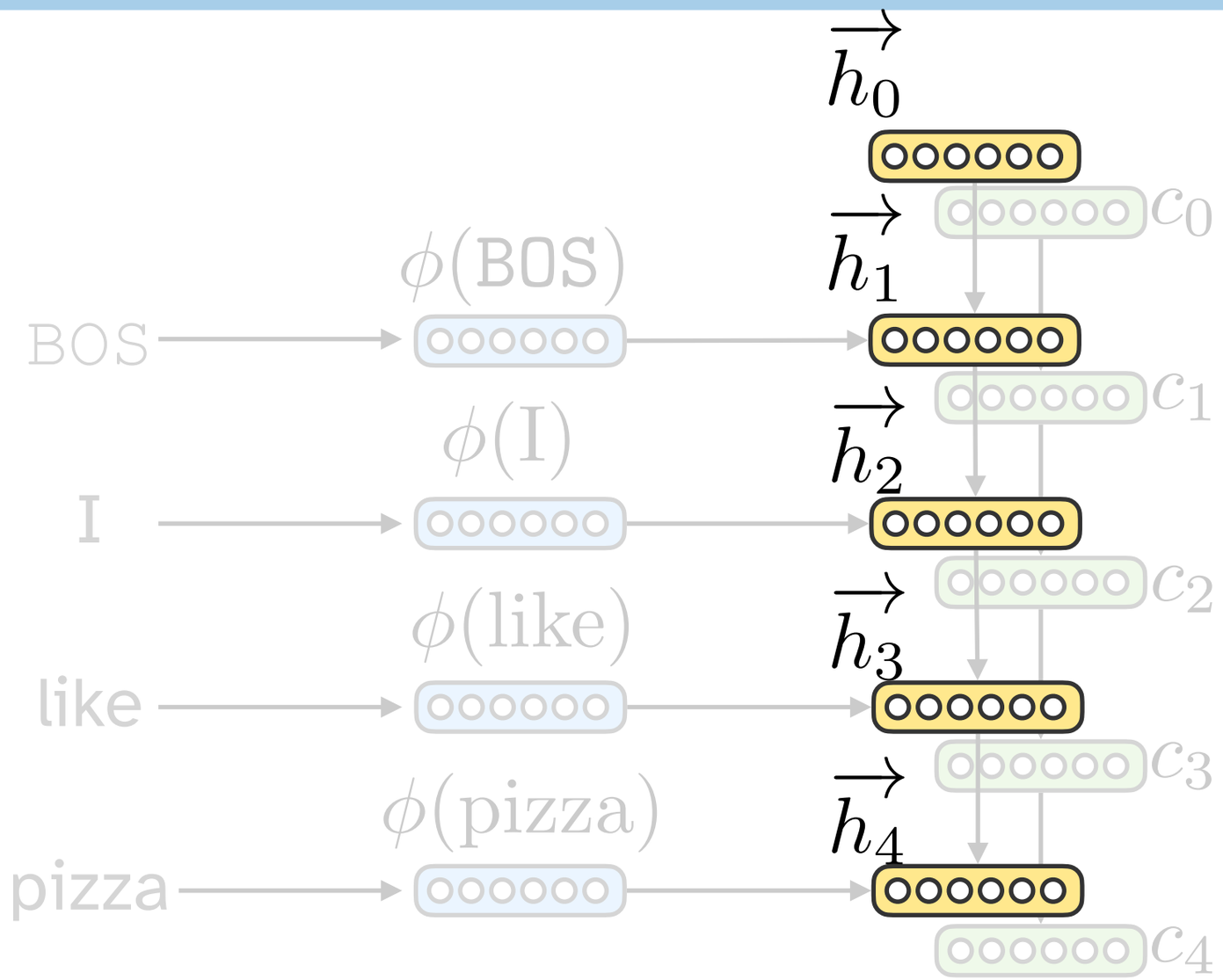


$$\phi(\overline{x}) = \phi(\text{I like pizza})$$
$$= h_{-1}$$

Option 1: take the last hidden state

But this is overly weighted by information from later tokens

Sequence Embeddings



$$\phi(\overline{x}) = \phi(\underset{|\overline{x}|}{\text{I like pizza}})$$

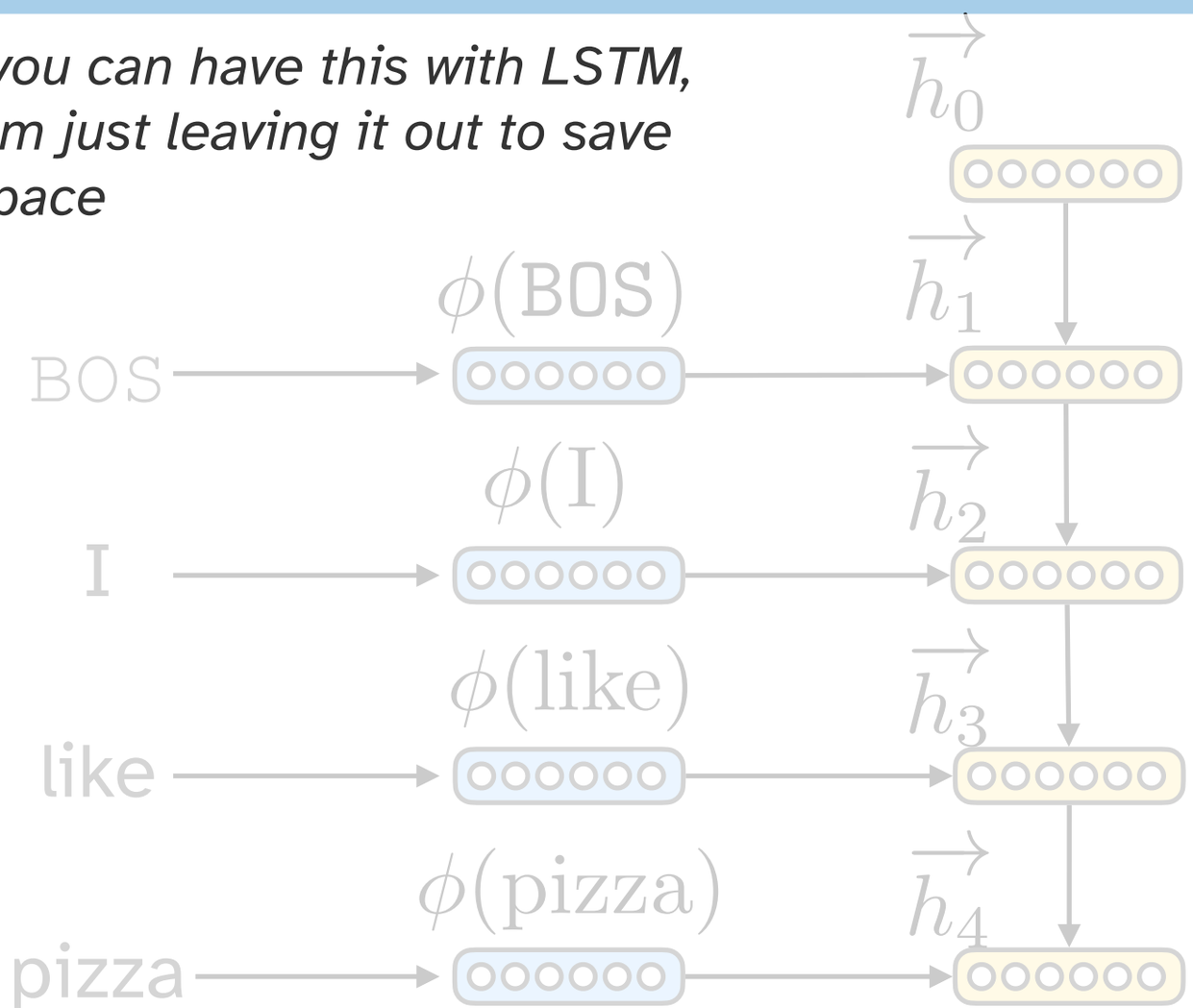
$$= \frac{1}{|\overline{x}|} \sum_{i=1}^{|\overline{x}|} h_i$$

