INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

IIIT Hyderabad

Neural Attention

Makarand Tapaswi

CS7.505 Spring 2024

14th February 2024

Attention is All You Need!

- Almost there ... but, before that

- Neural Machine Translation by Jointly Learning to Align and Translate
- Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio
- ICLR 2015 Oral
- https://arxiv.org/abs/1409.0473

Recurrent Neural Networks

$$h_t = f(h_{t-1}, x_t)$$

- Why recurrent?

- Different variants of function f (GRU, LSTM, etc.)

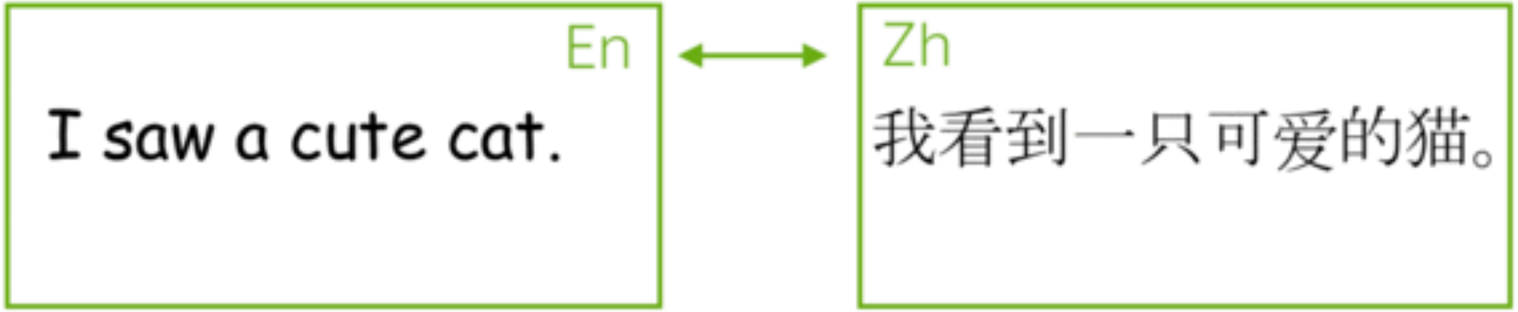
- Long Short-Term Memory Units
- Gated Recurrent Units

Examples of Sequence-to-Sequence tasks?

- Image Captioning
    - (strictly: image input to sequence)
- Sentiment Classification
    - (strictly: sequence to single output)
- Machine Translation
    - yes! sequence to sequence

Very nice material that we will follow!

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



Formulating machine translation

<u>Human Translation</u>

$$y^* = \arg\max_y p(y|x)$$

The "probability" is intuitive and is given by a human translator's expertise

<u>Machine Translation</u>

model parameters

$$y' = \arg\max_y p(y|x, \theta)$$

Questions we need to answer

- modeling

How does the model for $p(y|x, \theta)$ look like?

