This is AI4001

GCR : t37g47w

These slides are taken from Stanford course CS224N!

All credits goes to them.

References

https://web.stanford.edu/class/cs224n/slides/cs224n-2022-lecture08-final-project.pdf

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

NMT: perhaps the biggest success story of NLP Deep Learning?

Neural Machine Translation went from a fringe research attempt in **2014** to the leading standard method in **2016**

- **2014**: First seq2seq paper published

- **2016**: Google Translate switches from SMT to NMT – and by 2018 everyone has

- This is amazing!
  - **SMT** systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by a small group of engineers in a few months

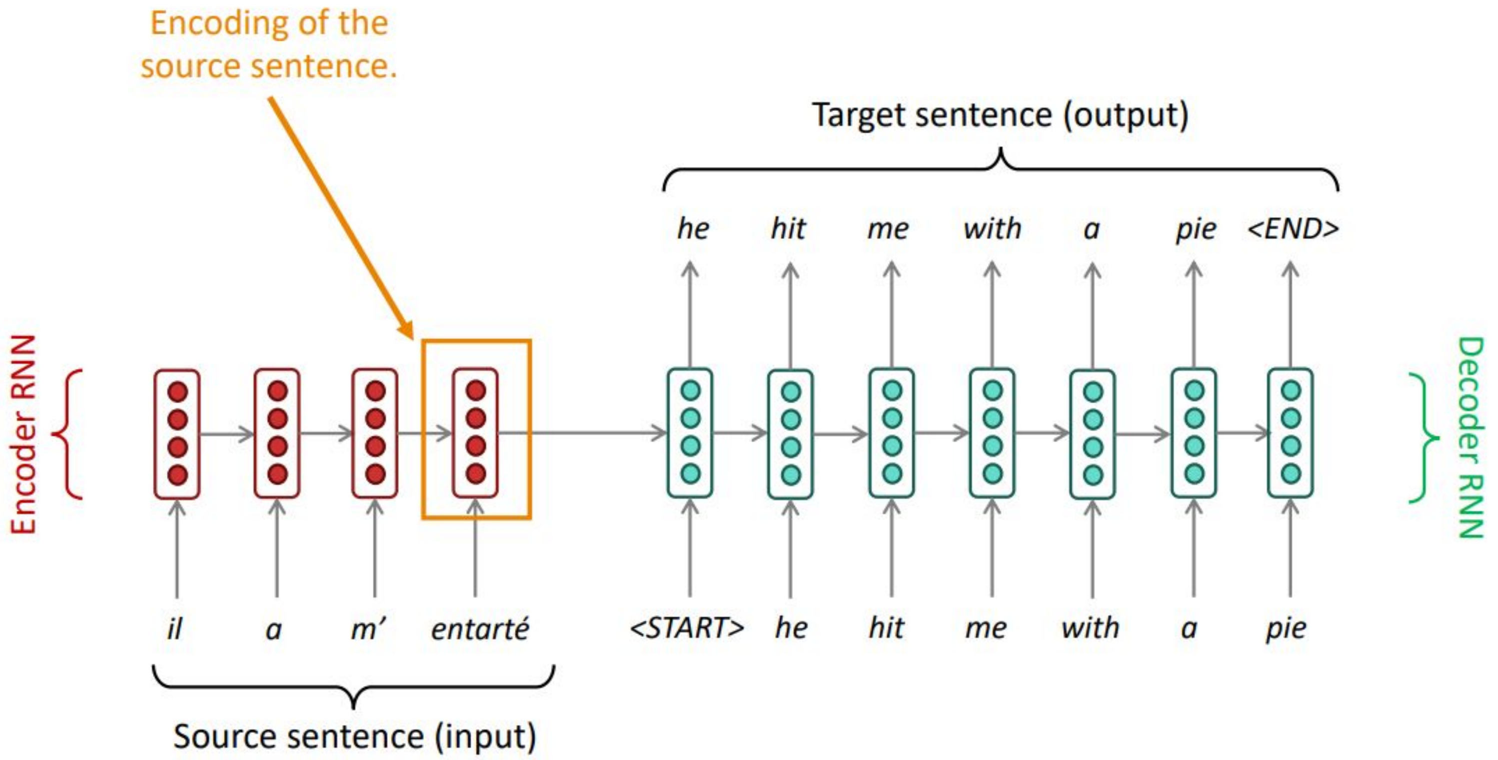
NMT research continues

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

- NMT research has pioneered many of the recent innovations of NLP Deep Learning
- NMT research continues to thrive
  - Researchers have found *many, many* improvements to the "vanilla" seq2seq NMT system we've just presented
  - But we'll present next one improvement so integral that it is the new vanilla…

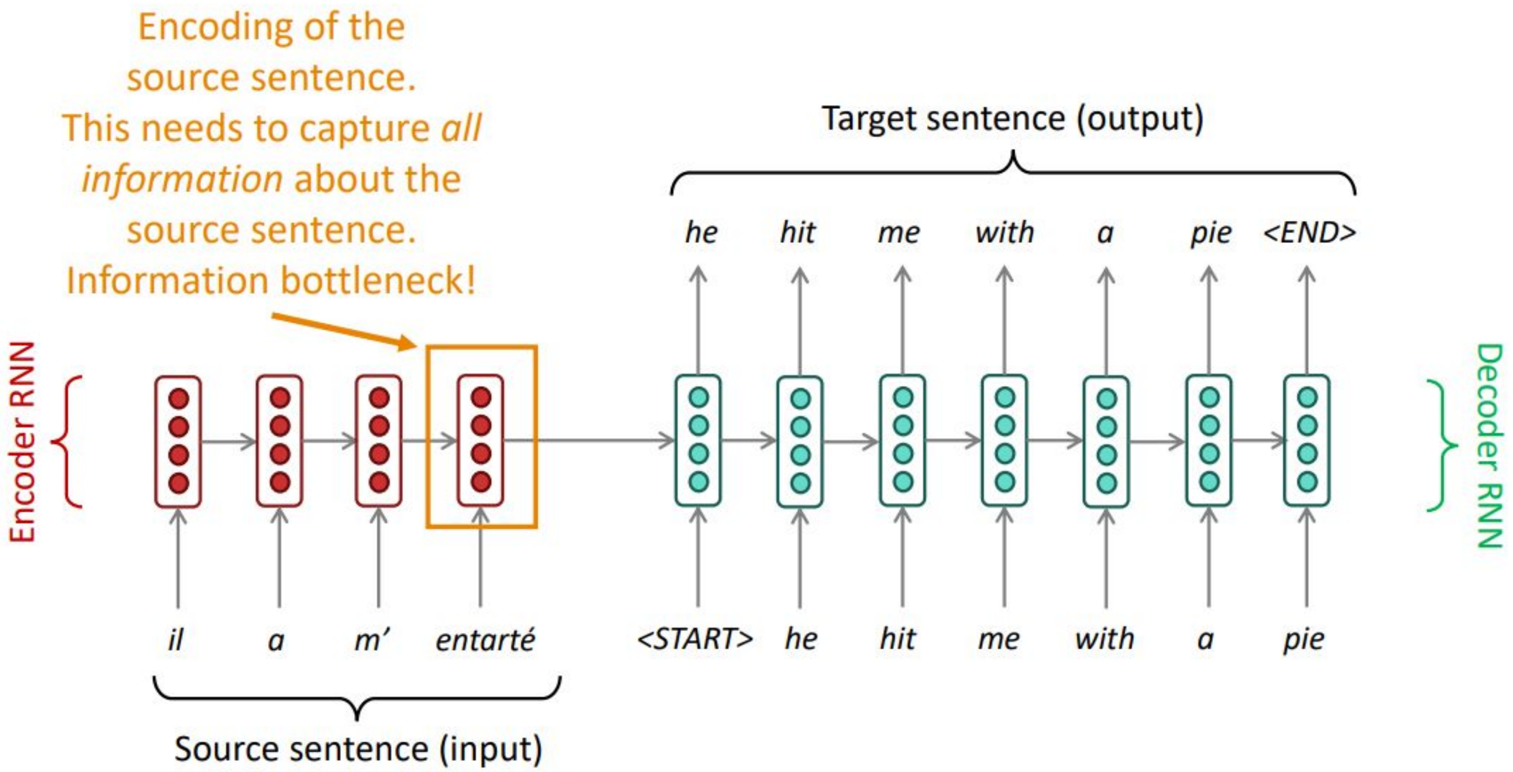
ATTENTION

1. Why attention? Sequence-to-sequence: the bottleneck problem



Problems with this architecture?

1. Why attention? Sequence-to-sequence: the bottleneck problem



Attention

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

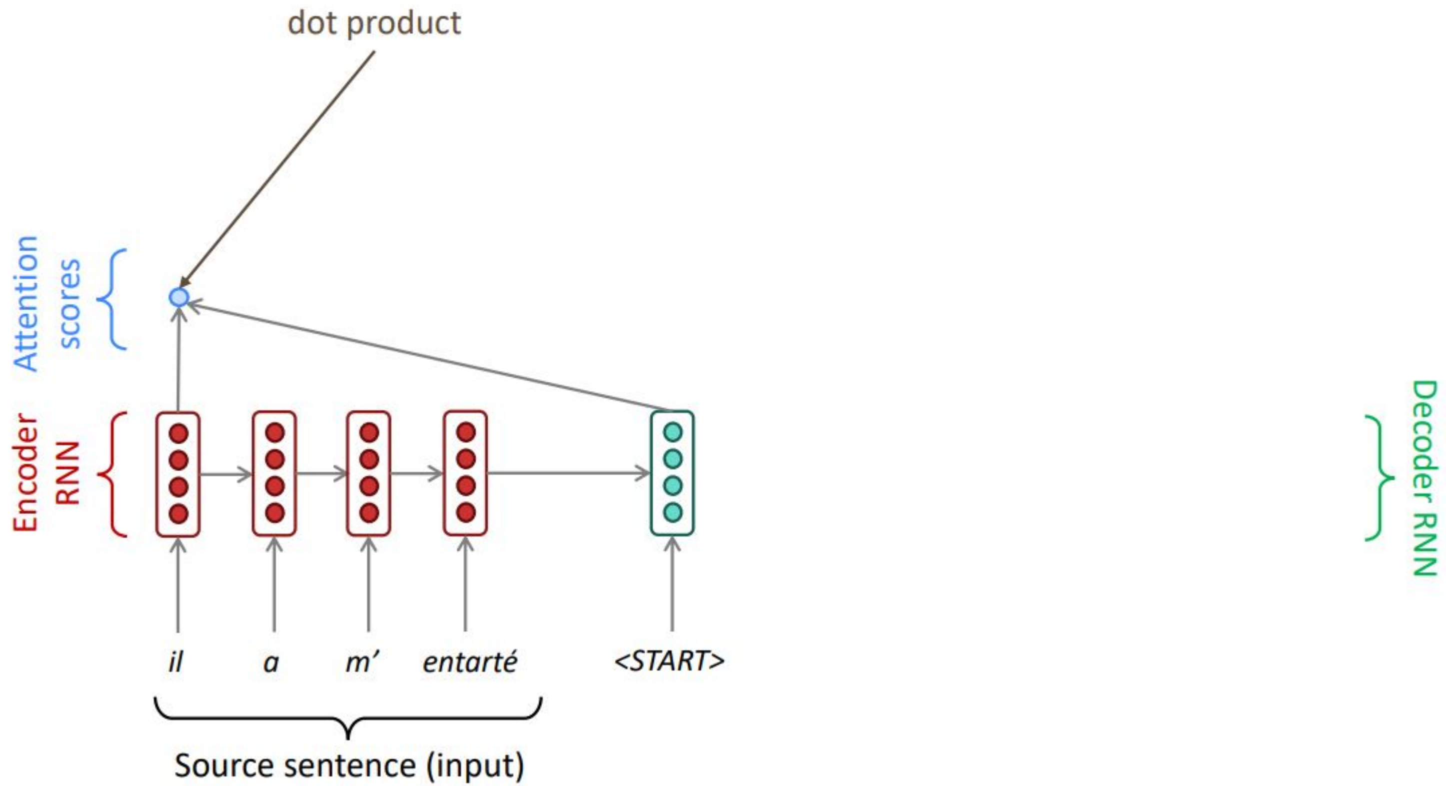
- Core idea: on each step of the decoder, *use direct connection to the encoder* to *focus on a particular part* of the source sequence

