Foundations for Natural Language Processing

Improving Encoder-Decoder, Attention

Ivan Titov
(with graphics/materials from Elena Voita)

School of informatics

Plan for today

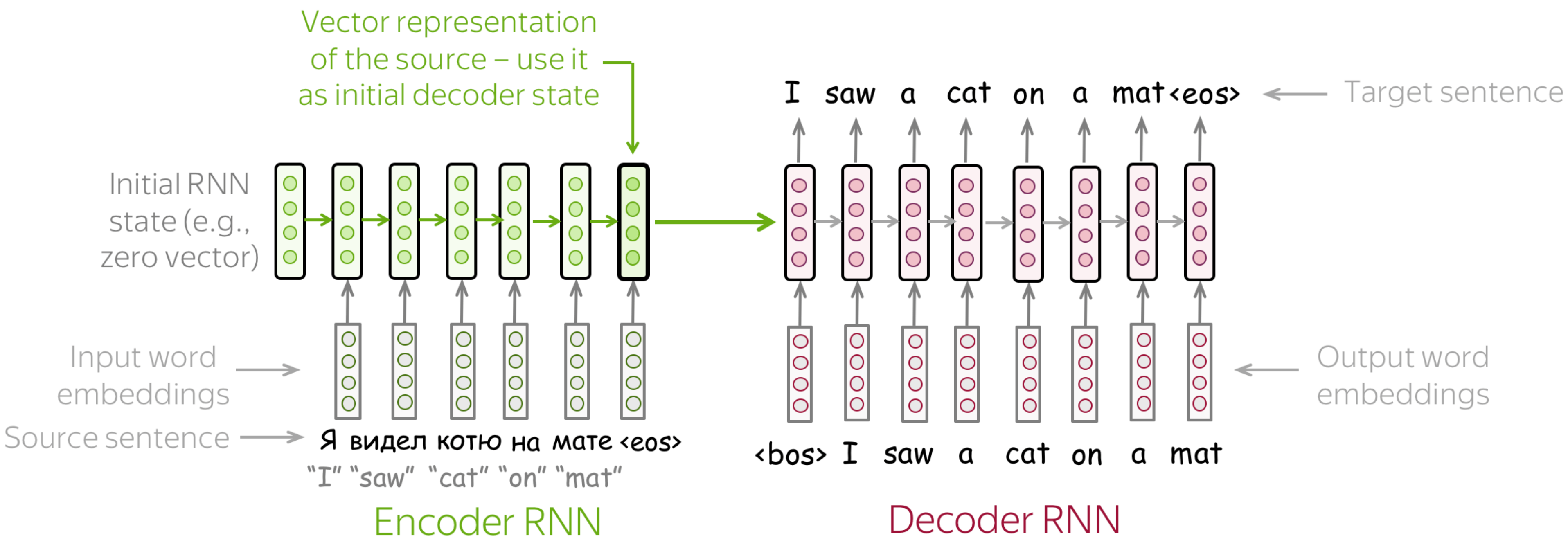
Last time:

- Vanilla Encoder-Decoder
- Text Generation (training, inference, evaluation)

Today

- Subword segmentation
- Improving Encoder-Decoder models
- Modeling Attention

Tokenization



We considered tokenization of sentences into 'words'

(whatever we mean by a 'word')

Tokenization

Tokenization

Word-level

- fixed vocabulary
- can process only a fixed number of words

Subword-level

- open vocabulary
- rare and unknown tokens are encoded as sequences of subword units

Instead of '*unrelated*', we get two tokens 'un@@' '*related*'

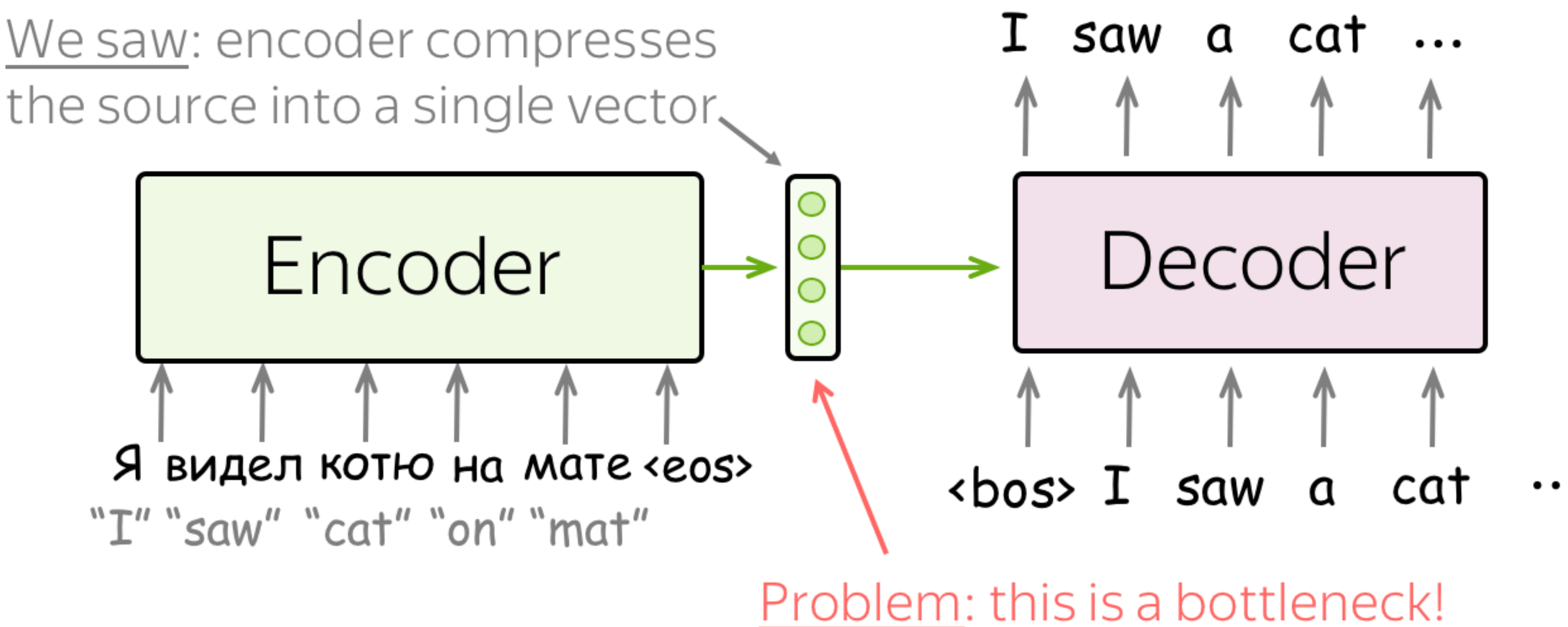
Subword segmentations reduces sparsity and results in a speeds-up (recall: softmax involves summation over all token types, few token typs -> faster computation)

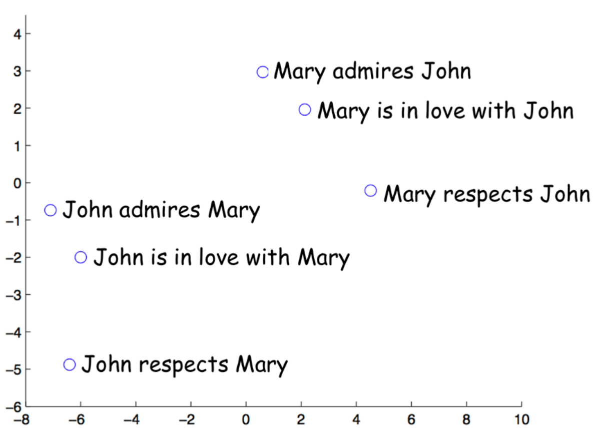
Crucial for morphologically-rich languages

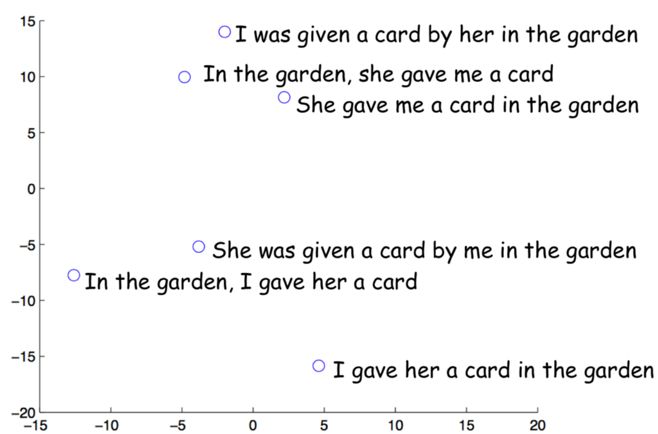
Standard segmentation algorithms rely on character ngram frequency (not on morphology (e.g., Byte-Pair Encoding)

