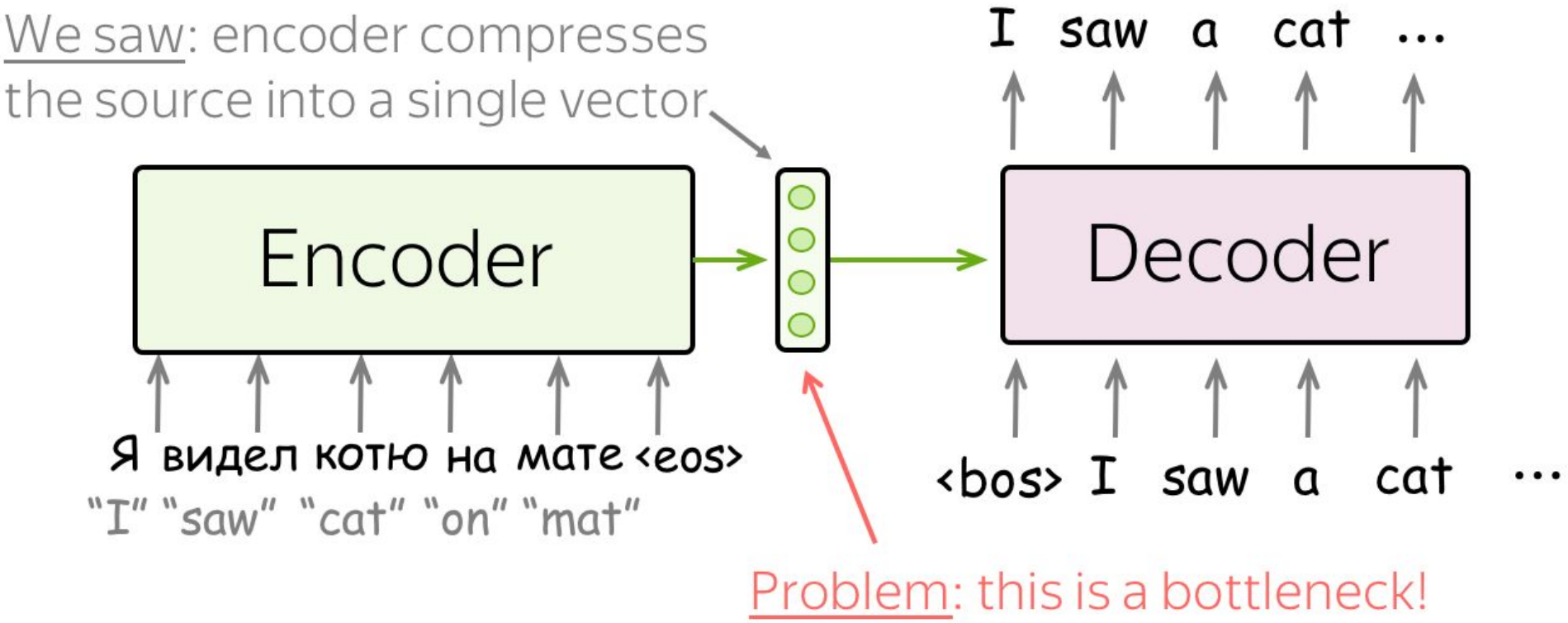
Transformer

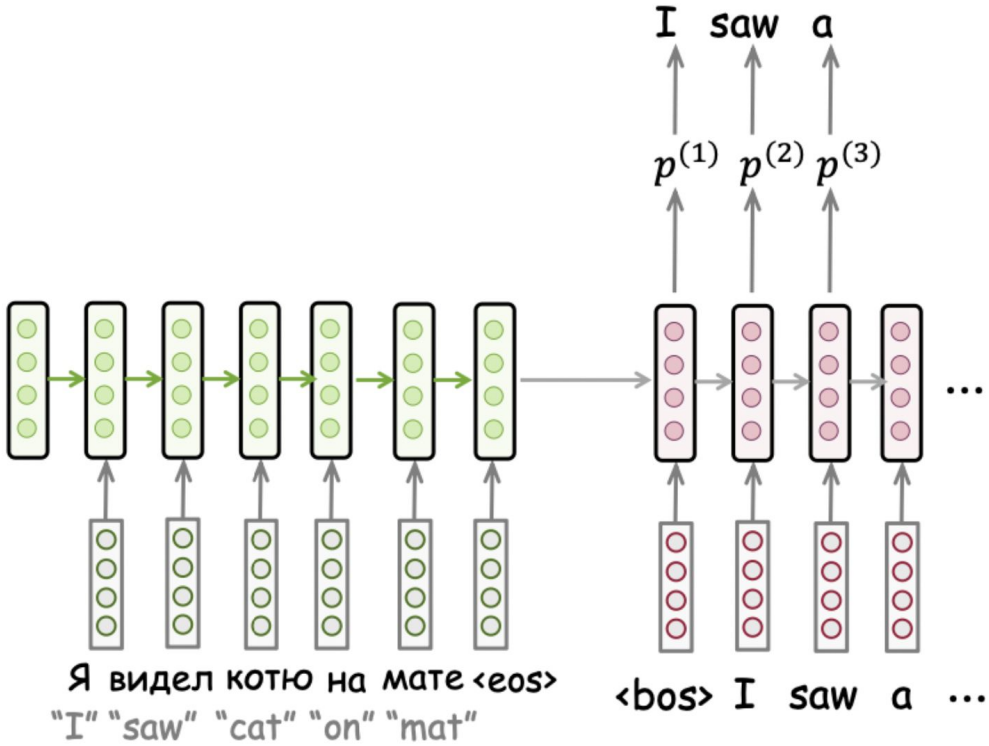
Марк Блуменау, магистратура ИИ

Почему RNN плохо?



