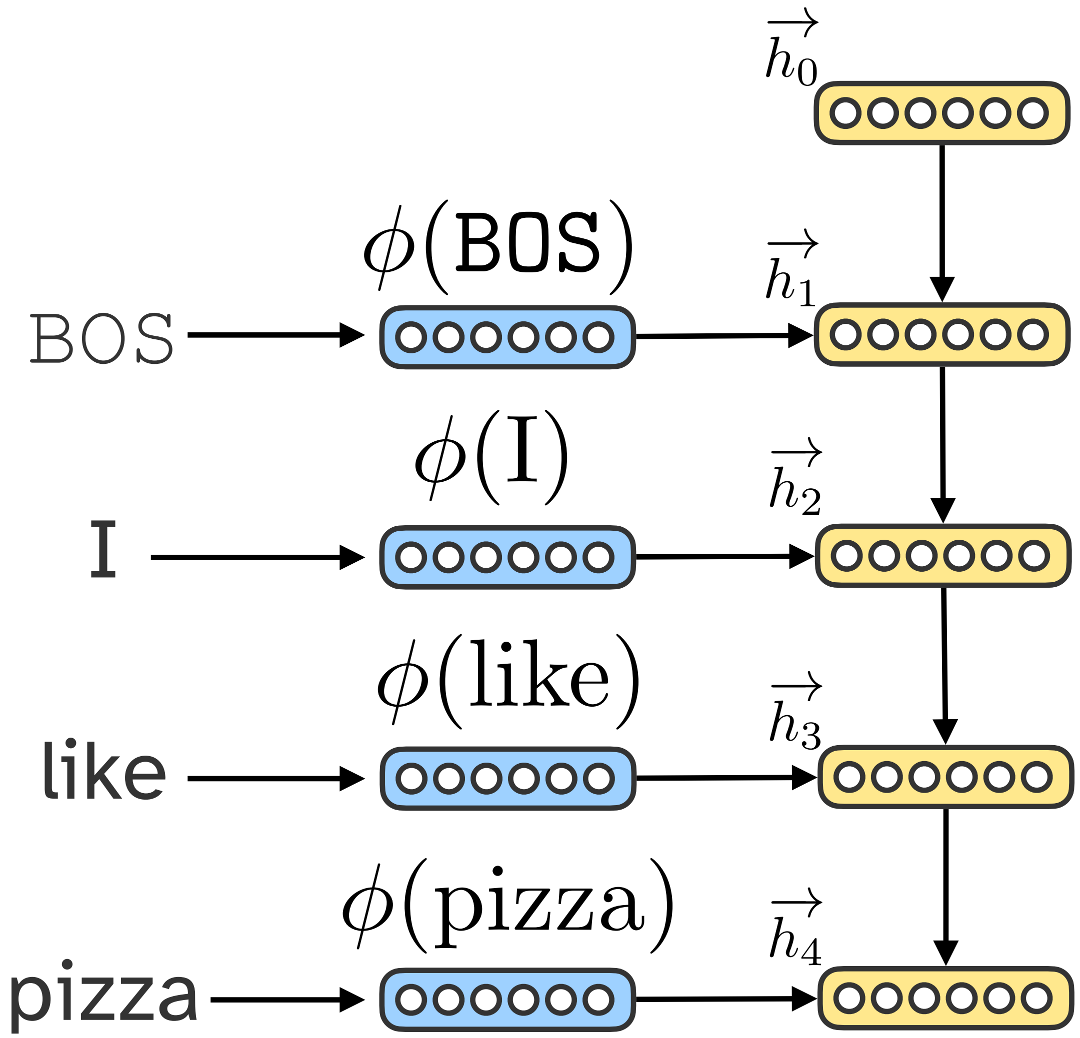
Sequence-to-Sequence Modeling

Berkeley

N L P

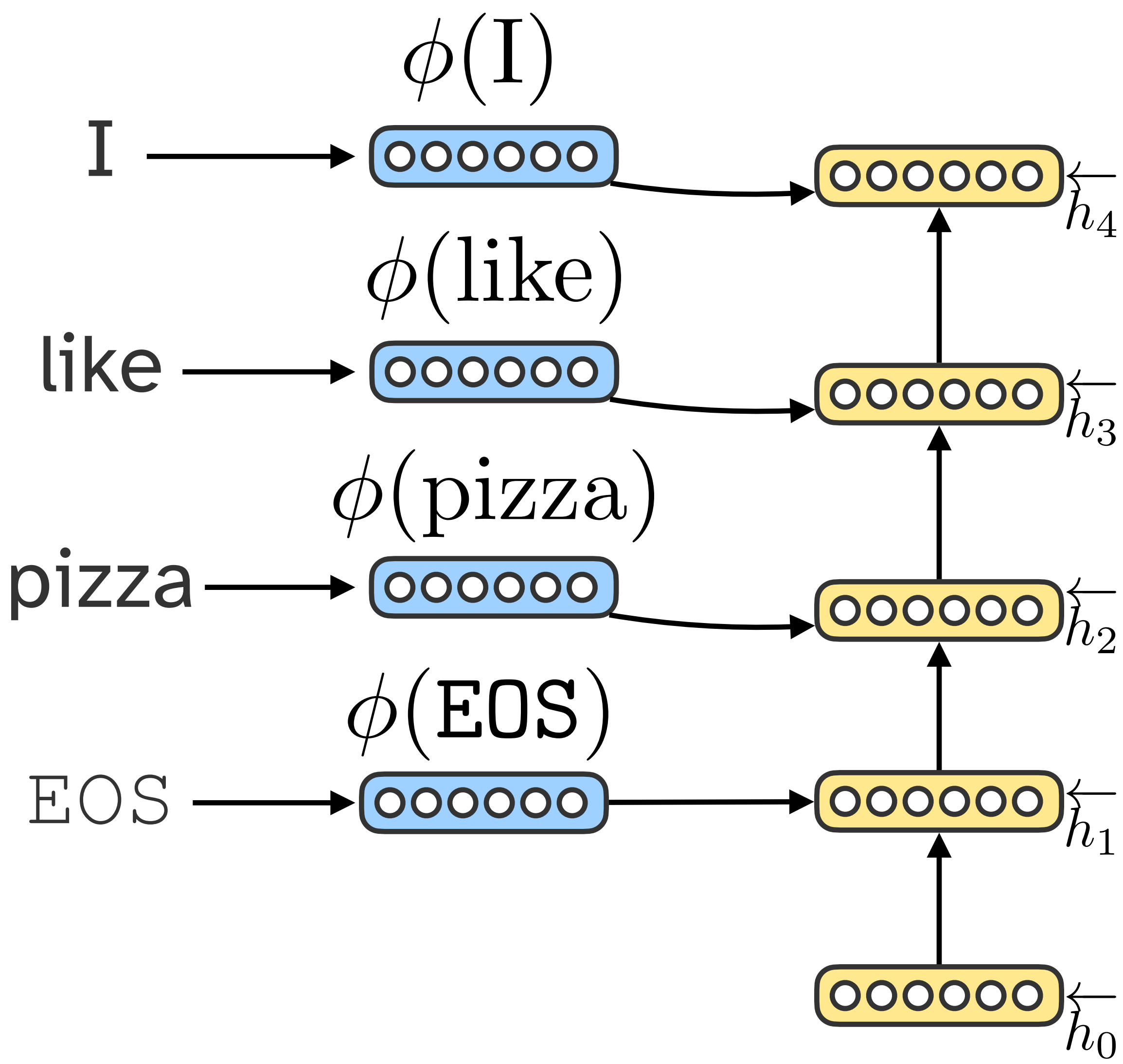
EECS 183/283a: Natural Language Processing

Recap: Recurrent Neural Networks



- Maintains a latent “hidden state” updated after each new token
- Forward RNN

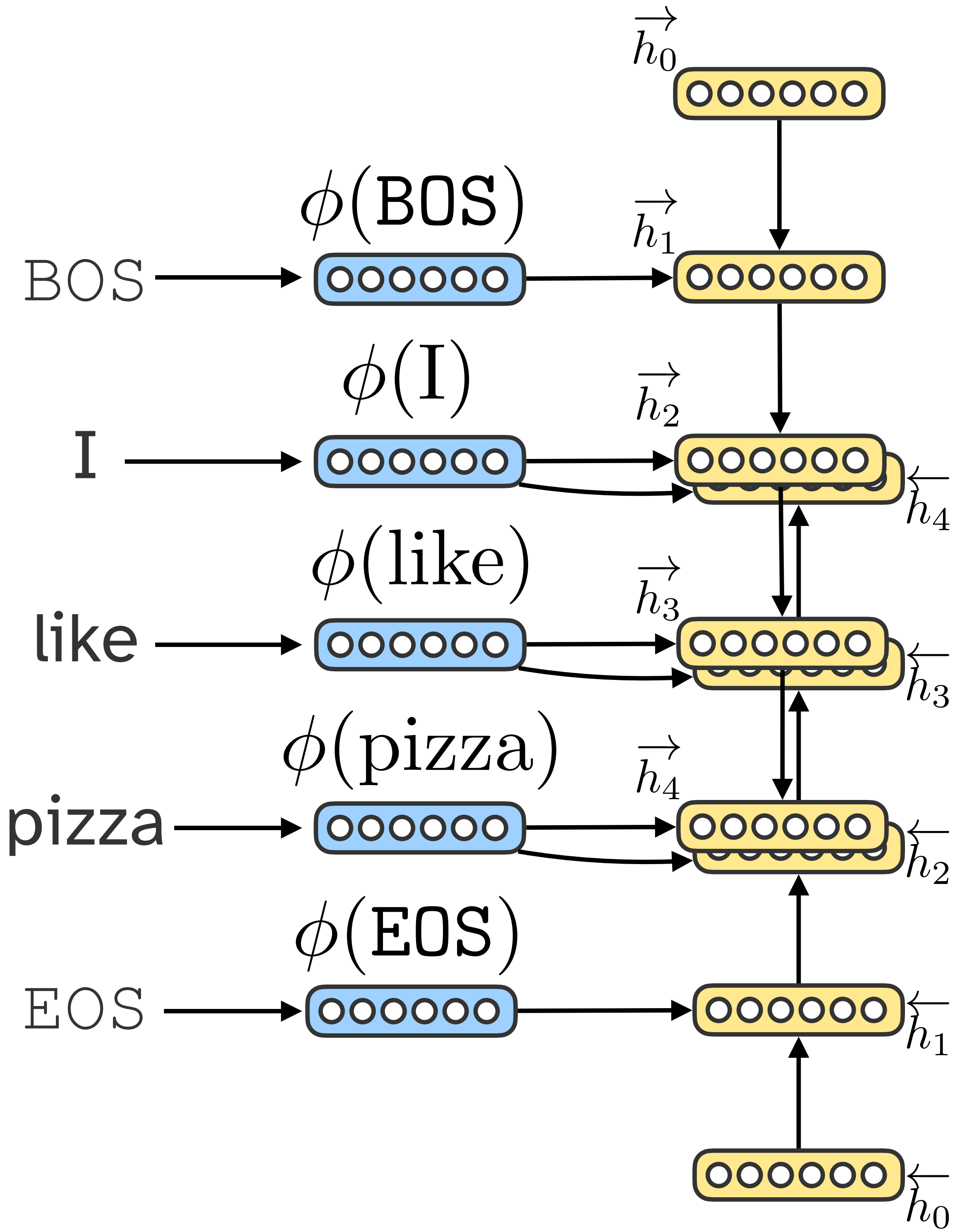
Recap:
Recurrent Neural Networks



- Maintains a latent "hidden state" updated after each new token

- Backward RNN

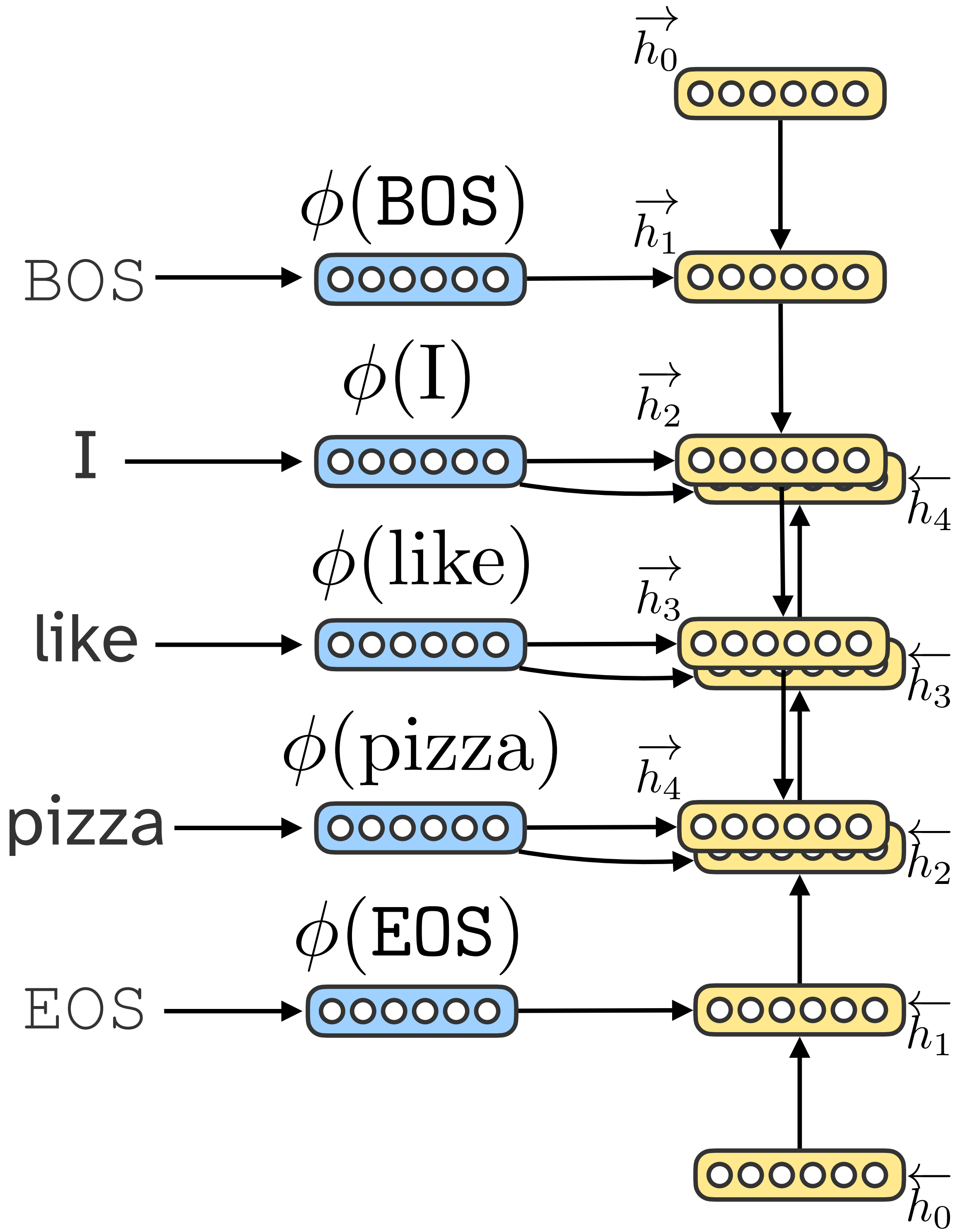
Recap:
Recurrent Neural Networks



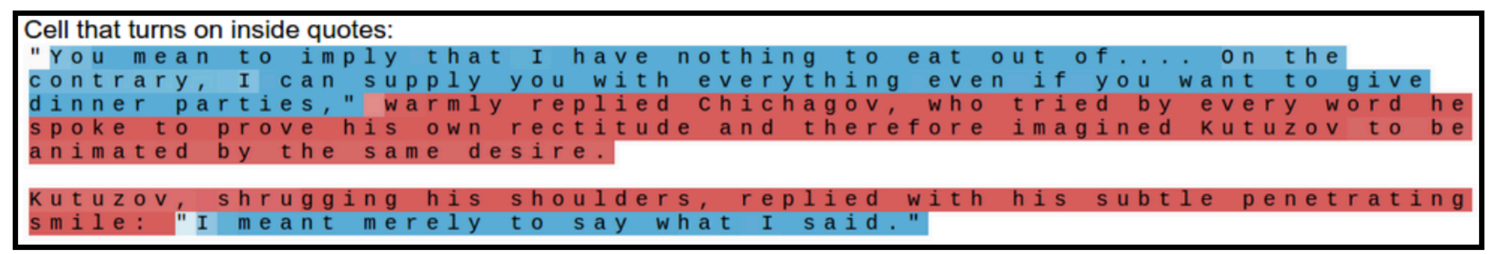
- Maintains a latent "hidden state" updated after each new token

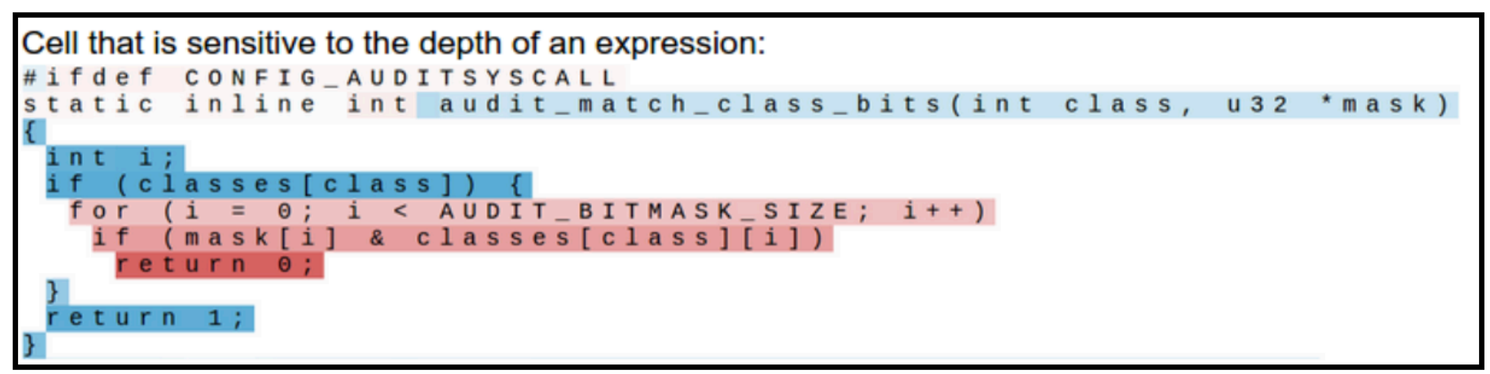
- Bidirectional RNN

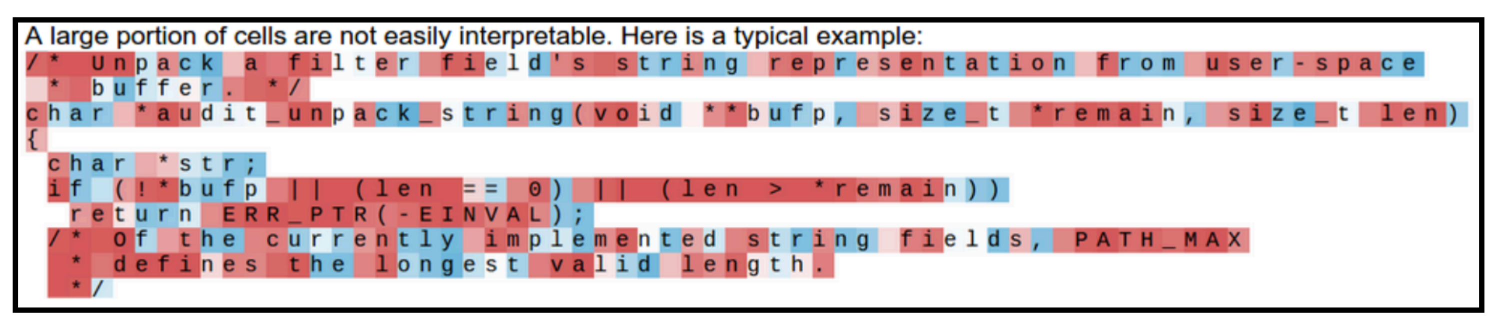
Recap:
Recurrent Neural Networks



- Maintains a latent "hidden state" updated after each new token
- Some parts of hidden states correlate with sentence features!