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

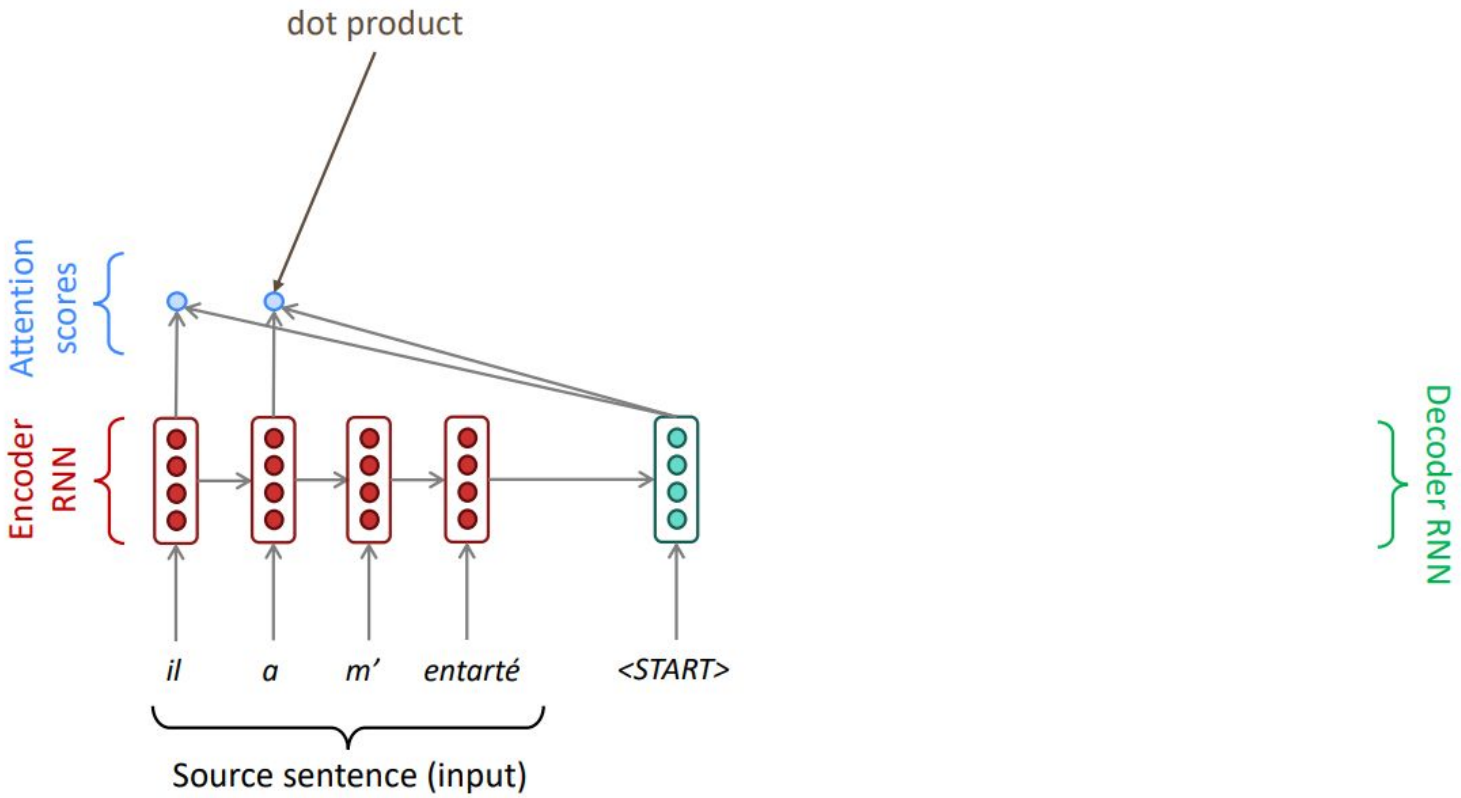
Attention:

At different steps, let a model "focus" on different parts of the input.

