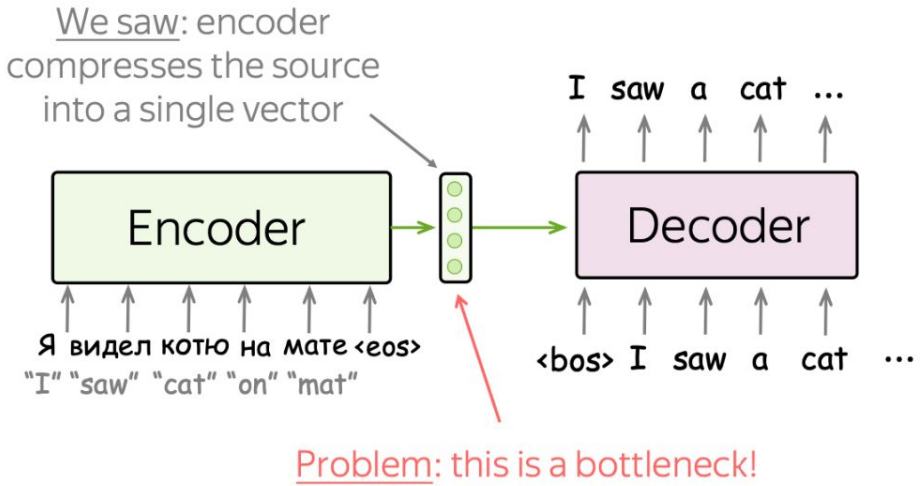
Transformers in Deep Learning

Lecture 2

Lecture 1 Recap: RNN bottelneck

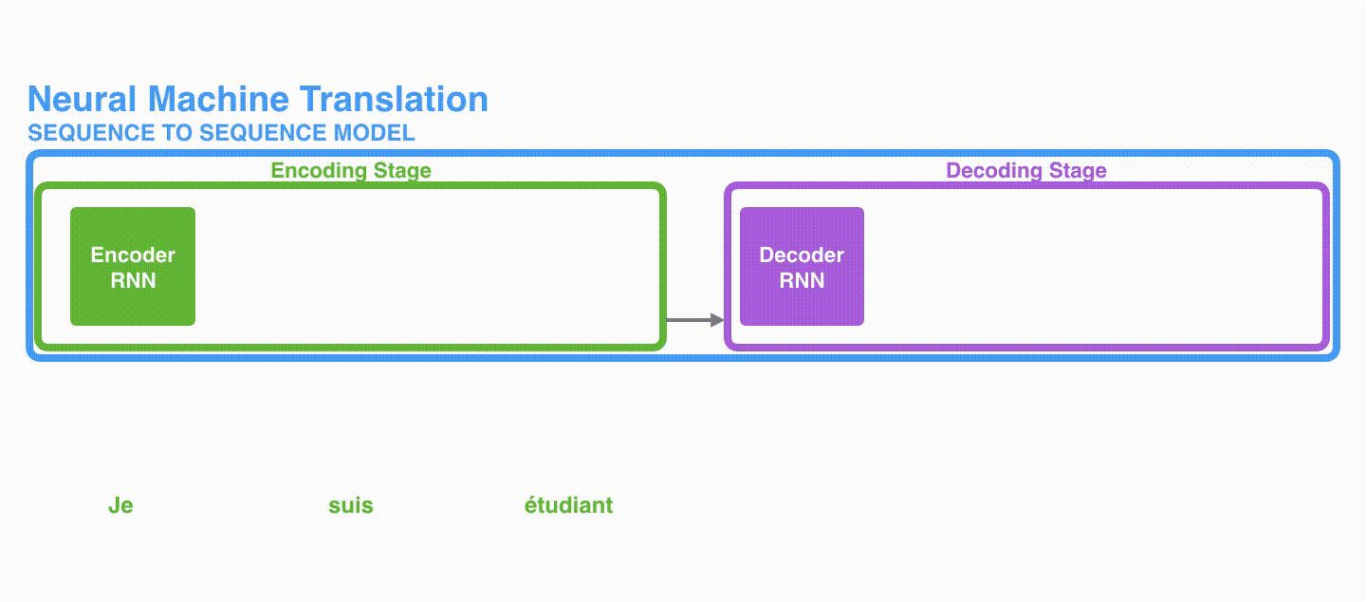
Fixed source representation is bad:

- for encoder, it is hard to compress a sentence
- for decoder, different information may be needed at different steps



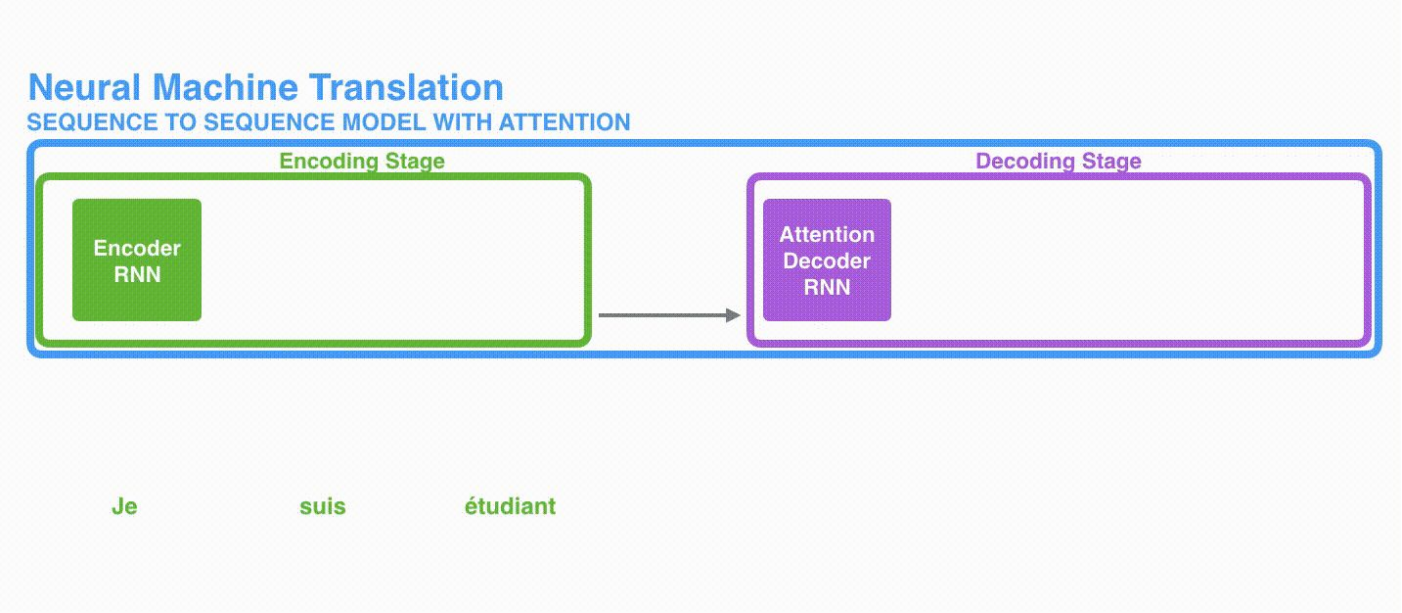
https://lena-voita.github.io/nlp_course.html

Lecture 1 Recap: RNN Seq2Seq



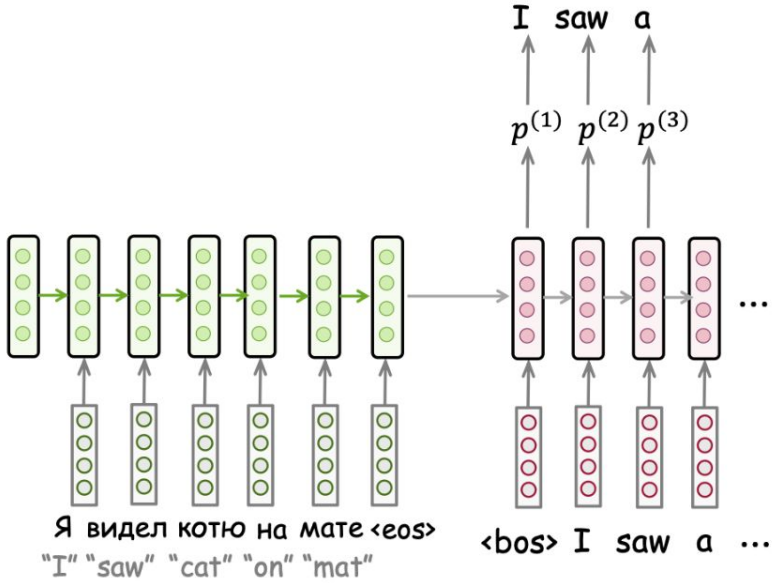
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Seq2Seq models: Encoder-Decoder Attention



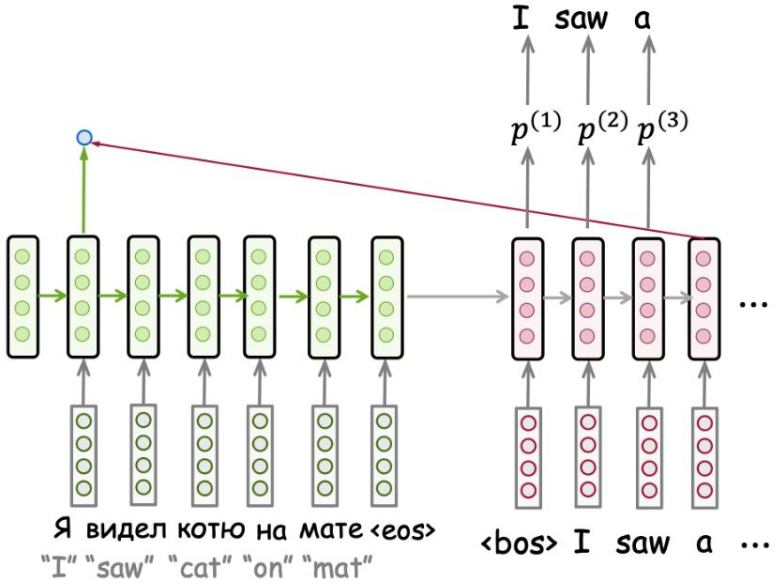
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Seq2Seq models: Encoder-Decoder Attention



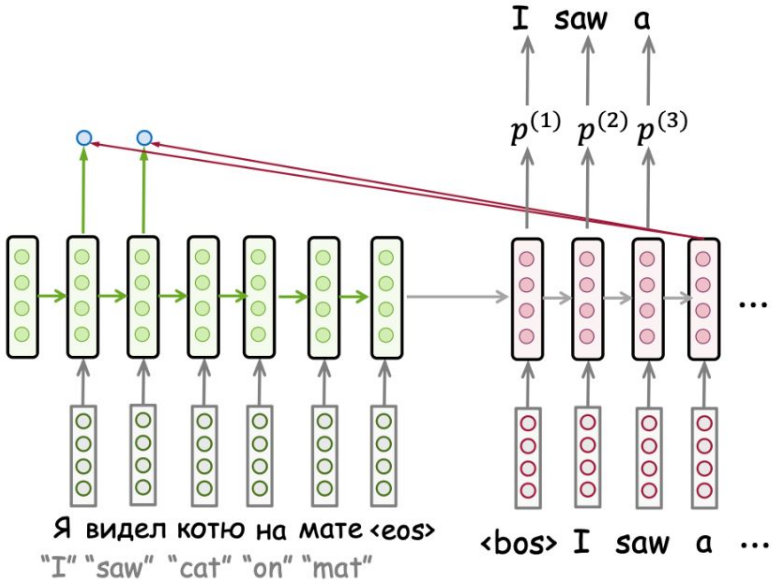
https://lena-voita.github.io/nlp_course.html

Seq2Seq models: Encoder-Decoder Attention



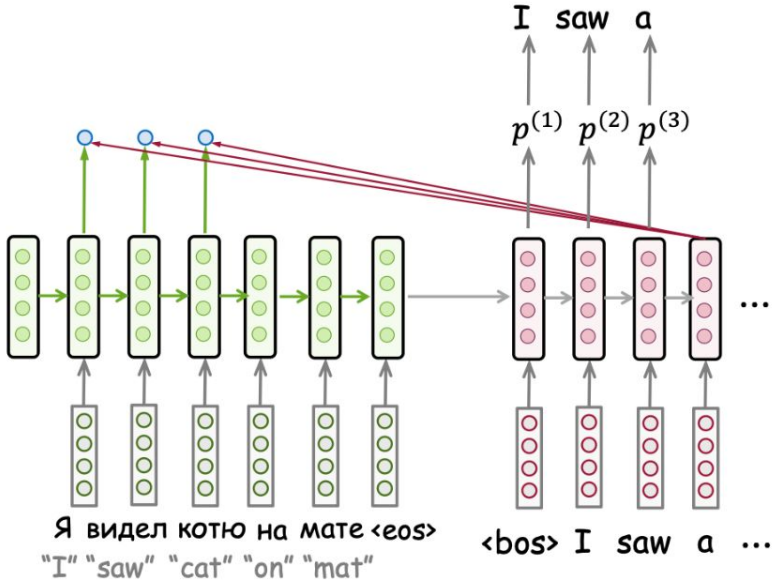
https://lena-voita.github.io/nlp_course.html

Seq2Seq models: Encoder-Decoder Attention



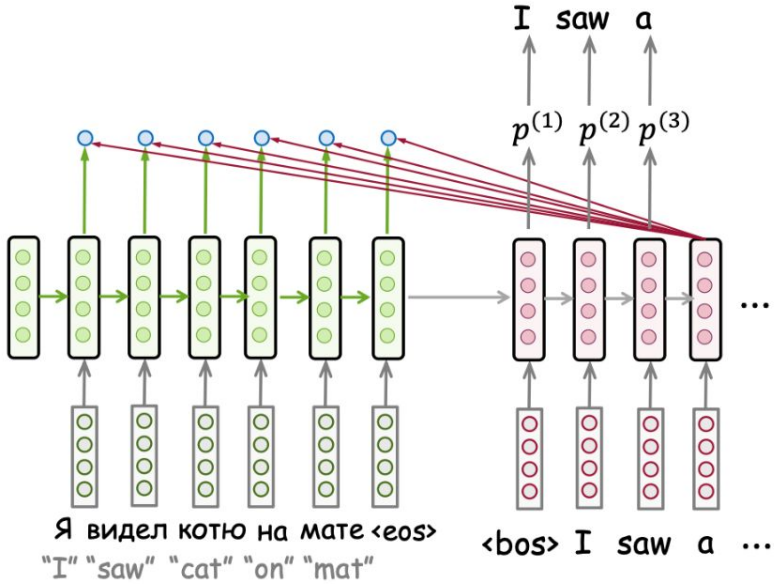
https://lena-voita.github.io/nlp_course.html

Seq2Seq models: Encoder-Decoder Attention



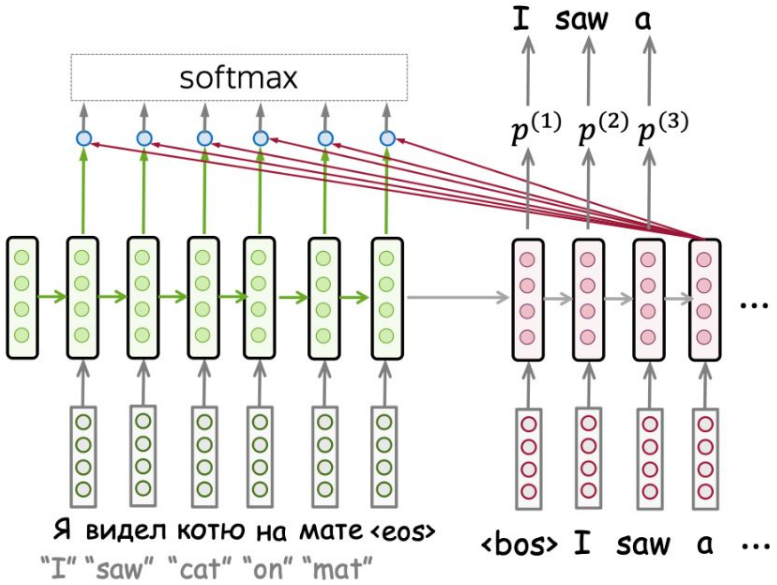
https://lena-voita.github.io/nlp_course.html