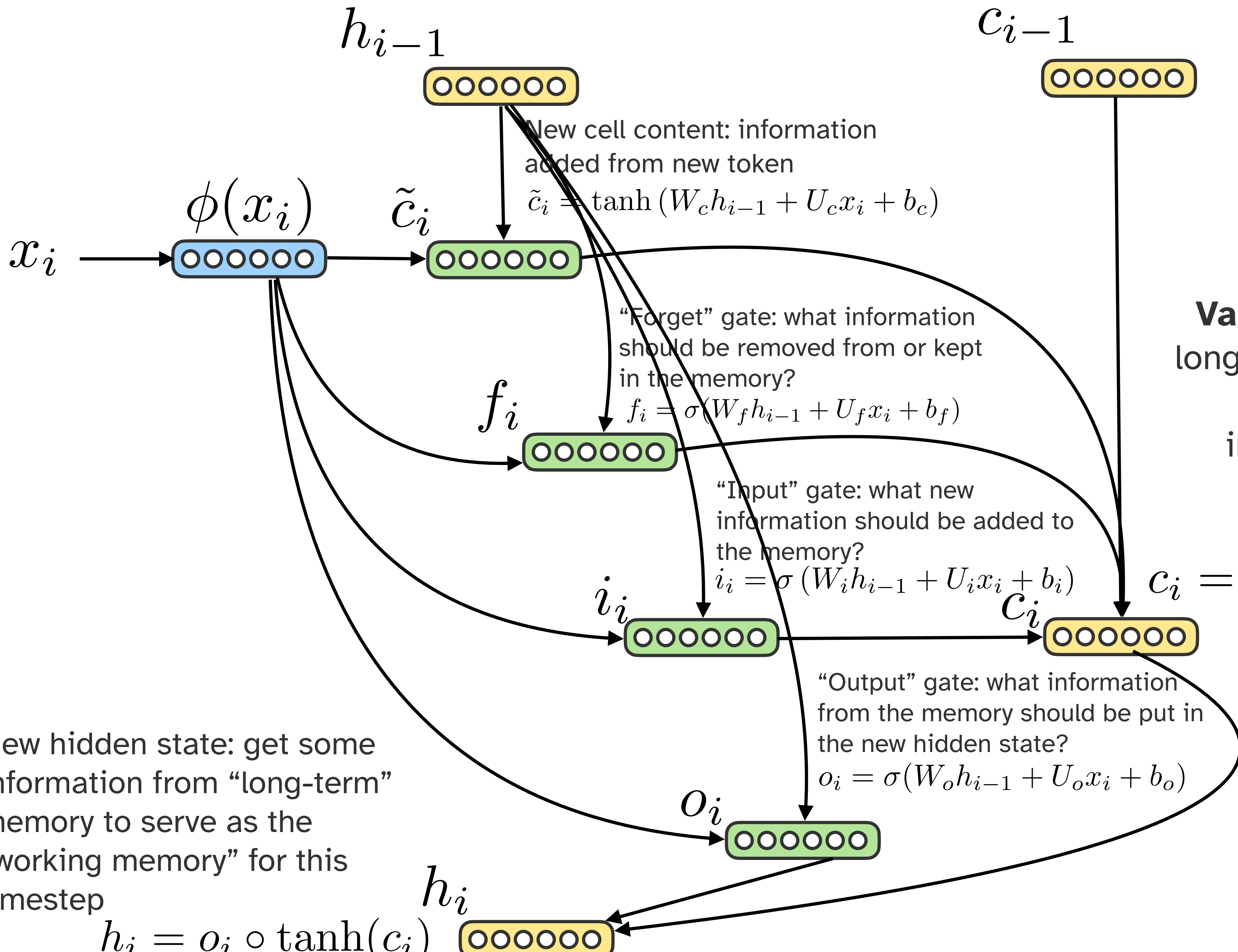






Karpathy et al. 2015

Long Short-Term Memory



Cell memory: same size as hidden state

Vanishing gradients: no longer a problem, because we can keep around information indefinitely

New information to add

$$c_i = i_i \circ \tilde{c}_i + f_i \circ c_{i-1}$$

Information to forget or keep

New hidden state: get some information from "long-term" memory to serve as the "working memory" for this timestep

$$h_i = o_i \circ \tanh(c_i)$$

Recap: LSTM

new cell content $\tilde{c}$: what information does the newest token give us?

- value changes depending on number (singular vs. plural) of most recent noun

input gate i: which information from the newest token should be added to long-term memory?

- "boy(s)" is likely the subject of the sentence, with some long-term dependency with the verb, so it should be stored
- "car(s)" terminates a prepositional phrase, so it's not necessary to store

forget gate f: what information from the memory should we keep versus forget?

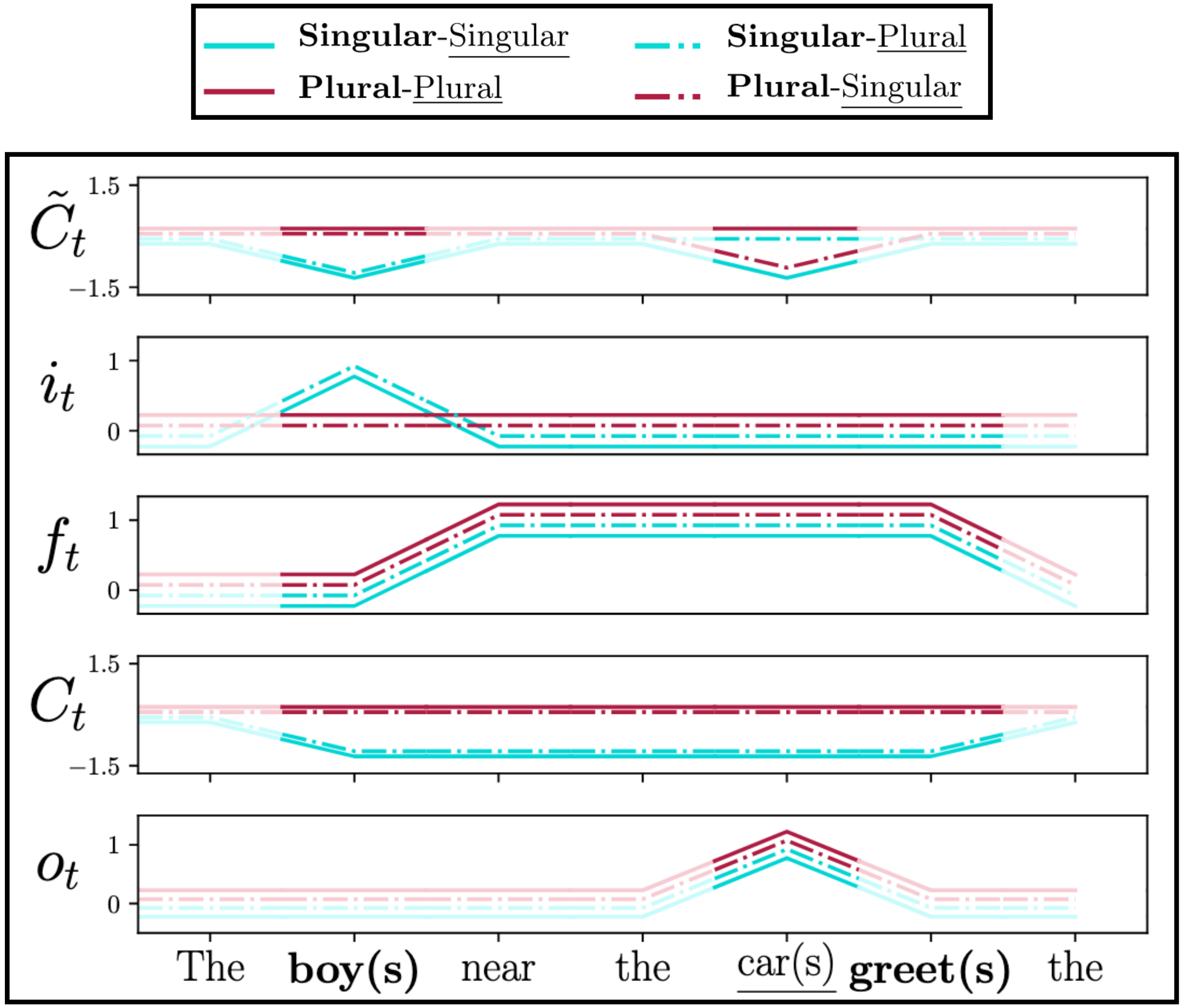
- when we see the prepositional phrase begin with "near", it's important to keep around the number of the subject ("boy(s)") because a verb will probably come later

cell memory c: what information is kept in the long-term memory?

- until the verb associated with the subject appears, we should keep around its number

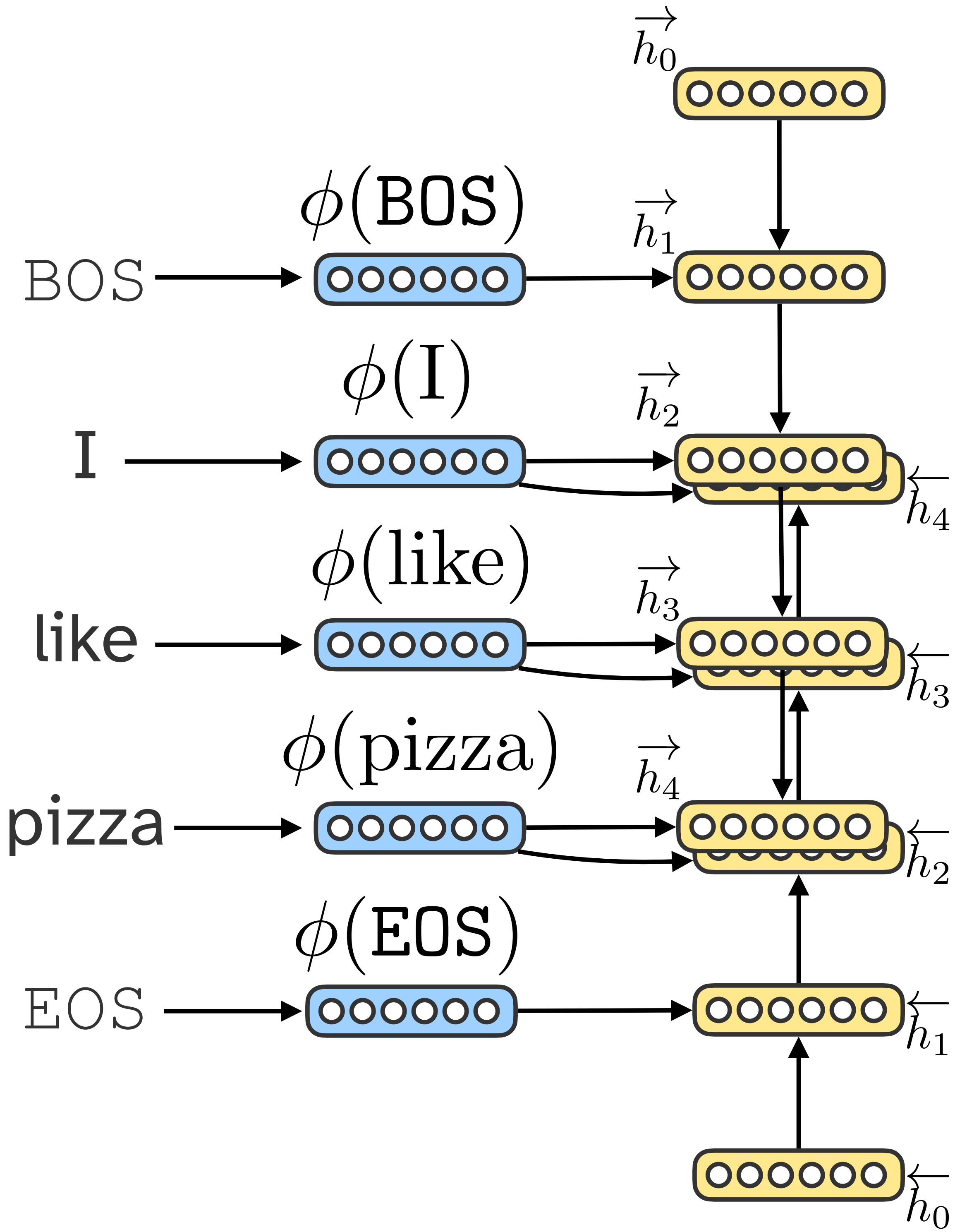
output gate o: what information from the cell memory is important for the next token's prediction?

- if we'll predict the verb next, we need to know the subject's number



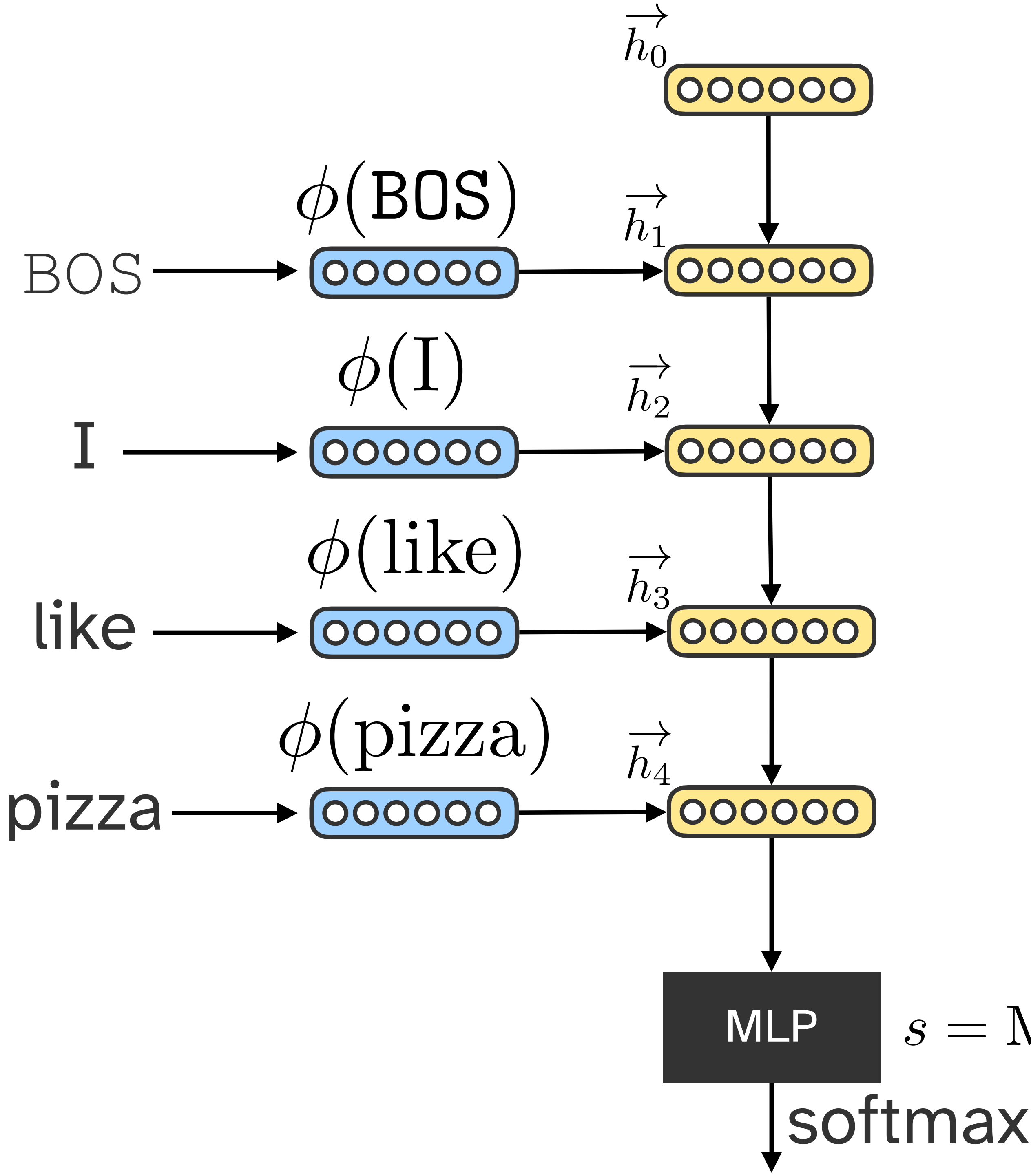
Lakretz et al. 2019

Recap: Recurrent Neural Networks



- Maintains a latent "hidden state" updated after each new token
- Hidden states can be used to:

Recap:
Recurrent Neural Networks

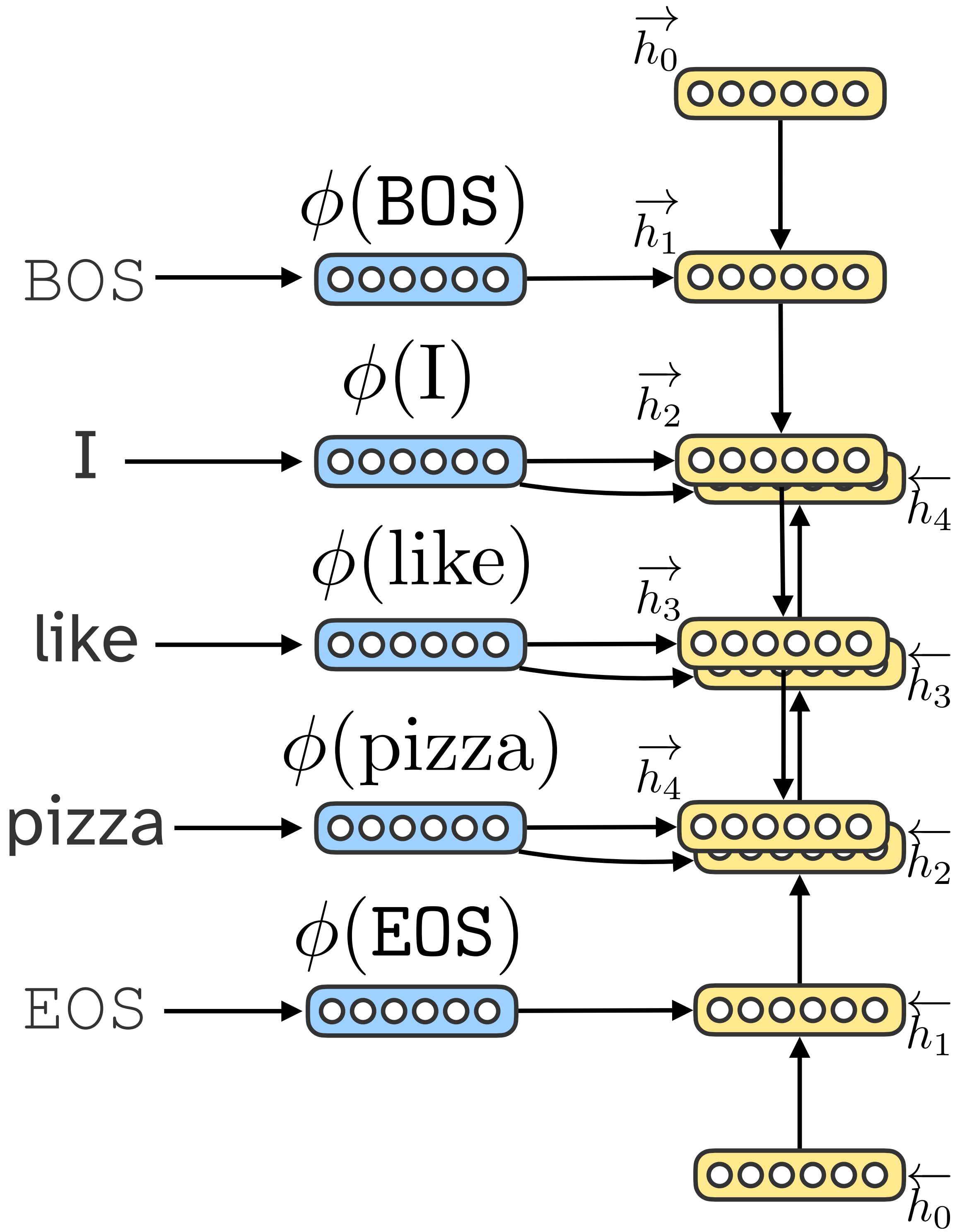


$s = \mathrm{MLP}(h_4) \quad s \in \mathbb{R}^{|\mathcal{V}|}$

$$p(X = x \mid x_1, \ldots, x_4) = \frac{s[x]}{\sum_{x' \in \mathcal{V}} s[x']}$$