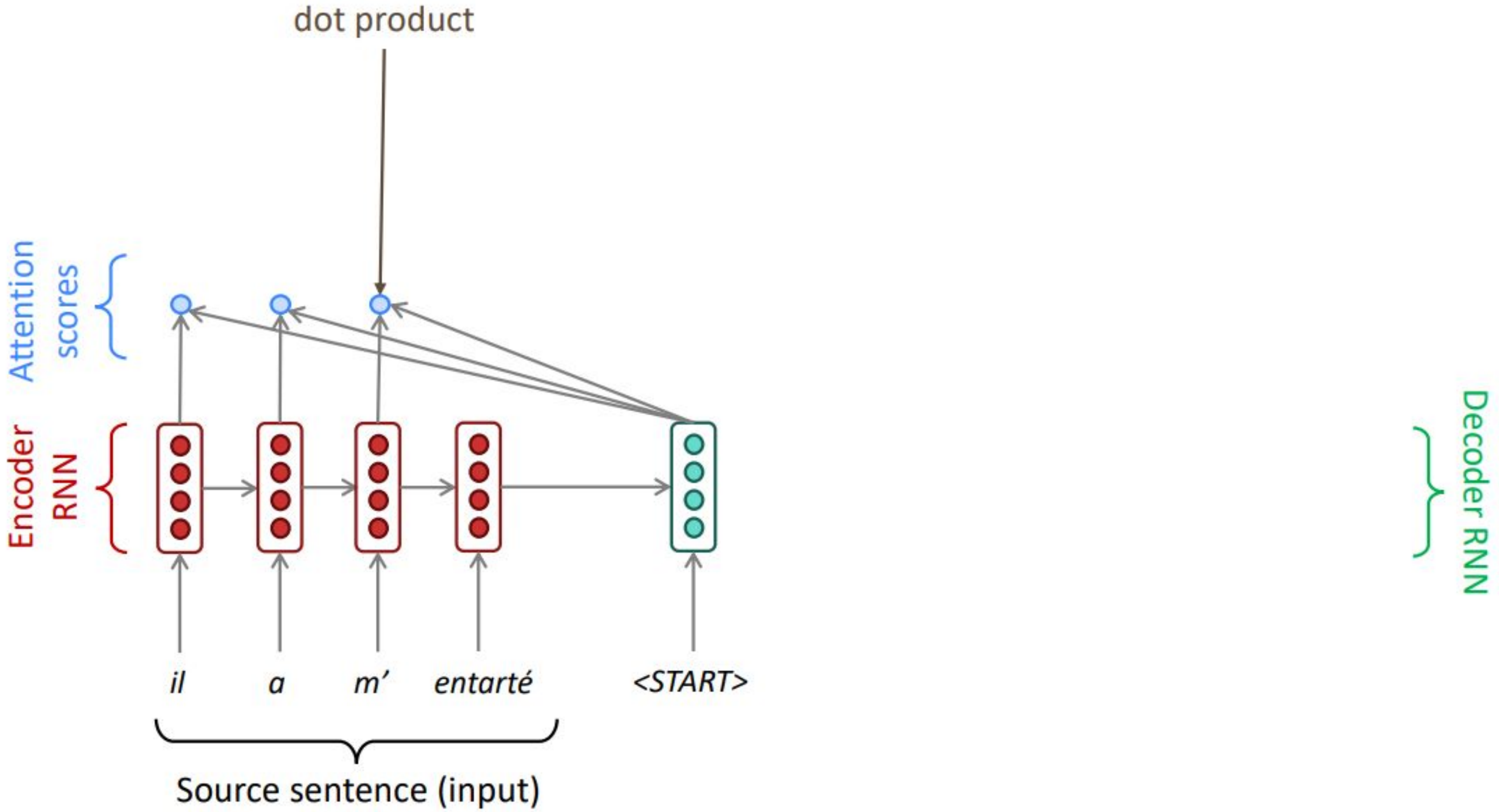
Sequence-to-sequence with attention



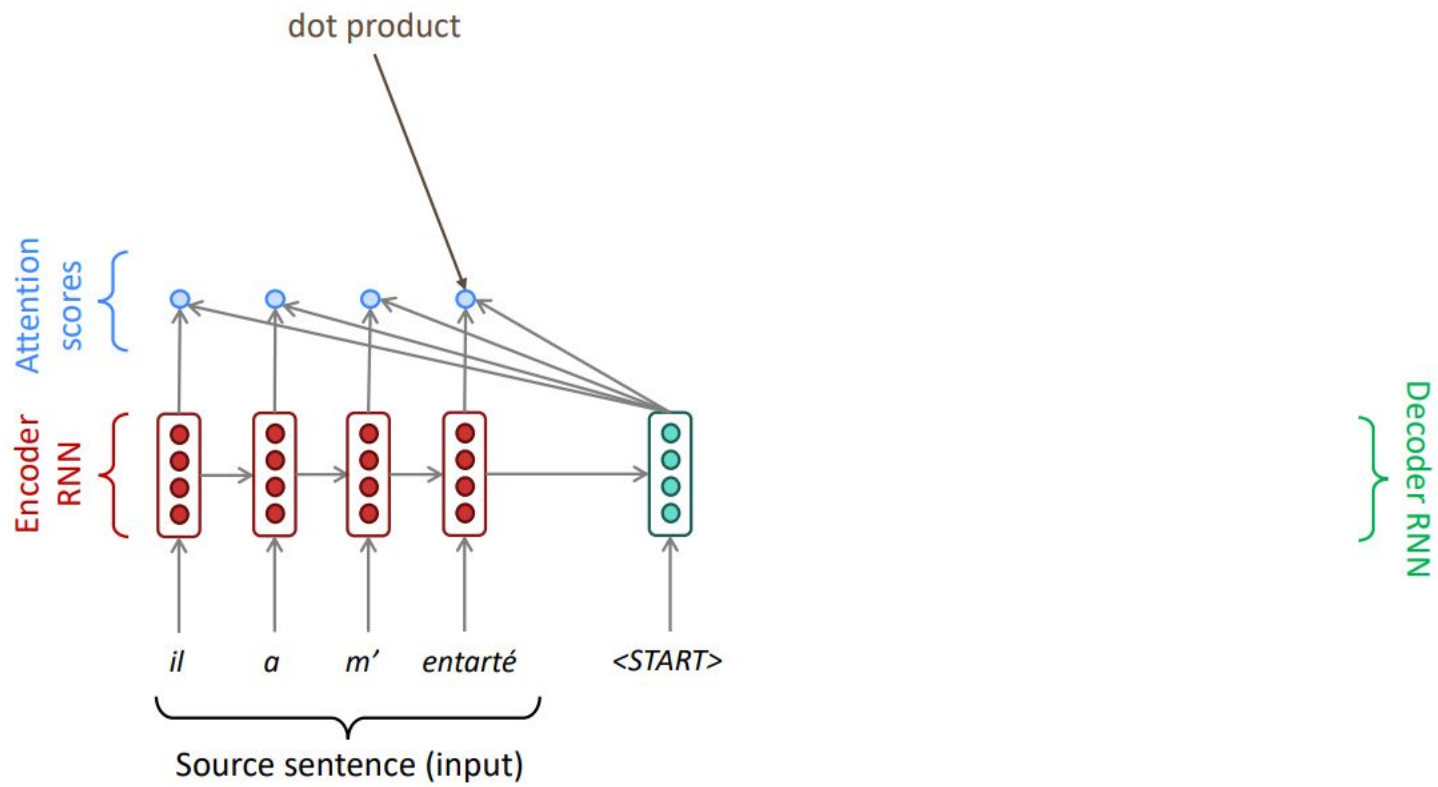
Sequence-to-sequence with attention



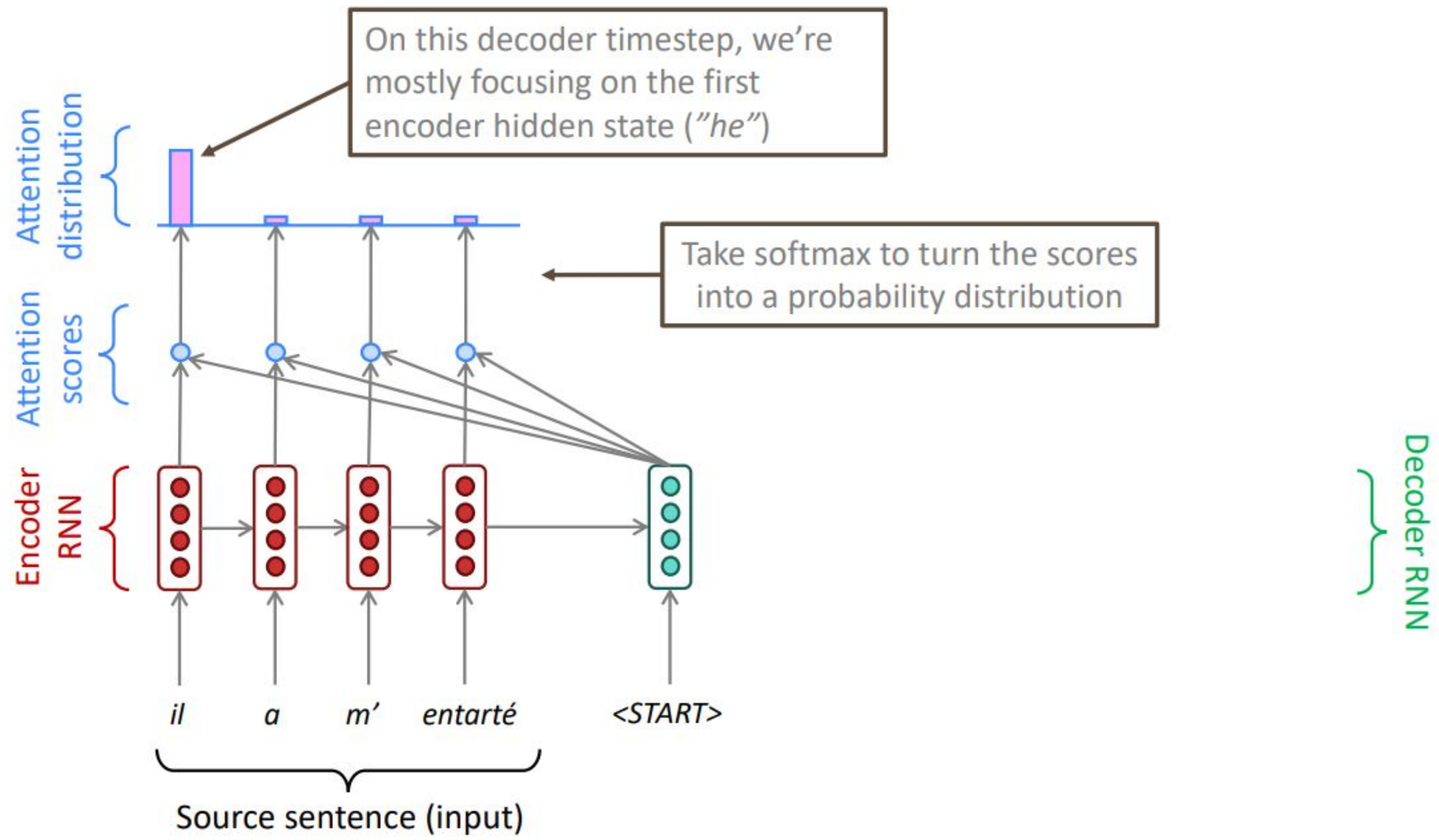
Sequence-to-sequence with attention



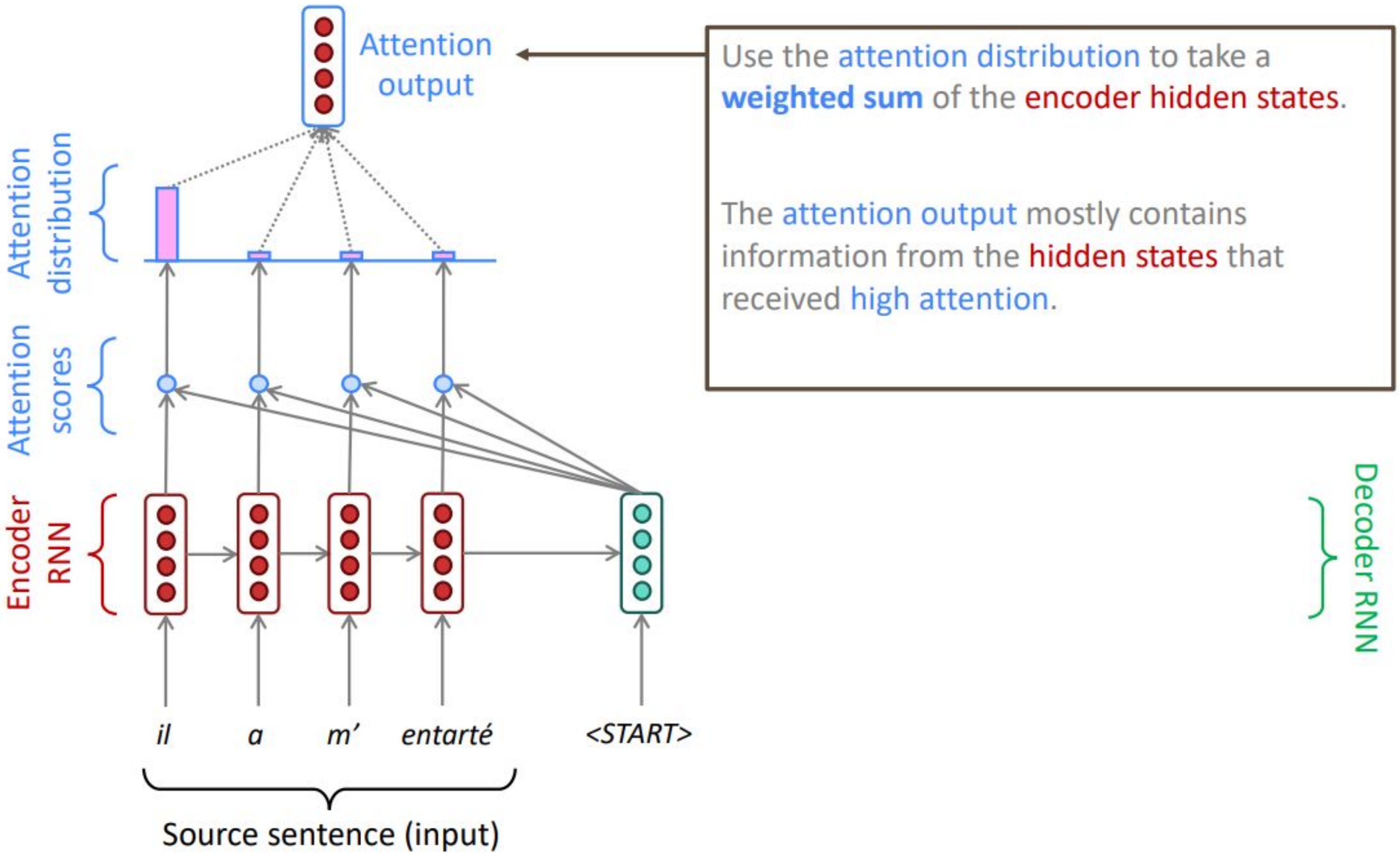
Sequence-to-sequence with attention



Sequence-to-sequence with attention



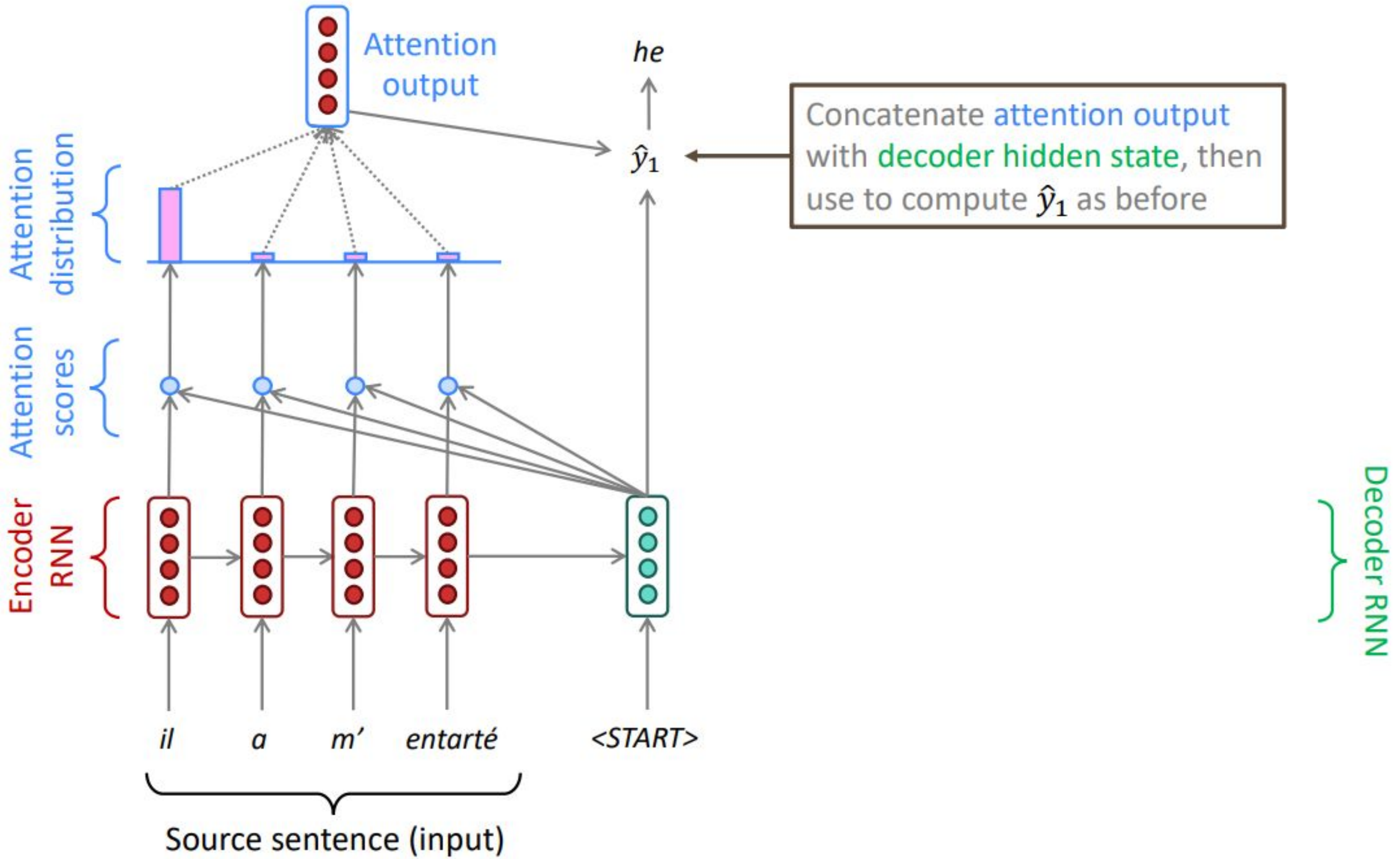
Sequence-to-sequence with attention



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

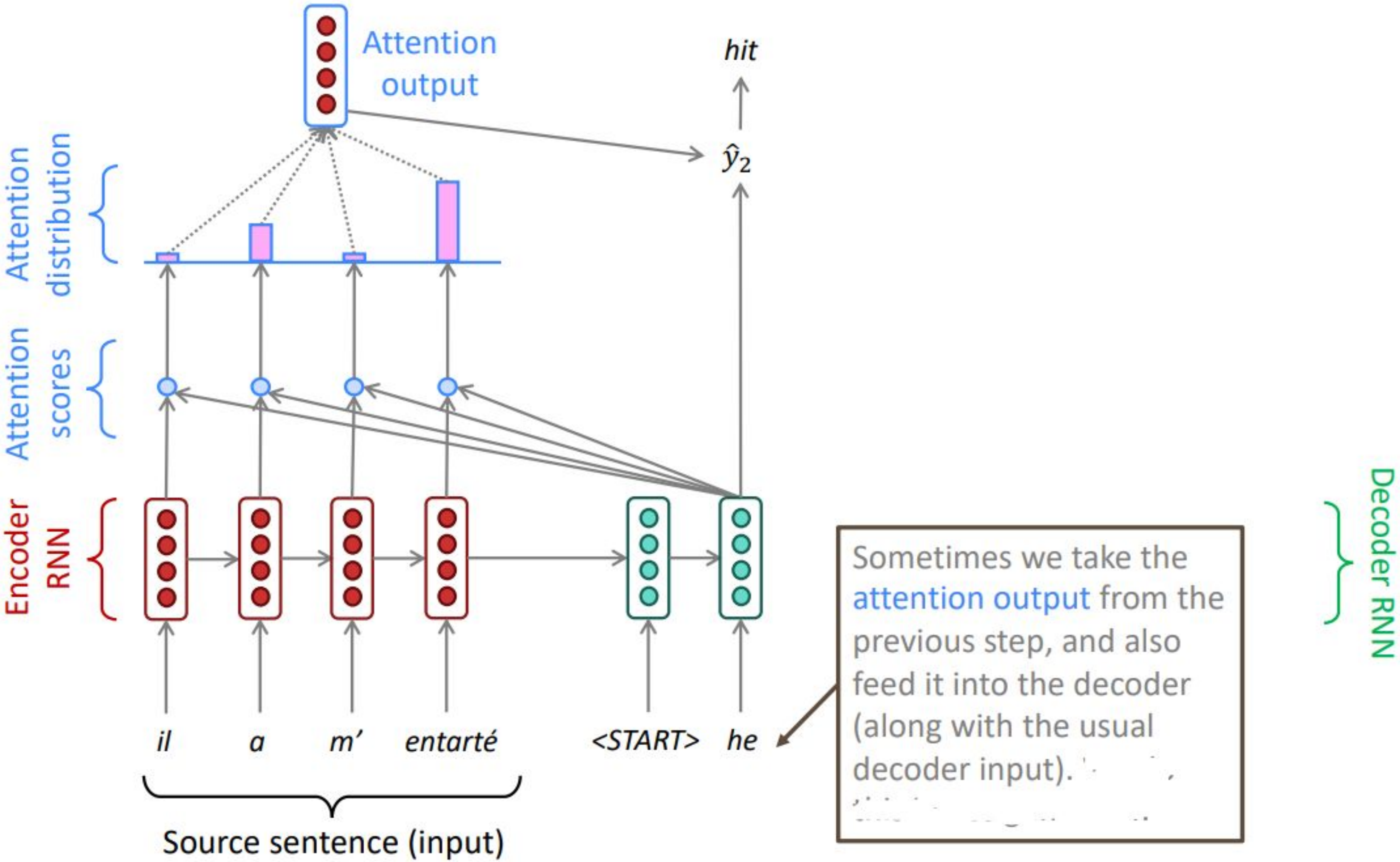
The attention output mostly contains information from the hidden states that received high attention.