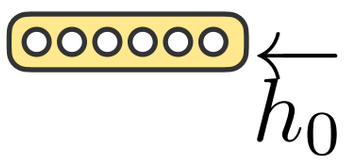
Option 2: average hidden states over time

But earlier states have no information about later ones, while later ones have full information of the sequence

Bidirectional RNN

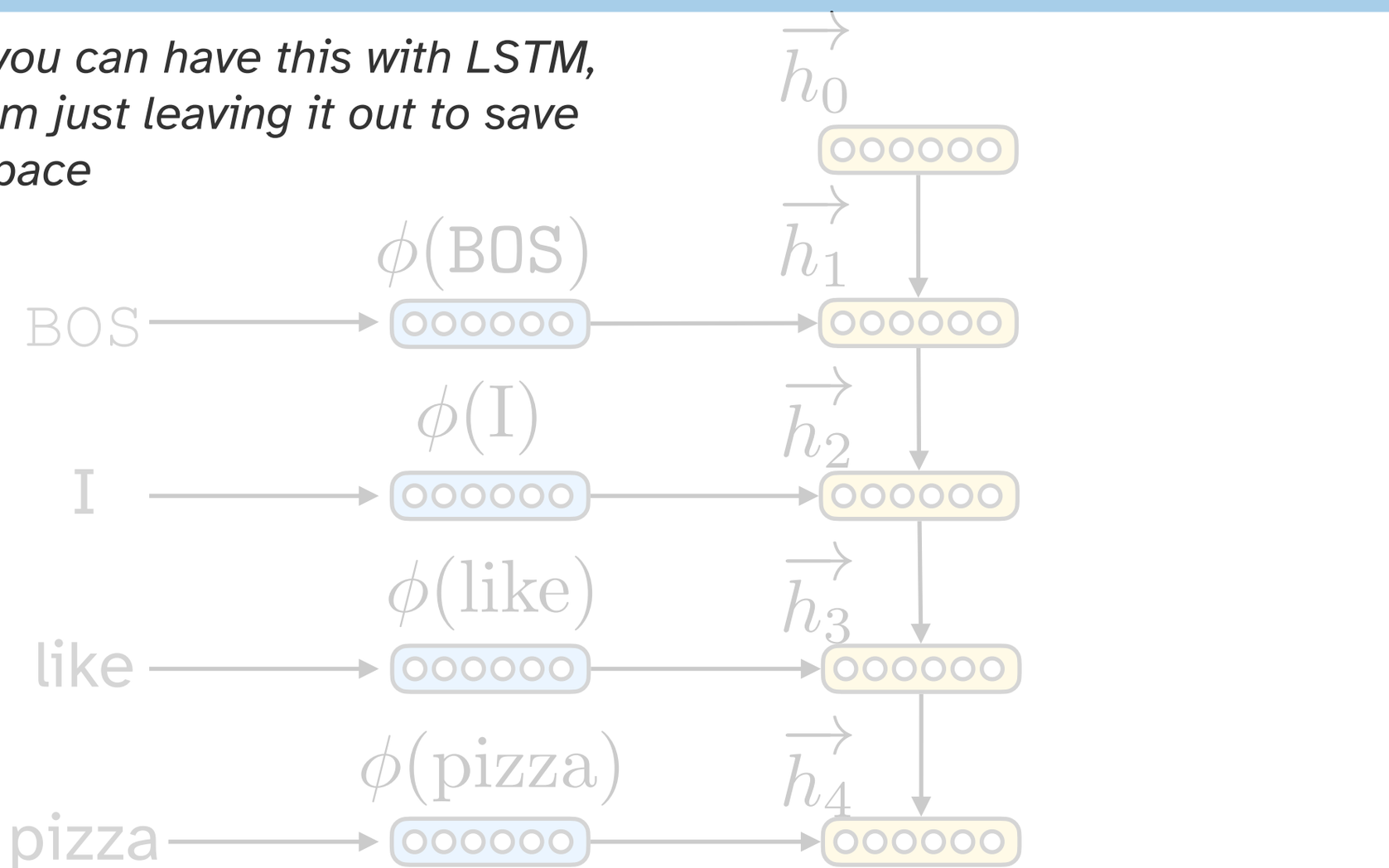
**you can have this with LSTM, I'm just leaving it out to save space*