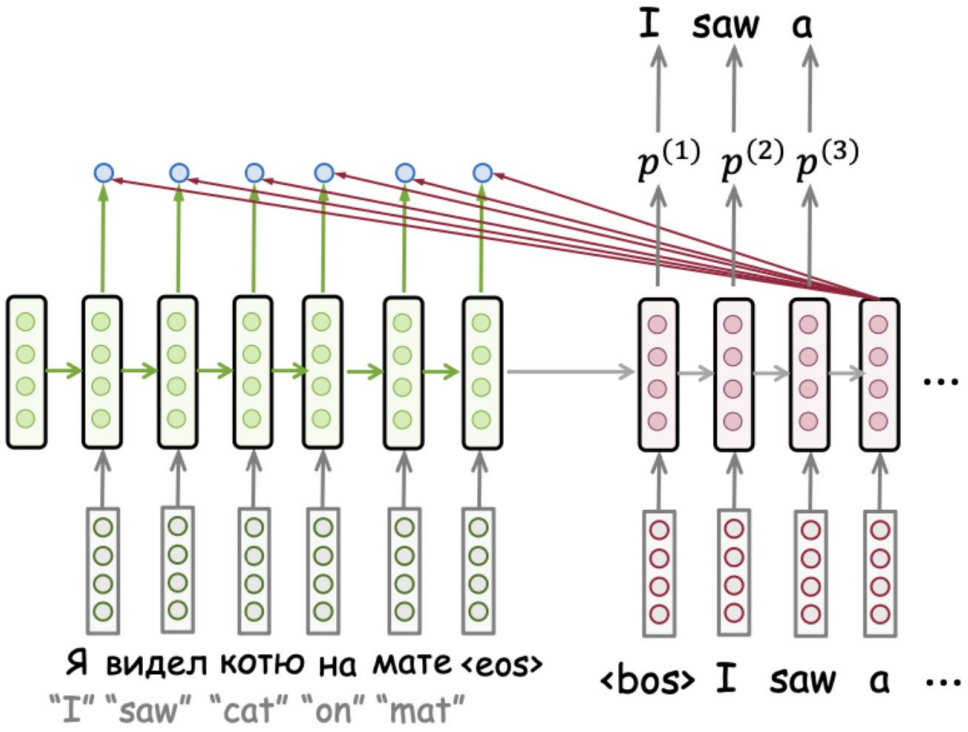
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention



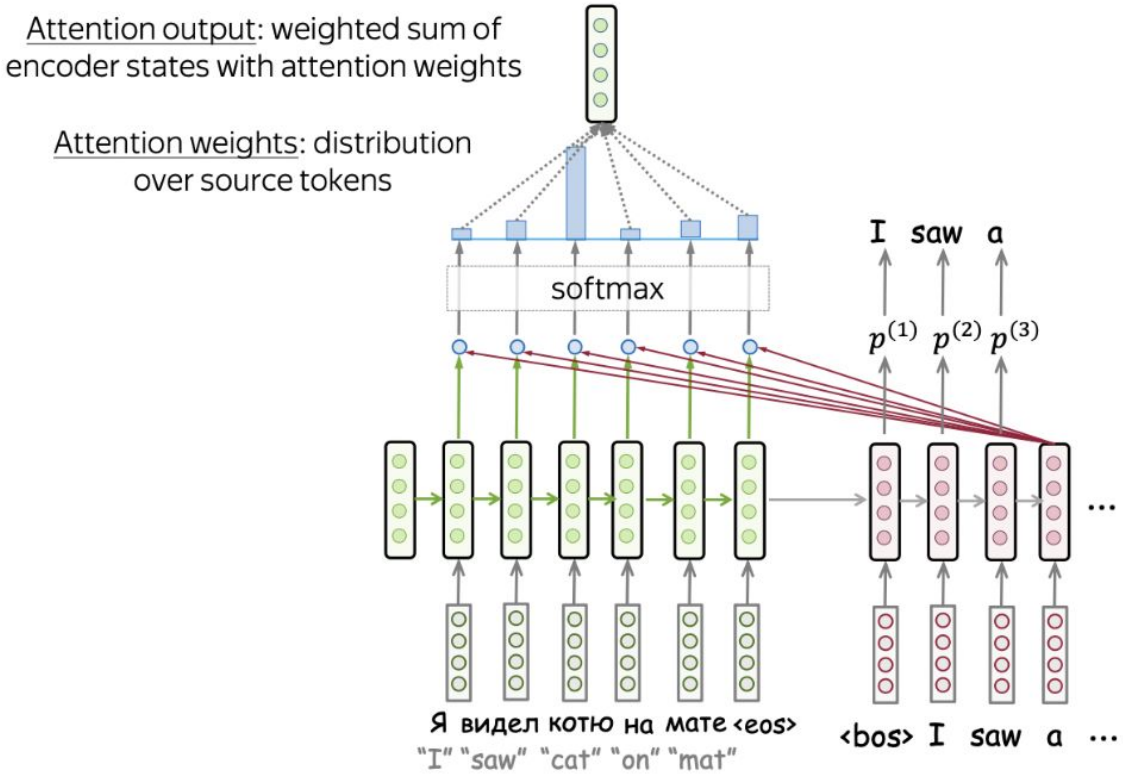
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

