Sequence to Sequence Models

Antoine Bosselut

EPFL

nlp lab

Section Outline

- **Sequence-to-sequence models:** Overview, Examples, Training
- **Sequence-to-sequence shortcomings:** Long-range dependencies, Temporal bottleneck
- **Improvements:** Attention mechanisms

Question

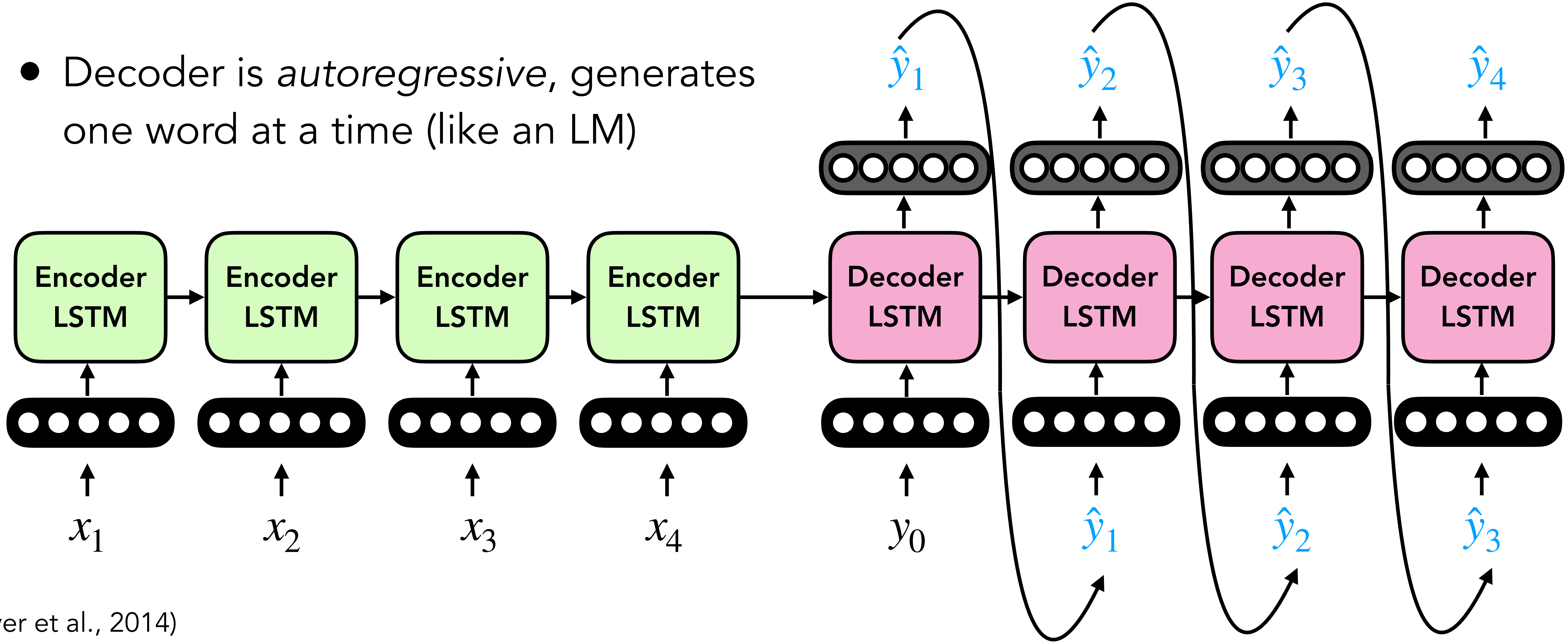
How can we use recurrent neural networks for tasks other than language modelling?

Machine Translation involves more than estimating the probability next word; requires generating a full translation of a given context into another language

Encoder-Decoder Models

- Encode a sequence fully with one model (**encoder**) and use its representation to seed a second model that decodes another sequence (**decoder**)

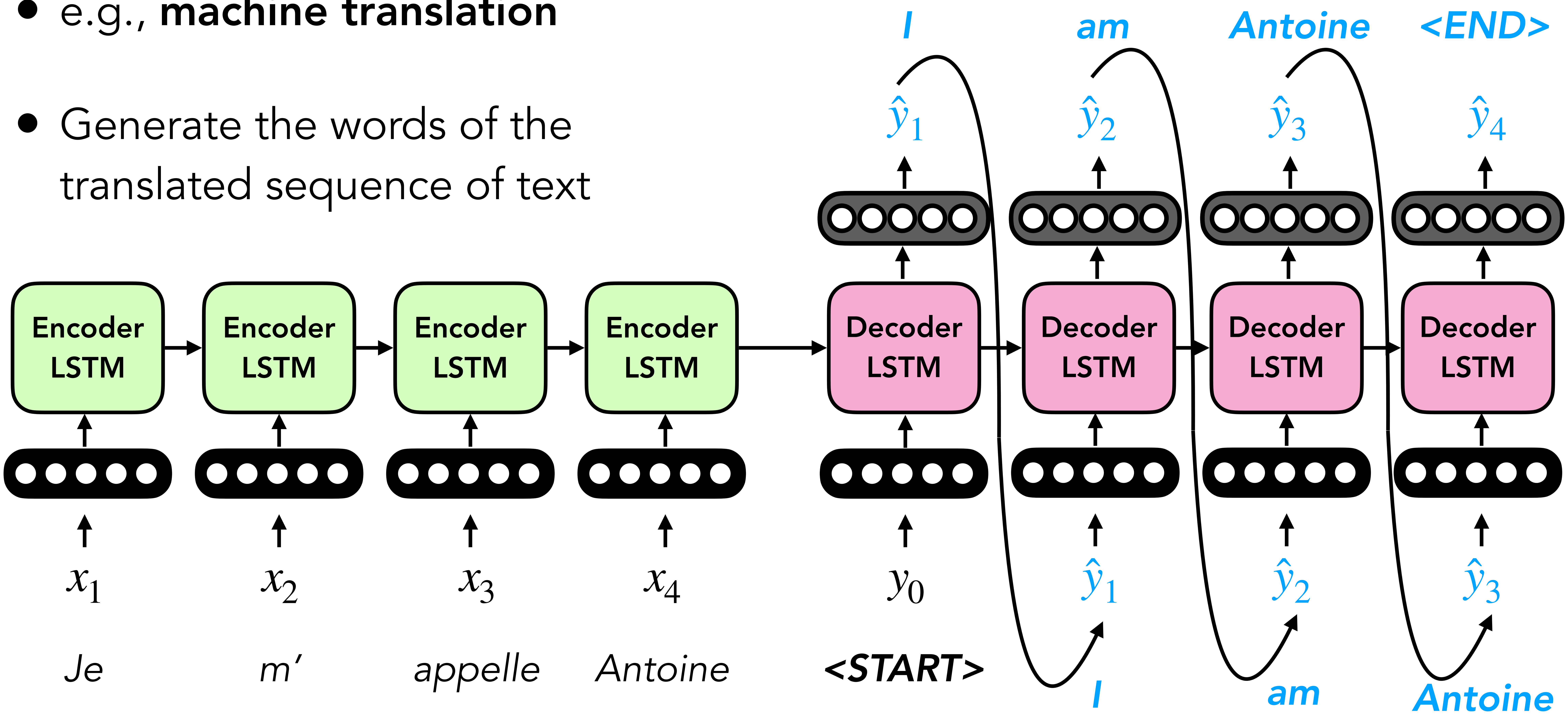
- Decoder is *autoregressive*, generates one word at a time (like an LM)



(Sutskever et al., 2014)

Encoder-Decoder Models

- e.g., **machine translation**
- Generate the words of the translated sequence of text



(Sutskever et al., 2014)

