ANLP

20 - attention & self-attention (seq2seq II)

David Schlangen
University of Potsdam, MSc Cognitive Systems
Winter 2019 / 2020

NMT research continues

NMT is the **flagship task** for NLP Deep Learning

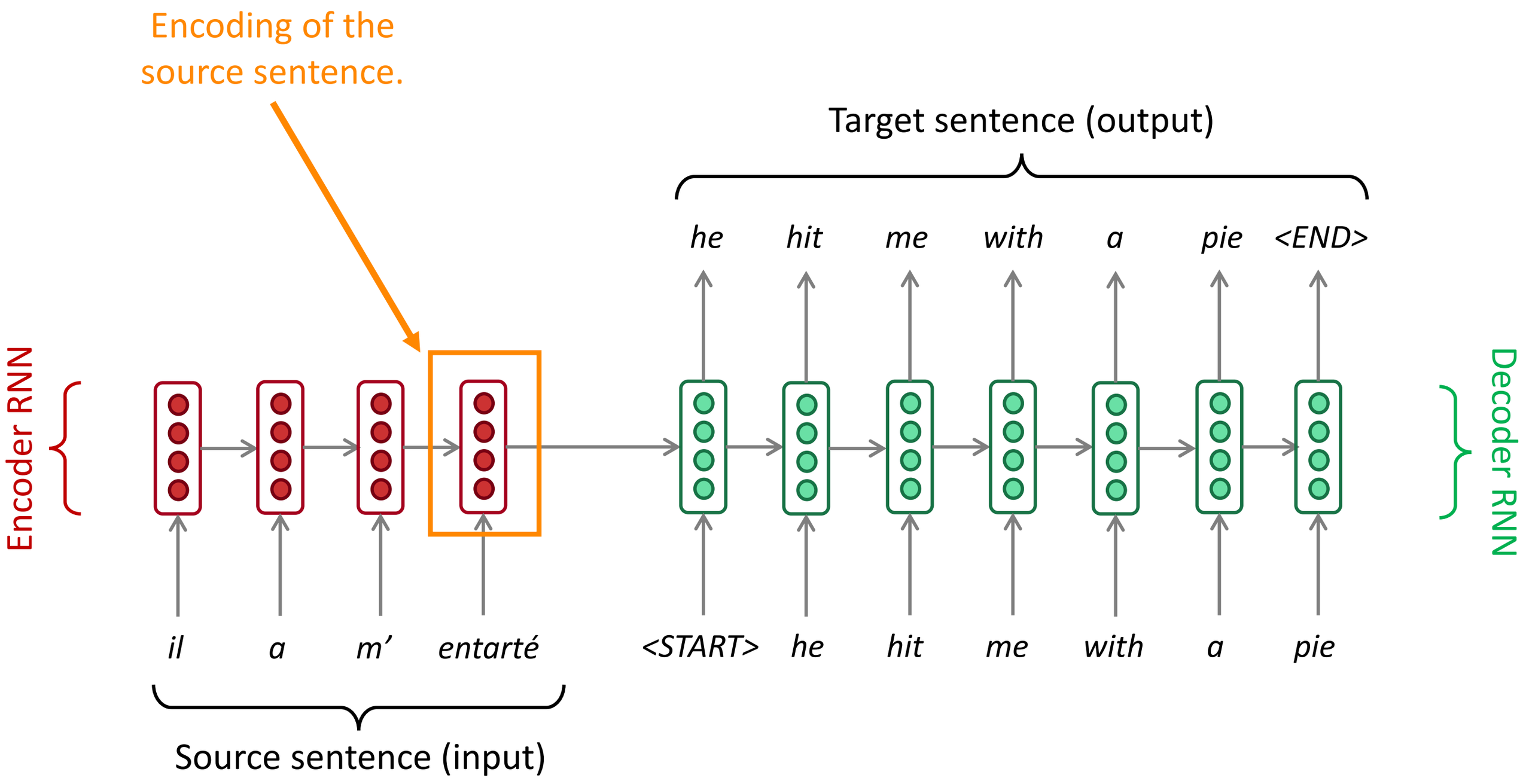
- NMT research has pioneered many of the recent innovations of NLP Deep Learning

- In **2019**: NMT research continues to thrive
  - Researchers have found *many, many* improvements to the "vanilla" seq2seq NMT system we've presented today
  - But one improvement is so integral that it is the new vanilla…

ATTENTION

Section 3: Attention

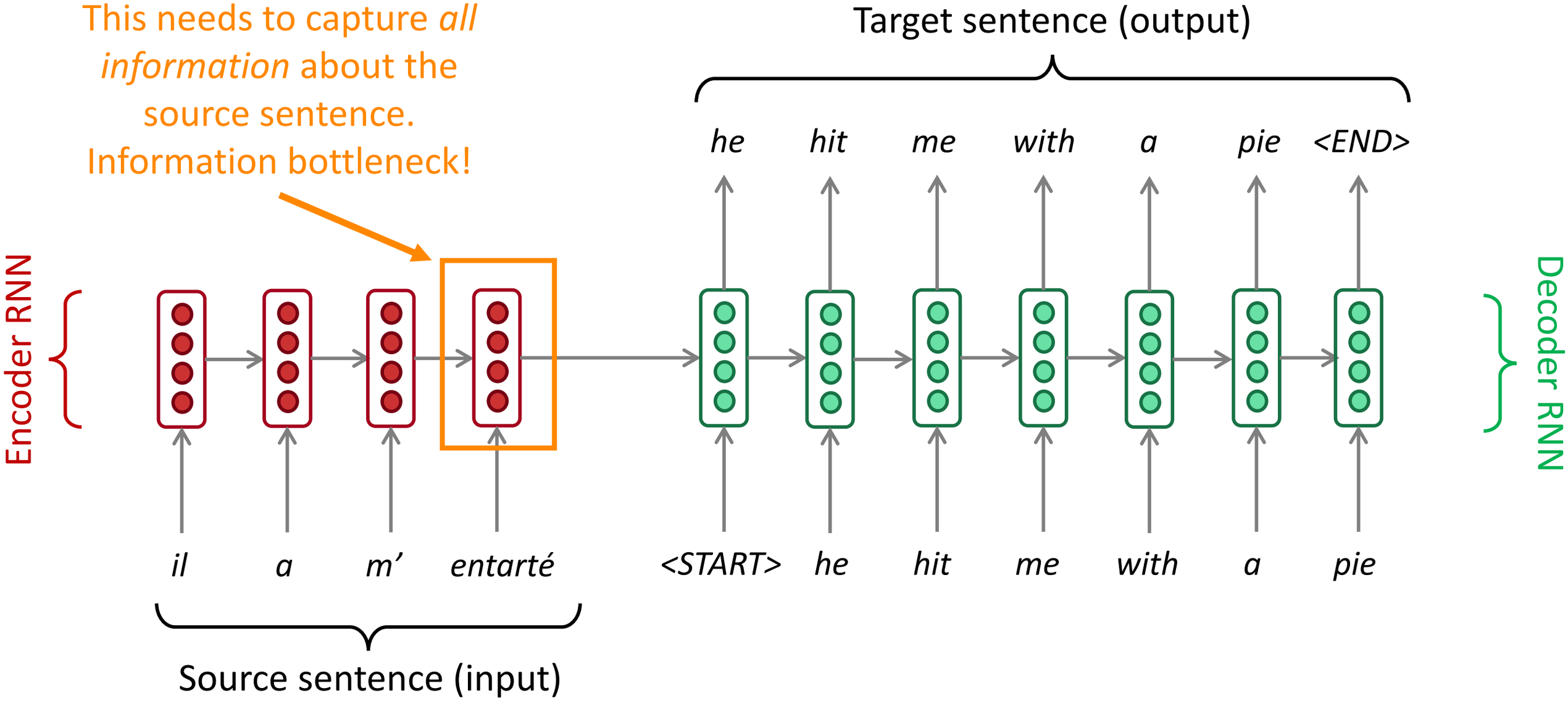
Sequence-to-sequence: the bottleneck problem



Problems with this architecture?

Sequence-to-sequence: the bottleneck problem

Encoding of the source sentence. This needs to capture *all information* about the source sentence. Information bottleneck!



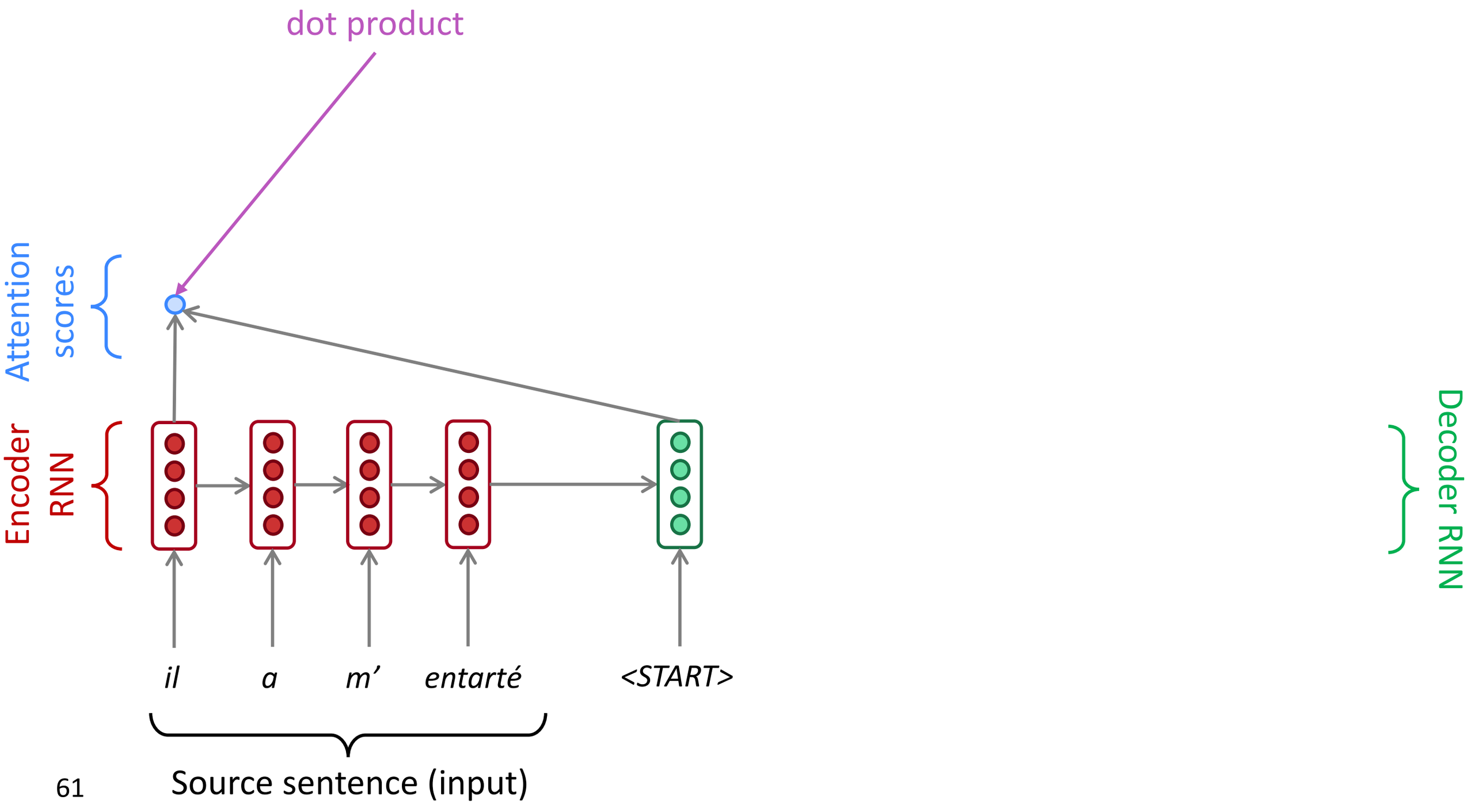
Attention

- **Attention** provides a solution to the bottleneck problem.

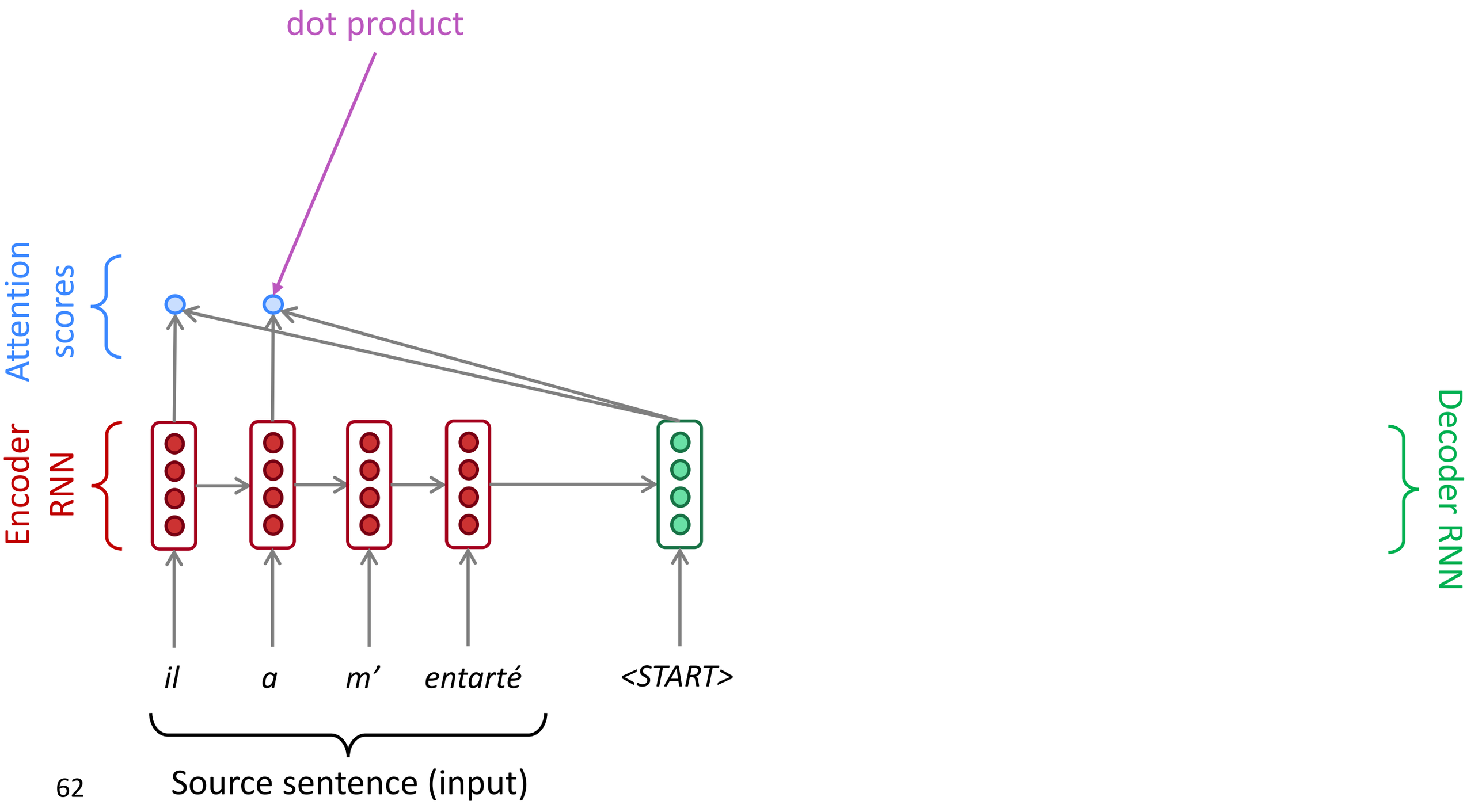
- Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence

- First we will show via diagram (no equations), then we will show with equations

