Foundations for Natural Language Processing

Improving Encoder-Decoder, Attention

Ivan Titov
(with graphics/materials from Elena Voita)

School of informatics

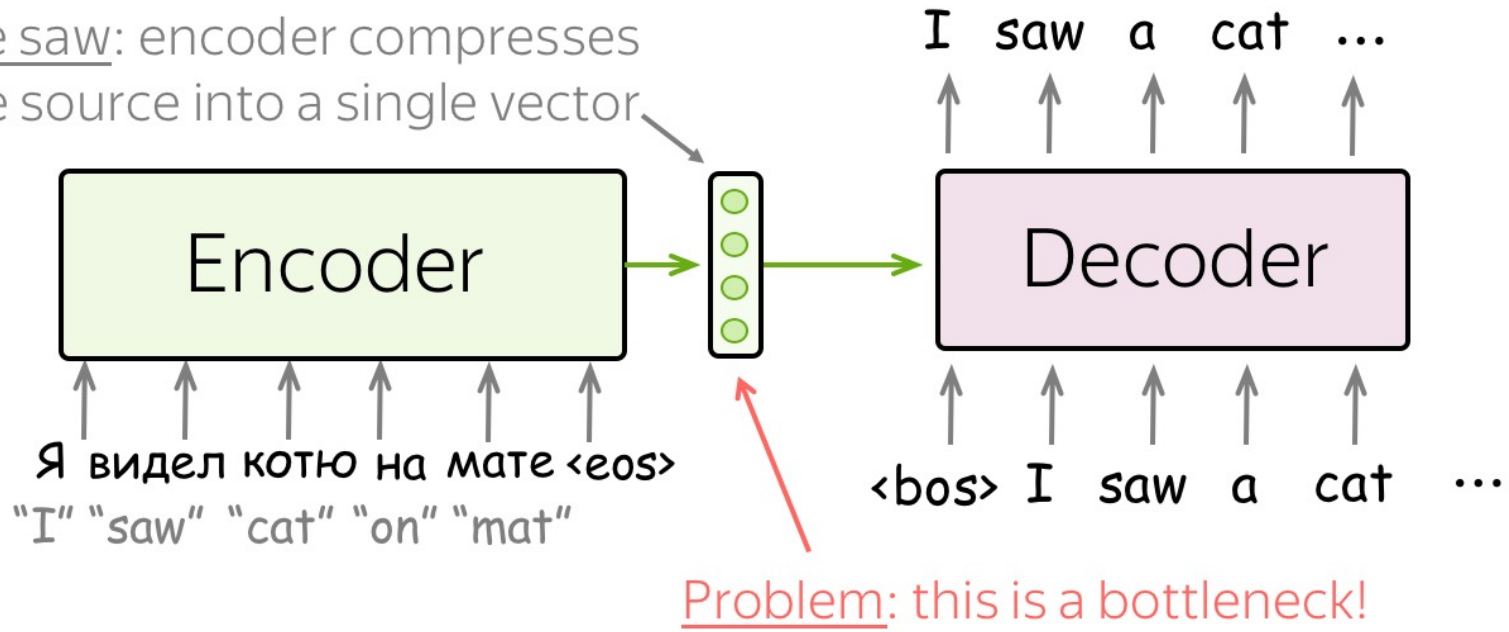
Plan for today

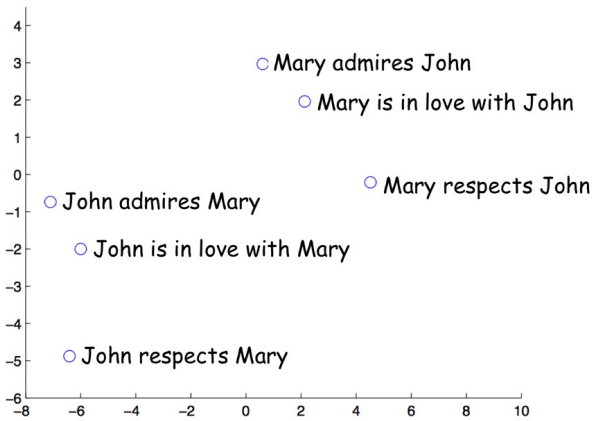
Today

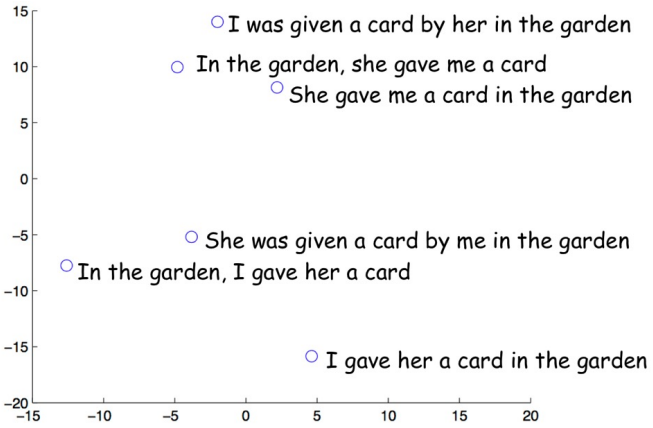
- Improving Encoder-Decoder models
- Modeling Attention in Encoder-Decoder
- Decomposable Attention Model (Pre-transformer)

The key problem with this approach

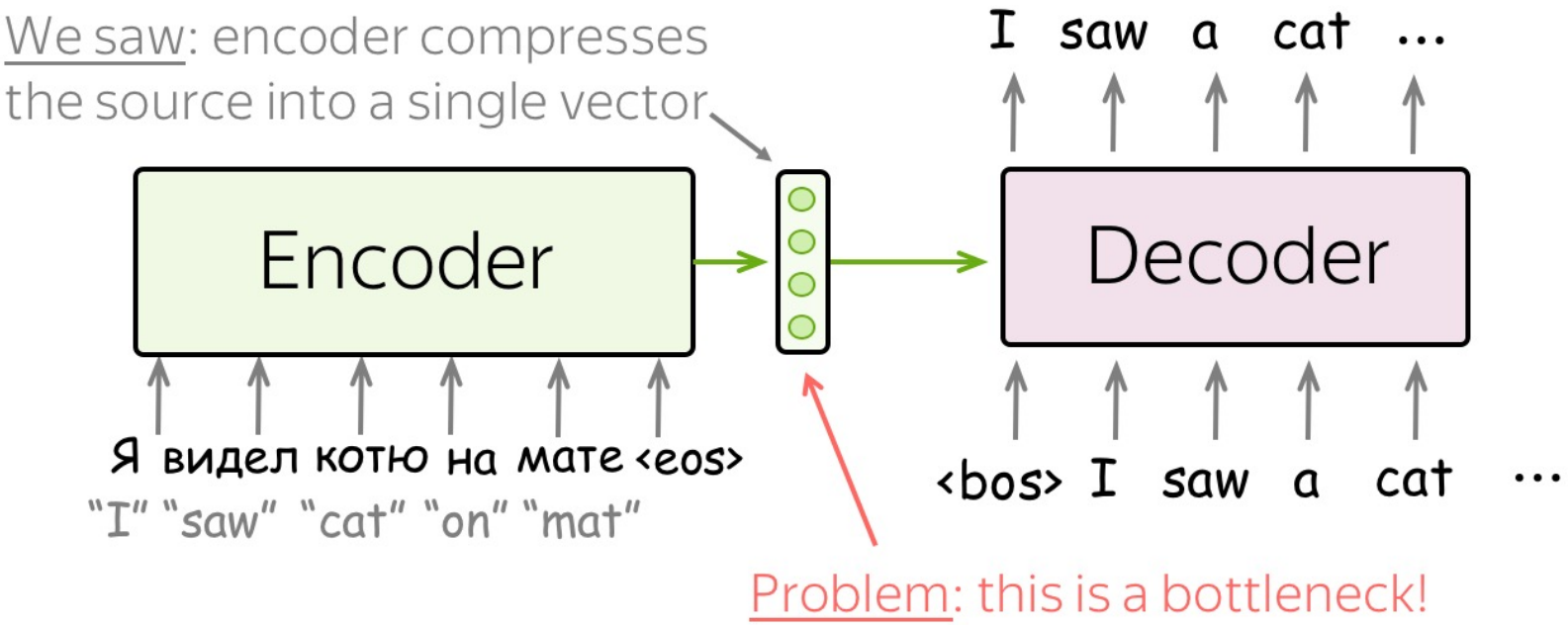
We saw: encoder compresses the source into a single vector