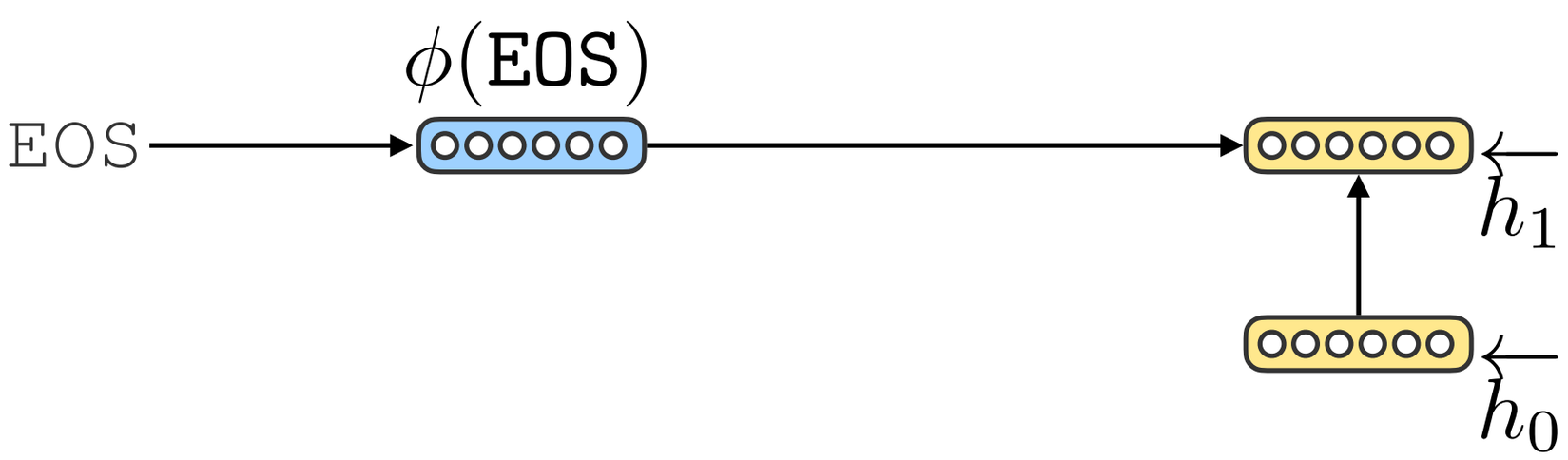




Bidirectional RNN

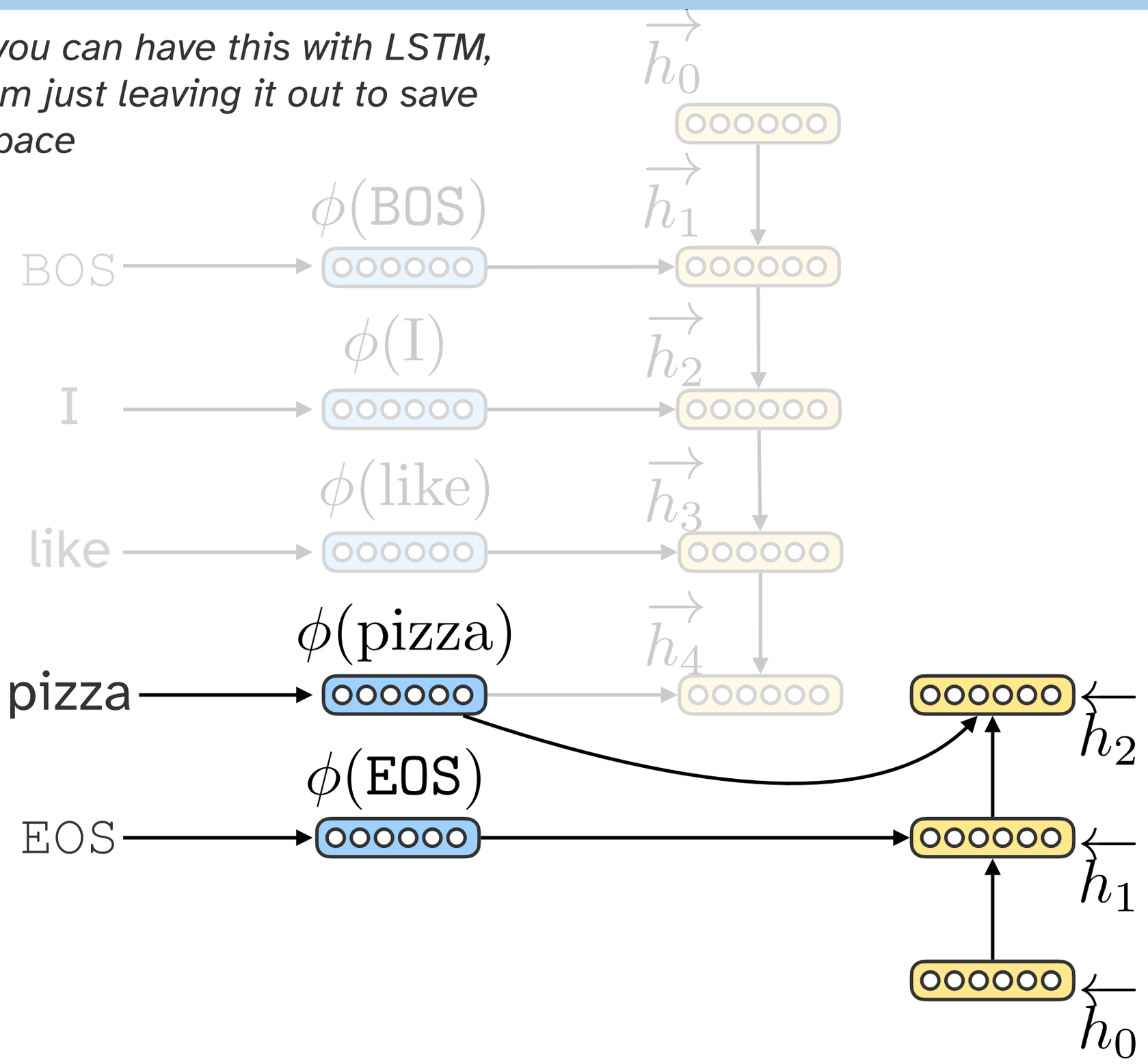
**you can have this with LSTM, I'm just leaving it out to save space*





