COS 484/584

L16: Neural Machine Translation - 1

Spring 2021

Last time: IBM Model 1

- Assume $p(a_m \mid m, M^{(s)}, M^{(t)}) = \dfrac{1}{M^{(t)}}$

- We then have:

$$p(w^{(s)}, w^{(t)}) = p(w^{(t)}) \sum_{A} (\frac{1}{M^{(t)}})^{M^{(s)}} \; p(w^{(s)} \mid w^{(t)})$$

- How do we estimate $p(w^{(s)} = v \mid w^{(t)} = u)$?

IBM Model 1

- If we have word-to-word alignments, we can compute the probabilities using the MLE:
  - $p(v \mid u) = \frac{count(u, v)}{count(u)}$
  - where $count(u, v)$ = #instances where target word u was aligned to source word v in the training set
- However, word-to-word alignments are often hard to come by

What can we do?

EM for Model I

- **(E-Step)** If we had an accurate translation model, we can estimate likelihood of each alignment as:

$$q_m(a_m \mid \boldsymbol{w}^{(s)}, \boldsymbol{w}^{(t)}) \propto \mathrm{p}(a_m \mid m, M^{(s)}, M^{(t)}) \times \mathrm{p}(w_m^{(s)} \mid w_{a_m}^{(t)}),$$

Remember these are fixed

- **(M Step)** Use expected count to re-estimate translation parameters:

How would you compute the new probabilities $p(v \mid u)$?

A) $\boxed{p(v \mid u) = \frac{E_q[count(u, v)]}{count(u)}}$

B) $p(v \mid u) = \frac{E_q[count(u, v)]}{count(v)}$

C) $p(v \mid u) = E_q[count(u, v)]$

EM for Model 1

- **(E-Step)** If we had an accurate translation model, we can estimate likelihood of each alignment as:

$$q_m(a_m \mid \boldsymbol{w}^{(s)}, \boldsymbol{w}^{(t)}) \propto \mathrm{p}(a_m \mid m, M^{(s)}, M^{(t)}) \times \mathrm{p}(w_m^{(s)} \mid w_{a_m}^{(t)}),$$

Remember these are fixed

- **(M Step)** Use expected count to re-estimate translation parameters:

$$E_q\left[\text{count}(u, v)\right] = \sum_m q_m(a_m \mid \boldsymbol{w}^{(s)}, \boldsymbol{w}^{(t)}) \times \delta(w_m^{(s)} = v) \times \delta(w_{a_m}^{(t)} = u).$$

$$p(v \mid u) = \frac{E_q[count(u, v)]}{count(u)}$$

Decoding: How do we translate?

- We want: $\arg\max_{w^{(t)}} p(w^{(t)} \mid w^{(s)}) = \arg\max_{w^{(t)}} \frac{p(w^{(s)}, w^{(t)})}{p(w^{(s)})}$

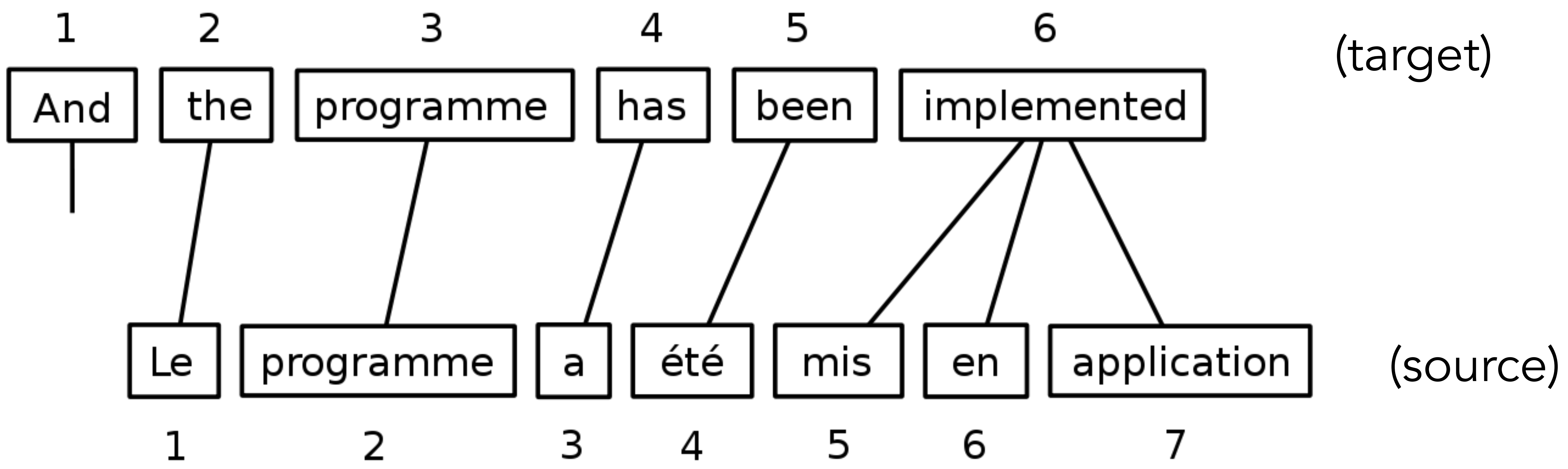
- Sum over all possible alignments:

$$\begin{aligned} \mathrm{p}(\boldsymbol{w}^{(s)}, \boldsymbol{w}^{(t)}) &= \sum_{\mathcal{A}} \mathrm{p}(\boldsymbol{w}^{(s)}, \boldsymbol{w}^{(t)}, \mathcal{A}) \\ &= \mathrm{p}(\boldsymbol{w}^{(t)}) \sum_{\mathcal{A}} \mathrm{p}(\mathcal{A}) \times \mathrm{p}(\boldsymbol{w}^{(s)} \mid \boldsymbol{w}^{(t)}, \mathcal{A}) \end{aligned}$$

- Alternatively, take the max over alignments

- Decoding: Greedy/beam search

Model 1: Decoding



At every step m, pick target word $w_m^{(t)}$ to maximize product of:

1. Language model: $p_{LM}(w_m^{(t)} \mid w_1^{(t)}, \ldots, w_{m-1}^{(t)})$

2. Translation model: $p(w_{b_m}^{(s)} \mid w_m^{(t)})$

where b_m is the inverse alignment from target to source

IBM Model 1

- Assume $p(a_m \mid m, M^{(s)}, M^{(t)}) = \dfrac{1}{M^{(t)}}$
- Each source word is aligned to at most one target word
- We then have:

$$p(w^{(s)}, w^{(t)}) = p(w^{(t)}) \sum_{A} (\frac{1}{M^{(t)}})^{M^{(s)}} \; p(w^{(s)} \mid w^{(t)})$$

Restrictive assumptions

IBM Model 2

- Slightly relaxed assumption:
  - $p(a_m \mid m, M^{(s)}, M^{(t)})$ is also estimated/learned

- Some independence assumptions from Model 1 still required:
  - Alignment probability factors across tokens:

$$\mathrm{p}(\mathcal{A} \mid \boldsymbol{w}^{(s)}, \boldsymbol{w}^{(t)}) = \prod_{m=1}^{M^{(s)}} \mathrm{p}(a_m \mid m, M^{(s)}, M^{(t)}).$$

  - Translation probability factors across tokens:

$$\mathrm{p}(\boldsymbol{w}^{(s)} \mid \boldsymbol{w}^{(t)}, \mathcal{A}) = \prod_{m=1}^{M^{(s)}} \mathrm{p}(w_m^{(s)} \mid w_{a_m}^{(t)}),$$

Other IBM models

Model 1: lexical translation

Model 2: additional absolute alignment model

Model 3: extra fertility model

Model 4: added relative alignment model

Model 5: fixed deficiency problem.

Model 6: Model 4 combined with a HMM alignment model in a log linear way

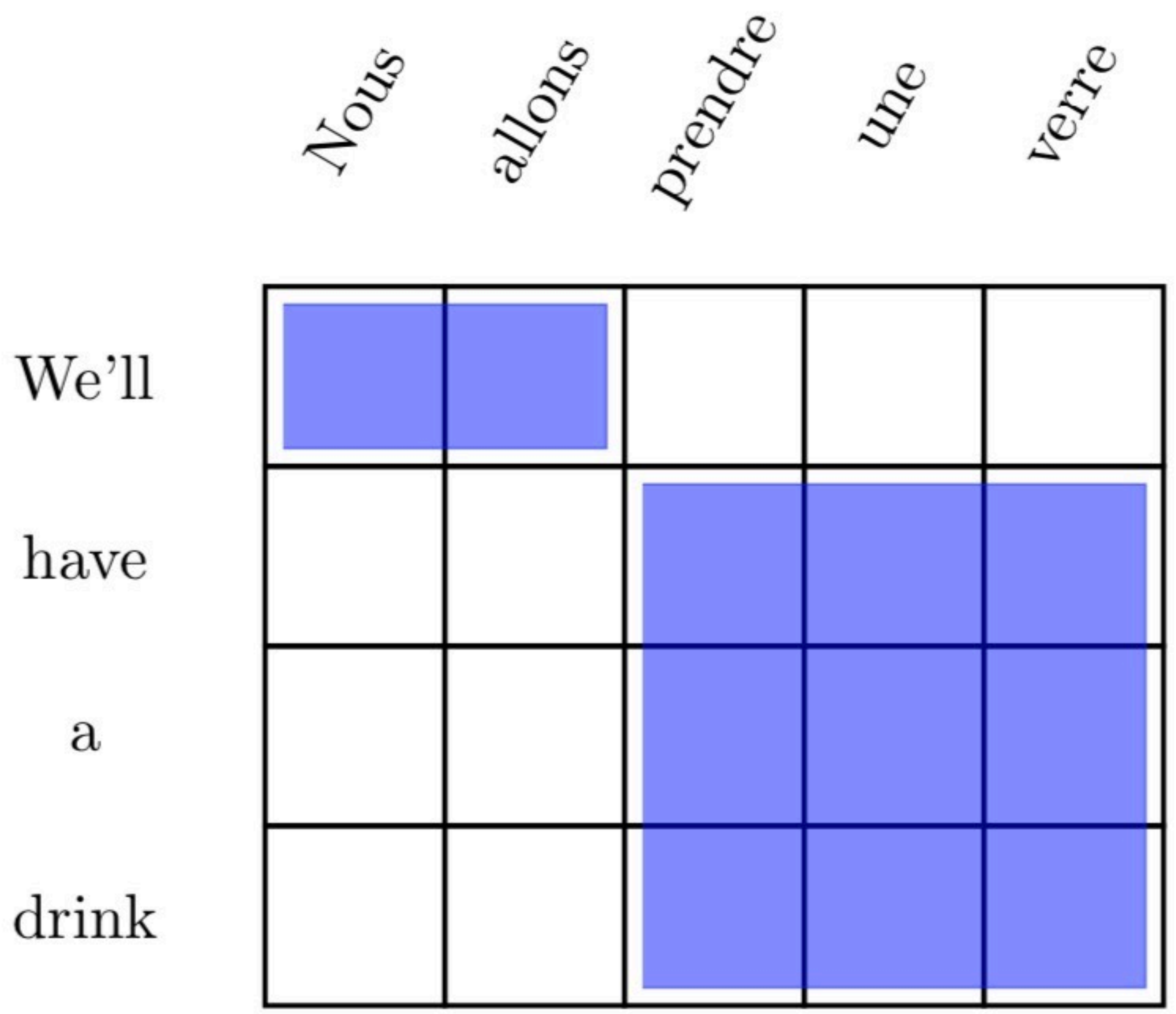
- Models 3 - 6 make successively weaker assumptions
  - But get progressively harder to optimize
- Simpler models are often used to ‘initialize’ complex ones
  - e.g train Model 1 and use it to initialize Model 2 translation parameters