Encoder-Decoder Models

- Input doesn't need to be text
- e.g., **image captioning**

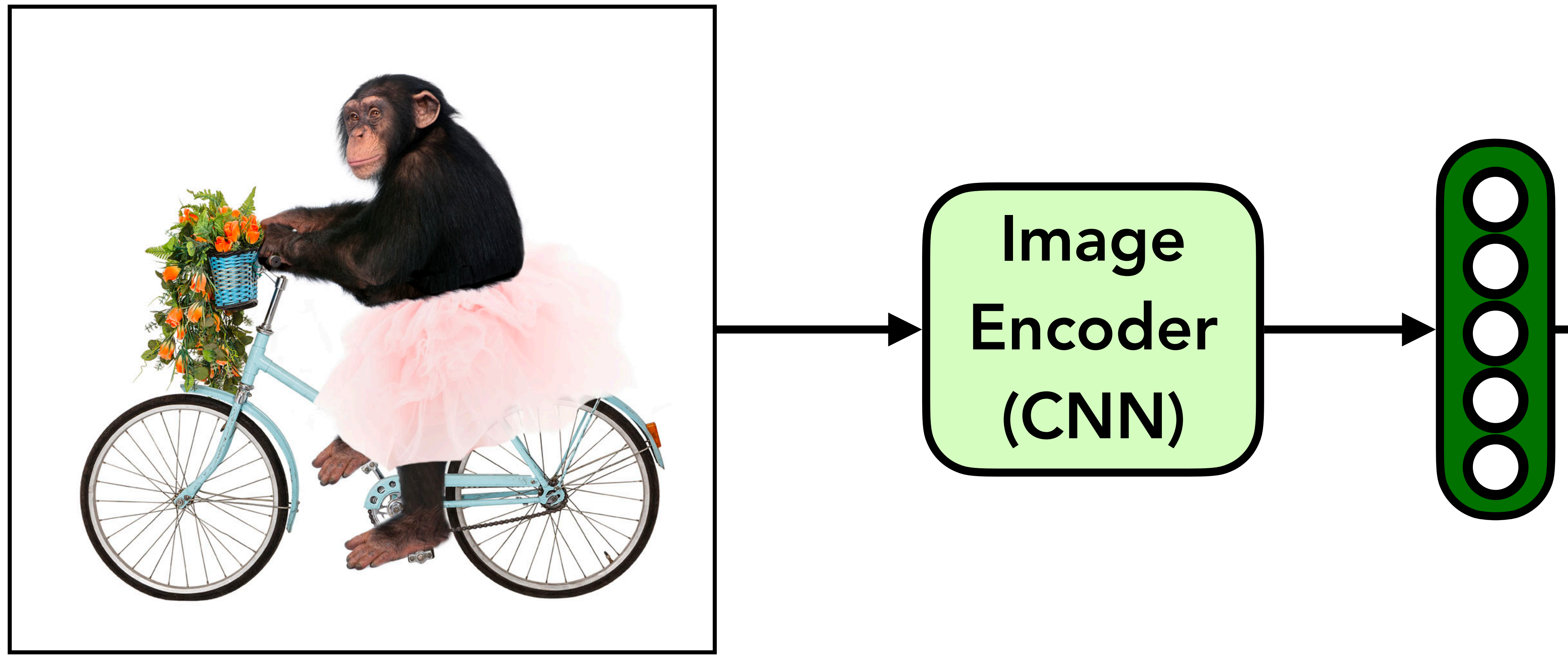
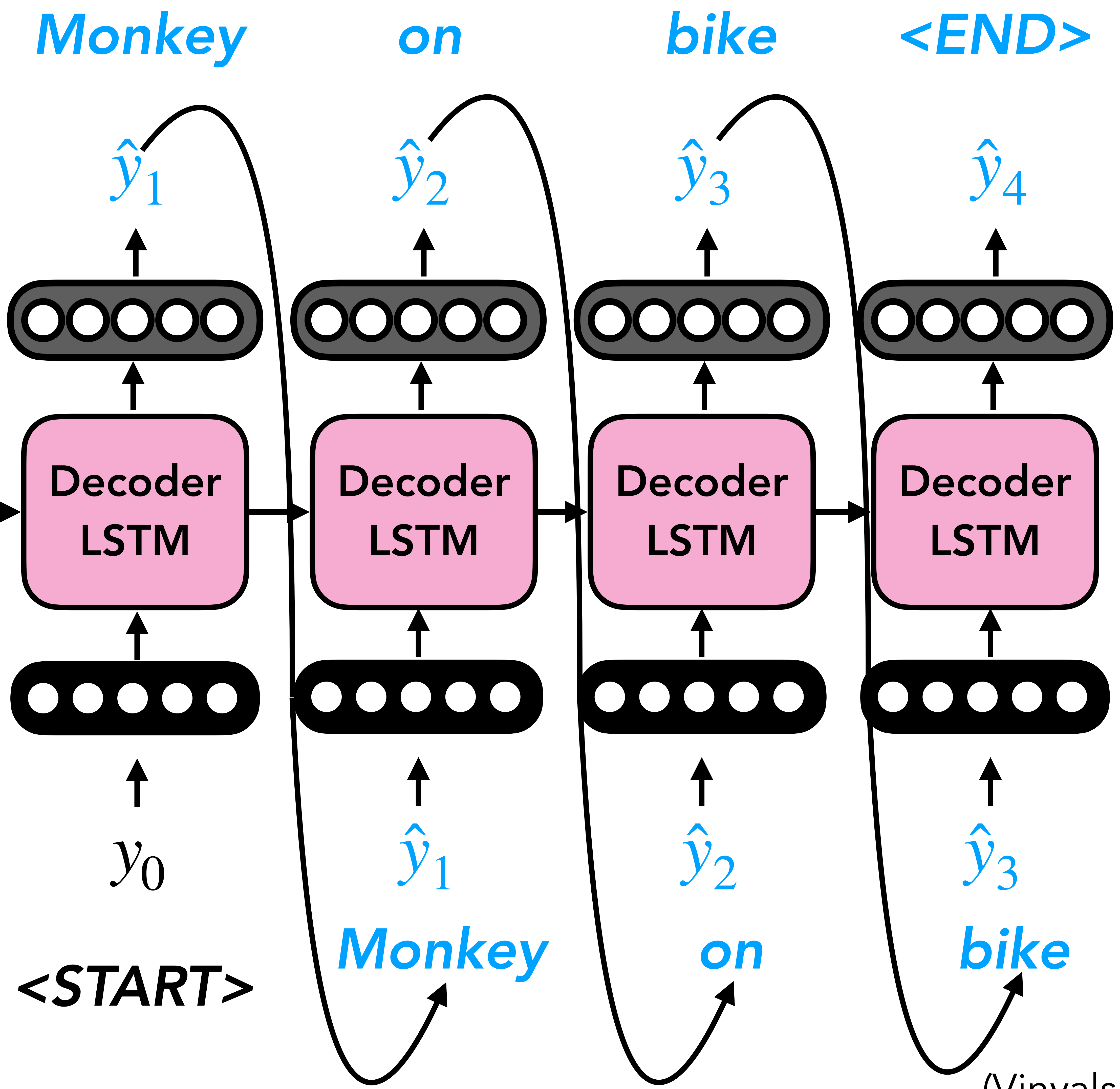


Photo credit: J Hovenstine Studios

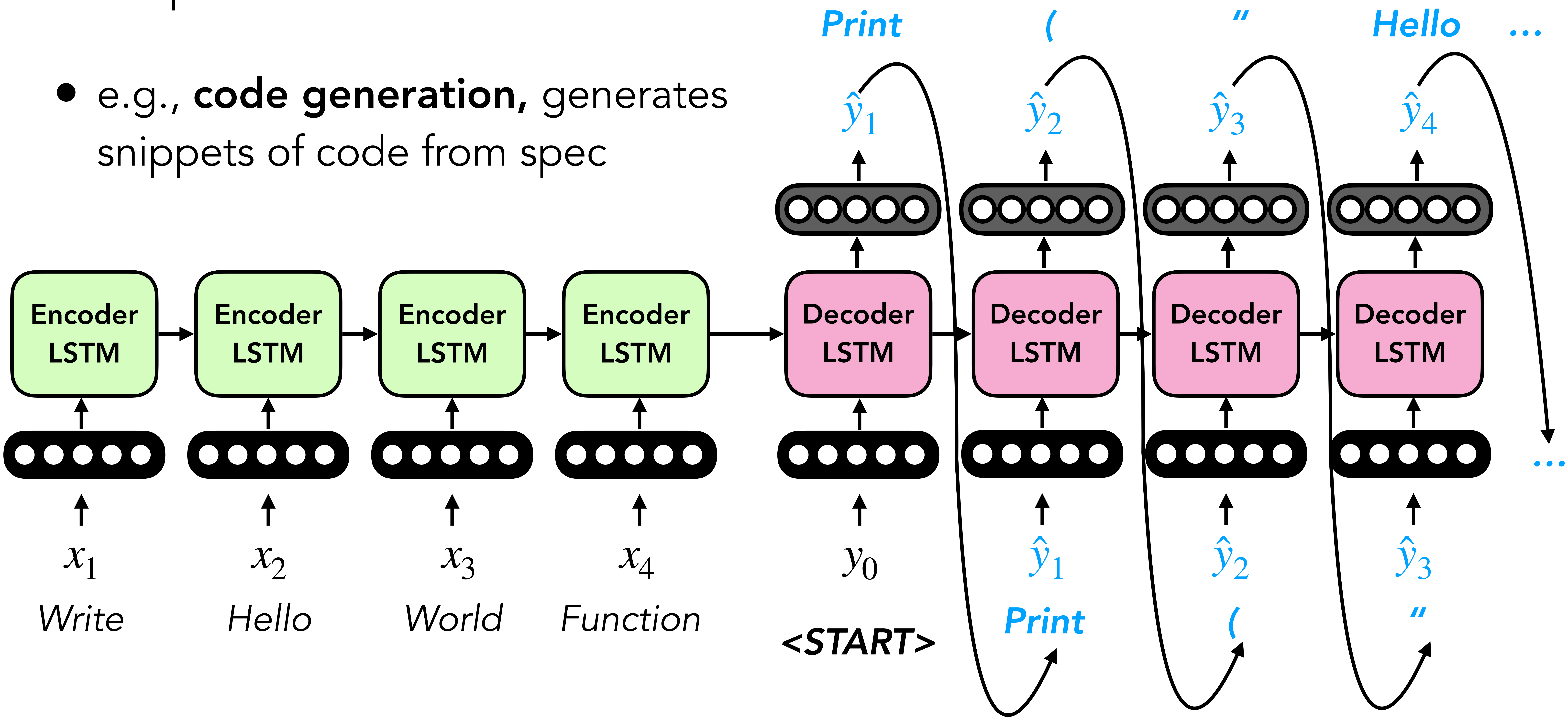


- Generate words of image description

(Vinyals et al., 2014)