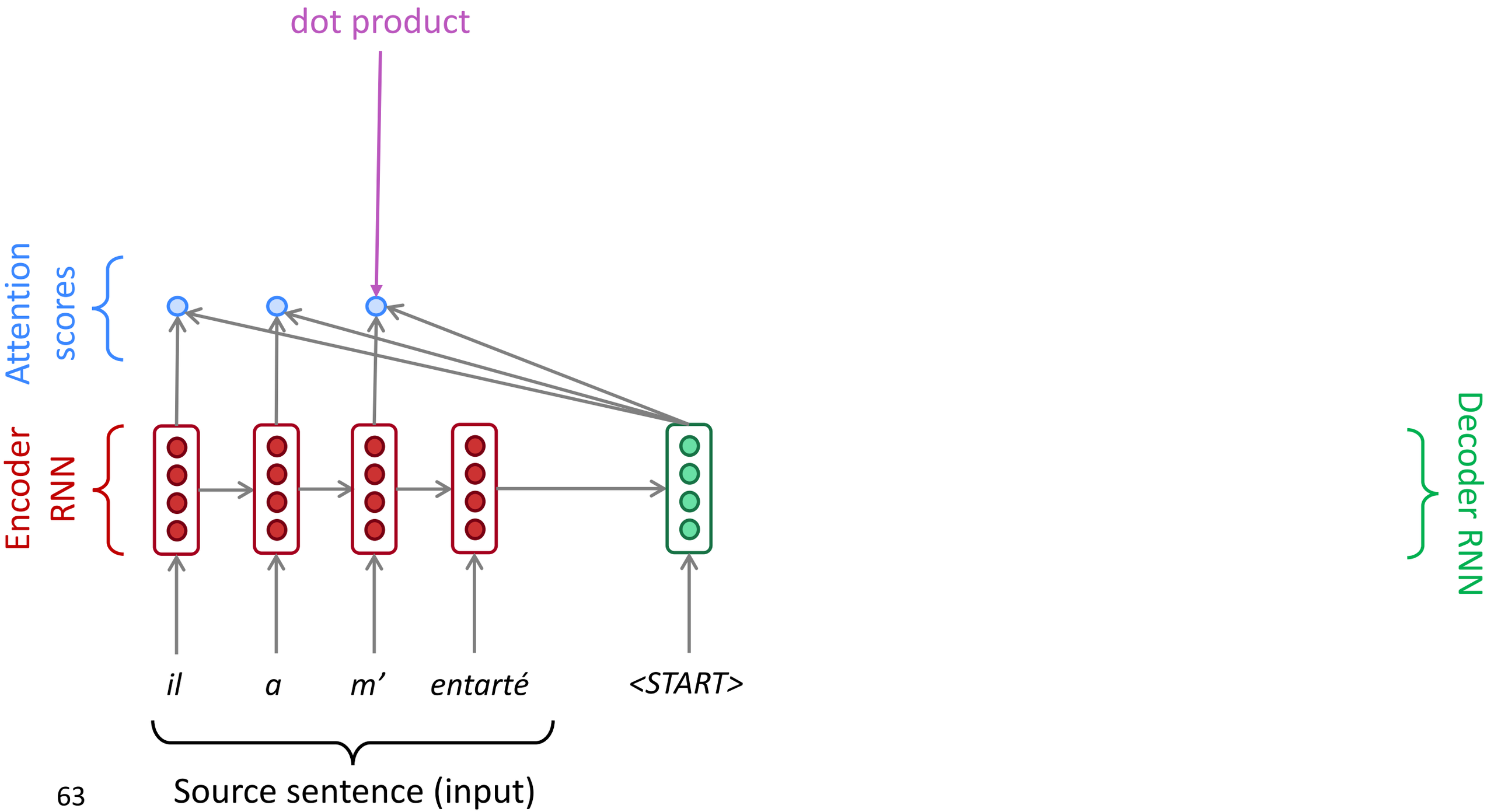
Sequence-to-sequence with attention



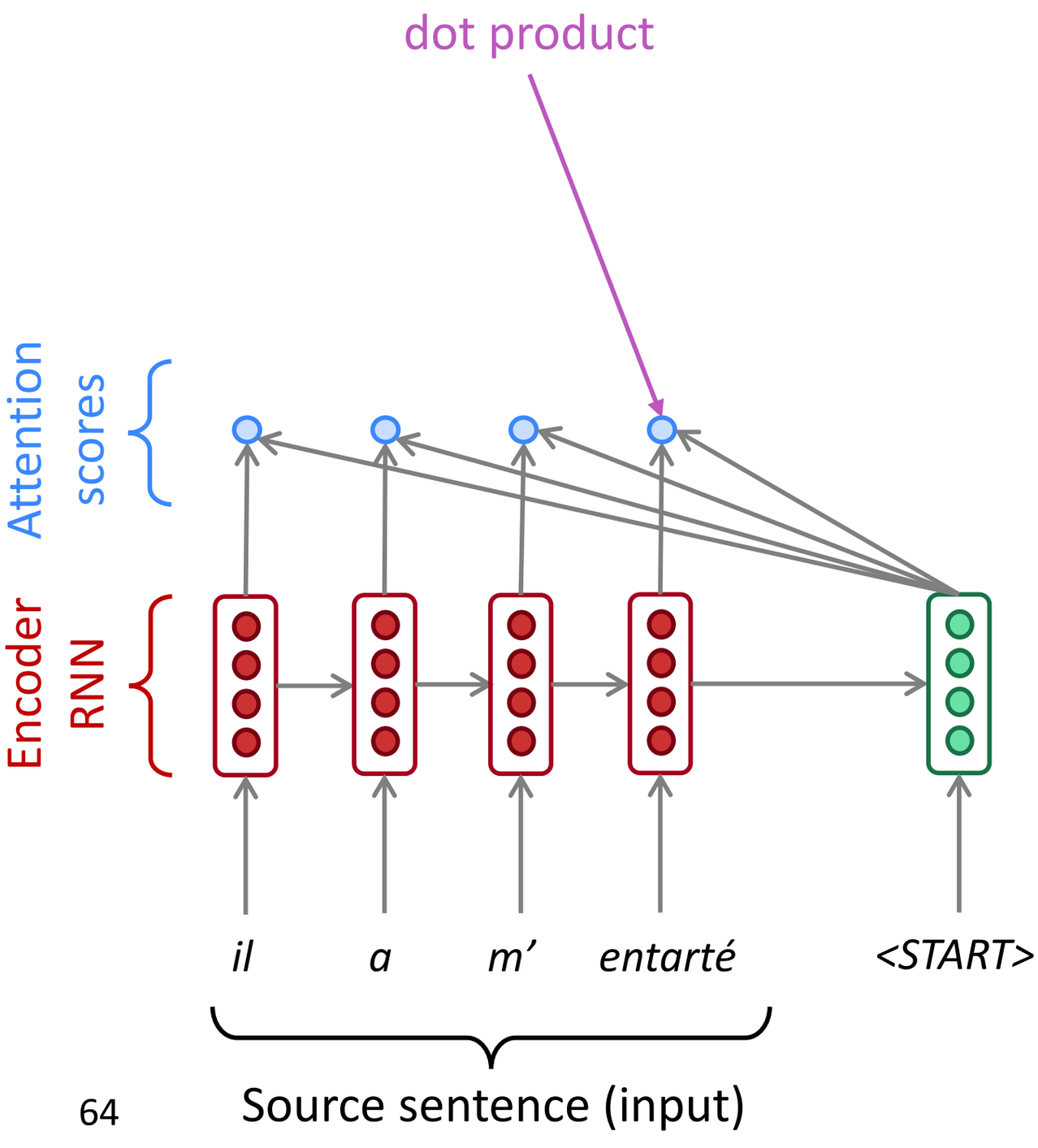
Sequence-to-sequence with attention



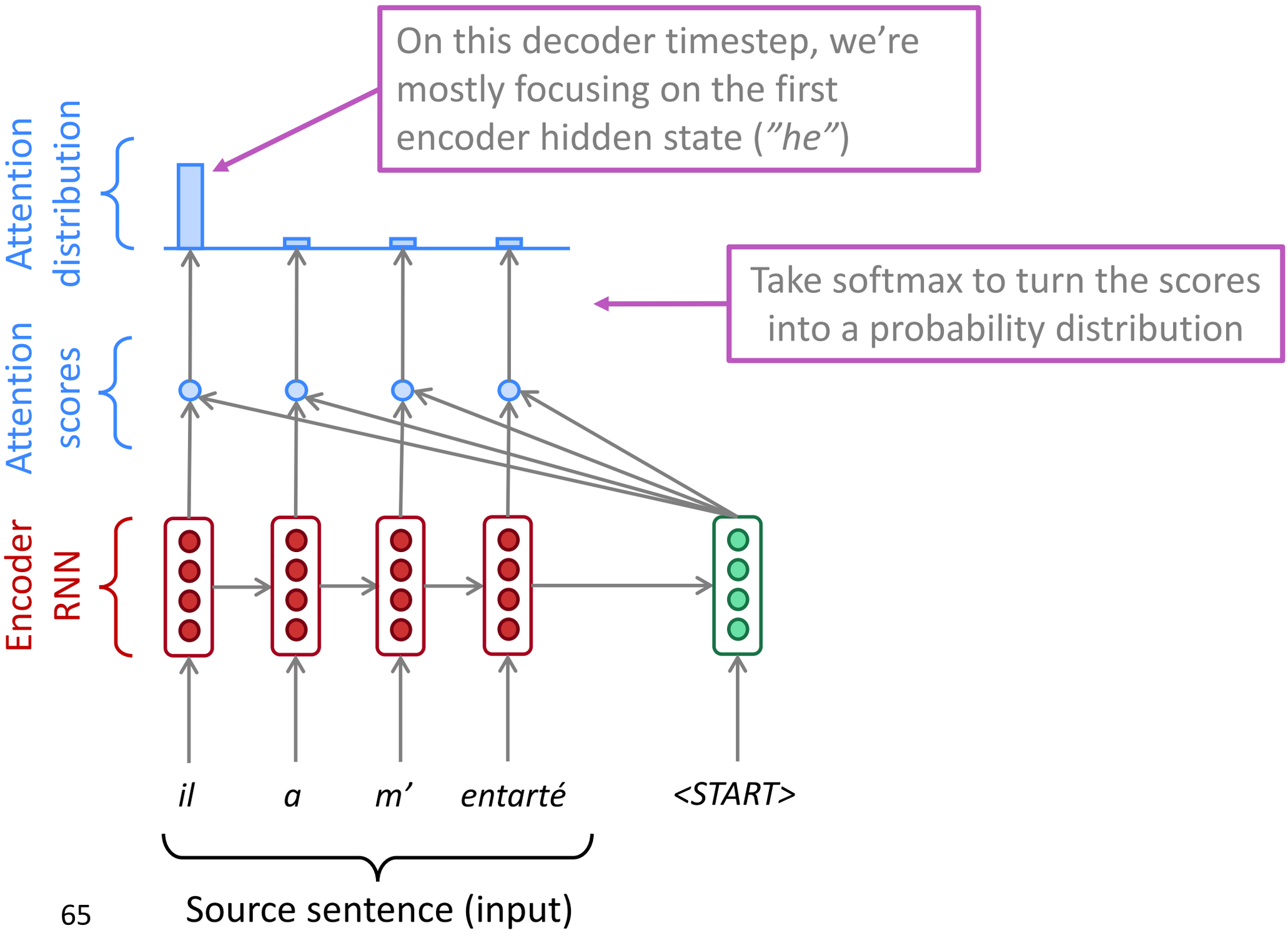
Sequence-to-sequence with attention



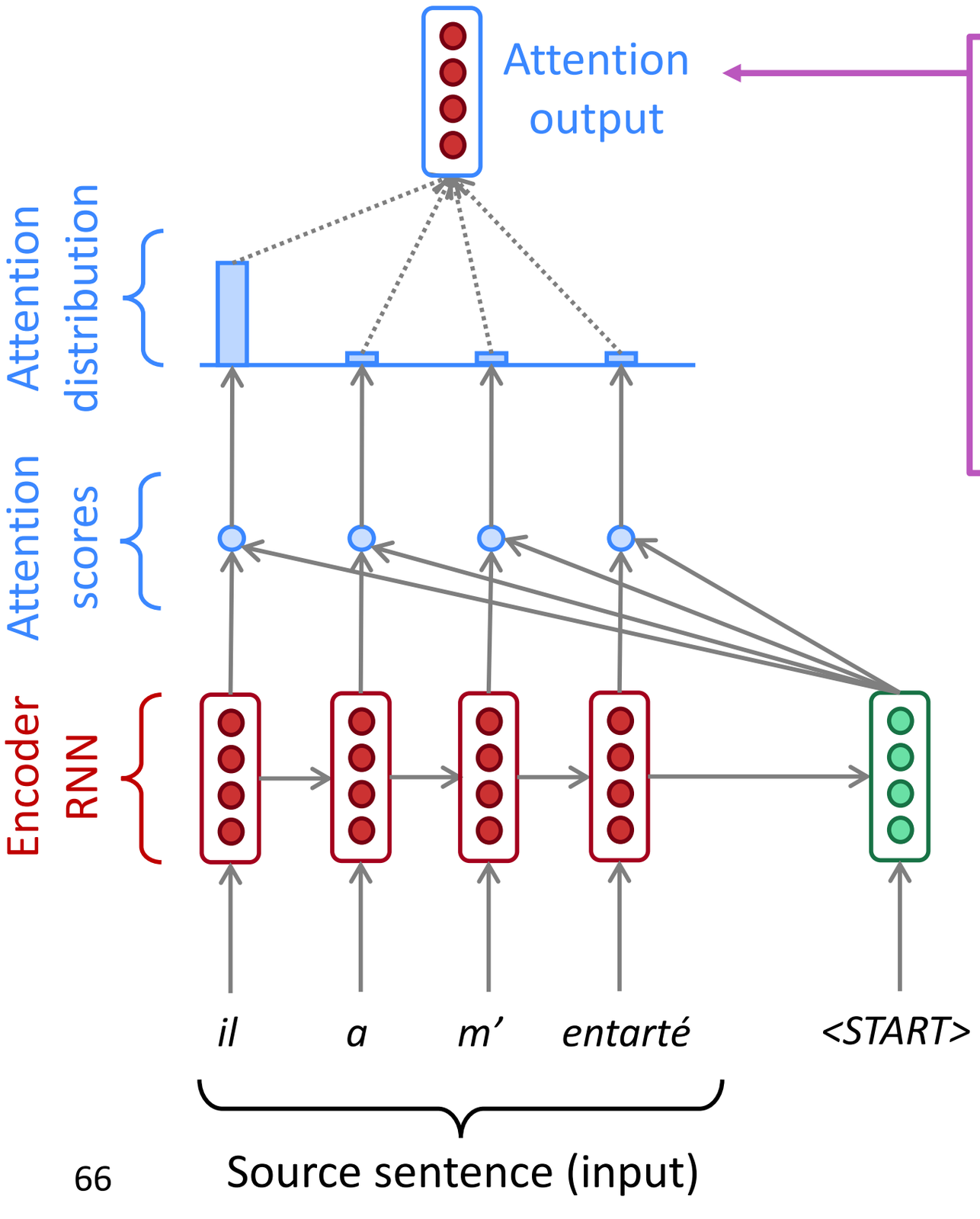
Sequence-to-sequence with attention



Sequence-to-sequence with attention



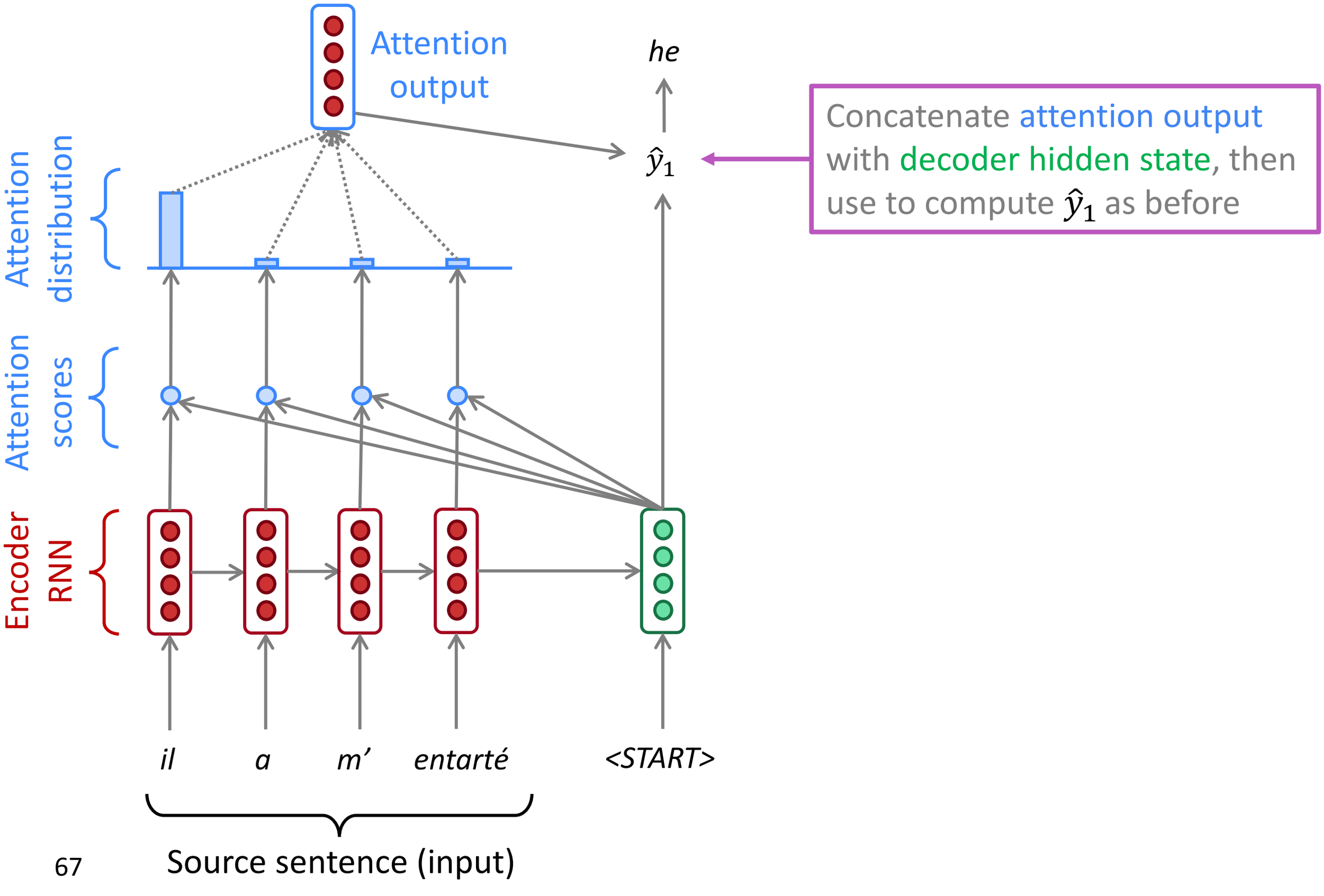
Sequence-to-sequence with attention



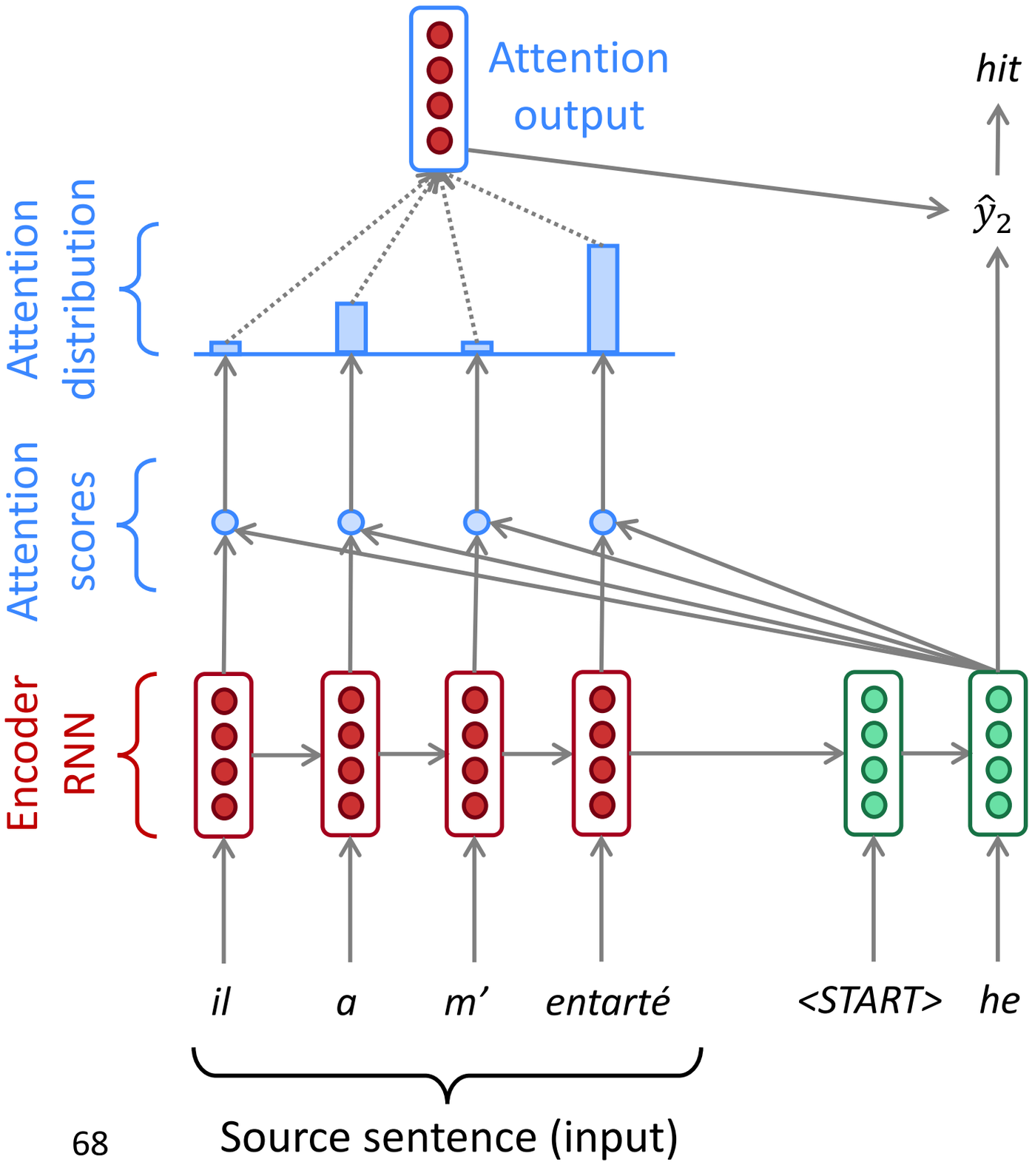
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

Sequence-to-sequence with attention

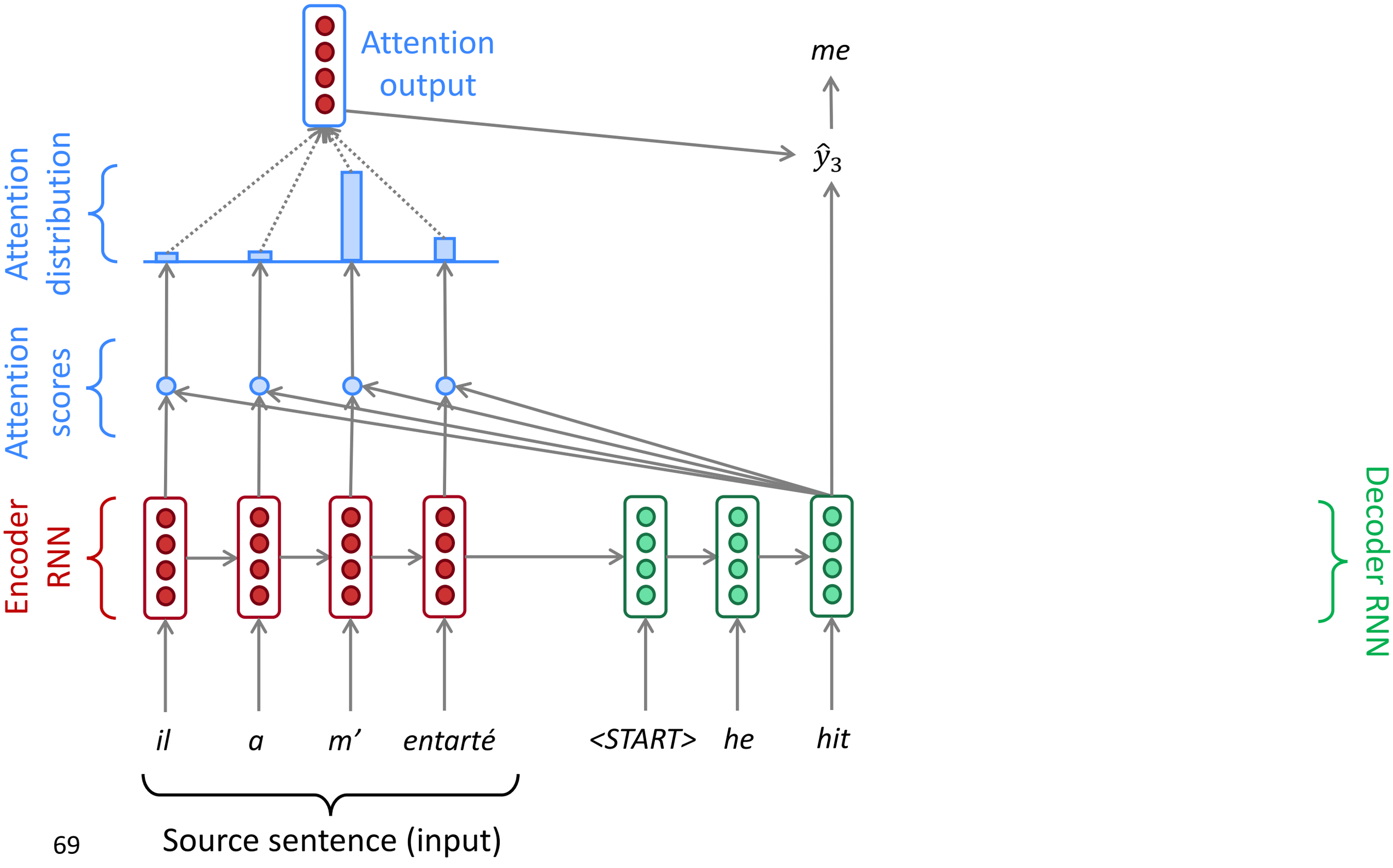


Sequence-to-sequence with attention

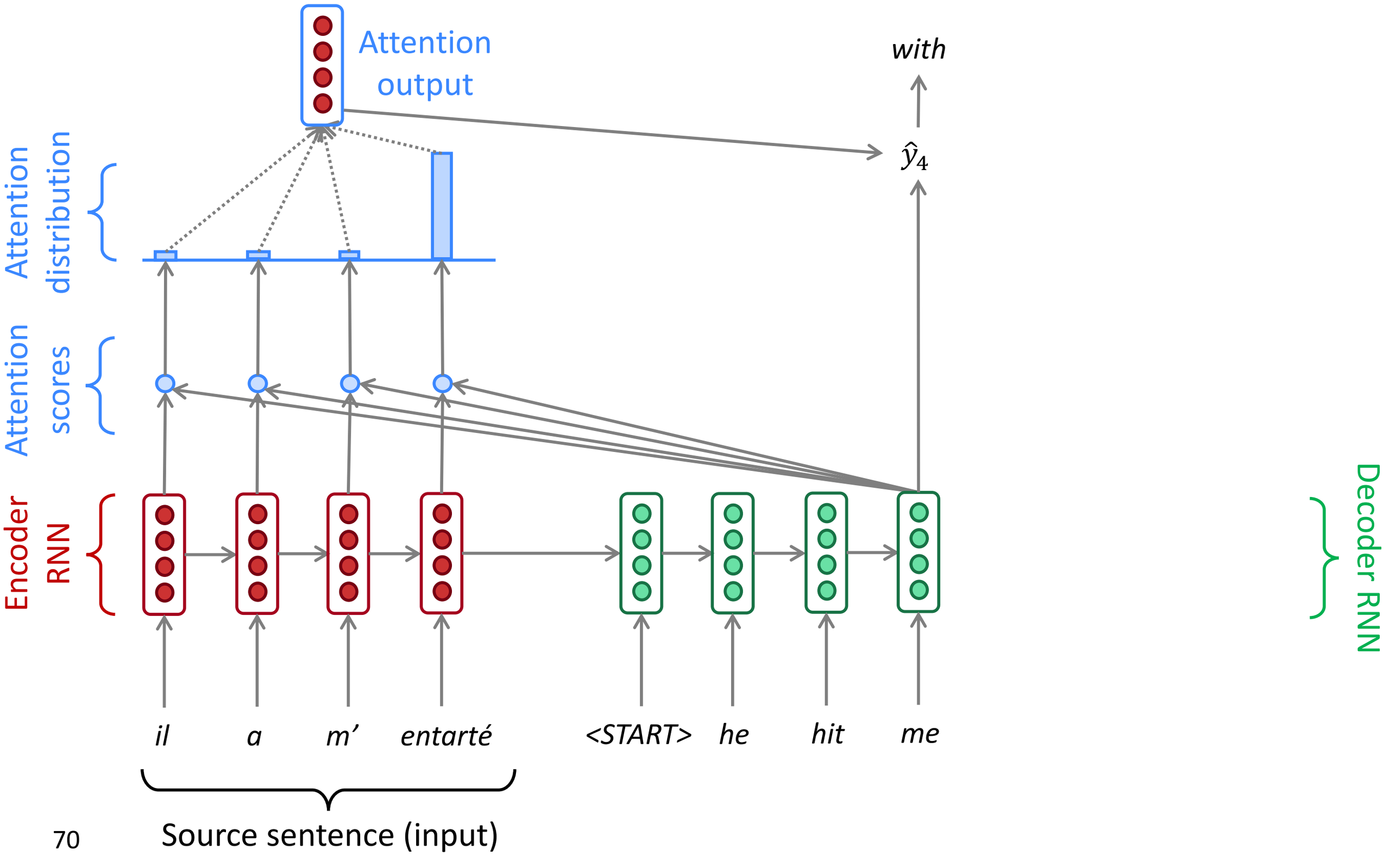


Sometimes we take the attention output from the previous step, and also feed it into the decoder (along with the usual decoder input).

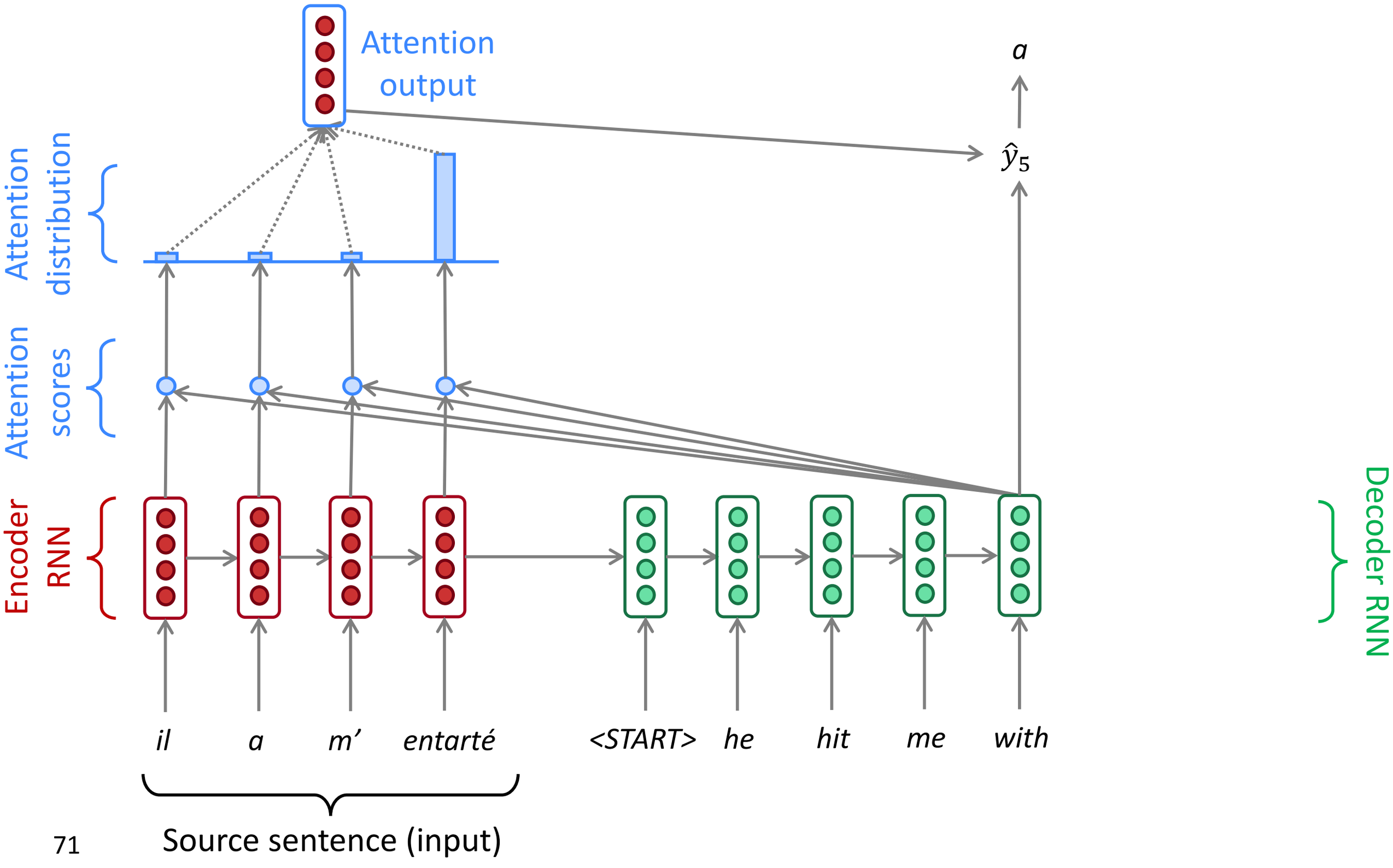
Sequence-to-sequence with attention



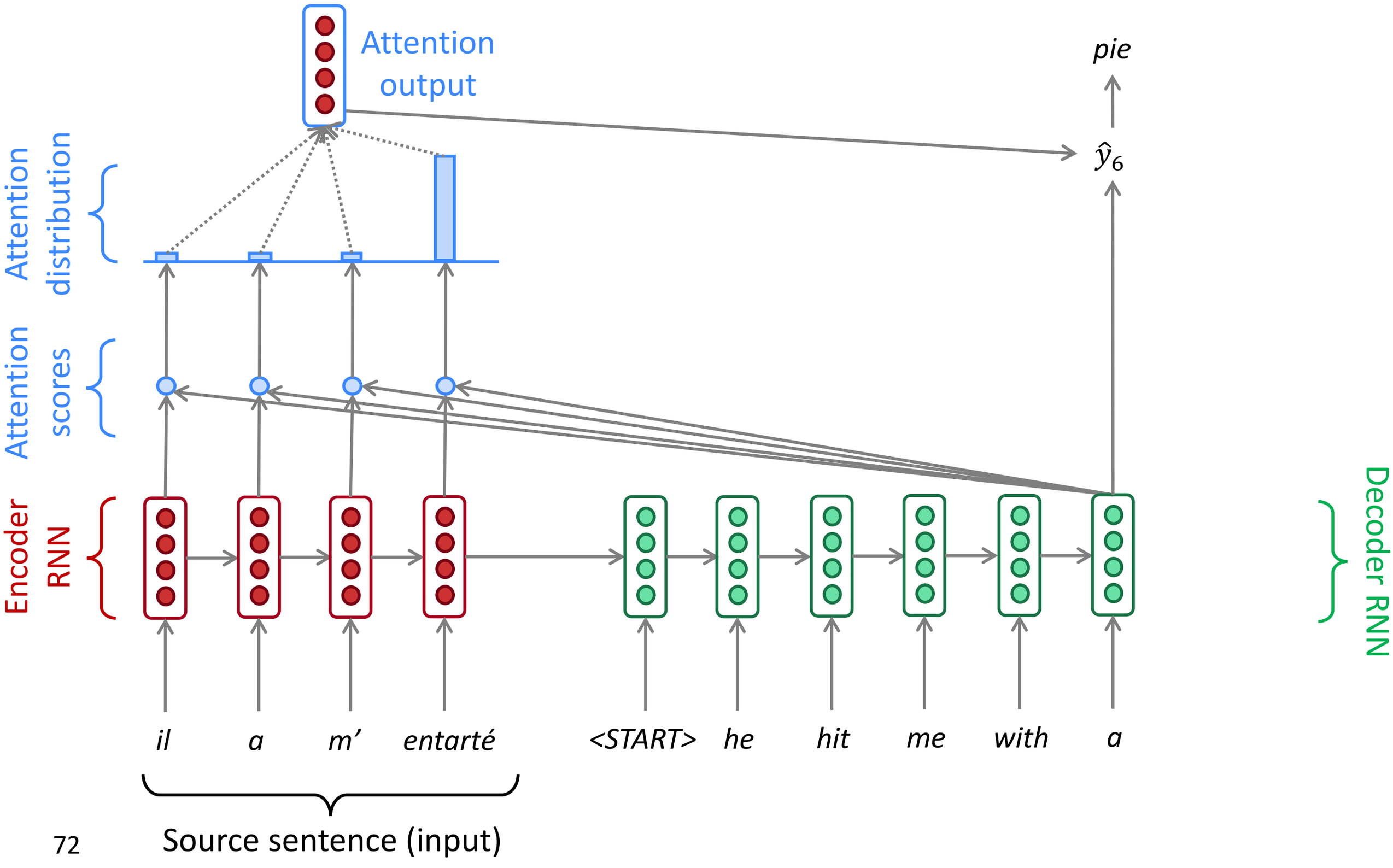
Sequence-to-sequence with attention



Sequence-to-sequence with attention



Sequence-to-sequence with attention



Attention: in equations

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

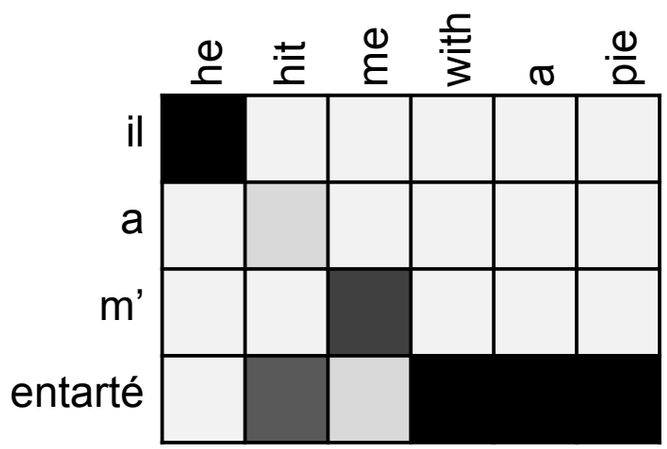
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Attention is great

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get (soft) alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



Attention is a *general* Deep Learning technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

- **More general definition of attention**:
  - Given a set of vector *values*, and a vector *query*, **attention** is a technique to compute a weighted sum of the values, dependent on the query.

- We sometimes say that the query *attends to* the values.
- For example, in the seq2seq + attention model, each decoder hidden state (query) *attends to* all the encoder hidden states (values).

Attention is a *general* Deep Learning technique

More general definition of attention:

Given a set of vector *values*, and a vector *query*, **attention** is a technique to compute a weighted sum of the values, dependent on the query.

Intuition:

- The weighted sum is a *selective summary* of the information contained in the values, where the query determines which values to focus on.
- Attention is a way to obtain a *fixed-size representation of an arbitrary set of representations* (the values), dependent on some other representation (the query).