Phrase-based MT

Nous allons prendre un verre

(literal) We will take a glass

(actual) We'll have a drink



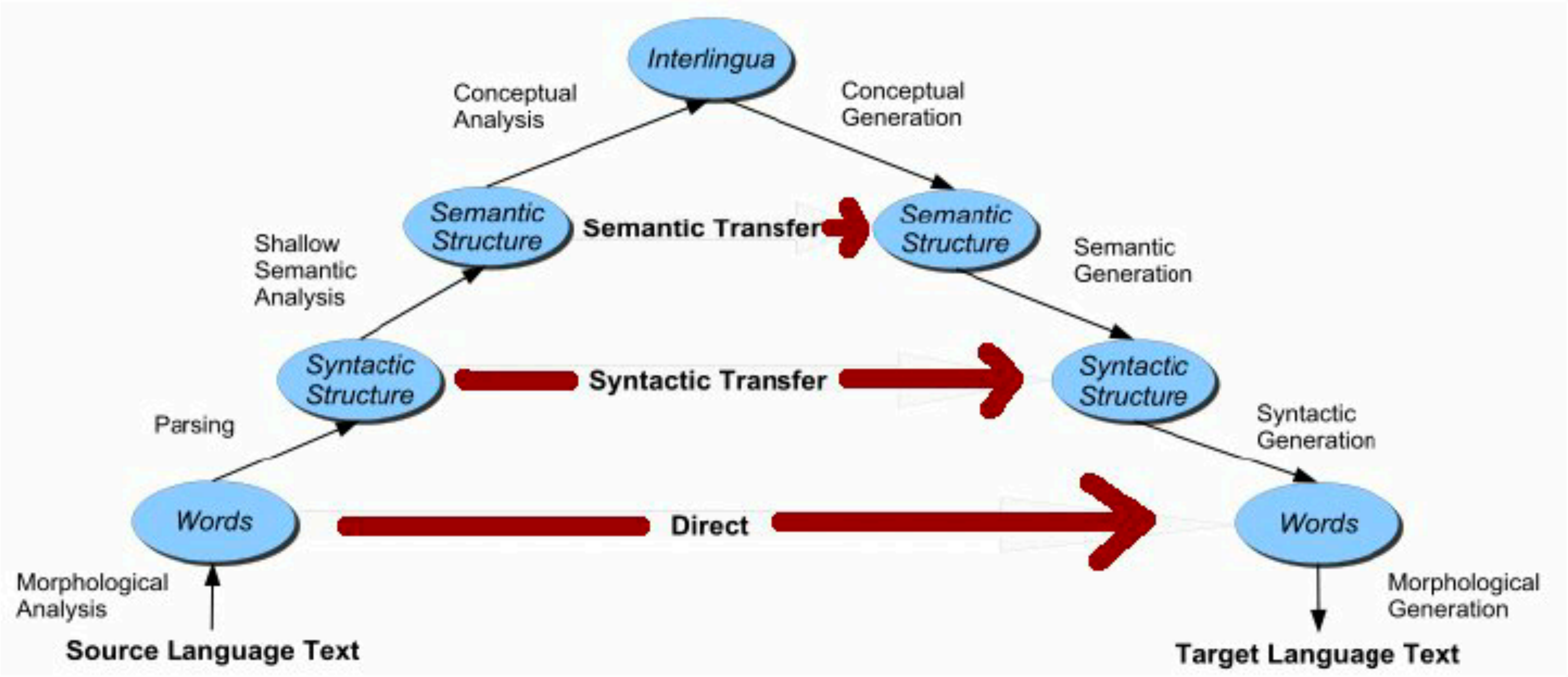
- Word-by-word translation is not sufficient in many cases
- Solution: build alignments and translation tables between multiword spans or "phrases"

Phrase-based MT

- Solution: build alignments and translation tables between multiword spans or "phrases"
- Translations condition on multi-word units and assign probabilities to multi-word units
- Alignments map from spans to spans

$$\mathrm{p}(\boldsymbol{w}^{(s)} \mid \boldsymbol{w}^{(t)}, \mathcal{A}) = \prod_{((i,j),(k,\ell))\in\mathcal{A}} \mathrm{p}_{w^{(s)}|w^{(t)}}(\{w^{(s)}_{i+1}, w^{(s)}_{i+2}, \ldots, w^{(s)}_{j}\} \mid \{w^{(t)}_{k+1}, w^{(t)}_{k+2}, \ldots, w^{(t)}_{\ell}\})$$

Vauquois Pyramid



- Hierarchy of concepts and distances between them in different languages
- Lowest level: individual words/ characters
- Higher levels: syntax, semantics
- Interlingua: Generic language-agnostic representation of meaning

Syntactic MT

- Rather than use phrases, use a *synchronous context-free grammar*: constructs "parallel" trees in two languages simultaneously

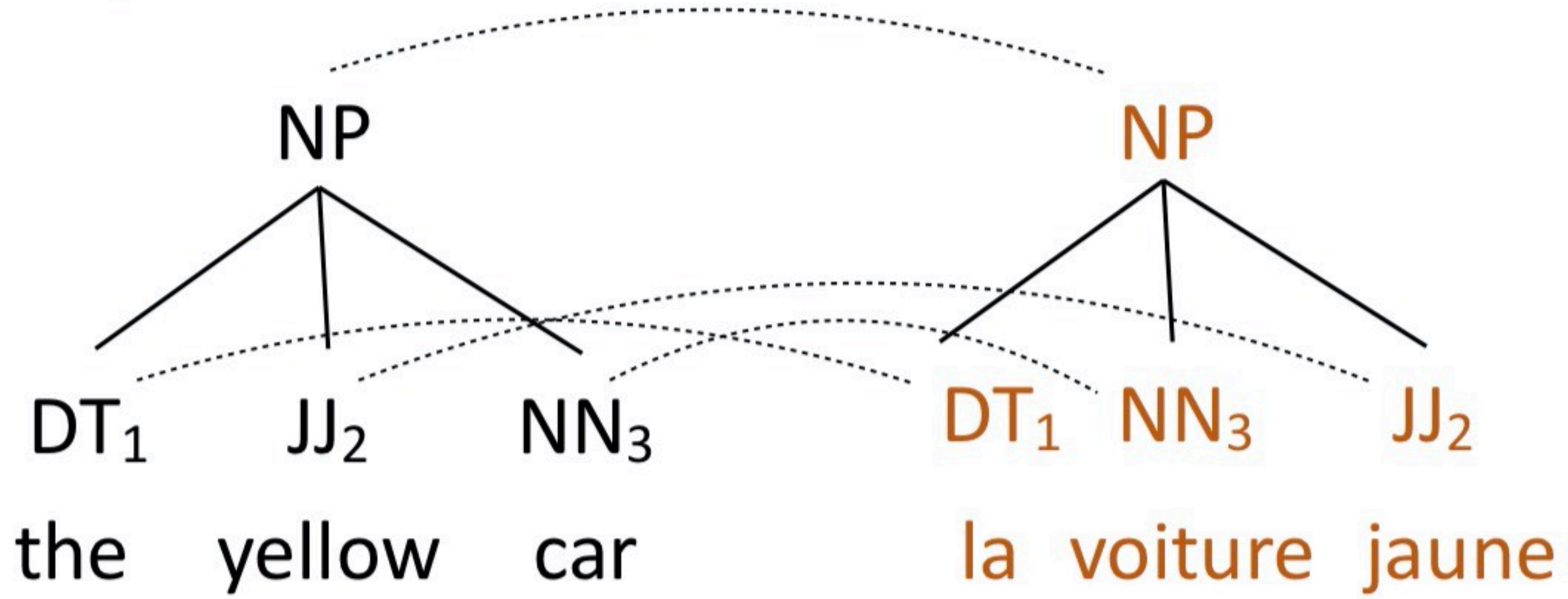
$NP \rightarrow [DT_1\ JJ_2\ NN_3;\ DT_1\ NN_3\ JJ_2]$

$DT \rightarrow$ [the, la]

$DT \rightarrow$ [the, le]

$NN \rightarrow$ [car, voiture]