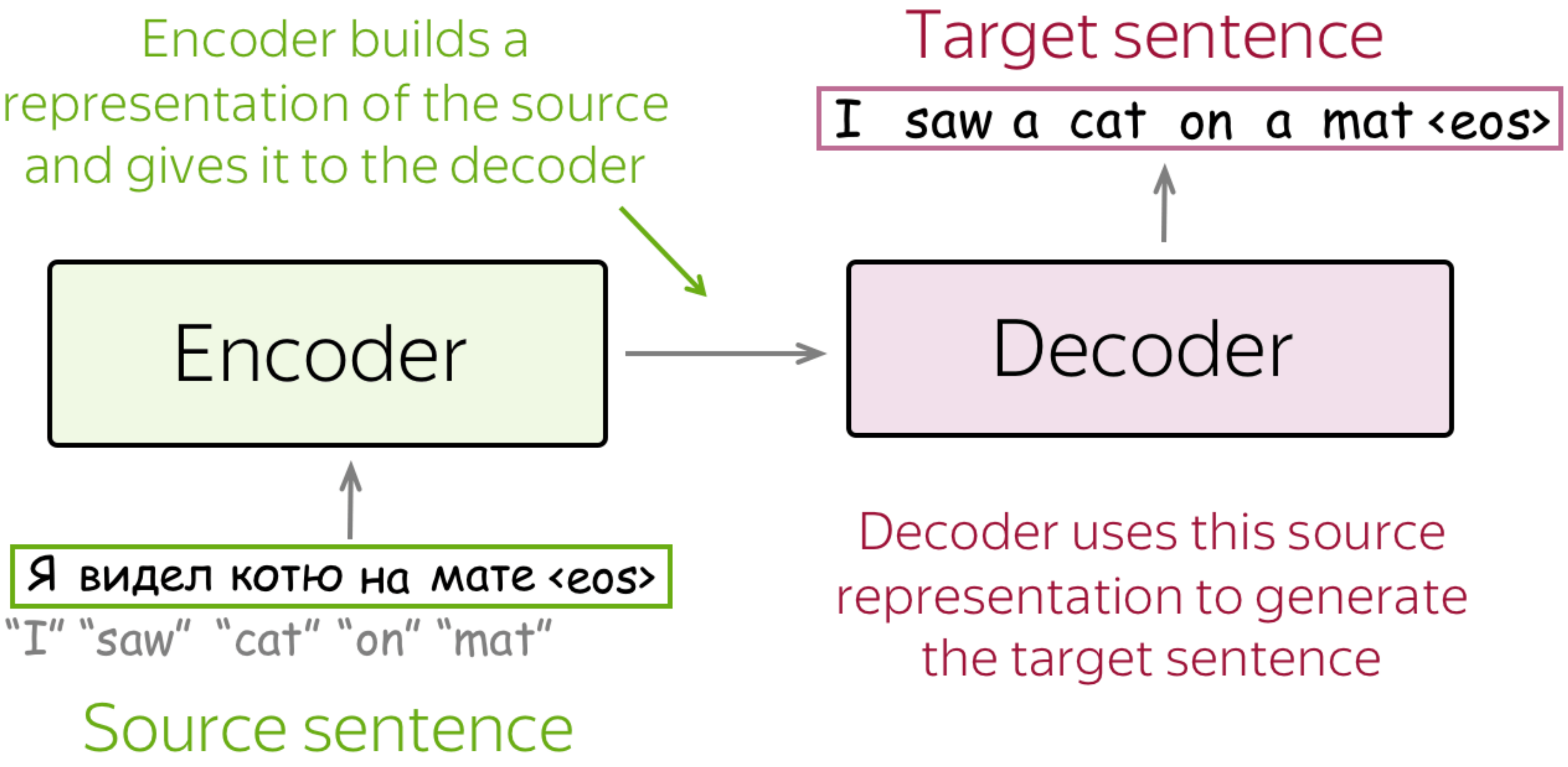
- learning

How to find θ?

- search

How to find the argmax?

