COS 484/584

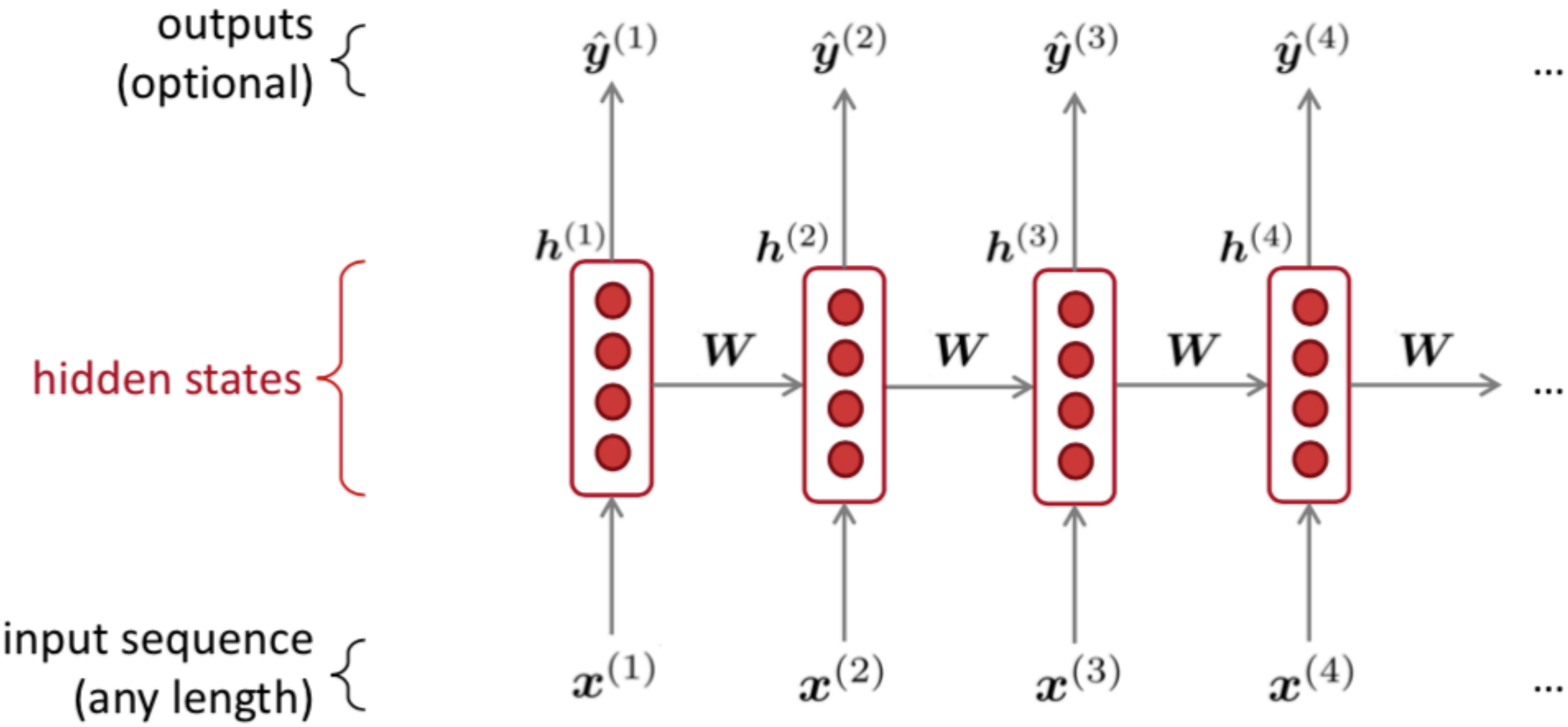
L16: Neural Machine Translation - 1

Neural Machine Translation

- A **single neural network** is used to translate from source to target language
- Architecture: Encoder-Decoder
  - Two main components:
    - Encoder: Convert source sentence (input) into a vector/matrix
    - Decoder: Convert encoding into a sentence in target language (output)

Recall: RNNs

$$\mathbf{h}_t = g(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}) \in \mathbb{R}^d$$



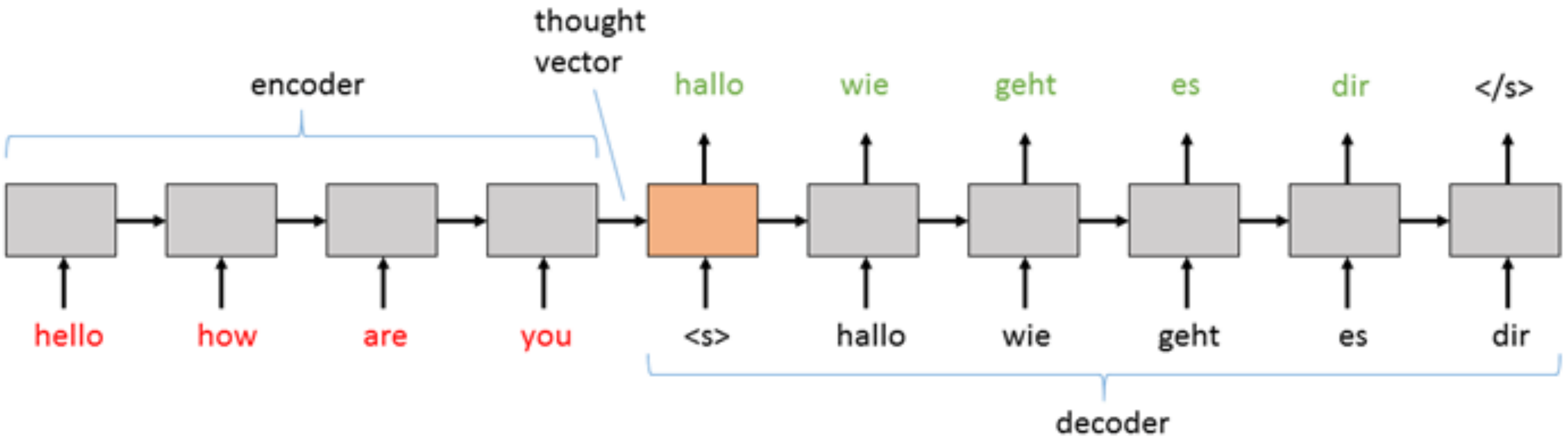
Recall: RNNs

$$\mathbf{h}_t = g(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}) \in \mathbb{R}^d$$

What is the maximum sequence length an RNN could theoretically take as input?