А хочется ведь так



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

Собираем

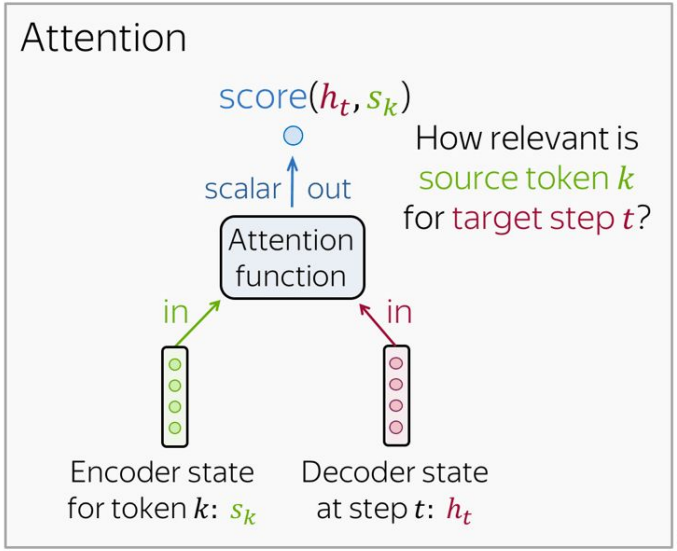


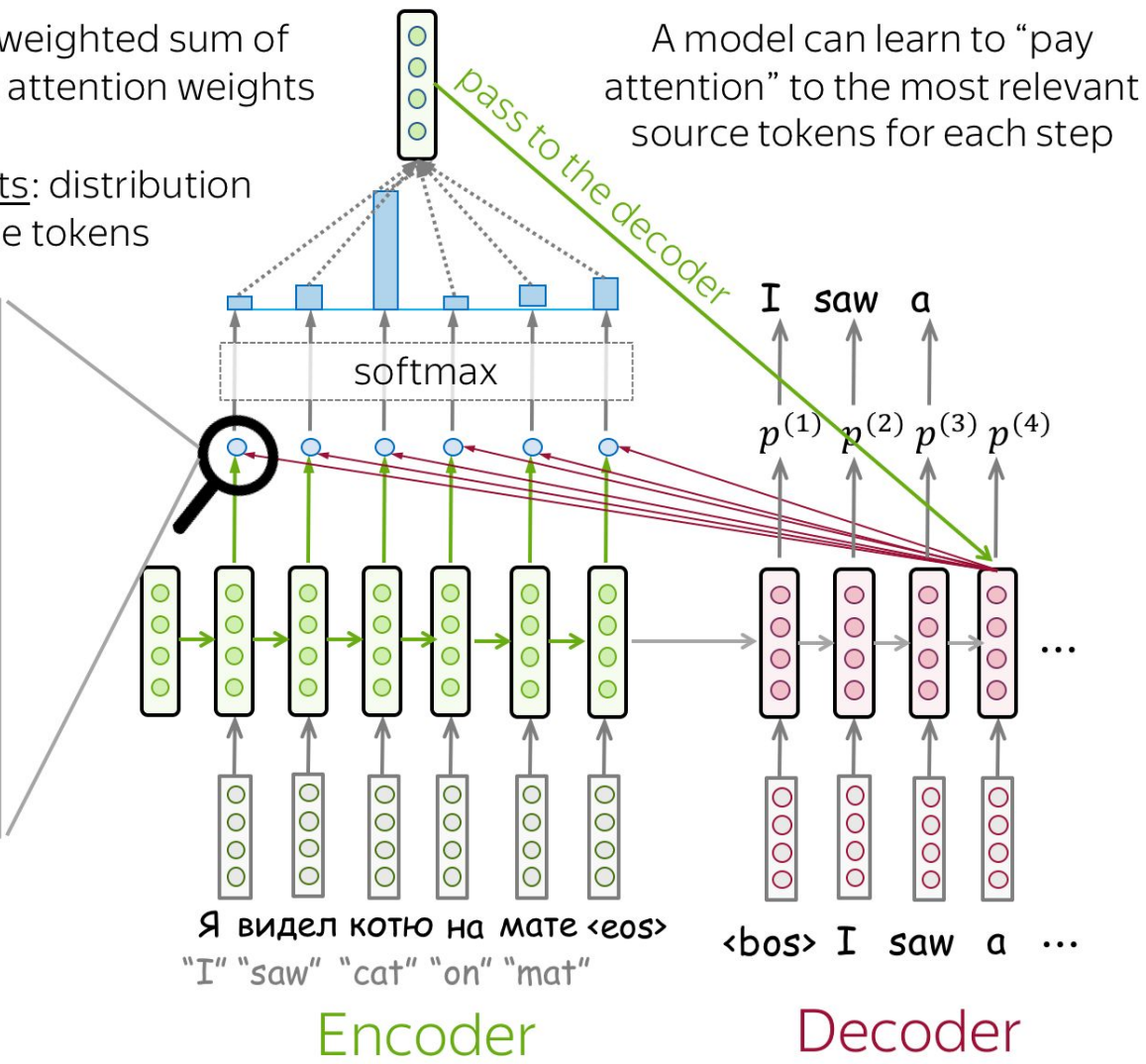
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

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





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

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?"