Sequence-to-sequence with attention

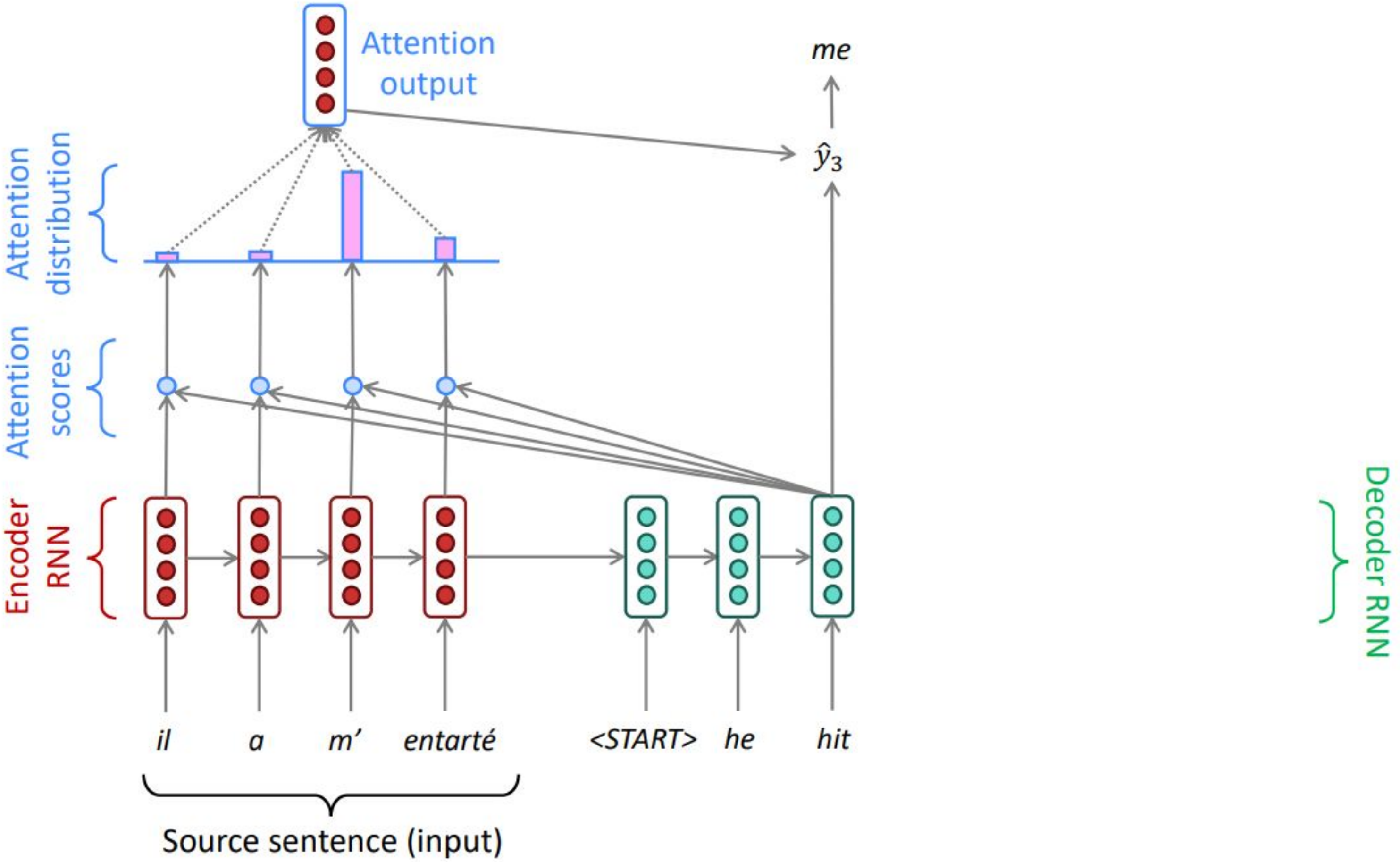


Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

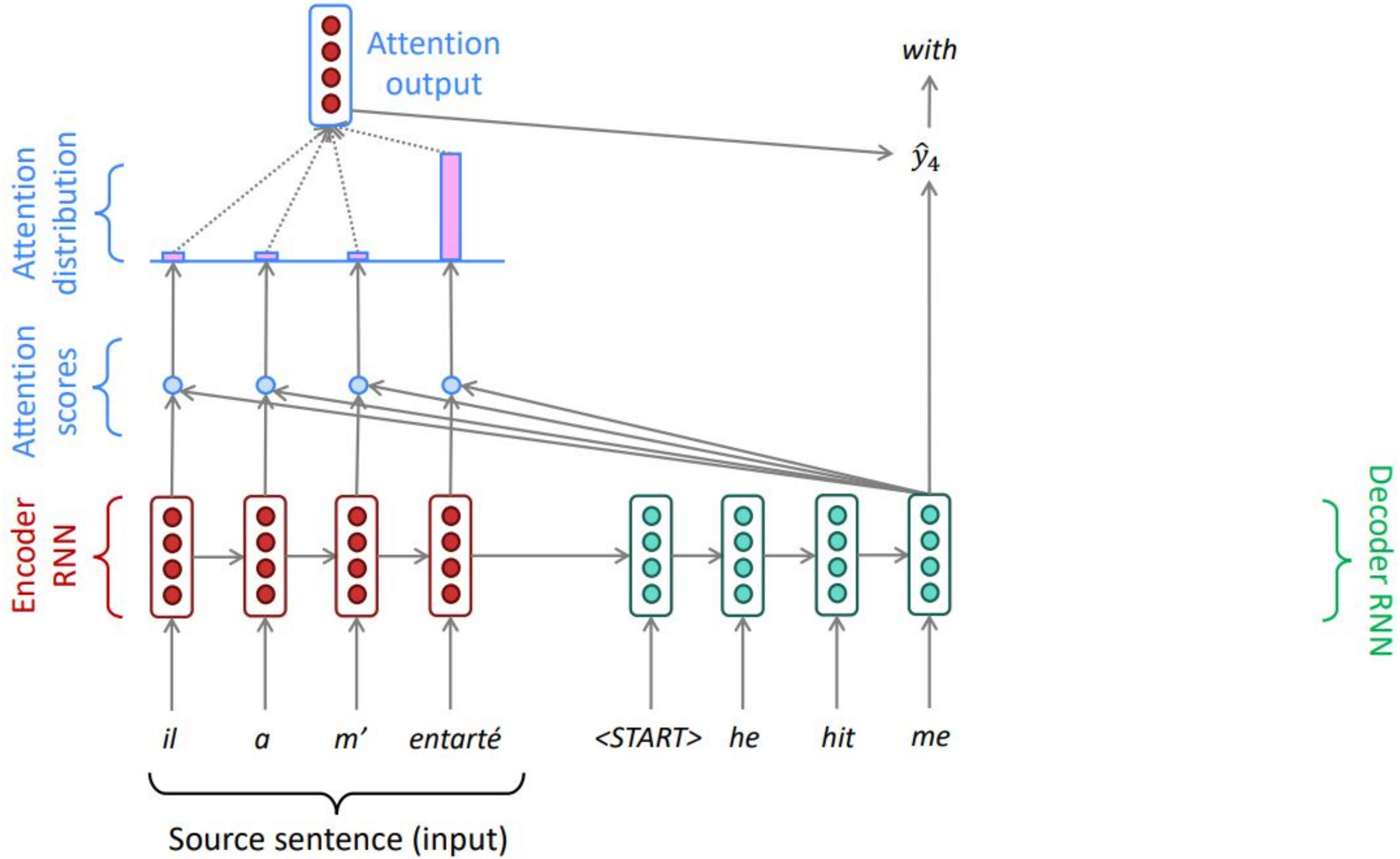
Sequence-to-sequence with attention



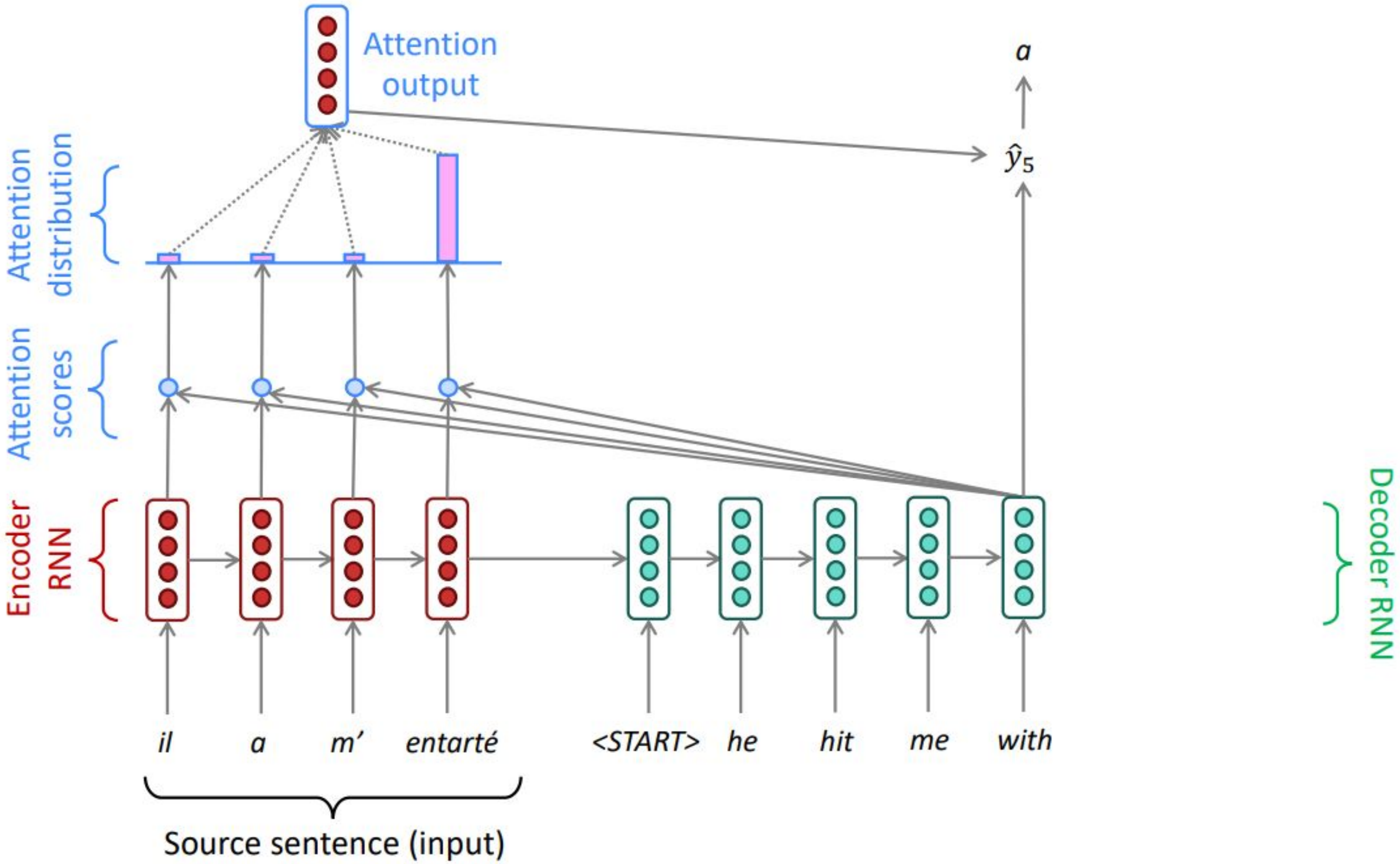
Sequence-to-sequence with attention



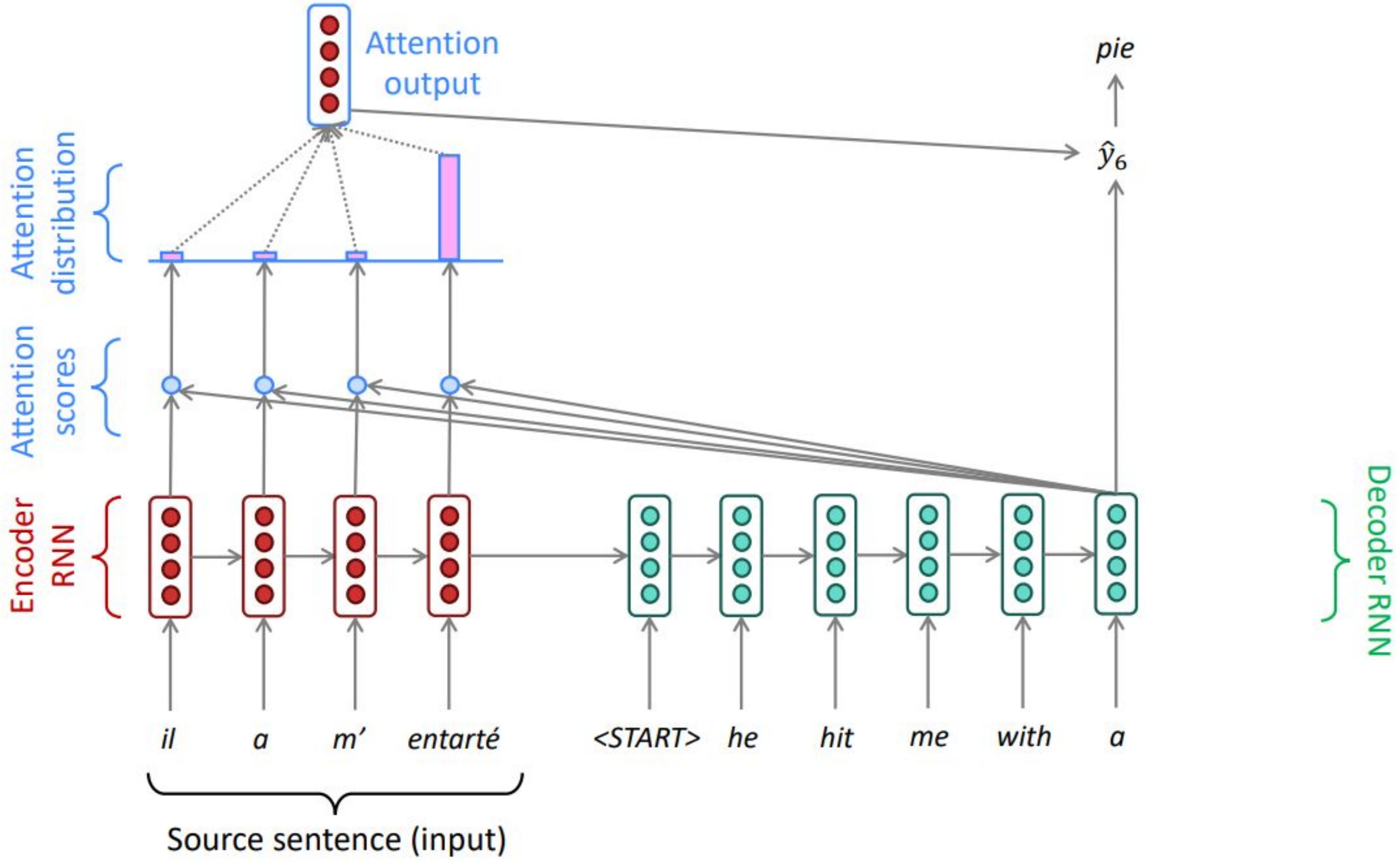
Sequence-to-sequence with attention



Sequence-to-sequence with attention



Sequence-to-sequence with attention



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)