A) 10
B) 128
C) ∞

Sequence to Sequence learning (Seq2seq)

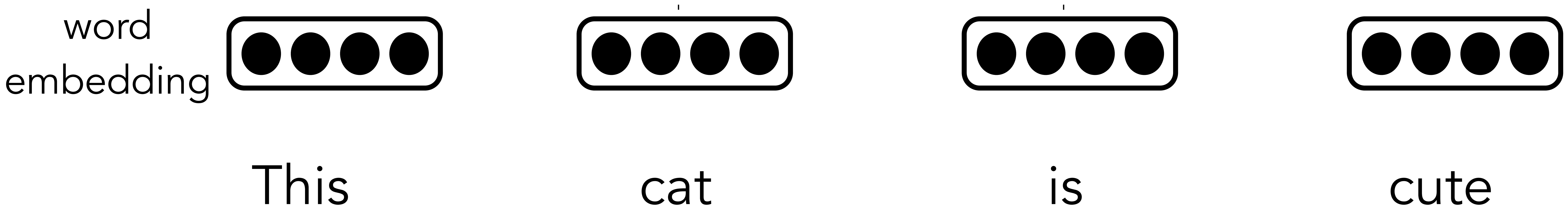


- Encode entire input sequence into a single vector **(using an RNN)**
- Decode one word at a time **(again, using an RNN!)**
- Beam search for better inference
- Learning is not trivial! (vanishing/exploding gradients)

(Sutskever et al., 2014)