Used in virtually any modern neural model

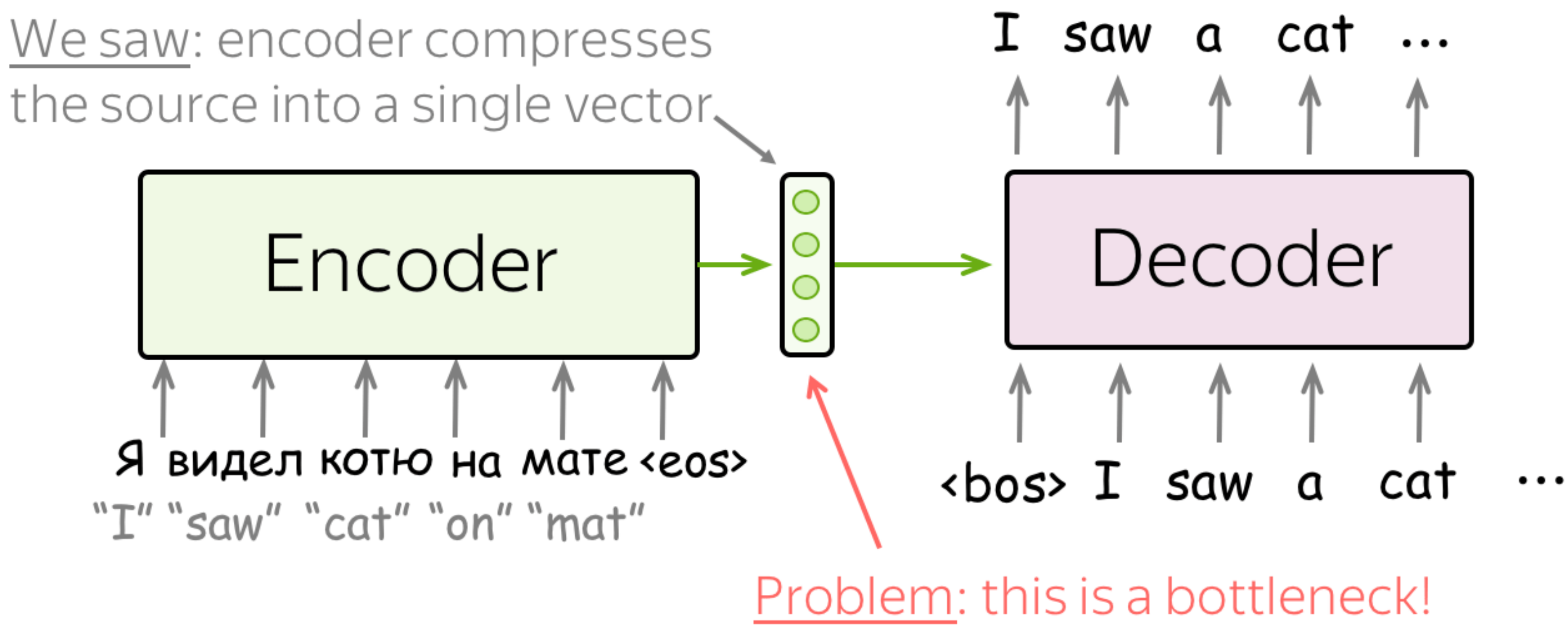
The key problem with this approach







The key problem with this approach



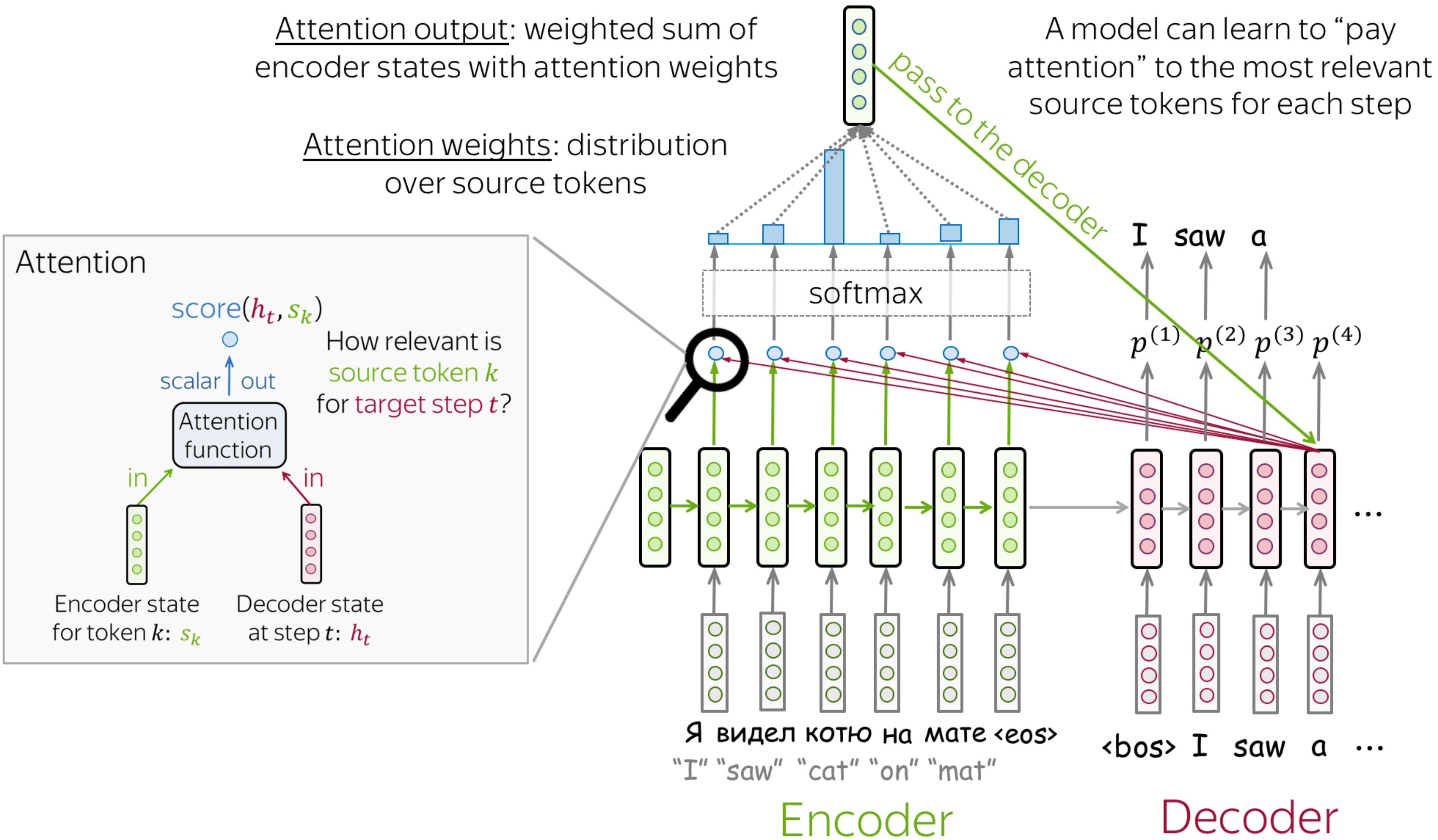
Problem: fixed source representation is suboptimal:

- for the encoder, it is hard to compress the sentence;
- for the decoder, different information may be relevant at different steps.

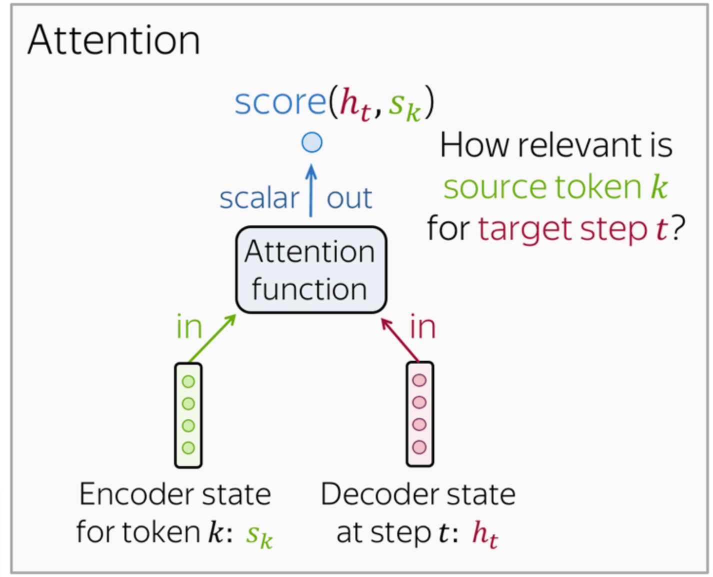
Solution: modeling "attention"

Attention: Intuition

At every step, the decoder decide on which input tokens to focus



Attention



At each decoder step, attention

- receives **attention input**: a decoder state h_t and all encoder states s_1, s_2, ..., s_m;
- computes **attention scores**
  For each encoder state s_k, attention computes its "relevance" for this decoder state h_t. Formally, it applies an attention function which receives one decoder state and one encoder state and returns a scalar value $score(h_t, s_k)$;
- computes **attention weights**: a probability distribution - softmax applied to attention scores;
- computes **attention output**: the weighted sum of encoder states with attention weights.