The key problem with this approach



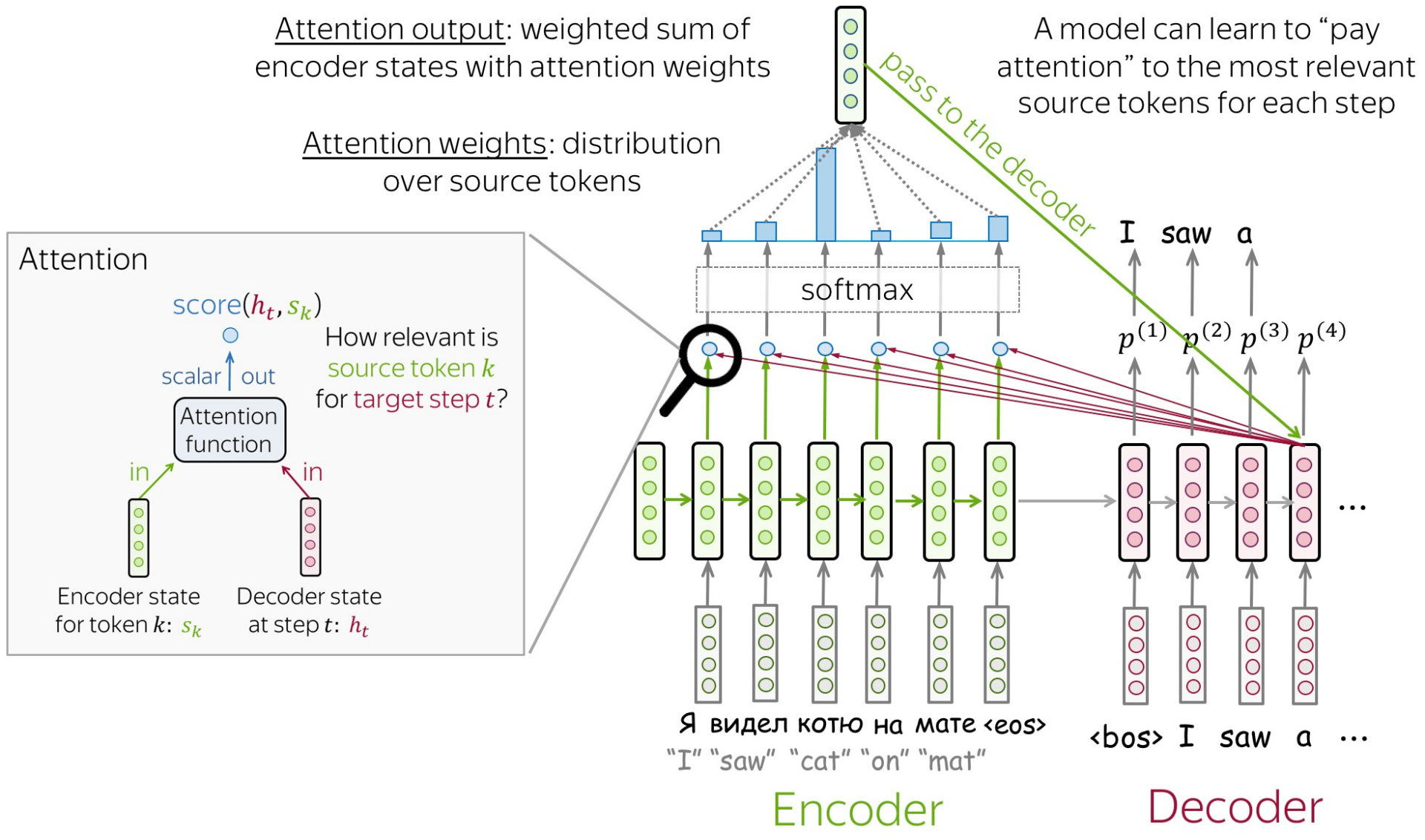
Problem: fixed source representation is suboptimal:

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

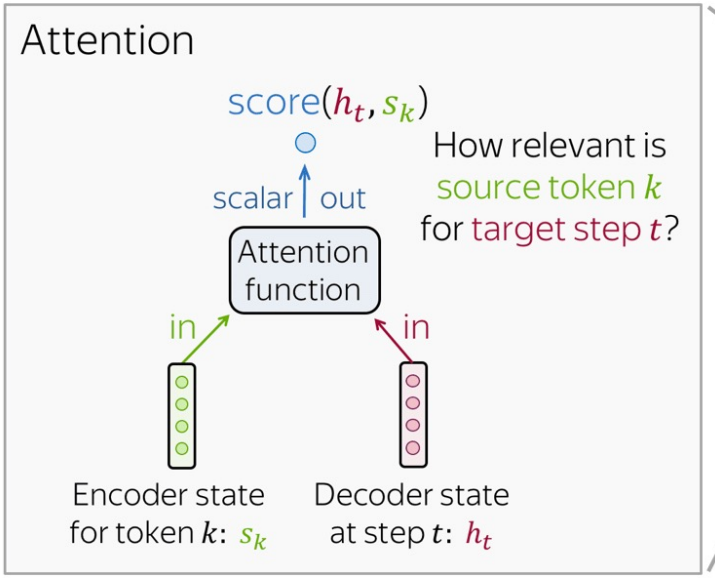
Solution: modeling "attention"

Attention: Intuition

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



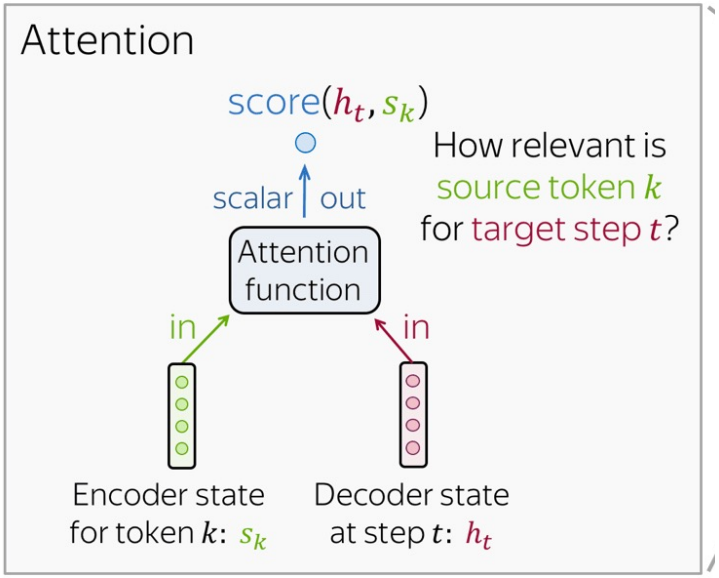
Attention



At each decoder step, attention

- receives **attention input**: a decoder state h_t and all encoder states s_1, s_2, ..., s_m;

Attention

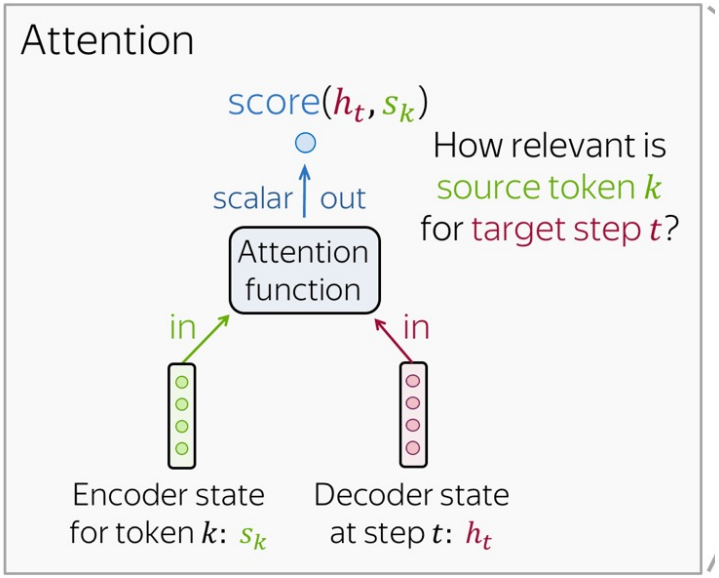


At each decoder step, attention

- 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 $score(h_t, s_k)$;

Attention

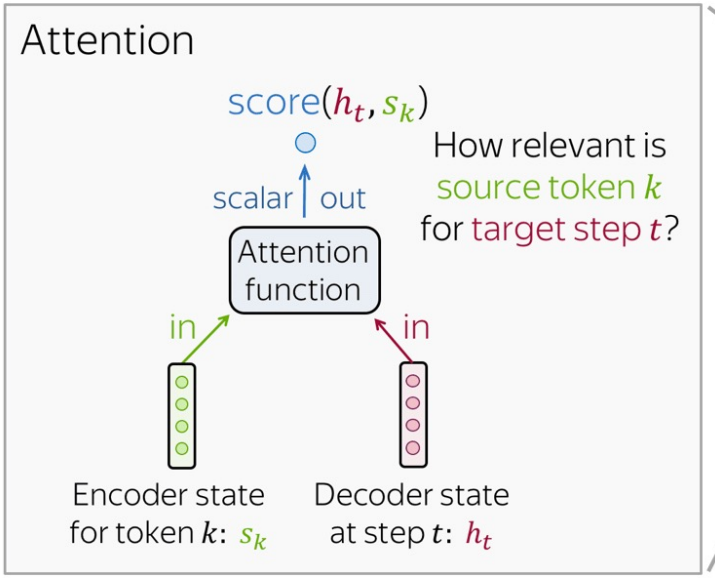


At each decoder step, attention

- 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;

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

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

"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)

Attention scores

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

"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

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)