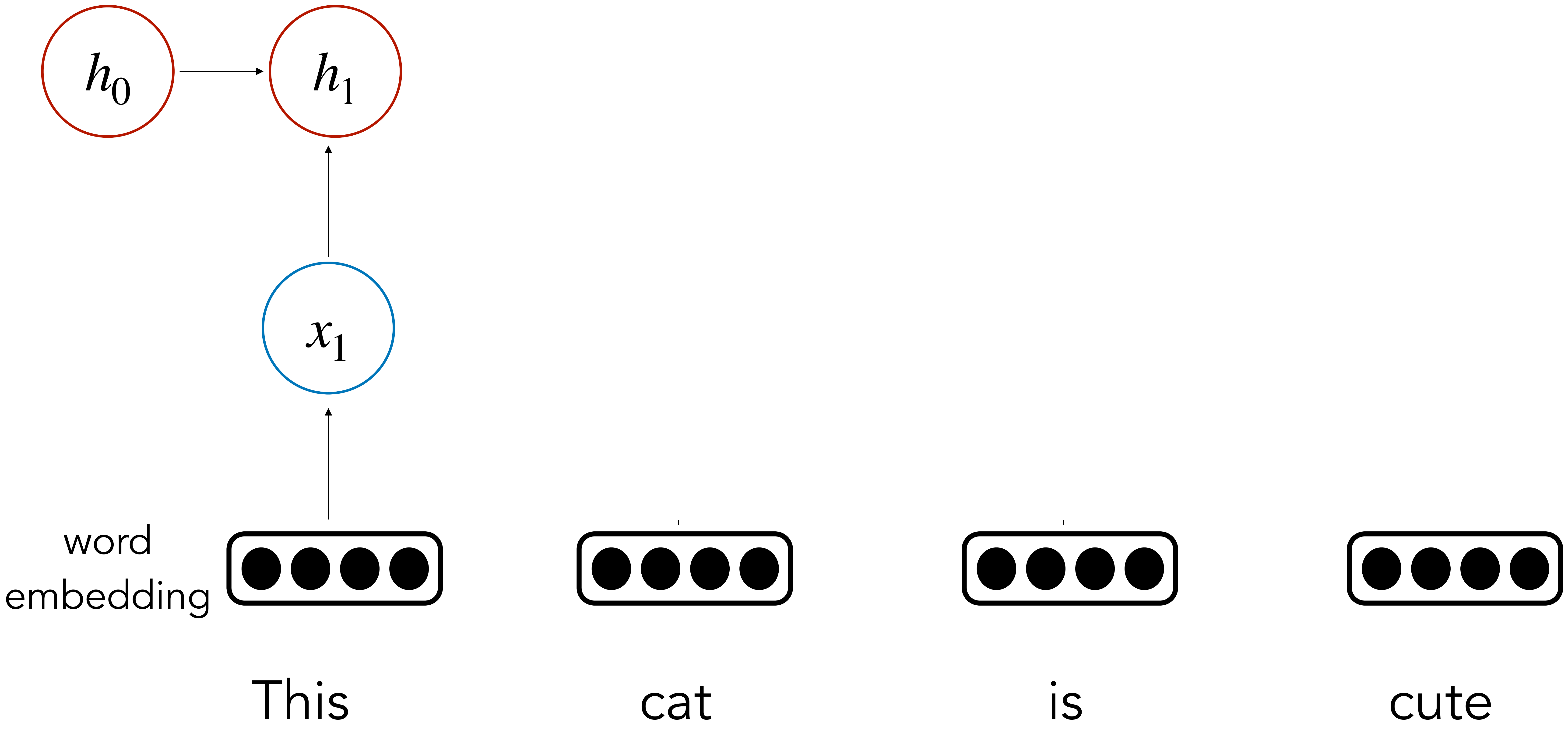
Encoder

Sentence: This cat is cute



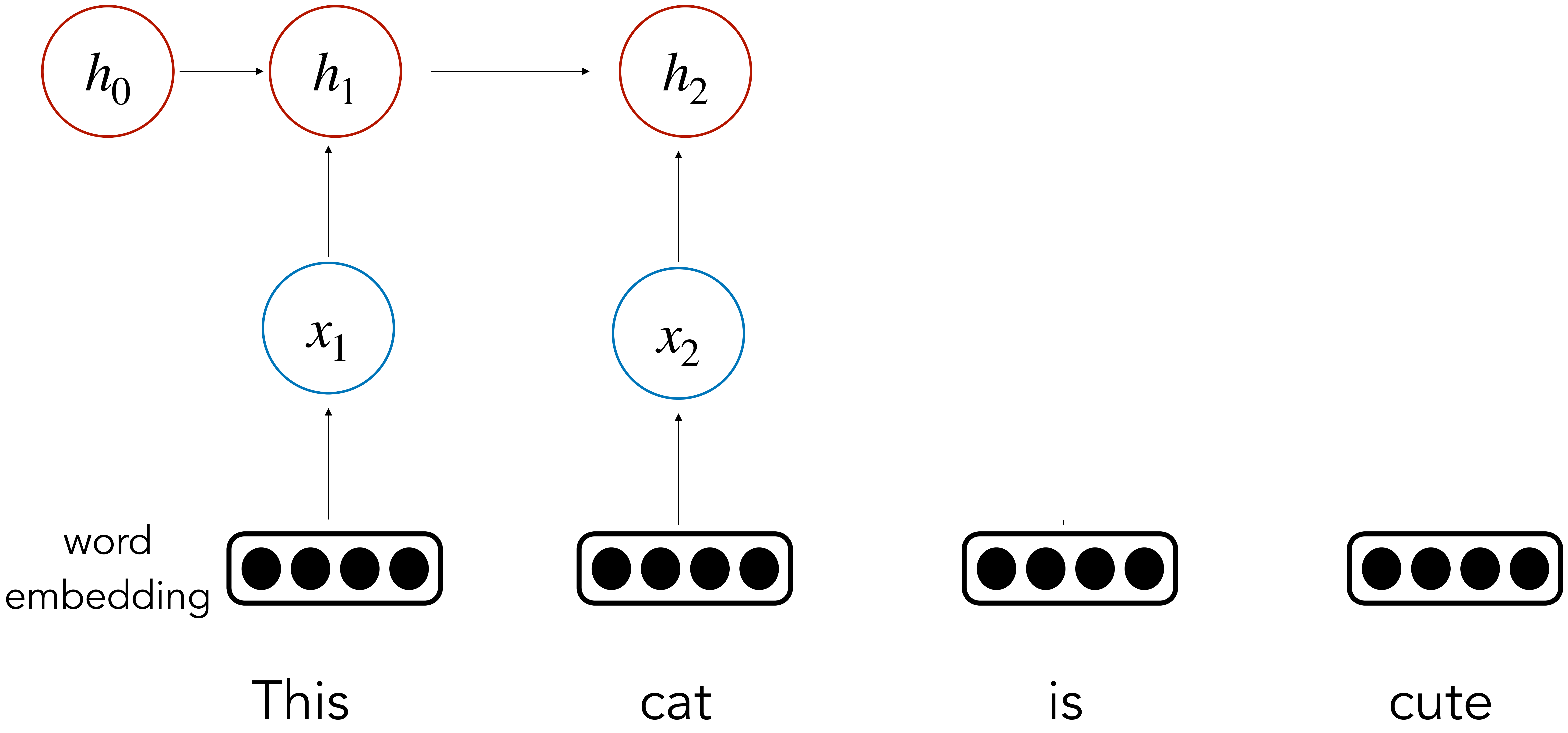
Encoder

Sentence: This cat is cute



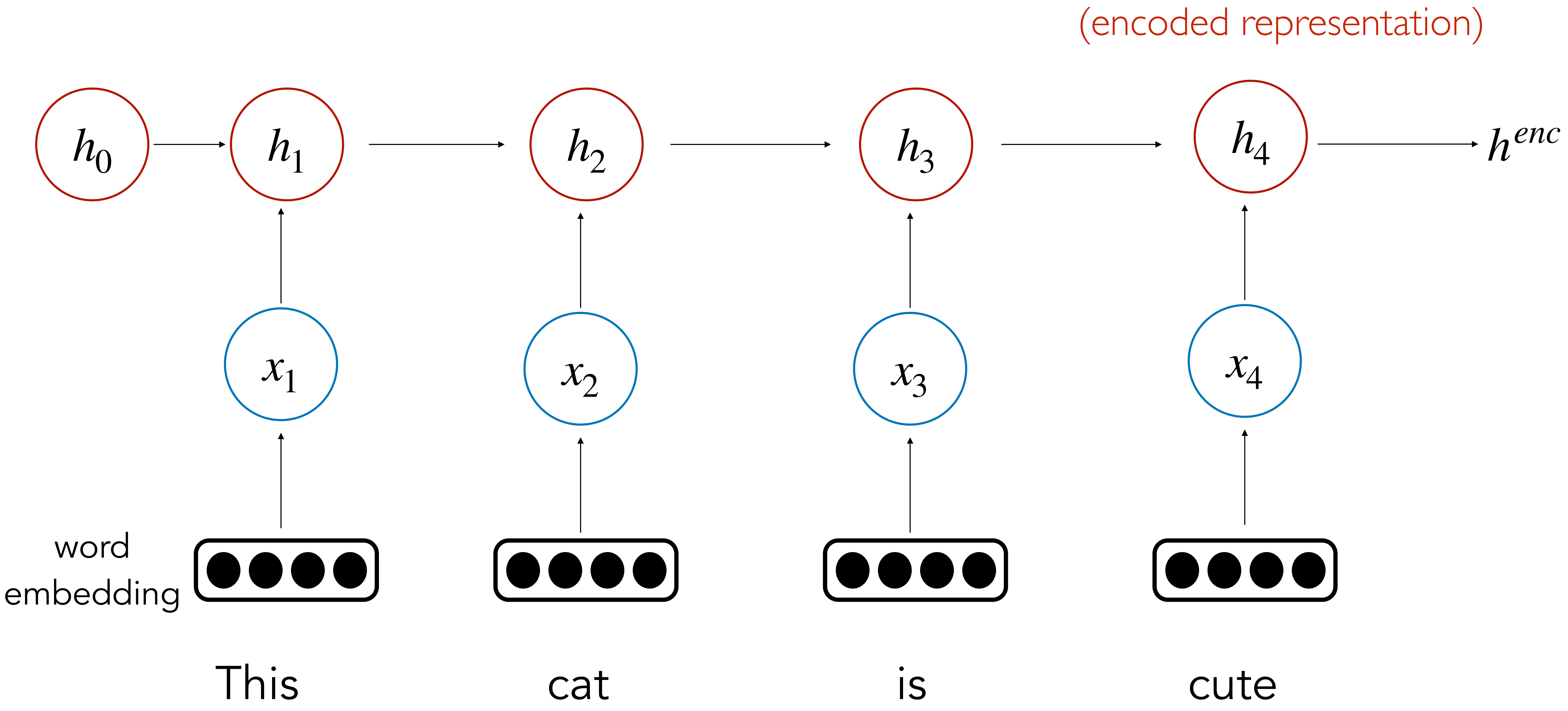
