

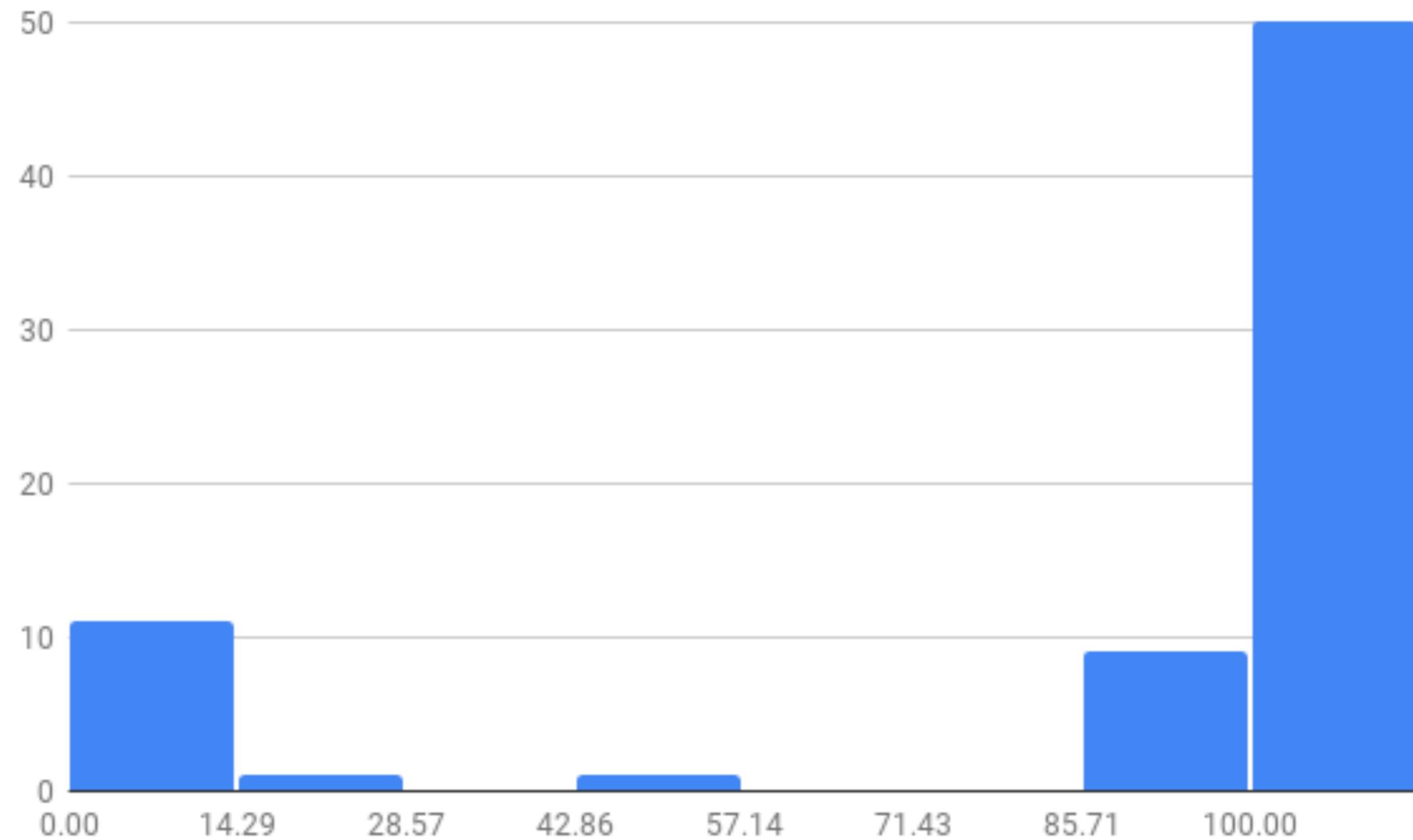


**SI 630**

# Natural Language Processing: Algorithms and People

Lecture 15: Machine Translation  
Feb. 7, 2018

# Nice work on Homework 2!



# Overview of Today

- Let's get reminded of Old School™ neural networks (ca. 2014)
- Out with the old and in with the new neural network overlords
- Parlez-vous Français?



(Reminder)

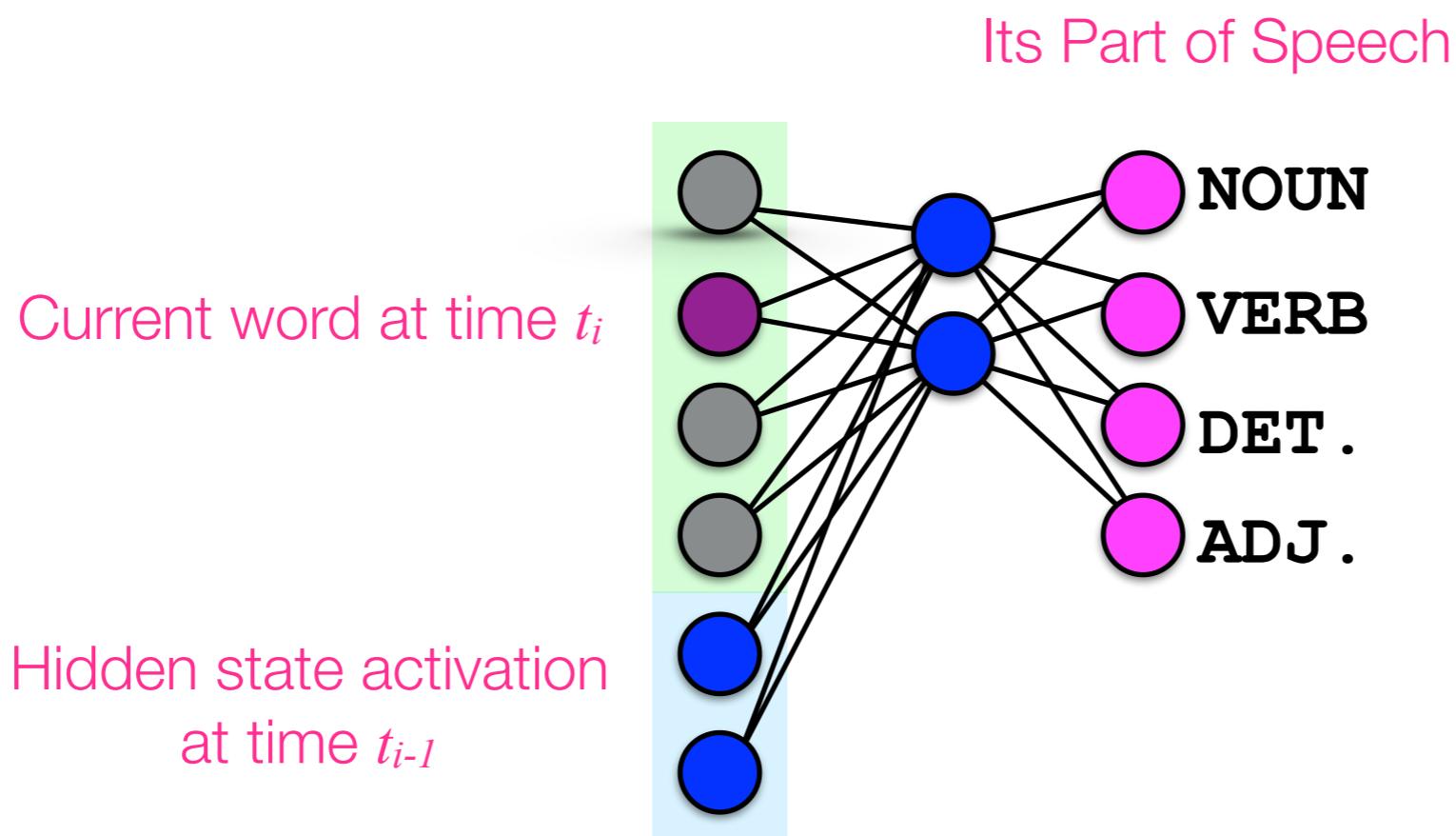
Recurrent Neural Nets

# Recurrent neural networks

- RNN allow arbitrarily-sized conditioning contexts; condition on the **entire sequence history**.

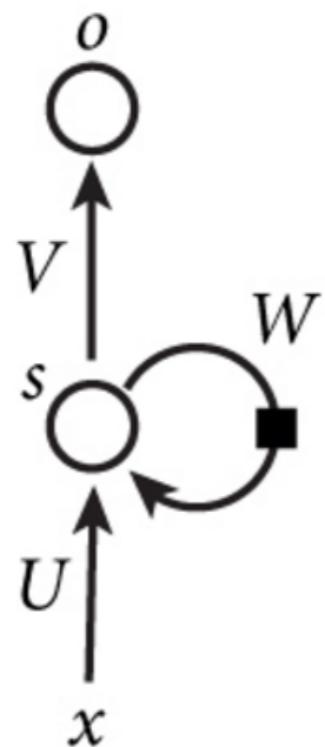
# Building a recurrent neural network using the hidden state to capture the previous context

“The play took two hours”



The input consists of the **current word** in the sequence and the **previous word's hidden state activation** which gives the network the **ability to remember** what came before

# RNN Diagram



$x_t$ : input at step t

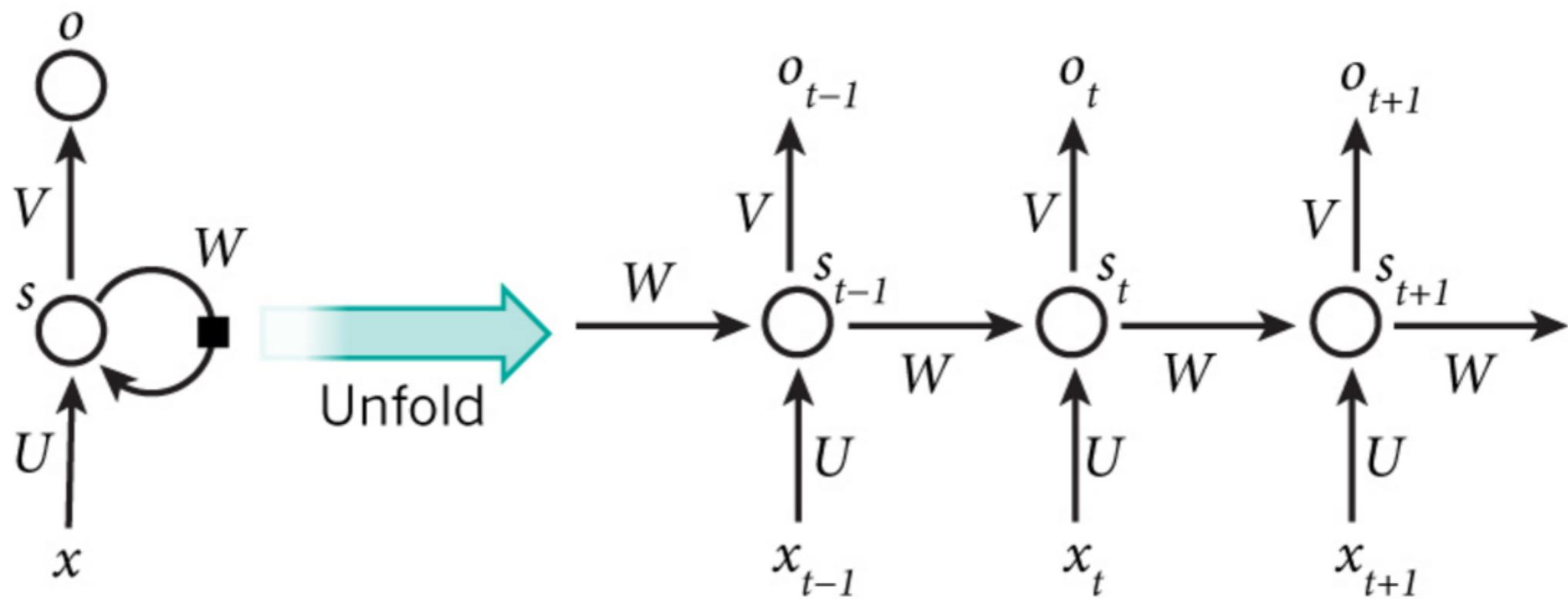
$o_t$ : output at step t

$s_t$ : hidden state at step t

$$s_t = \tanh(Ux_t + Ws_{t-1})$$

$$\hat{y}_t = \text{softmax}(Vs_t)$$

# RNN Diagram



$x_t$ : input at step t

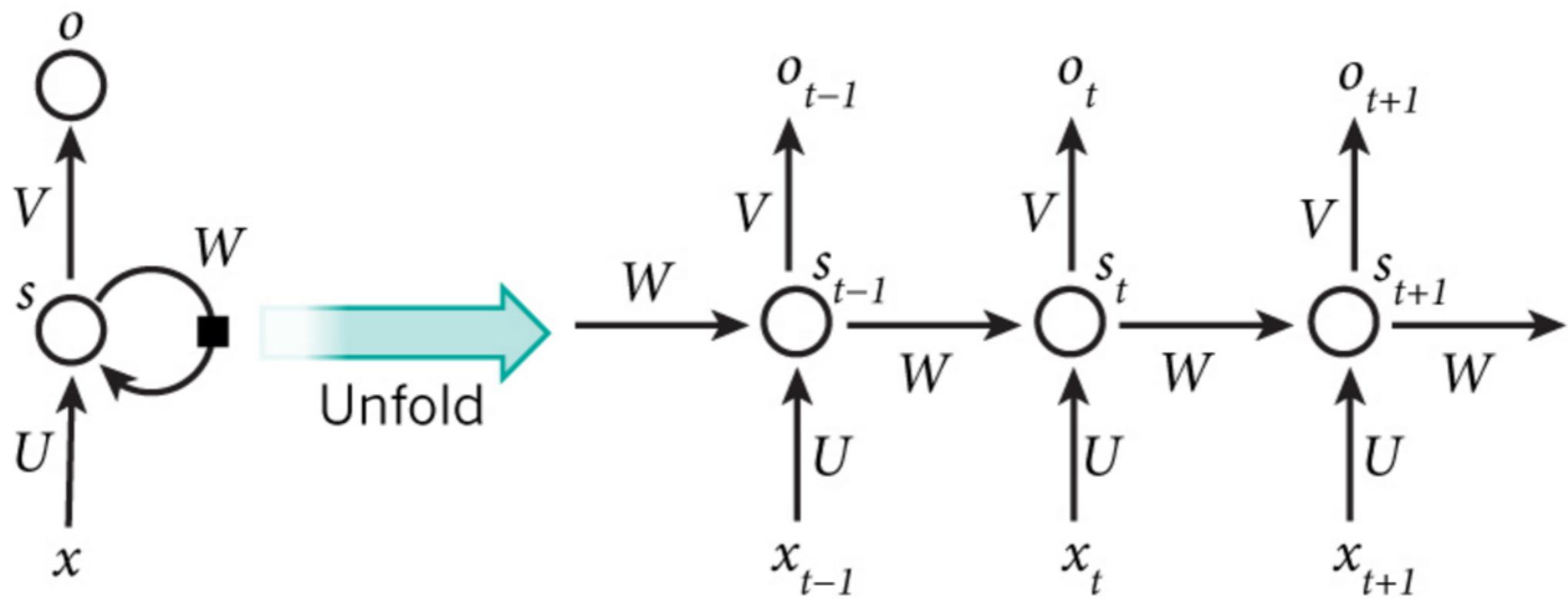
$o_t$ : output at step t

$s_t$ : hidden state at step t

$$s_t = \tanh(Ux_t + Ws_{t-1})$$

$$\hat{y}_t = \text{softmax}(Vs_t)$$

# RNN Diagram



$x_t$ : input at step t

$o_t$ : output at step t

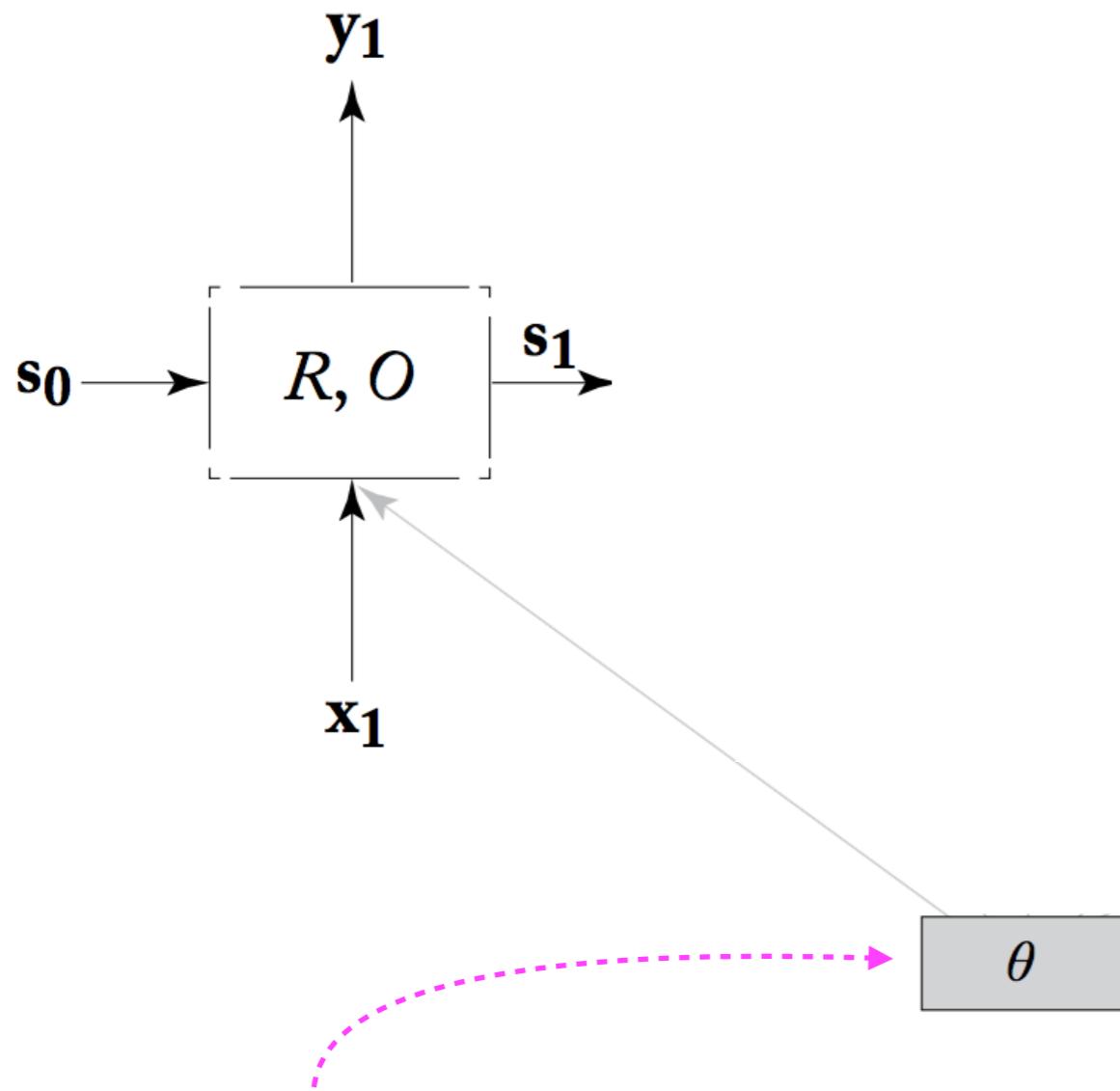
$s_t$ : hidden state at step t ← RNN's Memory

$$s_t = \tanh(Ux_t + Ws_{t-1})$$

$$\hat{y}_t = \text{softmax}(Vs_t)$$

# Recurrent neural network

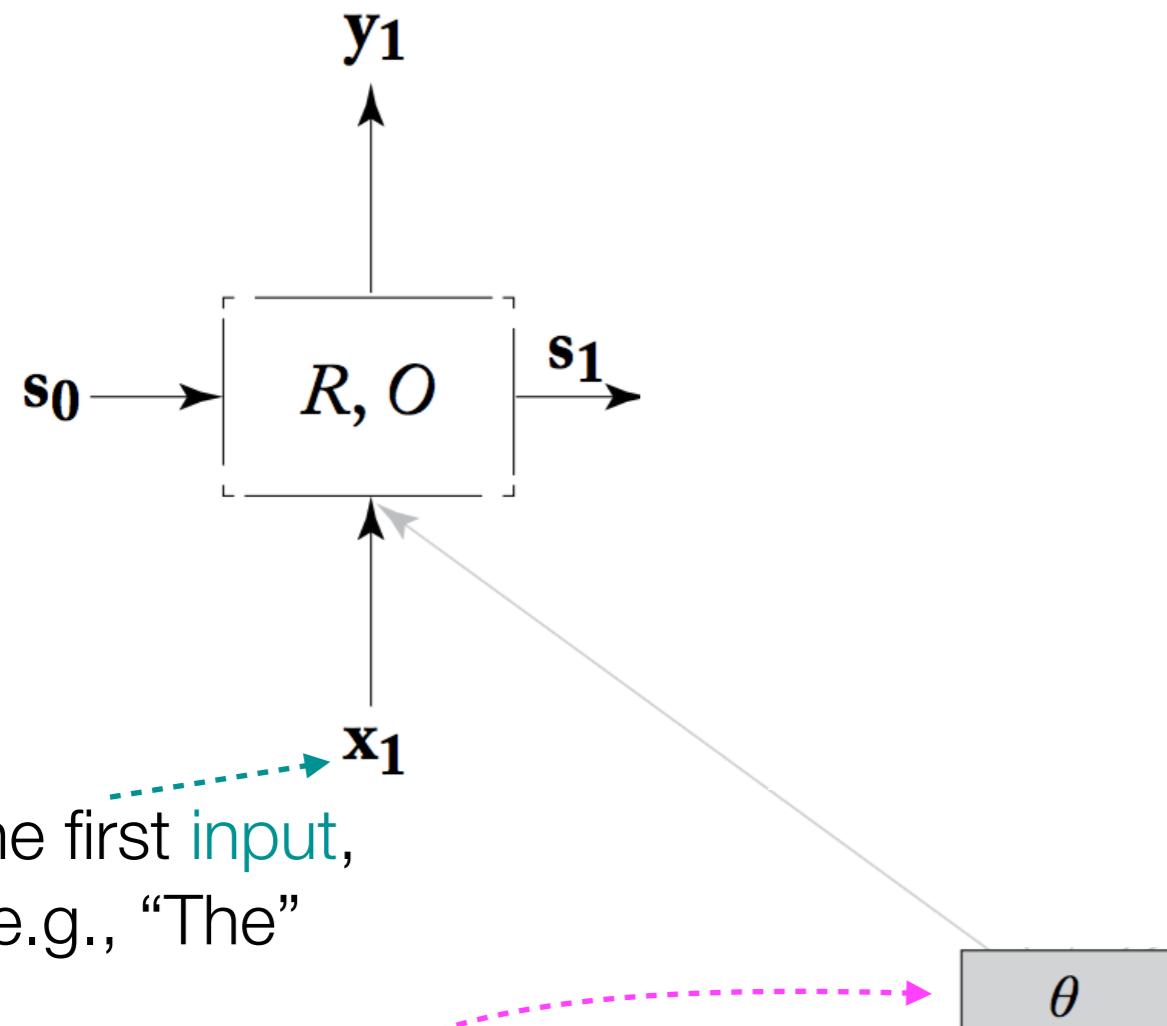
# Recurrent neural network



The **parameters** of the model

Goldberg 2017

# Recurrent neural network

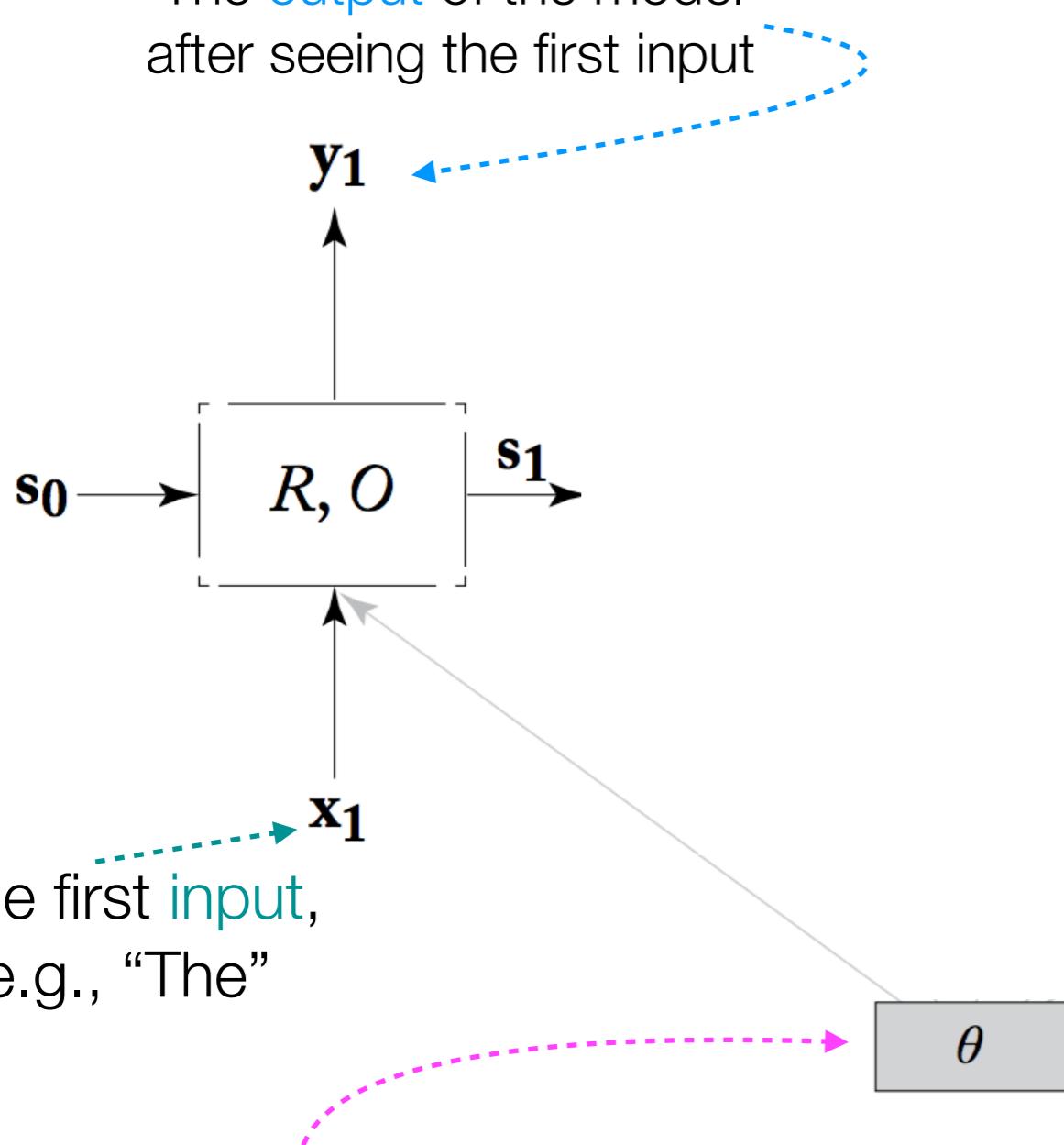


The first input,  
e.g., "The"

The parameters of the model

# Recurrent neural network

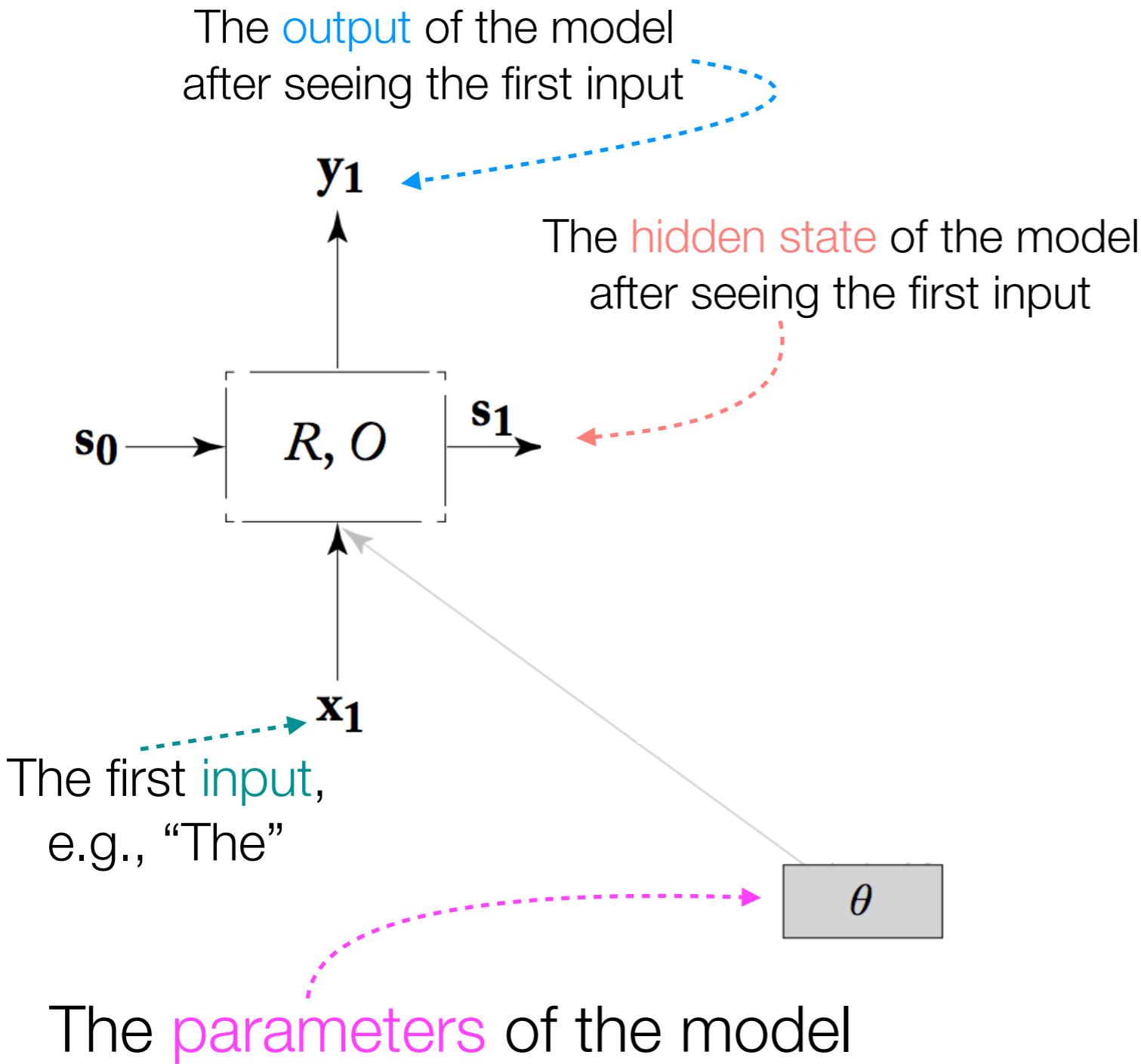
The **output** of the model  
after seeing the first input



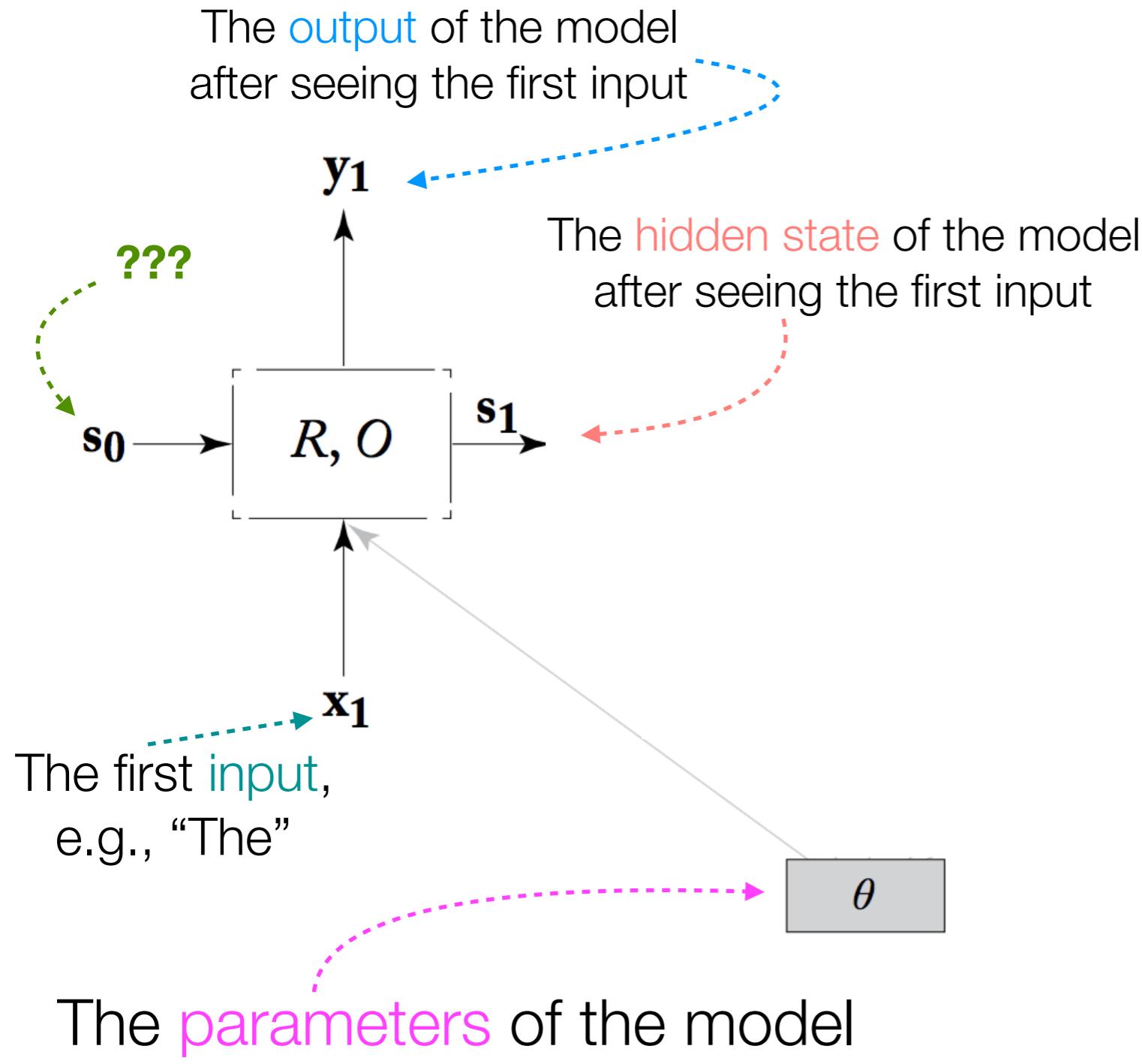
The **first input**,  
e.g., "The"

The **parameters** of the model

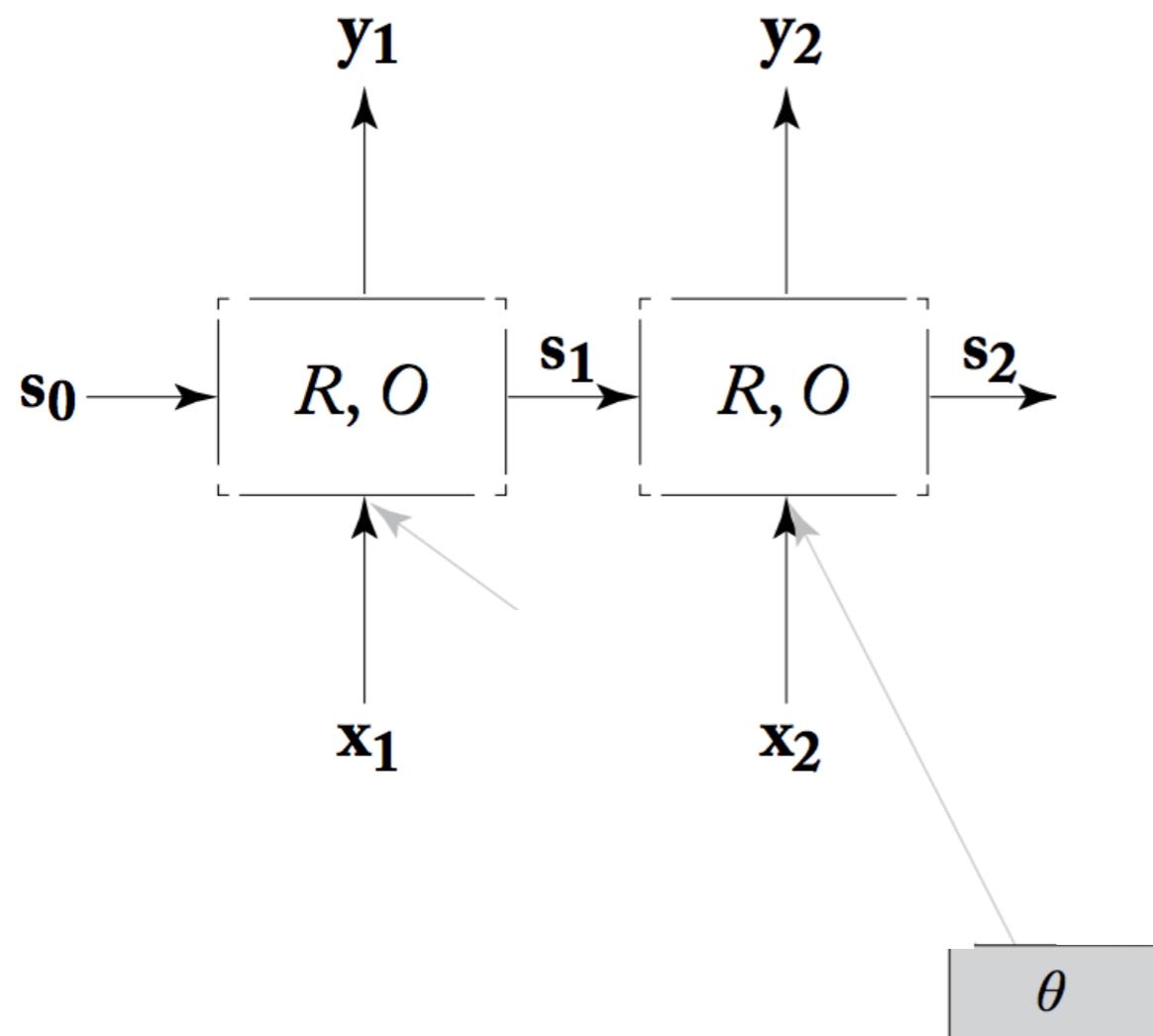
# Recurrent neural network



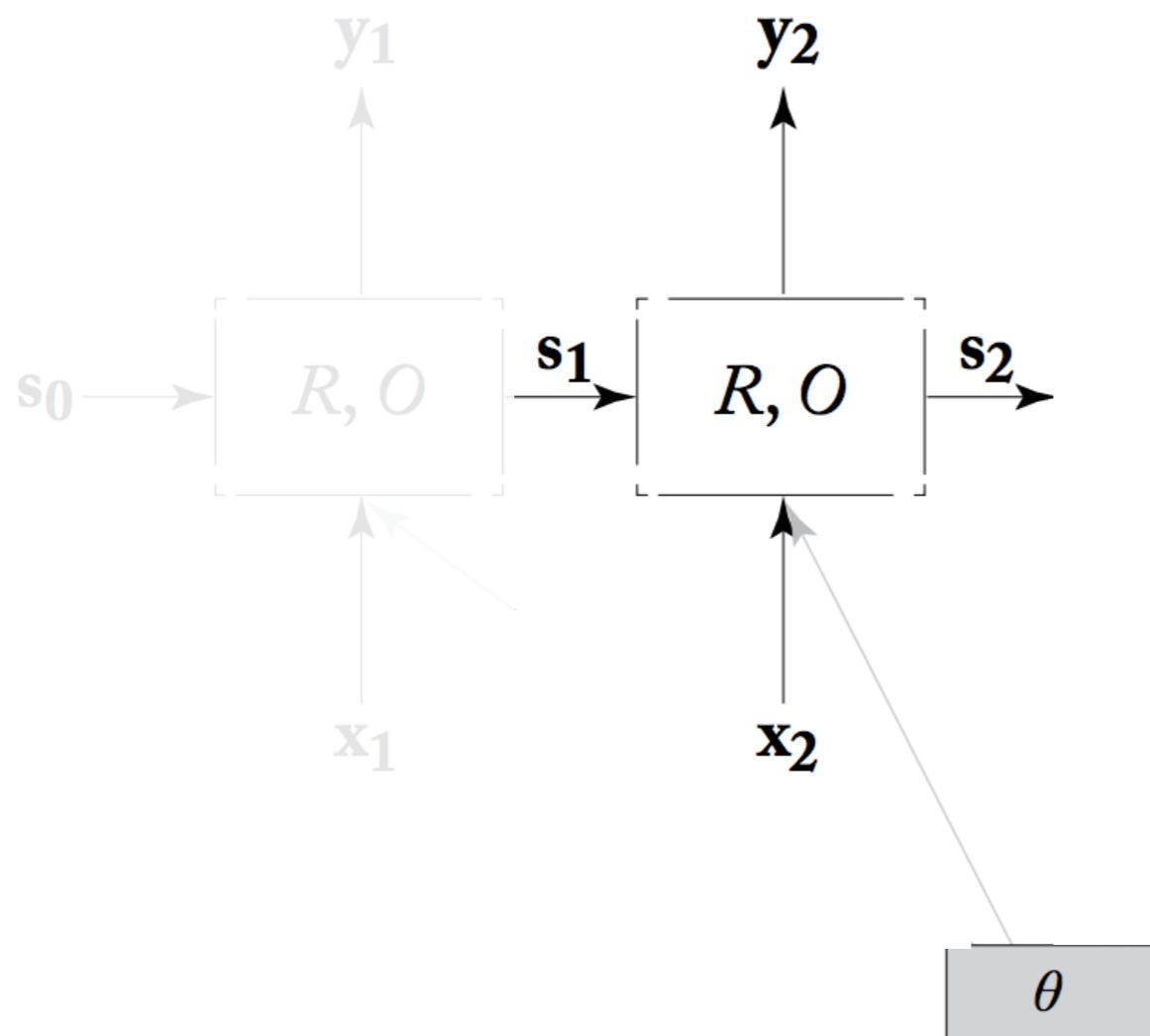
# Recurrent neural network



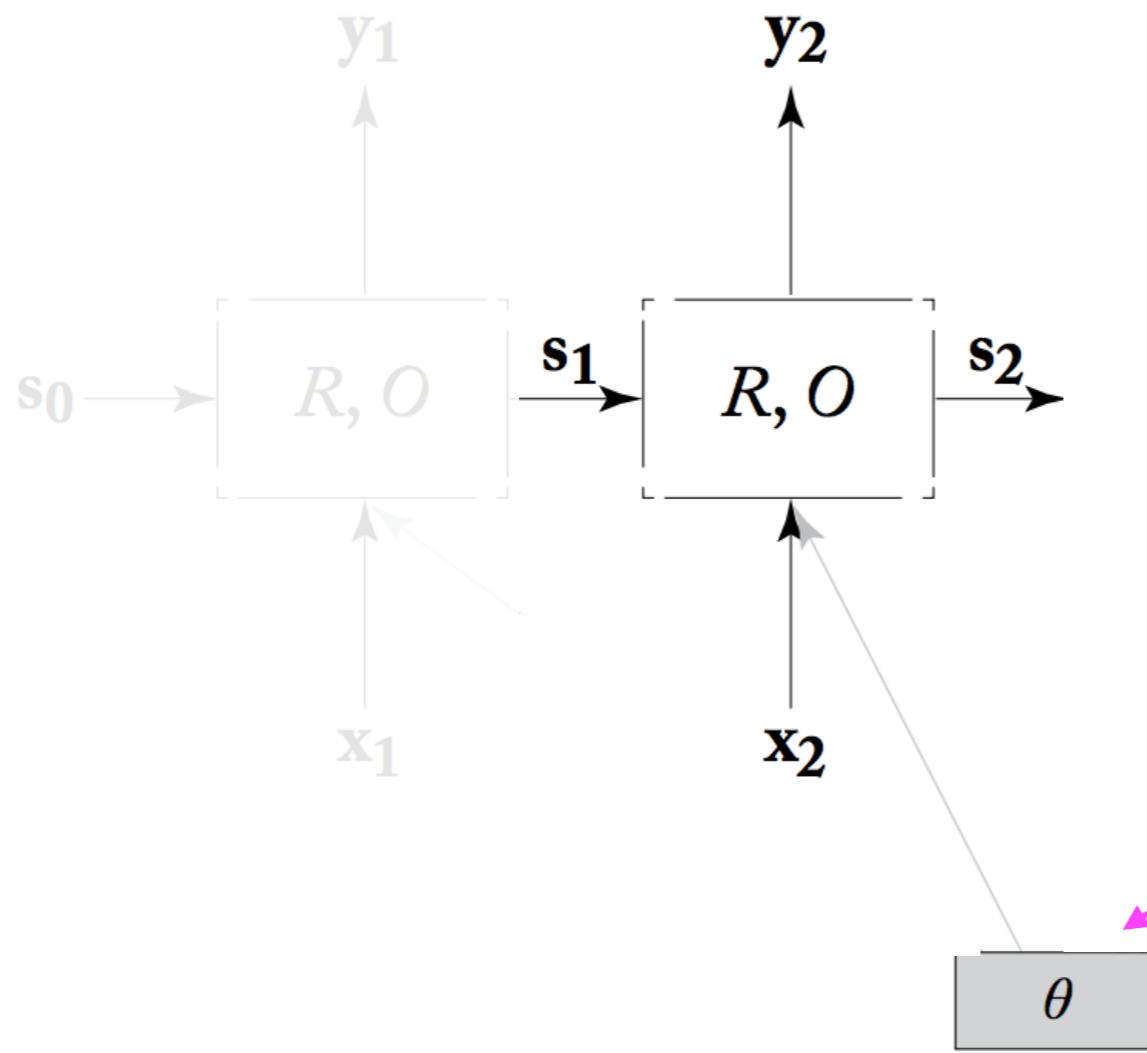
# Recurrent neural network



# Recurrent neural network

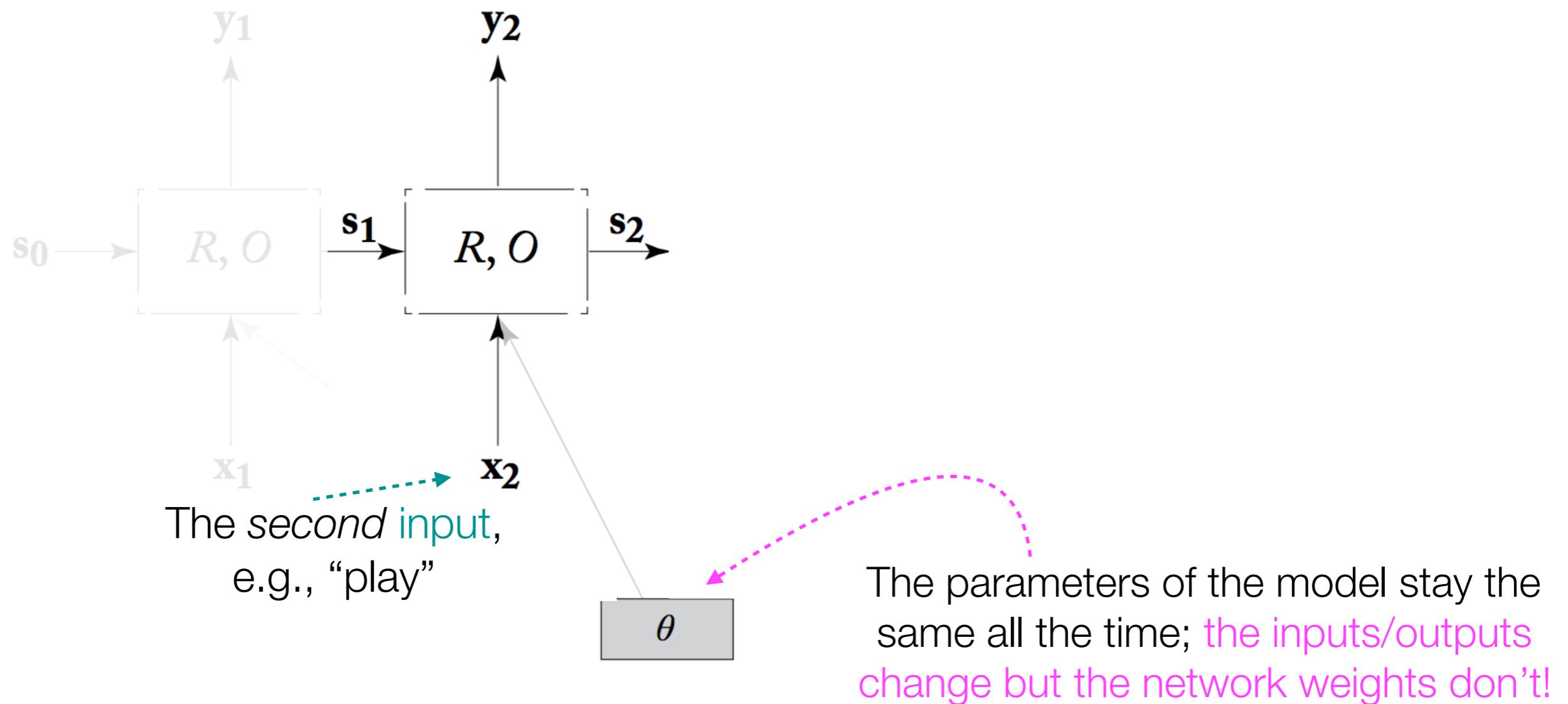


# Recurrent neural network

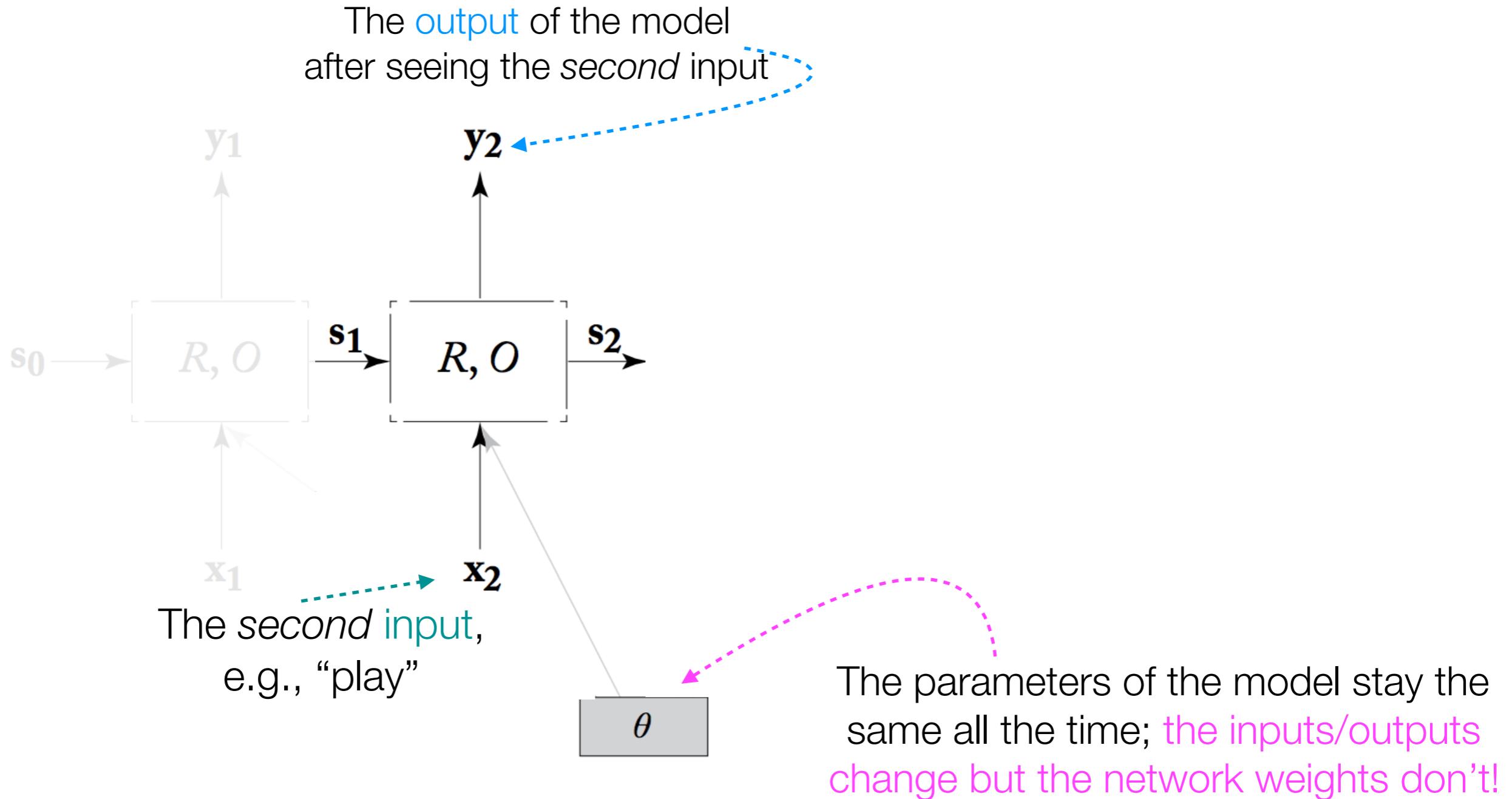


The parameters of the model stay the same all the time; the inputs/outputs change but the network weights don't!

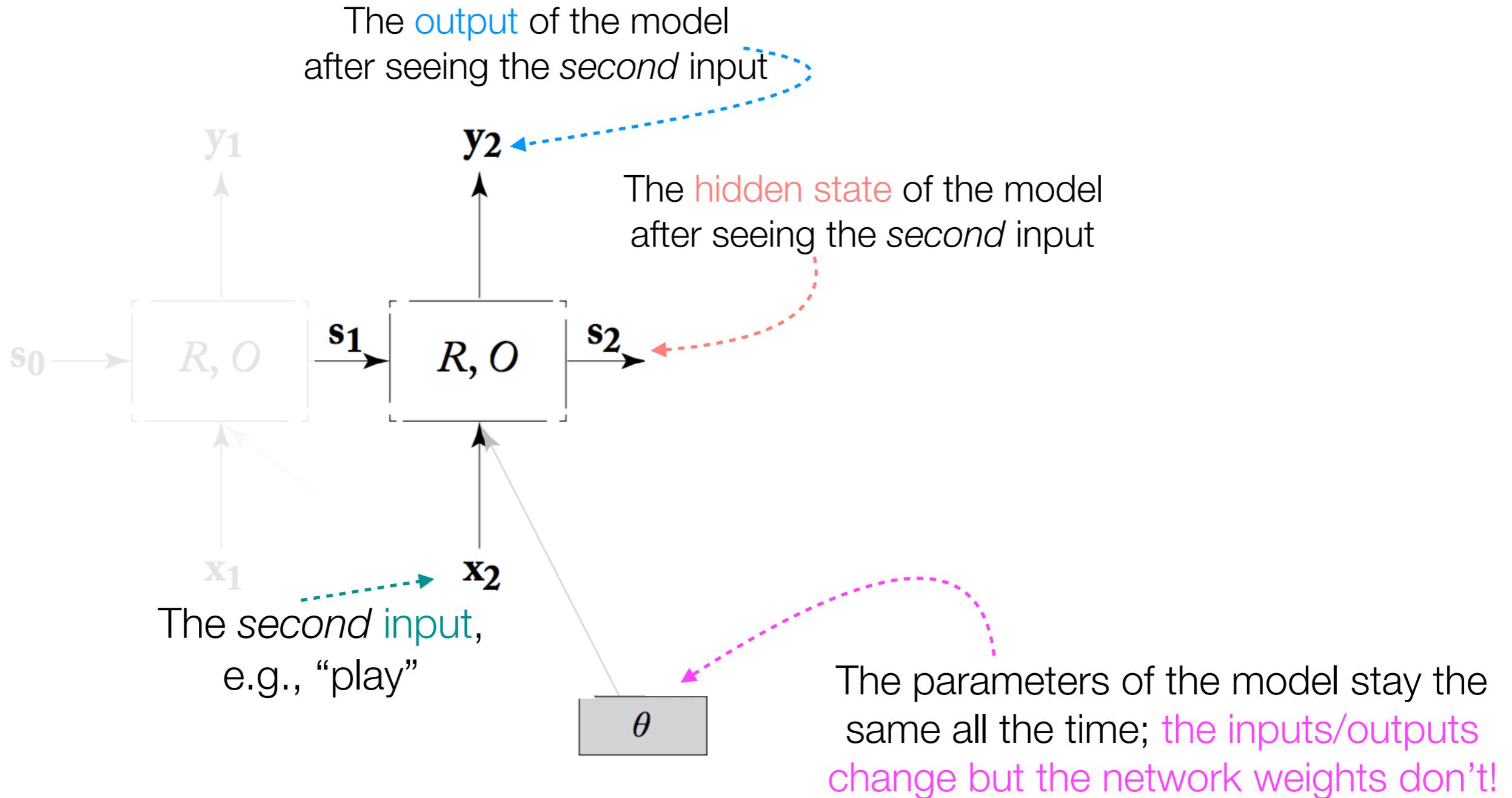
# Recurrent neural network



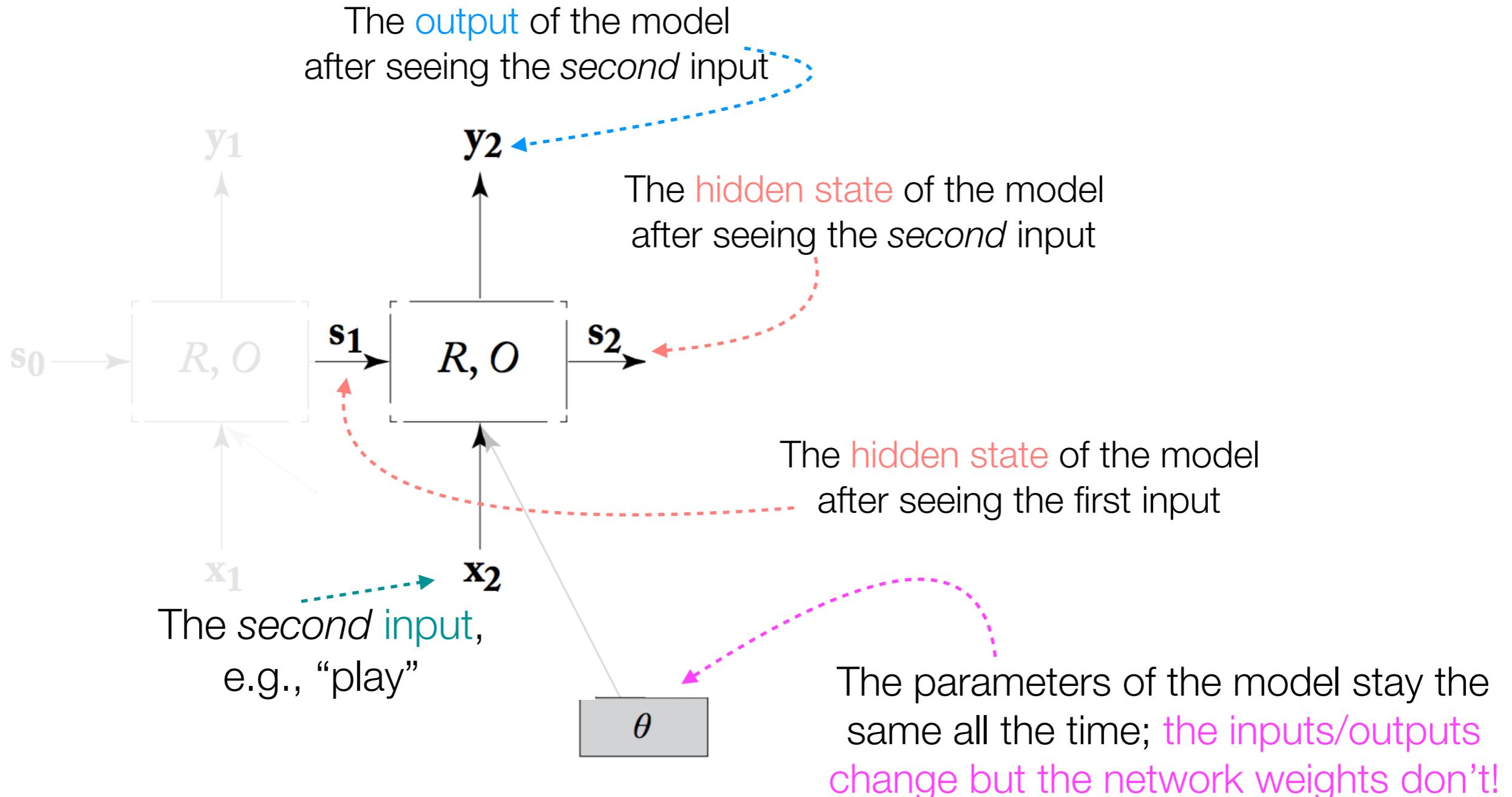
# Recurrent neural network



# Recurrent neural network

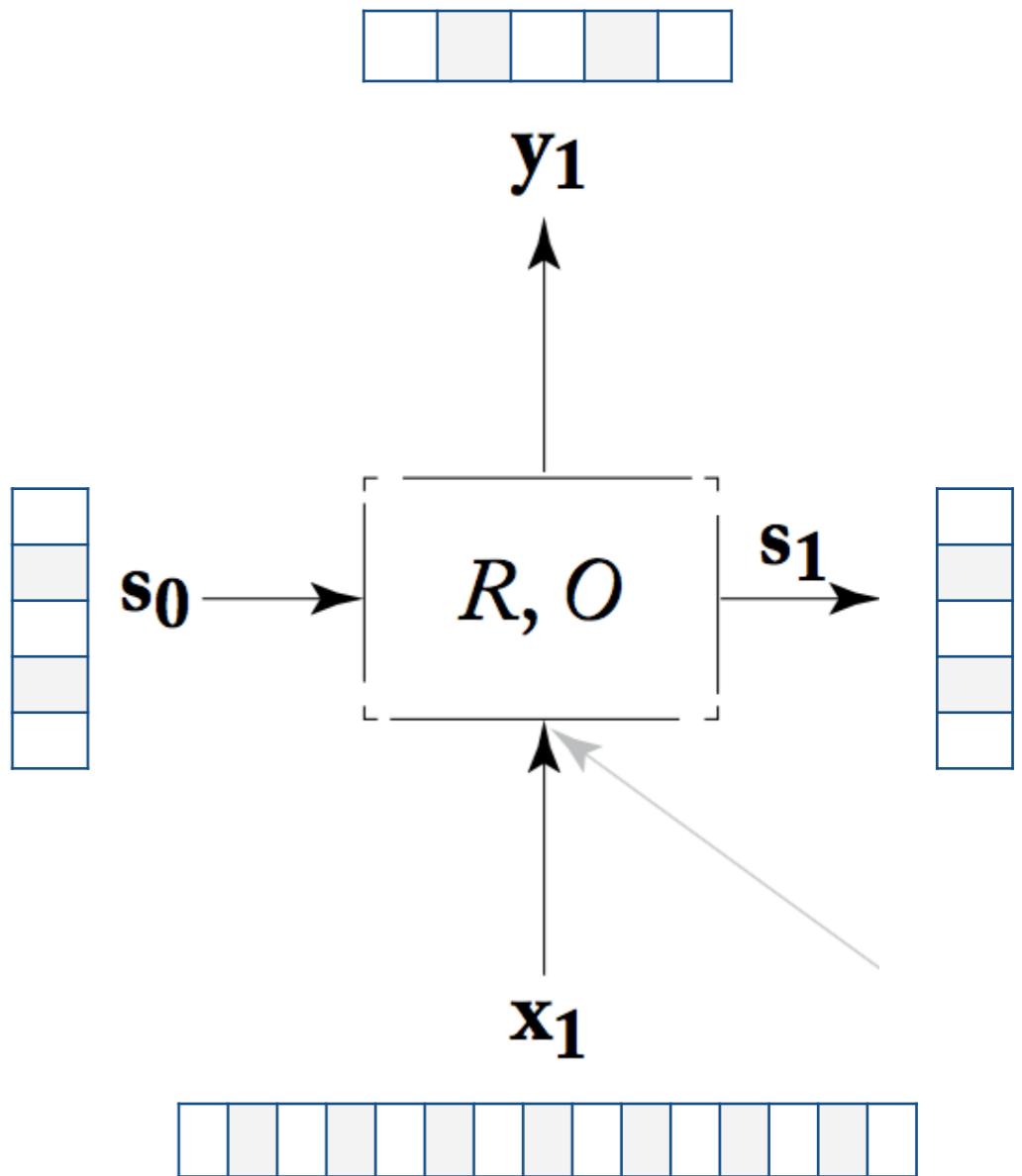


# Recurrent neural network



# Recurrent neural network

- Each time step has two inputs:
  - $x_i$  (the observation at time step  $i$ ); one-hot vector, feature vector or **distributed representation**.
  - $s_{i-1}$  (the output of the previous state); **base case:**  $s_0 = 0$  vector



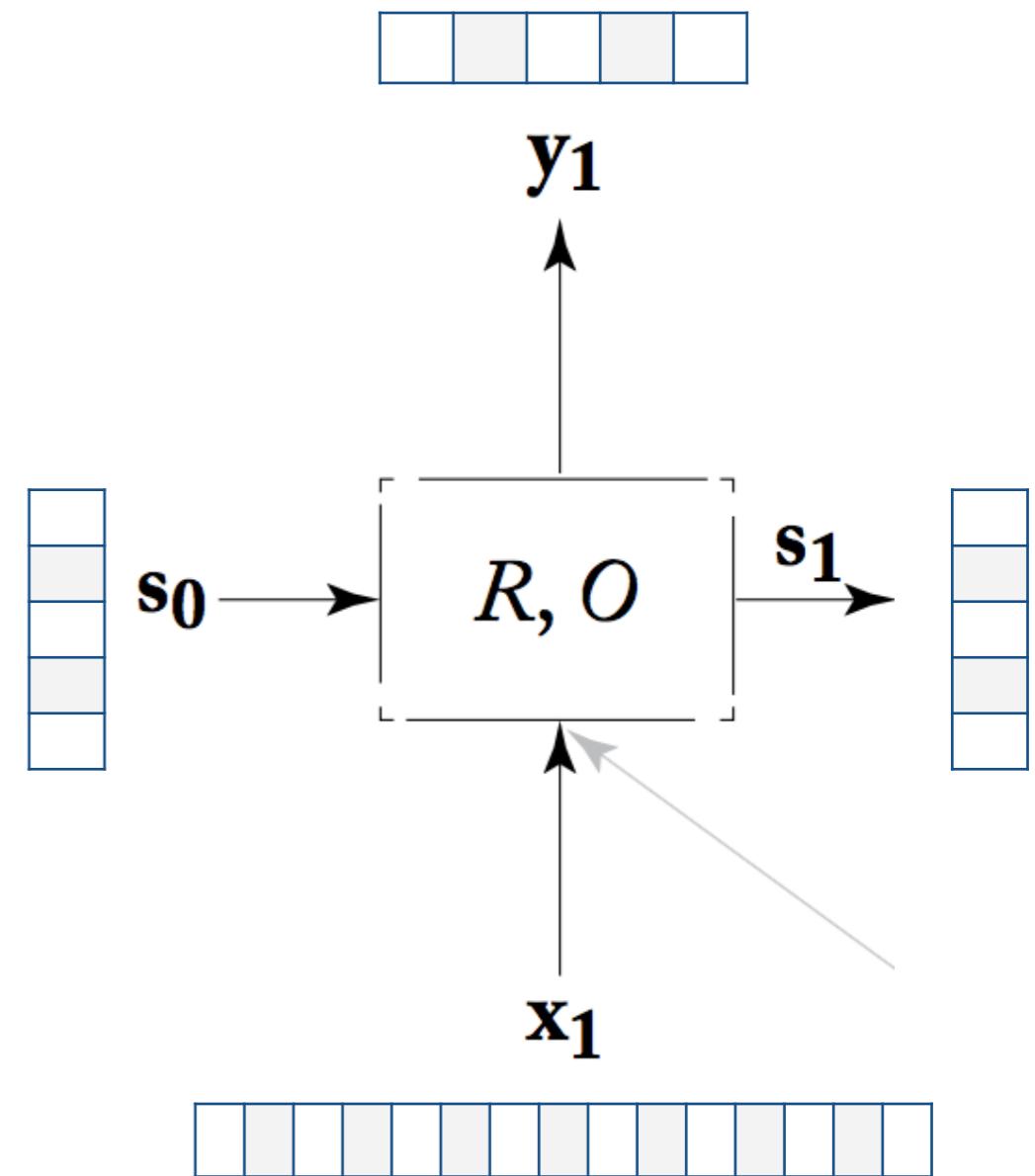
# Recurrent neural network

$$s_i = R(x_i, s_{i-1})$$

R computes the output state as a function of the current input and previous state

$$y_i = O(s_i)$$

O computes the output as a function of the current output state



# Simple RNN

$g = \tanh$  or  $\text{relu}$

$$s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b)$$

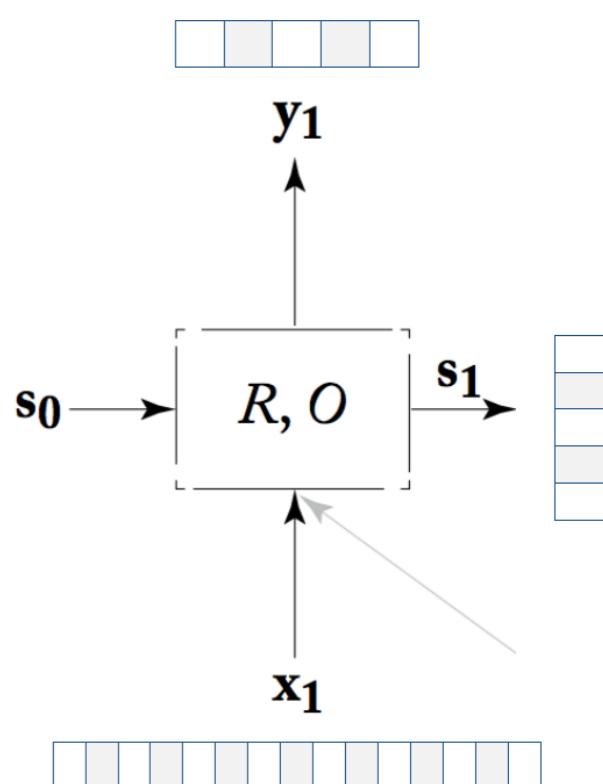
Different weight vectors  $W$  transform the previous state and current input before combining

$$W^s \in \mathbb{R}^{H \times H}$$

$$W^x \in \mathbb{R}^{D \times H}$$

$$b \in \mathbb{R}^H$$

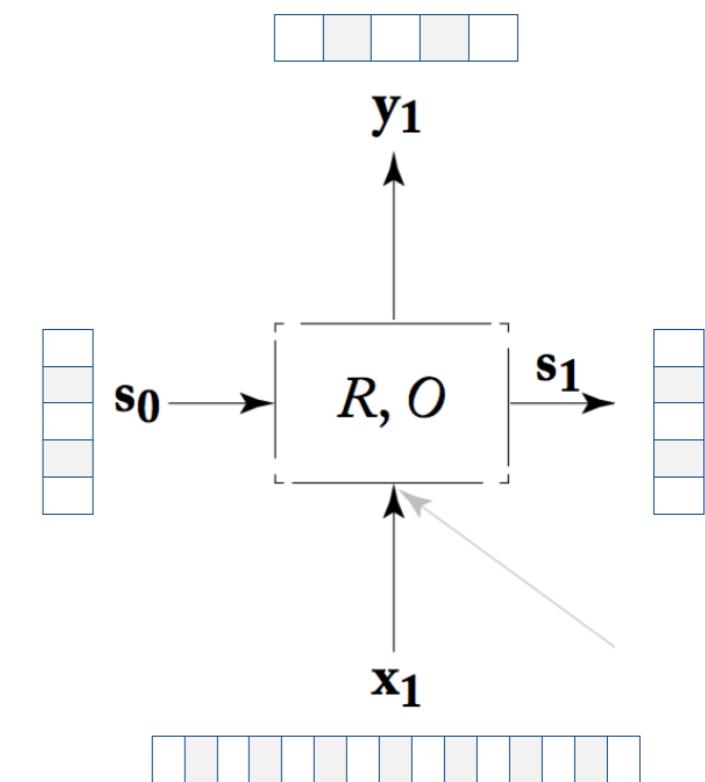
$$y_i = O(s_i) = s_i$$



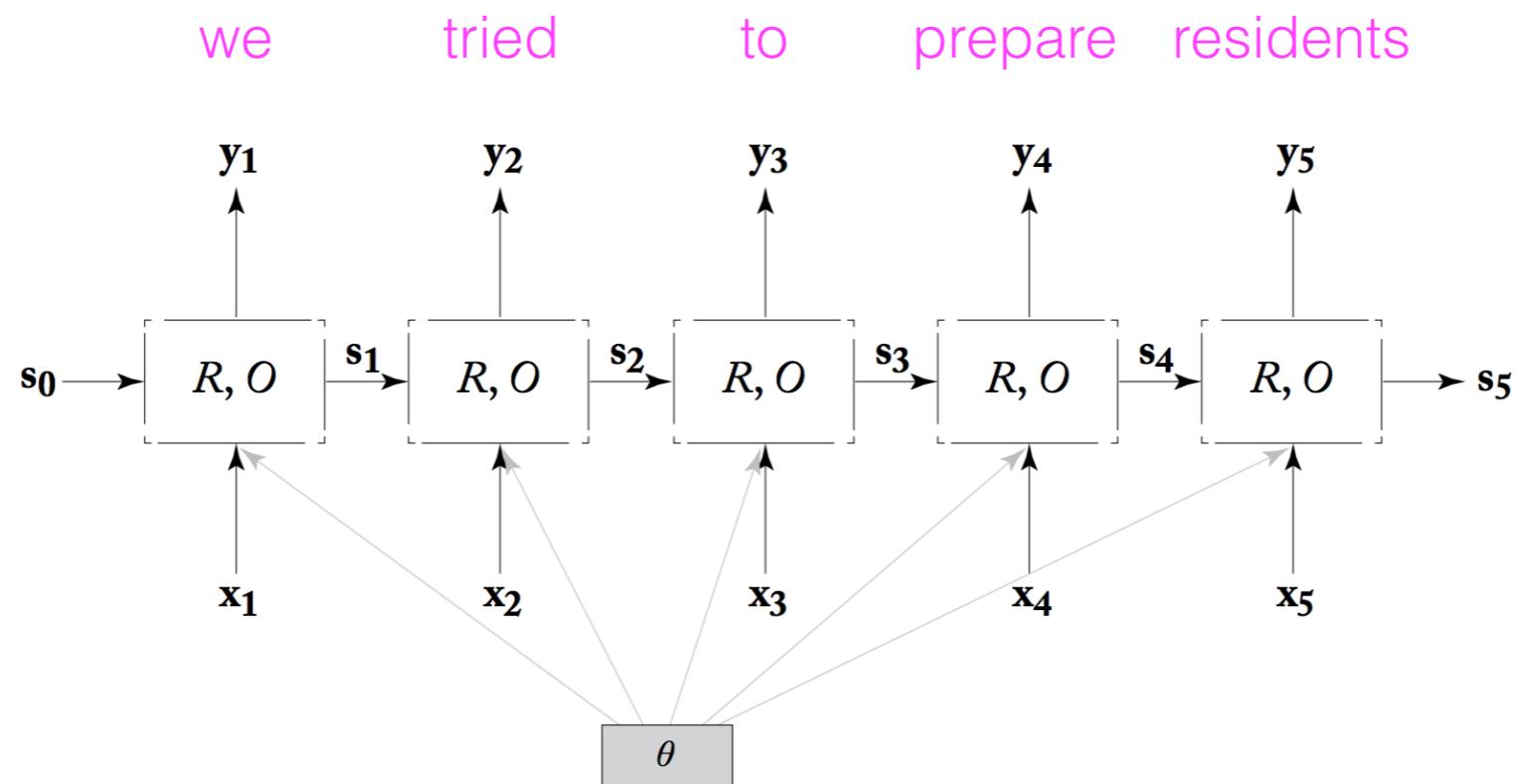
# RNN Language Model

- The output state  $s_i$  is an  $H$ -dimensional real vector; we can transform that vector into a probability distribution by passing it through an additional linear transformation followed by a softmax

$$y_i = O(s_i) = \text{softmax}(s_i W^o + b^o)$$

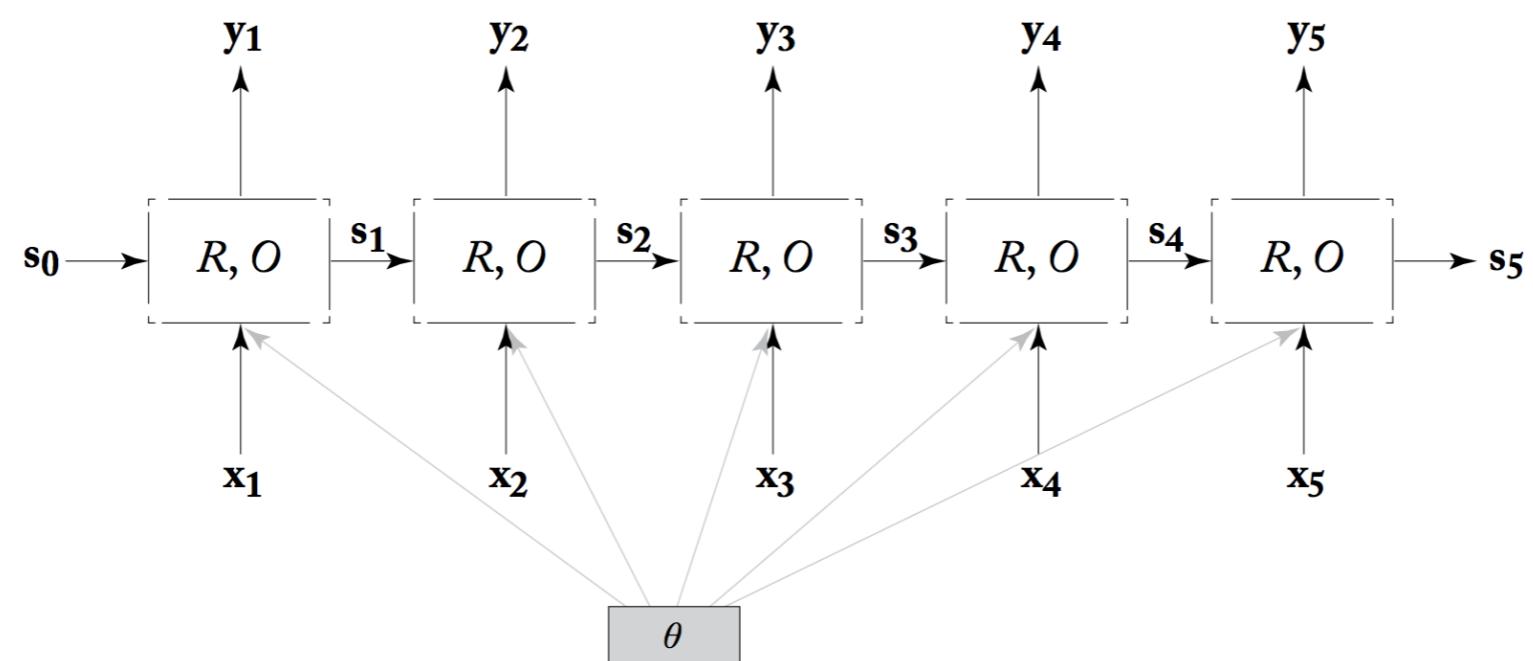


# Training RNNs

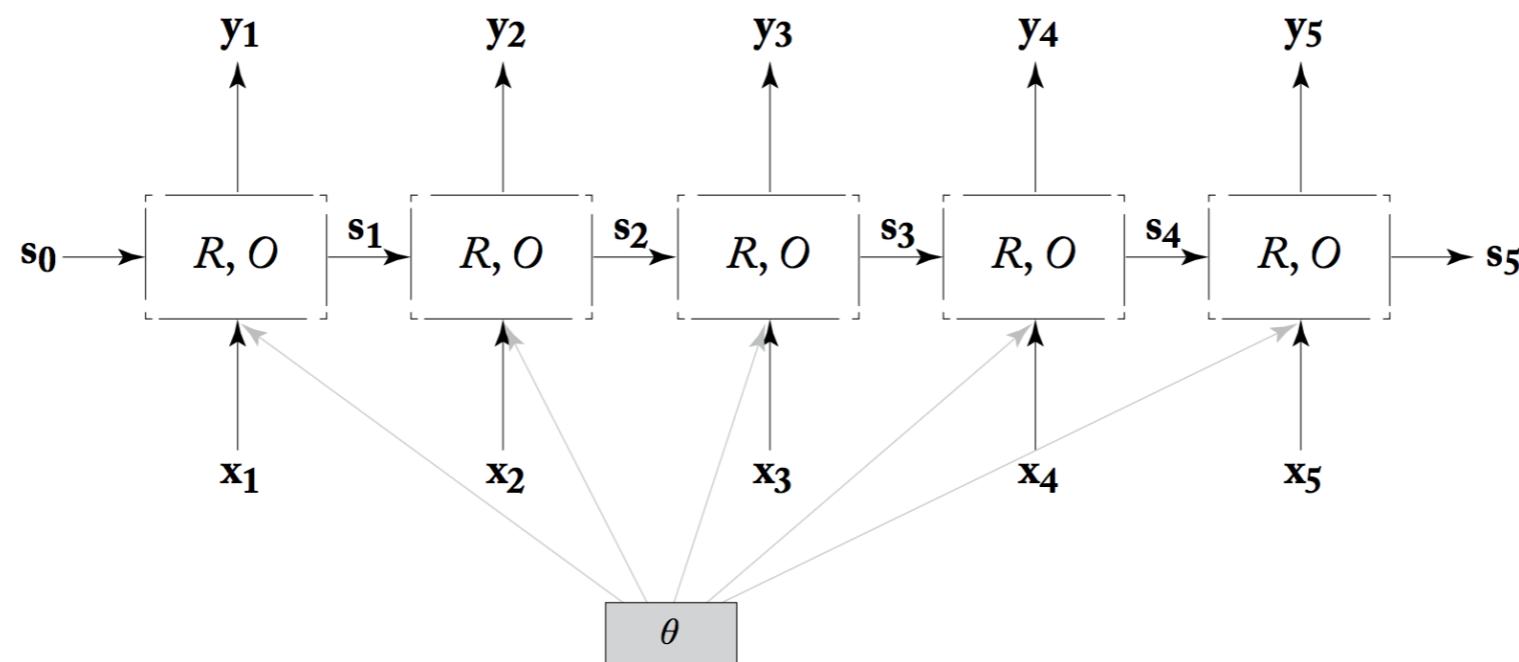


- At each time step, we make a prediction and incur a loss; we know the true  $y$  (the word we see in that position)

we tried to prepare residents



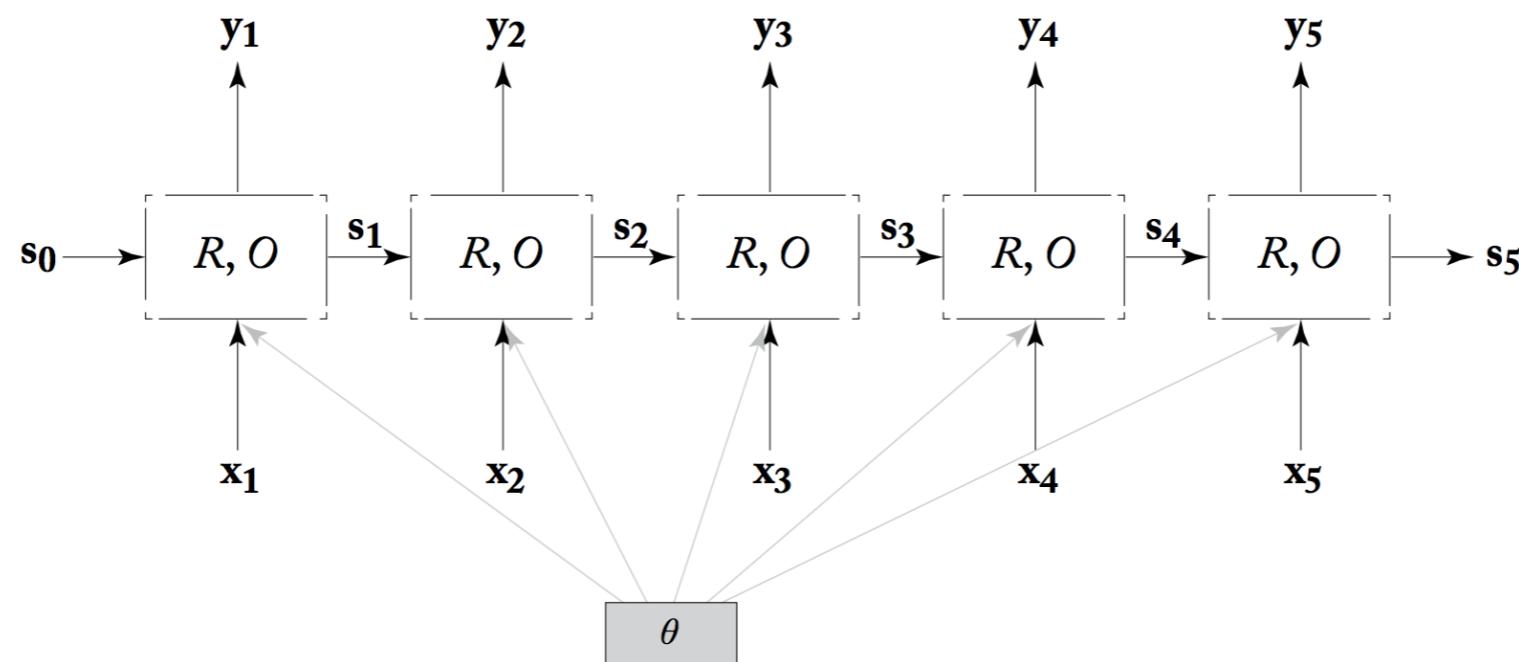
we tried to prepare residents



- Training here is standard **backpropagation**, taking the derivative of the loss we incur at step  $t$  with respect to the parameters we want to update.