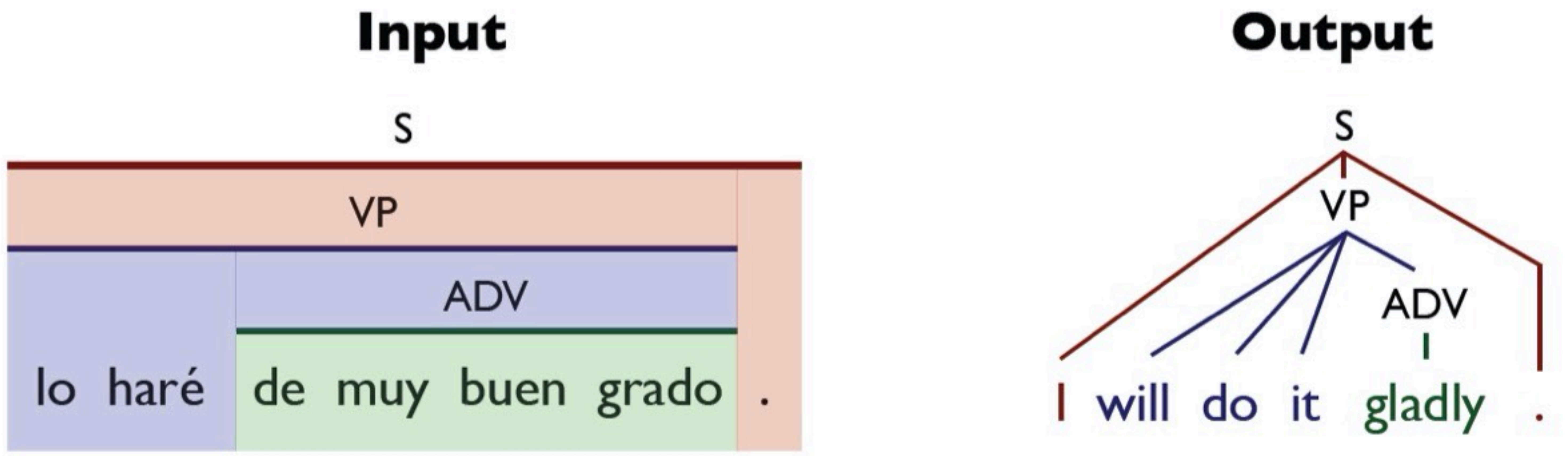
$JJ \rightarrow$ [yellow, jaune]



- Assumes parallel syntax up to reordering
- Translation = parse the input with "half" the grammar, read off other half

(Slide credit: Greg Durrett)

Syntactic MT



Grammar

S → ⟨ VP . ; I VP . ⟩ **OR** S → ⟨ VP . ; you VP . ⟩

VP → ⟨ lo haré ADV ; will do it ADV ⟩

S → ⟨ lo haré ADV . ; I will do it ADV . ⟩

ADV → ⟨ de muy buen grado ; gladly ⟩

- Relax this by using lexicalized rules, like “syntactic phrases”
- Leads to HUGE grammars, parsing is slow

Slide credit: Dan Klein

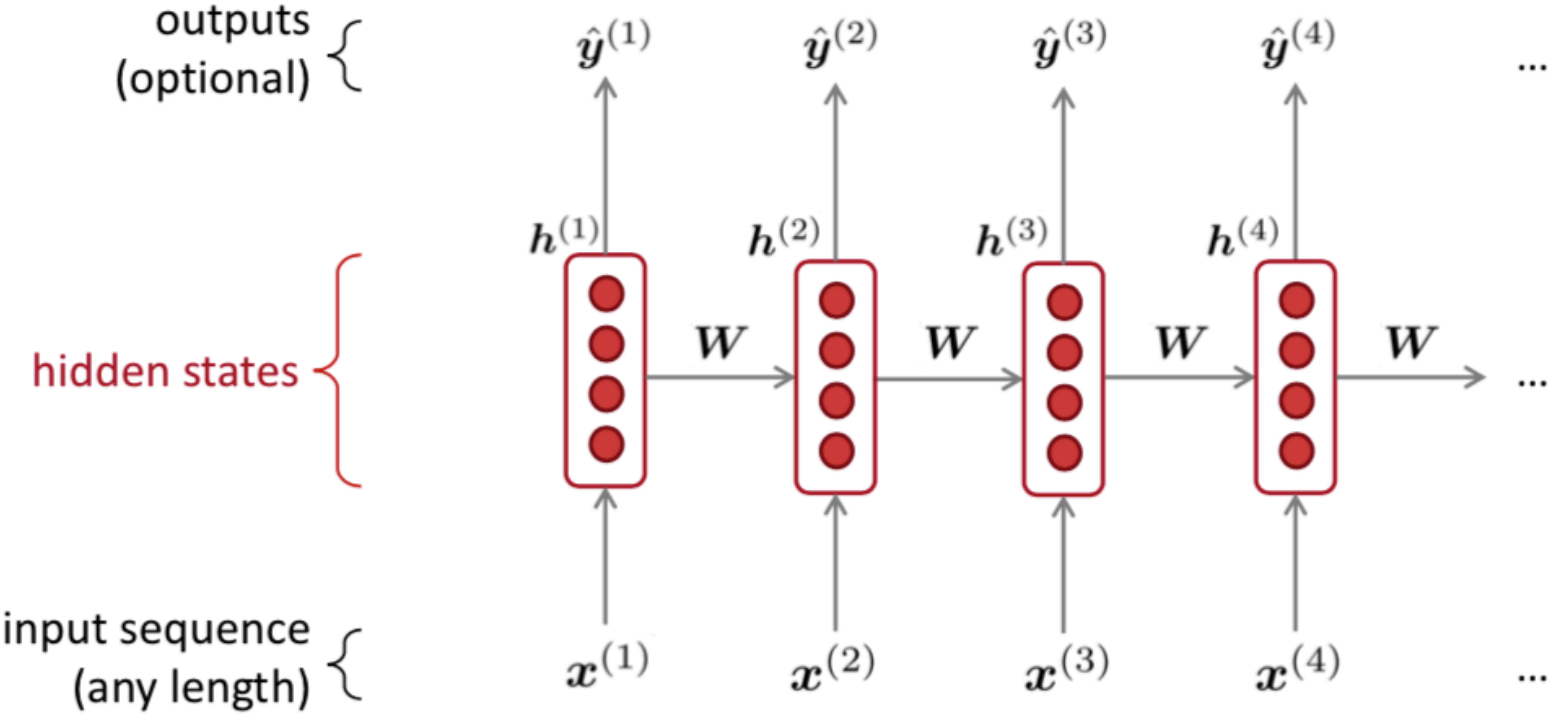
Neural Machine Translation

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

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

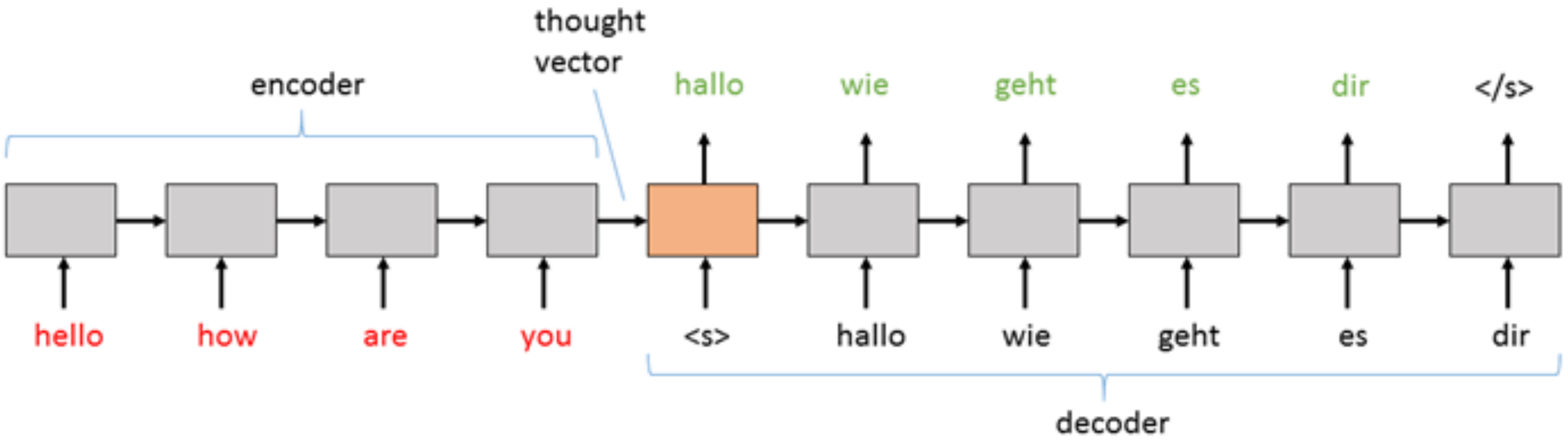


Recall: RNNs

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

A) 10
B) 128
C) ∞

Sequence to Sequence learning (Seq2seq)

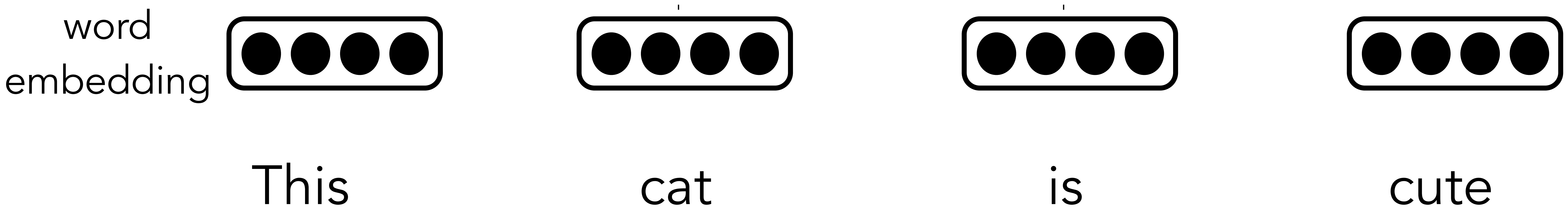


- Encode entire input sequence into a single vector **(using an RNN)**
- Decode one word at a time **(again, using an RNN!)**
- Beam search for better inference
- Learning is not trivial! (vanishing/exploding gradients)