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

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

"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

$s_1, s_2, \dots, s_m$

all encoder states

h_t

one decoder state

Attention

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

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

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

(softmax)

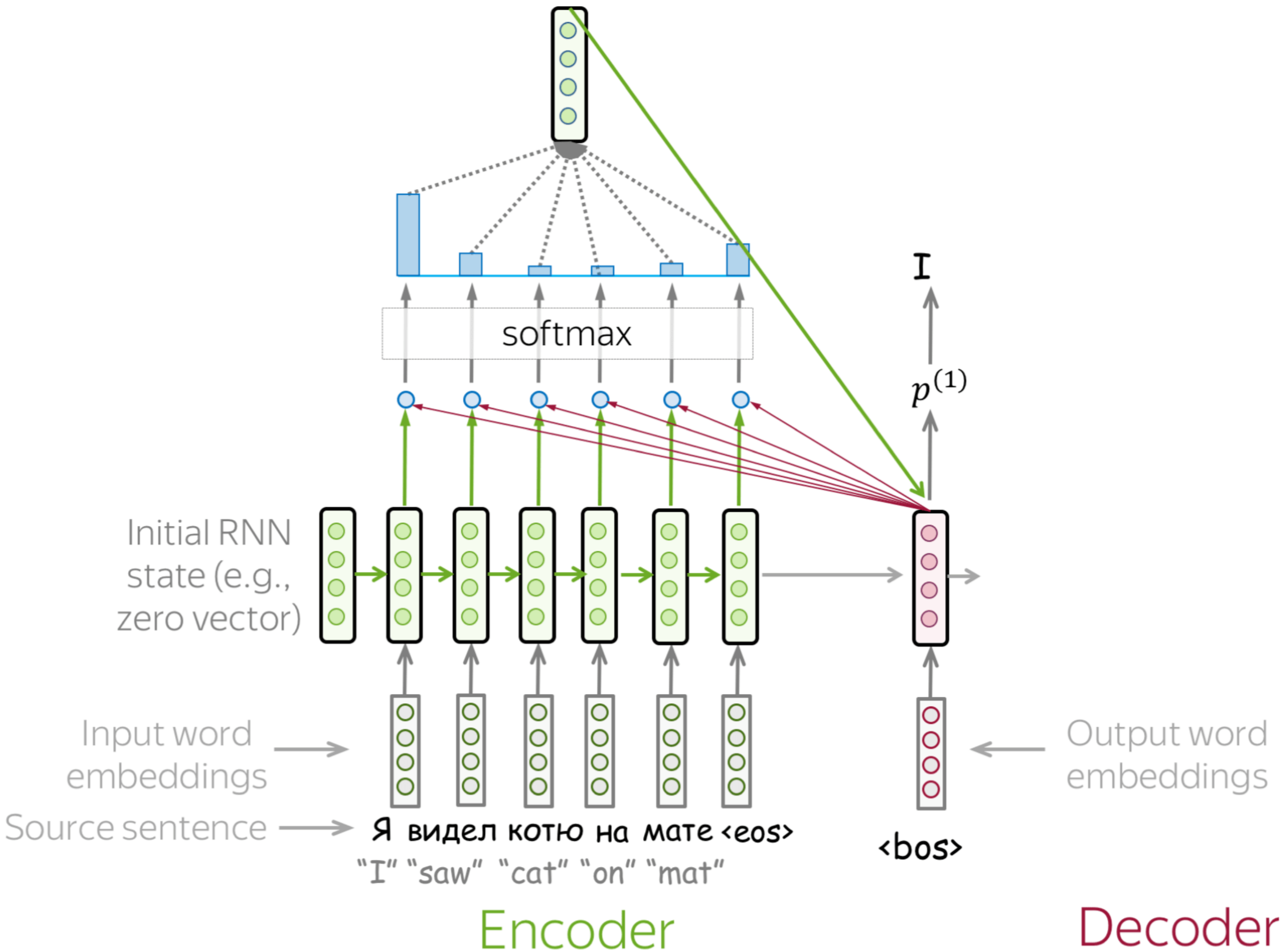
Att

Att

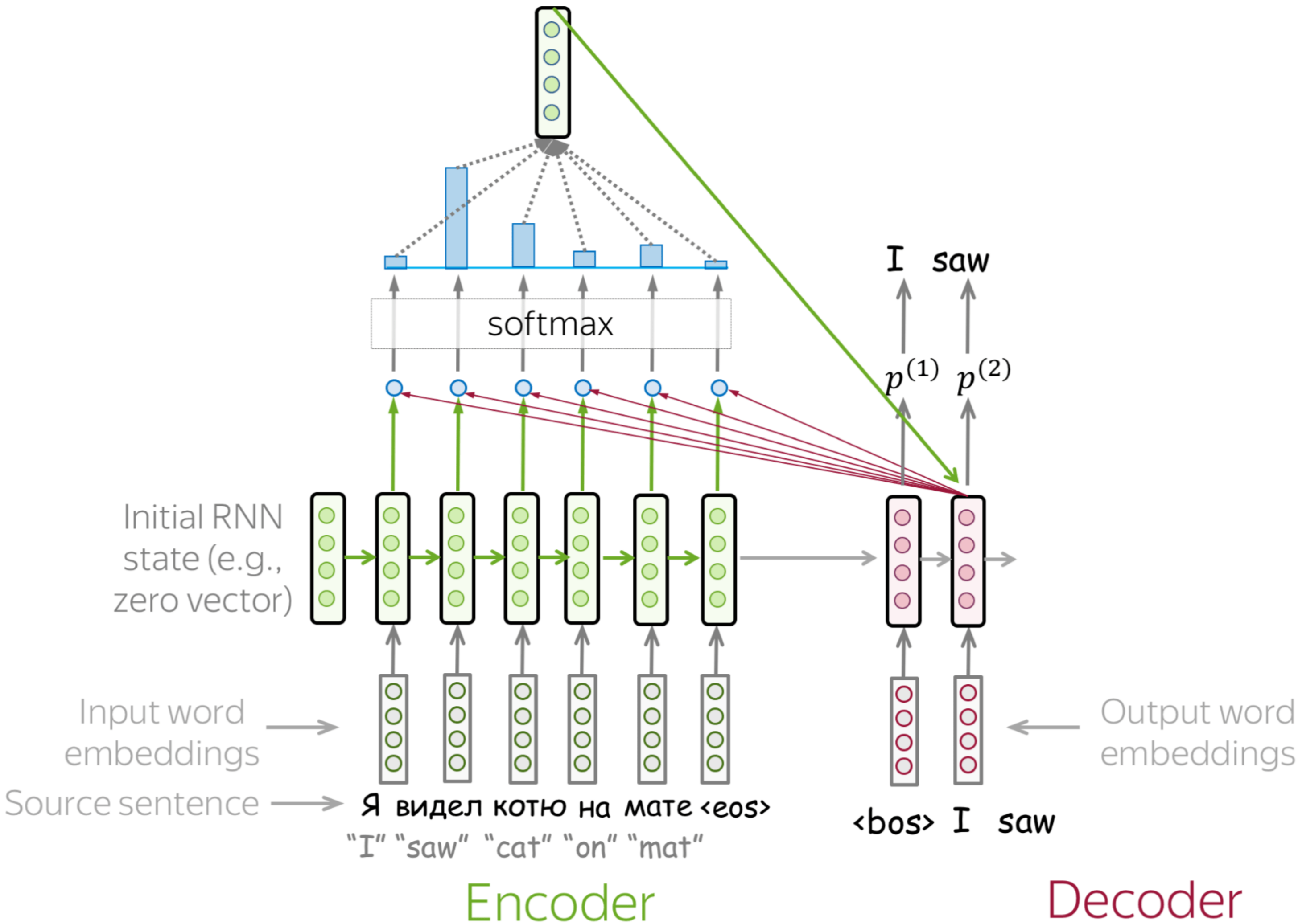
The model learns to choose relevant input tokens for every step

the computation is differentiable so
everything here is learned end-to-end
with backpropagation