Encoder

Sentence: This cat is cute

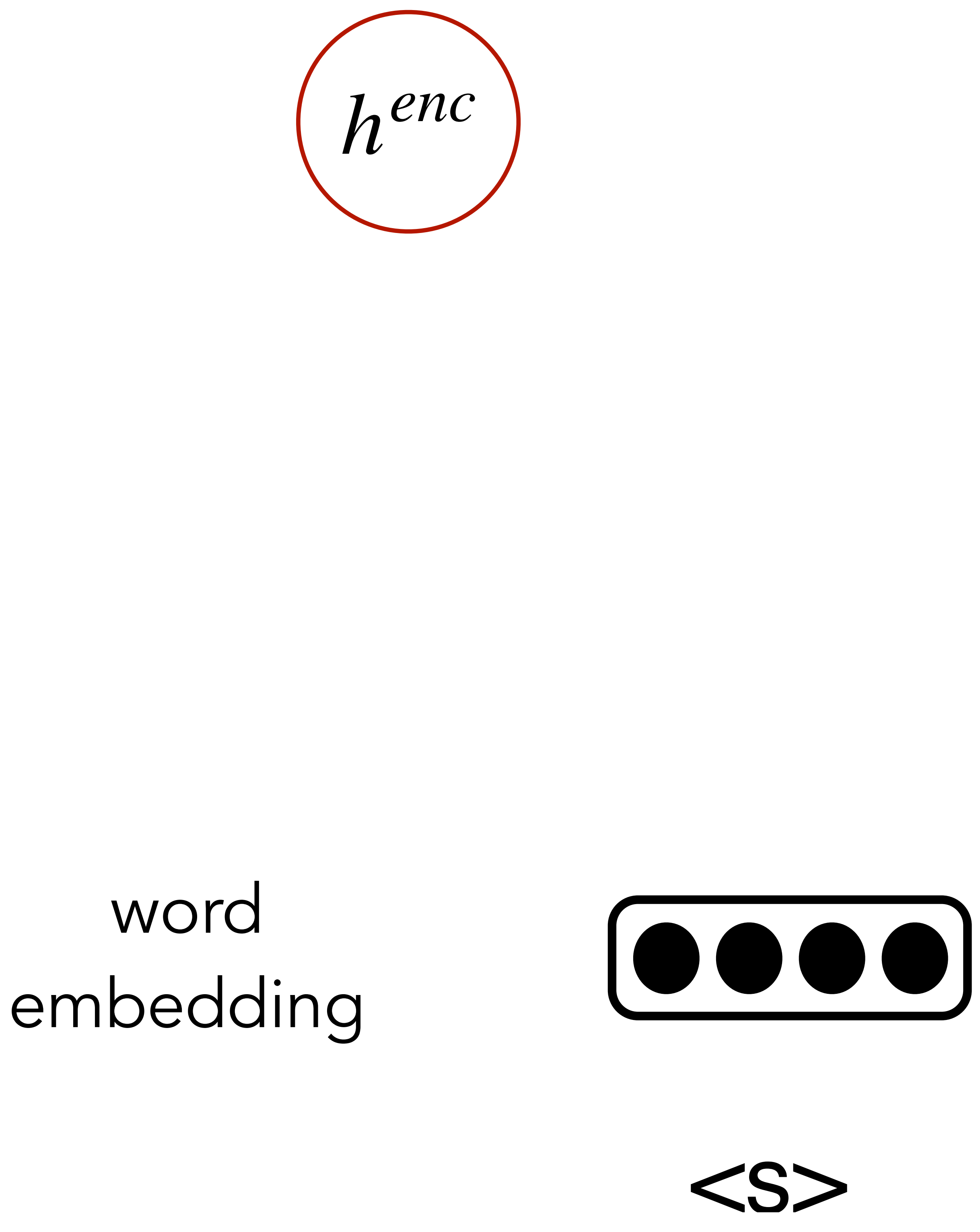


Encoder

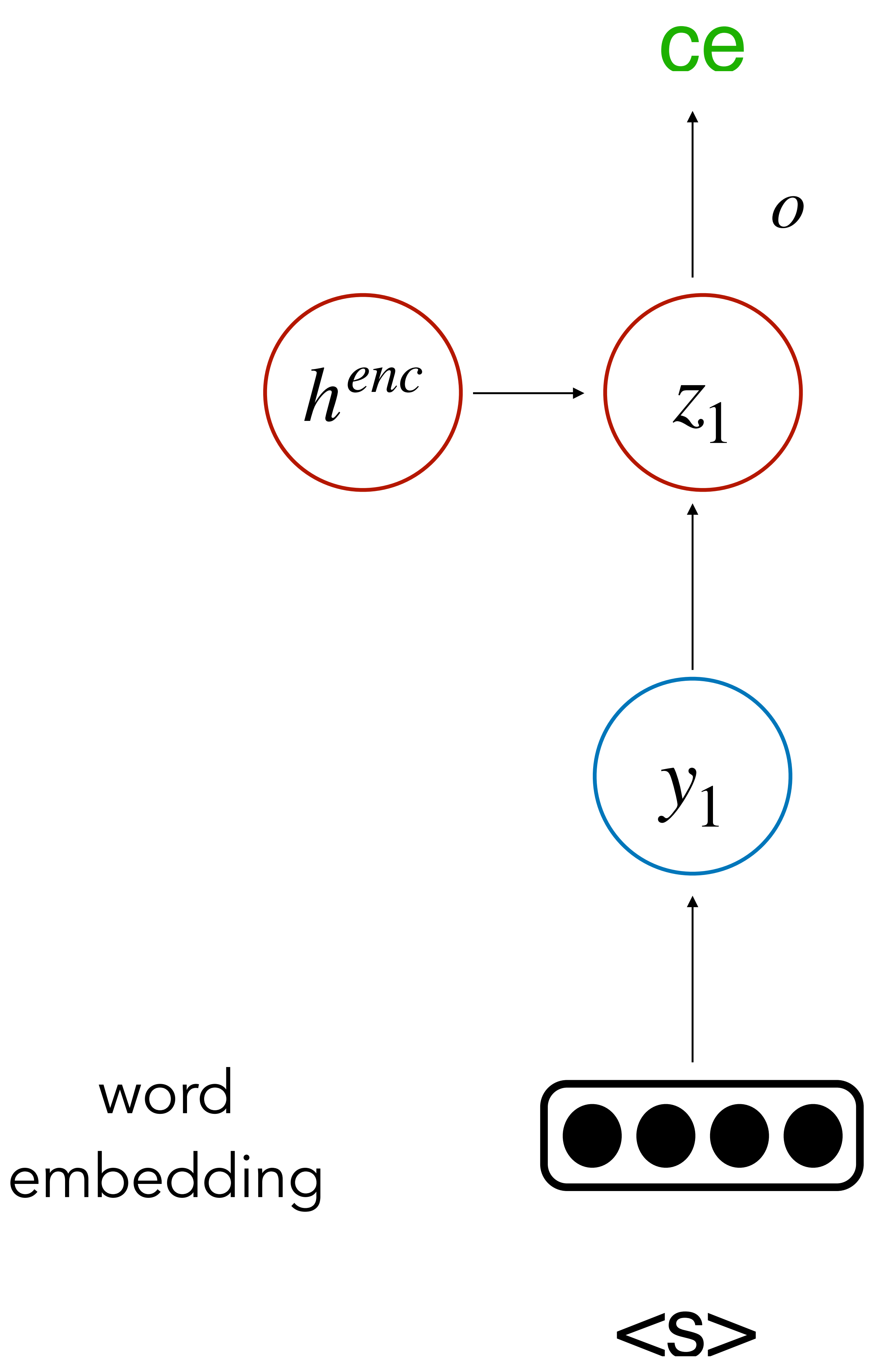
Sentence: This cat is cute



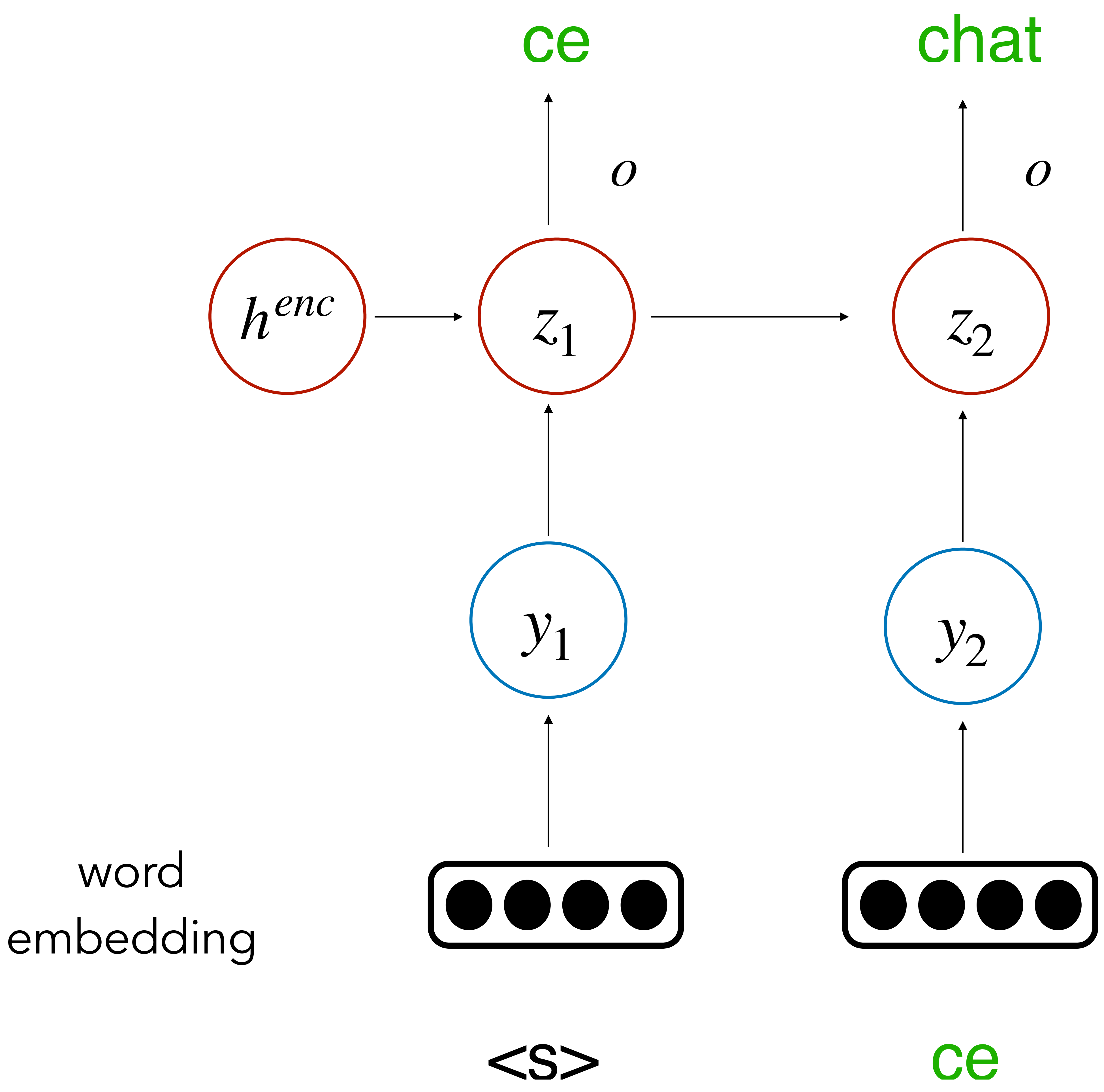