Encoder - decoder



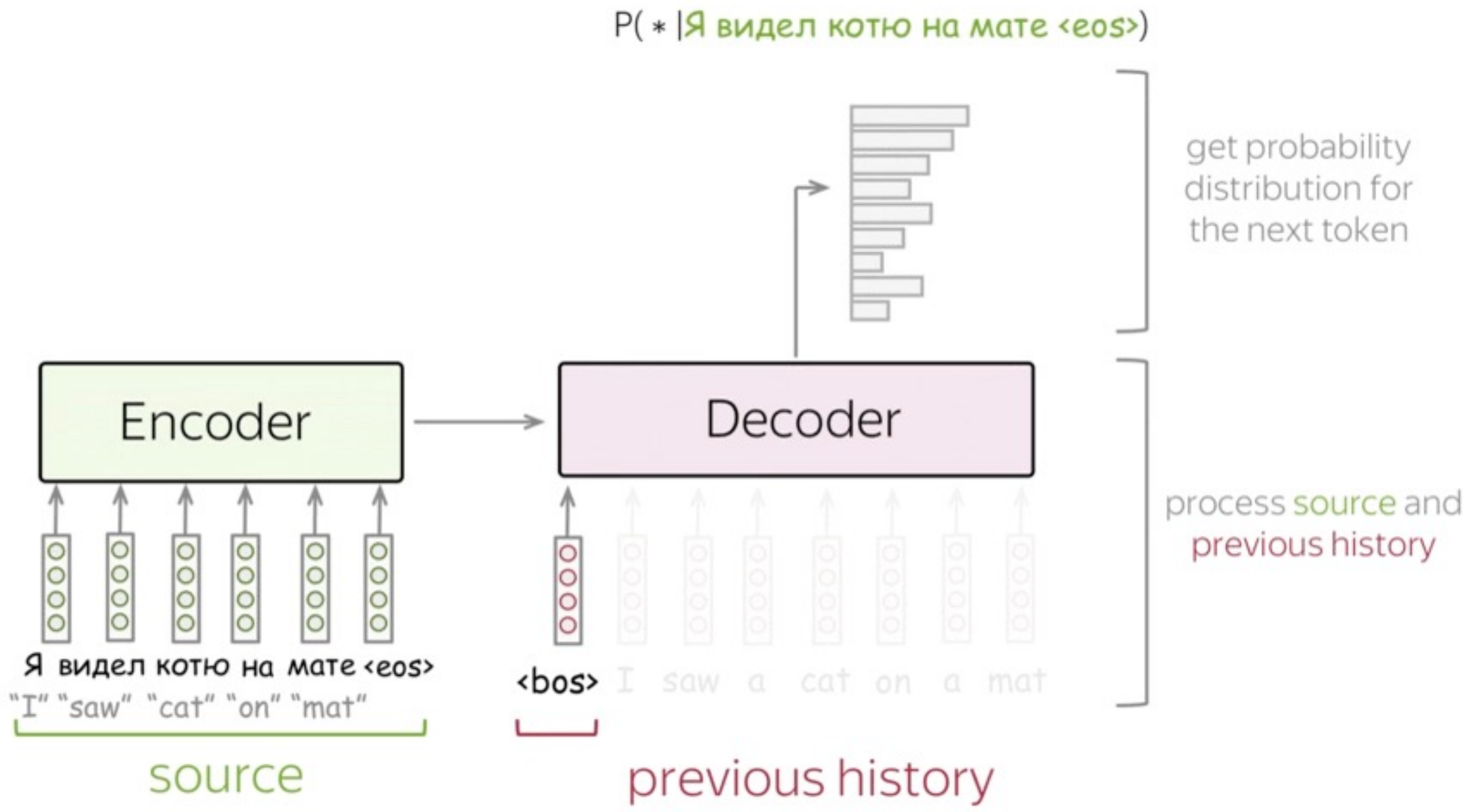
Conditional Language Models

Language Models: $P(y_1, y_2, \ldots, y_n) = \prod_{t=1}^{n} p(y_t | y_{<t})$

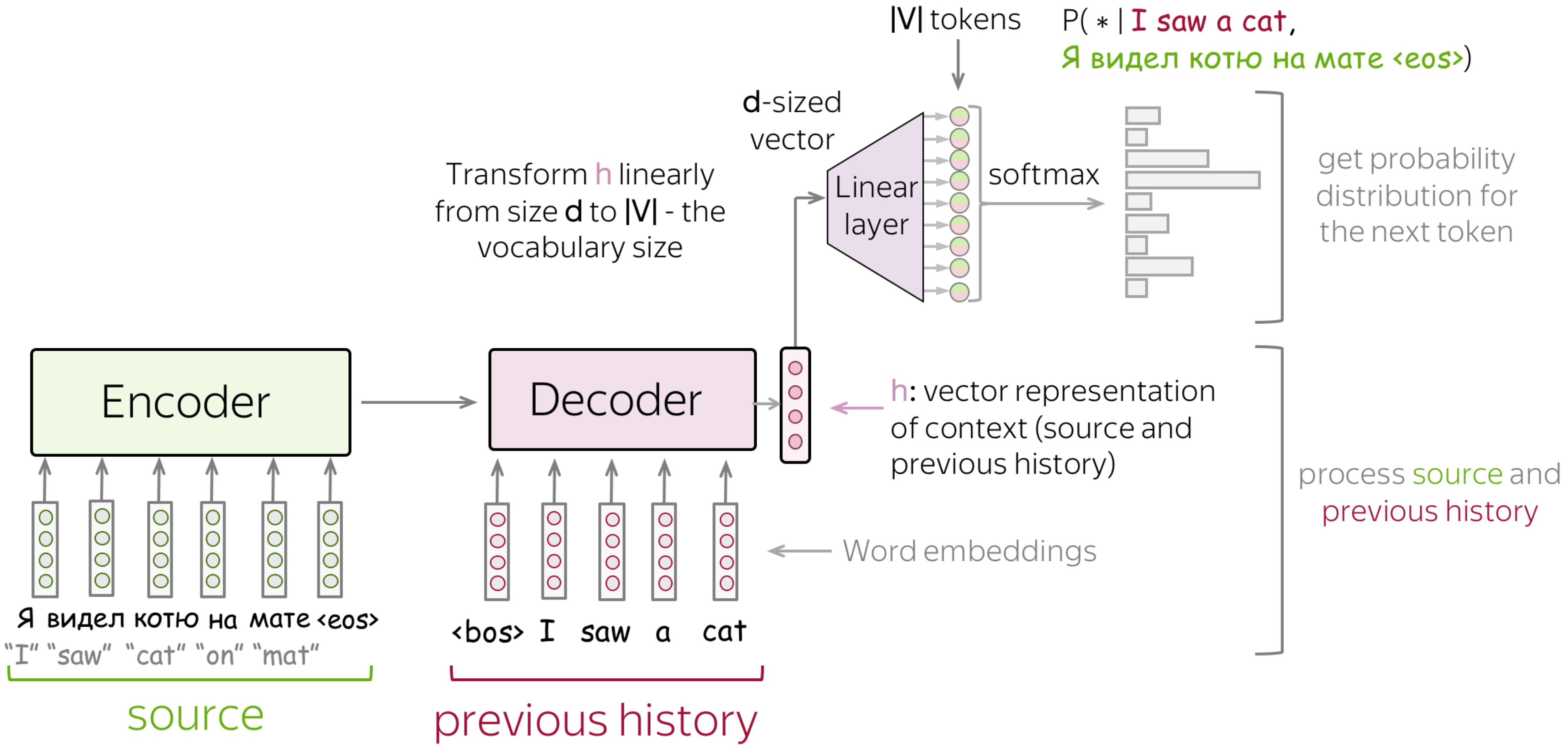
<u>Conditional</u>
Language Models: $P(y_1, y_2, \ldots, y_n, | x) = \prod_{t=1}^{n} p(y_t | y_{<t}, x)$