Atte

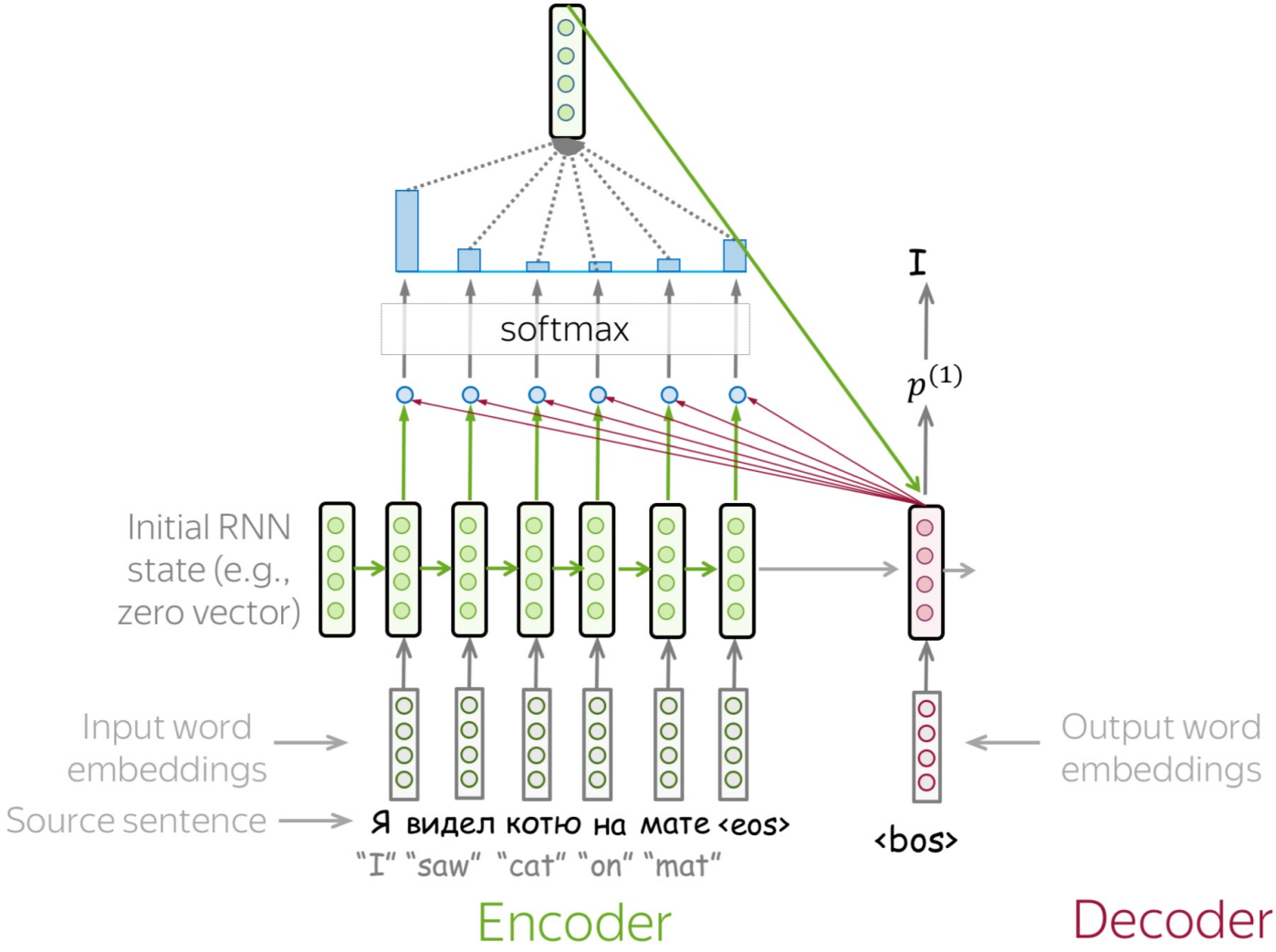
Atte

?"

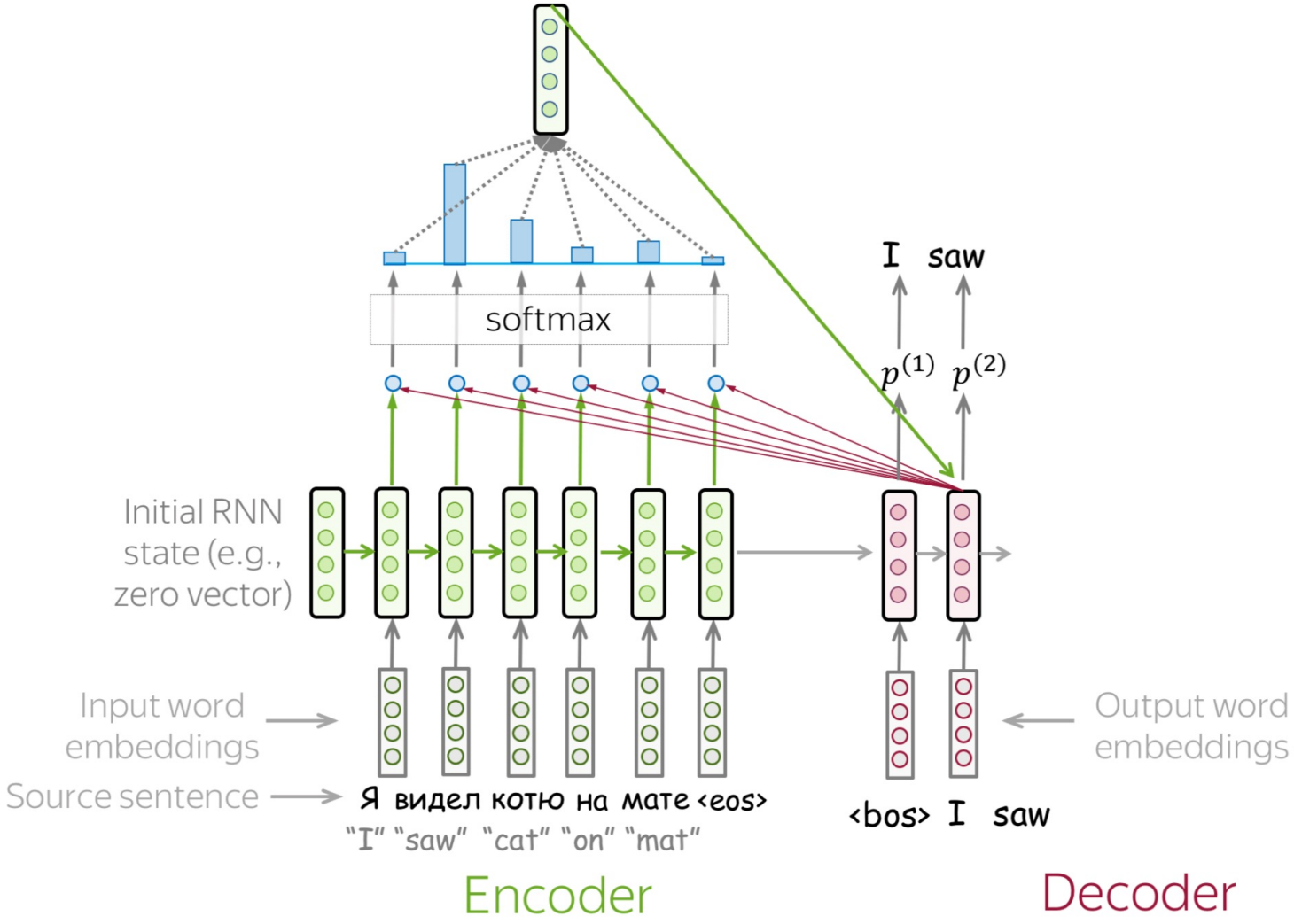
The model learns to choose relevant input tokens for every step;