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

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

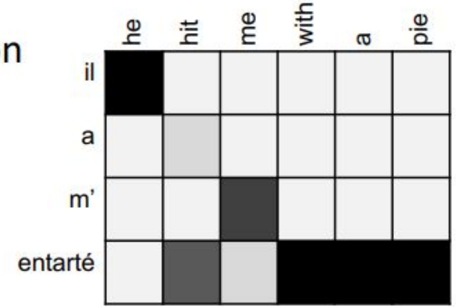
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

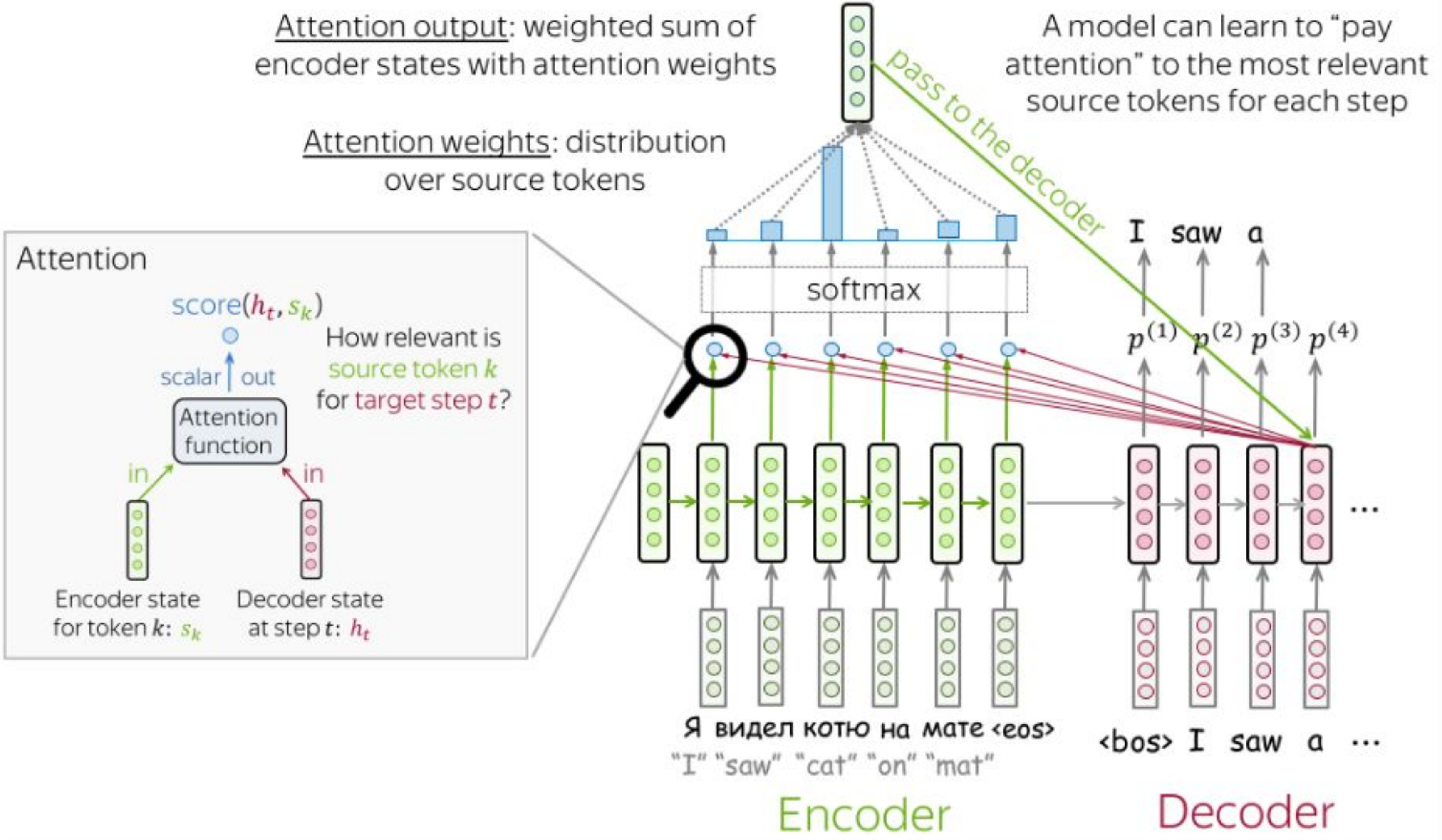
- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

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

Attention is great!

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention provides more "human-like" model of the MT process
  - You can look back at the source sentence while translating, rather than needing to remember it all
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with the vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we 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 output: weighted sum of encoder states with attention weights
A model can learn to "pay attention" to the most relevant source tokens for each step
pass to the decoder
Attention weights: distribution over source tokens
softmax
Attention
score(h_t, s_k)
scalar out
How relevant is source token k for target step t?
Attention function
in
in
Encoder state for token k: s_k
Decoder state at step t: h_t
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
...
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
Encoder
Decoder

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

(weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\mathrm{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\mathrm{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$$

"attention weight for source token k at decoder step t"

(softmax)

Attention scores

$$\mathrm{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$$

"How relevant is source token k for target step t?"