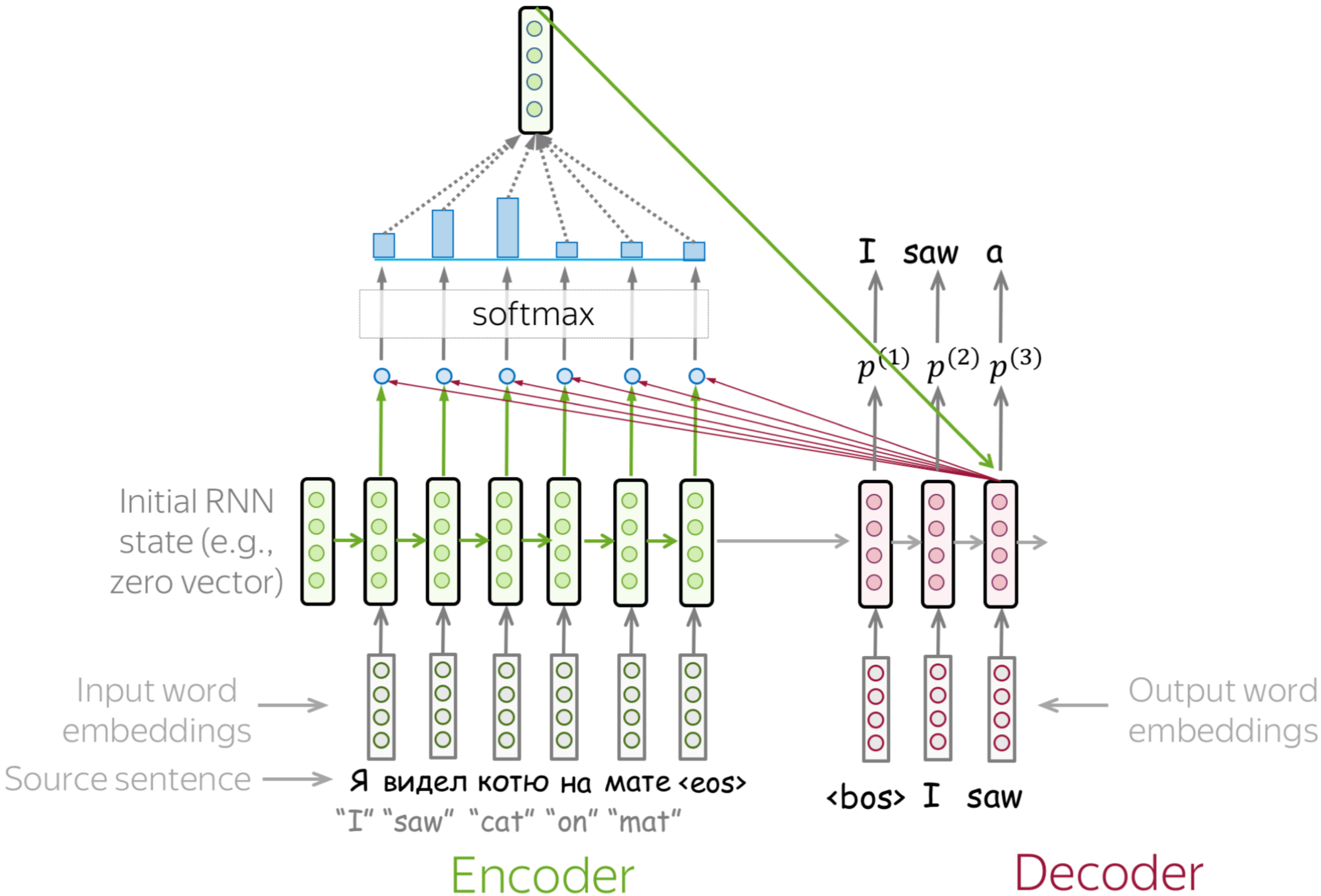
Attention: step by step



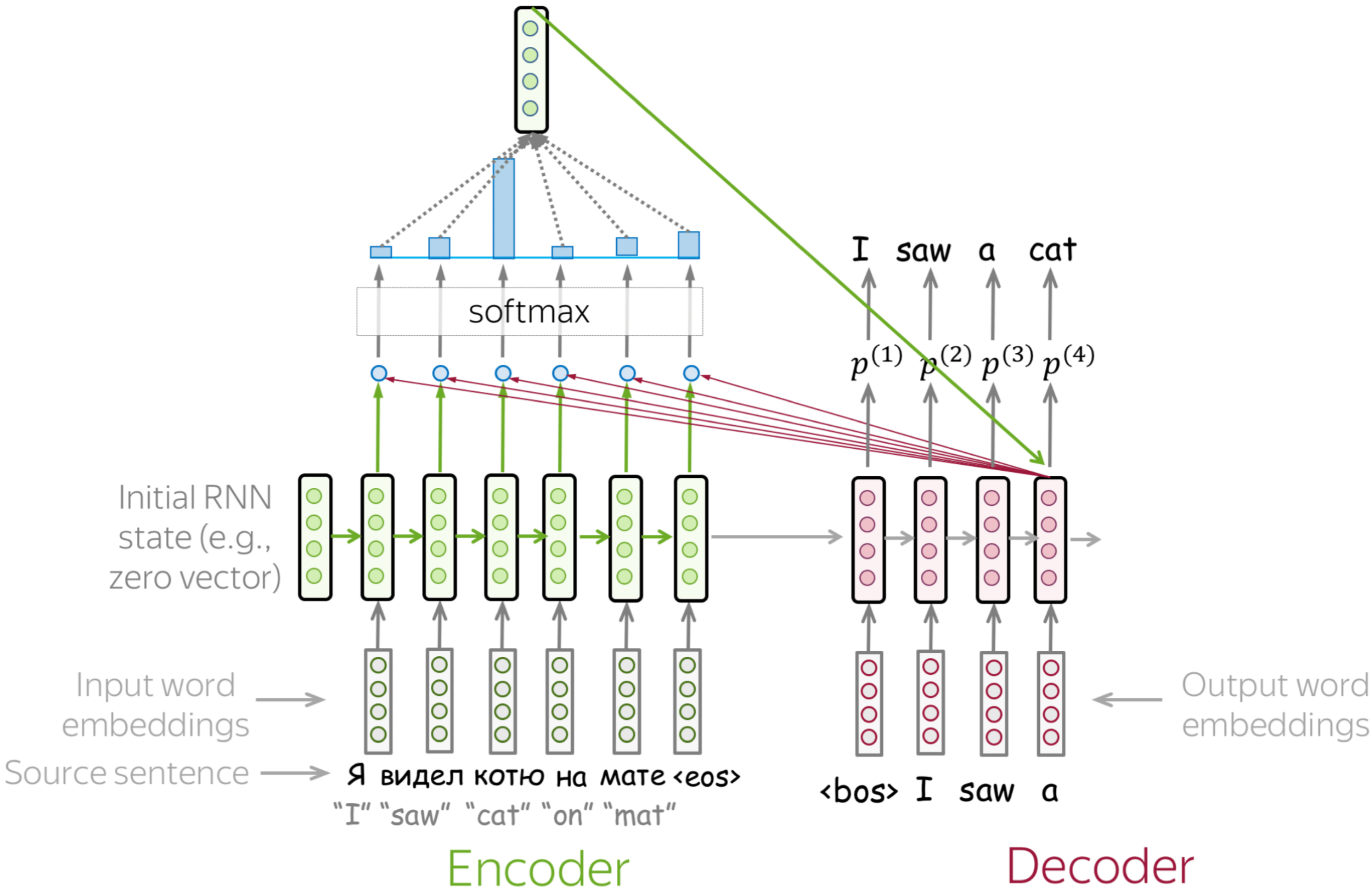
Attention: step by step



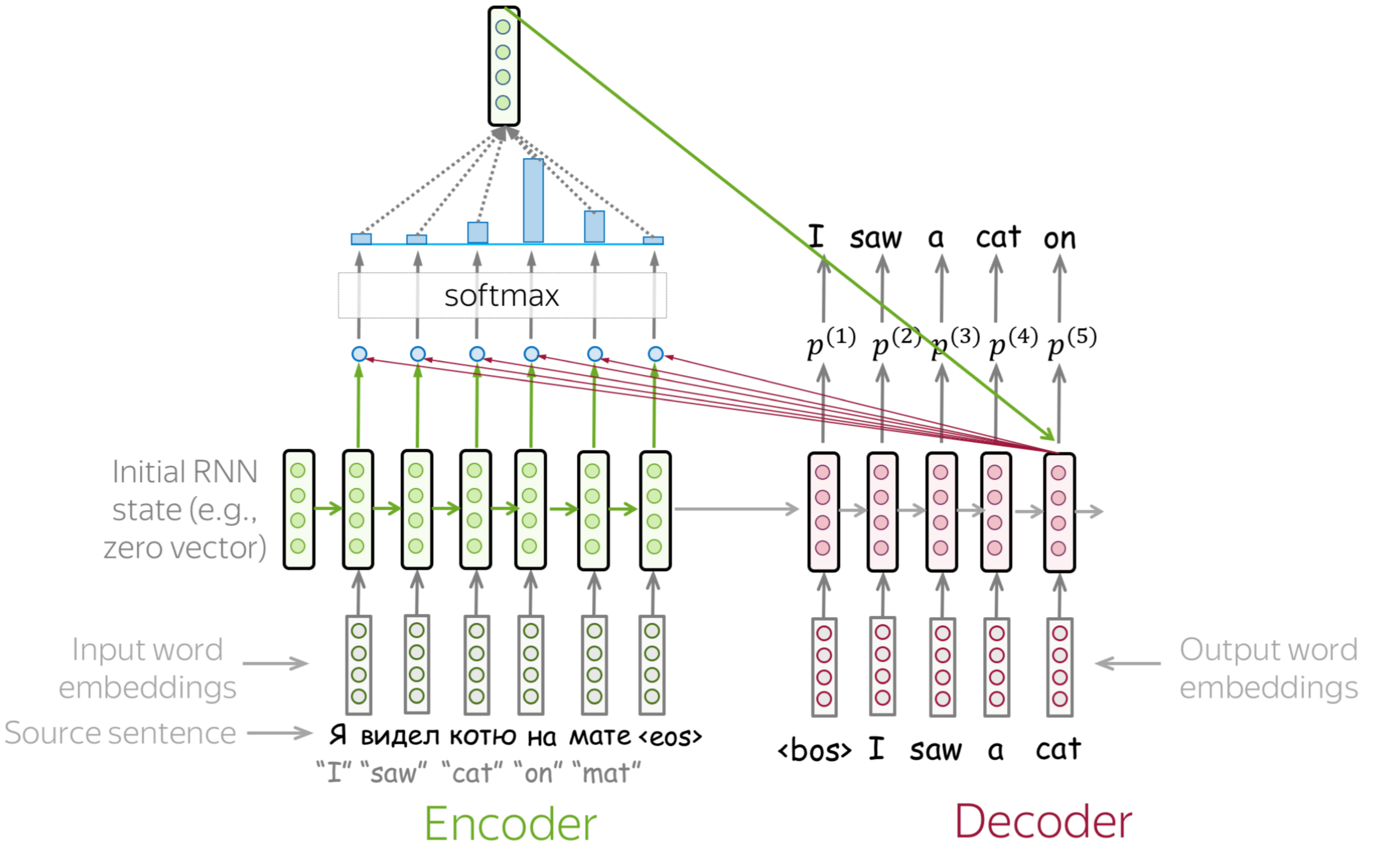
Attention: step by step



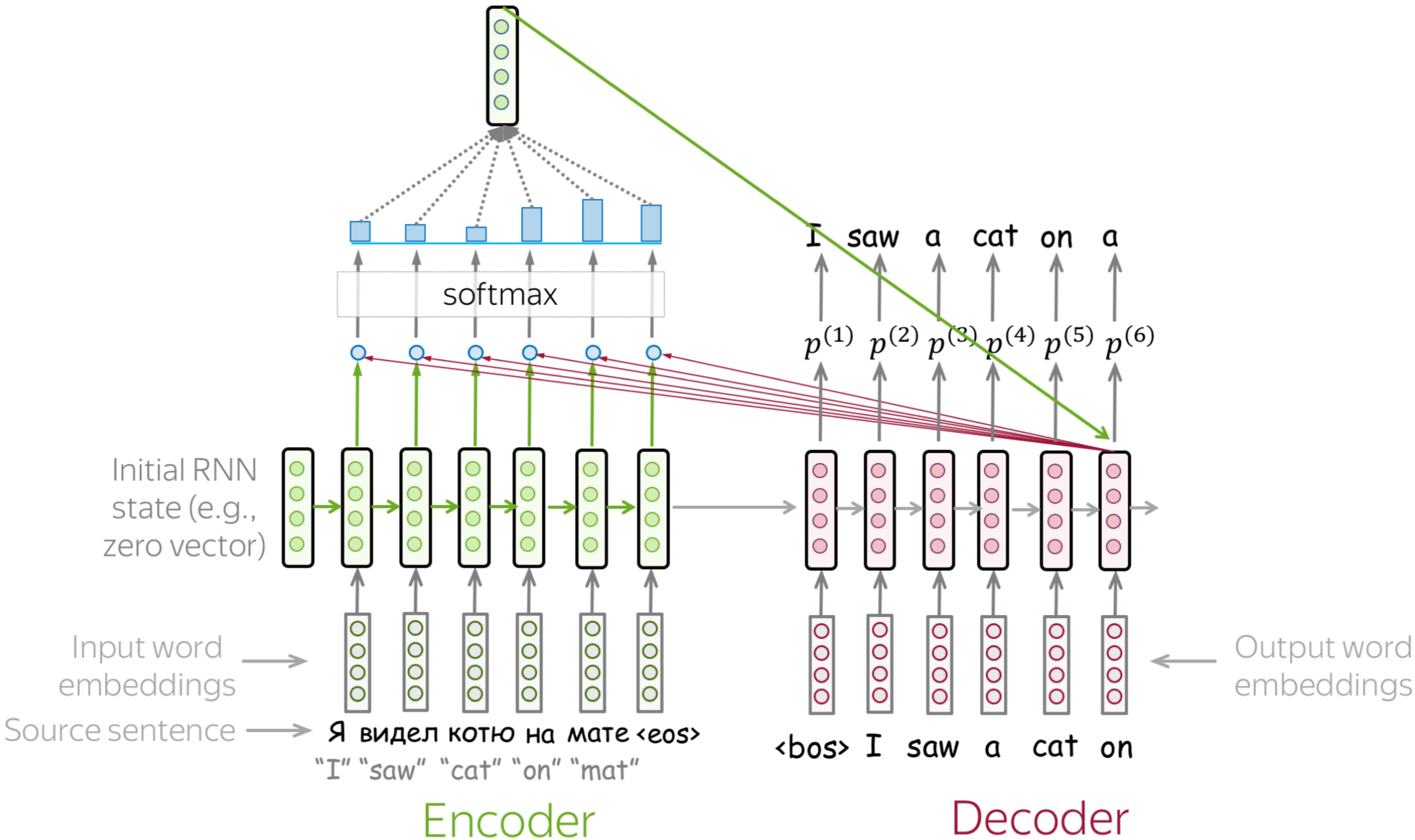
Attention: step by step



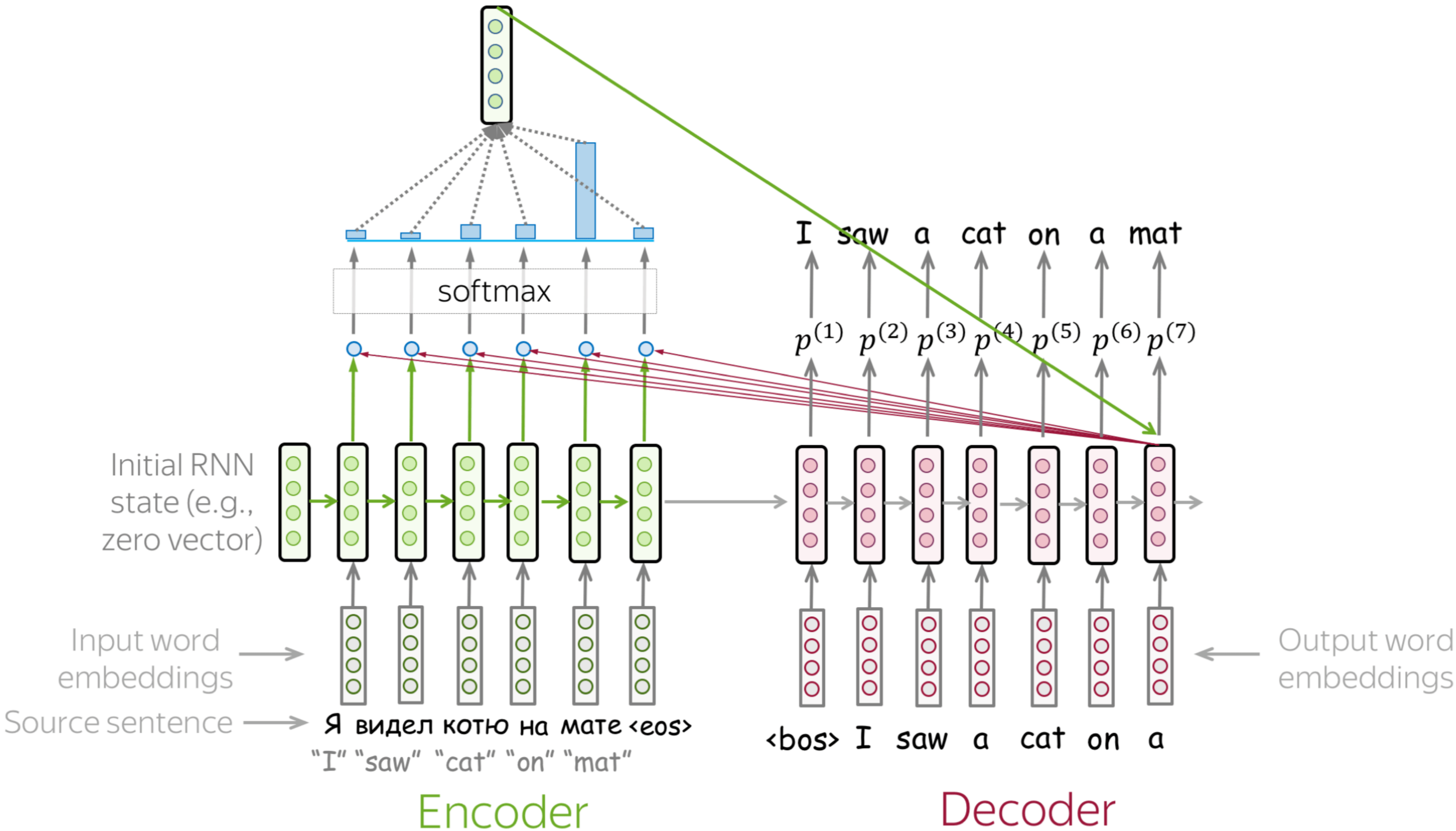
