Seq2seq and Attention

October 4, 2018

Elena Voita
Yandex Research,
University of Amsterdam
lena-voita@yandex-team.ru

Plan

- Machine translation task
- Encoder-decoder framework
- Decoding
- Bahdanau attention model
- Attention is all you need: Transformer
- Pointer networks
- Attention helps for understanding the model (context-aware NMT)
- Attention: other use cases
- Hack of the day!

Machine translation task (seq2seq)

$\boldsymbol{x} = (x_1, x_2, \ldots, x_{Tx})$ - source sentence

$\boldsymbol{y} = (y_1, y_2, \ldots, y_{Ty})$ - target sentence

Any type of machine translation system can be defined as a function

$$\hat{\boldsymbol{y}} = \mathrm{mt}(\boldsymbol{x})$$

Translation is equivalent to finding a target sentence that maximizes the conditional probability of $\boldsymbol{y}$ given a source sentence $\boldsymbol{x}$, i.e.

$$\arg\max_y p(\boldsymbol{y}|\boldsymbol{x})$$

Machine translation task (seq2seq)

Machine translation systems create a probabilistic model for the probability of $\boldsymbol{y}$ given $\boldsymbol{x}$,

$$p(\boldsymbol{y}|\boldsymbol{x}, \theta),$$

and find the target sentence that maximizes this probability:

$$\hat{\boldsymbol{y}} = \arg\max_{y} p(\boldsymbol{y}|\boldsymbol{x}, \theta).$$

(θ – the parameters of the model specifying the probability distribution)

Machine translation task (seq2seq)

- Modeling

What the model $\mathrm{p}(\boldsymbol{y}|\boldsymbol{x}, \theta)$ will look like?

- Learning

We need a method to learn appropriate values for parameters θ from training data.

- Search

We need to solve the problem of finding the most probable sentence (solving “argmax”)

Conditional Language Model

Seq2seq directly models probability $P(y|x)$:

$$P(y|x) = P(y_1|x)\, P(y_2|y_1, x)\, P(y_3|y_1, y_2, x) \ldots, P(y_T|y_1, \ldots, y_{T-1}, x)$$

Conditional Language Model

Seq2seq directly models probability $P(y|x)$:

$$P(y|x) = P(y_1|x)\, P(y_2|y_1, x)\, P(y_3|y_1, y_2, x) \ldots, P(y_T|y_1, \ldots, y_{T-1}, x)$$

Probability of next target words so far and source sentence *x*

Conditional Language Model

Seq2seq directly models probability $P(y|x)$:

$$P(y|x) = P(y_1|x)\, \underbrace{P(y_2|y_1, x)}\, P(y_3|y_1, y_2, x) \ldots, \underbrace{P(y_T|y_1, \ldots, y_{T-1}, x)}$$

Probability of next target words so far and source sentence *x*

Why conditional language model?

Conditional Language Model

Seq2seq directly models probability $P(y|x)$:

$$P(y|x) = P(y_1|x)\, P(y_2|y_1, x)\, P(y_3|y_1, y_2, x) \ldots, P(y_T|y_1, \ldots, y_{T-1}, x)$$

Probability of next target words so far and source sentence *x*

Why conditional language model?

- ➢ LM – predicts next word
- ➢ Conditional – predictions conditioned on source sentence

Encoder-Decoder Framework

RECAP: RNN Language Models

Output distribution

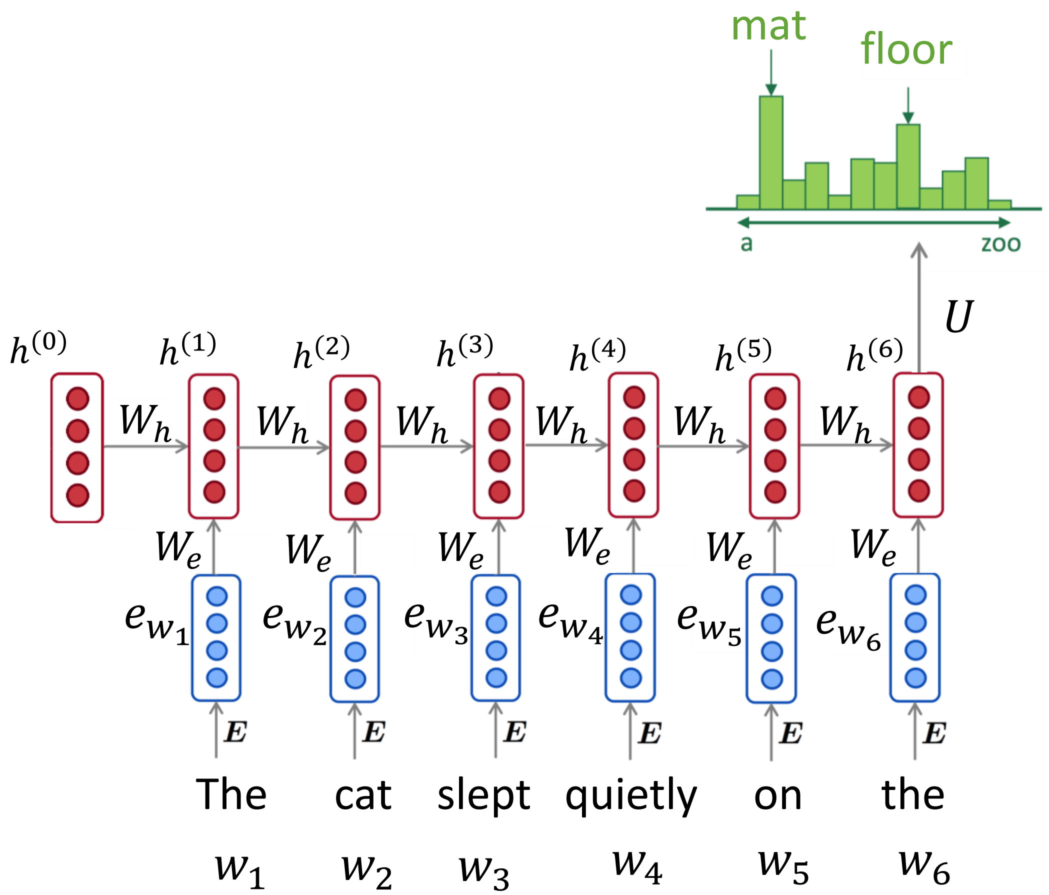
$$\hat{y}_t = \text{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$$

Hidden states

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e_{w_{t-1}} + b_1\right)$$

Word embeddings

Words/tokens

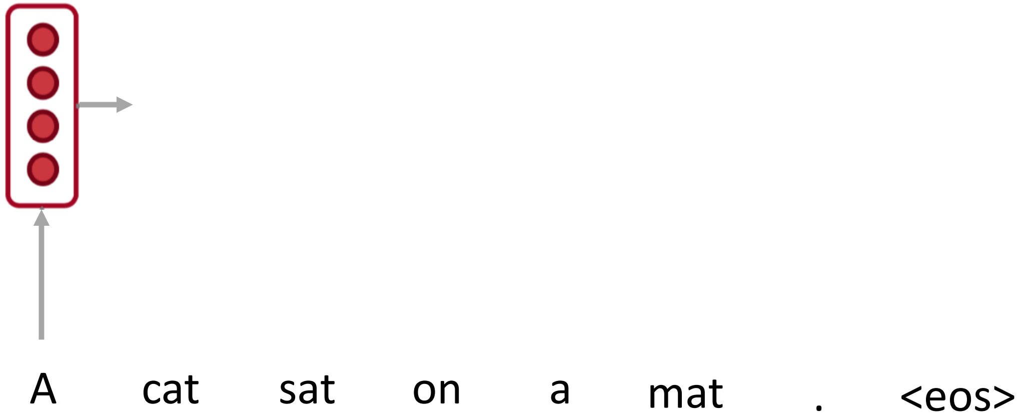


RNN Encoder-Decoder (Vanilla)

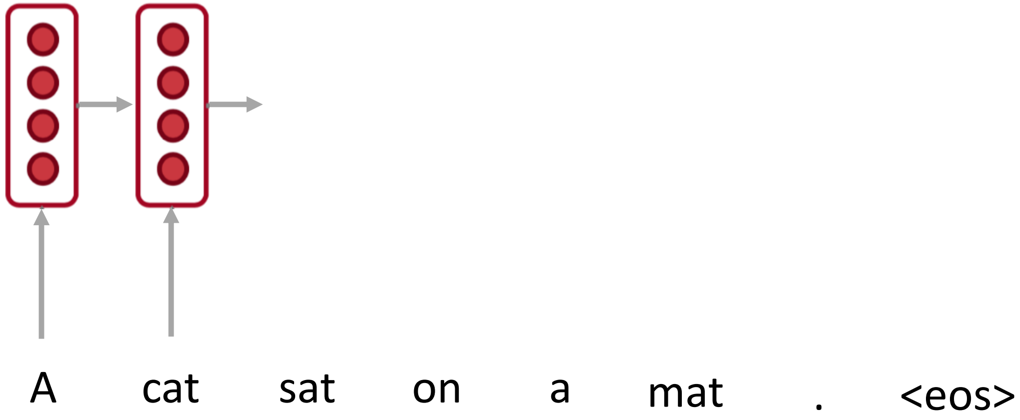
Source sentence

$\underbrace{\text{A} \quad \text{cat} \quad \text{sat} \quad \text{on} \quad \text{a} \quad \text{mat} \quad . \quad \text{<eos>}}$