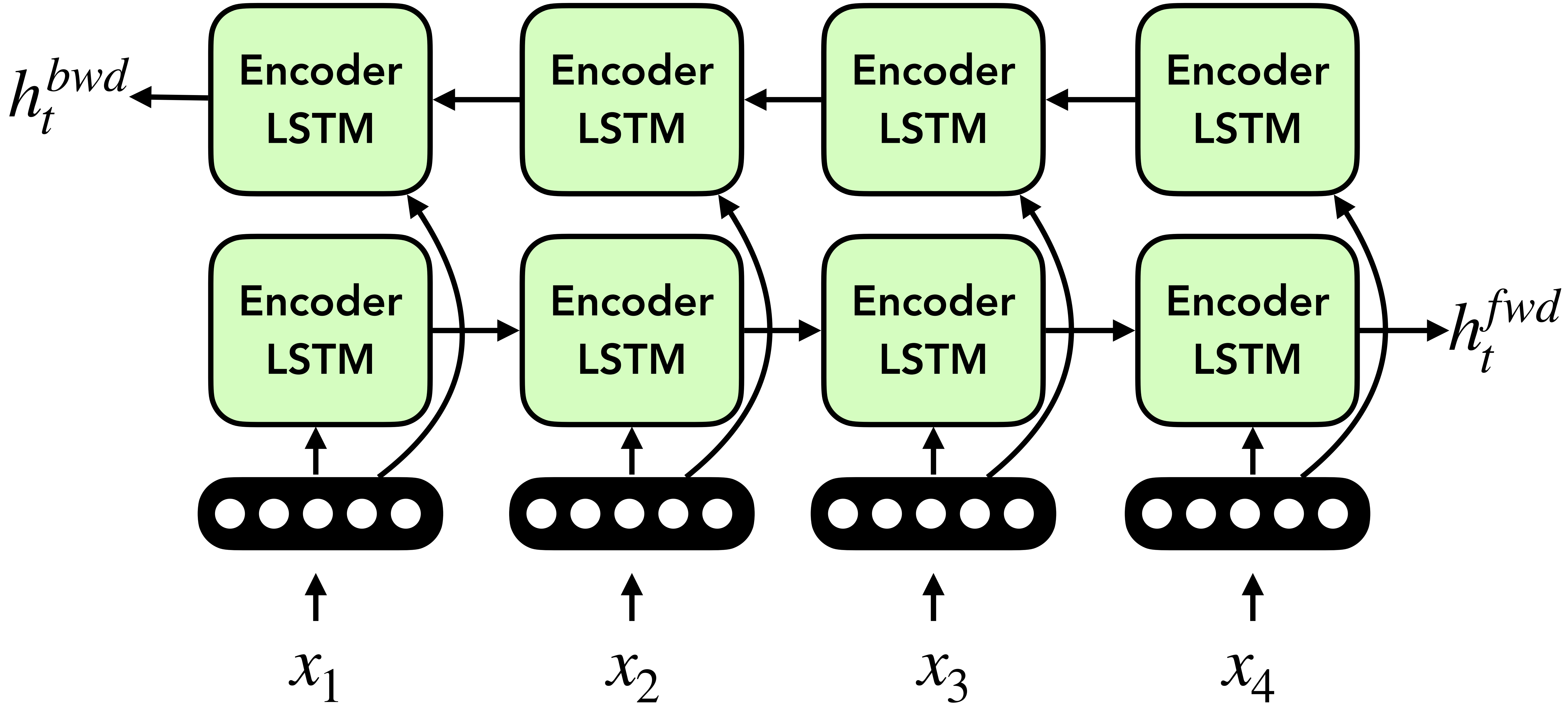
Encoder-Decoder Models

- Output can be other forms of text
- e.g., **code generation,** generates snippets of code from spec



Bidirectional Encoders

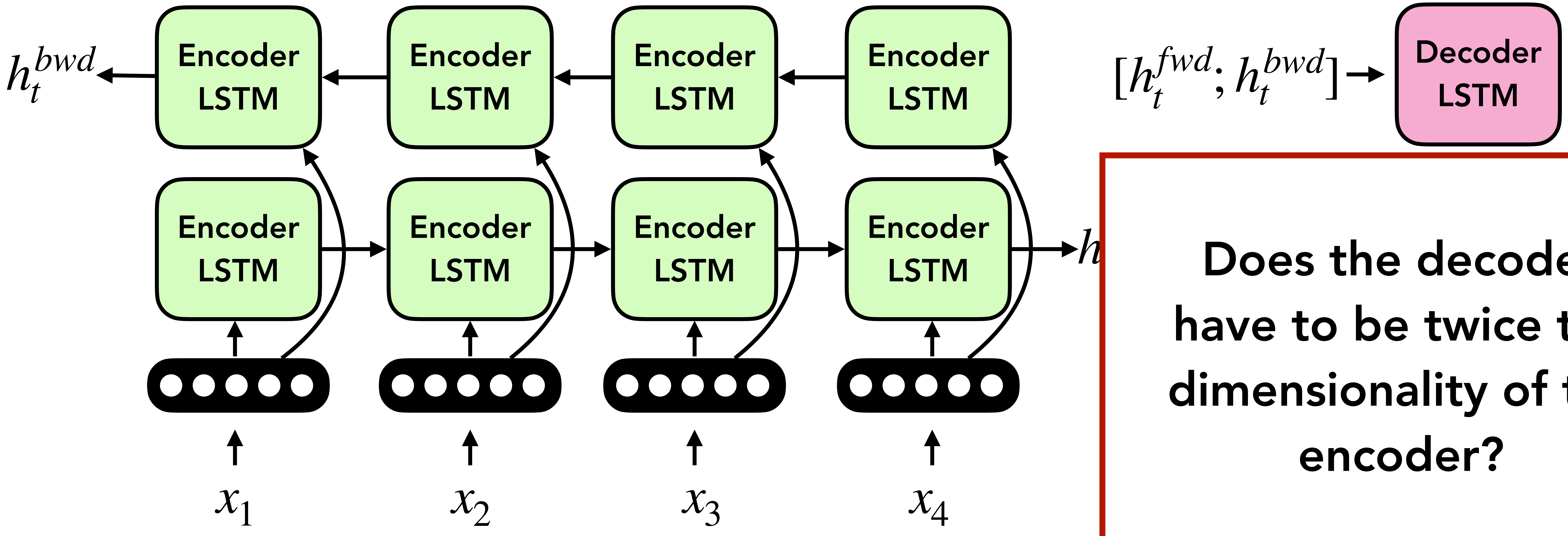
- Decoder needs to be unidirectional (autoregressive models can't know the future…)
- Encoder sequence representation augmented by encoding in both directions



(Schuster and Paliwal, 1997)

Bidirectional Encoders

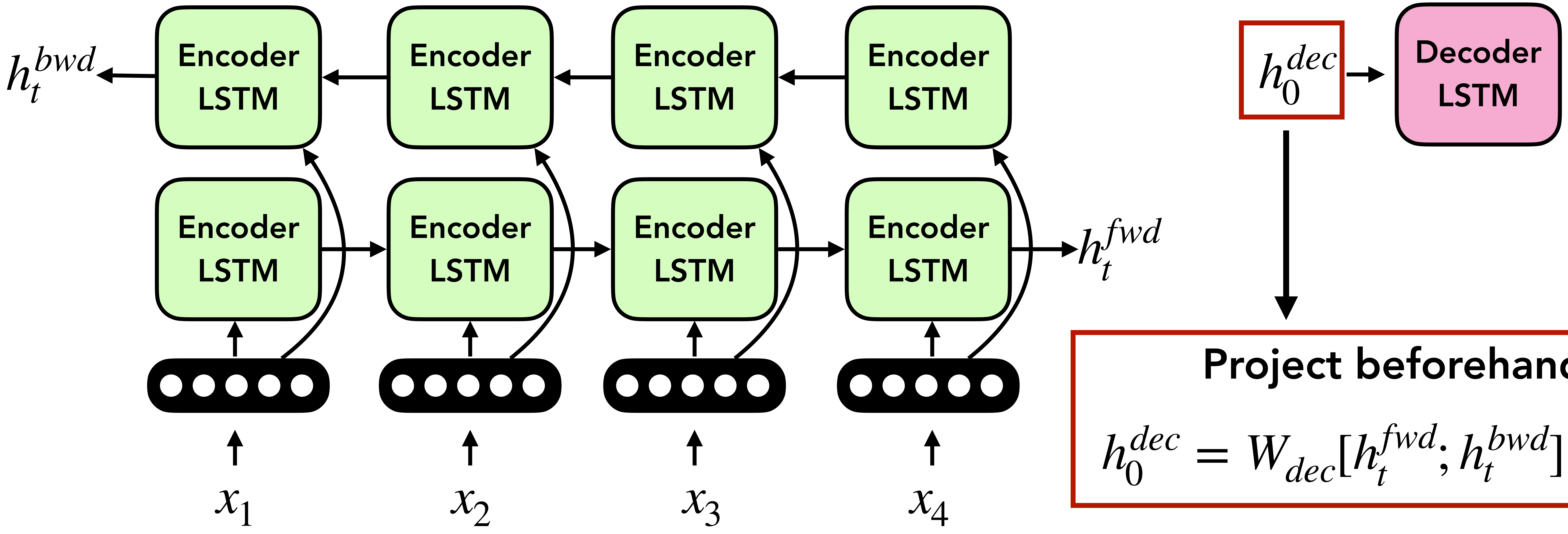
- Decoder needs to be unidirectional (autoregressive models can't know the future…)
- Encoder sequence representation augmented by encoding in both directions



Does the decoder have to be twice the dimensionality of the encoder?

Bidirectional Encoders

- Decoder needs to be unidirectional (autoregressive models can't know the future…)
- Encoder sequence representation augmented by encoding in both directions



Project beforehand:

$$h_0^{dec} = W_{dec}[h_t^{fwd}; h_t^{bwd}] + b_{dec}$$

Training Encoder-Decoder Models

- With a language model, we had practically unlimited data!
  - We were only learning which words followed others, so any text would do!
- With encoder-d quences align with others
  - **Machine Translat** t have the same meaning)
  - **Image Captioning**: Need paired image-text data (images and their description)
  - **Code Generation**: Need paired code-text data (e.g., code and their comments)
  - And so on… for other tasks!