There are *several* attention variants

- We have some *values* $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

- Attention always involves:
  1. Computing the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$
  2. Taking softmax to get *attention distribution* α:

$$\alpha = \text{softmax}(\boldsymbol{e}) \in \mathbb{R}^N$$

There are multiple ways to do this

  3. Using attention distribution to take weighted sum of values:

$$\boldsymbol{a} = \sum_{i=1}^{N} \alpha_i \boldsymbol{h}_i \in \mathbb{R}^{d_1}$$

thus obtaining the *attention output* $\boldsymbol{a}$ (sometimes called the *context vector*)

Attention variants

There are several ways you can compute $\boldsymbol{e} \in \mathbb{R}^N$ from $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and $\boldsymbol{s} \in \mathbb{R}^{d_2}$:

- Basic dot-product attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
  - Note: this assumes $d_1 = d_2$
  - This is the version we saw earlier

- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
  - Where $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix

- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$
  - Where $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}, \boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $\boldsymbol{v} \in \mathbb{R}^{d_3}$ is a weight vector.
  - d_3 (the attention dimensionality) is a hyperparameter

More information:

"Deep Learning for NLP Best Practices", Ruder, 2017. http://ruder.io/deep-learning-nlp-best-practices/index.html#attention

"Massive Exploration of Neural Machine Translation Architectures", Britz et al, 2017, https://arxiv.org/pdf/1703.03906.pdf

now you may think…

- is attention perhaps.. all I need?

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*]
illia.polosukhin@gmail.com

the remainder

- with this in hand, the next developments came quickly…

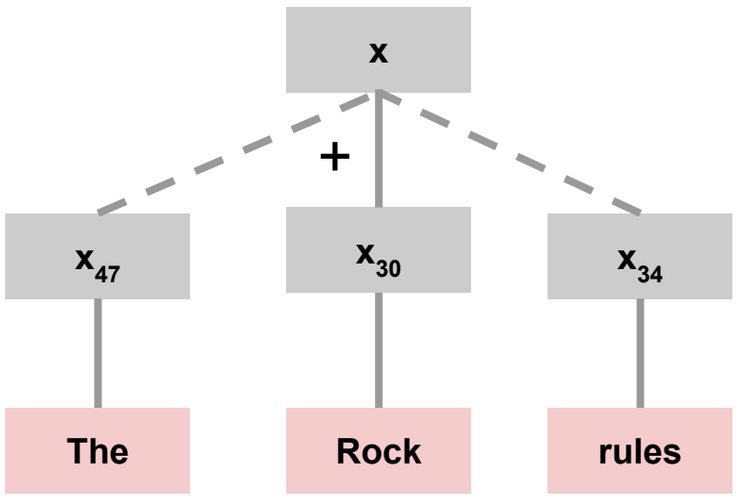
Overview

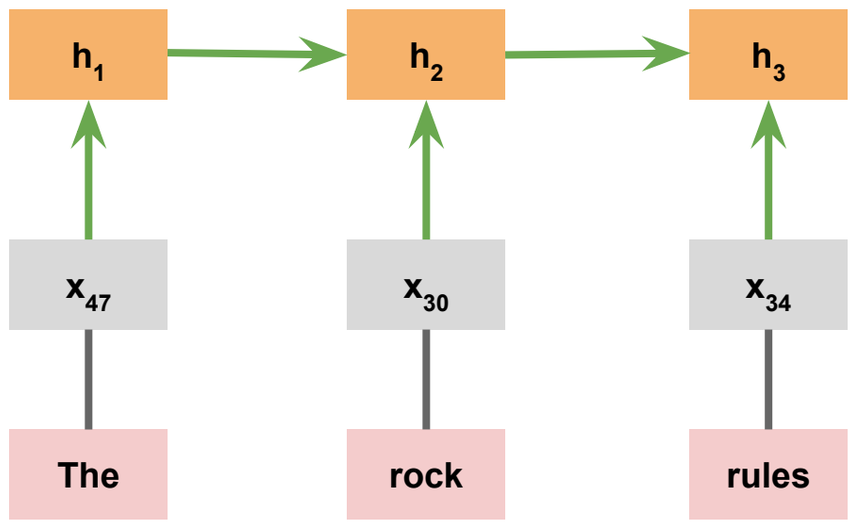
1. Overview: Resources and guiding insights
2. ELMo: **E**mbeddings from Language **Mo**dels
3. Transformers
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

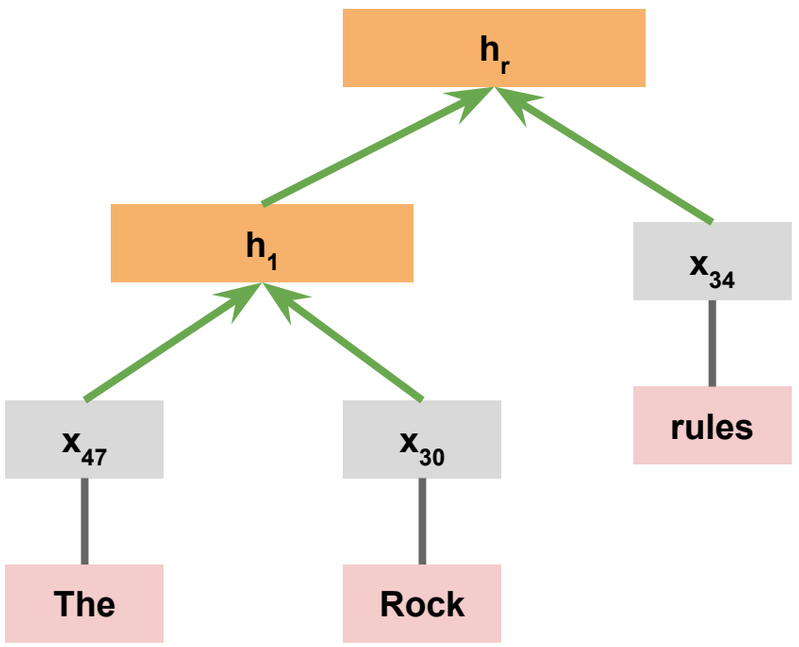
Associated materials

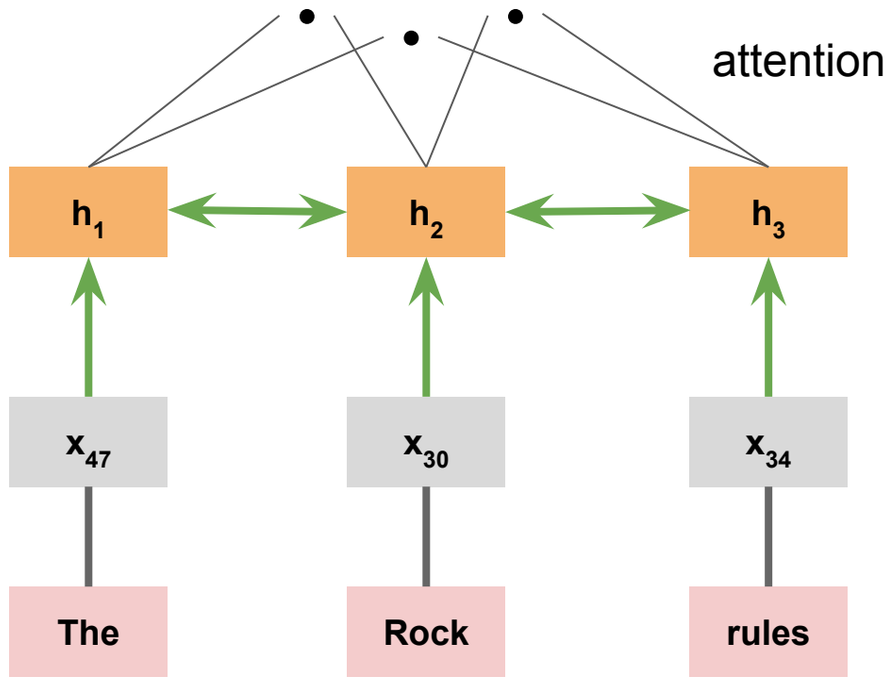
4. ELMo:
   - Peters et al. 2018
   - Project site: https://allennlp.org/elmo
5. Transformer
   - Vaswani et al. 2017
   - Alexander Rush: The Annotated Transformer [link]
6. BERT
   - Devlin et al. 2019
   - Project site: https://github.com/google-research/bert
   - bert-as-service [link]

Model structure and linguistic structure

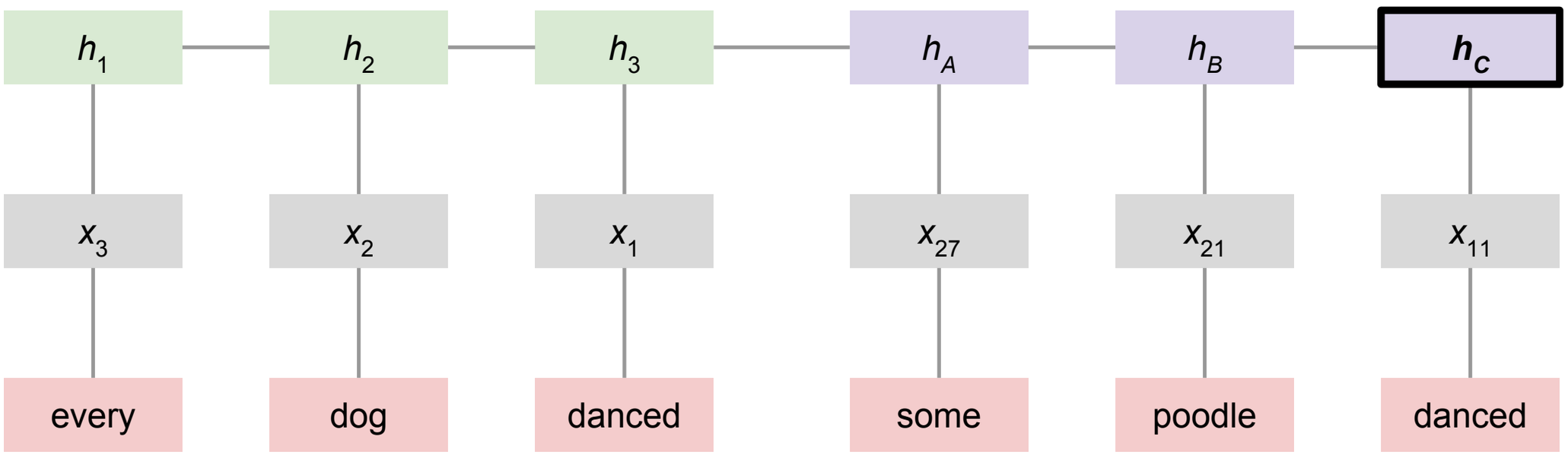






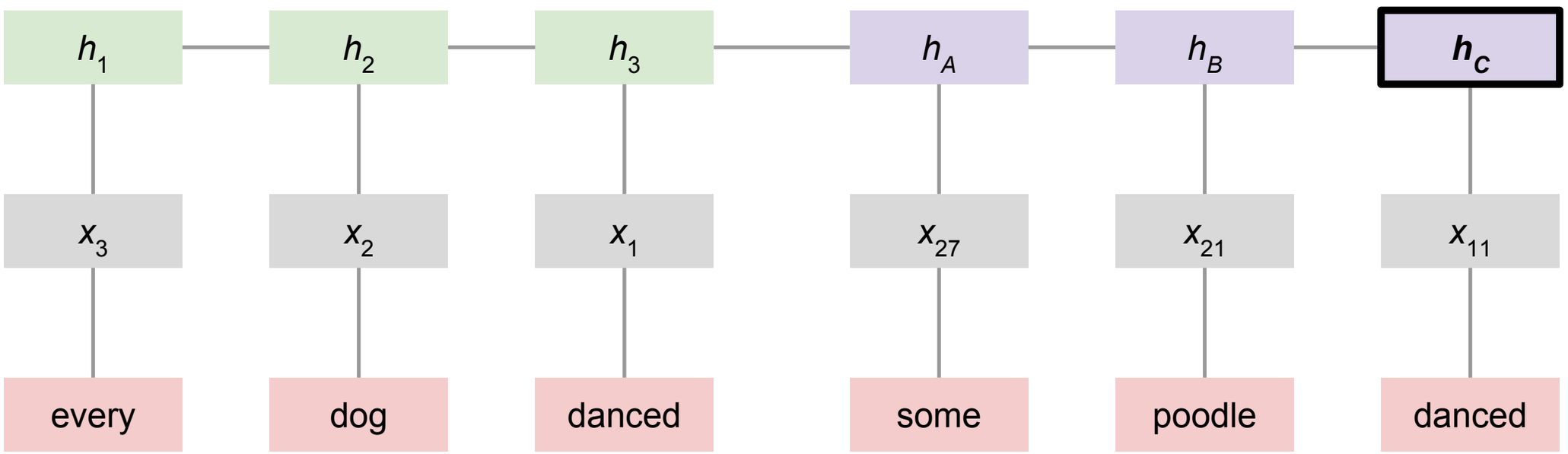


Guiding idea: Attention (from the NLI slides)



Guiding idea: Attention (from the NLI slides)

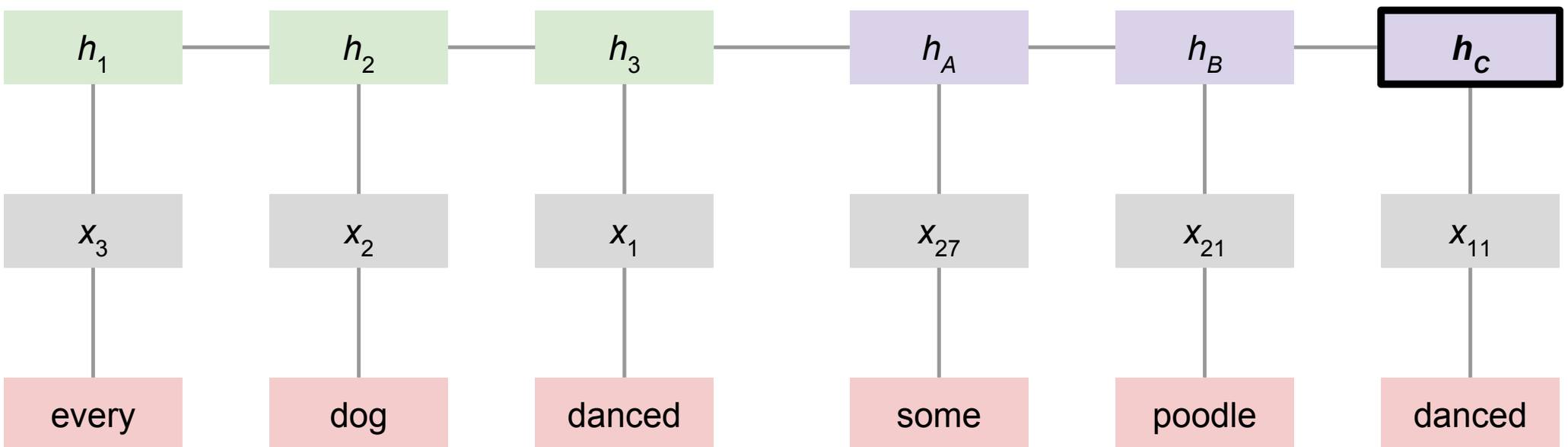
scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



Guiding idea: Attention (from the NLI slides)

attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$

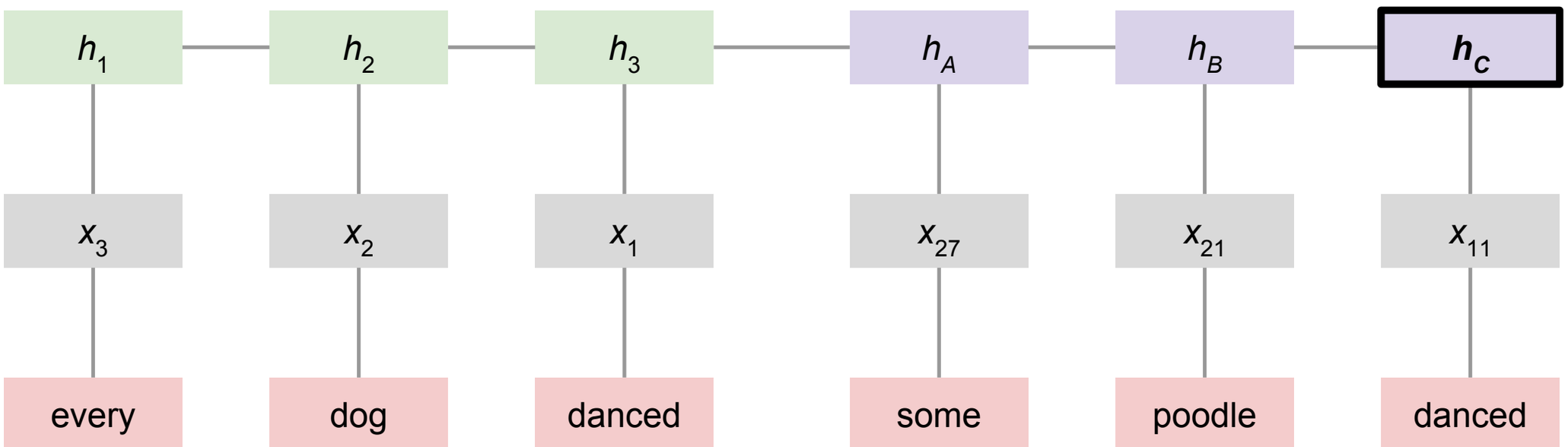


Guiding idea: Attention (from the NLI slides)

context $\kappa = \mathbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$

attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



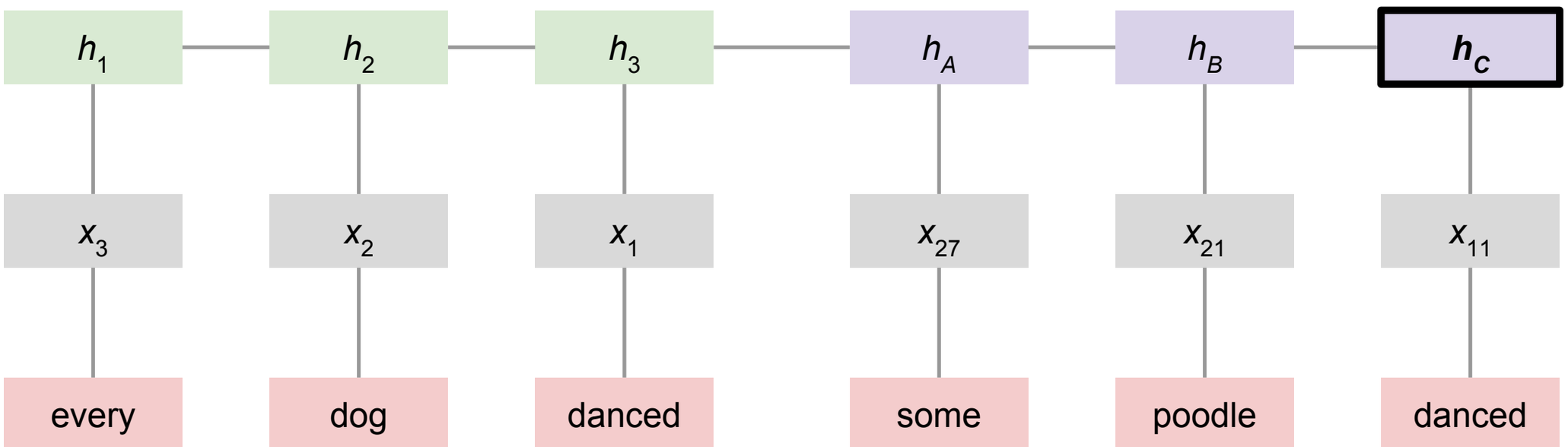
Guiding idea: Attention (from the NLI slides)

attention combo $\tilde{h} = \tanh([\kappa; h_C]W_\kappa)$

context $\kappa = \mathbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$

attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



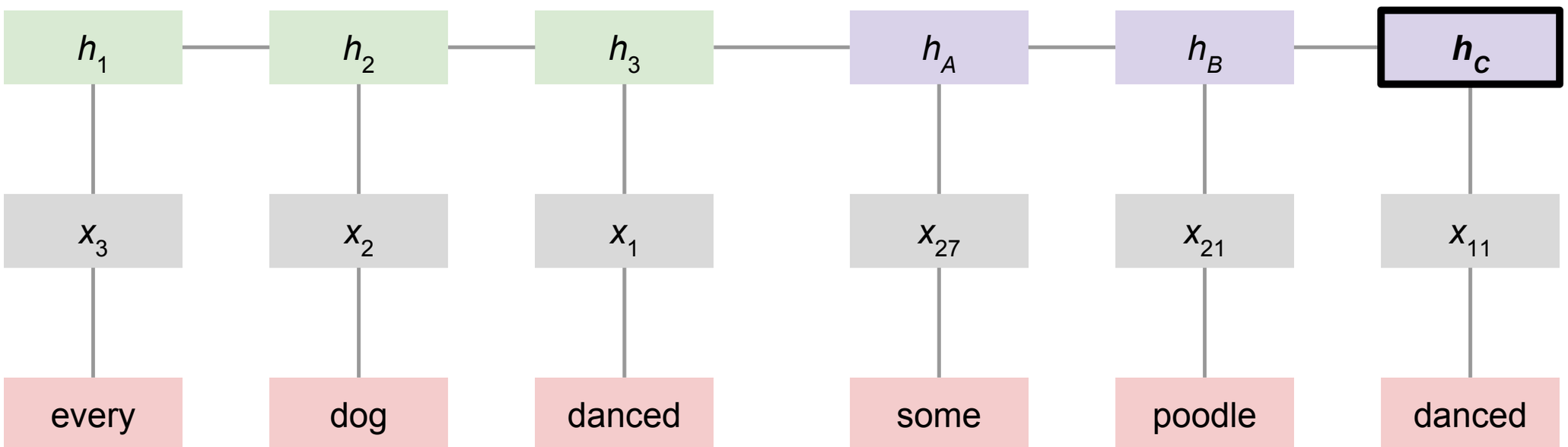
Guiding idea: Attention (from the NLI slides)

attention combo $\tilde{h} = \tanh([\kappa; h_C]W_\kappa)$ or $\tilde{h} = \tanh(\kappa W_\kappa + h_C W_h)$

context $\kappa = \mathbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$

attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



Guiding idea: Attention (from the NLI slides)

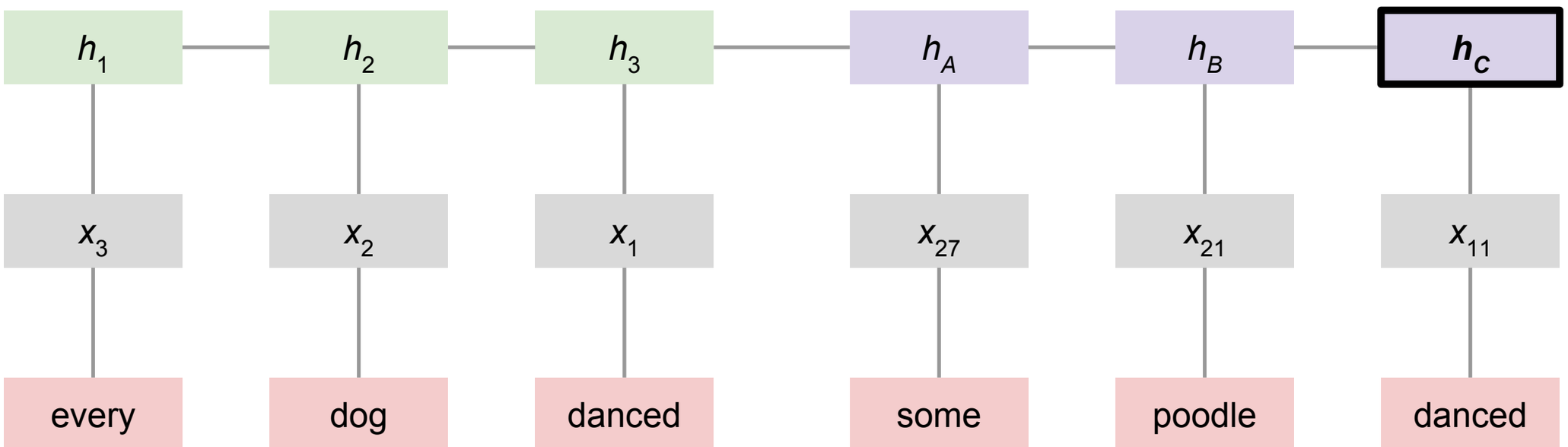
classifier $$y = \textbf{softmax}(\tilde{h}W + b)$$

attention combo $$\tilde{h} = \tanh([\kappa; h_C]W_\kappa)$$

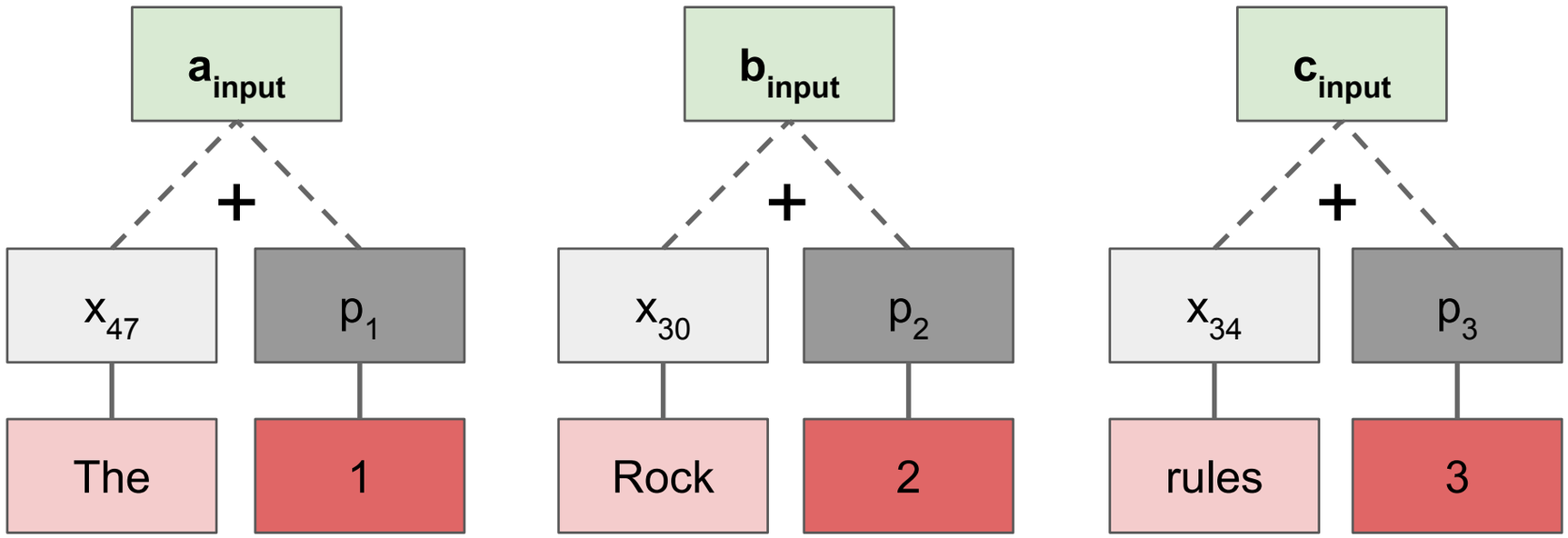
context $$\kappa = \textbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$$

attention weights $$\alpha = \textbf{softmax}(\tilde{\alpha})$$

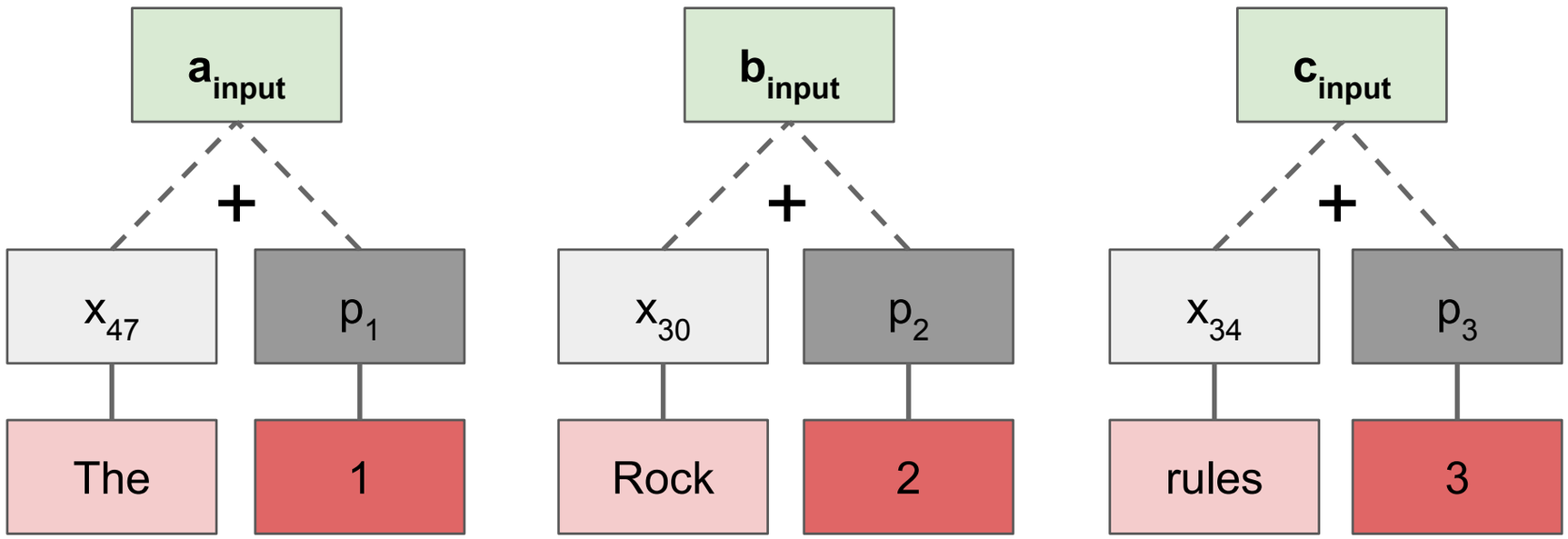
scores $$\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$$