Seq2Seq models: Encoder-Decoder Attention



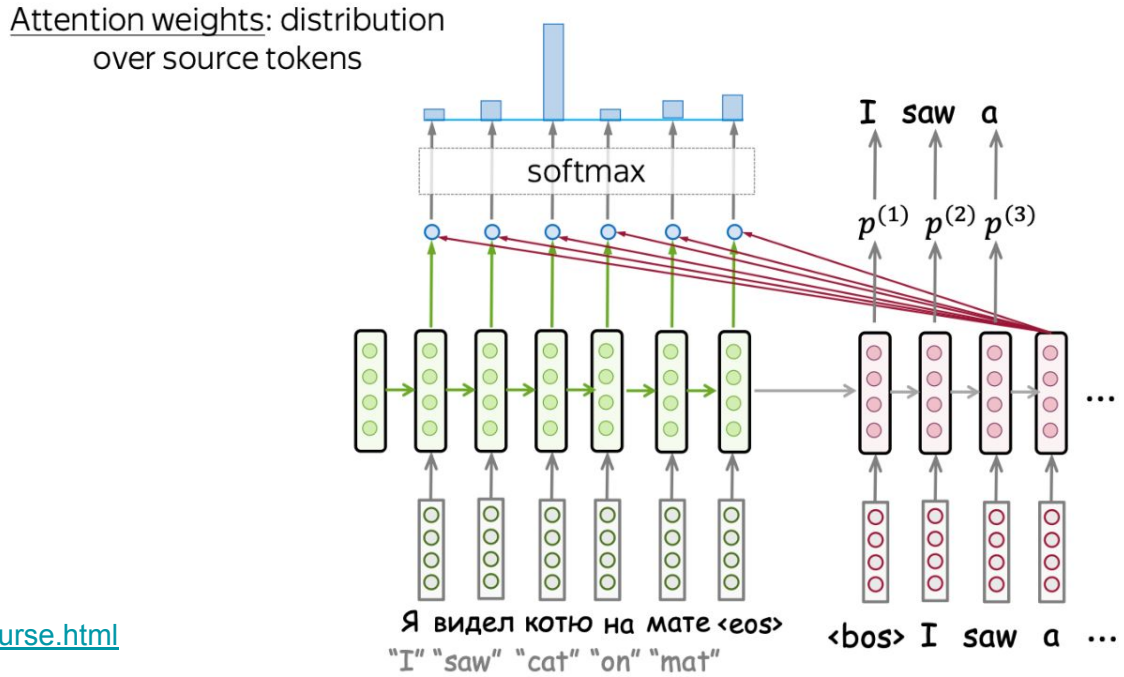
https://lena-voita.github.io/nlp_course.html

Seq2Seq models: Encoder-Decoder Attention



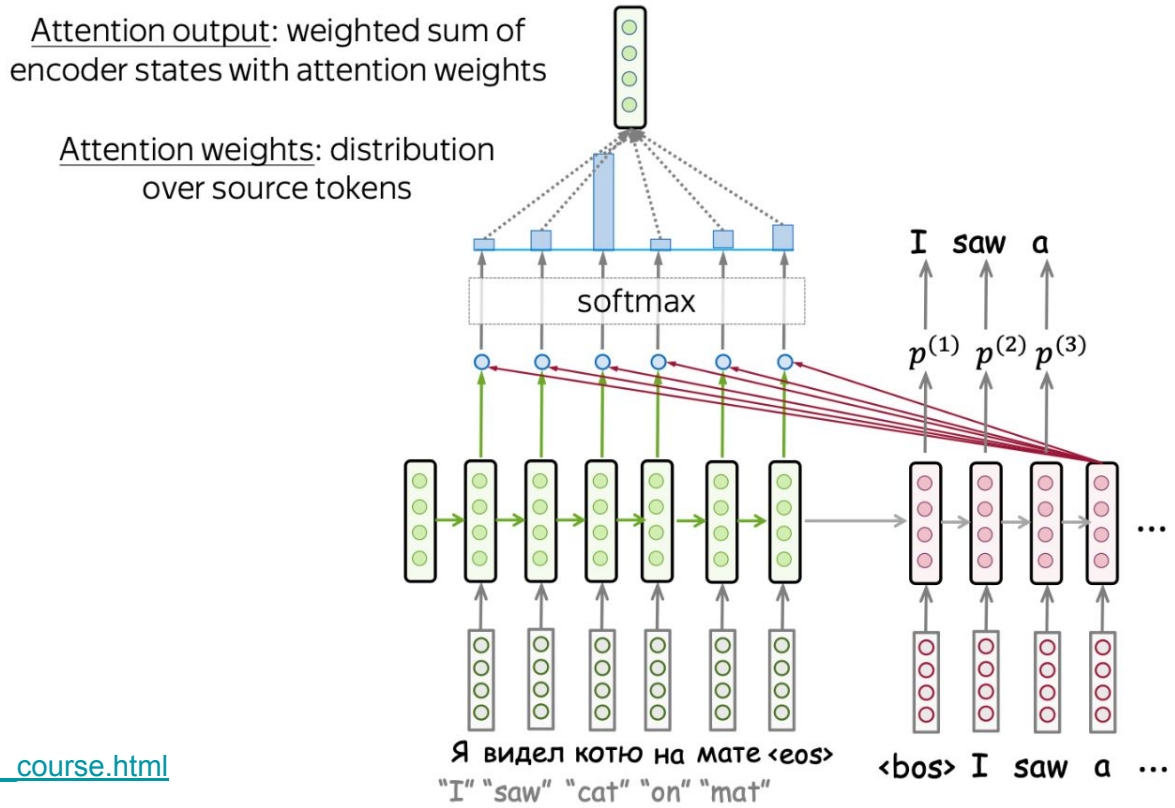
https://lena-voita.github.io/nlp_course.html

Seq2Seq models: Encoder-Decoder Attention



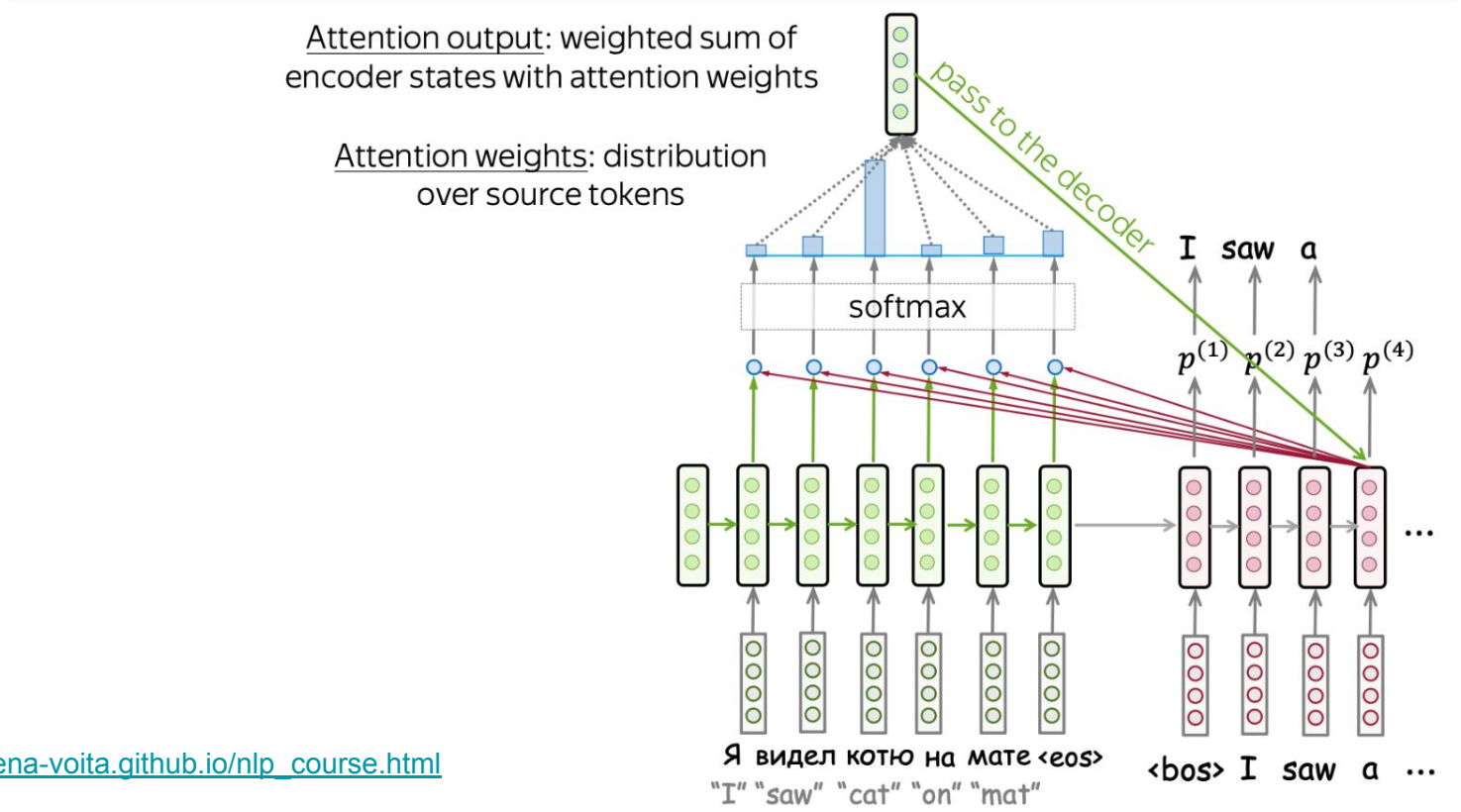
https://lena-voita.github.io/nlp_course.html

Seq2Seq models: Encoder-Decoder Attention



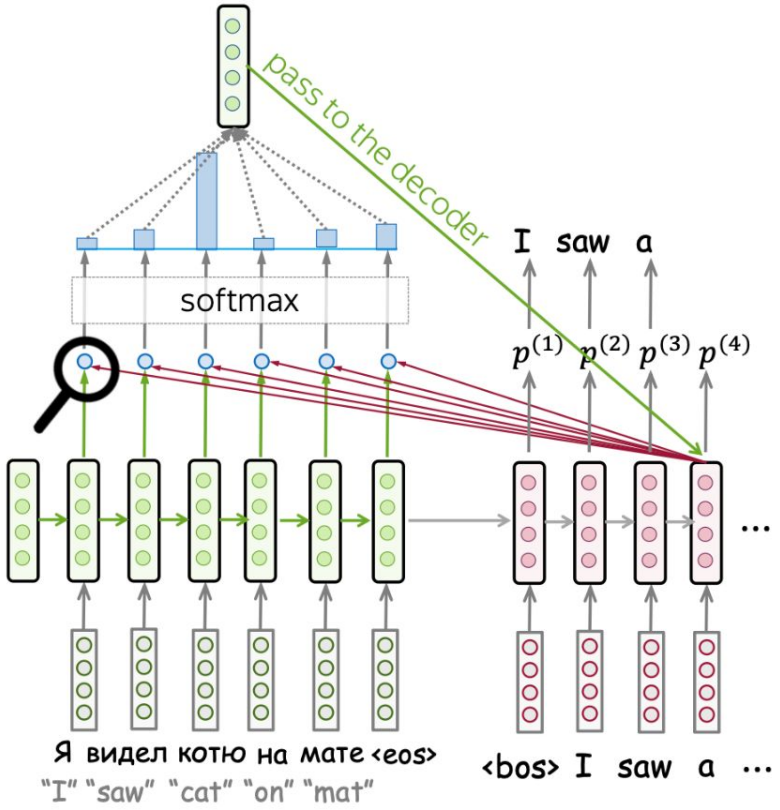
https://lena-voita.github.io/nlp_course.html

Seq2Seq models: Encoder-Decoder Attention



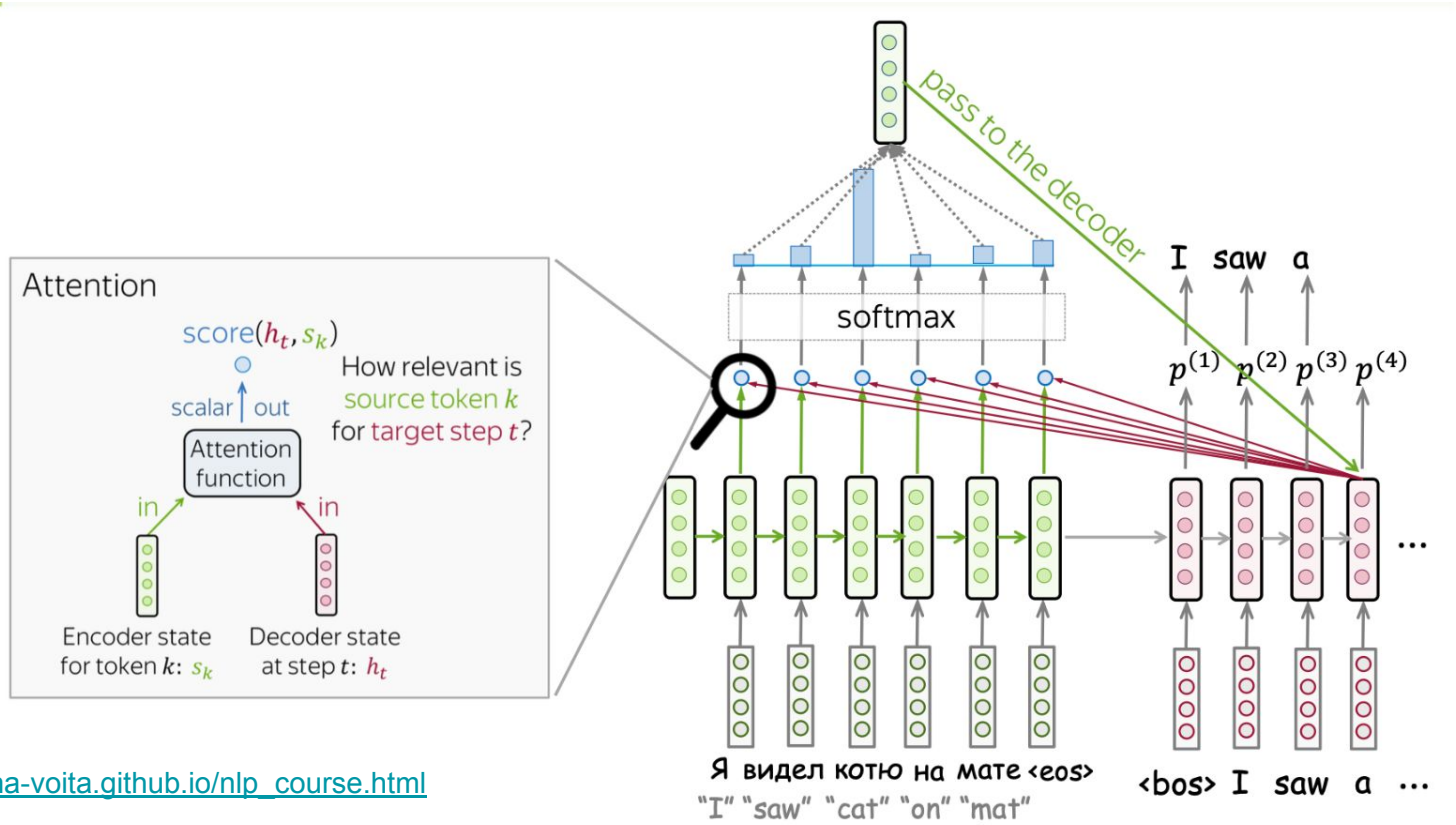
https://lena-voita.github.io/nlp_course.html