$$\frac{\partial L(\theta)_{y_1}}{\partial W^s}$$

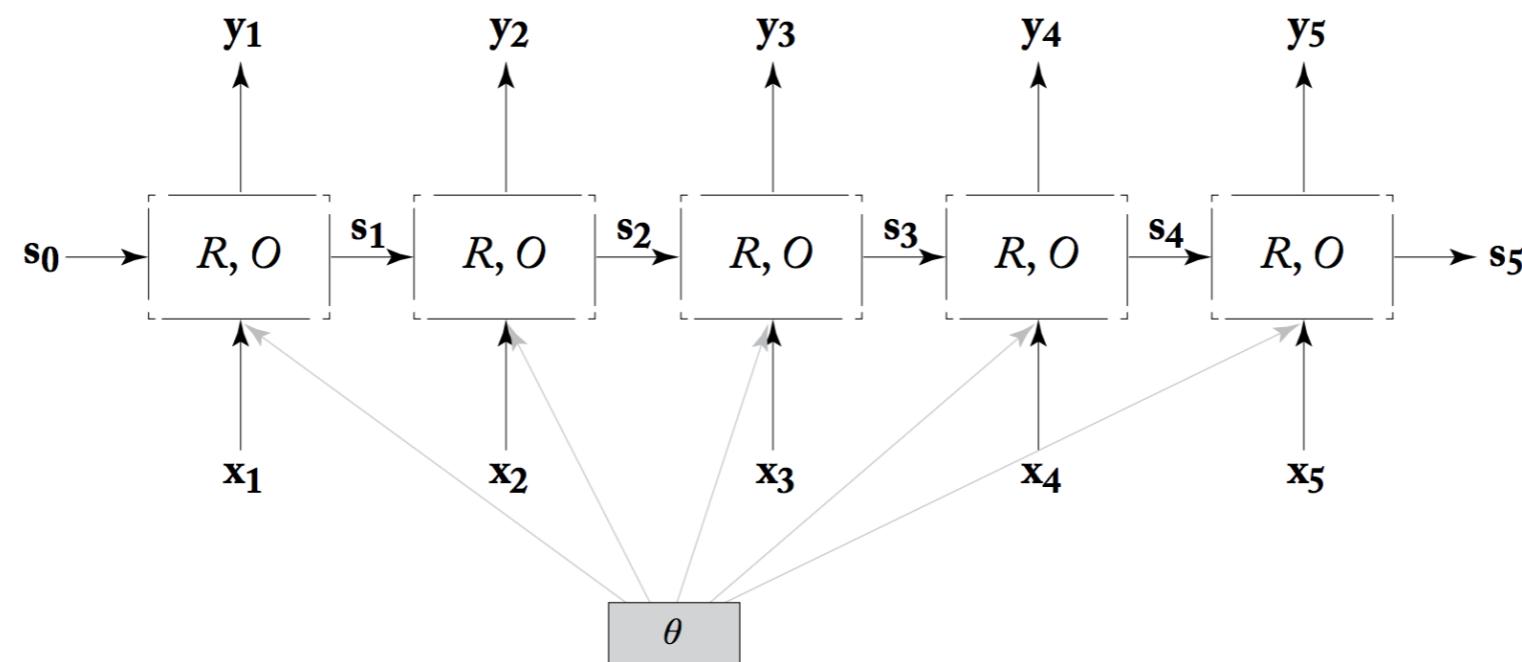
we tried to prepare residents



- Training here is standard **backpropagation**, taking the derivative of the loss we incur at step  $t$  with respect to the parameters we want to update.

$$\frac{\partial L(\theta)_{y_1}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_2}}{\partial W^s}$$

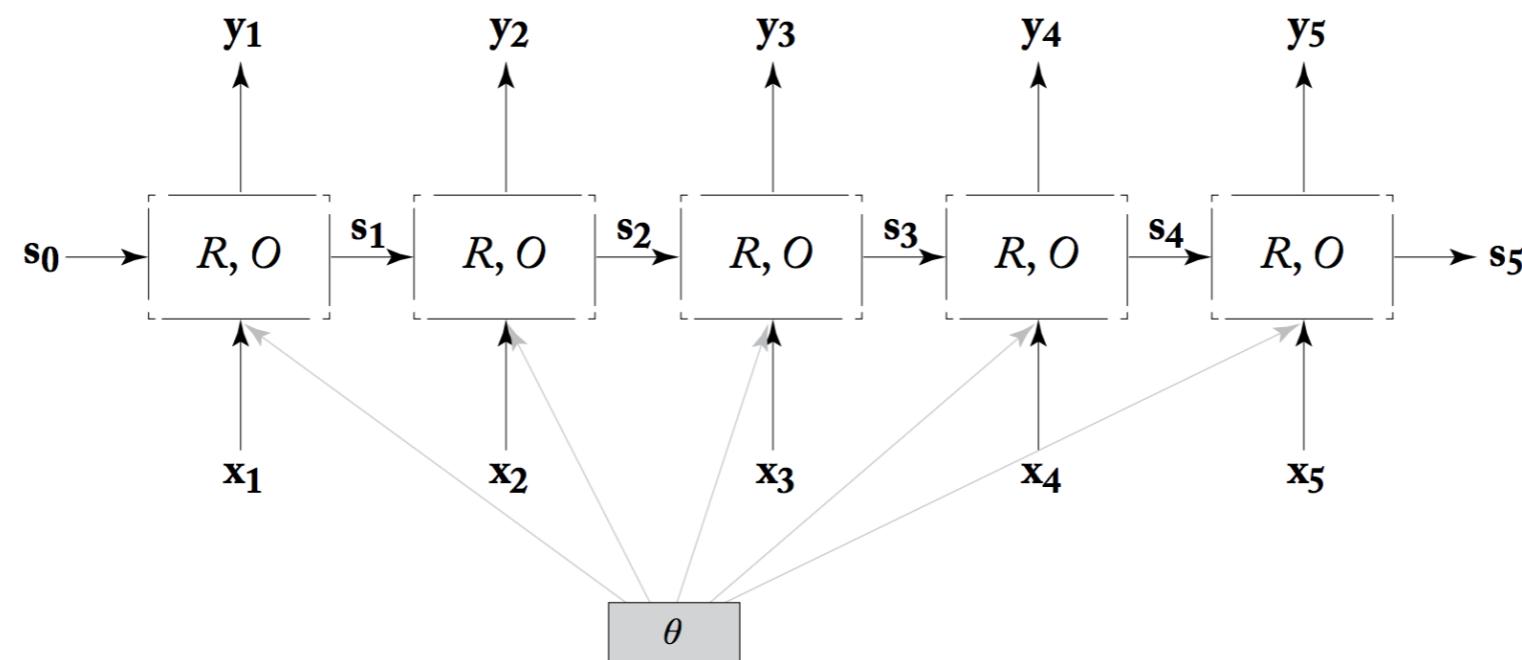
we tried to prepare residents



- Training here is standard **backpropagation**, taking the derivative of the loss we incur at step  $t$  with respect to the parameters we want to update.

$$\frac{\partial L(\theta)_{y_1}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_2}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_3}}{\partial W^s}$$

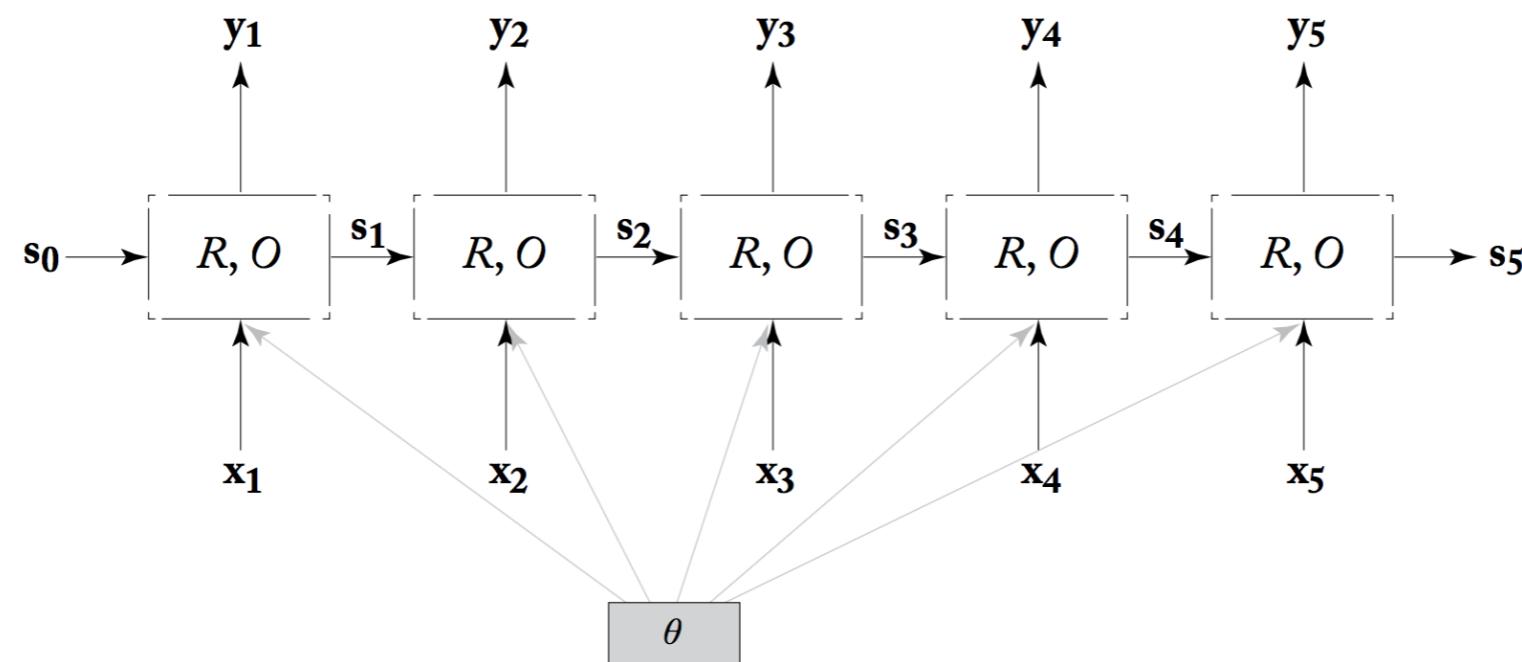
we tried to prepare residents



- Training here is standard **backpropagation**, taking the derivative of the loss we incur at step  $t$  with respect to the parameters we want to update.

$$\frac{\partial L(\theta)_{y_1}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_2}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_3}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_4}}{\partial W^s}$$

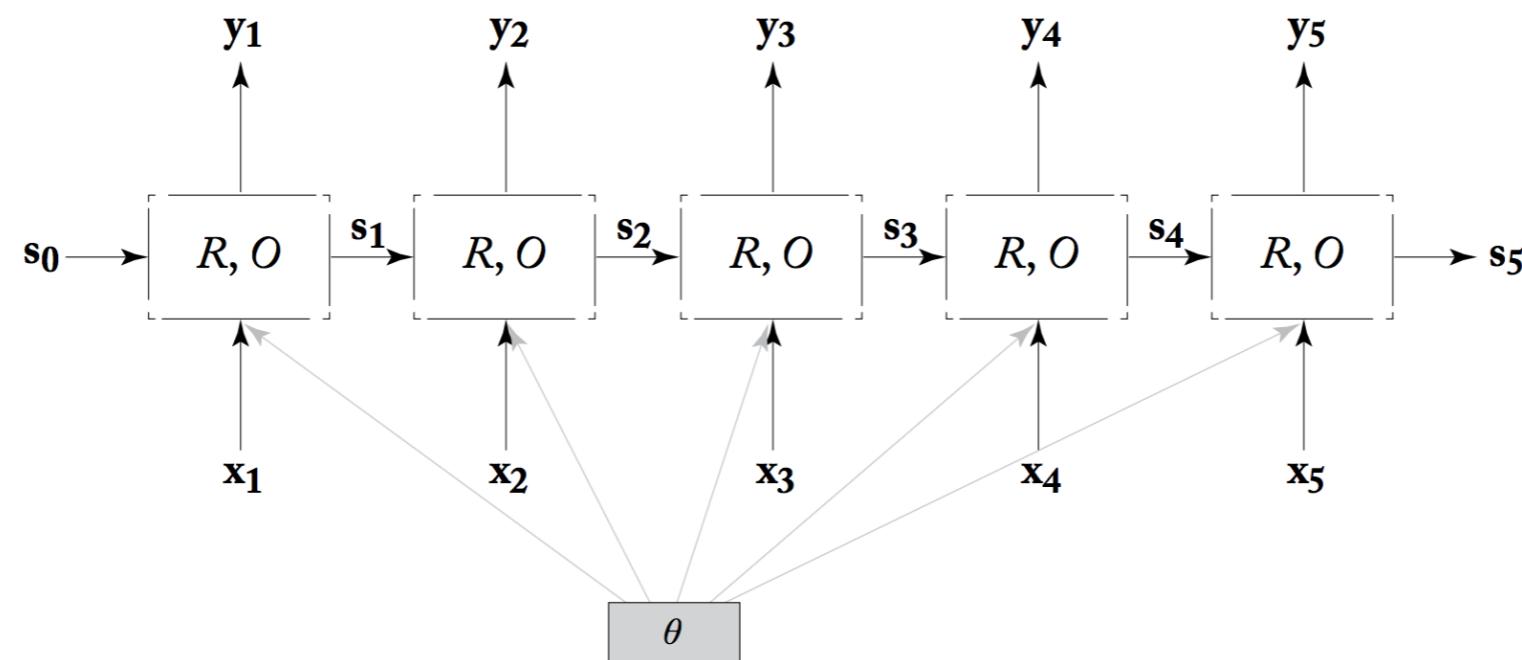
we tried to prepare residents



- Training here is standard **backpropagation**, taking the derivative of the loss we incur at step  $t$  with respect to the parameters we want to update.

$$\frac{\partial L(\theta)_{y_1}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_2}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_3}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_4}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_5}}{\partial W^s}$$

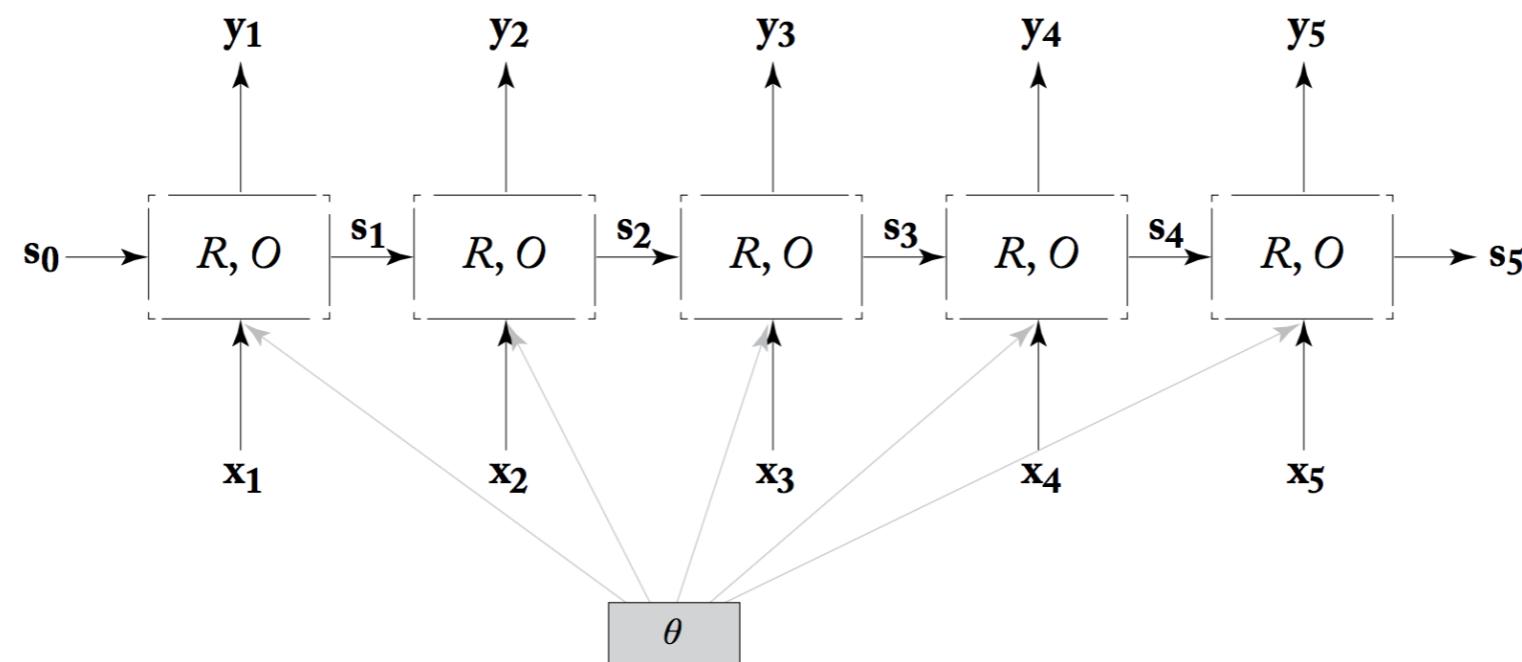
we tried to prepare residents



- Training here is standard **backpropagation**, taking the derivative of the loss we incur at step  $t$  with respect to the parameters we want to update.

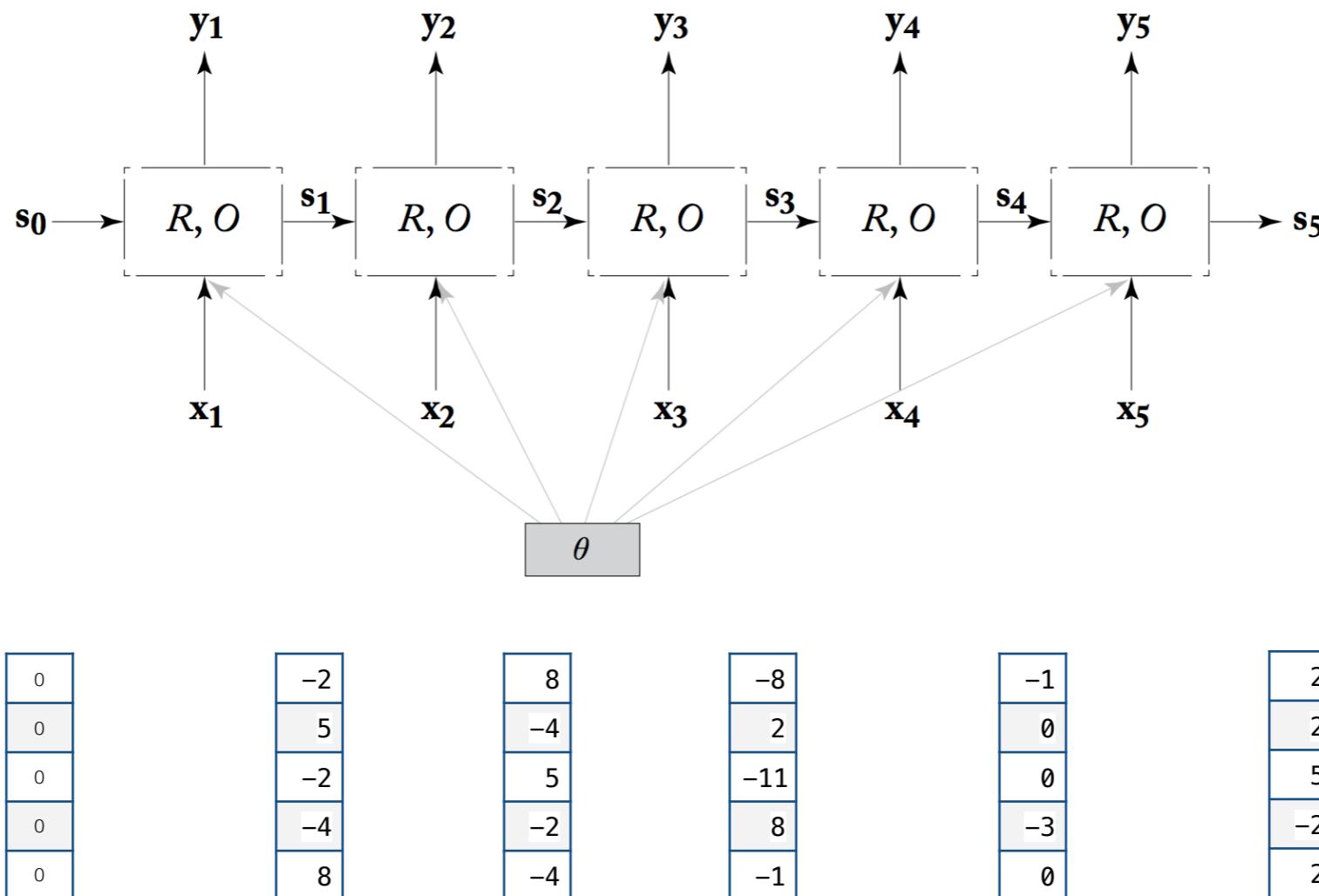
$$\frac{\partial L(\theta)_{y_1}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_2}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_3}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_4}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_5}}{\partial W^s}$$

we tried to prepare residents



- Training here is standard **backpropagation**, taking the derivative of the loss we incur at step  $t$  with respect to the parameters we want to update.
  - You might see this called **backpropagation through time** (BPTT)

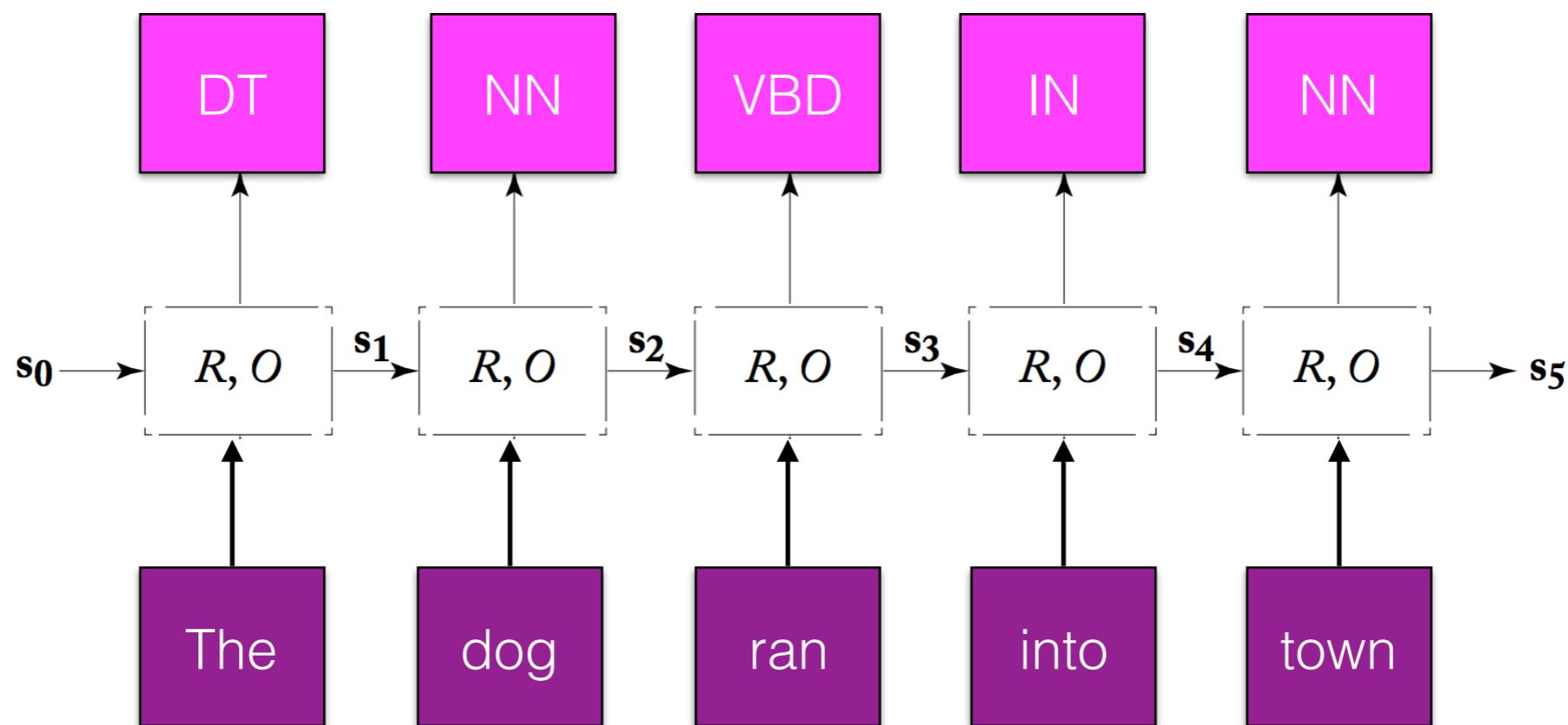
we tried to prepare residents



Each state  $i$  encodes information seen until time  $i$   
and its structure is optimized to predict the next word

# Recurrent neural network

- RNNs for POS tagging, predict the tag from  $\mathcal{Y}$  conditioned on the context



# RNNs for POS

NN TO VB

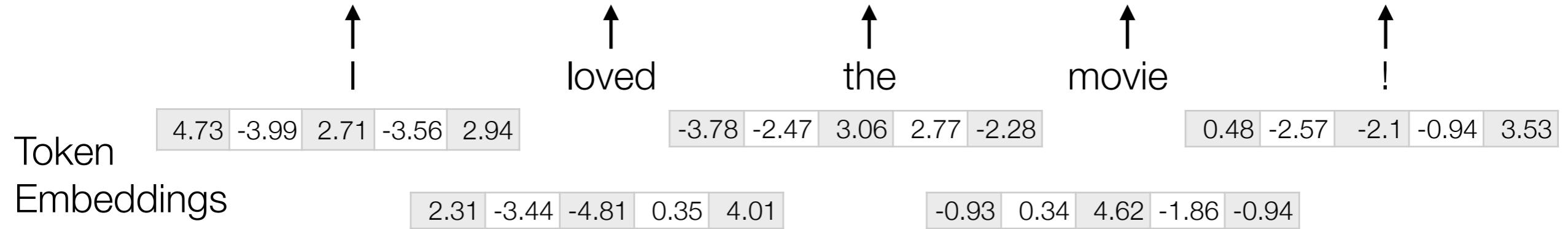
will to fight

- To make a prediction for  $y_t$ , RNNs condition on all input seen through time  $t$  ( $x_1, \dots, x_t$ )
- But knowing something about the **future** can help ( $x_{t+1}, \dots, x_n$ )

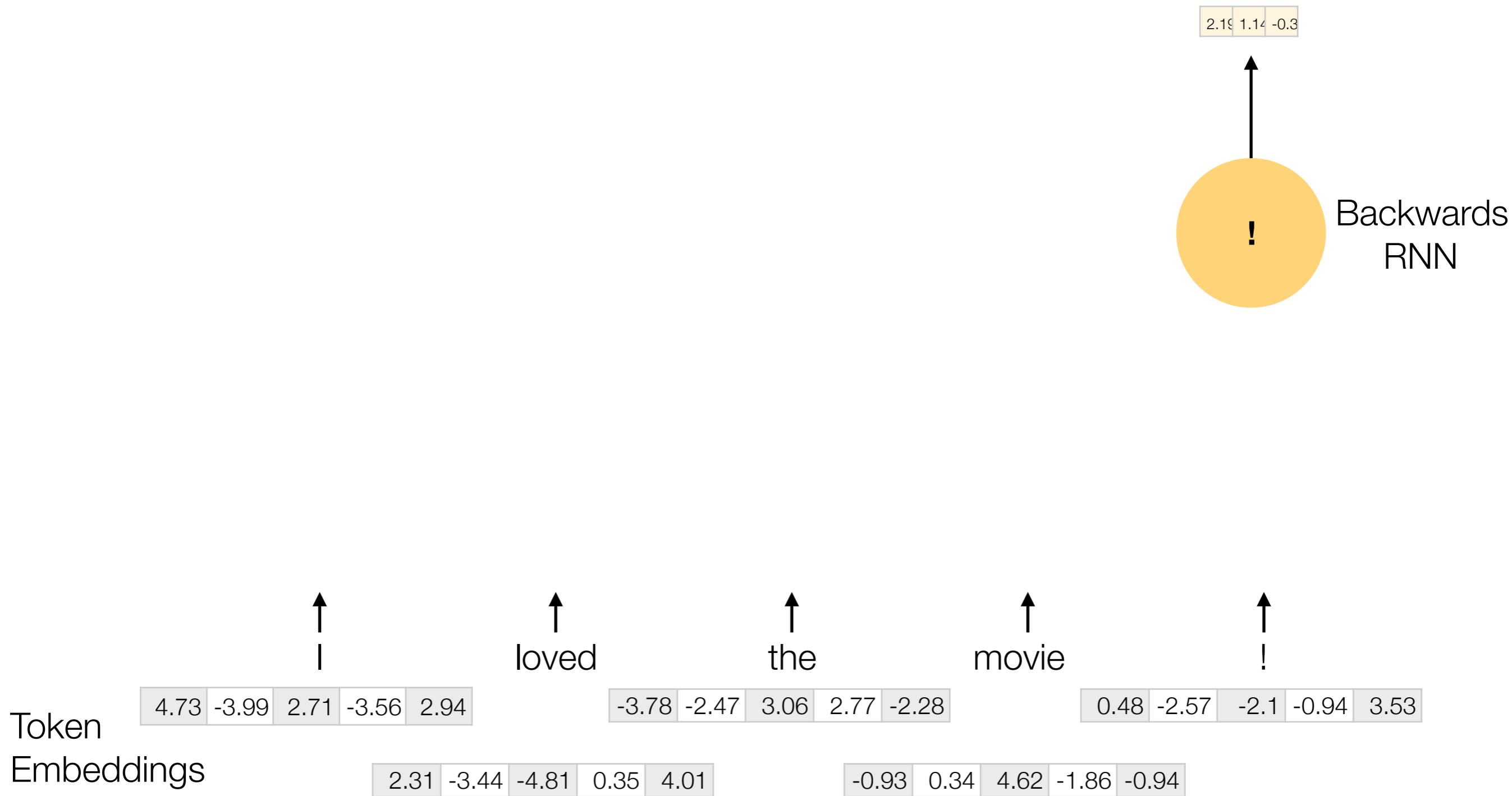
# Bidirectional RNN

- A powerful alternative is make predictions conditioning both on the **past** and the **future**.
- Two RNNs
  - One running left-to-right
  - One right-to-left
- Each produces an output vector at each time step, which we concatenate

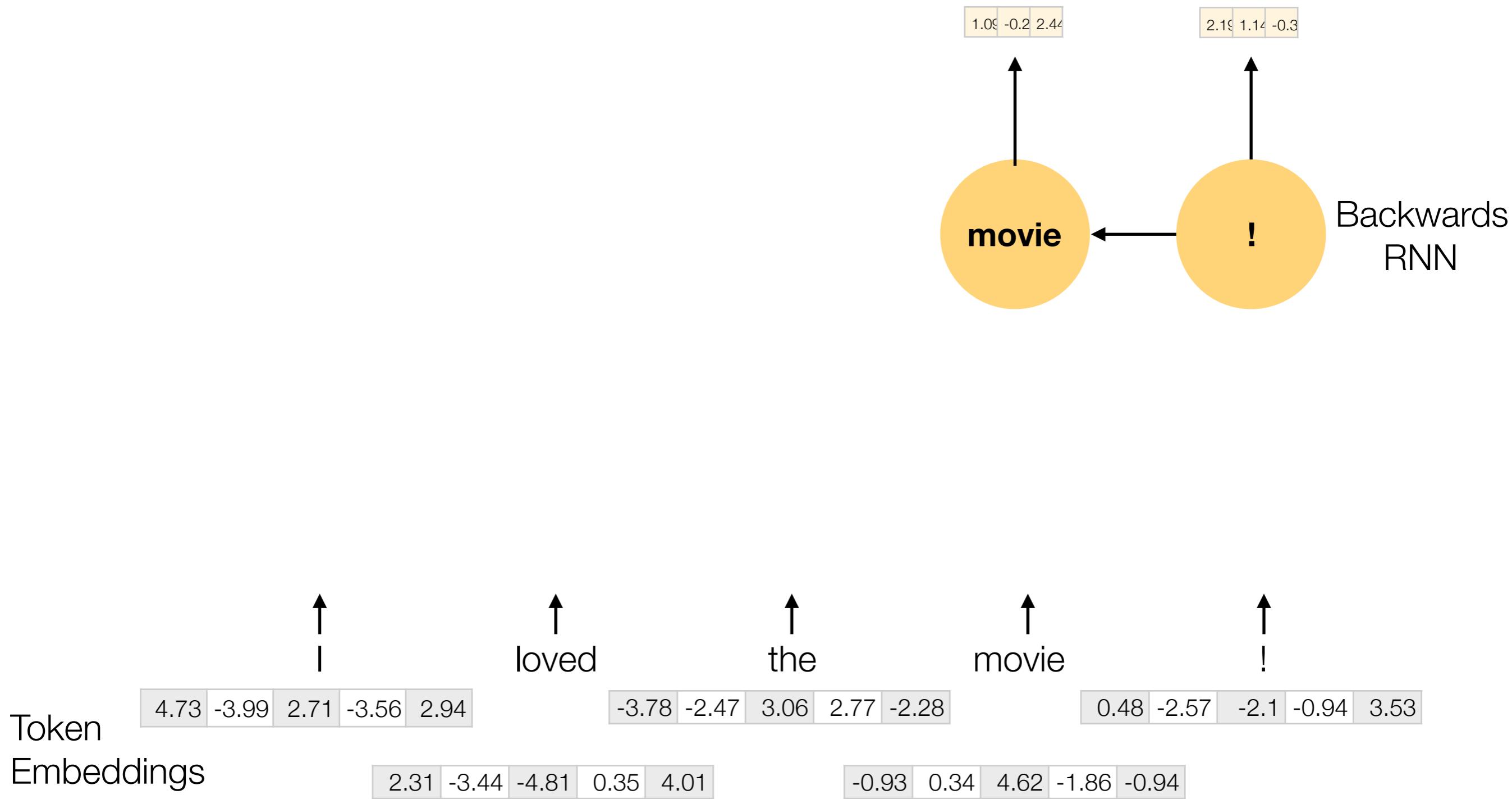
# Bidirectional RNN



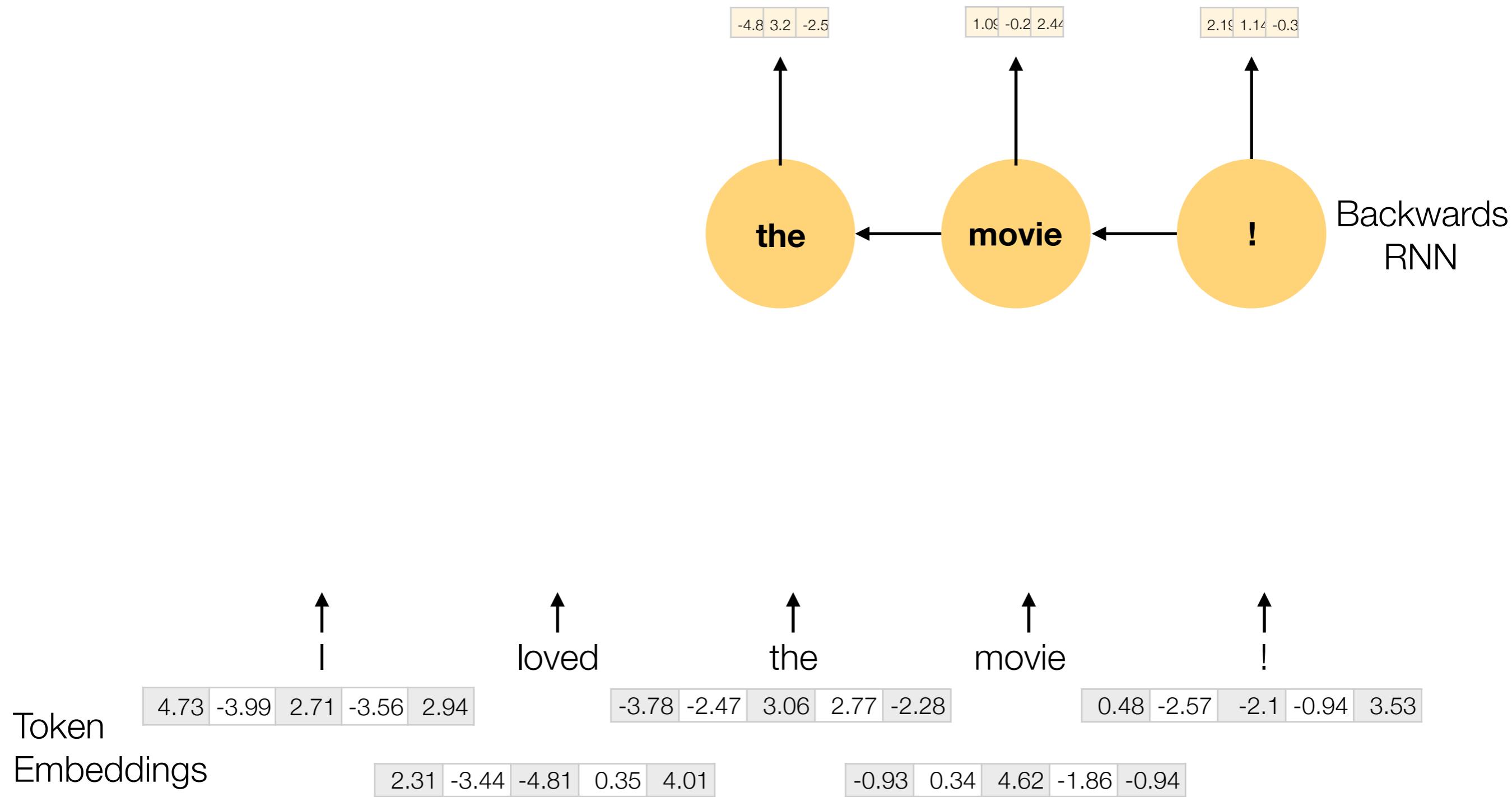
# Bidirectional RNN



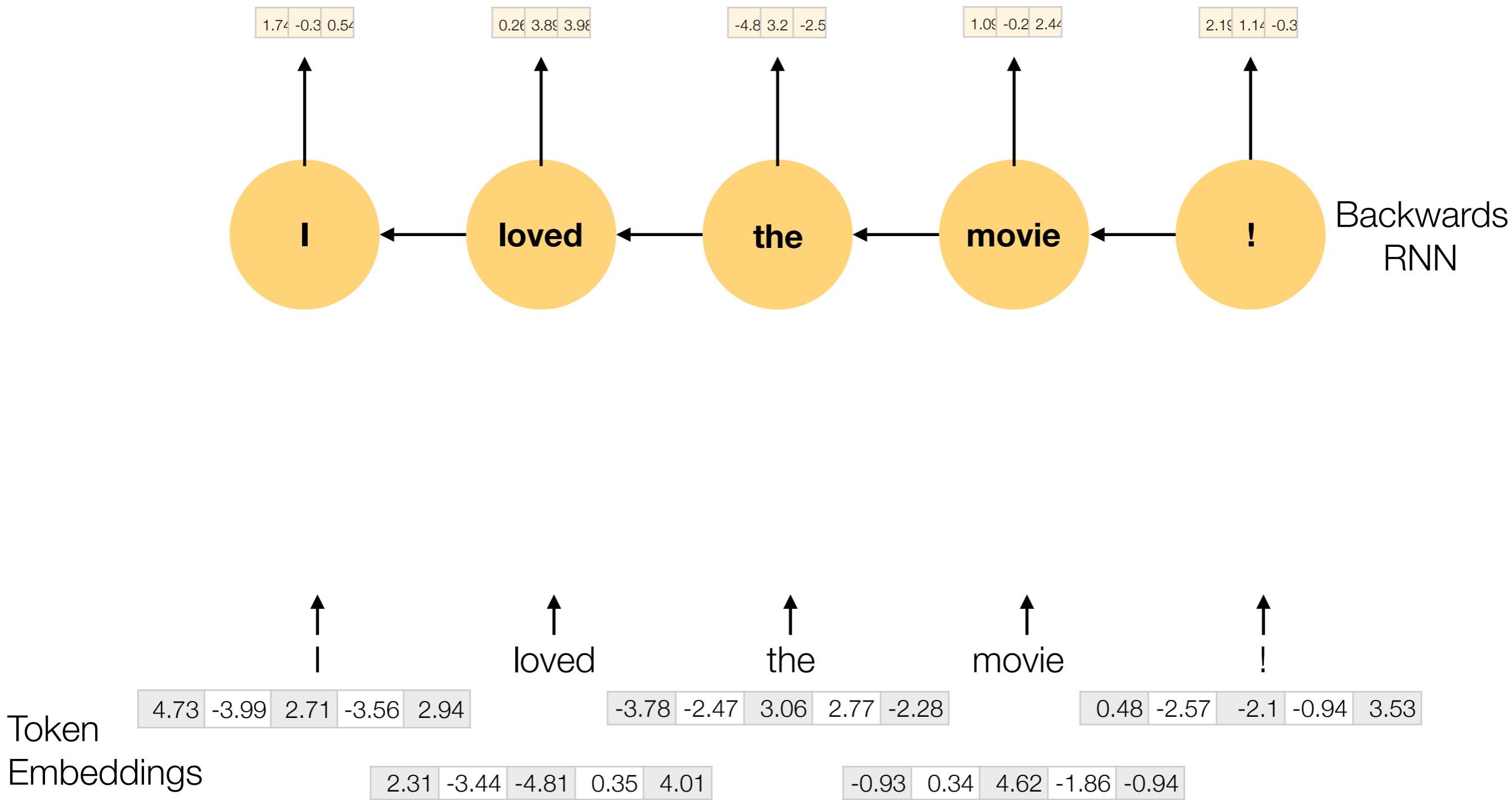
# Bidirectional RNN



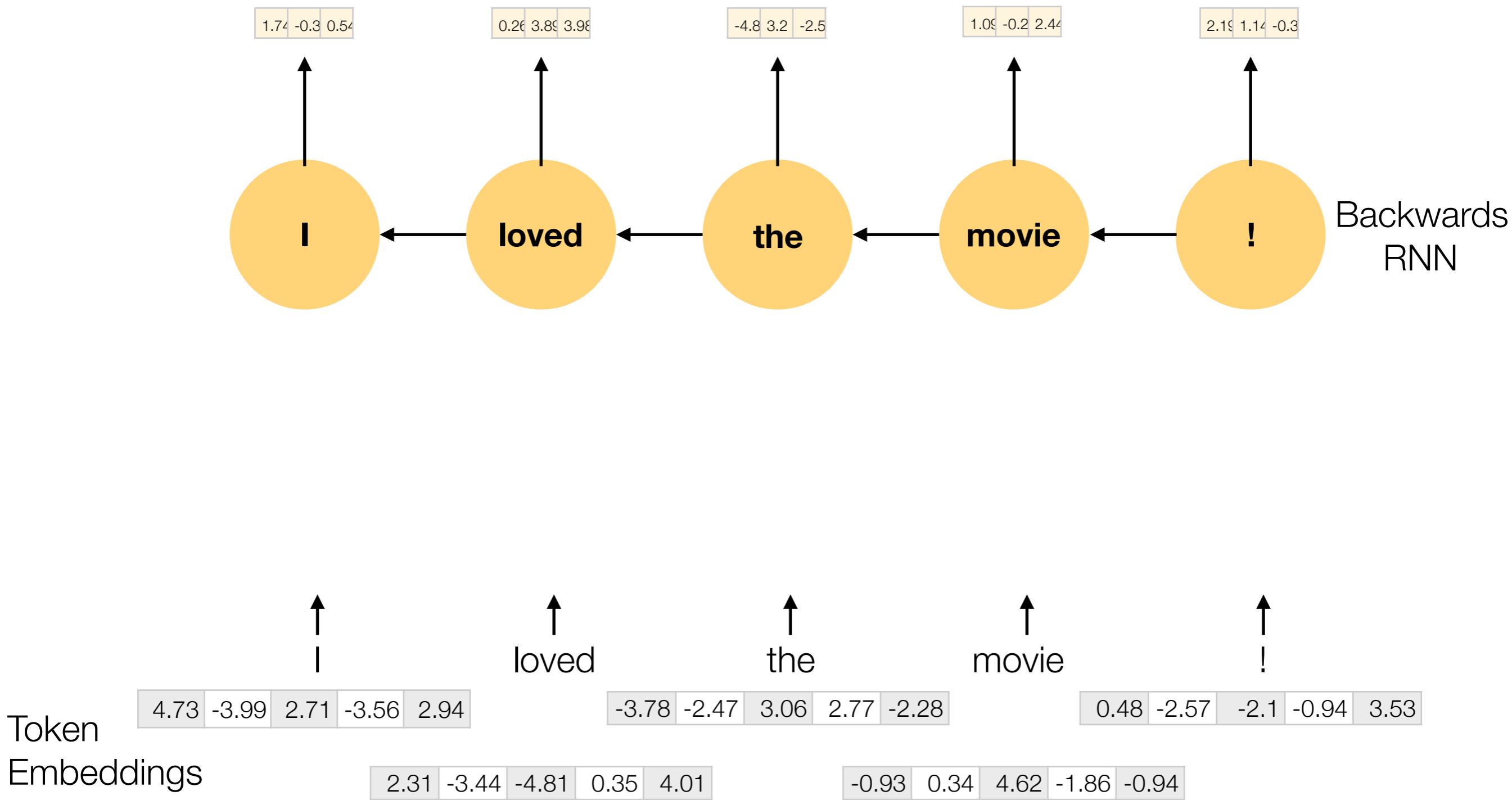
# Bidirectional RNN



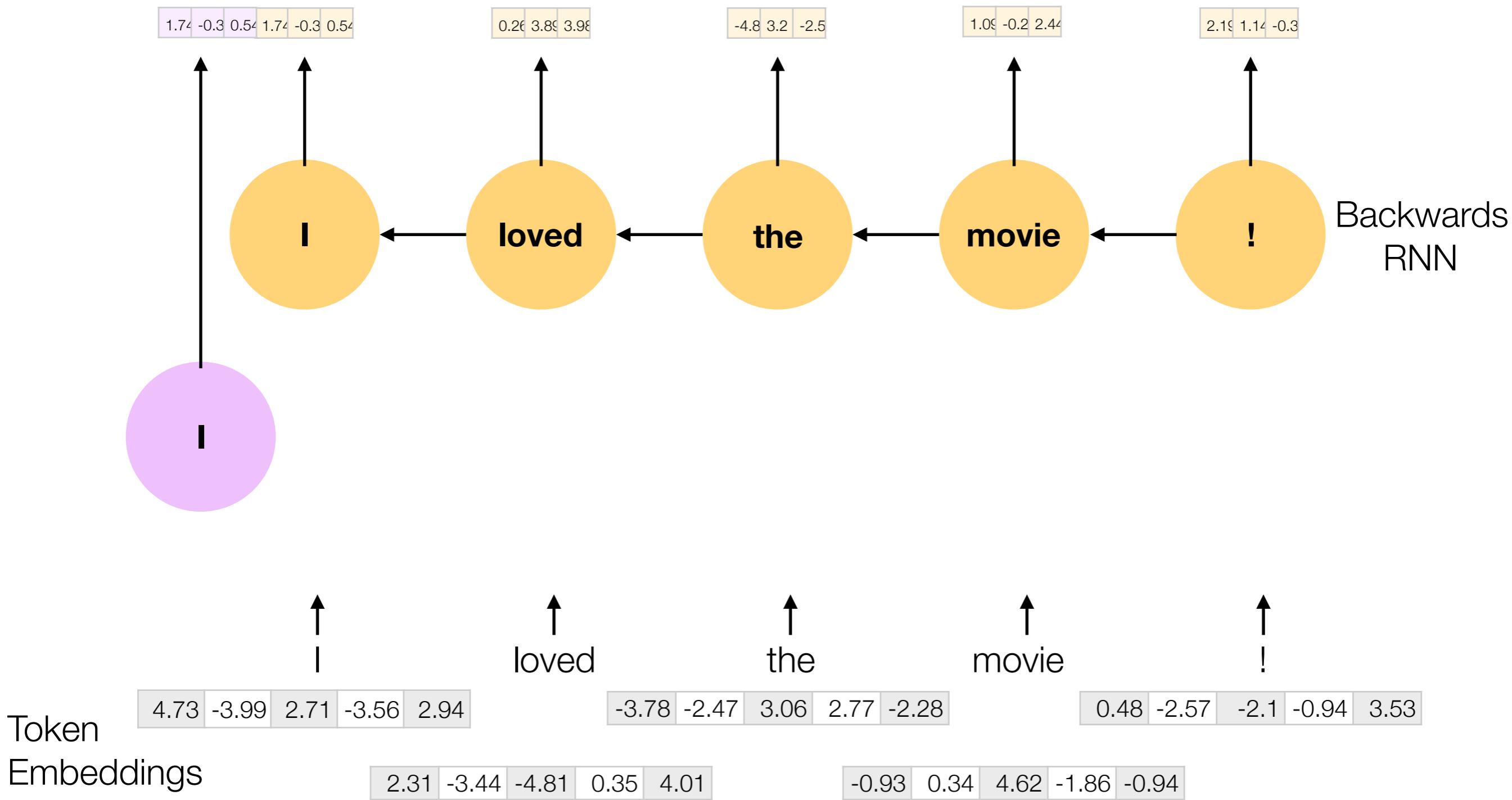
# Bidirectional RNN



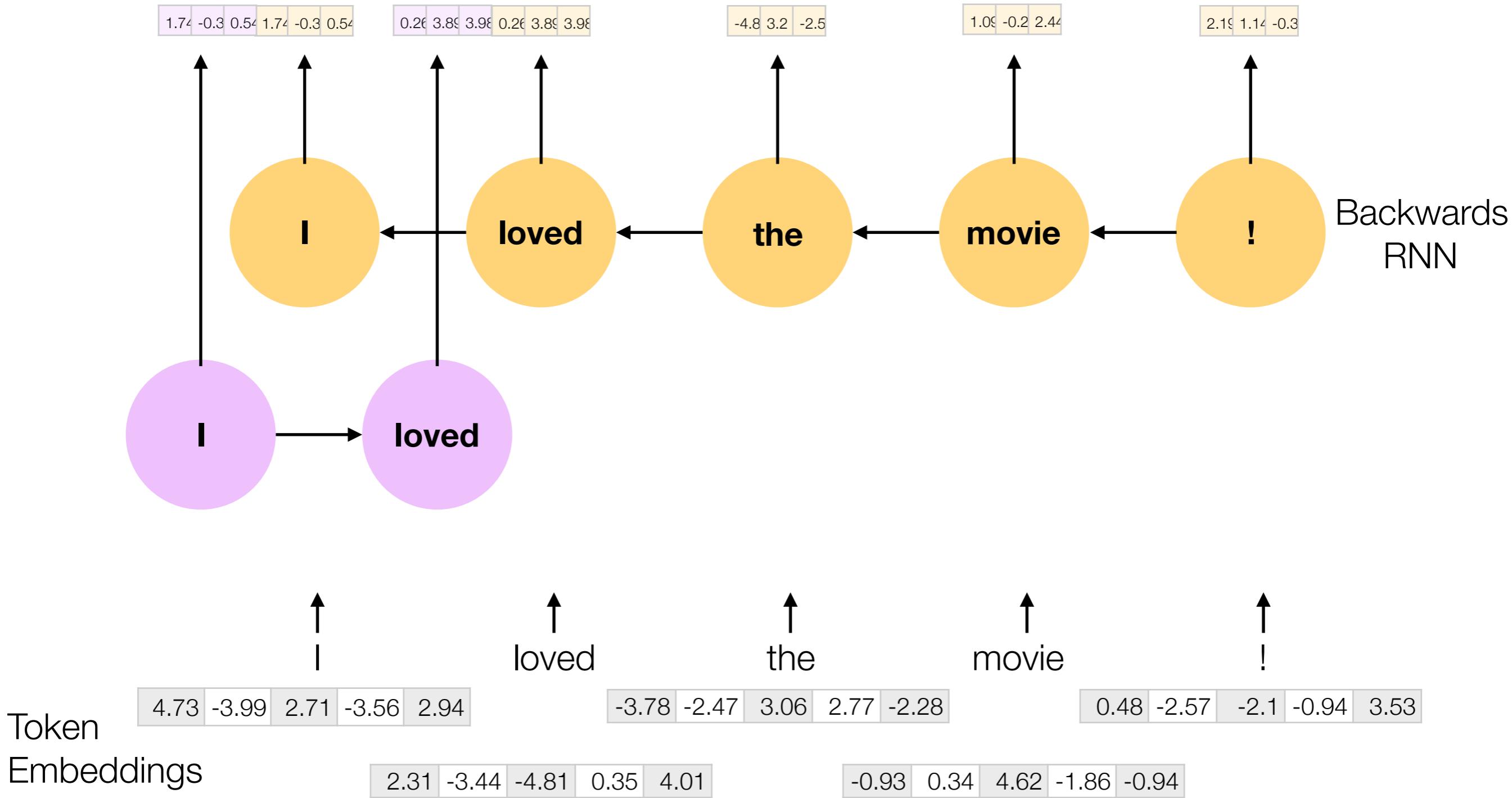
# Bidirectional RNN



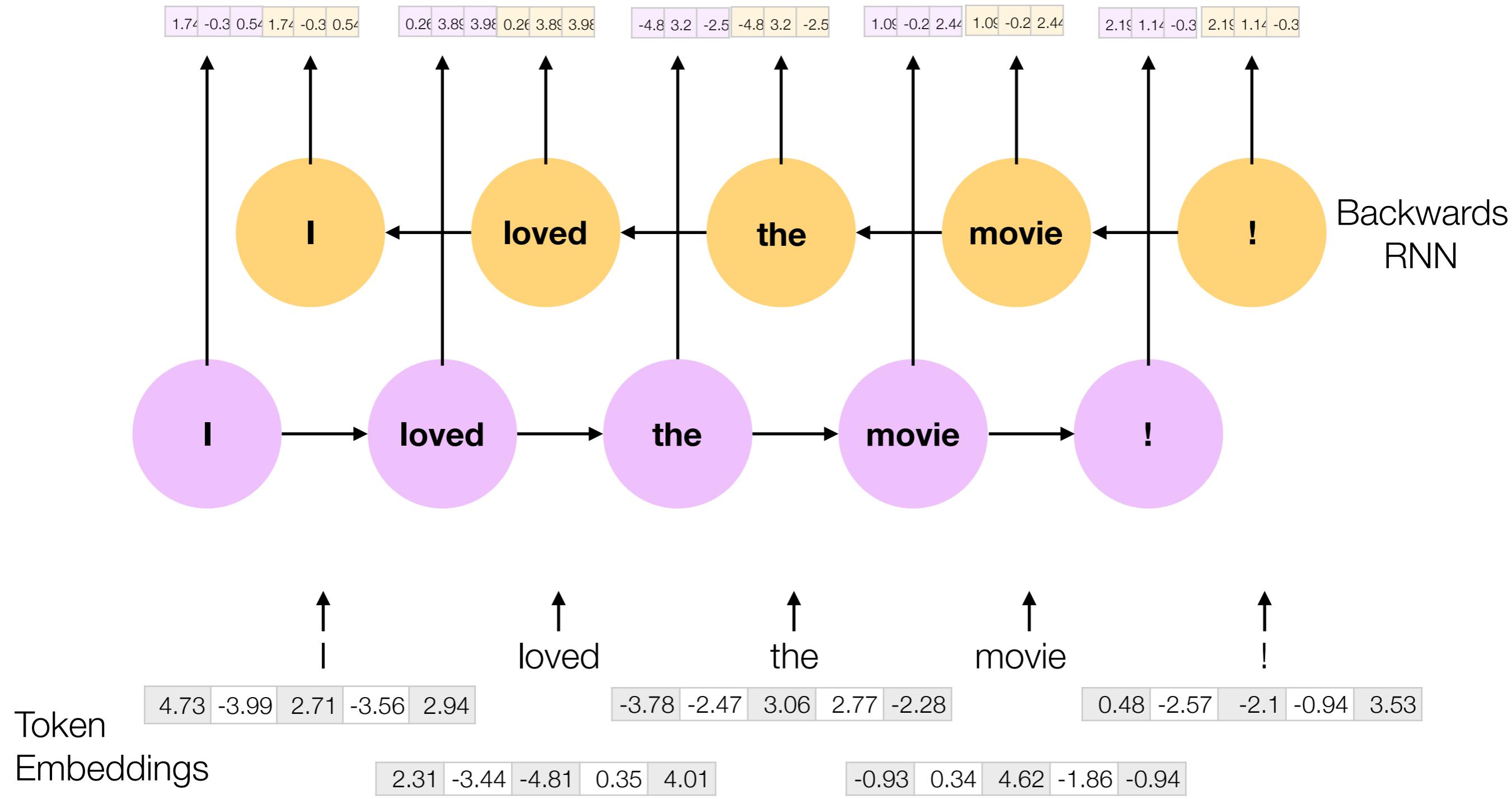
# Bidirectional RNN



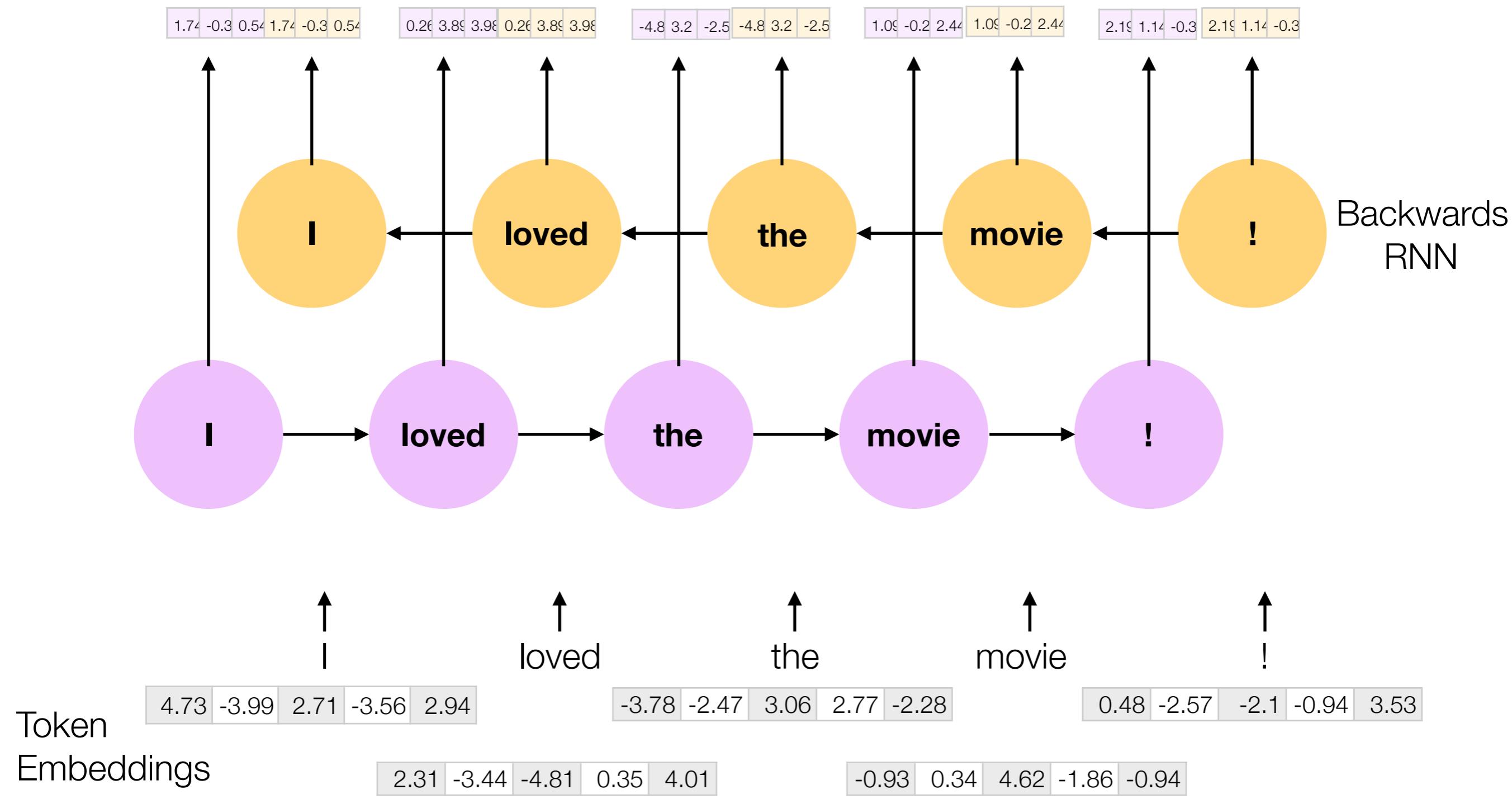
# Bidirectional RNN



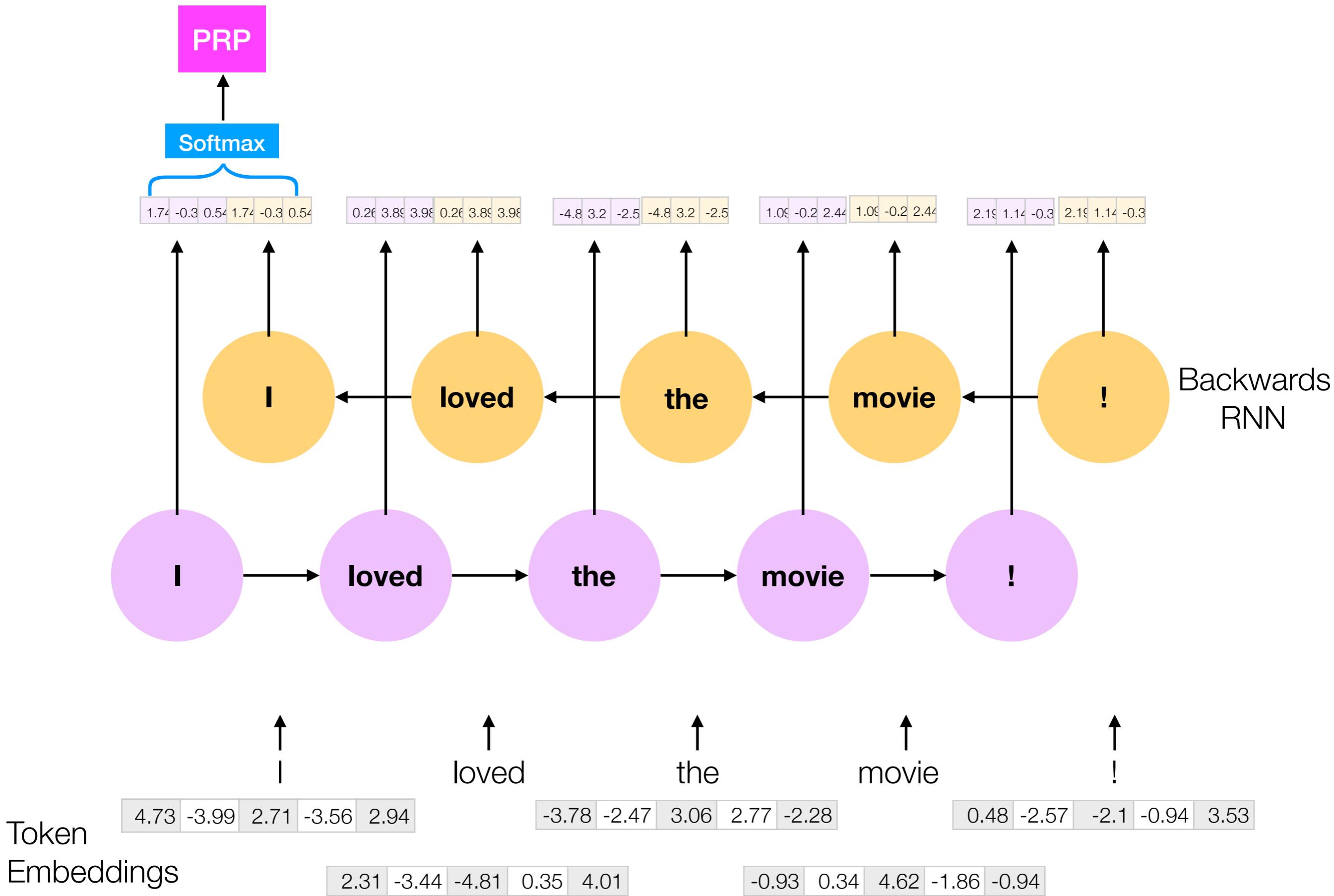
# Bidirectional RNN



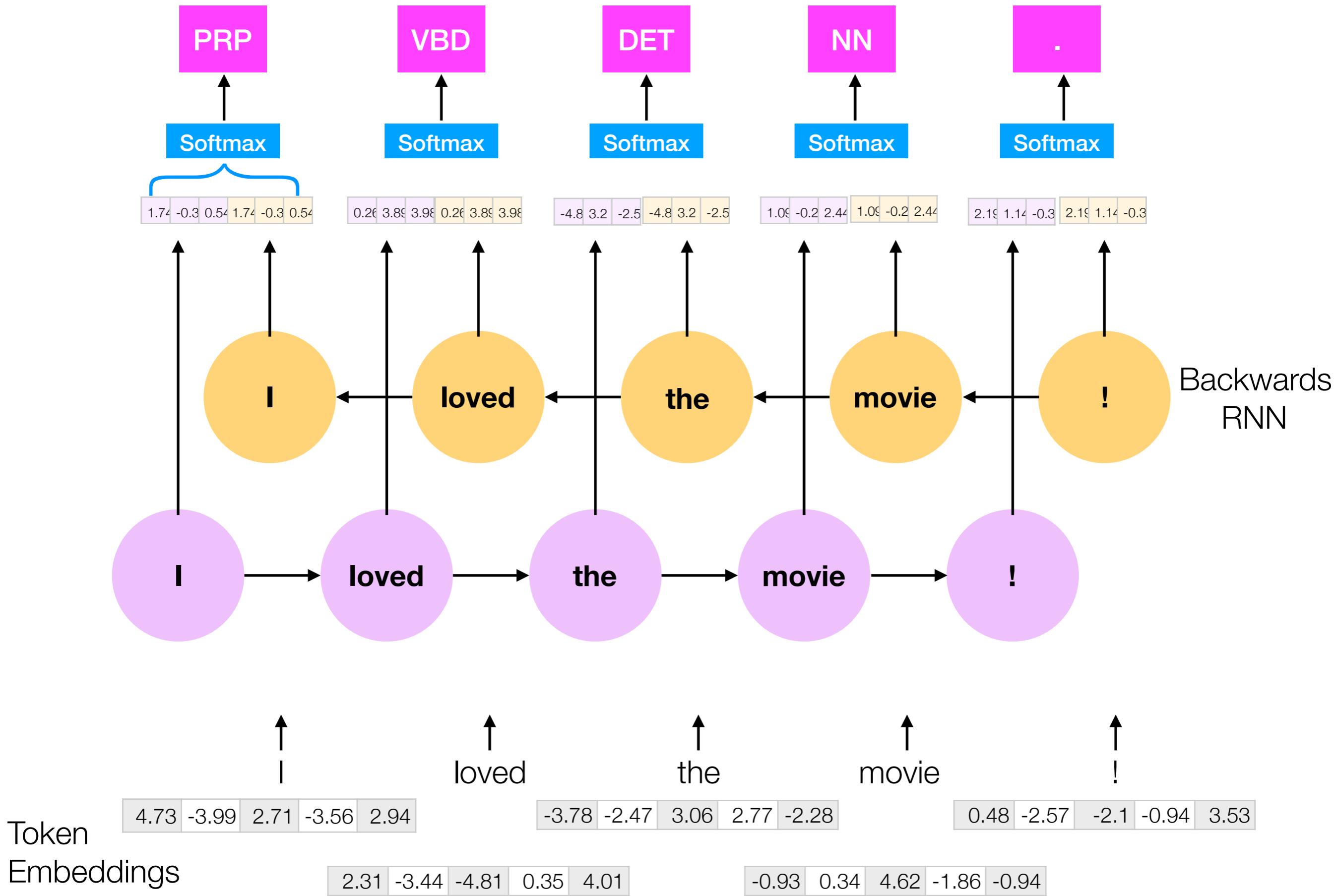
# Bidirectional RNN



# Bidirectional RNN

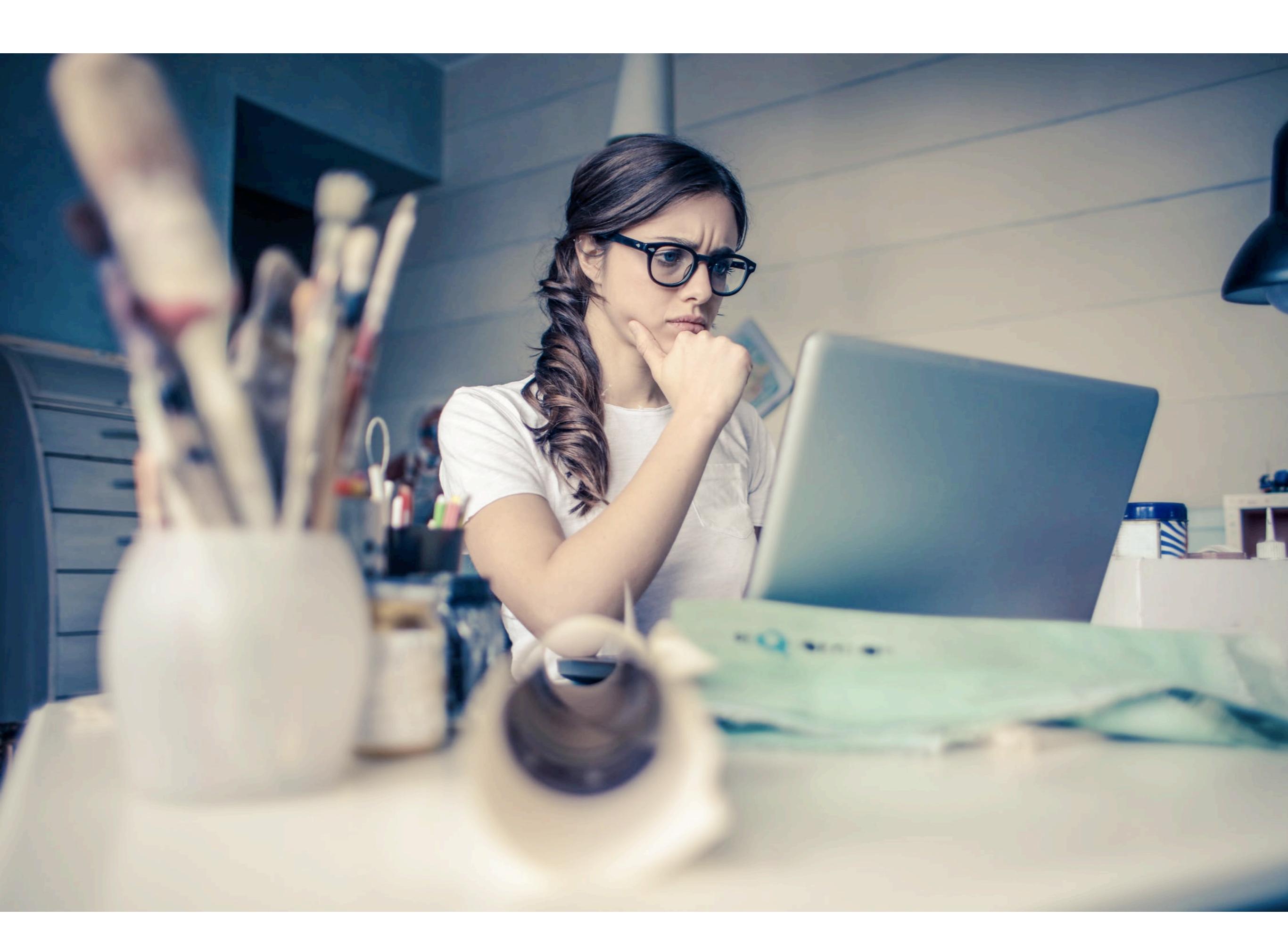


# Bidirectional RNN



# Long short-term memory network (LSTM)

- Designed to account for the vanishing gradient problem and replace **vanilla RNNs**
- Basic idea: split the  $s$  vector propagated between time steps into a **memory** component and a **hidden state** component

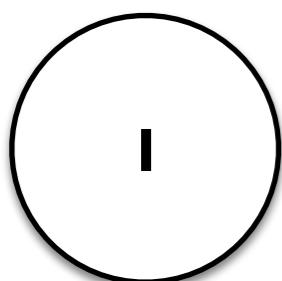


Thinking about  
neural nets

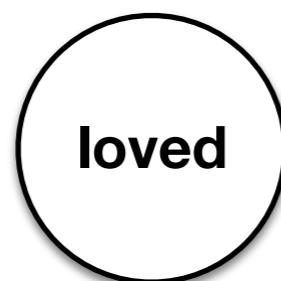
# How could we use word embeddings for document classification?