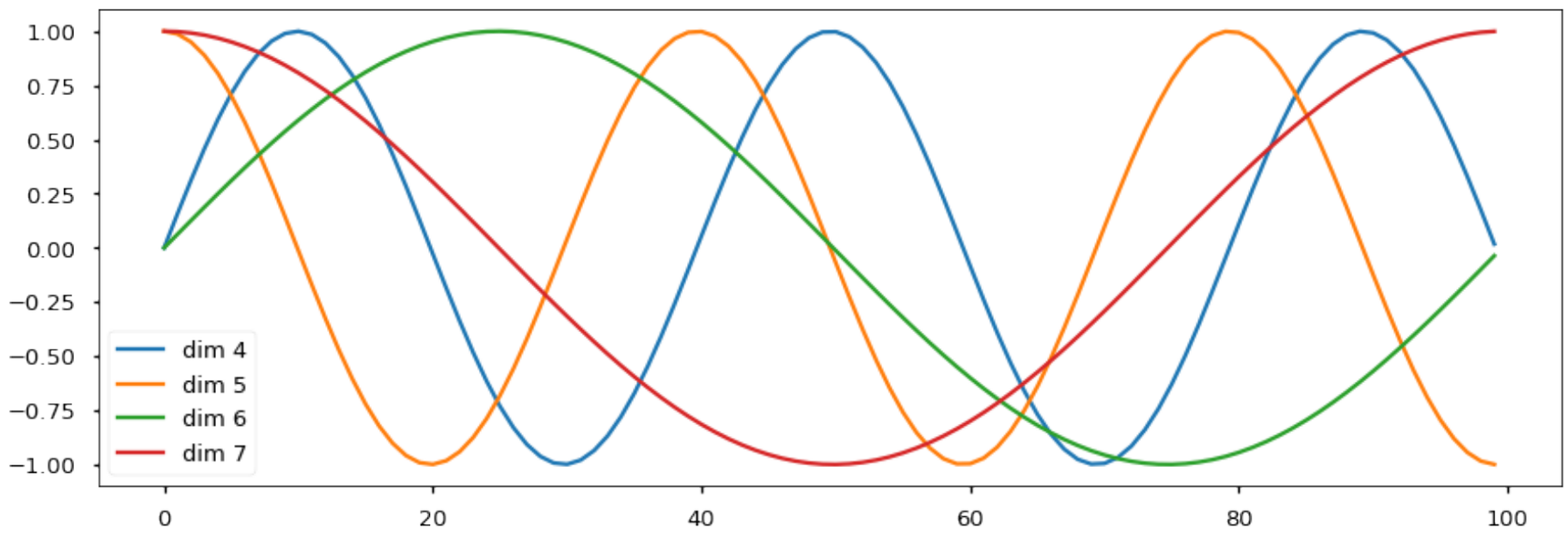


Guiding idea: Positional encoding



Guiding idea: Positional encoding





From ‘The Annotated Transformer’

ELMo

1. Overview: Resources and guiding insights

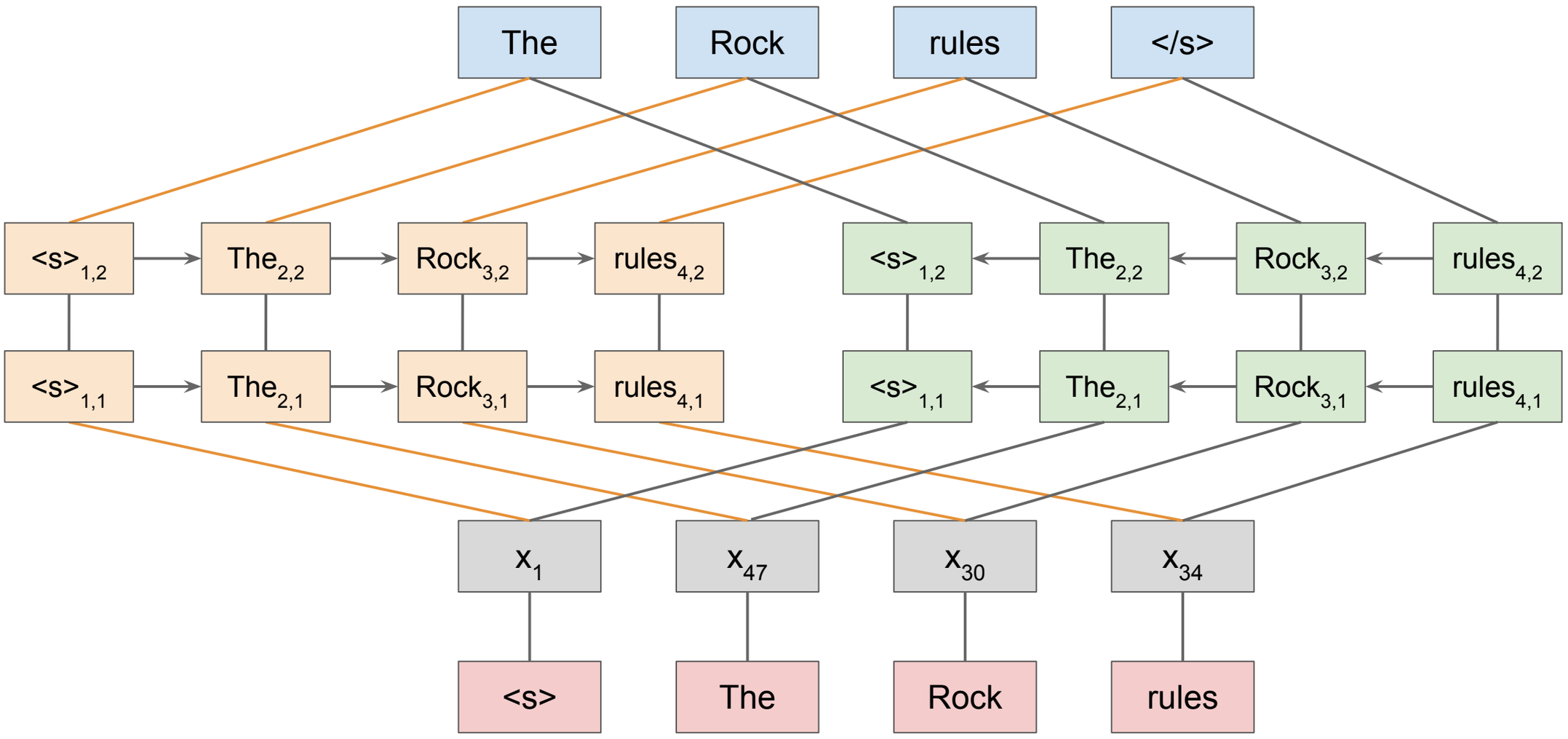
2. ELMo: **E**mbeddings from Language **Mo**dels

3. Transformers

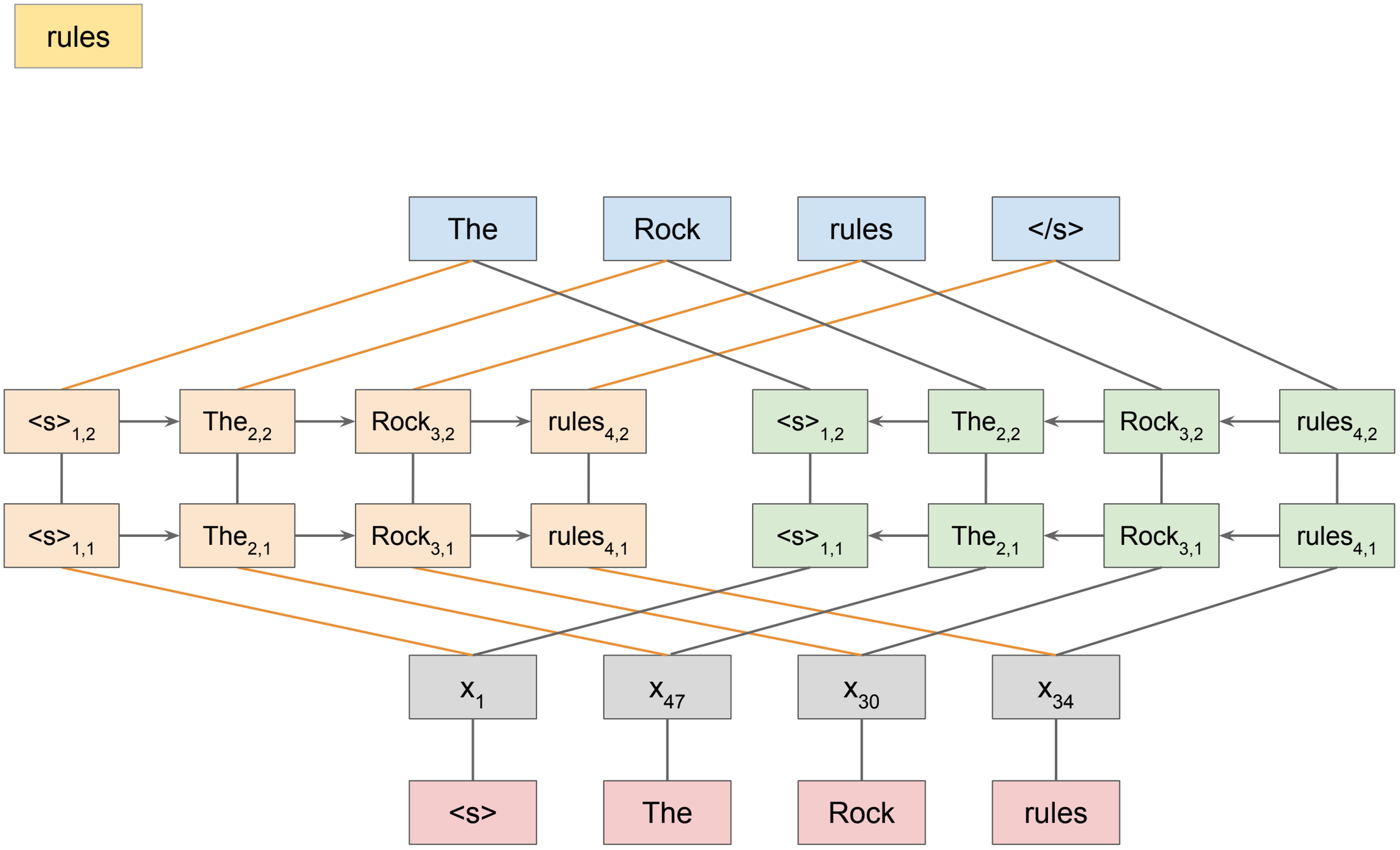
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

5. `contextualreps.ipynb`: Easy ways to bring ELMo and BERT into your project

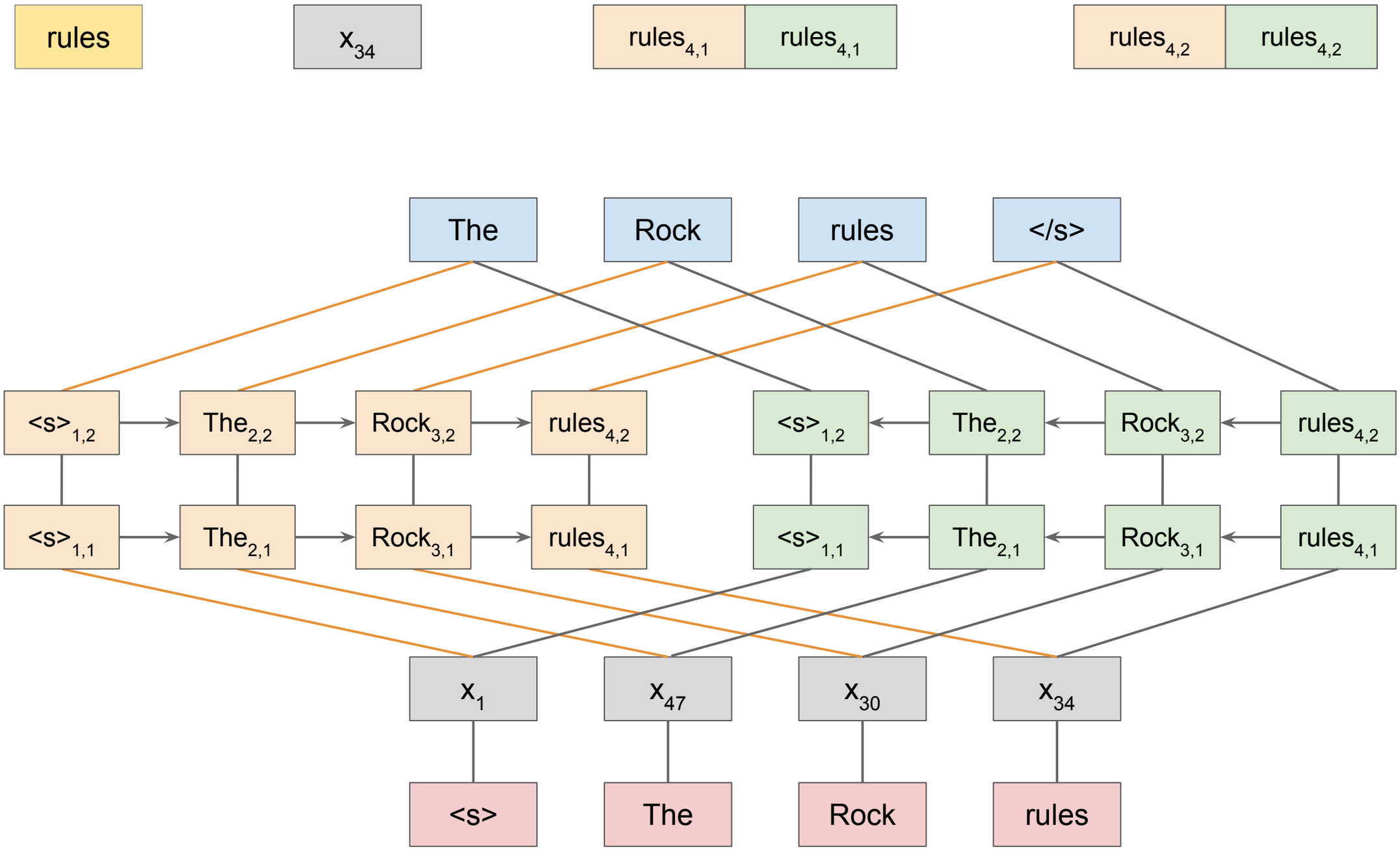
Core model structure



Core model structure

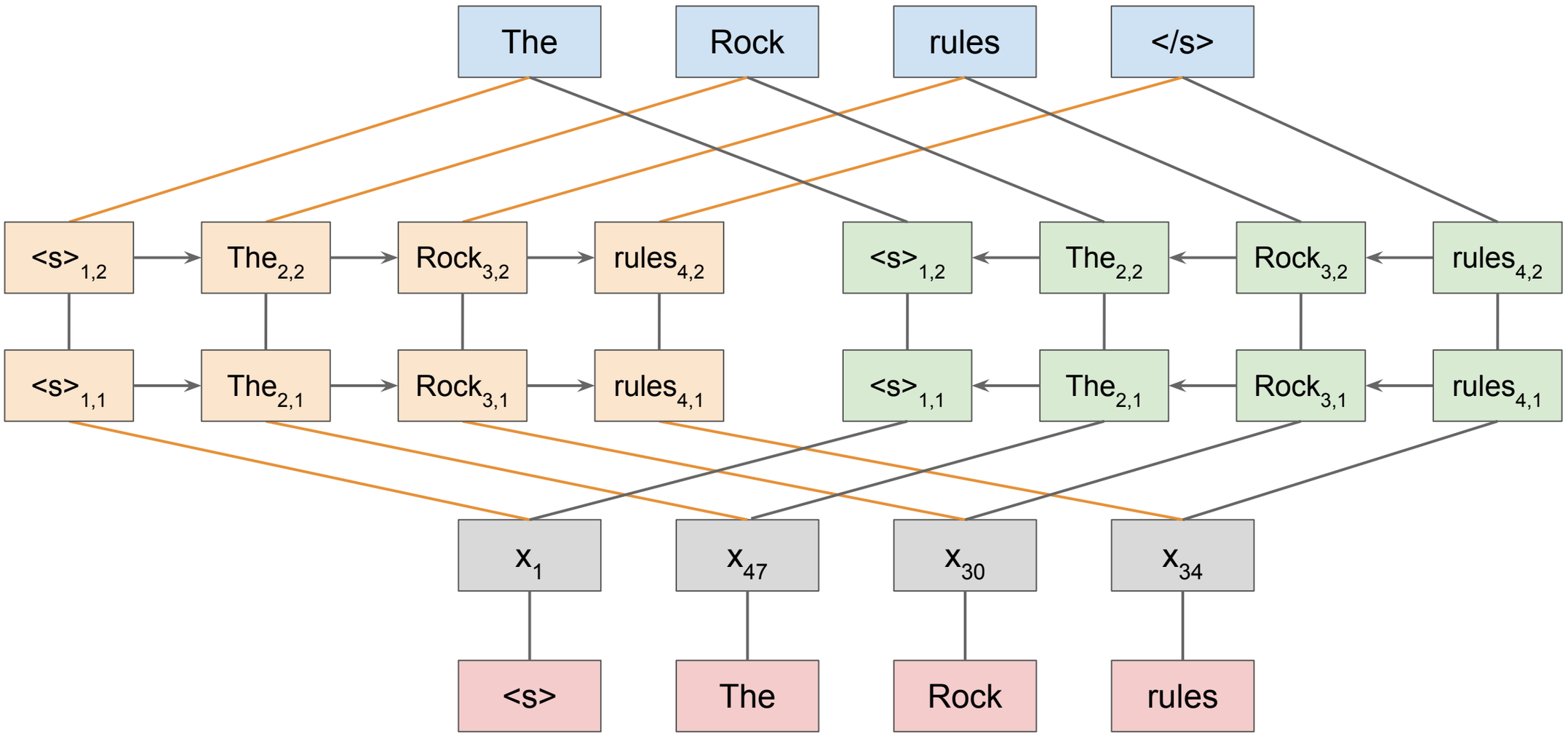


Core model structure



Core model structure

$$\text{rules} = s_0^{\text{task}} \cdot x_{34} + s_1^{\text{task}} \cdot [\text{rules}_{4,1}; \text{rules}_{4,1}] + s_2^{\text{task}} \cdot [\text{rules}_{4,2}; \text{rules}_{4,2}]$$



ELMo model releases

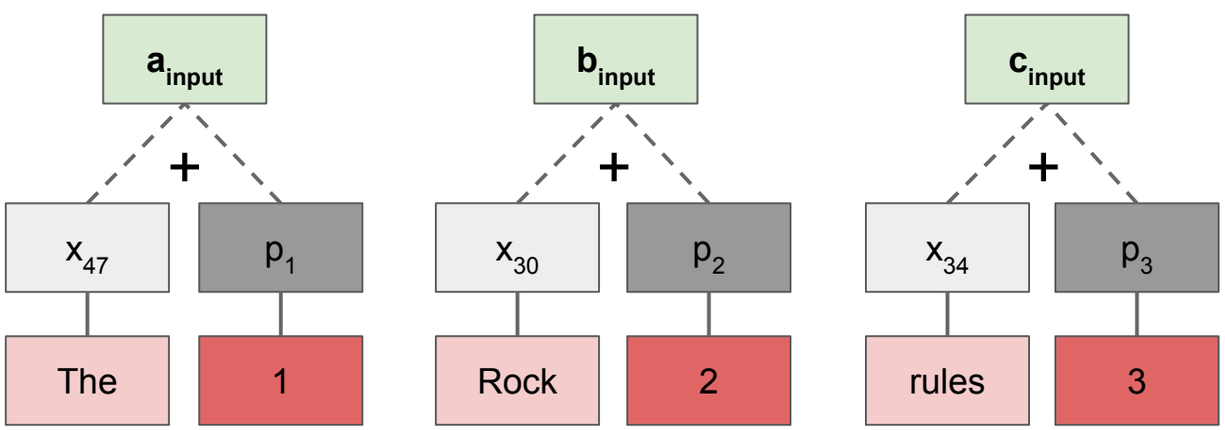
Model	Parameters	LSTM Hidden size	LSTM Output size	LSTM Highway layers
Small	13.6M	1024	128	1
Medium	28.0M	2048	256	1
Original	93.6M	4096	512	2
Original (5.5B)	93.6M	4096	512	2

Additional details at https://allennlp.org/elmo; the options files reveal additional information about the subword convolutional filters, activation functions, thresholds, and layer dimensions.

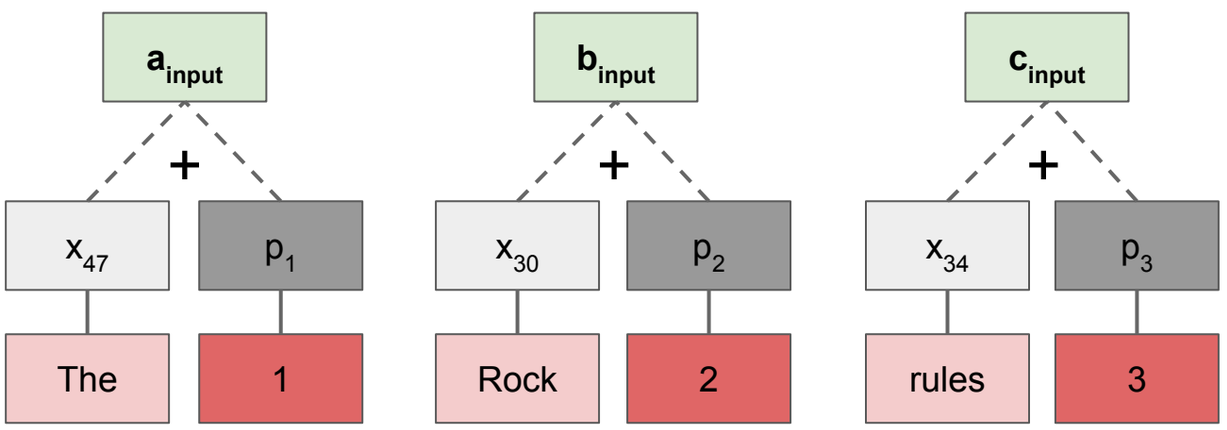
Transformers

1. Overview: Resources and guiding insights
2. ELMo: **E**mbeddings from Language **Mo**dels
3. Transformers
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
5. `contextualreps.ipynb`: Easy ways to bring ELMo and BERT into your project