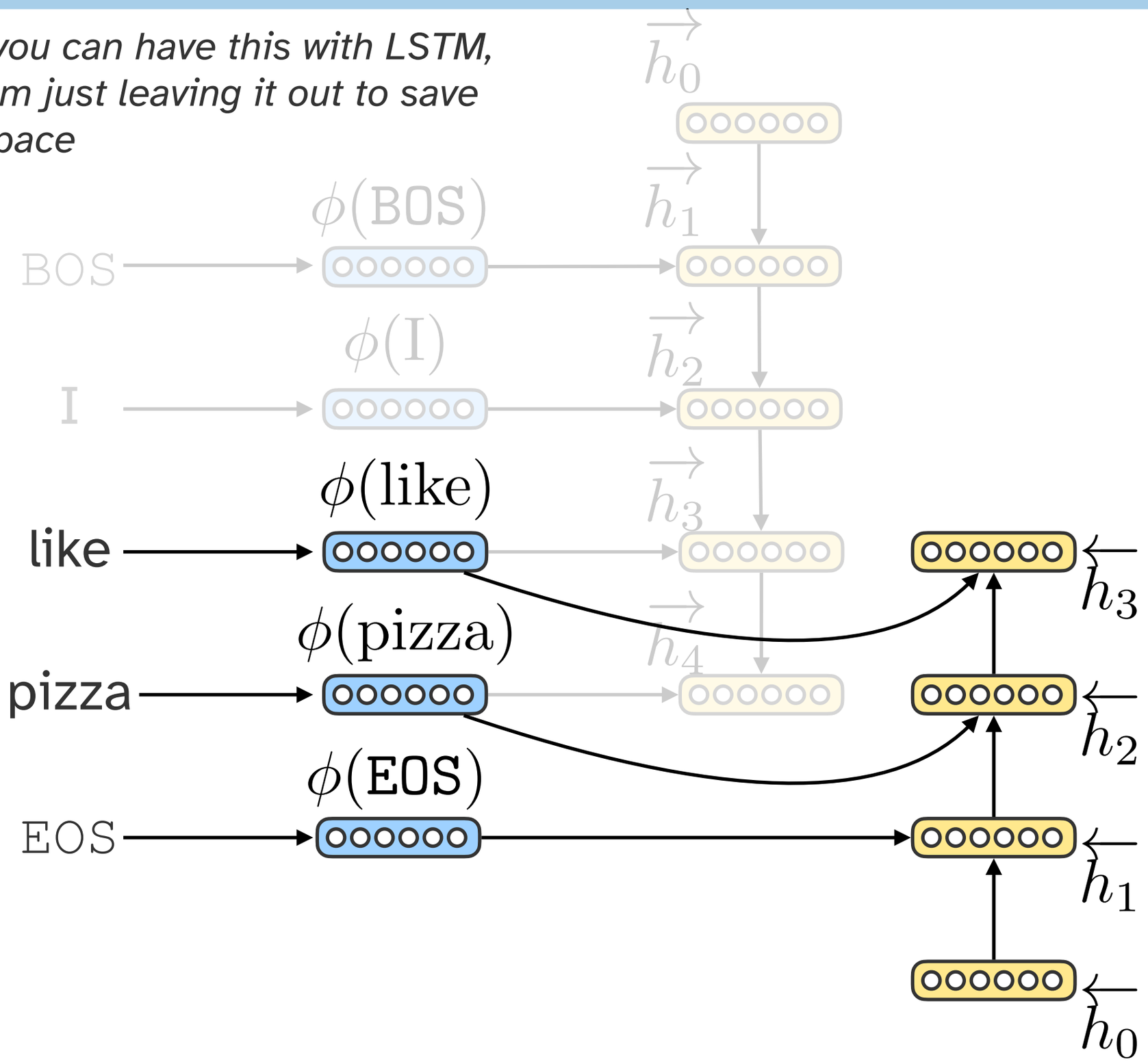
Bidirectional RNN

**you can have this with LSTM, I'm just leaving it out to save space*



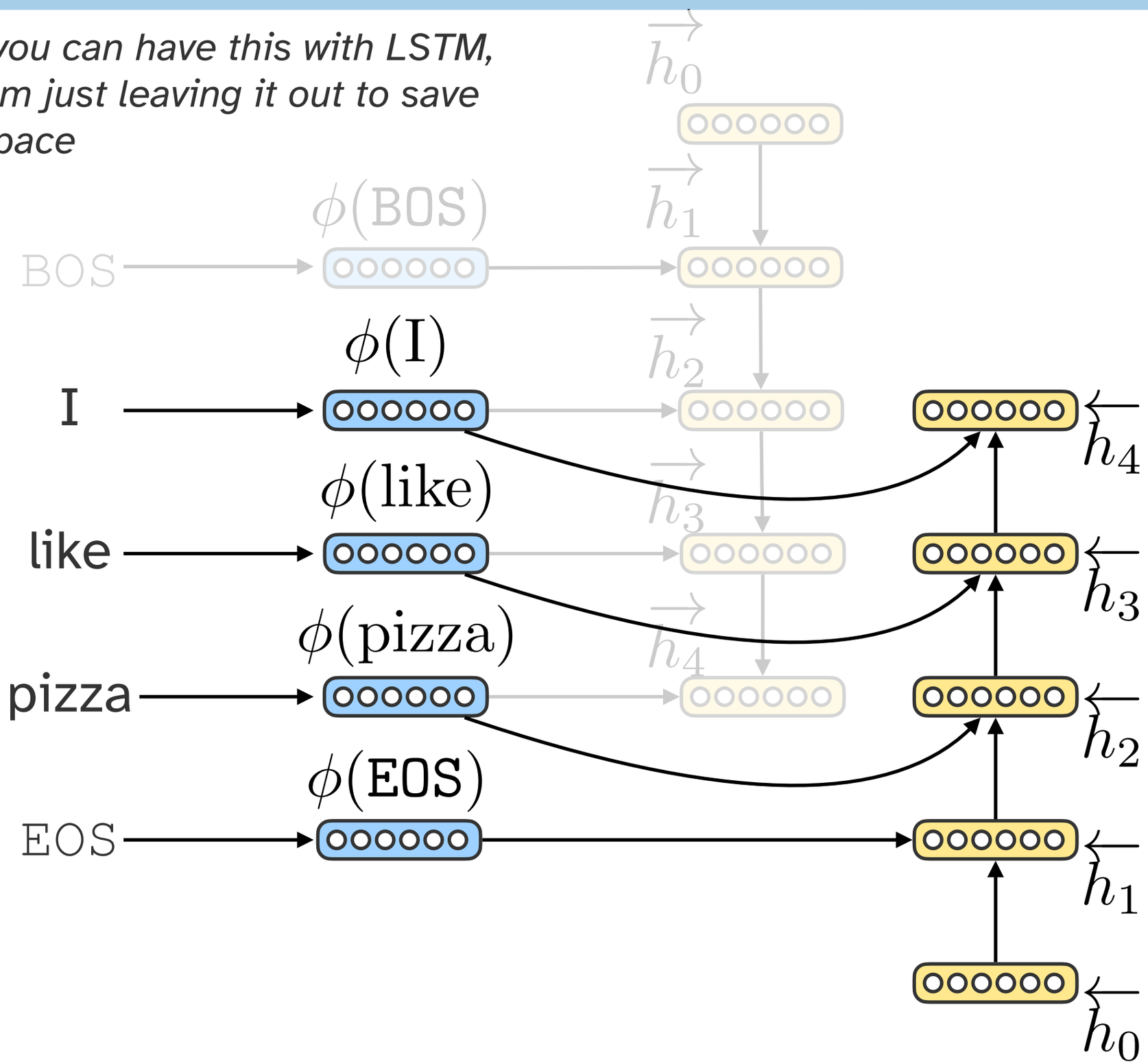
Bidirectional RNN

**you can have this with LSTM, I'm just leaving it out to save space*



Bidirectional RNN

**you can have this with LSTM, I'm just leaving it out to save space*



Bidirectional RNN

**you can have this with LSTM, I'm just leaving it out to save space*

BOS

I

like

pizza

EOS

$\phi(\texttt{BOS})$

$\phi(\mathrm{I})$

$\phi(\mathrm{like})$

$\phi(\mathrm{pizza})$

$\phi(\texttt{EOS})$

$\overrightarrow{h}_0$ $\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_4$

$\overleftarrow{h}_4$ $\overleftarrow{h}_3$ $\overleftarrow{h}_2$ $\overleftarrow{h}_1$ $\overleftarrow{h}_0$

MLP softmax

$\mathcal{L}_\theta(x_1) = -\log p(\texttt{BOS} \mid \texttt{EOS}, \mathrm{pizza}, \mathrm{like}, \mathrm{I}; \theta)$

MLP softmax

$\mathcal{L}_\theta(x_2) = -\log p(\mathrm{I} \mid \texttt{EOS}, \mathrm{pizza}, \mathrm{like}; \theta)$

MLP softmax

$\mathcal{L}_\theta(x_3) = -\log p(\mathrm{like} \mid \texttt{EOS}, \mathrm{pizza}; \theta)$