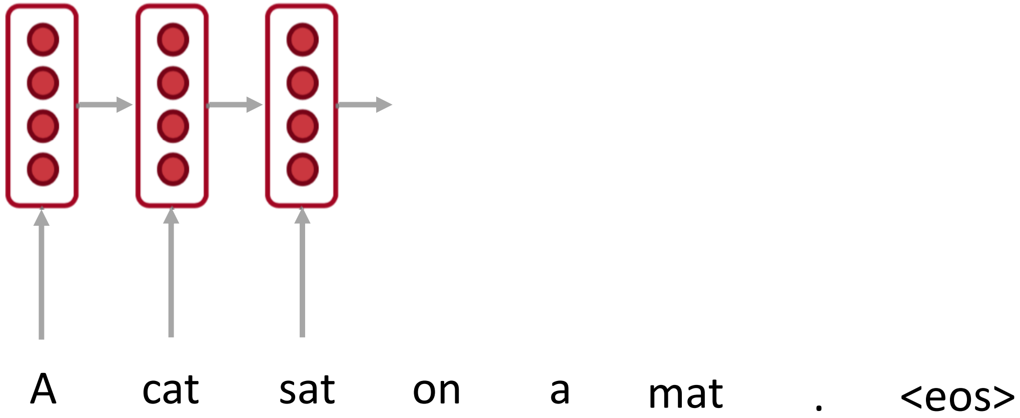
RNN Encoder-Decoder (Vanilla)



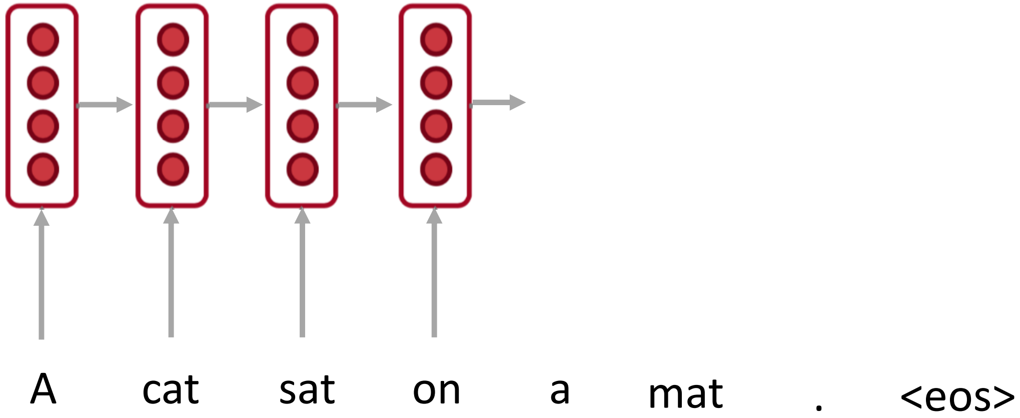
RNN Encoder-Decoder (Vanilla)



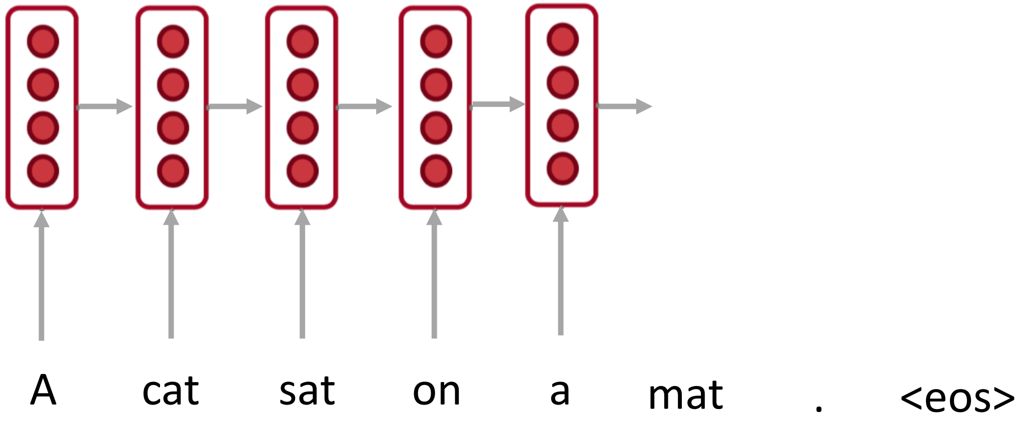
RNN Encoder-Decoder (Vanilla)



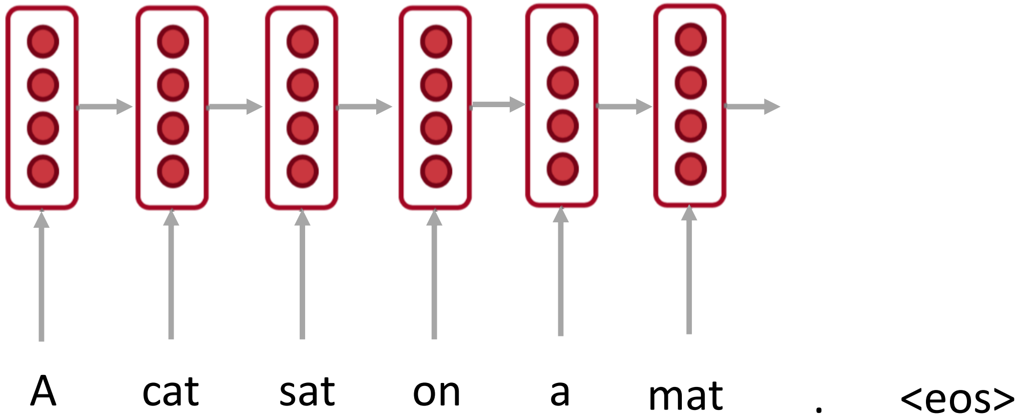
RNN Encoder-Decoder (Vanilla)



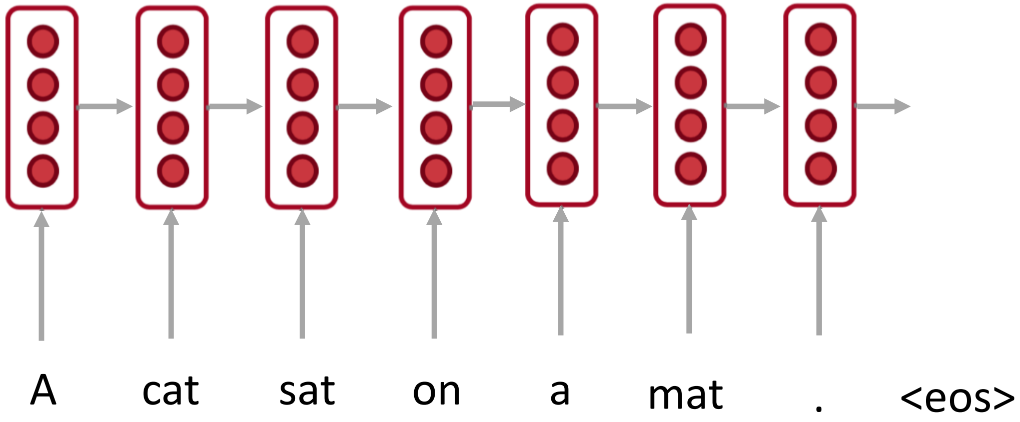
RNN Encoder-Decoder (Vanilla)



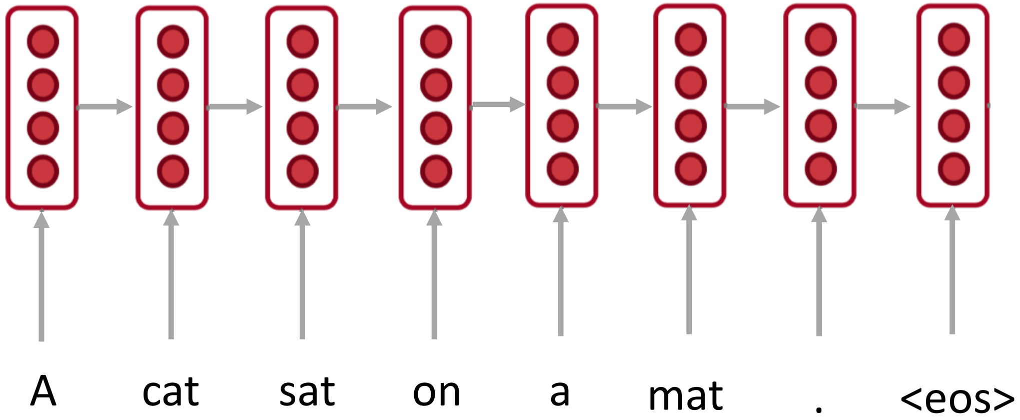
RNN Encoder-Decoder (Vanilla)



RNN Encoder-Decoder (Vanilla)

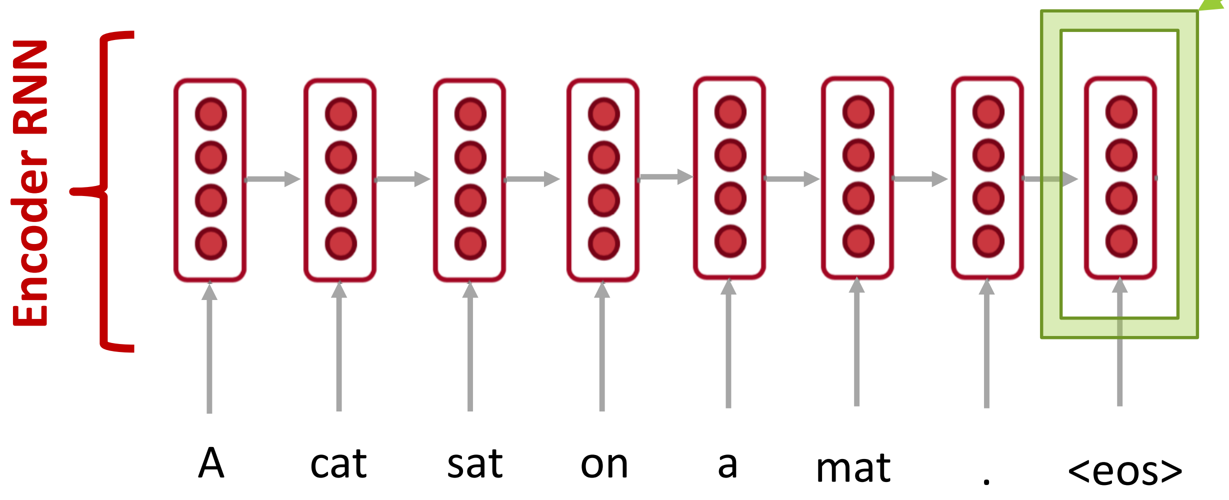


RNN Encoder-Decoder (Vanilla)