Seq2Seq models: Encoder-Decoder Attention



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

Seq2Seq models: Encoder-Decoder Attention



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

Computation Pipeline

Attention input

$s_1, s_2, \ldots, s_m$

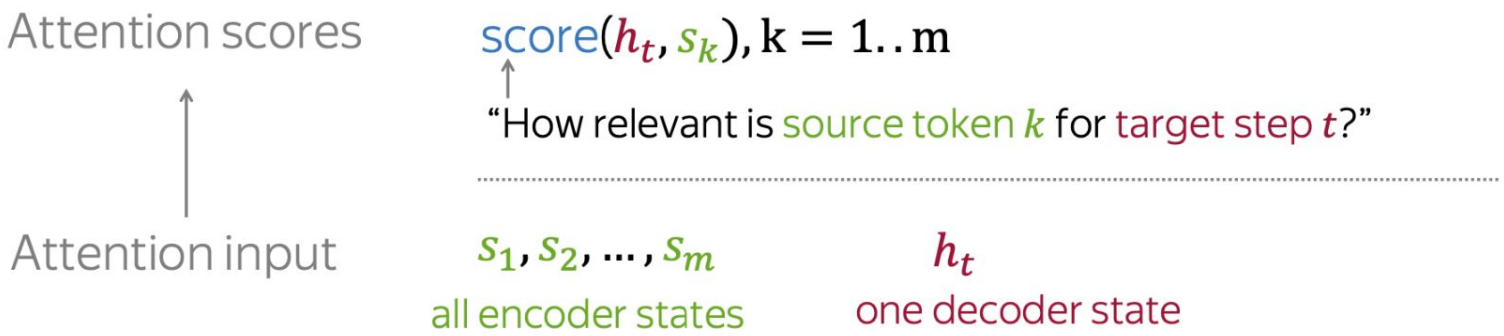
all encoder states

h_t

one decoder state

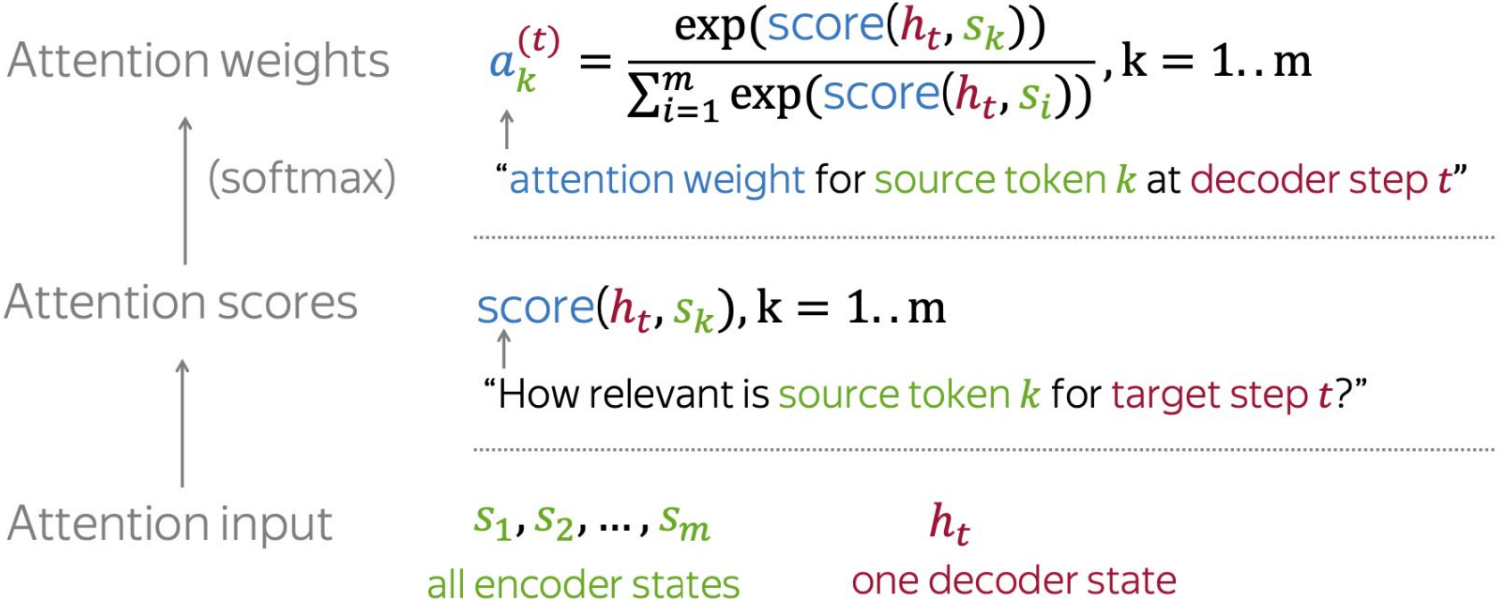
https://lena-voita.github.io/nlp_course.html

Computation Pipeline



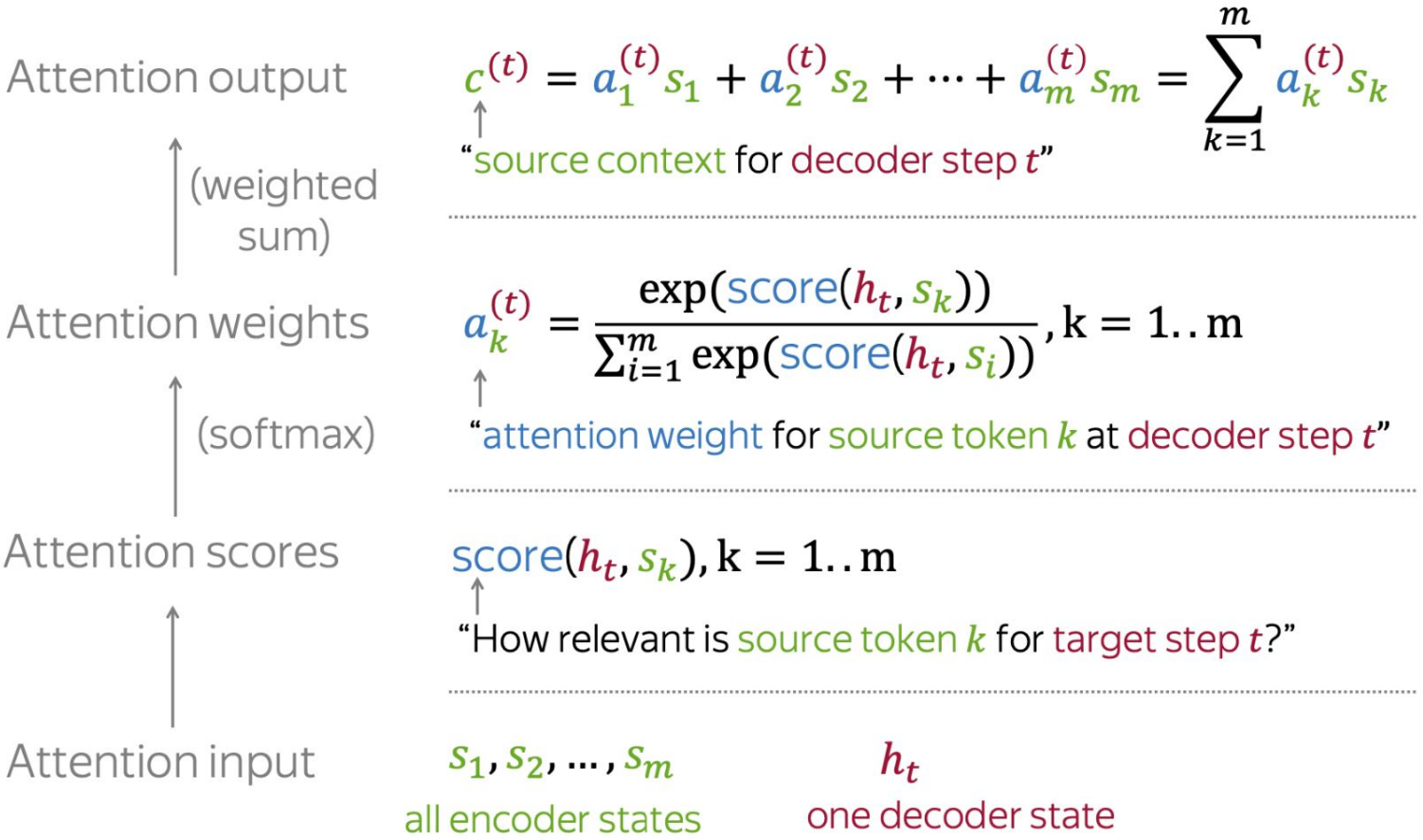
https://lena-voita.github.io/nlp_course.html

Computation Pipeline



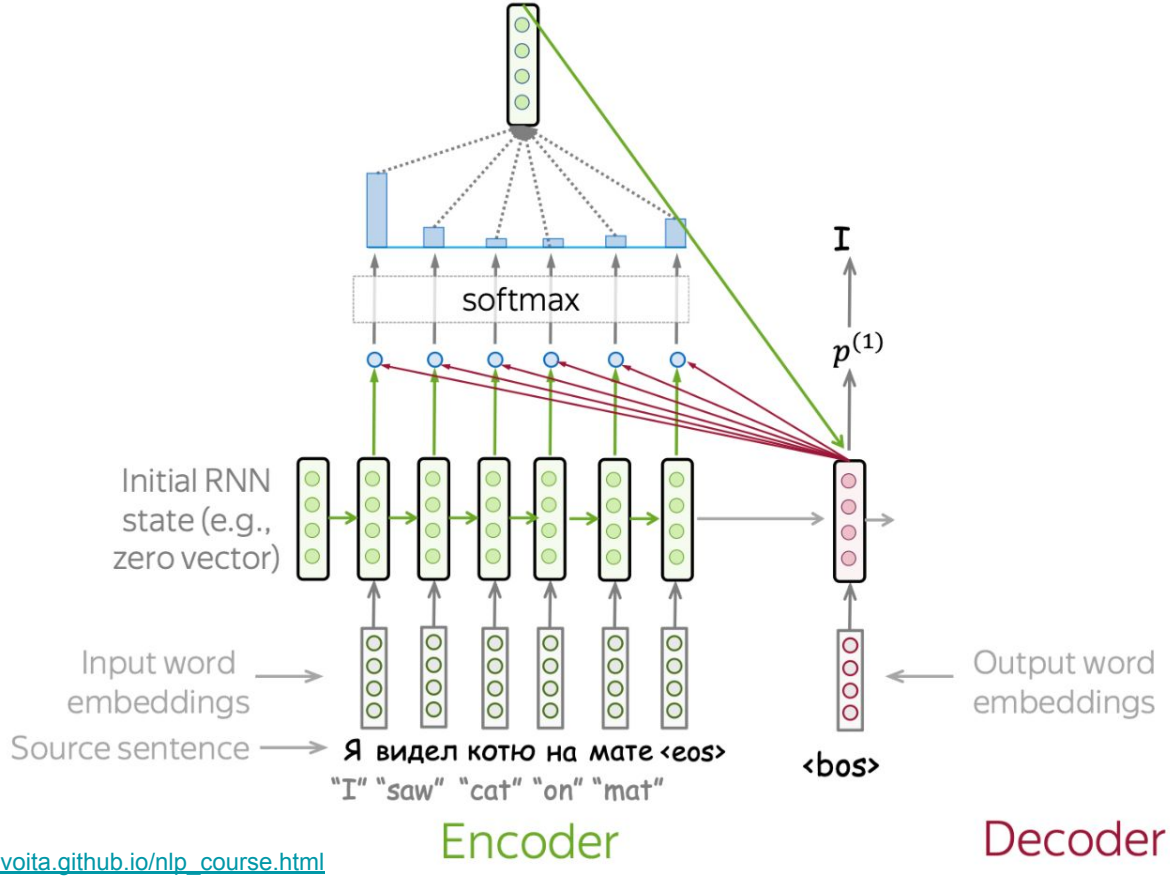
https://lena-voita.github.io/nlp_course.html

Computation Pipeline



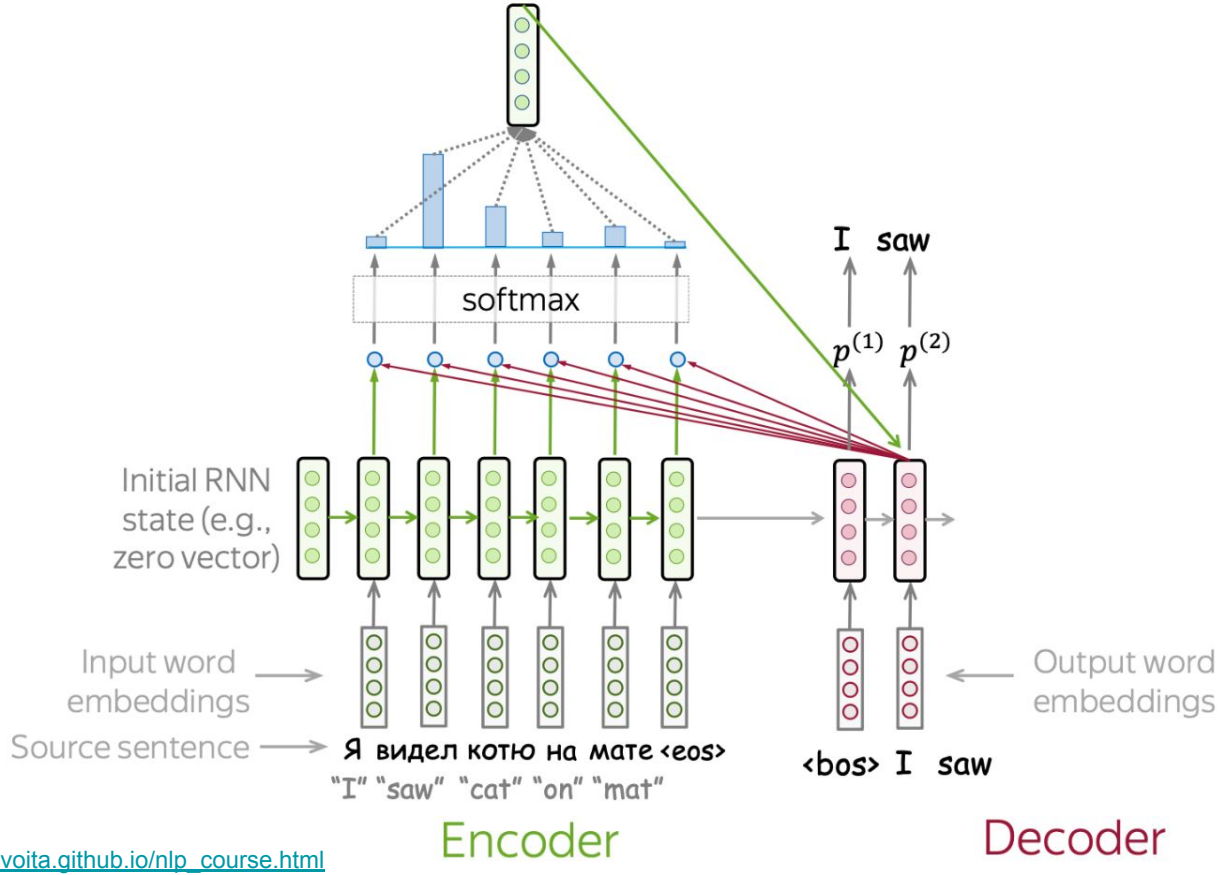
https://lena-voita.github.io/nlp_course.html

Model Learns to Pick Relevant Tokens



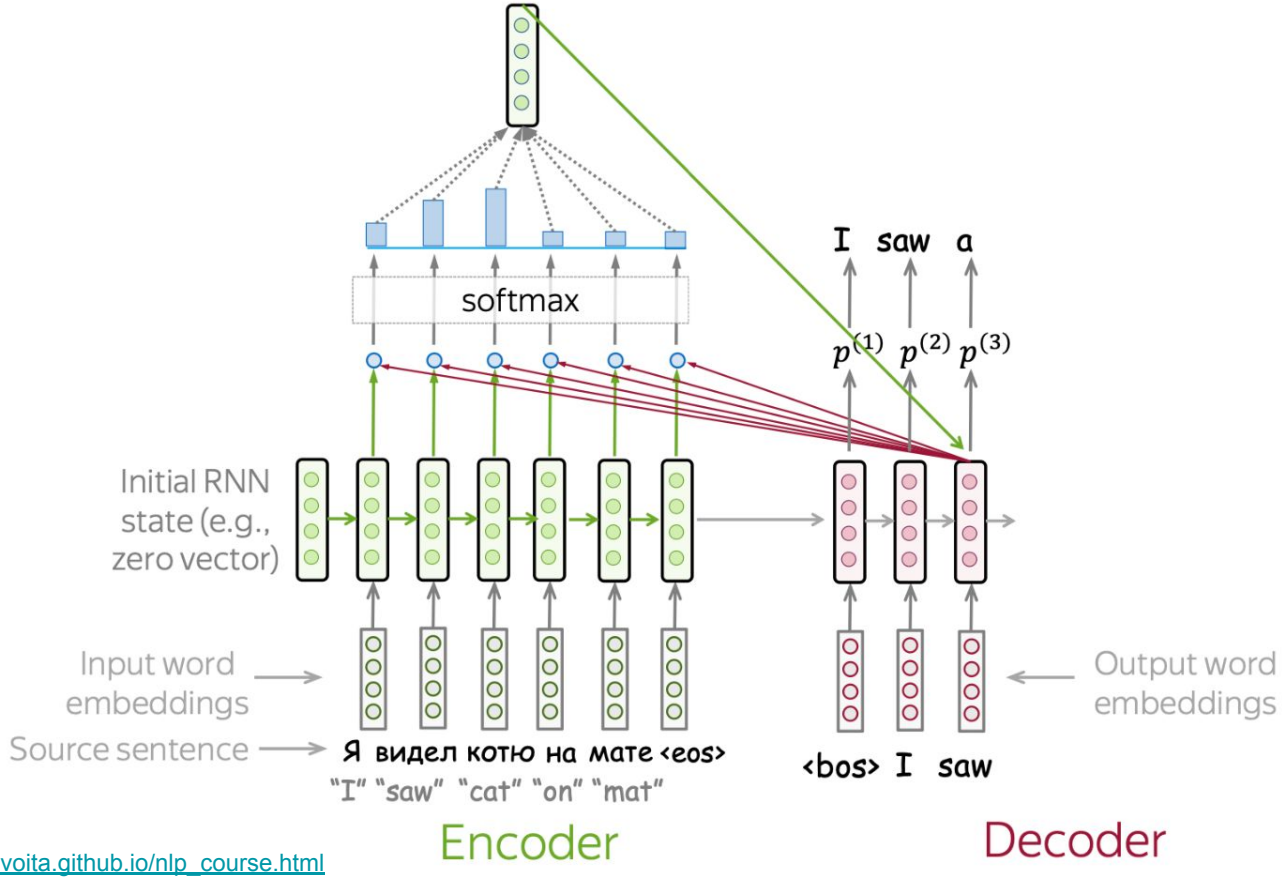
https://lena-voita.github.io/nlp_course.html