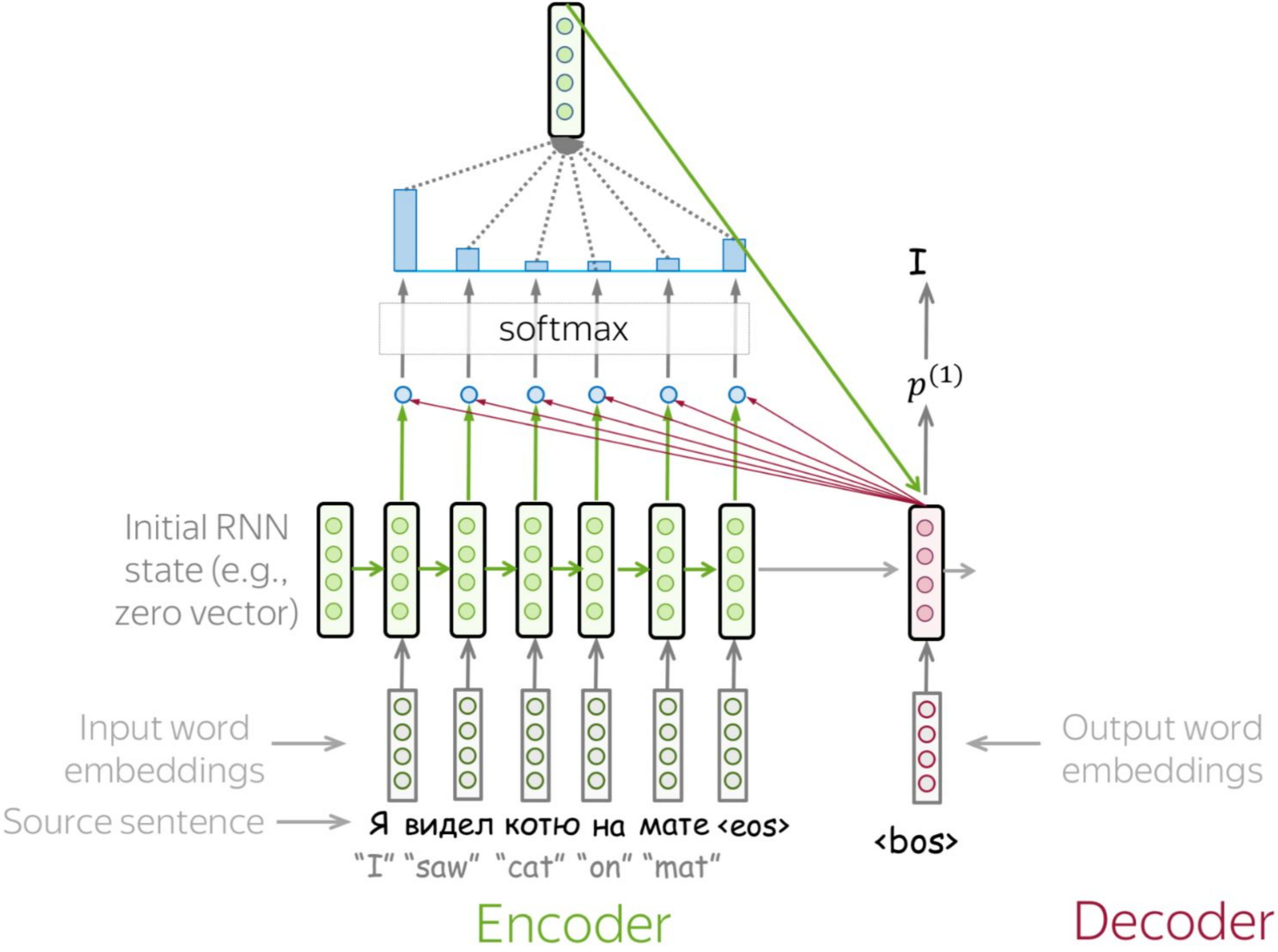
Attention input

$s_1, s_2, \ldots, s_m$

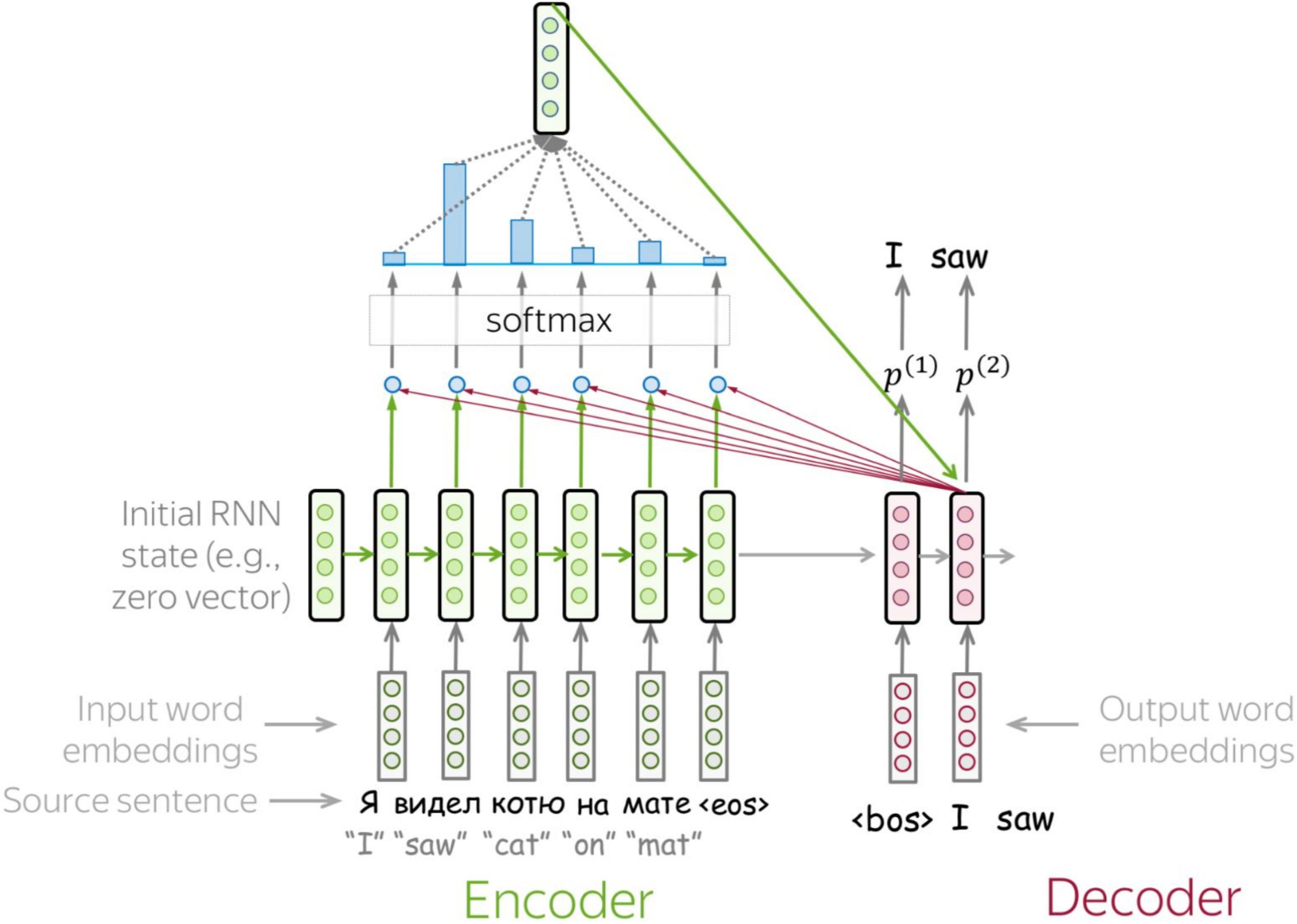
all encoder states

h_t

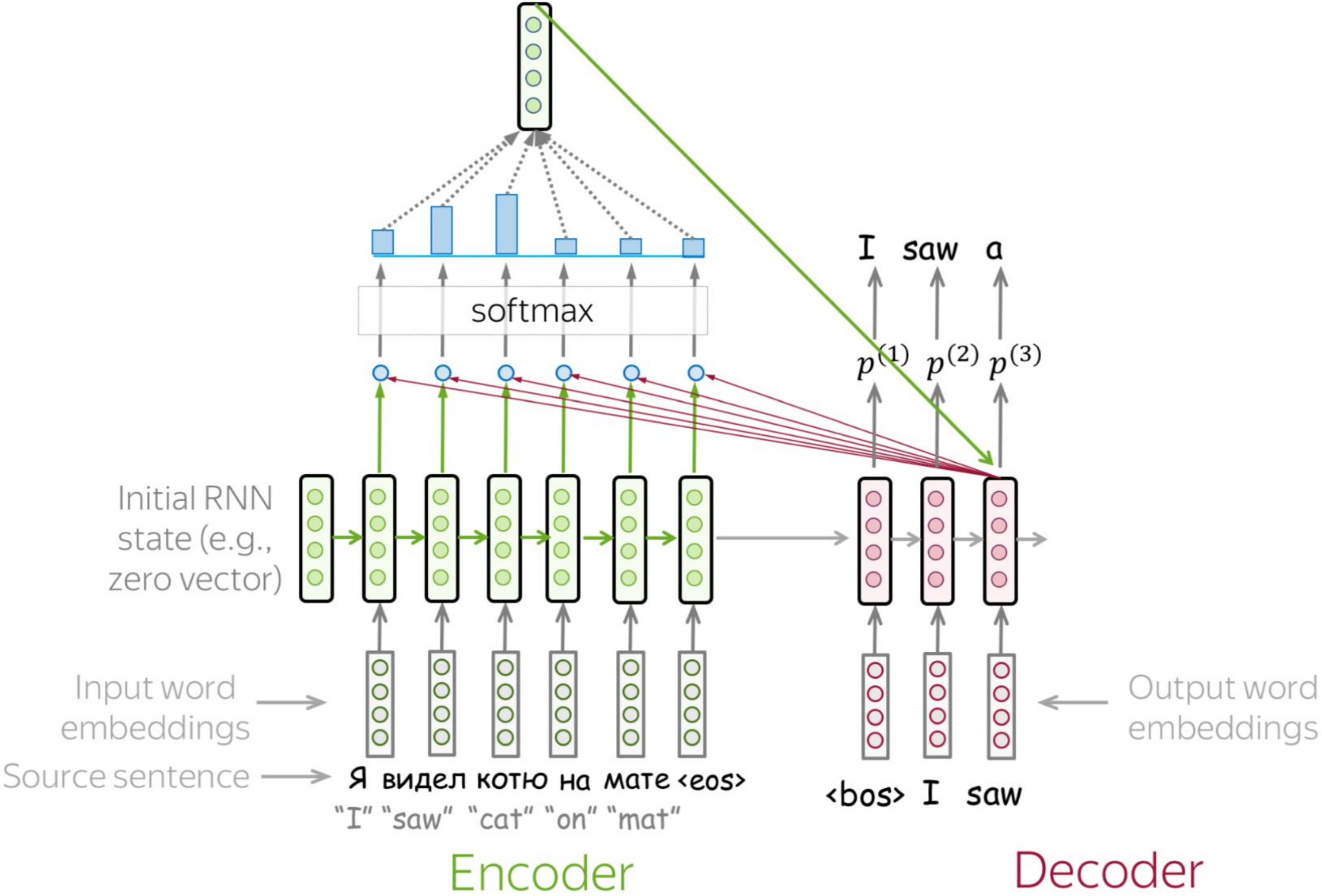
one decoder state

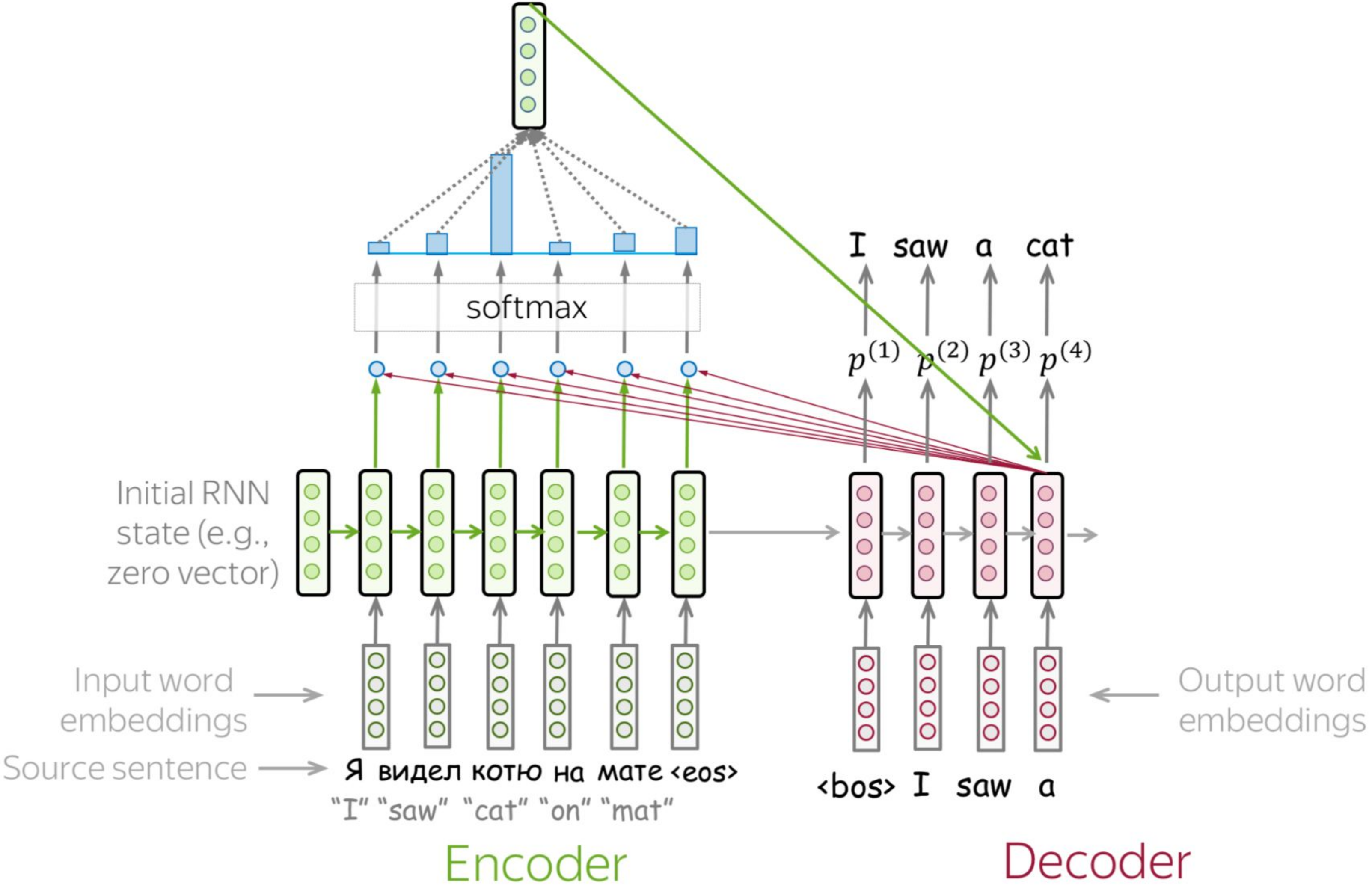


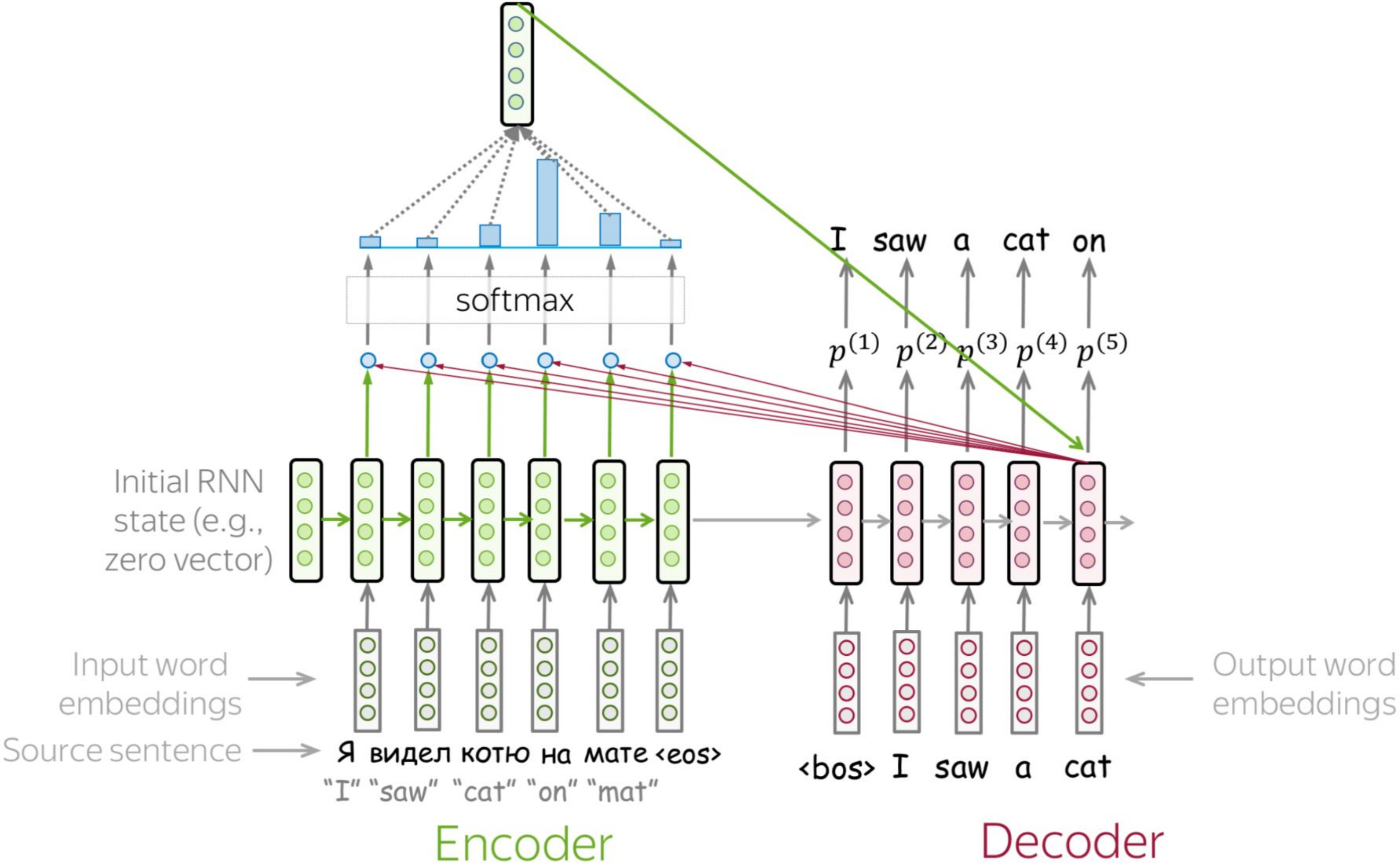
softmax
I
$p^{(1)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Output word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos>
Encoder
Decoder



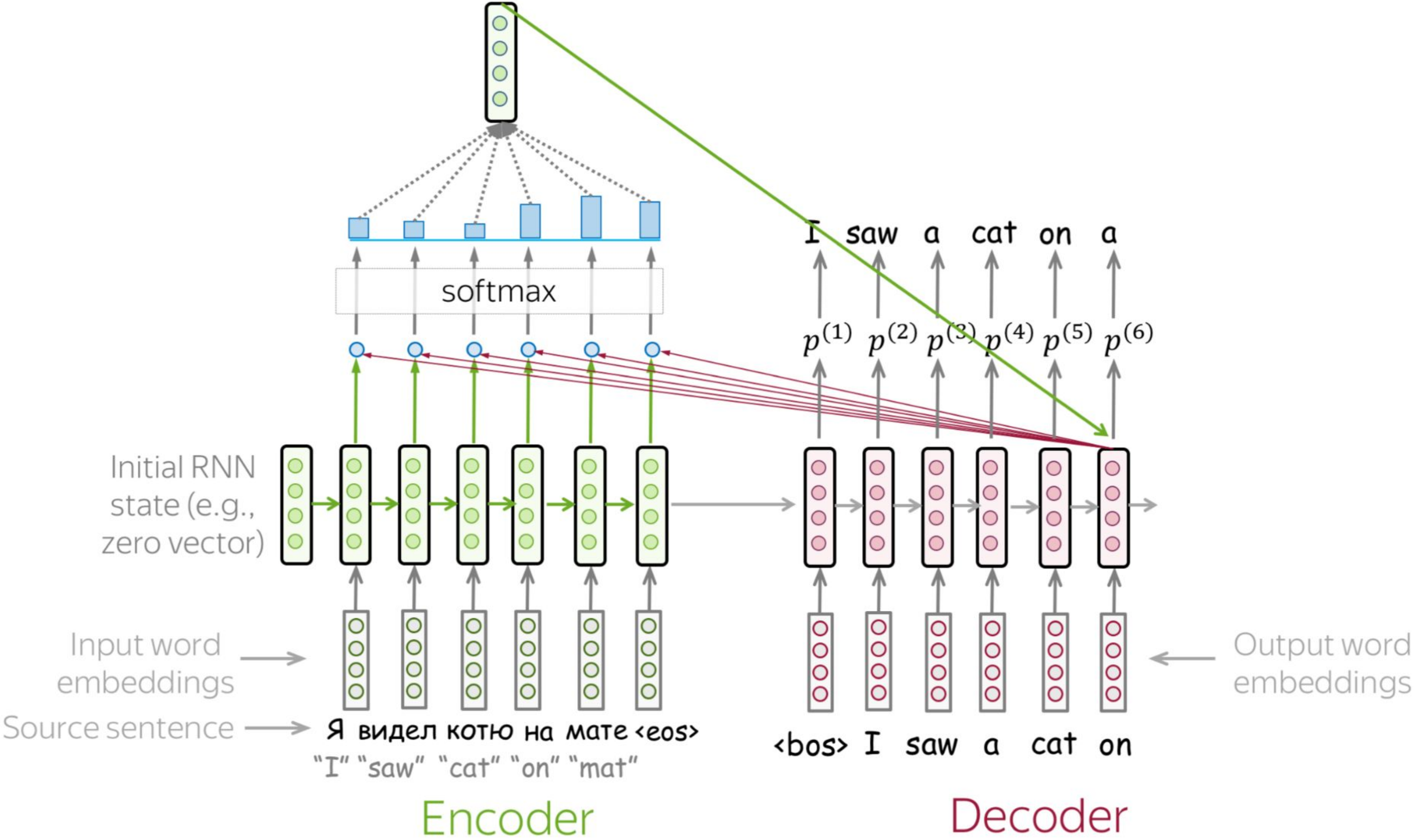
softmax
I saw
$p^{(1)}$ $p^{(2)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Output word embeddings
<bos> I saw
Encoder
Decoder


softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw
Output word embeddings
Encoder
Decoder