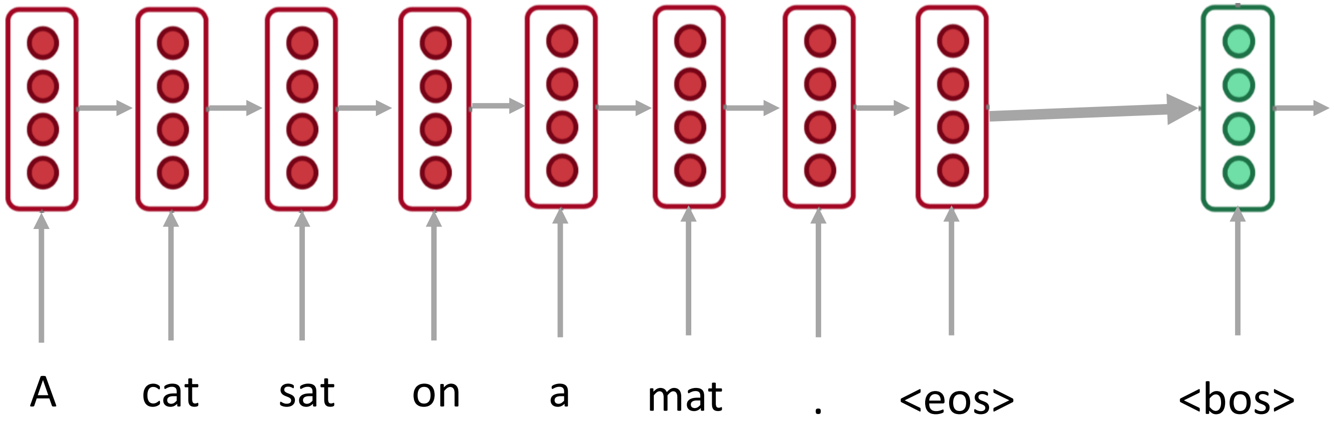


RNN Encoder-Decoder (Vanilla)

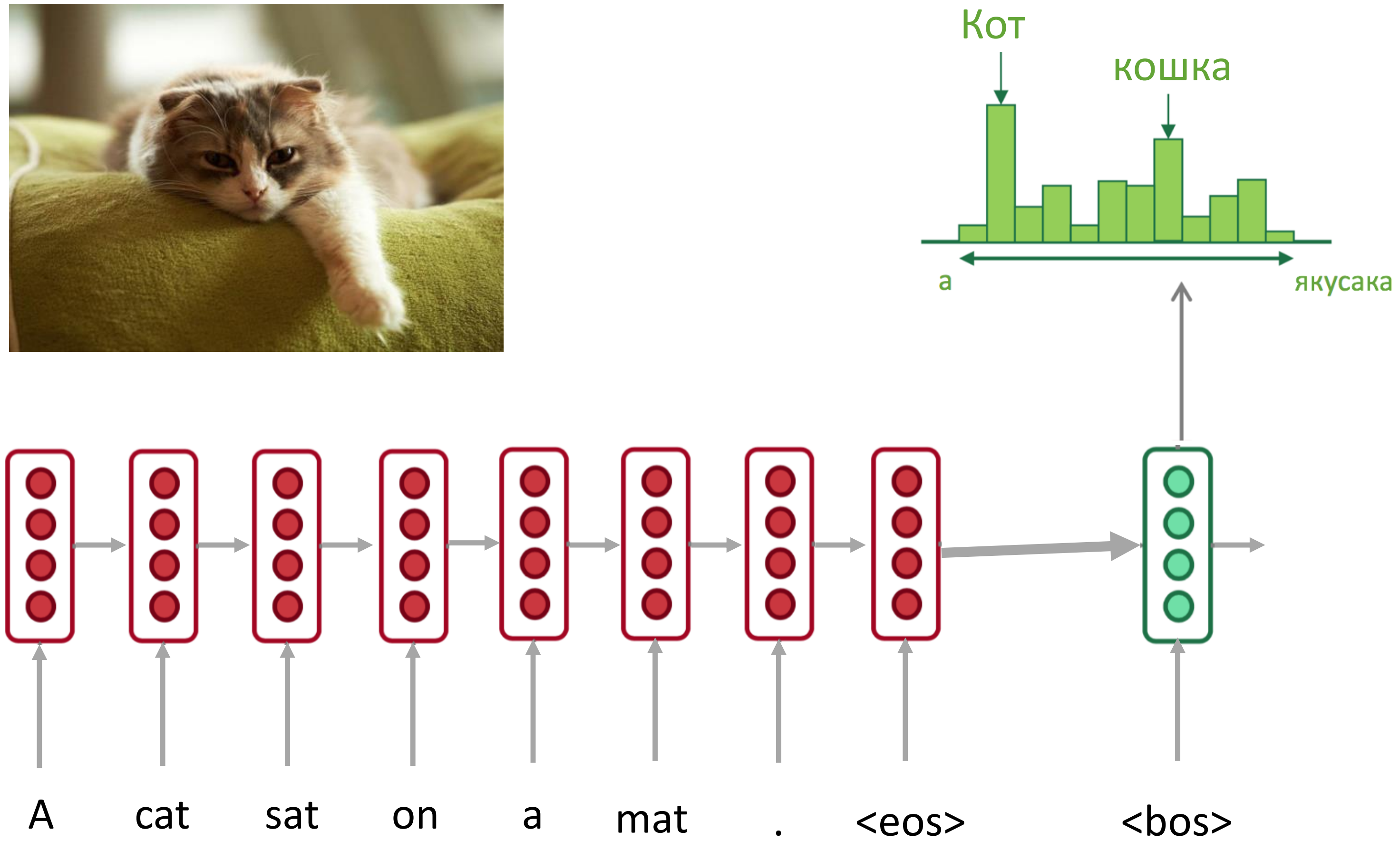
Encoding of a source sentence.
Provides an initial state for decoder RNN



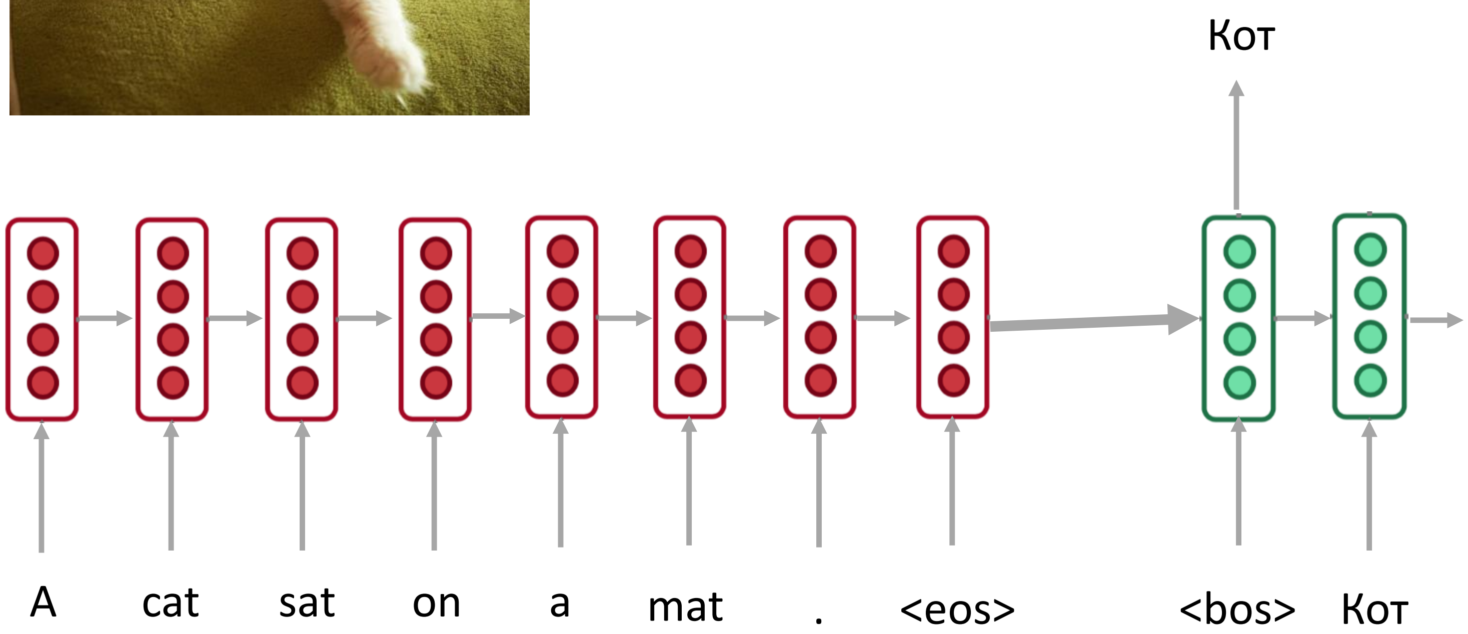
RNN Encoder-Decoder (Vanilla)