Attention: step by step



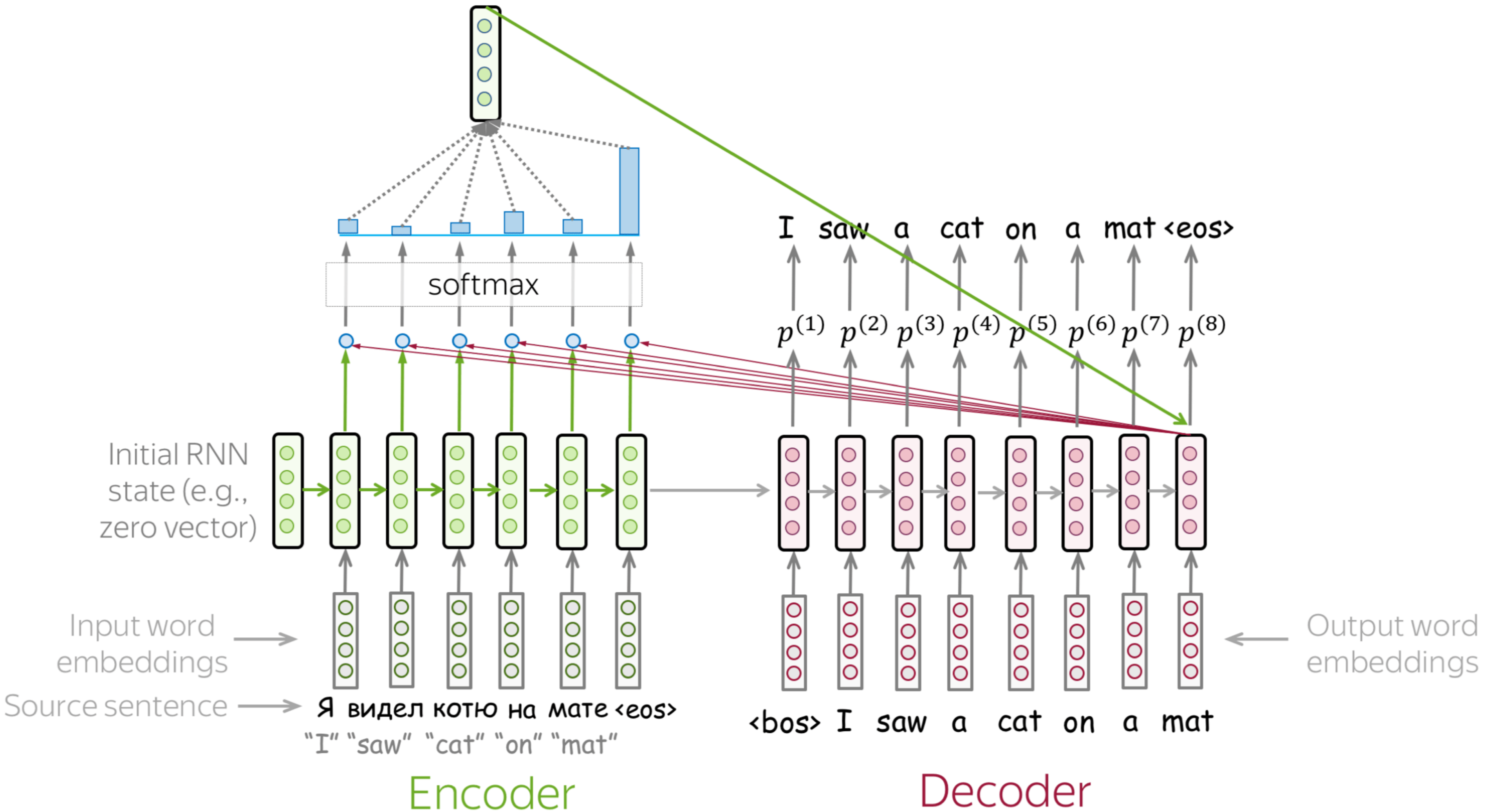
Attention: step by step



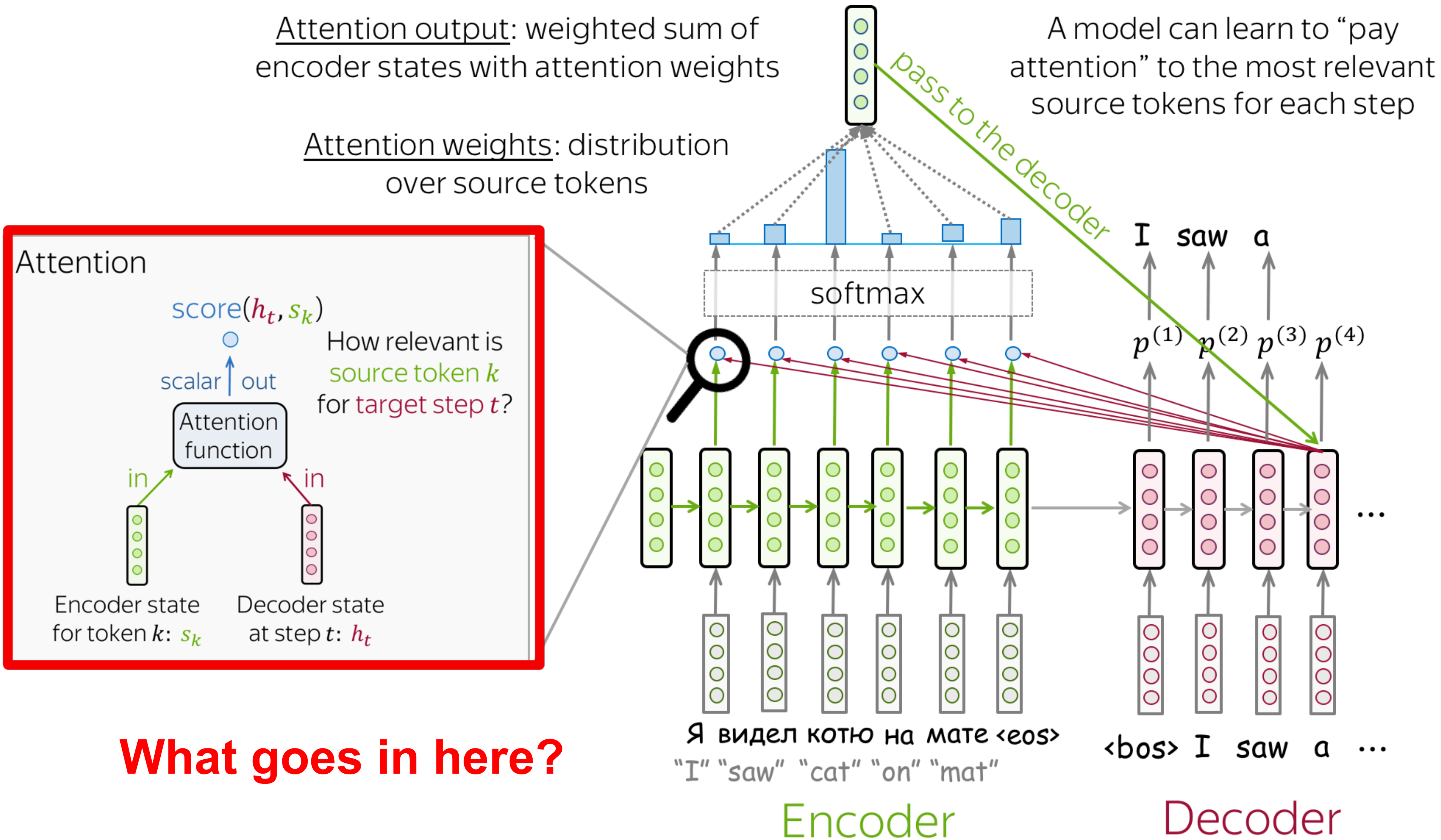
Attention: step by step



Attention: step by step

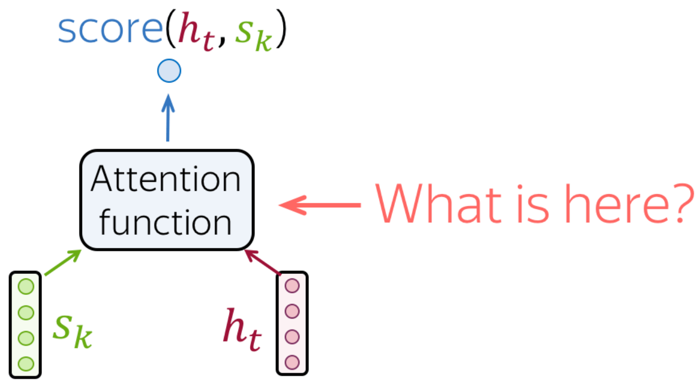


Attention

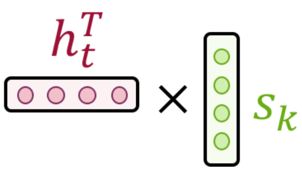


What goes in here?

How do we compute attention scores? Alternatives

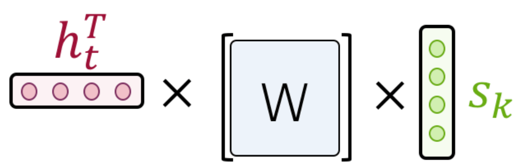


Dot-product



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

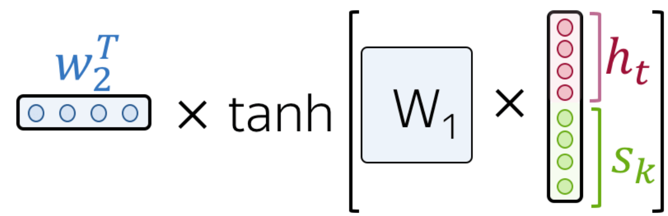
Bilinear



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

aka 'Luong attention'