(Sutskever et al., 2014)

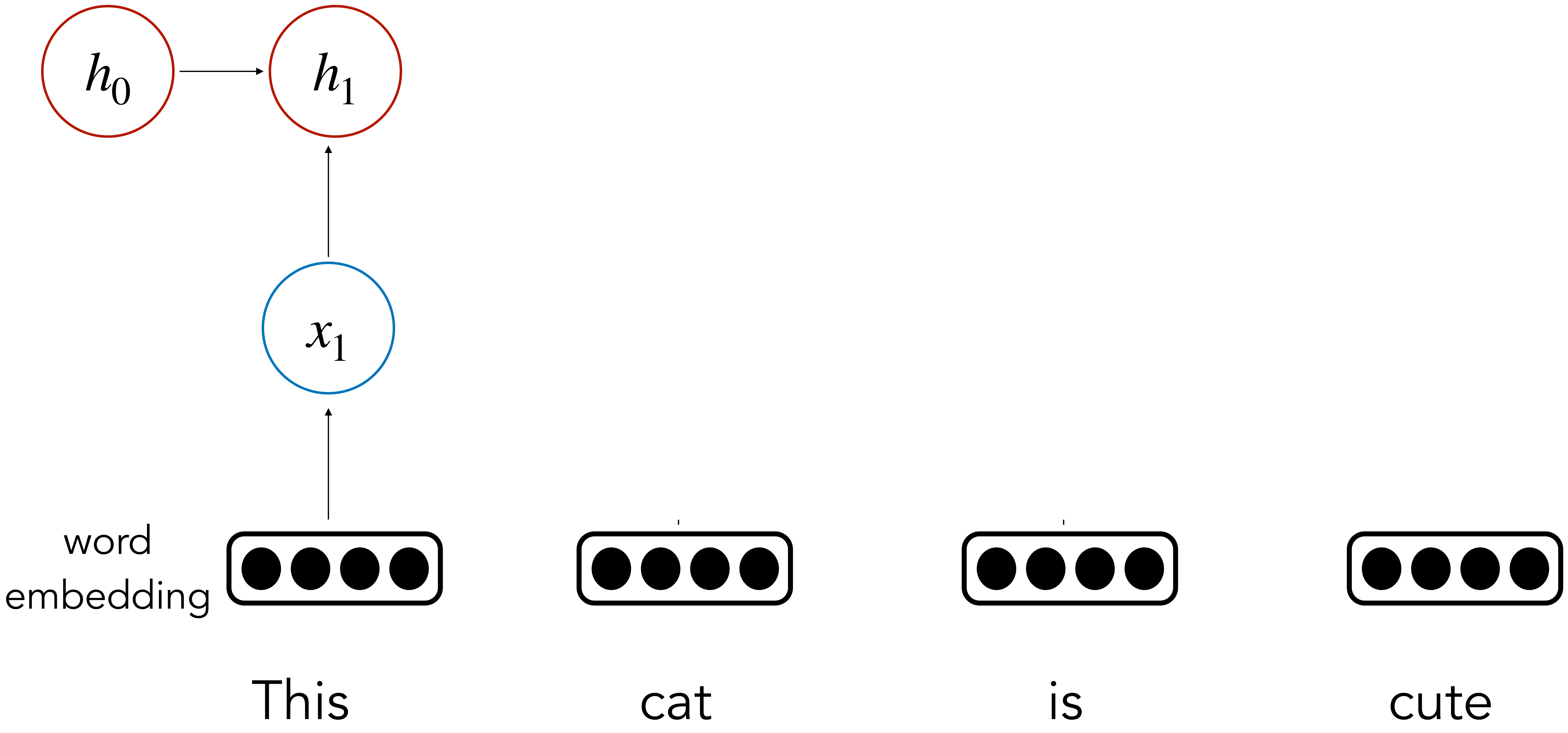
Encoder

Sentence: This cat is cute



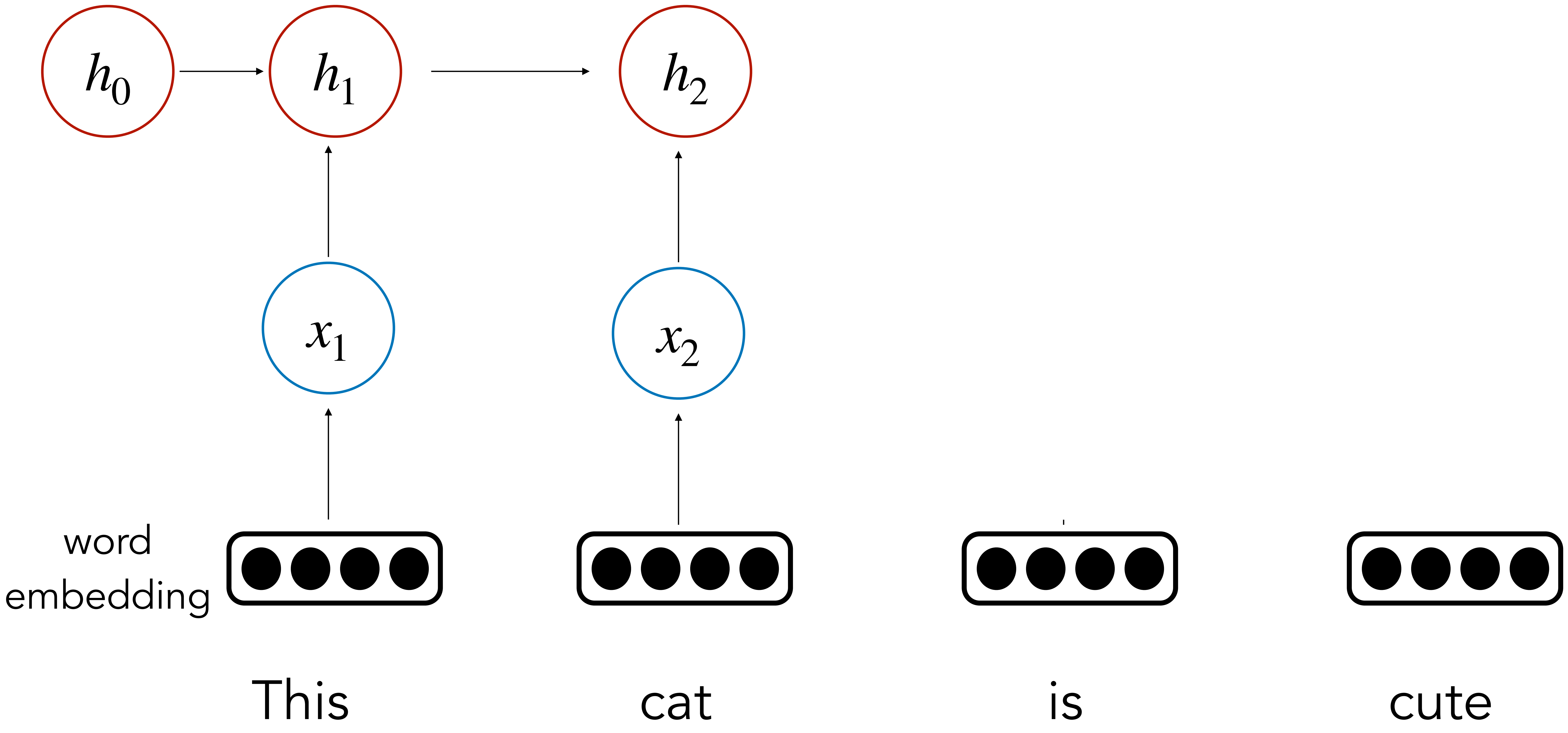
Encoder

Sentence: This cat is cute



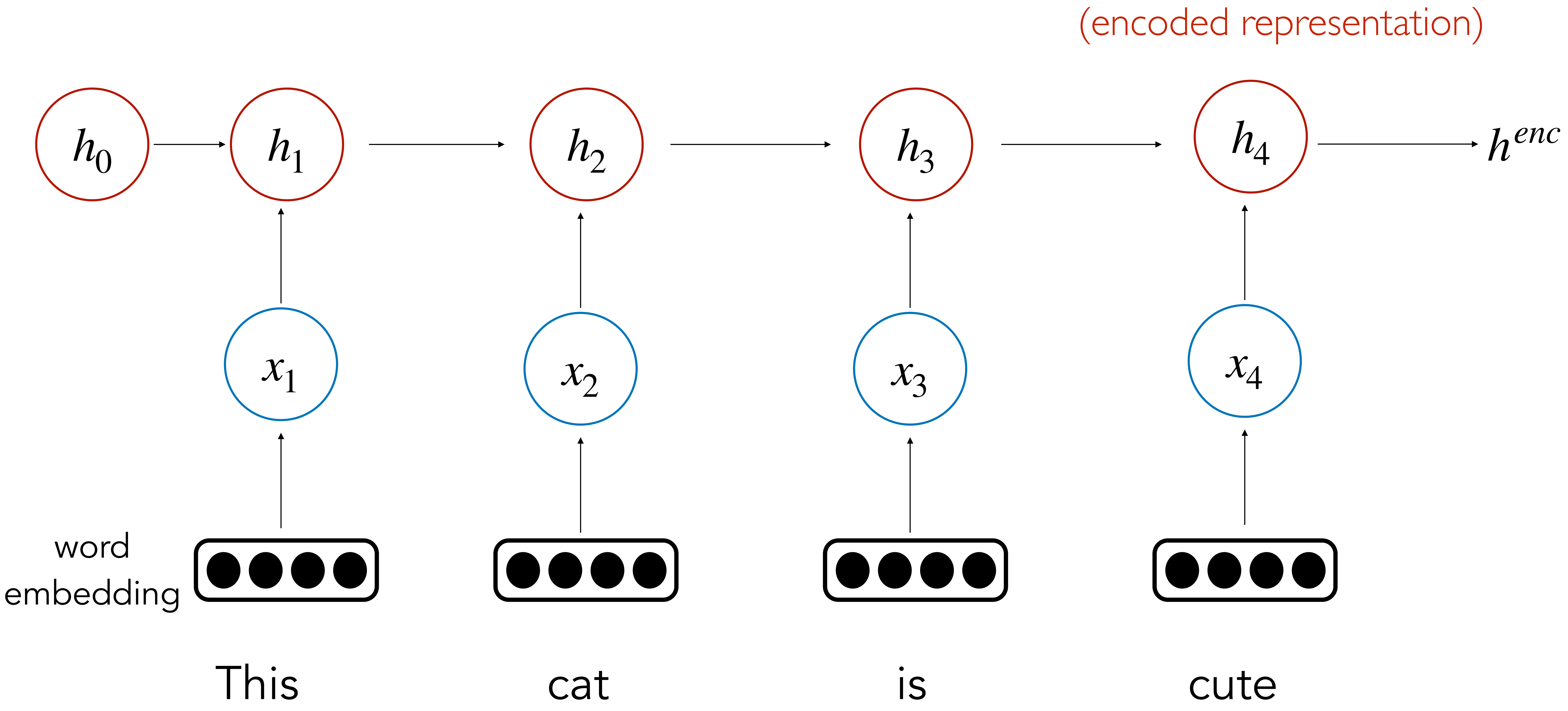
Encoder

Sentence: This cat is cute

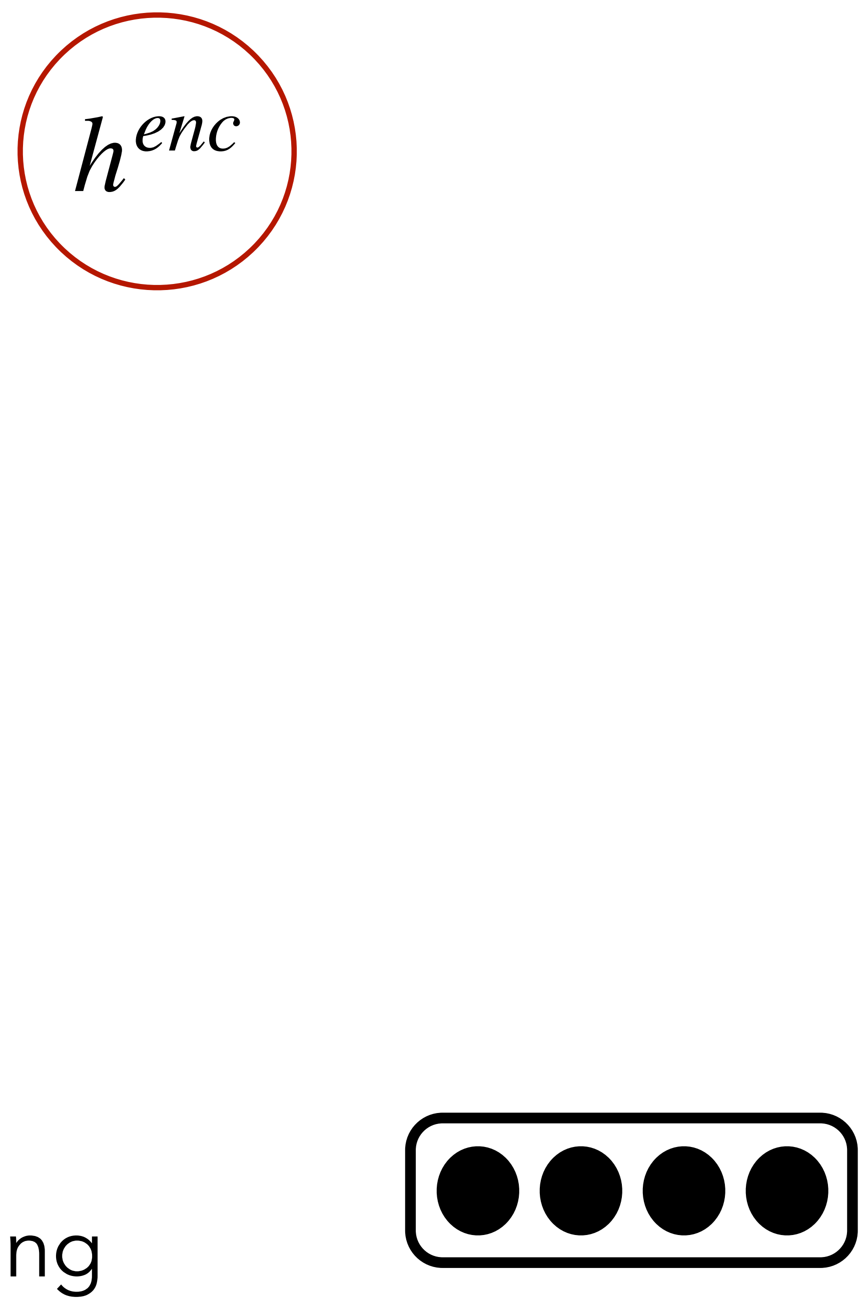