condition on source x

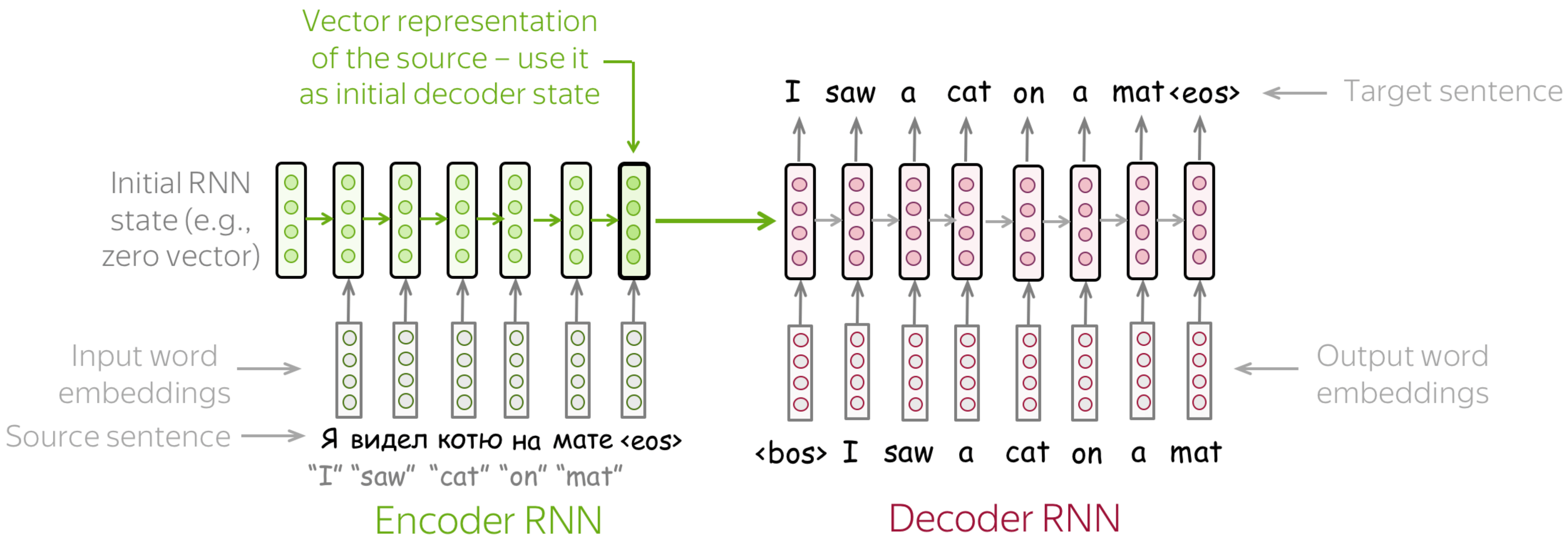
Encoder Decoder explained



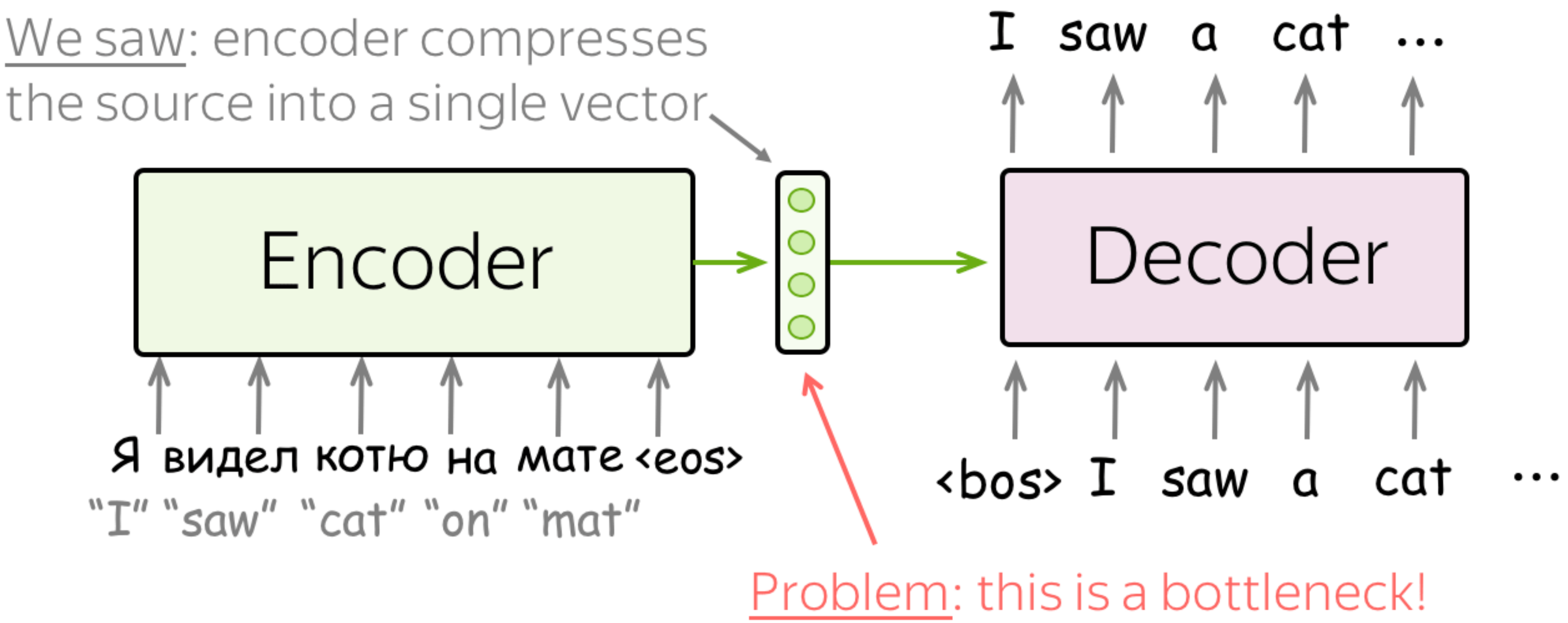
Using a decoder hidden representation



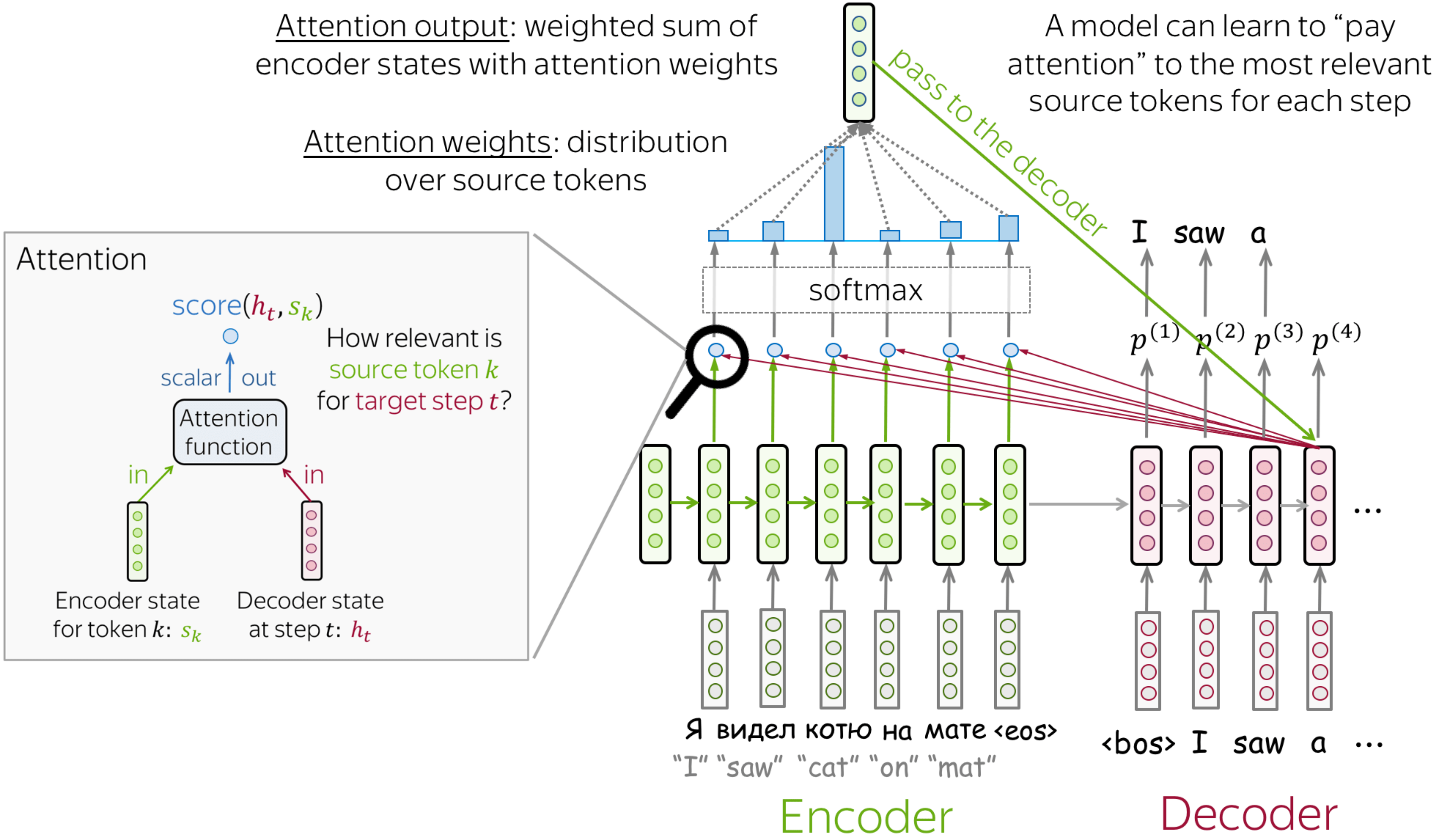
Seq2Seq RNNs



Fixed encoder representations are an issue



Attention



Attention output: weighted sum of encoder states with attention weights
Attention weights: distribution over source tokens
A model can learn to "pay attention" to the most relevant source tokens for each step
pass to the decoder
softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
...