Model Learns to Pick Relevant Tokens



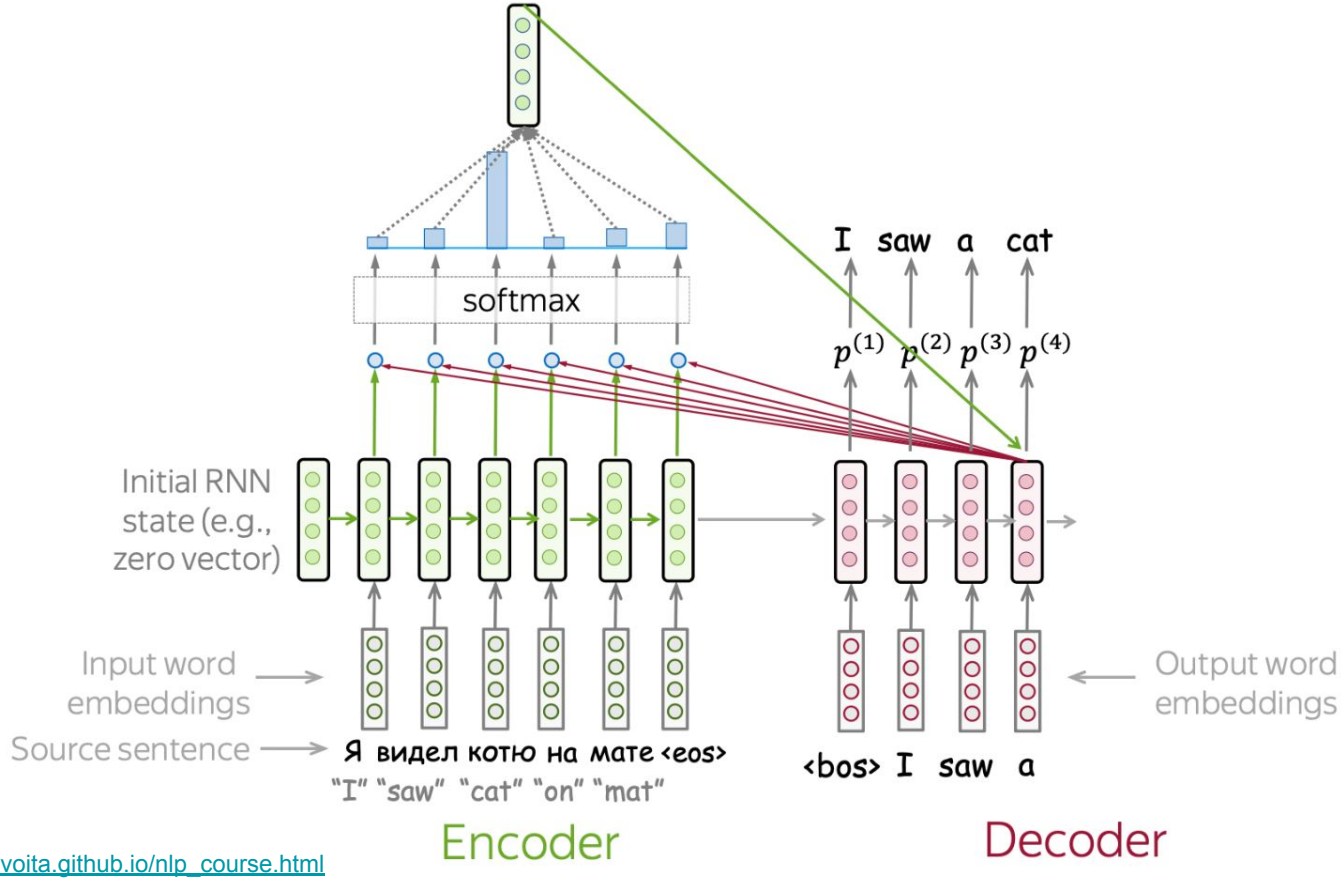
https://lena-voita.github.io/nlp_course.html

Model Learns to Pick Relevant Tokens



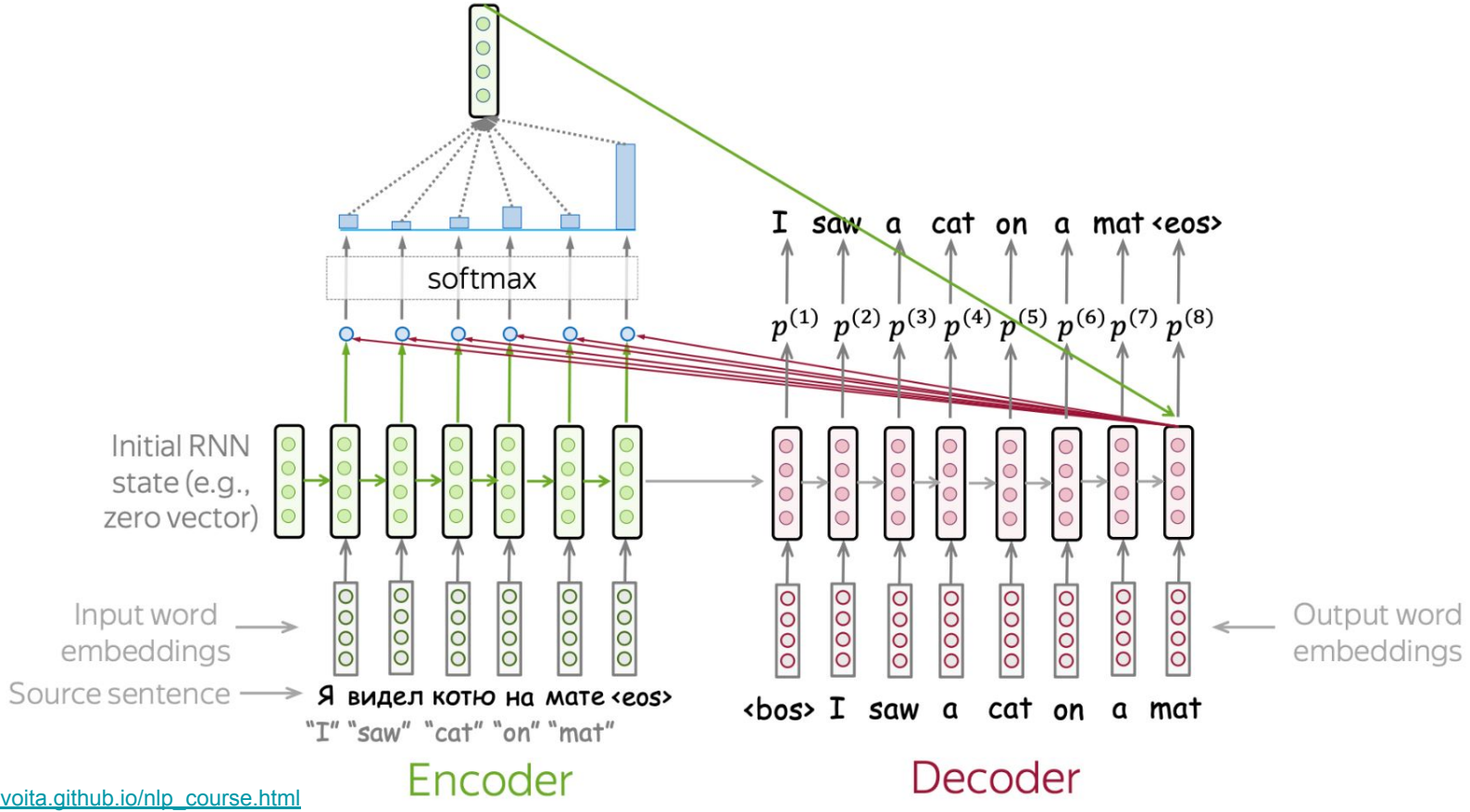
https://lena-voita.github.io/nlp_course.html

Model Learns to Pick Relevant Tokens



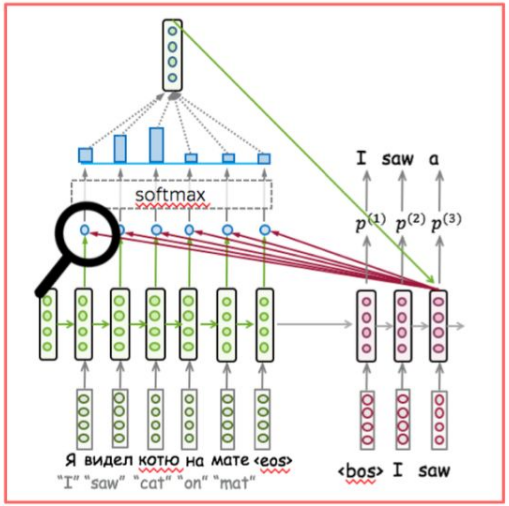
https://lena-voita.github.io/nlp_course.html

Model Learns to Pick Relevant Tokens

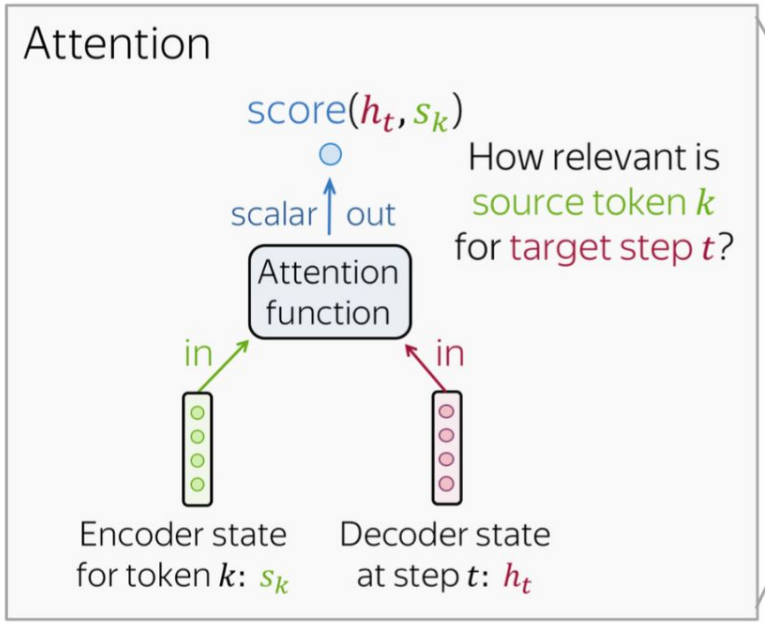


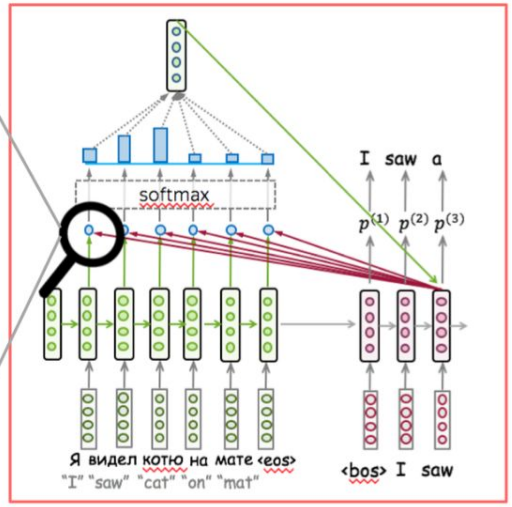
https://lena-voita.github.io/nlp_course.html

Attention Score Functions



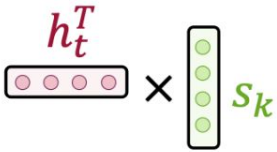
Attention Score Functions



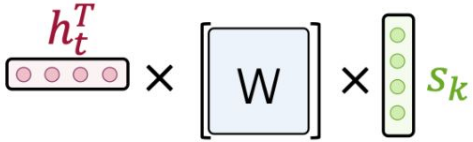


Attention Score Functions

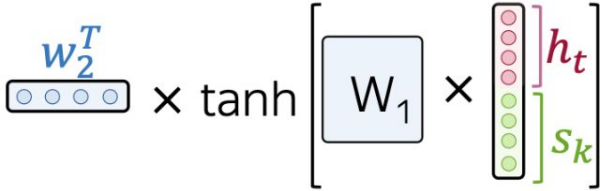
- Dot-product: $\text{score}(h_t, s_k) = h_t^T s_k$

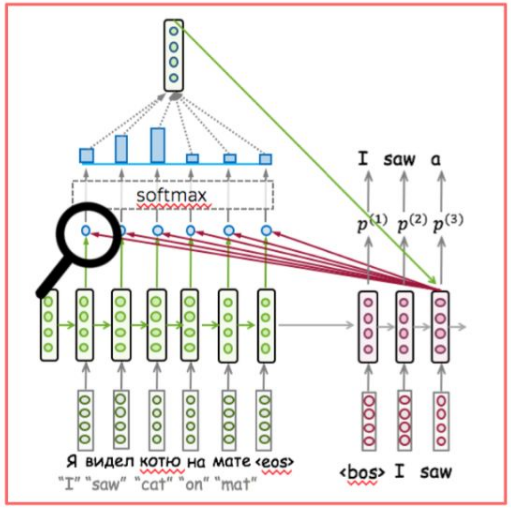


- Bilinear: $\text{score}(h_t, s_k) = h_t^T W s_k$



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





Bahdanau model (sep 2014)

Published as a conference paper at ICLR 2015

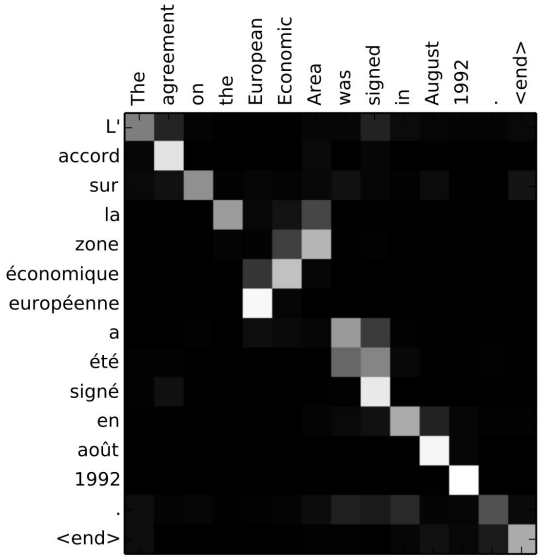
NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.



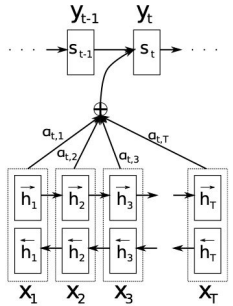


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \tag{6}$$

where

$$e_{ij} = a(s_{i-1}, h_j)$$

The probability α_{ij}, or its associated energy e_{ij}, reflects the importance of the annotation h_j with respect to the previous hidden state s_{i-1} in deciding the next state s_i and generating y_i. Intuitively, this implements a mechanism of attention in the decoder. The decoder decides parts of the source sentence to pay attention to. By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence into a fixed-length vector. With this new approach the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly.

https://arxiv.org/abs/1409.0473

Dzmitry Bahdanau

So I started thinking about how to avoid the bottleneck between encoder and decoder RNN. My first idea was to have a model with two “cursors”, one moving through the source sequence (encoded by a BiRNN) and another one moving through the target sequence. The cursor trajectories would be marginalized out using dynamic programming. KyungHyun Cho recognized this as an equivalent to Alex Graves’ RNN Transducer model. Following that, I may have also read Graves’ hand-writing recognition paper. The approach looked inappropriate for machine translation though.