- Maintains a latent "hidden state" updated after each new token
- Hidden states can be used to:
  - Get a distribution over the next token's value

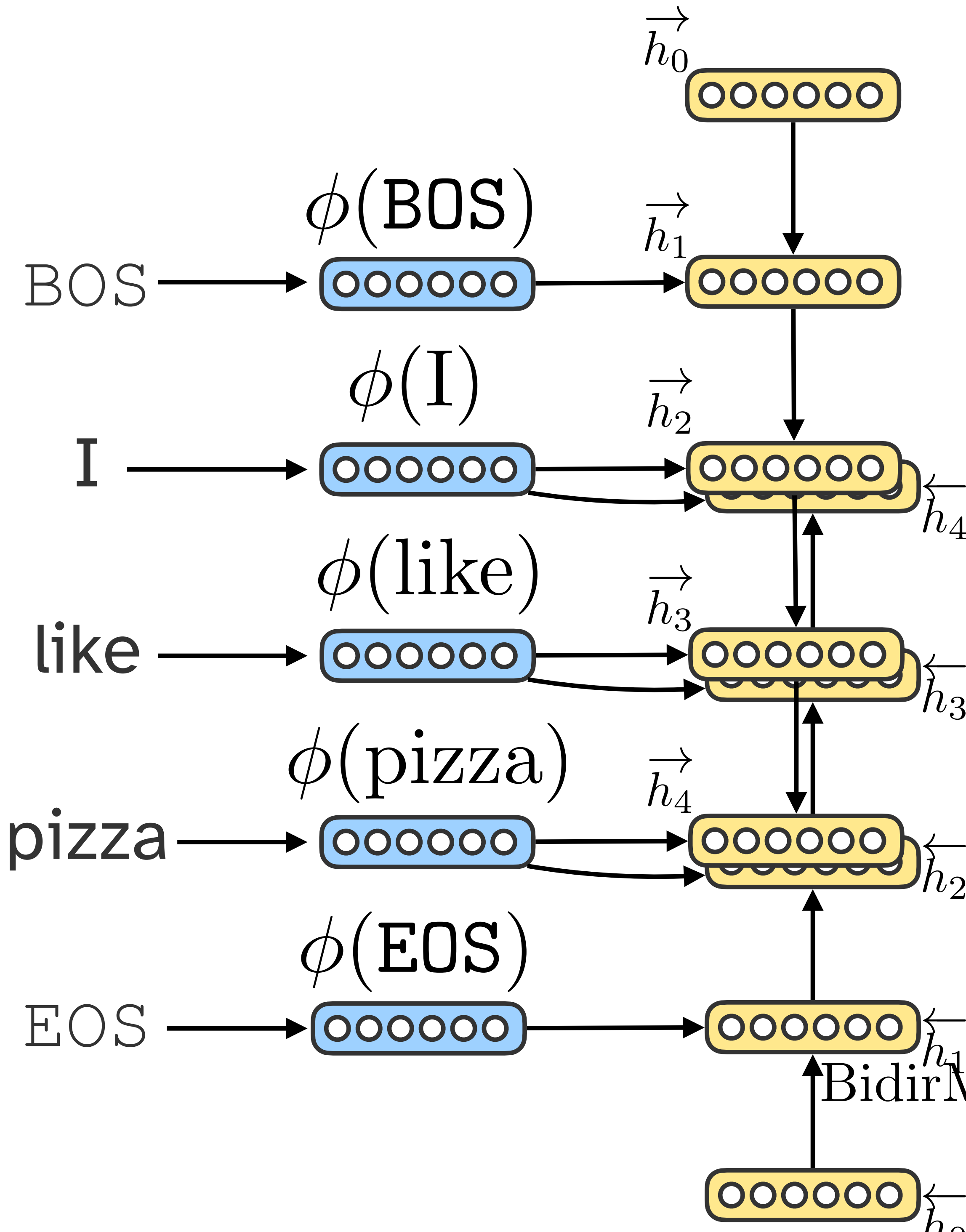
Recap: Recurrent Neural Networks



- Maintains a latent "hidden state" updated after each new token
- Hidden states can be used to:
  - Get a distribution over the next token's value
  - Get a vector representation of the entire sequence

$$\mathrm{Enc}(\overline{x}) = \mathrm{Pool}\left(h_1, \ldots, h_{|\overline{x}|}\right) \in \mathbb{R}^d$$

Recap: Recurrent Neural Networks



$$\mathrm{Enc}(\overline{x}) = \mathrm{Pool}\left(h_1, \ldots, h_{|\overline{x}|}\right) \in \mathbb{R}^d$$

"Pooling" functions map from a sequence of items of type *t* to a single item of type *t*

$$\mathrm{Pool} : \mathbb{R}^{d \times n} \to \mathbb{R}^d$$

n vectors of length *d*   single vector of length *d*

$$\mathrm{MeanPool}(h_1, \ldots, h_{|\overline{x}|}) = \frac{1}{|\overline{x}|} \sum_{i=1}^{|\overline{x}|} h_i$$

$$\mathrm{BidirMeanPool}(\overrightarrow{h_1}, \ldots, \overrightarrow{h_{|\overline{x}|}}; \overleftarrow{h_1}, \ldots, \overleftarrow{h_{|\overline{x}|}}) = \frac{1}{|\overline{x}|} \sum_{i=1}^{|\overline{x}|} \left[\overrightarrow{h_i}; \overleftarrow{h_i}\right]$$

$$\mathrm{BidirLastPool}(\overrightarrow{h_1}, \ldots, \overrightarrow{h_{|\overline{x}|}}; \overleftarrow{h_1}, \ldots, \overleftarrow{h_{|\overline{x}|}}) = \left[\overrightarrow{h_{|\overline{x}|}}; \overleftarrow{h_{|\overline{x}|}}\right]$$

Recap: Sequence Embeddings

- What can we do with sequence embeddings?
  - Text classification

$$f_\theta(\overline{x}) = \sigma(W\mathrm{Enc}(\overline{x}) + b) \qquad f : \mathcal{V}^+ \to \mathcal{C}$$

E.g., sentiment classification, language identification, ...

the first hour of this is a complete slog... ⟶ Negative

no me tires ⟶ Spanish

("don't throw me away")

Recap: Sequence Embeddings

- What can we do with sequence embeddings?
  - Text classification

$$f_\theta(\overline{x}) = \sigma(W\mathrm{Enc}(\overline{x}) + b) \qquad f : \mathcal{V}^+ \to \mathcal{C}$$

  - Multi-sentence classification, e.g., natural language inference

Natural Language Inference

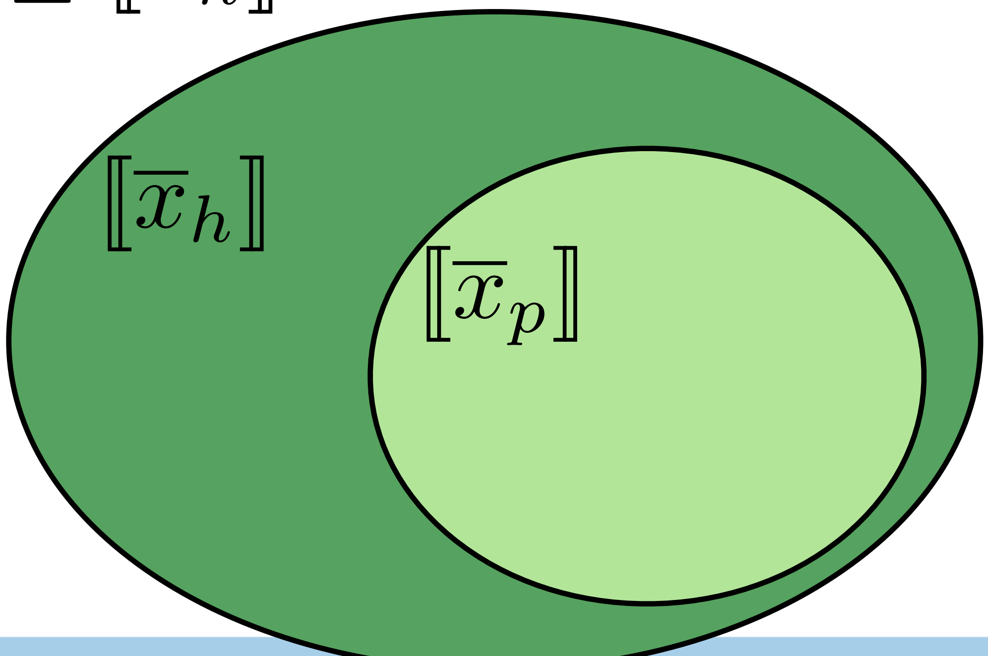
- Aka: recognizing textual entailment
- Given two sentences $\overline{x}_p$ (*premise*) and $\overline{x}_h$ (*hypothesis*), determine the following:
  - If $\overline{x}_p$ is true, then is $\overline{x}_h$ always true?

Natural Language Inference

- Aka: recognizing textual entailment
- Given two sentences $\overline{x}_p$(*premise*) and $\overline{x}_h$ (*hypothesis*), determine the following:
  - If $\overline{x}_p$ is true, then is $\overline{x}_h$ always true?
  - 3 possible labels:
    - Entailment: $\overline{x}_p \vDash \overline{x}_h \equiv [\![\overline{x}_h]\!] \subseteq [\![\overline{x}_h]\!]$

$\overline{x}_p =$ *I was born in Iowa.*

$\overline{x}_h =$ *I was born in the US.*

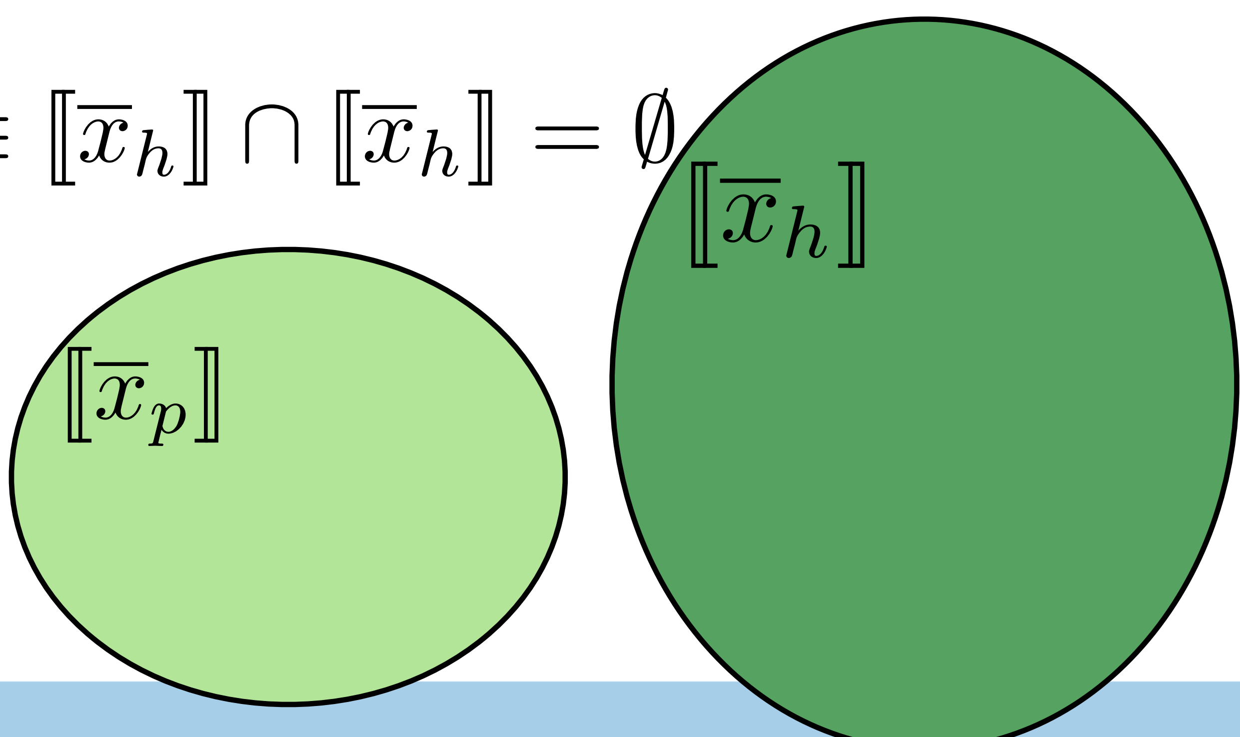


Natural Language Inference

- Aka: recognizing textual entailment
- Given two sentences $\overline{x}_p$(*premise*) and $\overline{x}_h$ (*hypothesis*), determine the following:
    - If $\overline{x}_p$ is true, then is $\overline{x}_h$ always true?
    - 3 possible labels:
        - Entailment: $\overline{x}_p \vDash \overline{x}_h$
        - Contradiction: $\overline{x}_p \vDash \neg\overline{x}_h \equiv [\![\overline{x}_h]\!] \cap [\![\overline{x}_h]\!] = \emptyset$

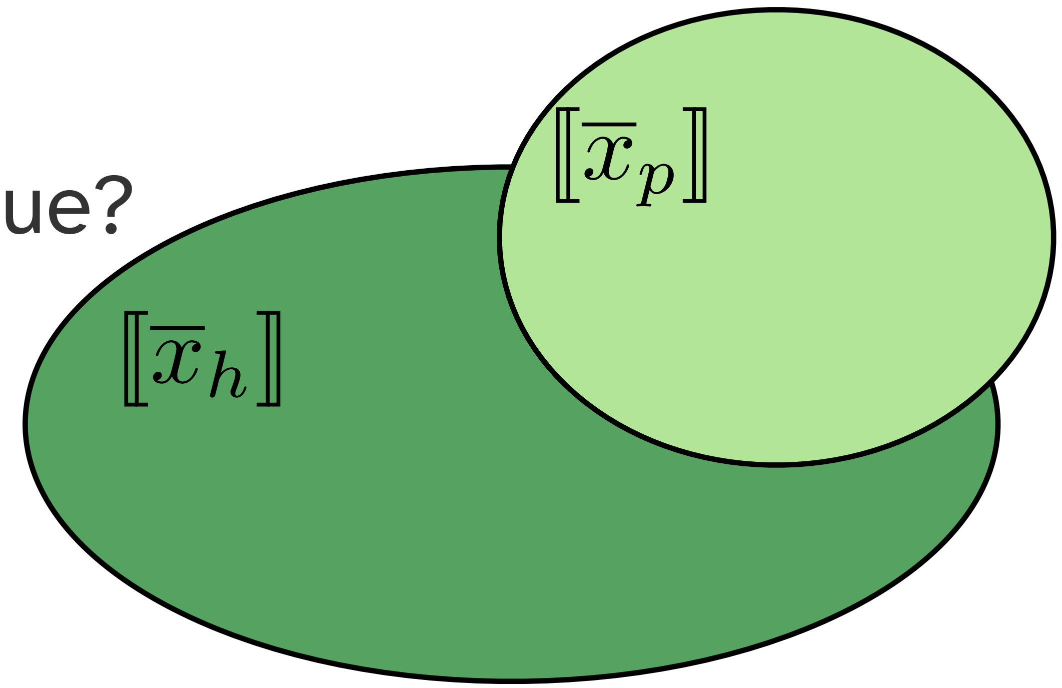
$\overline{x}_p =$ *I was born in Beijing.*

$\overline{x}_h =$ *I was born in the US.*



Natural Language Inference

- Aka: recognizing textual entailment
- Given two sentences $\overline{x}_p$(*premise*) and $\overline{x}_h$ (*hypothesis*), determine the following:
  - If $\overline{x}_p$ is true, then is $\overline{x}_h$ always true?
  - 3 possible labels:
    - Entailment: $\overline{x}_p \models \overline{x}_h$
    - Contradiction: $\overline{x}_p \models \neg\overline{x}_h$
    - Neutral: $\overline{x}_p \not\models \overline{x}_h \equiv (\llbracket\overline{x}_h\rrbracket \cap \llbracket\overline{x}_h\rrbracket \neq \emptyset) \wedge (\llbracket\overline{x}_h\rrbracket \not\subseteq \llbracket\overline{x}_h\rrbracket)$



$\overline{x}_p =$ *I was born in Beijing.*

$\overline{x}_h =$ *I live in the US.*

Natural Language Inference

- Aka: recognizing textual entailment
- Given two sentences $\overline{x}_p$(*premise*) and $\overline{x}_h$ (*hypothesis*), determine the following:
  - If $\overline{x}_p$ is true, then is $\overline{x}_h$ always true?
  - 3 possible labels:
    - Entailment: $\overline{x}_p \models \overline{x}_h$
    - Contradiction: $\overline{x}_p \models \neg\overline{x}_h$
    - Neutral: $\overline{x}_p \not\models \overline{x}_h$

$$f : \mathcal{V}^+ \times \mathcal{V}^+ \to \Delta^{\{\text{entailment},\text{contradiction},\text{neutral}\}}$$

- How might we use sentence embeddings to implement f ?

Natural Language Inference

$$f : \mathcal{V}^+ \times \mathcal{V}^+ \to \Delta^{\{\text{entailment},\text{contradiction},\text{neutral}\}}$$

vector concatenation

$$f(\overline{x}_p, \overline{x}_h) = \text{softmax}(W\,[\text{Enc}(\overline{x}_p); \text{Enc}(\overline{x}_h)] + b)$$

text encoding of premise

text encoding of hypothesis

Natural Language Inference

$$f : \mathcal{V}^+ \times \mathcal{V}^+ \to \Delta^{\{\text{entailment},\text{contradiction},\text{neutral}\}}$$

$$f(\overline{x}_p, \overline{x}_h) = \mathrm{softmax}(W \overbrace{[\underbrace{\mathrm{Enc}(\overline{x}_p)}_{\text{text encoding of premise}}; \underbrace{\mathrm{Enc}(\overline{x}_h)}_{\text{text encoding of hypothesis}}]}^{\text{vector concatenation}} + b)$$

- Train using labeled data
  $\overline{x}_p, \overline{x}_h, y \in \{\text{entailment}, \text{contradiction}, \text{neutral}\}$

Natural Language Inference

$$f : \mathcal{V}^+ \times \mathcal{V}^+ \rightarrow \Delta^{\{\text{entailment},\text{contradiction},\text{neutral}\}}$$

$$f(\overline{x}_p, \overline{x}_h) = \text{softmax}(W \underbrace{\overbrace{[\underbrace{\text{Enc}(\overline{x}_p)}_{\text{text encoding of premise}}; \underbrace{\text{Enc}(\overline{x}_h)}_{\text{text encoding of hypothesis}}]}^{\text{vector concatenation}}}_{} + b)$$

- Train using labeled data
  $\overline{x}_p, \overline{x}_h, y \in \{\text{entailment}, \text{contradiction}, \text{neutral}\}$

- Can be used to measure quality of sentence representations!