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

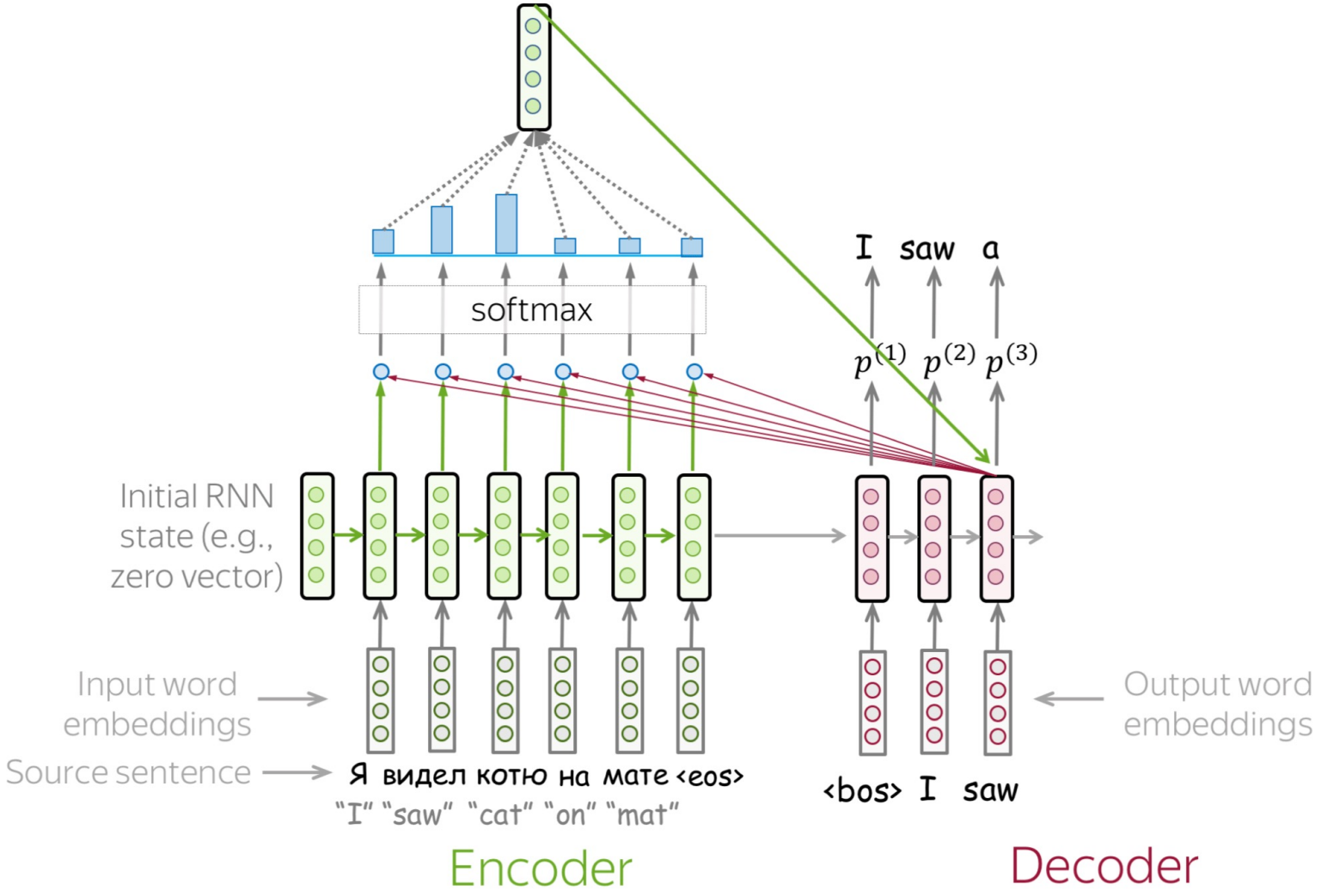
Attention: step by step



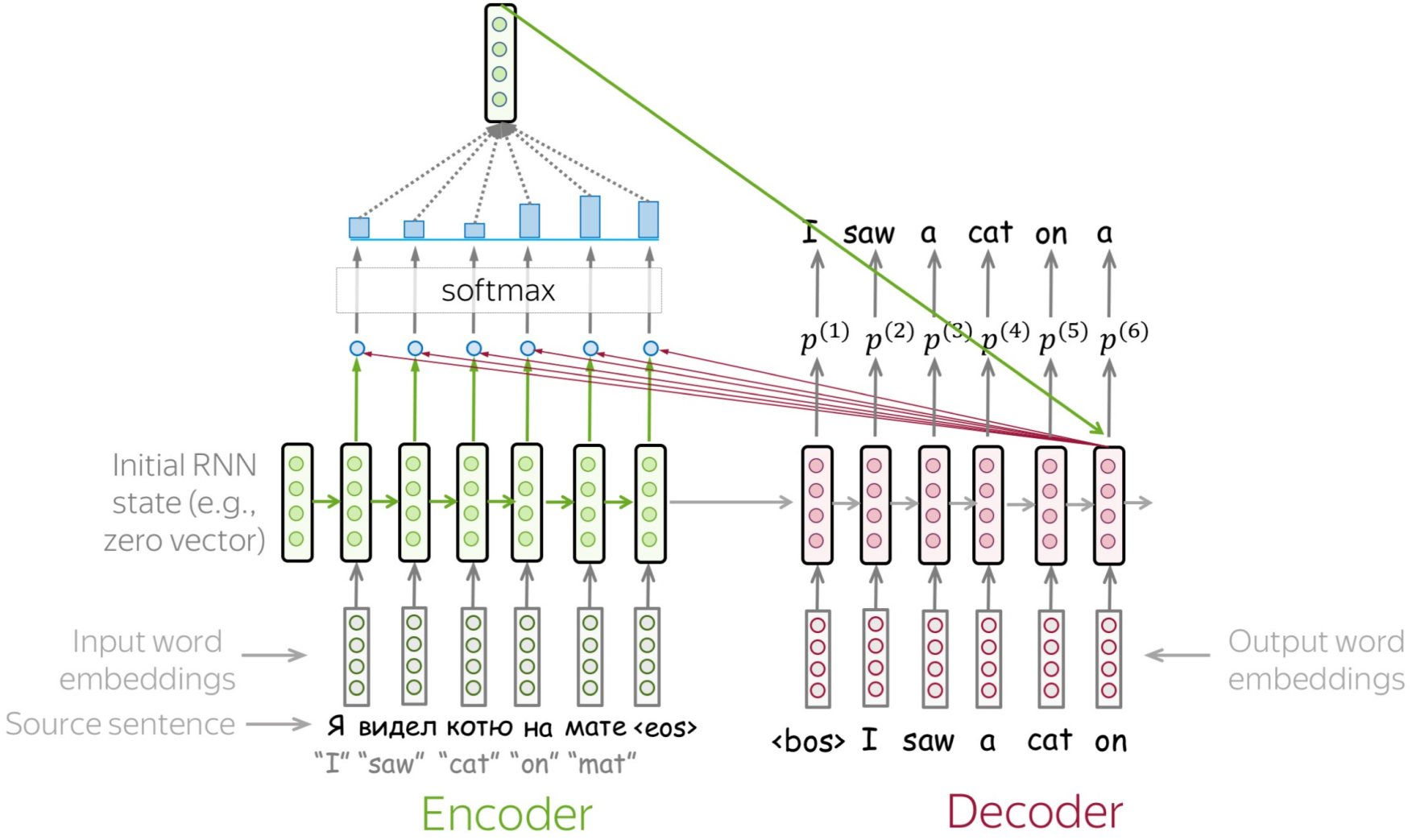
Attention: step by step



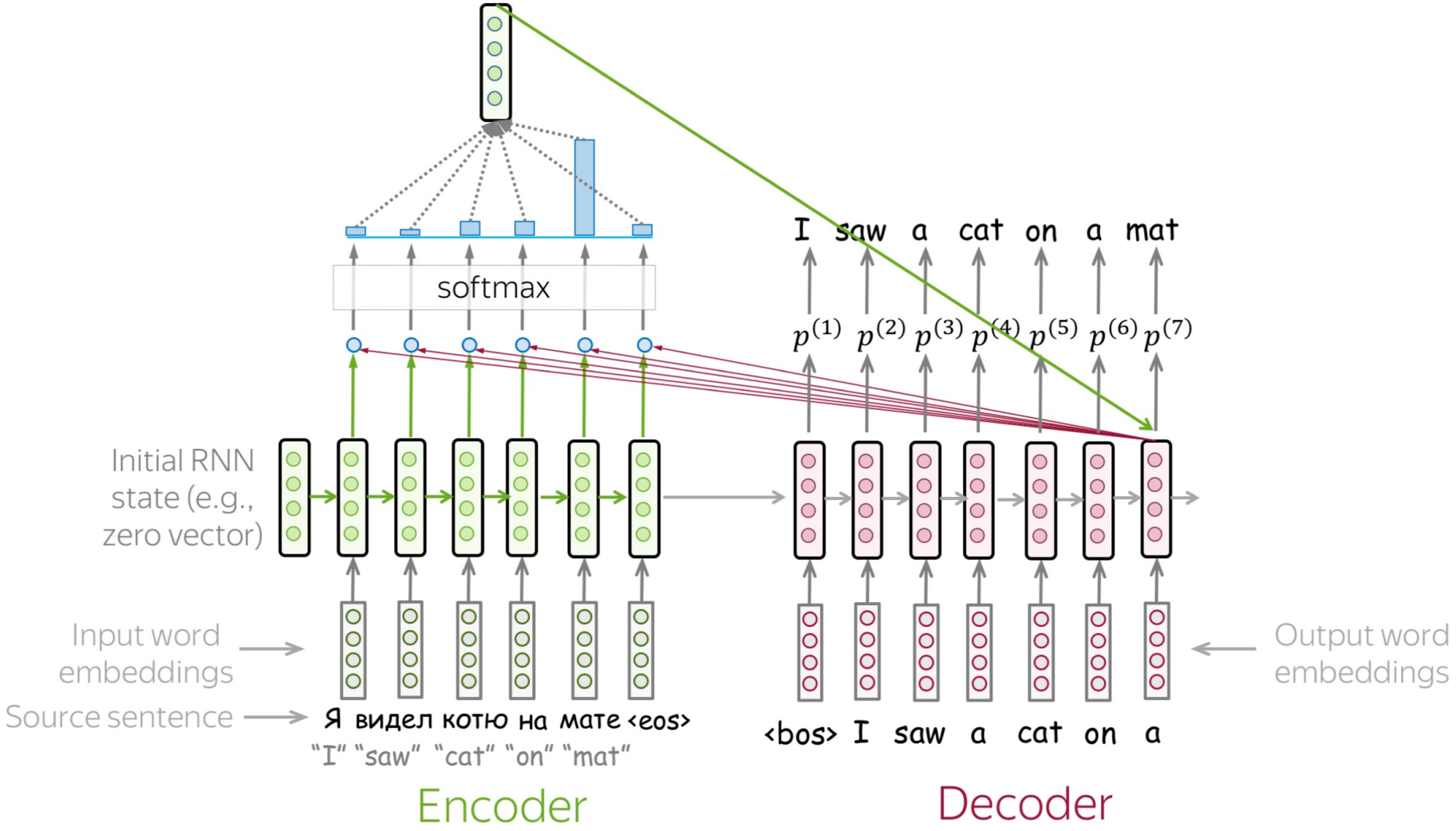
Attention: step by step



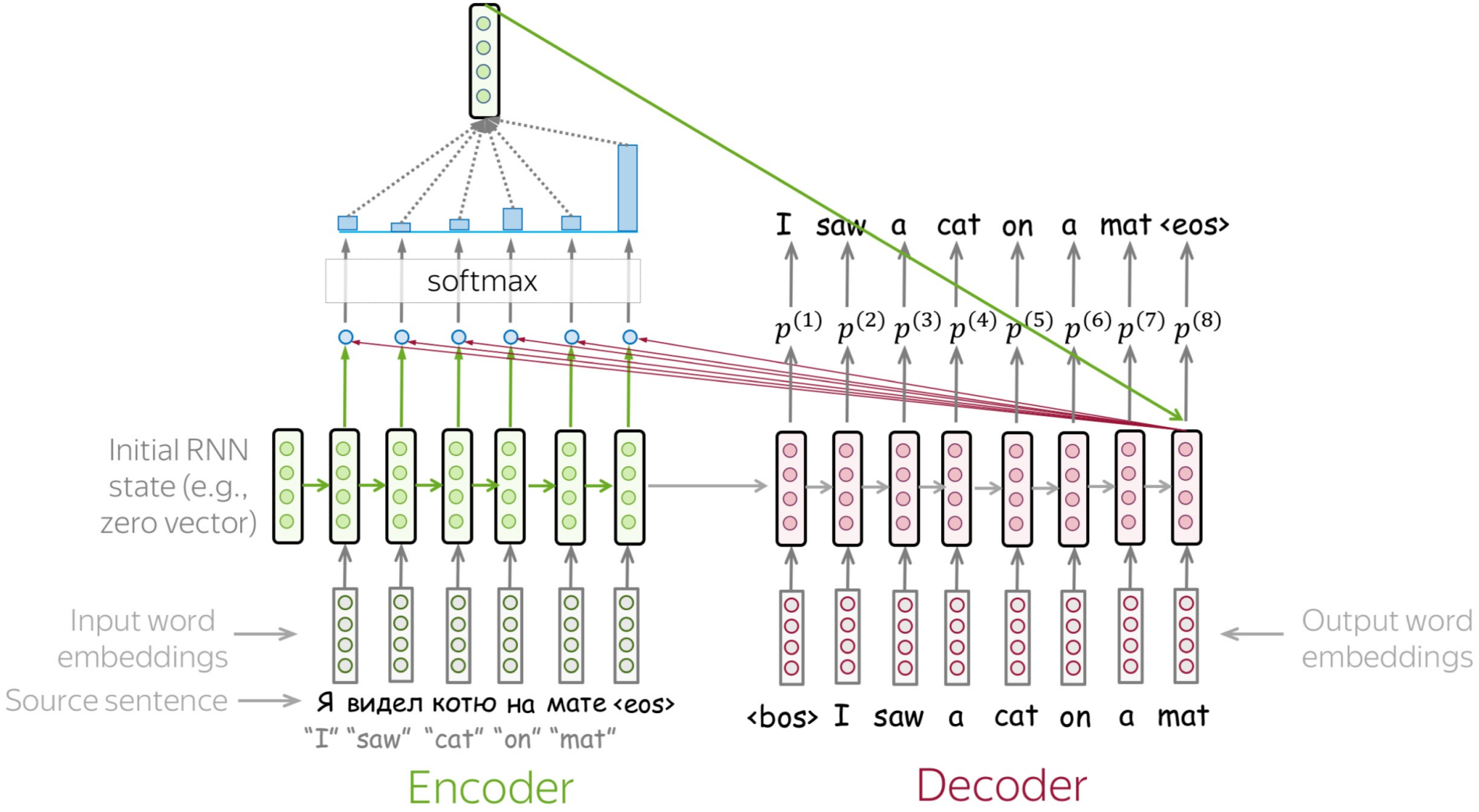
Attention: step by step



Attention: step by step



Attention: step by step