Core model structure

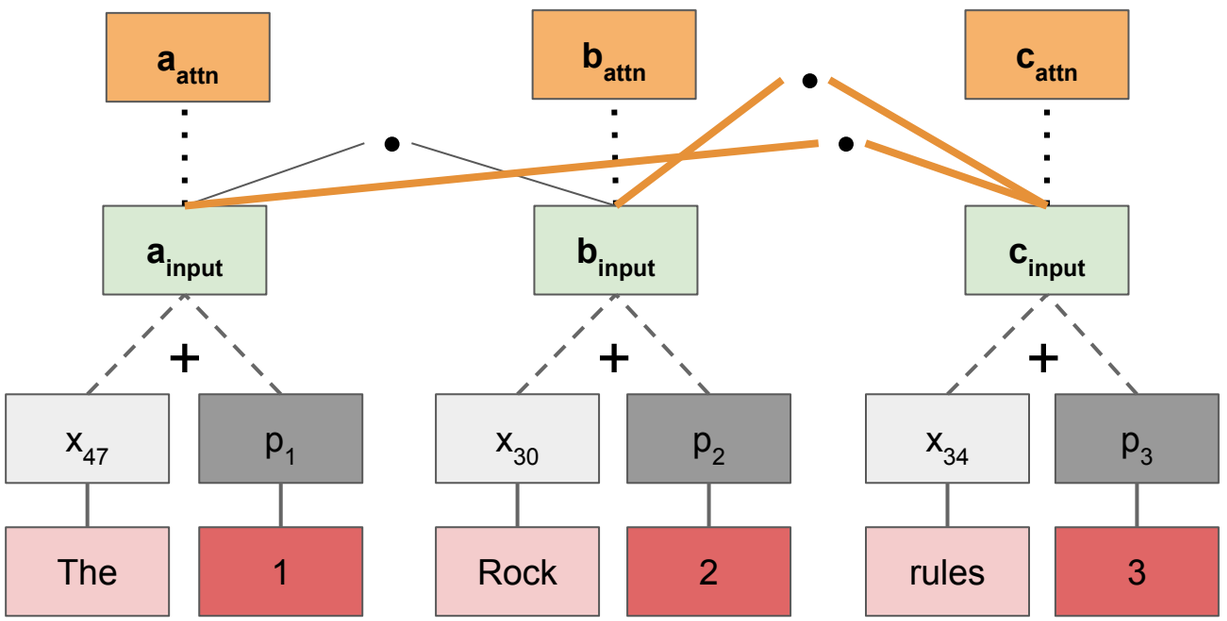


Core model structure



$$\mathrm{c}_{\mathrm{input}} = x_{34} + p_3$$

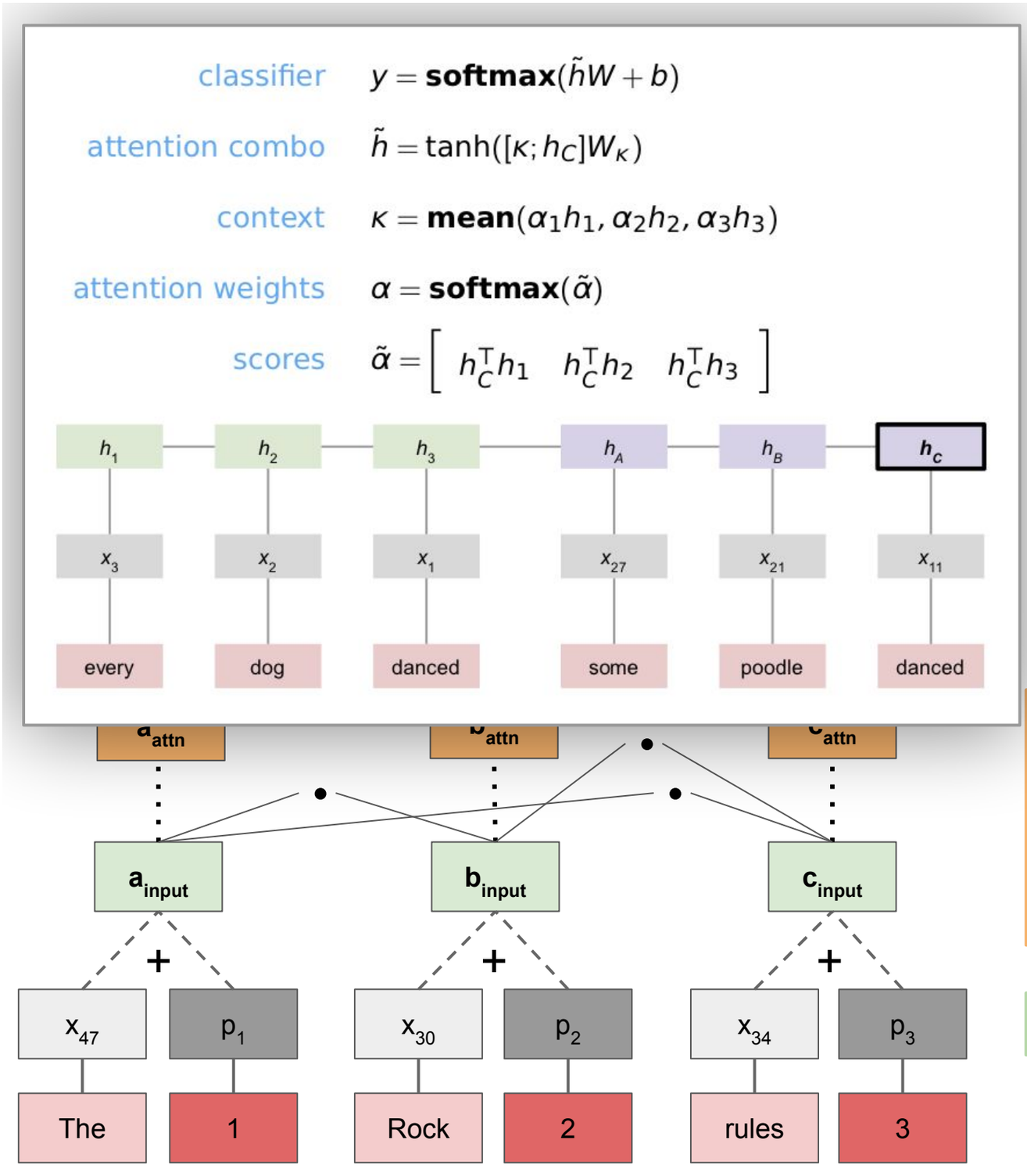
Core model structure



$$
\begin{aligned}
\mathrm{c}_{\mathrm{attn}} &= \mathbf{sum}\left([\alpha_1 \mathrm{a}_{\mathrm{input}}, \alpha_2 \mathrm{b}_{\mathrm{input}}]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{a}_{\mathrm{input}}}{\sqrt{d_k}}, \frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{b}_{\mathrm{input}}}{\sqrt{d_k}}\right]
\end{aligned}
$$

$$
\mathrm{c}_{\mathrm{input}} = x_{34} + p_3
$$

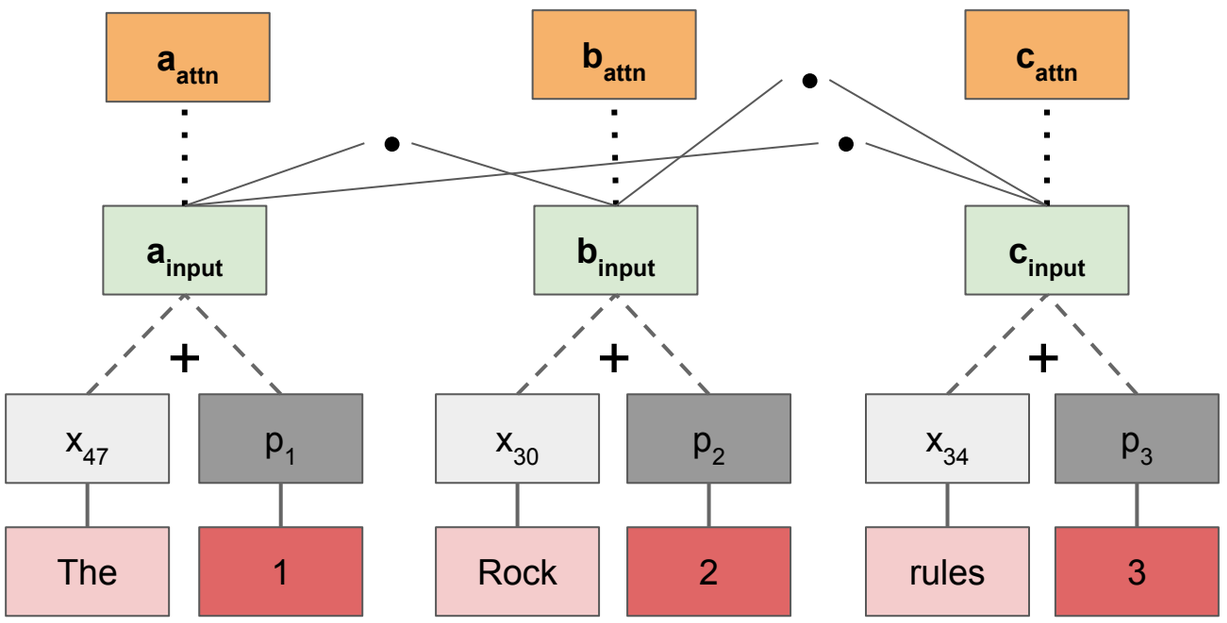
Core model structure



$$c_{attn} = \mathbf{sum}\left([\alpha_1 a_{input}, \alpha_2 b_{input}]\right)$$
$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$
$$\tilde{\alpha} = \left[\frac{c_{input}{}^\top a_{input}}{\sqrt{d_k}}, \frac{c_{input}{}^\top b_{input}}{\sqrt{d_k}}\right]$$

$$c_{input} = x_{34} + p_3$$

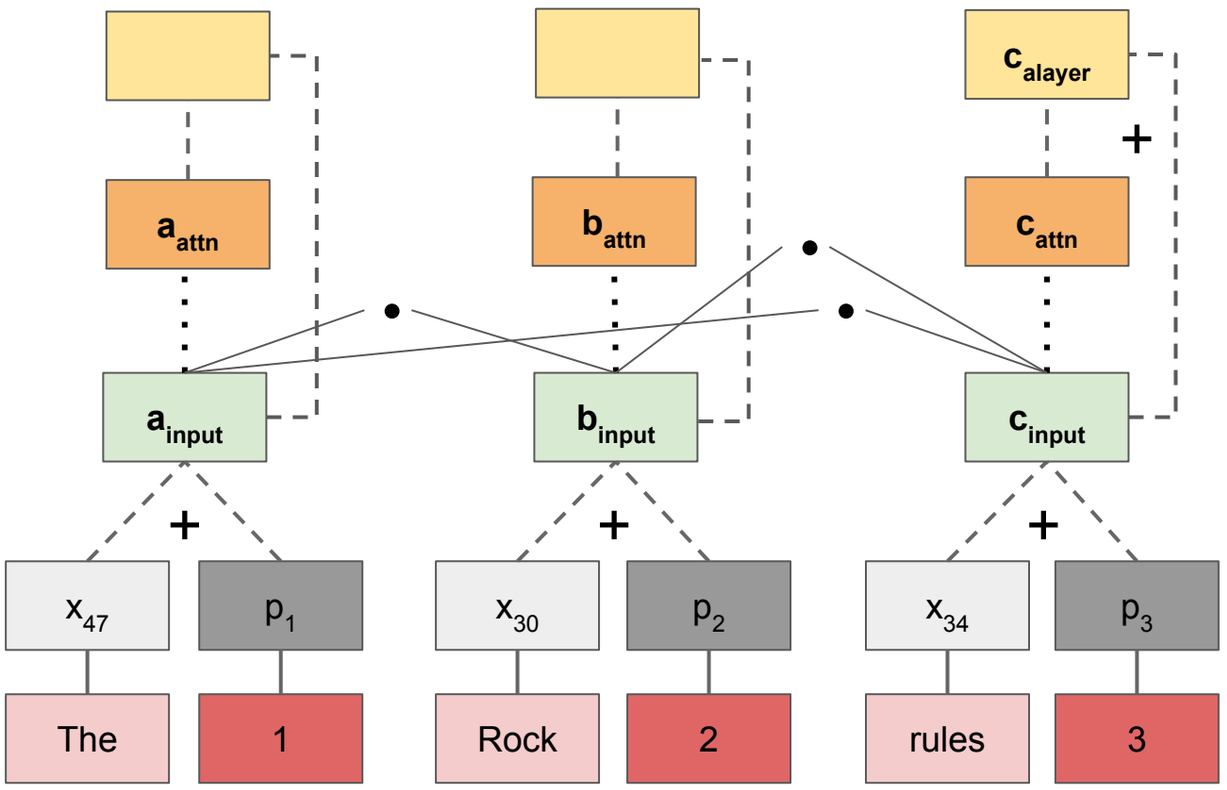
Core model structure



$$
\begin{aligned}
\mathrm{c}_{\mathrm{attn}} &= \mathbf{sum}\left([\alpha_1 \mathrm{a}_{\mathrm{input}}, \alpha_2 \mathrm{b}_{\mathrm{input}}]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{a}_{\mathrm{input}}}{\sqrt{d_k}}, \frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{b}_{\mathrm{input}}}{\sqrt{d_k}}\right]
\end{aligned}
$$

$$
\mathrm{c}_{\mathrm{input}} = x_{34} + p_3
$$

Core model structure

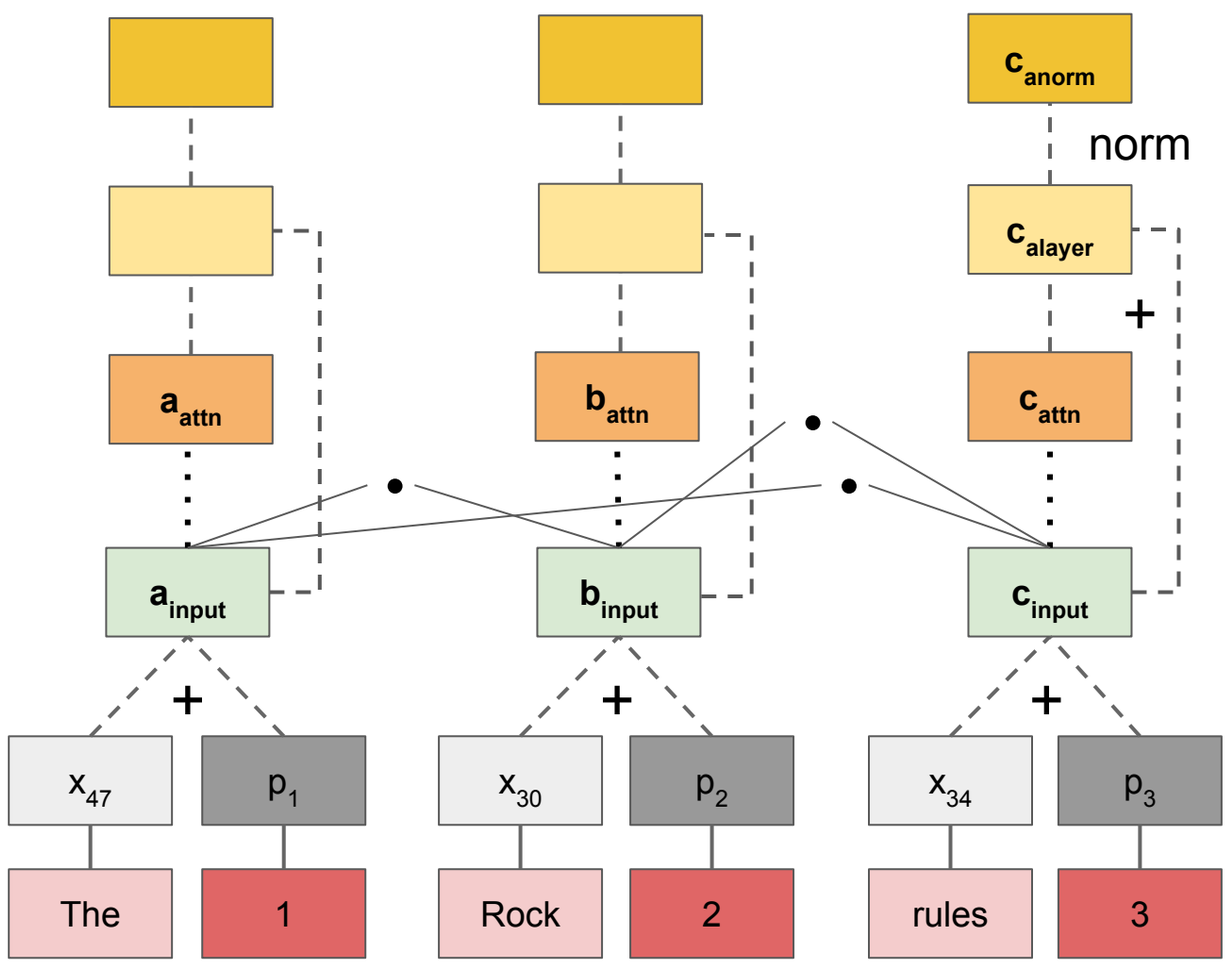


$$c_{alayer} = c_{attn} + \textbf{Dropout}(c_{input})$$

$$c_{attn} = \textbf{sum}\left([\alpha_1 a_{input}, \alpha_2 b_{input}]\right)$$
$$\alpha = \textbf{softmax}(\tilde{\alpha})$$
$$\tilde{\alpha} = \left[\frac{c_{input}{}^{\top} a_{input}}{\sqrt{d_k}}, \frac{c_{input}{}^{\top} b_{input}}{\sqrt{d_k}}\right]$$

$$c_{input} = x_{34} + p_3$$

Core model structure



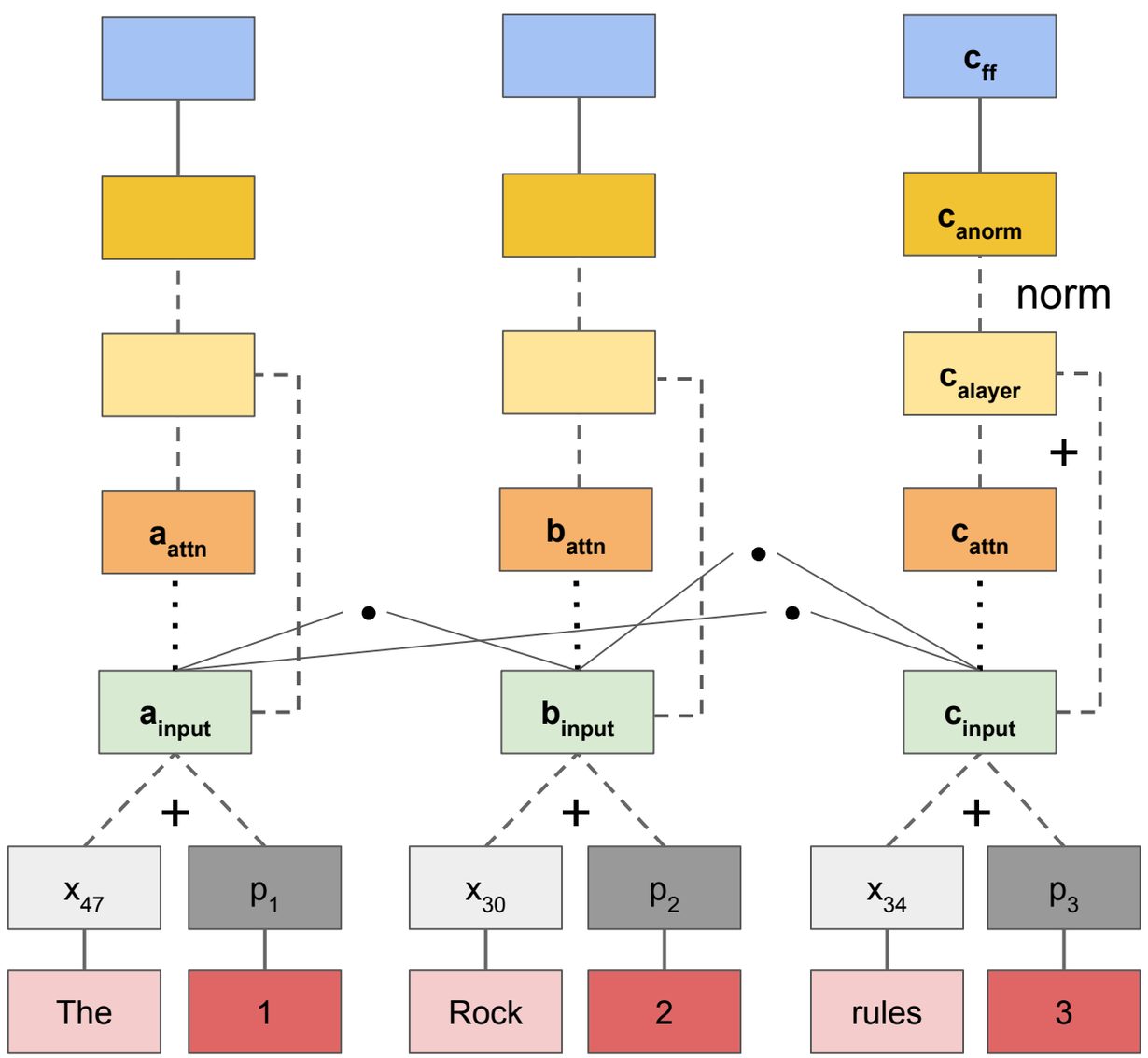
$$\mathrm{c}_{\mathrm{anorm}} = \frac{\mathrm{c}_{\mathrm{alayer}} - \mathbf{mean}(\mathrm{c}_{\mathrm{alayer}})}{\mathbf{std}(\mathrm{c}_{\mathrm{alayer}}) + \varepsilon}$$

$$\mathrm{c}_{\mathrm{alayer}} = \mathrm{c}_{\mathrm{attn}} + \mathbf{Dropout}(\mathrm{c}_{\mathrm{input}})$$

$$\mathrm{c}_{\mathrm{attn}} = \mathbf{sum}\left([\alpha_1 \mathrm{a}_{\mathrm{input}}, \alpha_2 \mathrm{b}_{\mathrm{input}}]\right)$$
$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$
$$\tilde{\alpha} = \left[\frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{a}_{\mathrm{input}}}{\sqrt{d_k}}, \frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{b}_{\mathrm{input}}}{\sqrt{d_k}}\right]$$

$$\mathrm{c}_{\mathrm{input}} = x_{34} + p_3$$

Core model structure



$$\mathrm{c_{ff}} = \mathbf{ReLU}(\mathrm{c_{anorm}} W_1 + b_1) W_2 + b_2$$

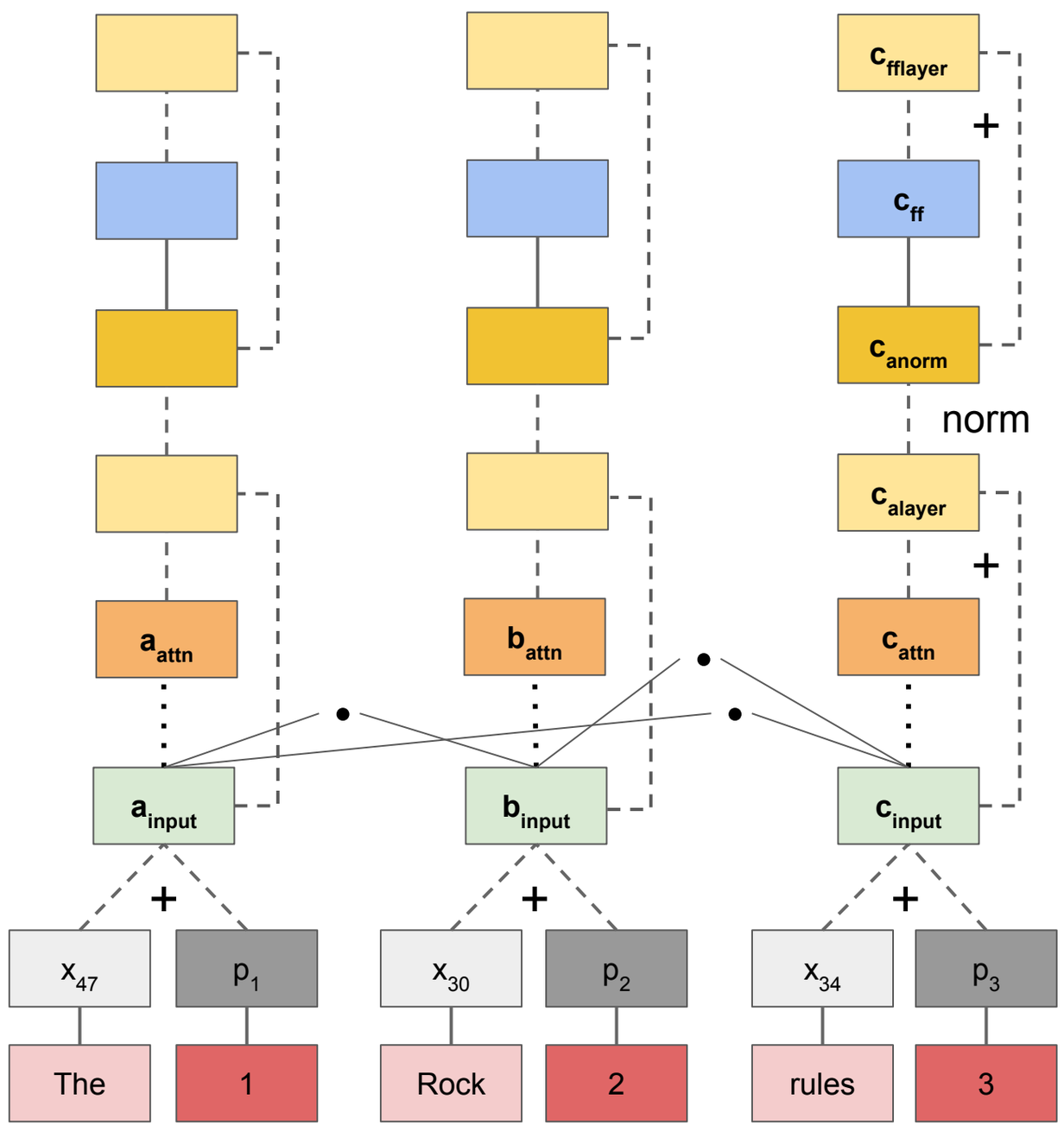
$$\mathrm{c_{anorm}} = \frac{\mathrm{c_{alayer}} - \mathbf{mean}(\mathrm{c_{alayer}})}{\mathbf{std}(\mathrm{c_{alayer}}) + \varepsilon}$$

$$\mathrm{c_{alayer}} = \mathrm{c_{attn}} + \mathbf{Dropout}(\mathrm{c_{input}})$$

$$\mathrm{c_{attn}} = \mathbf{sum}\left([\alpha_1 \mathrm{a_{input}}, \alpha_2 \mathrm{b_{input}}]\right)$$
$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$
$$\tilde{\alpha} = \left[\frac{\mathrm{c_{input}}^\top \mathrm{a_{input}}}{\sqrt{d_k}}, \frac{\mathrm{c_{input}}^\top \mathrm{b_{input}}}{\sqrt{d_k}}\right]$$

$$\mathrm{c_{input}} = x_{34} + p_3$$

Core model structure



$$\mathrm{c}_{\text{fflayer}} = \mathrm{c}_{\text{anorm}} + \mathbf{Dropout}(\mathrm{c}_{\text{ff}})$$

$$\mathrm{c}_{\text{ff}} = \mathbf{ReLU}(\mathrm{c}_{\text{anorm}} W_1 + b_1) W_2 + b_2$$

$$\mathrm{c}_{\text{anorm}} = \frac{\mathrm{c}_{\text{alayer}} - \mathbf{mean}(\mathrm{c}_{\text{alayer}})}{\mathbf{std}(\mathrm{c}_{\text{alayer}}) + \varepsilon}$$

$$\mathrm{c}_{\text{alayer}} = \mathrm{c}_{\text{attn}} + \mathbf{Dropout}(\mathrm{c}_{\text{input}})$$

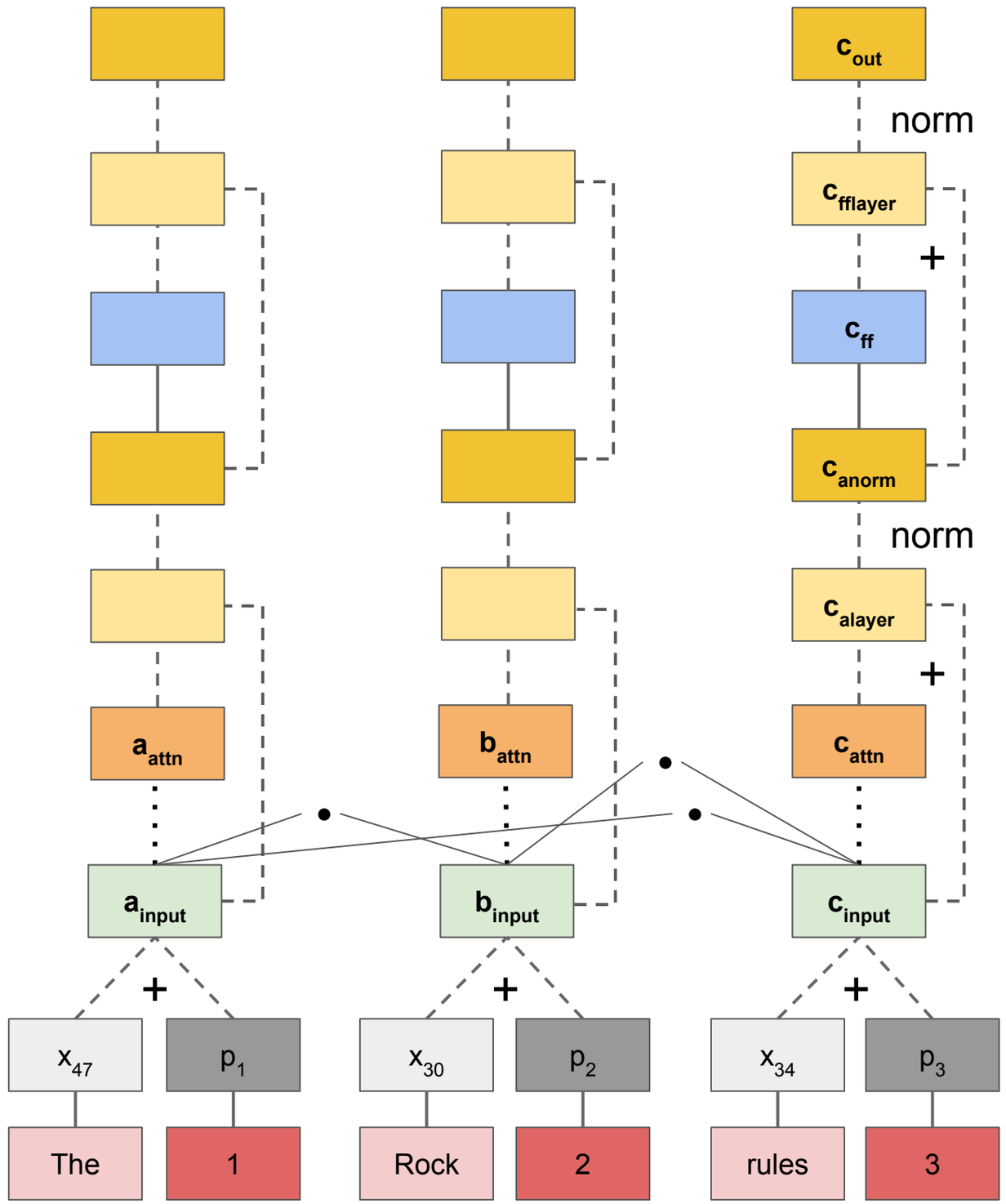
$$\mathrm{c}_{\text{attn}} = \mathbf{sum}\left([\alpha_1 \mathrm{a}_{\text{input}}, \alpha_2 \mathrm{b}_{\text{input}}]\right)$$