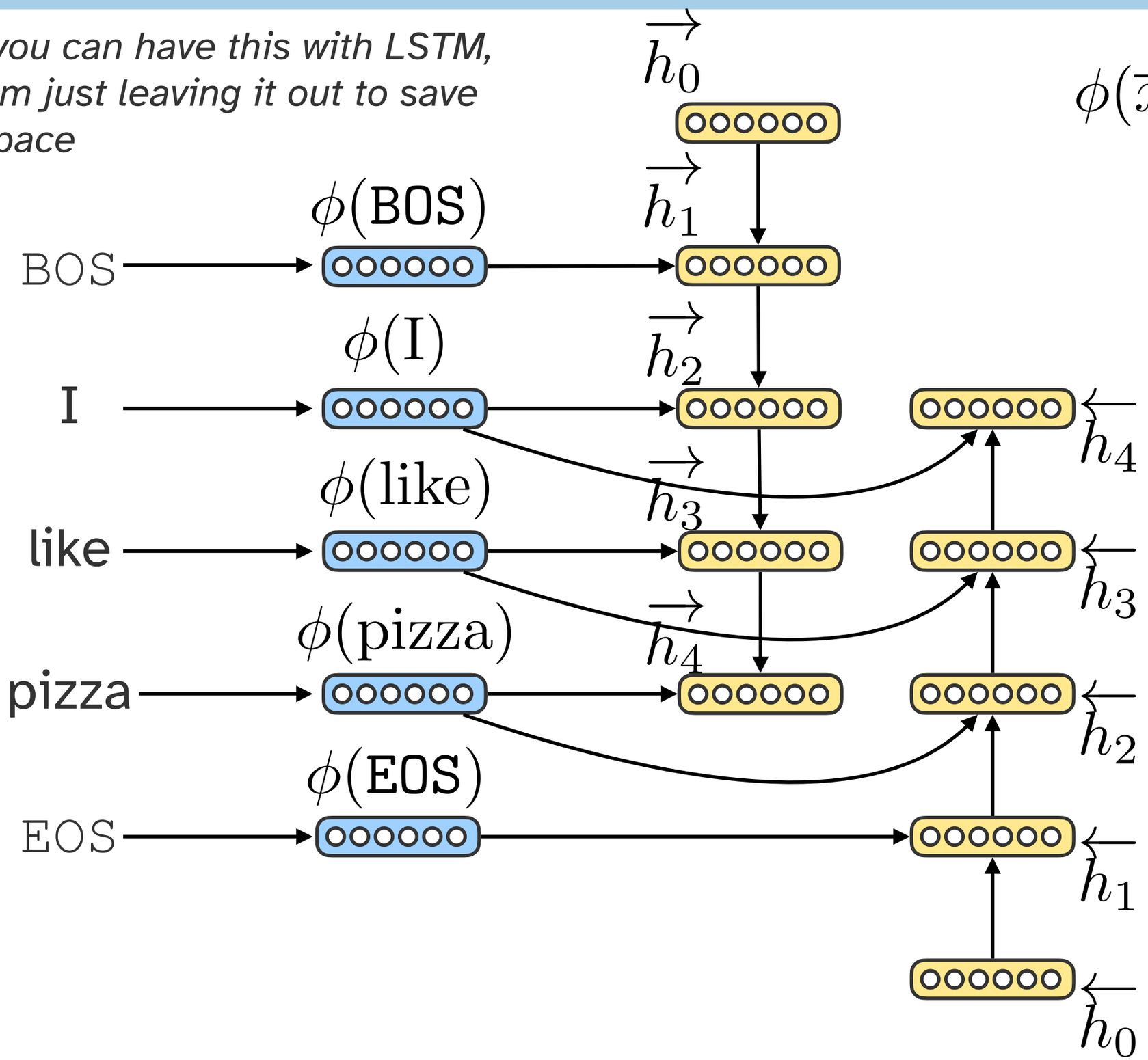
MLP softmax

$\mathcal{L}_\theta(x_4) = -\log p(\mathrm{pizza} \mid \texttt{EOS}; \theta)$

Sentence Embeddings

**you can have this with LSTM, I'm just leaving it out to save space*

$$\phi(\overline{x}) = \phi(\text{I like pizza})$$

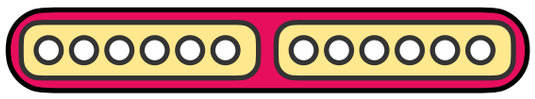


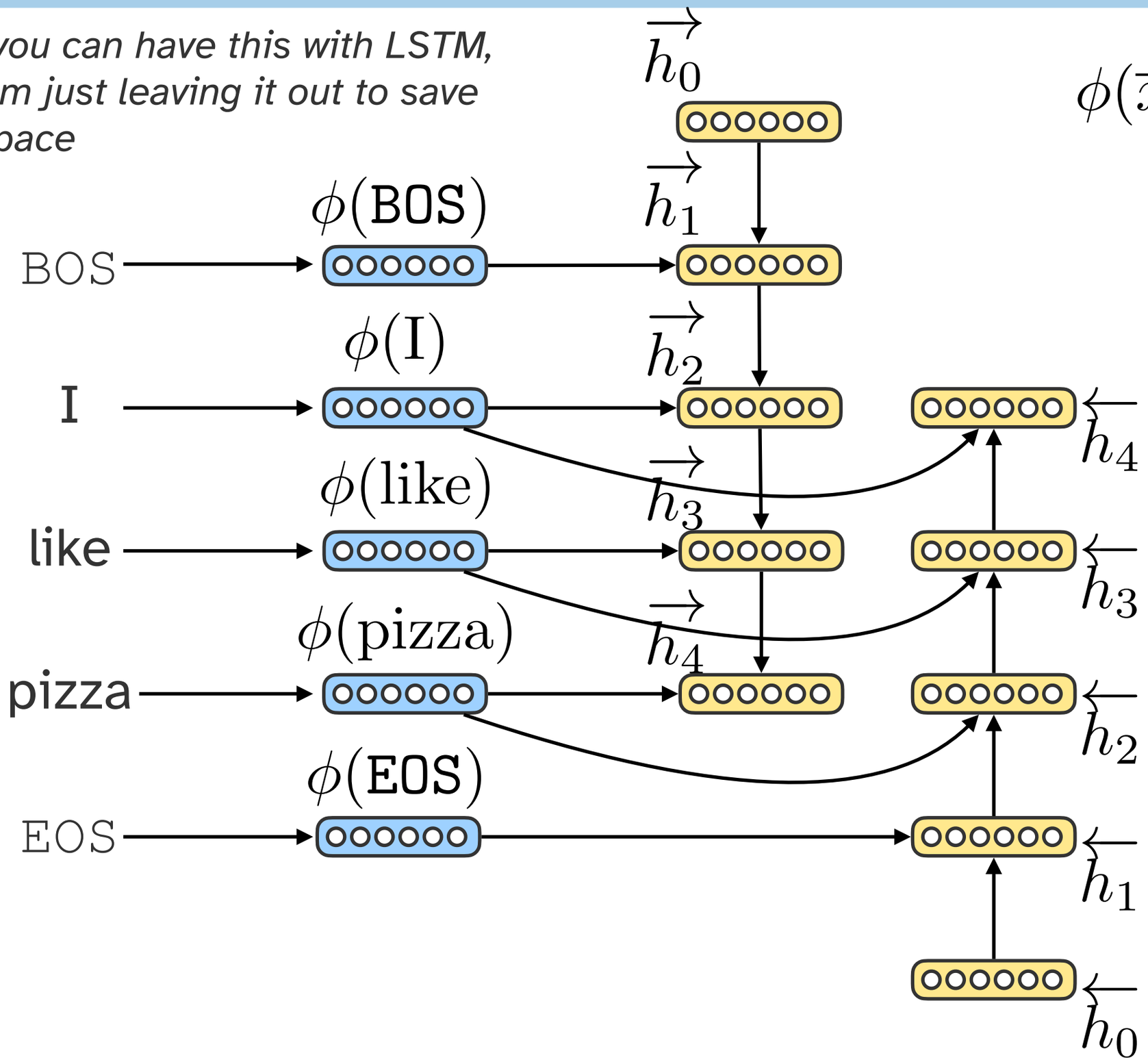
Sentence Embeddings

**you can have this with LSTM, I'm just leaving it out to save space*

$$\phi(\overline{x}) = \phi(\text{I like pizza})$$

$$= \left[\overrightarrow{h}_{|\overline{x}|}; \overleftarrow{h}_{|\overline{x}|}\right]$$