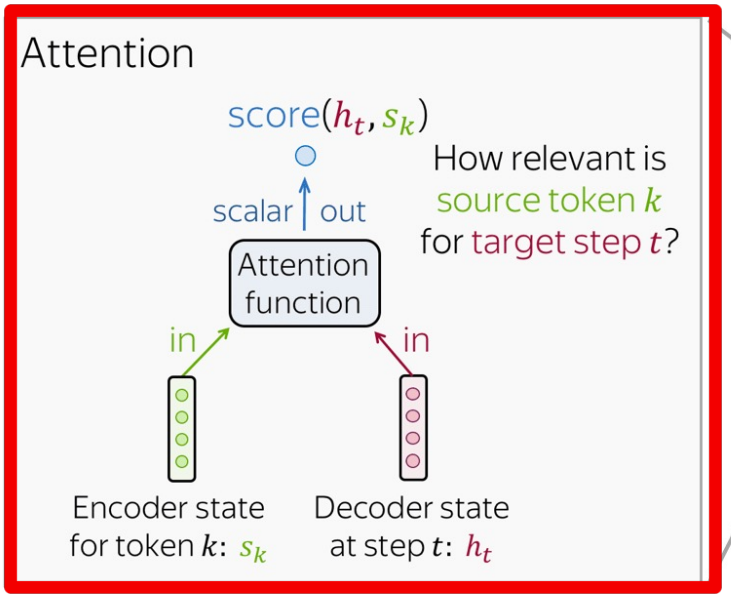


Attention

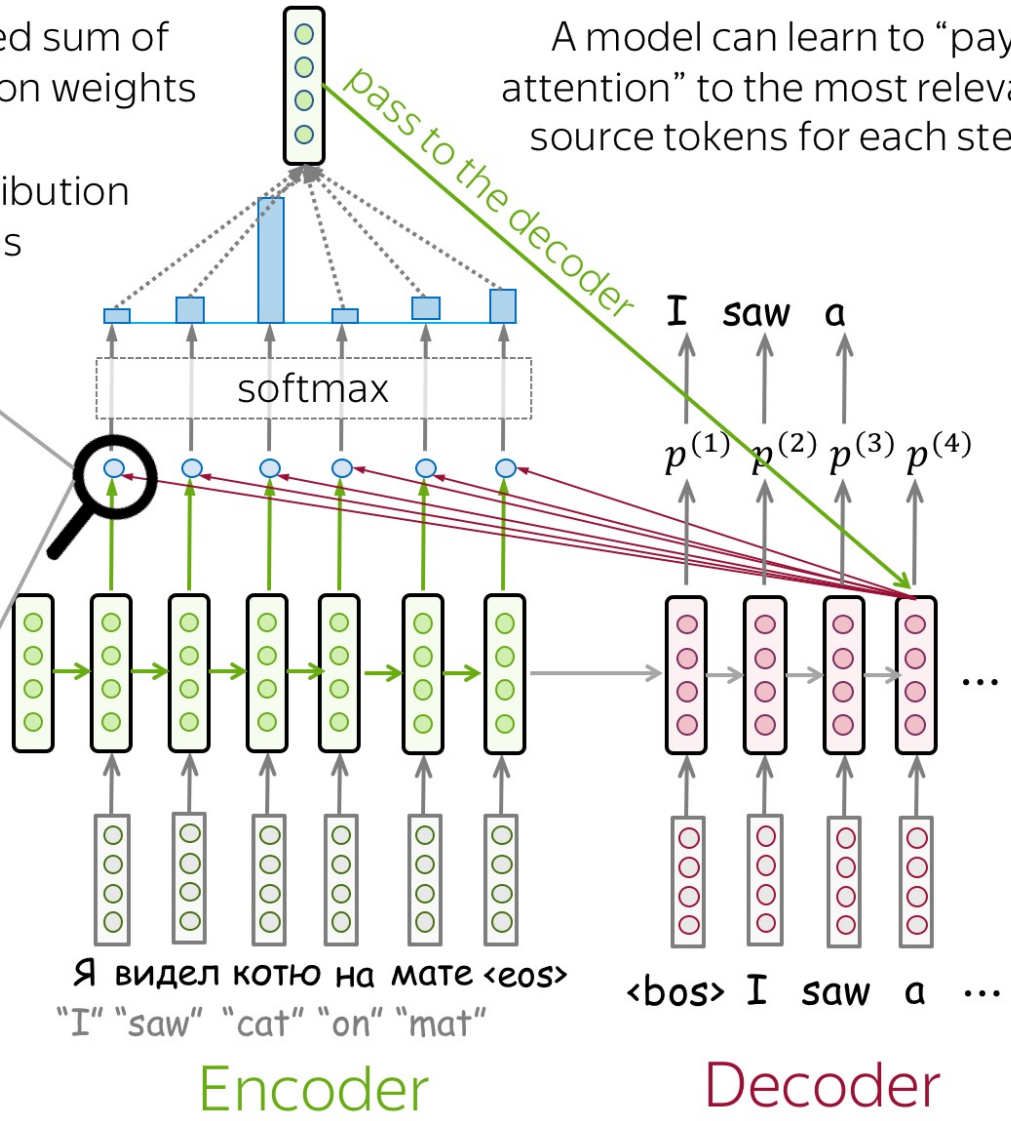
Attention output: weighted sum of encoder states with attention weights

Attention weights: distribution over source tokens

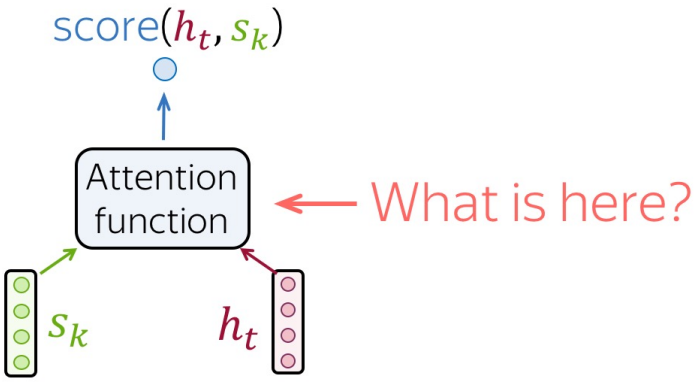
A model can learn to "pay attention" to the most relevant source tokens for each step



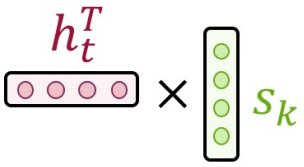
What goes in here?