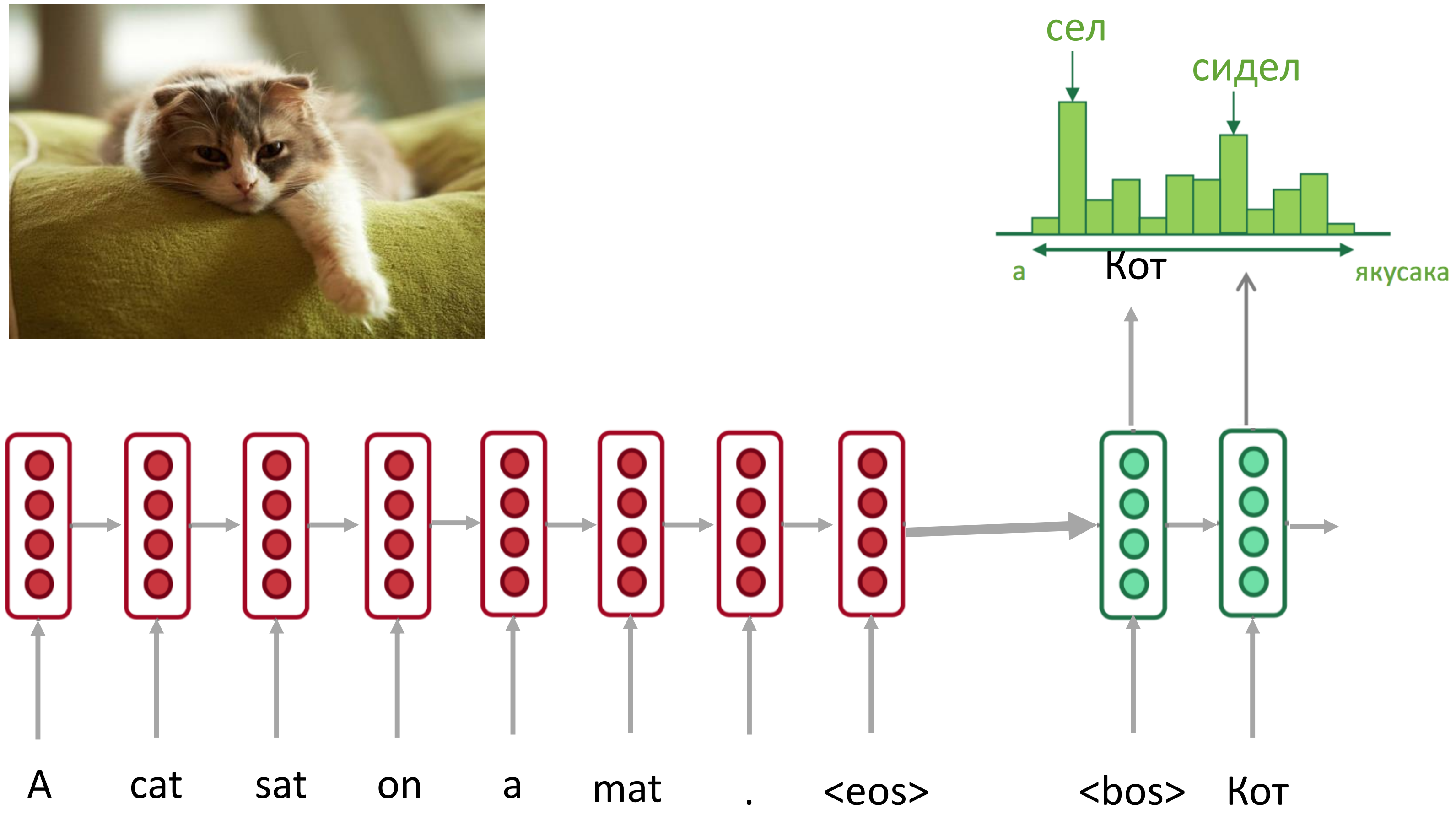
RNN Encoder-Decoder (Vanilla)



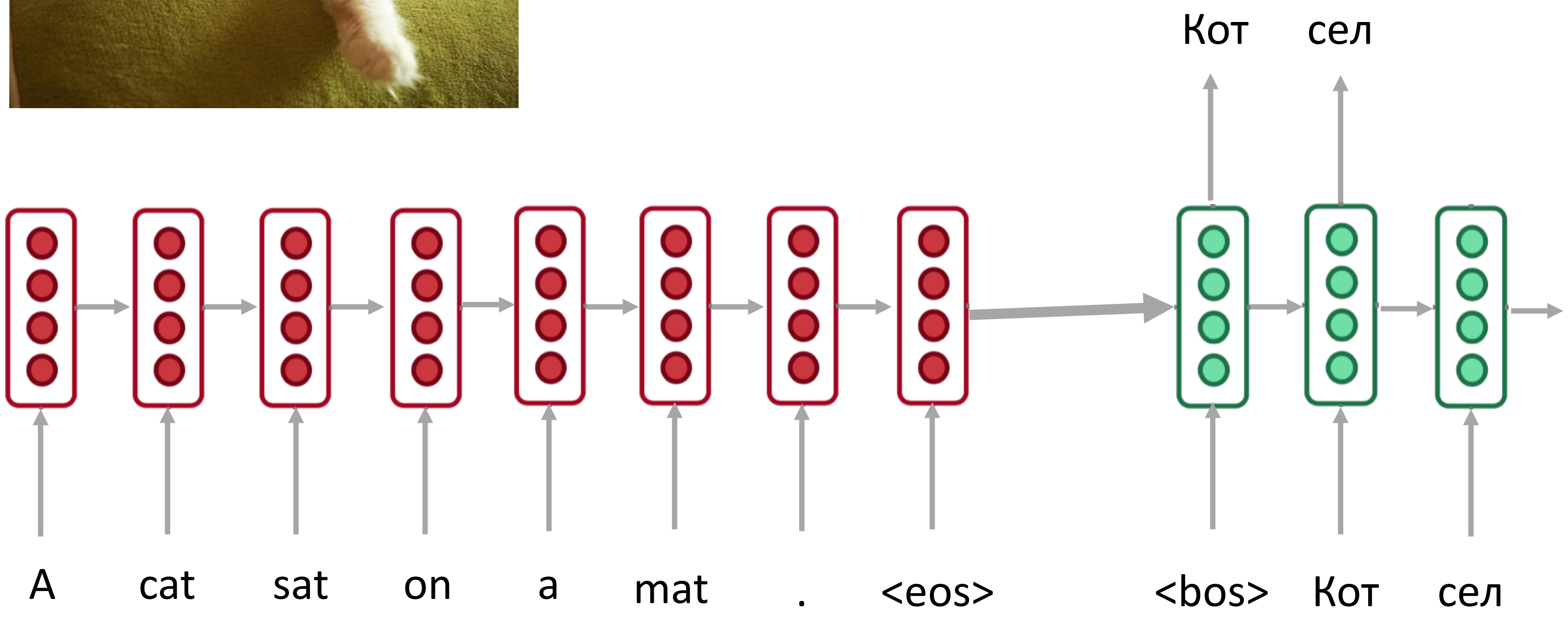
RNN Encoder-Decoder (Vanilla)



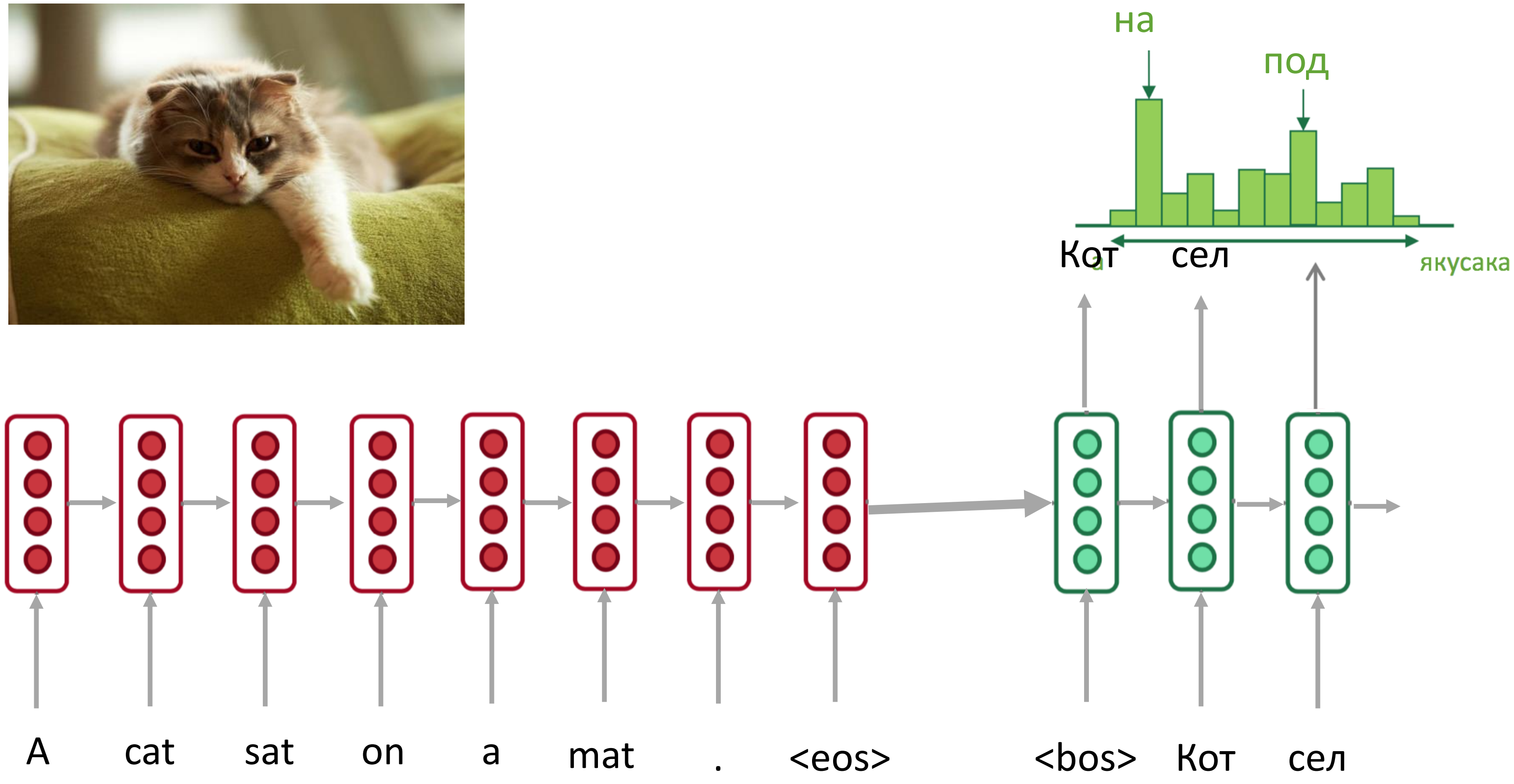
RNN Encoder-Decoder (Vanilla)