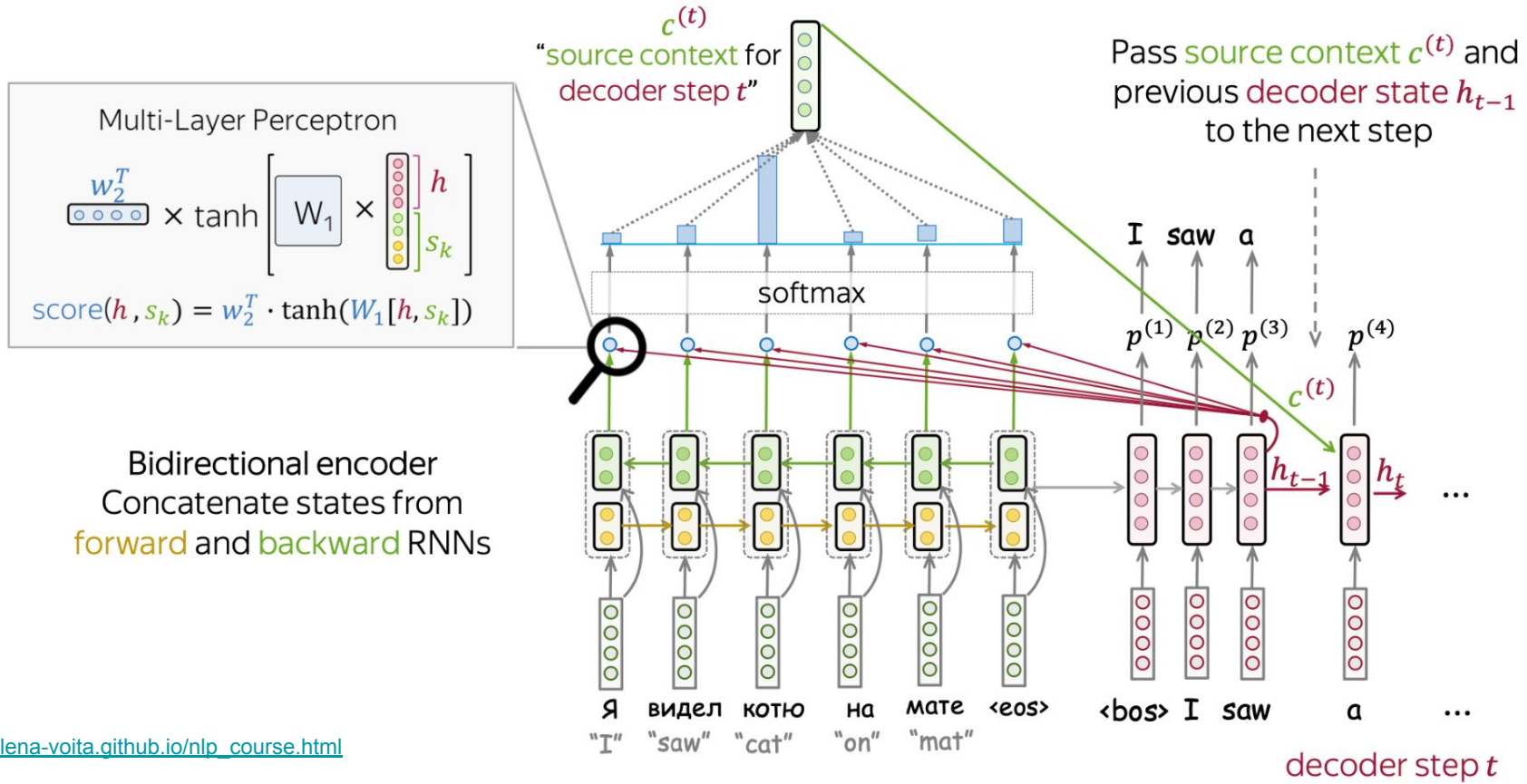
The above approach with cursors would be too hard to implement in the remaining 5 weeks of my internship. So I tried instead something simpler - two cursors moving at the same time synchronously (effectively hard-coded diagonal attention). That sort of worked, but the approach lacked elegance.

So one day I had this thought that it would be nice to enable the decoder RNN to learn to search where to put the cursor in the source sequence. This was sort of inspired by translation exercises that learning English in my middle school involved. Your gaze shifts back and forth between source and target sequence as you translate. I expressed the soft search as softmax and then weighted averaging of BiRNN states. It worked great from the very first try to my great excitement. I called the architecture RNNSearch, and we rushed to publish an ArXiV paper as we knew that Ilya and co at Google are somewhat ahead of us with their giant 8 GPU LSTM model (RNN Search still ran on 1 GPU).

As it later turned out, the name was not great. The better name (attention) was only added by Yoshua to the conclusion in one of the final passes.

https://www.youtube.com/watch?v=XfpMkf4rD6E

Bahdanau model (sep 2014)



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

Attention is all you need (jun 2017)

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 that include an encoder and a decoder. 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 ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

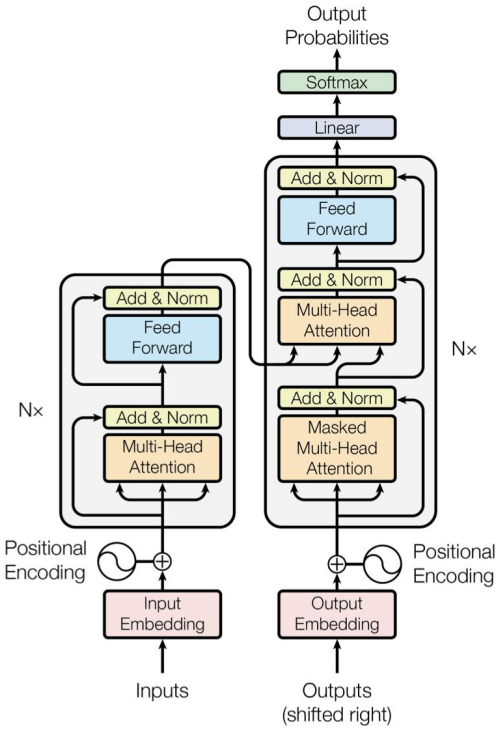


Figure 1: The Transformer - model architecture.

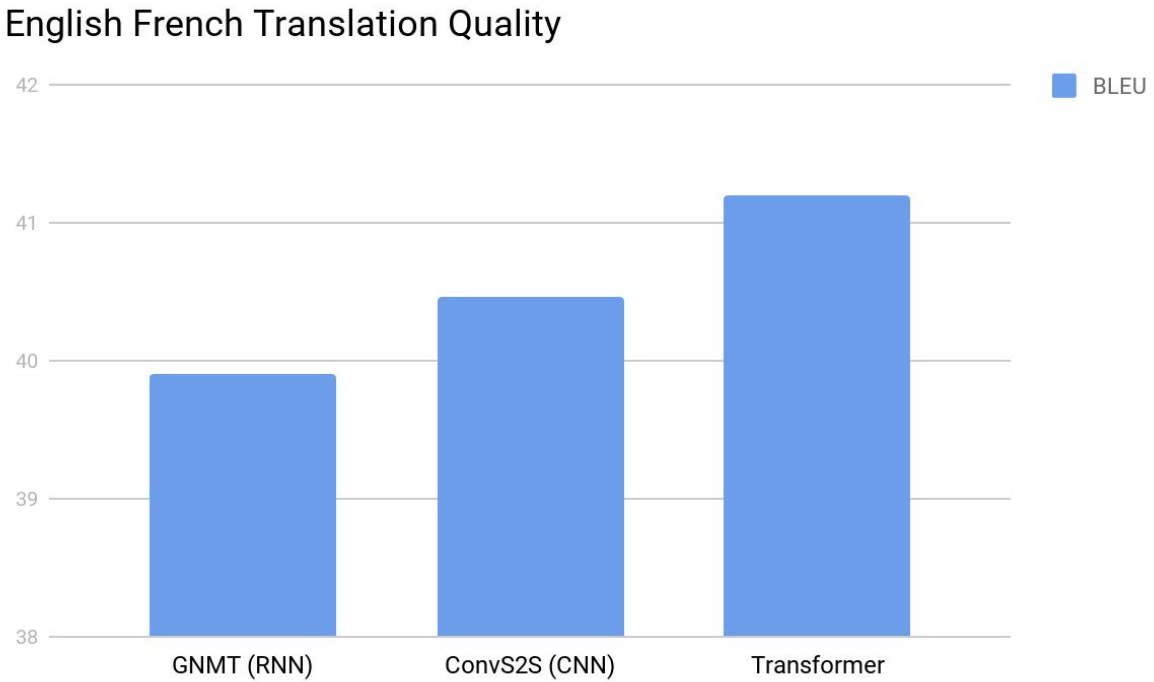
https://arxiv.org/abs/1706.03762

Transformers basics

	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

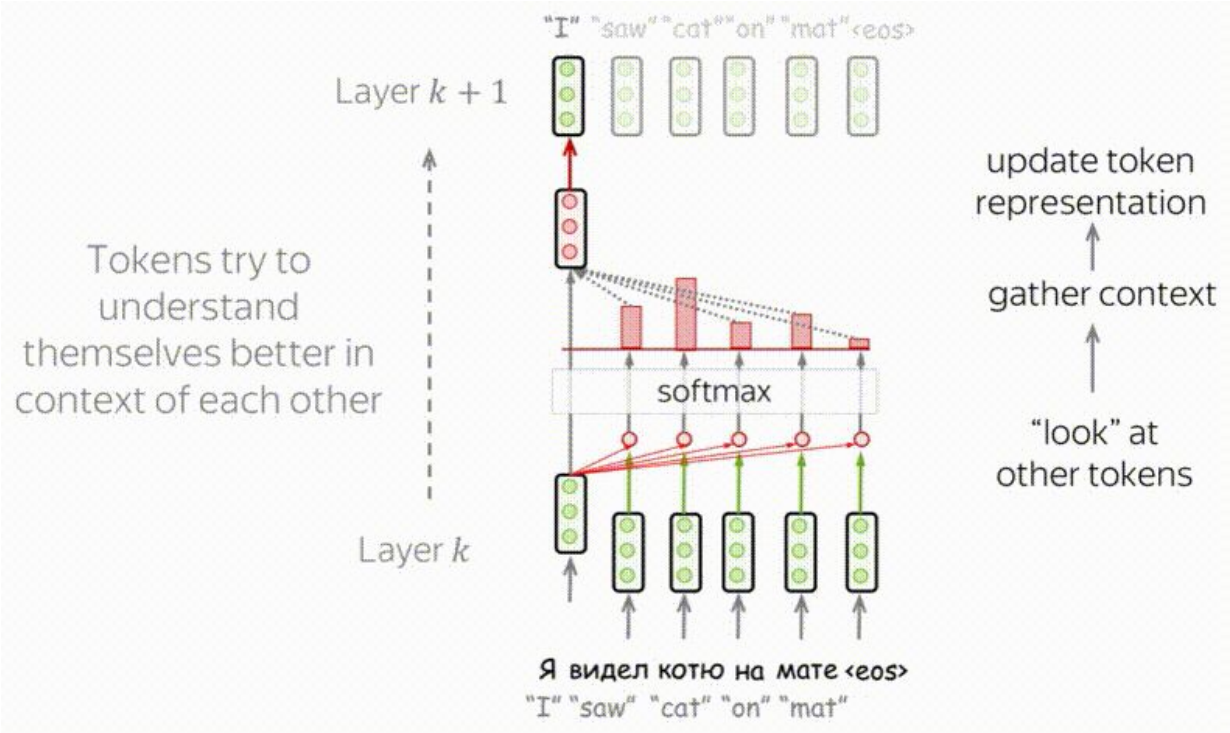
https://lena-voita.github.io/nlp_course.html

Attention is all you need (jun 2017)



https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding/

Self-Attention



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

Self-Attention

Each vector receives three representations ("roles")

$W_Q \times$ = Query: vector from which the attention is looking

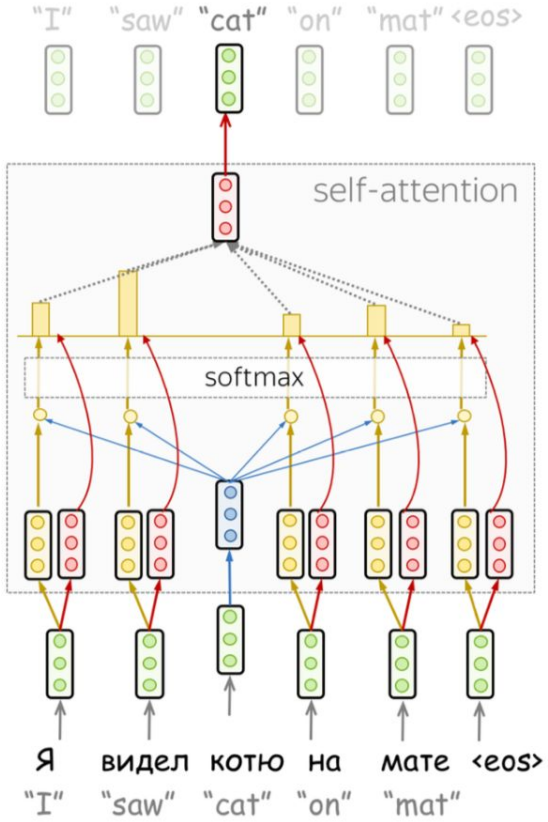
"Hey there, do you have this information?"

$W_K \times$ = Key: vector at which the query looks to compute weights

"Hi, I have this information – give me a large weight!"

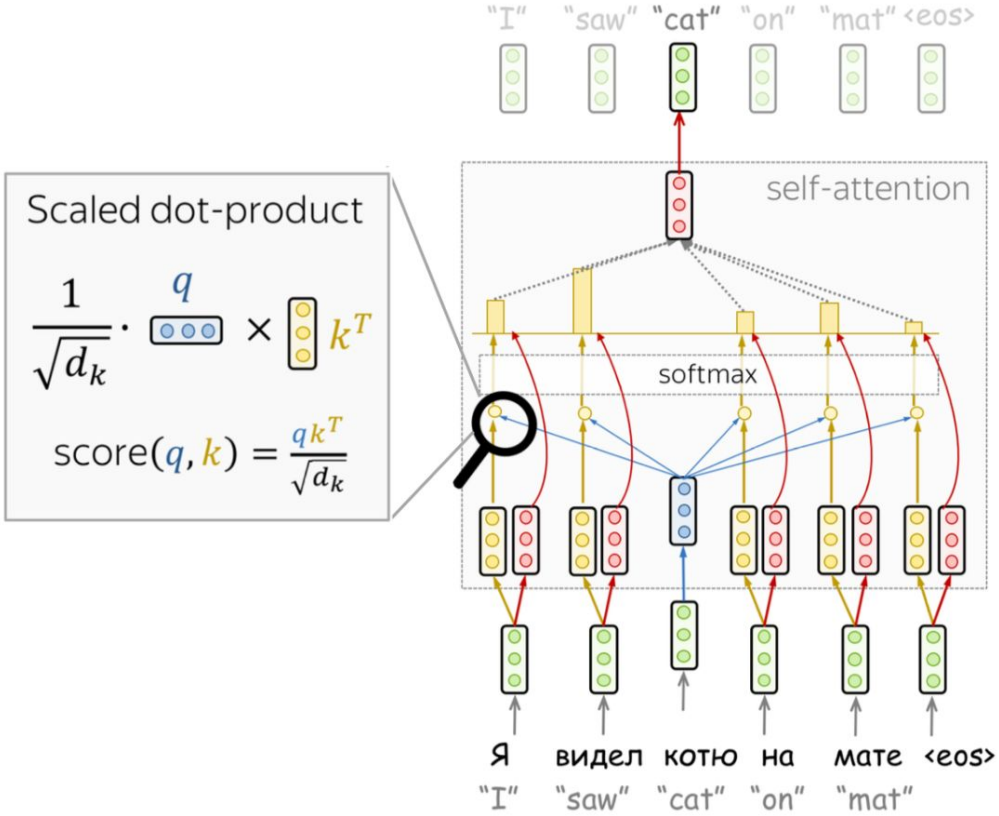
$W_V \times$ = Value: their weighted sum is attention output

"Here's the information I have!"