Token  
Embeddings

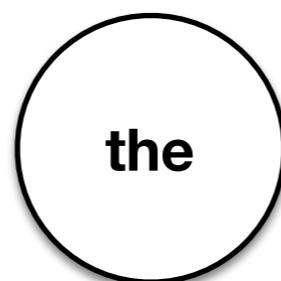
4.7	-3.9	2.7	-3.5	2.9
-----	------	-----	------	-----



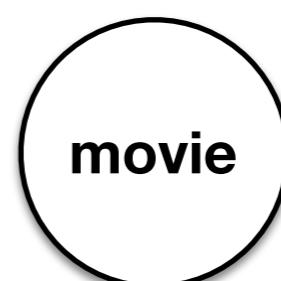
2.3	-3.4	-4.8	0.3	4.0
-----	------	------	-----	-----



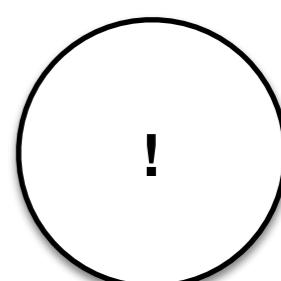
-3.7	-2.4	3.0	2.7	-2.2
------	------	-----	-----	------



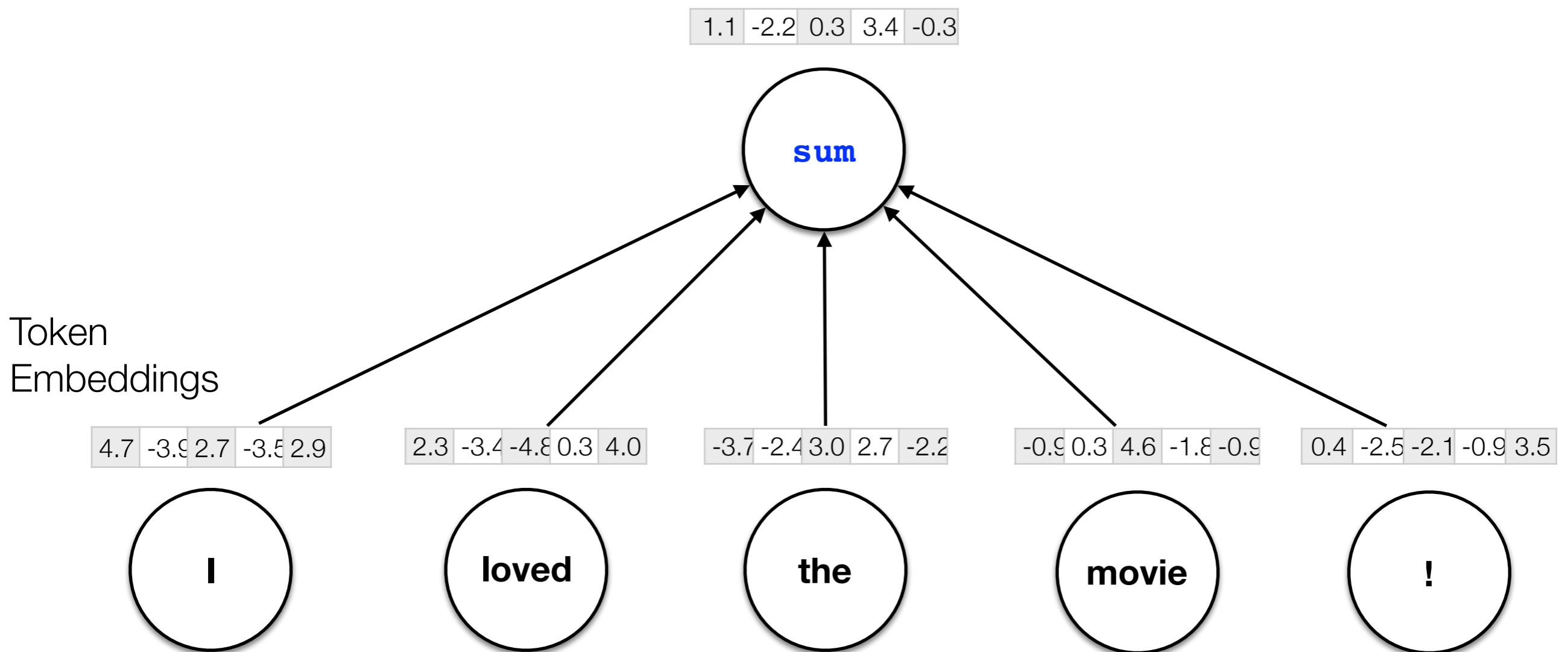
-0.9	0.3	4.6	-1.8	-0.9
------	-----	-----	------	------



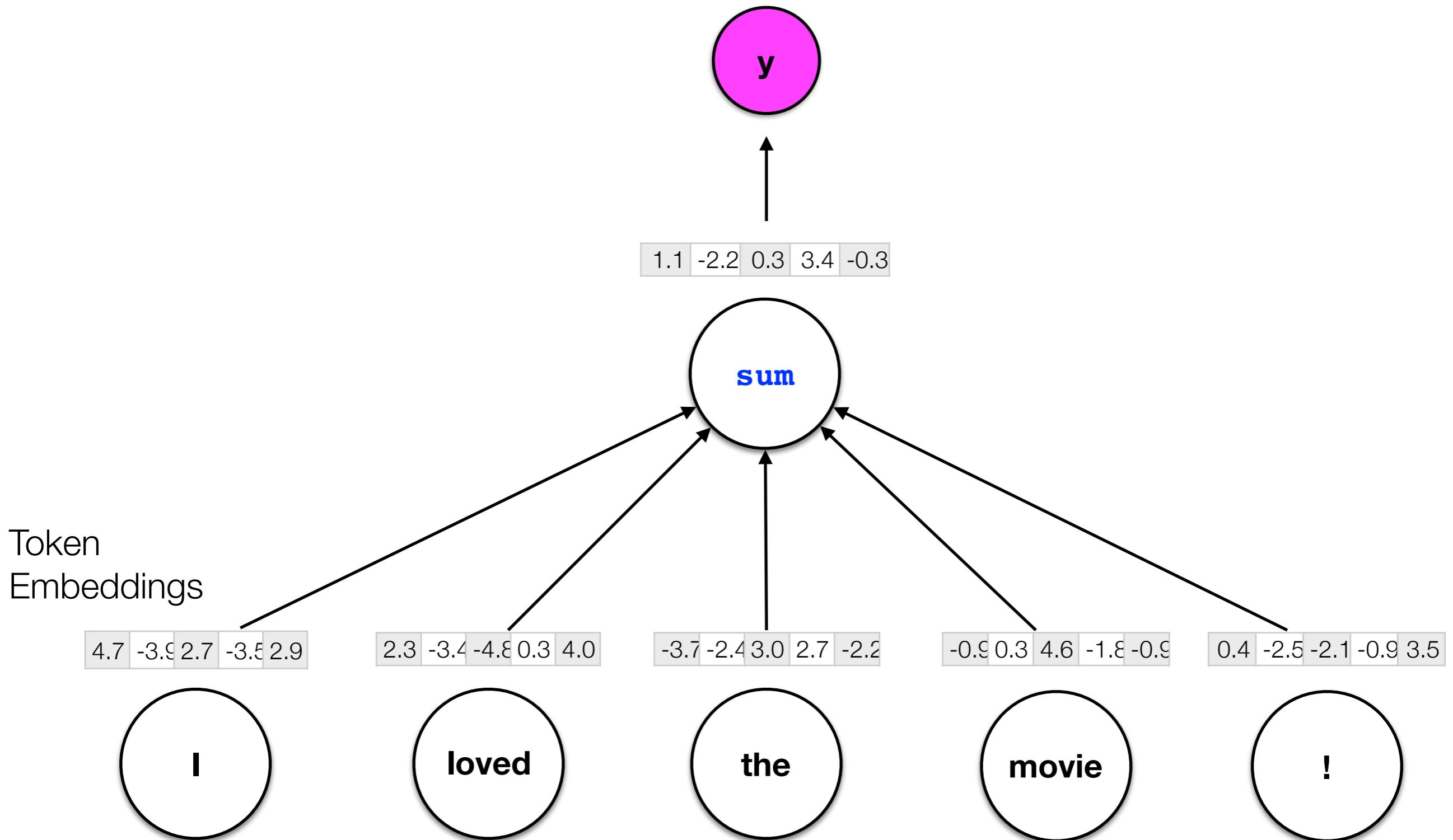
0.4	-2.5	-2.1	-0.9	3.5
-----	------	------	------	-----



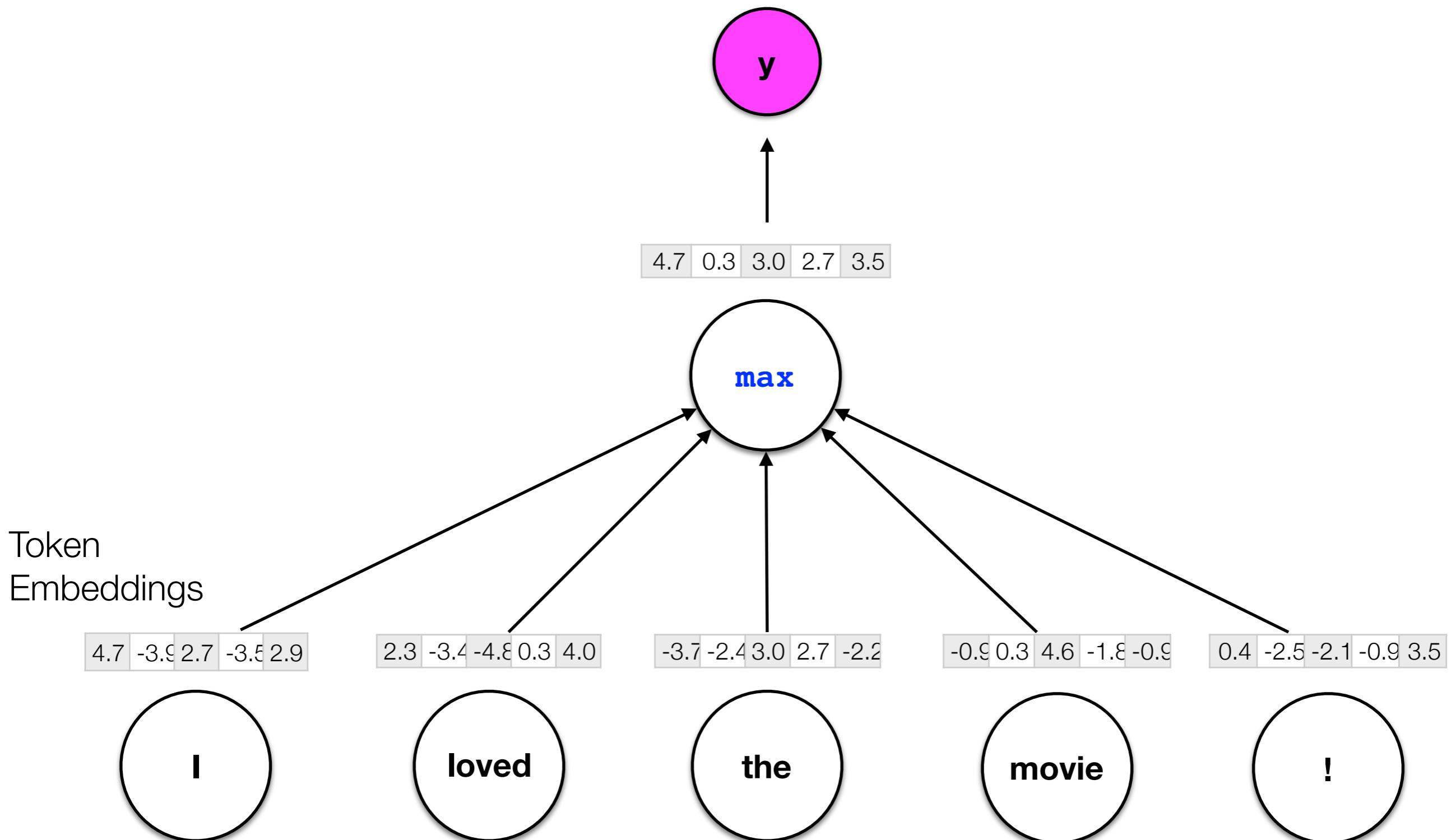
# How could we use word embeddings for document classification?



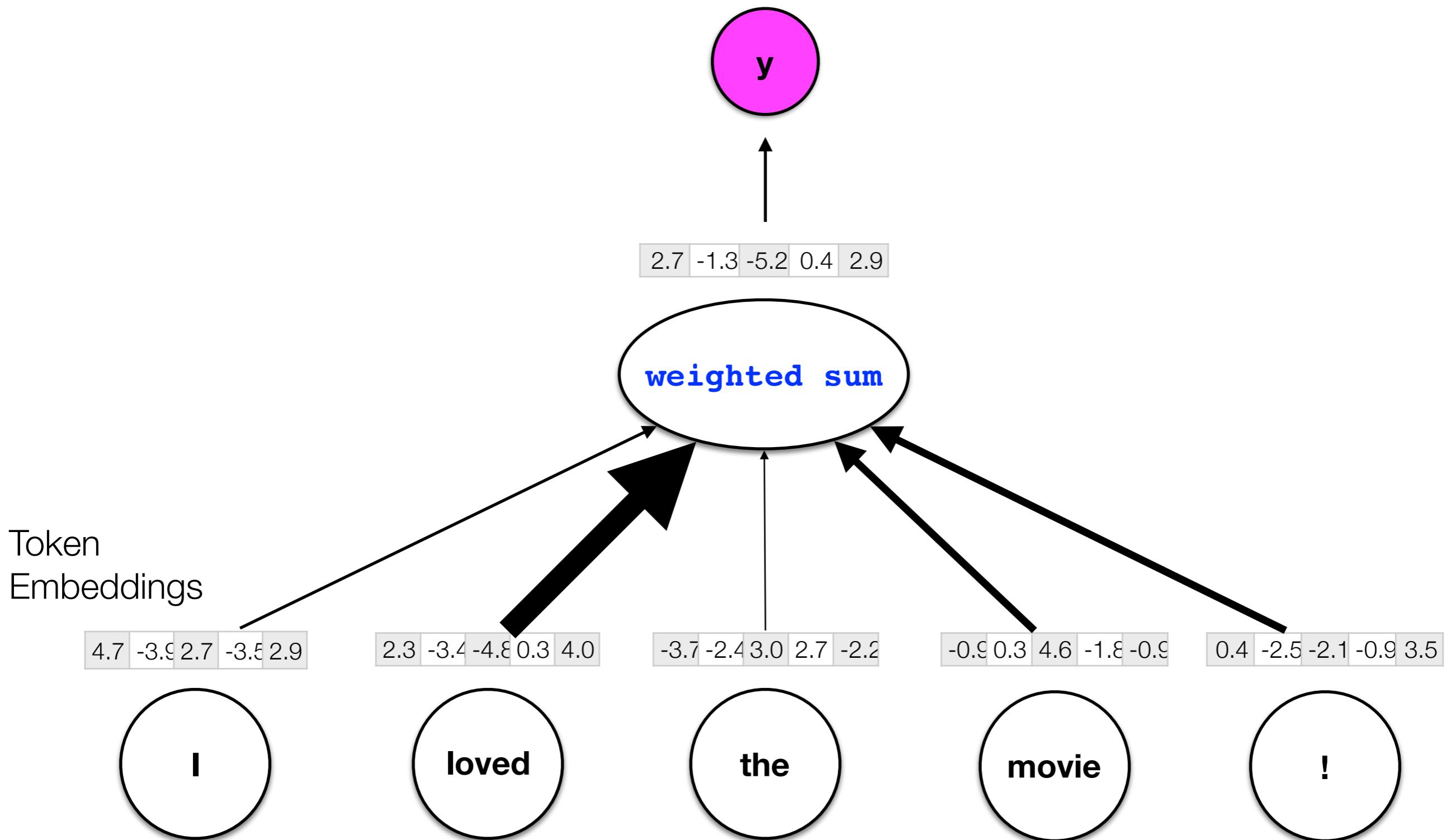
# How could we use word embeddings for document classification?



# How could we use word embeddings for document classification?



# How could we use word embeddings for document classification?



# Attention

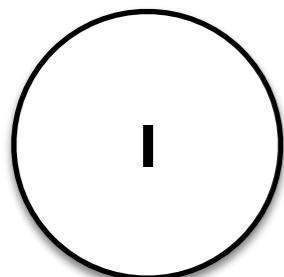
- Let's incorporate structure (and parameters) into a network that captures which elements in the input we should be **attending** to (and which we can ignore).

$v \in \mathbf{R}^H$

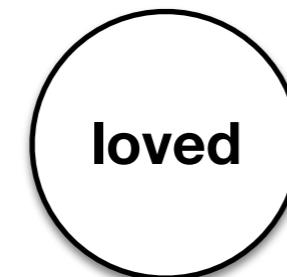
2.7	3.1	-1.4	2.3	0.7
-----	-----	------	-----	-----

Define  $v$  to be a vector to be learned; think of it as an “important word” vector. The dot product here measures how similar each input vector is to that “important word” vector

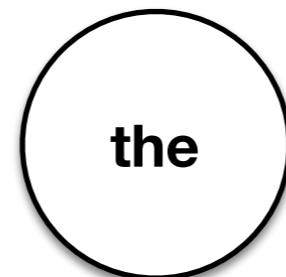
4.7	-3.9	2.7	-3.5	2.9
-----	------	-----	------	-----



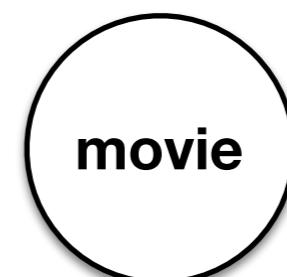
2.3	-3.4	-4.8	0.3	4.0
-----	------	------	-----	-----



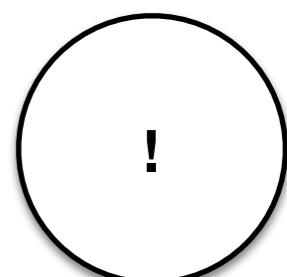
-3.7	-2.4	3.0	2.7	-2.2
------	------	-----	-----	------



-0.9	0.3	4.6	-1.8	-0.9
------	-----	-----	------	------



0.4	-2.5	-2.1	-0.9	3.5
-----	------	------	------	-----



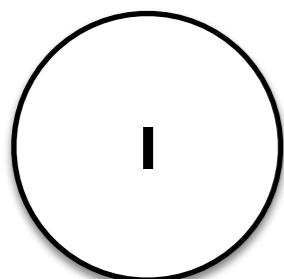
$v \in \mathbf{R}^H$

2.7	3.1	-1.4	2.3	0.7
-----	-----	------	-----	-----

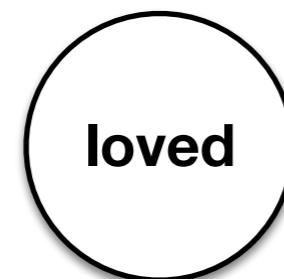
Define  $v$  to be a vector to be learned; think of it as an “important word” vector. The dot product here measures how similar each input vector is to that “important word” vector

$r_1 = v^\top x_1$

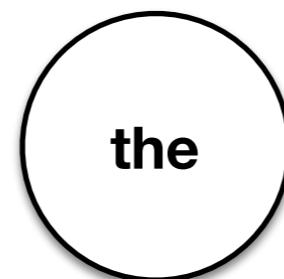
4.7	-3.9	2.7	-3.5	2.9
-----	------	-----	------	-----



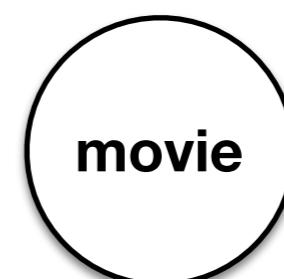
2.3	-3.4	-4.8	0.3	4.0
-----	------	------	-----	-----



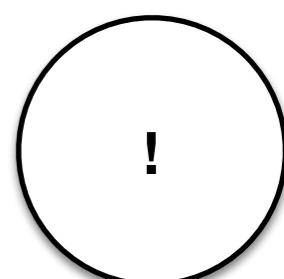
-3.7	-2.4	3.0	2.7	-2.2
------	------	-----	-----	------



-0.9	0.3	4.6	-1.8	-0.9
------	-----	-----	------	------



0.4	-2.5	-2.1	-0.9	3.5
-----	------	------	------	-----



$v \in \mathbf{R}^H$

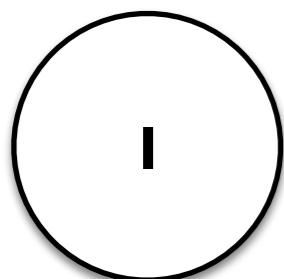
2.7	3.1	-1.4	2.3	0.7
-----	-----	------	-----	-----

Define  $v$  to be a vector to be learned; think of it as an “important word” vector. The dot product here measures how similar each input vector is to that “important word” vector

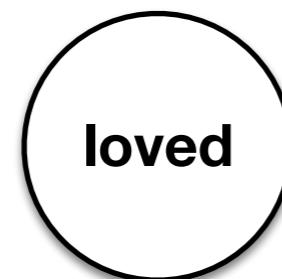
-3.4

$r_1 = v^\top x_1$

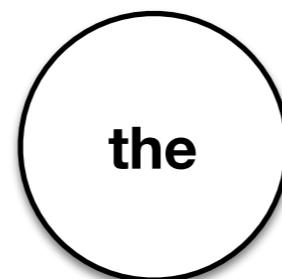
4.7	-3.9	2.7	-3.5	2.9
-----	------	-----	------	-----



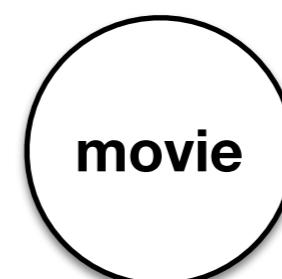
2.3	-3.4	-4.8	0.3	4.0
-----	------	------	-----	-----



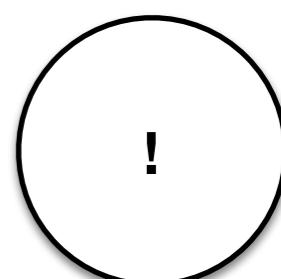
-3.7	-2.4	3.0	2.7	-2.2
------	------	-----	-----	------



-0.9	0.3	4.6	-1.8	-0.9
------	-----	-----	------	------



0.4	-2.5	-2.1	-0.9	3.5
-----	------	------	------	-----



$v \in \mathbf{R}^H$

2.7	3.1	-1.4	2.3	0.7
-----	-----	------	-----	-----

Define  $v$  to be a vector to be learned; think of it as an “important word” vector. The dot product here measures how similar each input vector is to that “important word” vector

-3.4

2.4

$r_1 = v^\top x_1$

$r_2 = v^\top x_2$

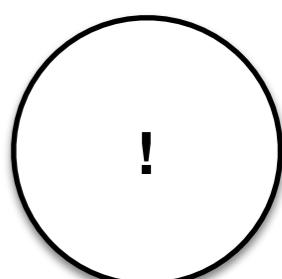
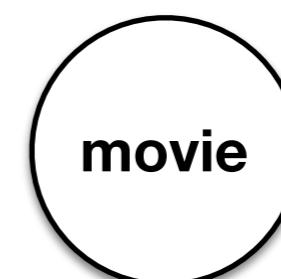
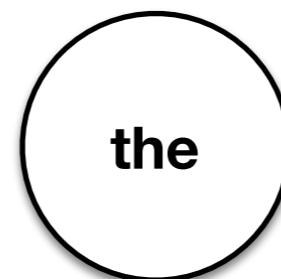
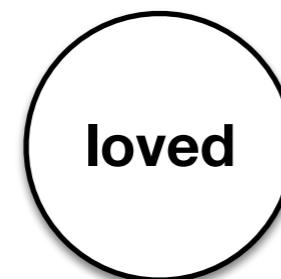
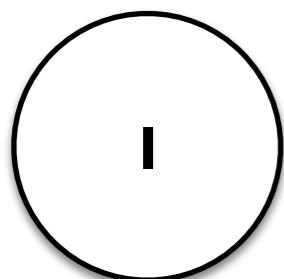
4.7	-3.9	2.7	-3.5	2.9
-----	------	-----	------	-----

2.3	-3.4	-4.8	0.3	4.0
-----	------	------	-----	-----

-3.7	-2.4	3.0	2.7	-2.2
------	------	-----	-----	------

-0.9	0.3	4.6	-1.8	-0.9
------	-----	-----	------	------

0.4	-2.5	-2.1	-0.9	3.5
-----	------	------	------	-----



$v \in \mathbf{R}^H$

2.7	3.1	-1.4	2.3	0.7
-----	-----	------	-----	-----

Define  $v$  to be a vector to be learned; think of it as an “important word” vector. The dot product here measures how similar each input vector is to that “important word” vector

-3.4

2.4

-0.8

-1.2

1.7

$r_1 = v^\top x_1$

$r_2 = v^\top x_2$

$r_3 = v^\top x_3$

$r_4 = v^\top x_4$

$r_5 = v^\top x_5$

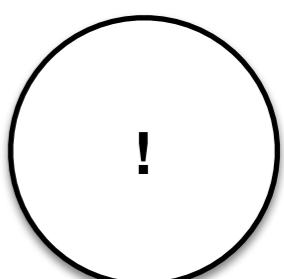
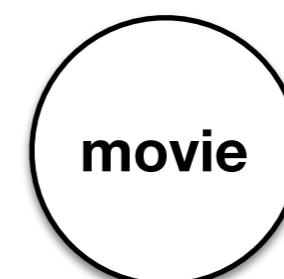
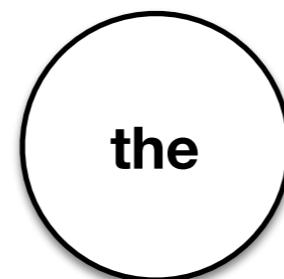
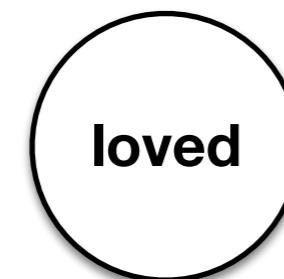
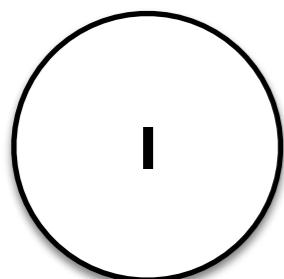
4.7	-3.9	2.7	-3.5	2.9
-----	------	-----	------	-----

2.3	-3.4	-4.8	0.3	4.0
-----	------	------	-----	-----

-3.7	-2.4	3.0	2.7	-2.2
------	------	-----	-----	------

-0.9	0.3	4.6	-1.8	-0.9
------	-----	-----	------	------

0.4	-2.5	-2.1	-0.9	3.5
-----	------	------	------	-----



Convert  $r$  into a vector of normalized weights that sum to 1.

$$r_1 = v^\top x_1$$

$$r_2 = v^\top x_2$$

$$r_3 = v^\top x_3$$

$$r_4 = v^\top x_4$$

$$r_5 = v^\top x_5$$

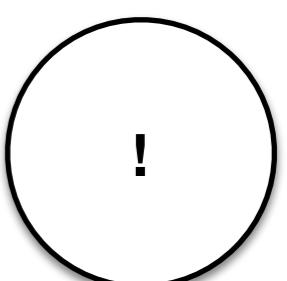
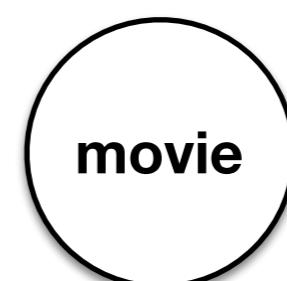
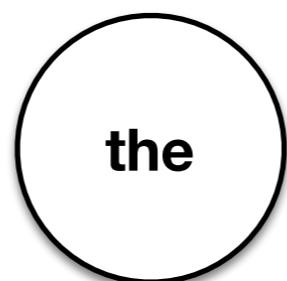
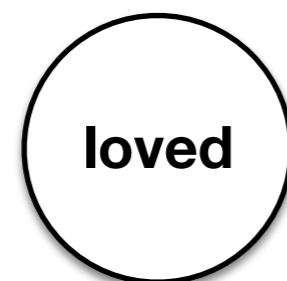
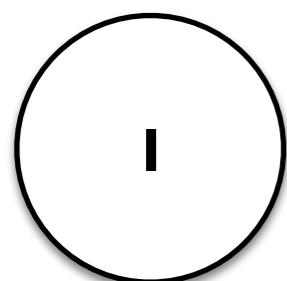
4.7	-3.9	2.7	-3.5	2.9
-----	------	-----	------	-----

2.3	-3.4	-4.8	0.3	4.0
-----	------	------	-----	-----

-3.7	-2.4	3.0	2.7	-2.2
------	------	-----	-----	------

-0.9	0.3	4.6	-1.8	-0.9
------	-----	-----	------	------

0.4	-2.5	-2.1	-0.9	3.5
-----	------	------	------	-----



Convert  $r$  into a vector of normalized weights that sum to 1.

$r$

-3.4	2.4	-0.8	-1.2	1.7
------	-----	------	------	-----

$$r_1 = v^\top x_1$$

$$r_2 = v^\top x_2$$

$$r_3 = v^\top x_3$$

$$r_4 = v^\top x_4$$

$$r_5 = v^\top x_5$$

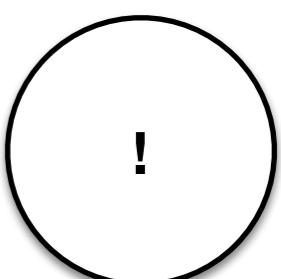
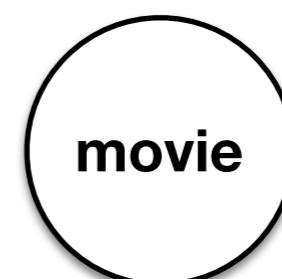
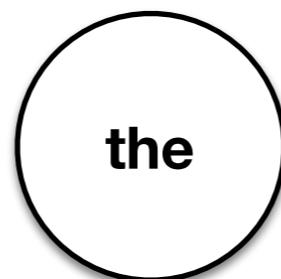
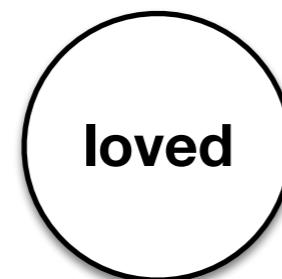
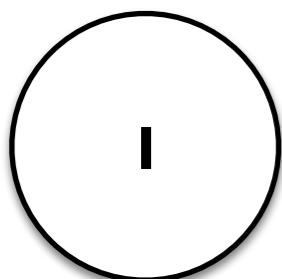
4.7	-3.9	2.7	-3.5	2.9
-----	------	-----	------	-----

2.3	-3.4	-4.8	0.3	4.0
-----	------	------	-----	-----

-3.7	-2.4	3.0	2.7	-2.2
------	------	-----	-----	------

-0.9	0.3	4.6	-1.8	-0.9
------	-----	-----	------	------

0.4	-2.5	-2.1	-0.9	3.5
-----	------	------	------	-----



Convert  $r$  into a vector of normalized weights that sum to 1.

$$a = \text{softmax}(r)$$

$r$

-3.4	2.4	-0.8	-1.2	1.7
------	-----	------	------	-----

$$r_1 = v^\top x_1$$

$$r_2 = v^\top x_2$$

$$r_3 = v^\top x_3$$

$$r_4 = v^\top x_4$$

$$r_5 = v^\top x_5$$

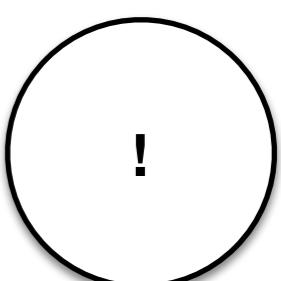
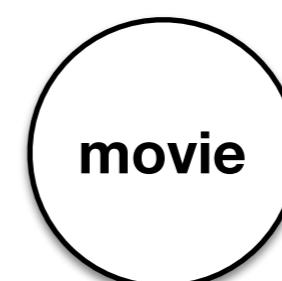
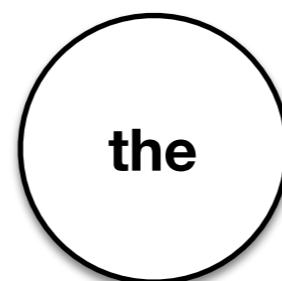
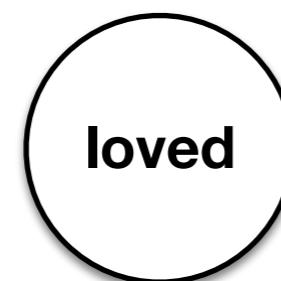
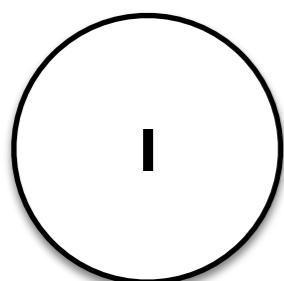
4.7	-3.9	2.7	-3.5	2.9
-----	------	-----	------	-----

2.3	-3.4	-4.8	0.3	4.0
-----	------	------	-----	-----

-3.7	-2.4	3.0	2.7	-2.2
------	------	-----	-----	------

-0.9	0.3	4.6	-1.8	-0.9
------	-----	-----	------	------

0.4	-2.5	-2.1	-0.9	3.5
-----	------	------	------	-----



Convert  $r$  into a vector of normalized weights that sum to 1.

$$a = \text{softmax}(r)$$

$a$	0	0.64	0.02	0.02	0.32
$r$	-3.4	2.4	-0.8	-1.2	1.7

$$r_1 = v^\top x_1$$

$$r_2 = v^\top x_2$$

$$r_3 = v^\top x_3$$

$$r_4 = v^\top x_4$$

$$r_5 = v^\top x_5$$

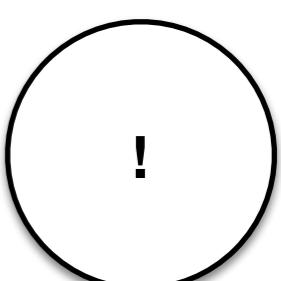
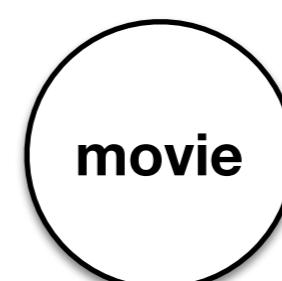
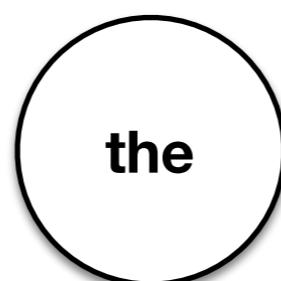
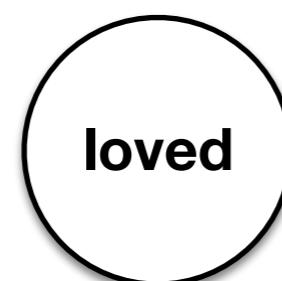
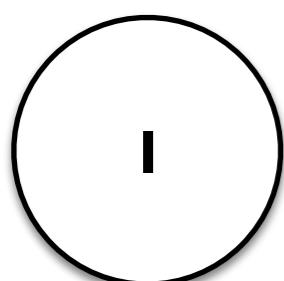
4.7	-3.9	2.7	-3.5	2.9
-----	------	-----	------	-----

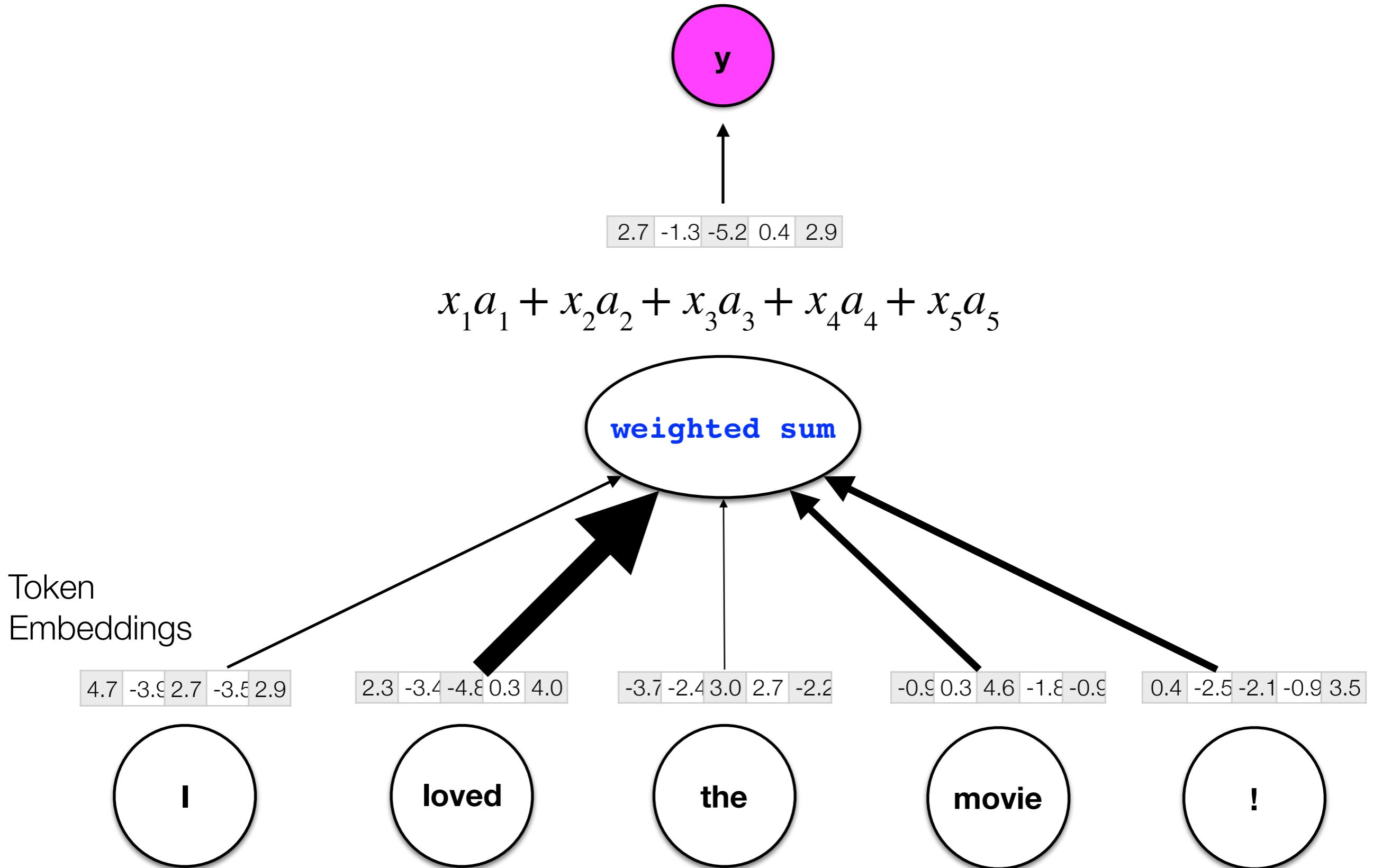
2.3	-3.4	-4.8	0.3	4.0
-----	------	------	-----	-----

-3.7	-2.4	3.0	2.7	-2.2
------	------	-----	-----	------

-0.9	0.3	4.6	-1.8	-0.9
------	-----	-----	------	------

0.4	-2.5	-2.1	-0.9	3.5
-----	------	------	------	-----



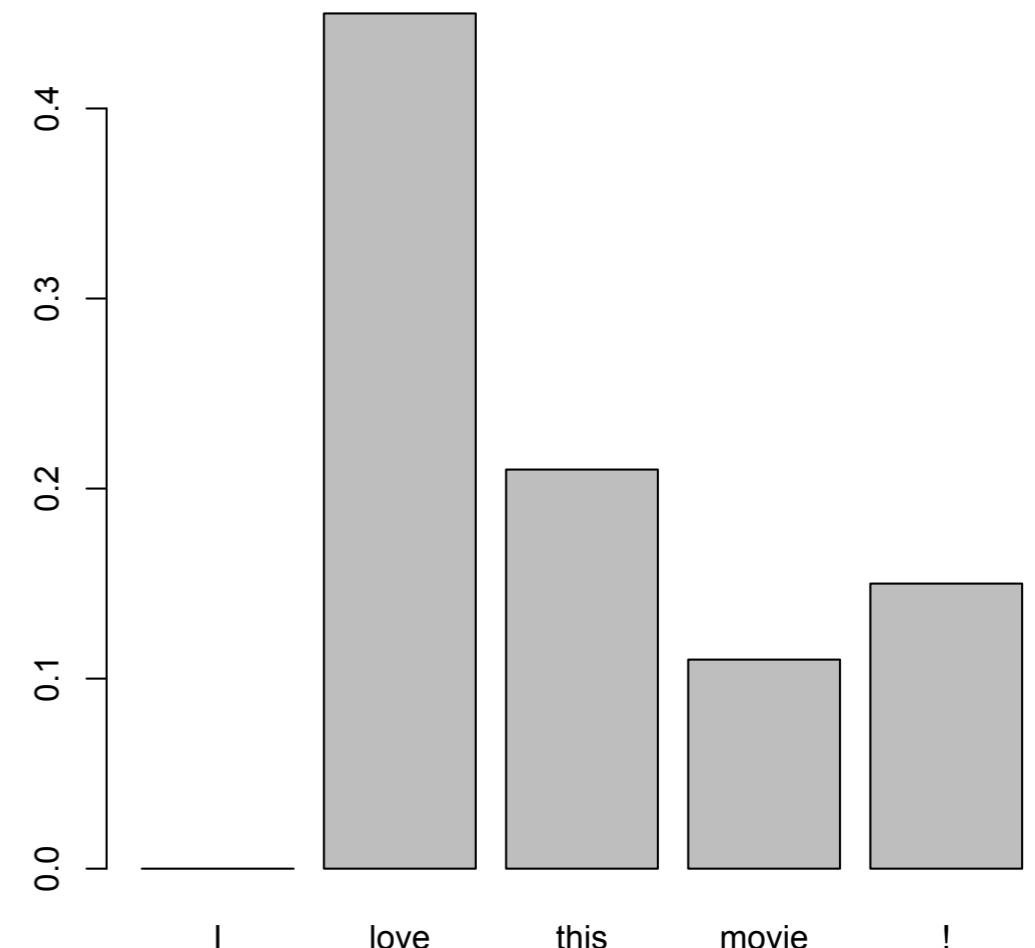


# Attention

- Lots of variations on attention:
  - Linear transformation of  $x$  into before dotting with  $v$
  - Non-linearities after each operation.
  - “Multi-head attention”: multiple  $v$  vectors to capture different phenomena that can be attended to in the input.
  - Hierarchical attention (sentence representation with attention over words + document representation with attention over sentences).

# Attention

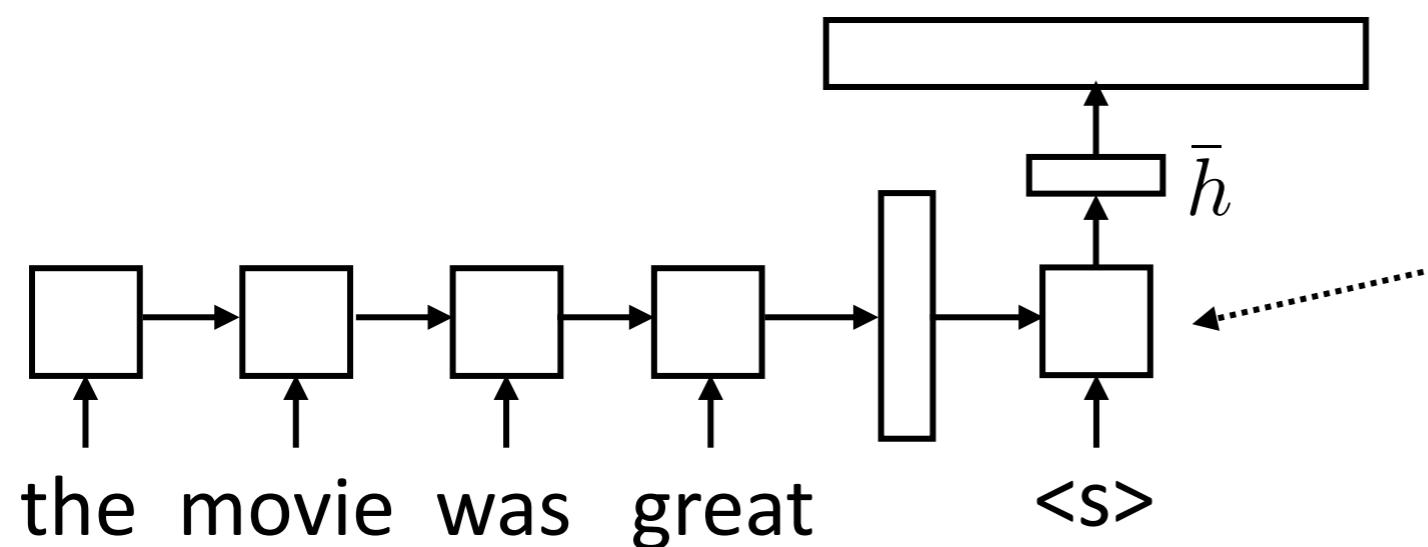
- Attention gives us a normalized weight for every token in a sequence that tells us how important that word was for the prediction
- This can be useful for visualization



# Attention in Networks

# Sequence to Sequence (seq2seq) models

- Generate next word conditioned on previous word as well as hidden state
- W size is  $|\text{vocab}| \times |\text{hidden state}|$ , softmax over entire vocabulary



Decoder has separate parameters from encoder, so this can learn to be a language model (produce a plausible next word given current one)

# Problems with seq2seq models

- Encoder-decoder models like to repeat themselves:

# Problems with seq2seq models

- Encoder-decoder models like to repeat themselves:

Un garçon joue dans la neige → A boy plays in the snow boy plays boy plays

# Problems with seq2seq models

- Encoder-decoder models like to repeat themselves:

Un garçon joue dans la neige → A boy plays in the snow boy plays boy plays

- Why does this happen?
  - Models trained poorly
  - LSTM state is not behaving as expected so it gets stuck in a “loop” of generating the same output tokens again and again

# Problems with seq2seq models

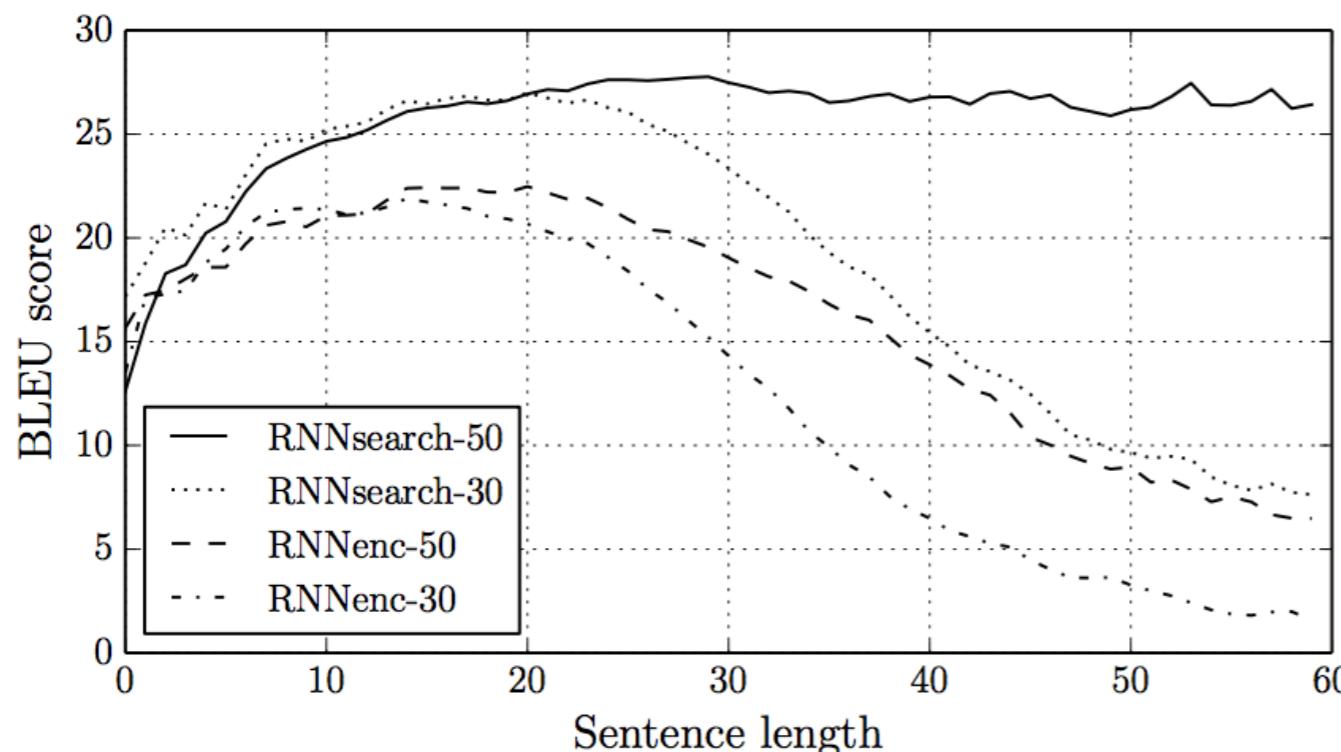
- Encoder-decoder models like to repeat themselves:

Un garçon joue dans la neige → A boy plays in the snow boy plays boy plays

- Why does this happen?
  - Models trained poorly
  - LSTM state is not behaving as expected so it gets stuck in a “loop” of generating the same output tokens again and again
- Need some notion of input coverage or what input words we’ve translated

# Problems with seq2seq models

- Bad at long sentences: 1) a fixed-size hidden representation doesn't scale; 2) LSTMs still have a hard time remembering for really long periods of time



Bahdanau et al. (2014)

# Problems with seq2seq models

- Unknown words:

en: The ecotax portico in Pont-de-Buis , ... [truncated] ..., was taken down on Thursday morning

fr: Le portique écotaxe de Pont-de-Buis , ... [truncated] ..., a été démonté jeudi matin

nn: Le unk de unk à unk , ... [truncated] ..., a été pris le jeudi matin

# Problems with seq2seq models

- Unknown words:

en: The ecotax portico in Pont-de-Buis , ... [truncated] ..., was taken down on Thursday morning

fr: Le portique écotaxe de Pont-de-Buis , ... [truncated] ..., a été démonté jeudi matin

nn: Le unk de unk à unk , ... [truncated] ..., a été pris le jeudi matin

- Encoding these rare words into a vector space is really hard

# Problems with seq2seq models

- Unknown words:

en: The ecotax portico in Pont-de-Buis , ... [truncated] ..., was taken down on Thursday morning

fr: Le portique écotaxe de Pont-de-Buis , ... [truncated] ..., a été démonté jeudi matin

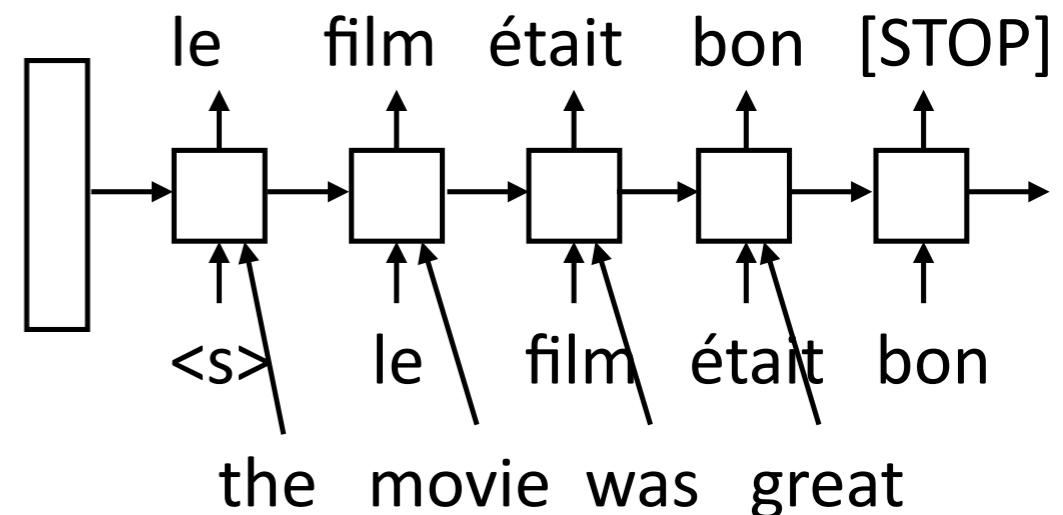
nn: Le unk de unk à unk , ... [truncated] ..., a été pris le jeudi matin

- Encoding these rare words into a vector space is really hard
- In fact, we don't want to encode them, we want a way of directly looking back at the input and copying them (Pont-de-Buis)

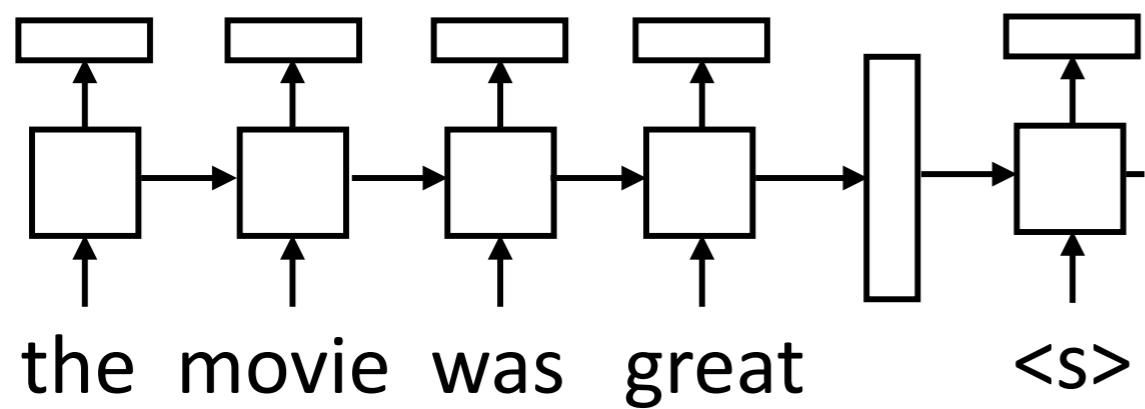
# Aligned Input

- Suppose we knew the source and target would be word-by-word translated
- Can look at the corresponding input word when translating — this could scale!
- Much less burden on the hidden state
- How can we achieve this without hardcoding it?

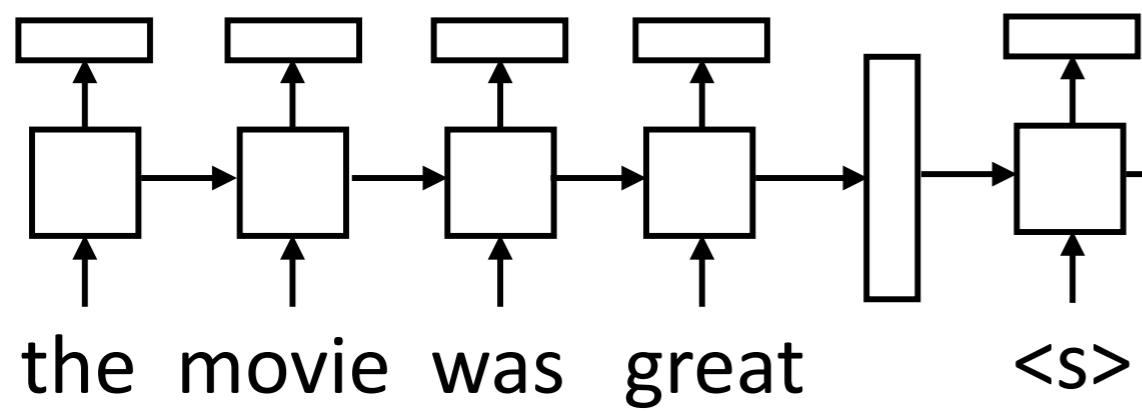
the movie was great  
| / / / /  
le film était bon



What if we could **attend** to specific input tokens when **decoding** the next output?

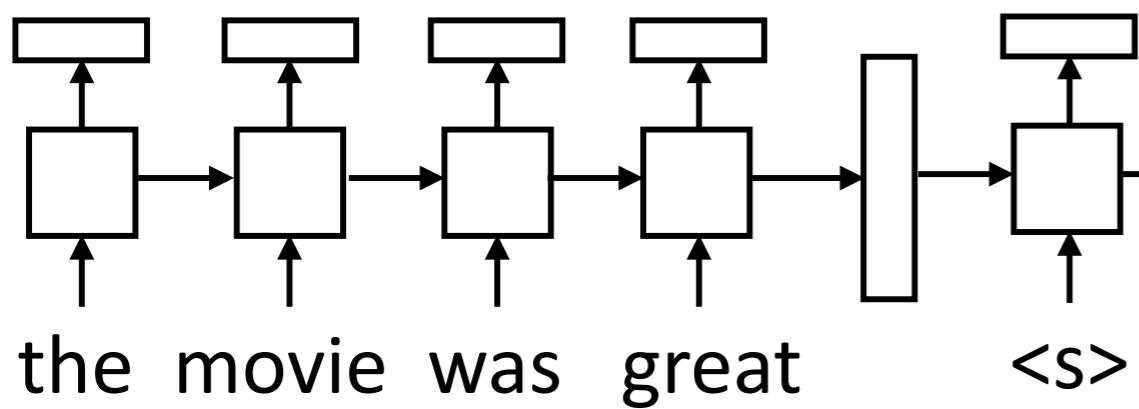
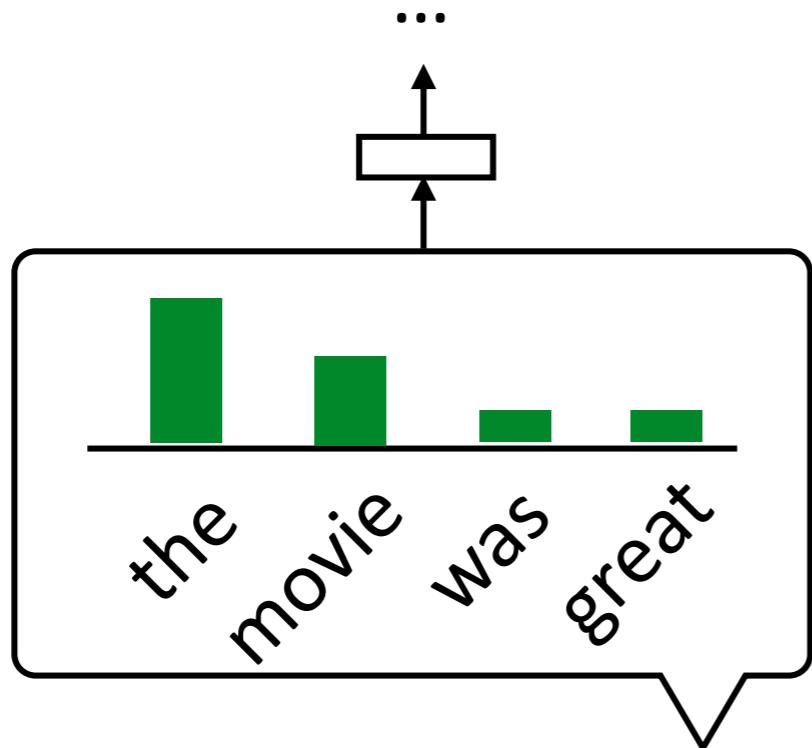


What if we could **attend** to specific input tokens when **decoding** the next output?



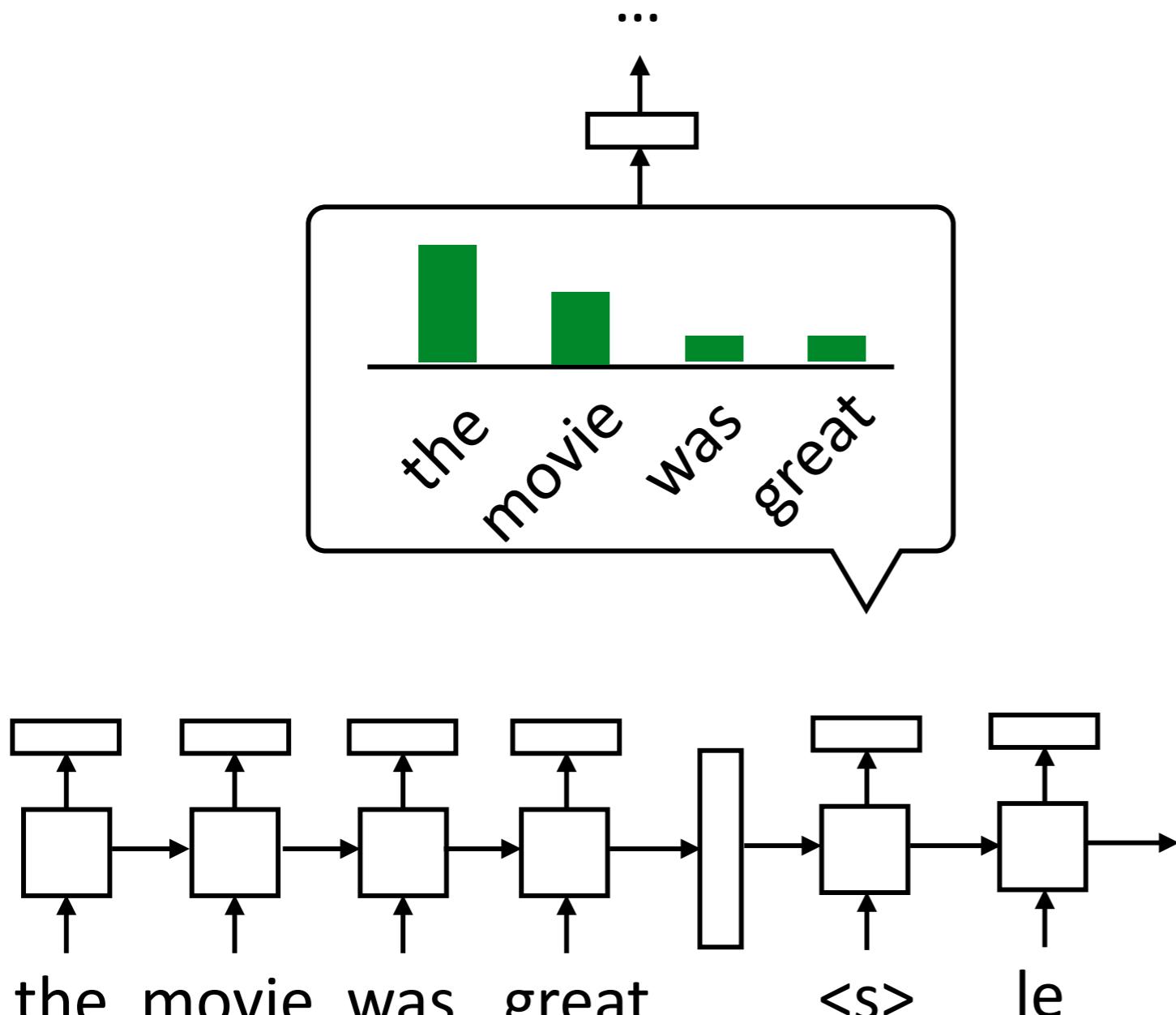
At each decoder state,  
compute a distribution  
over source inputs  
based on current  
decoder state, use that  
in output layer

What if we could **attend** to specific input tokens when **decoding** the next output?



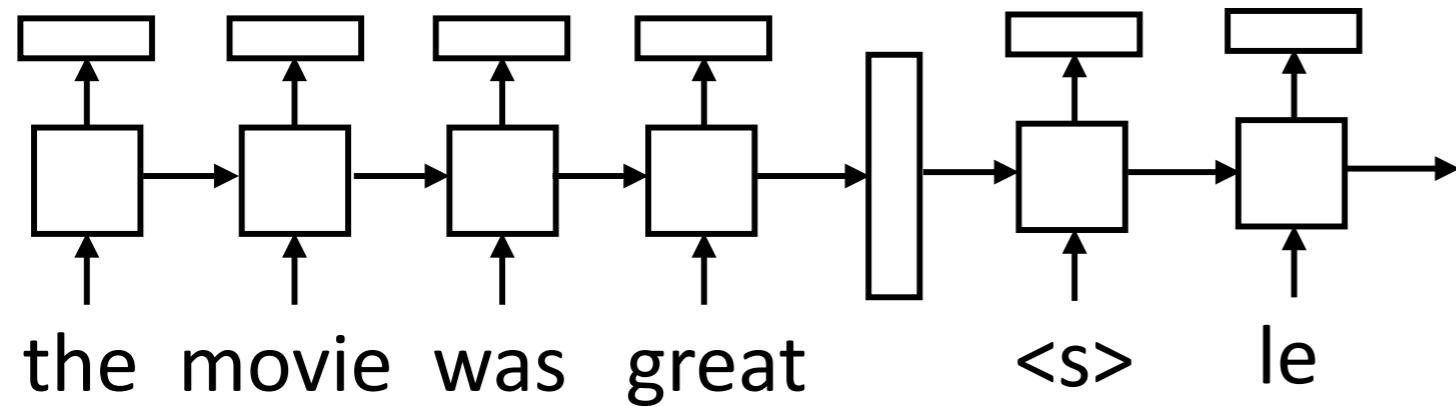
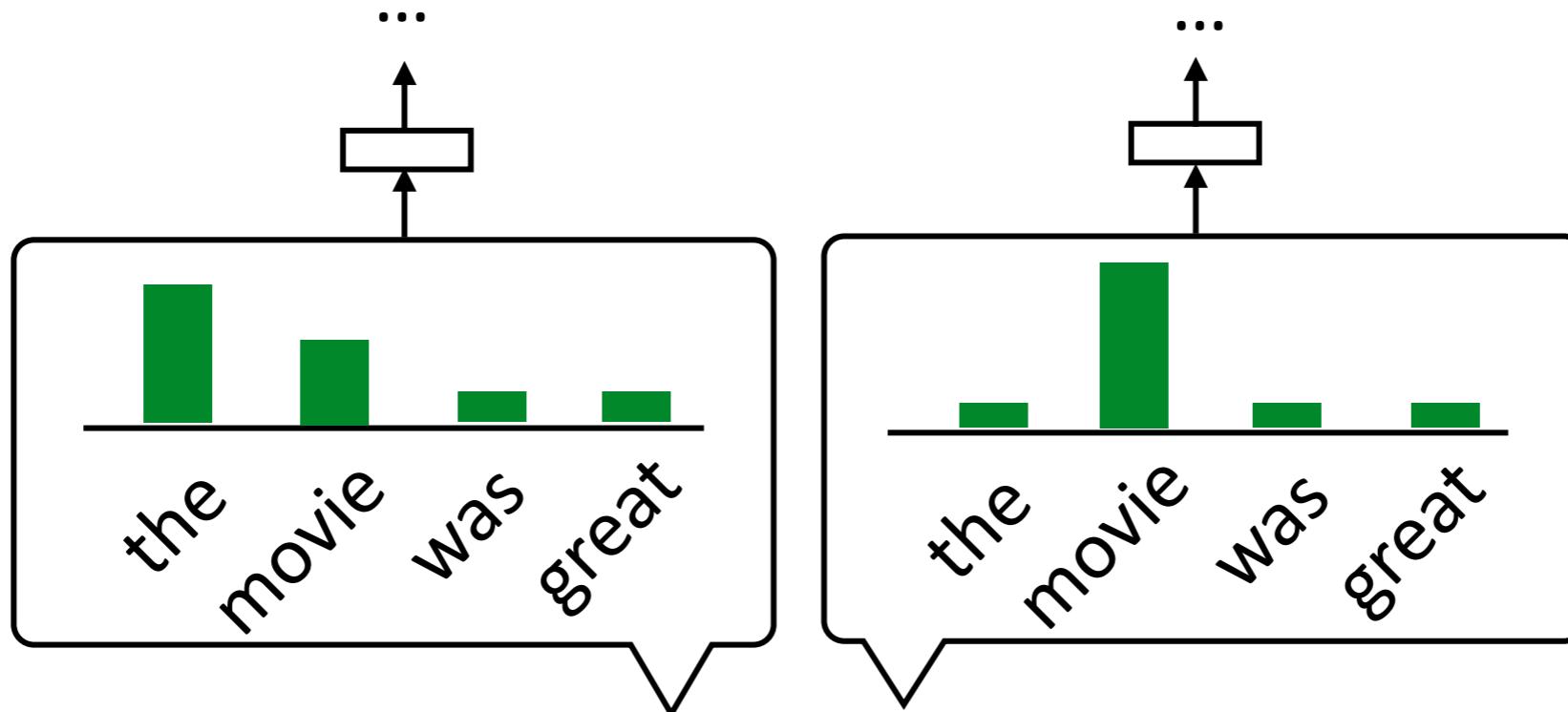
At each decoder state,  
compute a distribution  
over source inputs  
based on current  
decoder state, use that  
in output layer

What if we could **attend** to specific input tokens when **decoding** the next output?



At each decoder state,  
compute a distribution  
over source inputs  
based on current  
decoder state, use that  
in output layer

What if we could **attend** to specific input tokens when **decoding** the next output?