Recap: Sequence Embeddings

- What can we do with sequence embeddings?
  - Text classification

$$f_\theta(\overline{x}) = \sigma(W \mathrm{Enc}(\overline{x}) + b) \qquad f : \mathcal{V}^+ \to \mathcal{C}$$

  - Multi-sentence classification, e.g., natural language inference

$$f : \mathcal{V}^+ \times \mathcal{V}^+ \to \Delta^{\{\text{entailment},\text{contradiction},\text{neutral}\}}$$

$$f(\overline{x}_p, \overline{x}_h) = \mathrm{softmax}(W\,[\mathrm{Enc}(\overline{x}_p); \mathrm{Enc}(\overline{x}_h)] + b)$$

  - Retrieval

$$\arg\max_{\overline{x}' \in \mathcal{D}} \mathrm{sim}(\mathrm{Enc}(\overline{x}), \mathrm{Enc}(\overline{x}'))$$

set of reference documents

query document

Recap: Sequence Embeddings

- What can we do with sequence embeddings?
  - Text classification

$$f_\theta(\overline{x}) = \sigma(W\mathrm{Enc}(\overline{x}) + b) \qquad f : \mathcal{V}^+ \rightarrow \mathcal{C}$$

  - Multi-sentence classification, e.g., natural language inference

$$f : \mathcal{V}^+ \times \mathcal{V}^+ \rightarrow \Delta^{\{\mathrm{entailment},\mathrm{contradiction},\mathrm{neutral}\}}$$

$$f(\overline{x}_p, \overline{x}_h) = \mathrm{softmax}(W\left[\mathrm{Enc}(\overline{x}_p); \mathrm{Enc}(\overline{x}_h)\right] + b)$$

  - Retrieval

$$\arg\max_{\overline{x}' \in \mathcal{D}} \mathrm{sim}(\mathrm{Enc}(\overline{x}), \mathrm{Enc}(\overline{x}'))$$

- **Conditional sequence generation!**
  - Aka “sequence transduction”
  - E.g., machine translation, response generation in dialogue

Sequence Transduction: Machine Translation

- **Task:** map from text in one language (L1) to a distribution over texts in another language (L2), assigning high probability to texts that preserves the input text meaning

$$f : \mathcal{V}_{\mathrm{L1}}^{+} \to \Delta^{\mathcal{V}_{\mathrm{L2}}^{+}}$$

$\overline{x}_{\mathrm{English}} =$ *Oula A. Alrifai is a Syrian emigrant to the United States and writer for various Washington-based think tanks.*

$\downarrow$ Google Translate

$\overline{x}_{\mathrm{Malay}} =$ *Oula A. Alrifai ialah seorang pendatang Syria ke Amerika Syarikat dan penulis untuk pelbagai badan pemikir yang berpangkalan di Washington.*

Sequence Transduction: Machine Translation

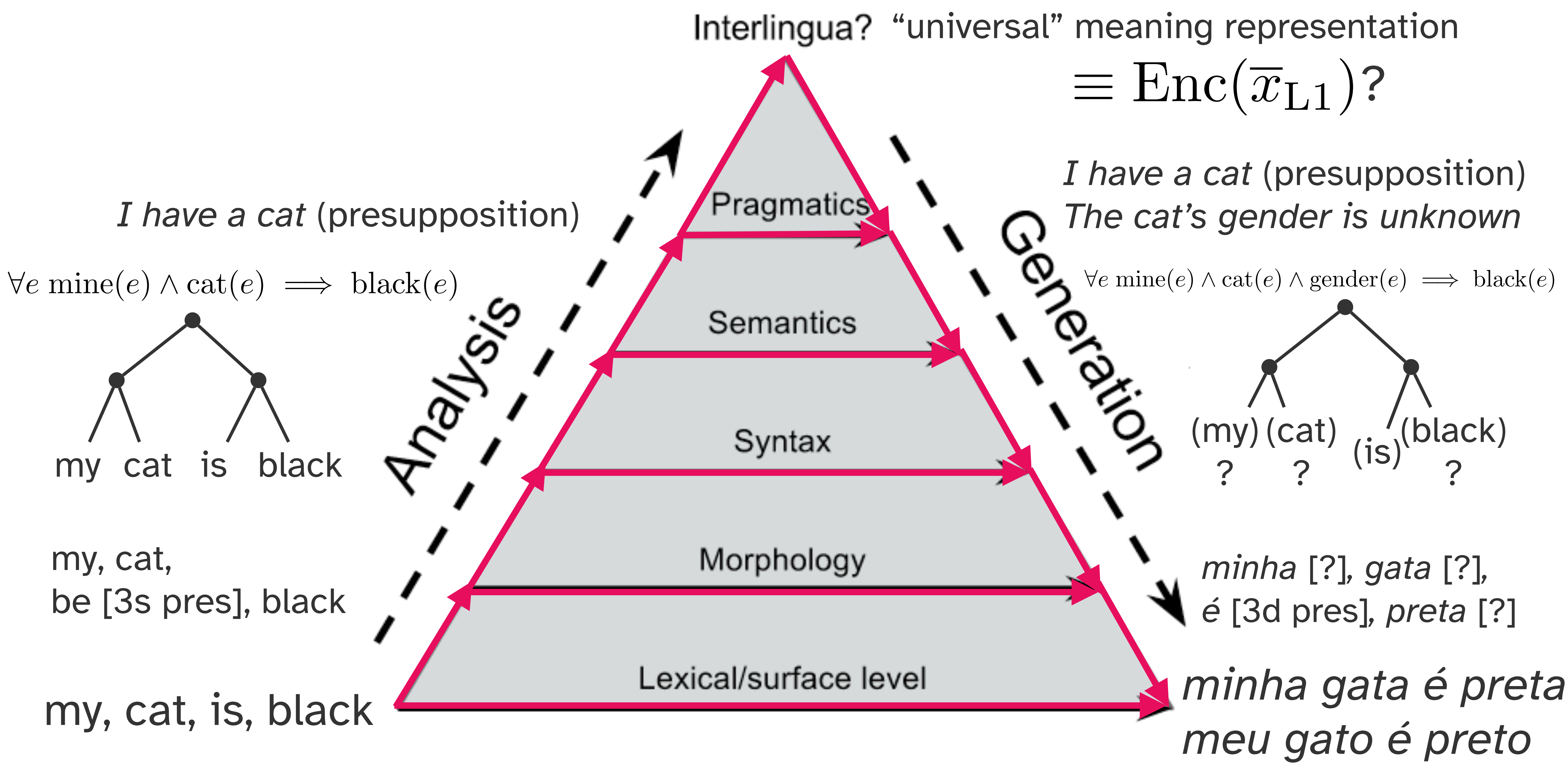
When I look at an article in Russian, I say: “This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.”

Warren Weaver (1949)

- Machine translation research began in the early 1950s on machines less powerful than high school calculators
    - Also, pre-“AI”!
- Heavily funded by military (which language?)

Sequence Transduction: Machine Translation

Vauquois triangle theory of machine translation



Sequence Transduction: Machine Translation

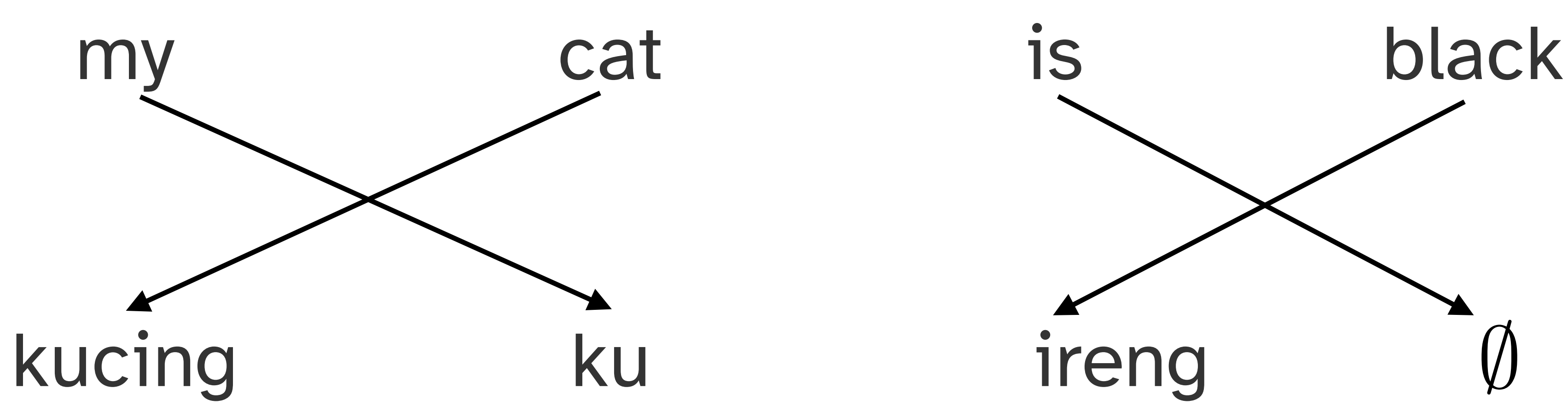
- End-to-end, we can think about the translation problem as generating an output sequence token-by-token by mapping from some input word to the target vocabulary, using the target language syntax:

my	cat	is	black
↓	↓	↓	↓
minha / meu	gata / gato	é	preta / preto

Portuguese

Sequence Transduction: Machine Translation

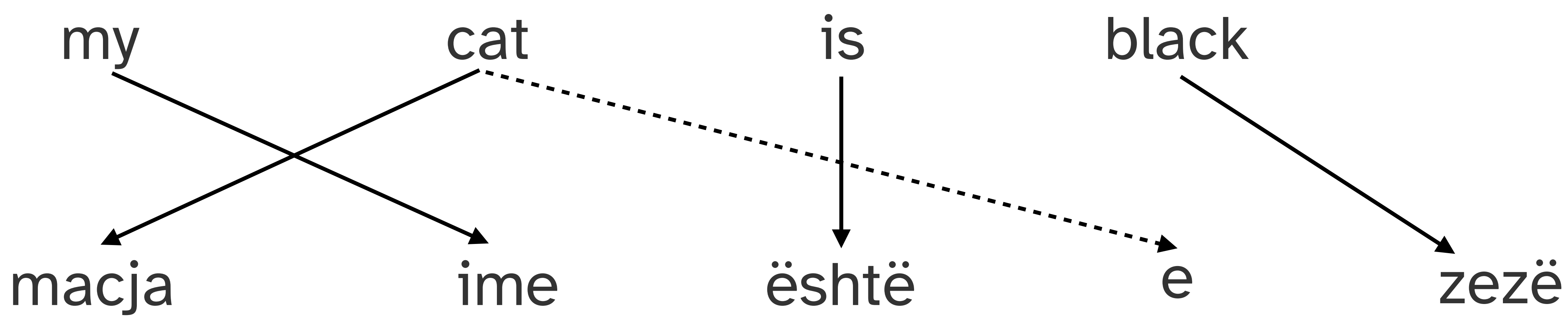
- End-to-end, we can think about the translation problem as generating an output sequence token-by-token by mapping from some input word to the target vocabulary, using the target language syntax:



Javanese

Sequence Transduction: Machine Translation

- End-to-end, we can think about the translation problem as generating an output sequence token-by-token by mapping from some input word to the target vocabulary, using the target language syntax:



Albanian

We'll talk more about alignment models later in the semester!

Putting it Together: Sequence-to-Sequence Models

- What do we get from sequence models?
  - Autoregressive (token-by-token) generation

my	cat	is	black
minha / meu	gata / gato	é	preta / preto

Putting it Together: Sequence-to-Sequence Models

- What do we get from sequence models?
  - Autoregressive (token-by-token) generation
  - Sentence embeddings $\mathrm{Enc}(\overline{x})$

Putting it Together: Sequence-to-Sequence Models

- What do we get from sequence models?
  - Autoregressive (token-by-token) generation
  - Sentence embeddings $\mathrm{Enc}(\overline{x})$
- **Key idea:** encode some input into a vector, decode it using autoregressive generation
  - Aka: sequence-to-sequence, encoder-decoder, sequence transduction, conditional language model…

Sequence-to-Sequence for Machine Translation

1. Encode input sentence: $\mathrm{Enc}(\overline{x}_{\mathrm{L1}})$

Sequence-to-Sequence for Machine Translation

1. Encode input sentence: $\mathrm{Enc}(\overline{x}_{\mathrm{L1}})$

2. Generate output sequence token-by-token, conditioning on previously-generated tokens **and the input sentence embedding:**

$$\overline{y}_{\mathrm{L2}} = \langle y_1, \ldots, y_n \rangle \quad y_n = \texttt{EOS}$$

$$y_i \sim p(Y_i \mid \underbrace{y_1, \ldots, y_{i-1}}_{\text{previously-generated tokens}}, \underbrace{\mathrm{Enc}(\overline{x}_{\mathrm{L1}})}_{\text{embedding of sequence to translate}}; \overbrace{\theta_{\mathrm{dec}}}^{\text{parameters of decoder model}})$$

parameters of decoder model (e.g., RNN parameters)

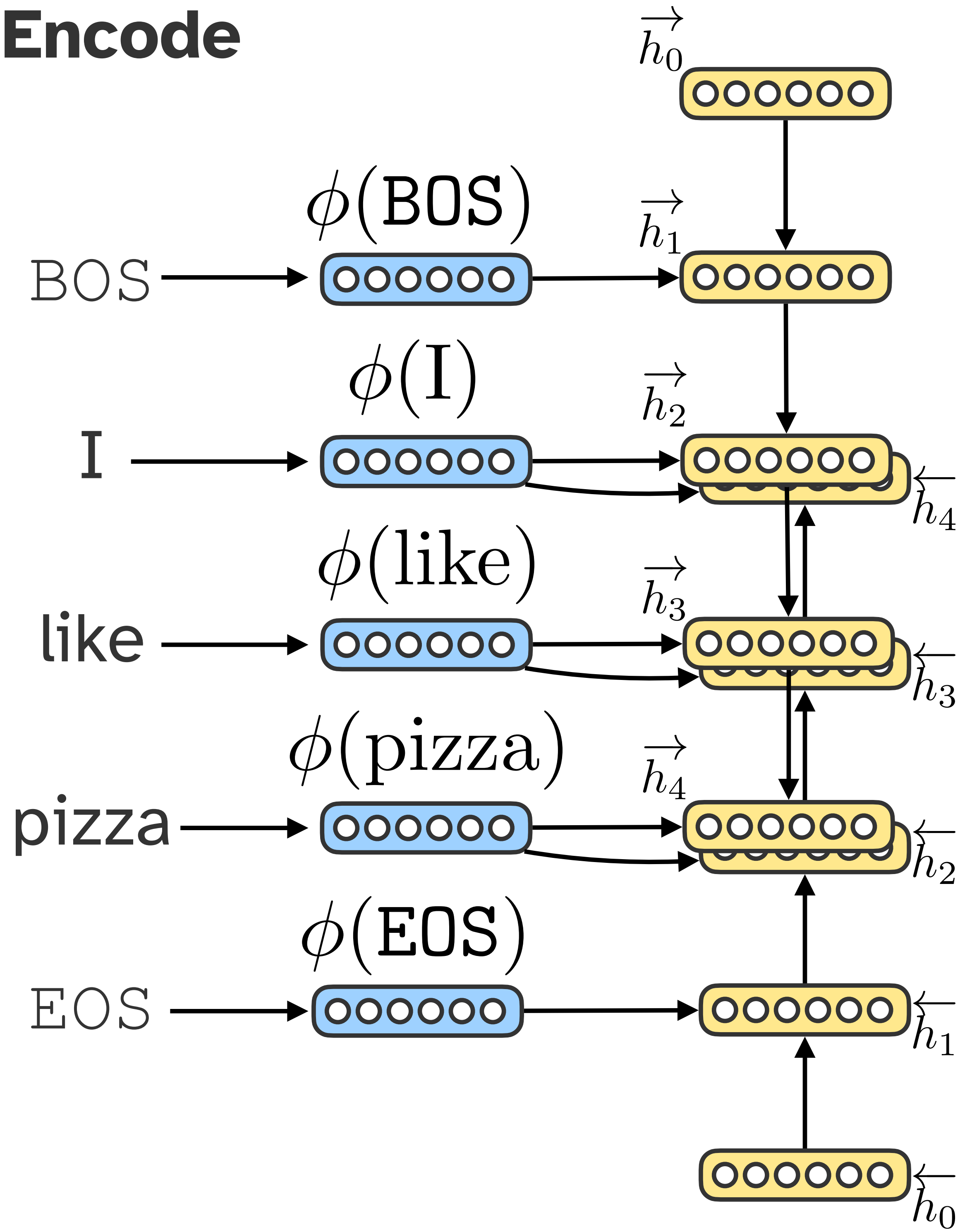
previously-generated tokens (empty when starting translation)
typically, we use *y* to denote output tokens

embedding of sequence to translate

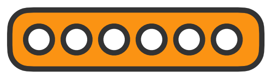
(you can condition on other things, too! e.g., images)

Sequence-to-Sequence with RNNs

Encode

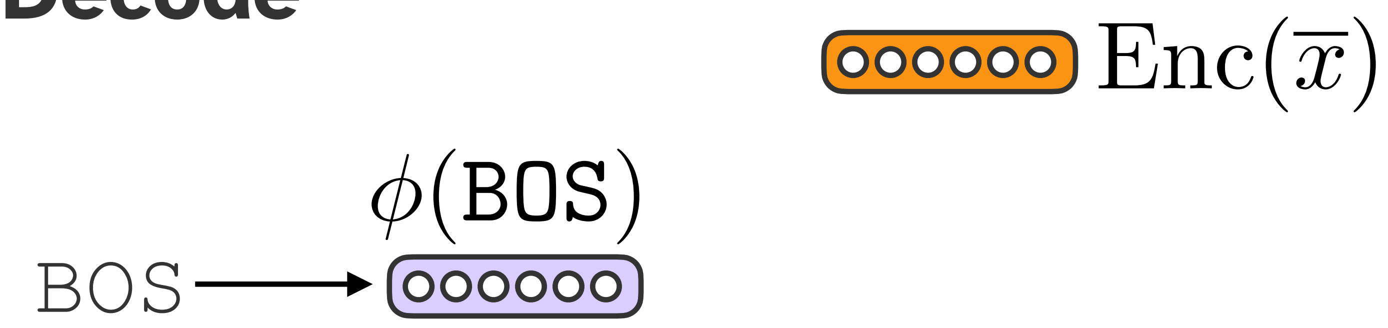


$$\mathrm{Enc}(\overline{x}) = \mathrm{Pool}\left(h_1, \ldots, h_{|\overline{x}|}\right) \in \mathbb{R}^d$$