How do we compute attention scores? Alternatives

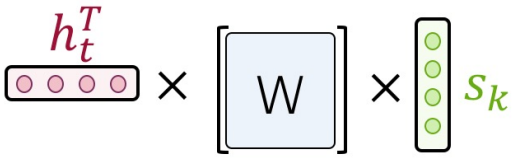


Dot-product



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

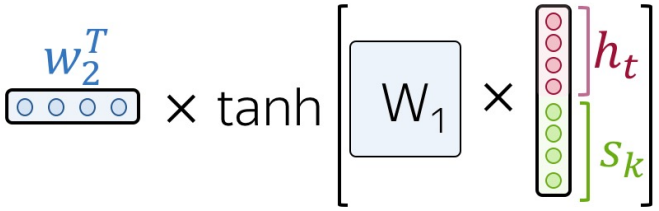
Bilinear



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

aka 'Luong attention'

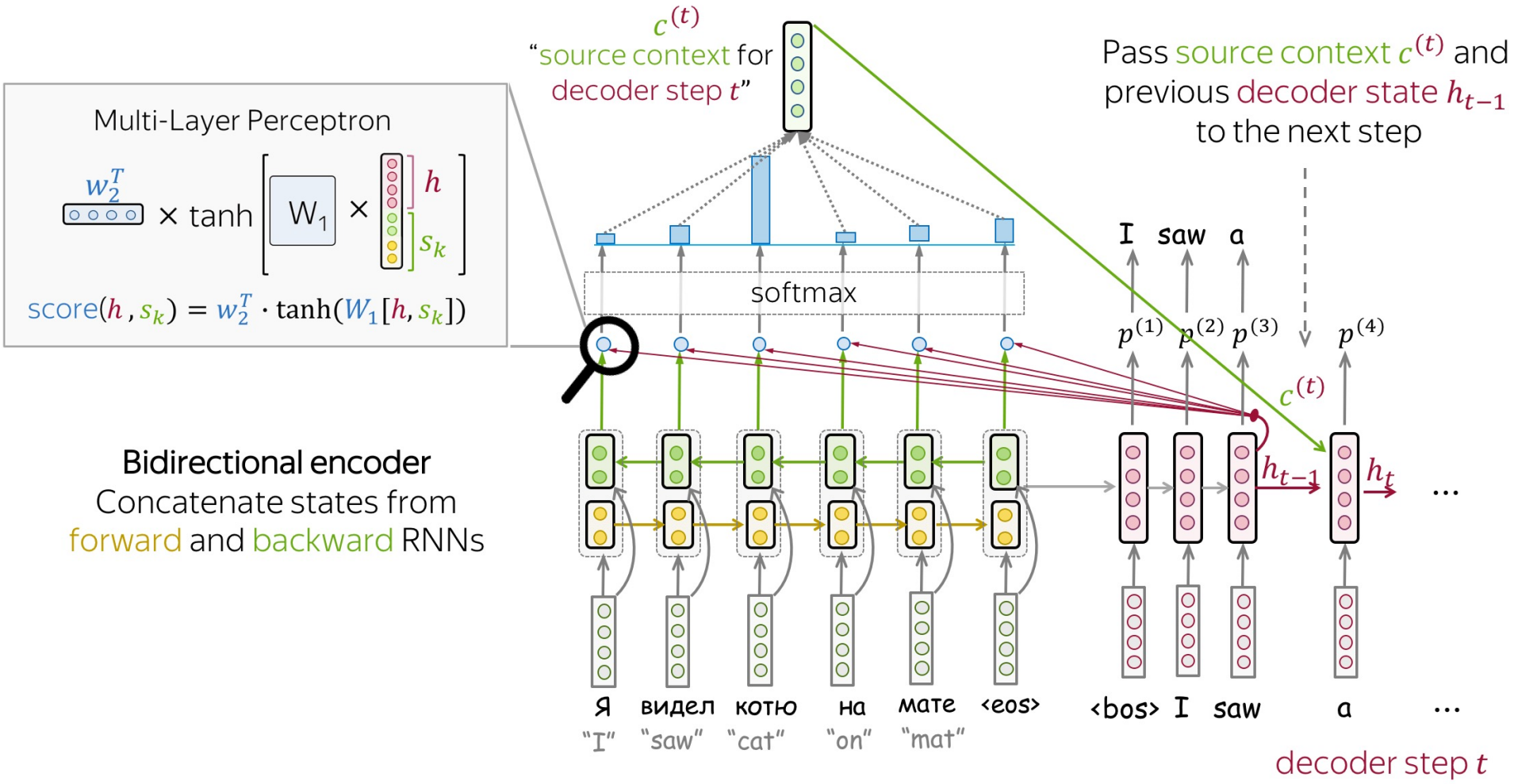
Multi-Layer Perceptron



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

aka 'Bahdanau' attention
(from the original paper)

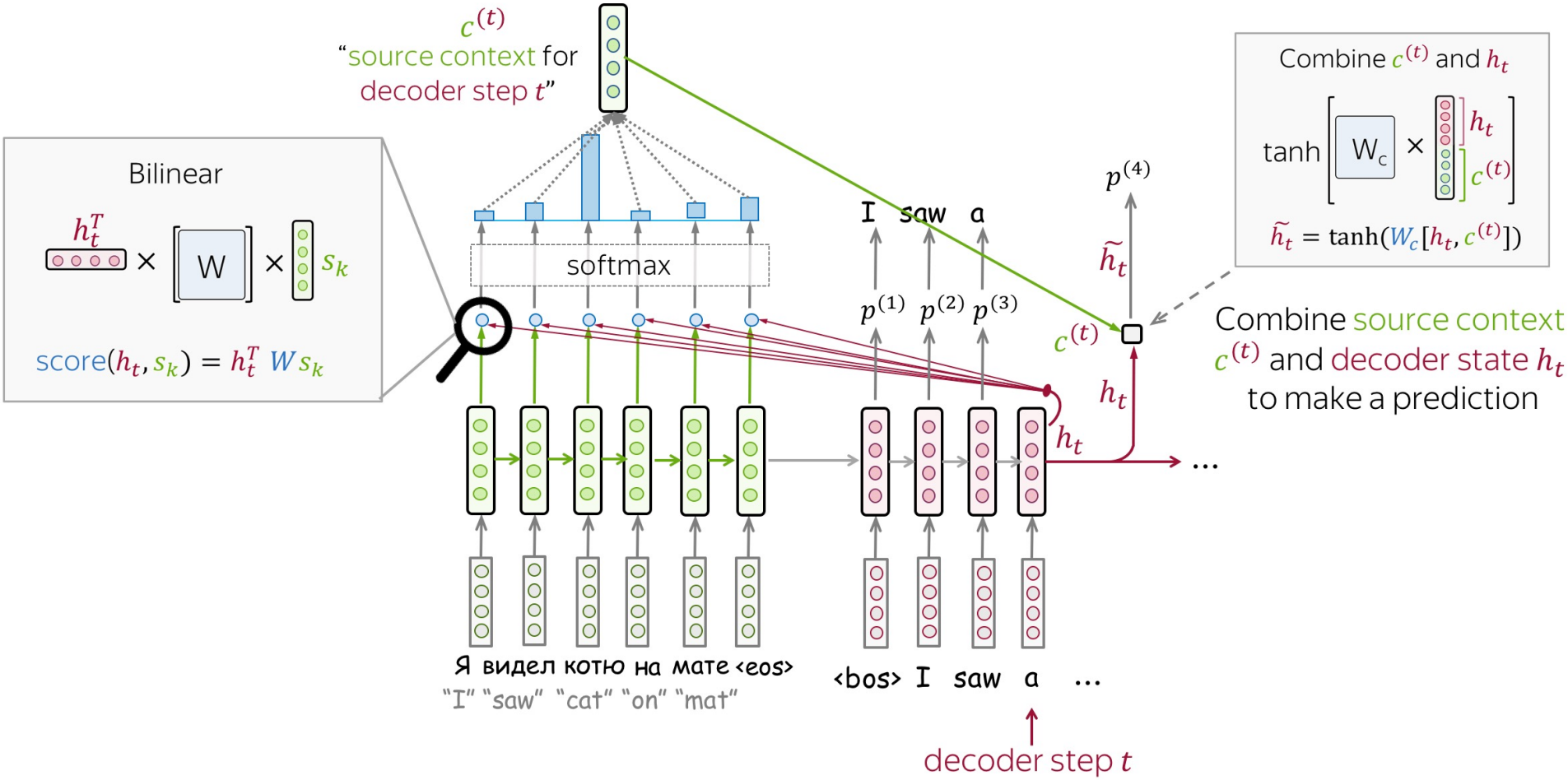
Encoder-Decoder variants: Bahdanau



Source context $c^{(t)}$ does not know what word $y^{(t-1)}$ was chosen on the previous step

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

Encoder-Decoder variants: Luong



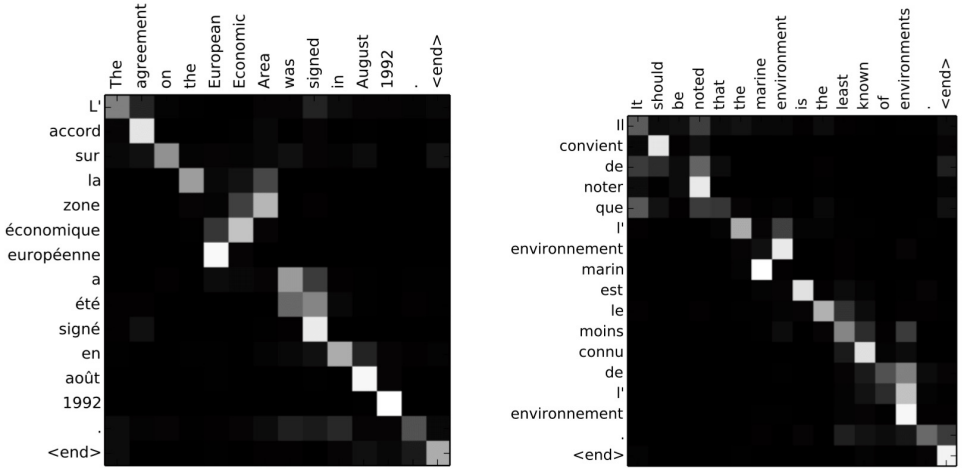
The context vector $c^{(t)}$ is not affecting future steps

https://arxiv.org/abs/1508.04025

Is attention interpretable?