At each decoder state,  
compute a distribution  
over source inputs  
based on current  
decoder state, use that  
in output layer

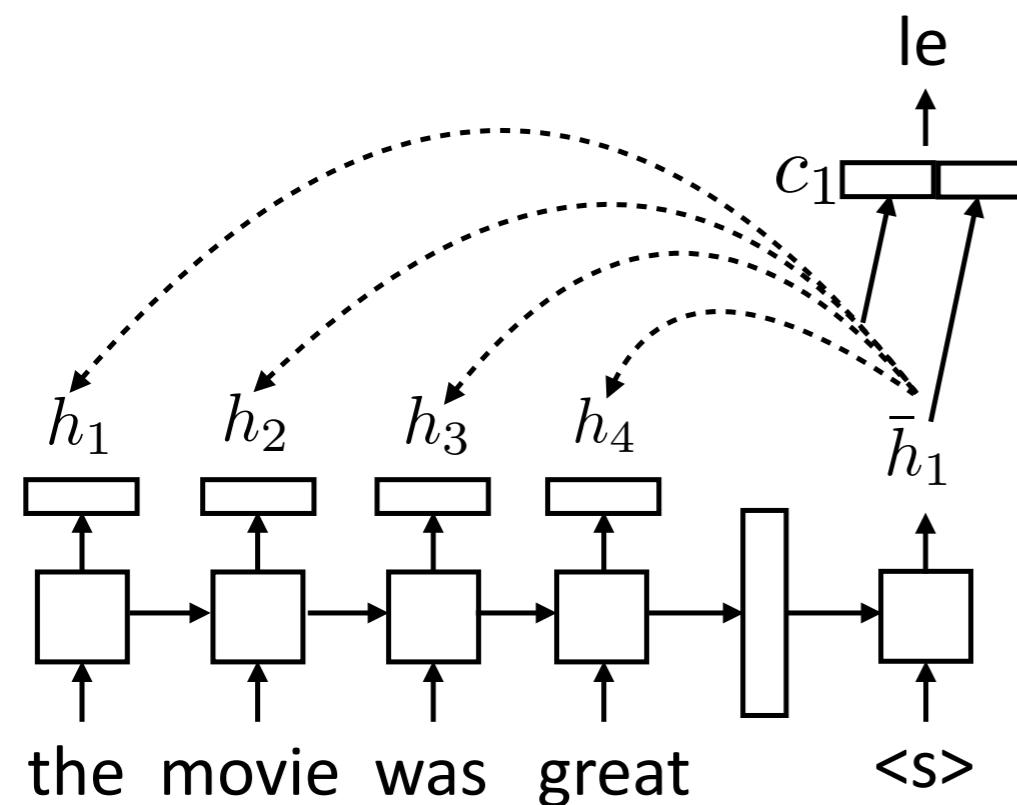
# How attention works at the math level

If there was no attention:  $P(y_i|x, y_1, \dots, y_{i-1}) = \text{softmax}(Wh_i)$

# How attention works at the math level

If there was no attention:  $P(y_i|x, y_1, \dots, y_{i-1}) = \text{softmax}(W h_i)$

With attention: for each decoder state, compute weighted sum of input states



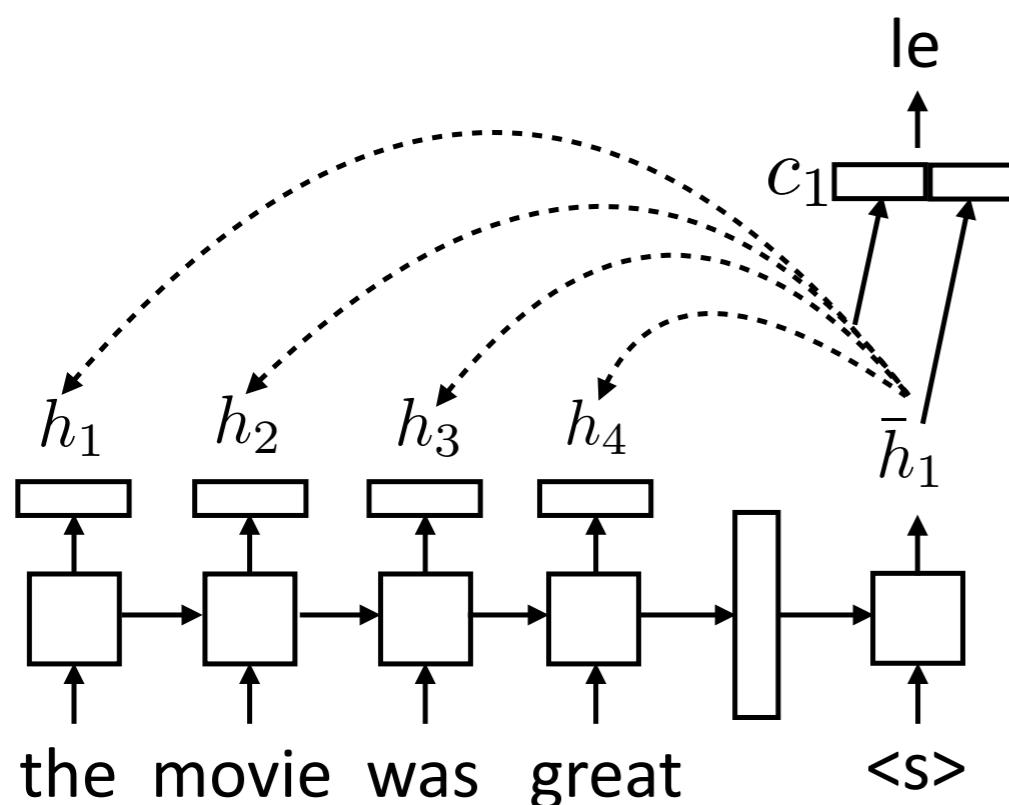
$P(y_i|x, y_1, \dots, y_{i-1}) = \text{softmax}(W[c_i; h_i])$

# How attention works at the math level

If there was no attention:  $P(y_i|x, y_1, \dots, y_{i-1}) = \text{softmax}(W[h_i])$

With attention: for each decoder state, compute weighted sum of input states

$P(y_i|x, y_1, \dots, y_{i-1}) = \text{softmax}(W[c_i; h_i])$



$$P(y_i|\mathbf{x}, y_1, \dots, y_{i-1})$$

$$c_i = \sum_j \alpha_{ij} h_j$$

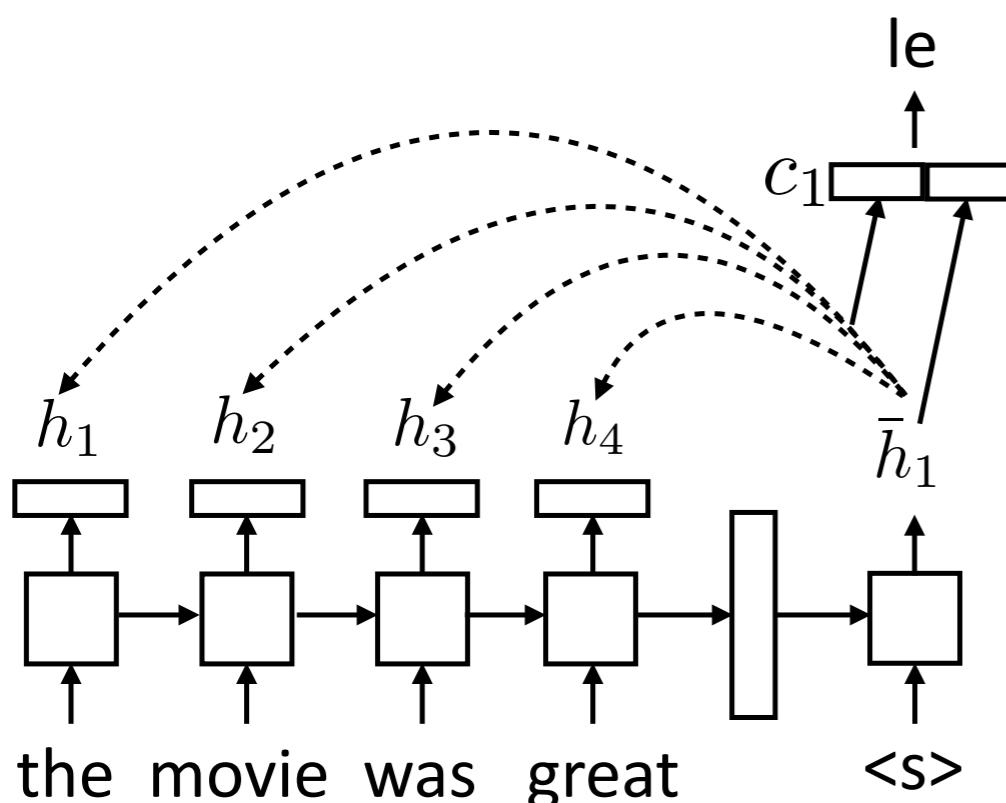
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

$$e_{ij} = f(\bar{h}_i, h_j)$$

# How attention works at the math level

If there was no attention:  $P(y_i|x, y_1, \dots, y_{i-1}) = \text{softmax}(W[h_i])$

With attention: for each decoder state, compute weighted sum of input states



$P(y_i|x, y_1, \dots, y_{i-1}) = \text{softmax}(W[c_i; h_i])$

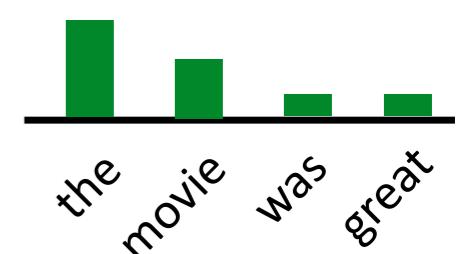
$$P(y_i|\mathbf{x}, y_1, \dots, y_{i-1})$$

$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

$$e_{ij} = f(\bar{h}_i, h_j)$$

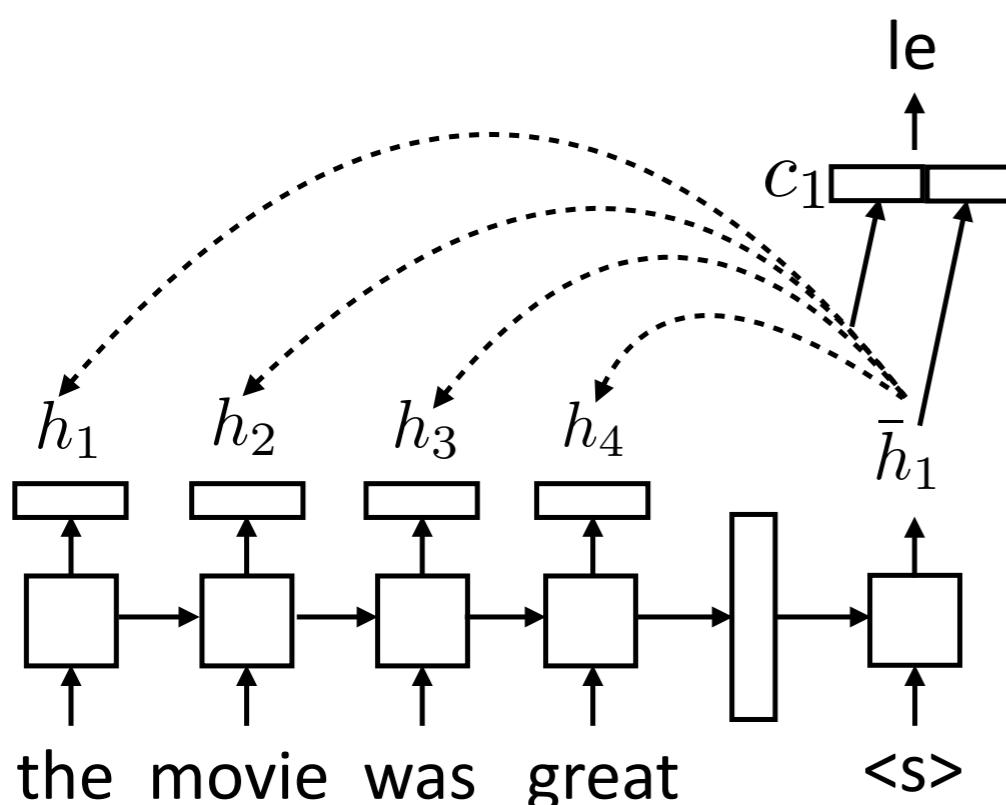
Weighted sum  
of input hidden  
states (vector)



# How attention works at the math level

If there was no attention:  $P(y_i|x, y_1, \dots, y_{i-1}) = \text{softmax}(W[h_i])$

With attention: for each decoder state, compute weighted sum of input states



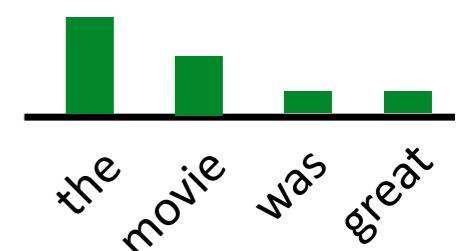
$$P(y_i|x, y_1, \dots, y_{i-1}) = \text{softmax}(W[c_i; h_i])$$

$$P(y_i|\mathbf{x}, y_1, \dots, y_{i-1})$$

$$c_i = \sum_j \alpha_{ij} h_j$$

Weighted sum  
of input hidden  
states (vector)

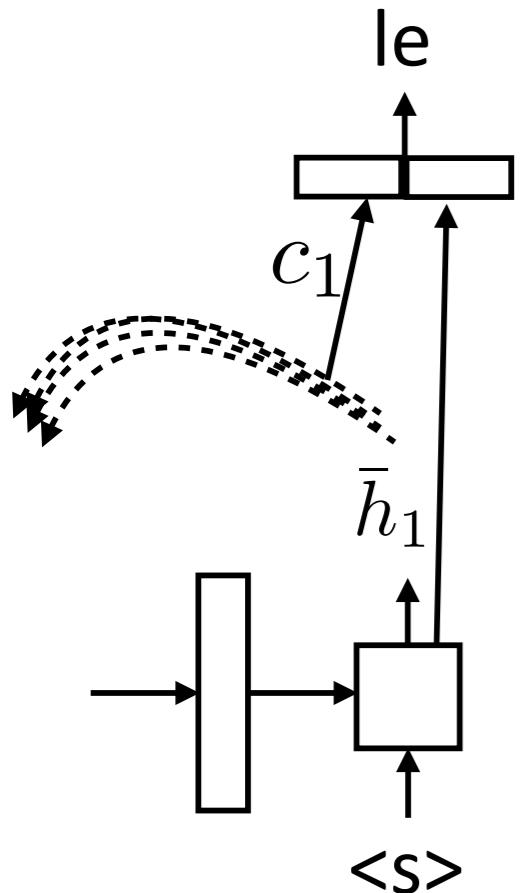
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$



$$e_{ij} = f(\bar{h}_i, h_j)$$

f is some  
function (TBD)

# How do we actually compute the attention?



$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

$$e_{ij} = f(\bar{h}_i, h_j)$$

$$f(\bar{h}_i, h_j) = \tanh(W[\bar{h}_i, h_j])$$

► Bahdanau+ (2014): additive

$$f(\bar{h}_i, h_j) = \bar{h}_i \cdot h_j$$

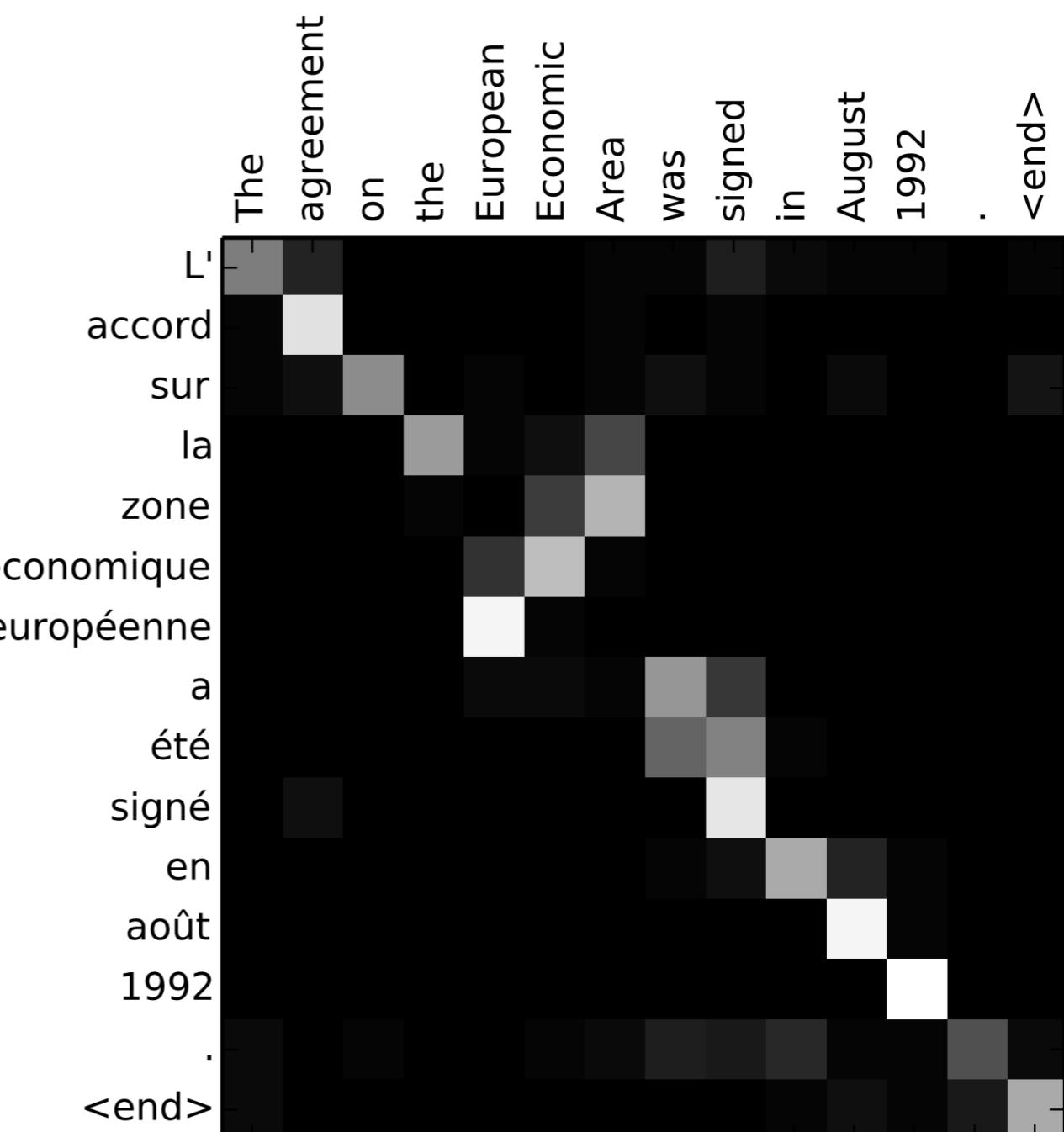
► Luong+ (2015): dot product

$$f(\bar{h}_i, h_j) = \bar{h}_i^\top W h_j$$

► Luong+ (2015): bilinear

# Attention can learn about positions and reordering

- Encoder hidden states capture contextual source word identity
- Decoder hidden states are now mostly responsible for selecting what to attend to
- Doesn't take a complex hidden state to walk monotonically through a sentence and spit out word-by-word translations





What would the fourth edition of your textbook include that isn't yet in the in the third version?

What would the fourth edition of your textbook include that isn't yet in the in the third version?

The third edition won't be done 'til next year and I still can't decide on the current topics. For example, Jim is writing the sequence modeling chapter right now using LSTMs, but of course the way people build sequence models will change, maybe Attention Is All You Need or maybe it will turn out we should have used dilated convolutions, or something else. So I'm not positive that the simplest, most general algorithm won't be something else by next year Sequence-to-sequence models is something that's changed a lot over time, from HMMs, to MEMMs, to CRFs, to RNNs.... Or maybe it'll turn out that really simple feed-forward networks that just walk across the input or something will work better, because somebody might come up with some simplification that makes it do so.

What would the fourth edition of your textbook include that isn't yet in the in the third version?

The third edition won't be done 'til next year and I still can't decide on the current topics. For example, Jim is writing the sequence modeling chapter right now using LSTMs, but of course the way people build sequence models will change, maybe *Attention Is All You Need* or maybe it will turn out we should have used dilated convolutions, or something else. So I'm not positive that the simplest, most general algorithm won't be something else by next year Sequence-to-sequence models is something that's changed a lot over time, from HMMs, to MEMMs, to CRFs, to RNNs.... Or maybe it'll turn out that really simple feed-forward networks that just walk across the input or something will work better, because somebody might come up with some simplification that makes it do so.

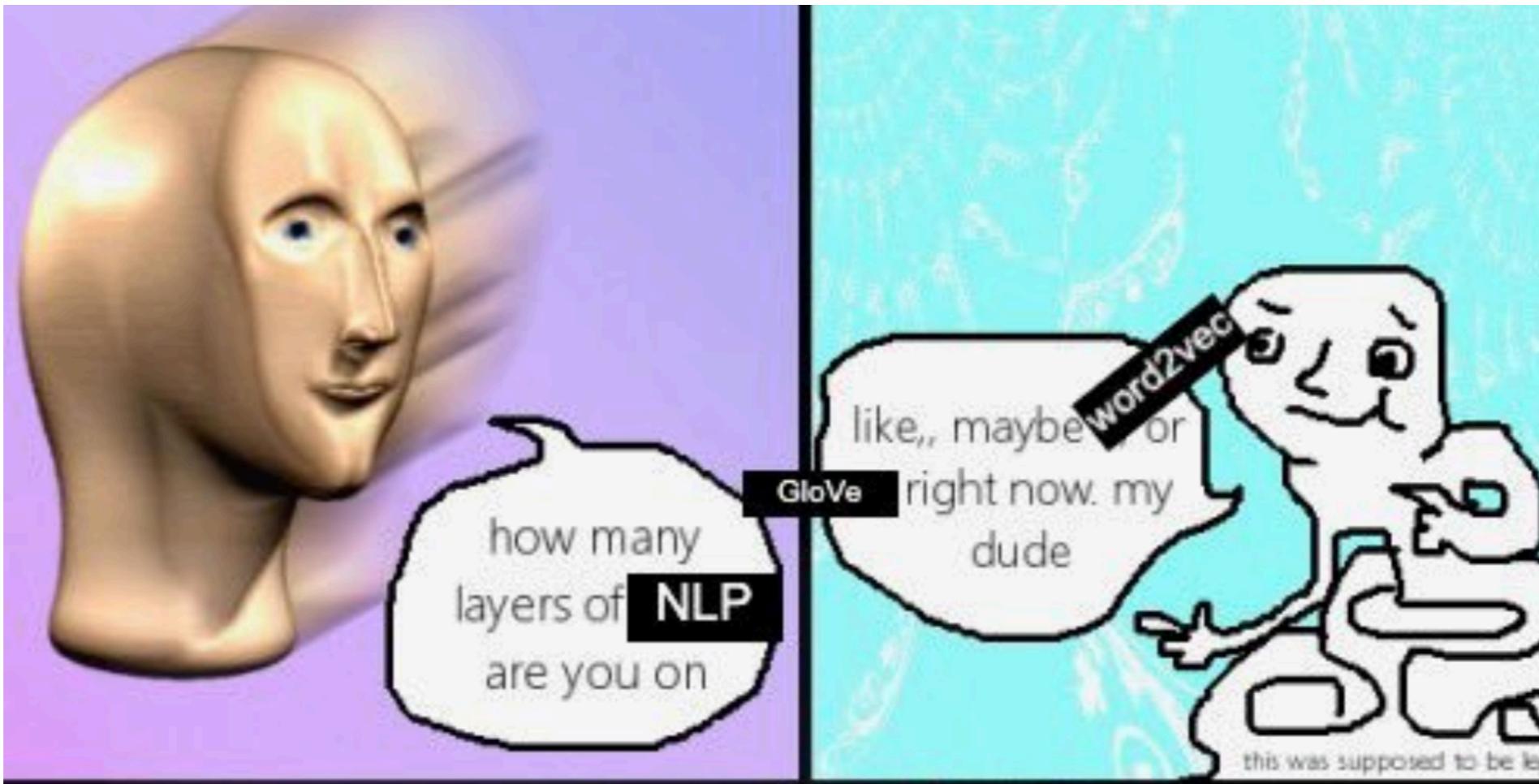


Image Credit: PBS

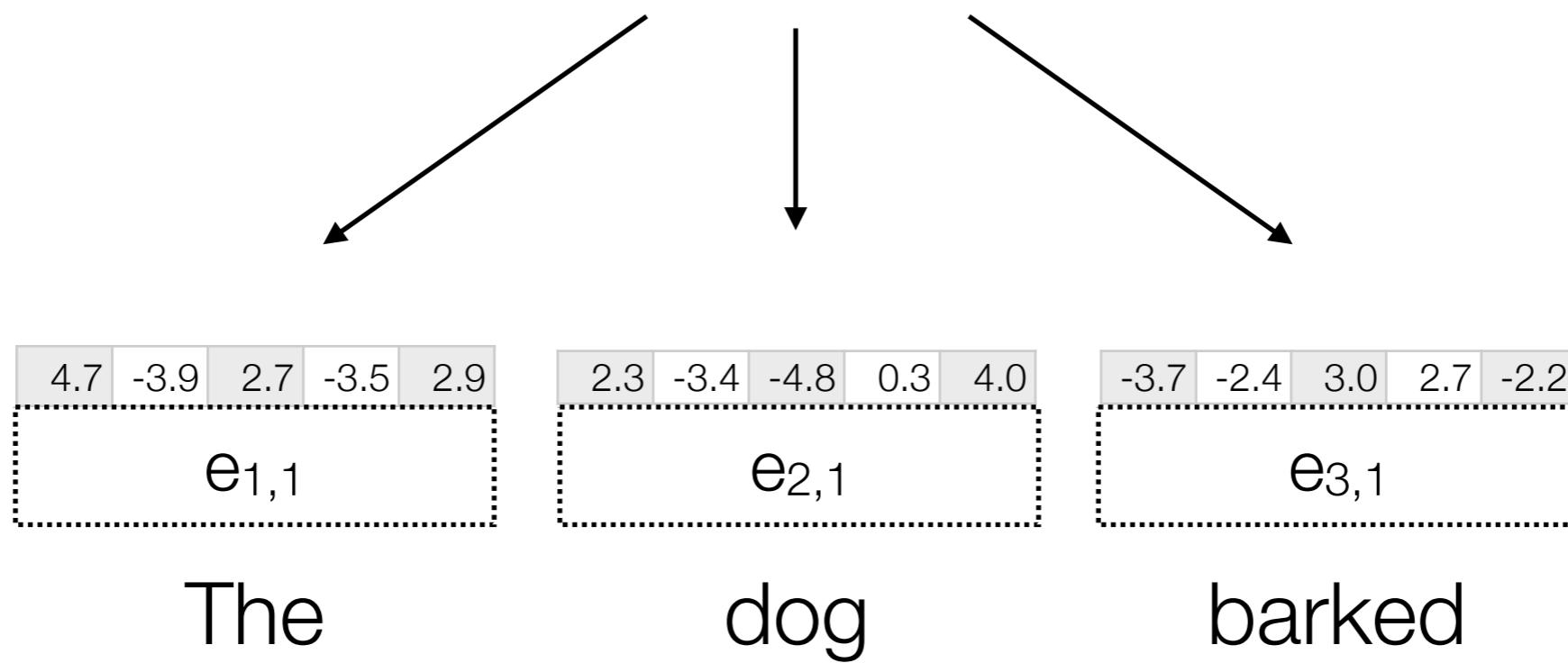
# The future of NLP: Pretrained Models

# BERT

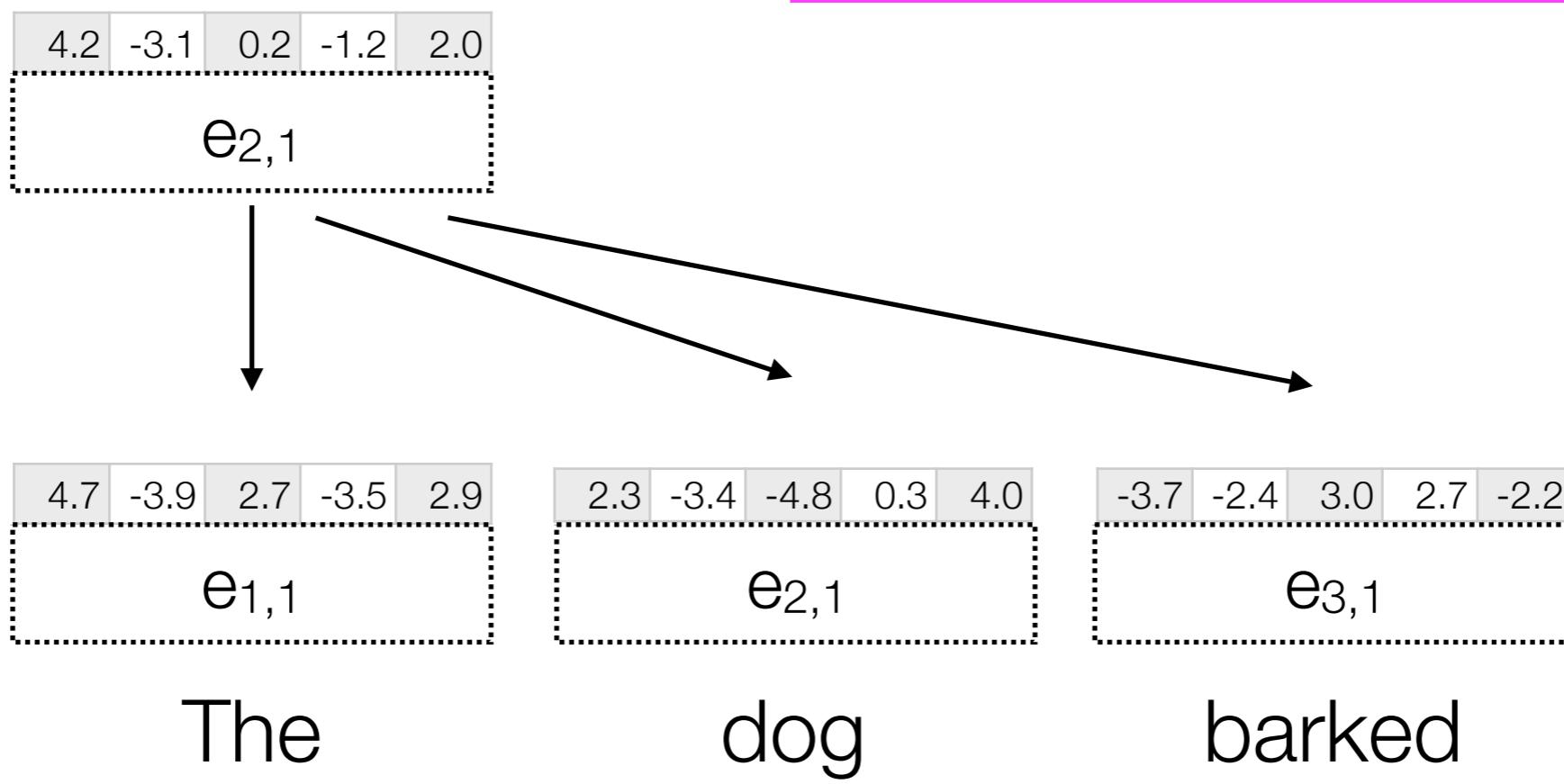
- Transformer-based model (Vaswani et al. 2017) to predict masked word using bidirectional context + next sentence prediction.
- Generates multiple layers of representations for each token sensitive to its context of use.

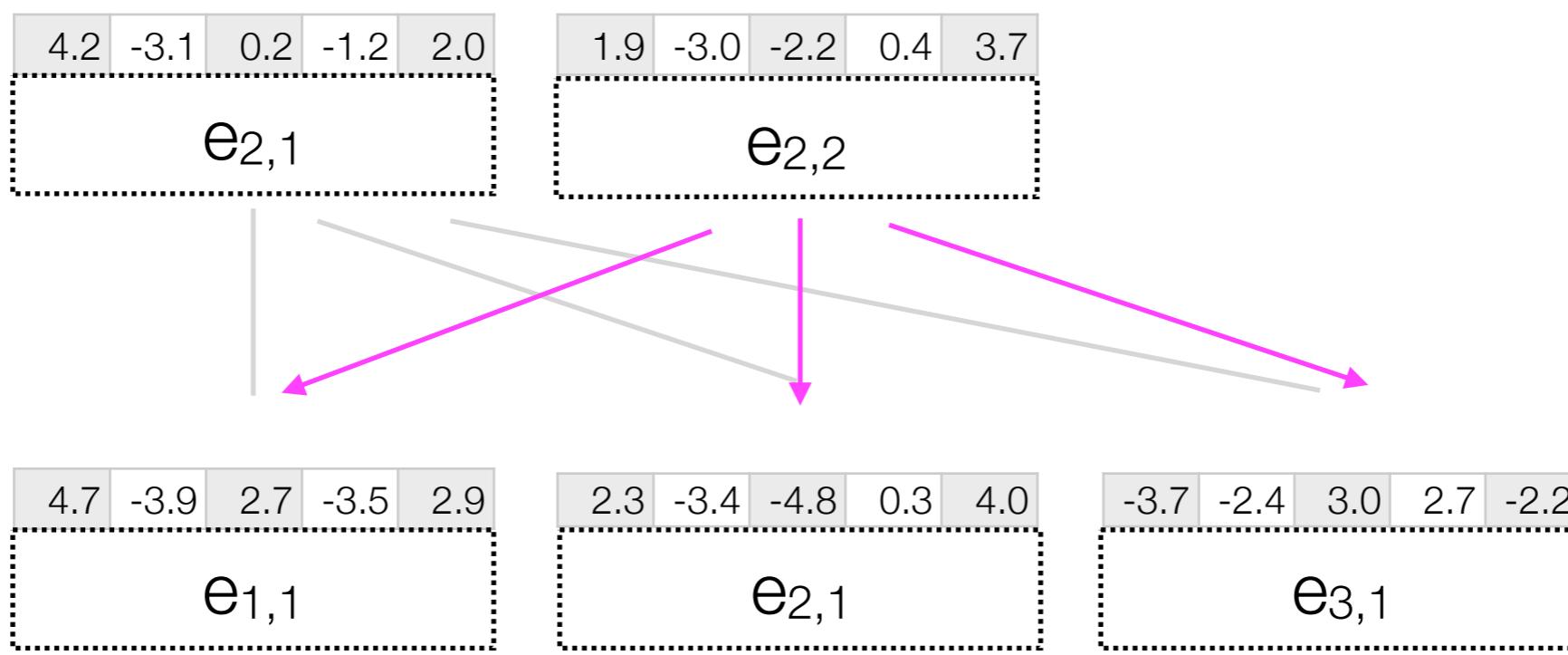


Each token in the input starts out represented by token and position embeddings



The value for time step  $j$  at layer  $i$  is  
the result of attention over all time  
steps in the previous layer  $i-1$

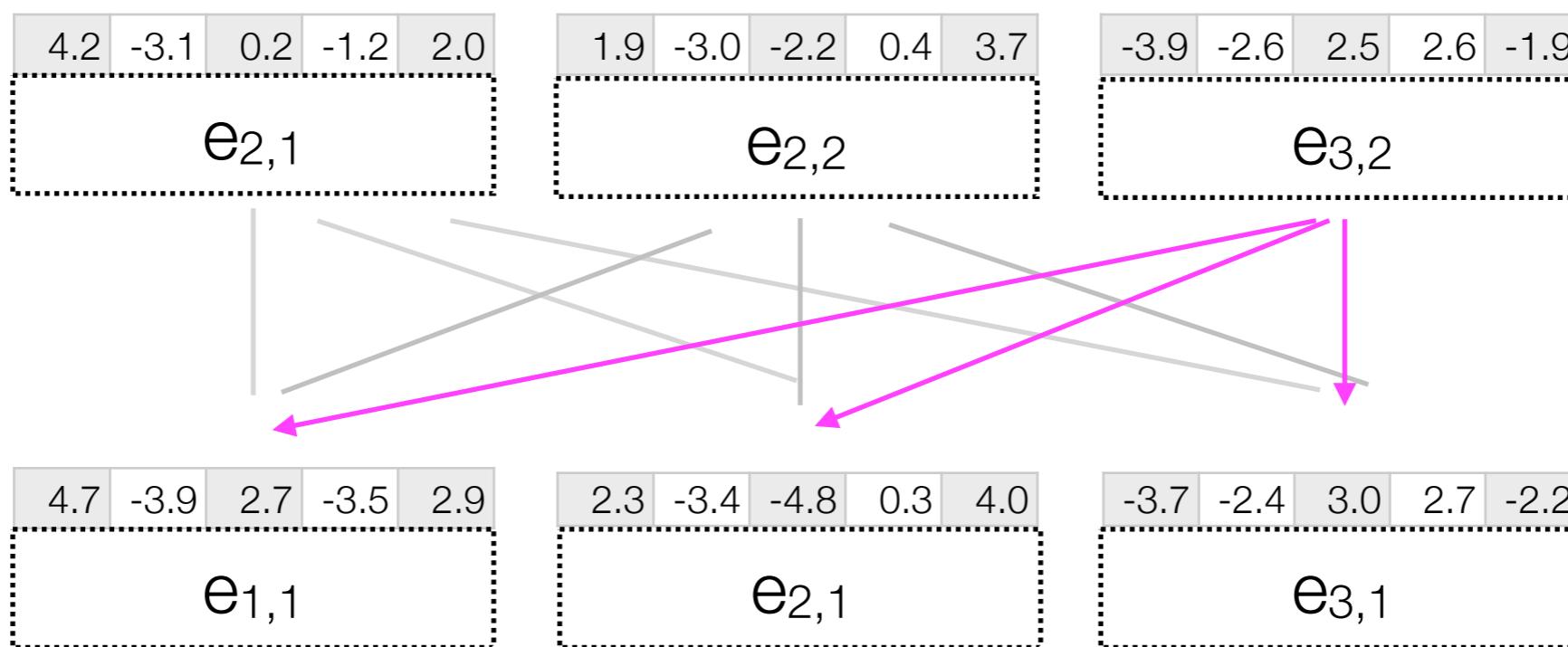




The

dog

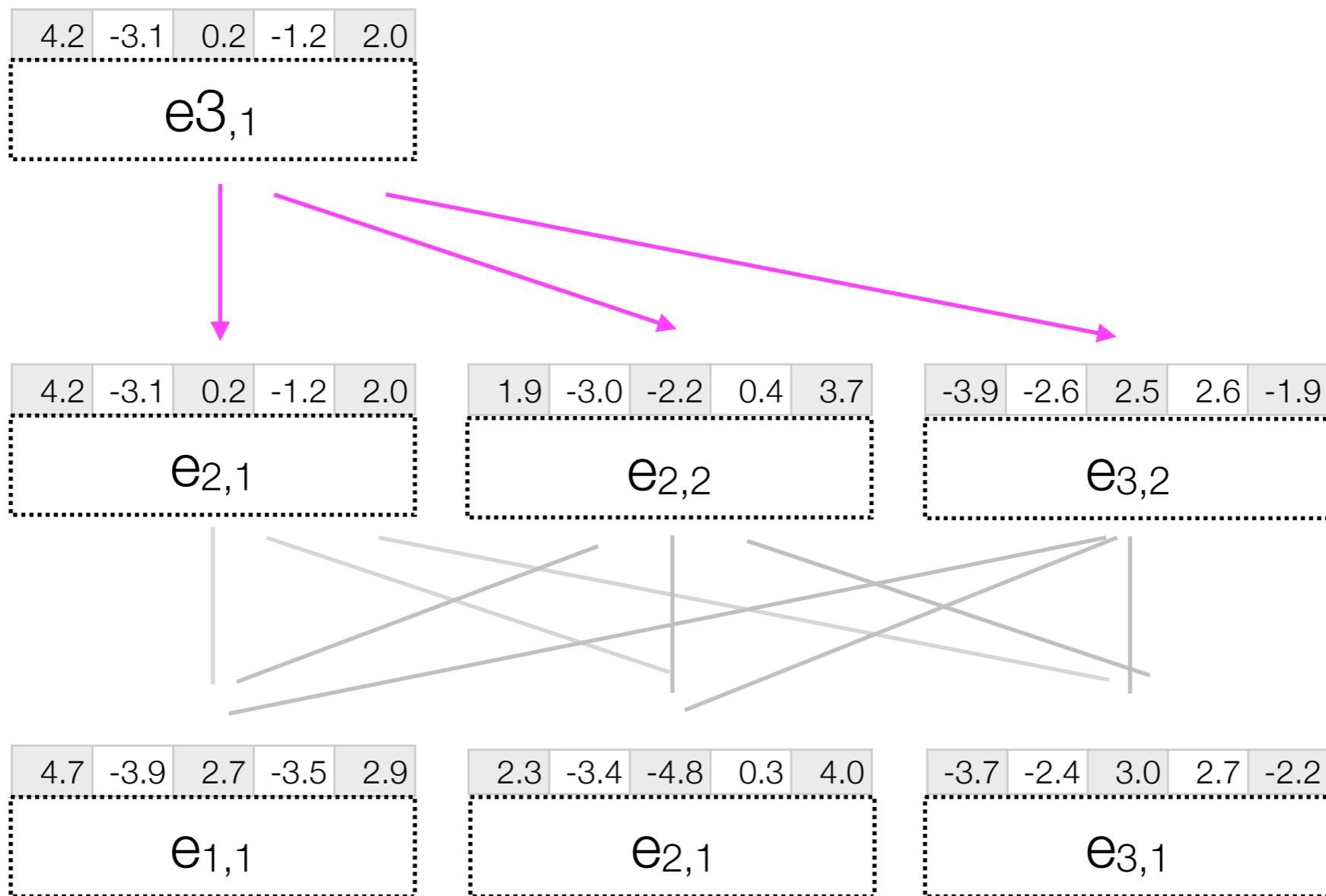
barked



The

dog

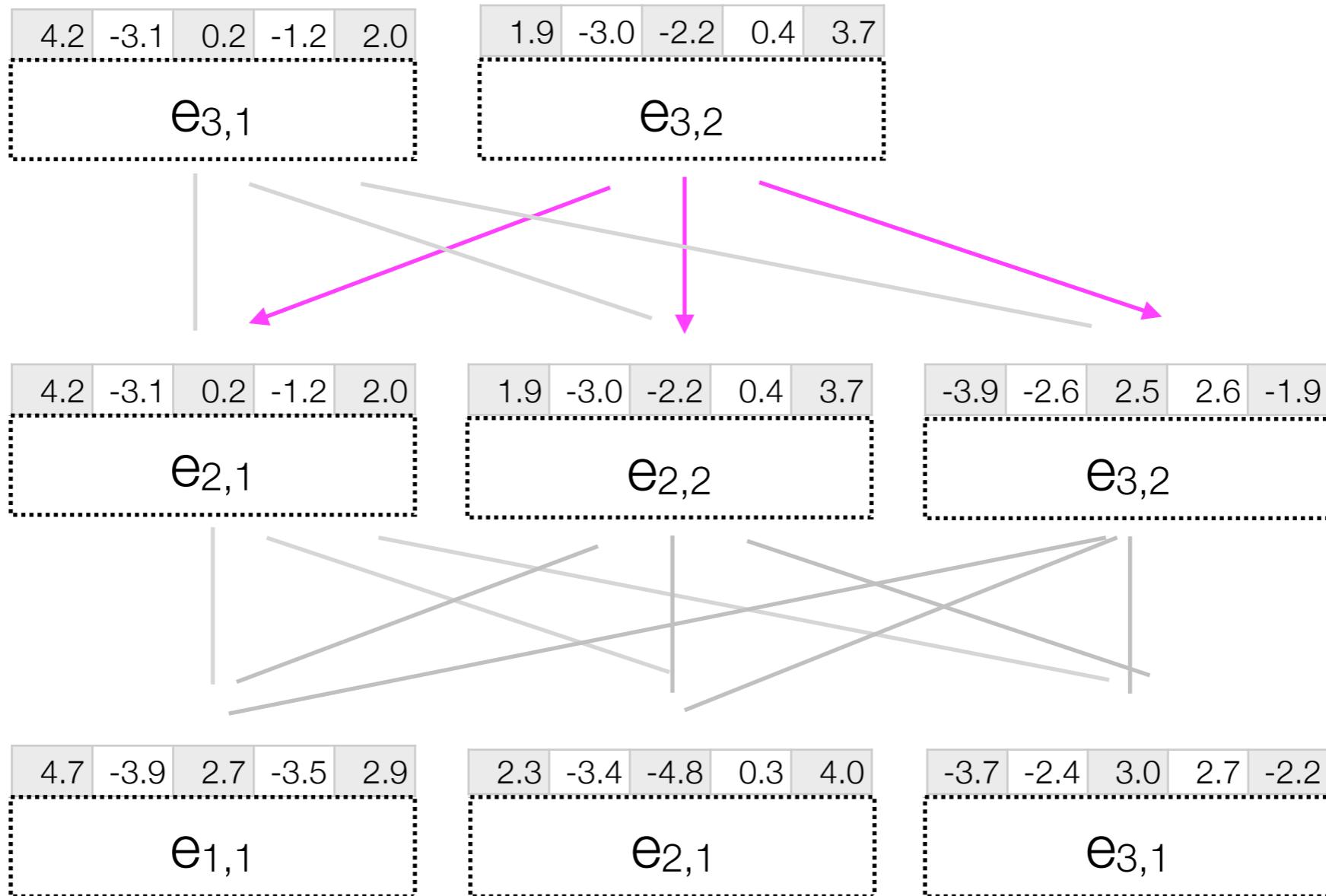
barked



The

dog

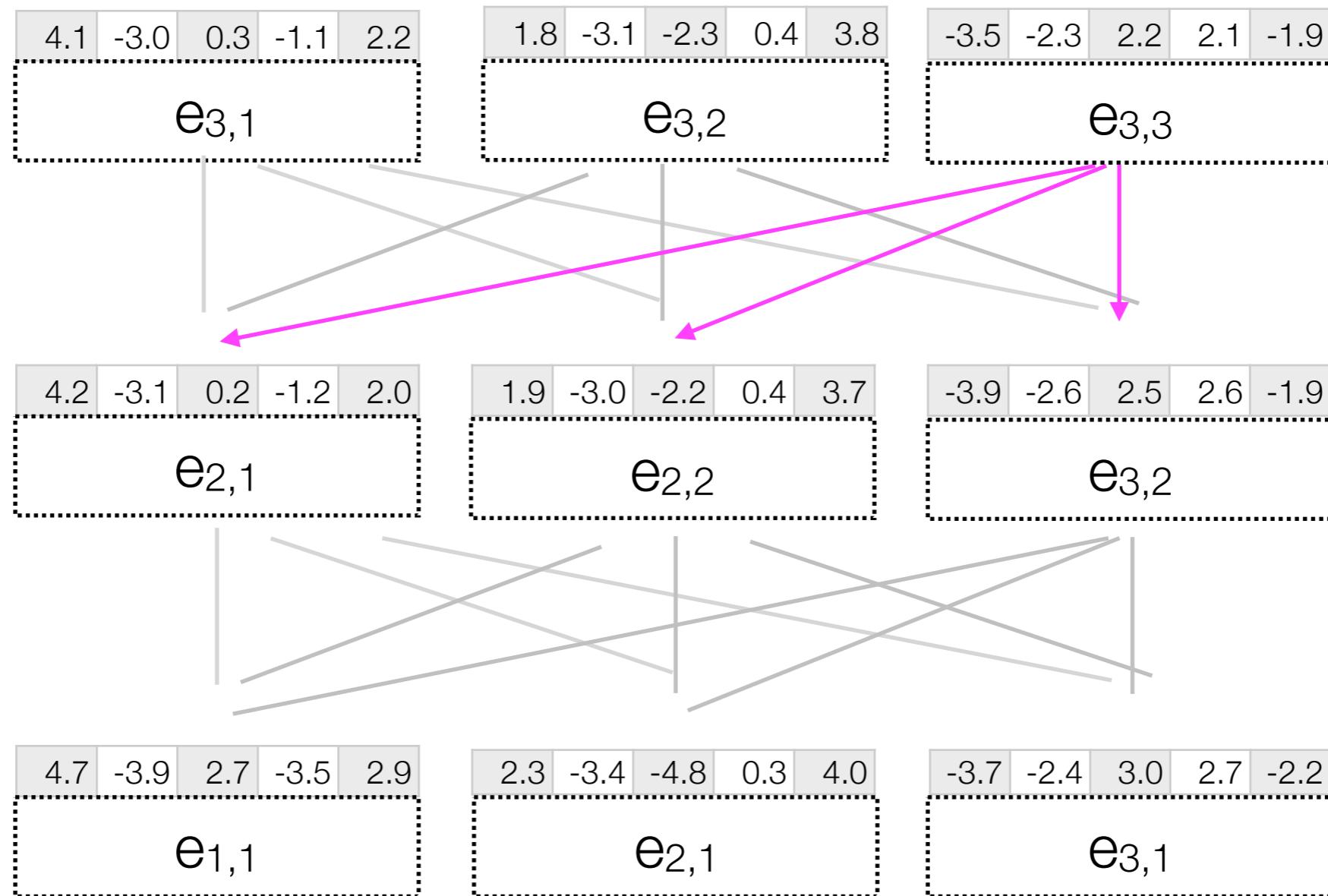
barked



The

dog

barked



The

dog

barked

At the end of this process, we have one representation **for each layer** for each token

4.1	-3.0	0.3	-1.1	2.2
$e_{3,1}$				

1.8	-3.1	-2.3	0.4	3.8
$e_{3,2}$				

-3.5	-2.3	2.2	2.1	-1.9
$e_{3,3}$				

4.2	-3.1	0.2	-1.2	2.0
$e_{2,1}$				

1.9	-3.0	-2.2	0.4	3.7
$e_{2,2}$				

-3.9	-2.6	2.5	2.6	-1.9
$e_{3,2}$				

4.7	-3.9	2.7	-3.5	2.9
$e_{1,1}$				

2.3	-3.4	-4.8	0.3	4.0
$e_{2,1}$				

-3.7	-2.4	3.0	2.7	-2.2
$e_{3,1}$				

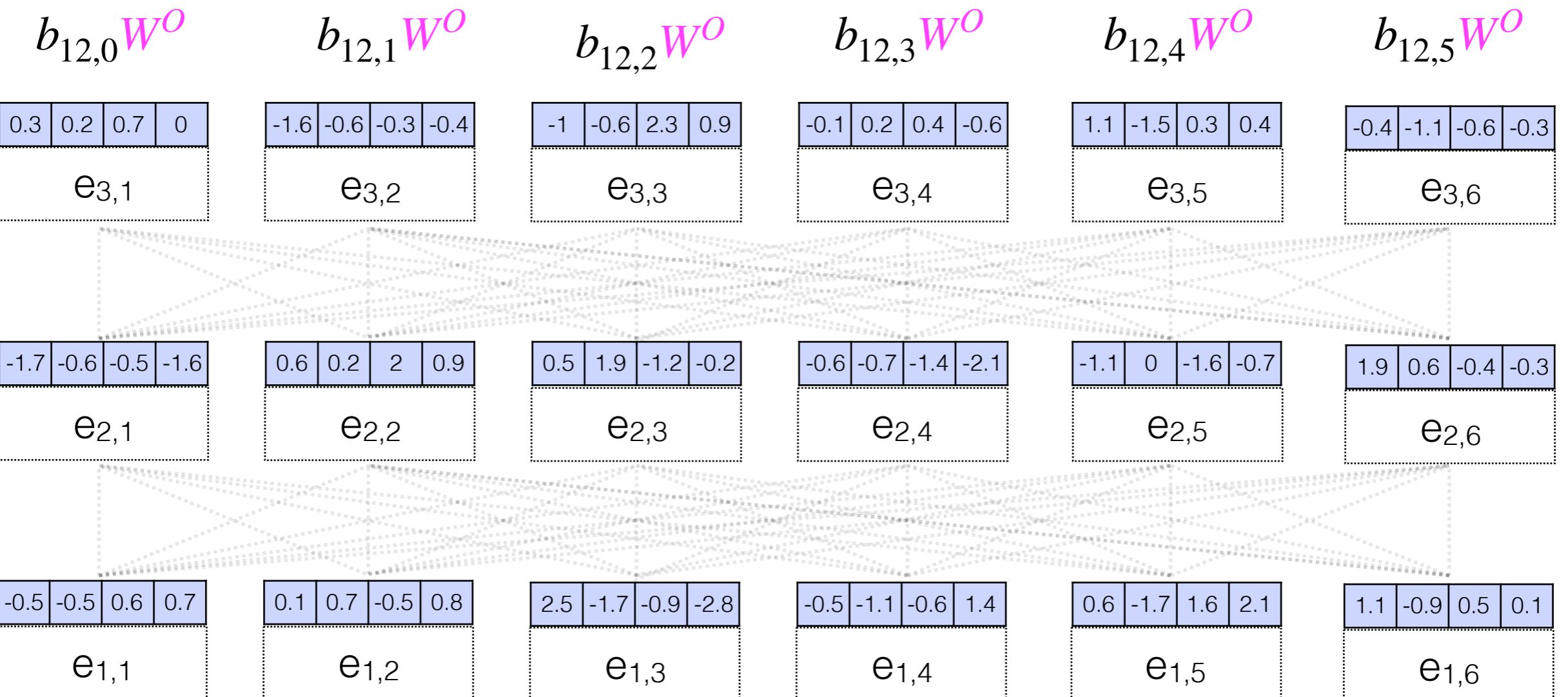
The

dog

barked

# BERT

- BERT can be used not only as a language model to generate contextualized word representations, but also as a predictive model whose parameters are fine-tuned to a task.



[CLS]

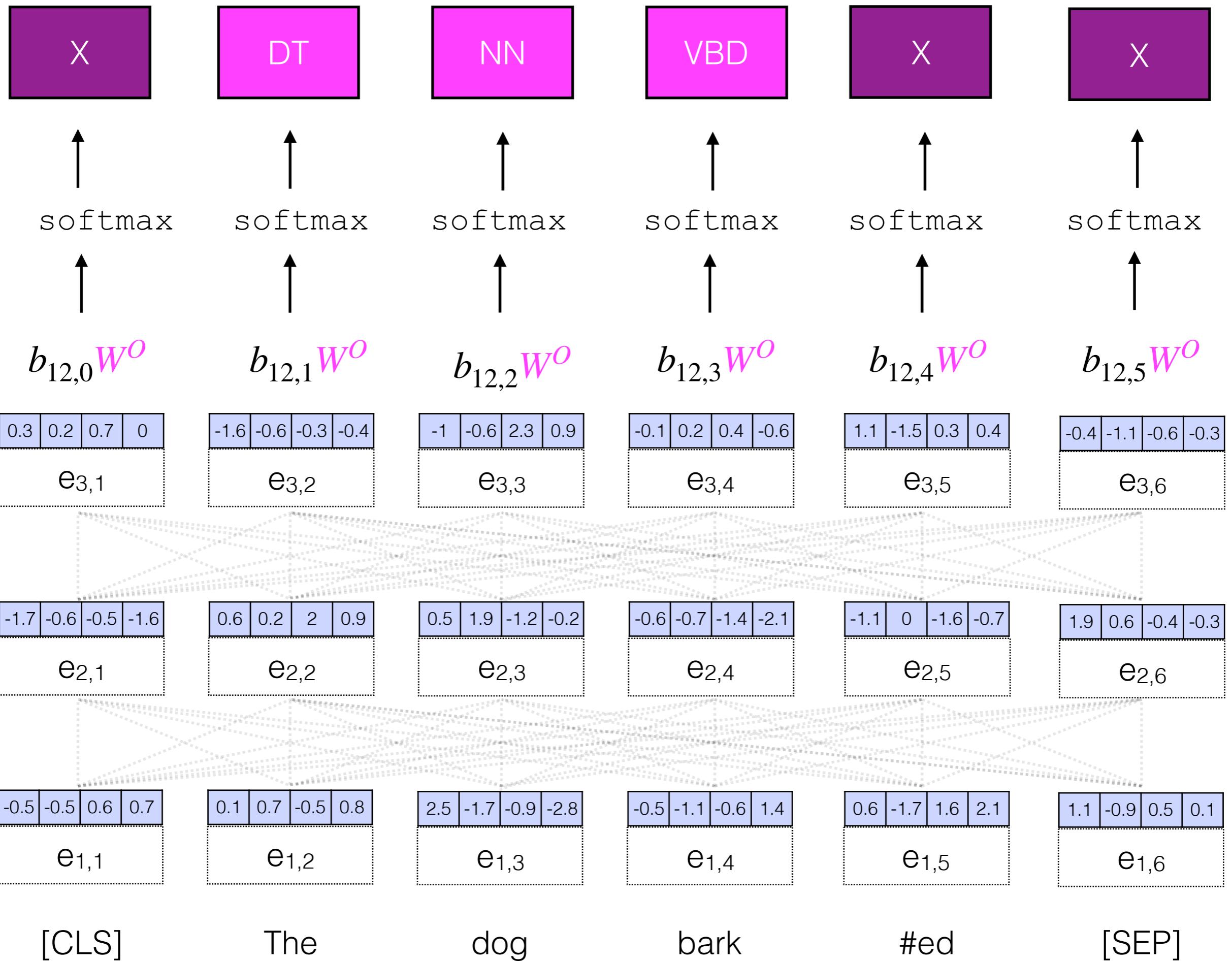
The

dog

bark

#ed

[SEP]



neutral



softmax



$b_{12,0} W^O$

0.3	0.2	0.7	0
e <sub>3,1</sub>			

-1.6	-0.6	-0.3	-0.4
e <sub>3,2</sub>			

-1	-0.6	2.3	0.9
e <sub>3,3</sub>			

-0.1	0.2	0.4	-0.6
e <sub>3,4</sub>			

1.1	-1.5	0.3	0.4
e <sub>3,5</sub>			

-0.4	-1.1	-0.6	-0.3
e <sub>3,6</sub>			

-1.7	-0.6	-0.5	-1.6
e <sub>2,1</sub>			

0.6	0.2	2	0.9
e <sub>2,2</sub>			

0.5	1.9	-1.2	-0.2
e <sub>2,3</sub>			

-0.6	-0.7	-1.4	-2.1
e <sub>2,4</sub>			

-1.1	0	-1.6	-0.7
e <sub>2,5</sub>			

1.9	0.6	-0.4	-0.3
e <sub>2,6</sub>			

-0.5	-0.5	0.6	0.7
e <sub>1,1</sub>			

0.1	0.7	-0.5	0.8
e <sub>1,2</sub>			

2.5	-1.7	-0.9	-2.8
e <sub>1,3</sub>			

-0.5	-1.1	-0.6	1.4
e <sub>1,4</sub>			

0.6	-1.7	1.6	2.1
e <sub>1,5</sub>			

1.1	-0.9	0.5	0.1
e <sub>1,6</sub>			

[CLS]

The

dog

bark

#ed

[SEP]

# BERT

- Pre-training: train BERT through masked language modeling and next-sentence prediction to learn the parameters of BERT layers. Trained on Wikipedia + BookCorpus.
- Task fine-tuning: add additional linear transformation + softmax to get distribution over output space. Trained on annotated data

# BERT

- Text “infilling” task: replace 15% of tokens with something else and try to predict the original
  - 80% of the time: MASK; 10%: random word; 10%: keep same

I went to the store and bought a gallon of milk . My favorite kind is 2% .

Transformer (12-24 Layer)

I went to the **MASK** and bought **MASK** gallon of dog . My **MASK** kind is 2% .

- Also generate “fake” sentence pairs and try to predict real from fake

I went to the **MASK** and bought **MASK** gallon of dog . I love karaoke!

# BERT Results

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>91.1</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>81.9</b>

- DramaRc gains on a range of sentence pair / single sentence tasks: paraphrase identification, entailment, sentiment, textual similarity, ...
- Not a generative model! But learns really effective representations...

# Probing BERT

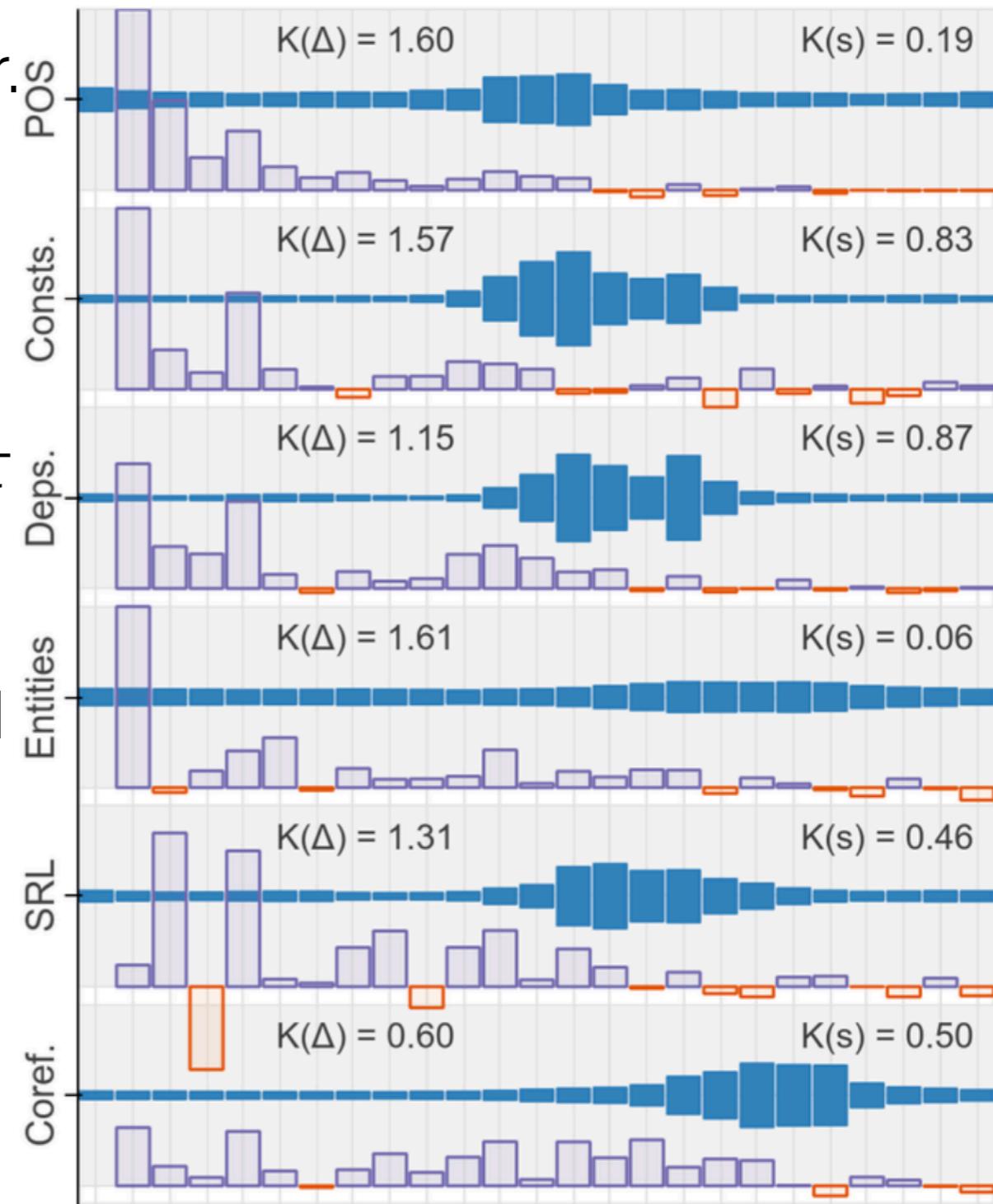
- Try to predict POS, etc. from each layer.  
Learn mixing weights

$$\mathbf{h}_{i,\tau} = \gamma_\tau \sum_{\ell=0}^L s_\tau^{(\ell)} \mathbf{h}_i^{(\ell)}$$



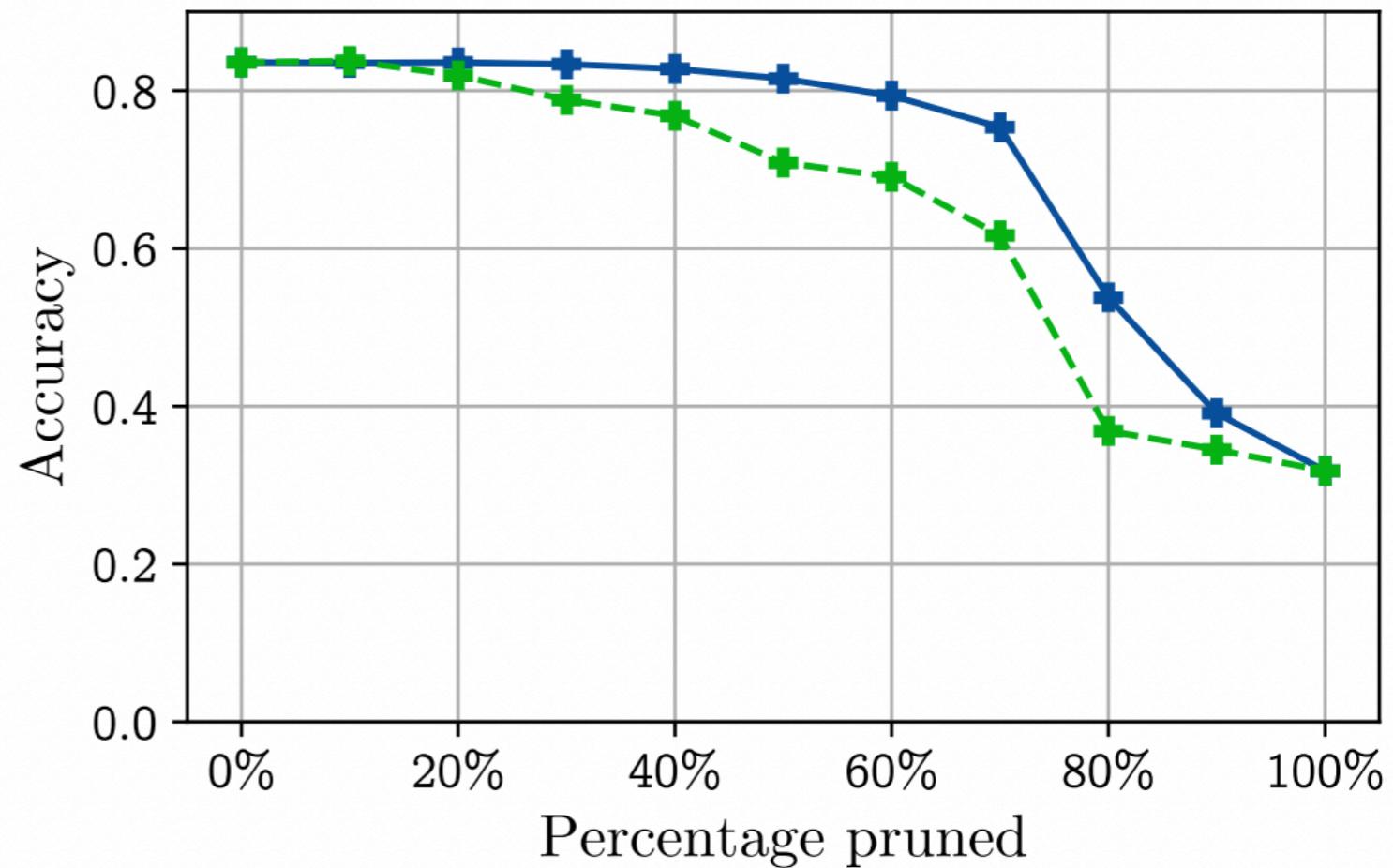
representation of word piece  $i$  for task  $\tau$

- Plot shows  $s$  weights (blue) and performance deltas when an additional layer is incorporated (purple)
- BERT “redisCOVERS the classical NLP pipeline”: first syntactic tasks then semantic ones



# Compressing BERT

- Remove 60+% of BERT's heads with minimal drop in performance
- DistilBERT (Sanh et al., 2019): nearly as good with half the parameters of BERT (via knowledge distillation)



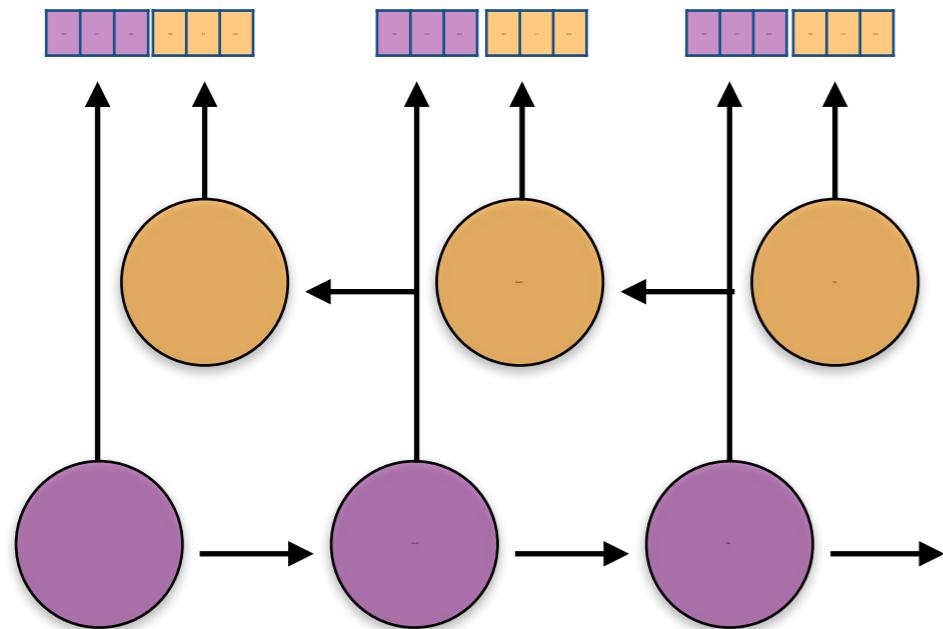
(b) Evolution of accuracy on the MultiNLI-matched validation set when heads are pruned from BERT according to  $I_h$  (solid blue) and accuracy difference (dashed green).

# BERT and EMLo are easier to use than you think!

- Tutorial and code from AllenNLP:  
<http://mlexplained.com/2019/01/30/an-in-depth-tutorial-to-allennlp-from-basics-to-elmo-and-bert/>
- Very (very) easy to use off-the-shelf example code for running various NLP tasks like classification using pertained BERT models:  
<https://github.com/huggingface/transformers>
  - This is going to be a part of HW5!

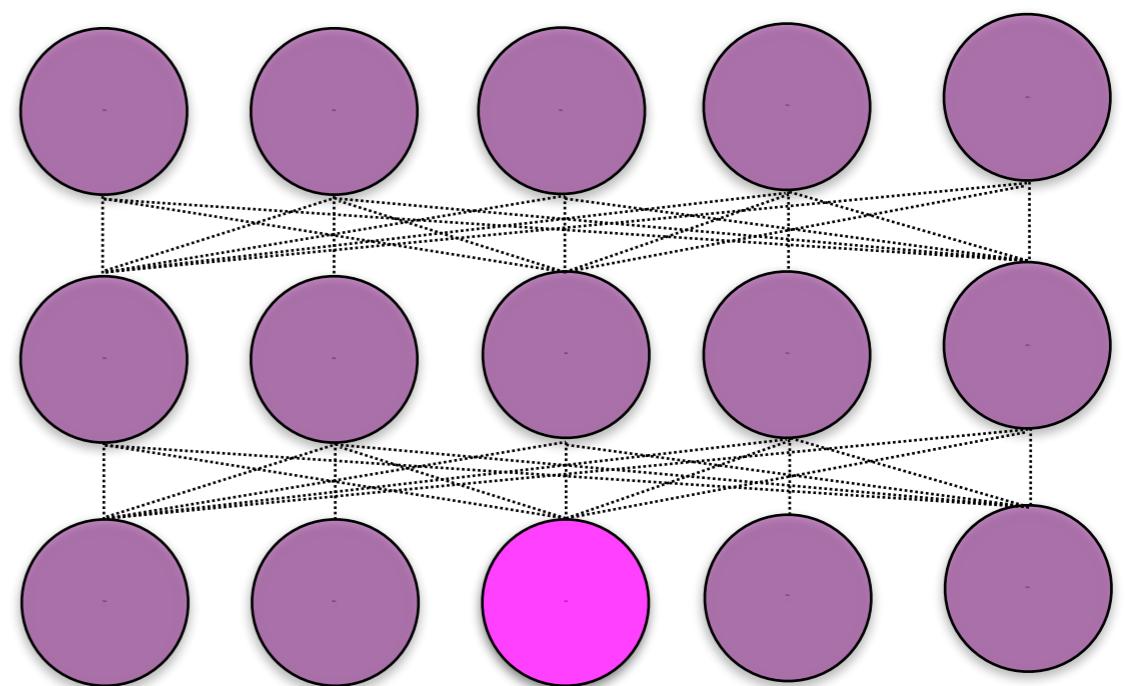
# ELMo and BERT

Stacked BiRNN trained to predict **next word** in language modeling task



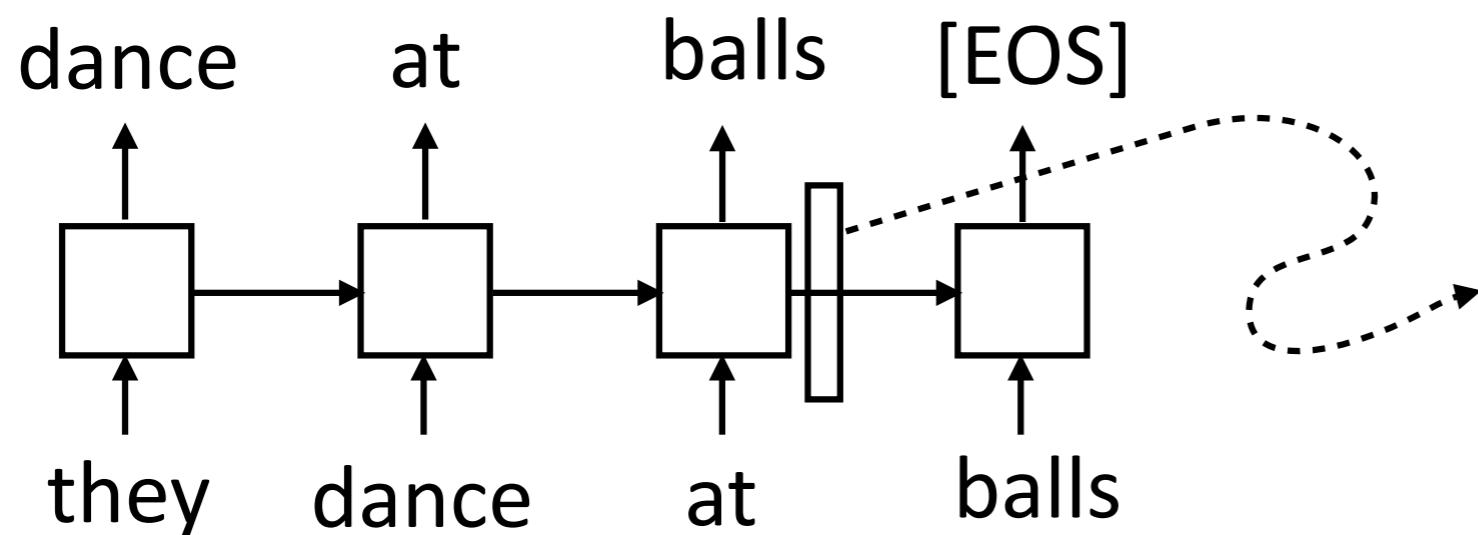
Peters et al. 2018

Transformer-based model to predict masked word using **bidirectional** context + next sentence prediction.



Devlin et al. 2019

# ELMo



learn a linear classifier on top of this vector to get a POS tagger with 97.3% accuracy (~SOTA)

$$P(x_i|x_1, \dots, x_{i-1}) = \text{LSTM}(x_1, \dots, x_{i-1})$$

- Generative model of the data!
- Train one model in each direction on 1B words, use the LSTM hidden states as context-aware token representations

# Pre-Training Cost (with Google/AWS)

- BERT: Base \$500, Large \$7000
- Grover-MEGA: \$25,000
- XLNet (BERT variant): \$30,000 — \$60,000 (unclear)

# Pre-Training Cost (with Google/AWS)

- BERT: Base \$500, Large \$7000
- Grover-MEGA: \$25,000
- XLNet (BERT variant): \$30,000 — \$60,000 (unclear)
- This is for a single pre-training run...developing new pre-training techniques may require many runs

# Pre-Training Cost (with Google/AWS)

- BERT: Base \$500, Large \$7000
- Grover-MEGA: \$25,000
- XLNet (BERT variant): \$30,000 — \$60,000 (unclear)
- This is for a single pre-training run...developing new pre-training techniques may require many runs
- Fine-tuning these models can typically be done with a single GPU (but may take 1-3 days for medium-sized datasets)

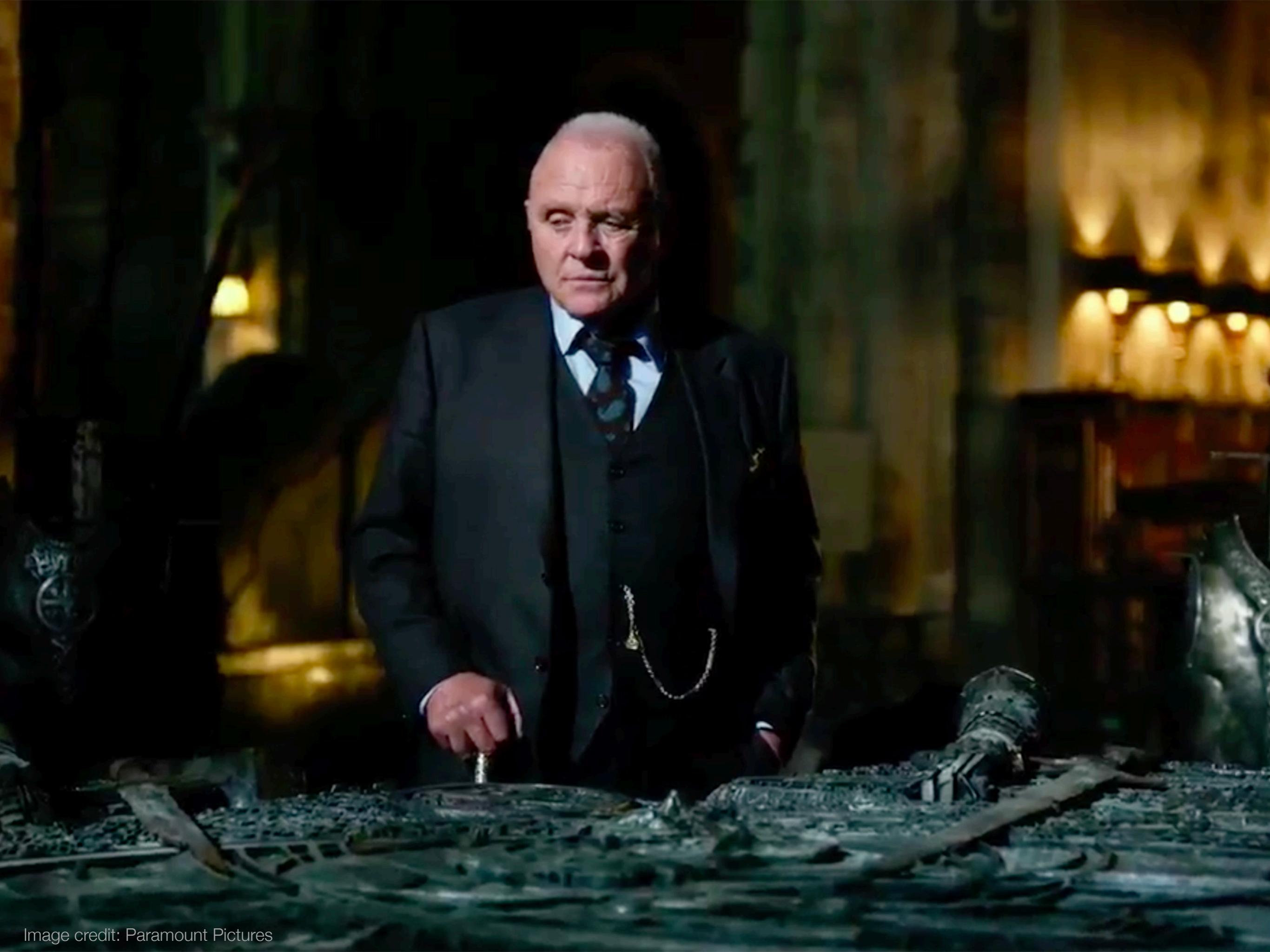
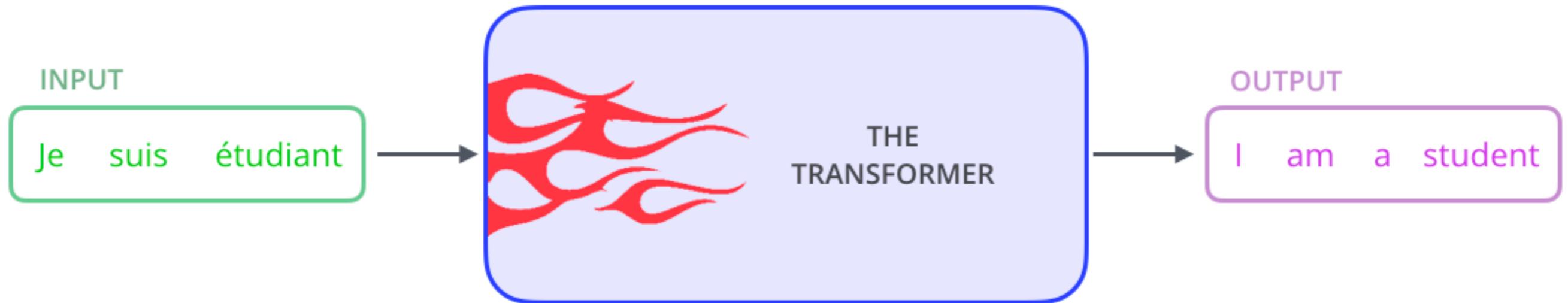


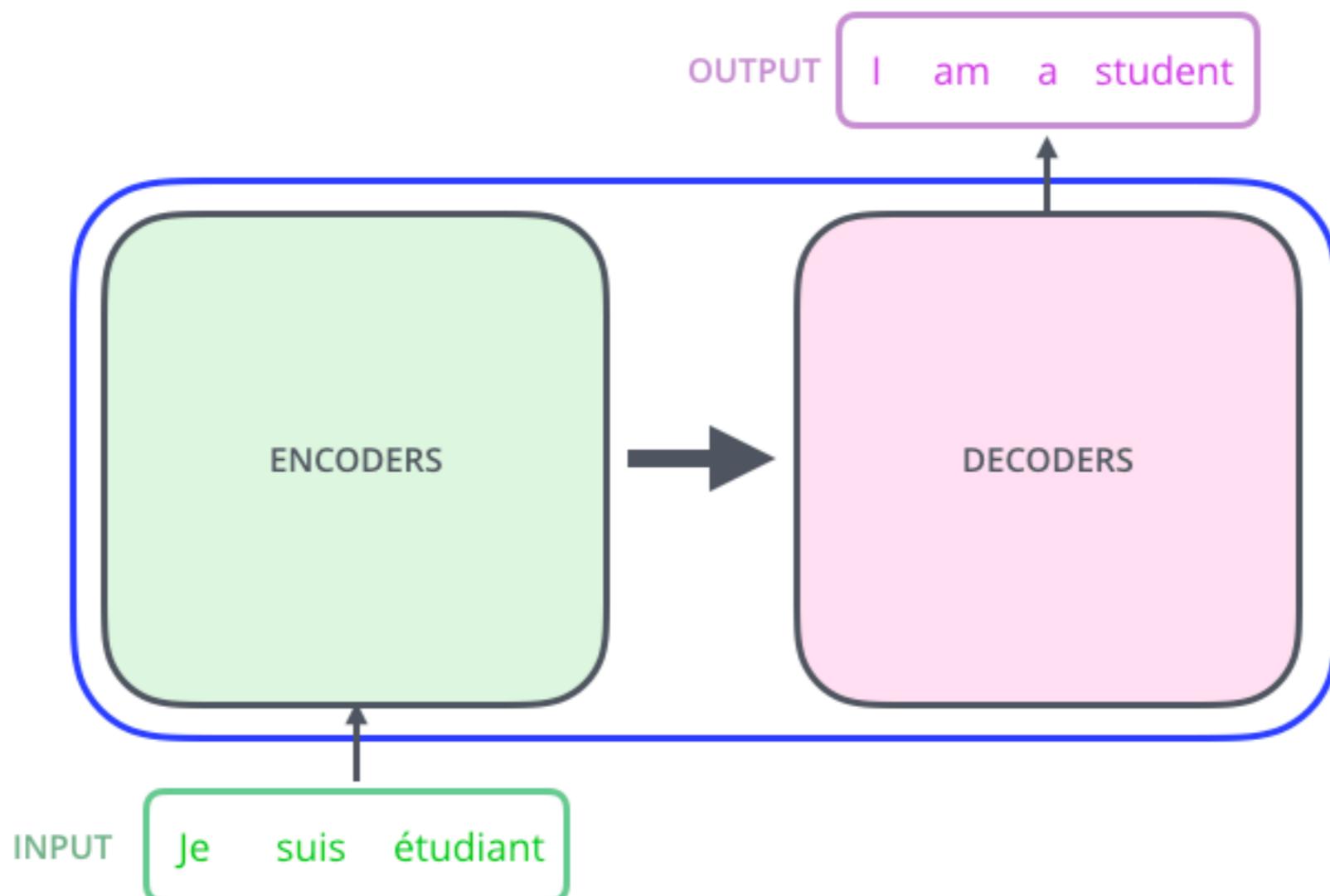
Image credit: Paramount Pictures

# Transformer Architecture

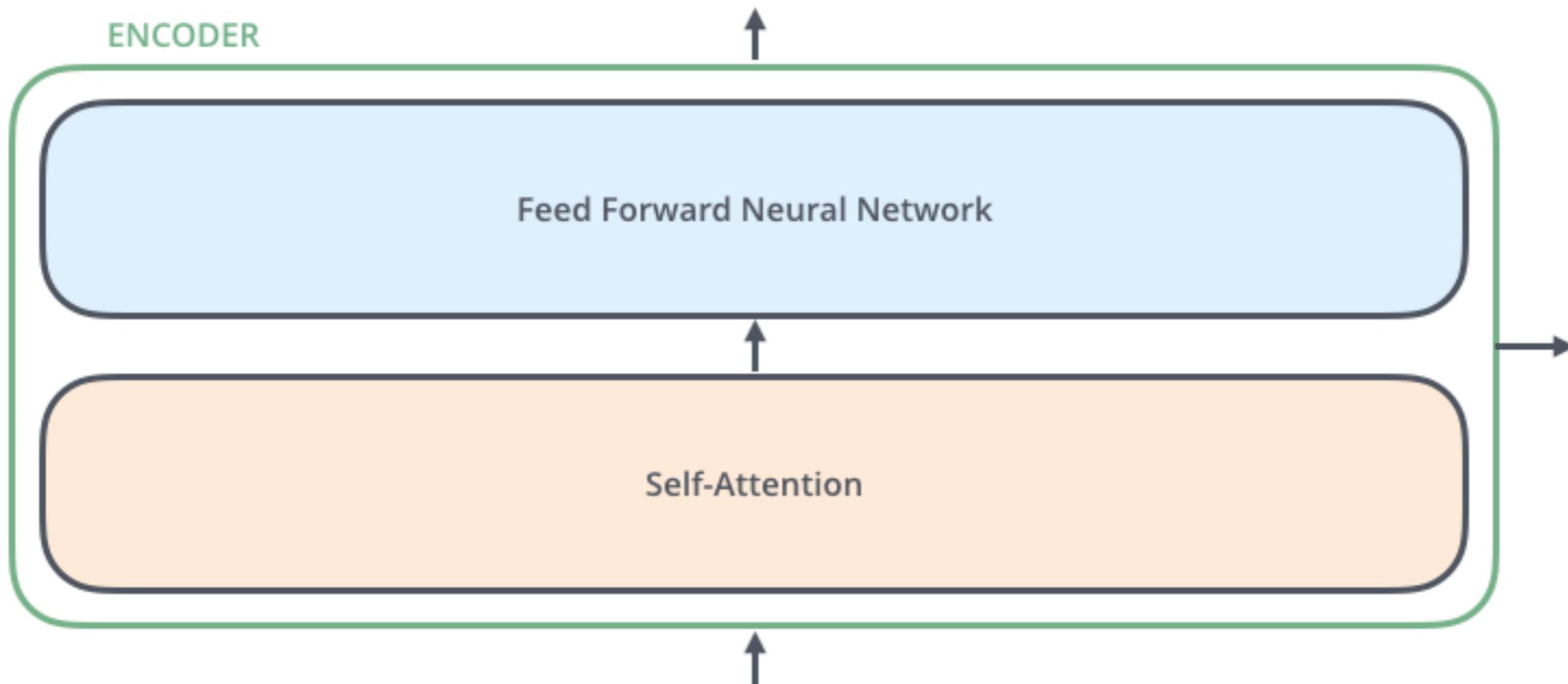
The transform follows the traditional  
encoder-decoder set up



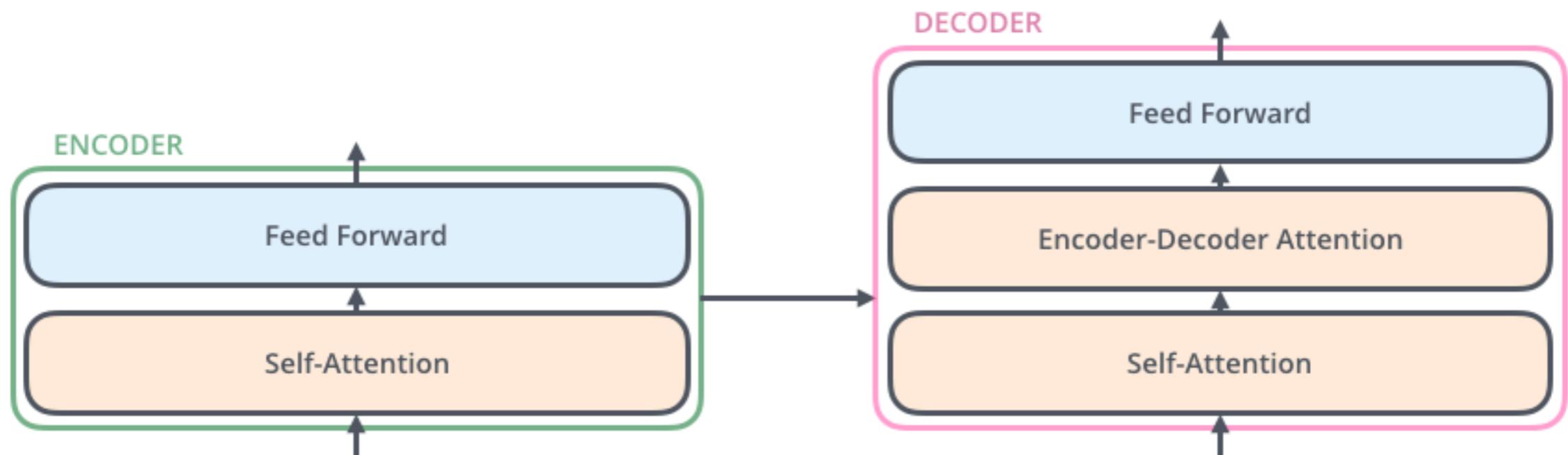
The transform follows the traditional  
encoder-decoder set up



Unlike the previous models, the encode is a feed forward network – no recurrence!