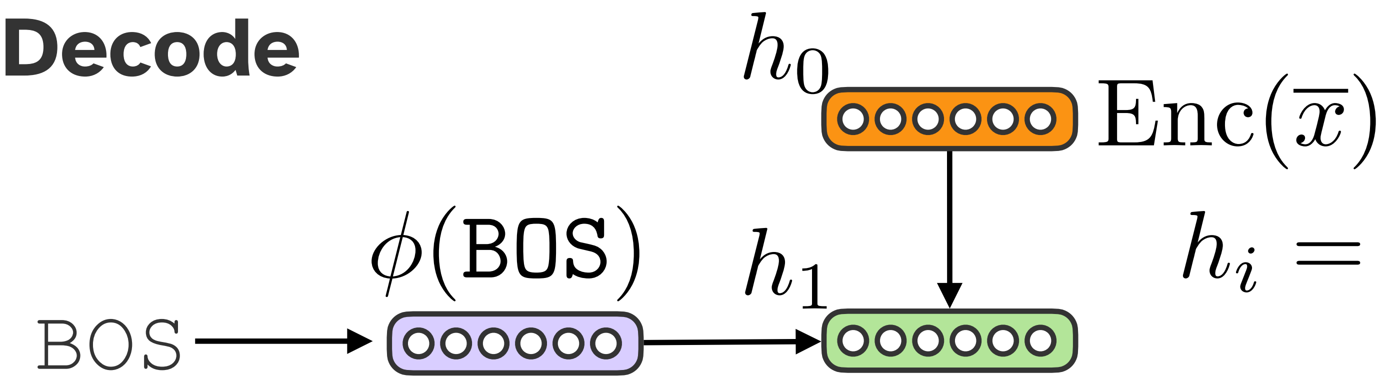
Sequence-to-Sequence with RNNs

Decode



Sequence-to-Sequence with RNNs

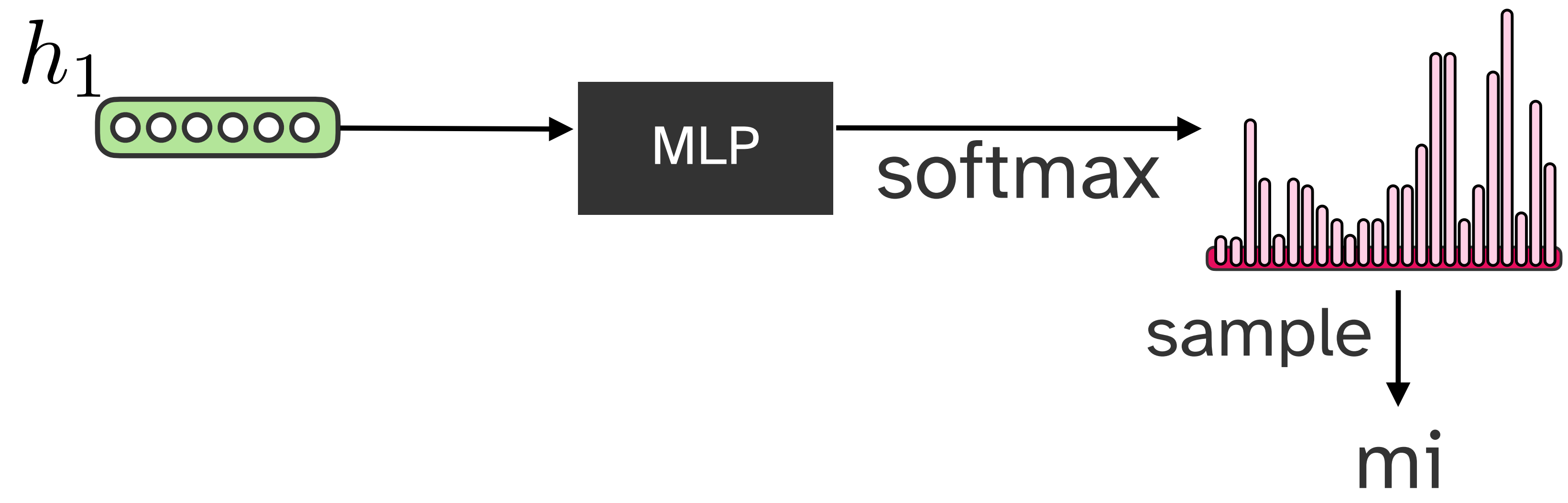
Decode



$$h_i = f(\phi(x_i), h_{i-1})$$

Sequence-to-Sequence with RNNs

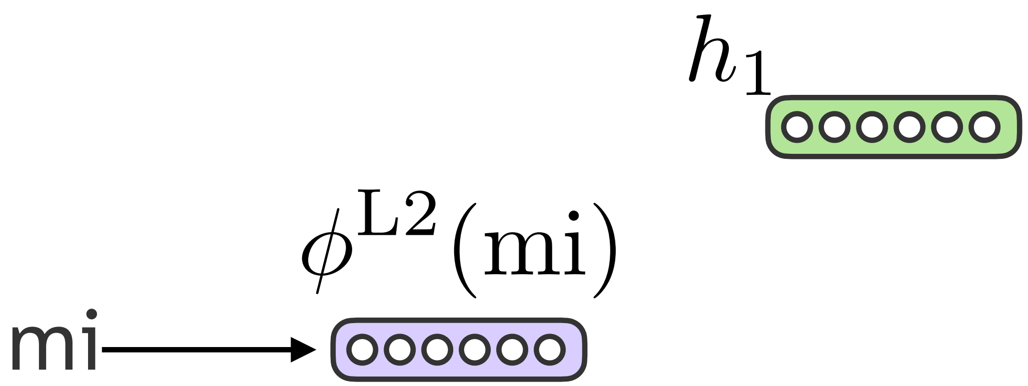
Decode



Sequence-to-Sequence with RNNs

Decode

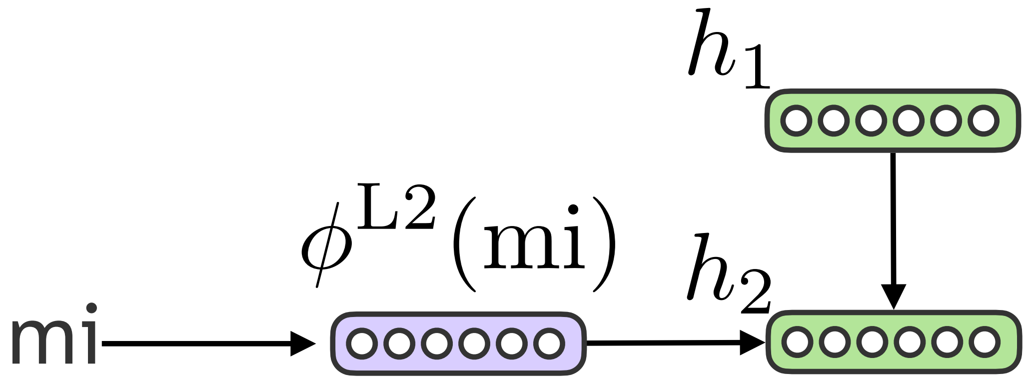
$\overline{y}_{\mathrm{L2}}$



Sequence-to-Sequence with RNNs

Decode

$\overline{y}_{\text{L2}}$



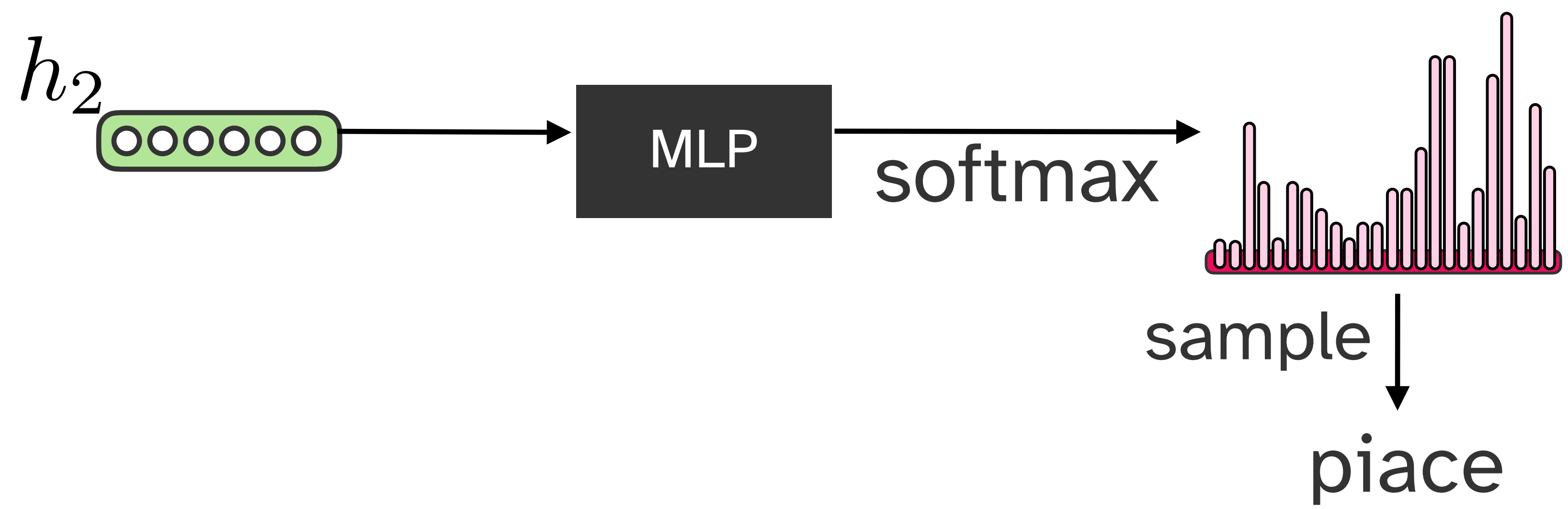
$$h_i = f(\phi(x_i), h_{i-1})$$

Sequence-to-Sequence with RNNs

Decode

$\overline{y}_{\mathrm{L2}}$

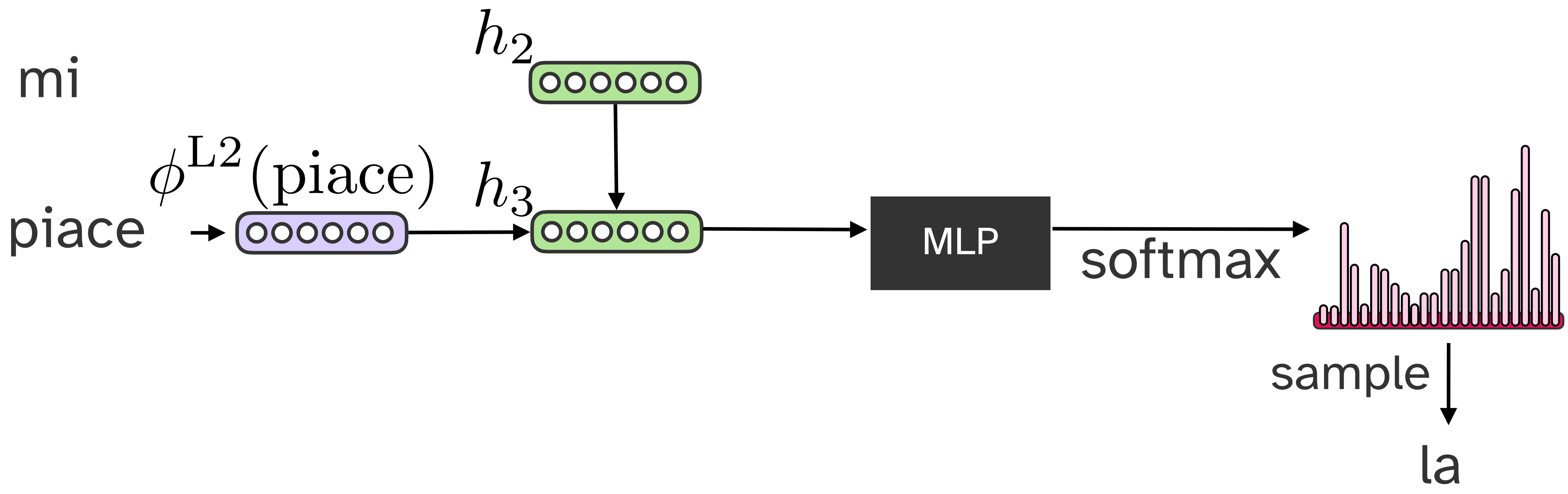
mi



Sequence-to-Sequence with RNNs

Decode

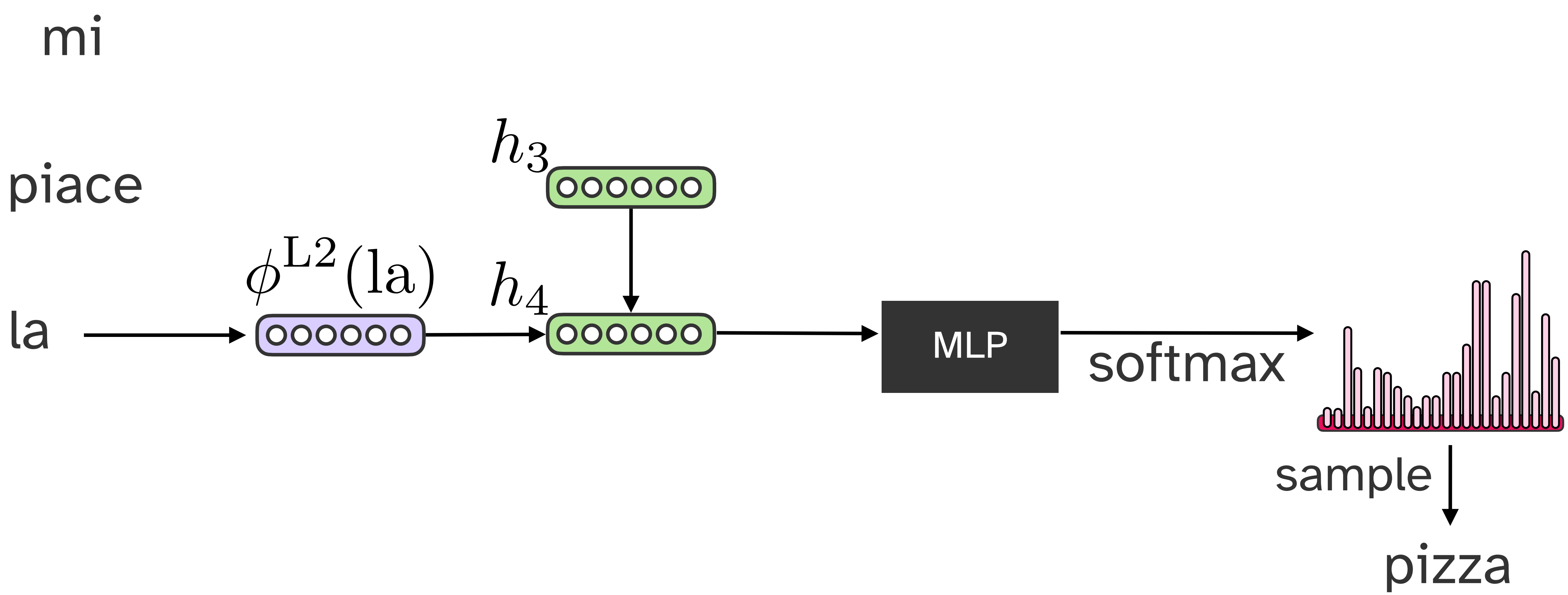
$\overline{y}_{\mathrm{L2}}$



Sequence-to-Sequence with RNNs

Decode

$\overline{y}_{\mathrm{L2}}$



Sequence-to-Sequence with RNNs

Decode

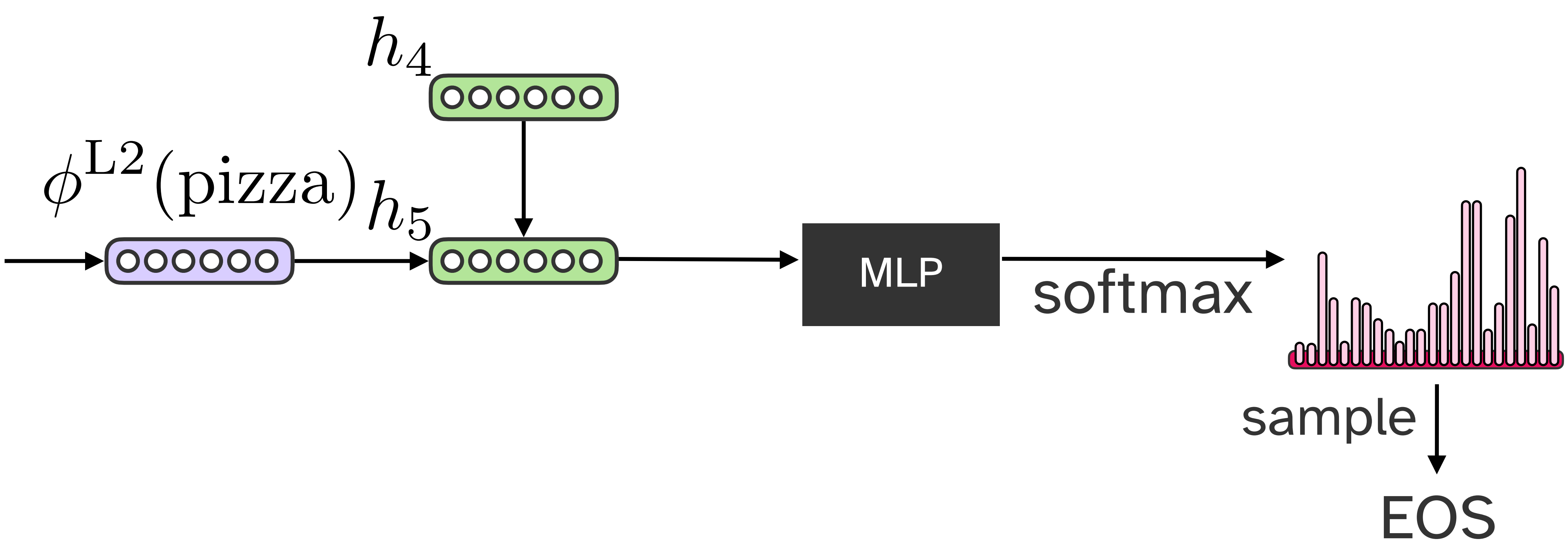
$\overline{y}_{\mathrm{L2}}$

mi

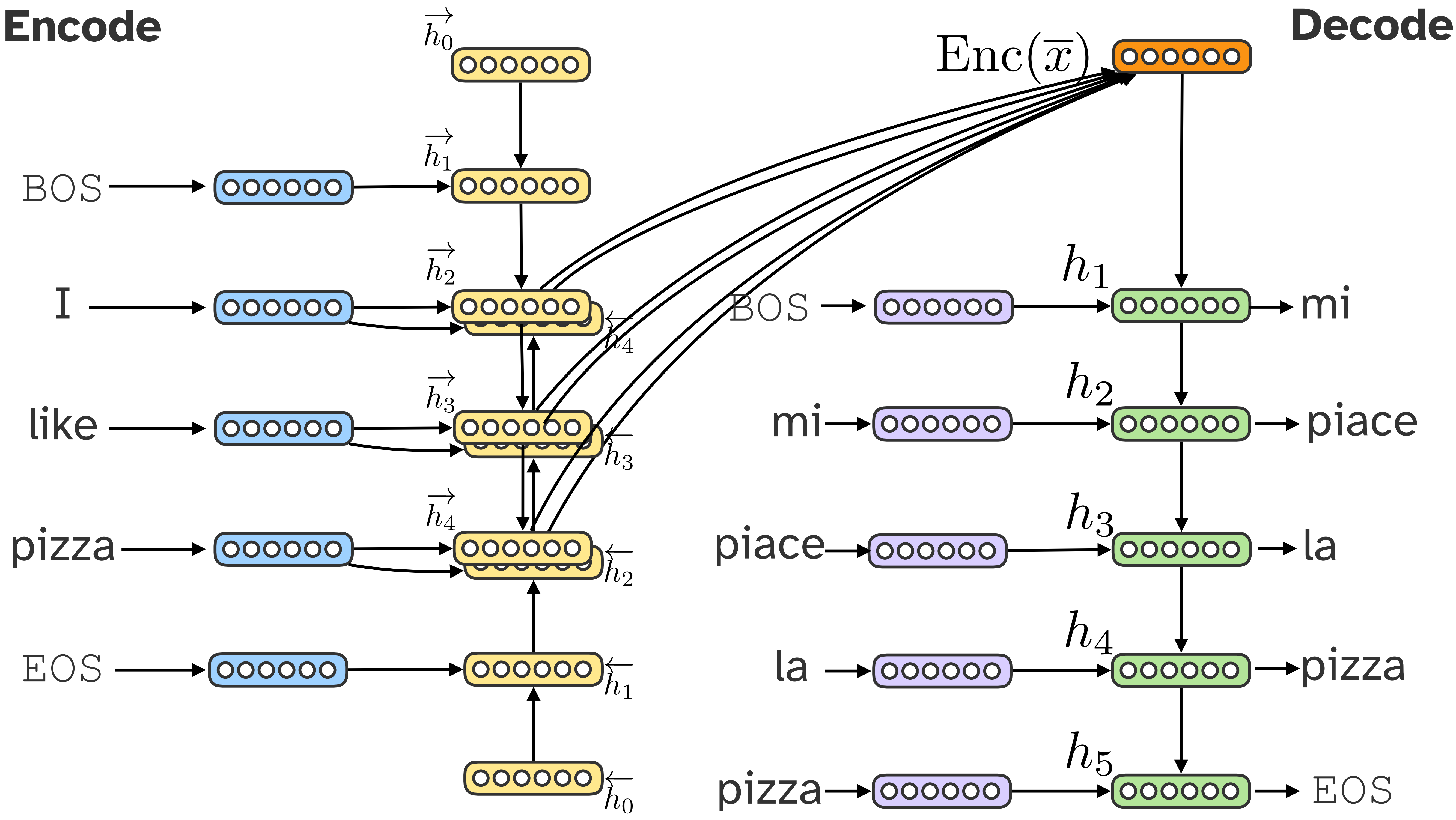
piace

la

pizza



Sequence-to-Sequence with RNNs



Training

- Given a training dataset of *bitexts* $\mathcal{D} = \left\{ \left(\overline{x}^{(i)}_{\mathrm{L1}}, \overline{y}^{(i)}_{\mathrm{L2}} \right) \right\}_{i=1}^{M}$
- We want to find parameters that maximize the probability of target sequences $\overline{y}$ conditioned on input sequence $\overline{x}$

$$\theta^* = \arg\max_\theta \left(- \sum_{i=1}^{M} \sum_{j=1}^{\left|\overline{y}^{(i)}_{\mathrm{L2}}\right|} \log p \left(y^{(i)}_j \mid \underbrace{y^{(i)}_1, \ldots, y^{(i)}_{j-1}}_{\text{“previously-generated” (teacher-forcing) tokens for example (i)}}, \underbrace{\mathrm{Enc}_{\theta_{\mathrm{enc}}} \left(\overline{x}^{(i)}_{\mathrm{L1}} \right)}_{\text{embedding of sequence to translate}} ; \underbrace{\theta_{\mathrm{dec}}}_{\text{parameters of decoder model}} \right) \right)$$