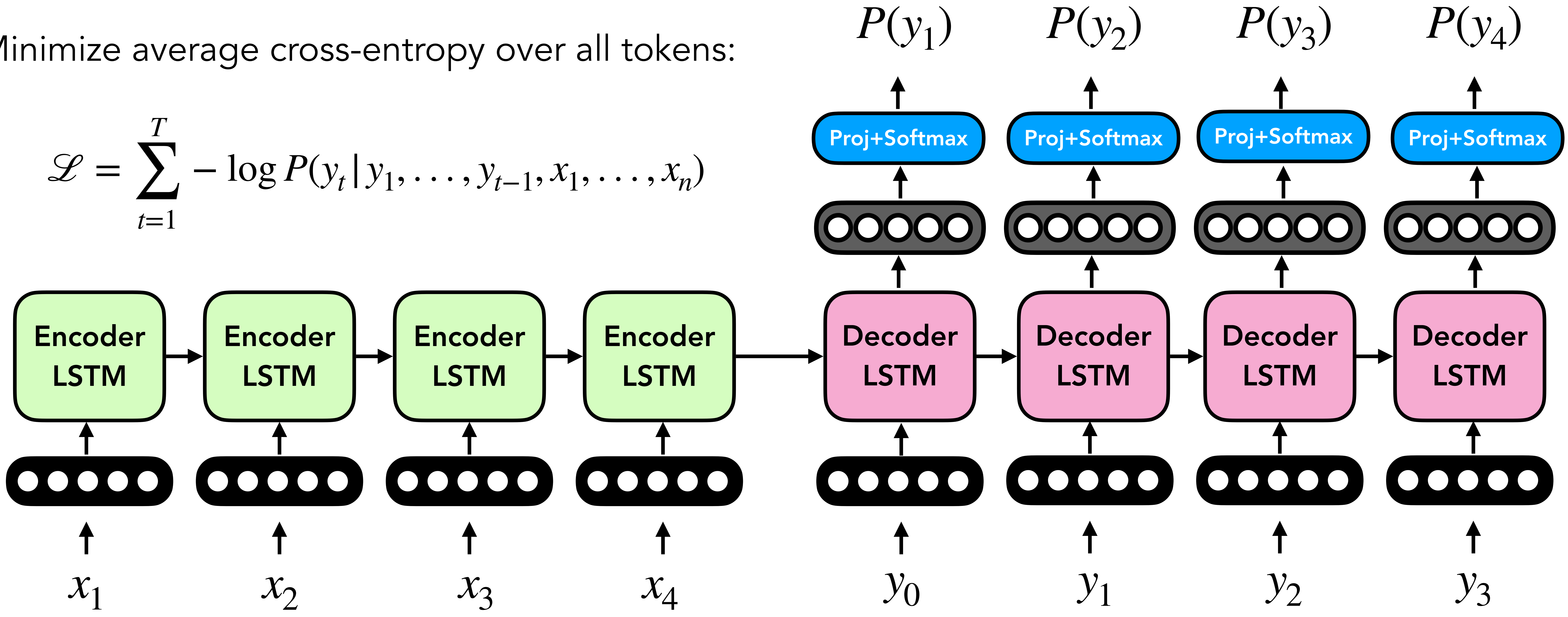
Paired data can be much more challenging to find in the wild

Training Encoder-Decoder Models

Similar to training a language model!

Minimize average cross-entropy over all tokens:

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

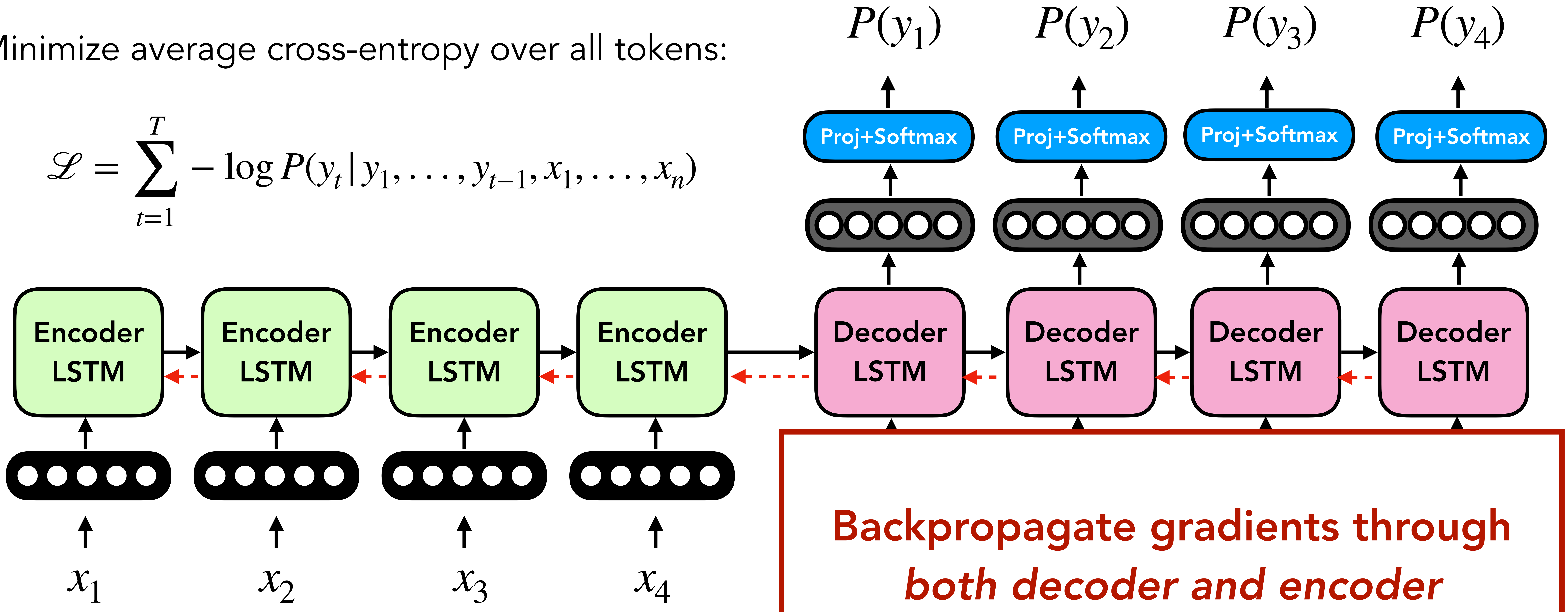


Training Encoder-Decoder Models

Similar to training a language model!

Minimize average cross-entropy over all tokens:

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



Backpropagate gradients through *both decoder and encoder*

“you can’t cram the meaning of a whole %&@#&ing sentence into a single $*(&@ing vector!”

— Ray Mooney (NLP professor at UT Austin)

Issue with Recurrent Models

- State represented as a single vector —> massive compression of information
- At every step, it must be re-computed, making it challenging to learn long-range dependencies without ignoring immediate ones

LAMBADA dataset, 2016

Issue with Recurrent Models

- State represented as a single vector —> massive compression of information
- At every step, it must be re-computed, making it challenging to learn long-range dependencies without ignoring immediate ones

They tuned, discussed for a moment, then struck up a lively ***jig****. Everyone joined in, turning the courtyard into an even more chaotic scene, people now* ***dancing*** *in circles,* ***swinging*** *and* ***spinning*** *in circles, everyone making up their own* ***dance steps****. I felt my feet tapping, my body wanting to move. Aside from writing, I 've always loved* **dancing** *.*

LAMBADA dataset, 2016

Issue with Recurrent Models

- State represented as a single vector —> massive compression of information
- At every step, it must be re-computed, making it challenging to learn long-range dependencies without ignoring immediate ones

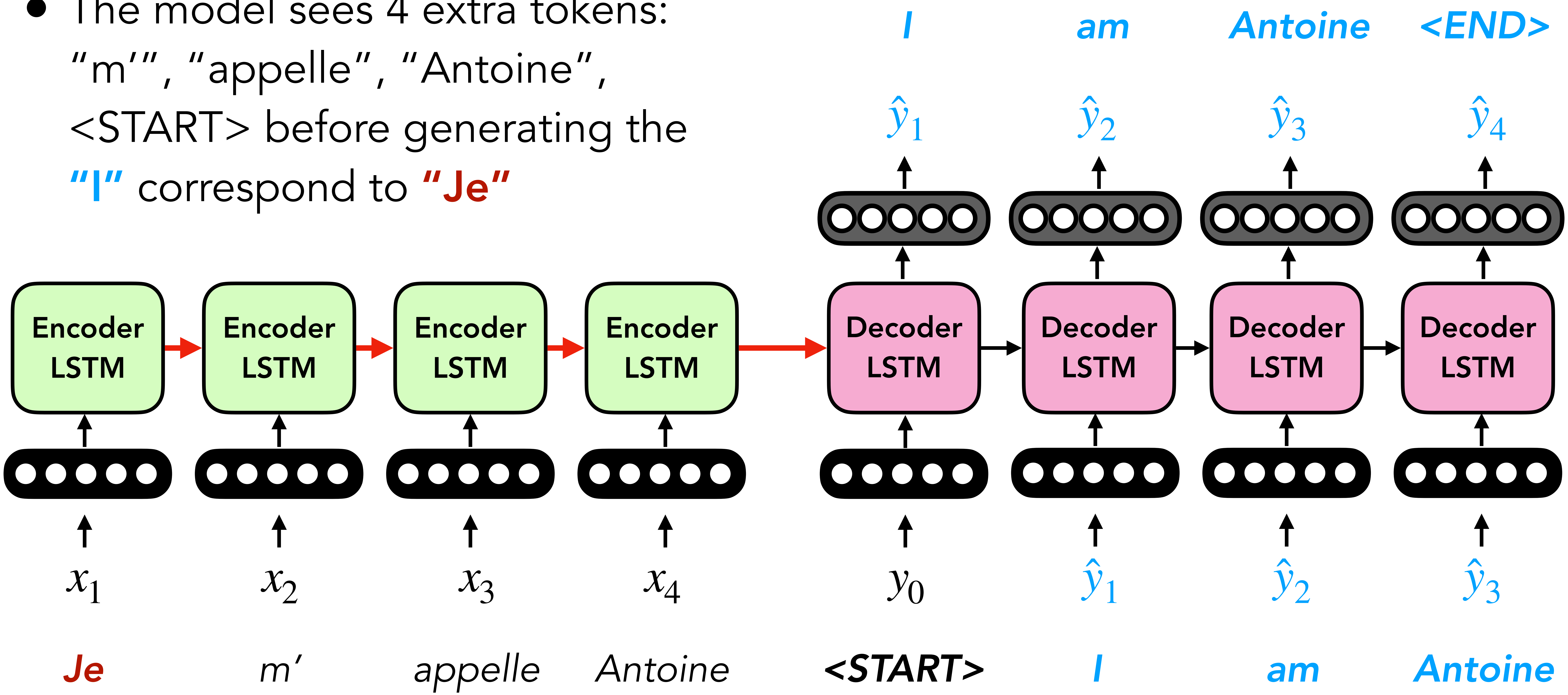
They tuned, discussed for a moment, then struck up a lively ***jig****. Everyone joined in, turning the courtyard into an even more chaotic scene, people now* ***dancing*** *in circles,* ***swinging*** *and* ***spinning*** *in circles, everyone making up their own* ***dance steps****. I felt my feet tapping, my body wanting to move.*
Aside from writing, I 've always loved **dancing** *.*