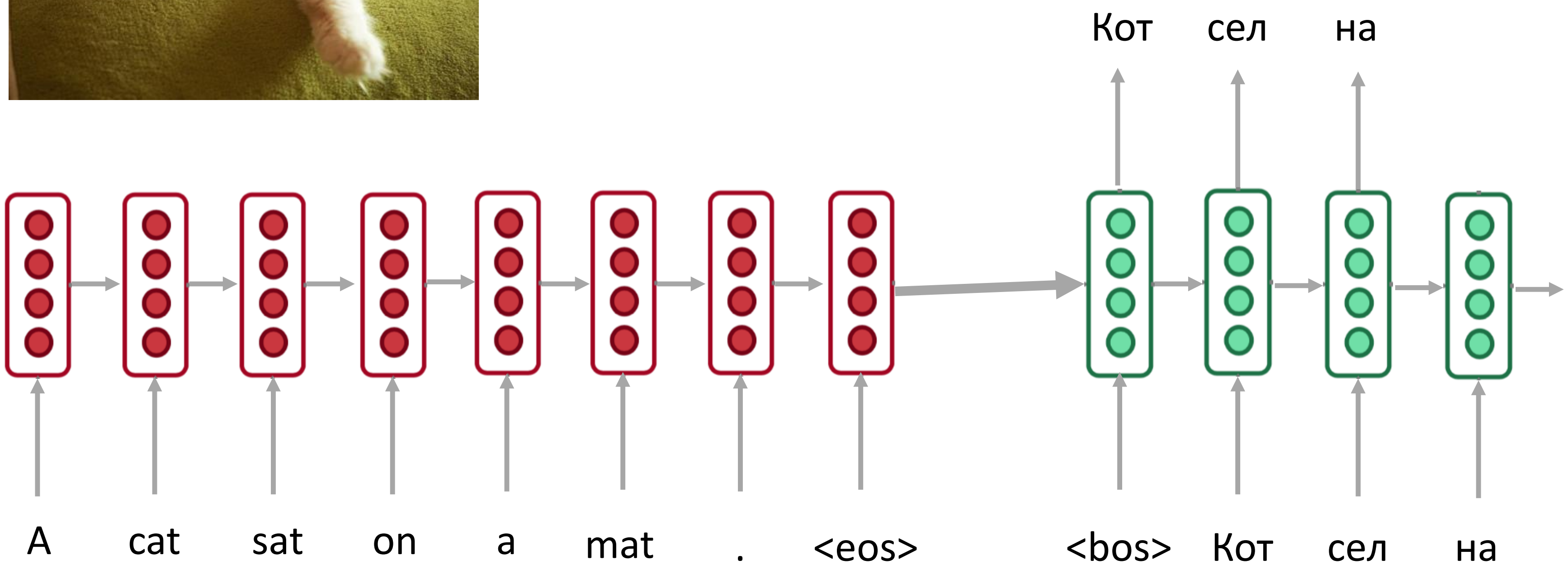
RNN Encoder-Decoder (Vanilla)



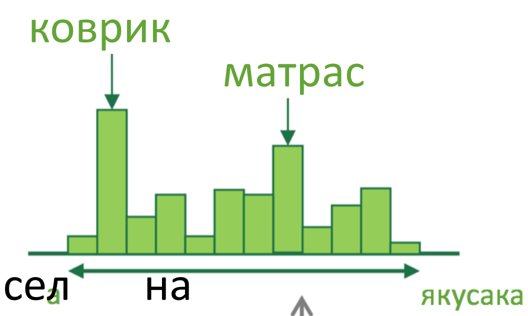
RNN Encoder-Decoder (Vanilla)

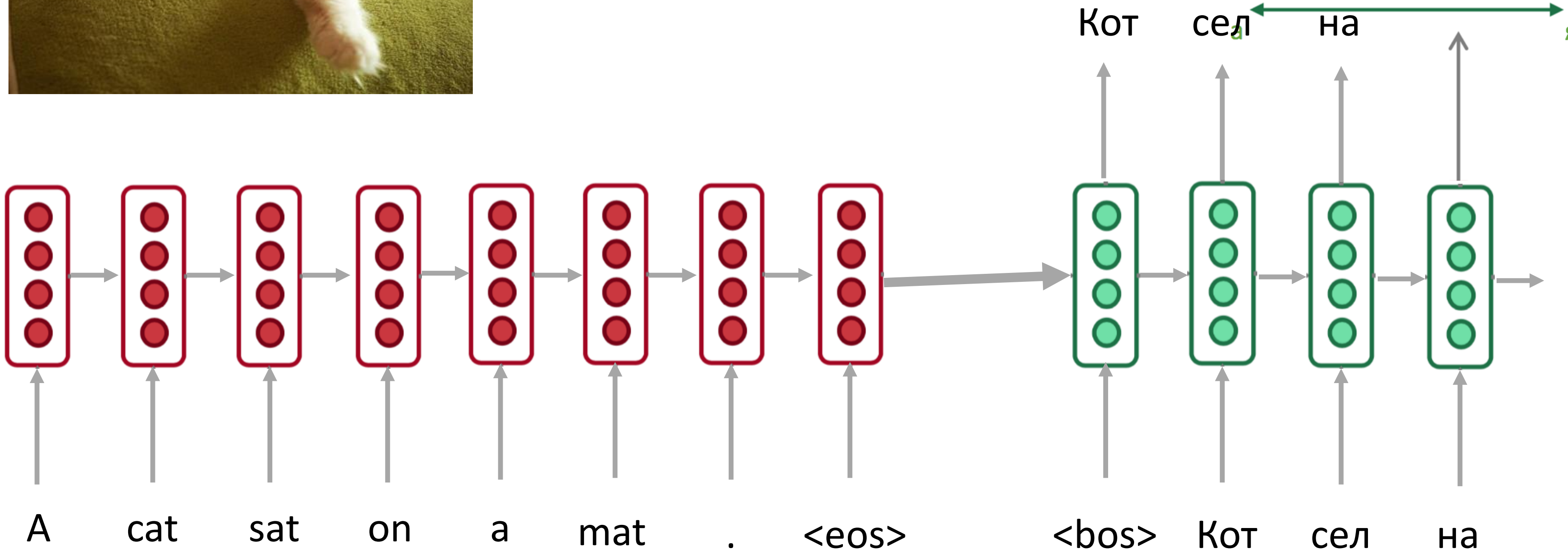


RNN Encoder-Decoder (Vanilla)

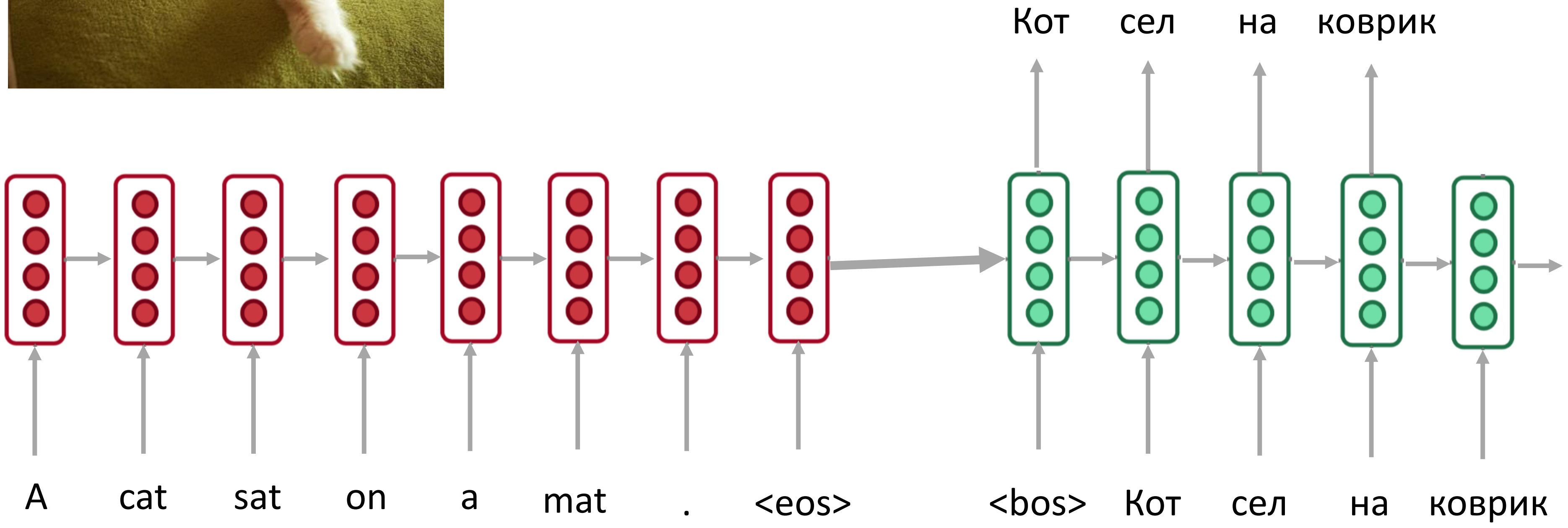


RNN Encoder-Decoder (Vanilla)