$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$

$$\tilde{\alpha} = \left[\frac{\mathrm{c}_{\text{input}}{}^{\top} \mathrm{a}_{\text{input}}}{\sqrt{d_k}}, \frac{\mathrm{c}_{\text{input}}{}^{\top} \mathrm{b}_{\text{input}}}{\sqrt{d_k}}\right]$$

$$\mathrm{c}_{\text{input}} = x_{34} + p_3$$

Core model structure



$$\mathrm{c_{out}} = \frac{\mathrm{c_{fflayer}} - \mathbf{mean}(\mathrm{c_{fflayer}})}{\mathbf{std}(\mathrm{c_{fflayer}}) + \varepsilon}$$

$$\mathrm{c_{fflayer}} = \mathrm{c_{anorm}} + \mathbf{Dropout}(\mathrm{c_{ff}})$$

$$\mathrm{c_{ff}} = \mathbf{ReLU}(\mathrm{c_{anorm}} W_1 + b_1) W_2 + b_2$$

$$\mathrm{c_{anorm}} = \frac{\mathrm{c_{alayer}} - \mathbf{mean}(\mathrm{c_{alayer}})}{\mathbf{std}(\mathrm{c_{alayer}}) + \varepsilon}$$

$$\mathrm{c_{alayer}} = \mathrm{c_{attn}} + \mathbf{Dropout}(\mathrm{c_{input}})$$

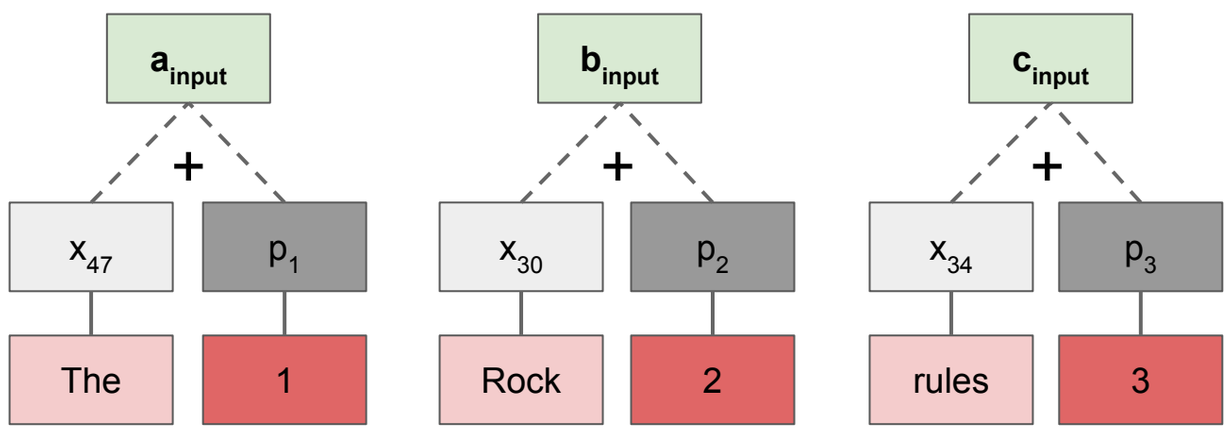
$$\mathrm{c_{attn}} = \mathbf{sum}\left([\alpha_1 \mathrm{a_{input}}, \alpha_2 \mathrm{b_{input}}]\right)$$

$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$

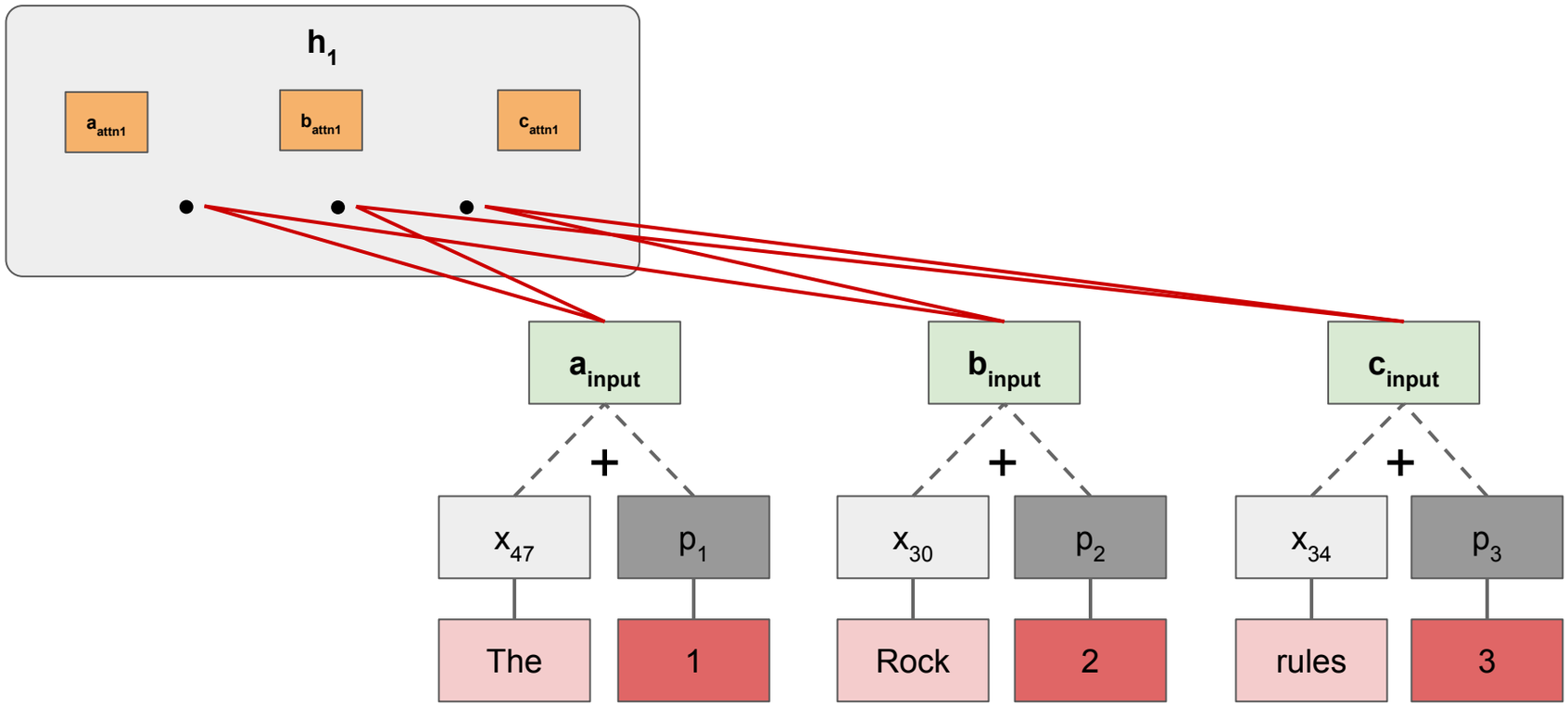
$$\tilde{\alpha} = \left[\frac{\mathrm{c_{input}}^\top \mathrm{a_{input}}}{\sqrt{d_k}}, \frac{\mathrm{c_{input}}^\top \mathrm{b_{input}}}{\sqrt{d_k}}\right]$$

$$\mathrm{c_{input}} = x_{34} + p_3$$

Multi-headed attention



Multi-headed attention

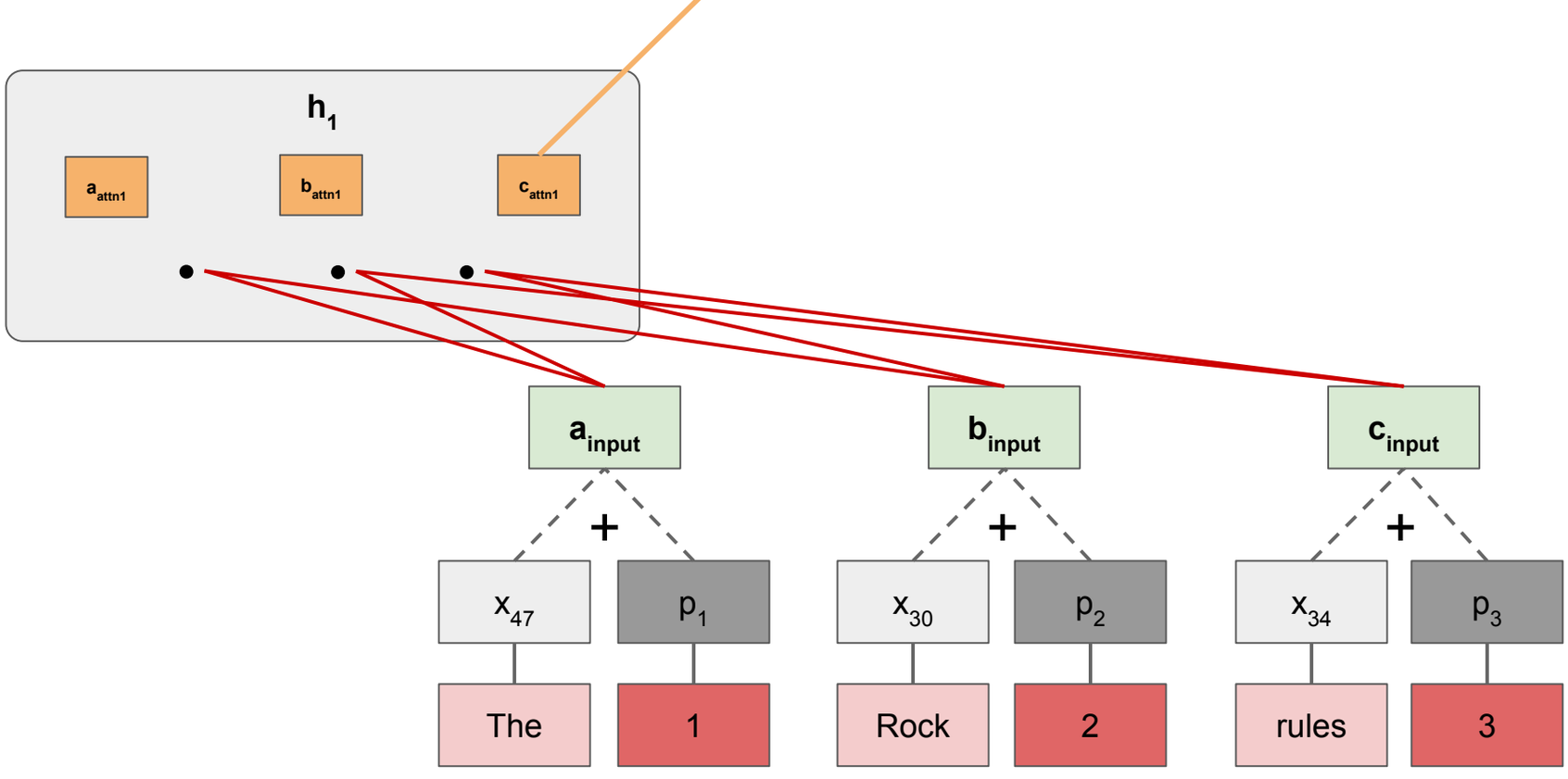


Multi-headed attention

$$\mathrm{c}_{\mathrm{attn1}} = \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_1^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_1^V)\right]\right)$$

$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$

$$\tilde{\alpha} = \left[\frac{(\mathrm{c}_{\mathrm{input}} W_1^Q)^\top (\mathrm{a}_{\mathrm{input}} W_1^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_1^Q)^\top (\mathrm{b}_{\mathrm{input}} W_1^K)}{\sqrt{d_k}}\right]$$

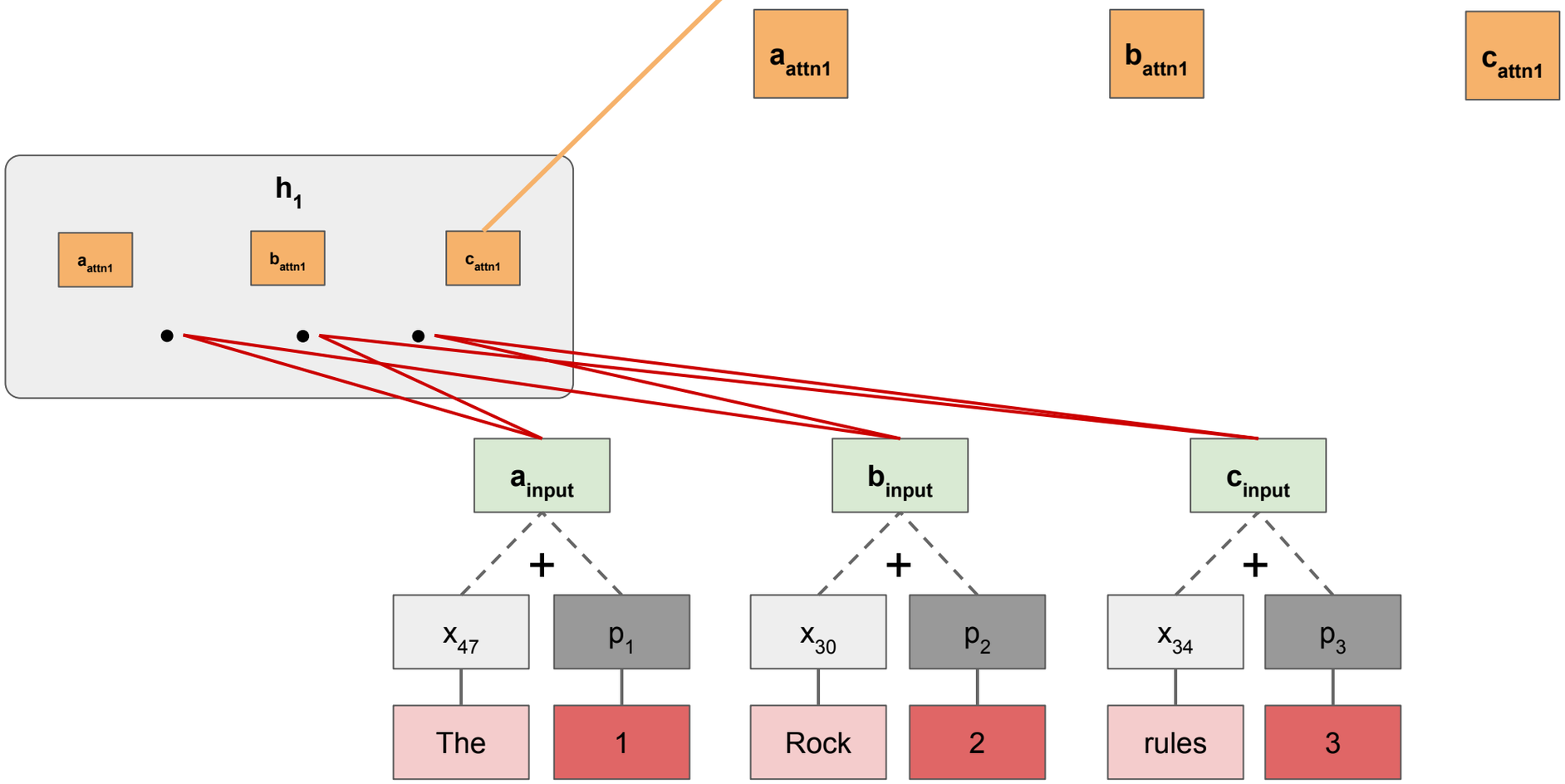


Multi-headed attention

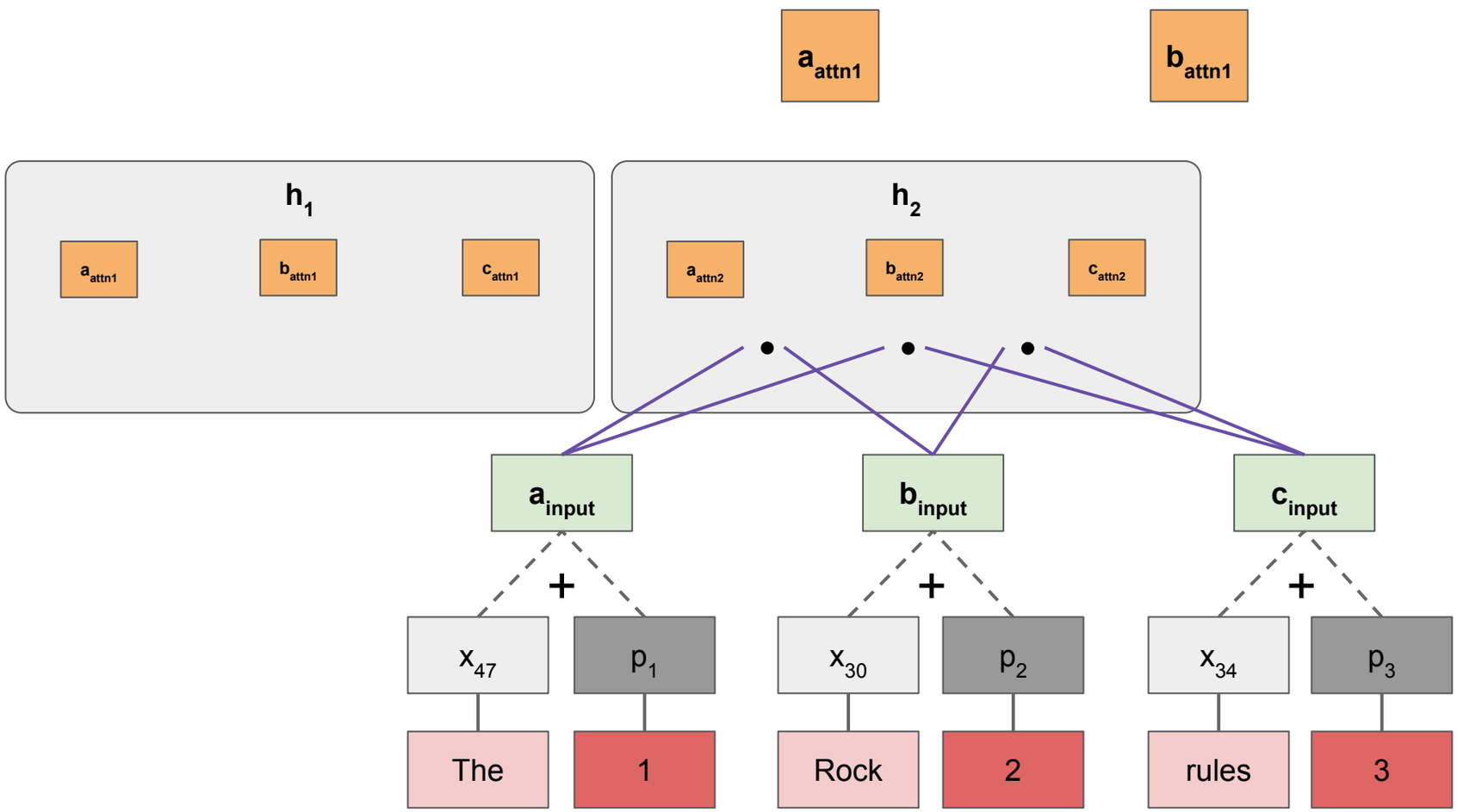
$$\mathrm{c}_{\mathrm{attn1}} = \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_1^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_1^V)\right]\right)$$

$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$

$$\tilde{\alpha} = \left[\frac{(\mathrm{c}_{\mathrm{input}} W_1^Q)^\top (\mathrm{a}_{\mathrm{input}} W_1^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_1^Q)^\top (\mathrm{b}_{\mathrm{input}} W_1^K)}{\sqrt{d_k}}\right]$$



Multi-headed attention

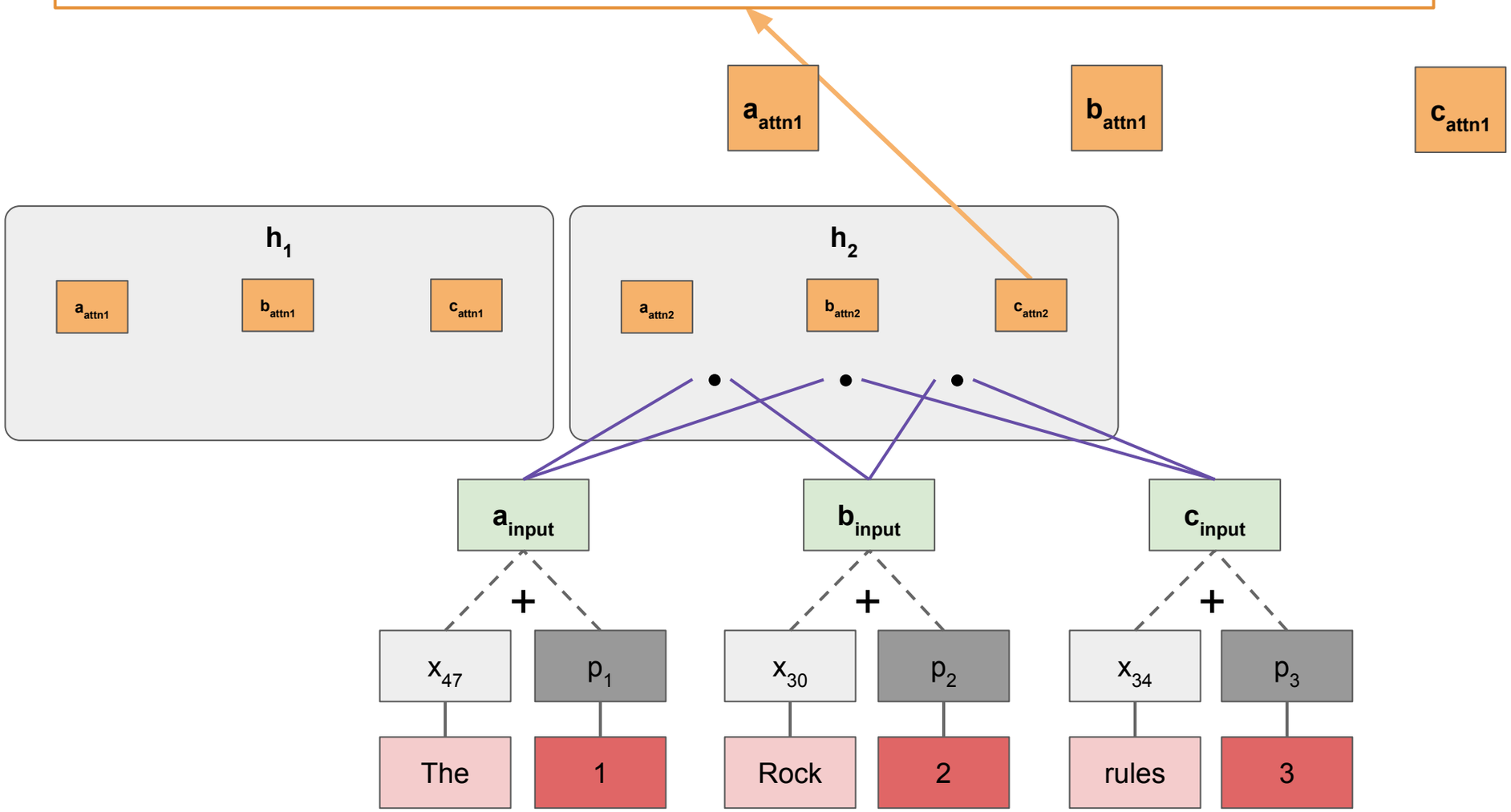


Multi-headed attention

$$\mathrm{c}_{\mathrm{attn2}} = \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_2^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_2^V)\right]\right)$$

$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$

$$\tilde{\alpha} = \left[\frac{(\mathrm{c}_{\mathrm{input}} W_2^Q)^\top (\mathrm{a}_{\mathrm{input}} W_2^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_2^Q)^\top (\mathrm{b}_{\mathrm{input}} W_2^K)}{\sqrt{d_k}}\right]$$