- Nearby words should affect each other more than farther ones, but RNNs make it challenging to learn **any** long-range interactions

LAMBADA dataset, 2016

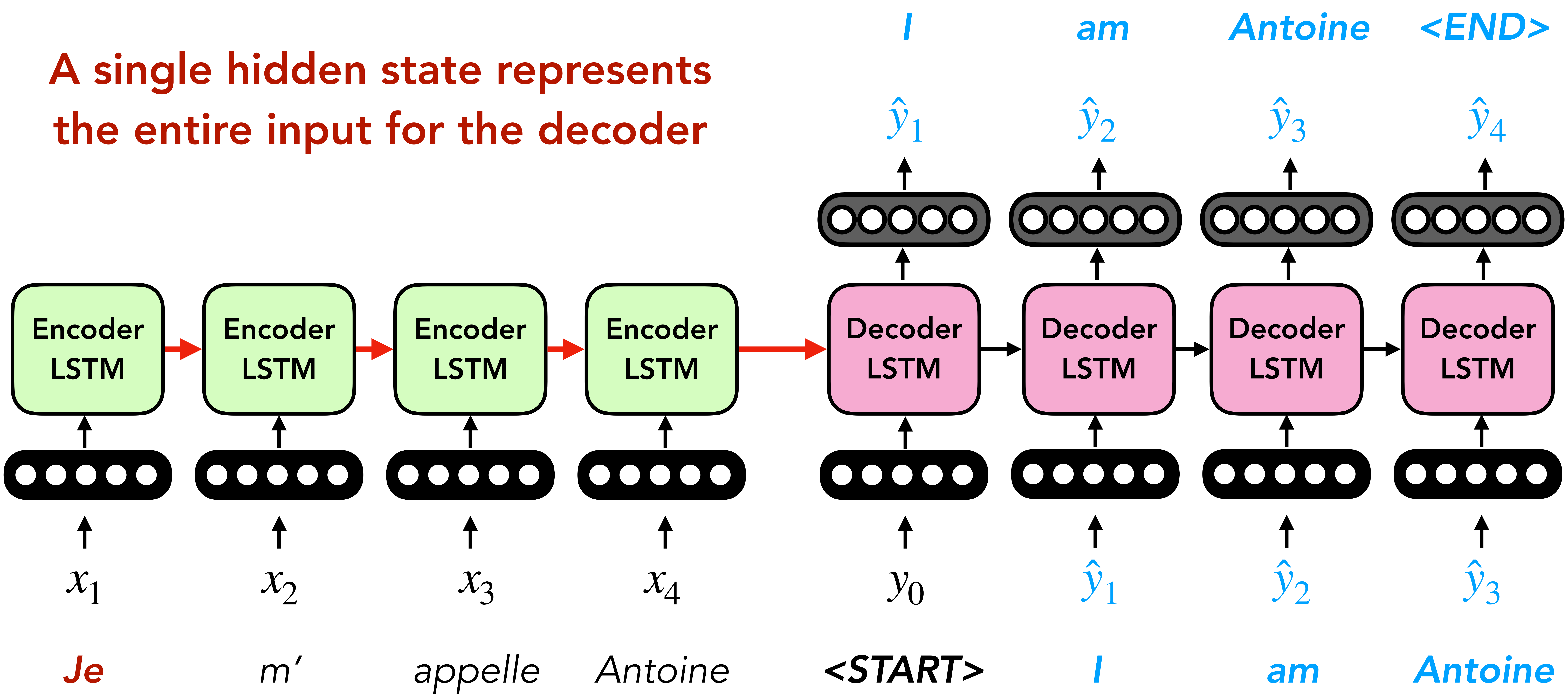
Toy Example

- The model sees 4 extra tokens: "m'", "appelle", "Antoine", <START> before generating the **"I"** correspond to **"Je"**



(Sutskever et al., 2014)

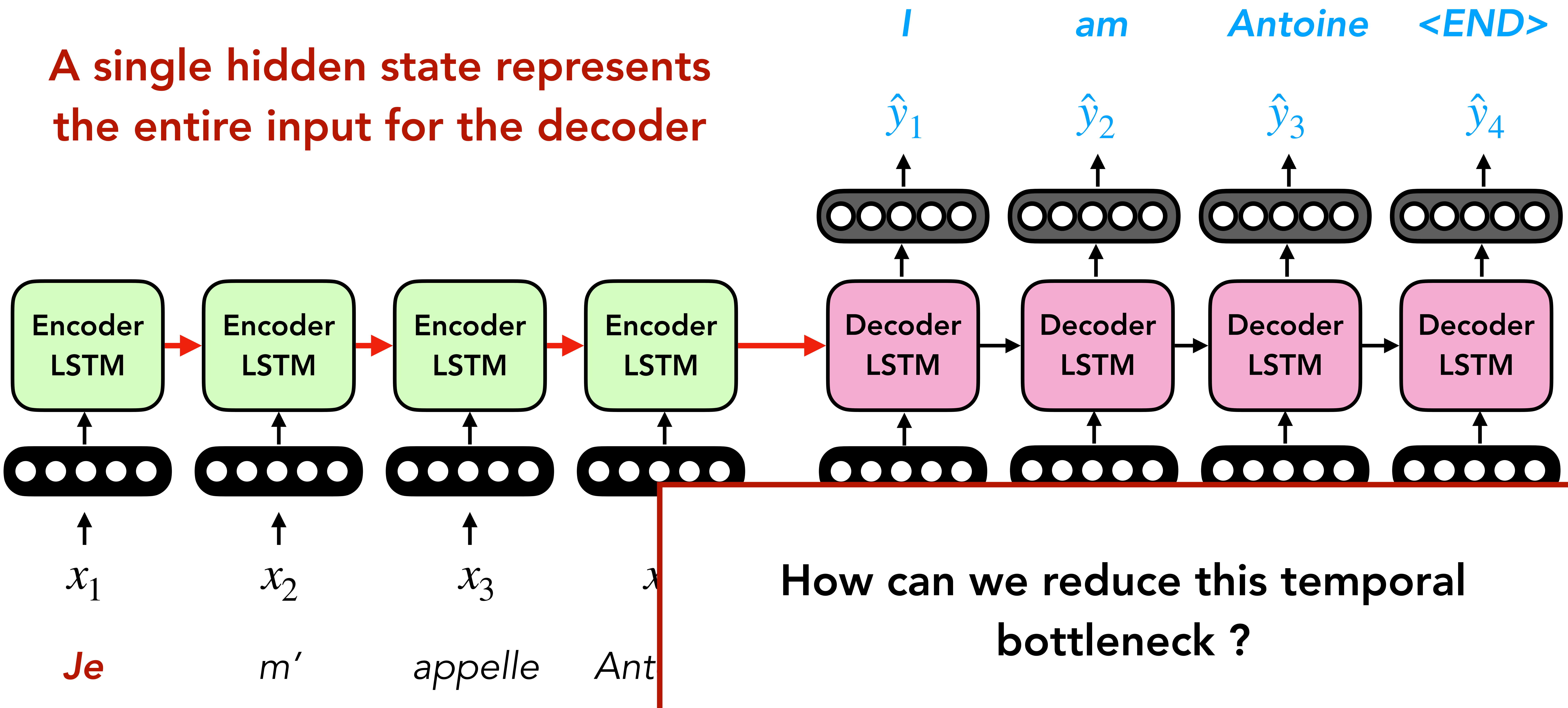
Toy Example



(Sutskever et al., 2014)

Toy Example

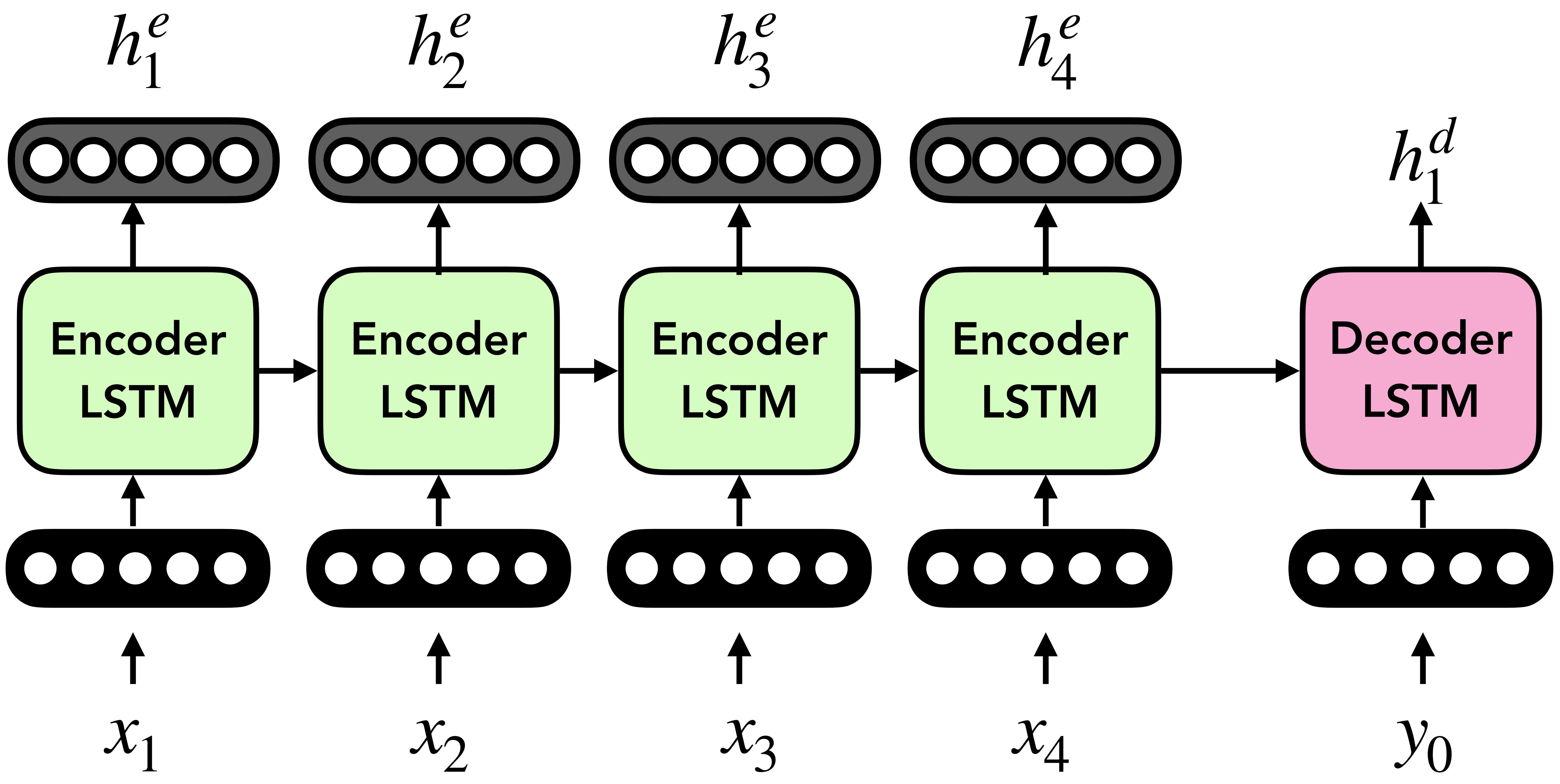
A single hidden state represents the entire input for the decoder