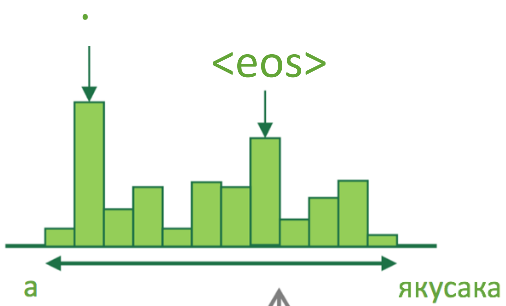


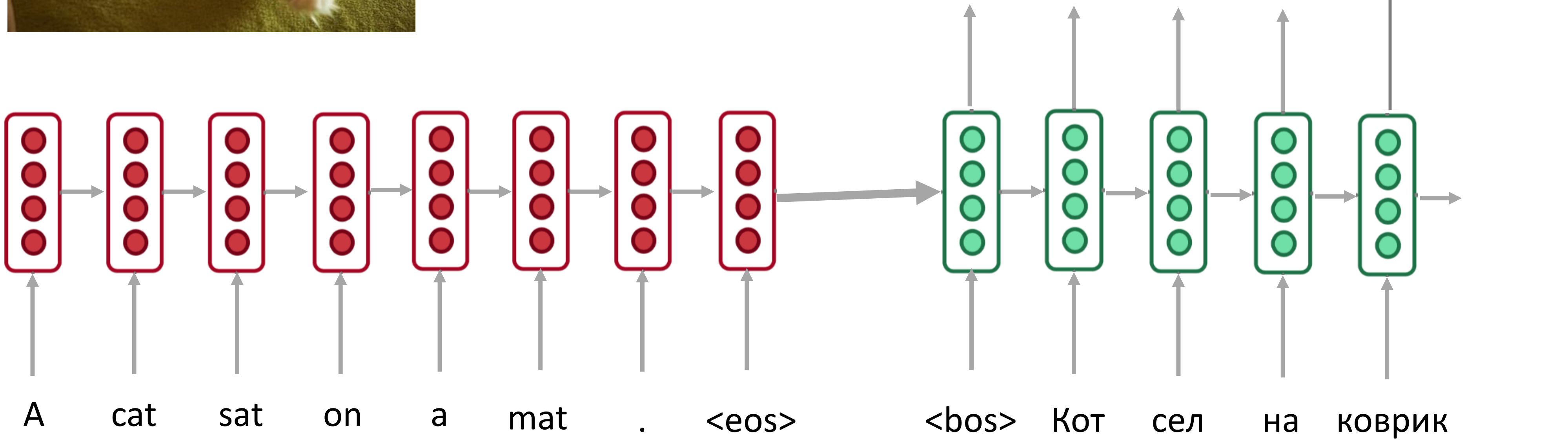


RNN Encoder-Decoder (Vanilla)

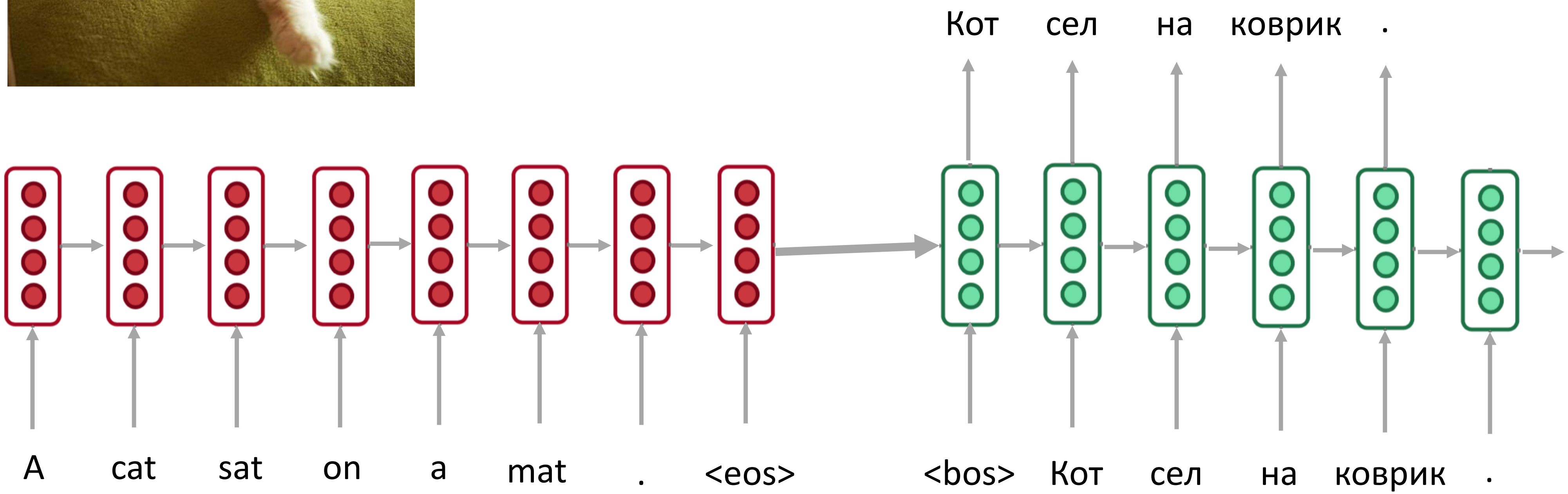


RNN Encoder-Decoder (Vanilla)

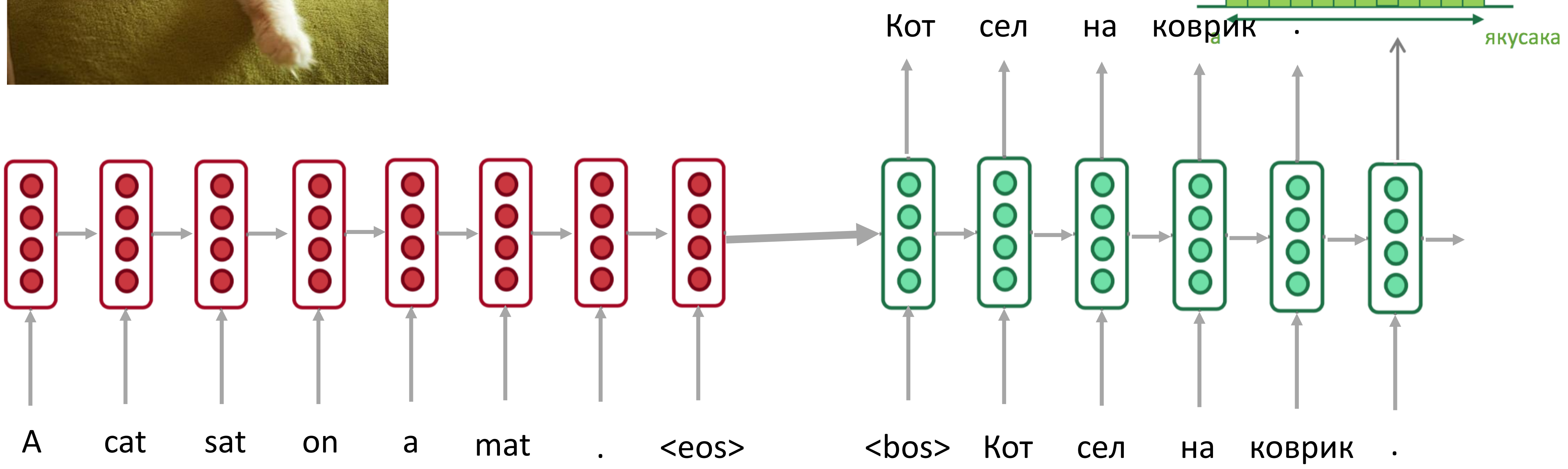




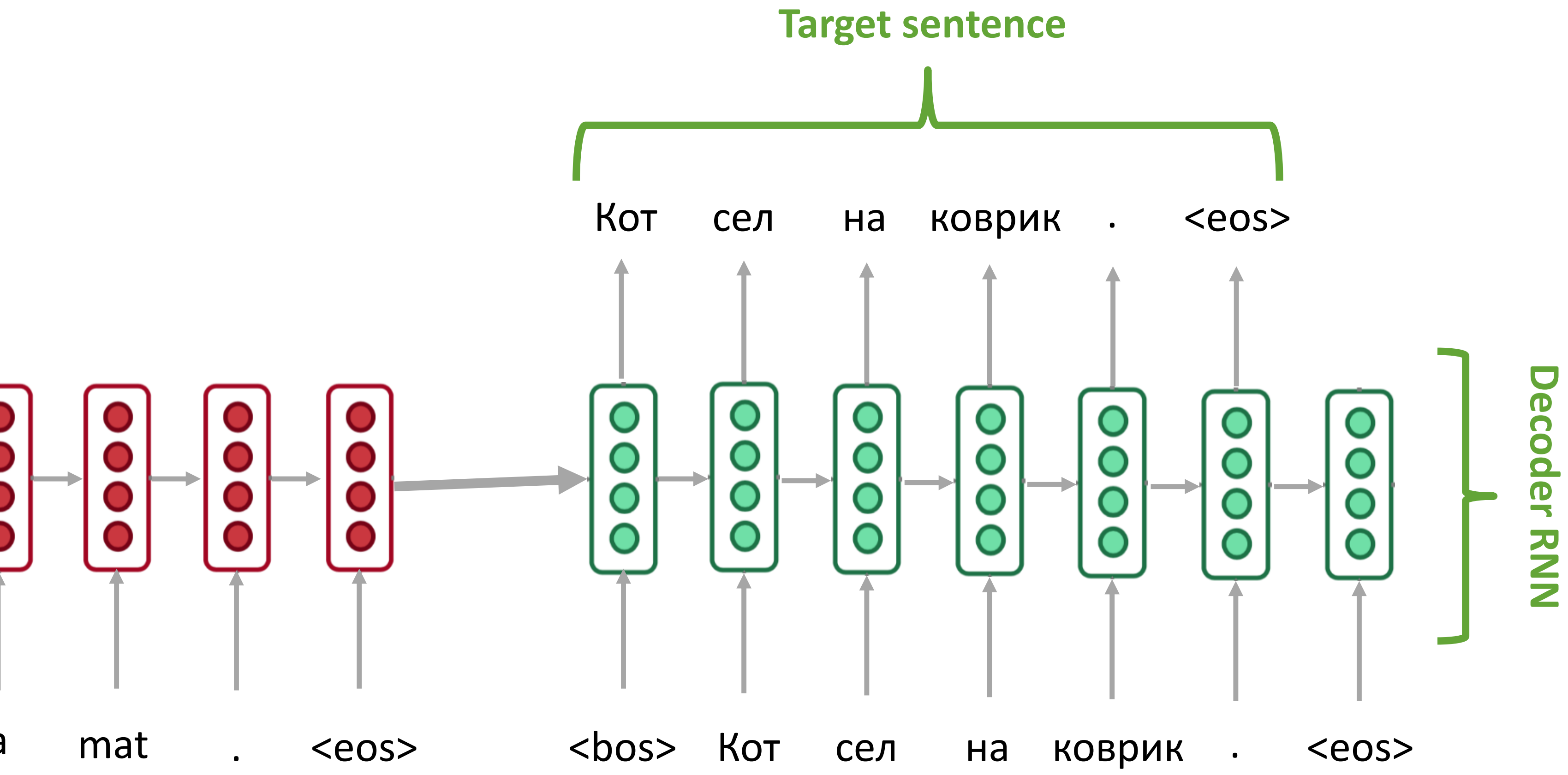
RNN Encoder-Decoder (Vanilla)