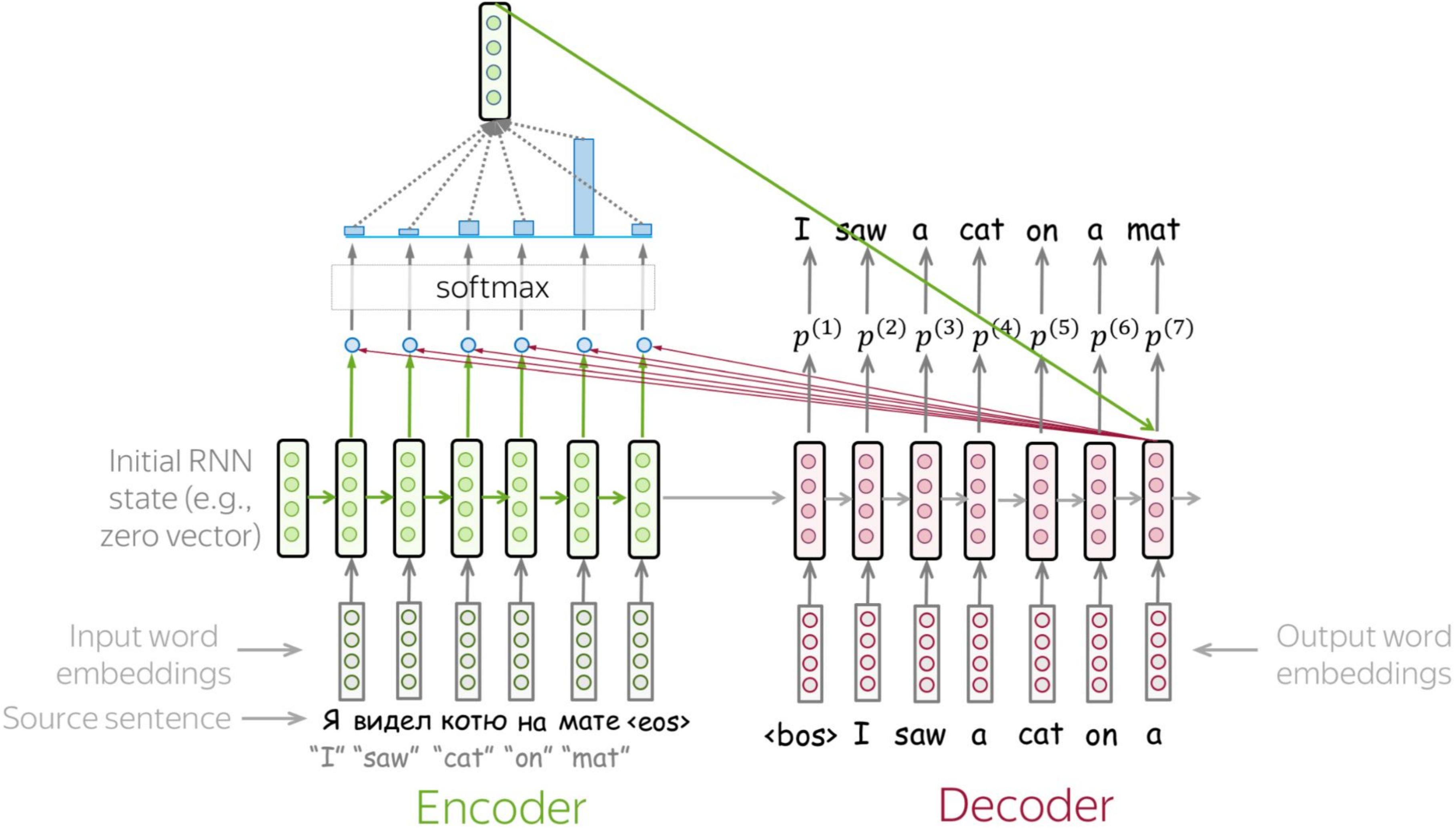

softmax
I saw a cat
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a
Output word embeddings
Encoder
Decoder

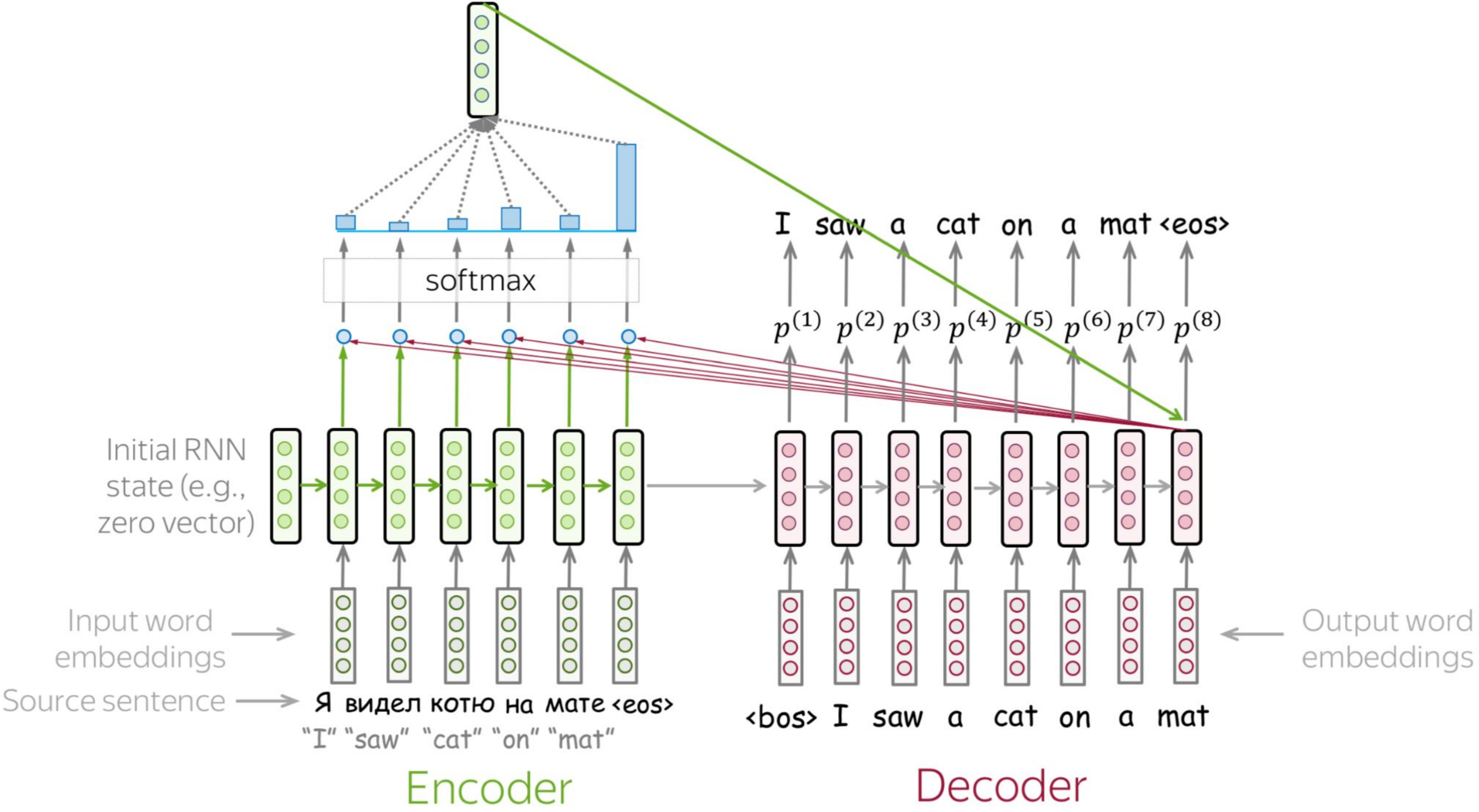


softmax
I saw a cat on
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$ $p^{(5)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a cat
Output word embeddings
Encoder
Decoder



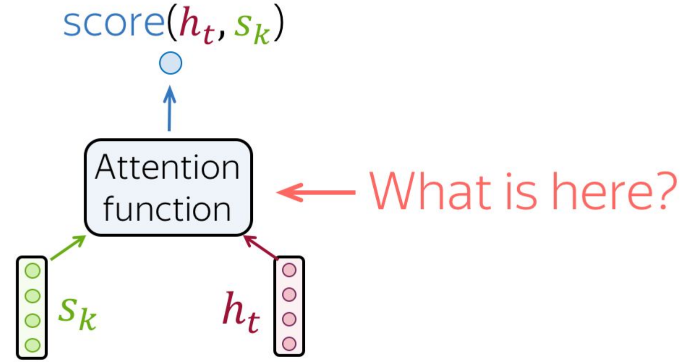
softmax
I saw a cat on a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$ $p^{(5)}$ $p^{(6)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Output word embeddings
<bos> I saw a cat on
Encoder
Decoder


softmax
I saw a cat on a mat
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$ $p^{(5)}$ $p^{(6)}$ $p^{(7)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Output word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a cat on a
Encoder
Decoder

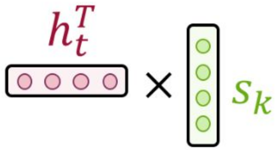


softmax
I saw a cat on a mat <eos>
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$ $p^{(5)}$ $p^{(6)}$ $p^{(7)}$ $p^{(8)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Output word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a cat on a mat
Encoder
Decoder

How to Compute Attention Score?

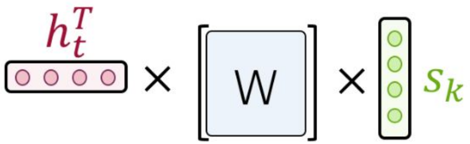


Dot-product



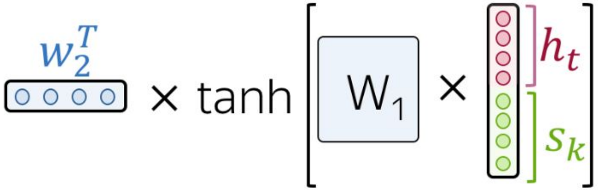
$$\text{score}(h_t, s_k) = h_t^T \, s_k$$

Bilinear



$$\text{score}(h_t, s_k) = h_t^T \, W s_k$$

Multi-Layer Perceptron

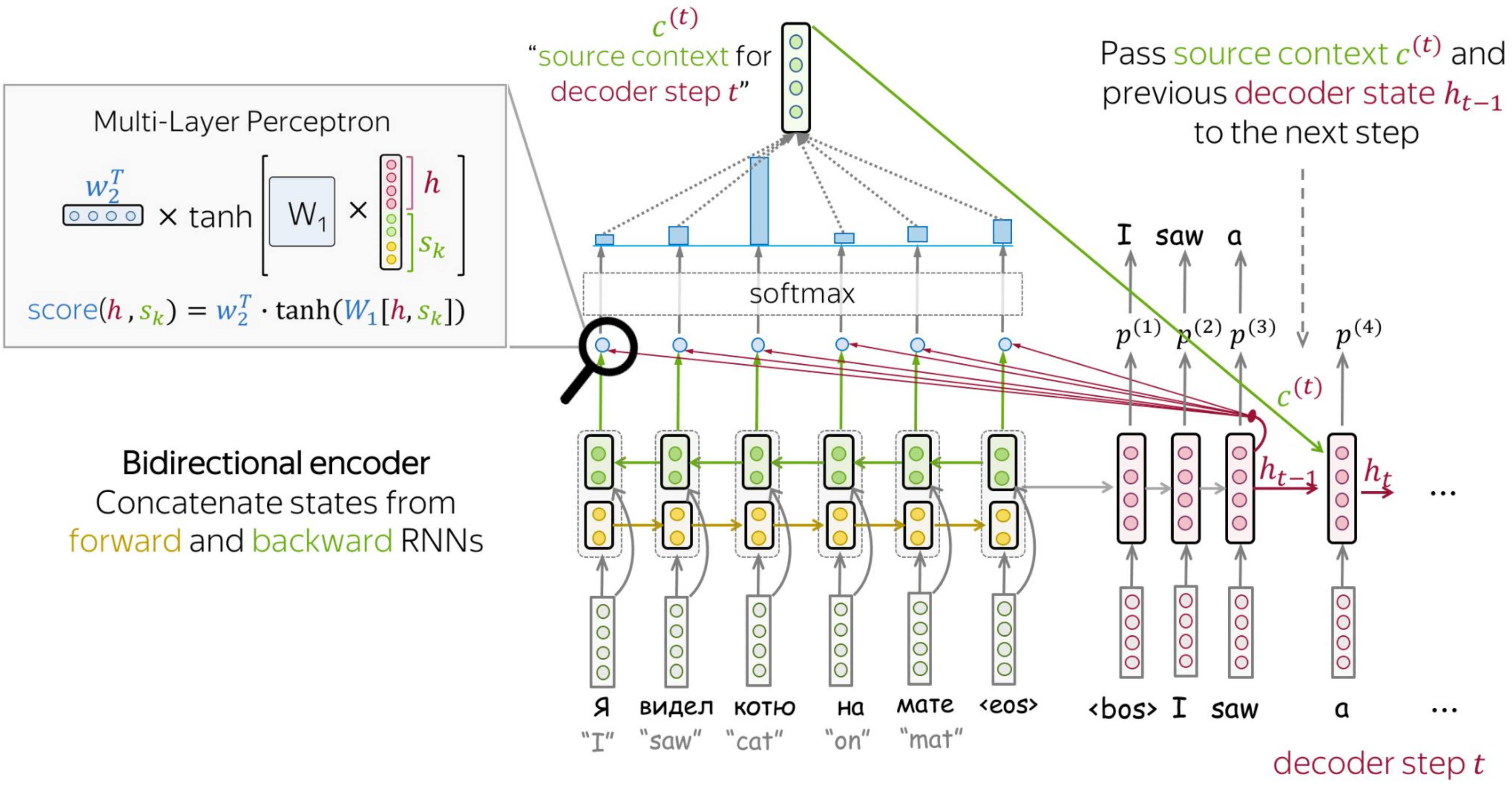


$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1[h_t, s_k])$$

How to Compute Attention Score?