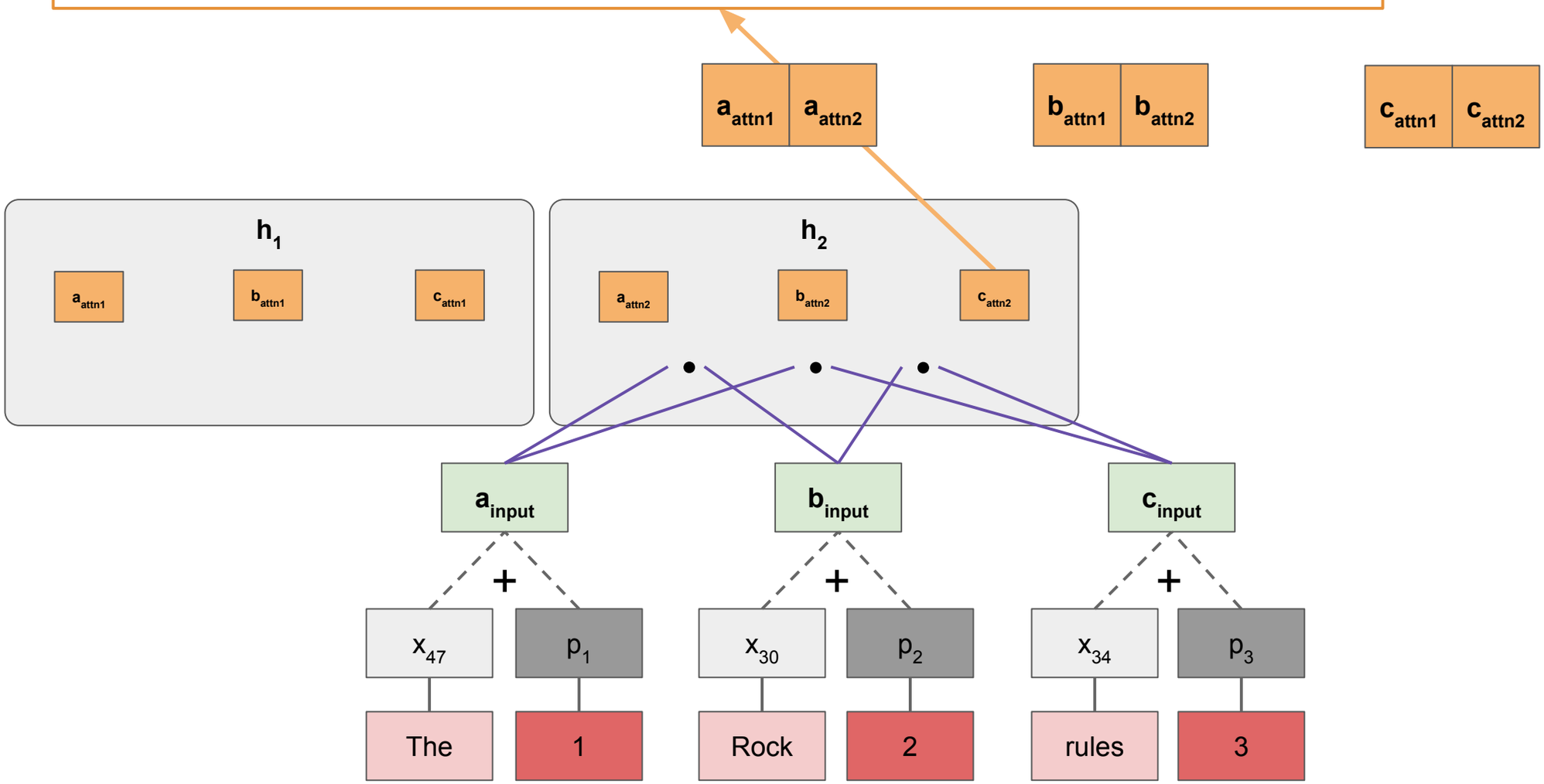


Multi-headed attention

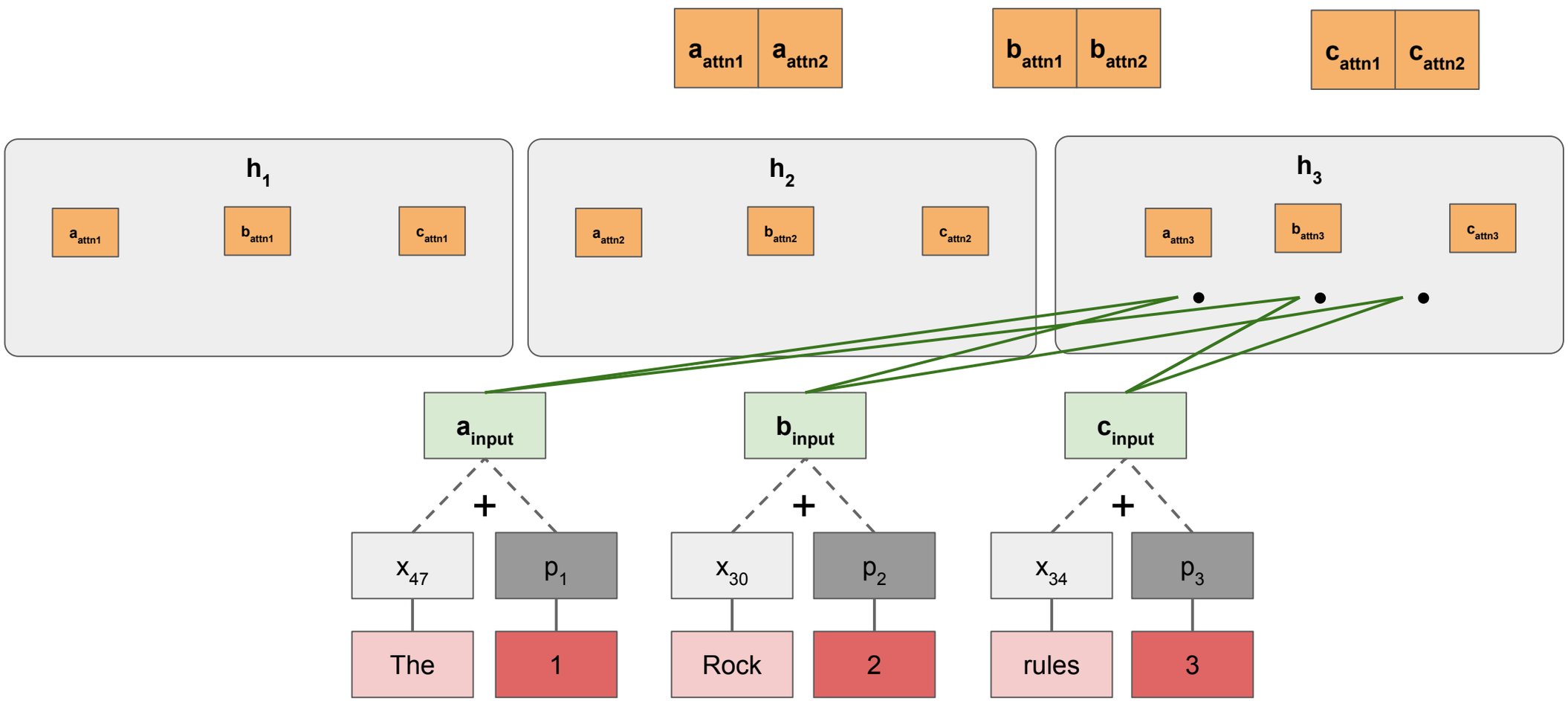
$$\mathrm{c}_{\mathrm{attn2}} = \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_2^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_2^V)\right]\right)$$

$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$

$$\tilde{\alpha} = \left[\frac{(\mathrm{c}_{\mathrm{input}} W_2^Q)^\top (\mathrm{a}_{\mathrm{input}} W_2^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_2^Q)^\top (\mathrm{b}_{\mathrm{input}} W_2^K)}{\sqrt{d_k}}\right]$$



Multi-headed attention

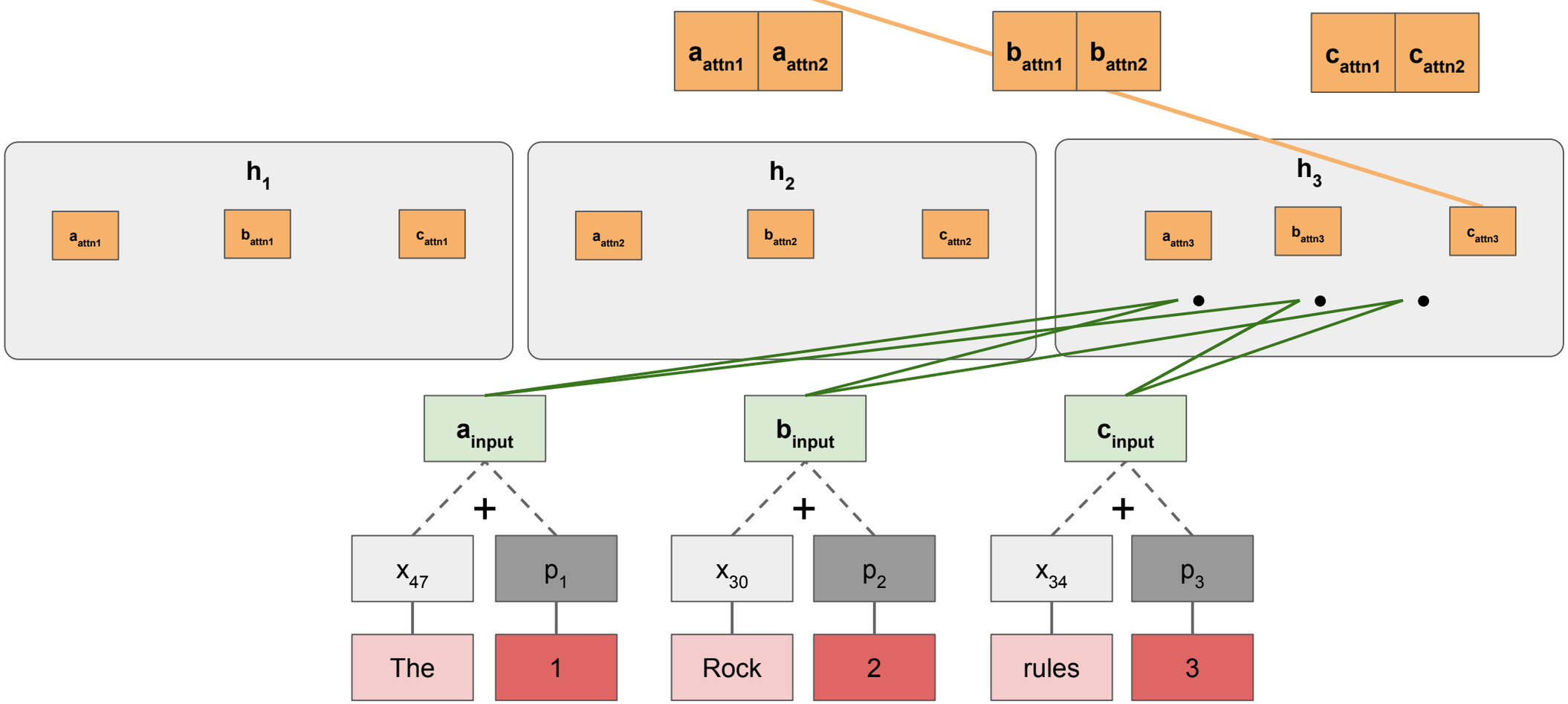


Multi-headed attention

$$\mathrm{c}_{\mathrm{attn3}} = \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_3^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_3^V)\right]\right)$$

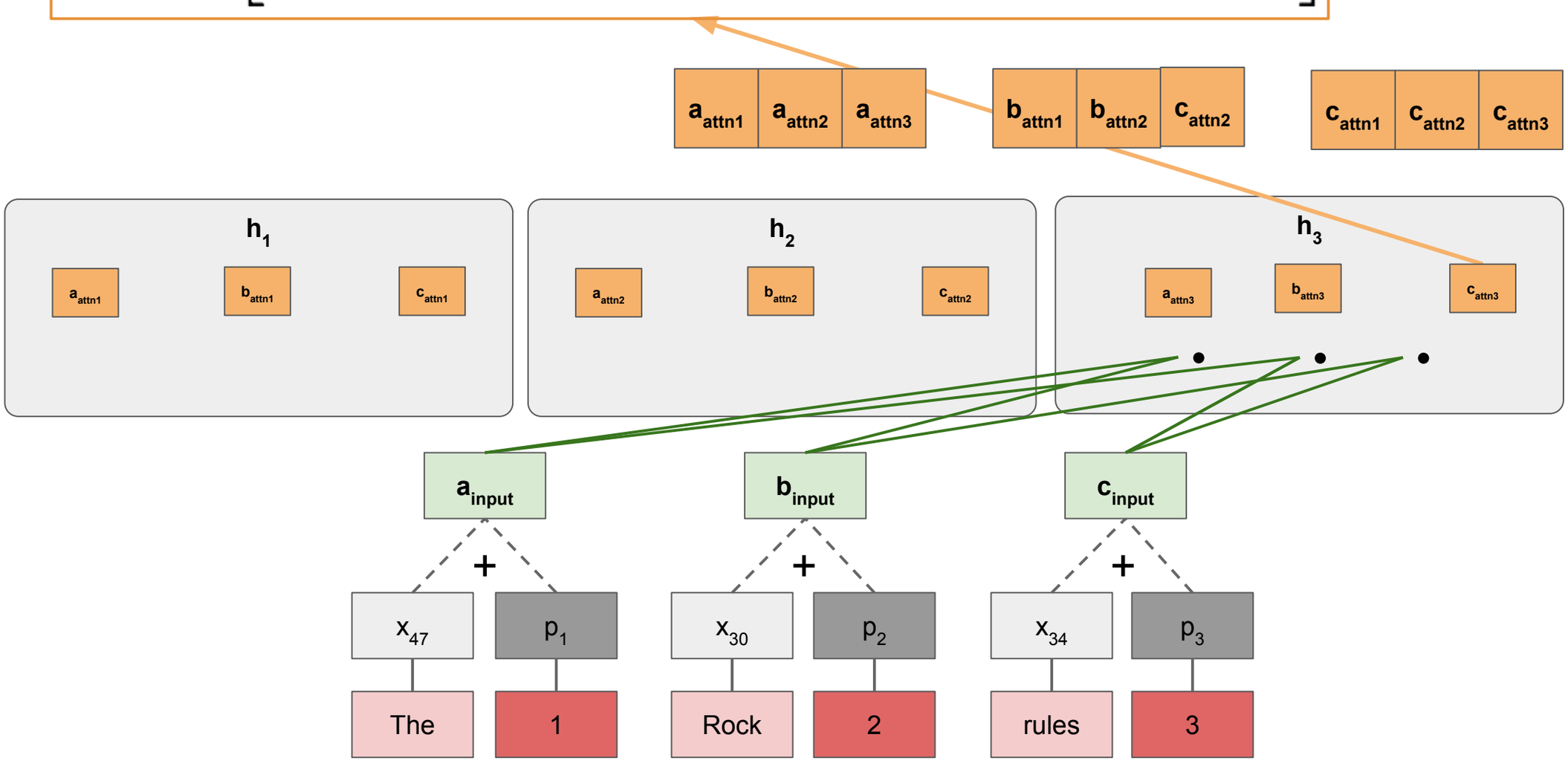
$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$

$$\tilde{\alpha} = \left[\frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{a}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{b}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}\right]$$



Multi-headed attention

$$\begin{aligned}
\mathrm{c}_{\mathrm{attn3}} &= \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_3^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_3^V)\right]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{a}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{b}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}\right]
\end{aligned}$$

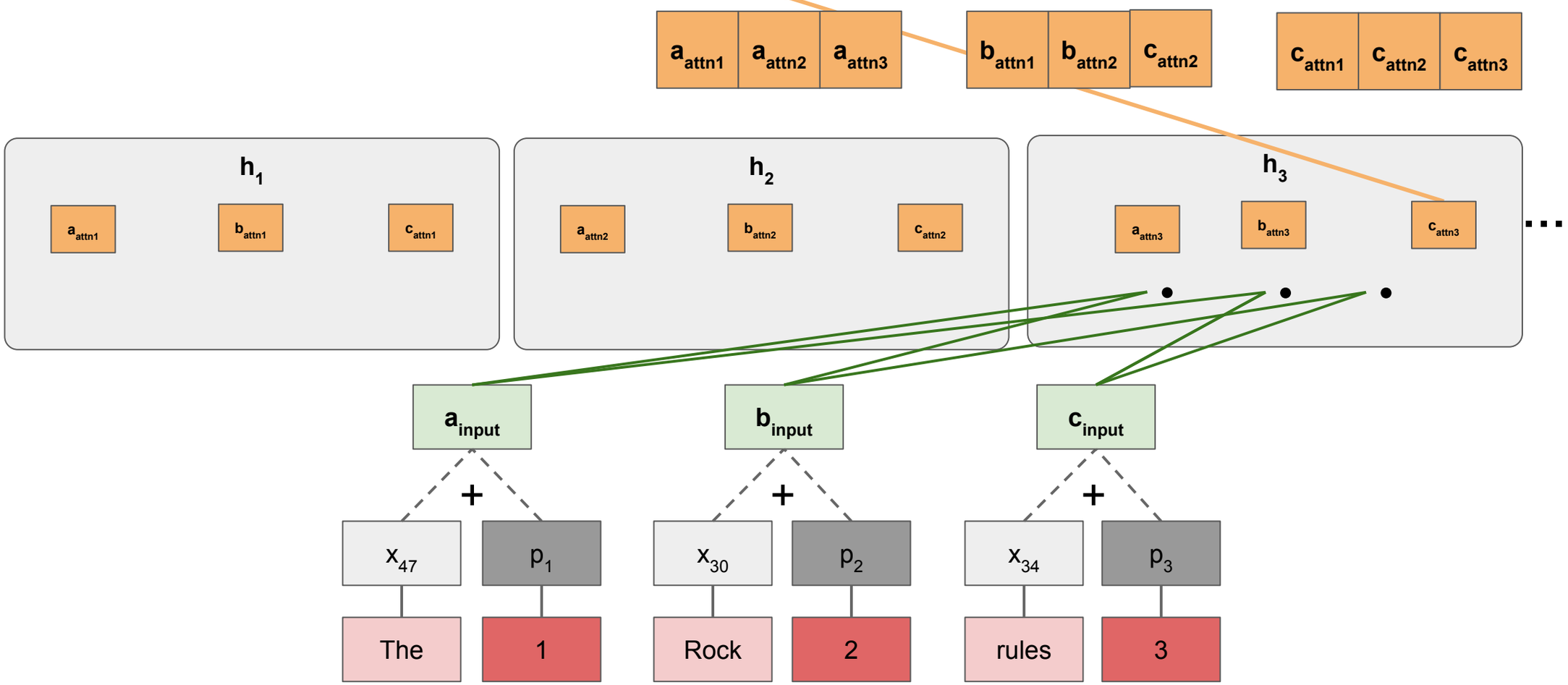


Multi-headed attention

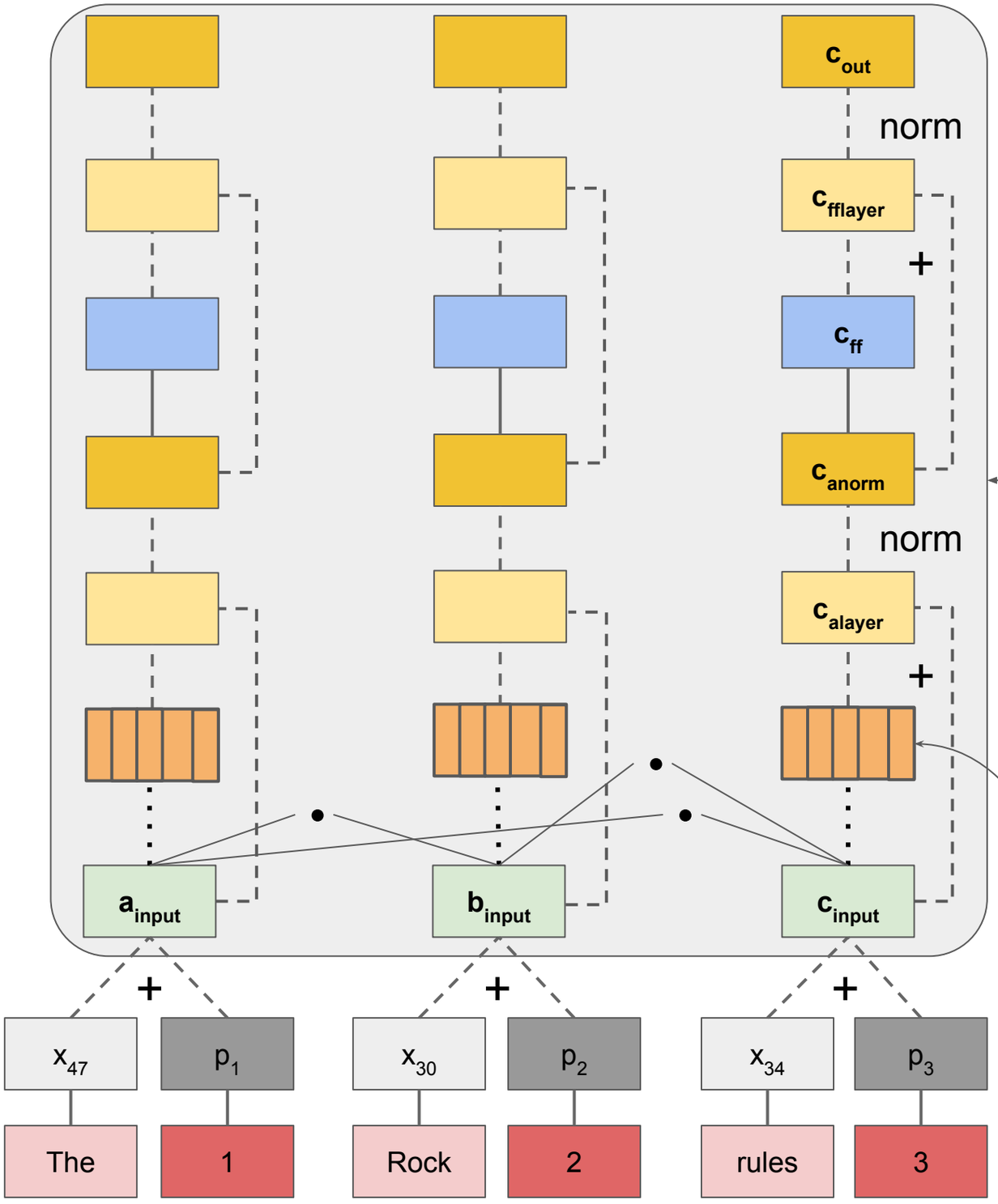
$$\mathrm{c}_{\mathrm{attn3}} = \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_3^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_3^V)\right]\right)$$

$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$

$$\tilde{\alpha} = \left[\frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{a}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{b}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}\right]$$



Repeated transformer blocks



Repeated 6 times, with c_{out} serving as c_{input} to each successive layer

Reminder that we also do multi-headed attention in each layer

The architecture diagram

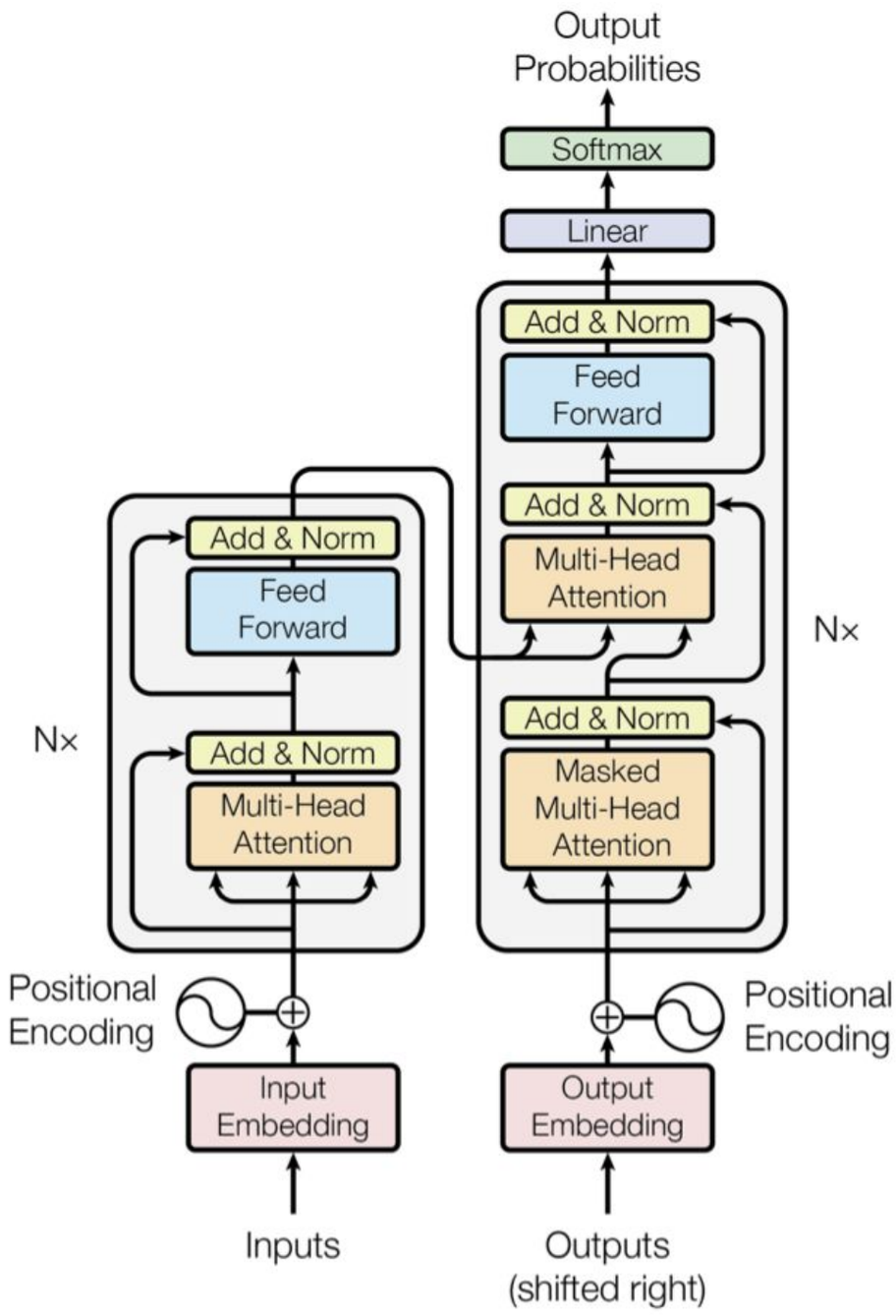


Figure 1: The Transformer - model architecture.

The architecture diagram

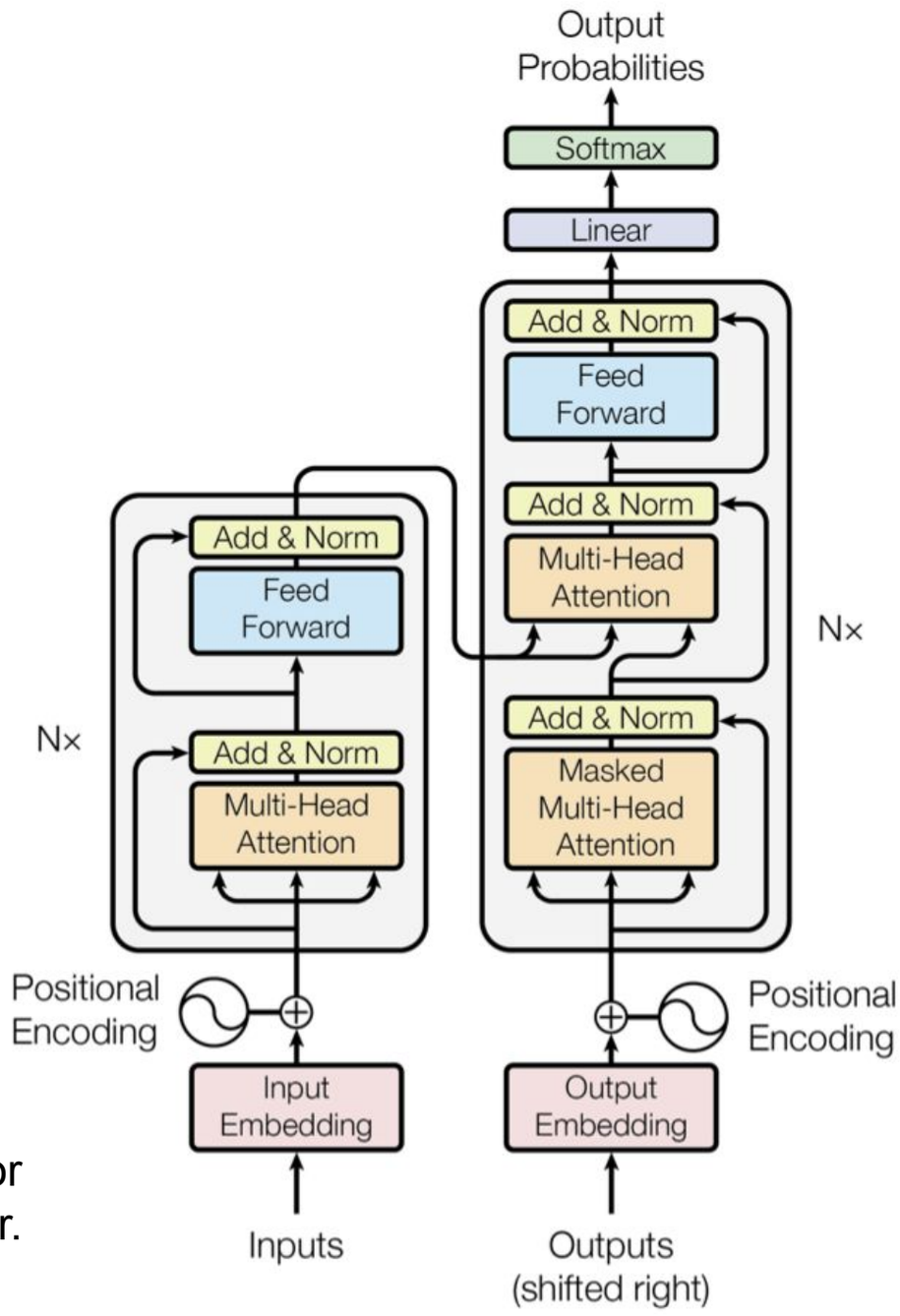
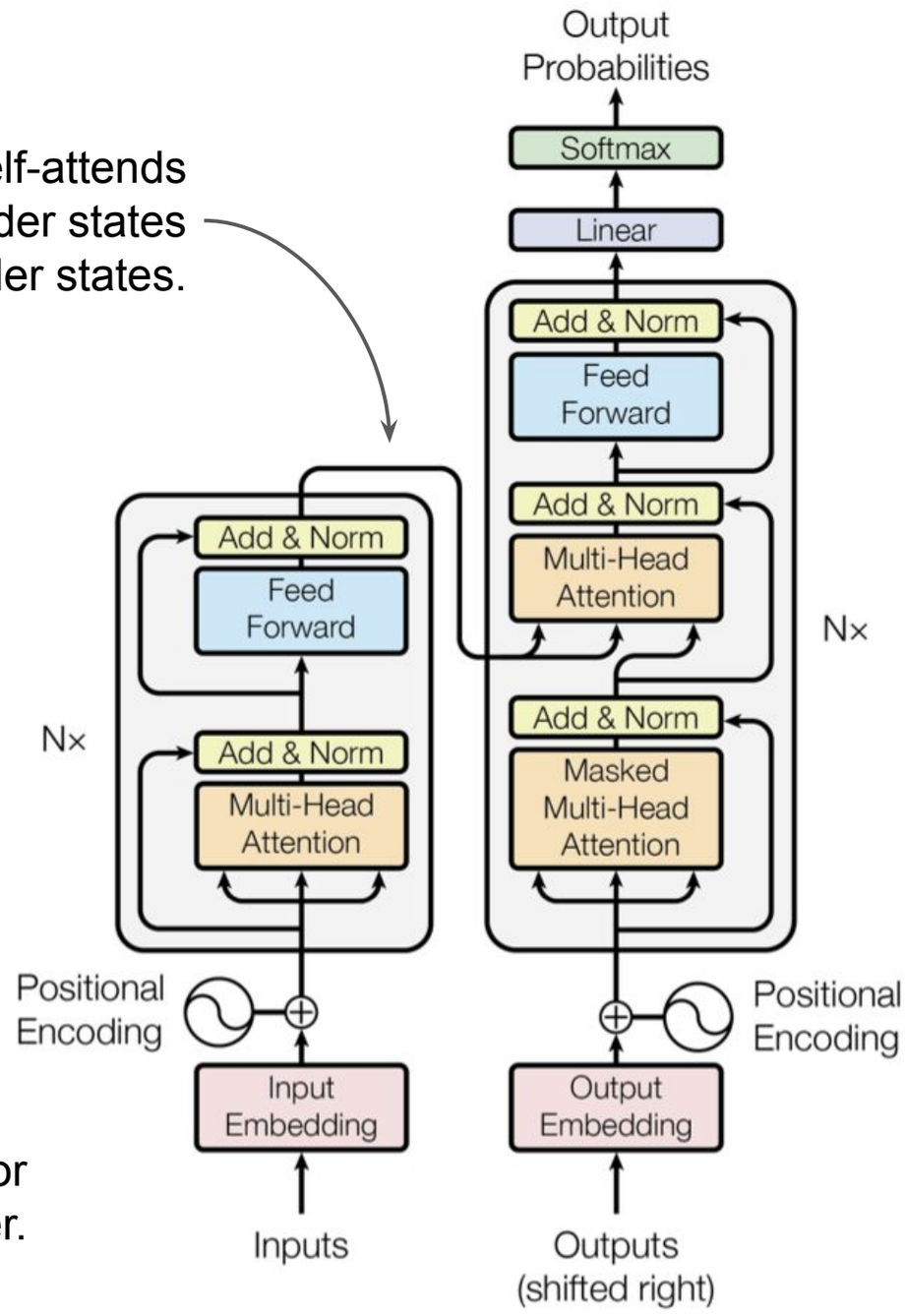


Figure 1: The Transformer - model architecture.