Training

- Given a training dataset of *bitexts* $\mathcal{D} = \left\{ \left(\overline{x}_{\mathrm{L1}}^{(i)}, \overline{y}_{\mathrm{L2}}^{(i)} \right) \right\}_{i=1}^{M}$
- We want to find parameters that maximize the probability of target sequences $\overline{y}$ conditioned on input sequence $\overline{x}$

$$\theta^* = \arg\max_{\theta} \left(- \sum_{i=1}^{M} \sum_{j=1}^{\left|\overline{y}_{\mathrm{L2}}^{(i)}\right|} \log p \left(y_j^{(i)} \mid \underbrace{y_1^{(i)}, \ldots, y_{j-1}^{(i)},}_{\substack{\text{“previously-generated”} \\ \text{(teacher-forcing) tokens} \\ \text{for example (i)}}} \underbrace{\mathrm{Enc}_{\theta_{\mathrm{enc}}} \left(\overline{x}_{\mathrm{L1}}^{(i)} \right)}_{\substack{\text{embedding of} \\ \text{sequence to} \\ \text{translate}}} ; \underbrace{\theta_{\mathrm{dec}}}_{\substack{\text{parameters of} \\ \text{decoder model}}} \right) \right)$$

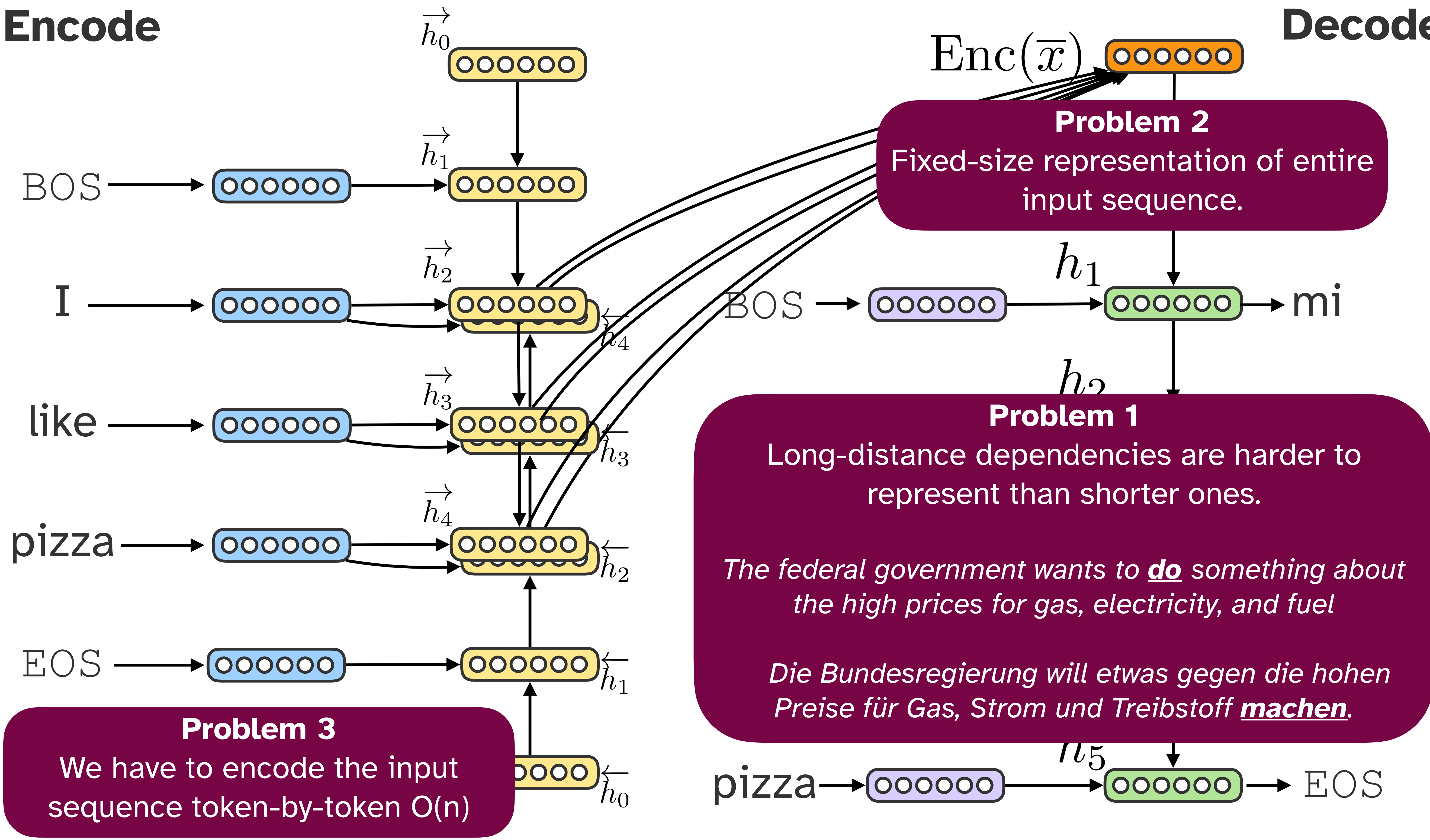
- Options for backprop: use pre-trained θ_{enc} and freeze, or finetune by backprop directly $\theta = \theta_{\mathrm{enc}} \cup \theta_{\mathrm{dec}}$

Sequence-to-Sequence Tasks

$$p(y \mid x)$$

- Machine translation
- Summarization (long text → short text)
- Dialogue (previous utterances → next utterance)
- Syntactic parsing (input sentence → serialized parse)
- Code generation (natural language → Python code)

Limitations of Vanilla Seq2Seq RNN



Problem 2
Fixed-size representation of entire input sequence.

Problem 1
Long-distance dependencies are harder to represent than shorter ones.

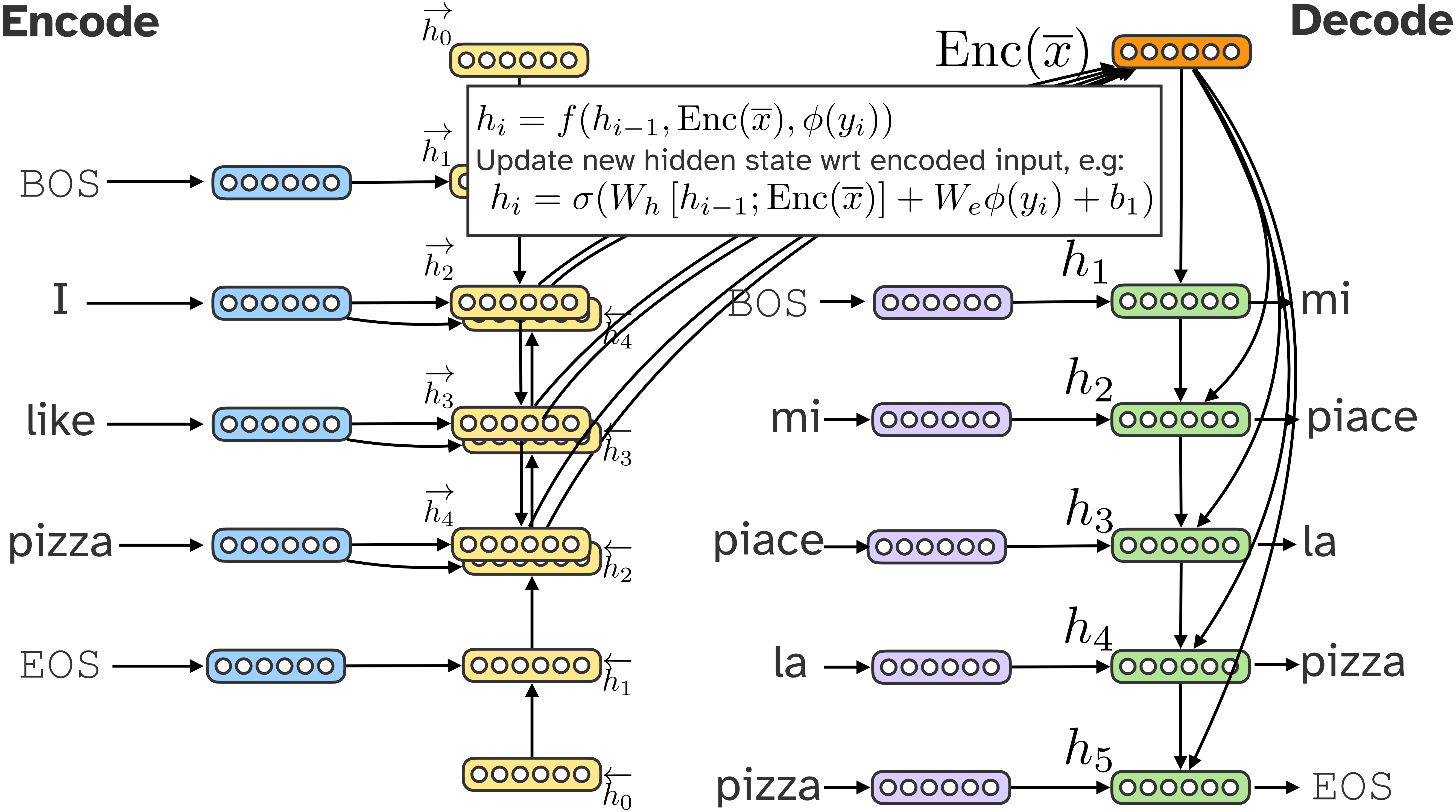
*The federal government wants to **do** something about the high prices for gas, electricity, and fuel*

*Die Bundesregierung will etwas gegen die hohen Preise für Gas, Strom und Treibstoff **machen**.*

Problem 3
We have to encode the input sequence token-by-token O(n)

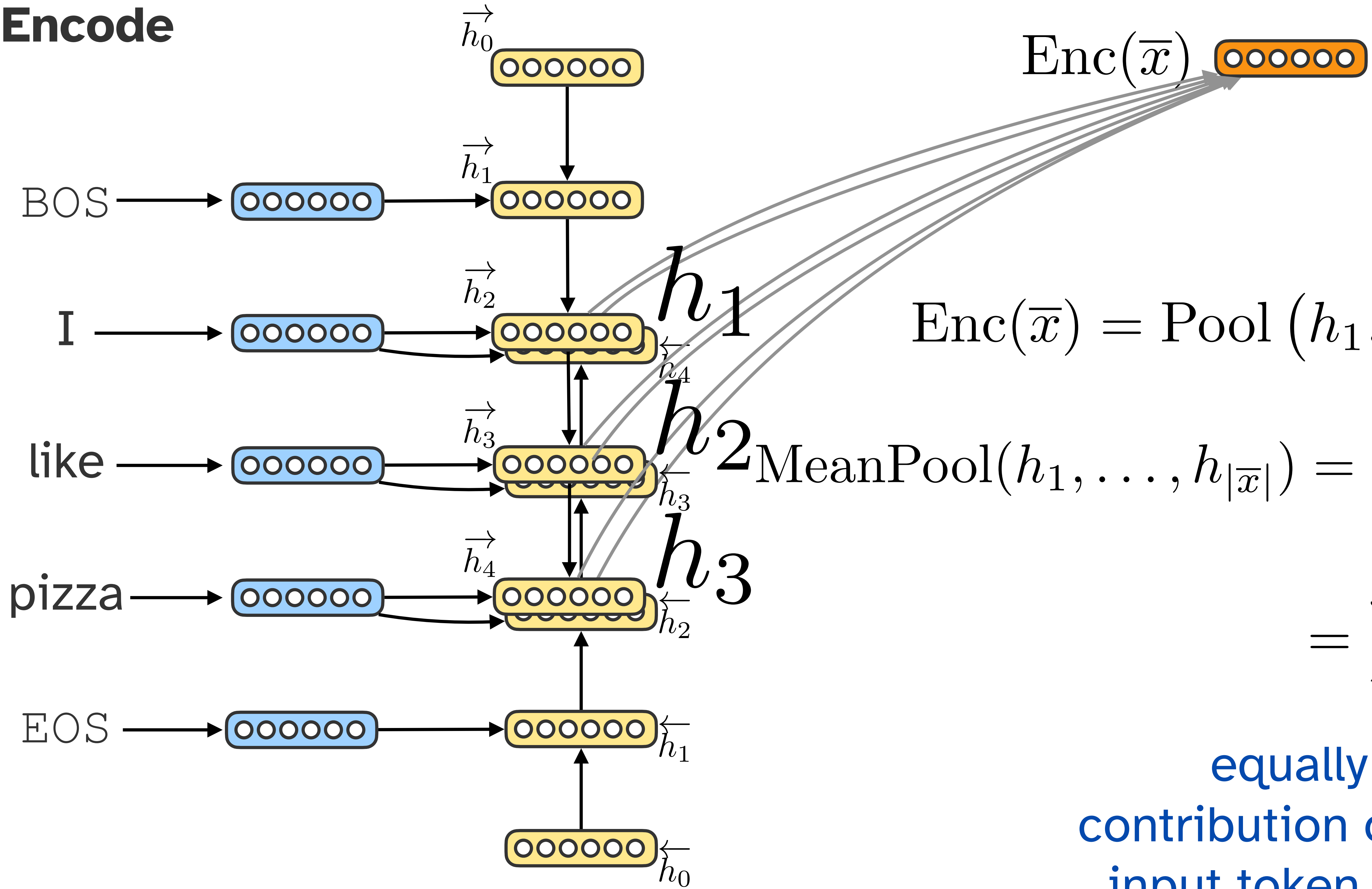
Problem 1: Long-Distance Dependencies

Encode **Decode**



Problem 2: Fixed-Size Sentence Embedding

Encode



$$\mathrm{Enc}(\overline{x}) = \mathrm{Pool}\left(h_1, \ldots, h_{|\overline{x}|}\right)$$

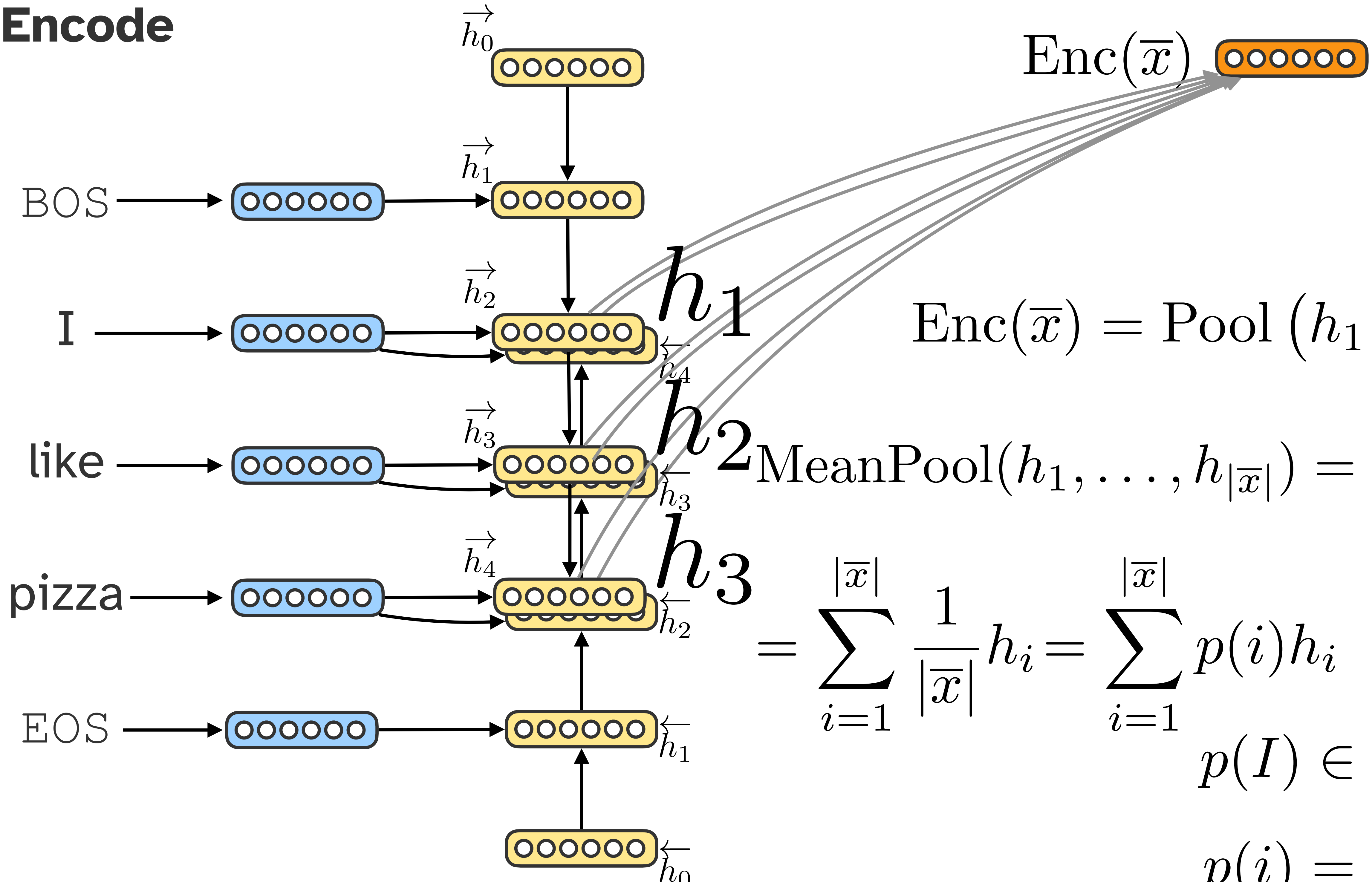
$$\mathrm{MeanPool}(h_1, \ldots, h_{|\overline{x}|}) = \frac{1}{|\overline{x}|} \sum_{i=1}^{|\overline{x}|} h_i$$

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

equally weight contribution of each input token / index

Problem 2: Fixed-Size Sentence Embedding

Encode



$$\mathrm{Enc}(\overline{x}) = \mathrm{Pool}\left(h_1, \ldots, h_{|\overline{x}|}\right)$$

$$\mathrm{MeanPool}(h_1, \ldots, h_{|\overline{x}|}) = \frac{1}{|\overline{x}|}\sum_{i=1}^{|\overline{x}|} h_i$$