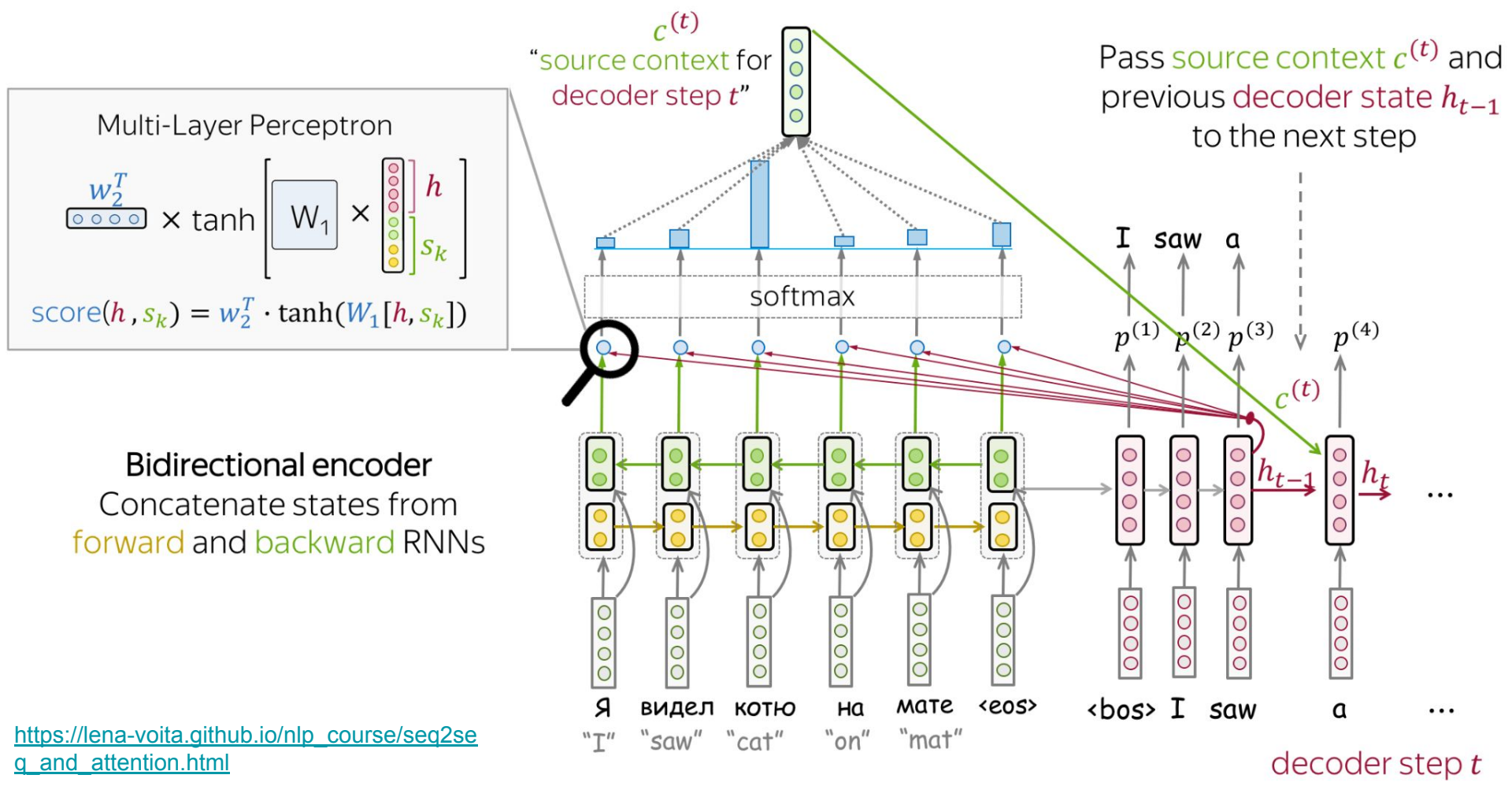
Attention input

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

h_t one decoder state

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

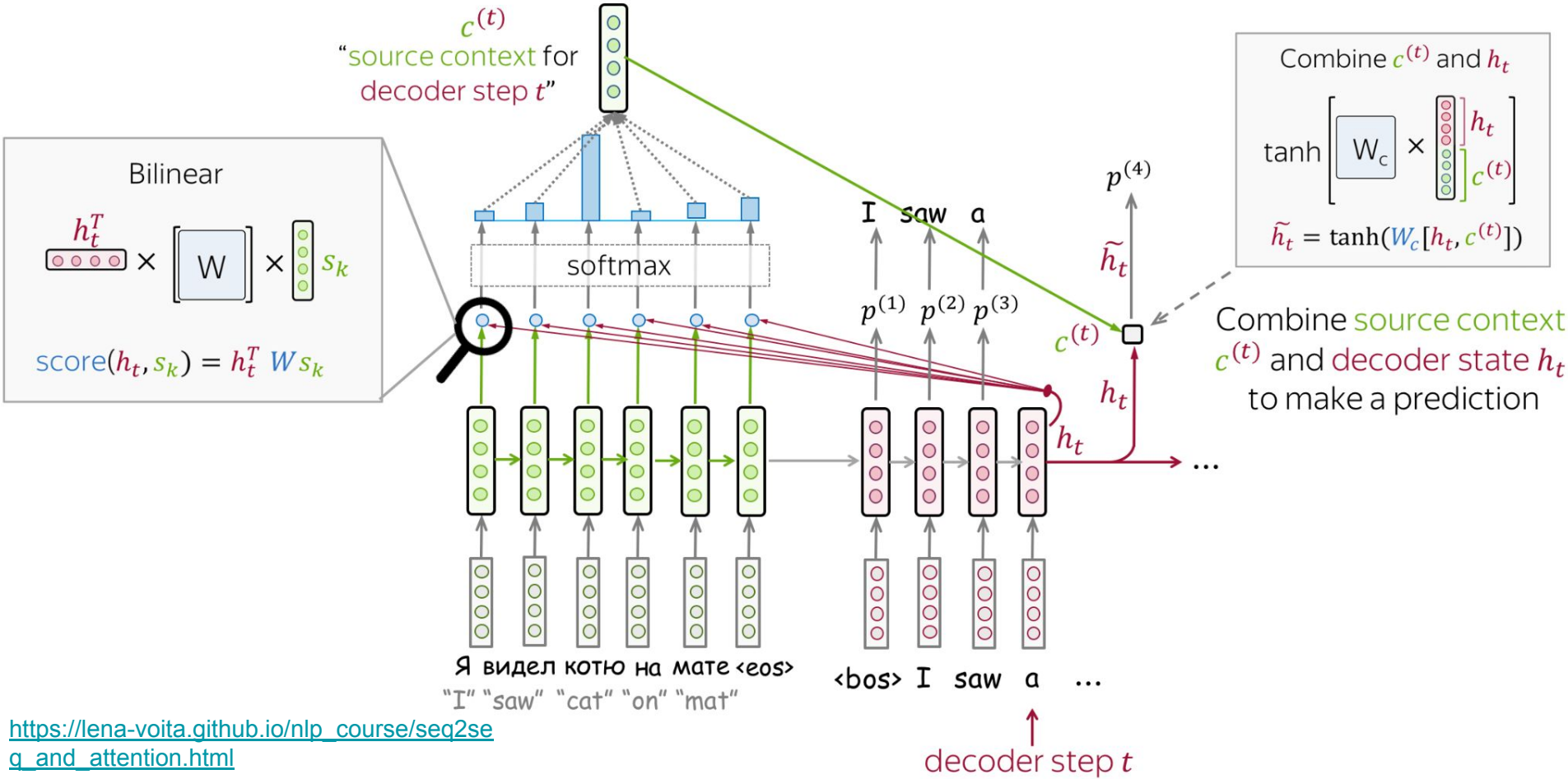
Badhau



Bidirectional encoder
Concatenate states from forward and backward RNNs

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

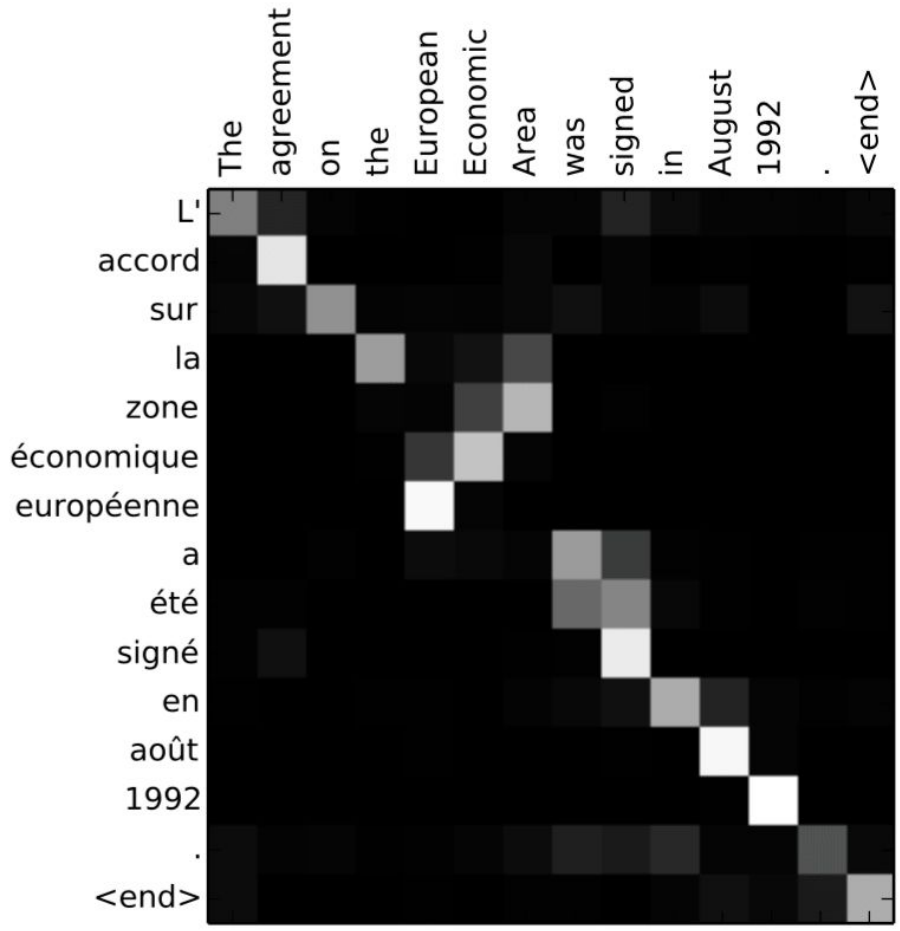
Luong

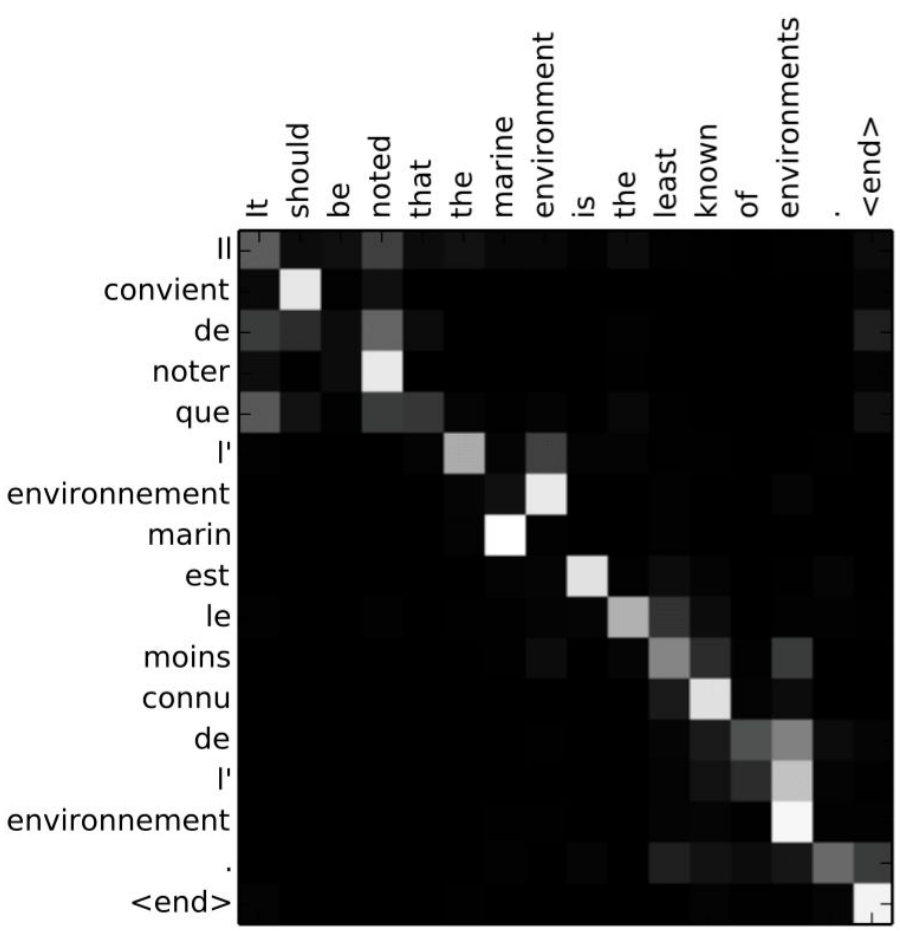


https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

А что получается?

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





Ну давайте его везде теперь засунем

	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/seq2seq_and_attention.html

Who is who

Encoder

Who is doing:

- all source tokens

What they are doing:

- look at each other
- update representations

repeat N times

Decoder

Who is doing:

- target token at the current step

What they are doing:

- looks at previous target tokens
- looks at source representations
- update representation

repeat N times

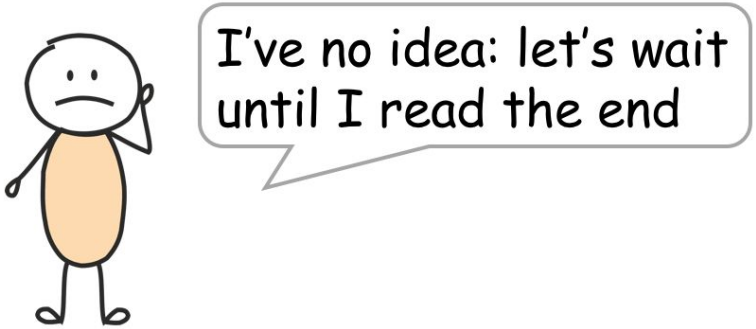
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

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

Напоминание: RNN медленный

I arrived at the bank after crossing the ... ...street? ...river?

What does **bank** mean in this sentence?



RNNs

O(N) steps to process a sentence with length N

Transformer

Constant number of steps to process any sentence

Стоп, а как смотреть самим на себя???

Зеркала > Зеркало в раме в античном стиле золотое купить в интернет-магазине

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

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

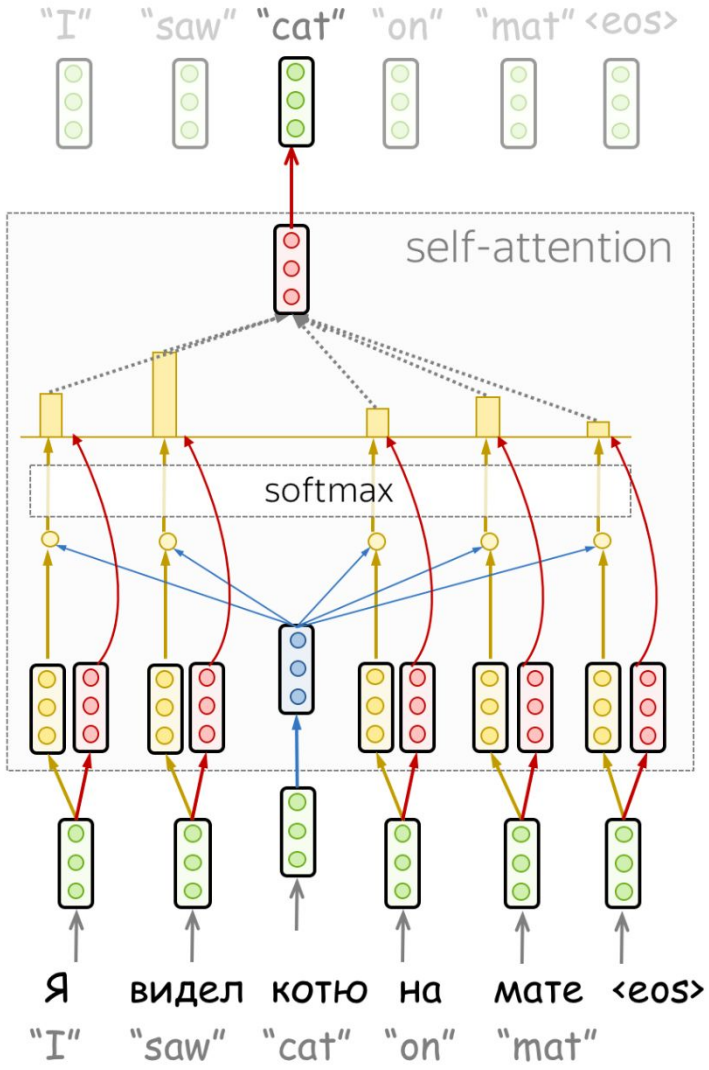
"Hi, I have this information – give me a large weight!"