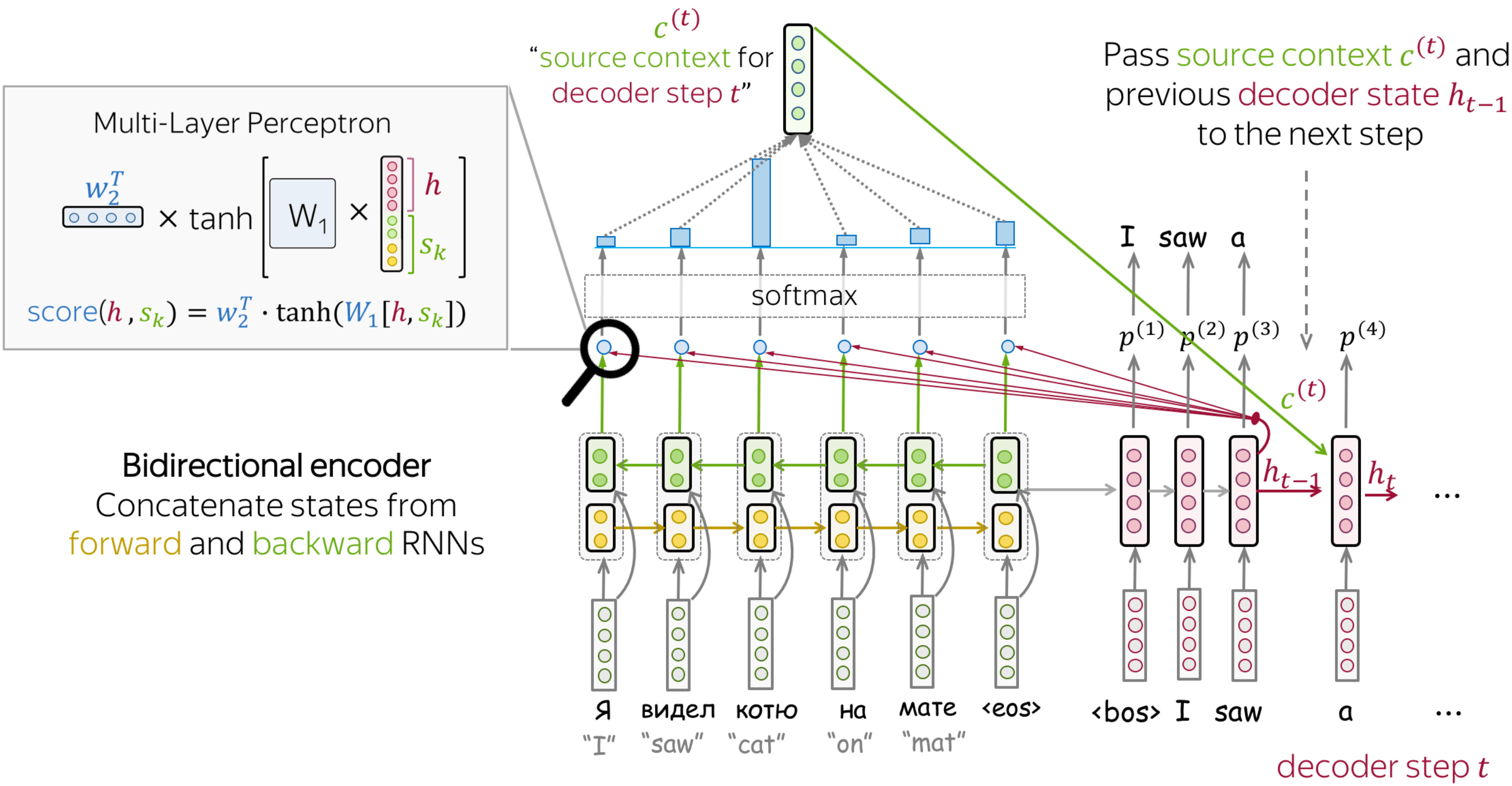
Multi-Layer Perceptron



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

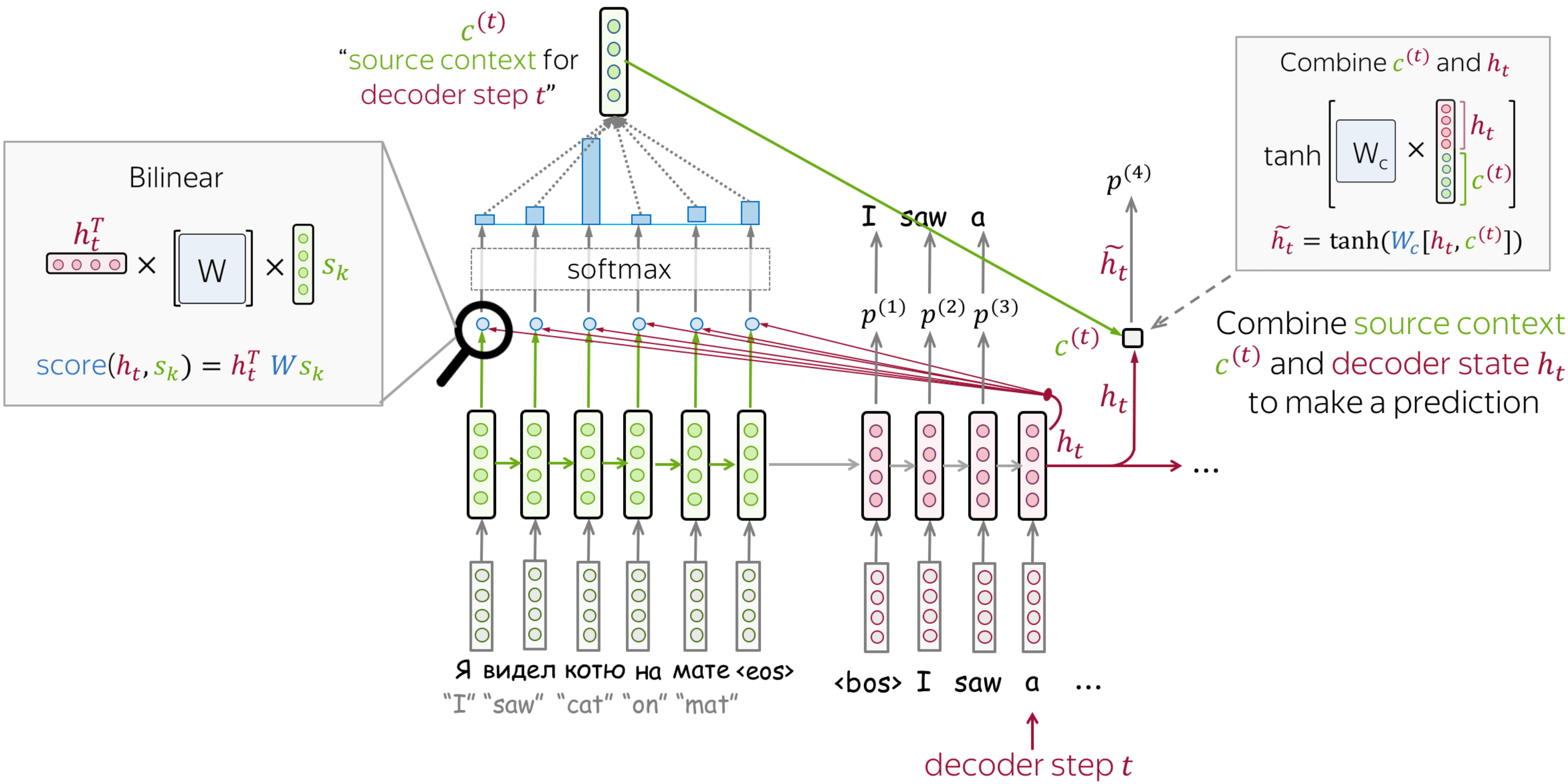
aka 'Bahdanau' attention
(from the original paper)

Encoder-Decoder variants: Bahdanau



https://arxiv.org/pdf/1409.0473.pdf

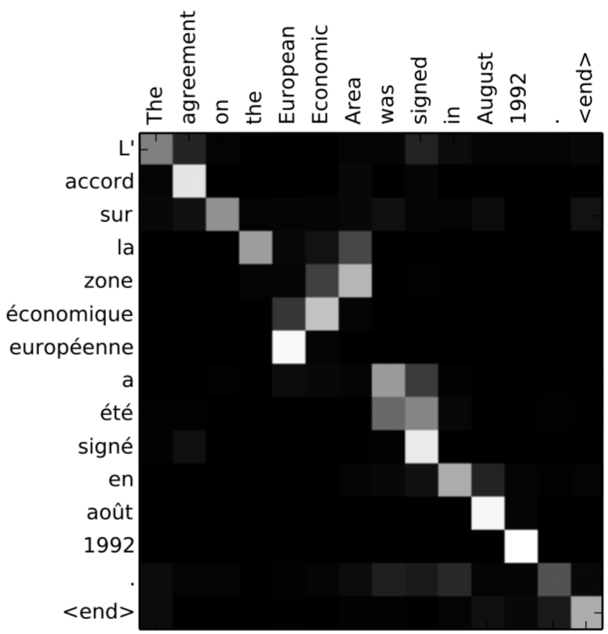
Encoder-Decoder variants: Luong

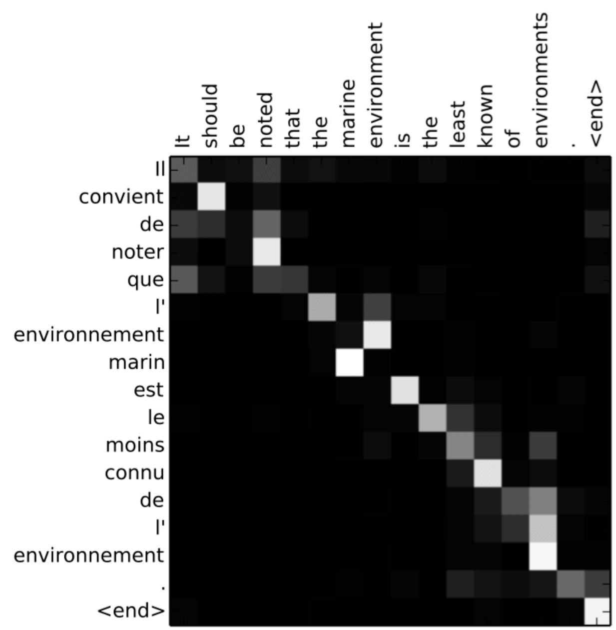


https://arxiv.org/abs/1508.04025

Is attention interpretable?