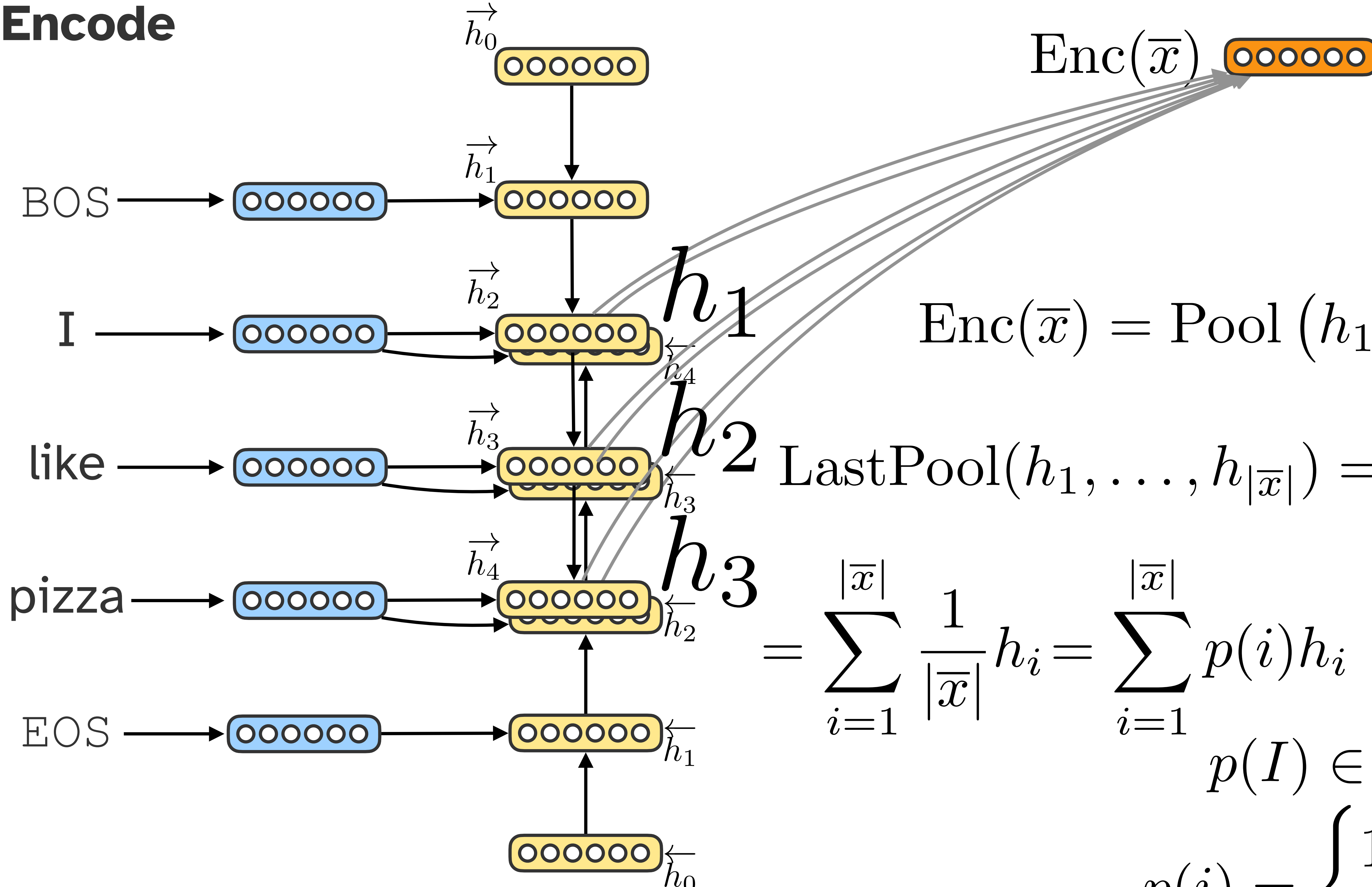
$$= \sum_{i=1}^{|\overline{x}|} \frac{1}{|\overline{x}|} h_i = \sum_{i=1}^{|\overline{x}|} p(i) h_i$$

$$p(I) \in \Delta^{\mathbb{N}_{1:|\overline{x}|}}$$

$$p(i) = \frac{1}{|\overline{x}|}$$

Problem 2: Fixed-Size Sentence Embedding

Encode



$$\mathrm{Enc}(\overline{x}) = \mathrm{Pool}\left(h_1, \ldots, h_{|\overline{x}|}\right)$$

$$\mathrm{LastPool}(h_1, \ldots, h_{|\overline{x}|}) = h_{|\overline{x}|}$$

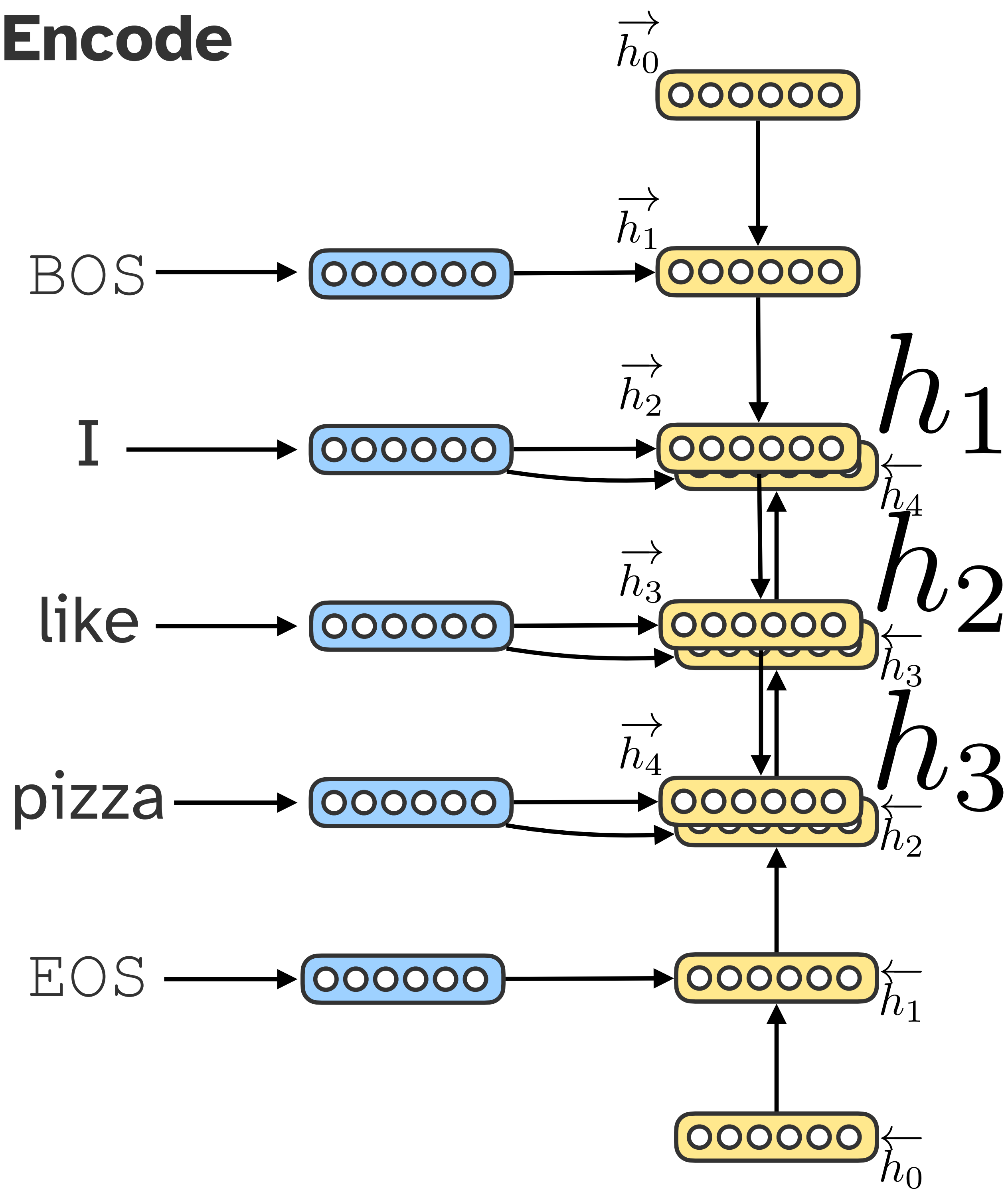
$$= \sum_{i=1}^{|\overline{x}|} \frac{1}{|\overline{x}|} h_i = \sum_{i=1}^{|\overline{x}|} p(i) h_i$$

$$p(I) \in \Delta^{\mathbb{N}_{1:|\overline{x}|}}$$

$$p(i) = \begin{cases} 1 & i = |\overline{x}| \\ 0 & i < |\overline{x}| \end{cases}$$

Attention

Encode



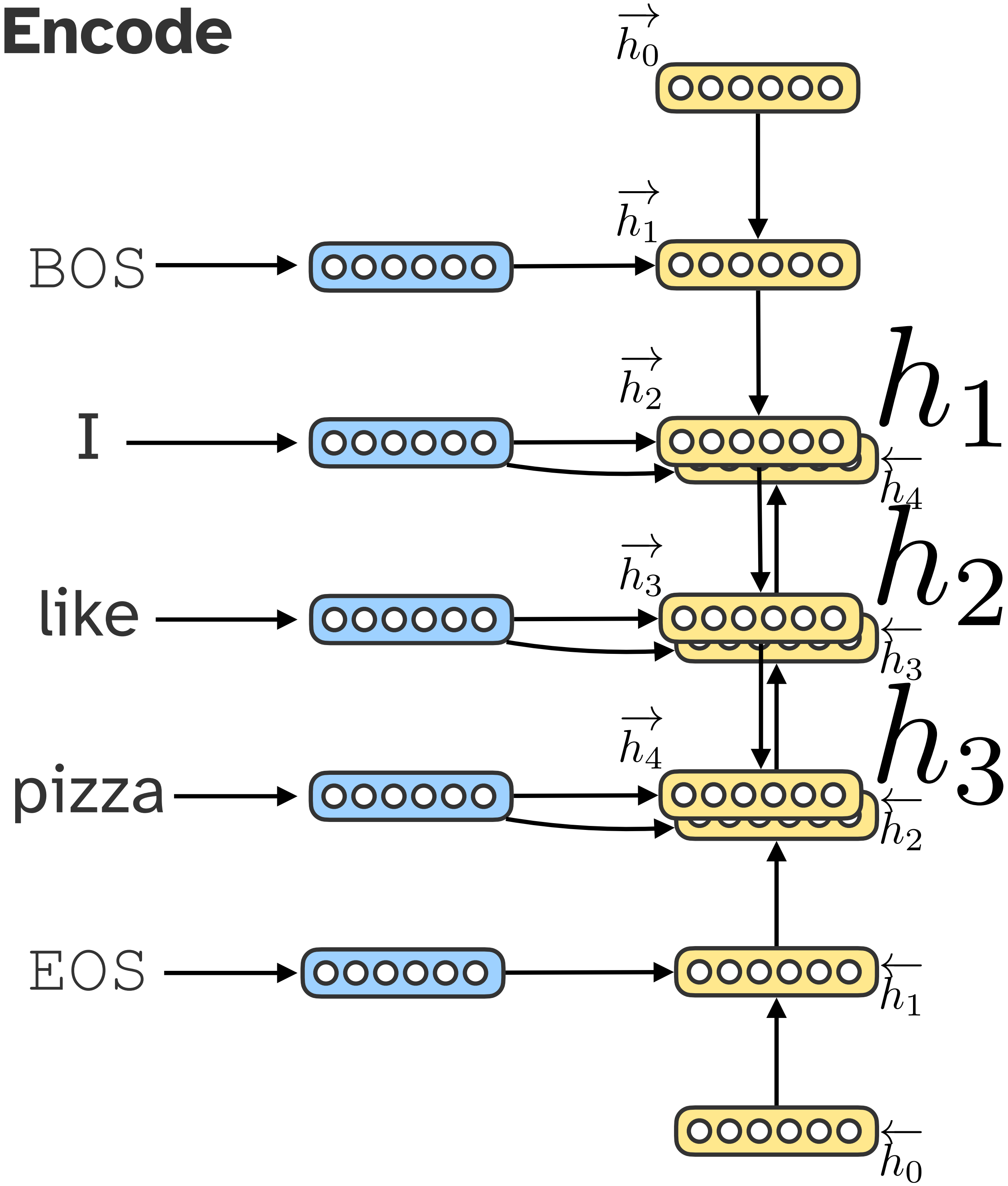
$$\mathrm{Enc}(\overline{x}) = \sum_{i=1}^{|\overline{x}|} p(i) h_i$$

$$p(I) \in \Delta^{\mathbb{N}_{1:|\overline{x}|}}$$

- Generalization of pooling: assign a **probability distribution** over input indices, and compute **weighted** sum over hidden states
- What if this probability distribution could change during generation?

Attention

Encode

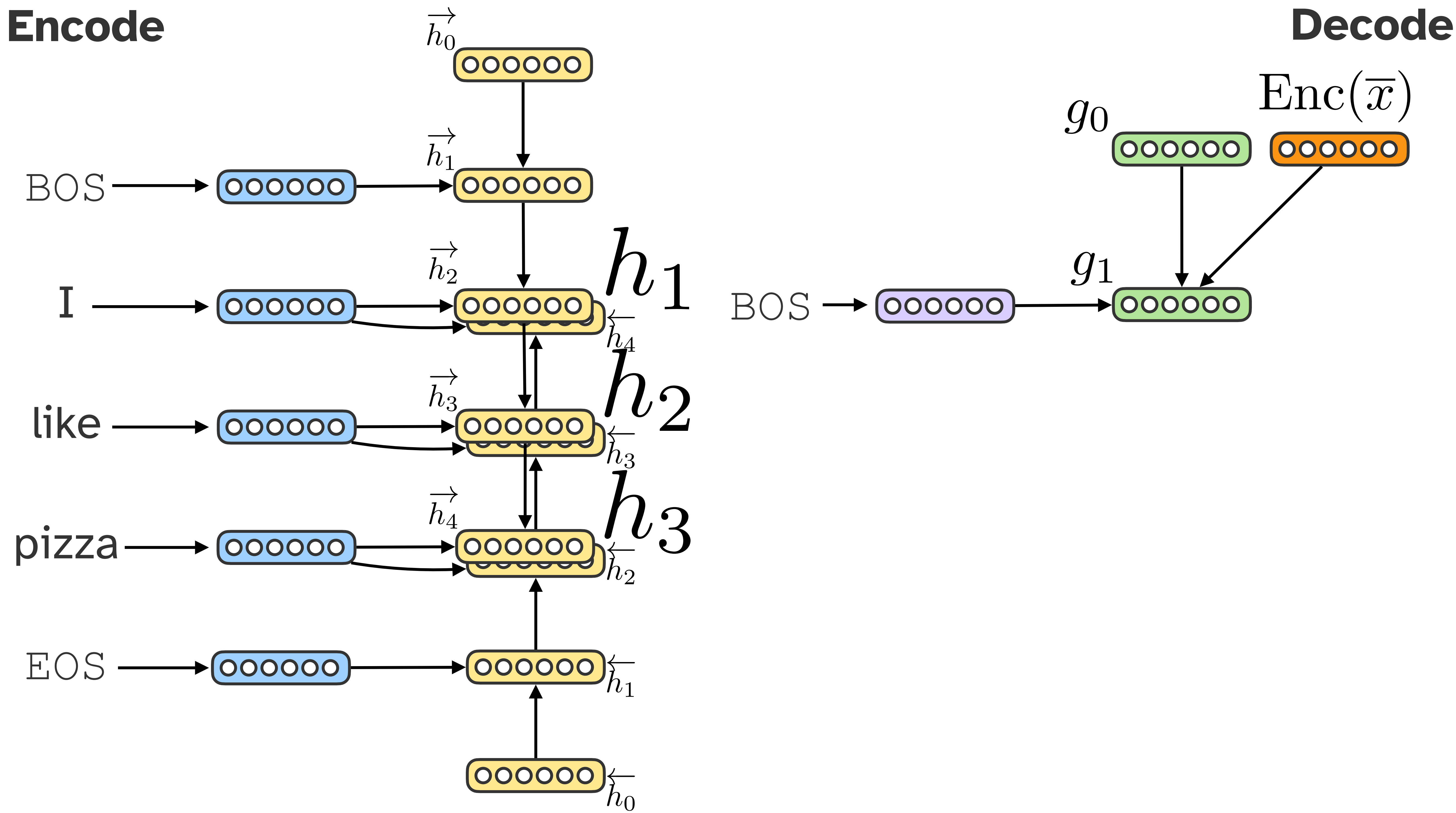


$$\mathrm{Enc}(\overline{x}, g) = \sum_{i=1}^{|\overline{x}|} p(i \mid g) h_i$$

$$g \in \mathbb{R}^d \quad p : \mathbb{R}^d \to \Delta^{\mathbb{N}_{1:|\overline{x}|}}$$

- Compute a new distribution over input indices depending on some other vector g
- Two questions:
  - Where does g come from?
  - How to implement p?

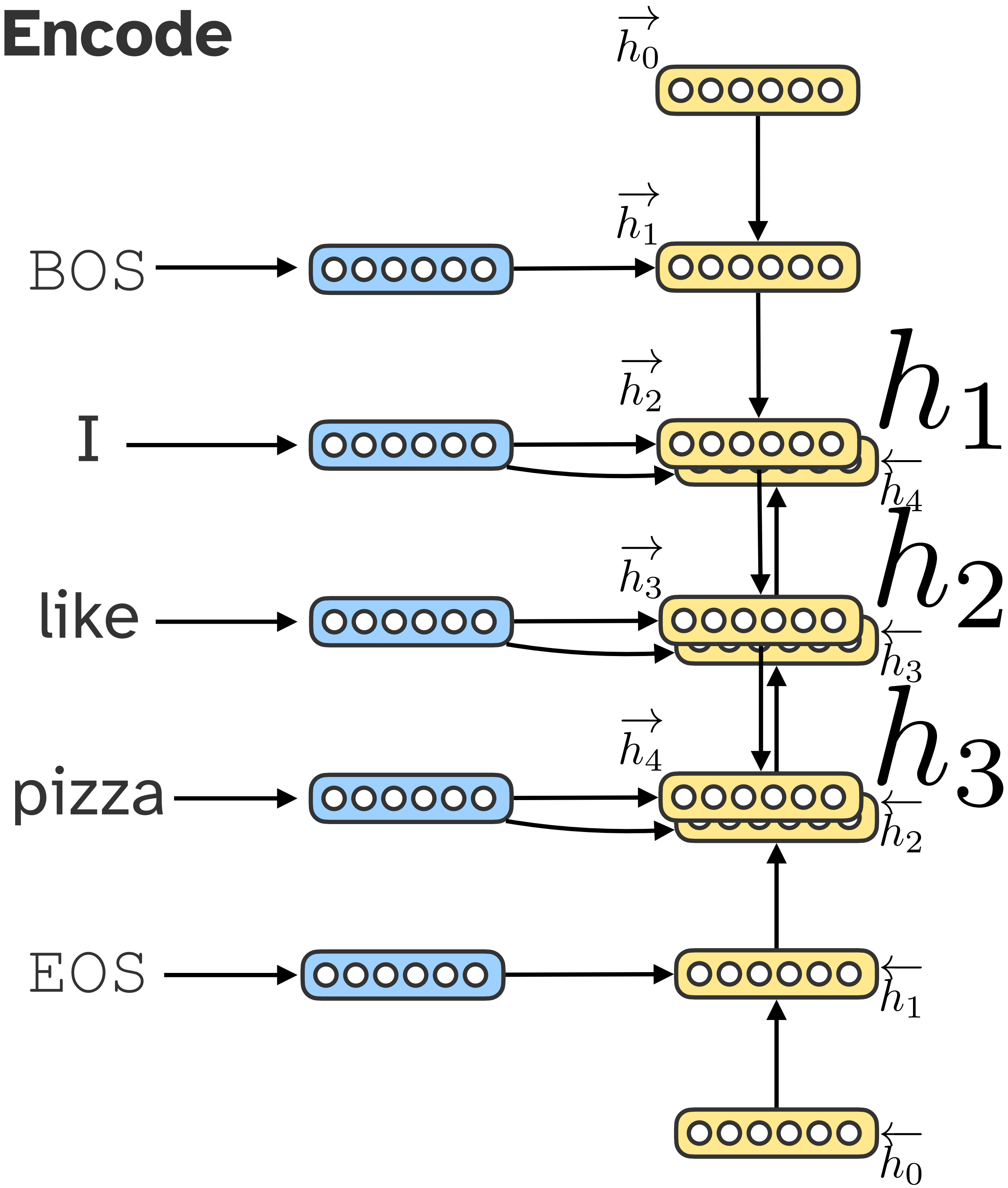
Attention



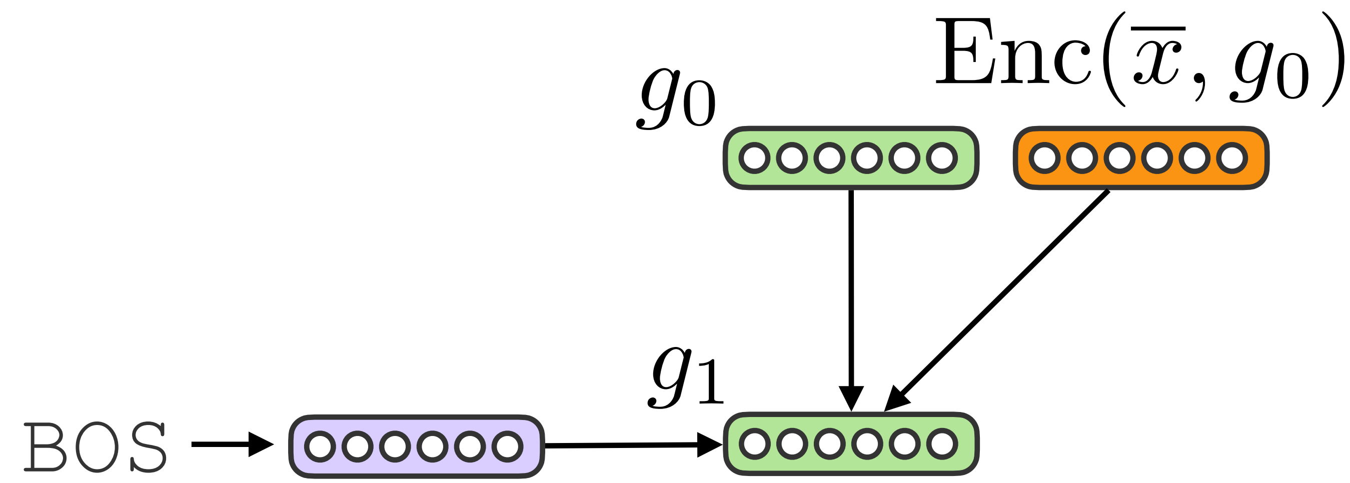
Encode
Decode
$\overrightarrow{h}_0$
$\overrightarrow{h}_1$
$\overrightarrow{h}_2$
$\overrightarrow{h}_3$
$\overrightarrow{h}_4$
BOS
I
like
pizza
EOS
h_1
h_2
h_3
$\overleftarrow{h}_4$
$\overleftarrow{h}_3$
$\overleftarrow{h}_2$
$\overleftarrow{h}_1$
$\overleftarrow{h}_0$
g_0
$\mathrm{Enc}(\overline{x})$
g_1
BOS