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

"Here's the information I have!"

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

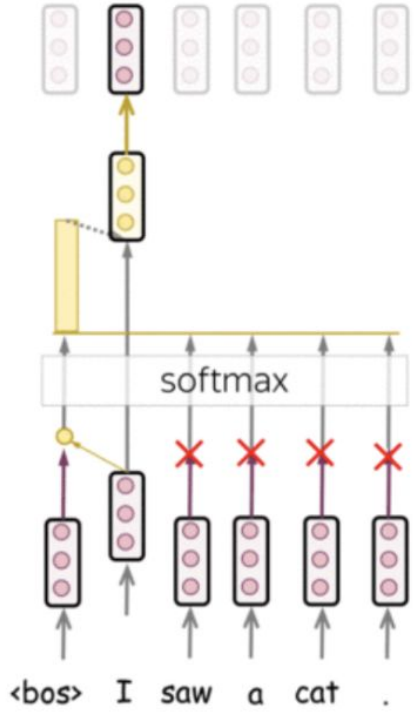
from (q) · to (k) · vector dimensionality of K, V



Смотреть в будущее оставим Ванге

In the decoder, we forbid looking at future tokens – we don't know them

Note: in training, decoder processes all target tokens at once – without masks, it would see future



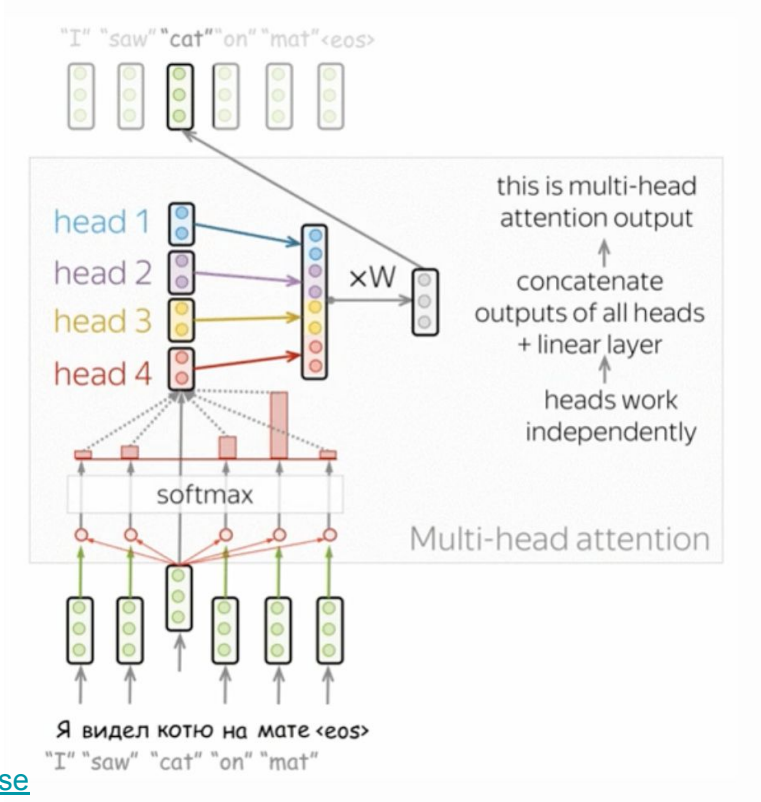
update token representation

gather context

"look" at the previous tokens (future tokens are masked out)

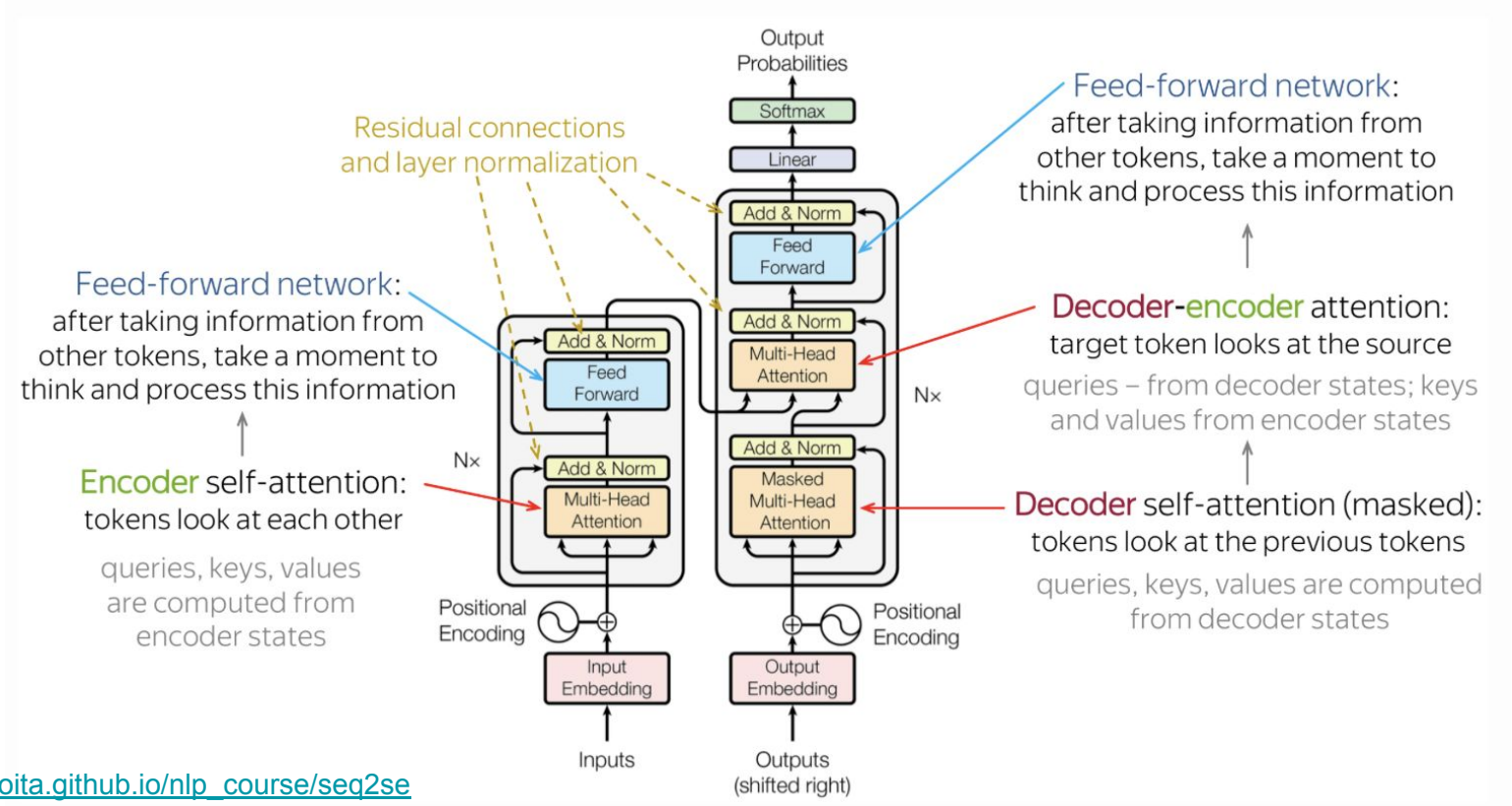
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Одна голова хорошо, а две – лучше