Decoder



Decoder

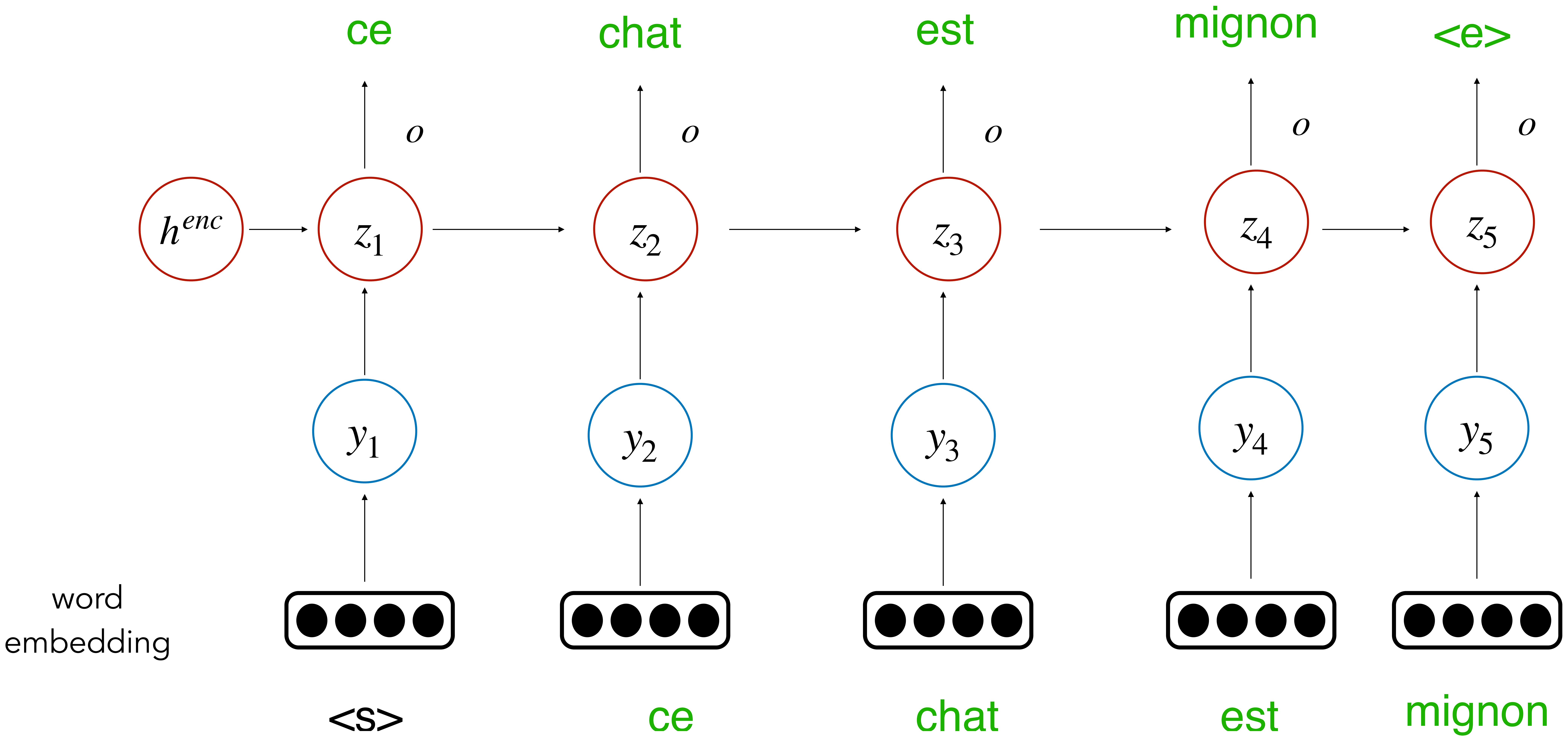


Decoder



Decoder

- A conditioned language model



Seq2seq training

- Similar to training a language model!
- Minimize cross-entropy loss:

$$\sum_{t=1}^{T} -\log P(y_t \mid y_1, \ldots, y_{t-1}, x_1, \ldots, x_n)$$

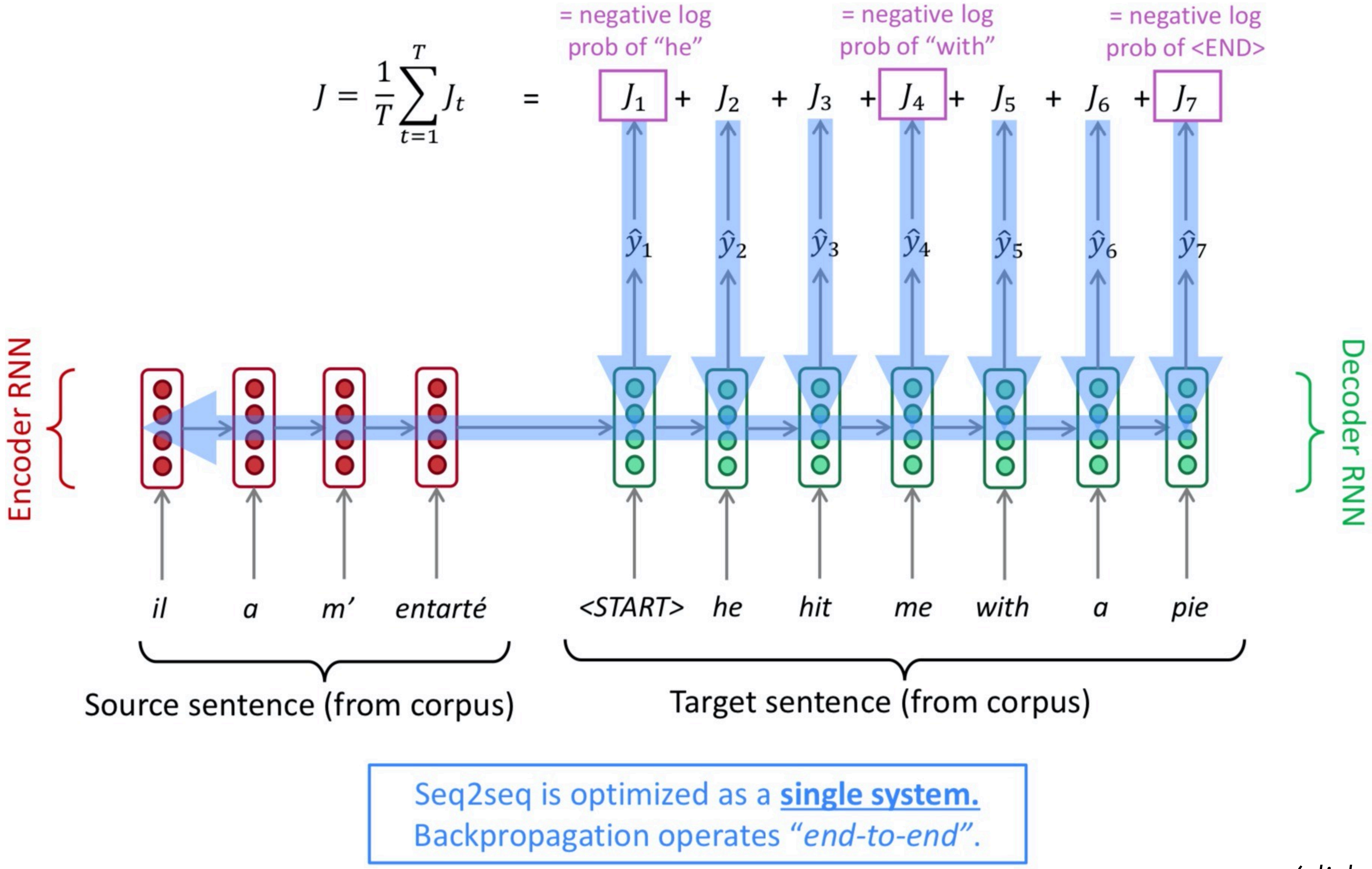
- Back-propagate gradients through *both decoder and encoder*
- Need a really big corpus

36M sentence pairs

Russian: Машинный перевод - это круто!

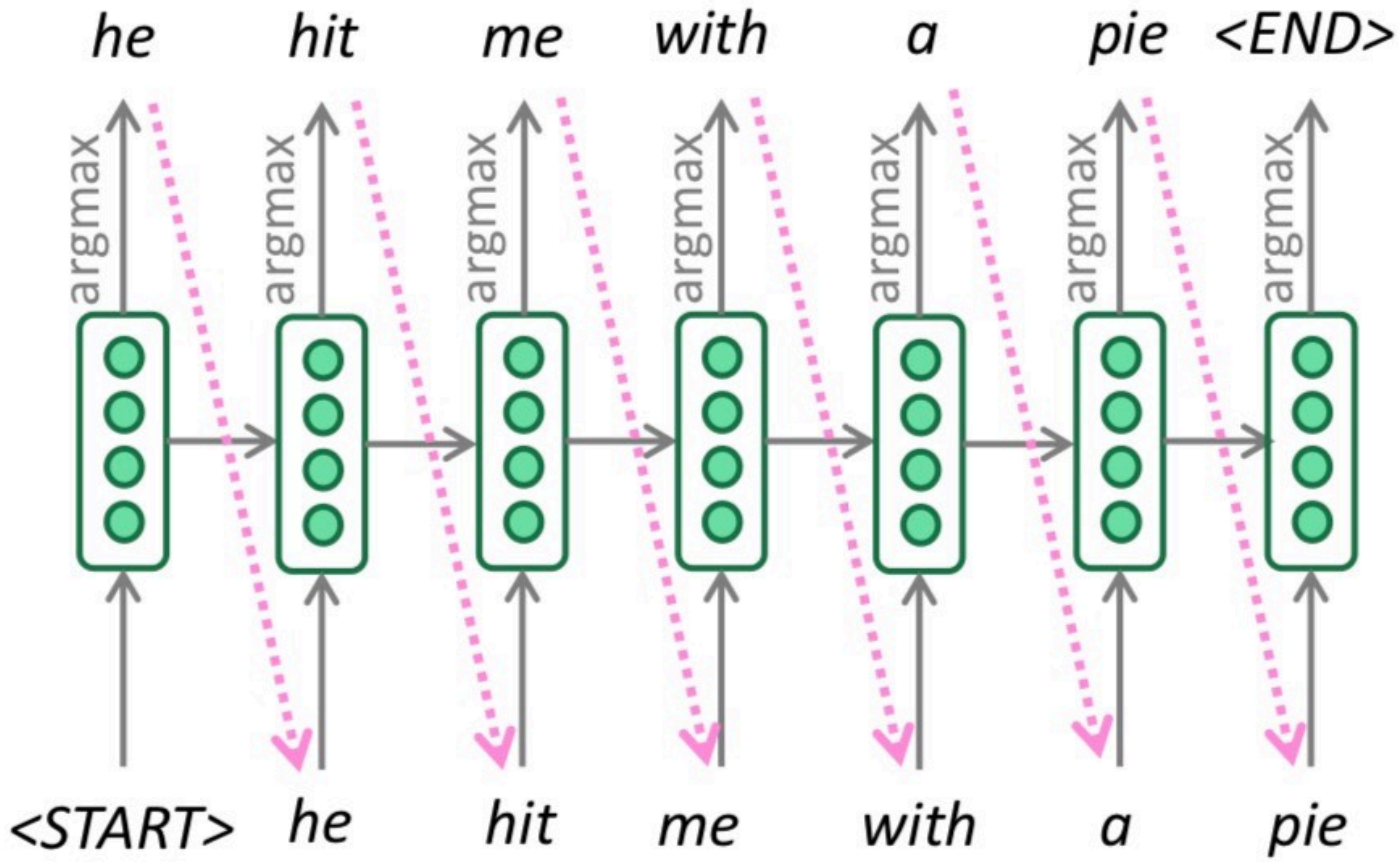
English: Machine translation is cool!

Seq2seq training



(slide credit: Abigail See)

Greedy decoding



- Compute argmax at every step of decoder to generate word
- What's wrong?

Exhaustive search?

- Find $\arg\max_{y_1,\ldots,y_T} P(y_1, \ldots, y_T | x_1, \ldots, x_n)$

- Requires computing all possible sequences

V - Vocabulary
T - length of sequence

What is the complexity of doing this search?