Attention

Encode



Decode



$$g_i = f(g_{i-1}, \mathrm{Enc}(\overline{x}, g_{i-1}), \phi(y_i))$$

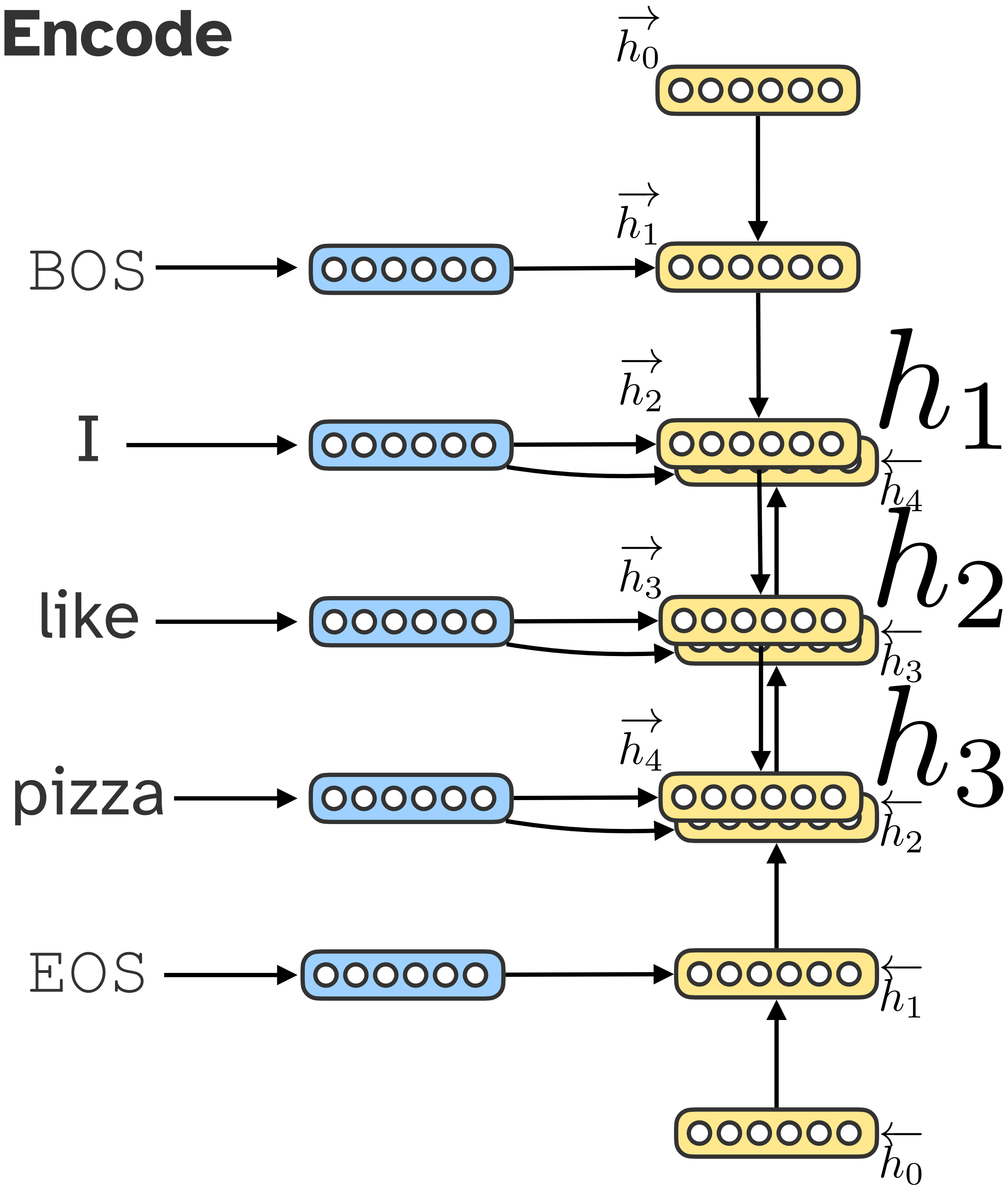
text encoding also depends on the previous state!

Recall:

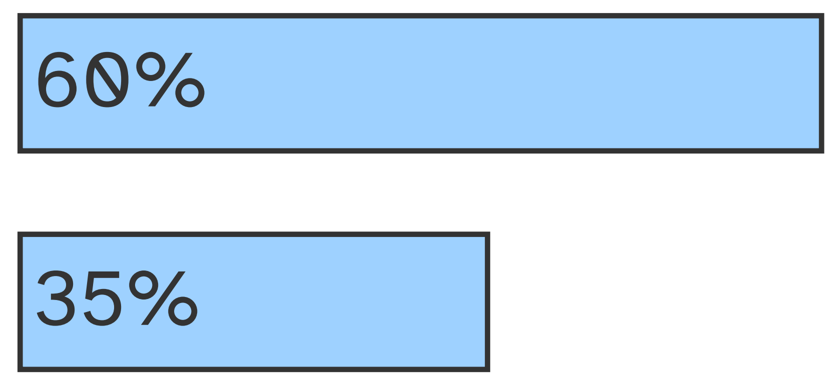
$$\mathrm{Enc}(\overline{x}, g) = \sum_{i=1}^{|\overline{x}|} p(i \mid g) h_i$$

Attention

Encode **Decode**



$$p(I \mid g_0)$$



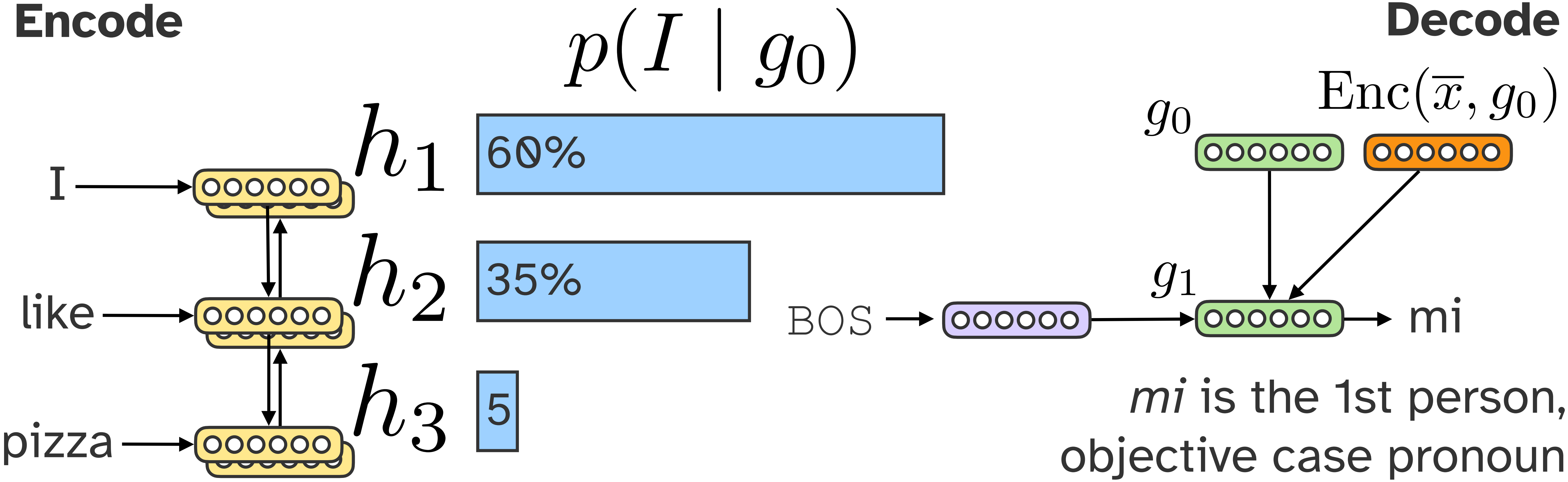
Weighted sum:

$$\text{Enc}(\overline{x}, g_0) = \sum_{i=1}^{|\overline{x}|} p(i \mid g_0) h_i$$

Recall:

$$\text{Enc}(\overline{x}, g) = \sum_{i=1}^{|\overline{x}|} p(i \mid g) h_i$$

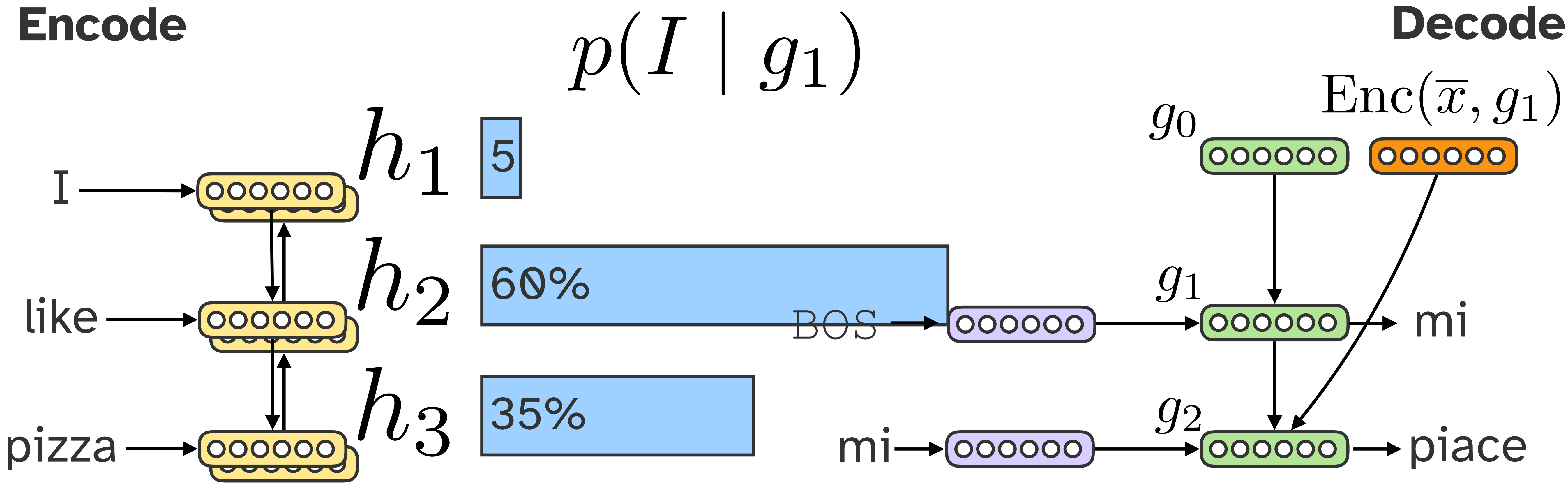
Attention



Recall:

$$\mathrm{Enc}(\overline{x}, g) = \sum_{i=1}^{|\overline{x}|} p(i \mid g) h_i$$

Attention

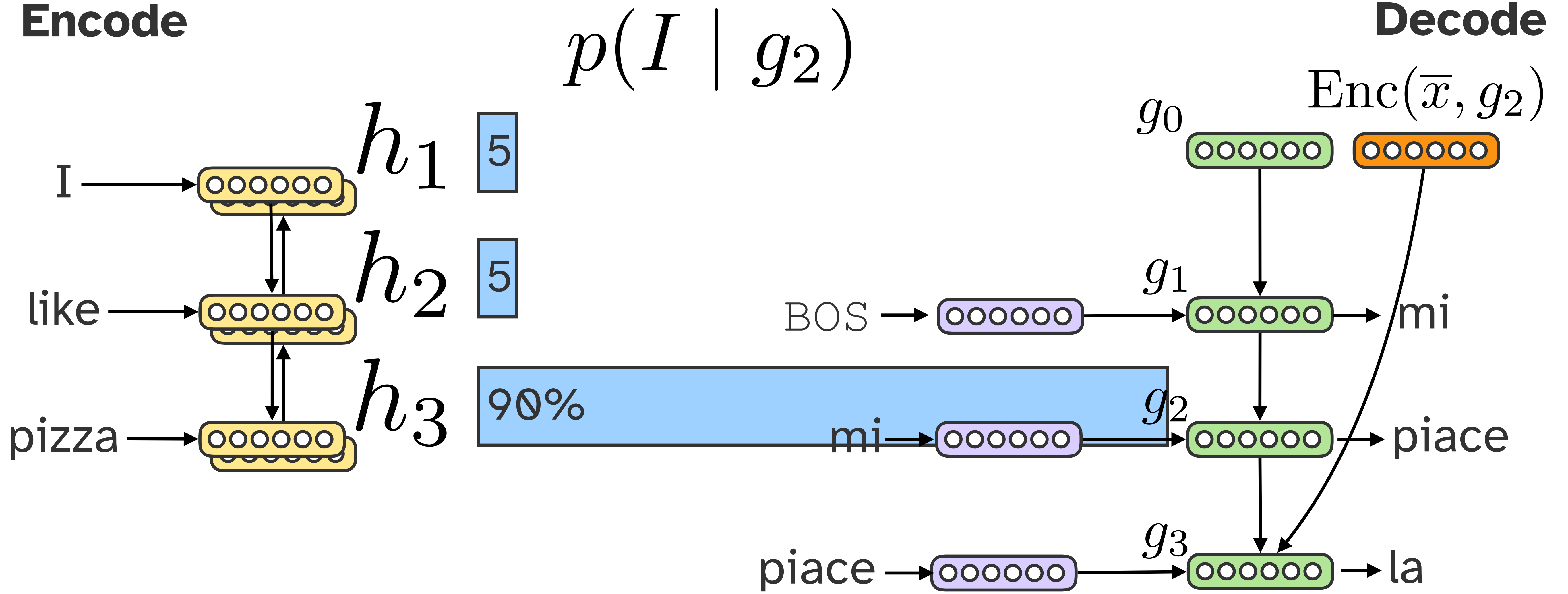


piace is the 3rd person singular declension of *piacere*

Recall:

$$\mathrm{Enc}(\overline{x}, g) = \sum_{i=1}^{|\overline{x}|} p(i \mid g) h_i$$

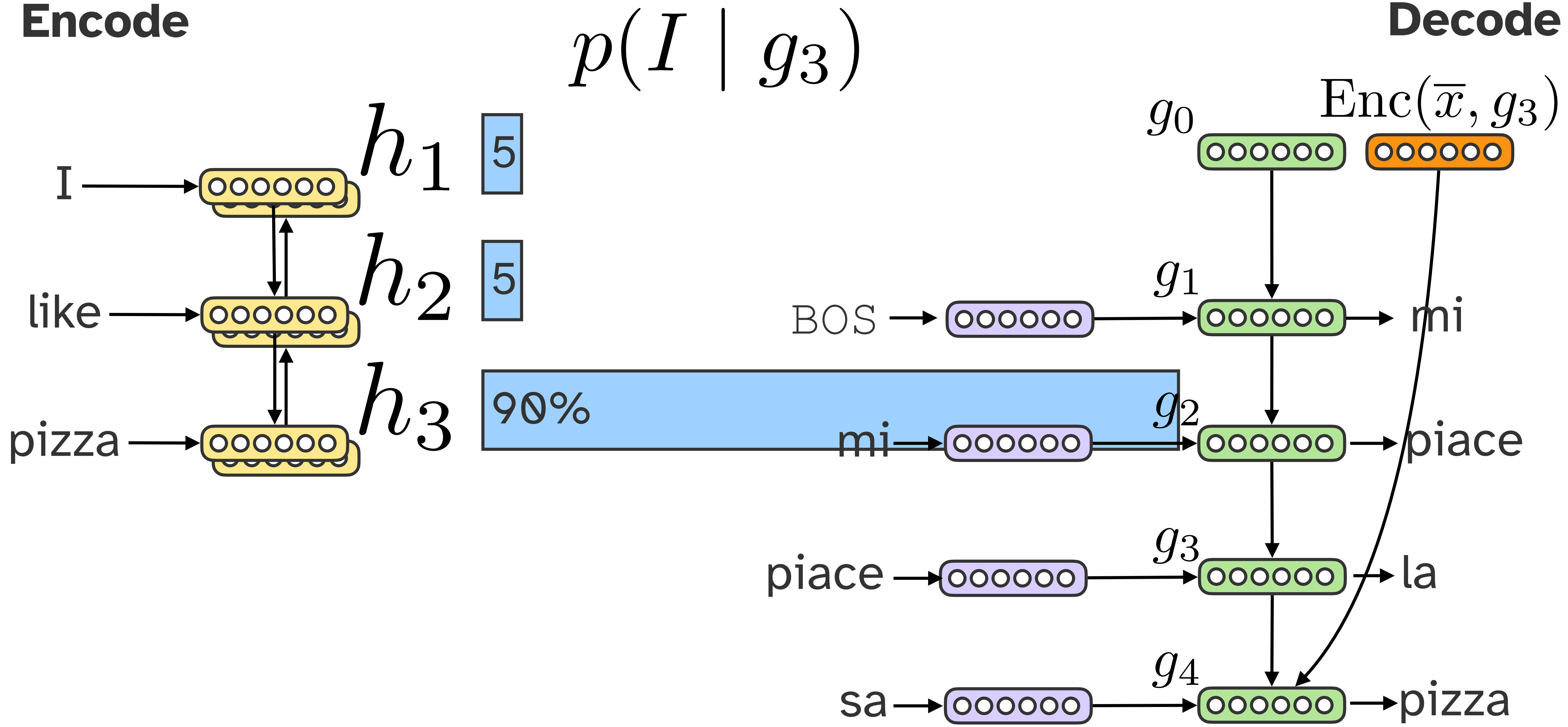
Attention



Recall:

$$\mathrm{Enc}(\overline{x}, g) = \sum_{i=1}^{|\overline{x}|} p(i \mid g) h_i$$

Attention



Recall:

$$\text{Enc}(\overline{x}, g) = \sum_{i=1}^{|\overline{x}|} p(i \mid g) h_i$$

Attention

- Update decoding hidden state based on an updated version of the input sequence encoding

$$g_i = f(g_{i-1}, \text{Enc}(\overline{x}, g_{i-1}), \phi(y_i))$$

- The updated sequence is determined via a weighted sum over input hidden states

$$\text{Enc}(\overline{x}, g) = \sum_{i=1}^{|\overline{x}|} p(i \mid g) h_i \qquad p : \mathbb{R}^d \to \Delta^{\mathbb{N}_{1:|\overline{x}|}}$$

- Next time: how do we compute $p(I \mid g)$?