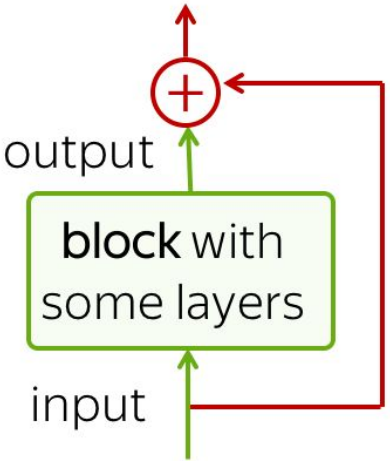
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Ура давайте собрать лего

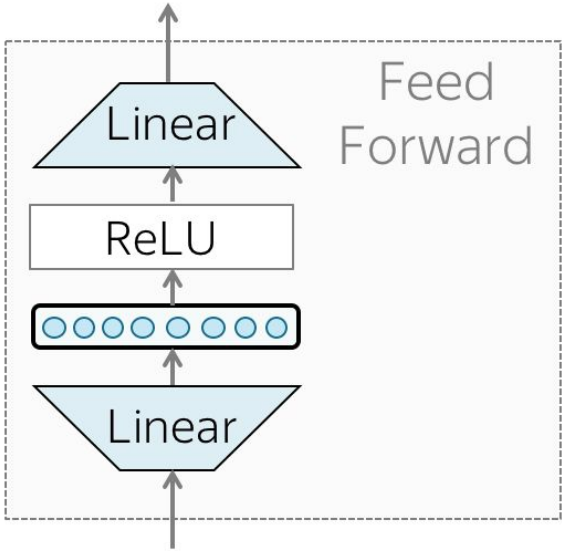


https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Эти покемоны нам известны