For Bahdanau – before even chosing the previous token

- The attention may be perceived as modelling *alignment* between input and output words, but
  - attention is computed **before** choosing 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 or mark than nothing is relevant)
  - attention is (?) not an explanation



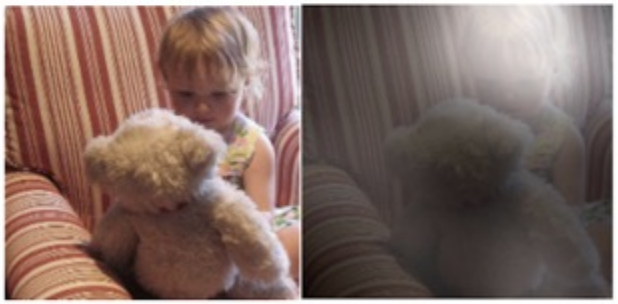
Enoder-decoders and attention are a very general idea

For example, caption generation (encoder process the image, not a sentence)

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"

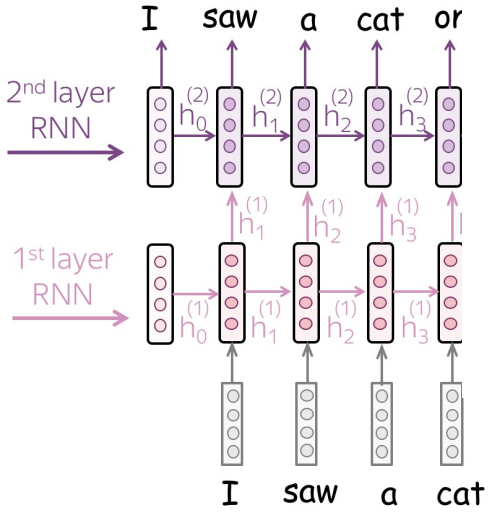
$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$$

↑ "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 be affected change

Attention is not necessarily faithful

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



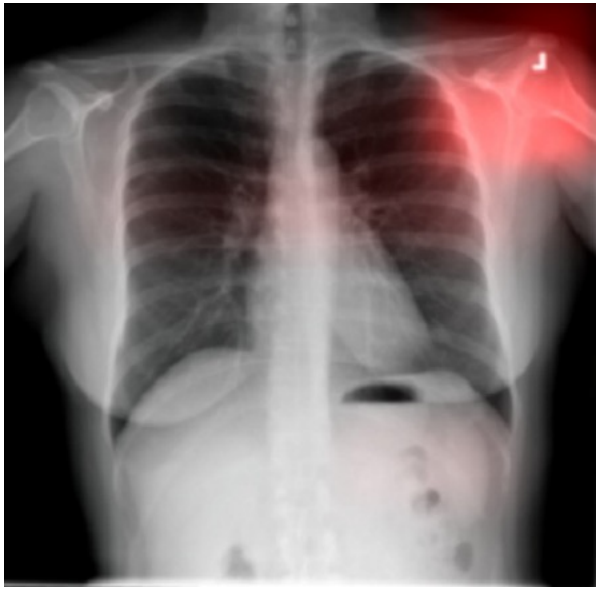
There are attribution methods (e.g., norm x gradient, layer-wise relevant propagation, integrated gradients, attention flow, zero valuing, …) which attempt to address these issues, but they also have their own pitfalls

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 and/or data

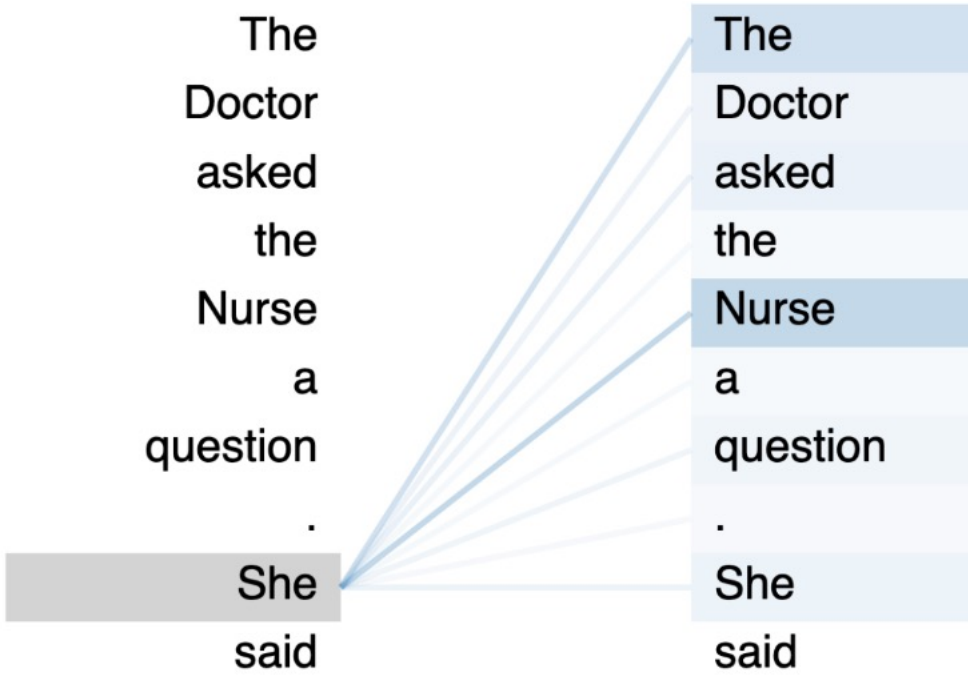
Detecting pneumonia from an x-ray, the model 'looks' at the corner of an image, rather than at lungs. Any thoughts on why?



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

Decomposable attention (pre-Transformer)

A Decomposable Attention Model for Natural Language Inference

Ankur P. Parikh
Google
New York, NY

Oscar Täckström
Google
New York, NY

Dipanjan Das
Google
New York, NY

Jakob Uszkoreit
Google
Mountain View, CA

{aparikh,oscart,dipanjand,uszkoreit}@google.com

Abstract

We propose a simple neural architecture for natural language inference. Our approach uses attention to decompose the problem into subproblems that can be solved separately, thus making it trivially parallelizable. On the Stanford Natural Language Inference (SNLI) dataset, we obtain state-of-the-art results with almost an order of magnitude fewer parameters than previous work and without relying on any word-order information. Adding intra-sentence attention that takes a minimum amount of order into account yields further improvements.

1 Introduction

LSTMs henceforth) with the goal of deeper sentence comprehension. While these approaches have yielded impressive results, they are often computationally very expensive, and result in models having millions of parameters (excluding embeddings).

Here, we take a different approach, arguing that for natural language inference it can often suffice to simply align bits of local text substructure and then aggregate this information. For example, consider the following sentences:

- *Bob is in his room, but because of the thunder and lightning outside, he cannot sleep.*
- *Bob is awake.*
- *It is sunny outside.*

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

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including

Natural Language Inference (aka textual entailment)

Classification task:

Premise: Bob is in his room, but because of the thunder and lightning outside, he cannot sleep.

Hypothesis: It is sunny outside.

Is H entailed by P (i.e. logically follows), contradicts P or neither ?

Natural Language Inference (aka textual entailment)

Classification task:

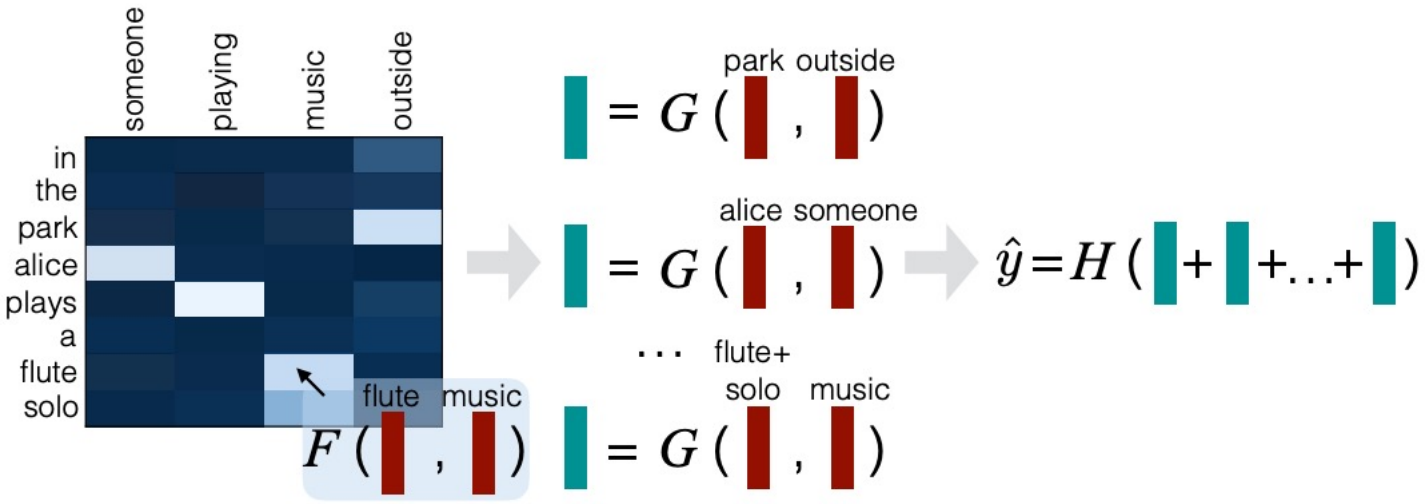
Premise: Bob is in his room, but because of the thunder and lightning outside, he cannot sleep.