Encoder

Sentence: This cat is cute



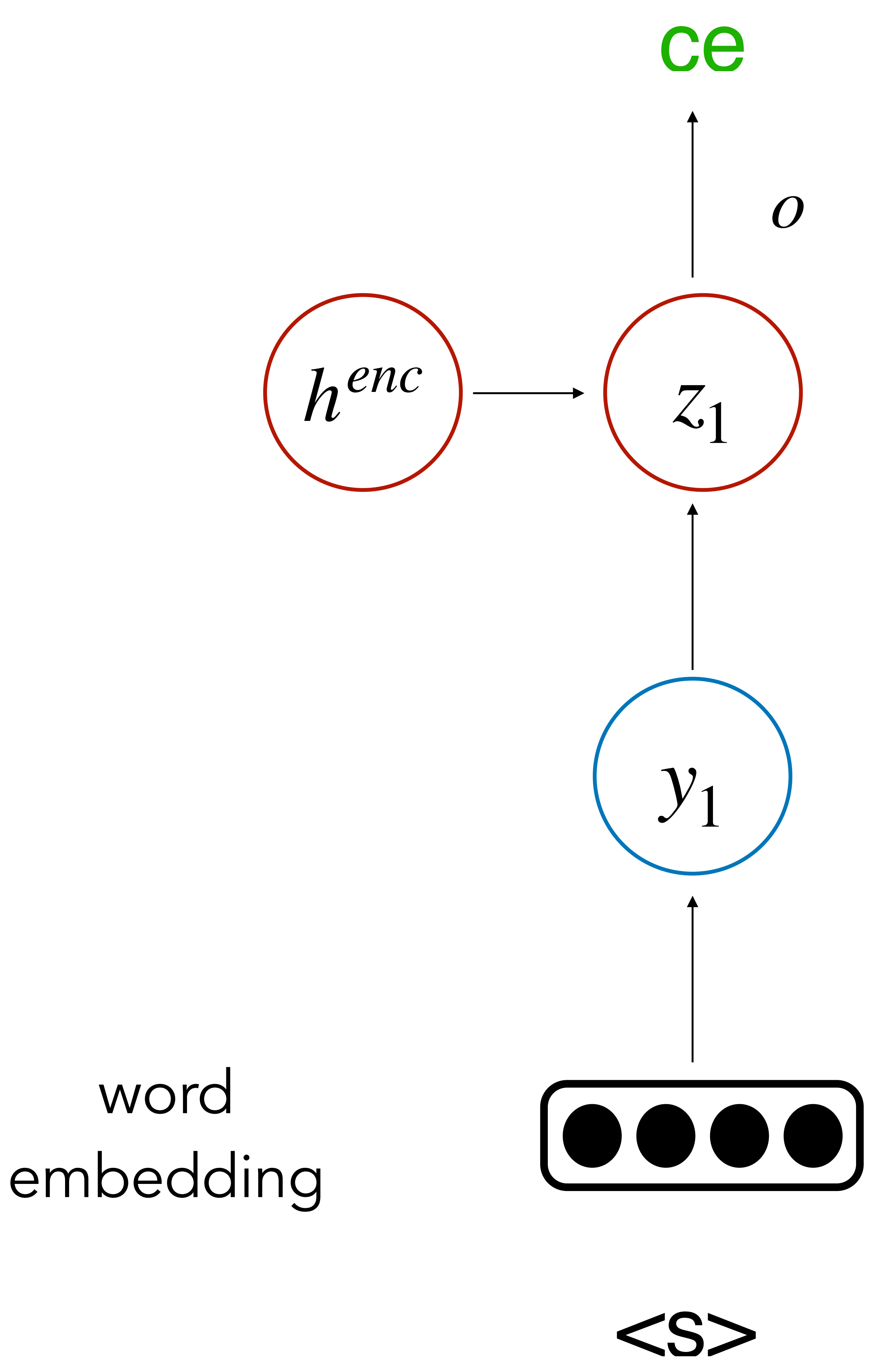
Decoder



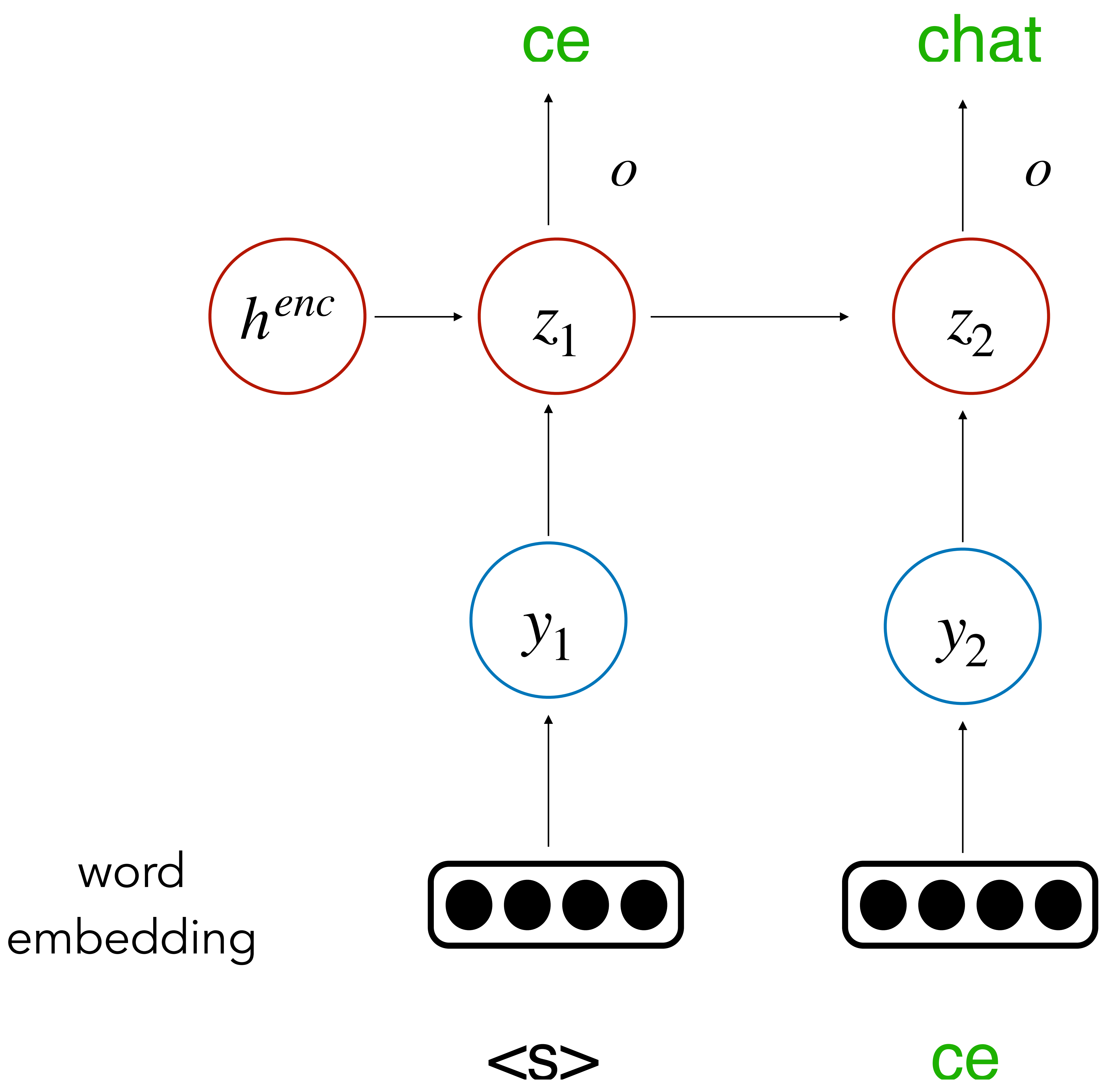
word
embedding

<s>

Decoder

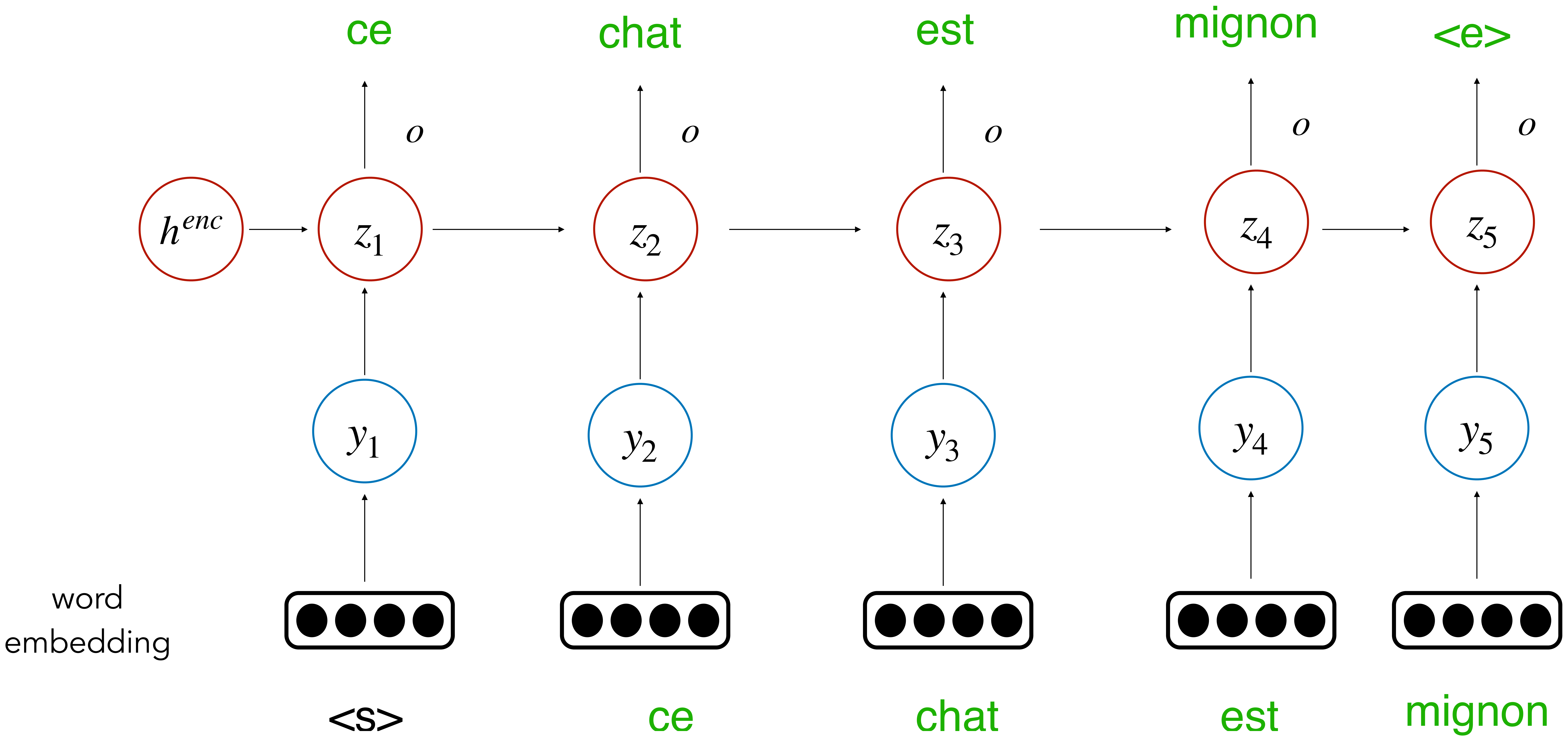


Decoder



Decoder

- A conditioned language model



Seq2seq training