The most popular ways to compute attention scores are:

- dot-product - the simplest method
- bilinear function (aka "Luong attention") - used in the paper Effective Approaches to Attention-based Neural Machine Translation
- multi-layer perceptron (aka "Bahdanau attention") - the method proposed in the original paper (NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE).



$c^{(t)}$
“source context for decoder step t”
Pass source context $c^{(t)}$ and previous decoder state h_{t-1} to the next step
Multi-Layer Perceptron
$w_2^T \times \tanh \left[W_1 \times \begin{bmatrix} h \\ s_k \end{bmatrix} \right]$
$\text{score}(h, s_k) = w_2^T \cdot \tanh(W_1[h, s_k])$
softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
$c^{(t)}$
h_{t-1}
h_t
...
Bidirectional encoder
Concatenate states from forward and backward RNNs
Я видел котю на мате <eos>
“I” “saw” “cat” “on” “mat”
<bos> I saw a ...
decoder step t

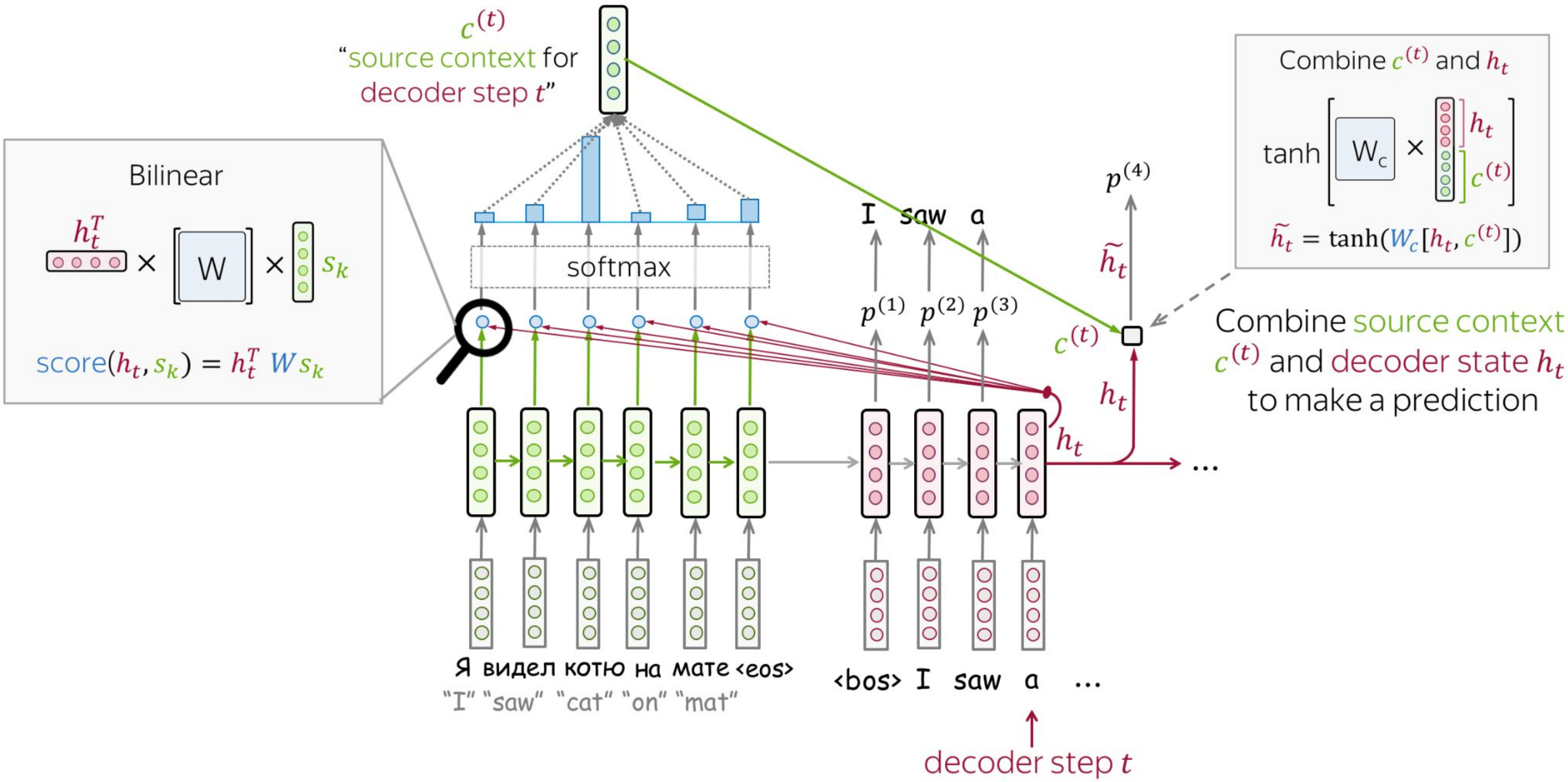
Bahdanau Model

encoder: bidirectional

To better encode each source word, the encoder has two RNNs, forward and backward, which read input in the opposite directions. For each token, states of the two RNNs are concatenated.

attention score: multi-layer perceptron

To get an attention score, apply a multi-layer perceptron (MLP) to an encoder state and a decoder state.



$c^{(t)}$
"source context for decoder step t"
Bilinear
$h_t^T \times [W] \times s_k$
score$(h_t, s_k) = h_t^T\, W s_k$
softmax
Combine $c^{(t)}$ and h_t
tanh $\left[W_c \times \begin{bmatrix} h_t \\ c^{(t)} \end{bmatrix} \right]$
$\tilde{h}_t = \tanh(W_c[h_t, c^{(t)}])$
$p^{(4)}$
$\tilde{h}_t$
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
$c^{(t)}$
Combine source context $c^{(t)}$ and decoder state h_t to make a prediction
h_t
h_t
...
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
decoder step t

Luong Model

encoder: bidirectional

To better encode each source word, the encoder has two RNNs, forward and backward, which read input in the opposite directions. For each token, states of the two RNNs are concatenated.

attention score: multi-layer perceptron

To get an attention score, apply a multi-layer perceptron (MLP) to an encoder state and a decoder state.

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:

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

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* ***a*** (sometimes called the *context vector*)

Attention is a *general* Deep Learning technique

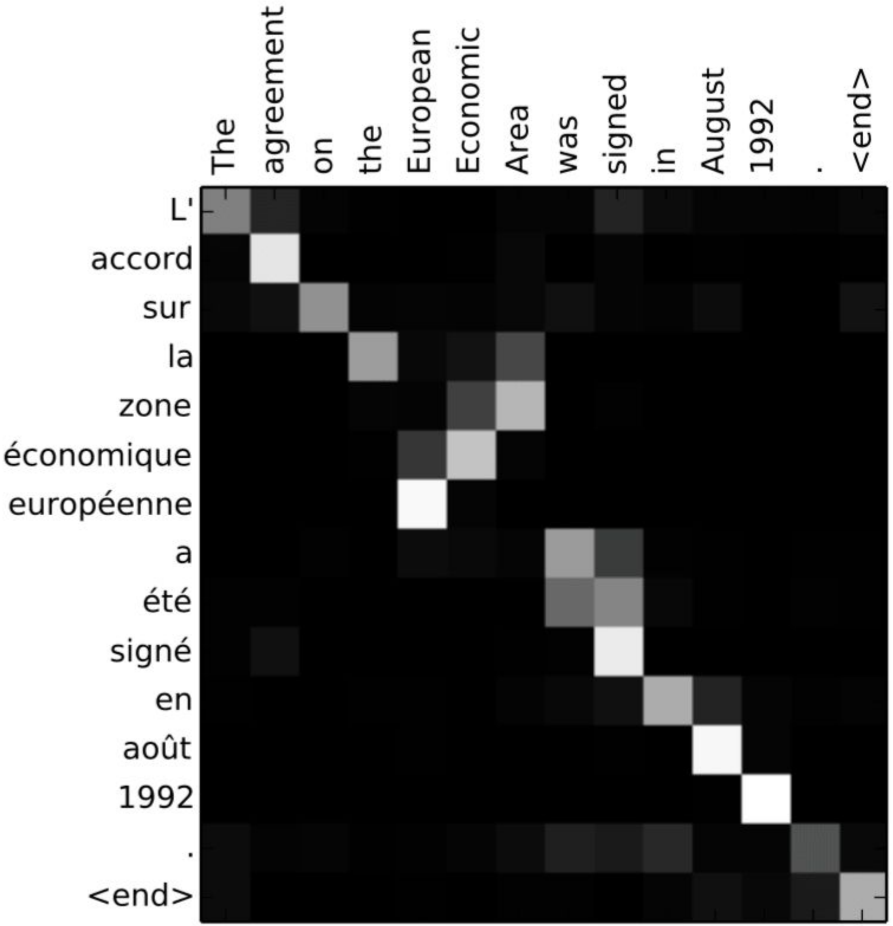
- More general definition of attention:
  - Given a set of vector *values*, and a vector *query*, attention is a technique to compute a weighted sum of the values, dependent on the query.

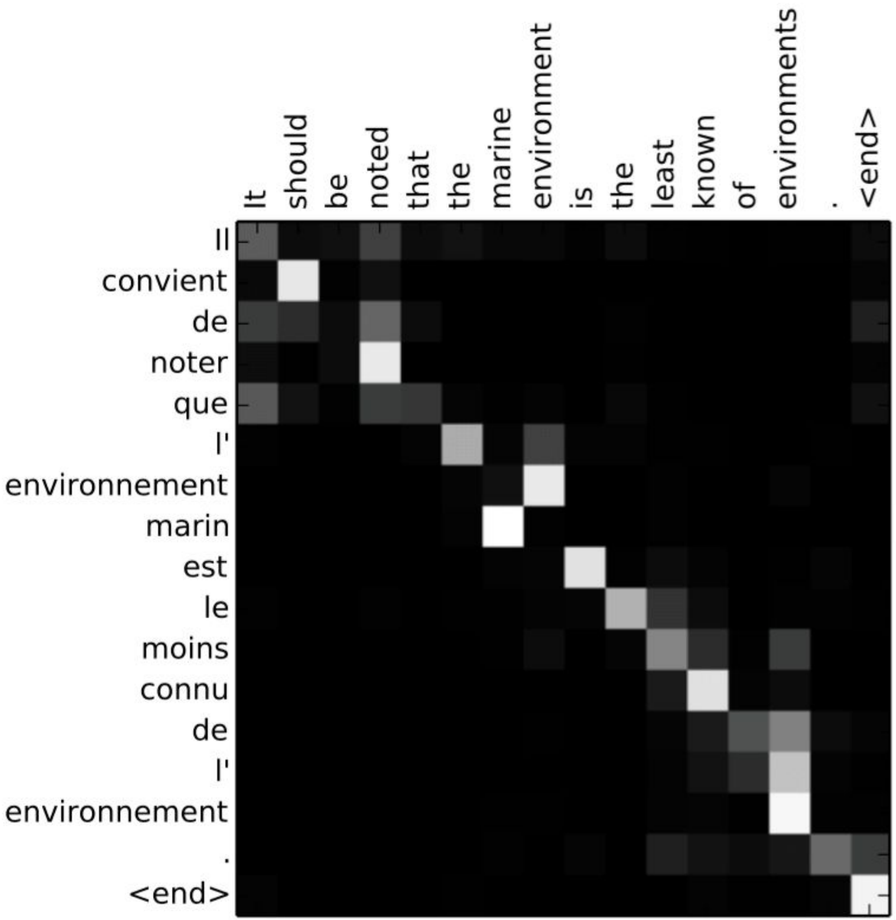
Intuition:

- 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).

Upshot:

- Attention has become the powerful, flexible, general way pointer and memory manipulation in all deep learning models. A new idea from after 2010! From NMT!


The
agreement
on
the
European
Economic
Area
was
signed
in
August
1992
.
<end>
L'
accord
sur
la
zone
économique
européenne
a
été
signé
en
août
1992
.
<end>


It
should
be
noted
that
the
marine
environment
is
the
least
known
of
environments
.
<end>
Il
convient
de
noter
que
l'
environnement
marin
est
le
moins
connu
de
l'
environnement
.
<end>

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


ATTENTION IS ALL YOU NEED?
FALSE. YOU NEED WATER AND RATIONS.
imgflip.com