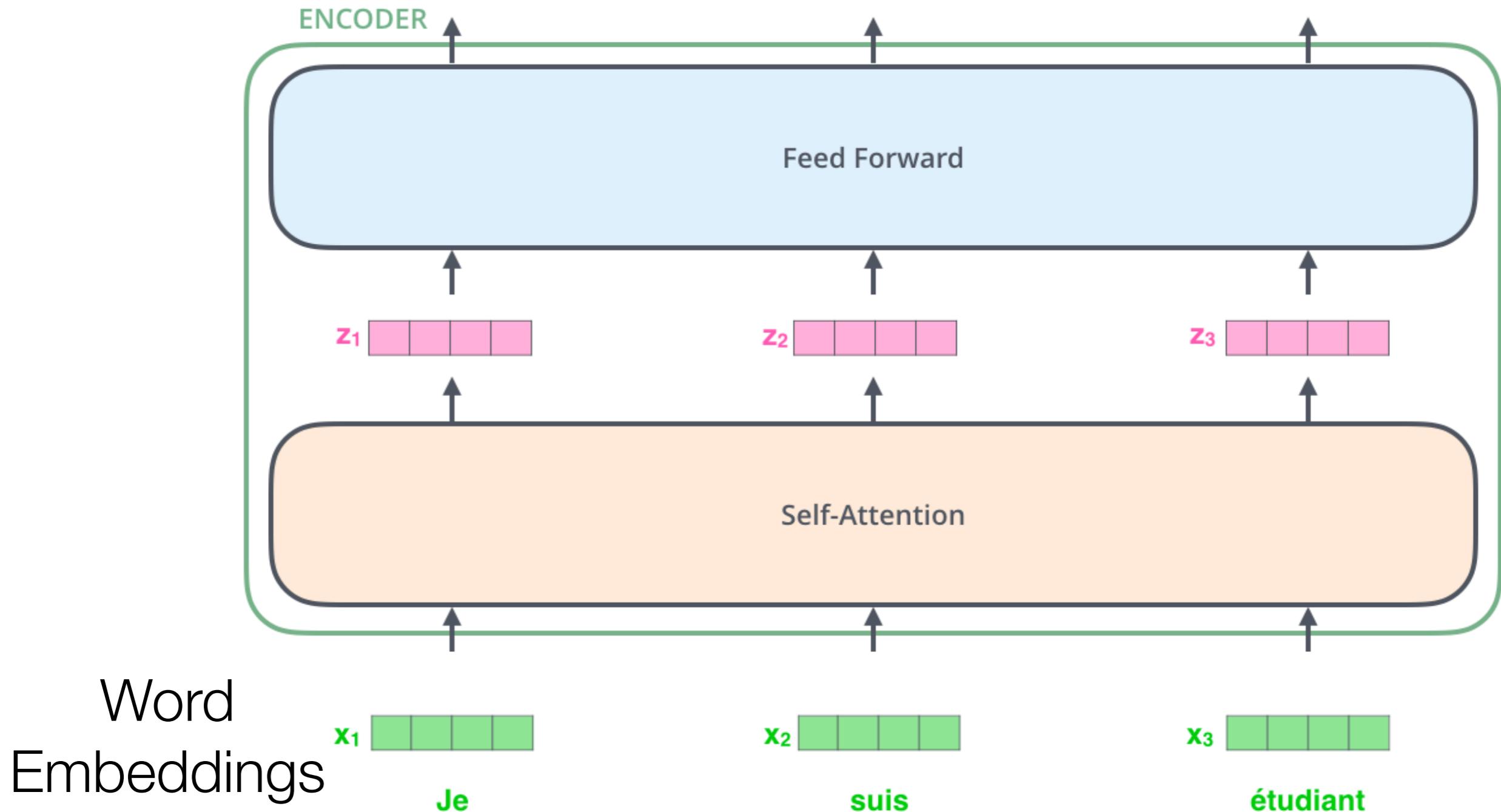
The decoder has access to the state of the encoder (not just its own state)



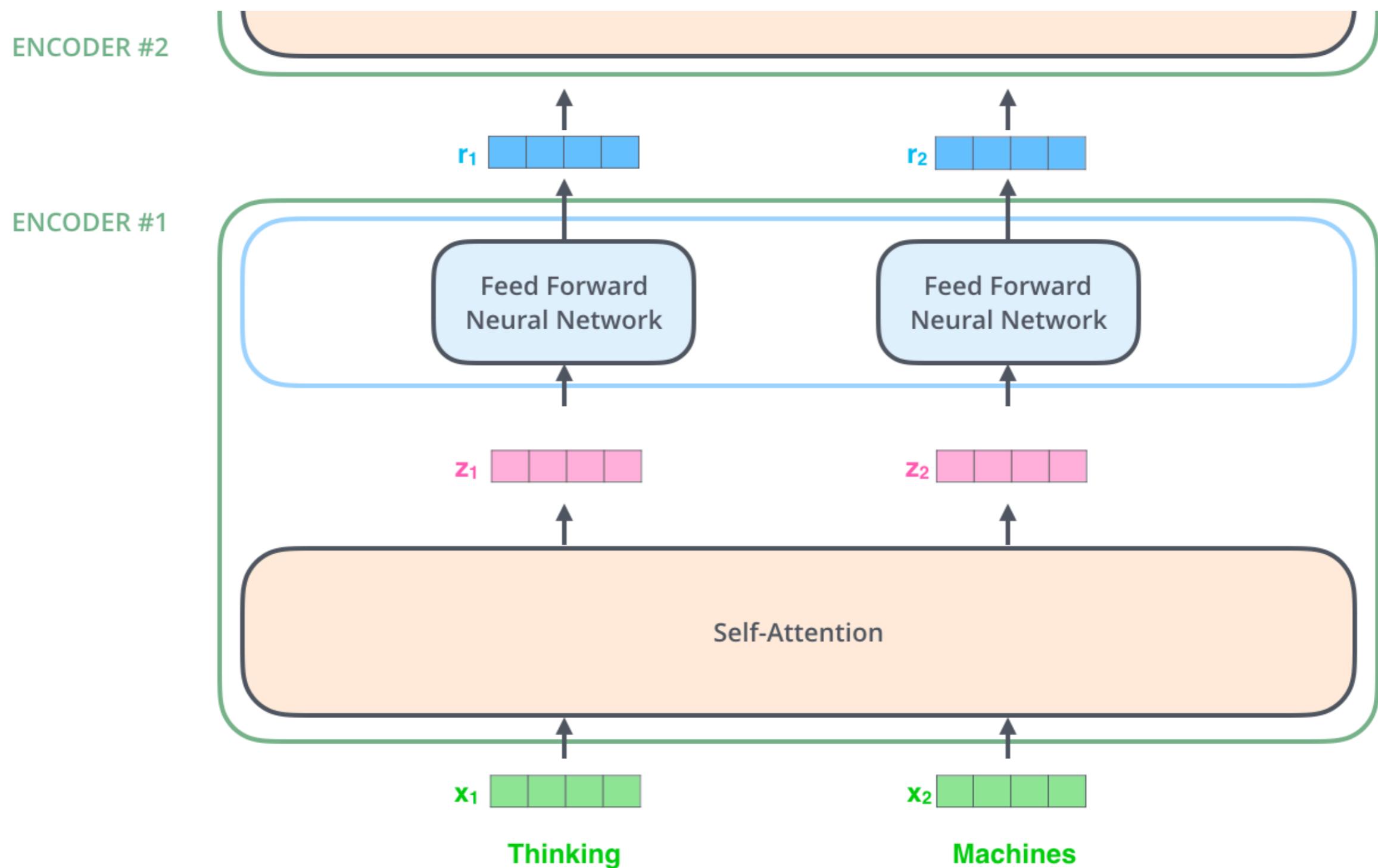
# A lot of the magic comes from self-attention



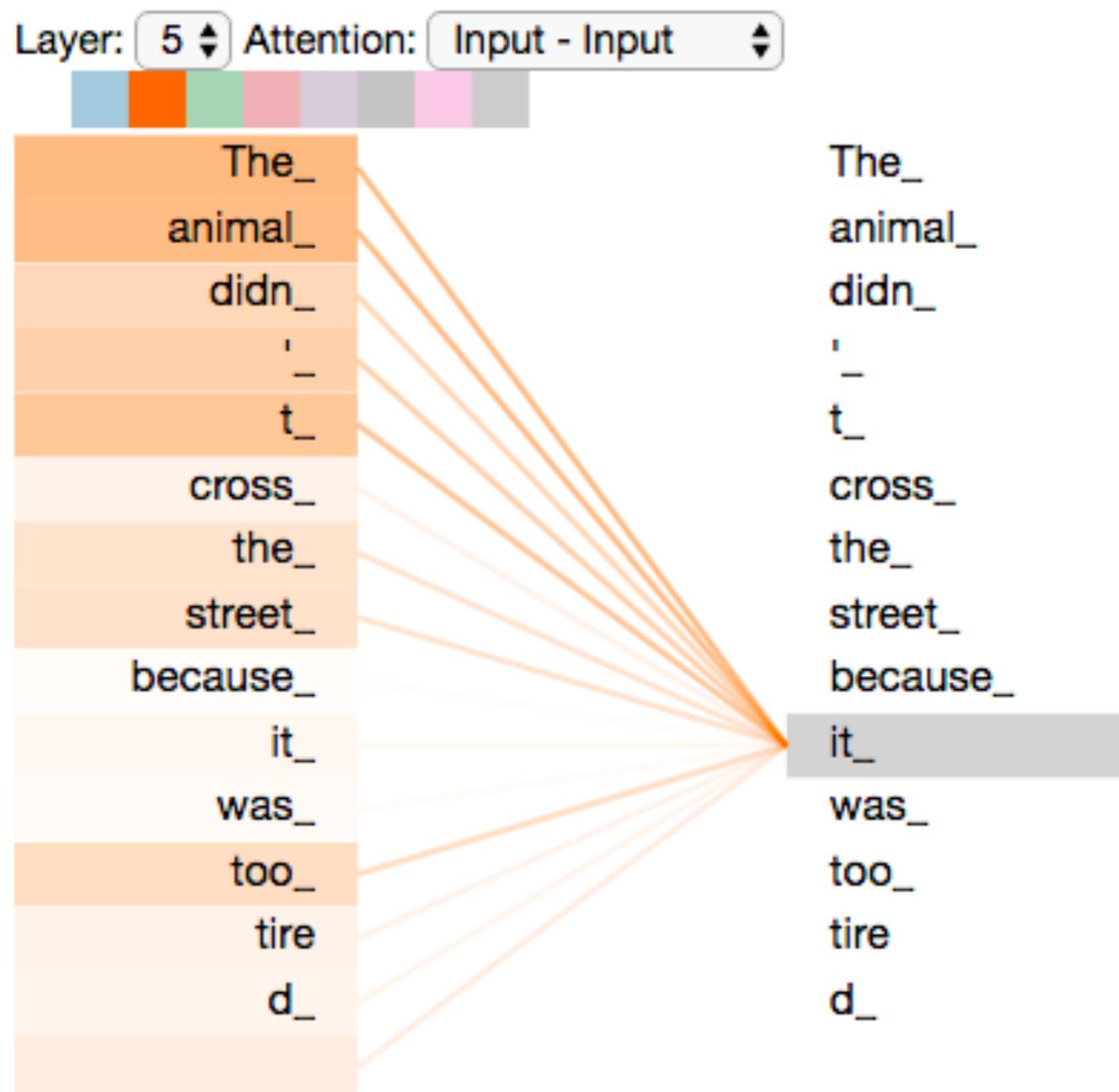
# A lot of the magic comes from self-attention



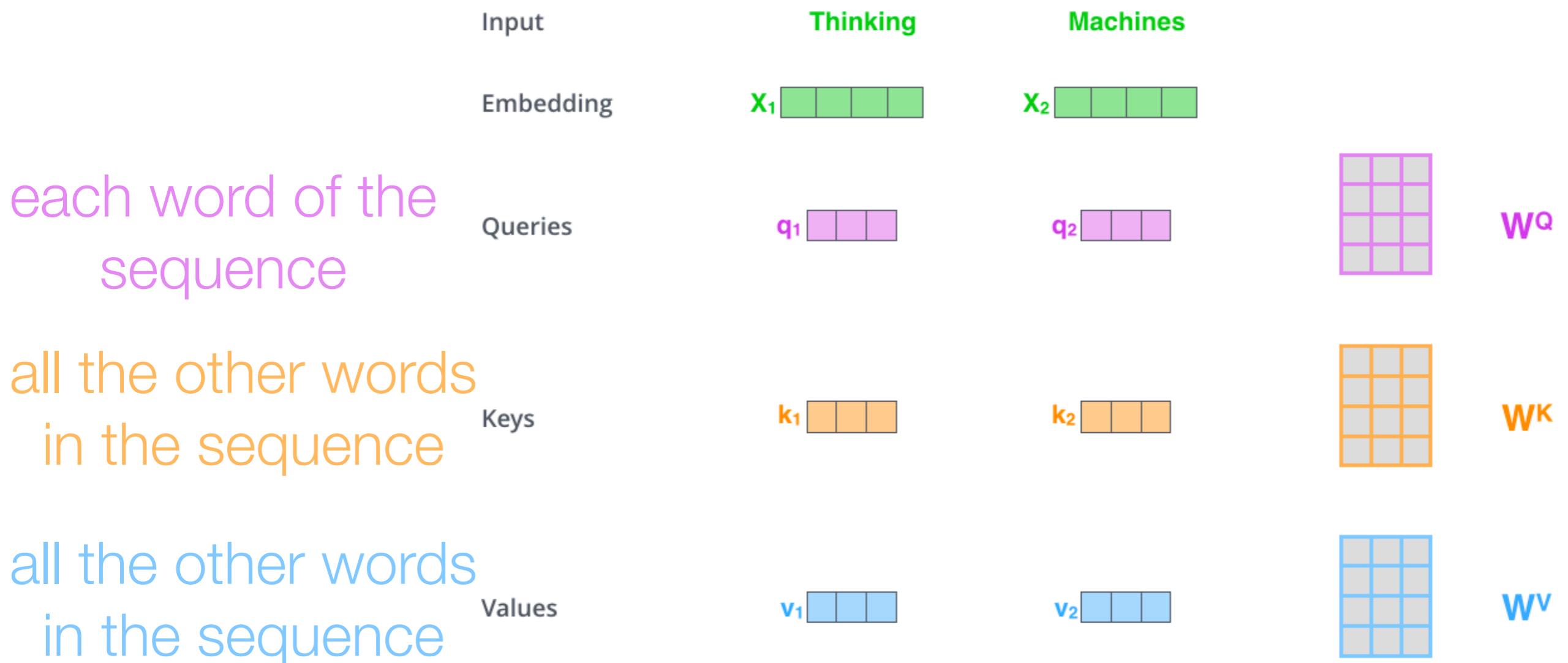
Input encoders can be stacked on top of each other. Google's architecture uses six!



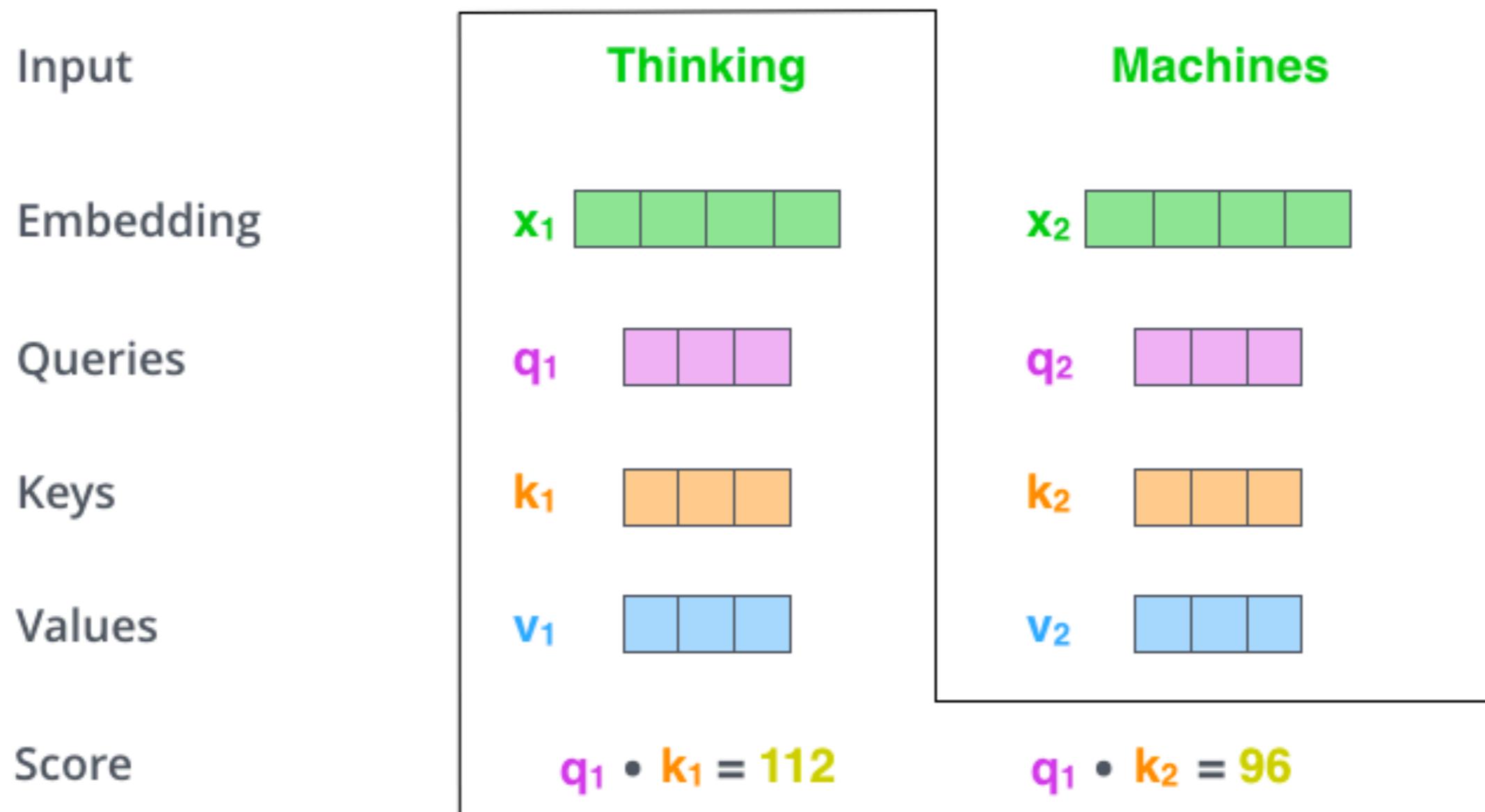
# Self-attention determines how relevant is every other input word to the current word



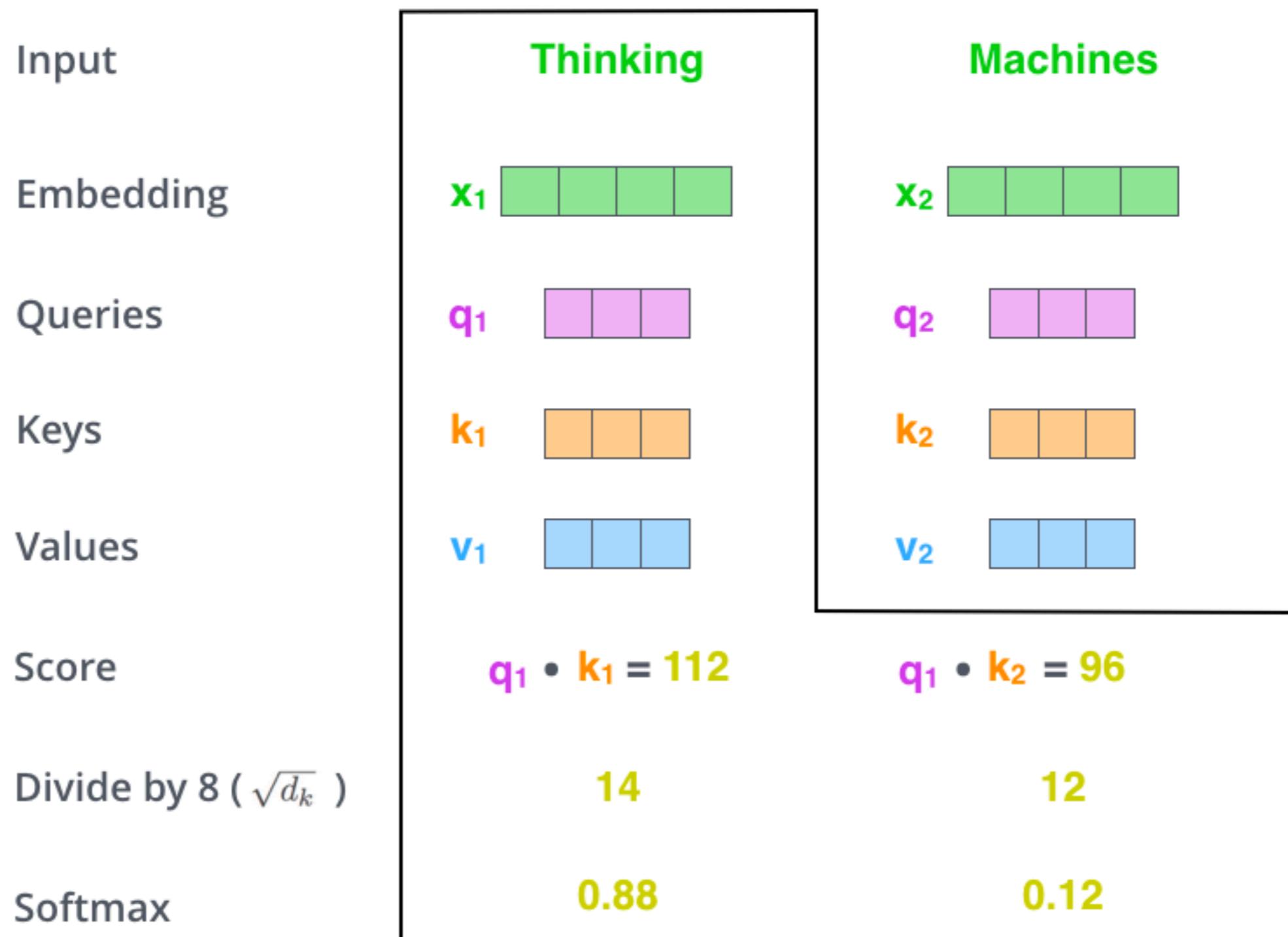
# Breaking down the steps of self attention



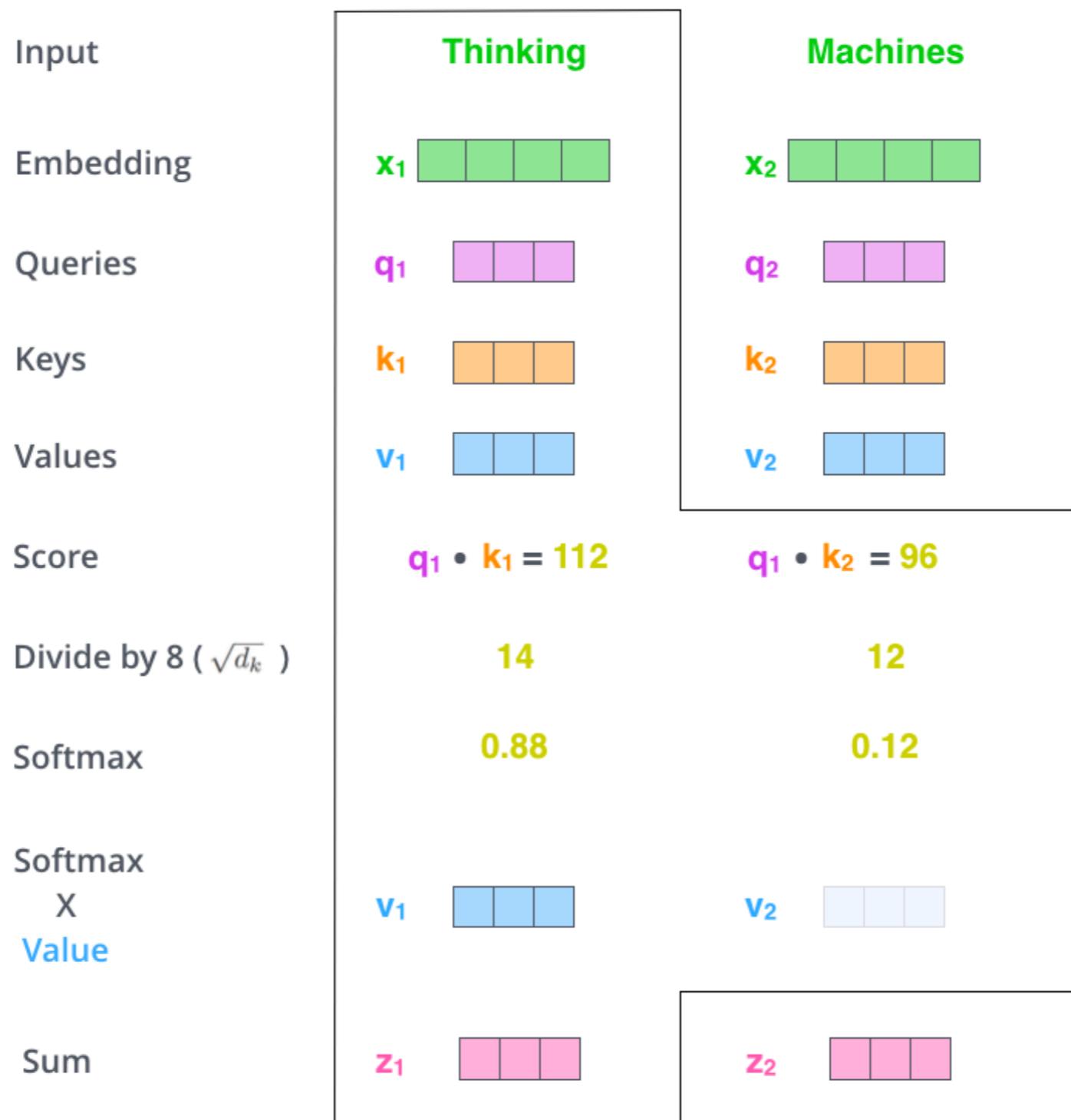
# $Q, K, V$ are embeddings of the input



$q^*k$  represents how related the center word is to the context word

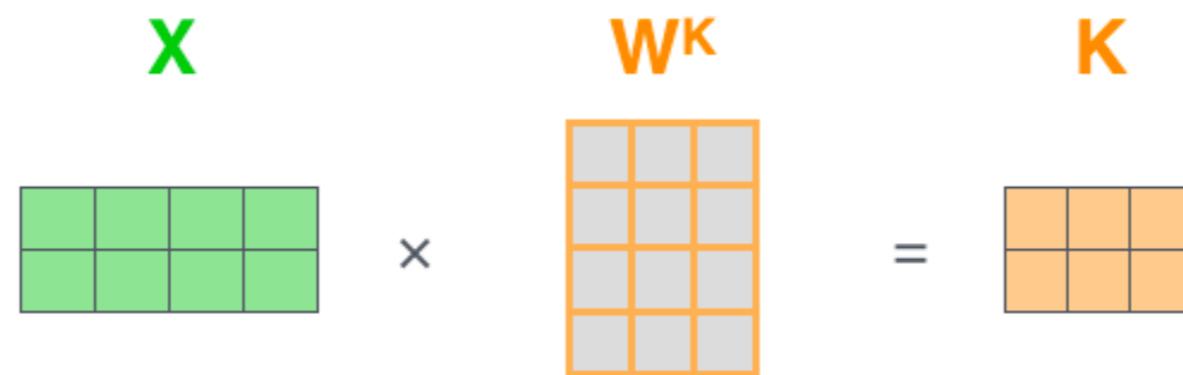


We scale the **vector** for the other words by related that word is ( $q^*k$ )



The **query**, **key**, and **value** vectors are just transforms of the existing word vectors

$$\mathbf{X} \times \mathbf{W}^Q = \mathbf{Q}$$


$$\mathbf{X} \times \mathbf{W}^K = \mathbf{K}$$


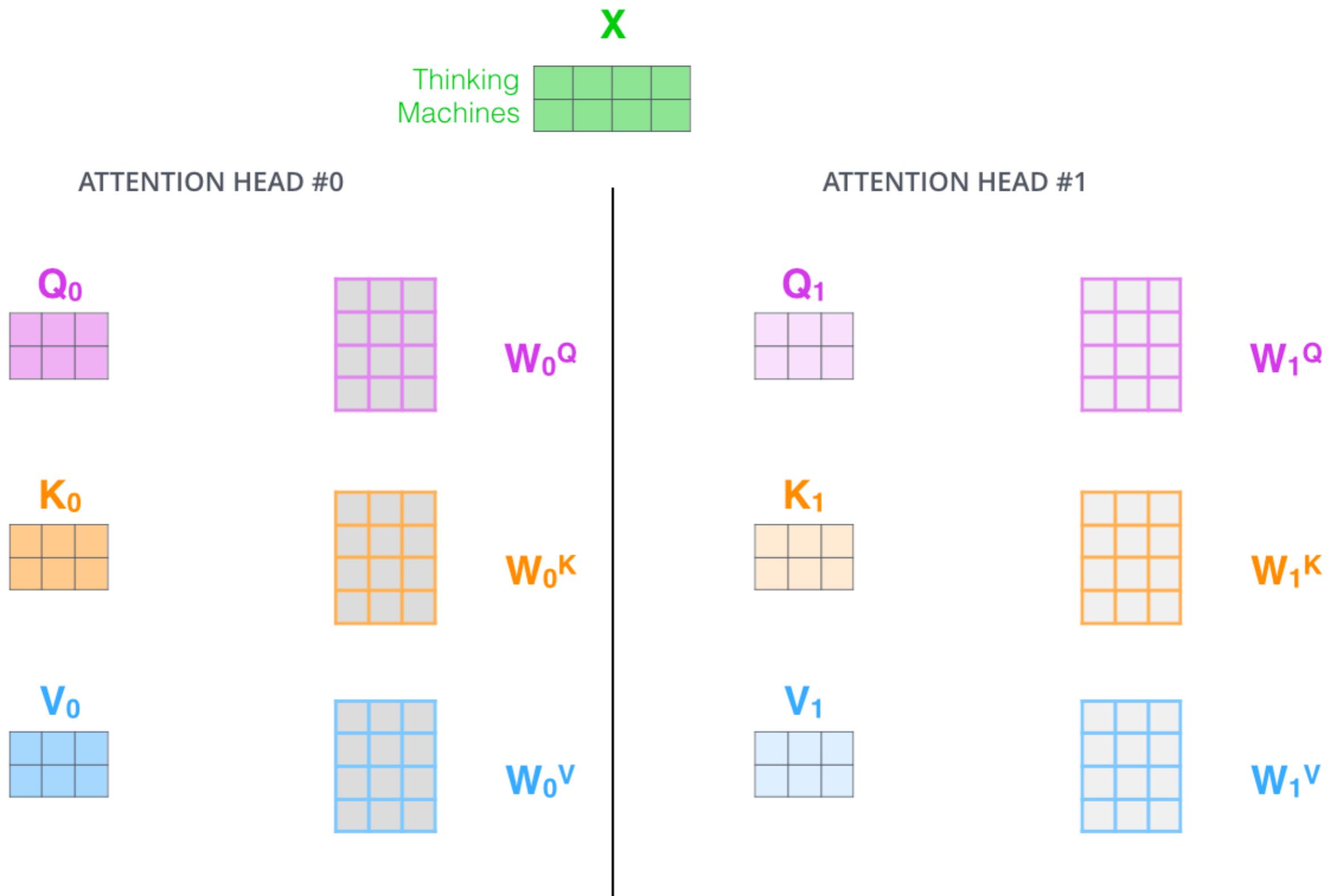
$$\mathbf{X} \times \mathbf{W}^V = \mathbf{V}$$


The whole attention gets computed at once using matrix multiplication!

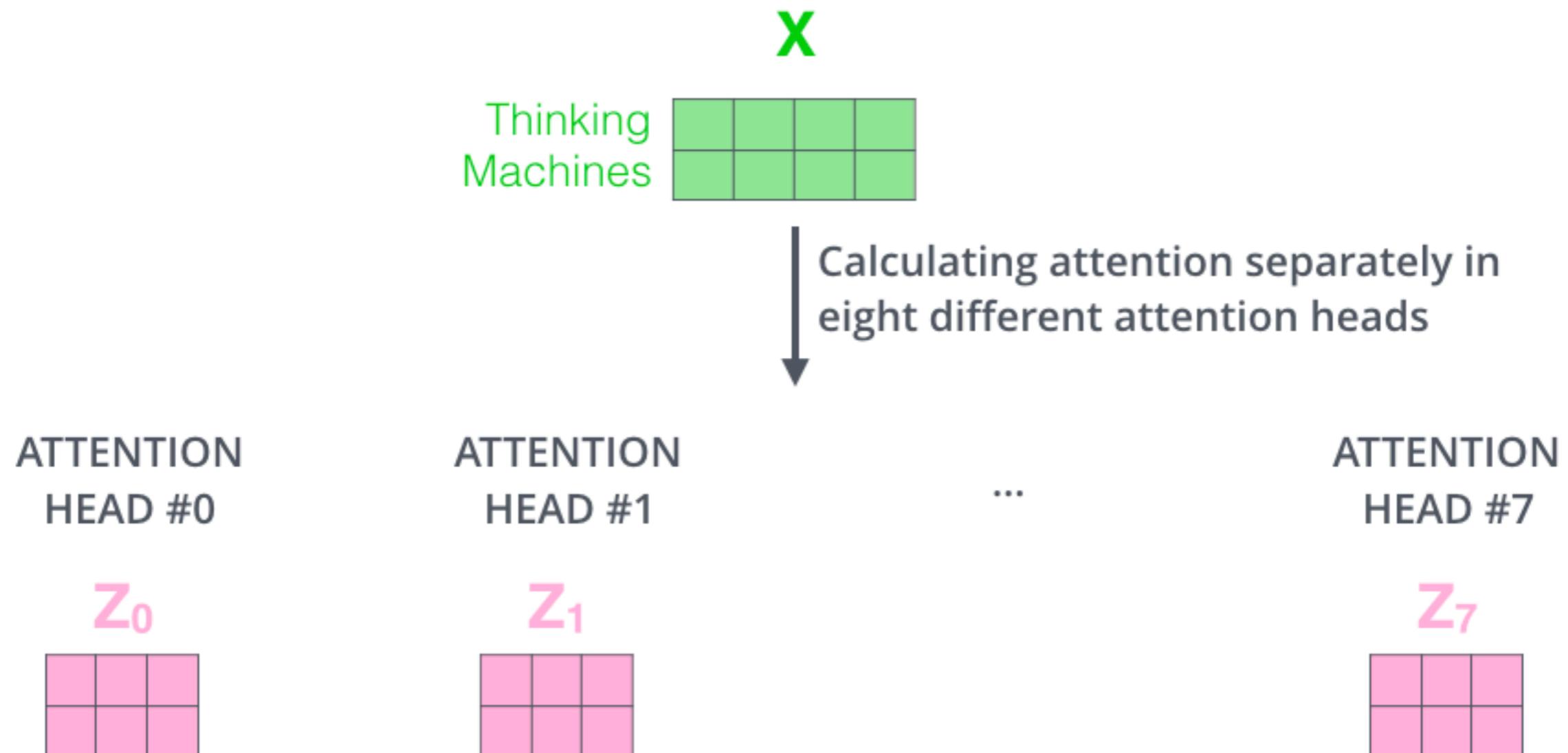
$$\text{softmax} \left( \frac{\begin{matrix} \mathbf{Q} & \mathbf{K}^T \\ \hline \end{matrix} \times \begin{matrix} \mathbf{V} \\ \hline \end{matrix}}{\sqrt{d_k}} \right) = \mathbf{Z}$$

The diagram illustrates the computation of attention scores. It shows three matrices:  $\mathbf{Q}$  (purple, 3x3),  $\mathbf{K}^T$  (orange, 3x3), and  $\mathbf{V}$  (blue, 3x3). The matrices  $\mathbf{Q}$  and  $\mathbf{K}^T$  are multiplied together, and the result is divided by  $\sqrt{d_k}$ . This result is then passed through a softmax function to produce the attention weights  $\mathbf{Z}$  (pink, 3x3).

We want the model to be able to interact words in multiple ways, so we use multi-headed attention



Each attention head is just a matrix that says how to combine the **query**, **key**, and **value** vectors



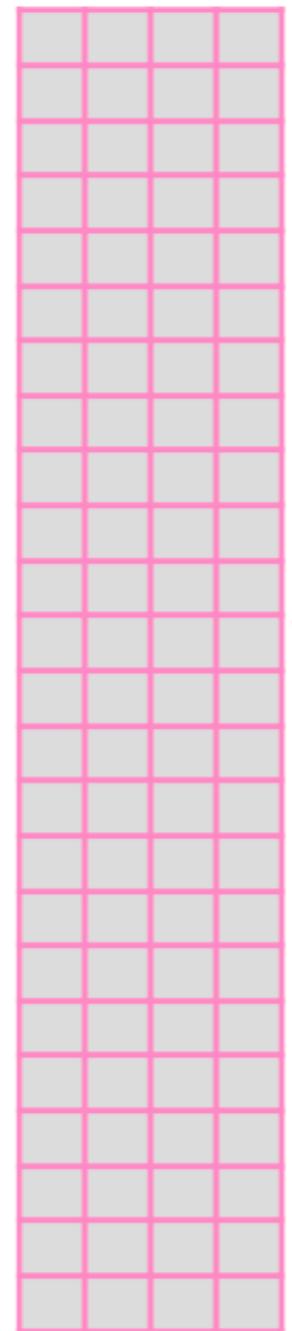
# Multi-headed attention itself is just a big matrix multiply!

1) Concatenate all the attention heads



2) Multiply with a weight matrix  $W^o$  that was trained jointly with the model

$X$



3) The result would be the  $Z$  matrix that captures information from all the attention heads. We can send this forward to the FFNN

$$= \begin{matrix} Z \\ \hline \end{matrix}$$

# Putting the whole thing together

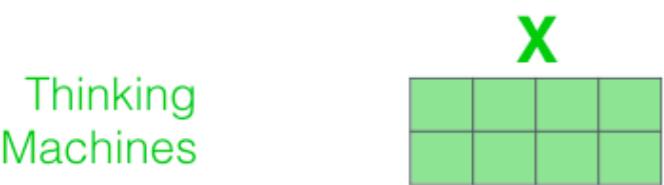
1) This is our  
input sentence\*

Thinking  
Machines

\* In all encoders other than #0,  
we don't need embedding.  
We start directly with the output  
of the encoder right below this one

# Putting the whole thing together

- 1) This is our input sentence\*
- 2) We embed each word\*



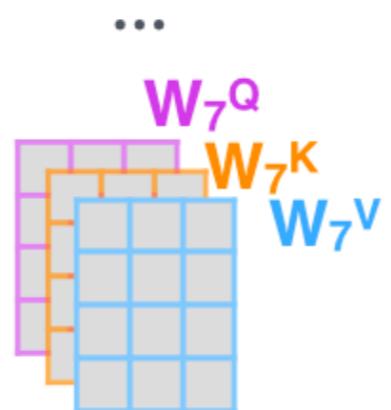
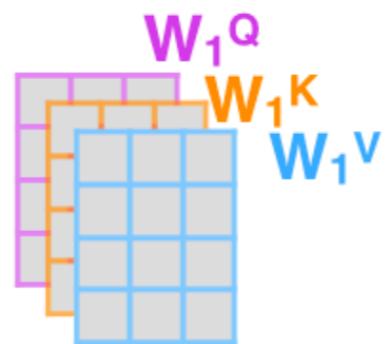
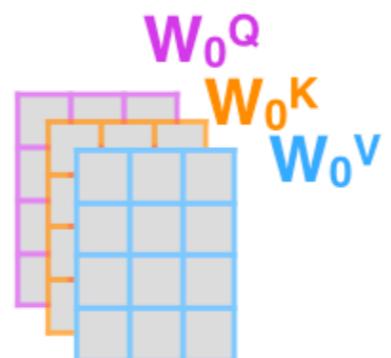
\* In all encoders other than #0,  
we don't need embedding.  
We start directly with the output  
of the encoder right below this one



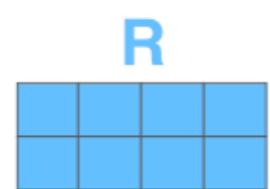
# Putting the whole thing together

1) This is our input sentence\*    2) We embed each word\*

3) Split into 8 heads.  
We multiply  $X$  or  $R$  with weight matrices



\* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



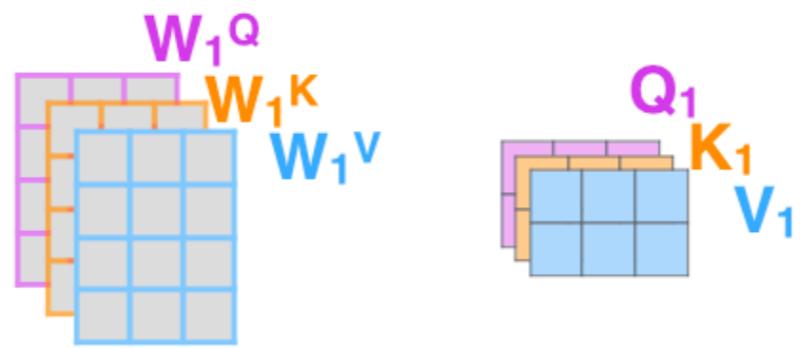
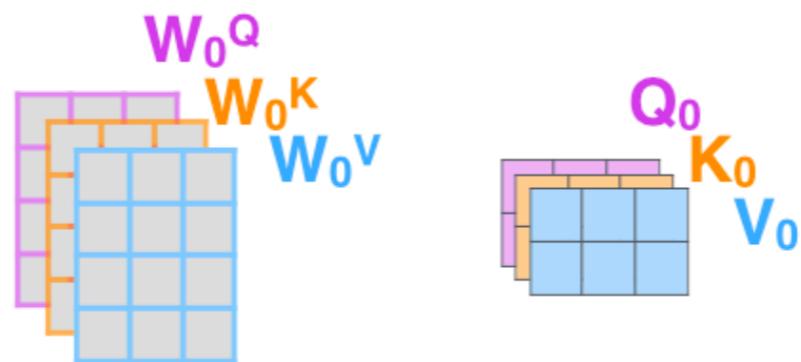
# Putting the whole thing together

1) This is our input sentence\*

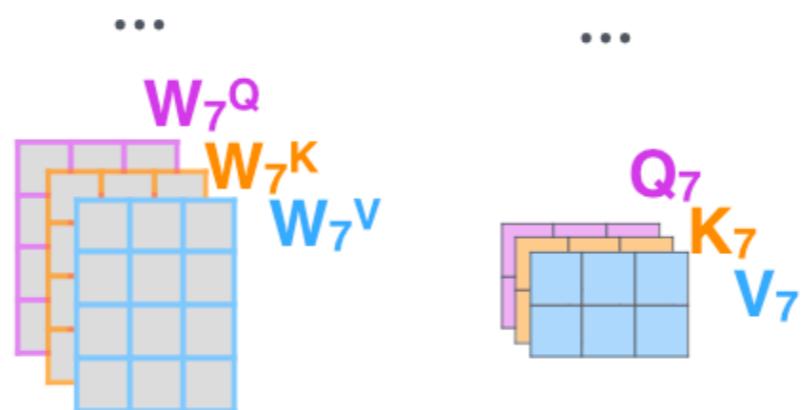
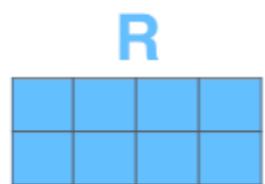
2) We embed each word\*

3) Split into 8 heads.  
We multiply  $X$  or  $R$  with weight matrices

4) Calculate attention using the resulting  $Q/K/V$  matrices

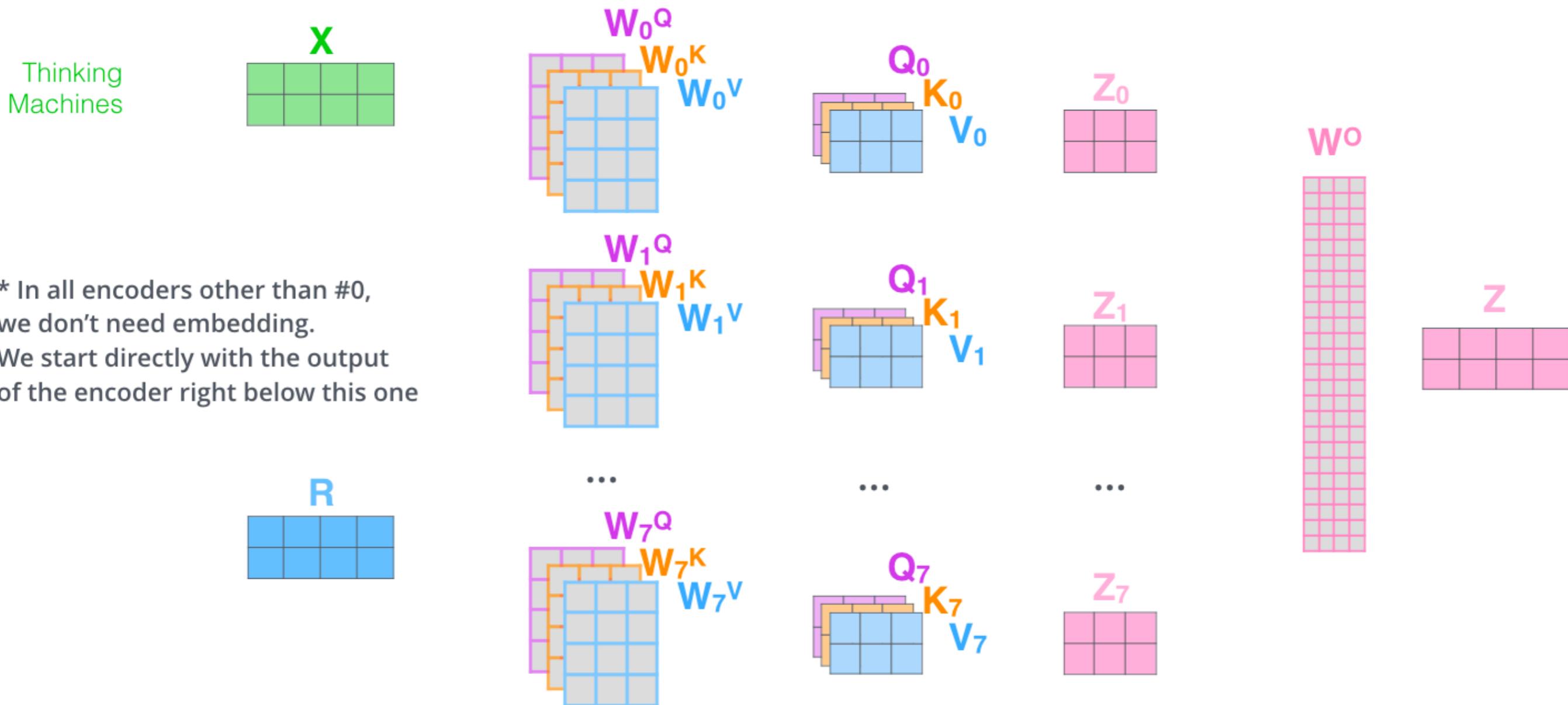


\* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

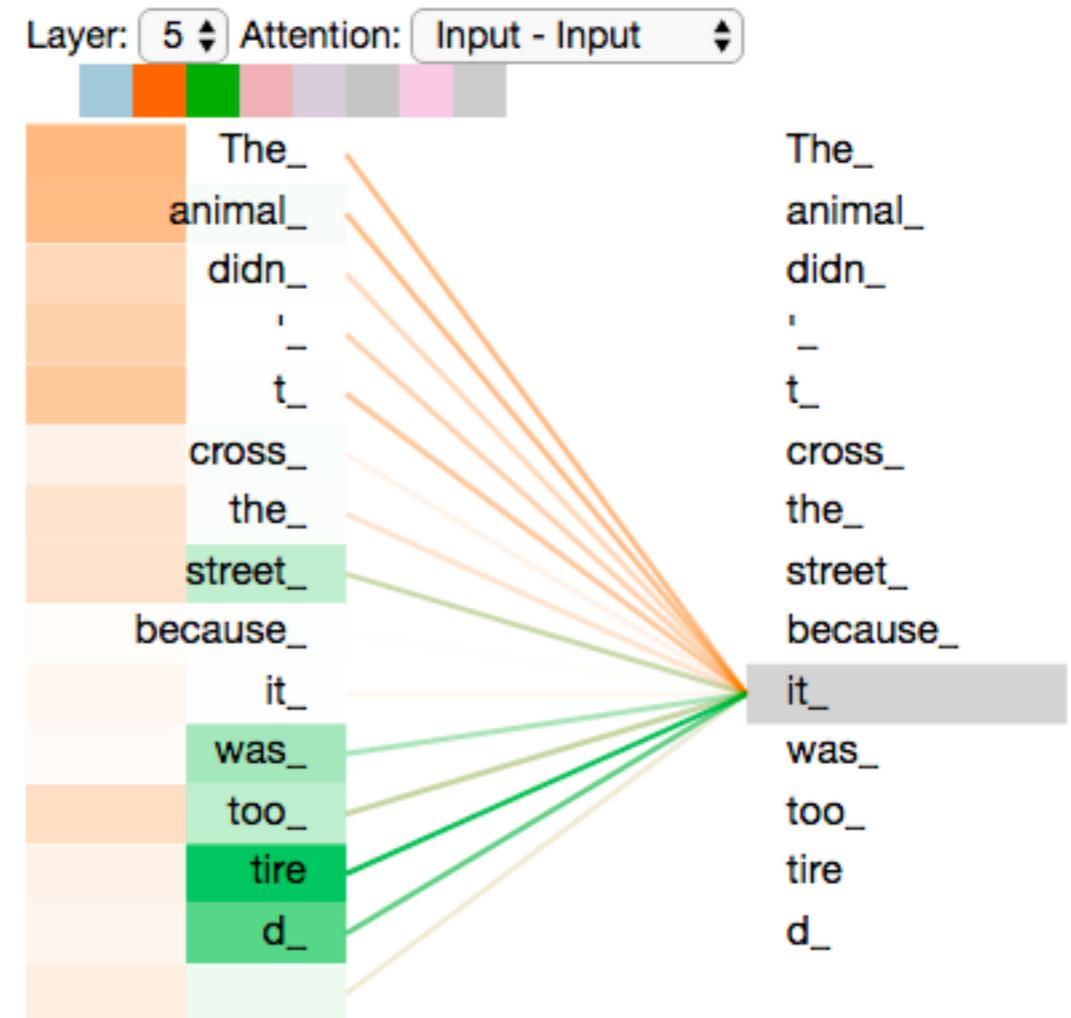
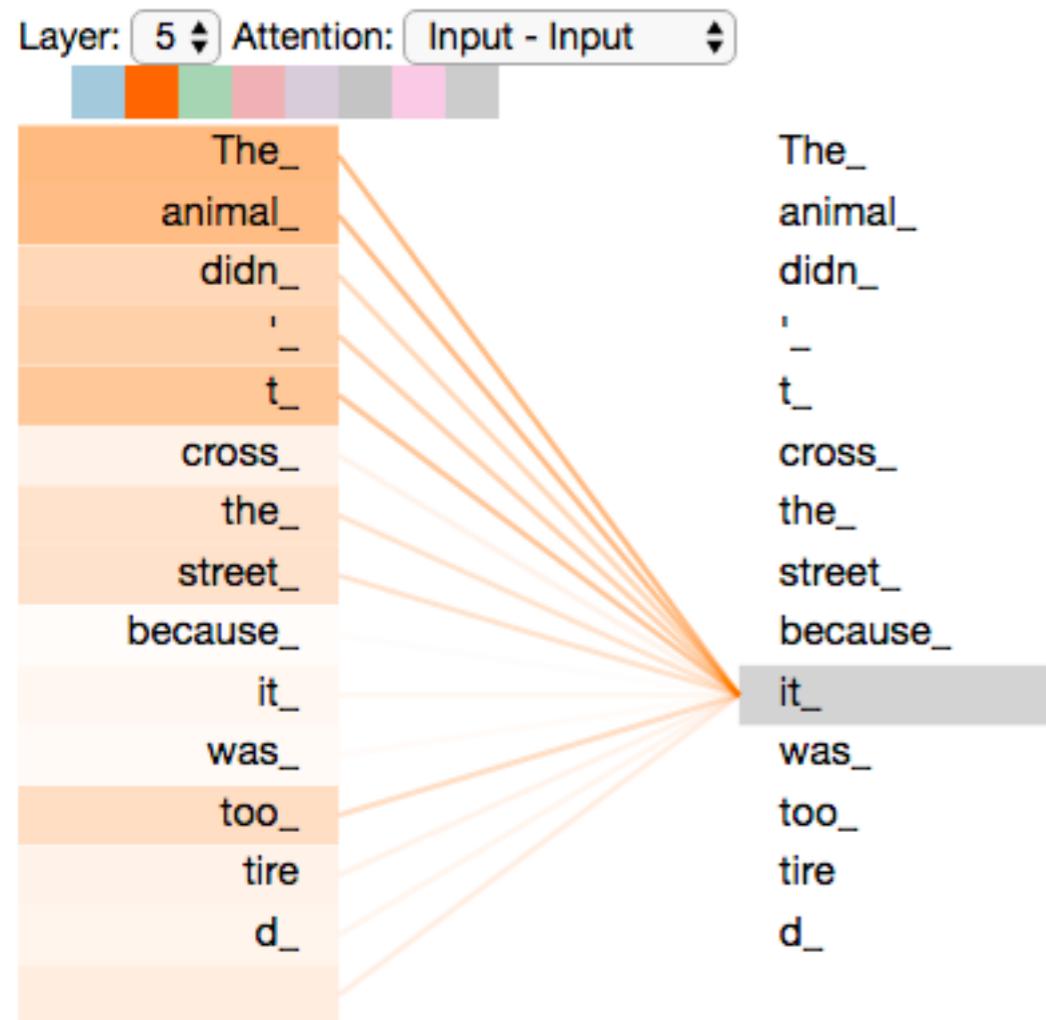


# Putting the whole thing together

- 1) This is our input sentence\*  $X$
- 2) We embed each word\*  $R$
- 3) Split into 8 heads. We multiply  $X$  or  $R$  with weight matrices  $W_0^Q, W_0^K, W_0^V$ ,  $W_1^Q, W_1^K, W_1^V$ , ...,  $W_7^Q, W_7^K, W_7^V$
- 4) Calculate attention using the resulting  $Q/K/V$  matrices  $Q_0, K_0, V_0$ ,  $Q_1, K_1, V_1$ , ...,  $Q_7, K_7, V_7$
- 5) Concatenate the resulting  $Z$  matrices, then multiply with weight matrix  $W^O$  to produce the output of the layer



# The attention heads learn how to pay attention to different parts of the sentence



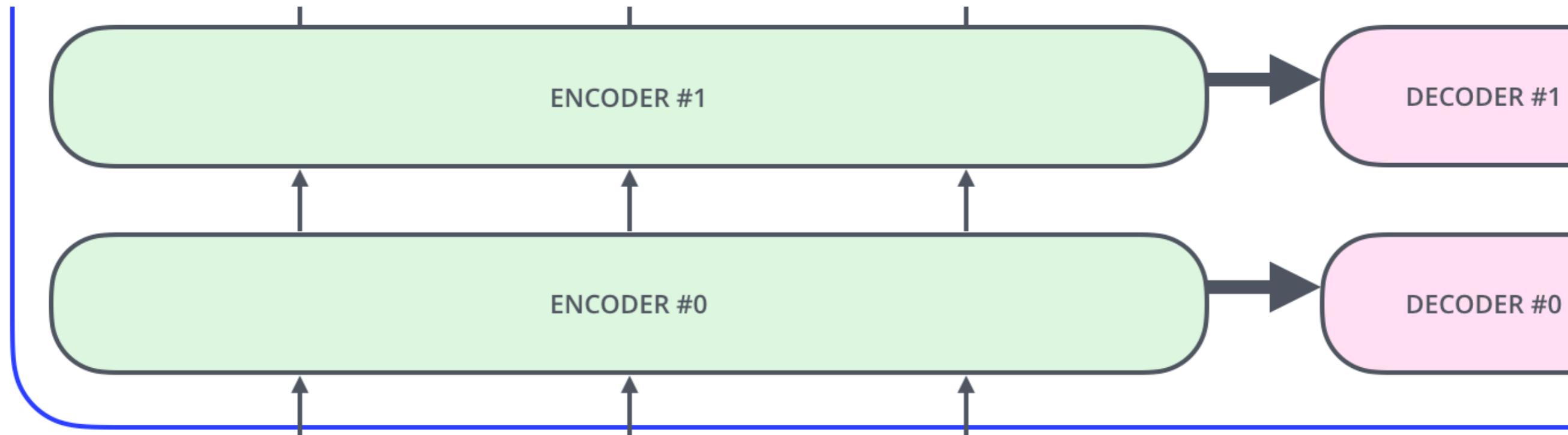
The  
animal  
didn't  
cross  
the  
street  
because  
it  
was  
too  
tired  
.

The  
animal  
didn't  
cross  
the  
street  
because  
it  
was  
too  
tired

The  
animal  
didn't  
cross  
the  
street  
because  
it  
was  
too  
wide  
.

The  
animal  
didn't  
cross  
the  
street  
because  
it  
was  
too  
wide

# How do we keep track of where the word is?: Positional Encodings



EMBEDDING  
WITH TIME  
SIGNAL

$$\mathbf{x}_1 \quad \boxed{\text{light green}} \quad \boxed{\text{light green}} \quad \boxed{\text{light green}} \quad \boxed{\text{light green}}$$

$$\mathbf{x}_2 \quad \boxed{\text{light green}} \quad \boxed{\text{light green}} \quad \boxed{\text{light green}} \quad \boxed{\text{light green}}$$

$$\mathbf{x}_3 \quad \boxed{\text{light green}} \quad \boxed{\text{light green}} \quad \boxed{\text{light green}} \quad \boxed{\text{light green}}$$

POSITIONAL  
ENCODING

$$\mathbf{t}_1 \quad \boxed{\text{yellow}} \quad \boxed{\text{yellow}} \quad \boxed{\text{yellow}} \quad \boxed{\text{yellow}}$$

$$\mathbf{t}_2 \quad \boxed{\text{yellow}} \quad \boxed{\text{yellow}} \quad \boxed{\text{yellow}} \quad \boxed{\text{yellow}}$$

$$\mathbf{t}_3 \quad \boxed{\text{yellow}} \quad \boxed{\text{yellow}} \quad \boxed{\text{yellow}} \quad \boxed{\text{yellow}}$$

EMBEDDINGS

$$\mathbf{x}_1 \quad \boxed{\text{green}} \quad \boxed{\text{green}} \quad \boxed{\text{green}} \quad \boxed{\text{green}}$$

$$\mathbf{x}_2 \quad \boxed{\text{green}} \quad \boxed{\text{green}} \quad \boxed{\text{green}} \quad \boxed{\text{green}}$$

$$\mathbf{x}_3 \quad \boxed{\text{green}} \quad \boxed{\text{green}} \quad \boxed{\text{green}} \quad \boxed{\text{green}}$$

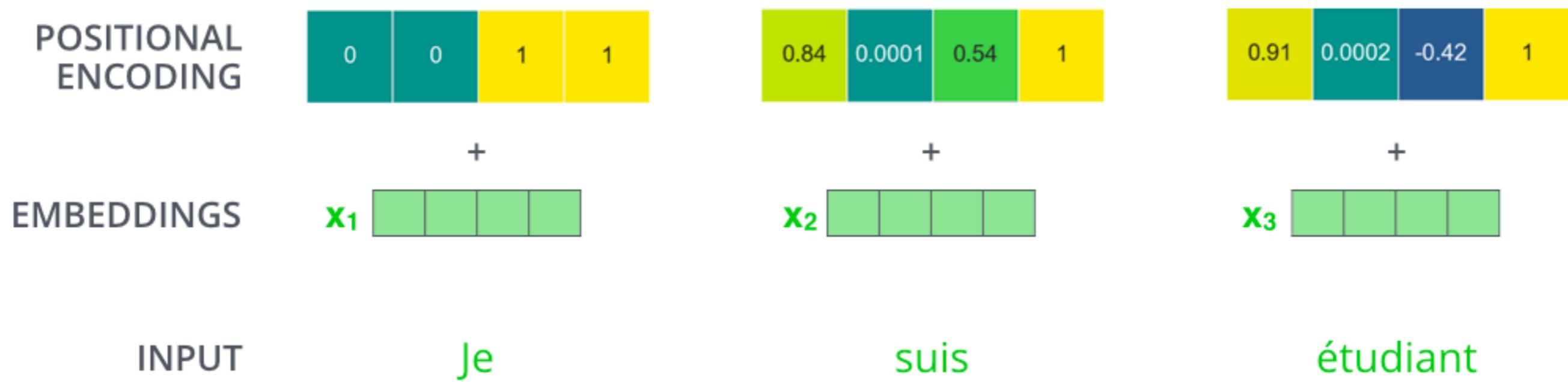
INPUT

Je

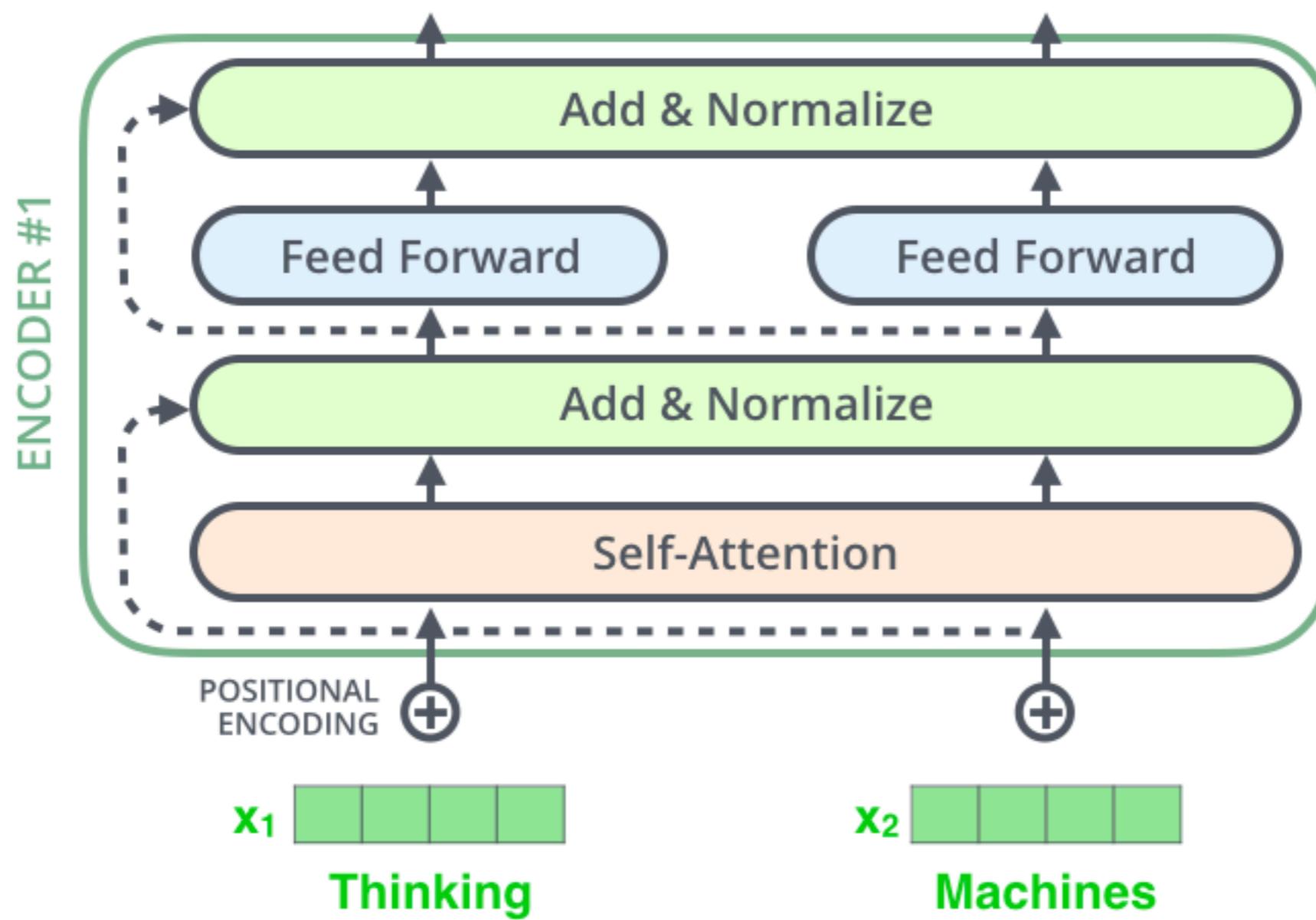
suis

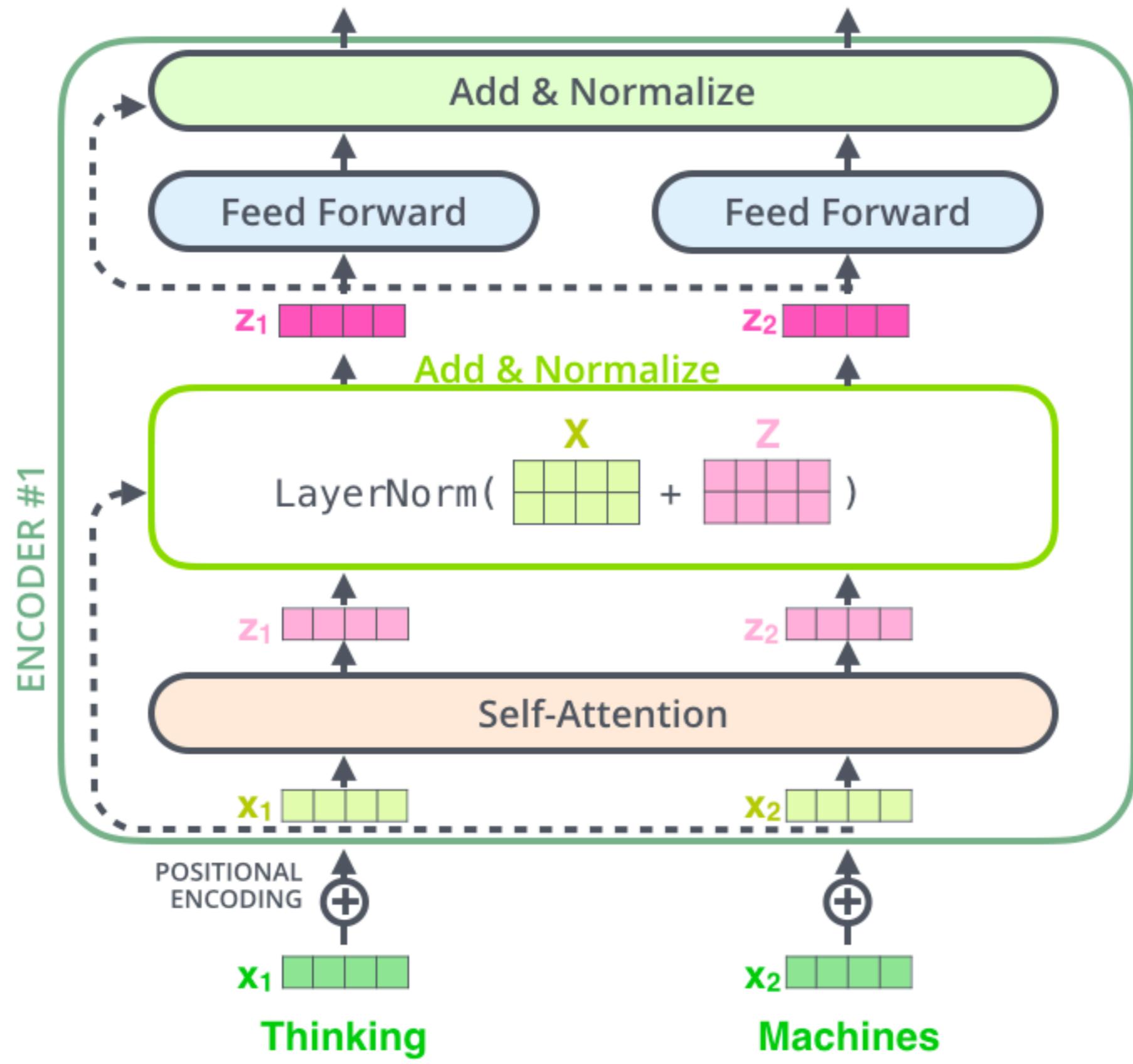
étudiant

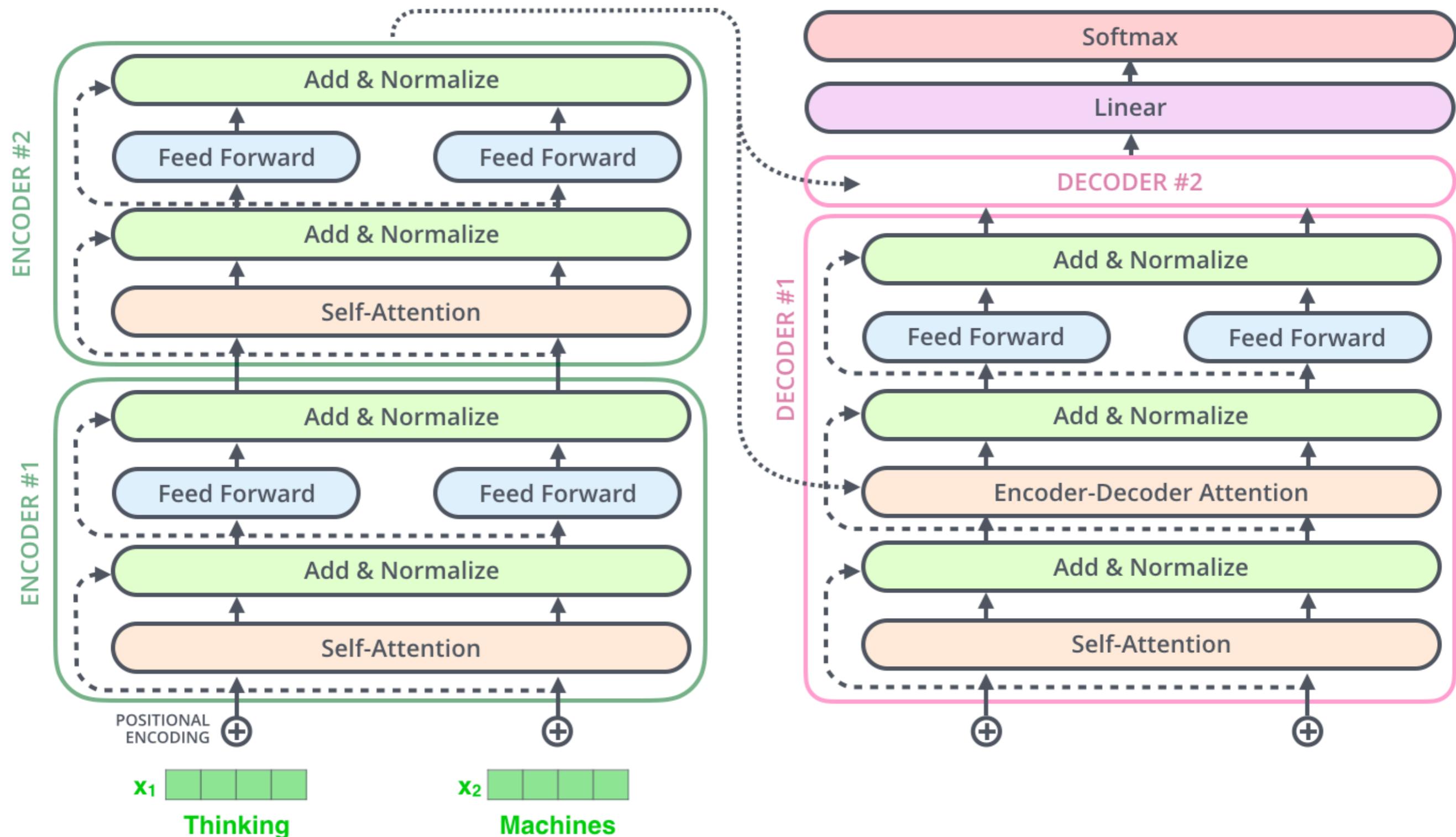
Each position in the sentence gets a unique vector that is added to the input embedding



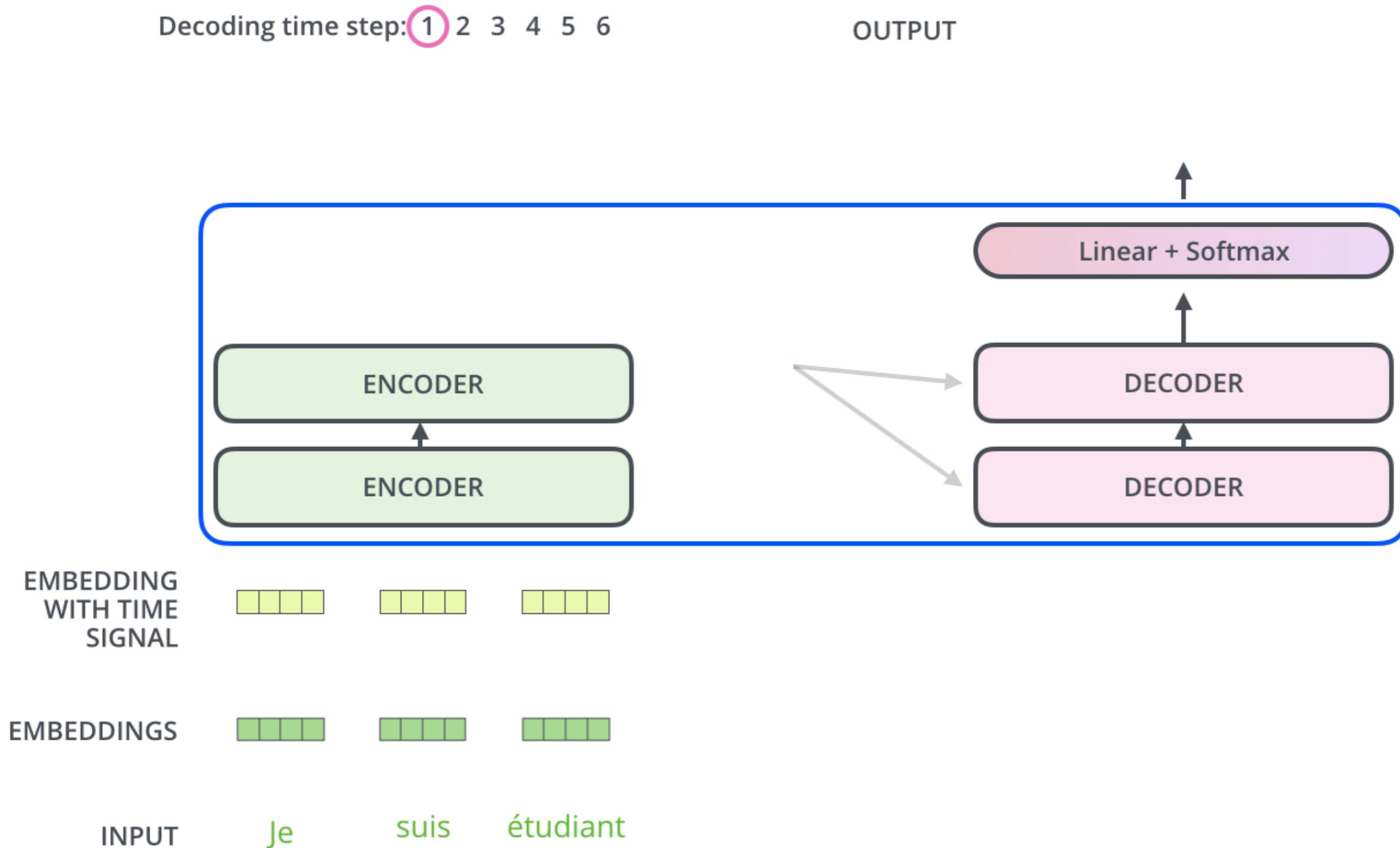
# The whole encoder



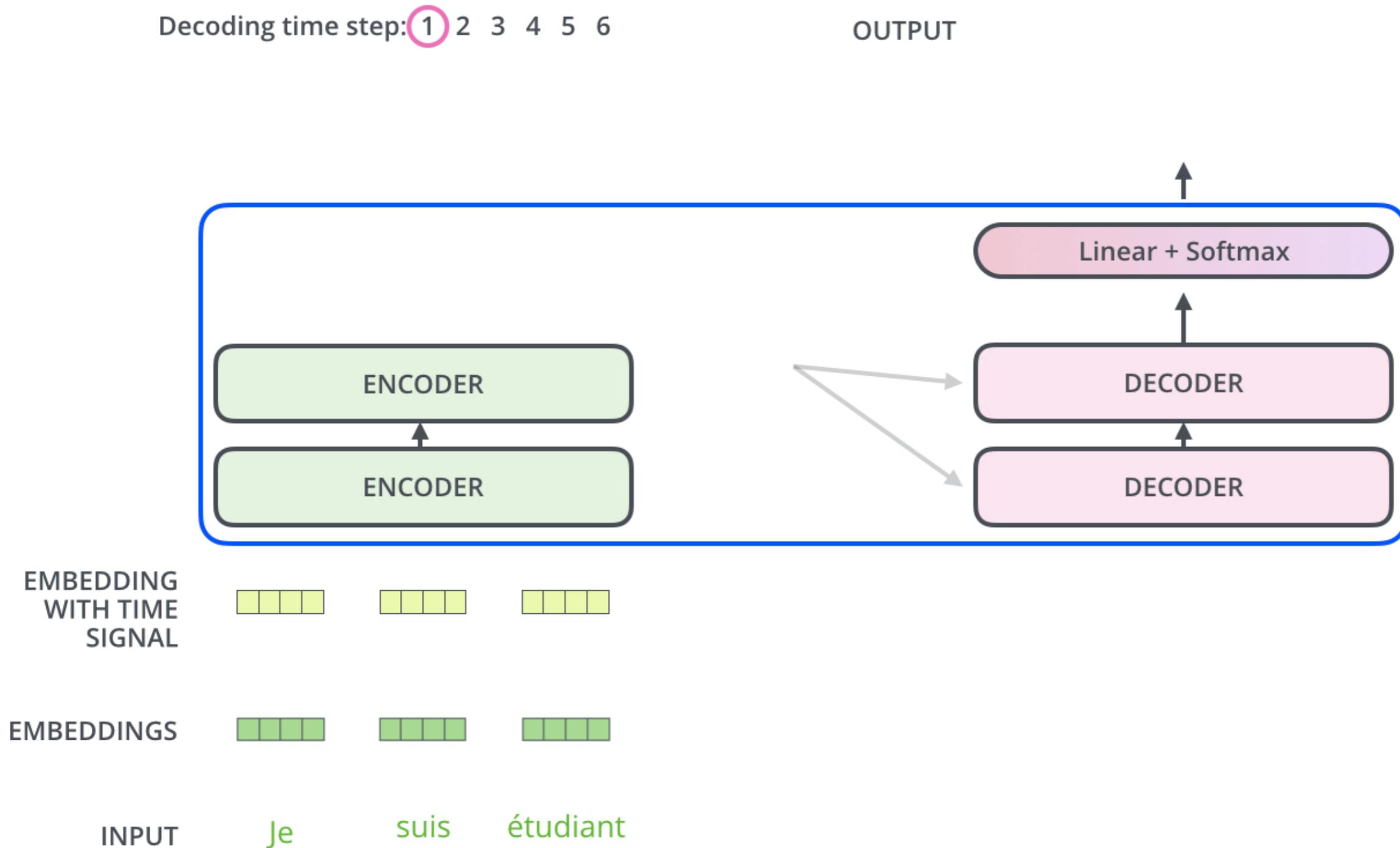




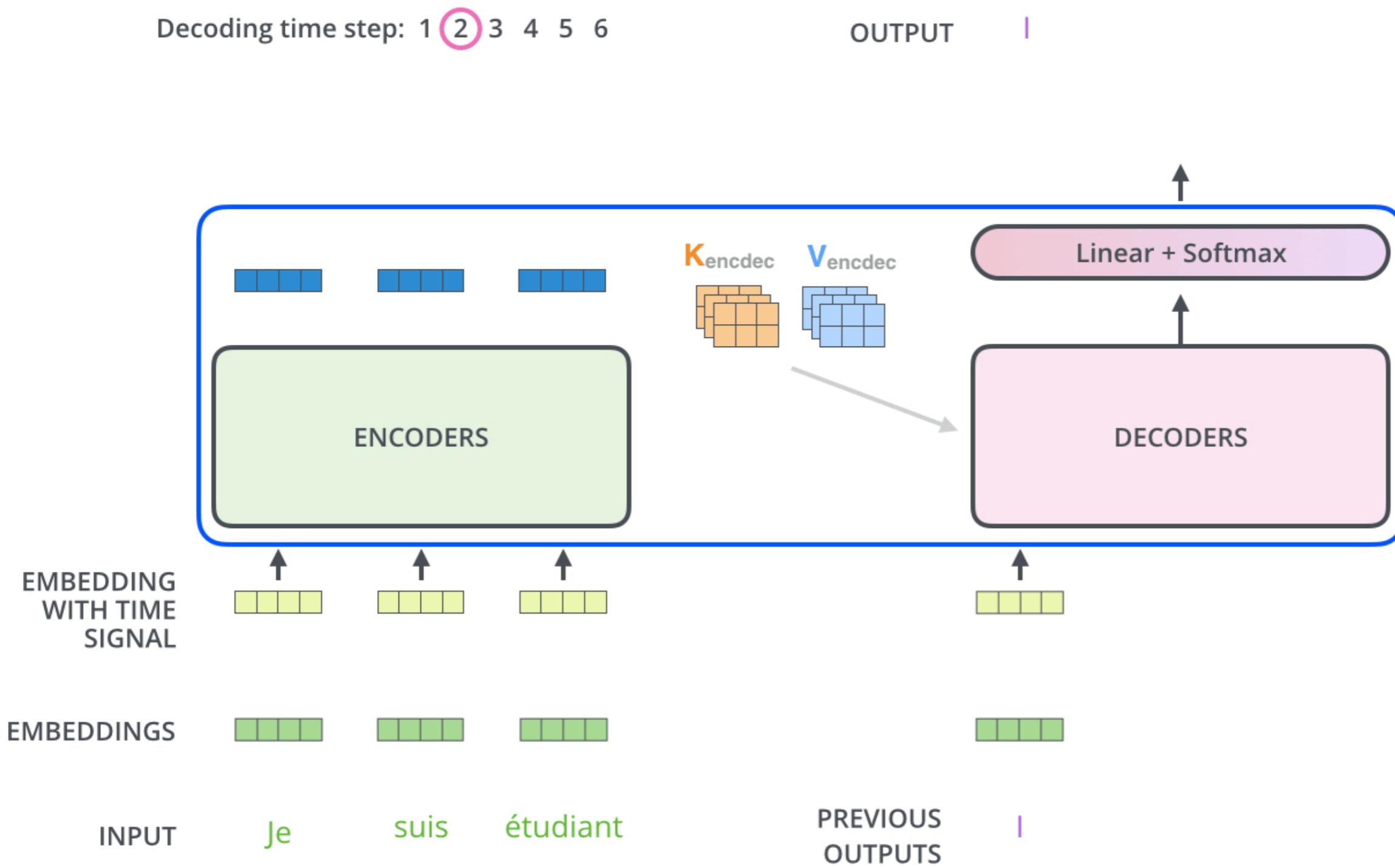
# Feeding the encoder into the decoder...