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

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

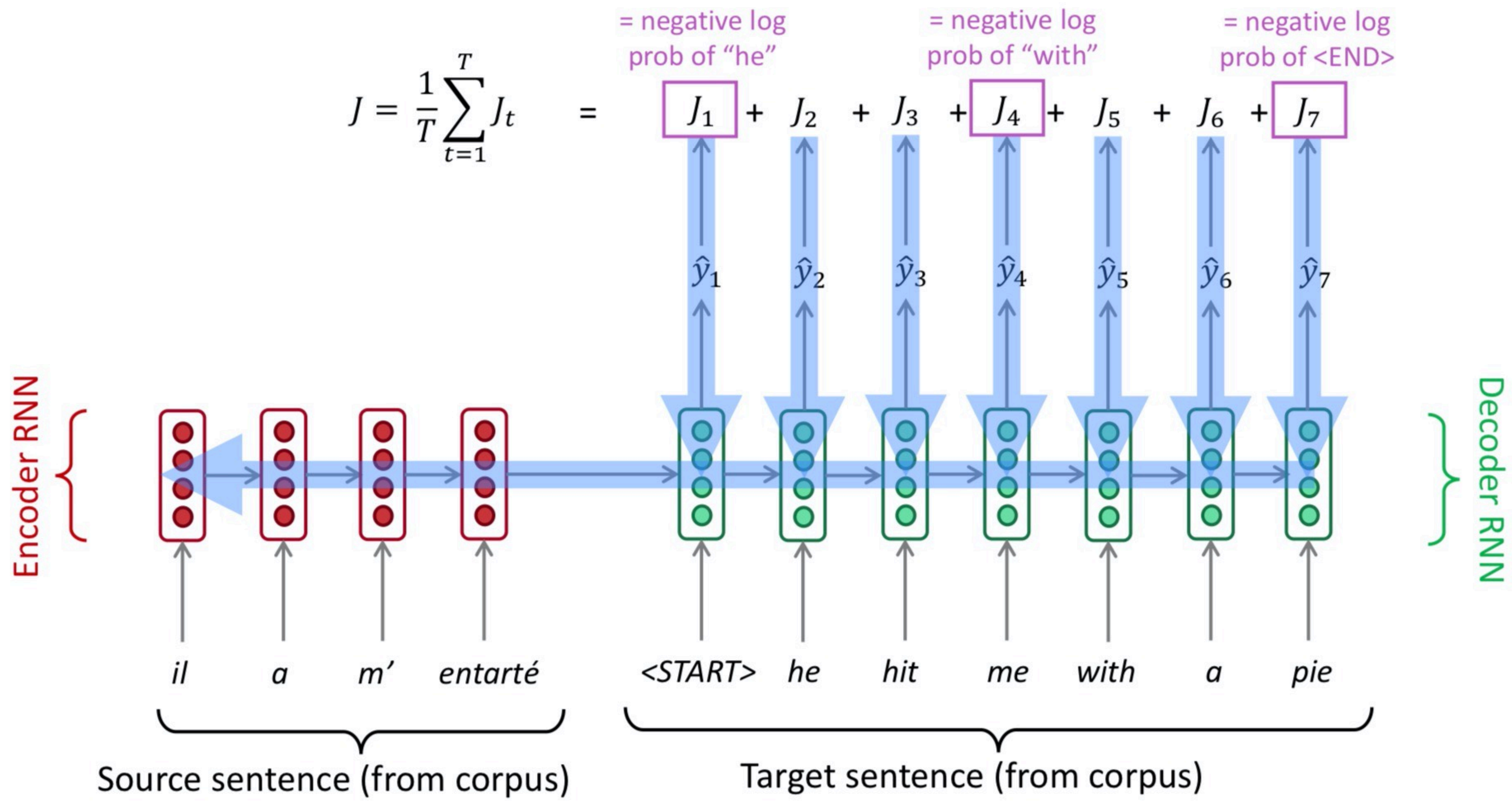
- Back-propagate gradients through *both decoder and encoder*
- Need a really big corpus

36M sentence pairs

Russian: Машинный перевод - это круто!