How can we reduce this temporal bottleneck ?

2014)

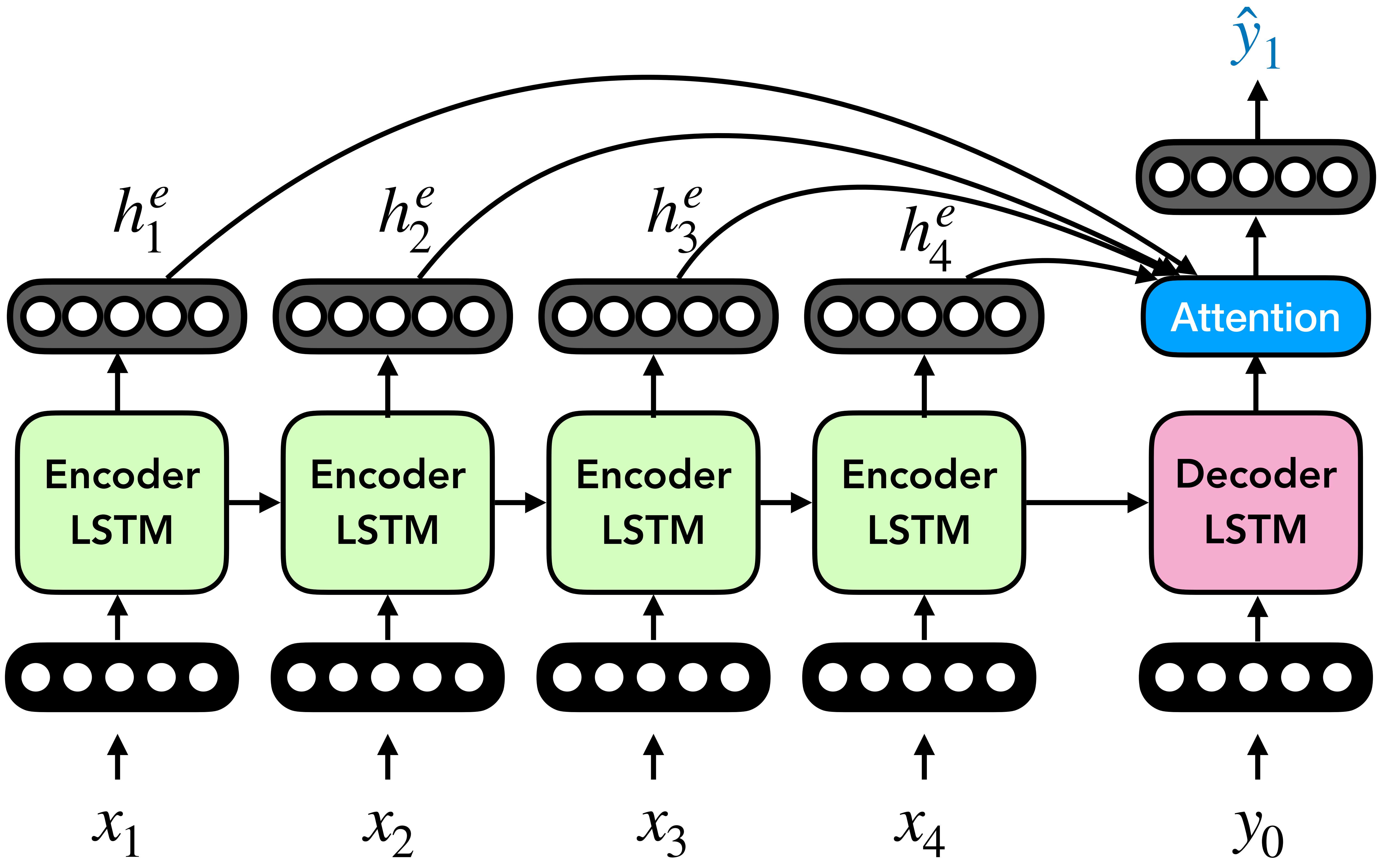
Attentive Encoder-Decoder Models



- **Recall:** At each encoder time step, there is an output of the RNN!

(Bahdanau et al., 2015)

Attentive Encoder-Decoder Models



- **Recall:** At each encoder time step, there is an output of the RNN!
- **Idea:** Use the output of the Decoder LSTM to compute an **attention** (i.e., a mixture) over all the h_t^e outputs of the encoder LSTM
- **Intuition:** focus on different parts of the input at each time step

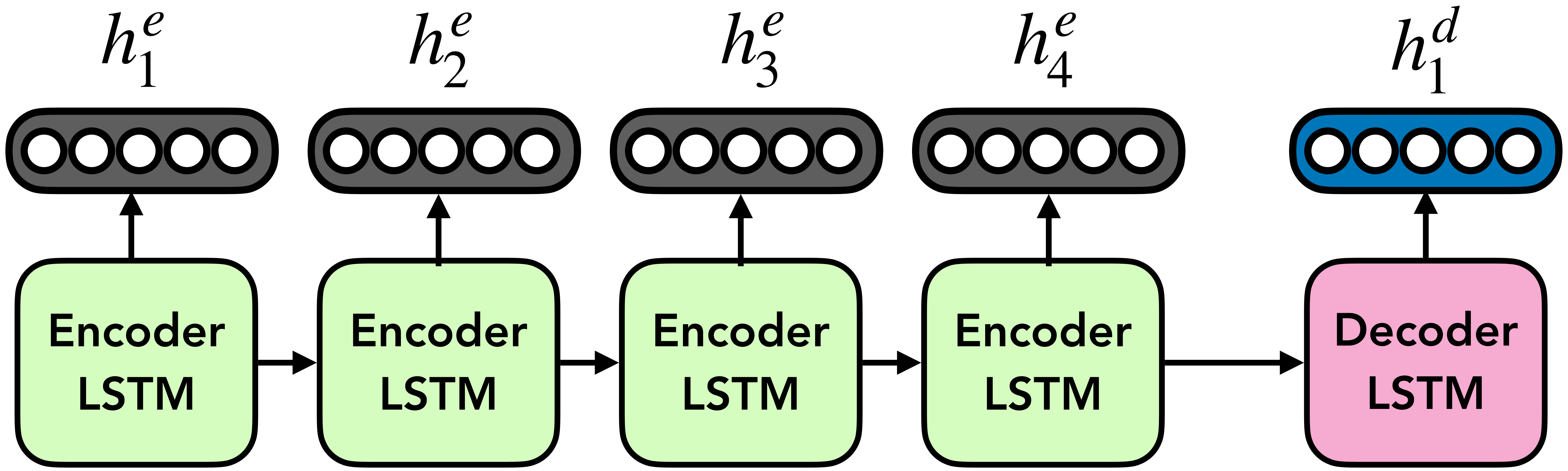
(Bahdanau et al., 2015)

What is attention?

- Attention is a **weighted average over a set of inputs**

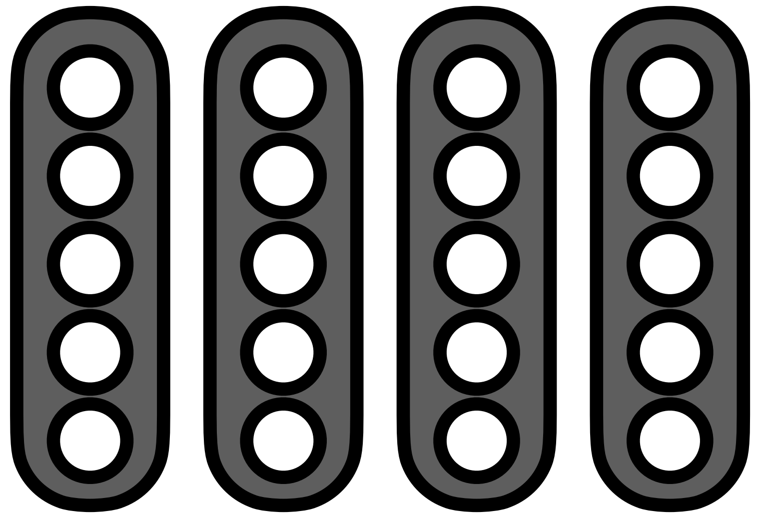
h_t^e = encoder output hidden states

- How should we compute this weighted average?