A) O(VT)

B) O(V^T)

C) O(T^V)

A middle ground: Beam search

- **Key idea:** At every step, keep track of the k most probable partial translations (hypotheses)

- Score of each hypothesis = log probability of sequence so far

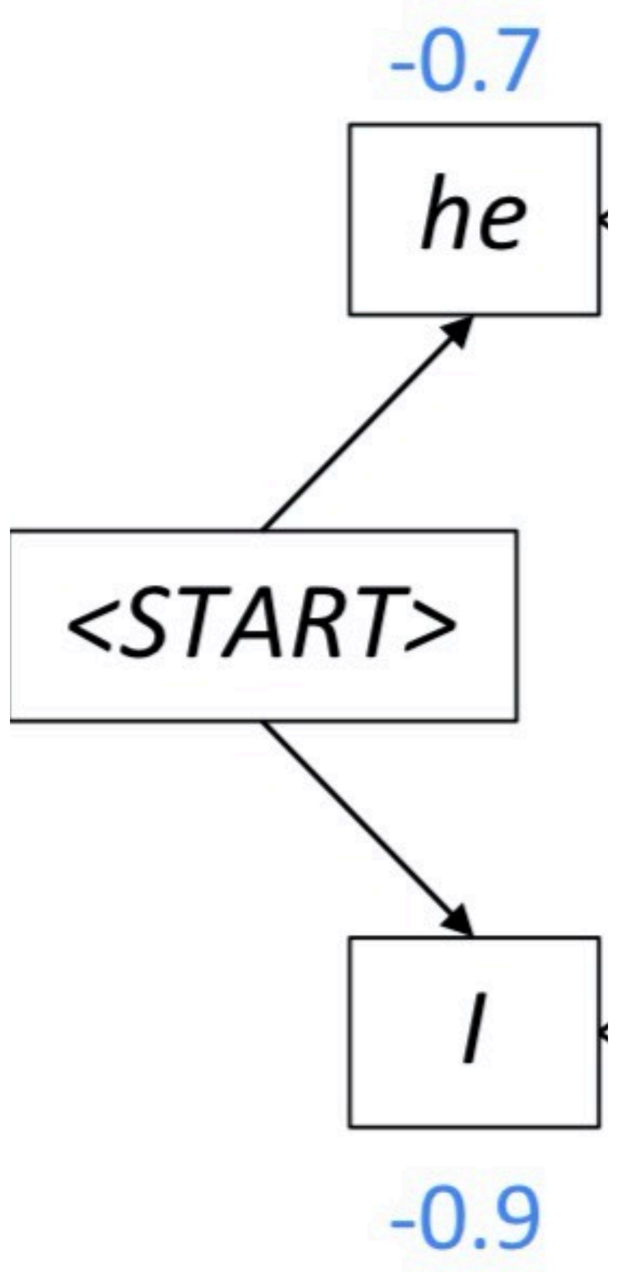
$$\sum_{t=1}^{j} \log P(y_t \mid y_1, \ldots, y_{t-1}, x_1, \ldots, x_n)$$

- Not guaranteed to be optimal

- More efficient than exhaustive search

Beam decoding

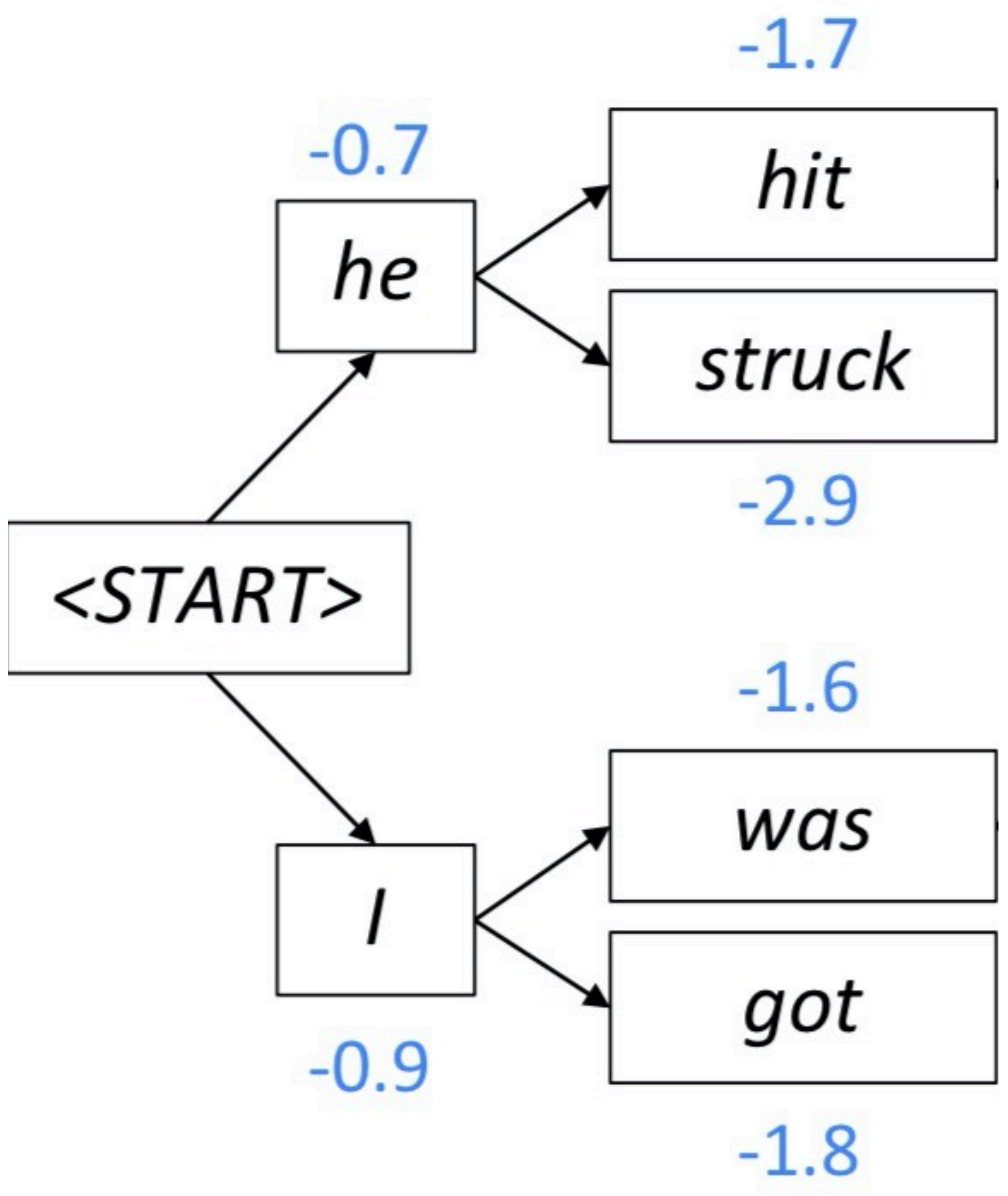
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