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

Attention score

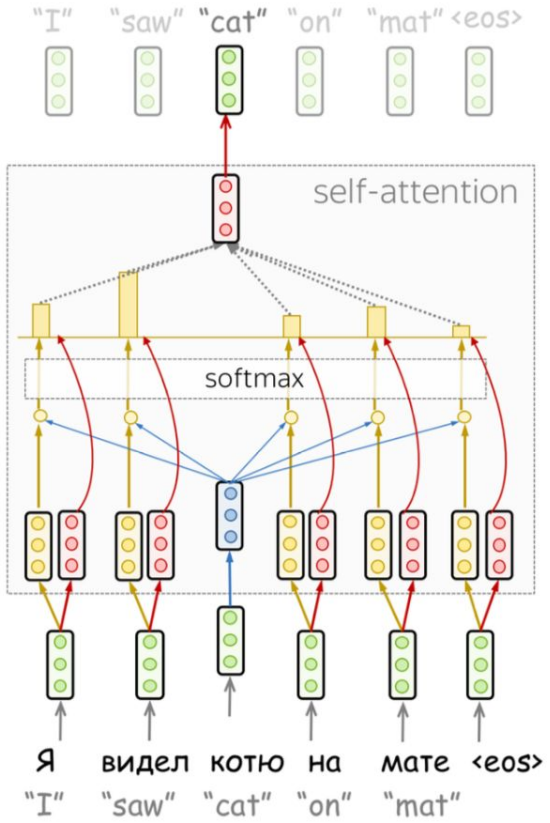


https://lena-voita.github.io/nlp_course.html

Self-Attention

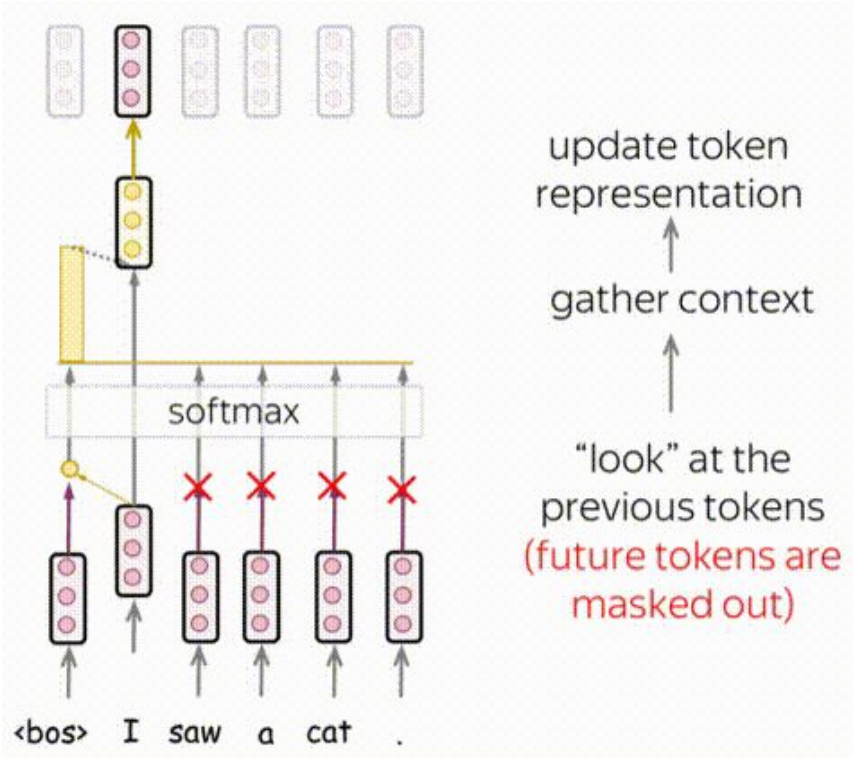
$$Attention(q, k, v) = \overbrace{softmax\left(\frac{qk^T}{\sqrt{d_k}}\right)v}^{\text{Attention weights}}$$

from (q) · to (k, v) · vector dimensionality of K, V



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

Masked Self-Attention

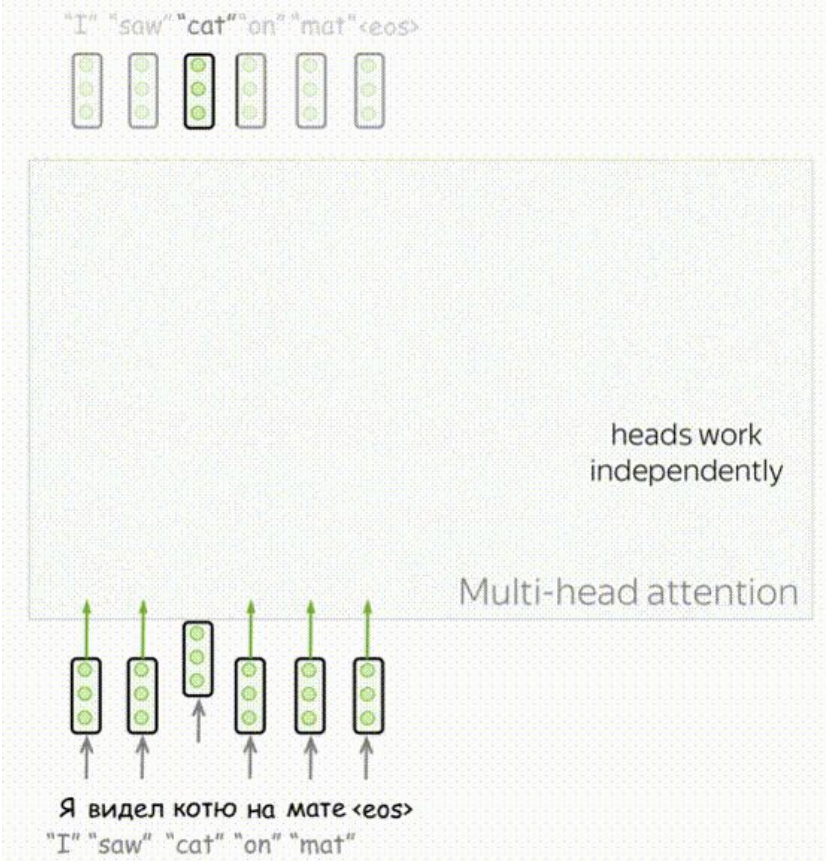


https://lena-voita.github.io/nlp_course.html

Multihead Self-Attention

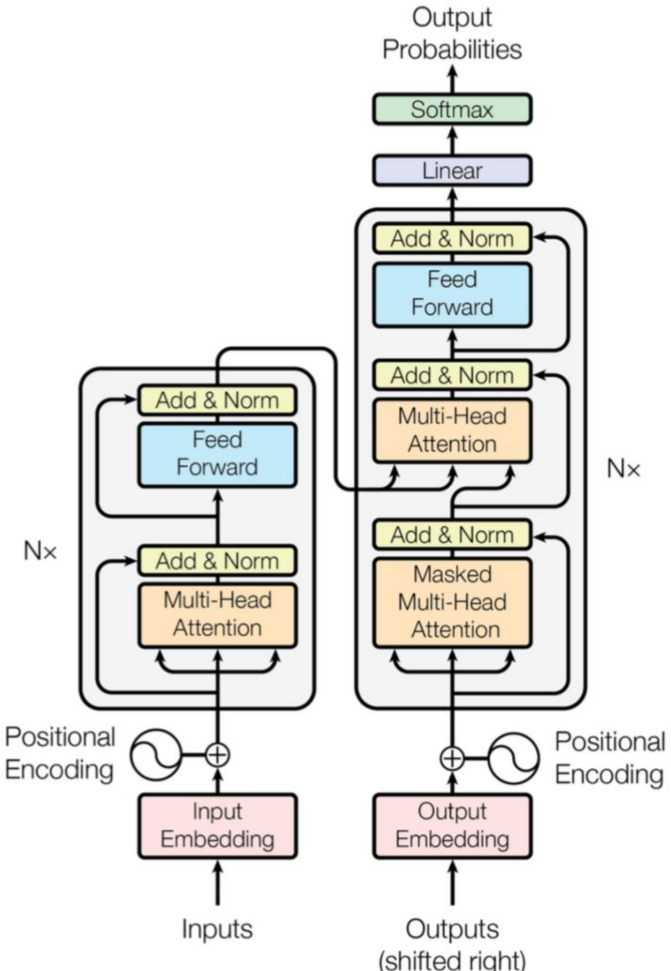
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_n)W_o,$$

$$\text{head}_i = \text{Attention}(QW_Q^i, KW_K^i, VW_V^i)$$



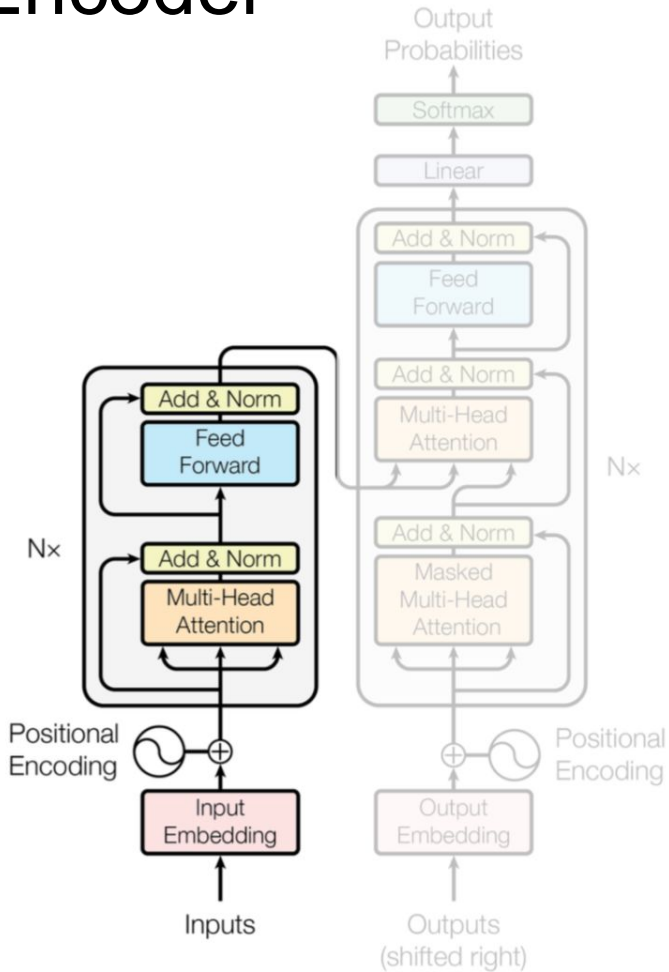
https://lena-voita.github.io/nlp_course.html

Transformer



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

Transformer Encoder

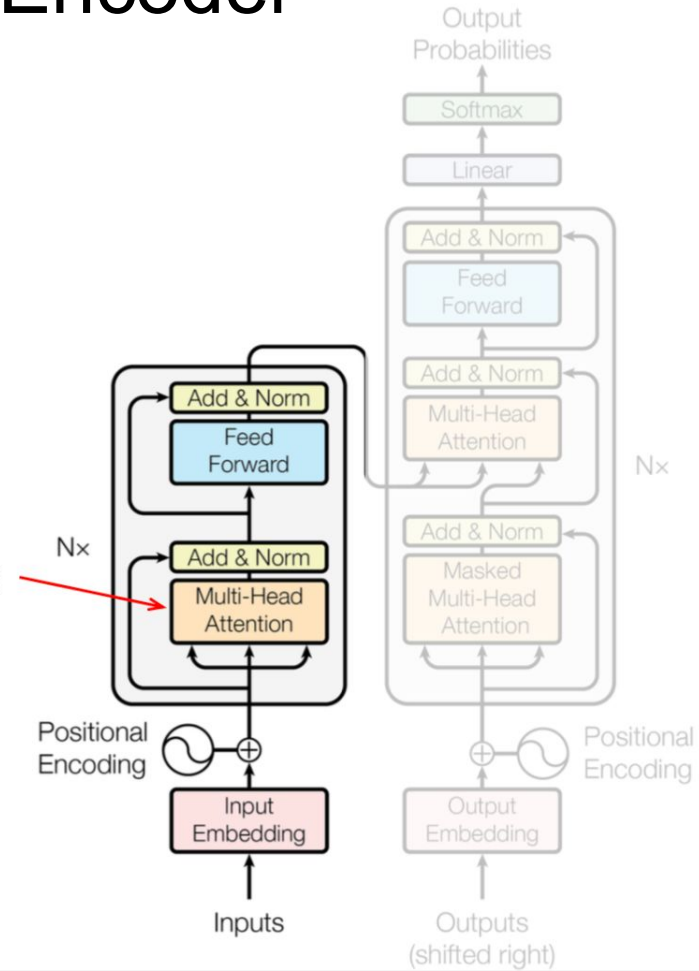


https://lena-voita.github.io/nlp_course.html

Transformer Encoder

Encoder self-attention:
tokens look at each other

queries, keys, values are computed from encoder states



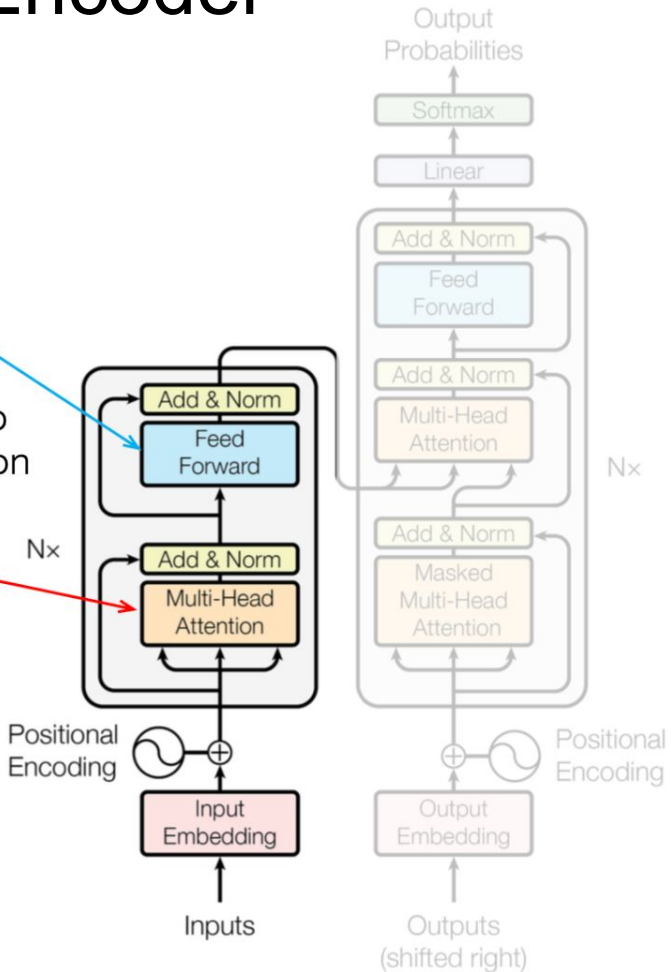
https://lena-voita.github.io/nlp_course.html

Transformer Encoder

Feed-forward network:
after taking information from other tokens, take a moment to think and process this information

Encoder self-attention:
tokens look at each other

queries, keys, values are computed from encoder states



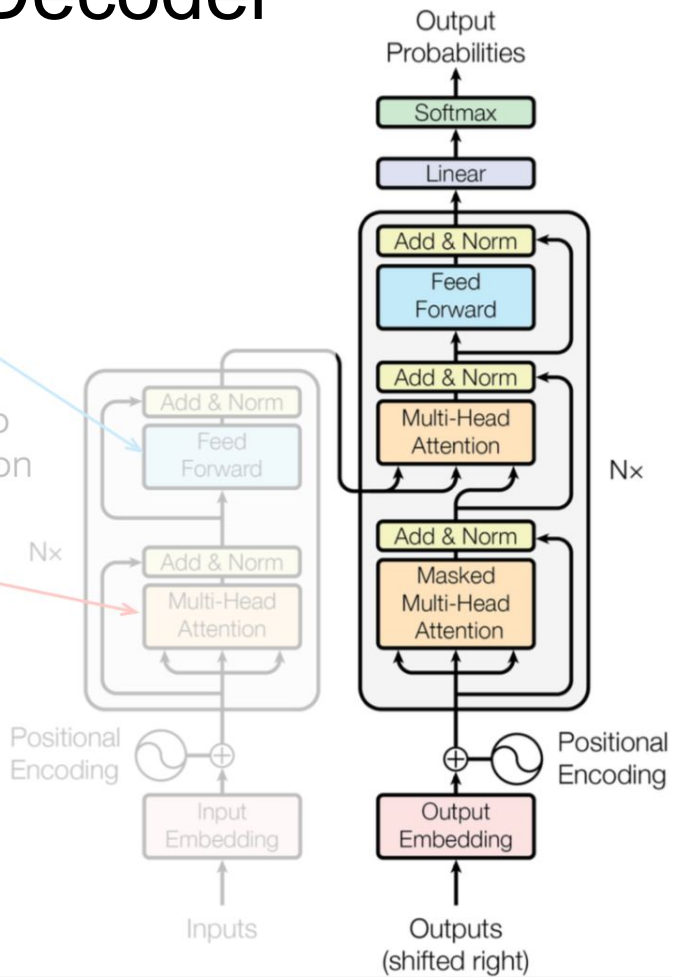
https://lena-voita.github.io/nlp_course.html