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
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
Encoder
Decoder

How to compute attention?

Attention output

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

"source context for decoder step t"

(weighted sum)

Attention weights

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

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

(softmax)

Attention scores

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

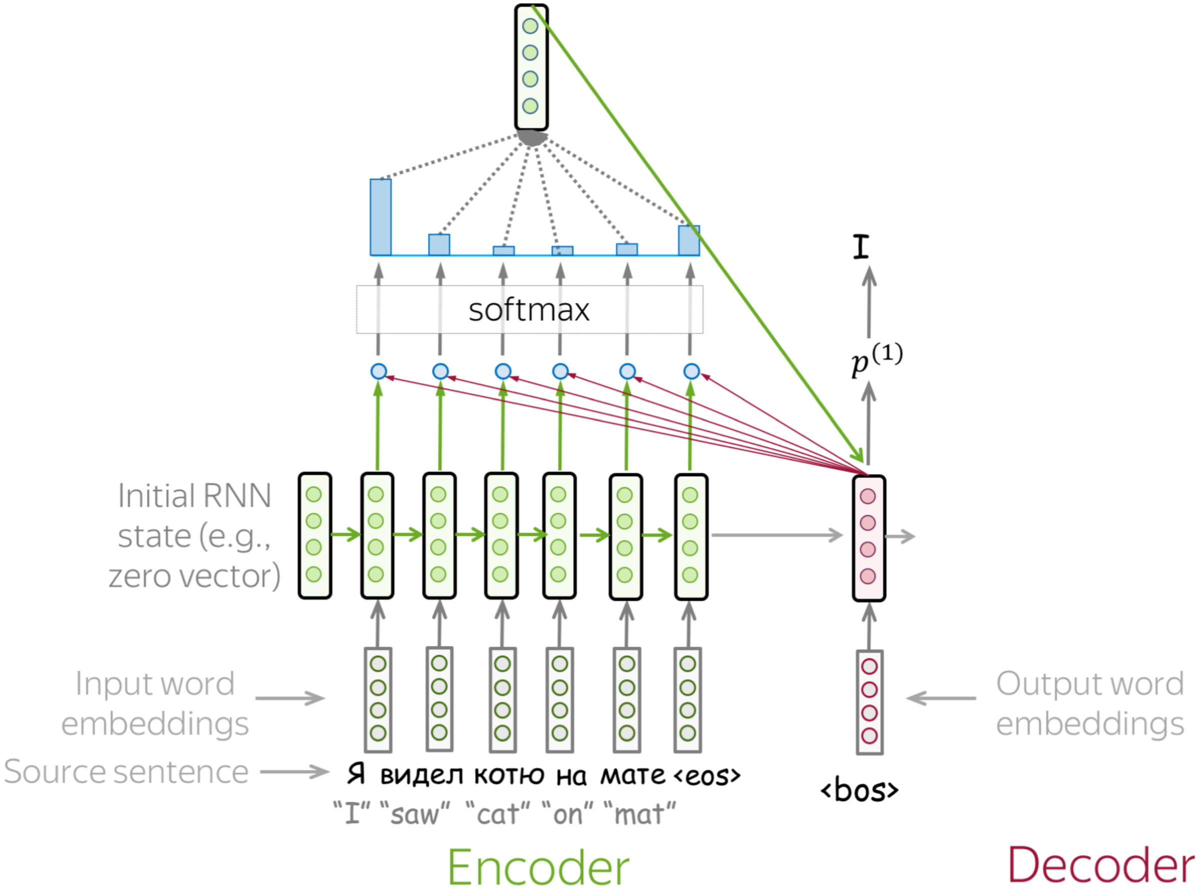
"How relevant is source token k for target step t?"