English: Machine translation is cool!

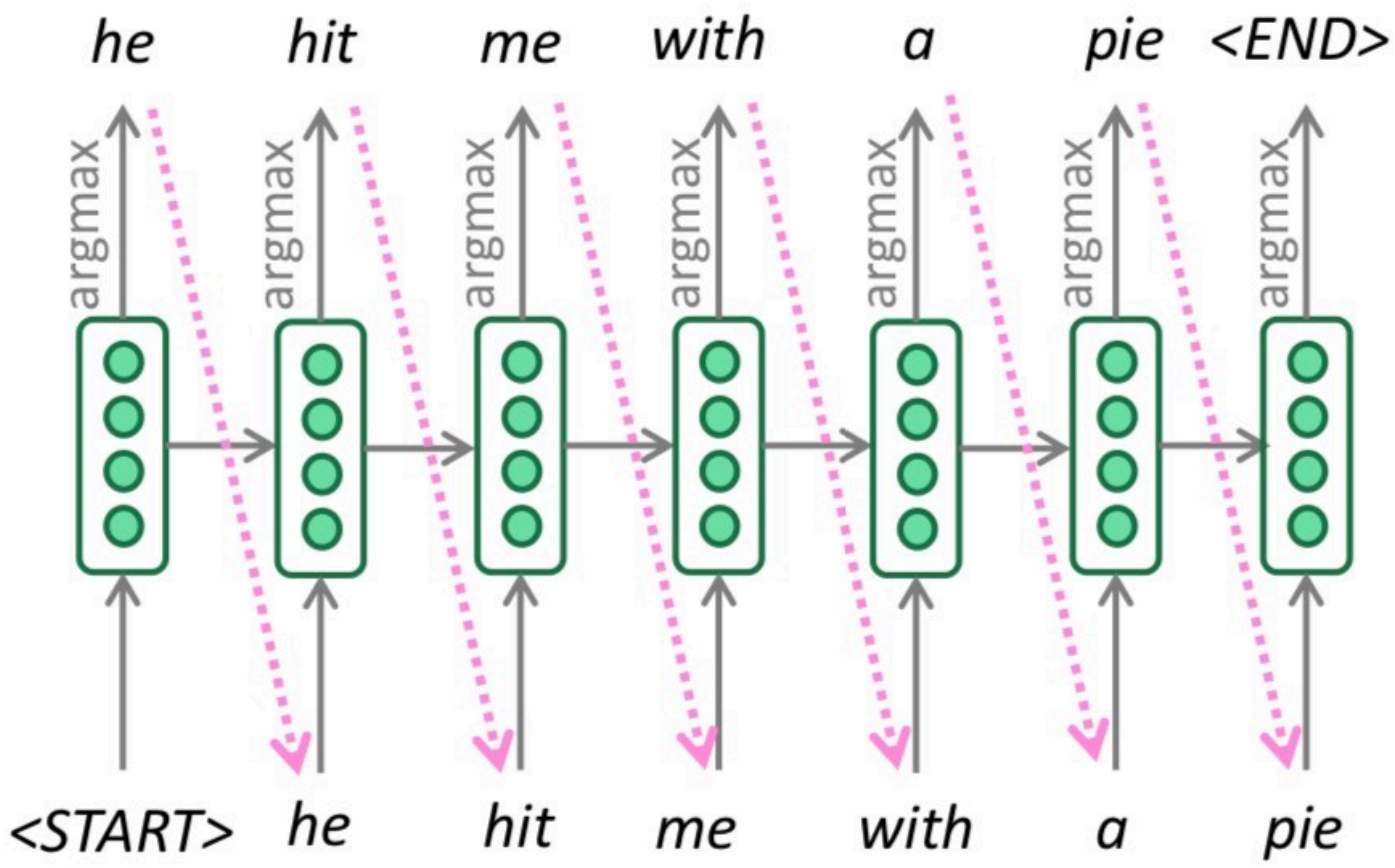
Seq2seq training



Seq2seq is optimized as a **single system.**
Backpropagation operates *"end-to-end"*.

(slide credit: Abigail See)

Greedy decoding



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

Exhaustive search?

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

- 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

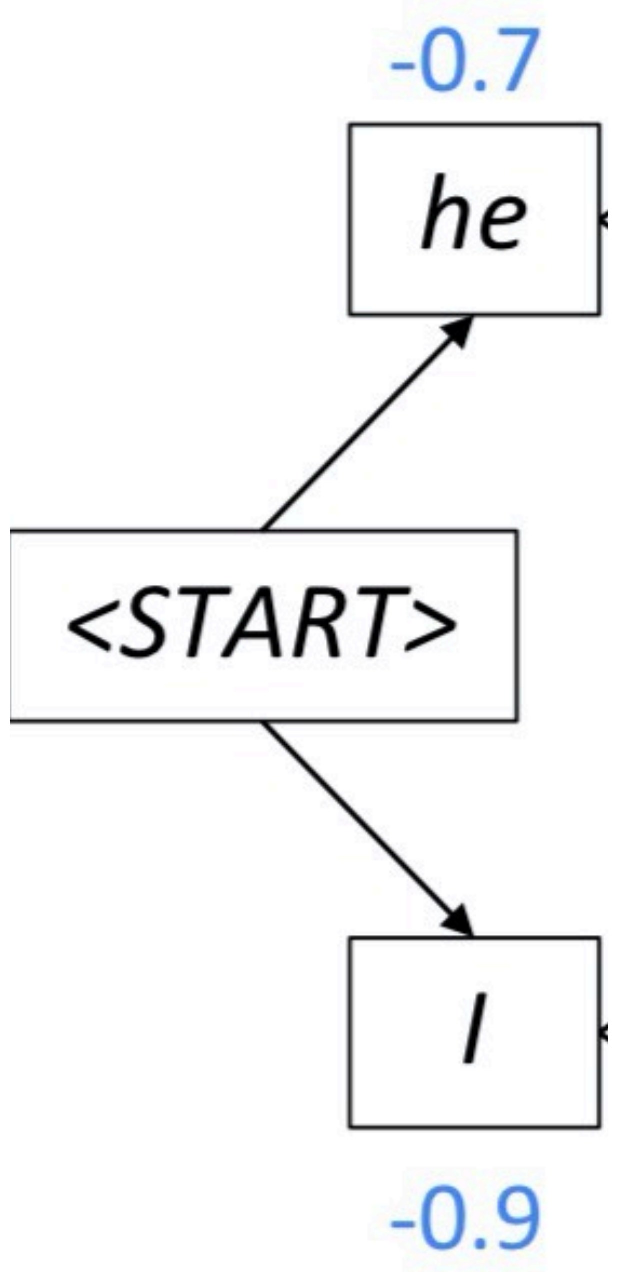
$$\sum_{t=1}^{j} \log P(y_t \mid y_1, \ldots, y_{t-1}, x_1, \ldots, x_n)$$