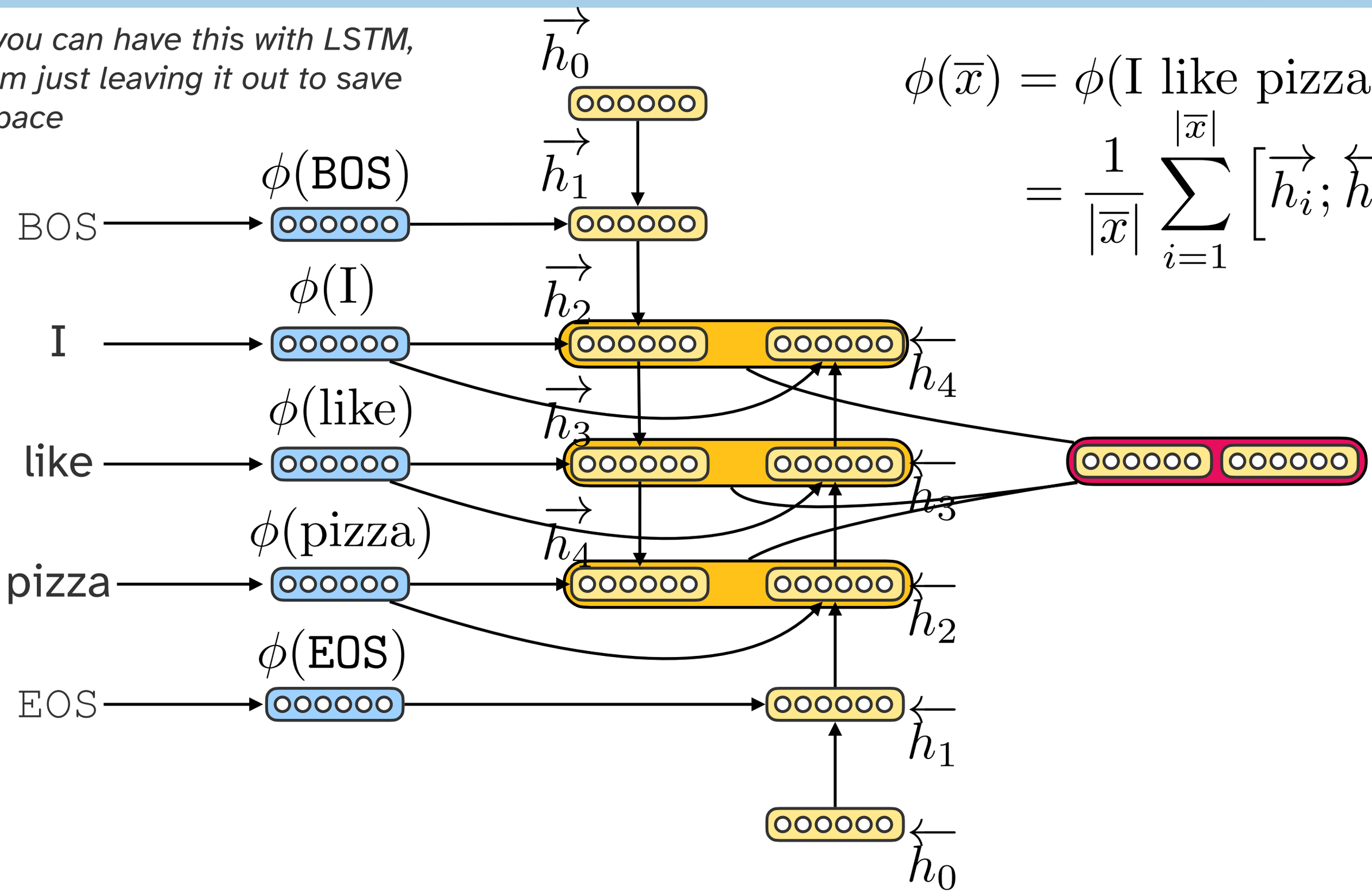




Sentence Embeddings

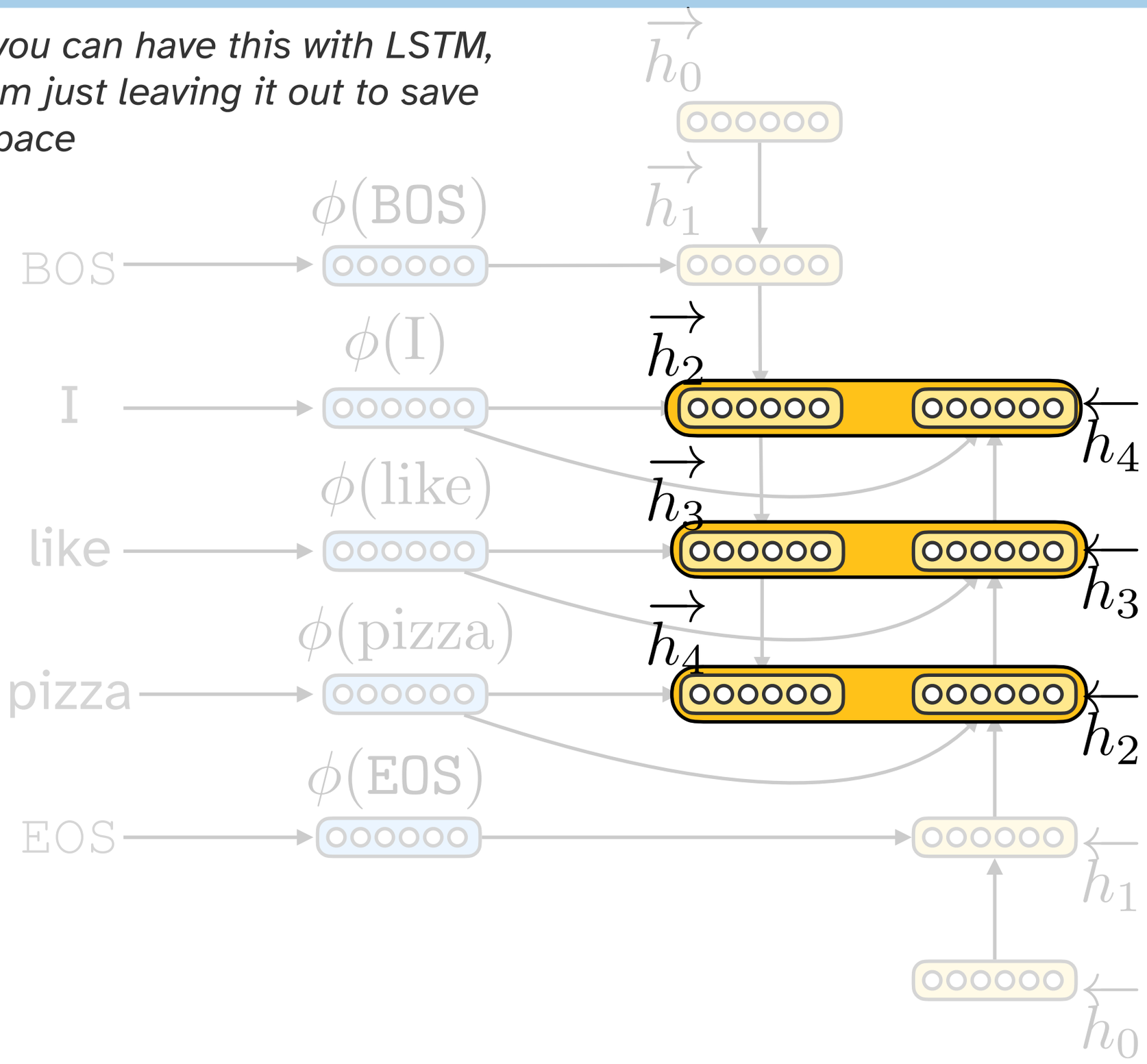
**you can have this with LSTM, I'm just leaving it out to save space*

$$\phi(\overline{x}) = \phi(\text{I like pizza})$$
$$= \frac{1}{|\overline{x}|} \sum_{i=1}^{|\overline{x}|} \left[\overrightarrow{h_i}; \overleftarrow{h_i} \right]$$