Attention input

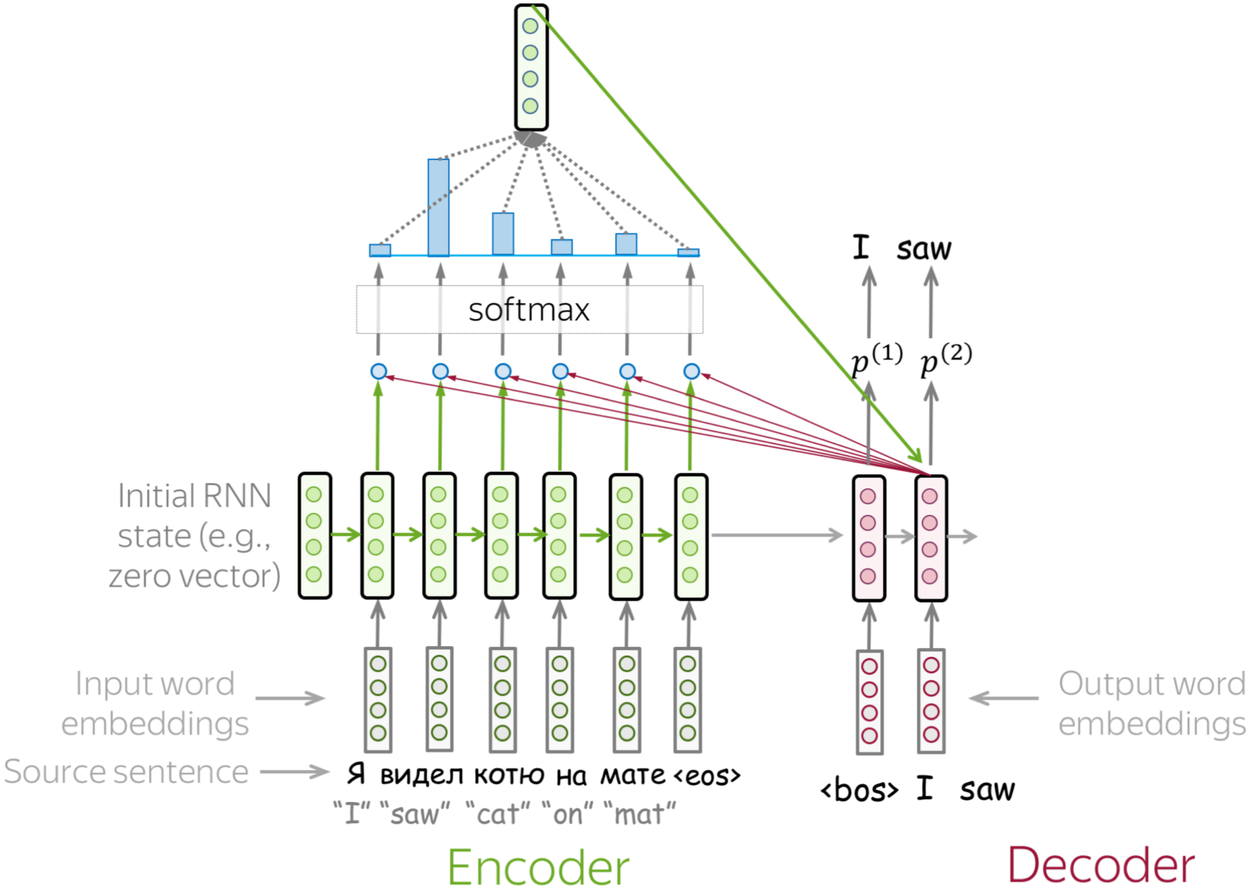
$s_1, s_2, \dots, s_m$ all encoder states