Transformer Decoder

Feed-forward network:
after taking information from other tokens, take a moment to think and process this information

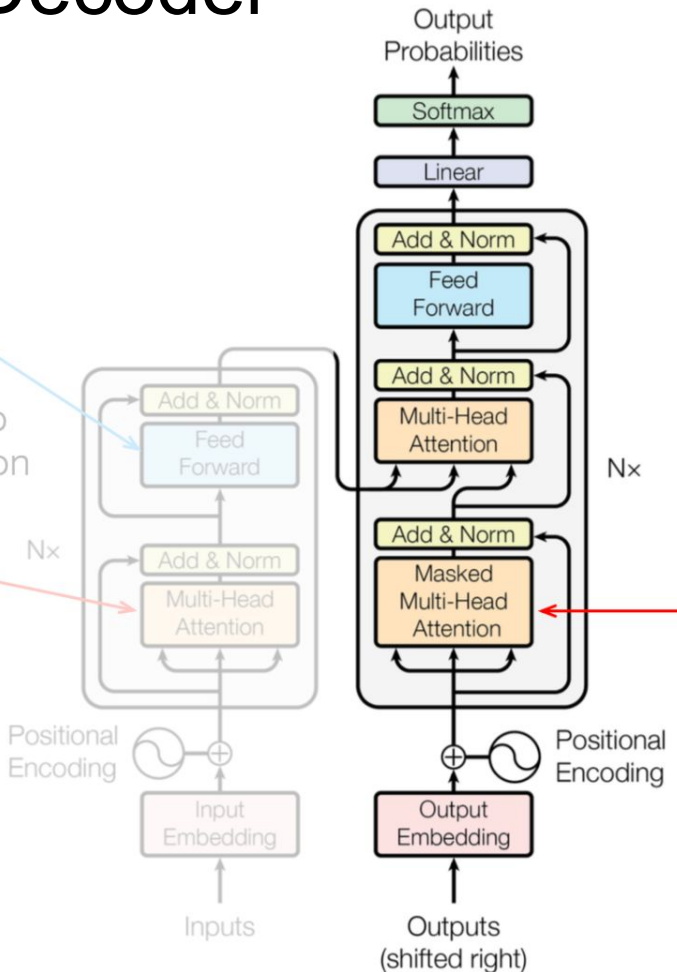
Encoder self-attention:
tokens look at each other

queries, keys, values are computed from encoder states



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

Transformer Decoder



Feed-forward network:
after taking information from other tokens, take a moment to think and process this information

Encoder self-attention:
tokens look at each other

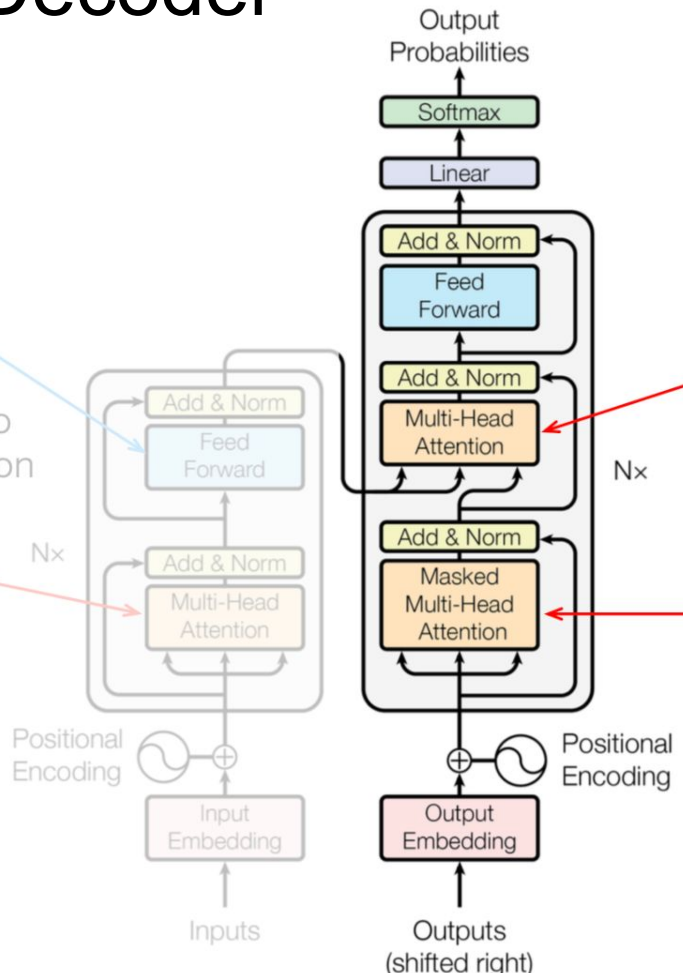
queries, keys, values are computed from encoder states

Decoder self-attention (masked):
tokens look at the previous tokens

queries, keys, values are computed from decoder states

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

Transformer Decoder



Feed-forward network:
after taking information from other tokens, take a moment to think and process this information

Encoder self-attention:
tokens look at each other

queries, keys, values are computed from encoder states

Decoder-encoder attention:
target token looks at the source

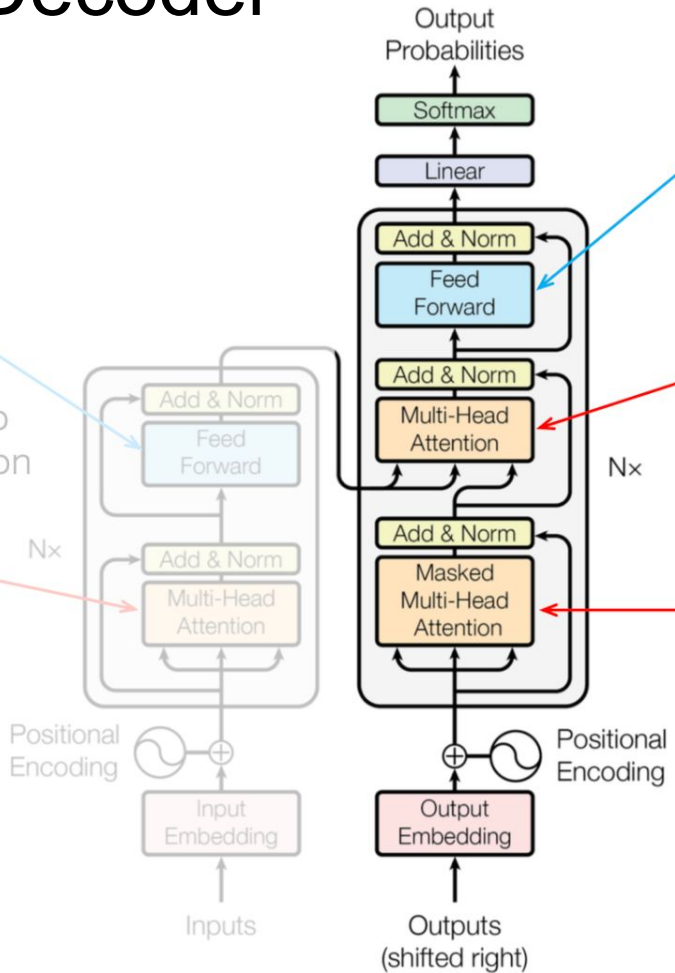
queries – from decoder states; keys and values from encoder states

Decoder self-attention (masked):
tokens look at the previous tokens

queries, keys, values are computed from decoder states

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

Transformer Decoder



Feed-forward network:
after taking information from other tokens, take a moment to think and process this information

Decoder-encoder attention:
target token looks at the source
queries – from decoder states; keys and values from encoder states

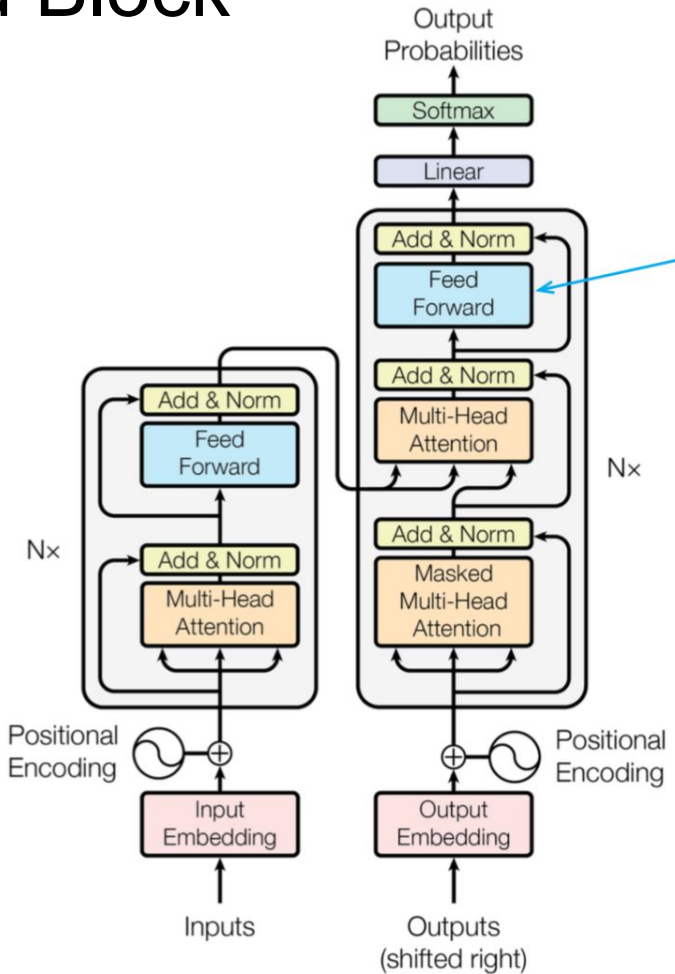
Decoder self-attention (masked):
tokens look at the previous tokens
queries, keys, values are computed from decoder states

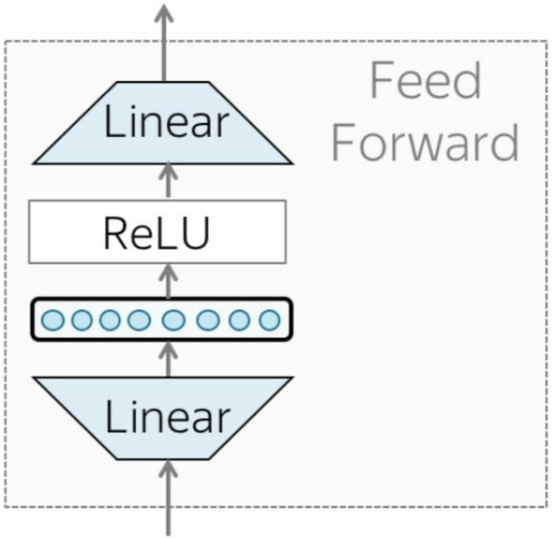
Feed-forward network:
after taking information from other tokens, take a moment to think and process this information

Encoder self-attention:
tokens look at each other
queries, keys, values are computed from encoder states

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

Feed Forward Block

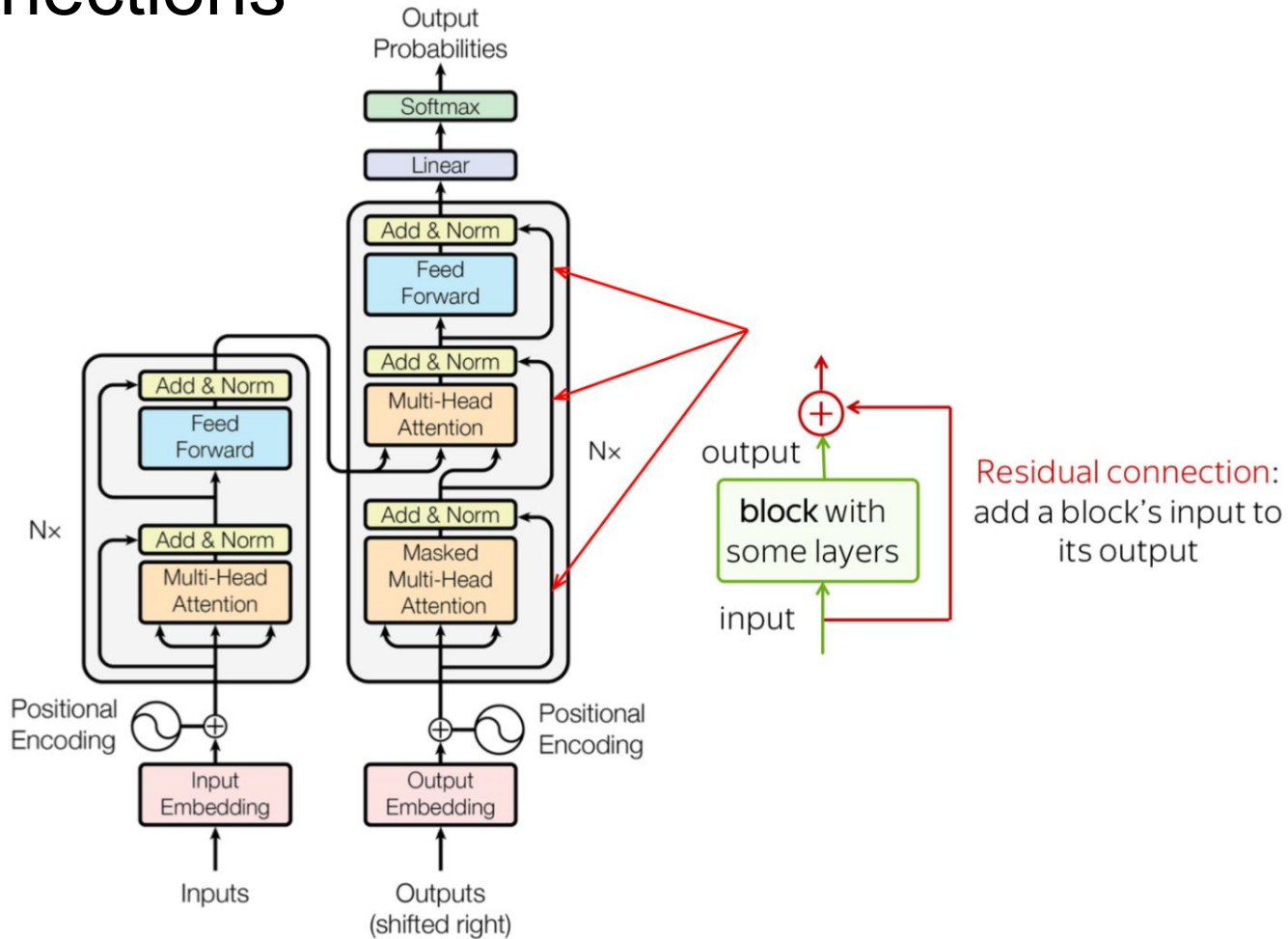




$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

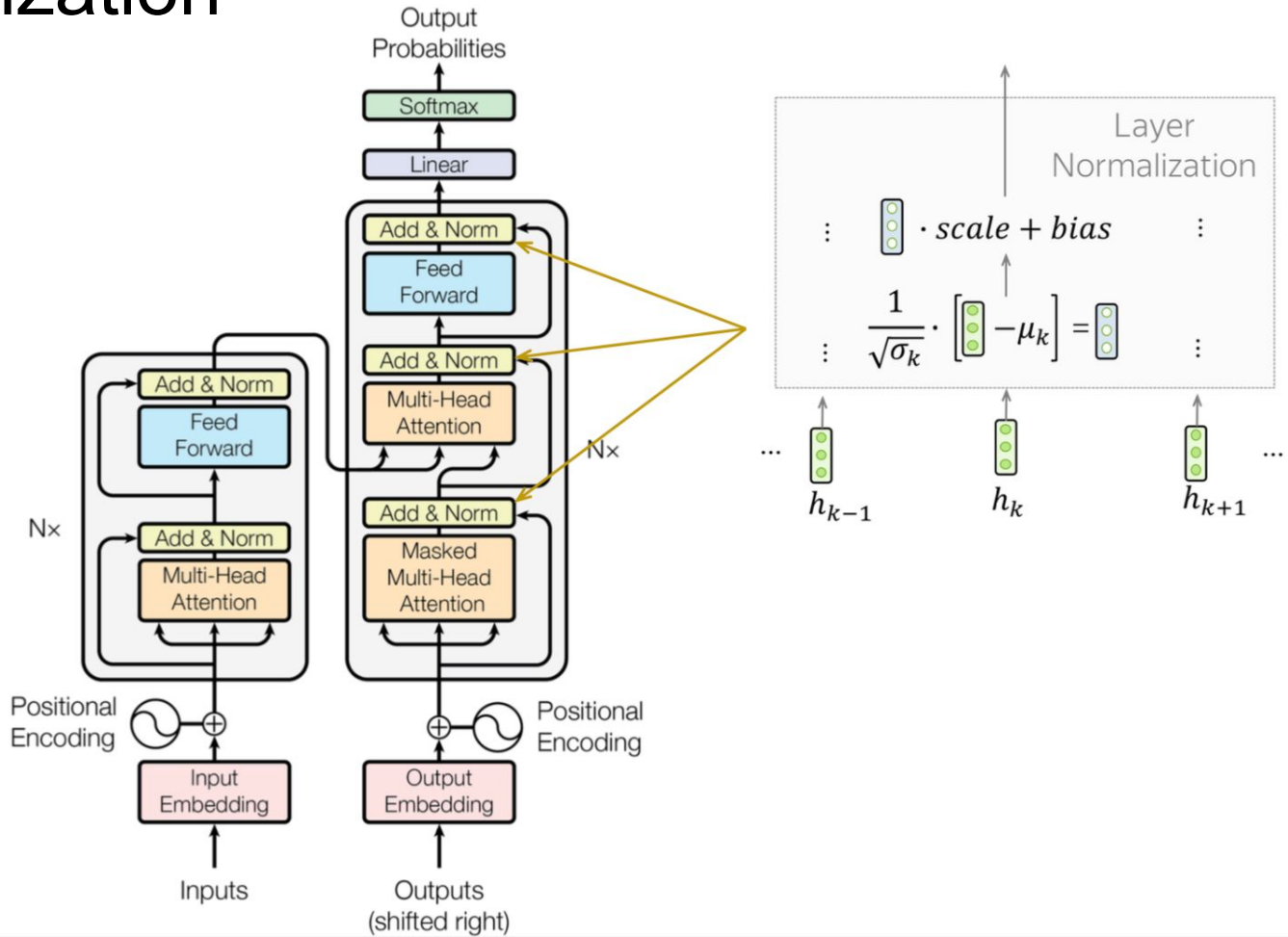
https://lena-voita.github.io/nlp_course.html