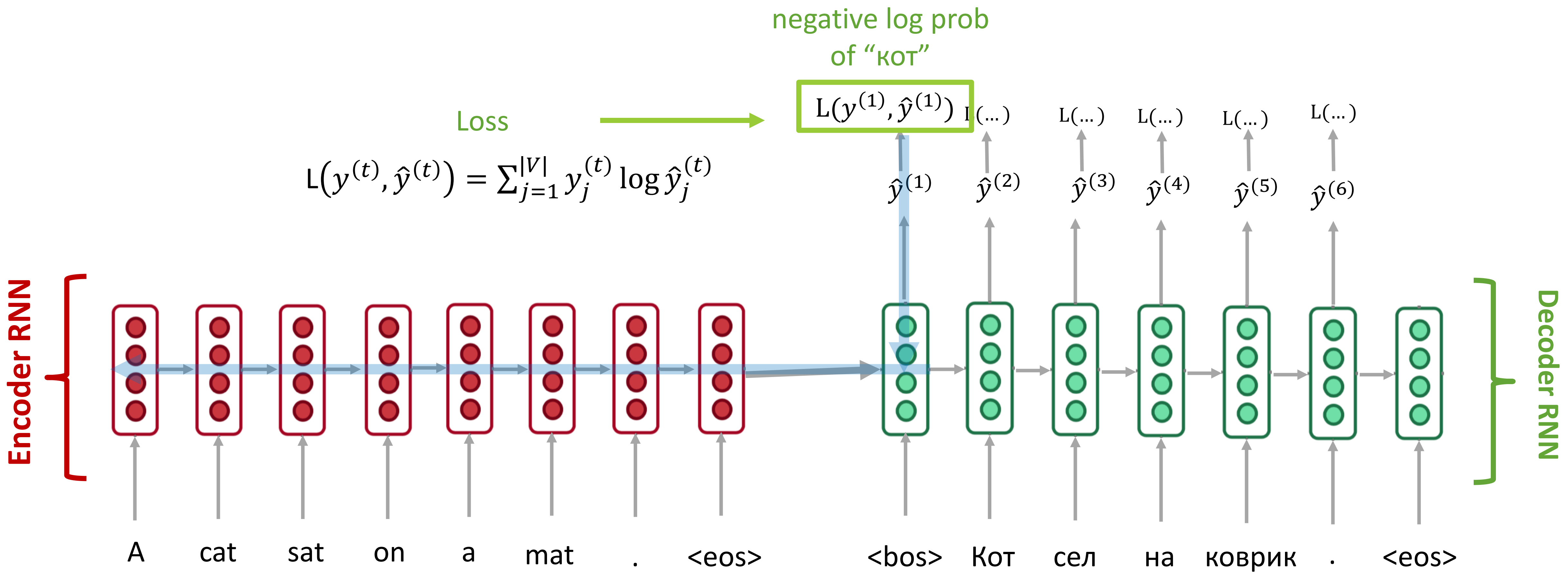
RNN Encoder-Decoder (Vanilla)



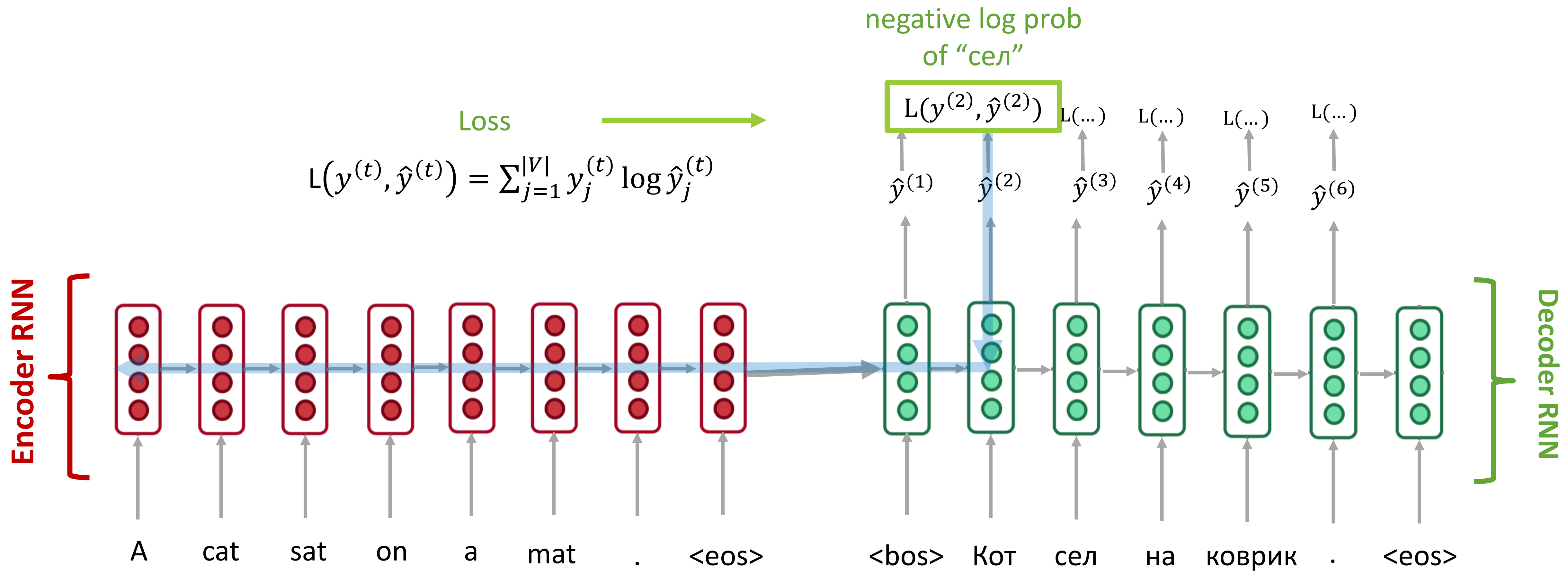
RNN Encoder-Decoder (Vanilla)



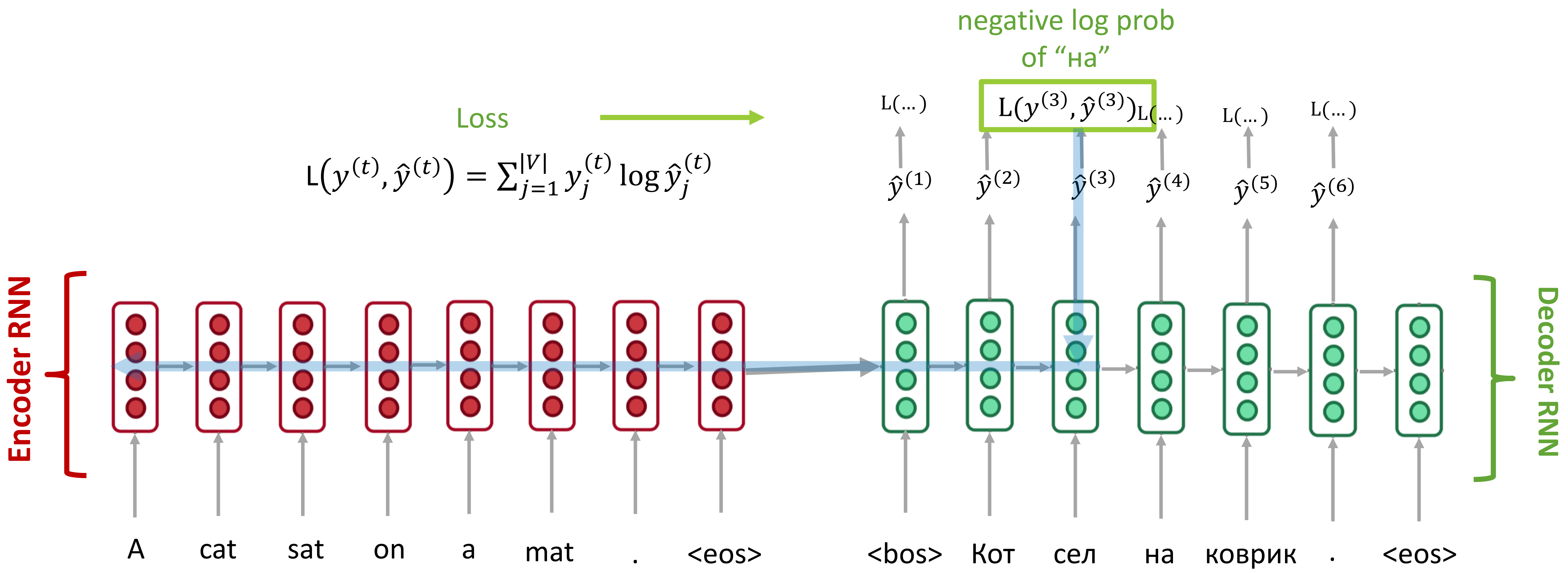
RNN Encoder-Decoder (Vanilla)