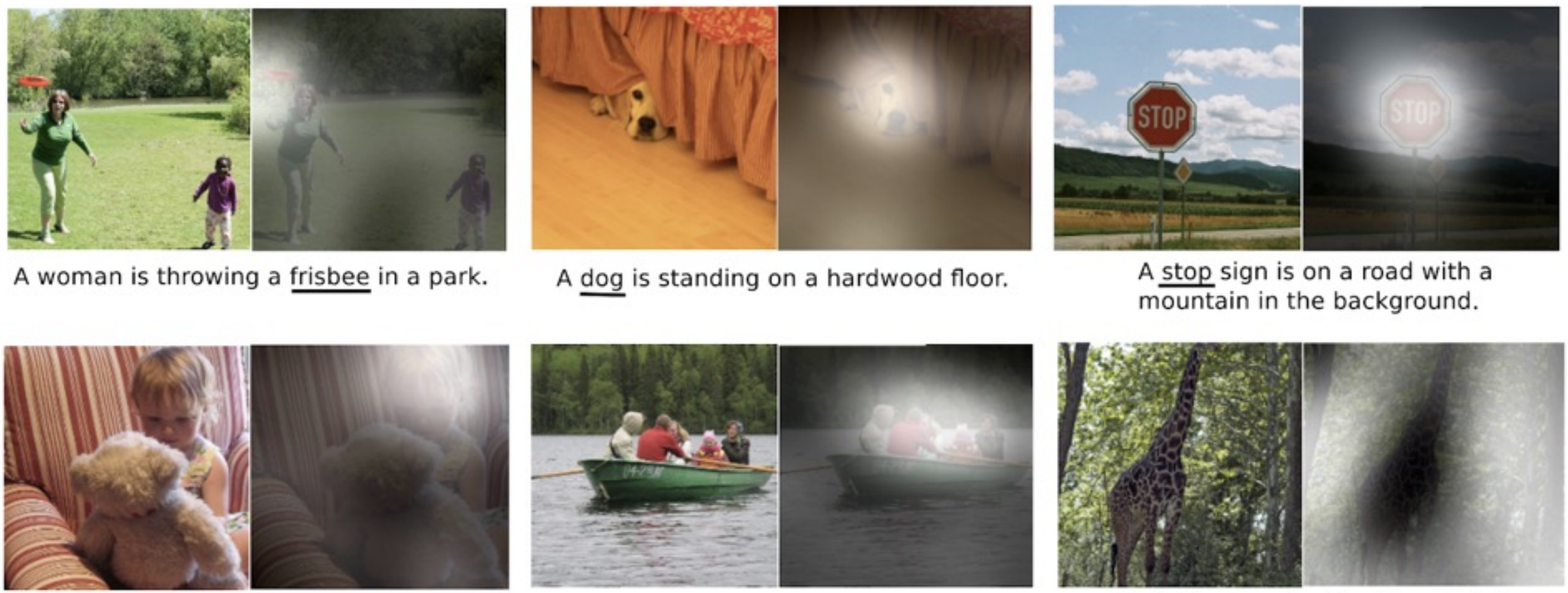
- The attention may be perceived as modelling alignment between input and output words, but
  - decoder computes attention **before** generating the target token (i.e. the choice of the token does not influence attention)
  - sometimes states encode something unexpected (e.g., <eos> may capture the general topic of the sentence)
  - attention is (?) not an explanation





Enoder-decoders and attention are a very general idea

For example, caption generation



A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

From https://arxiv.org/pdf/1502.03044.pdf

Attention is not necessarily faithful

Generally, you can make a change to model parameters such that attention score a_k for a state s_k decreases N-fold ($a'_k = s_k / N$) whereas s_k magnitude increases N-fold ($s'_k = s_k$ N)

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

"source context for decoder step t"

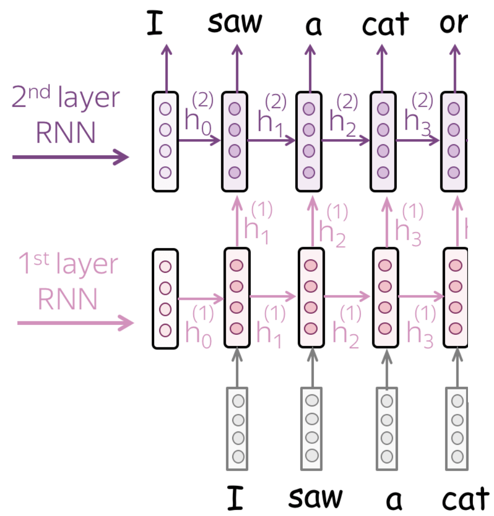
$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$$

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

Since the attention output is the weighted sum of embedding encoders states, the attention output (c) will not change

Attention is not necessarily faithful

Also, if we use a multilayer encoder (and we usually will), there is no <u>guarantee</u> that a state in a *n*-th layers (n > 1) encodes the n-th token



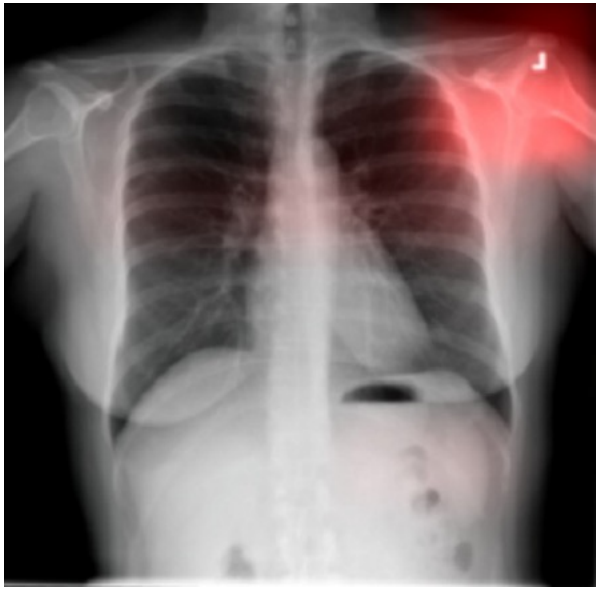
There are attribution methods (e.g., norm x gradient, layer-wise relevant propagation, integrated gradients, attention flow, zero valuing, …) which attempt to address these issues, but they also have pitfalls

Generally: attention cannot be trusted blindly but it can signal some issues with our models

Attribution can help us detect issues with our models

Even if not perfectly faithful attention and attribution techniques help us detect issues with our models

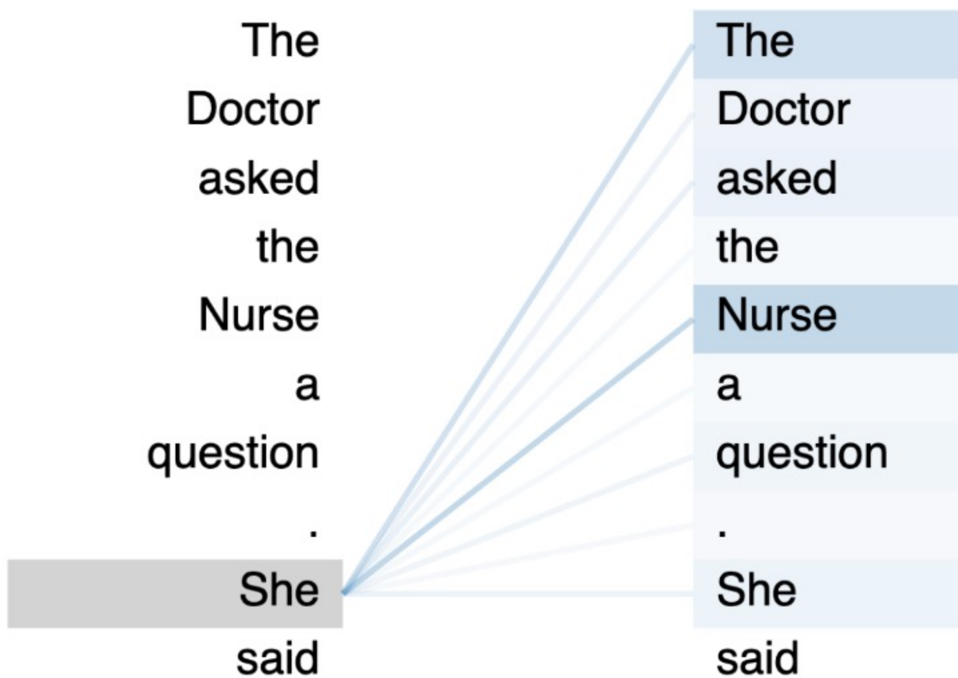
Detecting pneumonia from an x-ray, the model 'looks' at the corner of an image, rather than at lungs.   Any thoughts on why?



Zech et al (PLOS Medecine 2018)

Attribution can help us detect issues with our models

The model decides that pronoun ‘she’ refers to ’Nurse’ rather than Doctor (gender bias)



https://www.comet.com/site/blog/explainable-ai-for-transformers/

It is not an encoder-decoder attention, it is attention with a language model (i.e. what we will consider on Friday)

Transformer

Attention is all you need

	Seq2seq without attention	Seq2seq with attention	Transformer
processing within encoder	RNN/CNN	RNN/CNN	attention
processing within decoder	RNN/CNN	RNN/CNN	attention
decoder-encoder interaction	static fixed-sized vector	attention	attention

Summary

- The general idea of attention
- Attention between encoder and decoder
- Is attention alignment / is it interpretable?
- Many applications in NLP and beyond