h_t one decoder state

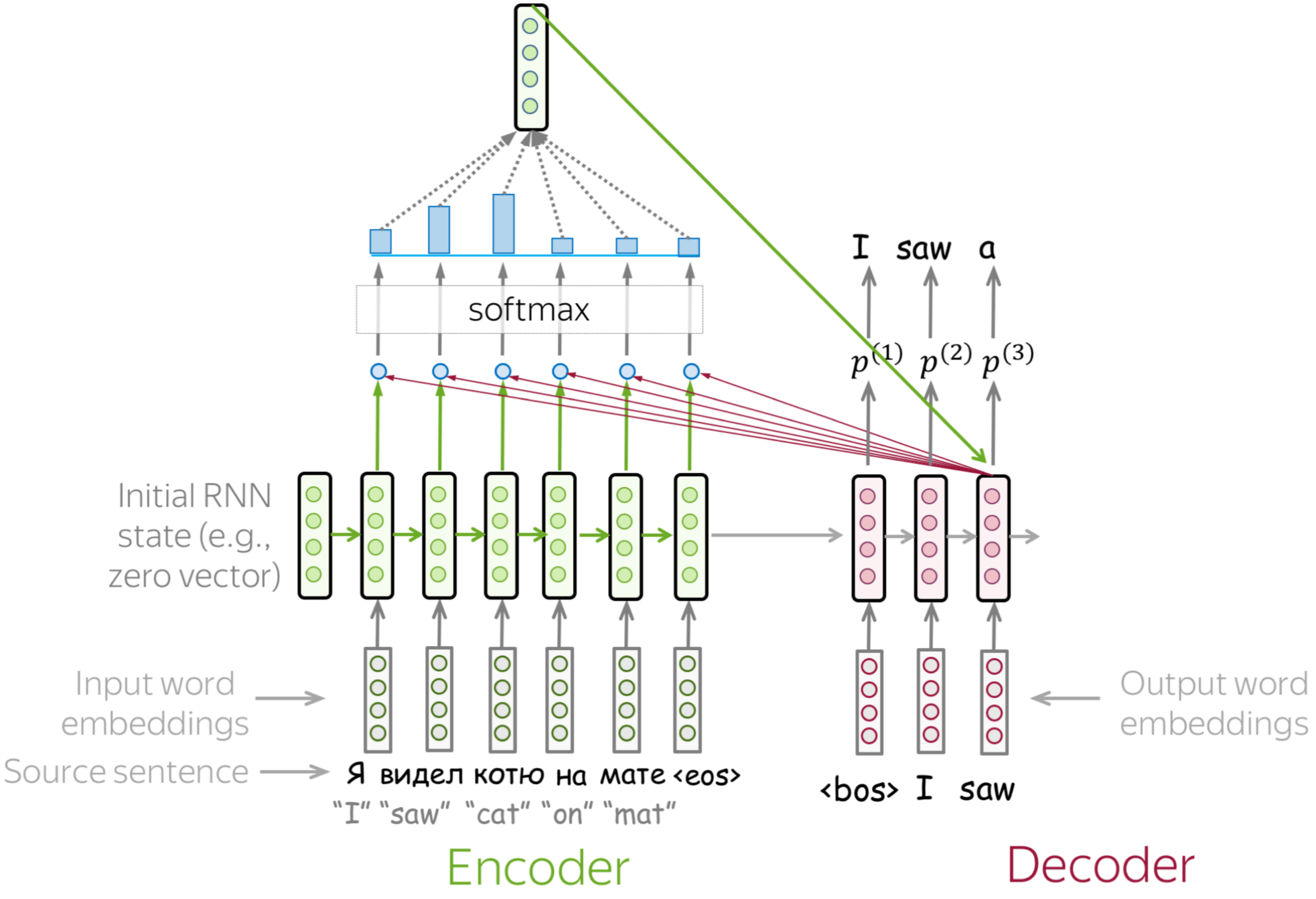
Decoding word 1



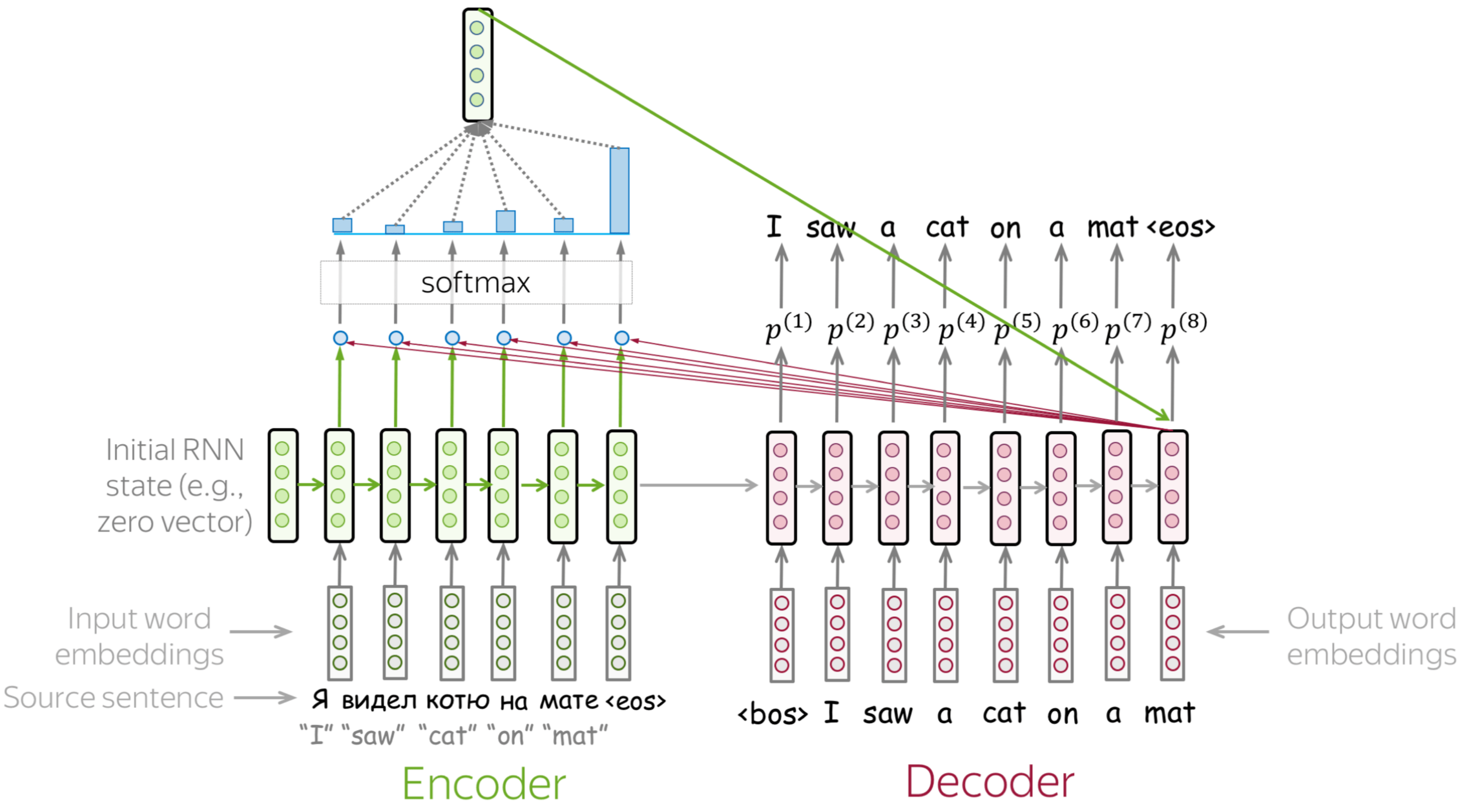
Decoding word 2



Decoding word 3

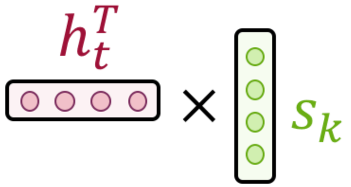


Decoding word N



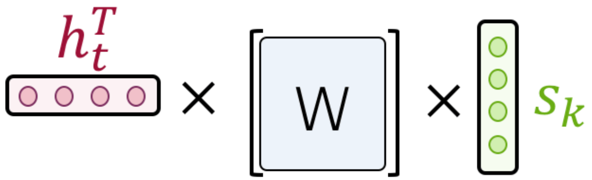
Computing Attention

Dot-product