Attention Function

- **Compute** pairwise similarity between each encoder hidden state and decoder hidden state ("idea of what to decode")



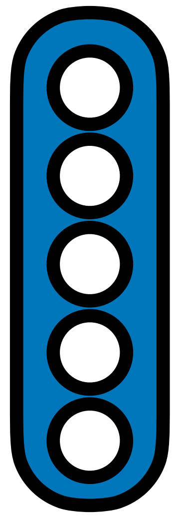
h_1^e h_2^e h_3^e h_4^e

h_t^e = encoder output hidden states

Also known as a "keys"

h_t^d = decoder output hidden state

Also known as a "query"



Attention Function

- **Compute** pairwise similarity between each encoder hidden state and decoder hidden state ("idea of what to decode")

h_t^e = encoder output hidden states

Also known as a "keys"

h_t^d = decoder output hidden state

Also known as a "query"

- **We have a single query vector for multiple key vectors**

Attention Function

Attention Function	Formula
Multiplicative	$a = h^e \mathbf{W} h^d$
Linear	$a = v^T \phi(\mathbf{W}[h^e; h^d])$
Scaled Dot Product	$a = \frac{(\mathbf{W}h^e)^T(\mathbf{U}h^d)}{\sqrt{d}}$

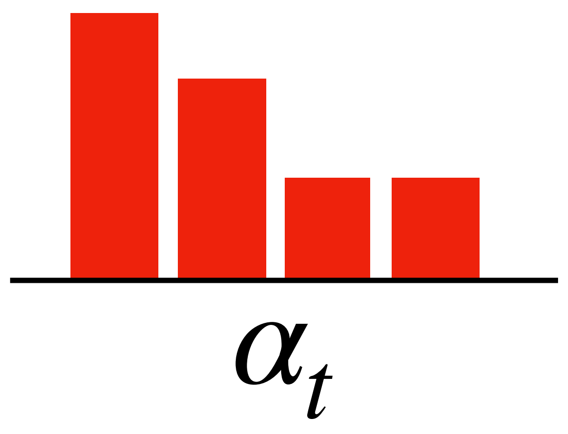
Attention Function

- **Compute** pairwise similarity between each encoder hidden state and decoder hidden state ("idea of what to decode")

$$a_1 = f(h_1^e, h_1^d) \qquad a_2 = f(h_2^e, h_1^d) \qquad a_3 = f(h_3^e, h_1^d)$$

- **Convert** pairwise similarity scores to probability **distribution** (using softmax!) over encoder hidden states and compute weighted average:

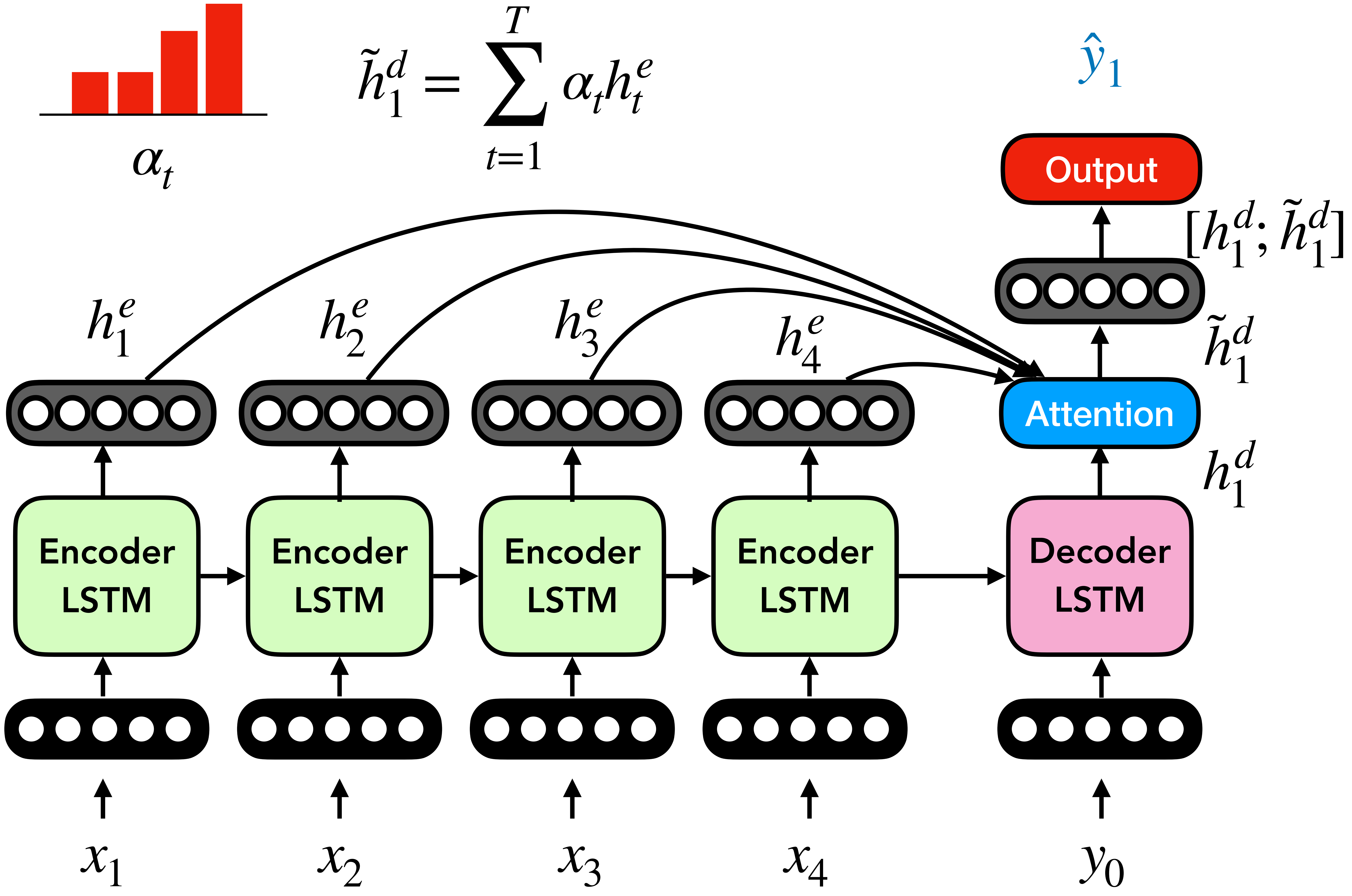
Softmax! $\boxed{\alpha_t = \dfrac{e^{a_t}}{\sum_j e^{a_j}}}$ $\rightarrow$



$\rightarrow$ $\tilde{h}_1^d = \sum_{t=1}^{T} \alpha_t h_t^e$

Here h_t^e is known as the "value"

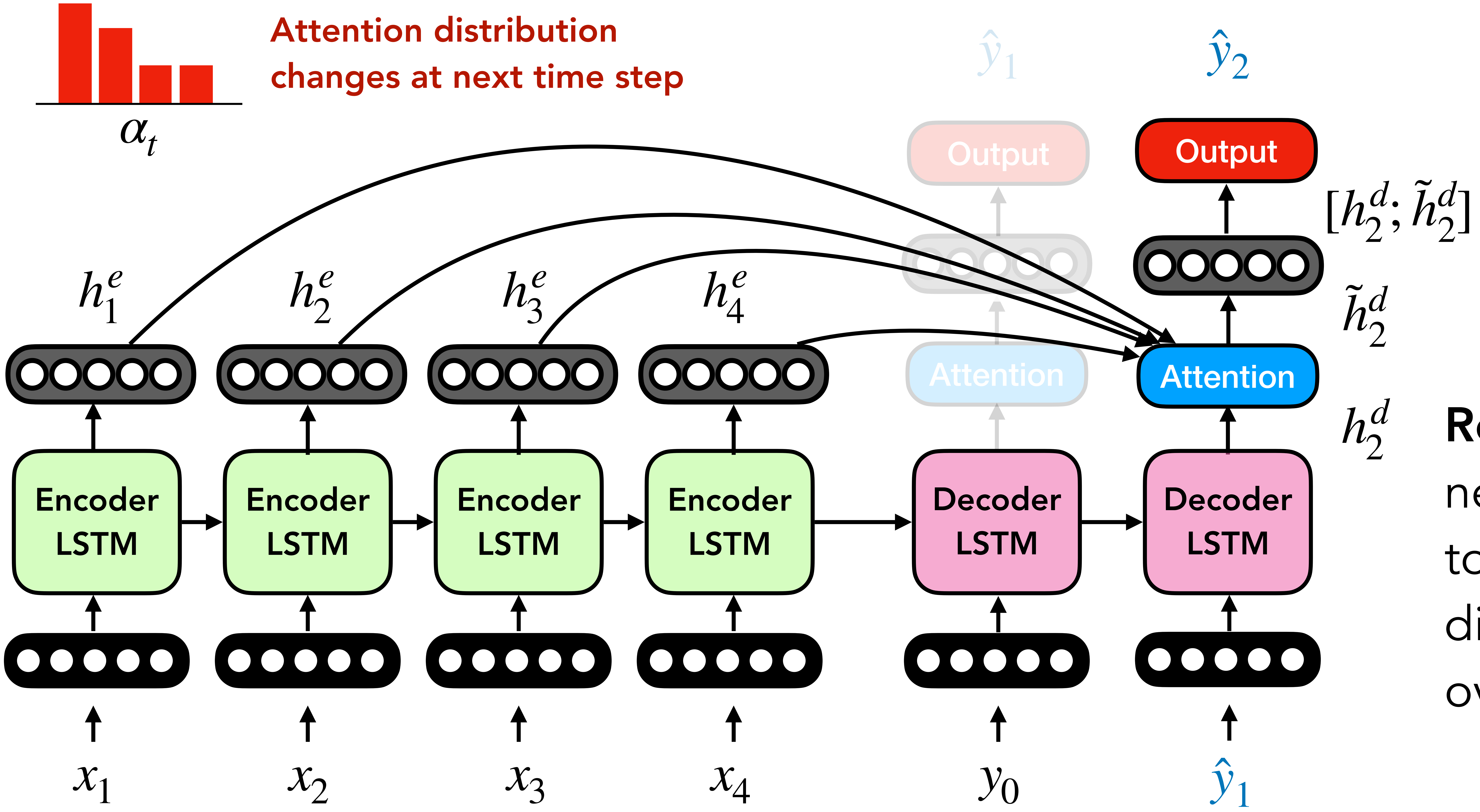
Attentive Encoder-Decoder Models



- **Intuition:** $\tilde{h}^d_1$ contains information about hidden states that got **high** attention
- Typically, $\tilde{h}^d_1$ is concatenated (or composed in some other manner) with h^d_1 (the original decoder state) before being passed to the **output** layer
- **Output** layer predicts the most likely output token $\hat{y}_1$