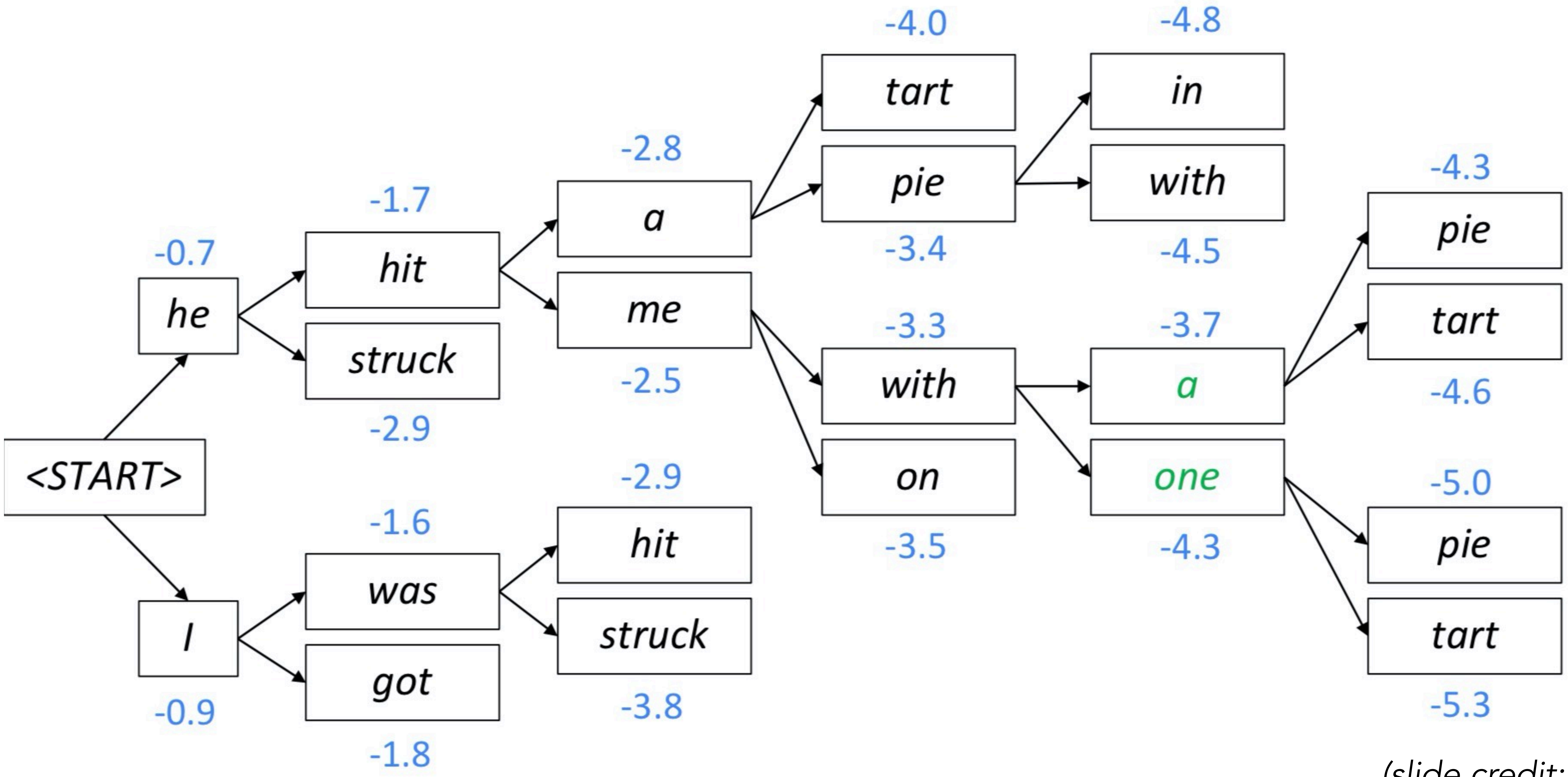
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

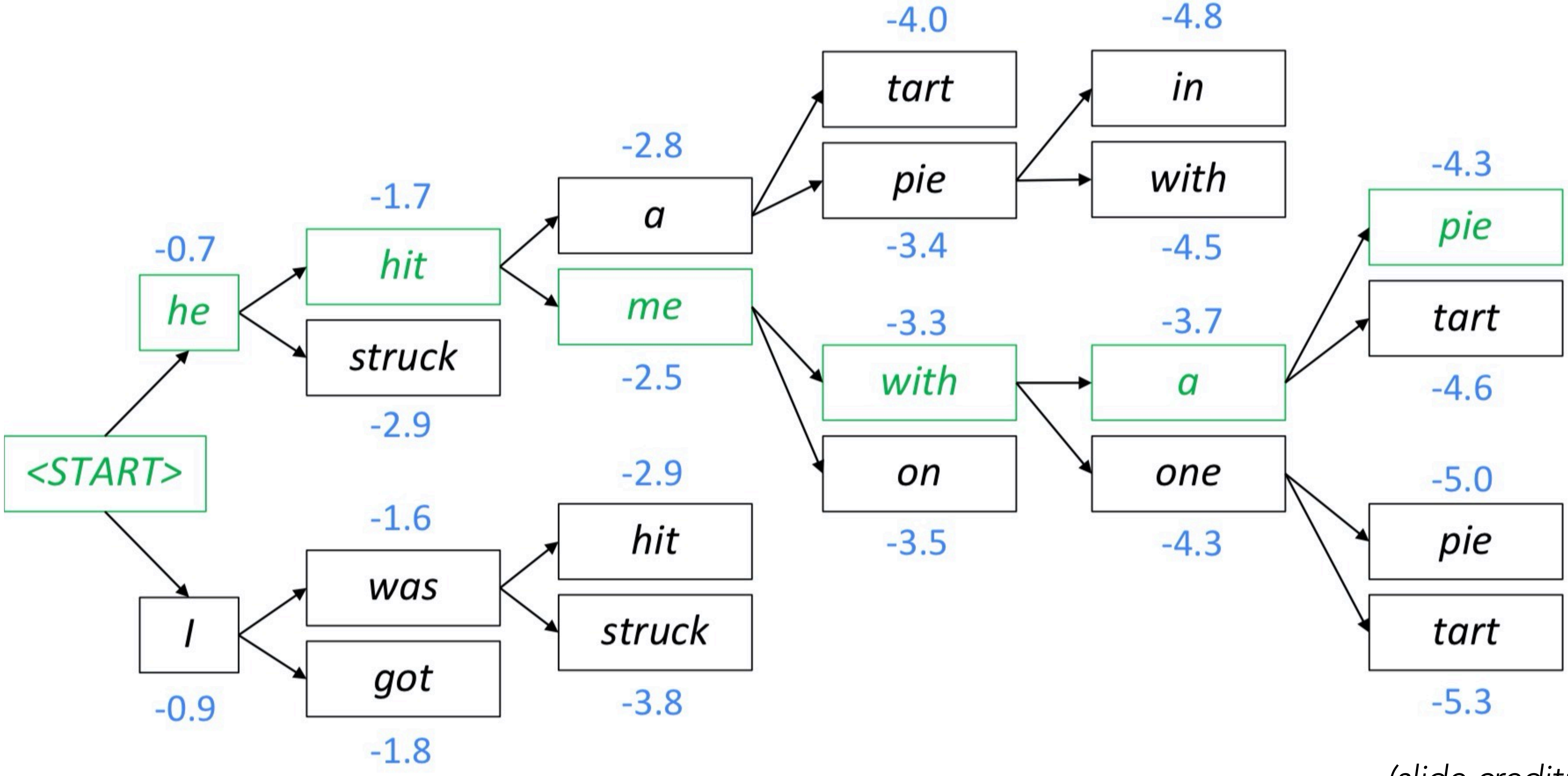
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Backtrack

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

- Different hypotheses may produce $\langle e \rangle$ (end) token at different time steps
  - When a hypothesis produces $\langle e \rangle$, stop expanding it and place it aside
- Continue beam search until:
  - All k hypotheses produce $\langle e \rangle$ OR
  - Hit max decoding limit T
- Select top hypotheses using the *normalized* likelihood score

$$\frac{1}{T}\sum_{t=1}^{T}\log P(y_t \,|\, y_1, \ldots, y_{t-1}, x_1, \ldots, x_n)$$

  - Otherwise shorter hypotheses have higher scores