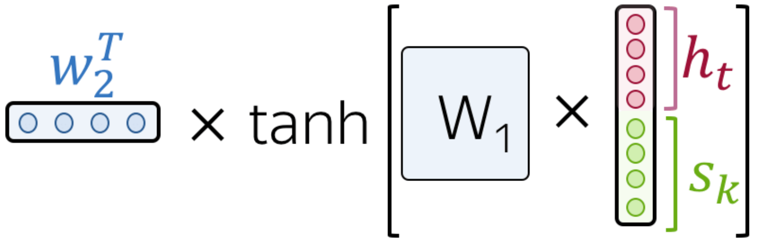
$\text{score}(h_t, s_k) = h_t^T\ s_k$

Bilinear



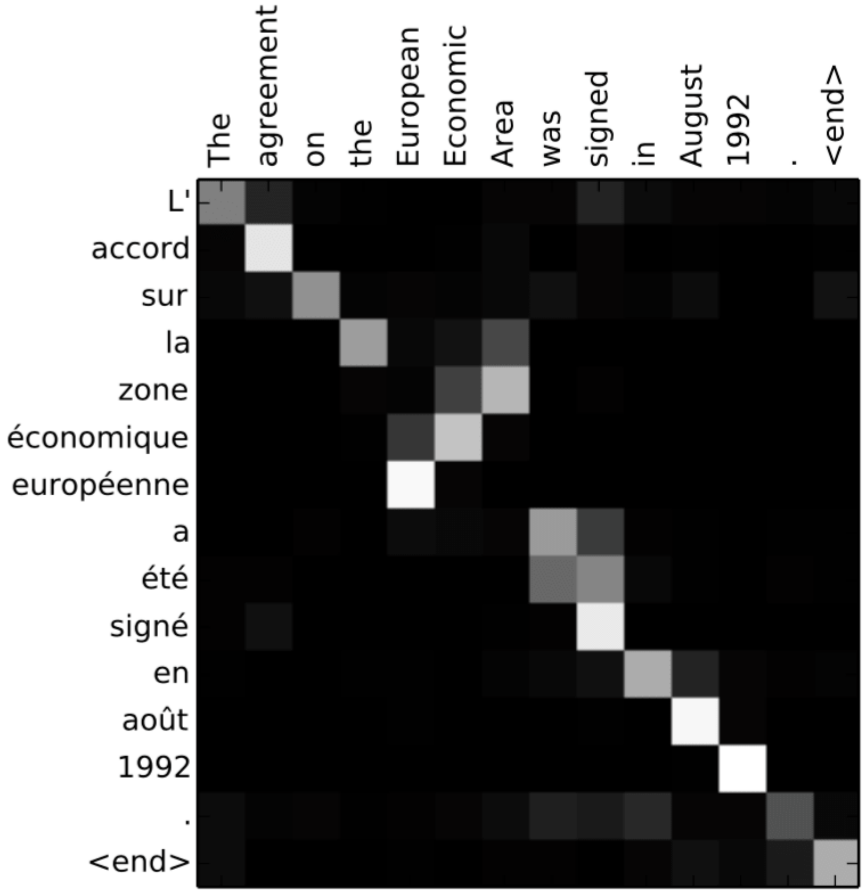
$\text{score}(h_t, s_k) = h_t^T\ W s_k$

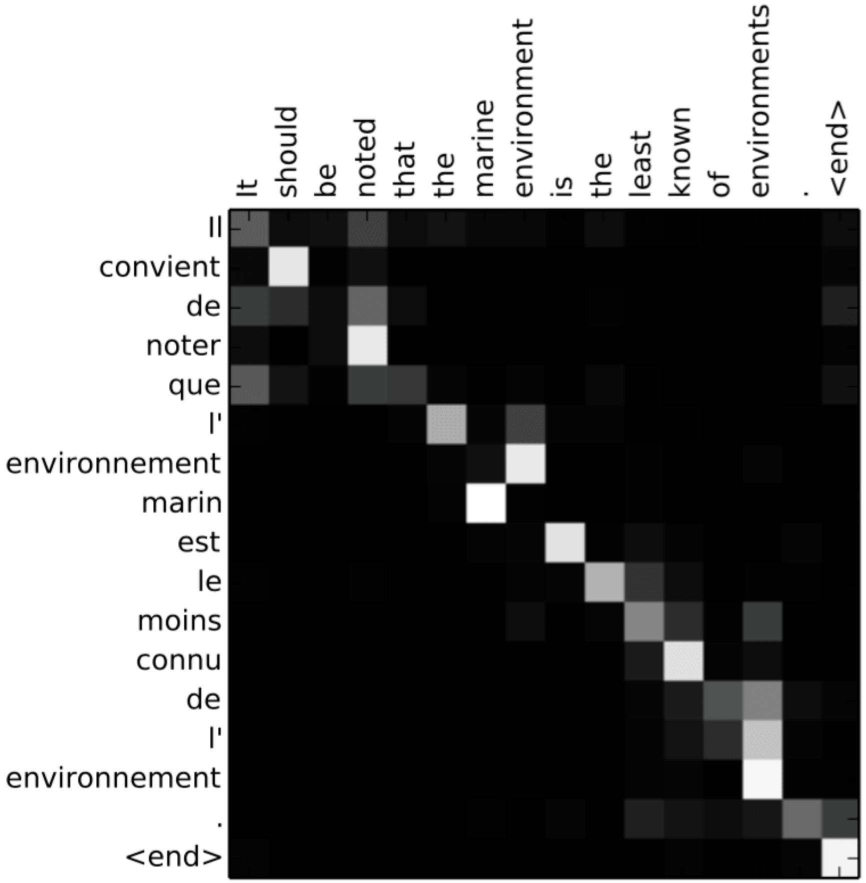
Multi-Layer Perceptron



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

Attention Learns Alignment!





Thank you!