(Bahdanau et al., 2015)

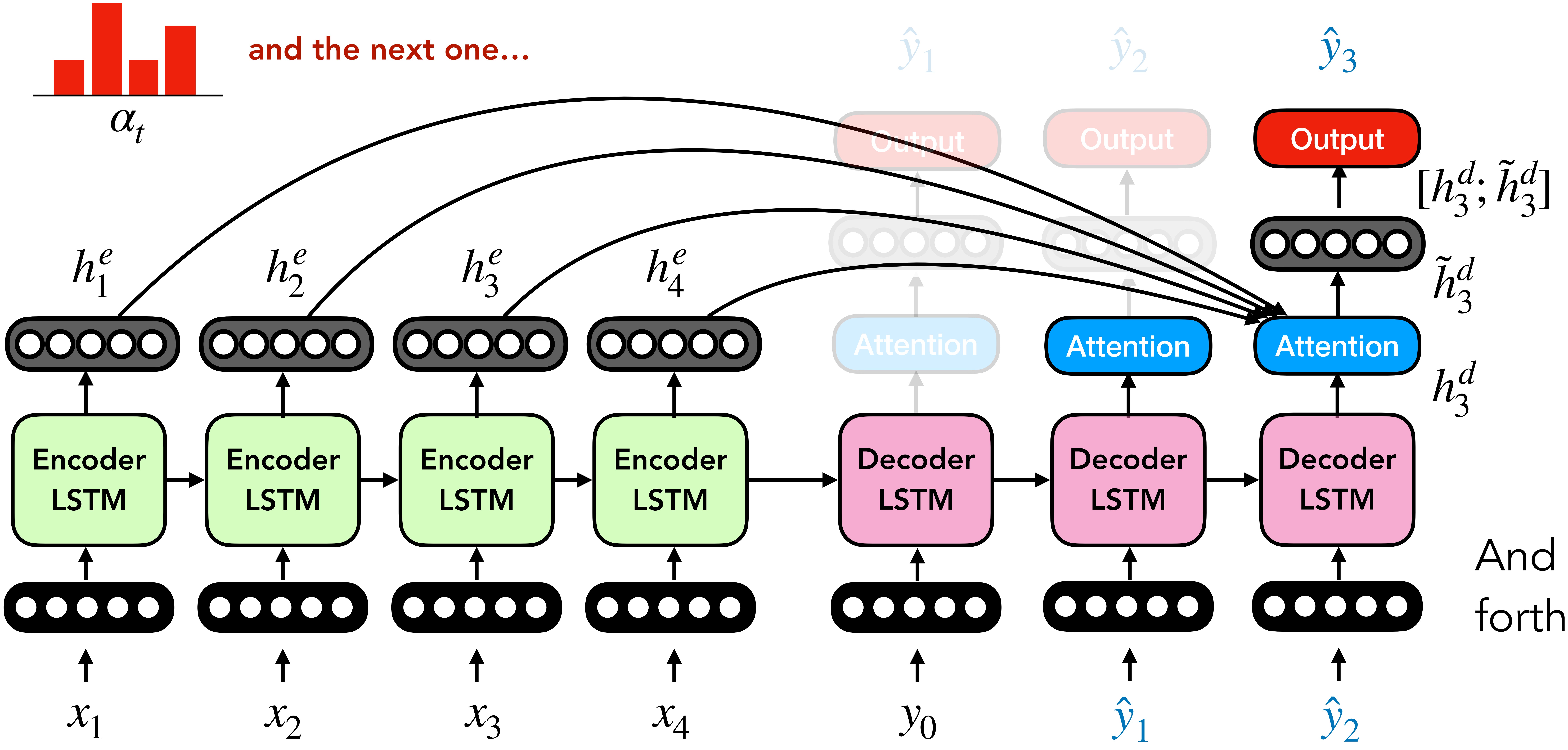
Attentive Encoder-Decoder Models



Repeat in next time step to get new distribution over states

(Bahdanau et al., 2015)

Attentive Encoder-Decoder Models



And so forth…

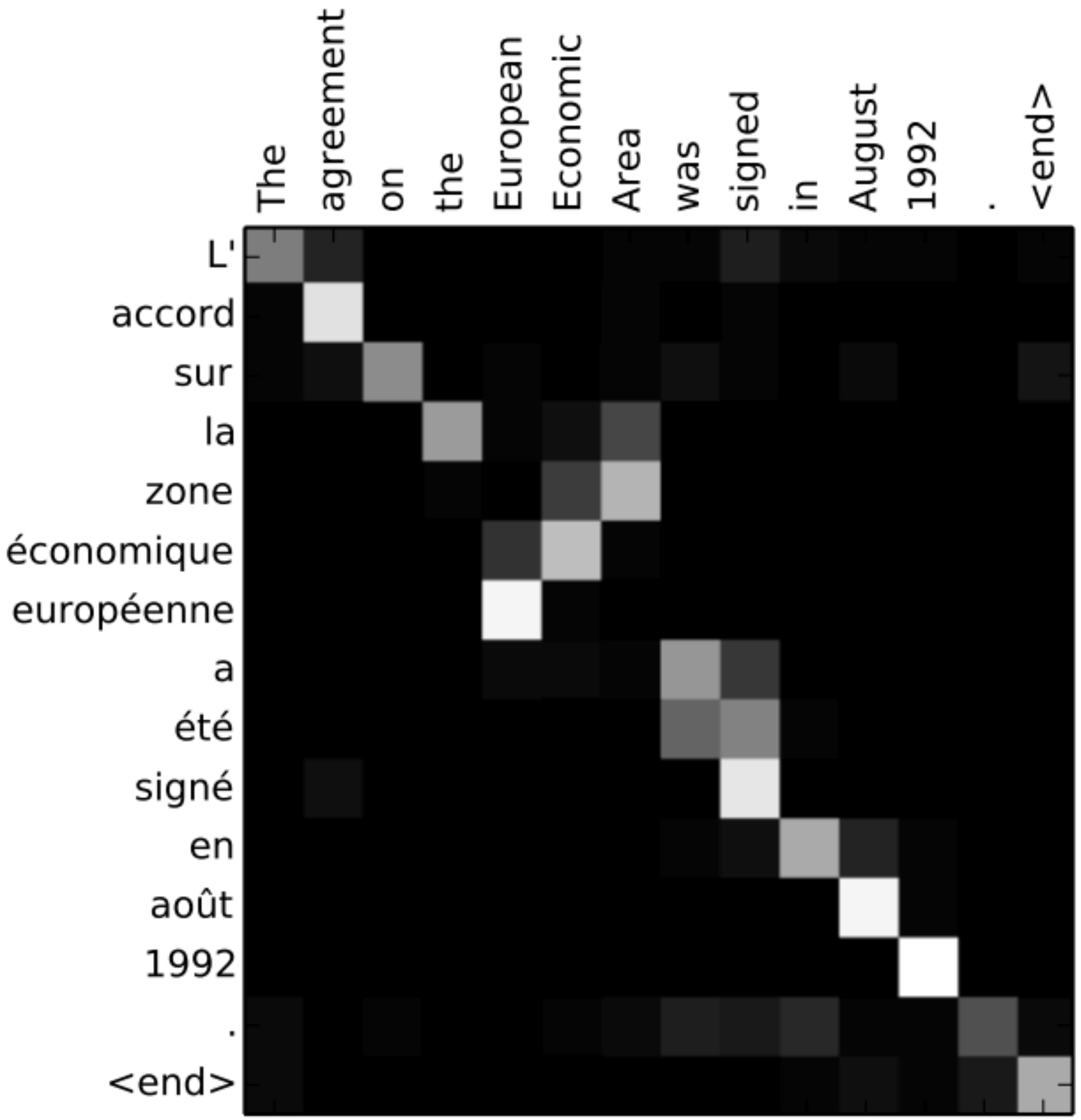
(Bahdanau et al., 2015)

Interpretability?

- **Main Idea:** Attention can be visualised based on the score given to each encoder hidden state

- What is focused on when each word is generated?

- Training with attention gives us implicit alignment for free!



Question

How does attention address the temporal bottleneck in sequence to sequence models?

Attention Recap

- **Main Idea:** Decoder computes a weighted sum of encoder outputs
  - Compute pairwise score between each encoder hidden state and initial decoder hidden state ("idea of what to decode")
- Many possible functions for computing scores (dot product, bilinear, etc.)
- **Temporal Bottleneck Fixed! Direct link** between decoder and encoder states
  - Helps with vanishing gradients!
- **Interpretability** allows us to investigate model behavior!
- Attention is **agnostic** to the type of RNN used in the encoder and decoder!

Question

In what range can an attention value fall ?

[0, 1]

Looking Forward

- **Tomorrow**: Guest Lecture by Gail Weiss
  - *"Theoretical properties of RNNs"*
- **Next week:** More attention, transformers, GPT
- **Exercise Session:** Sequence-to-sequence models; Attention

References

- Sutskever, I., Vinyals, O., & Le, Q.V. (2014). Sequence to Sequence Learning with Neural Networks. *NIPS*.
- Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2014). Show and tell: A neural image caption generator. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3156-3164.
- Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q.N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., & Fernández, R. (2016). The LAMBADA dataset: Word prediction requiring a broad discourse context. *ArXiv, abs/1606.06031*.
- Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR, abs/1409.0473*.