- Not guaranteed to be optimal

- More efficient than exhaustive search

Beam decoding

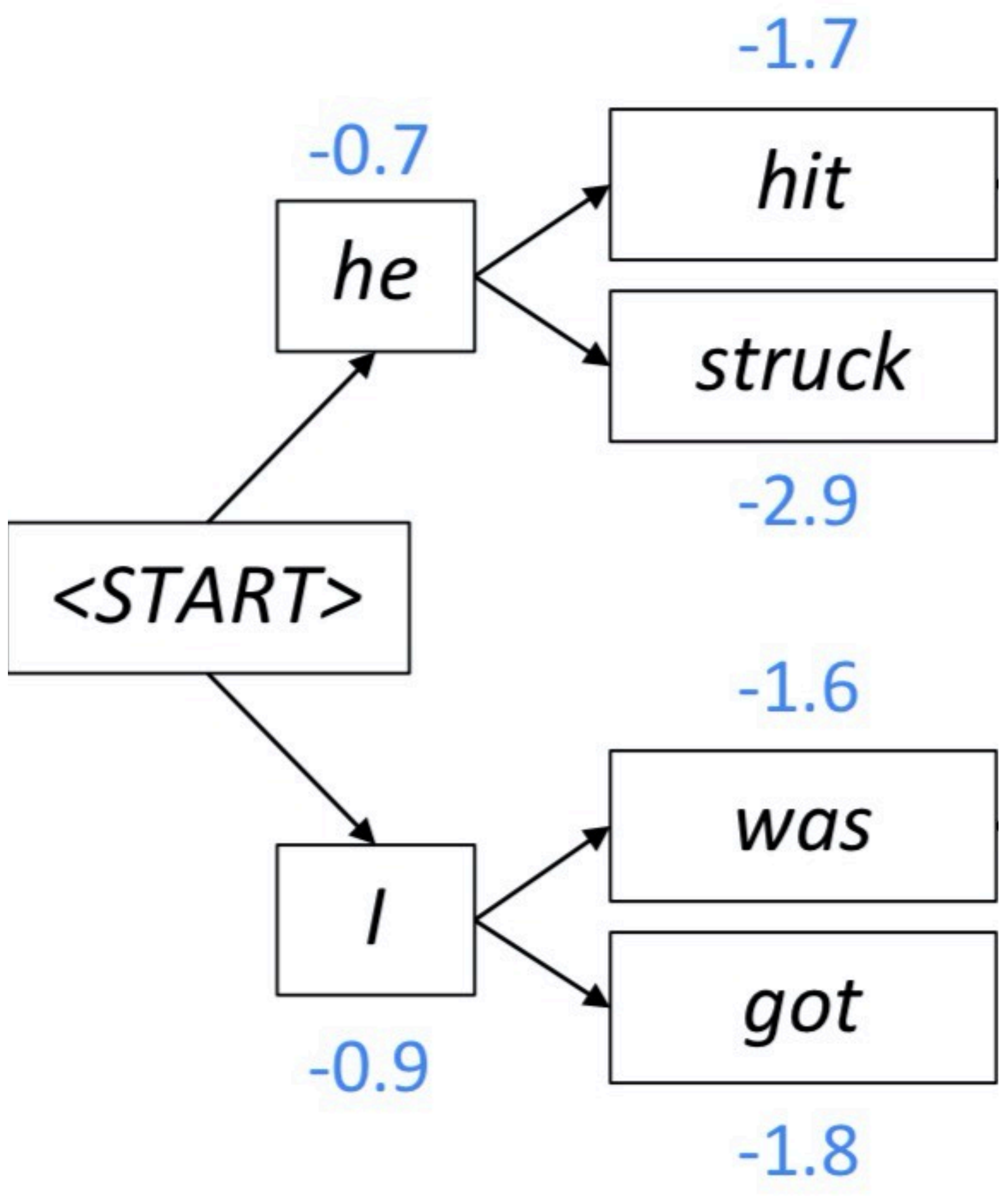
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

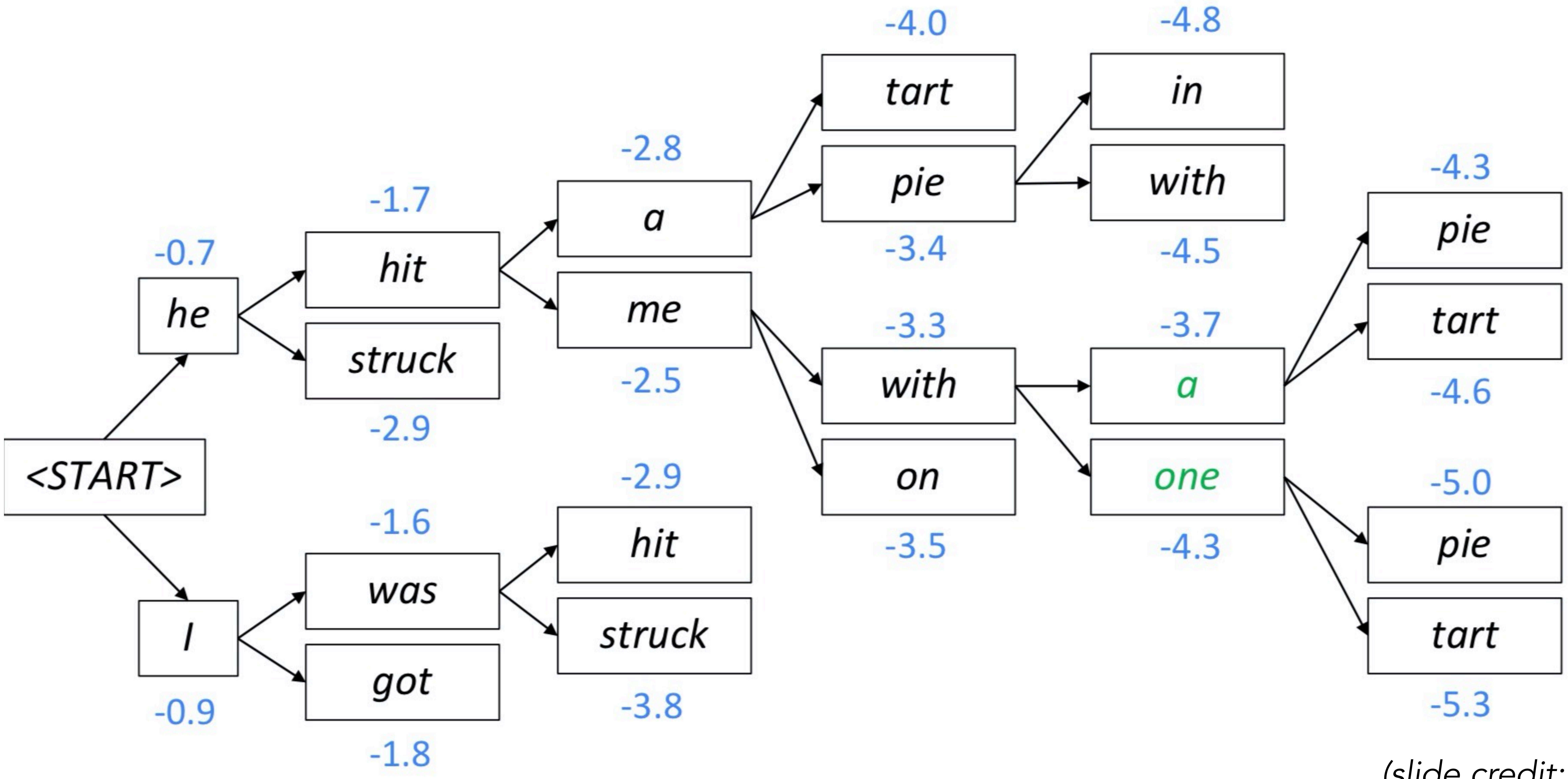
Beam size = k = 2. Blue numbers = $\mathrm{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\mathrm{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

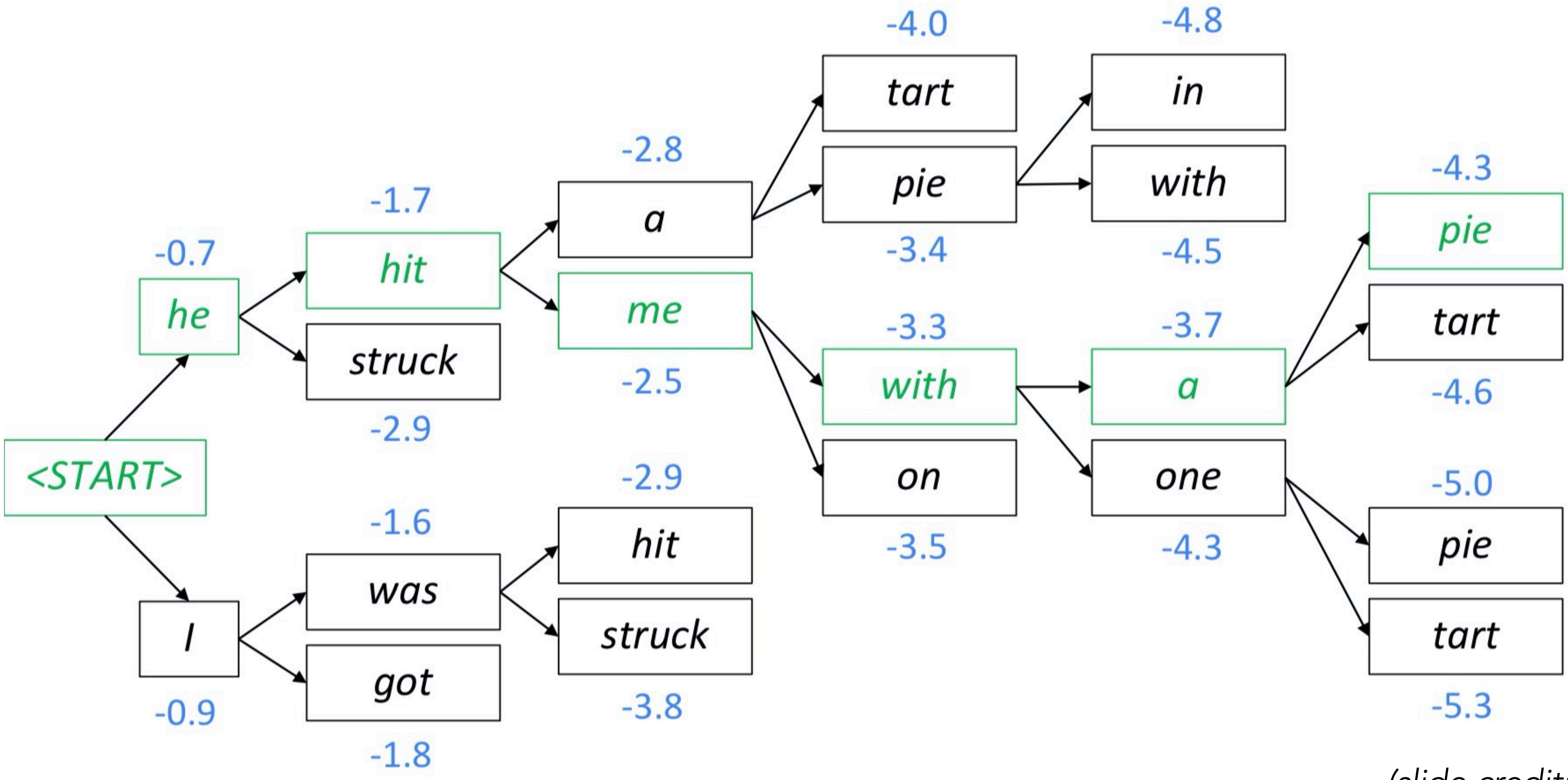
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(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

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

  - Otherwise shorter hypotheses have higher scores

NMT vs SMT

Pros

-

Cons

-