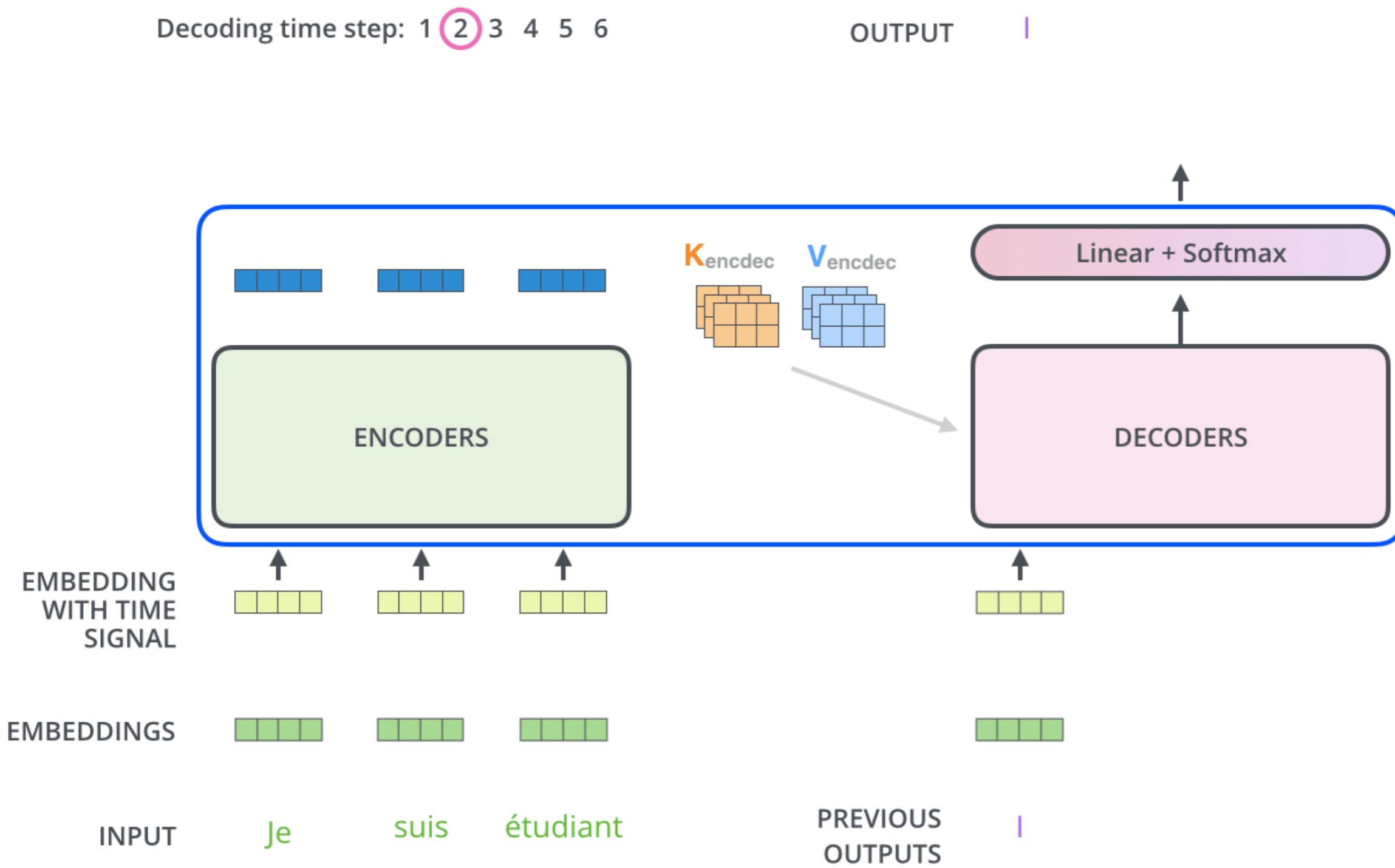
# Feeding the encoder into the decoder...



# Feeding the encoder into the decoder...



# Feeding the encoder into the decoder...



# But does it work?

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1		$3.3 \cdot 10^{18}$
Transformer (big)	<b>28.4</b>	<b>41.0</b>		$2.3 \cdot 10^{19}$

# Lots of fun good readings

- <http://jalammar.github.io/illustrated-transformer/> – Source for all these figures
- <http://nlp.seas.harvard.edu/2018/04/03/attention.html> – PyTorch code!
- <http://mlexplained.com/2017/12/29/attention-is-all-you-need-explained/> – more PyTorch with commentary



# **Machine Translation**

## **(MT)**

# Machine Translation

# Machine Translation

- Fully automatic

# Machine Translation

- Fully automatic

Enter Source Text:

Translation from Stanford's *Phrasal*:

# Machine Translation

- Fully automatic

Enter Source Text:

这 不 过 是 一 个 时 间 的 问 题 .

Translation from Stanford's *Phrasal*:

# Machine Translation

- Fully automatic

Enter Source Text:

这 不 过 是 一 个 时 间 的 问 题 .

Translation from Stanford's *Phrasal*:

This is only a matter of time.

# Machine Translation

- Fully automatic
- Helps human translators

Enter Source Text:

这 不 过 是 一 个 时 间 的 问 题 .

Translation from Stanford's *Phrasal*:

This is only a matter of time.

# Machine Translation

- Fully automatic
- Helps human translators

Enter Source Text:

这 不 过 是 一 个 时 间 的 问 题 .

Translation from Stanford's *Phrasal*:

This is only a matter of time.

Enter Source Text:

لـة عنيفة في مجلس النواب الذي انعقد امس في جلسة تشريعية عادلة تحولت  
على موقف +ه من المحكمة الدولية و "الملاحظات" التي ادلى ب#+ها  
حول هذا الموضوع .

Translate Clear

Enter Translation:

lebanese |

president  
suffered  
exposed

# Google Translate

- Fried ripe plantains:
- [http://laylita.com/recetas/2008/02/28/  
platanos-maduros-fritos/](http://laylita.com/recetas/2008/02/28/platanos-maduros-fritos/)

# Machine Translation

黛玉自在枕上感念宝钗。。。又听见窗外竹梢焦叶之上，  
雨声淅沥，清寒透幕，不觉又滴下泪来。

- The Story of the Stone (“The Dream of the Red Chamber”)
  - Cao Xueqin 1792
- **Chinese gloss:** Dai-yu alone at bed on think-of-with-gratitude Bao-chai... again listen to window outside bamboo tip plantain leaf of on, rain sound sigh drop, clear cold penetrate curtain, not feeling again fall down tears come.
- **Hawkes translation:** As she lay there alone, Dai-yu's thoughts turned to Bao-chai... Then she listened to the insistent rustle of the rain on the bamboos and plantains outside her window. The coldness penetrated the curtains of her bed. Almost without noticing it she had begun to cry.

# Difficulties in Chinese to English translation

# Difficulties in Chinese to English translation

- Long Chinese sentences: 4 English sentences to 1 Chinese

# Difficulties in Chinese to English translation

- Long Chinese sentences: 4 English sentences to 1 Chinese
- Chinese no pronouns or articles (English **the, a**)

# Difficulties in Chinese to English translation

- Long Chinese sentences: 4 English sentences to 1 Chinese
- Chinese no pronouns or articles (English **the, a**)
- Chinese has locative post-positions, English prepositions
  - Chinese **bed on, window outside**, English **on the bed, outside the window**

# Difficulties in Chinese to English translation

- Long Chinese sentences: 4 English sentences to 1 Chinese
- Chinese no pronouns or articles (English **the, a**)
- Chinese has locative post-positions, English prepositions
  - Chinese **bed on, window outside**, English **on the bed, outside the window**
- Chinese rarely marks tense:
  - English **as, turned to, had begun,**
  - Chinese **tou, 'penetrate'** -> English **penetrated**

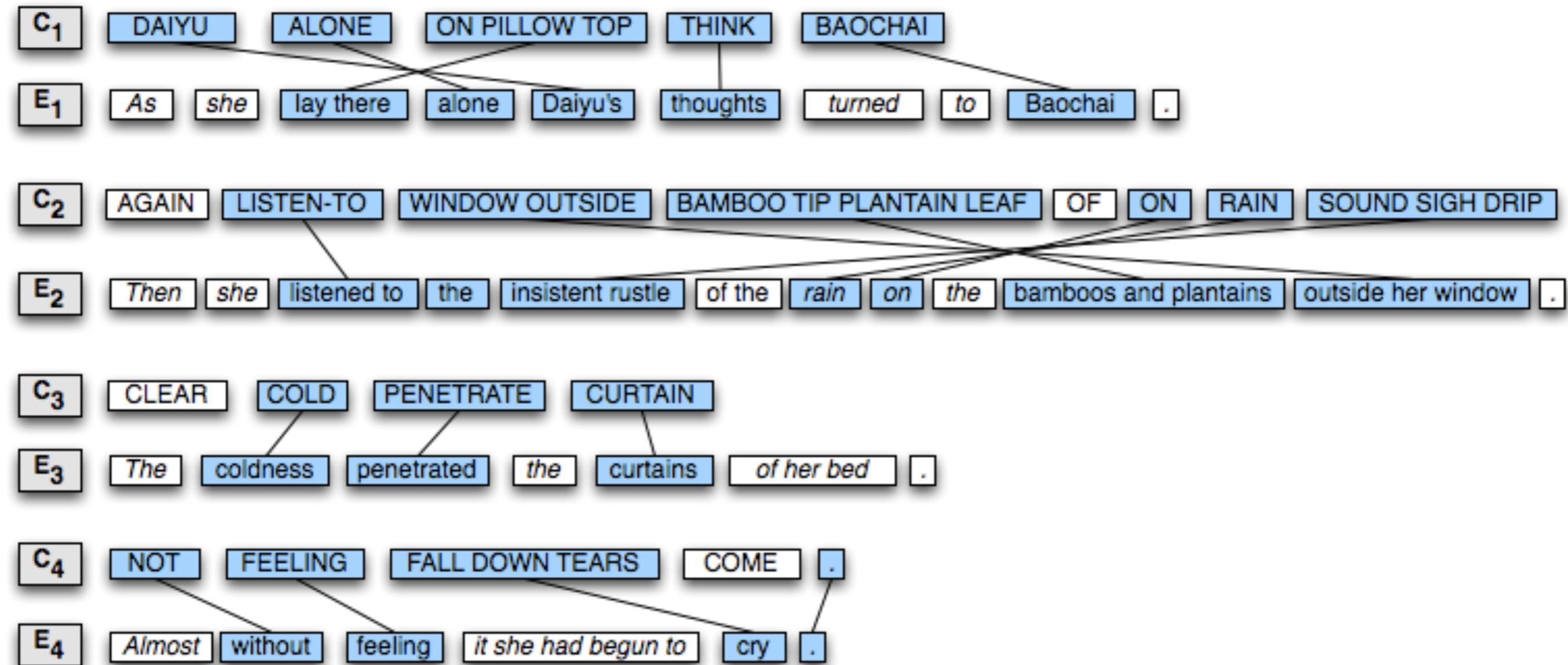
# Difficulties in Chinese to English translation

- Long Chinese sentences: 4 English sentences to 1 Chinese
- Chinese no pronouns or articles (English **the, a**)
- Chinese has locative post-positions, English prepositions
  - Chinese **bed on, window outside**, English **on the bed, outside the window**
- Chinese rarely marks tense:
  - English **as, turned to, had begun,**
  - Chinese **tou, 'penetrate'** -> English **penetrated**
- Chinese relative clauses are before the noun, English after
  - Chinese: **[window outside bamboo on] rain**
  - English: **rain [on the bamboo outside the window]**

# Difficulties in Chinese to English translation

- Long Chinese sentences: 4 English sentences to 1 Chinese
- Chinese no pronouns or articles (English **the, a**)
- Chinese has locative post-positions, English prepositions
  - Chinese **bed on, window outside**, English **on the bed, outside the window**
- Chinese rarely marks tense:
  - English **as, turned to, had begun,**
  - Chinese **tou, 'penetrate'** -> English **penetrated**
- Chinese relative clauses are before the noun, English after
  - Chinese: **[window outside bamboo on] rain**
  - English: **rain [on the bamboo outside the window]**
- Stylistic and cultural differences
  - Chinese **bamboo tip plaintain leaf** -> bamboos and plantains
  - Chinese **rain sound sigh drop** -> insistent rustle of the rain
  - Chinese **ma 'curtain'** -> curtains of her bed

# Alignment in Machine Translation



# Early MT History

1946 Booth and Weaver discuss MT in New York

1947-48 idea of dictionary-based direct translation

1947 Warren Weaver suggests translation by computer

1949 Weaver memorandum

1952 all 18 MT researchers in world meet at MIT

1954 IBM/Georgetown Demo Russian-English MT

1955-65 lots of labs take up MT

<http://www.hutchinsweb.me.uk/PPF-TOC.htm>

# Warren Weaver (1947)



ingcmpnqsnwf cv fpn owoktvcv

hu ihgzswnfv rqcffnw cw owgcnwf

kowazoanv . . .

# Warren Weaver (1947)



e        e        e                    e  
**ingcmpnqsnwf cv fpn owoktvcv**  
                  e                    e                    e  
**hu ihgzswnfv rqcffnw cw owgcnwf**  
                  e  
**kowazoanv . . .**

# Warren Weaver (1947)



e e e                    the  
ingcmpnqsnwf cv fpn owoktvcv  
e                        e                        e  
hu ihgzswnfv rqcffnw cw owgcnwf  
e  
kowazoanv . . .

# Warren Weaver (1947)



e      he    e                the  
**ingcmpnqsnwf cv fpn owoktvcv**  
e                                  e                                    e t  
**hu ihgzswnfv rqcffnw cw owgcnwf**  
e  
**kowazoanv . . .**

# Warren Weaver (1947)



e he e of the  
**ingcmpnqsnwf cv fpn owoktvcv**  
e e e t  
**hu ihgzswnfv rqcffnw cw owgcnwf**  
e  
**kowazoanv . . .**

# Warren Weaver (1947)



e he e of the f of  
**ingcmpnqsnwf cv fpn owoktvcv**  
e f o e o oe t  
**hu ihgzswnfv rqcffnw cw owgcnwf**  
ef  
**kowazoanv . . .**

# Warren Weaver (1947)



e      he    e      ~~of~~the  
**ingcmpnqsnwf**    cv    fpn    owoktvcv  
e                        e                        e t  
**hu ihgzswnfv**    **rqcffnw**    cw    **owgcnwf**  
e  
**kowazoanv** . . .

# Warren Weaver (1947)



e he e is the sis  
**ingcmpnqsnwf cv fpn owoktvcv**  
e s i e i ie t  
**hu ihgzswnfv rqcffnw cw owgcnwf**  
es  
**kowazoanv . . .**

# Warren Weaver (1947)



decipherment is the analysis  
**ingcmpnqsnwf cv fpn owoktvcv**  
of documents written in ancient  
**hu ihgzswnfv rqcffnw cw owgcnwf**  
languages . . .  
**kowazoanv . . .**

# Warren Weaver (1947)

Can this be  
computerized?

The non-Turkish guy next to  
me is even deciphering  
Turkish! All he needs is a  
statistical table of letter-pair  
frequencies in Turkish ...



Collected mechanically from a  
Turkish body of text, or *corpus*



“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, March 1947

<http://www.mt-archive.info/Weaver-1949.pdf>



“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, March 1947

<http://www.mt-archive.info/Weaver-1949.pdf>



“... as to the problem of mechanical translation, I frankly am afraid that the [semantic] boundaries of words in different languages are too vague ... to make any quasi-mechanical translation scheme very hopeful.”

- Norbert Wiener, April 1947

# The History of MT: Pessimism

- 1959/1960
- Yehoshua Bar-Hillel “Report on the state of MT in US and GB”
  - Fully automatic high quality (FAHQ) MT too hard because we would have to encode all of human knowledge
  - Instead we should work on computer tools for human translators

# The claim that fully automatic high quality MT is impossible

Yehoshua Bar-Hillel. 1960. A Demonstration of the Nonfeasibility of Fully Automatic High Quality Translation.

- Little John was looking for his toy box. Finally he found it. *The box was in the pen.* John was very happy.

# The claim that fully automatic high quality MT is impossible

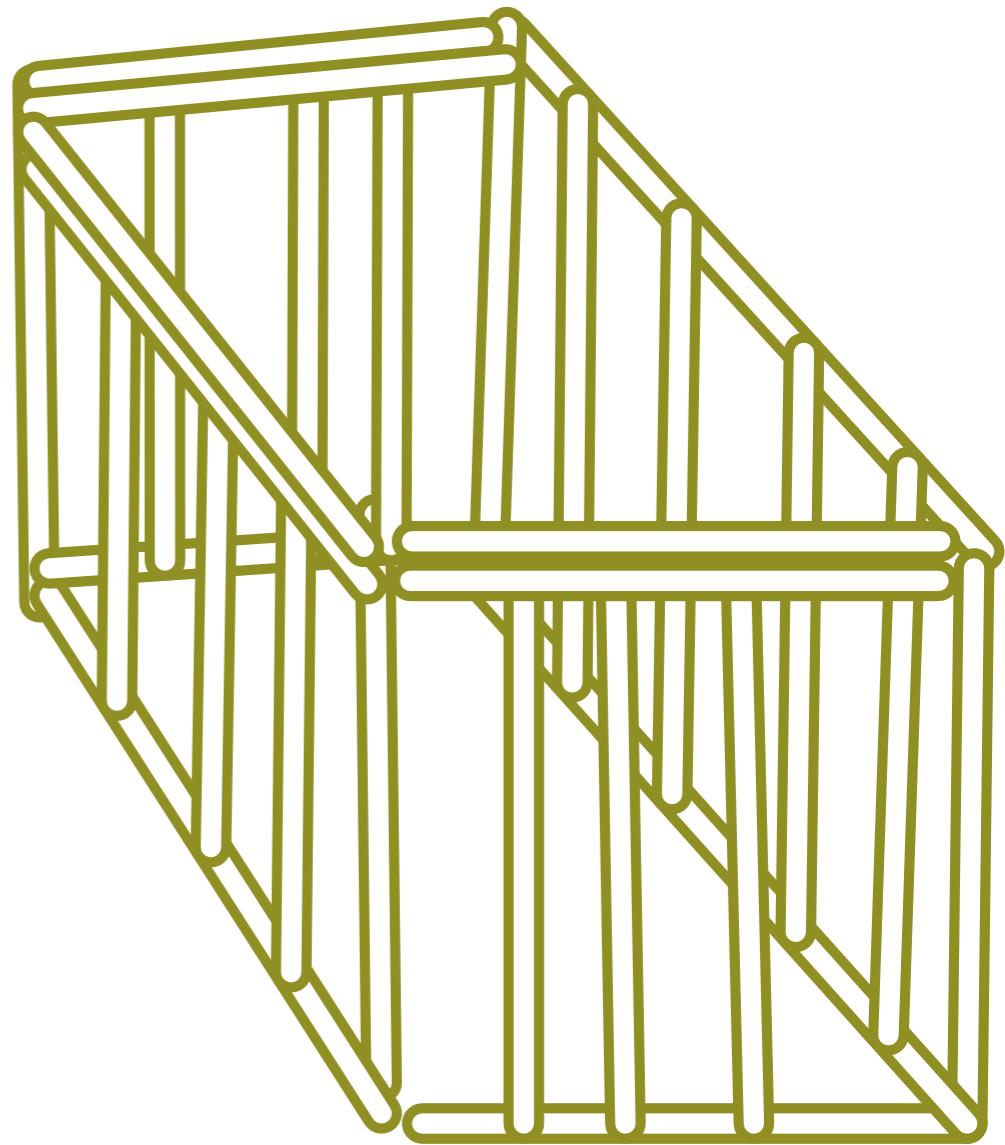
Yehoshua Bar-Hillel. 1960. A Demonstration of the Nonfeasibility of Fully Automatic High Quality Translation.

- Little John was looking for his toy box. Finally he found it. *The box was in the pen.* John was very happy.
  - Pen<sub>1</sub>: Enclosure for small children
  - Pen<sub>2</sub>: Writing utensil

The claim that fully automatic  
high quality MT is impossible

- Pen<sub>1</sub>: Enclosure for small children

- The box was in the pen.



# The claim that fully automatic high quality MT is impossible

Yehoshua Bar-Hillel, 1960

“I now claim that no existing or imaginable program will enable an electronic computer to determine...”

# The state of the art in MT

The image shows a screenshot of the Google Translate interface. At the top left is the Google logo. Below it, the word "Translate" is written in red. To the right are four buttons: "From: English" with a dropdown arrow, a bidirectional arrow icon, "To: French" with a dropdown arrow, and a blue "Translate" button. Below these buttons, there are language selection tabs for Chinese, English, Spanish, French, and Chinese. The English tab is highlighted. On the left side of the interface, there are two input fields. The top field contains the sentence "The box was in the pen". The bottom field contains the sentence "The pen was in the box". On the right side, the corresponding French translations are shown: "La boîte était dans l'enclos" and "La plume était dans la boîte". A small "X" icon is located above the second French translation, and a microphone icon is located below the first French translation.

From: English ▾

To: French ▾

Translate

Chinese English Spanish

English French Chinese

The box was in the pen

The pen was in the box

La boîte était dans l'enclos

La plume était dans la boîte

# History of MT: Further Pessimism

## The ALPAC report

# History of MT: Further Pessimism

## The ALPAC report

- Headed by John R. Pierce of Bell Labs

# History of MT: Further Pessimism

## The ALPAC report

- Headed by John R. Pierce of Bell Labs
- Conclusions:
  - MT doesn't work
    - MT a failure: all current MT work had to be post-edited
    - Intelligibility and informativeness worse than human
  - We don't need MT anyhow
    - Already too many human translators from Russian

# History of MT: Further Pessimism

## The ALPAC report

- Headed by John R. Pierce of Bell Labs
- Conclusions:
  - MT doesn't work
    - MT a failure: all current MT work had to be post-edited
    - Intelligibility and informativeness worse than human
  - We don't need MT anyhow
    - Already too many human translators from Russian
- Results: MT research suffered
  - Funding loss
  - Number of research labs declined
  - Association for Machine Translation and Computational Linguistics dropped MT from its name

# MT in the modern age

# MT in the modern age

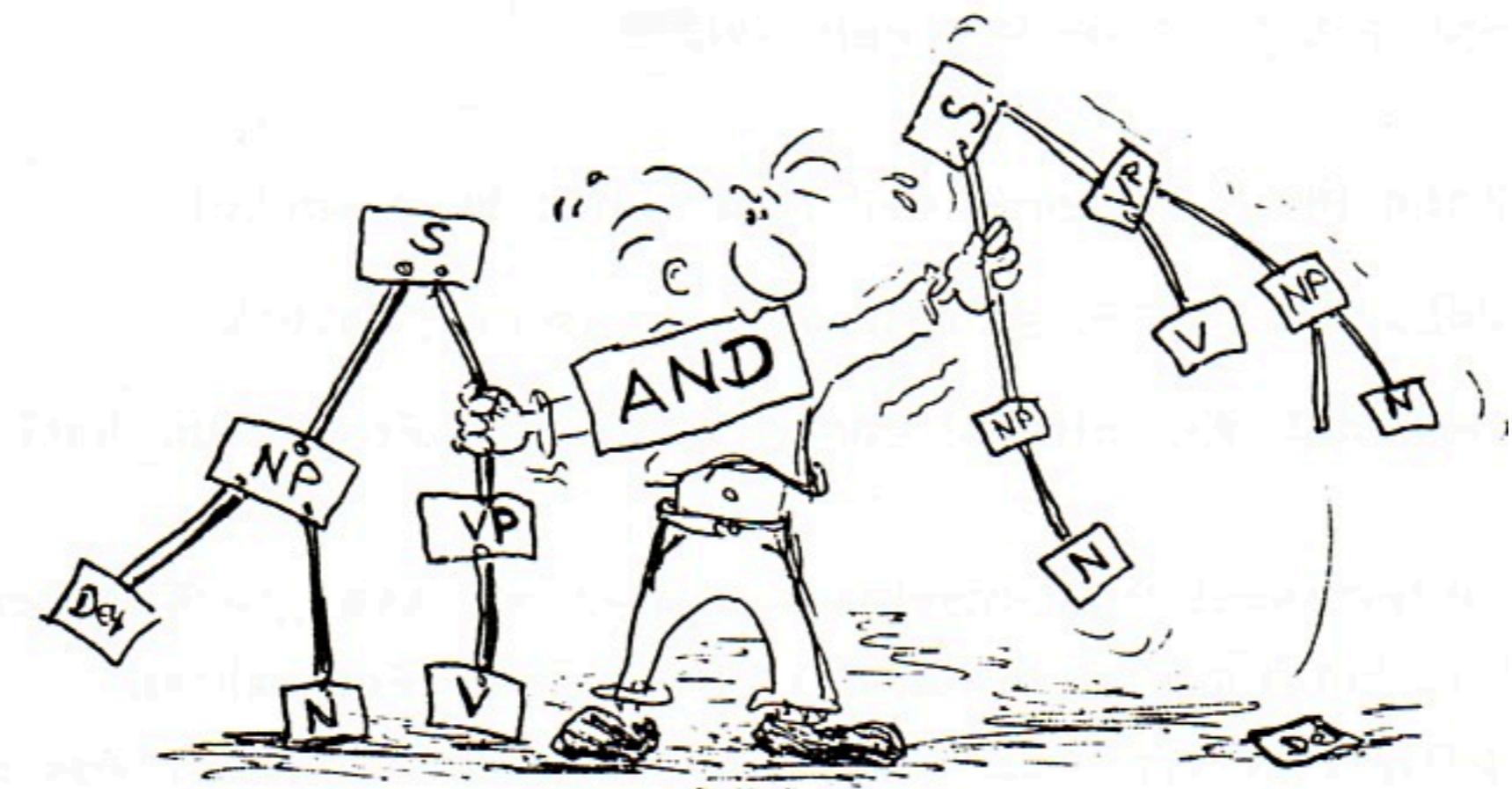
- 1975-1985: Resurgence of MT in Europe and Japan
  - Domain-specific rule-based systems

# MT in the modern age

- 1975-1985: Resurgence of MT in Europe and Japan
  - Domain-specific rule-based systems
- 1990-2013: Rise of Statistical Machine Translation

# MT in the modern age

- 1975-1985: Resurgence of MT in Europe and Japan
  - Domain-specific rule-based systems
- 1990-2013: Rise of Statistical Machine Translation
- 2013-present: Rise of *Neural* Machine Translation



# The Linguistics of MT

# Language Similarities and Divergences

- Typology:
  - the study of systematic cross-linguistic similarities and differences
- What are the dimensions along which human languages vary?

# Syntactic Variation: Basic Word Orders

In many languages one word order is more basic

# Syntactic Variation: Basic Word Orders

In many languages one word order is more basic

- SVO (Subject-Verb-Object) languages

English, German, French, Mandarin

I baked a pizza

# Syntactic Variation: Basic Word Orders

In many languages one word order is more basic

- SVO (Subject-Verb-Object) languages

English, German, French, Mandarin

I baked a pizza

- SOV Languages

Japanese, Hindi

English: He adores listening to music

Japanese: kare ha ongaku wo kiku no ga daisuki desu  
he music to listening adores

# Syntactic Variation: Basic Word Orders

In many languages one word order is more basic

- SVO (Subject-Verb-Object) languages

English, German, French, Mandarin

I baked a pizza

- SOV Languages

Japanese, Hindi

English: He adores listening to music

Japanese: kare ha ongaku wo kiku no ga daisuki desu  
he music to listening adores

- VSO languages

- Irish, Classical Arabic, Tagalog

# Morphology

# Morphology

- Morpheme: “Minimal meaningful unit of language”

Word = Morpheme + Morpheme + Morpheme +...

# Morphology

- Morpheme: “Minimal meaningful unit of language”

Word = Morpheme + Morpheme + Morpheme +...

- Stems: (base form, root)

hope+ing → hop<sup>ing</sup>   hop → hopp<sup>ing</sup>

# Morphology

- Morpheme: “Minimal meaningful unit of language”

Word = Morpheme + Morpheme + Morpheme + ...

- Stems: (base form, root)

hope+ing → hop<sup>ing</sup>   hop → hopping

- Affixes

- **Prefixes:** Antidisestablishmentarianism

- **Suffixes:** Antidisestablishmentarianism

- **Infixes:** hingi (borrow) – hum<sup>ingi</sup> (borrower) in Tagalog

- **Circumfixes:** sagen (say) – gesagt (said) in German

# Morphemes per Word

Joseph Greenberg. 1954. A Quantitative Approach to the Morphological Typology of Language. IJAL 26:3.

# Morphemes per Word

Joseph Greenberg. 1954. A Quantitative Approach to the Morphological Typology of Language. IJAL 26:3.

isolating

1



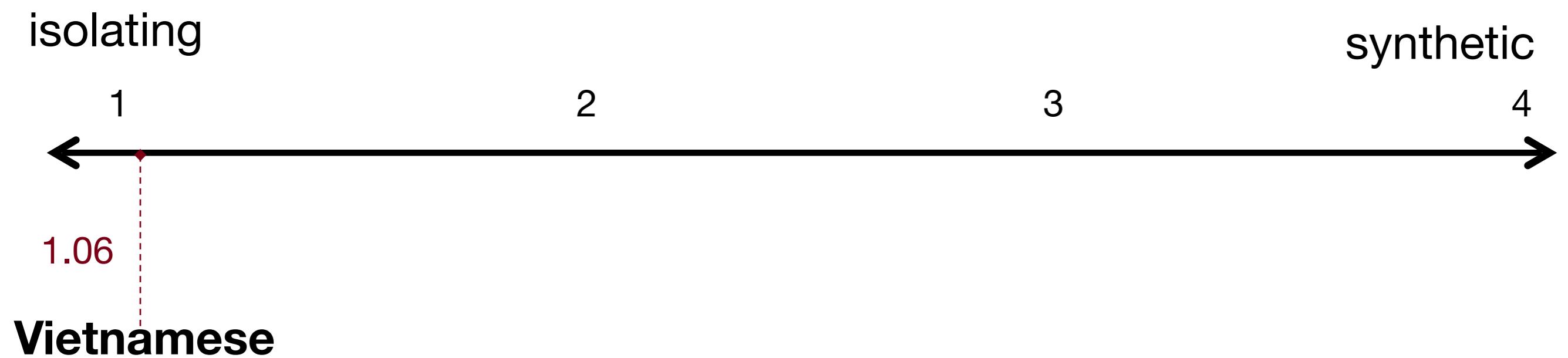
# Morphemes per Word

Joseph Greenberg. 1954. A Quantitative Approach to the Morphological Typology of Language. IJAL 26:3.



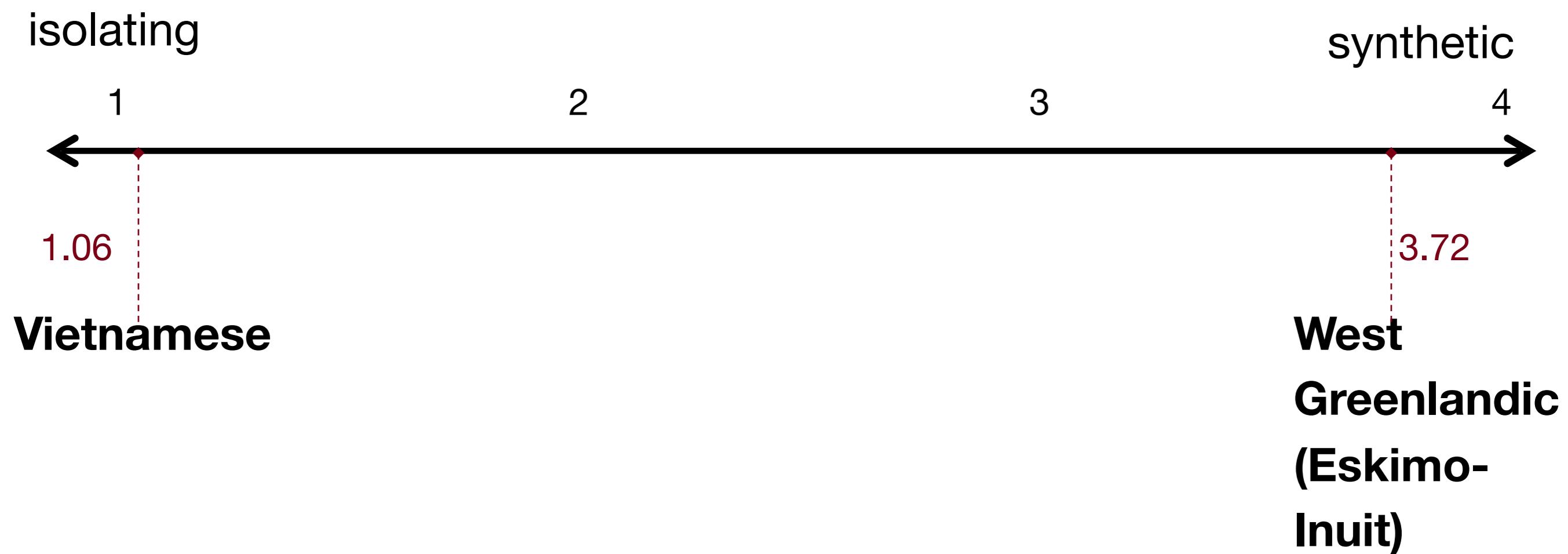
# Morphemes per Word

Joseph Greenberg. 1954. A Quantitative Approach to the Morphological Typology of Language. IJAL 26:3.



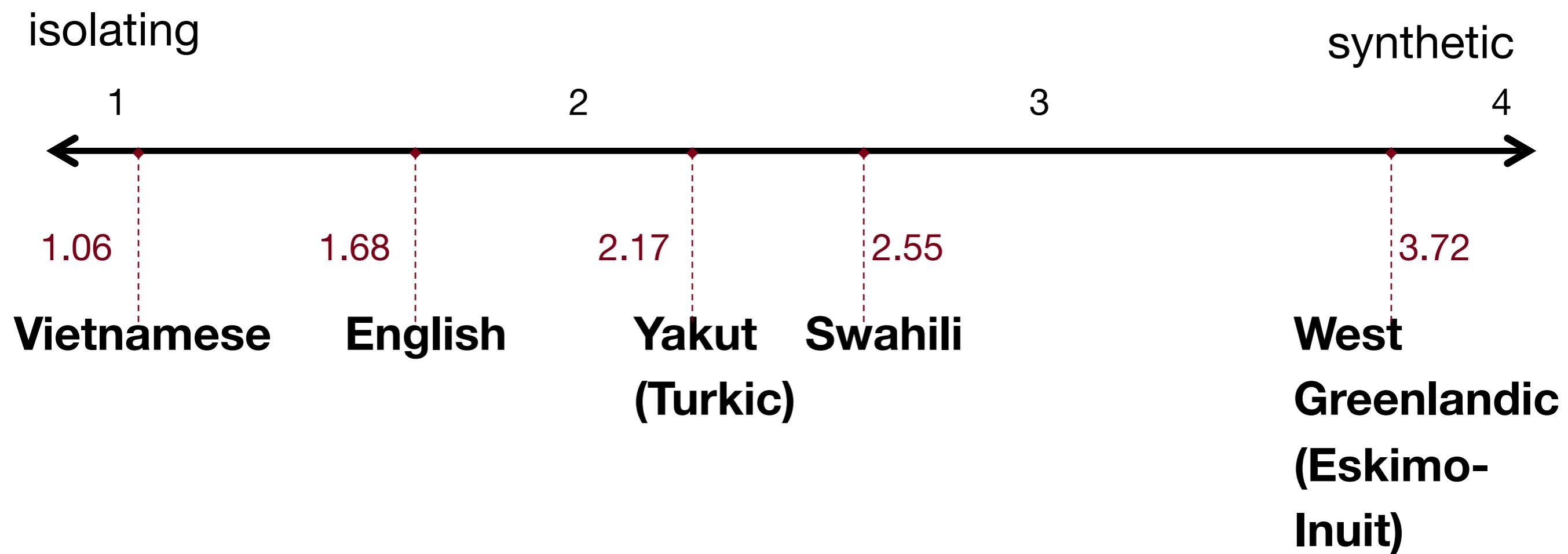
# Morphemes per Word

Joseph Greenberg. 1954. A Quantitative Approach to the Morphological Typology of Language. IJAL 26:3.



# Morphemes per Word

Joseph Greenberg. 1954. A Quantitative Approach to the Morphological Typology of Language. IJAL 26:3.



# Few morphemes per word: Cantonese

“He said this was the biggest building in the whole country”

Each word in this sentence has one morpheme (and one syllable):

keui wa chyuhn gwok jeui daaih gaan nguk haih li gaan  
he say entire country most big bldg house is this bldg

# Many Morphemes per word: Turkish

uygarlaştıramadıklarımızdanmışsınızcasına  
uygar+laş+tır+ama+dık+lar+ımız+dan+mış+sınız+  
casına

*Behaving as if you are among those whom we could  
not cause to become civilized*

# Word Segmentation

## Are word boundaries marked in writing?

- Some writing systems: boundaries between words not marked
  - Chinese, Japanese, Thai
  - Word segmentation becomes an important part of text normalization for MT
- Some languages tend to have sentences that are quite long, closer to English paragraphs than sentences:
  - Modern Standard Arabic, Chinese
  - Sentence segmentation may be necessary for MT between these languages and languages like English

# Inferential Load: cold vs. hot languages

Balthasar Bickel. 2003. Referential density in discourse and syntactic typology. Language 79:2, 708-36

- **Hot languages:**
  - Who did what to whom is marked explicitly
  - English
- **Cold languages:**
  - The hearer has more “figuring out” of who the various actors in the various events are
  - Japanese, Chinese

# Inferential Load: The blue noun phrases are not in the Chinese original

飓风丽塔已经减弱为第三级飓风，

Rita weakened and was downgraded to a Category 3 storm;

❶ 迫近美国德课萨斯州和路易斯安那州，

[Rita/it/the storm] is moving close to Texas and Louisiana;

当局表示，

the authorities announced;

虽然 ❶ 在登陆前可能再稍微减弱，

although [Rita/it/the storm] might weaken again before landing,

但 ❶ 仍然会非常危险，

[Rita/it/the storm] is still very dangerous;

❶ 预料 ❶ 会在当地时间星期六凌晨在德州和路易斯安那州之间登陆，

[the authorities] predict [Rita/it/the storm] will arrive at the Texas-Louisiana border on Saturday morning local time;

❶ 直接吹袭休斯敦市东面的主要炼油设施。

[Rita/it/the storm] will directly hit the oil-refining industry east of Houston.

# Lexical Divergences

- Word to phrases:
  - English computer science
  - French informatique
- Part of Speech divergences
  - English She likes to sing
  - German Sie singt gerne [She sings likefully]
  - English I'm hungry
  - Spanish Tengo hambre [I have hunger]

# Lexical Specificity Divergences

- Grammatical specificity
  - Spanish: plural pronouns have gender (**ellos/ellas**)
  - English: plural pronouns no gender (**they**)
- So translating “**they**” from English to Spanish, need to figure out gender of the referent!

# Lexical Divergences: Semantic Specificity

English      **brother**

Mandarin **gege** (older brother), **didi** (younger brother)

English      **wall**

German **Wand** (inside)      **Mauer** (outside)

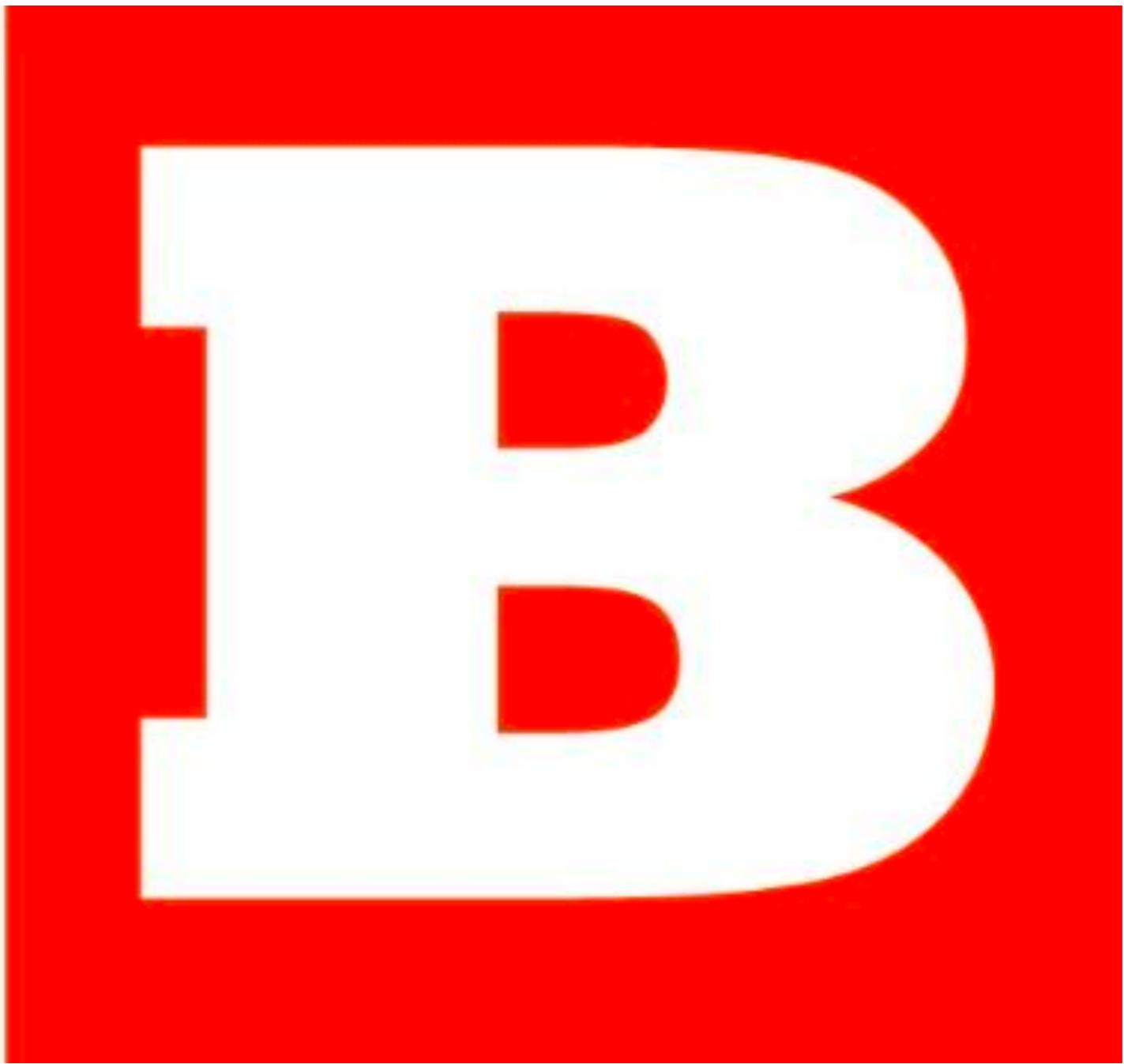
English      **fish**

Spanish **pez** (the creature)      **pescado** (fish as food)

Cantonese      **ngau**

English      **cow**      **beef**

**BREITBART**



# Statistical Machine Translation

# Statistical MT

- The intuition for Statistical MT comes from the **impossibility** of perfect translation
- Why perfect translation is impossible
  - Goal: Translating Hebrew *adonai roi* (“the lord is my shepherd”) for a culture without sheep or shepherds

# Statistical MT

- The intuition for Statistical MT comes from the **impossibility** of perfect translation
- Why perfect translation is impossible
  - Goal: Translating Hebrew *adonai roi* (“the lord is my shepherd”) for a culture without sheep or shepherds
- Two options:
  - Something **fluent** and understandable, but not faithful:  
The Lord will look after me

# Statistical MT

- The intuition for Statistical MT comes from the **impossibility** of perfect translation
- Why perfect translation is impossible
  - Goal: Translating Hebrew *adonai roi* (“the lord is my shepherd”) for a culture without sheep or shepherds
- Two options:
  - Something **fluent** and understandable, but not faithful:  
The Lord will look after me
  - Something **faithful**, but not fluent or natural  
The Lord is for me like somebody who looks after animals with cotton-like hair

# A good translation is:

- Faithful
  - Has the same meaning as the source
  - (Causes the reader to draw the same inferences as the source would have)

# A good translation is:

- Faithful
  - Has the same meaning as the source
  - (Causes the reader to draw the same inferences as the source would have)
- Fluent
  - Is natural, fluent, grammatical in the target

# A good translation is:

- Faithful
  - Has the same meaning as the source
  - (Causes the reader to draw the same inferences as the source would have)
- Fluent
  - Is natural, fluent, grammatical in the target
- Real translations trade off these two factors



“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, March 1947

The required statistical tables have millions of entries...?

Too much for the computers of Weaver's day.

-> Not enough RAM!

# IBM Candide Project (1988-1994)

- How to get quantities of human translation in computer readable form?
  - *parallel corpus*

IBM's John Cocke,  
inventor of  
CKY parsing &  
RISC processors



Canadian  
bureaucrat

# IBM Candide Project (1988-1994)

- How to get quantities of human translation in computer readable form?
  - *parallel corpus*

IBM's John Cocke,  
inventor of  
CKY parsing &  
RISC processors

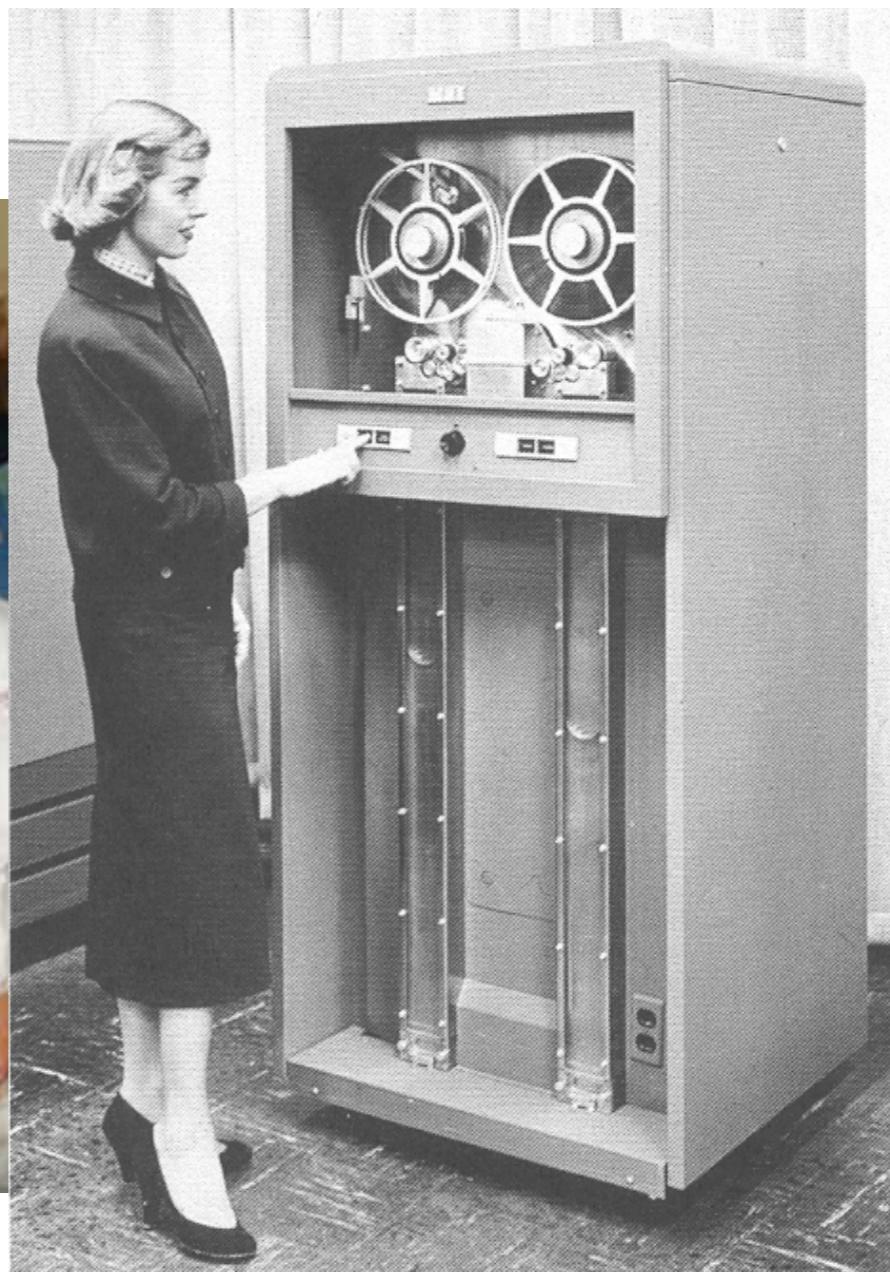


Canadian  
bureaucrat

# IBM Candide Project (1988-1994)

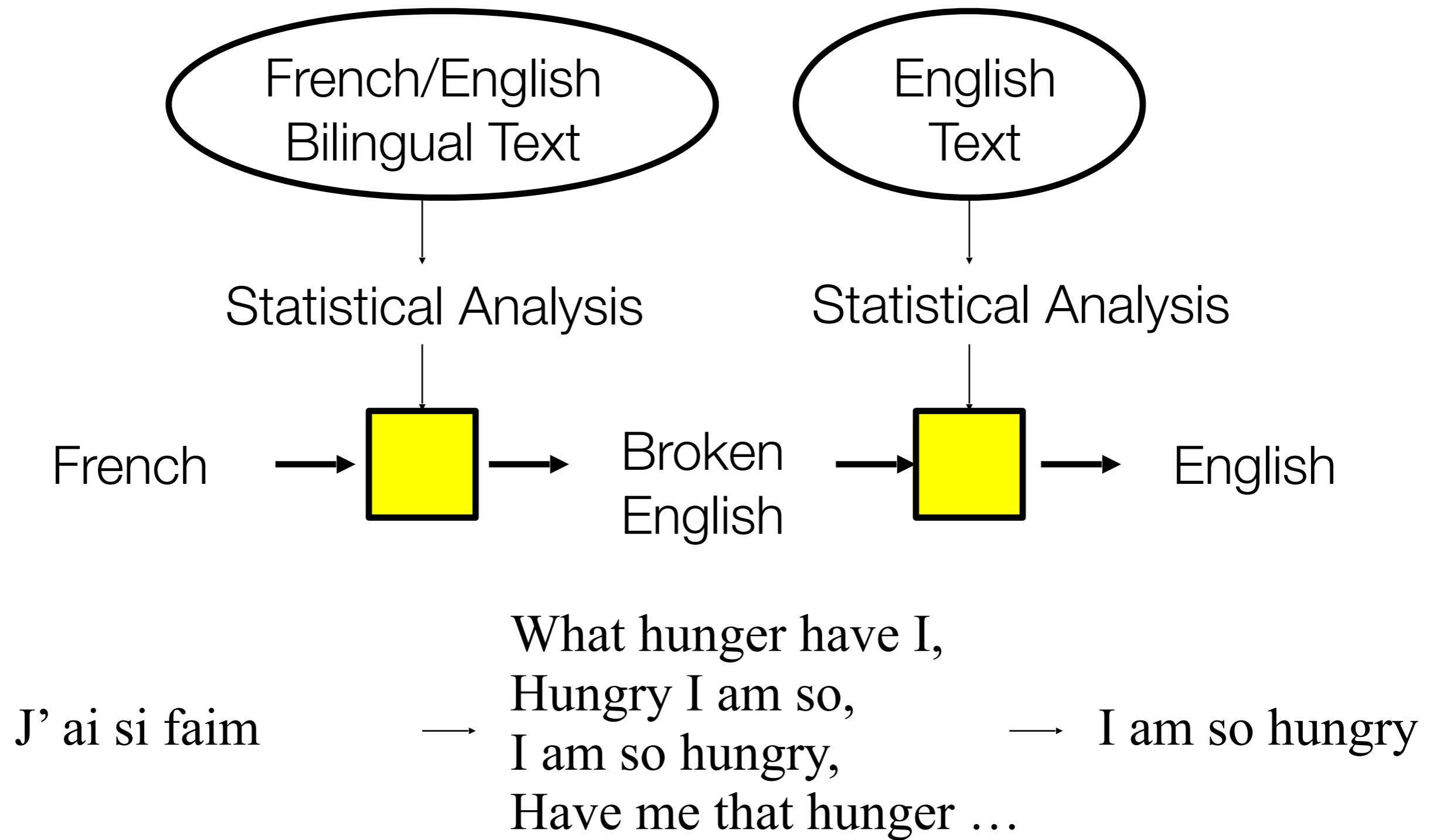
- How to get quantities of human translation in computer readable form?
  - *parallel corpus*

IBM's John Cocke,  
inventor of  
CKY parsing &  
RISC processors



# IBM Candide Project

[Brown et al 93]



# Mathematical Formulation

Given source sentence  $f$ :

# Mathematical Formulation

Given source sentence  $f$ :

$$\operatorname{argmax}_e P(e | f) =$$

# Mathematical Formulation

Given source sentence  $f$ :

$$\operatorname{argmax}_e P(e | f) =$$

# Mathematical Formulation

Given source sentence  $f$ :

$$\operatorname{argmax}_e P(e | f) =$$

$$\operatorname{argmax}_e P(f | e) \cdot P(e) / P(f) = \text{by Bayes Rule}$$

# Mathematical Formulation

Given source sentence  $f$ :

$$\operatorname{argmax}_e P(e | f) =$$

$$\operatorname{argmax}_e P(f | e) \cdot P(e) / P(f) = \text{by Bayes Rule}$$

# Mathematical Formulation

Given source sentence  $f$ :

$$\operatorname{argmax}_e P(e | f) =$$

$$\operatorname{argmax}_e P(f | e) \cdot P(e) / P(f) = \text{by Bayes Rule}$$

$$\operatorname{argmax}_e P(f | e) \cdot P(e) \quad P(f) \text{ same for all } e$$

# Mathematical Formulation

Given source sentence  $f$ :

$$\operatorname{argmax}_e P(e | f) =$$

$$\operatorname{argmax}_e P(f | e) \cdot P(e) / P(f) = \text{by Bayes Rule}$$

$$\operatorname{argmax}_e P(f | e) \cdot P(e) \quad P(f) \text{ same for all } e$$



# Mathematical Formulation

Given source sentence  $f$ :

$$\operatorname{argmax}_e P(e | f) =$$

$$\operatorname{argmax}_e P(f | e) \cdot P(e) / P(f) = \text{by Bayes Rule}$$

$$\operatorname{argmax}_e P(f | e) \cdot P(e) \quad P(f) \text{ same for all } e$$



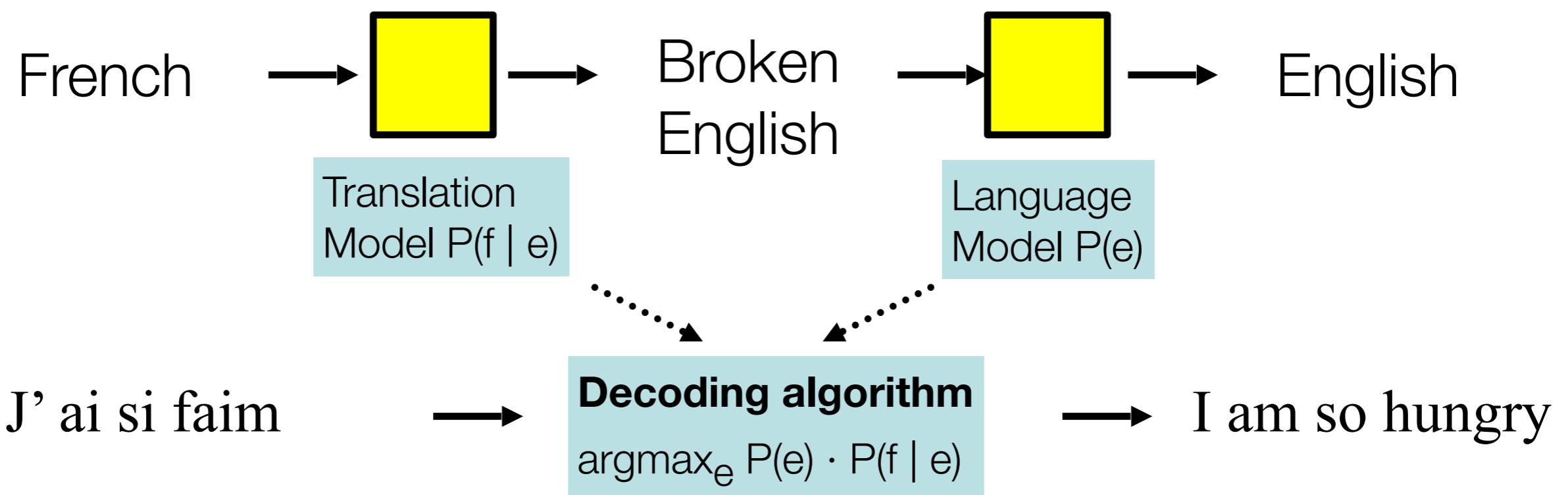
# Mathematical Formulation

Given source sentence  $f$ :

$$\operatorname{argmax}_e P(e | f) =$$

$$\operatorname{argmax}_e P(f | e) \cdot P(e) / P(f) = \text{by Bayes Rule}$$

$$\operatorname{argmax}_e P(f | e) \cdot P(e) \quad P(f) \text{ same for all } e$$



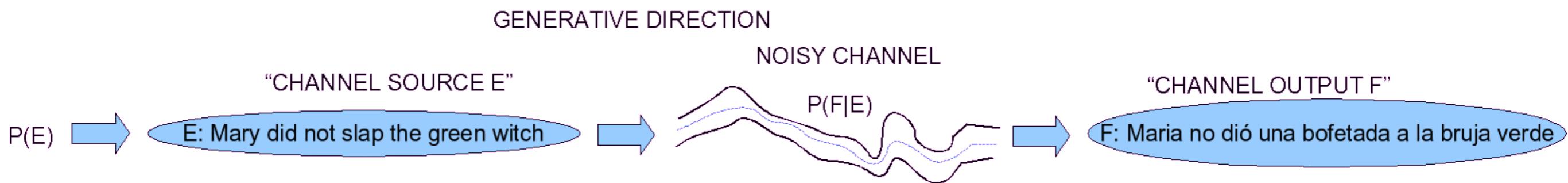
# Convention in Statistical MT

- We always refer to translating
  - from input F, the foreign language (originally F = French)
  - to output E, English.

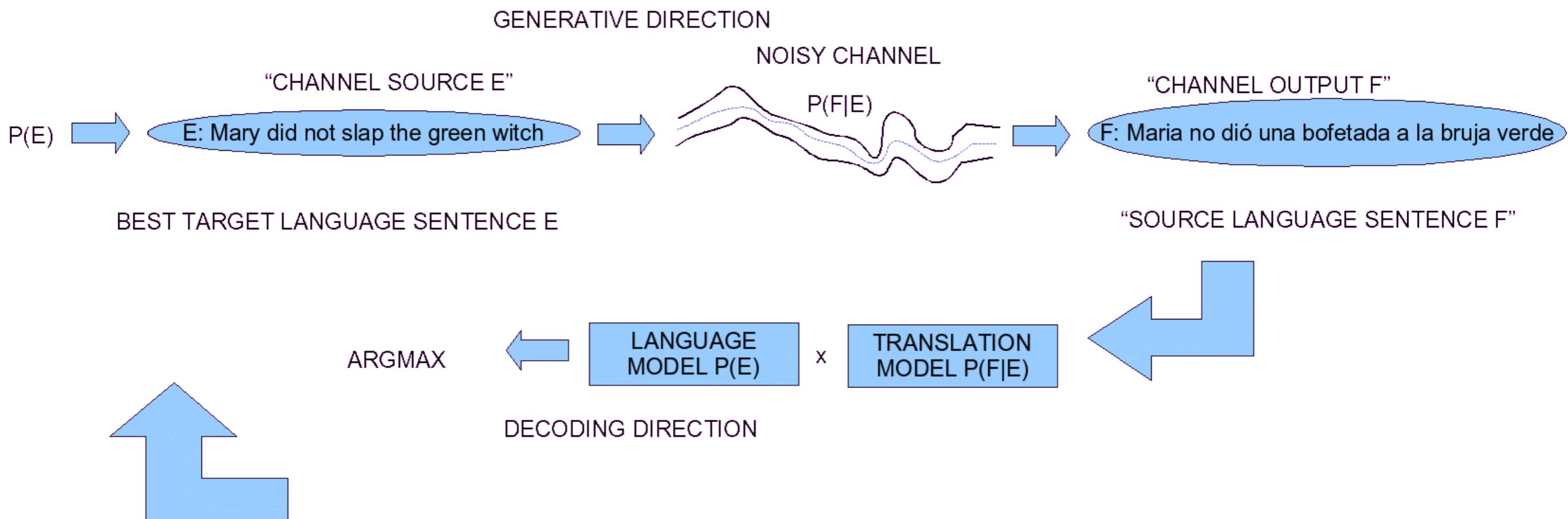
# Convention in Statistical MT

- We always refer to translating
  - from input F, the foreign language (originally F = French)
  - to output E, English.
- Obviously statistical MT can translate from English into another language or between any pair of languages
- The convention helps avoid confusion about which way the probabilities are conditioned for a given example
- I will call the input F, or sometimes French, or Spanish.

# The noisy channel model for MT



# The noisy channel model for MT



# Fluency: $P(E)$

- We need a metric that ranks this sentence

**That car almost crash to me**

as less fluent than this one:

**That car almost hit me.**

# Fluency: $P(E)$

- We need a metric that ranks this sentence

`That car almost crash to me`

as less fluent than this one:

`That car almost hit me.`

- Answer: language models (N-grams!)

$P(me|hit) > P(to|crash)$

- And we can use any other more sophisticated model of grammar
- Advantage: this is **monolingual** knowledge!

# Faithfulness: $P(F|E)$

- Spanish:
  - Maria no dió una bofetada a la bruja verde
- English candidate translations:
  - Mary didn't slap the green witch
  - Mary not give a slap to the witch green
  - The green witch didn't slap Mary
  - Mary slapped the green witch

# Faithfulness: $P(F|E)$

- Spanish:
  - Maria no dió una bofetada a la bruja verde
- English candidate translations:
  - Mary didn't slap the green witch
  - Mary not give a slap to the witch green
  - The green witch didn't slap Mary
  - Mary slapped the green witch
- More faithful translations will be composed of phrases that are high probability translations
  - How often was “slapped” translated as “dió una bofetada” in a large bitext (parallel English-Spanish corpus)
  - We’ll need to align phrases and words to each other in bitext

# We treat Faithfulness and Fluency as independent factors

- $P(F|E)$ 's job is to model “bag of words”; which words come from English to Spanish.
  - $P(F|E)$  doesn't have to worry about internal facts about English word order.

# We treat Faithfulness and Fluency as independent factors

- $P(F|E)$ 's job is to model “bag of words”; which words come from English to Spanish.
  - $P(F|E)$  doesn't have to worry about internal facts about English word order.
- $P(E)$ 's job is to do bag generation: put the following words in order:
  - a ground there in the hobbit hole lived a in

# Three Problems for Statistical MT

- Language Model: given  $E$ , compute  $P(E)$   
good English string  $\rightarrow$  high  $P(E)$   
random word sequence  $\rightarrow$  low  $P(E)$
- Translation Model: given  $(F, E)$  compute  $P(F | E)$   
 $(F, E)$  look like translations  $\rightarrow$  high  $P(F | E)$   
 $(F, E)$  don't look like translations  $\rightarrow$  low  $P(F | E)$
- Decoding algorithm: given LM, TM, F, find  $\hat{E}$   
Find translation  $E$  that maximizes  $P(E) * P(F | E)$

# Language Model

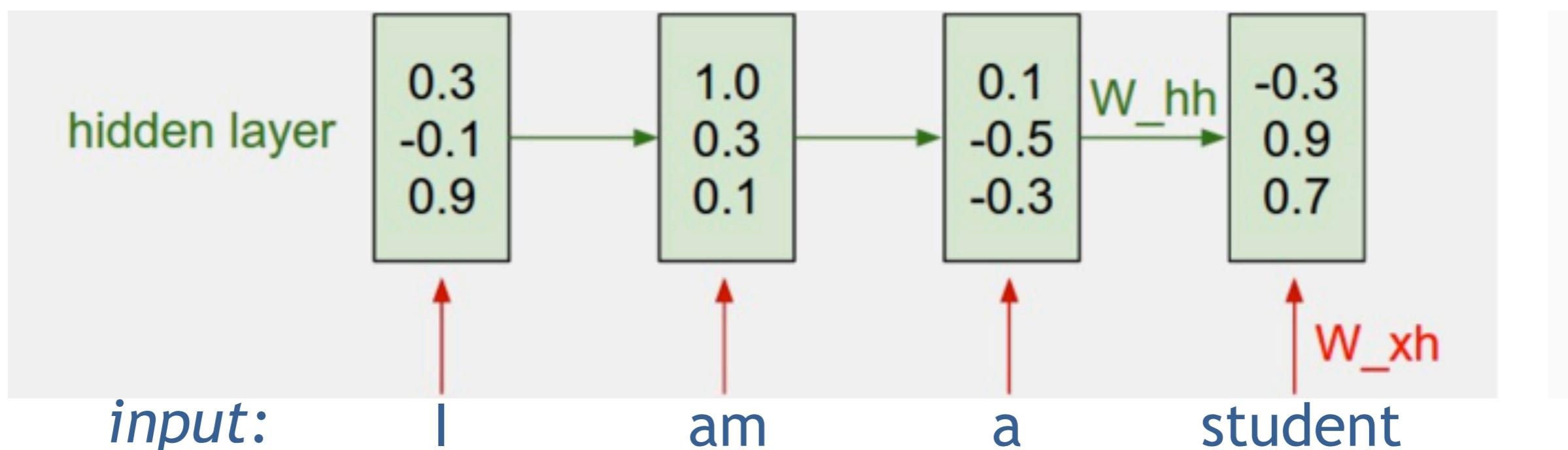
- Use a standard  $n$ -gram language model for  $P(E)$ .
- Can be trained on a large mono-lingual corpus
  - 5-gram grammar of English from terabytes of web data
  - More sophisticated parser-based language models can also help
- You learned about these in Week 3!