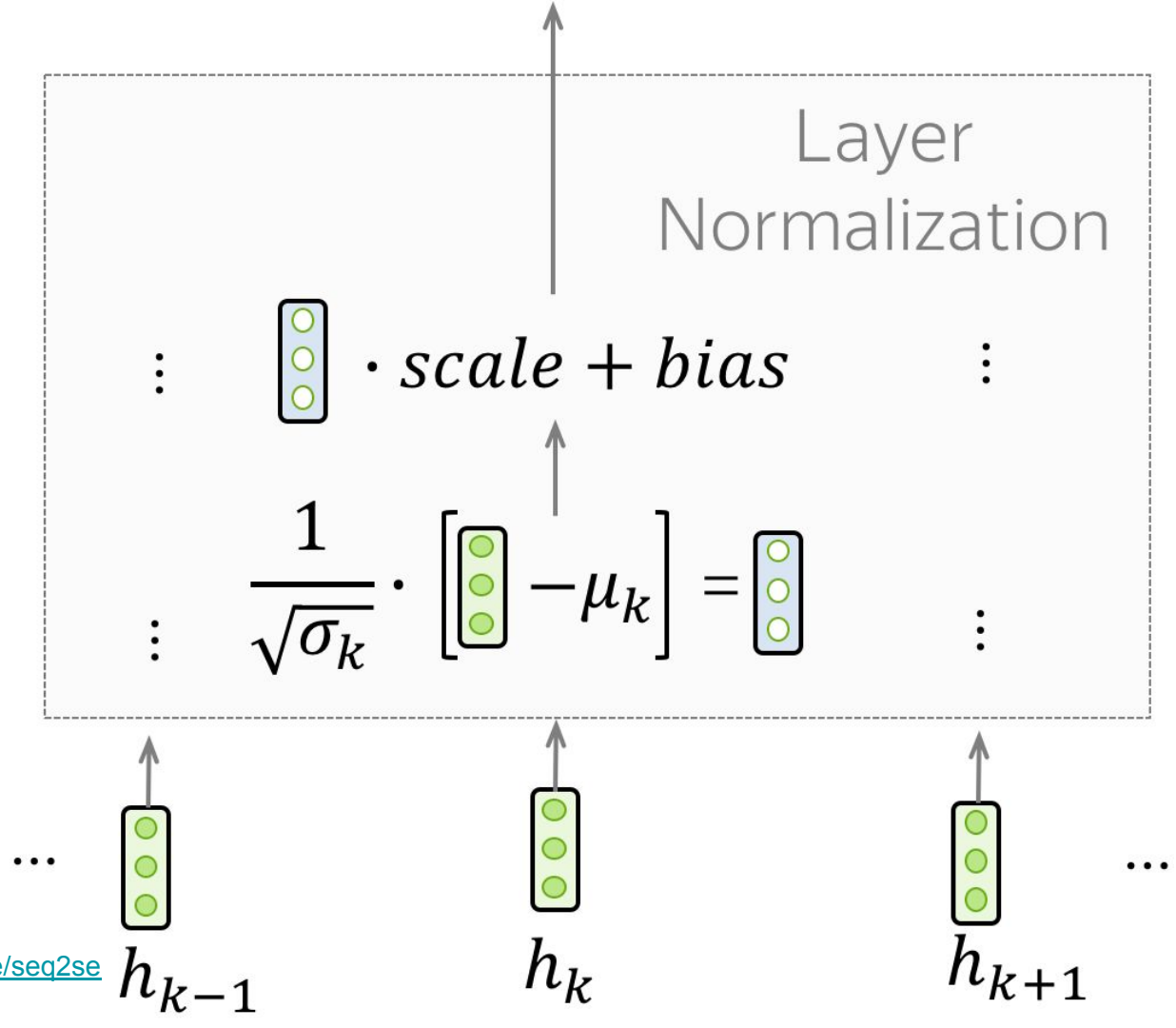


Residual connection: add a block's input to its output



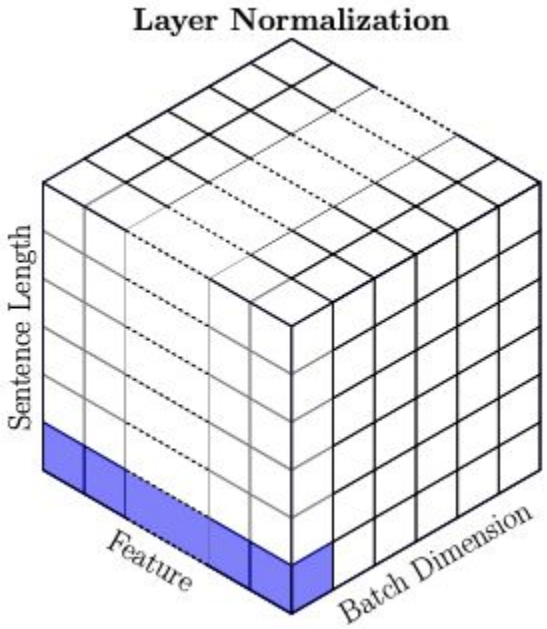
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

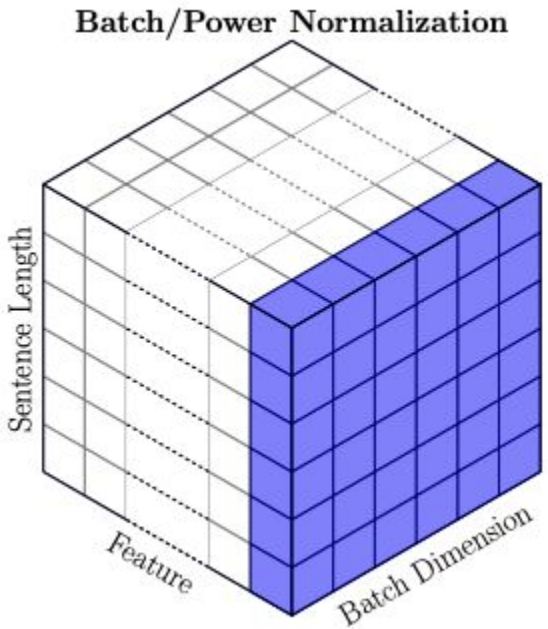
А этот нет



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

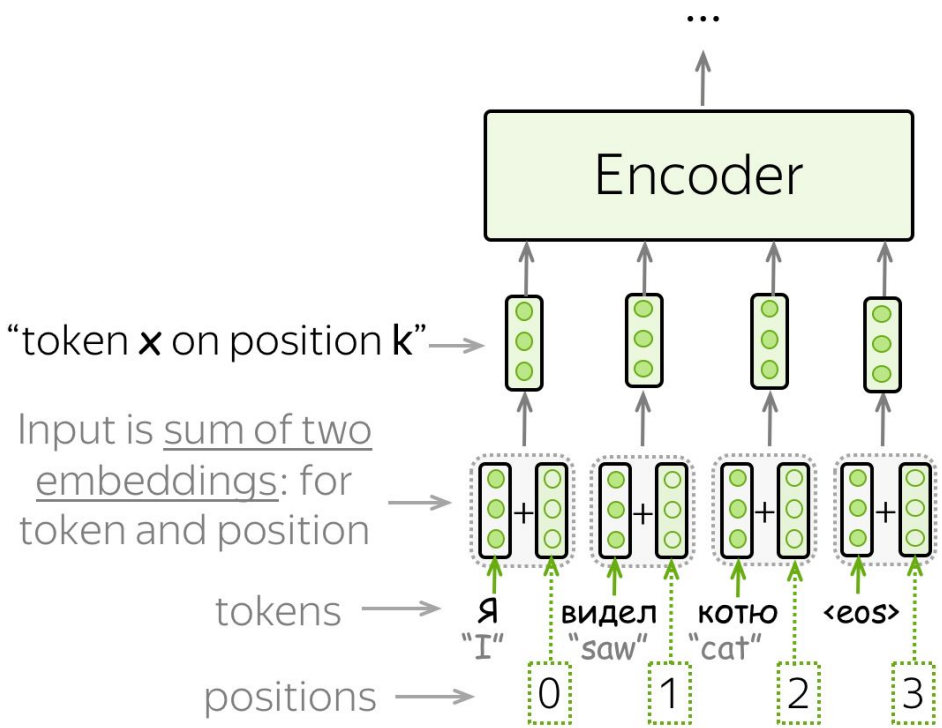
Просто ось поменяли





https://stats.stackexchange.com/questions/474440/why-do-transformers-use-layer-norm-instead-of-batch-norm

А позиция как работает?



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

Perplexity

$$L(y_{1:M}) = L(y_1, y_2, \ldots, y_M) = \sum_{t=1}^{M} \log_2 p(y_t | y_{<t})$$

Log-likelihood of the text

$$Loss(y_{1:M}) = -\sum_{t=1}^{M} \log p(y_t | y_{<t})$$

Note: cross-entropy (our loss) is negative log-likelihood

$$\boldsymbol{Perplexity}(y_{1:M}) = 2^{-\frac{1}{M} L(y_{1:M})}.$$

$$\boldsymbol{Perplexity}(y_{1:M}) = 2^{-\frac{1}{M} L(y_{1:M})} = 2^{-\frac{1}{M} \sum_{t=1}^{M} \log_2 p(y_t | y_{1:t-1})} = 2^{-\frac{1}{M} \cdot M \cdot \log_2 \frac{1}{|V|}} = 2^{\log_2 |V|} = |V|.$$