Residual Connections



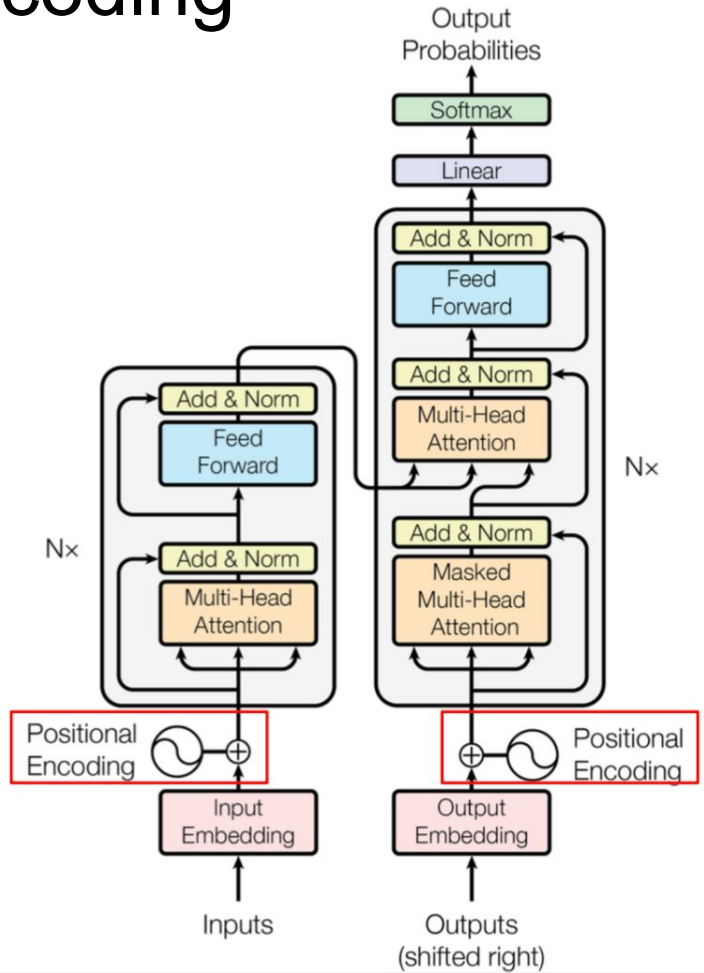
https://lena-voita.github.io/nlp_course.html

Layer Normalization



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

Positional Encoding



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

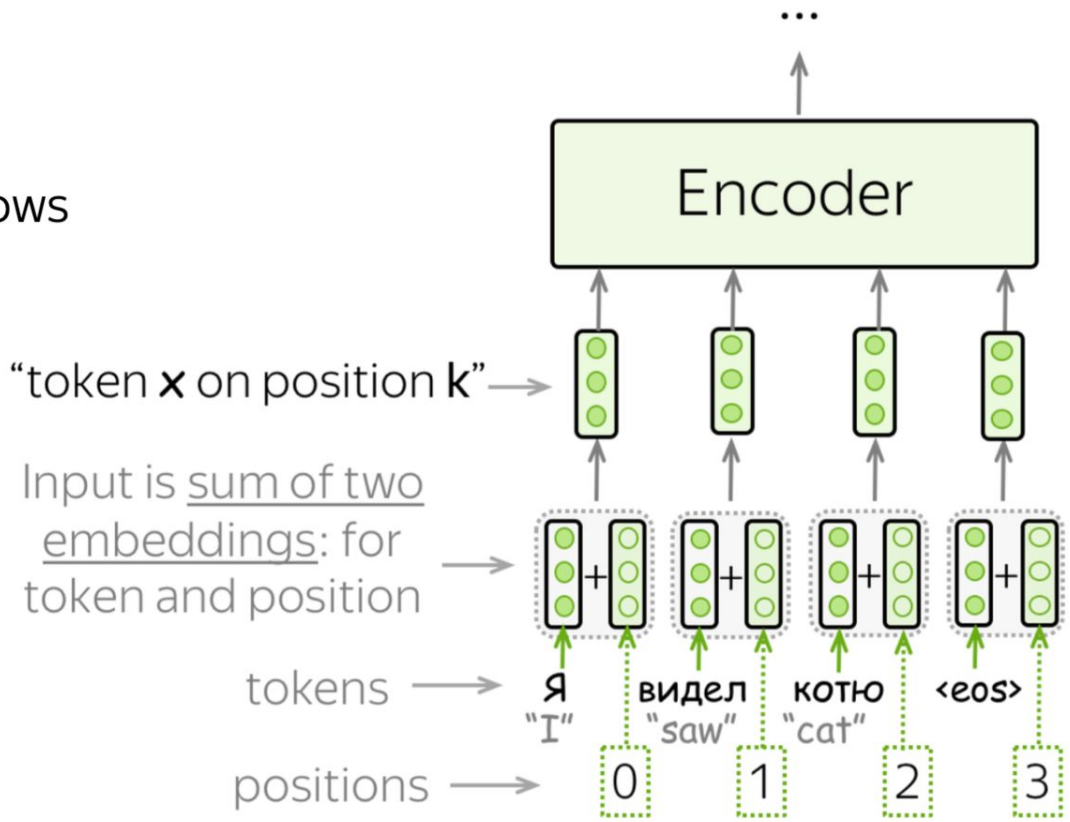
Positional Encoding

Problem:

- without recurrence or convolution, the model knows nothing about position

Solution:

- encode position explicitly and add



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

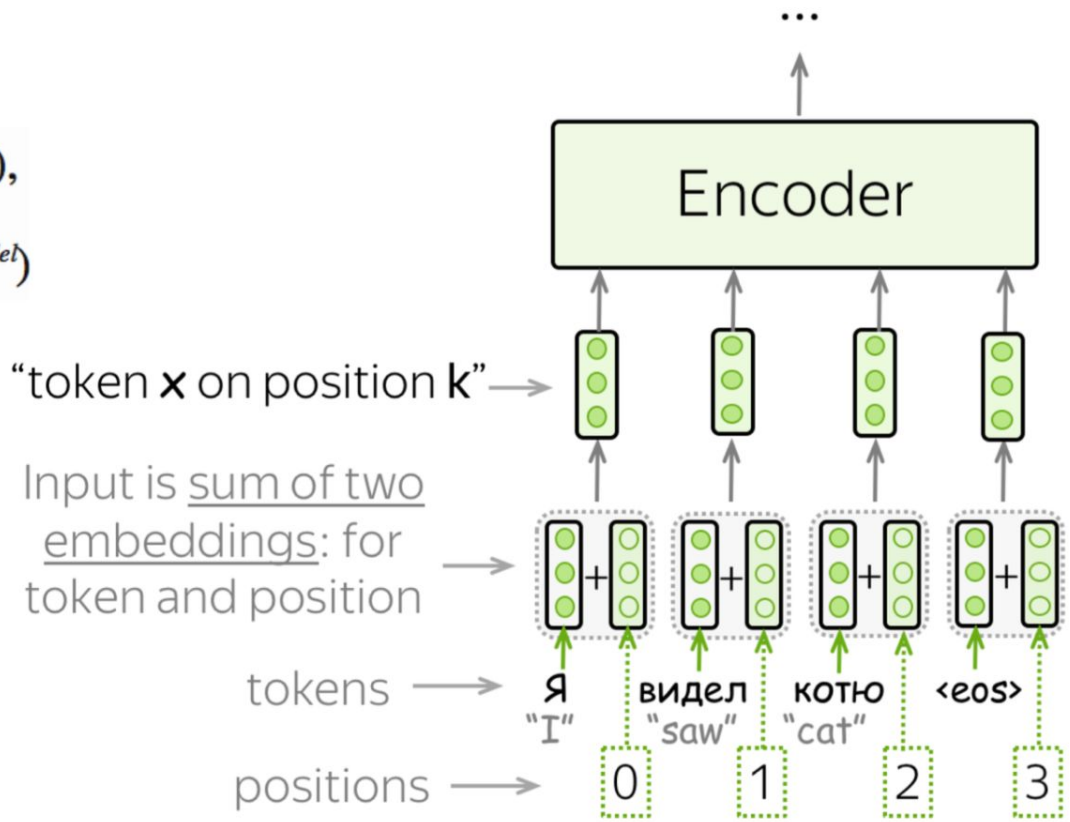
Positional Encoding

Fixed encodings:

$$PE_{pos,2i} = \sin(pos/10000^{2i/d_{model}}),$$

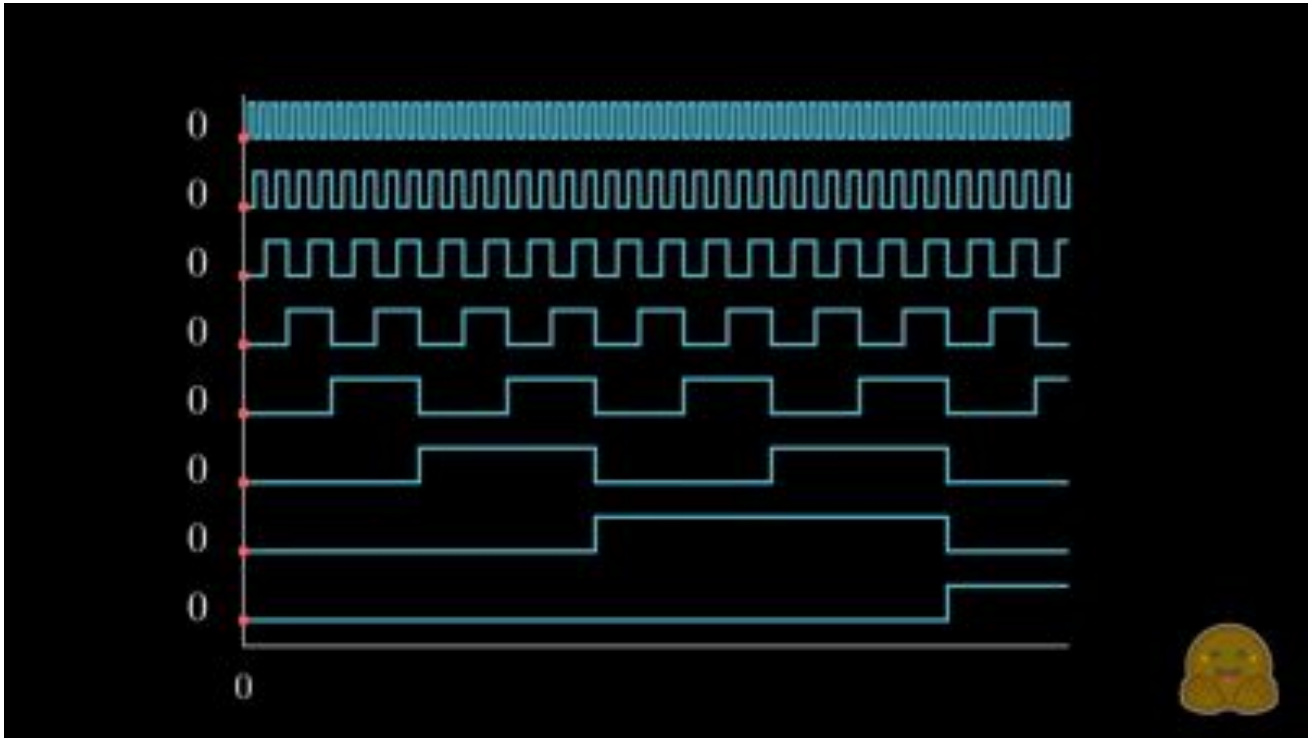
$$PE_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}})$$

pos – position, i - dimension



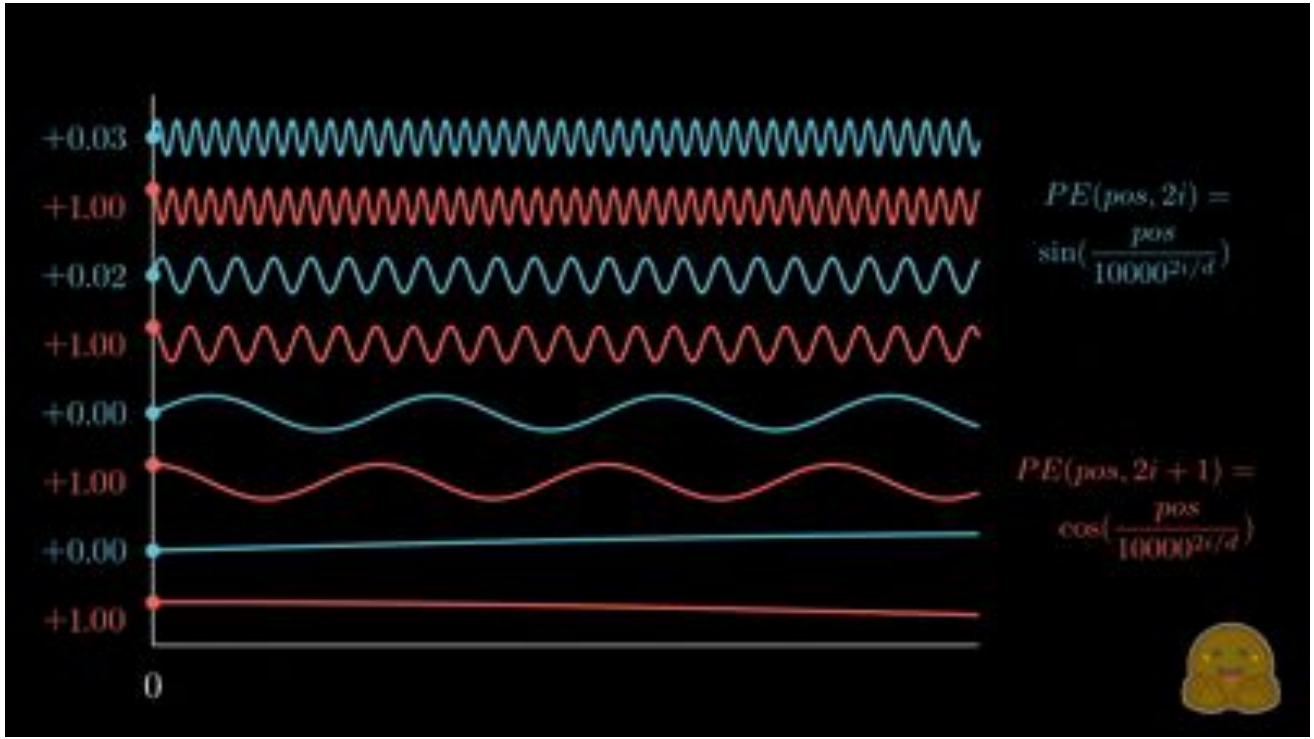
https://lena-voita.github.io/nlp_course.html

Binary Positional Encoding



https://huggingface.co/blog/designing-positional-encoding

Sinusoidal Positional Encoding



https://huggingface.co/blog/designing-positional-encoding