# Neural Machine Translation (NMT)

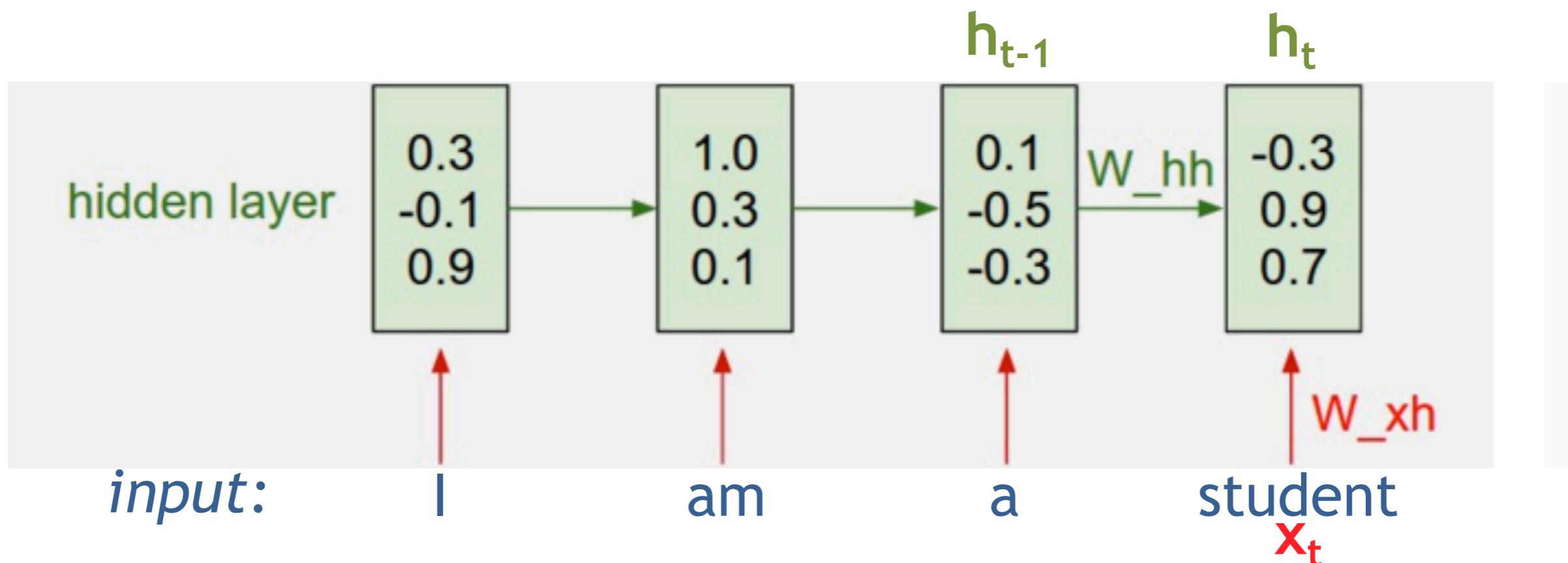
# Recurrent Neural Networks (RNNs)



(Picture adapted from Andrej Karpathy)

# Recurrent Neural Networks (RNNs)

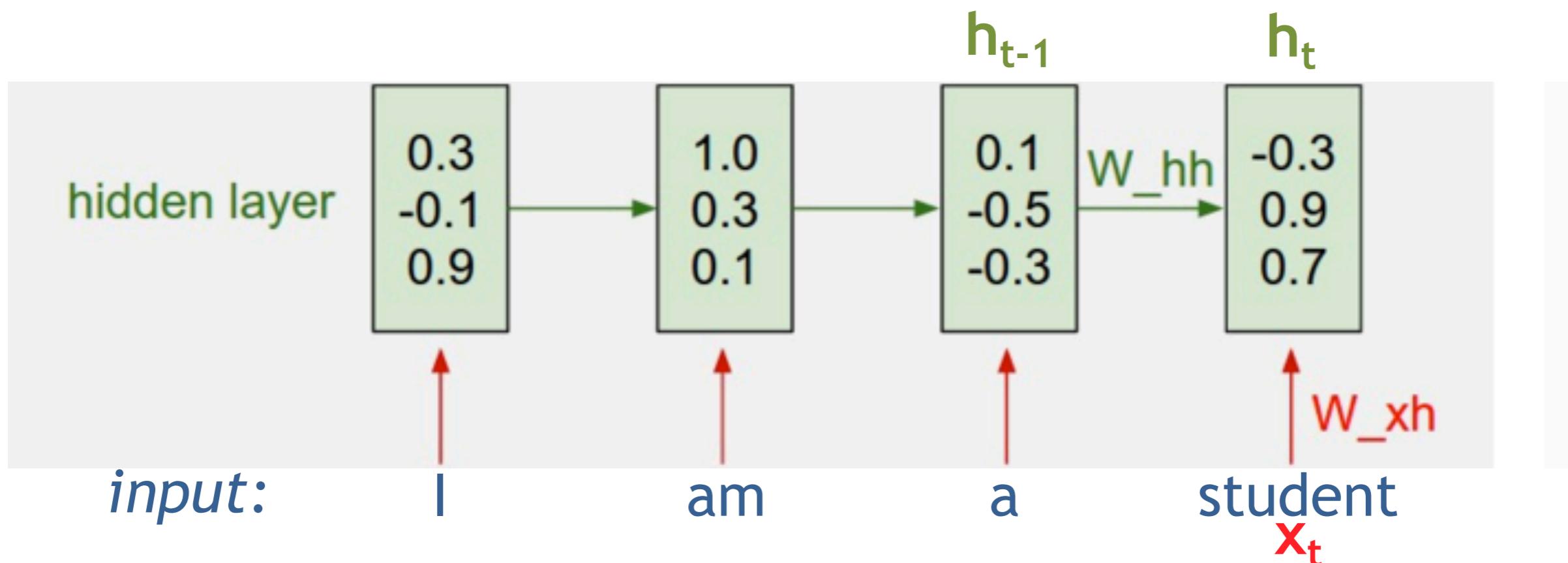
$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1})$$



(Picture adapted from Andrej Karpathy)

# Recurrent Neural Networks (RNNs)

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1})$$



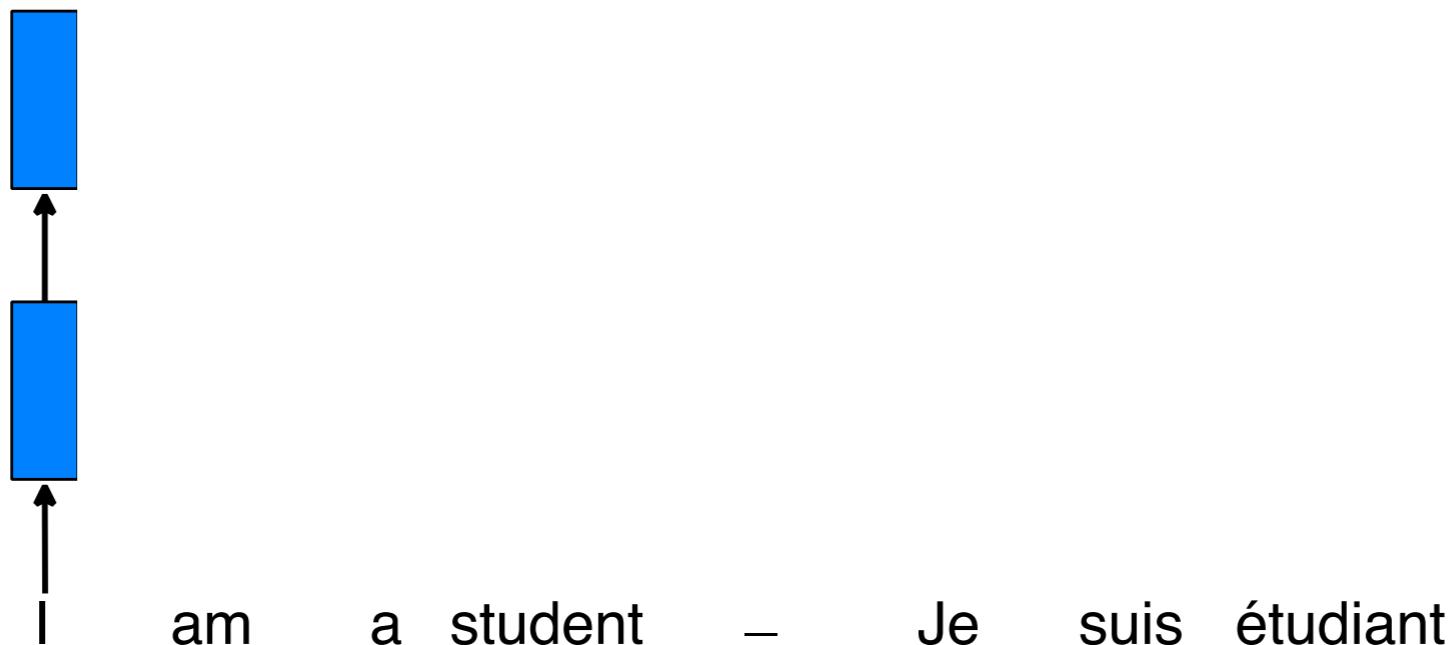
RNNs to represent sequences!

# Neural Machine Translation (NMT)

I am a student – Je suis étudiant

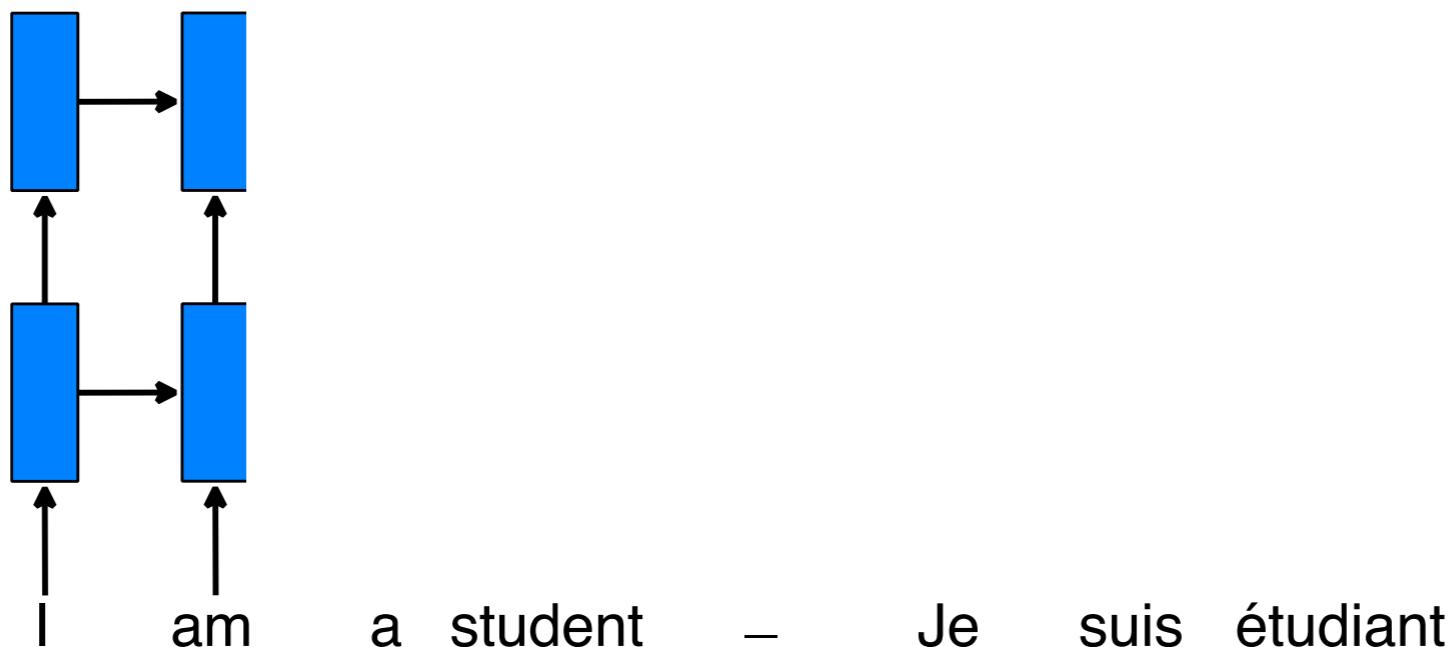
- Model  $P(\text{target} \mid \text{source})$  directly.

# Neural Machine Translation (NMT)



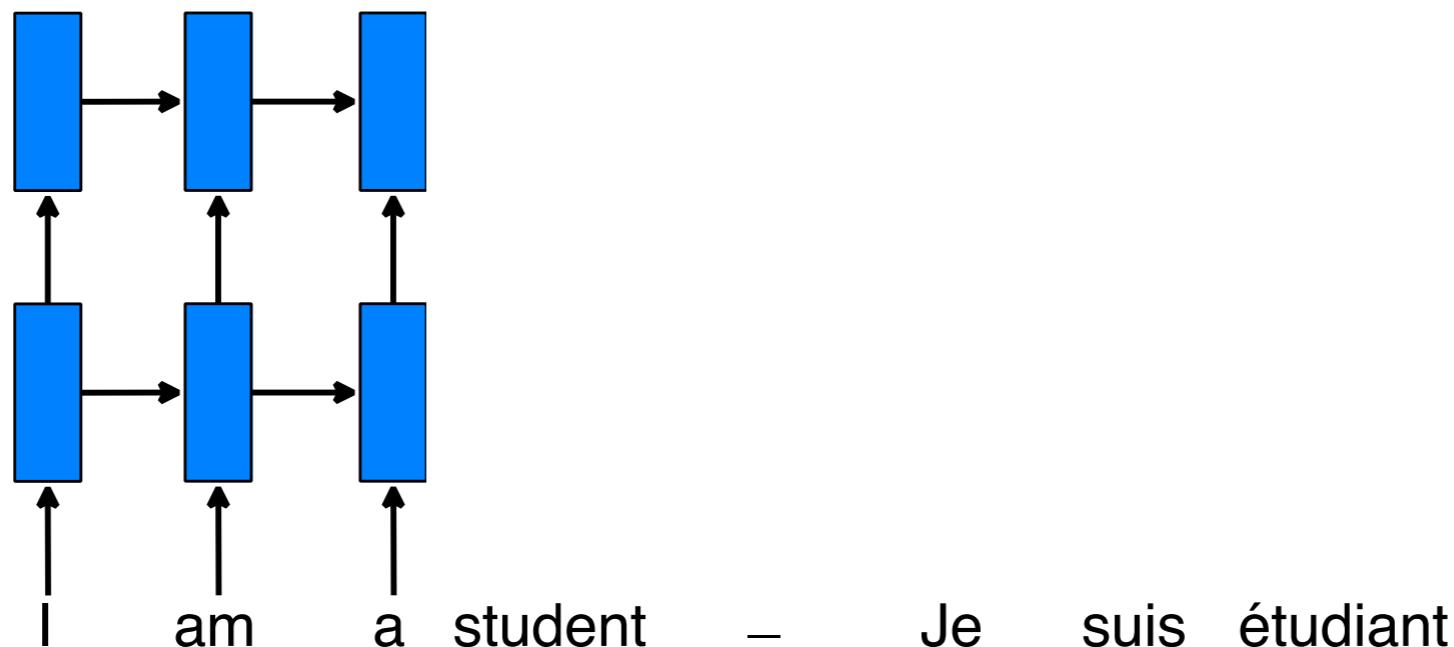
- RNNs trained **end-to-end** (Sutskever et al., 2014).

# Neural Machine Translation (NMT)



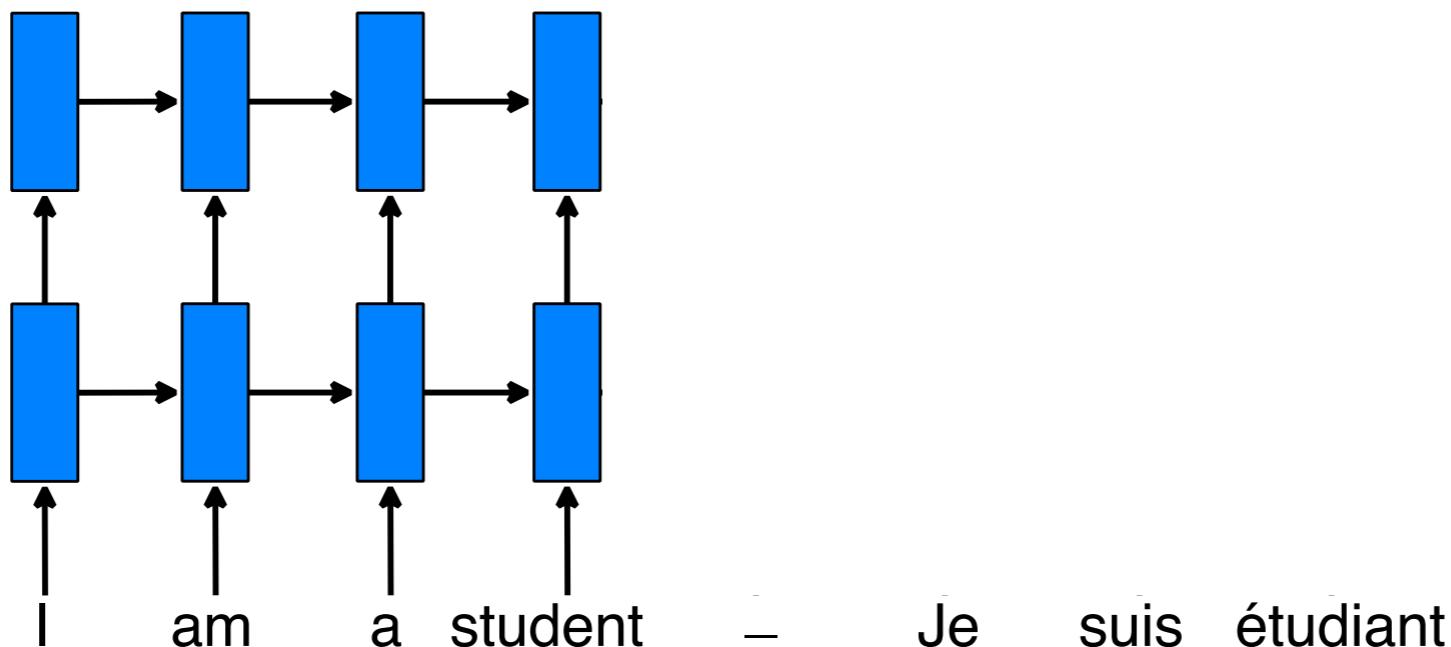
- RNNs trained end-to-end (Sutskever et al., 2014).

# Neural Machine Translation (NMT)



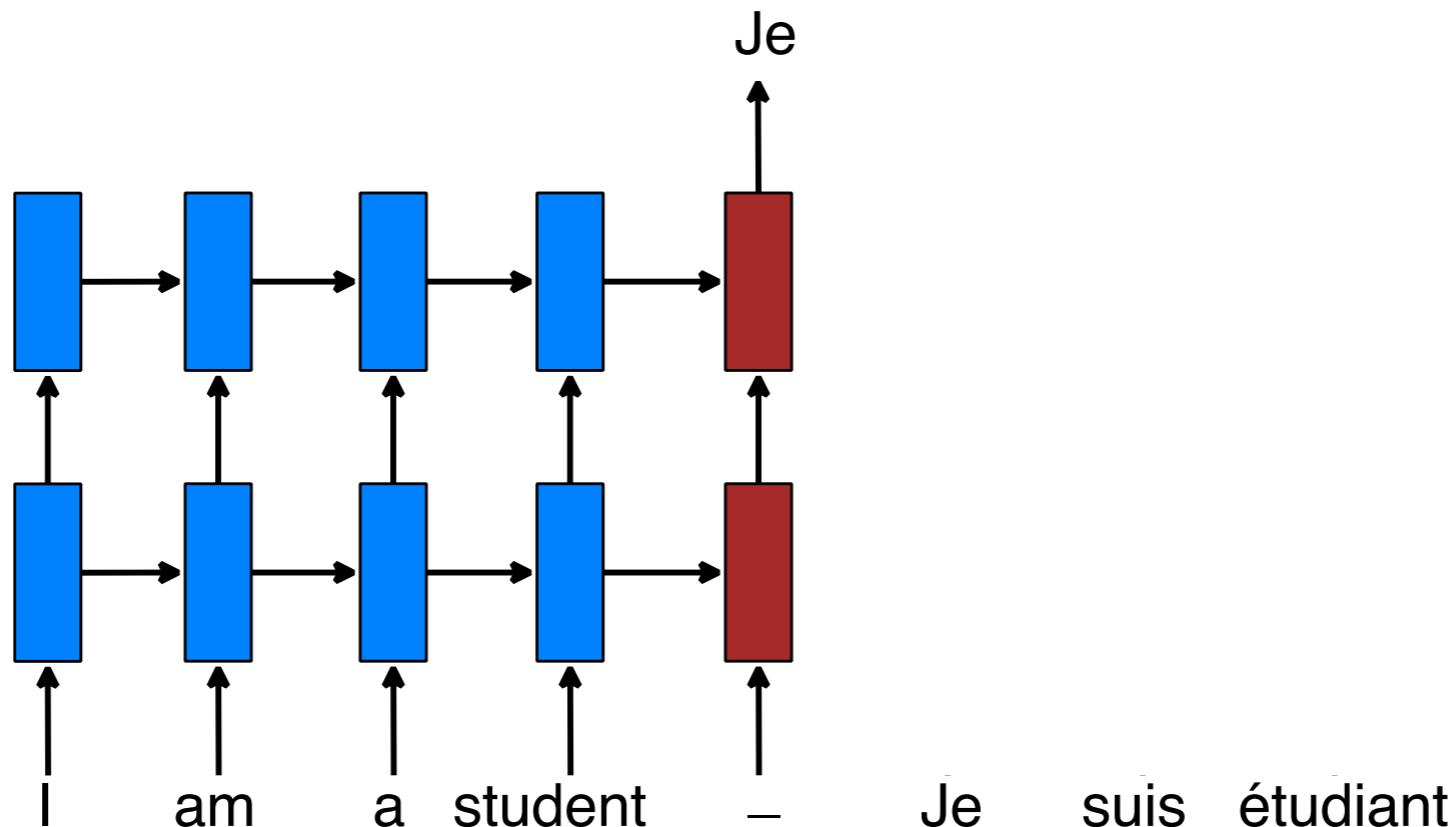
- RNNs trained end-to-end (Sutskever et al., 2014).

# Neural Machine Translation (NMT)



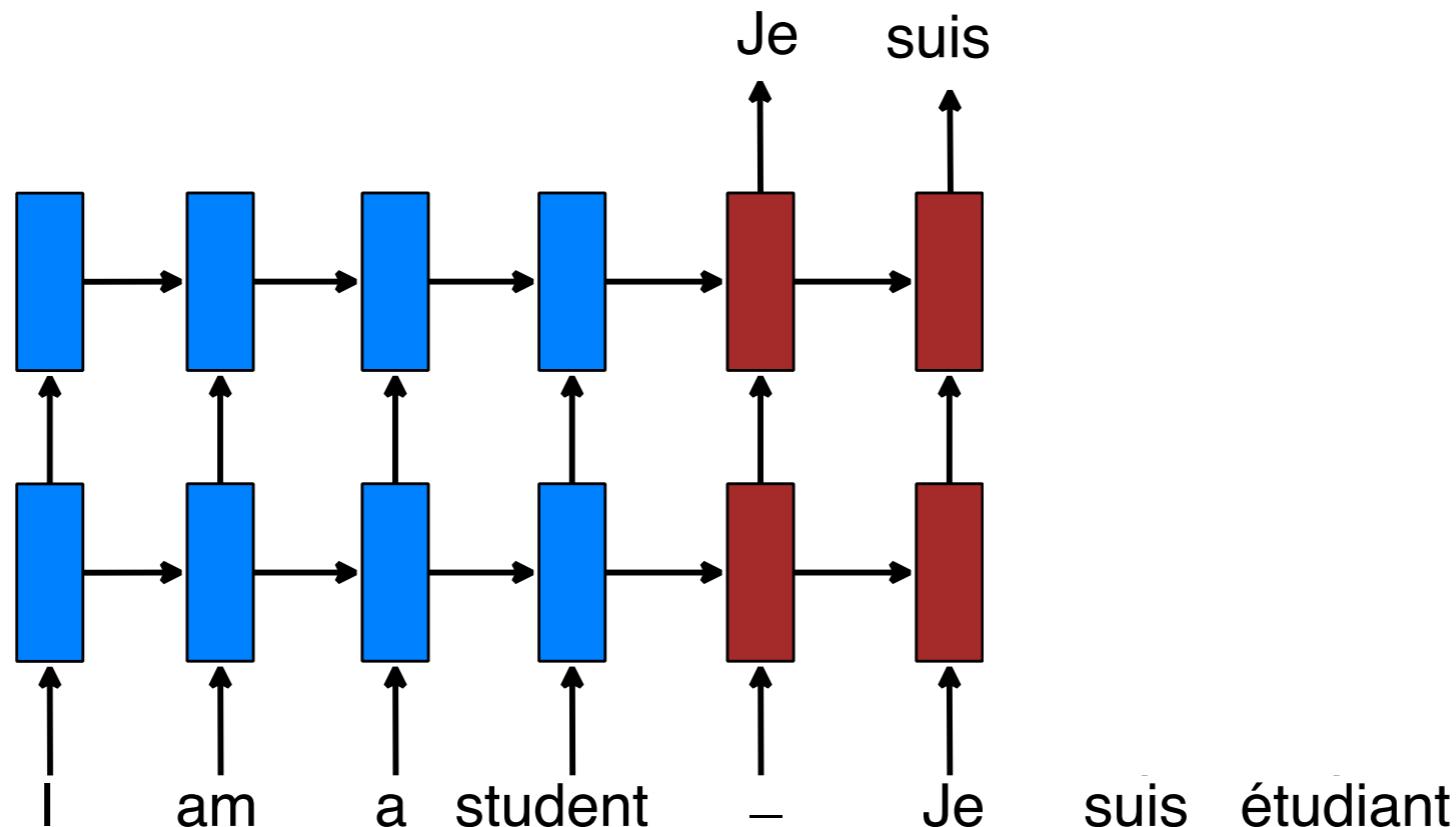
- RNNs trained end-to-end (Sutskever et al., 2014).

# Neural Machine Translation (NMT)



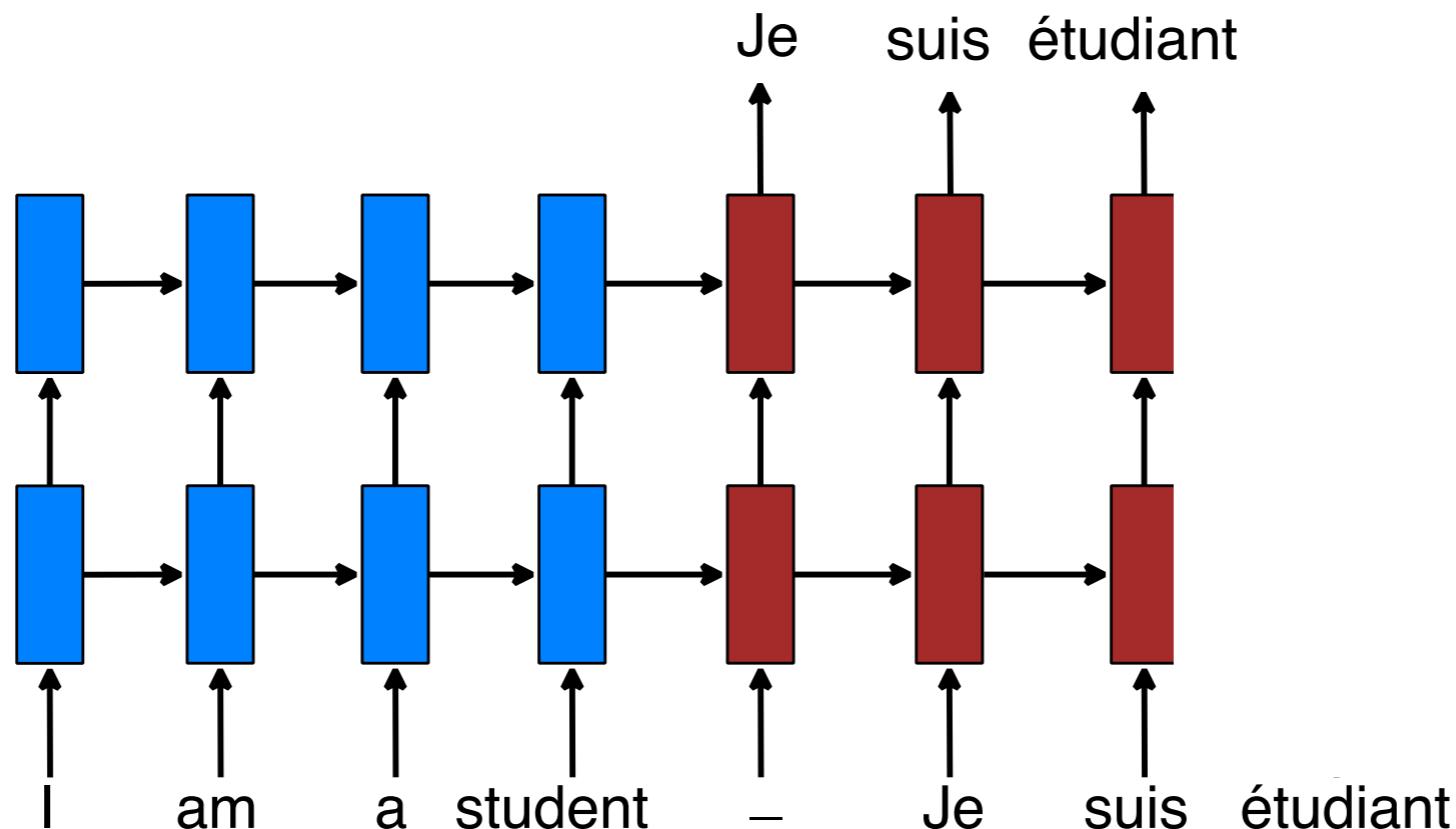
- RNNs trained end-to-end (Sutskever et al., 2014).

# Neural Machine Translation (NMT)



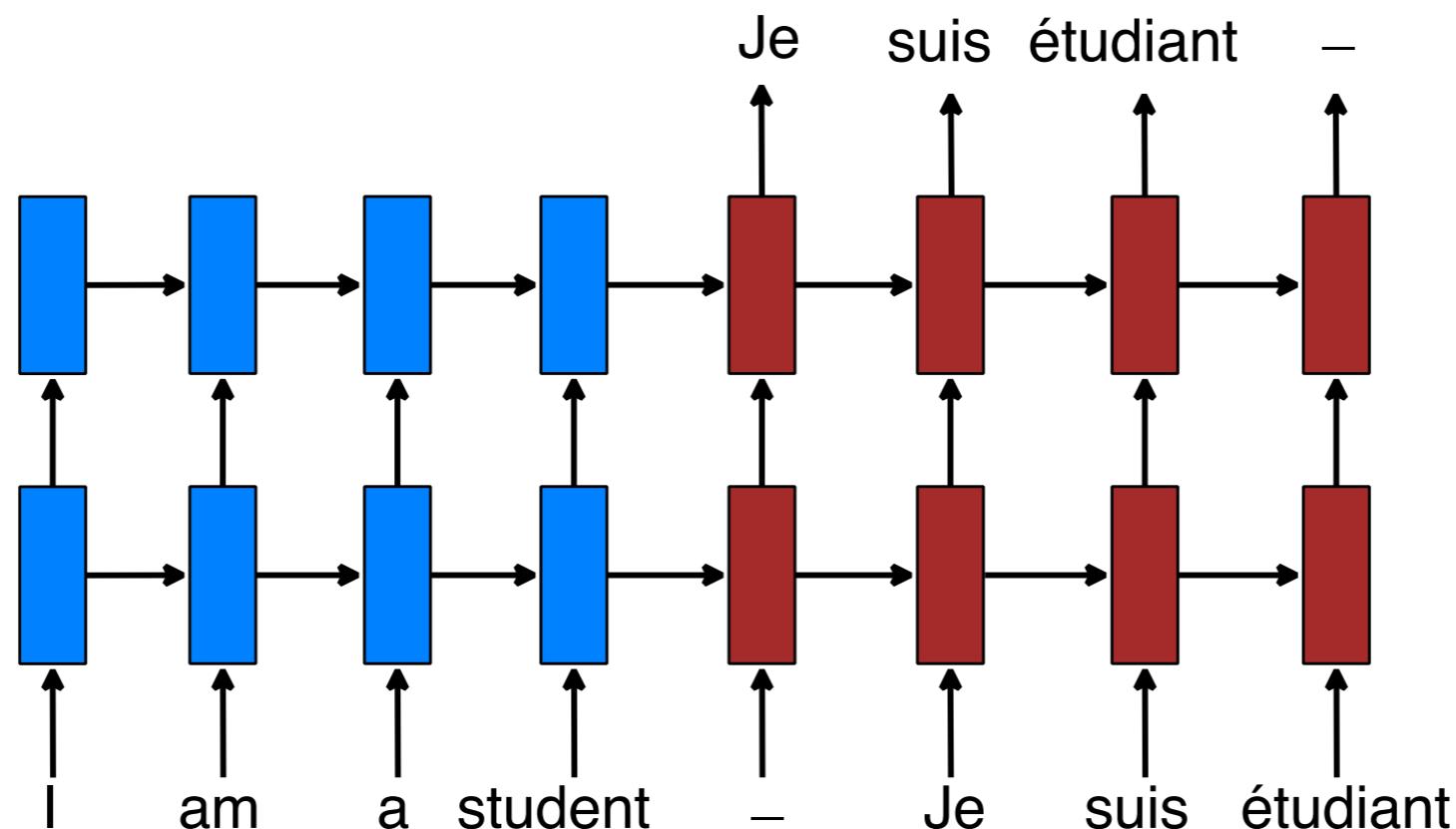
- RNNs trained end-to-end (Sutskever et al., 2014).

# Neural Machine Translation (NMT)

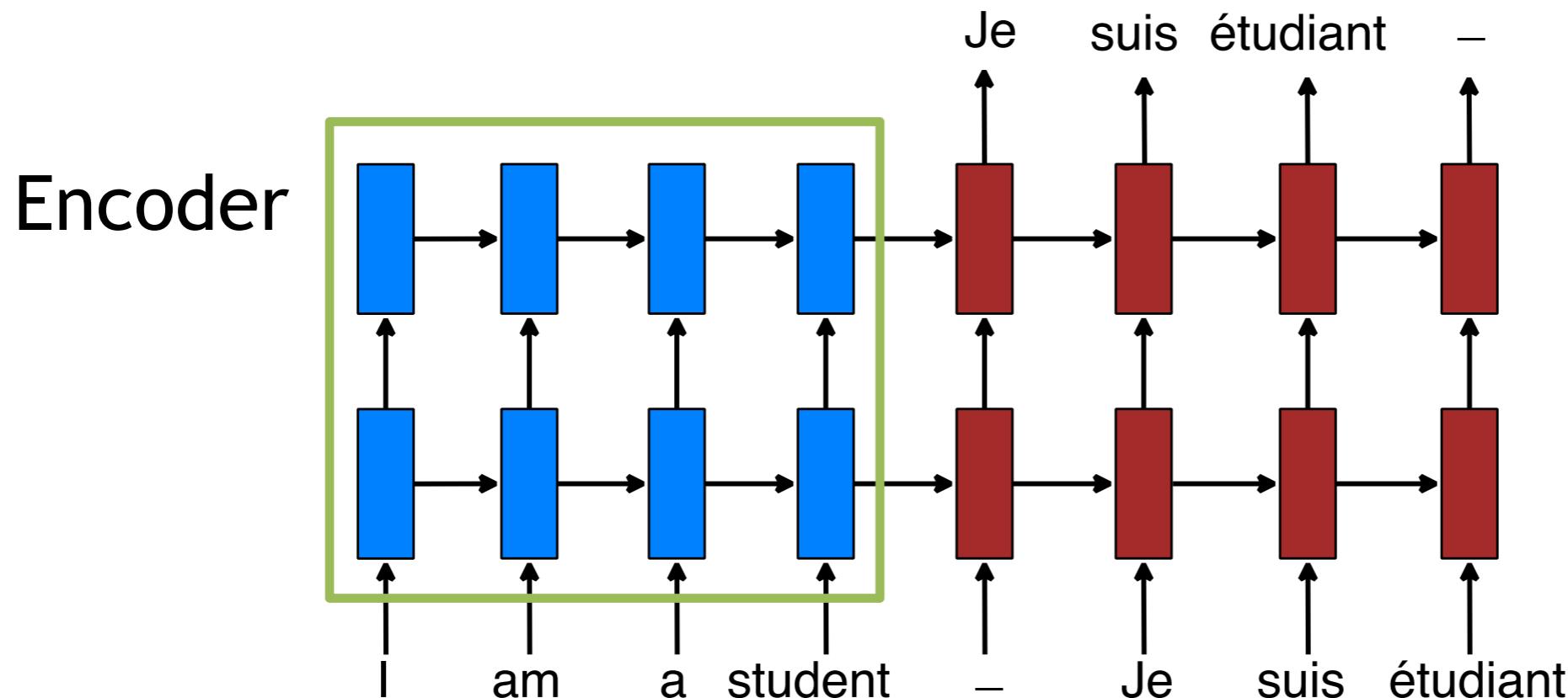


- RNNs trained end-to-end (Sutskever et al., 2014).

# Neural Machine Translation (NMT)

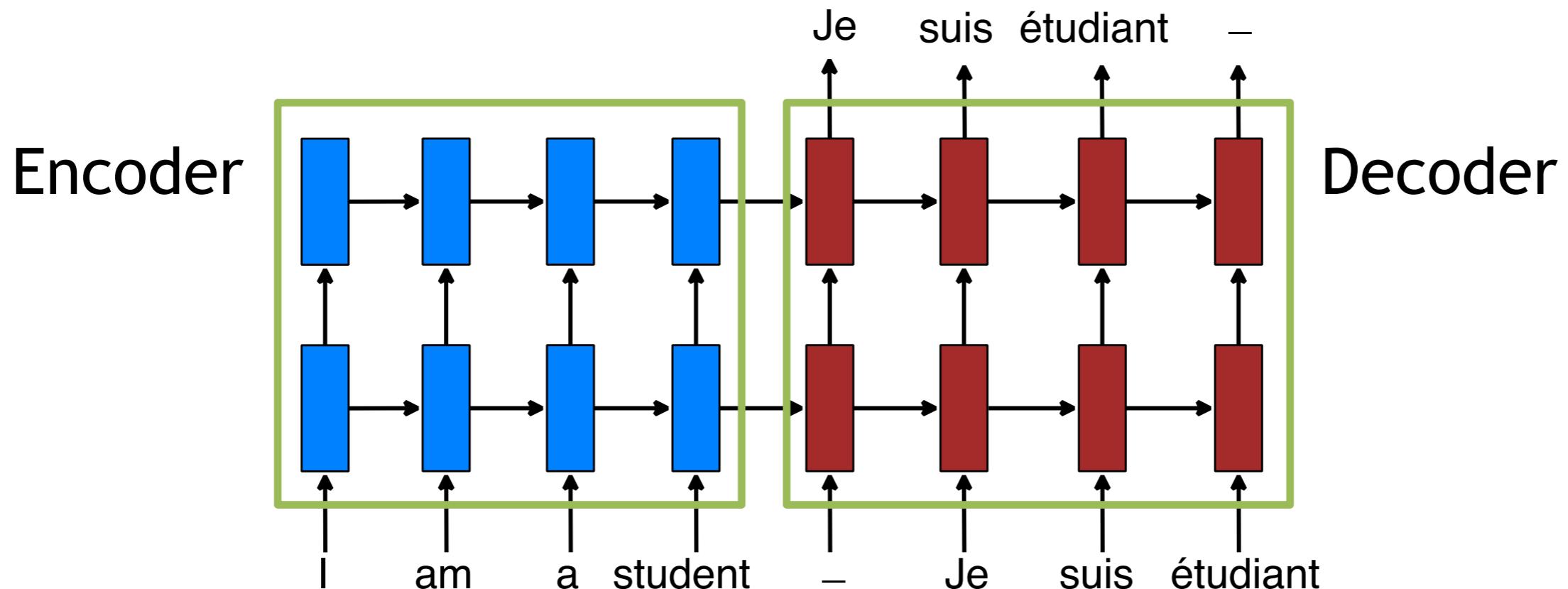


# Neural Machine Translation (NMT)



- RNNs trained **end-to-end** (Sutskever et al., 2014).
- Encoder-decoder approach.

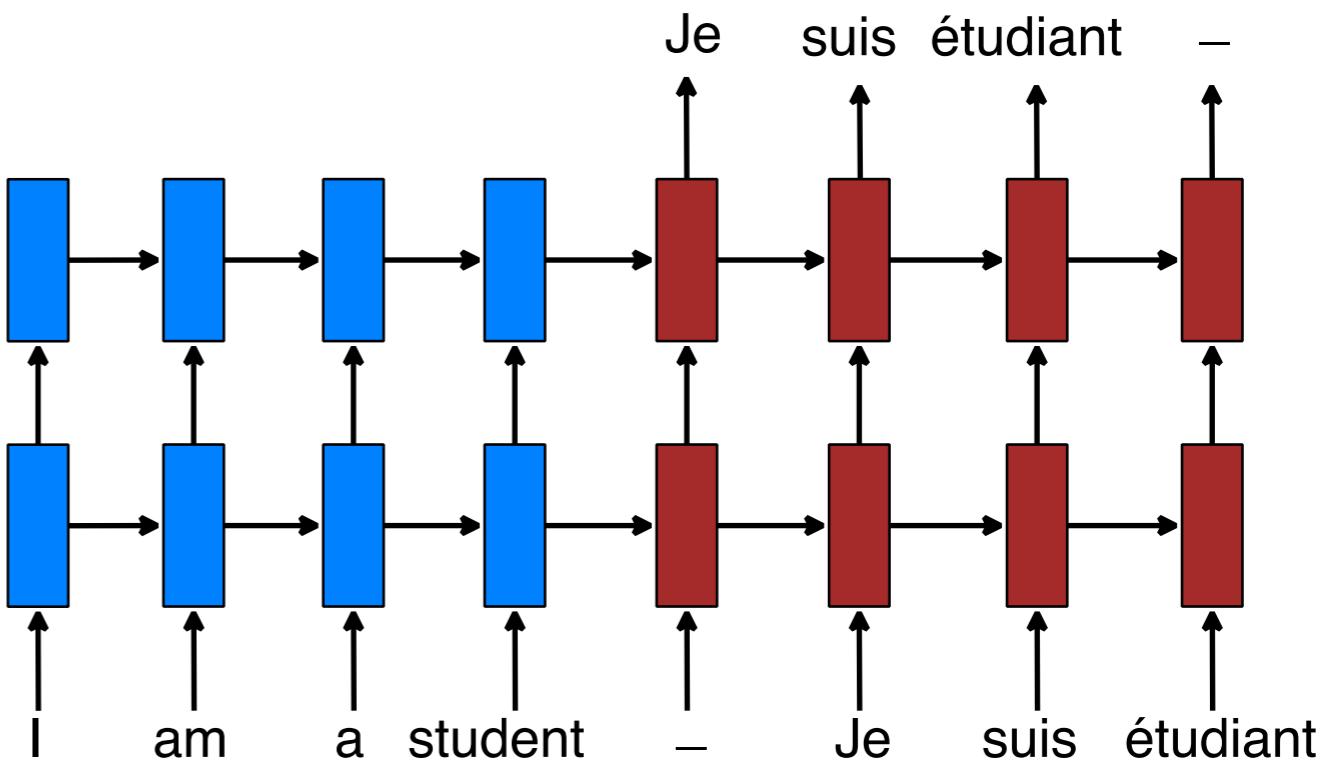
# Neural Machine Translation (NMT)



- RNNs trained **end-to-end** (Sutskever et al., 2014).
- Encoder-decoder approach.

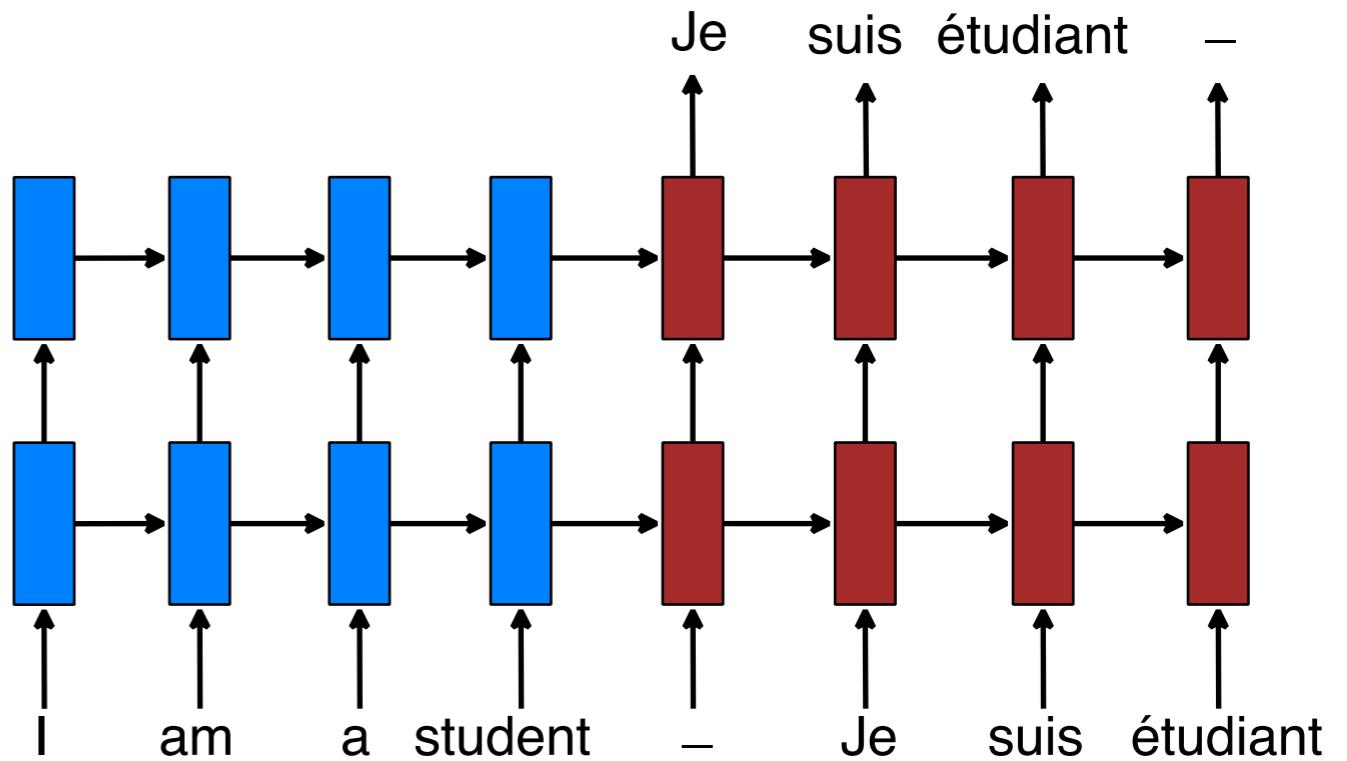
# Training vs. Testing

- *Training*
  - Correct translations are available.

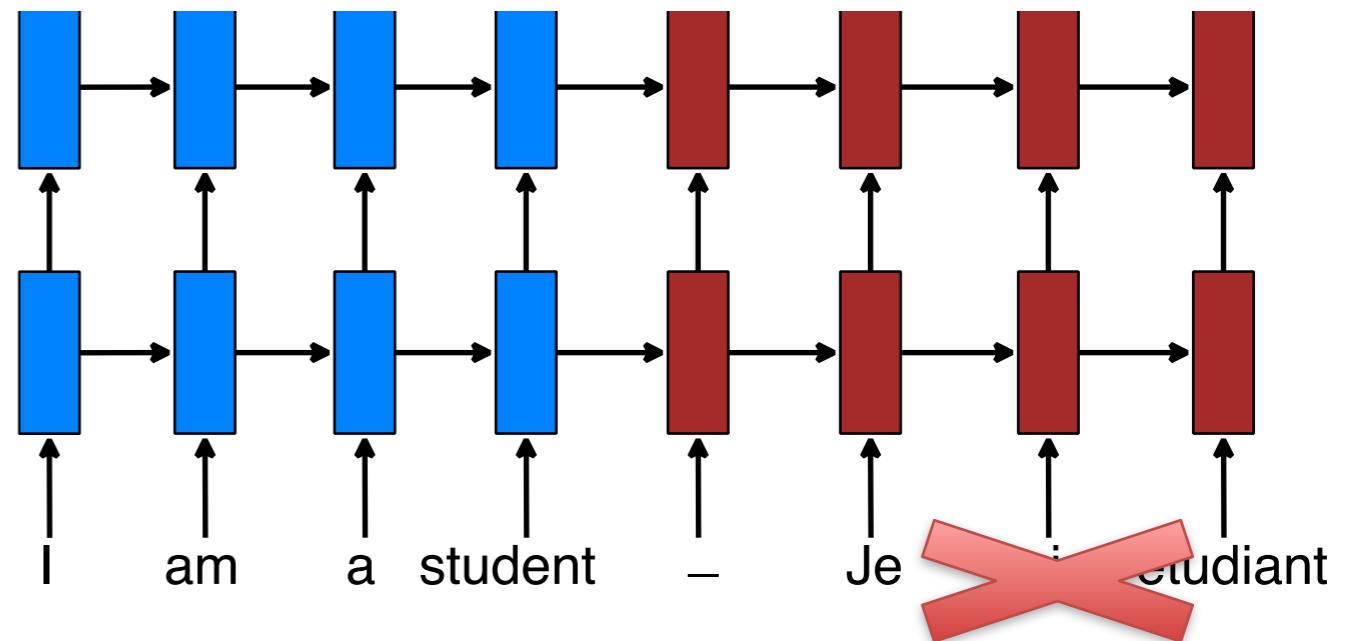


# Training vs. Testing

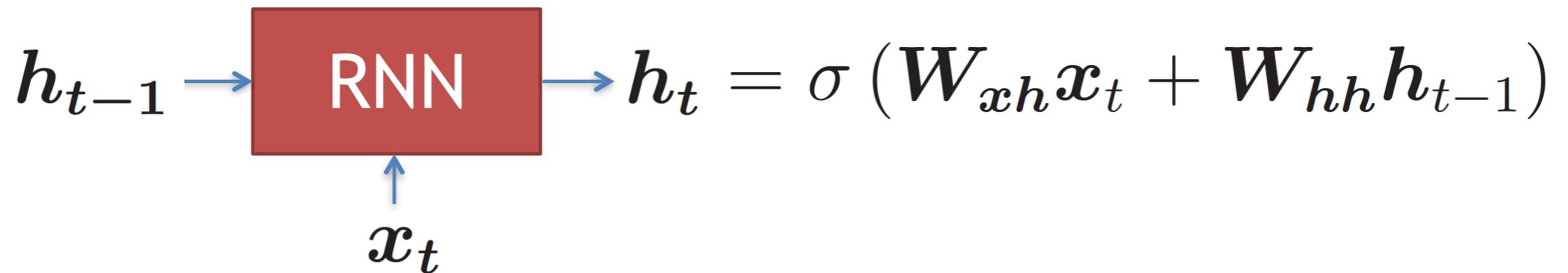
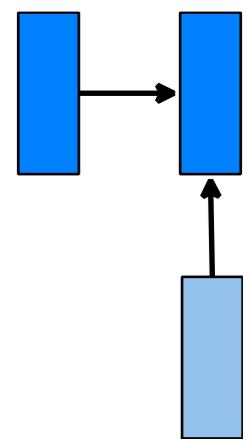
- *Training*
  - Correct translations are available.



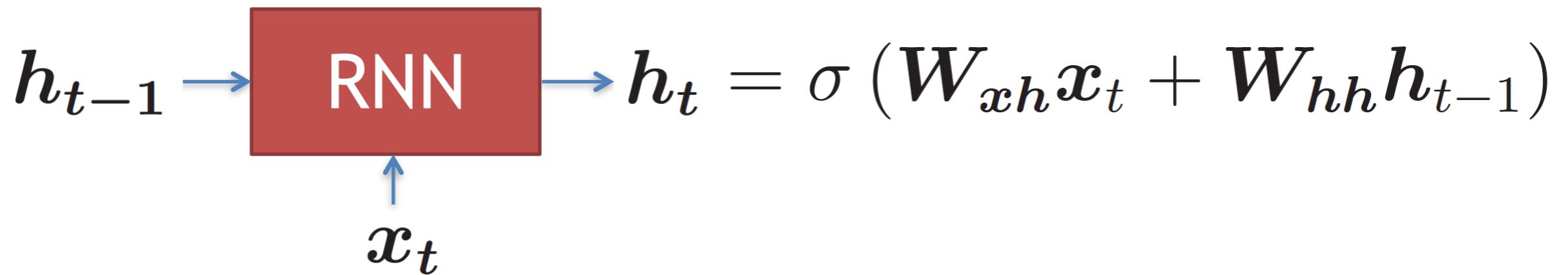
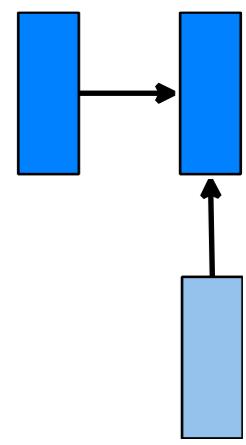
- *Testing*
  - Only source sentences are given.



# Recurrent types – vanilla RNN

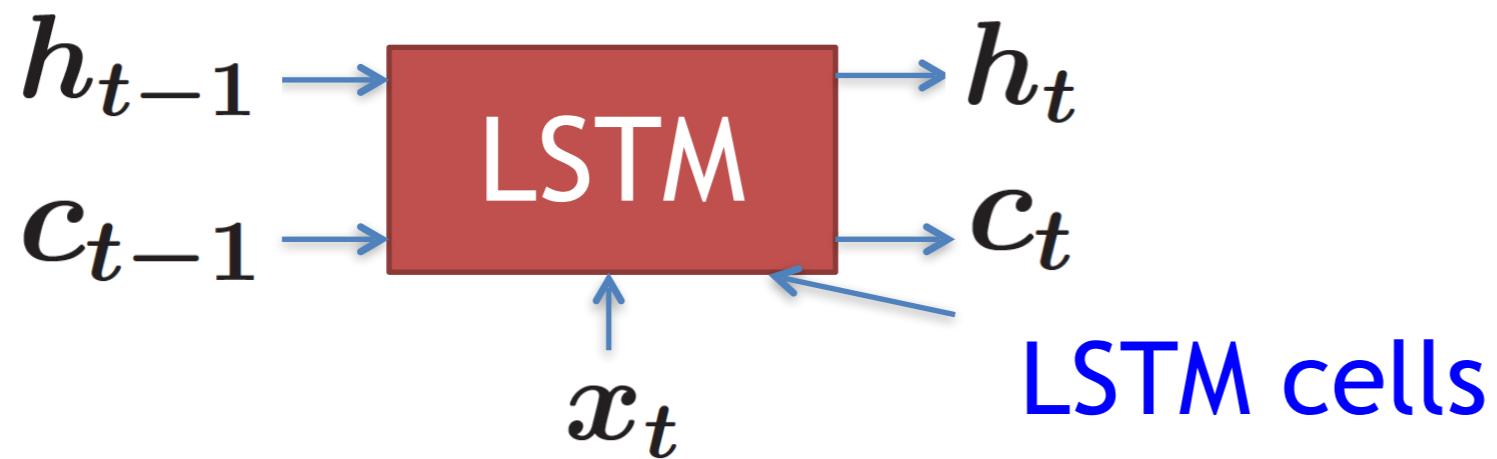


# Recurrent types – vanilla RNN



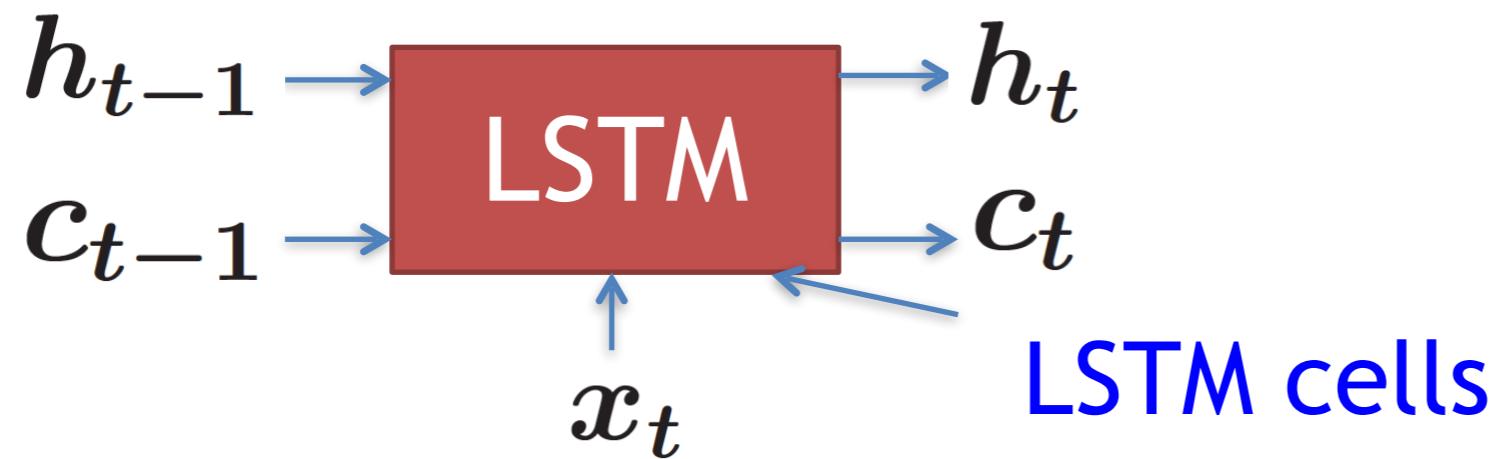
Vanishing gradient problem!

# Recurrent types – LSTM



# Recurrent types – LSTM

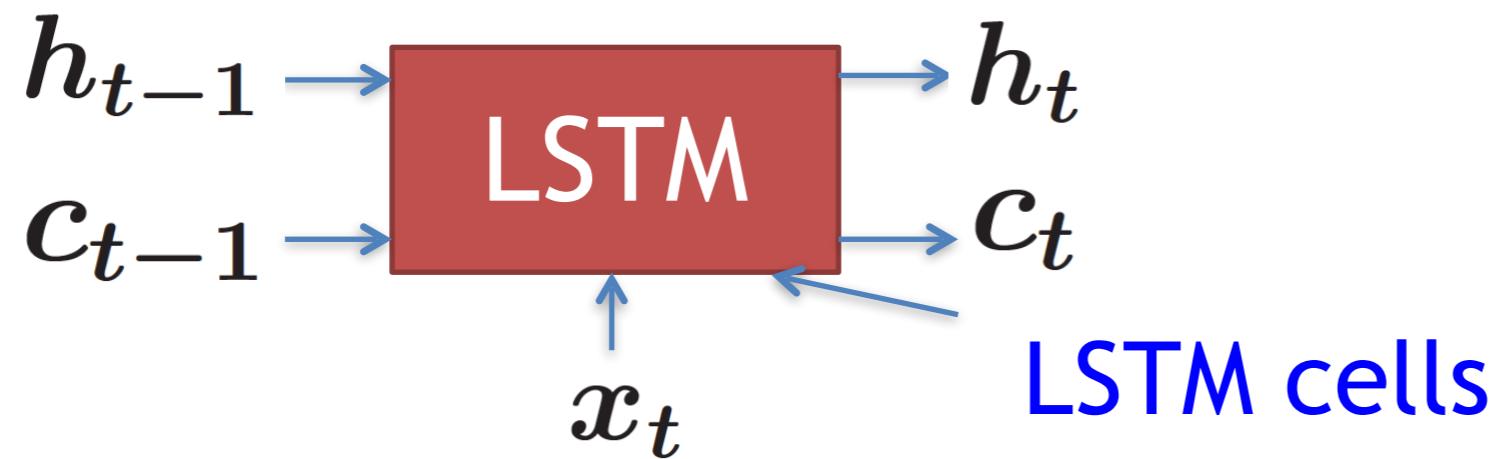
C'mon, it's  
been around  
for 20 years!



- Long-Short Term Memory (LSTM)
  - (Hochreiter & Schmidhuber, 1997)

# Recurrent types – LSTM

C'mon, it's  
been around  
for 20 years!

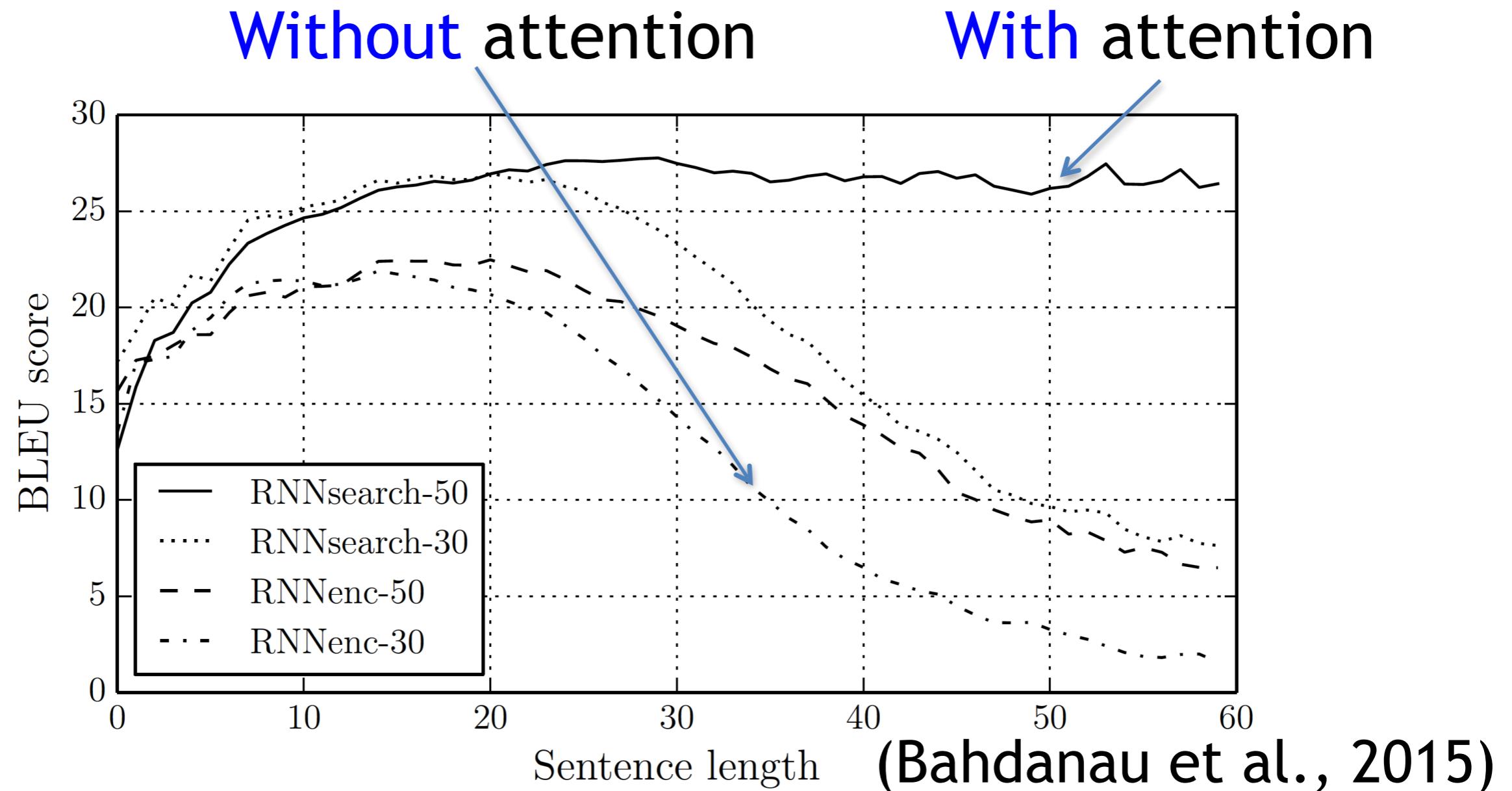


- Long-Short Term Memory (LSTM)
  - (Hochreiter & Schmidhuber, 1997)
- LSTM cells are **additively** updated
  - Make backprop through time easier.

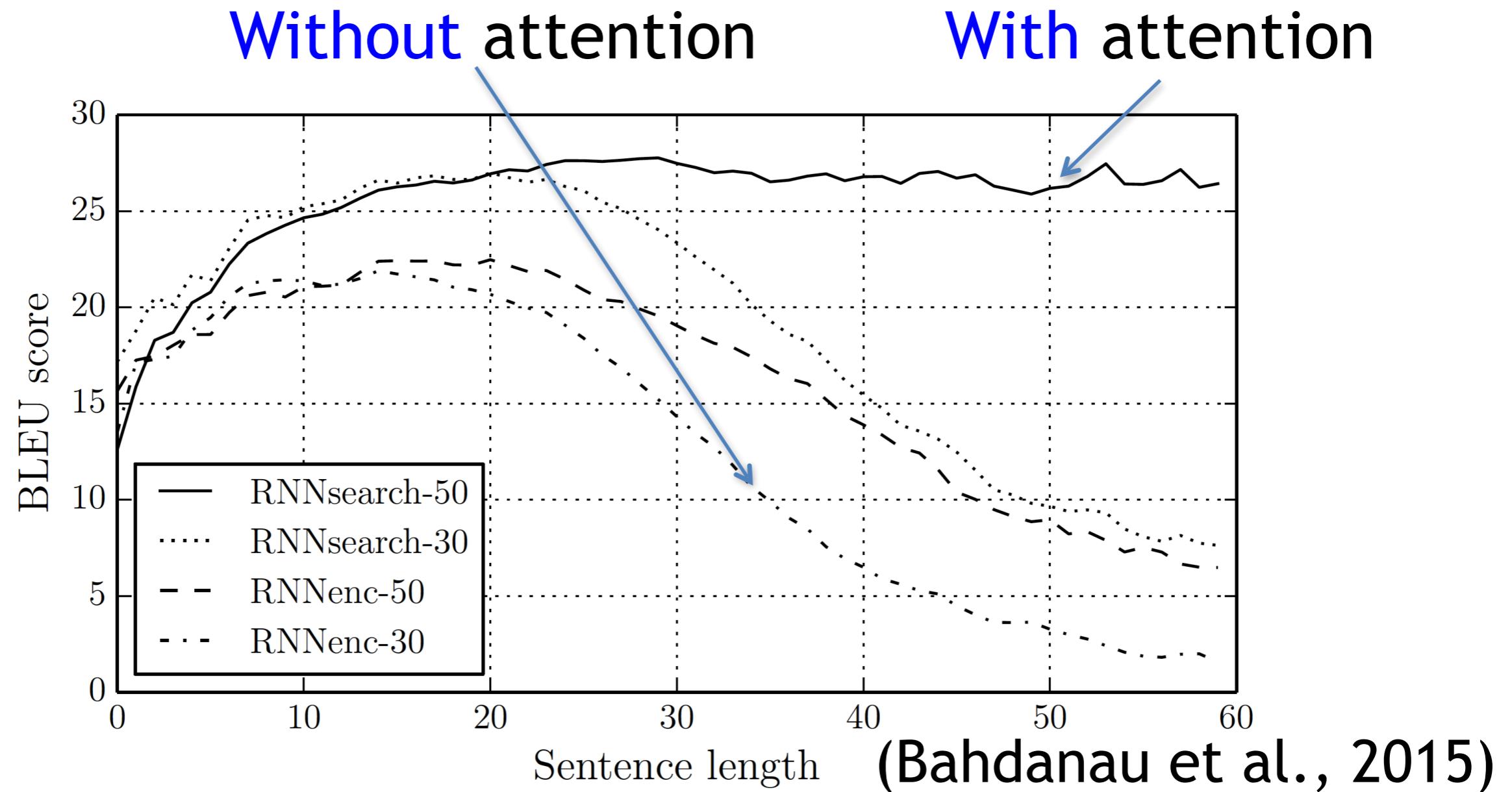
# Summary – NMT

- Few linguistic assumptions.
- Simple beam-search decoders.
- Good generalization to long sequences.

# Sentence Length Problem

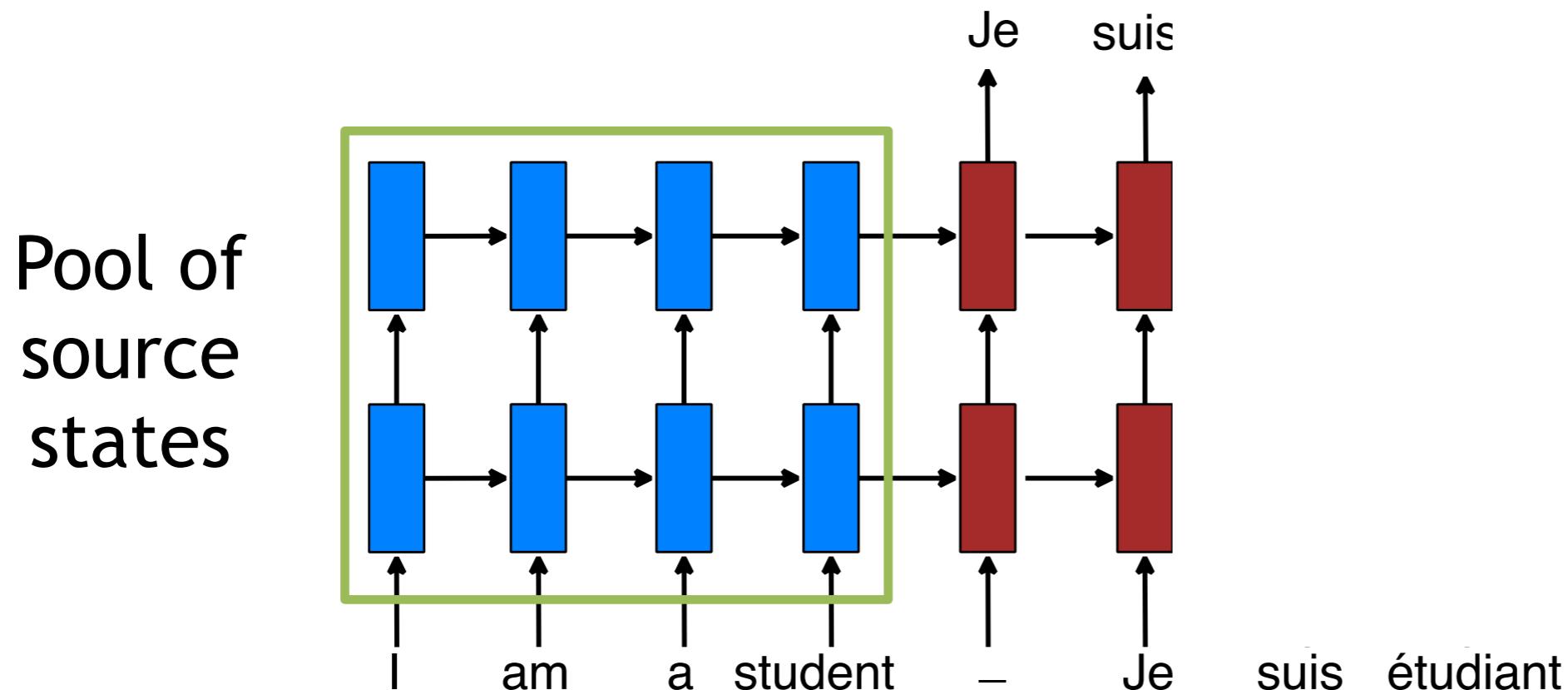


# Sentence Length Problem

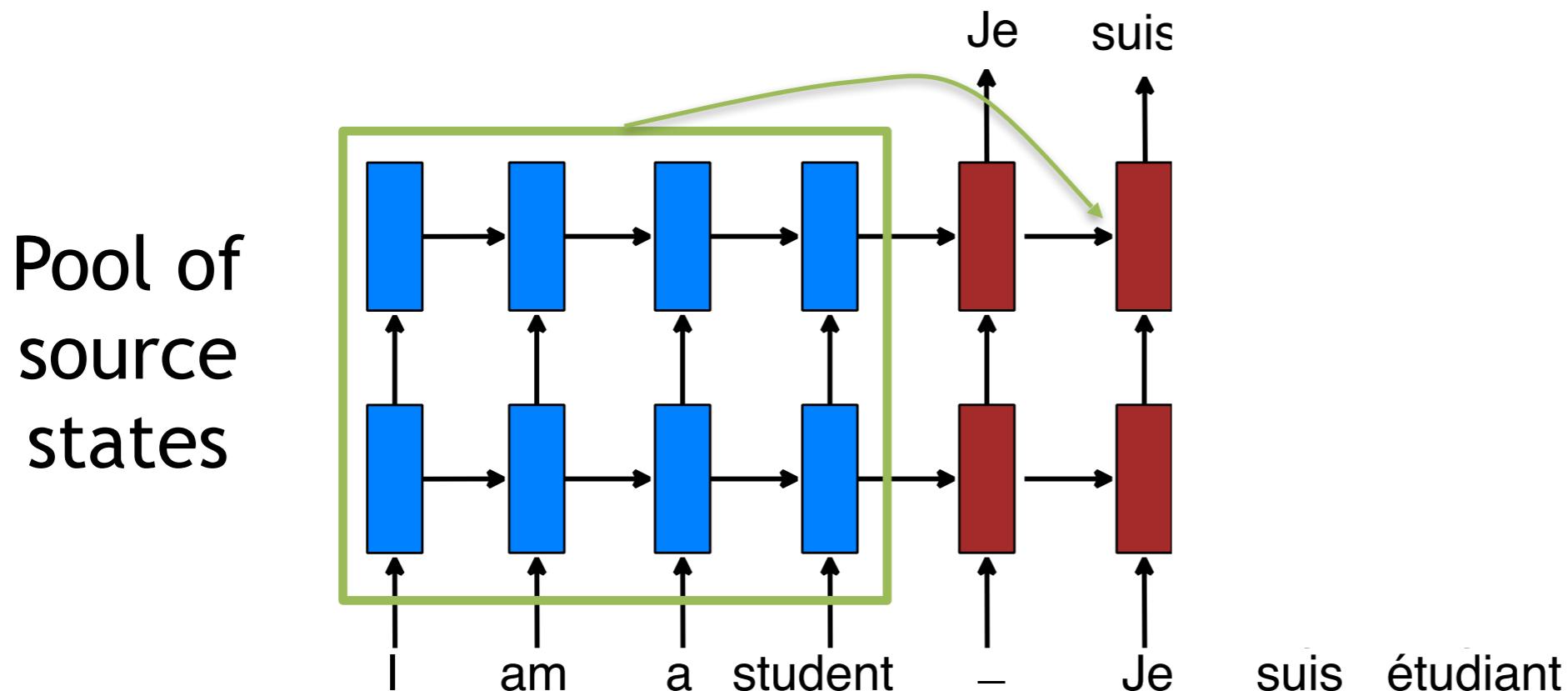


*Problem:* sentence meaning is represented by a fixed-dimensional vector.

# Attention Mechanism

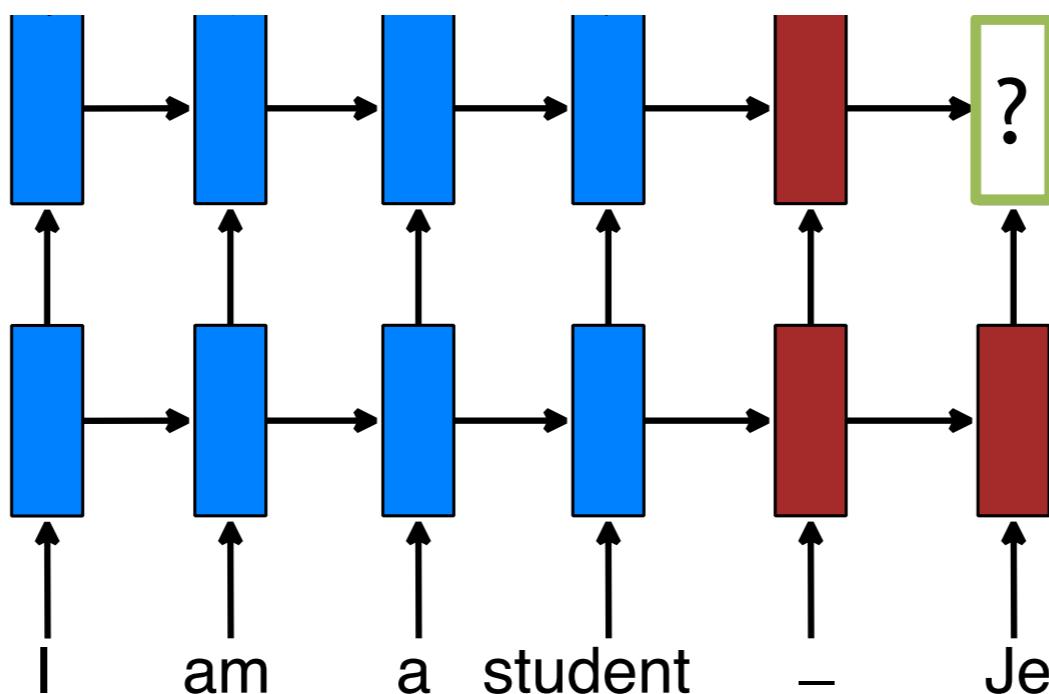


# Attention Mechanism



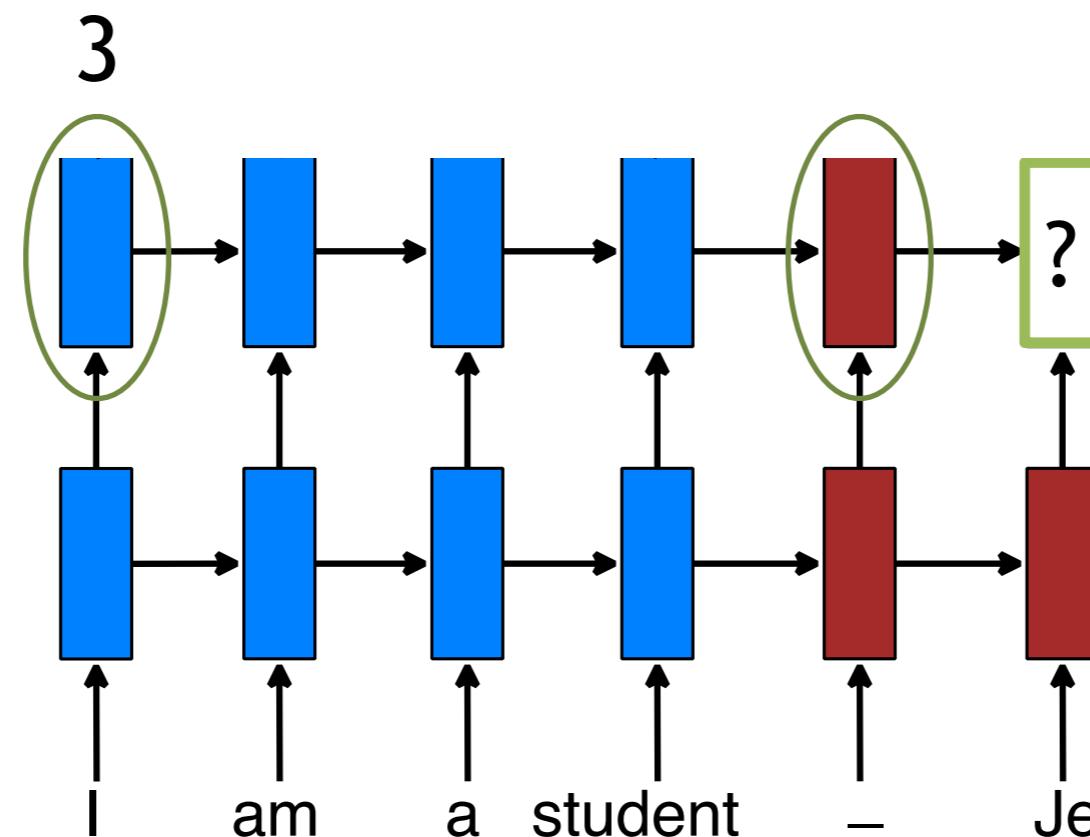
- Solution: random access memory
  - Retrieve as needed.