Aside: Contextual Word Embeddings

**you can have this with LSTM, I'm just leaving it out to save space*



These hidden states can be interpreted as "context-aware" word embeddings

Solves the problem of polysemy (lexical ambiguity)

ELMo, Peters et al. 2018

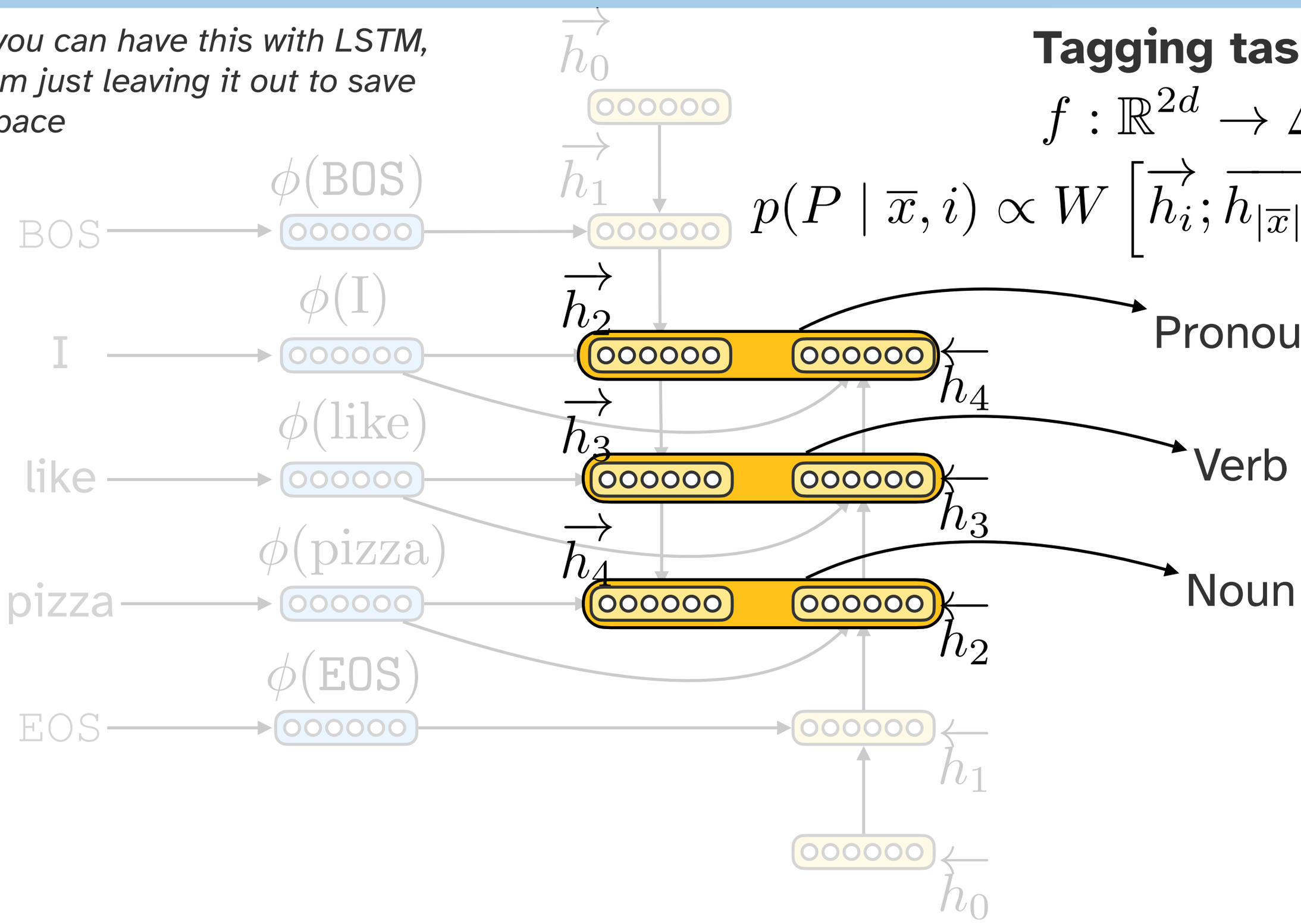
Aside: Contextual Word Embeddings

**you can have this with LSTM, I'm just leaving it out to save space*

Tagging tasks:

$$f : \mathbb{R}^{2d} \rightarrow \Delta^{\mathcal{P}}$$

$$p(P \mid \overline{x}, i) \propto W \left[\overrightarrow{h_i}; \overrightarrow{h_{|\overline{x}|-i}} \right] + b$$



ELMo, Peters et al. 2018

Text Classification

- Let's say we have some embedding function (aka **encoder**)

$$\text{Enc} : \mathcal{V}^+ \rightarrow \mathbb{R}^n$$

Note: $\text{Enc} \equiv \phi$
is the notation we'll use for encoding sequences

- And we want to learn to classify text
  - E.g.: spam vs. not spam

- We want to learn the optimal parameters of some function

$$f_\theta : \mathbb{R}^n \rightarrow \Delta^{\{\text{spam}, \text{not spam}\}}$$

such that for some labeled dataset $\mathcal{D} = \{\overline{x}_i, y_i\}_{i=1}^M$

$$\theta^* = \arg\max_\theta \prod_{i=1}^M f(y_i \mid \text{Enc}(\overline{x}_i); \theta)$$

Text Classification

- Our classifier might just look like a linear transformation or MLP, e.g.,

$$f_\theta(\overline{x}) = \sigma(W\mathrm{Enc}(\overline{x}) + b)$$
$$\theta = \{W, b\}$$

- And we can train this classifier by fixing the parameters of the Encoder
  - Or, we could also backpropagate into the encoder itself $\theta = \{W, b\} \cup \theta_{\mathrm{RNN}}$