Hypothesis: It is sunny outside.

Is H entailed by P (i.e. logically follows), contradicts P or neither ?

The idea: the prediction can be done (or approximated) by matching / comparing parts of H and P

Classification task



Steps:

(1) attend / align (F),

(2) compare (G)

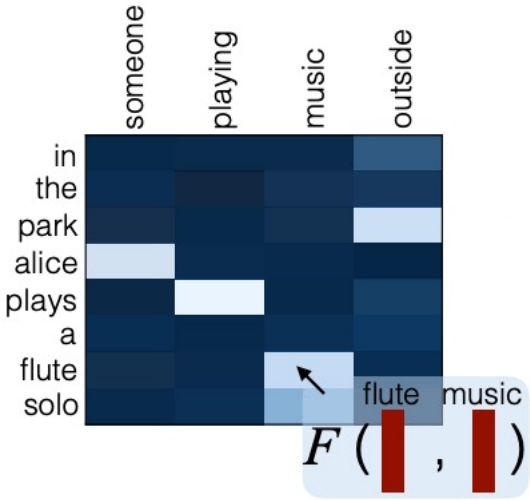
(3) aggregate and classify (H)

Step 1: Attend (align)

Embedding of i-th token in P

$$e_{ij} := F(a_i)^T F(b_j)$$

Embedding of j-th token in T



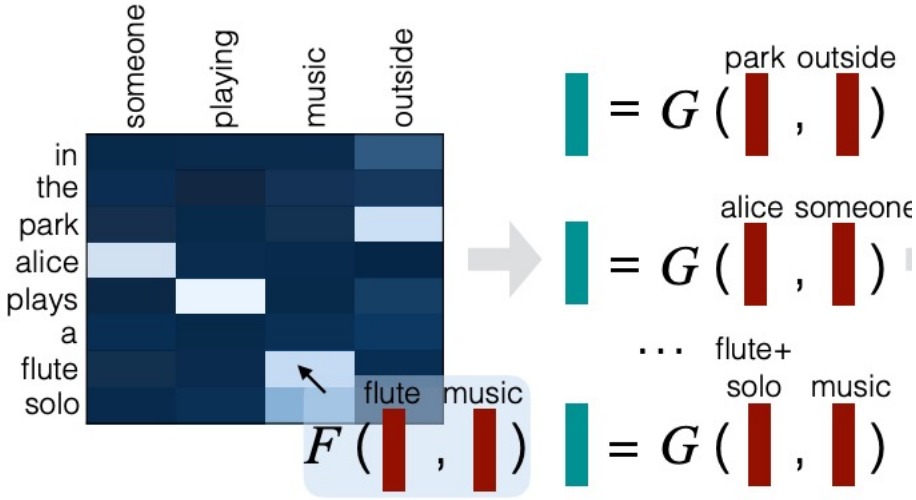
Representation of tokens in H aligned to i-th token in P:

$$\beta_i := \sum_{j=1}^{\ell_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_b} \exp(e_{ik})} b_j$$

The same in the opposite direction:

$$\alpha_j := \sum_{i=1}^{\ell_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_a} \exp(e_{kj})} a_i$$

Step 2: Compare



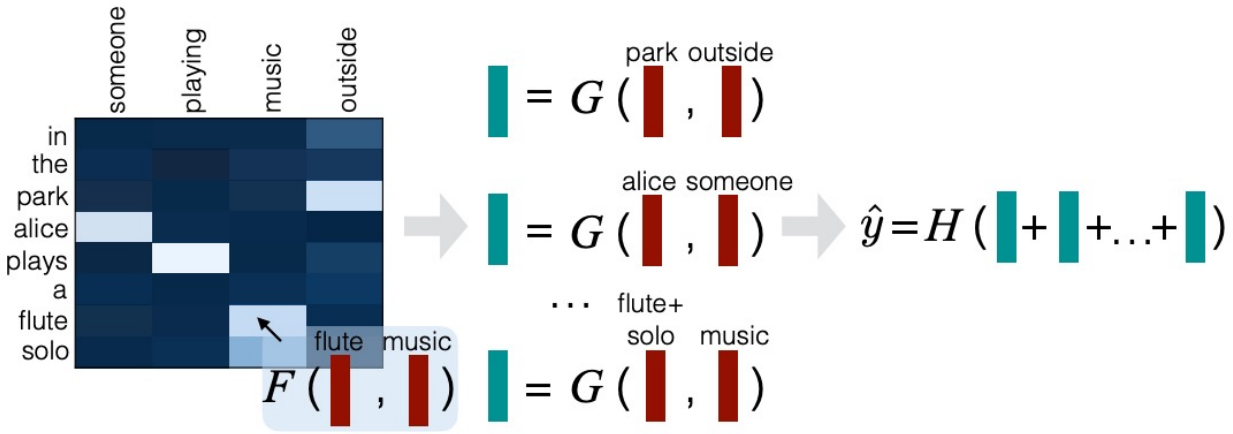
$$\mathbf{v}_{1,i} := G([a_i, \beta_i])$$
$$\mathbf{v}_{2,j} := G([b_j, \alpha_j])$$

← a representation of i-th token in H, informed by all tokens in T

G is just a feedforward neural network, *v* is a vector

Note: the input to G is concatenation, so not invariant to switching the arguments around

Step 2: Aggregate / classify

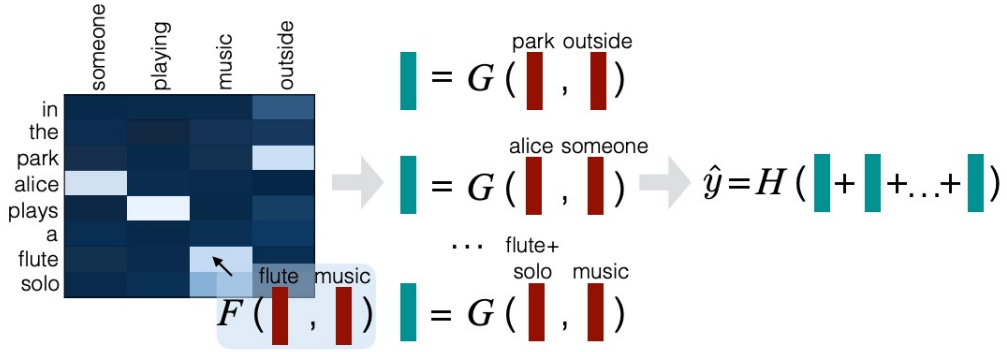


$$\mathbf{v}_1 = \sum_{i=1}^{\ell_a} \mathbf{v}_{1,i} \quad , \quad \mathbf{v}_2 = \sum_{j=1}^{\ell_b} \mathbf{v}_{2,j} \, .$$

$$H([\mathbf{v}_1, \mathbf{v}_2])$$

Just a linear classification layer, followed by softmax to get a distribution over 3 classes

All together now



Softmax

Linear layer

Sum pooling

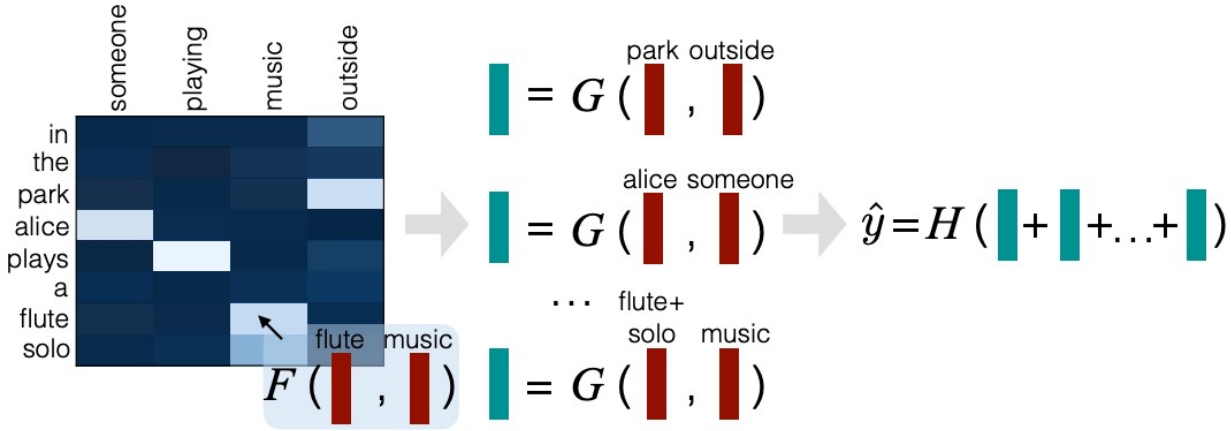
Feedforward layer

Attention layer

Embeddings

In this layer you produce a representations of each token in H informed by every token in T (and vice versa)

All together now



This simple and very fast to train model without RNNs or CNNs components achieved very strong results on the NLP task

(a couple of small extra tricks were needed to inform the model about word position + to better handle unaligned tokens; we will discuss their counterparts in the Transformer)

Summary

- The general idea of attention
- Attention between encoder and decoder
- Does attention represent an alignment? / is it interpretable?
- Many applications in NLP and beyond, often as a crucial component of Transformers but not only
- We made the first step towards Transformers which we will discuss after the flexible learning week