# Attention Mechanism



# Attention Mechanism – Scoring

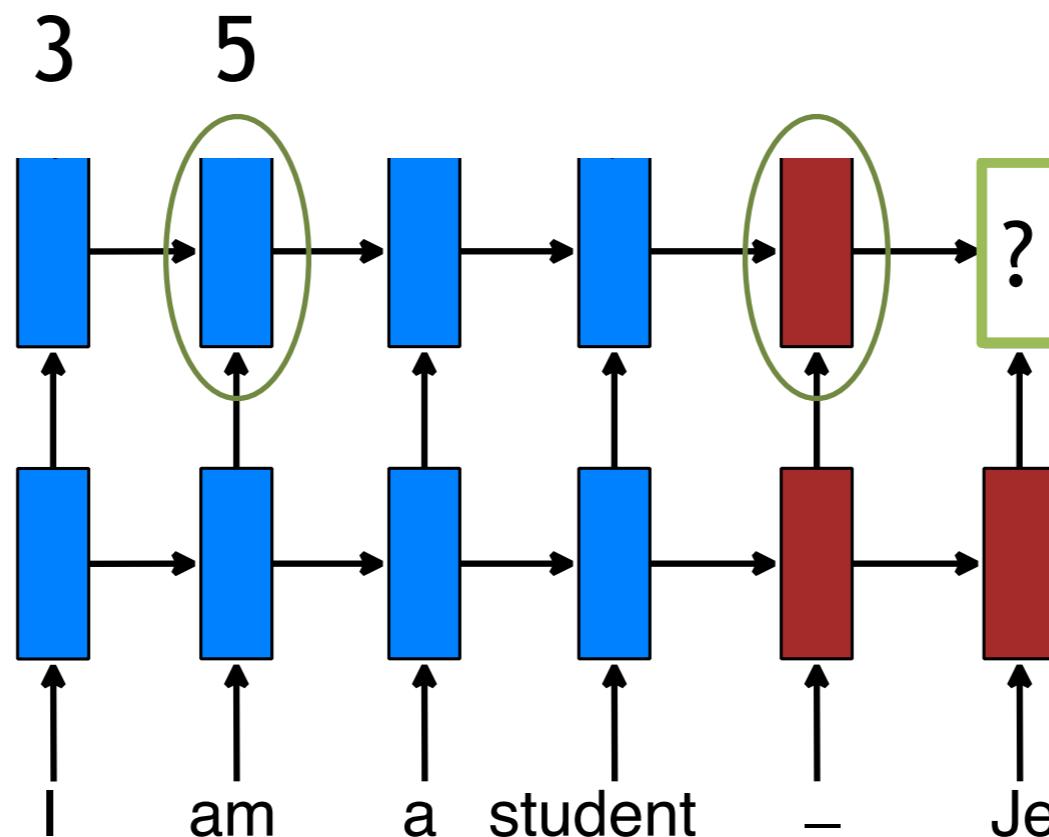
$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s)$$



- Compare target and source hidden states.

# Attention Mechanism – Scoring

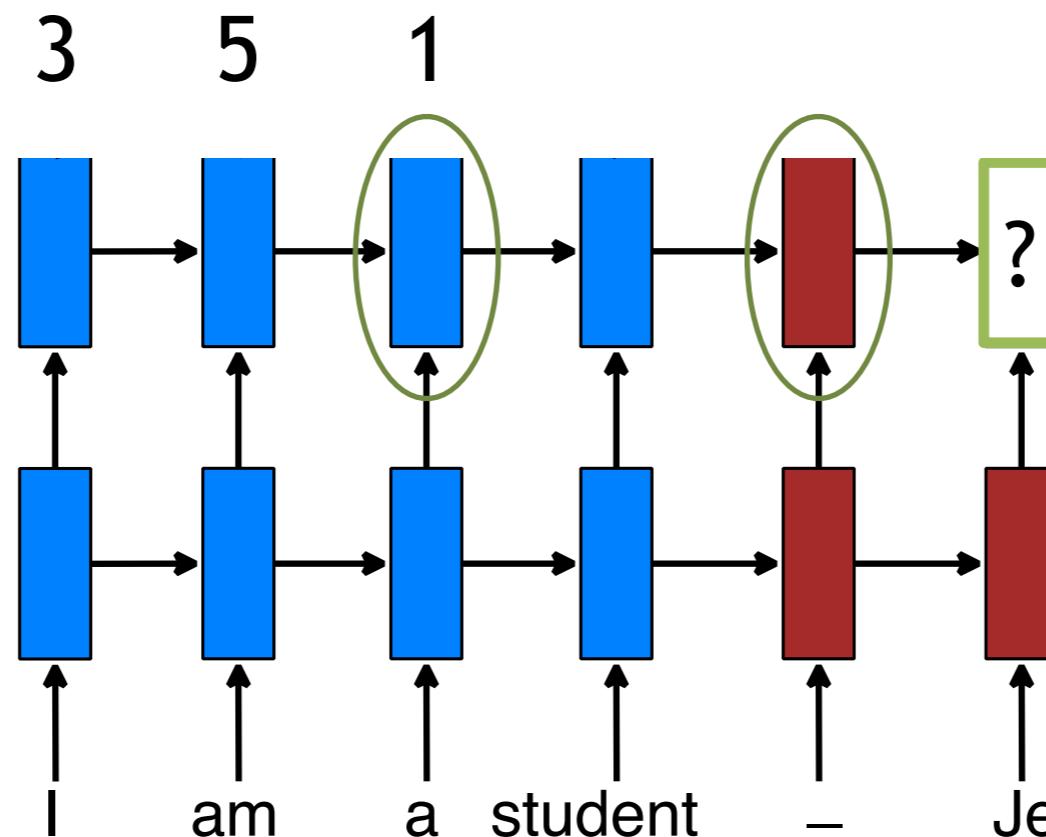
$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s)$$



- Compare target and source hidden states.

# Attention Mechanism – Scoring

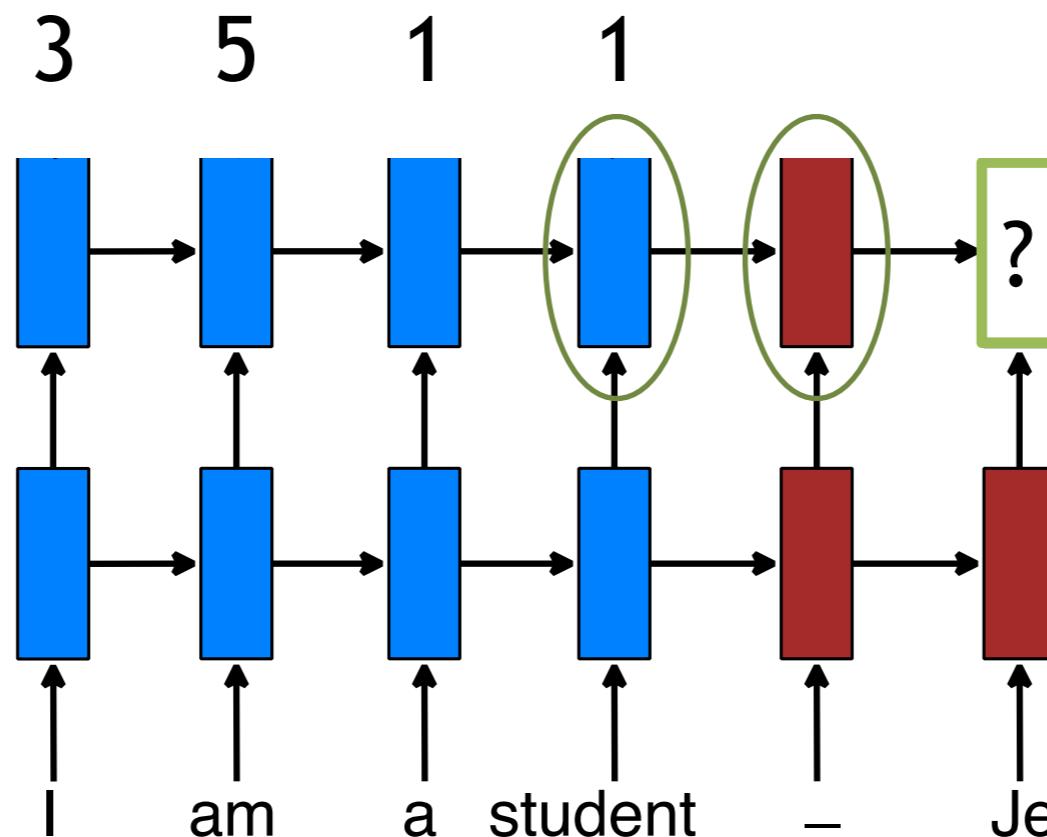
$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s)$$



- Compare target and source hidden states.

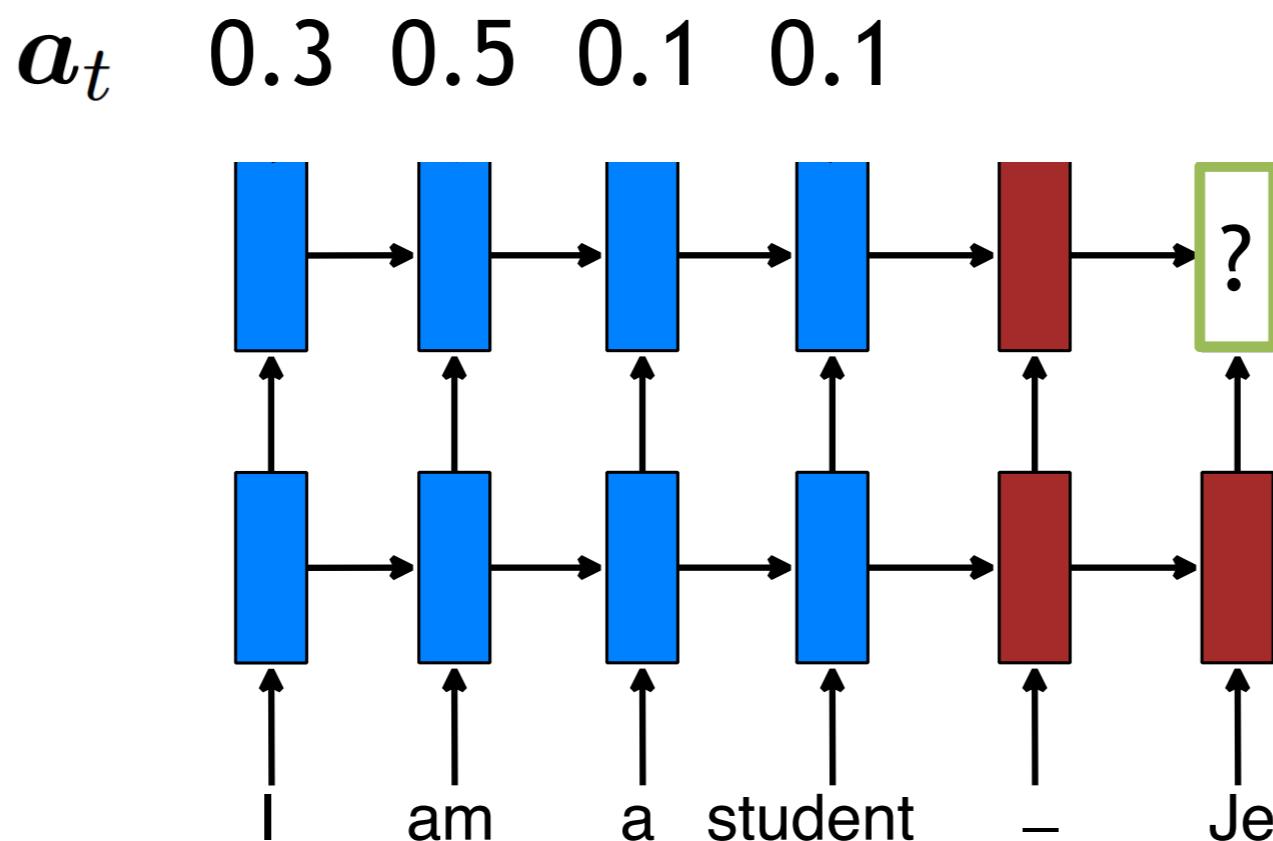
# Attention Mechanism – Scoring

$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s)$$



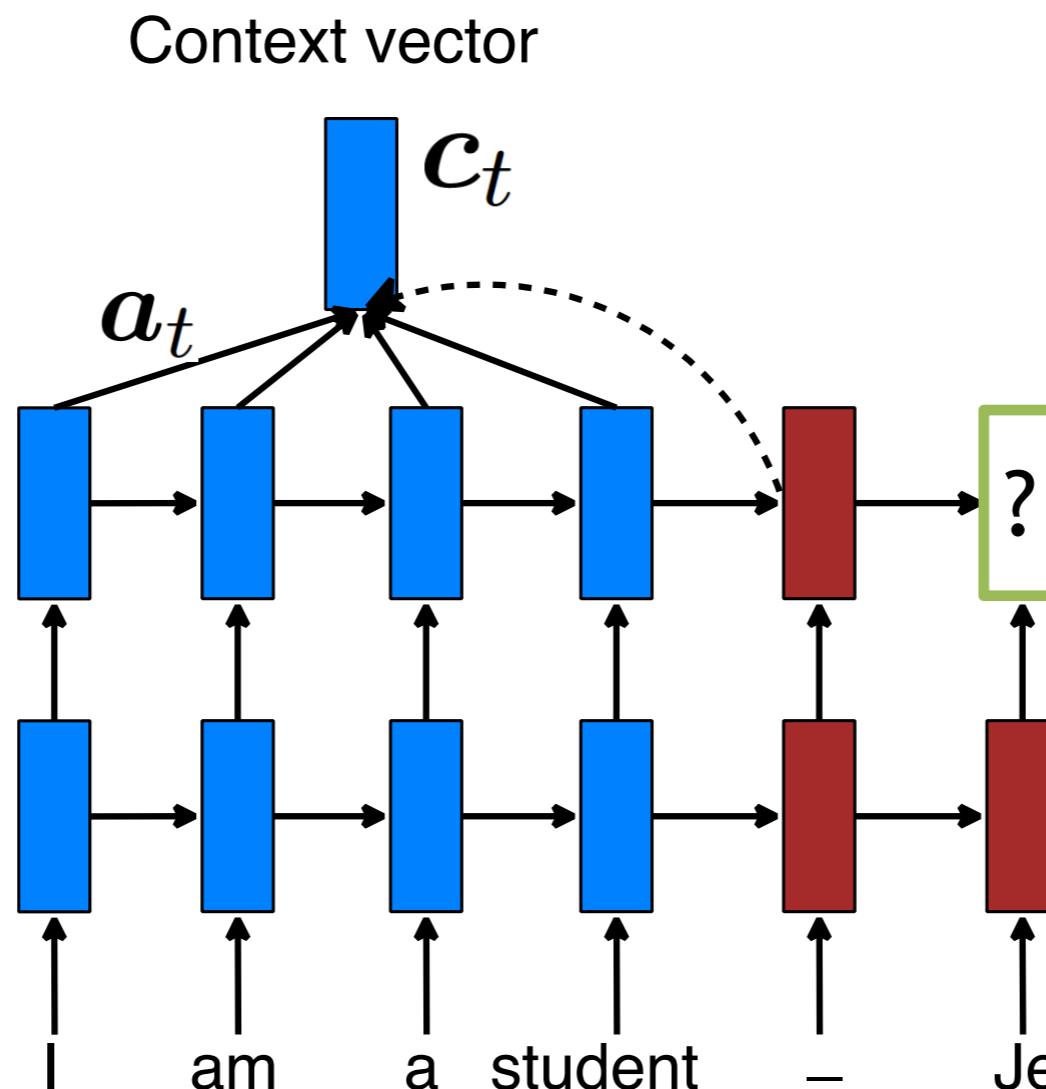
- Compare target and source hidden states.

# Attention Mechanism – Normalization



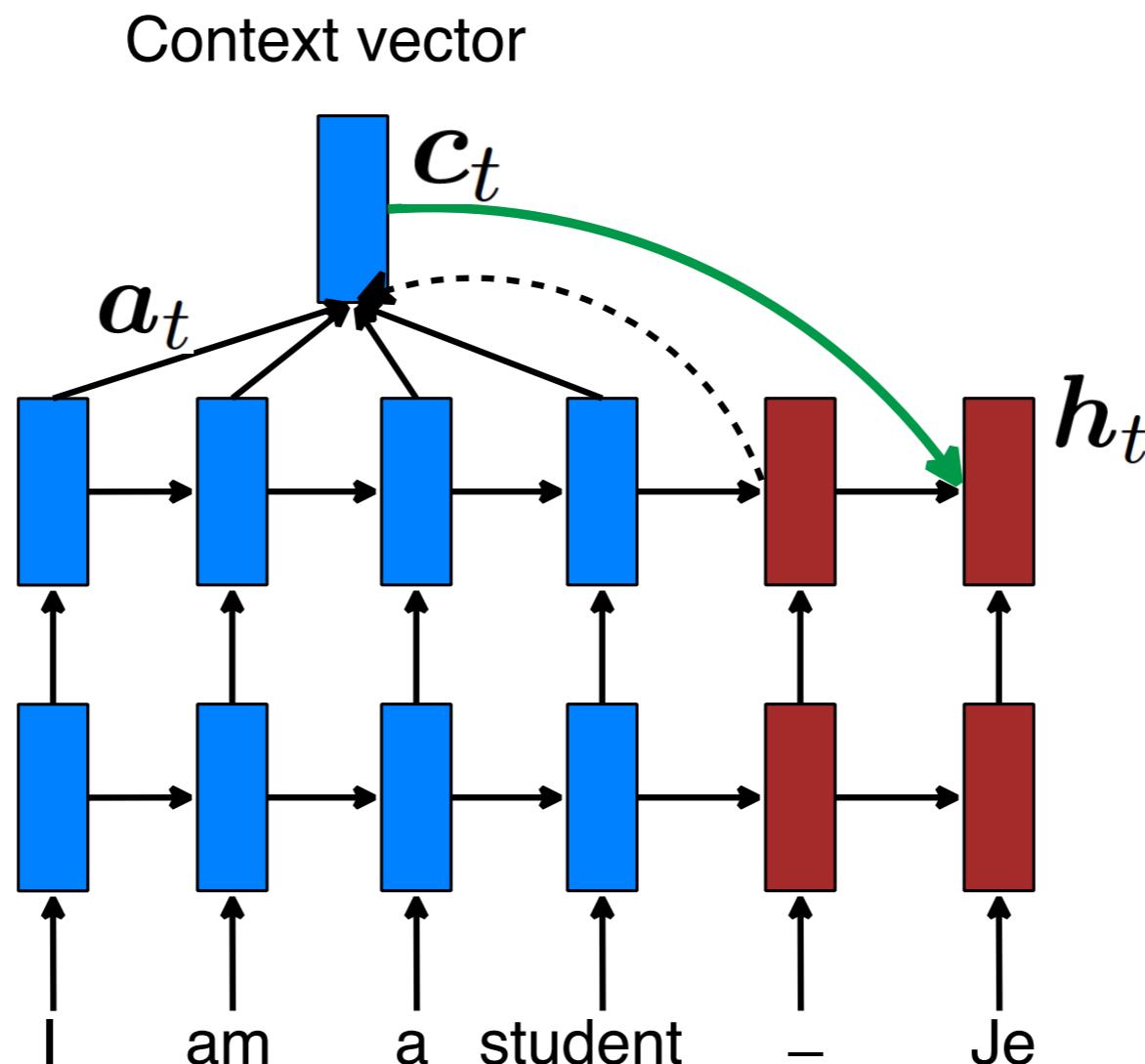
- Convert into alignment weights.

# Attention Mechanism – *Context vector*



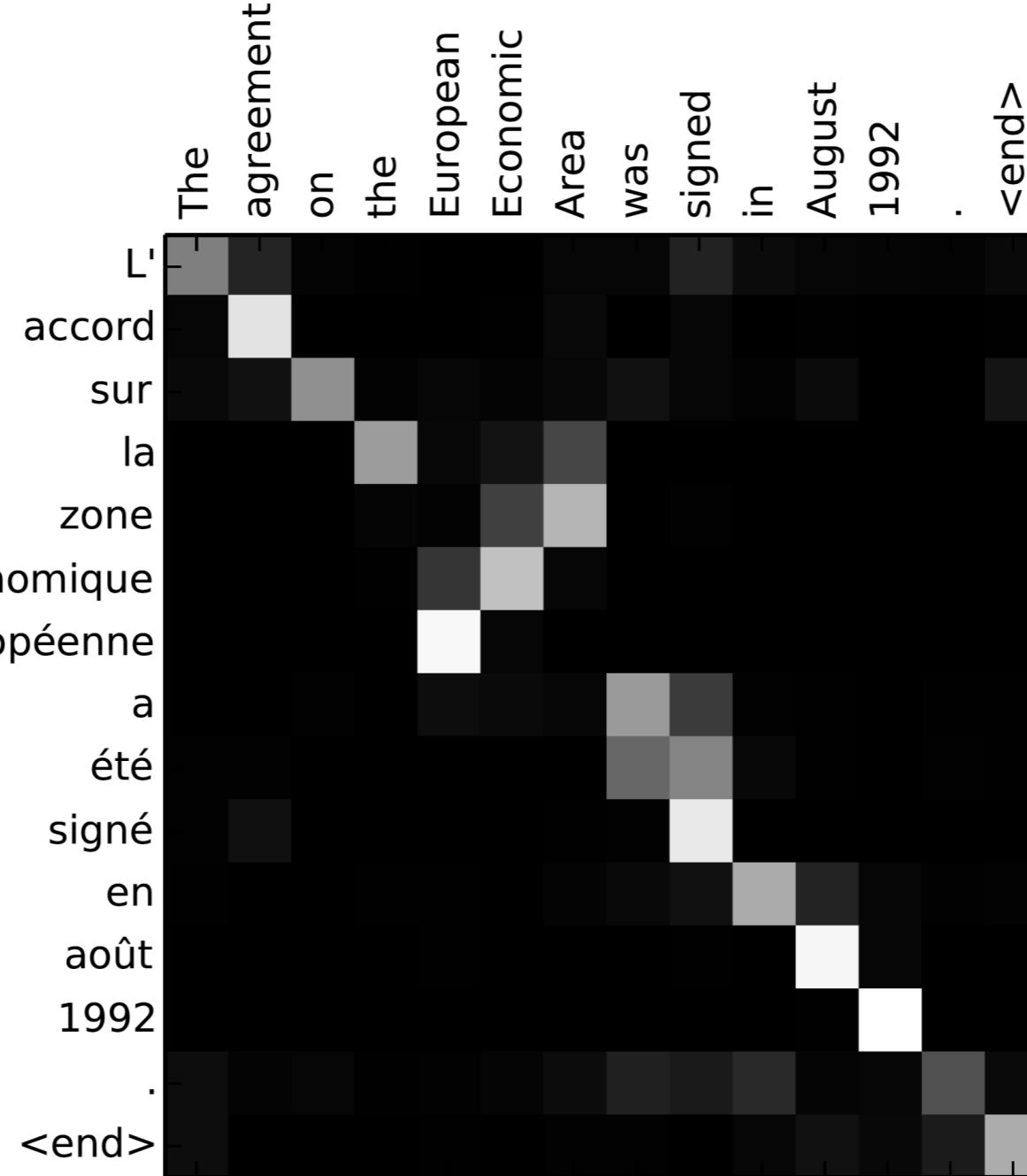
- Build **context** vector: weighted average.

# Attention Mechanism – *Hidden state*



- Compute the **next hidden state**.

# Alignments as a by-product



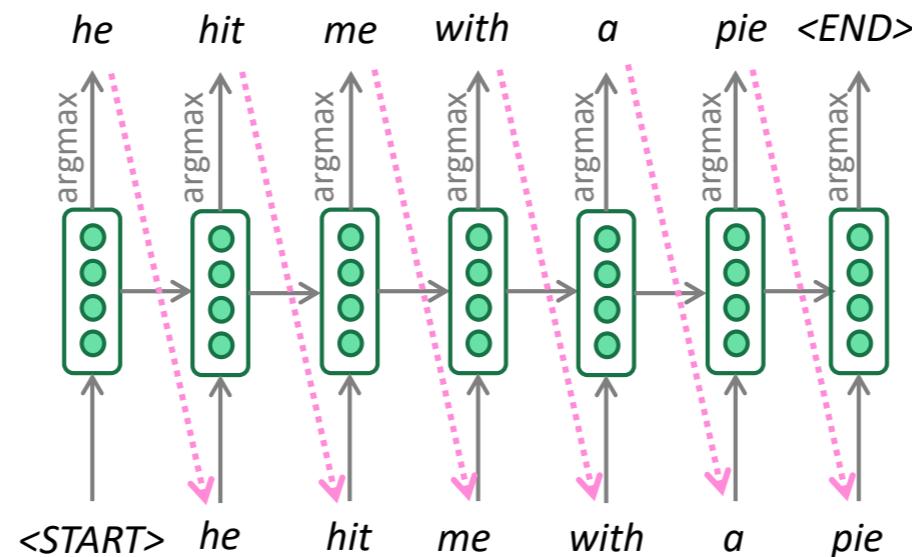
(Bahdanau et al., 2015)



# Beam Search

# Greedy Decoding

- We saw how to generate (or “decode”) the target sentence by taking argmax on each step of the decoder



- This is greedy decoding (take most probable word on each step)
- Problems with this method?

# Problems with Greedy Decoding

# Problems with Greedy Decoding

- Greedy decoding has no way to undo decisions!

# Problems with Greedy Decoding

- Greedy decoding has no way to undo decisions!
- Input: il a m'entarté (he hit me with a pie)

# Problems with Greedy Decoding

- Greedy decoding has no way to undo decisions!
- Input: il a m'entarté (he hit me with a pie)
  - → he \_\_\_\_\_

# Problems with Greedy Decoding

- Greedy decoding has no way to undo decisions!
- Input: il a m'entarté (he hit me with a pie)
  - → he \_\_\_\_\_
  - → he hit \_\_\_\_\_

# Problems with Greedy Decoding

- Greedy decoding has no way to undo decisions!
- Input: il a m'entarté (he hit me with a pie)
  - → he \_\_\_\_\_
  - → he hit \_\_\_\_\_
  - → he hit a \_\_\_\_\_ (whoops! no going back now...)

# Problems with Greedy Decoding

- Greedy decoding has no way to undo decisions!
- Input: il a m'entarté (he hit me with a pie)
  - → he \_\_\_\_\_
  - → he hit \_\_\_\_\_
  - → he hit a **a** \_\_\_\_\_ (whoops! no going back now...)
- How to fix this?

# Exhaustive search decoding

# Exhaustive search decoding

- Ideally we want to find a (length T) translation  $y$  that maximizes

$$\begin{aligned} P(y|x) &= P(y_1|x) P(y_2|y_1, x) P(y_3|y_1, y_2, x) \dots, P(y_T|y_1, \dots, y_{T-1}, x) \\ &= \prod_{t=1}^T P(y_t|y_1, \dots, y_{t-1}, x) \end{aligned}$$

# Exhaustive search decoding

- Ideally we want to find a (length T) translation  $y$  that maximizes

$$\begin{aligned} P(y|x) &= P(y_1|x) P(y_2|y_1, x) P(y_3|y_1, y_2, x) \dots, P(y_T|y_1, \dots, y_{T-1}, x) \\ &= \prod_{t=1}^T P(y_t|y_1, \dots, y_{t-1}, x) \end{aligned}$$

- We could try computing **all possible sequences**  $y$ 
  - This means that on each step  $t$  of the decoder, we're tracking  $V^t$  possible

# Exhaustive search decoding

- Ideally we want to find a (length T) translation  $y$  that maximizes

$$\begin{aligned} P(y|x) &= P(y_1|x) P(y_2|y_1, x) P(y_3|y_1, y_2, x) \dots, P(y_T|y_1, \dots, y_{T-1}, x) \\ &= \prod_{t=1}^T P(y_t|y_1, \dots, y_{t-1}, x) \end{aligned}$$

- We could try computing **all possible sequences**  $y$ 
  - This means that on each step  $t$  of the decoder, we're tracking  $V^t$  possible
  - partial translations, where  $V$  is vocab size
    - This  $O(V^T)$  complexity is far too expensive!

# Beam search decoding

# Beam search decoding

- **Core idea:** On each step of decoder, keep track of the  $k$  most probable partial translations (which we call hypotheses)
  - $k$  is the beam size (in practice around 5 to 10)

# Beam search decoding

- **Core idea:** On each step of decoder, keep track of the  $k$  most probable partial translations (which we call hypotheses)

- $k$  is the beam size (in practice around 5 to 10)
- A hypothesis has a score which is its log probability:

$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

- Scores are all negative, and higher score is better
- We search for high-scoring hypotheses, tracking top  $k$  on each step

# Beam search decoding

- **Core idea:** On each step of decoder, keep track of the  $k$  most probable partial translations (which we call hypotheses)

- $k$  is the beam size (in practice around 5 to 10)
- A hypothesis has a score which is its log probability:

$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

- Scores are all negative, and higher score is better
- We search for high-scoring hypotheses, tracking top  $k$  on each step
- Beam search is not guaranteed to find optimal solution

# Beam search decoding

- **Core idea:** On each step of decoder, keep track of the  $k$  most probable partial translations (which we call hypotheses)

- $k$  is the beam size (in practice around 5 to 10)
- A hypothesis has a score which is its log probability:

$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

- Scores are all negative, and higher score is better
- We search for high-scoring hypotheses, tracking top  $k$  on each step
- Beam search is not guaranteed to find optimal solution
- But much more efficient than exhaustive search!

# Beam search decoding: example

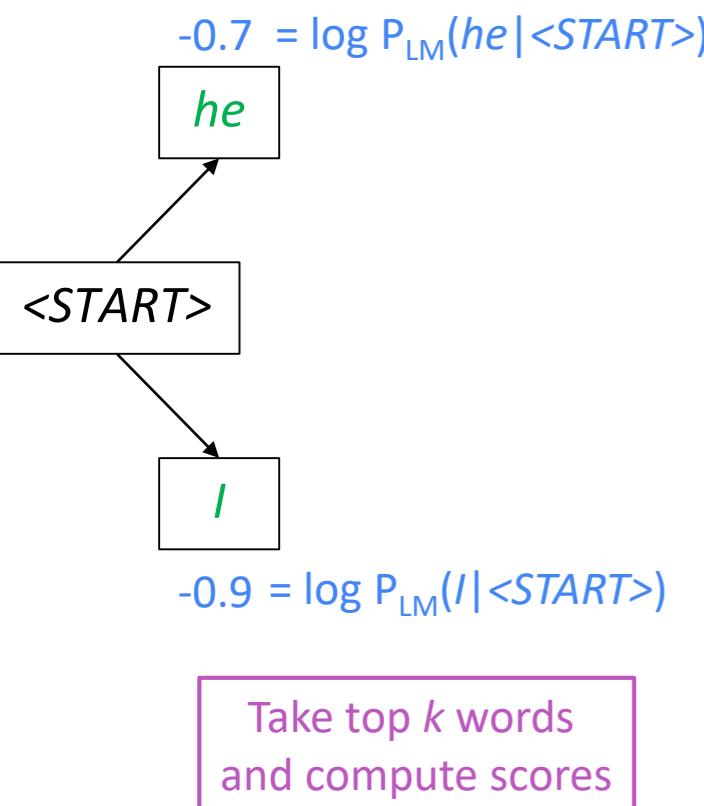
- Beam size =  $k = 2$ . Blue numbers =  $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$

<START>

Calculate prob  
dist of next word

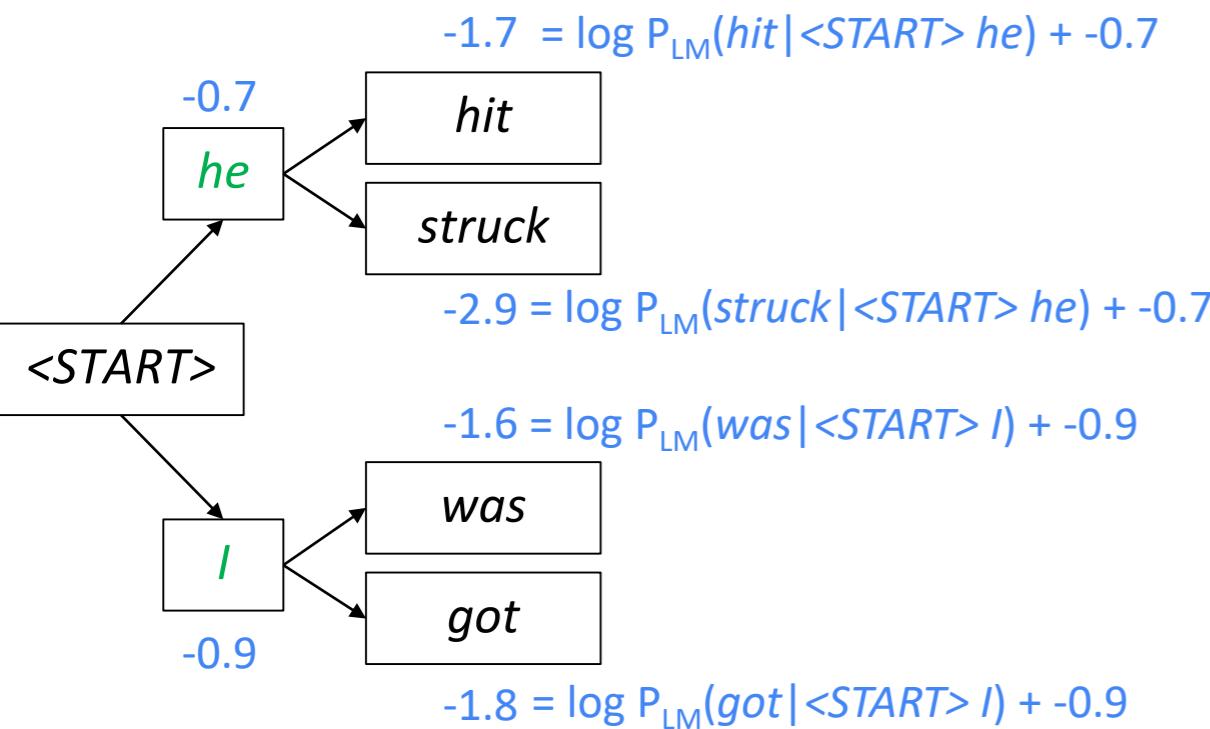
# Beam search decoding: example

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



# Beam search decoding: example

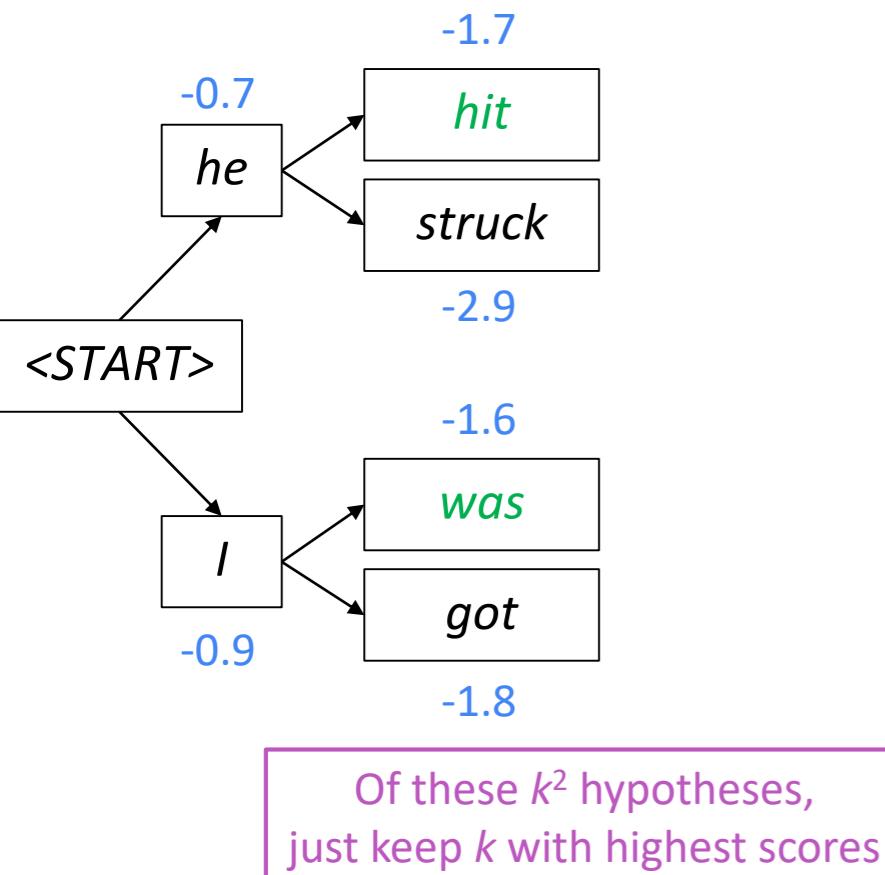
- Beam size =  $k = 2$ . Blue numbers =  $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the  $k$  hypotheses, find  
top  $k$  next words and calculate scores

# Beam search decoding: example

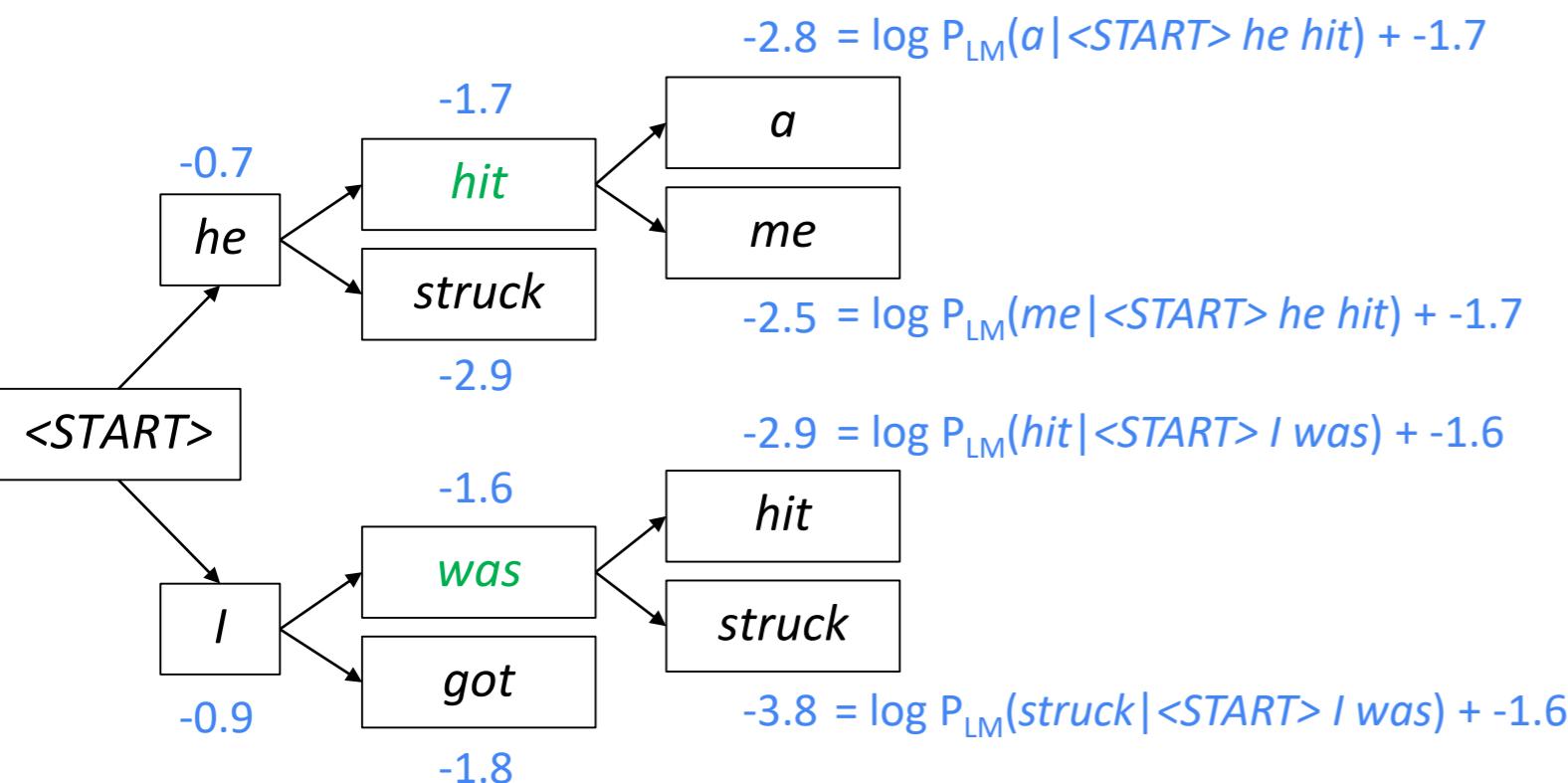
- Beam size =  $k = 2$ . Blue numbers =  $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



# Beam search decoding: example

- Beam size =  $k = 2$ . Blue numbers =

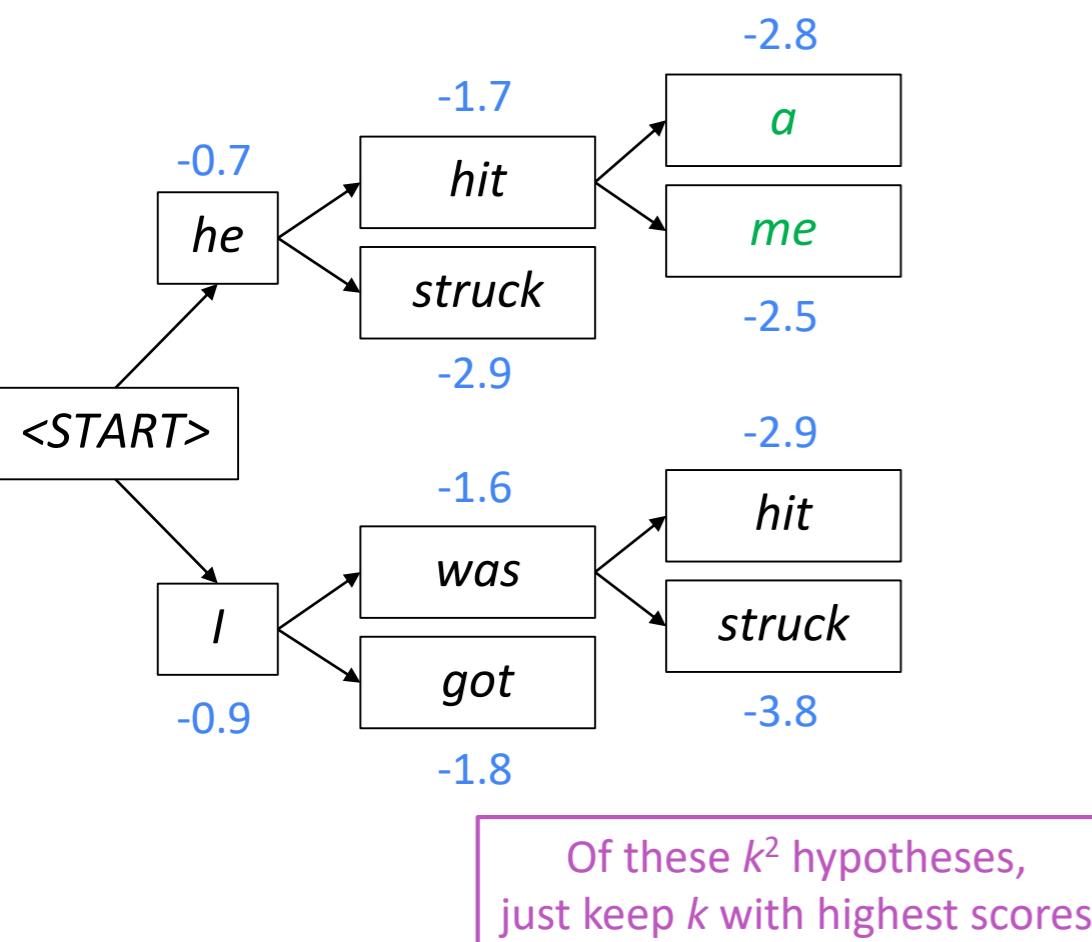
$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$



For each of the  $k$  hypotheses, find  
top  $k$  next words and calculate scores

# Beam search decoding: example

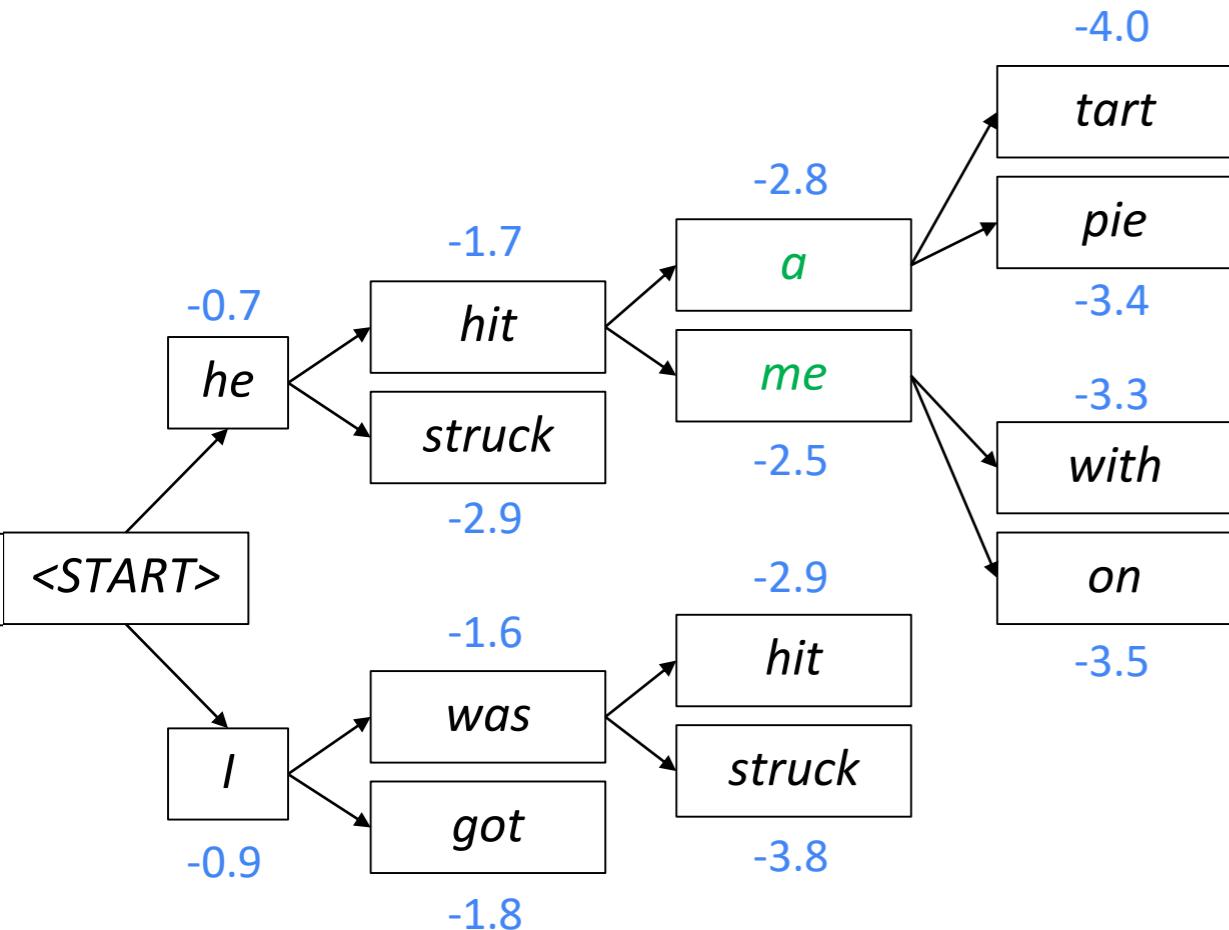
- Beam size =  $k = 2$ . Blue numbers =  $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



# Beam search decoding: example

- Beam size =  $k = 2$ . Blue numbers =

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

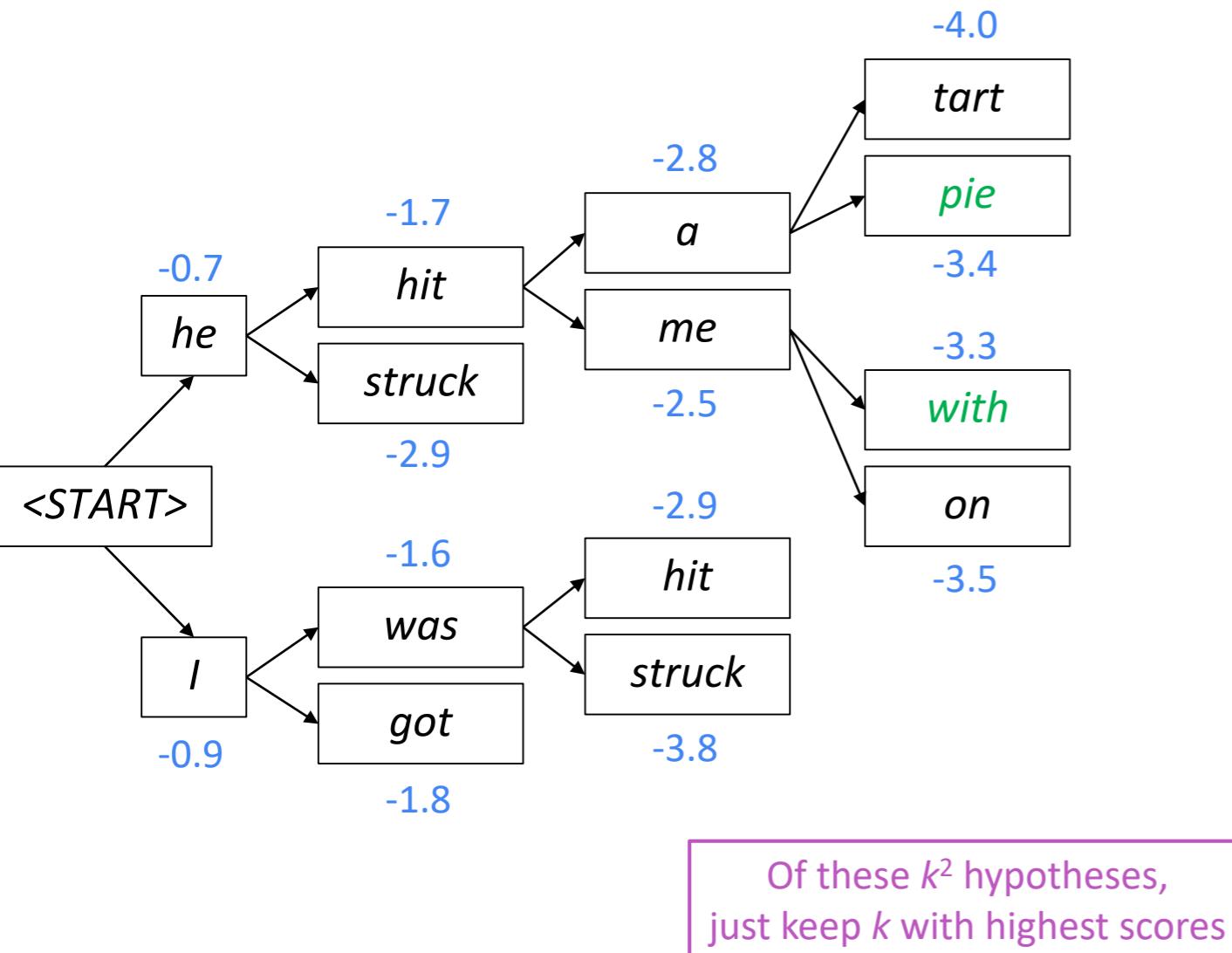


For each of the  $k$  hypotheses, find  
top  $k$  next words and calculate scores

# Beam search decoding: example

- Beam size =  $k = 2$ . Blue numbers =

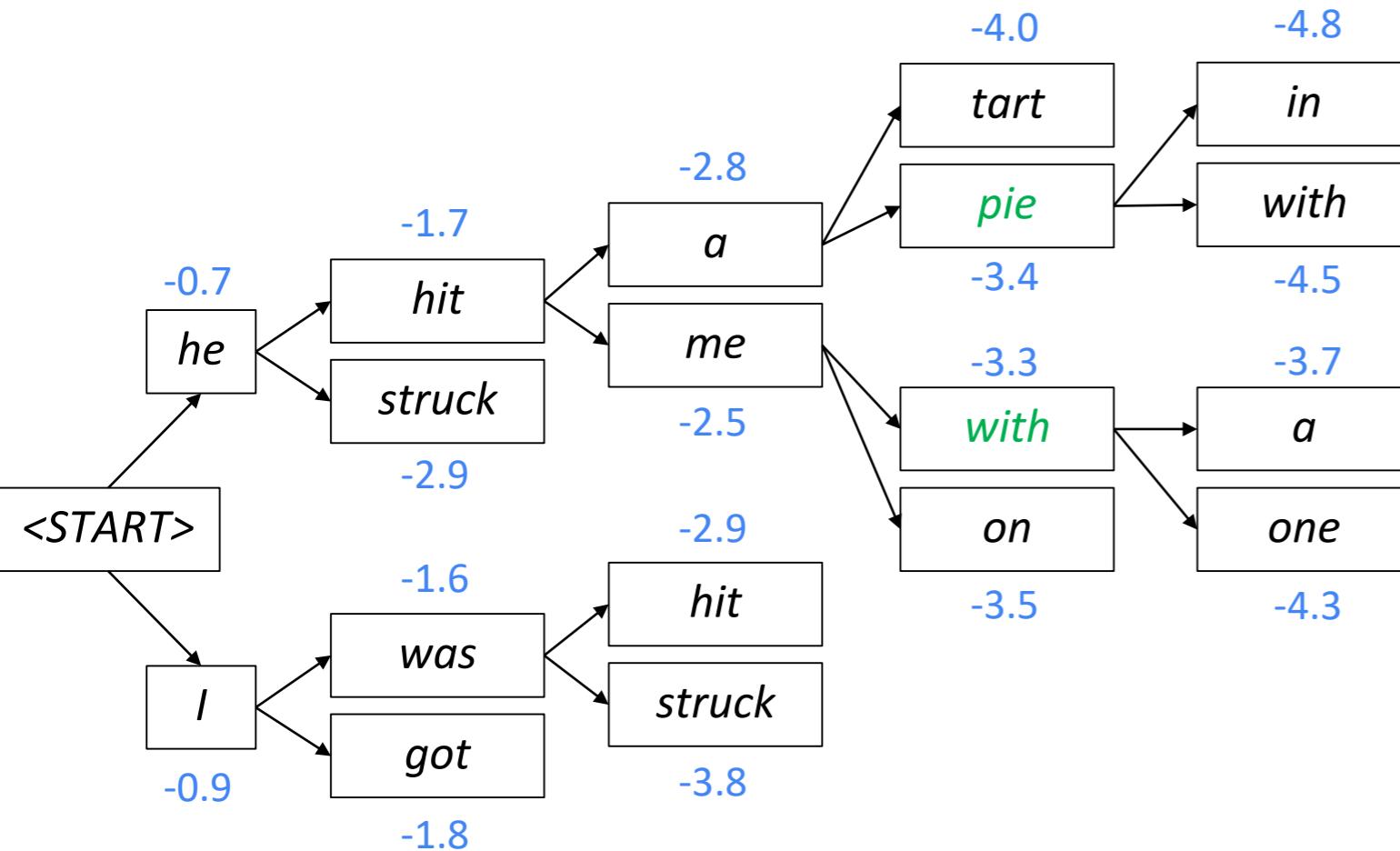
$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$



# Beam search decoding: example

- Beam size =  $k = 2$ . Blue numbers =

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

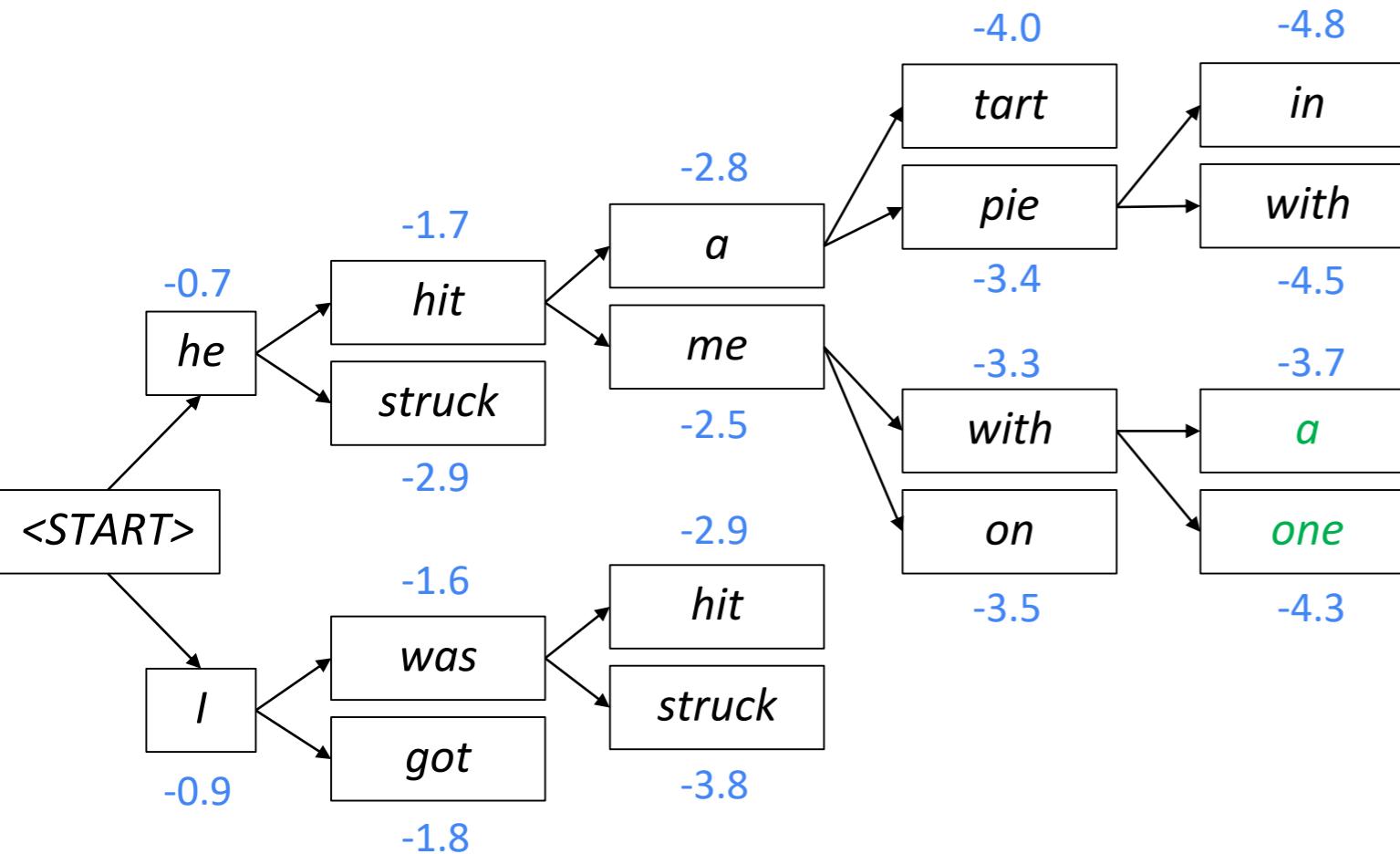


For each of the  $k$  hypotheses, find  
top  $k$  next words and calculate scores

# Beam search decoding: example

- Beam size =  $k = 2$ . Blue numbers =

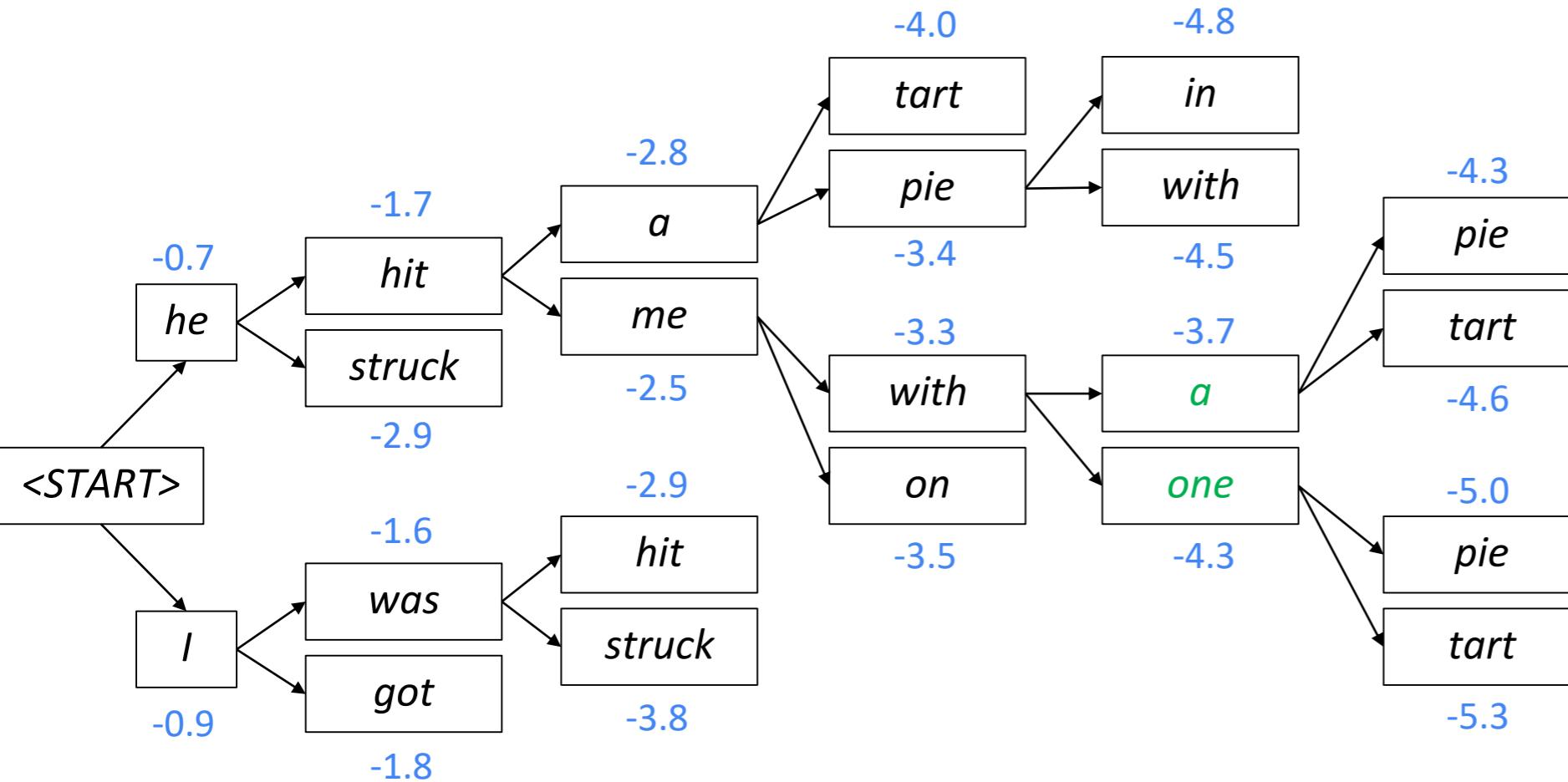
$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$



Of these  $k^2$  hypotheses,  
just keep  $k$  with highest scores

# Beam search decoding: example

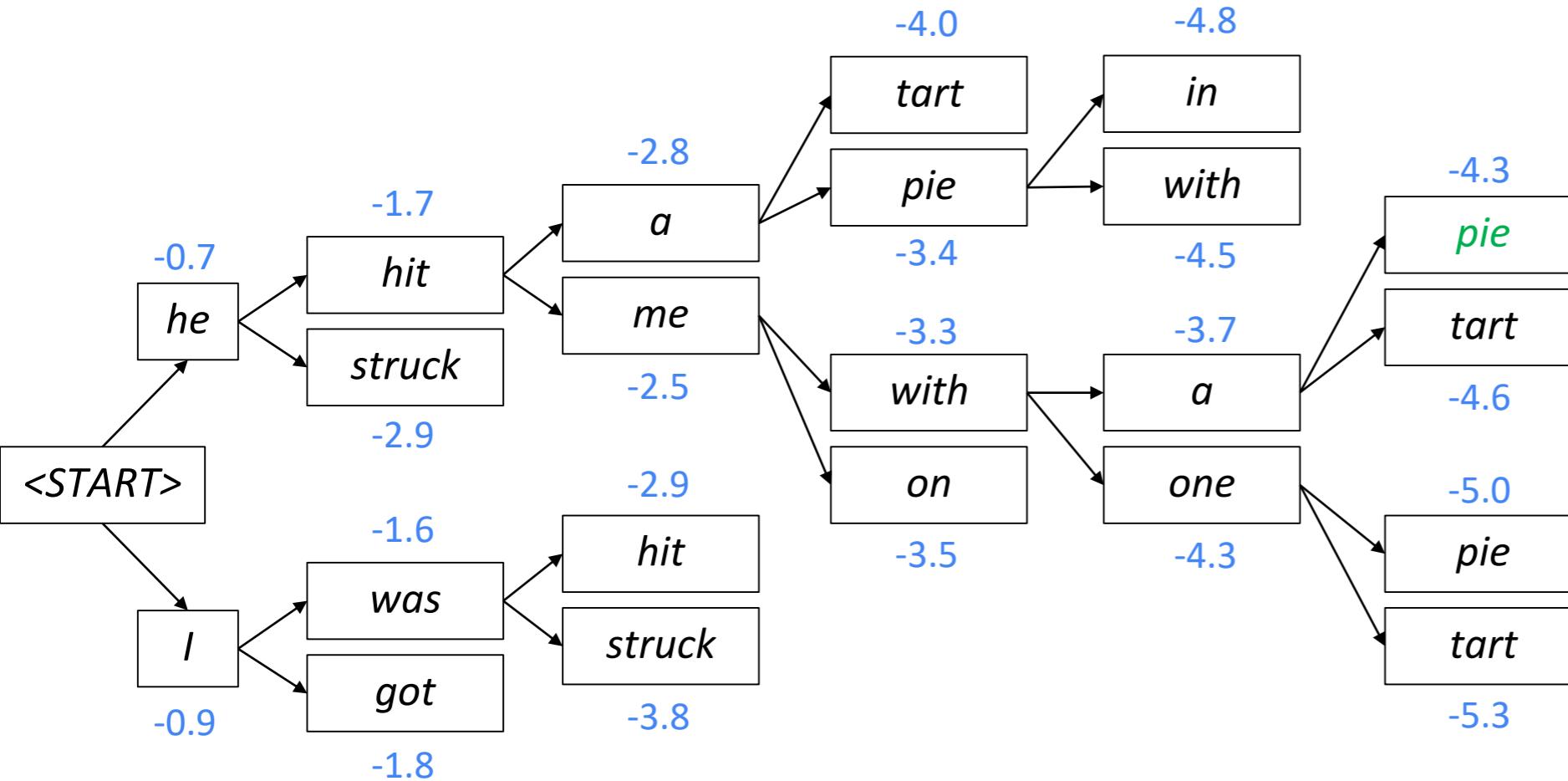
- Beam size =  $k = 2$ . Blue numbers =  $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the  $k$  hypotheses, find top  $k$  next words and calculate scores

# Beam search decoding: example

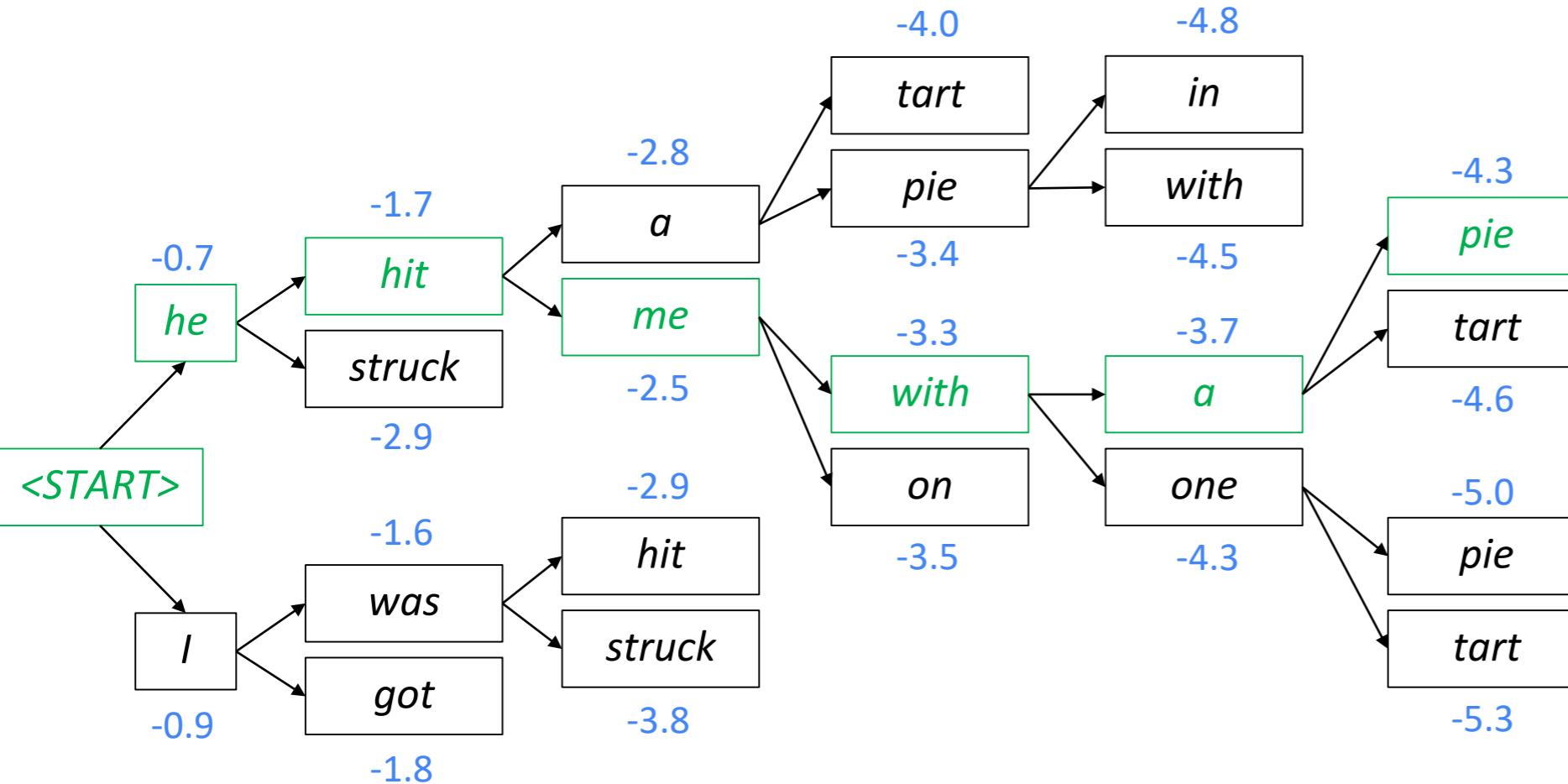
- Beam size =  $k = 2$ . Blue numbers =  $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



This is the top-scoring hypothesis!

# Beam search decoding: example

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



Backtrack to obtain the full hypothesis

# Beam search decoding: stopping criterion

# Beam search decoding: stopping criterion

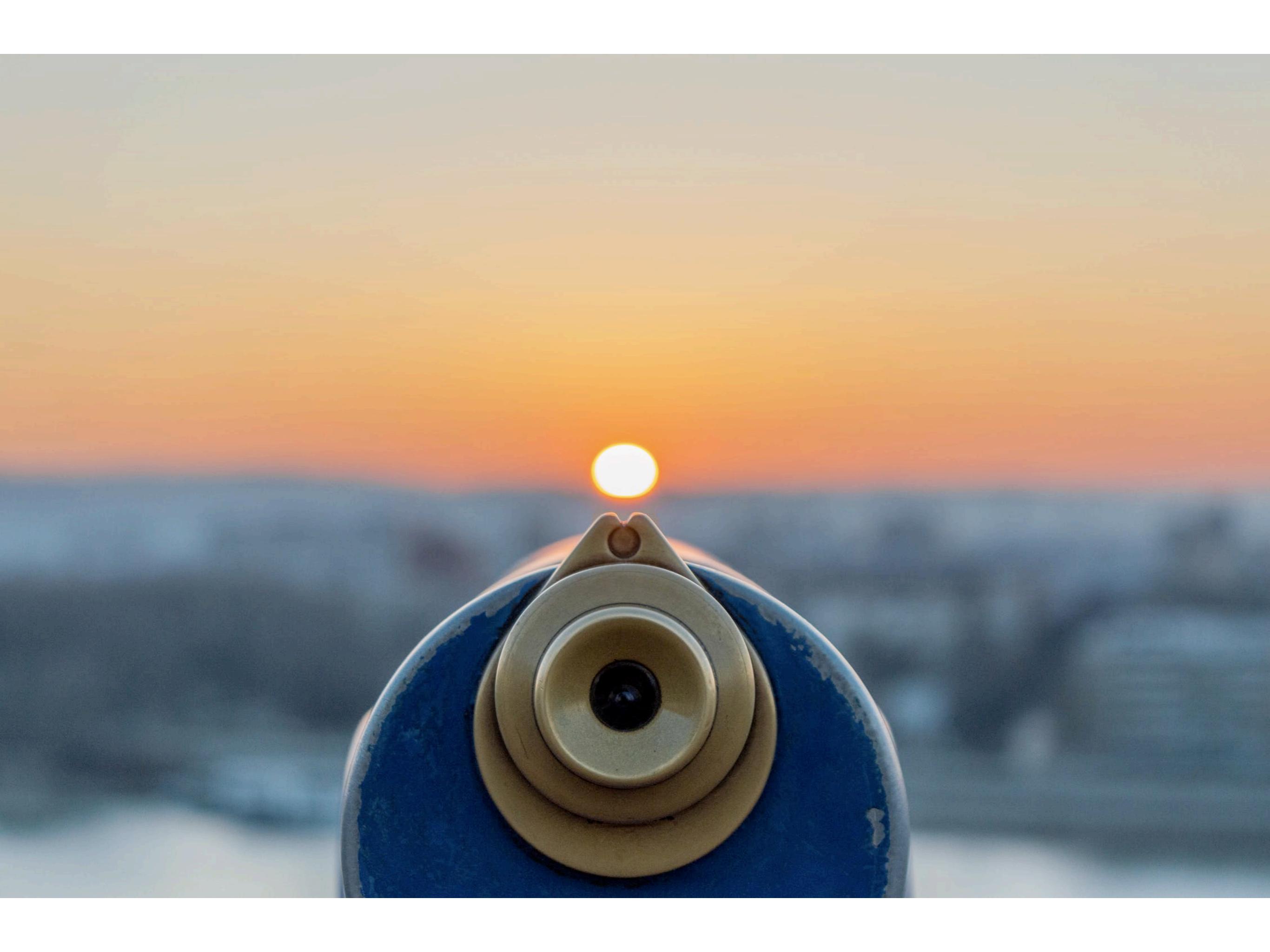
- In **greedy decoding**, usually we decode until the model produces a **<END>** token
  - For example: <START> he hit me with a pie <END>

# Beam search decoding: stopping criterion

- In **greedy decoding**, usually we decode until the model produces a **<END> token**
  - For example: <START> he hit me with a pie <END>
- In **beam search decoding**, different hypotheses may produce <END> tokens on **different timesteps**
  - When a hypothesis produces <END>, that hypothesis is **complete**.
  - **Place it aside** and continue exploring other hypotheses via beam search.

# Beam search decoding: stopping criterion

- In **greedy decoding**, usually we decode until the model produces a **<END> token**
  - For example: <START> he hit me with a pie <END>
- In **beam search decoding**, different hypotheses may produce <END> tokens on **different timesteps**
  - When a hypothesis produces <END>, that hypothesis is **complete**.
  - **Place it aside** and continue exploring other hypotheses via beam search.
- Usually we continue beam search until:
  - We reach timestep T (where T is some pre-defined cutoff), or
  - We have at least n completed hypotheses (where n is pre-defined cutoff)



Looking forward in the  
semester

Homework 4: How do you  
learn the parameters for LDA?

# Homework 4: How do you learn the parameters for LDA?

- One week homework where we give you two weeks

# Homework 4: How do you learn the parameters for LDA?

- One week homework where we give you two weeks
- You'll be implementing LDA (well, a small part of it)

# Homework 4: How do you learn the parameters for LDA?

- One week homework where we give you two weeks
- You'll be implementing LDA (well, a small part of it)
- We'll give you a bunch of skeleton code; your part is ~15 lines

# Homework 4: How do you learn the parameters for LDA?

- One week homework where we give you two weeks
- You'll be implementing LDA (well, a small part of it)
- We'll give you a bunch of skeleton code; your part is ~15 lines
- Your goal is to learn how the update rule to LDA works—the Gibbs Sampling part
  - The lecture slides will be very helpful here

# Project Updates are due ~~next~~ in two weeks

- Extra time due to chaos in the world
- Big goal: generate some baseline system and run your model all the way through it
- Update to your current report
- More details posted tonight

# Midterm is in three weeks

- Take-home midterm
- We'll vote on a date
- 4 open-ended questions + A few if-you-took-NLP-you-should-know this short answer questions
- We'll do a review session the week before