The left side is repeated for every state in the encoder.

The architecture diagram

Each decoder state self-attends with all of its fellow decoder states and with all the encoder states.

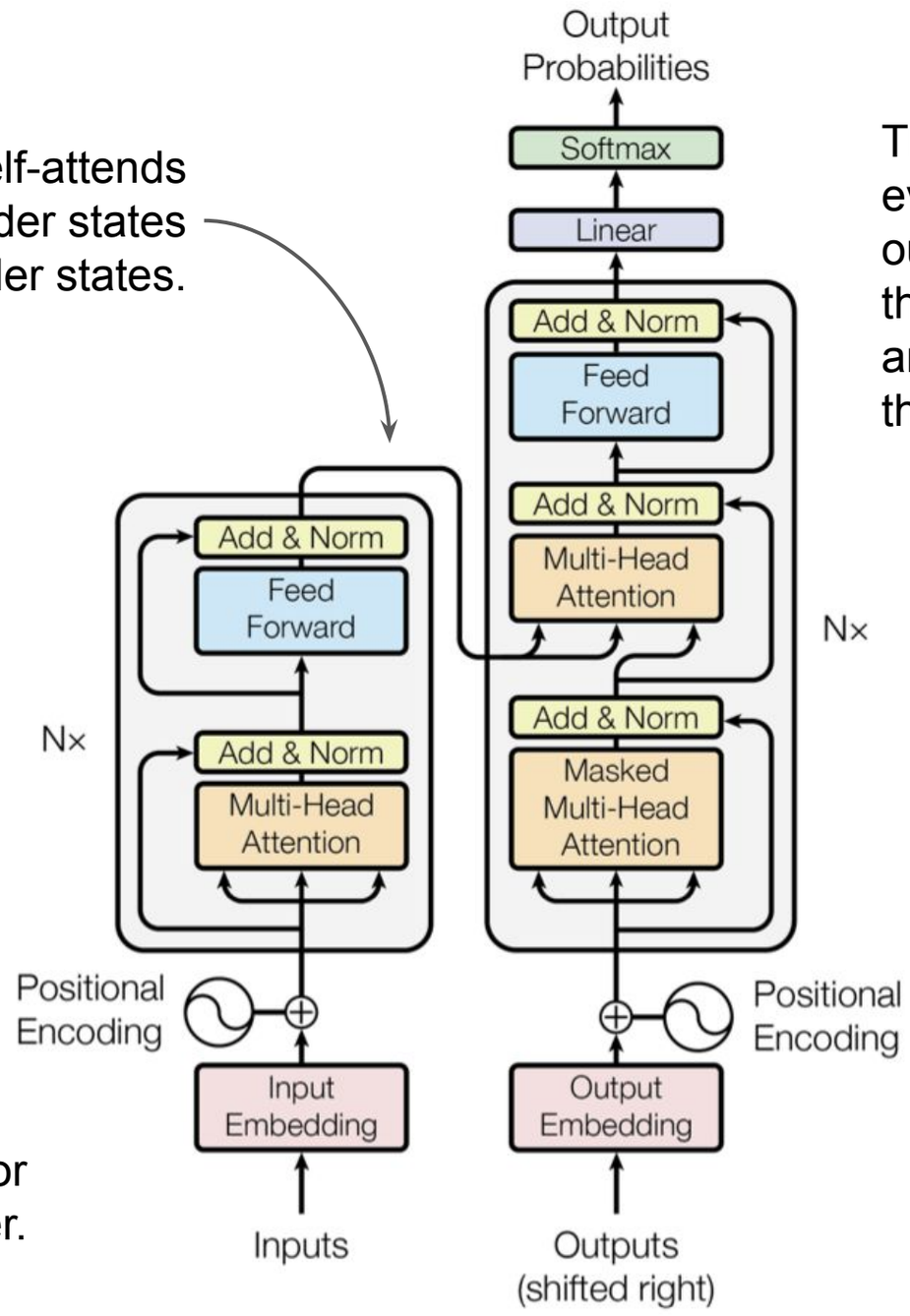


The left side is repeated for every state in the encoder.

Figure 1: The Transformer - model architecture.

The architecture diagram

Each decoder state self-attends with all of its fellow decoder states and with all the encoder states.



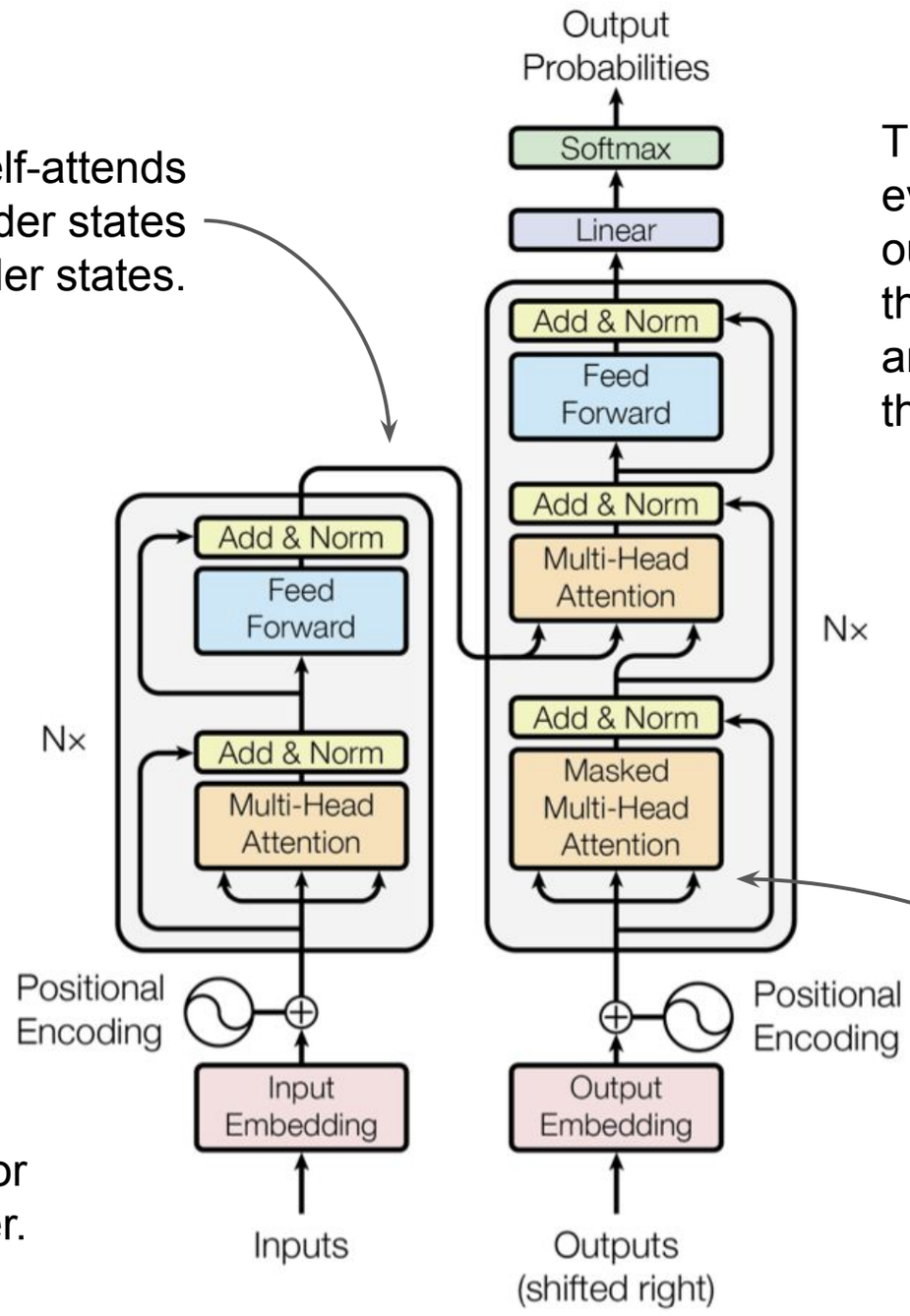
The right side is repeated for every decoder state, with outputs for each state that has them (all of them for dialogue and machine translation, only the final one for NLI).

The left side is repeated for every state in the encoder.

Figure 1: The Transformer - model architecture.

The architecture diagram

Each decoder state self-attends with all of its fellow decoder states and with all the encoder states.



The right side is repeated for every decoder state, with outputs for each state that has them (all of them for dialogue and machine translation, only the final one for NLI).

The left side is repeated for every state in the encoder.

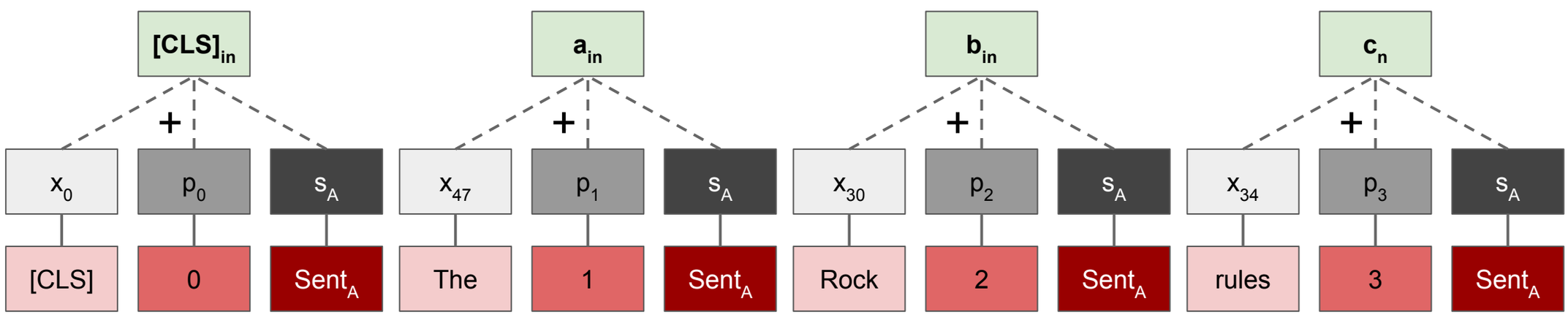
In the decoder, self-attention is limited to preceding words.

Figure 1: The Transformer - model architecture.

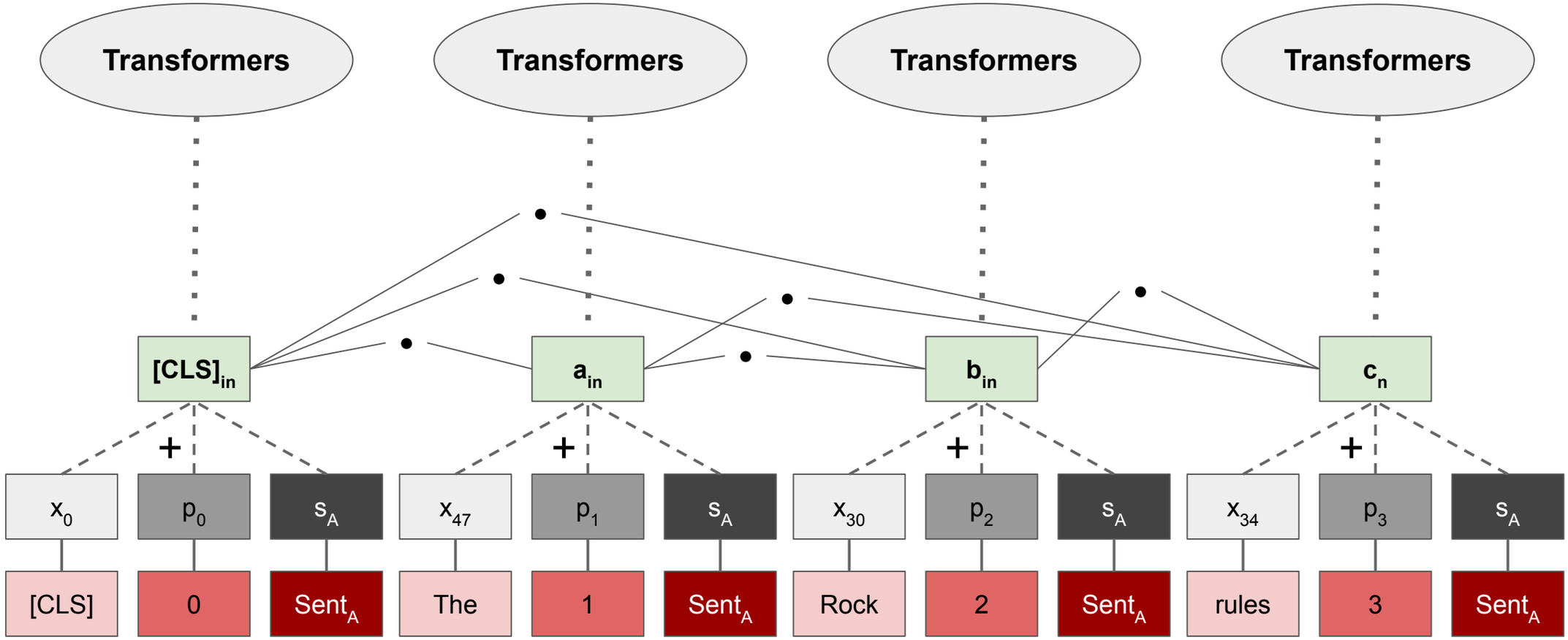
BERT

1. Overview: Resources and guiding insights
2. ELMo: **E**mbeddings from Language **Mo**dels
3. Transformers
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
5. `contextualreps.ipynb`: Easy ways to bring ELMo and BERT into your project

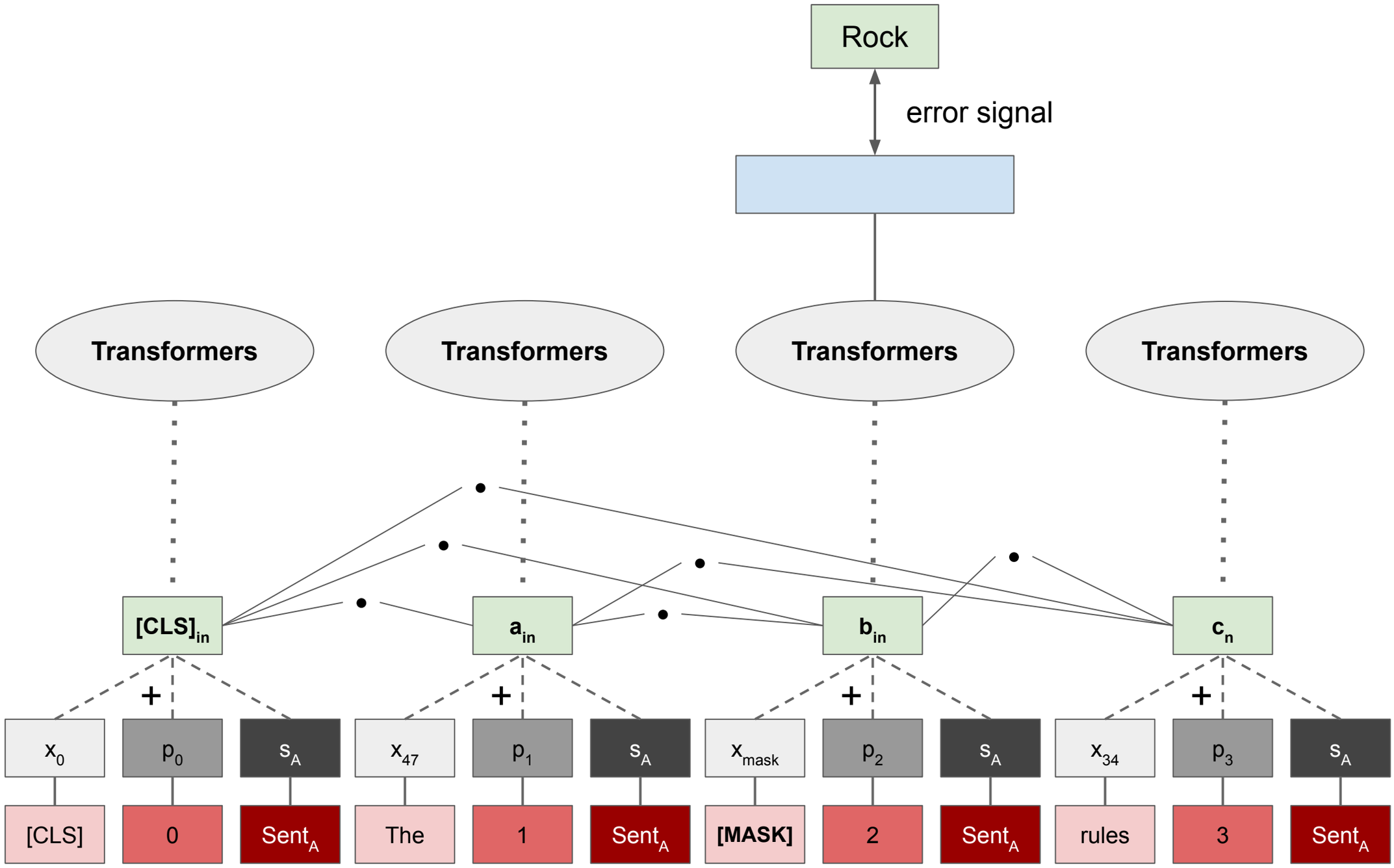
Core model structure



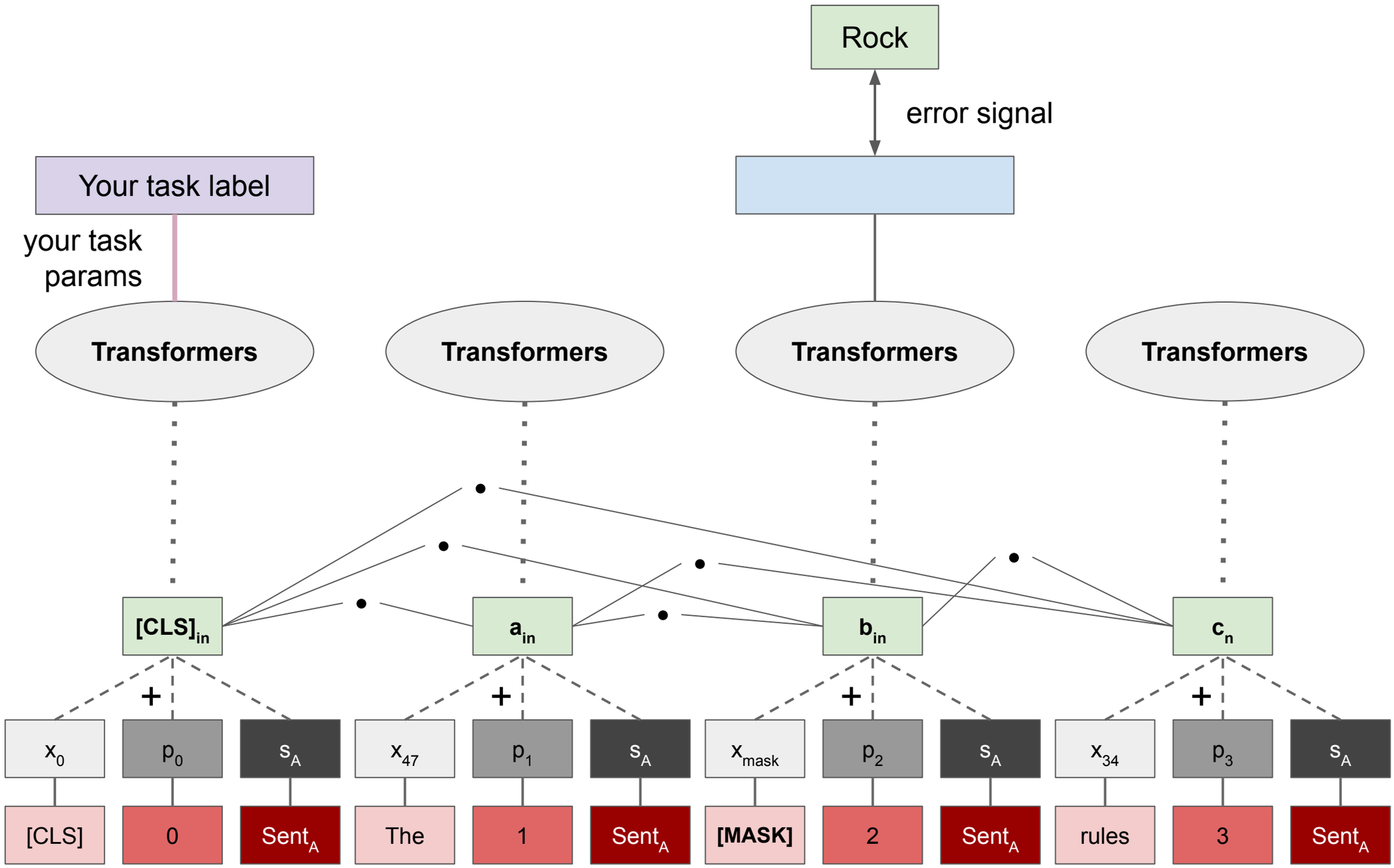
Core model structure



Masked Language Modeling (MLM)



Transfer learning and fine-tuning



Binary sentence prediction pretraining

Positive: Actual sentence sequences

- [CLS] the man went to [MASK] store [SEP]
- he bought a gallon [MASK] milk [SEP]
- Label: `IsNext`

Negative: Randomly chosen second sentence

- [CLS] the man went to [MASK] store [SEP]
- penguin [MASK] are flight ##less birds [SEP]
- Label: `NotNext`

BERT model releases

Base

- Transformer layers: 12
- Hidden representations: 768 dimensions
- Attention heads: 12
- Total parameters: 110M

Large

- Transformer layers: 24
- Hidden representations: 1024 dimensions
- Attention heads: 16
- Total parameters: 340M

Limited to sequences of 512 tokens due to dimensionality of the positional embeddings.

References I

Devlin, Jacob, Ming-Wei Chang, Kenton Lee & Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the north american association of computational linguistics*, Stroudsburg, PA: Association for Computational Linguistics.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee & Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)*, 2227–2237. Association for Computational Linguistics. http://aclweb.org/anthology/N18-1202.

Smith, Noah A. 2019. Contextual word representations: A contextual introduction. ArXiv:1902.06006v2.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser & Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (eds.), *Advances in neural information processing systems 30*, 5998–6008. Curran Associates, Inc. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

resources

- very useful detailed explanation of the transformer architecture: the illustrated transformer

- walk-through explanation of the transformer code: the annotated transformer

and that’s a wrap!

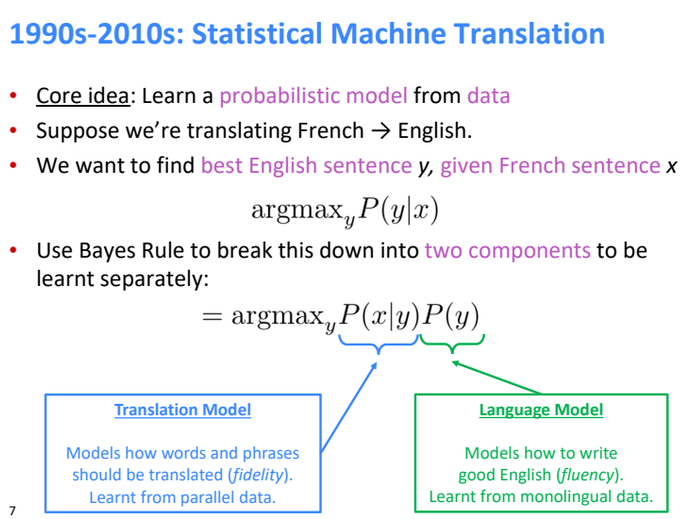
- we’ve seen many ways to represent linguistic objects, and do something with the representations

  - knowledge, learning, search

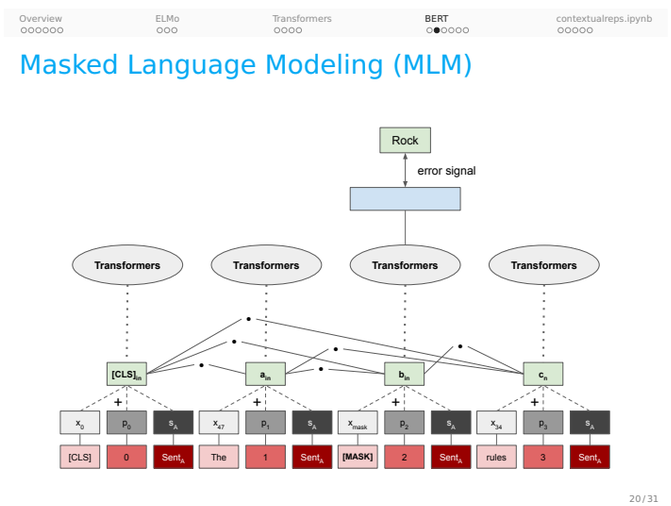
slide credits

slides that look like this

come from



Chris Manning & Abigail See's Stanford cs224n course, 2019 edition



Chris Pott's Stanford cs224u course, 2019 edition

and their use is gratefully acknowledged. I try to make any modifications obvious, but if there are errors on a slide, assume that I added them.