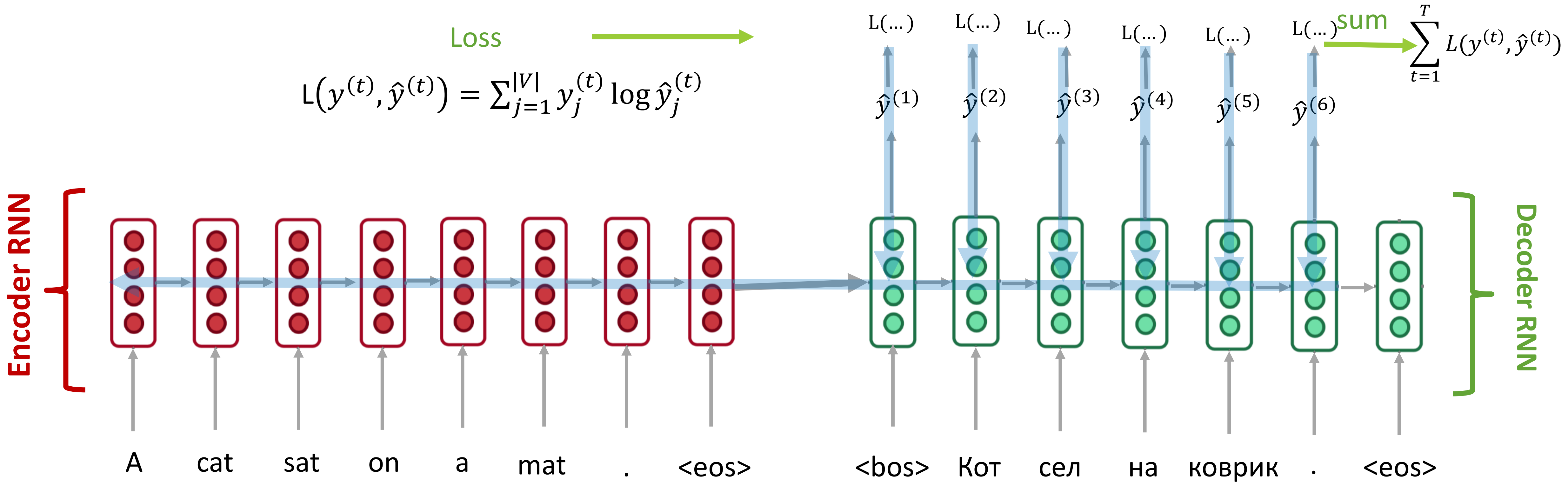
RNN Encoder-Decoder (Vanilla)



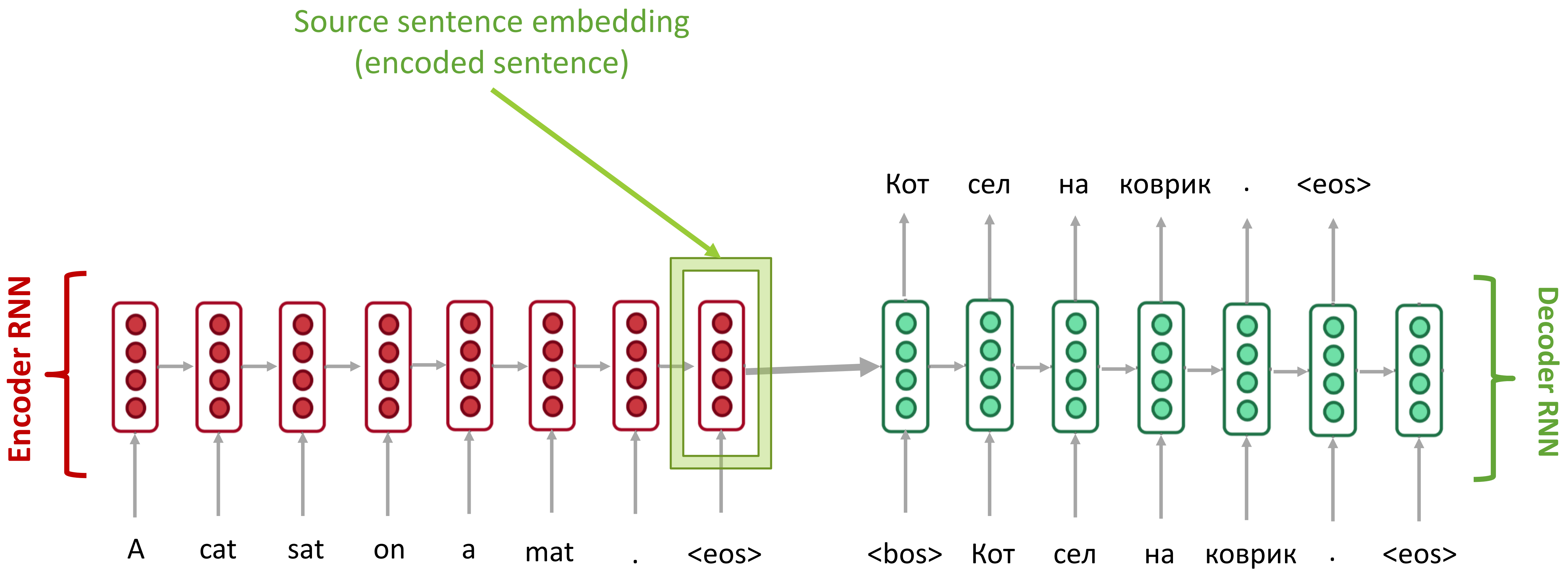
RNN Encoder-Decoder (Vanilla)



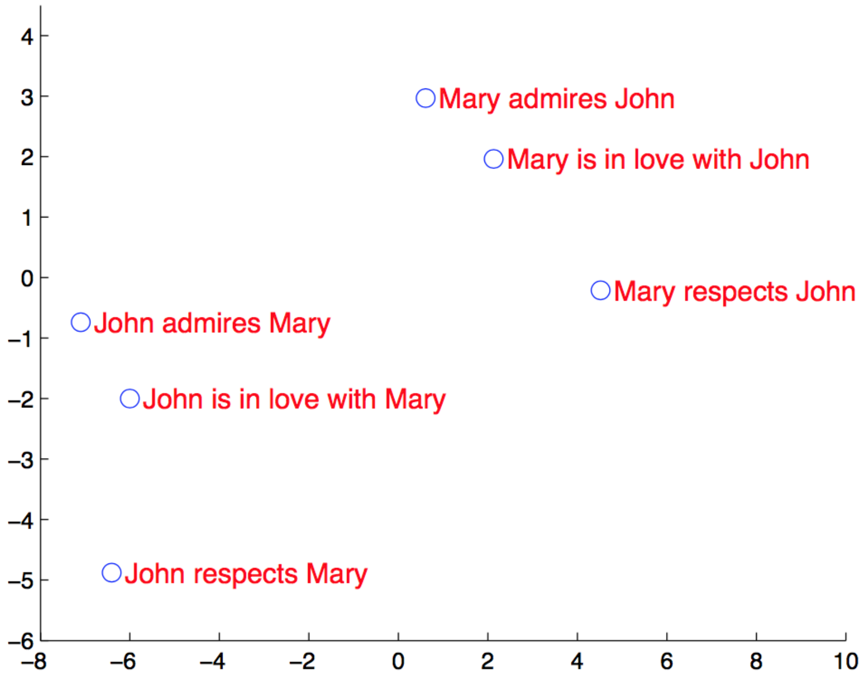
RNN Encoder-Decoder (Vanilla)

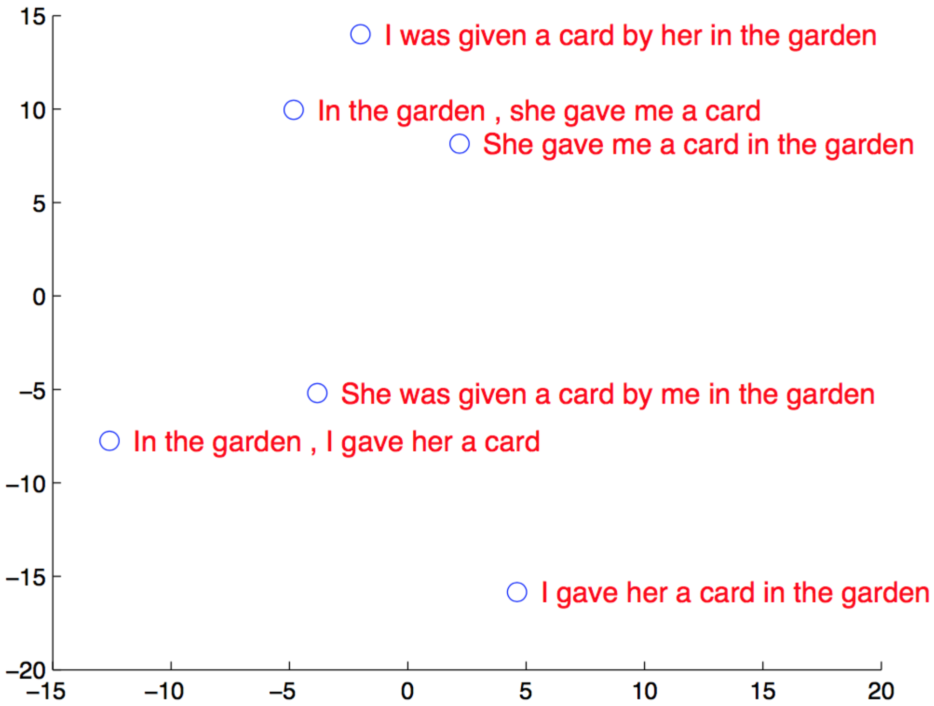


Let’s look at sentence embedding



Let's look at sentence embedding

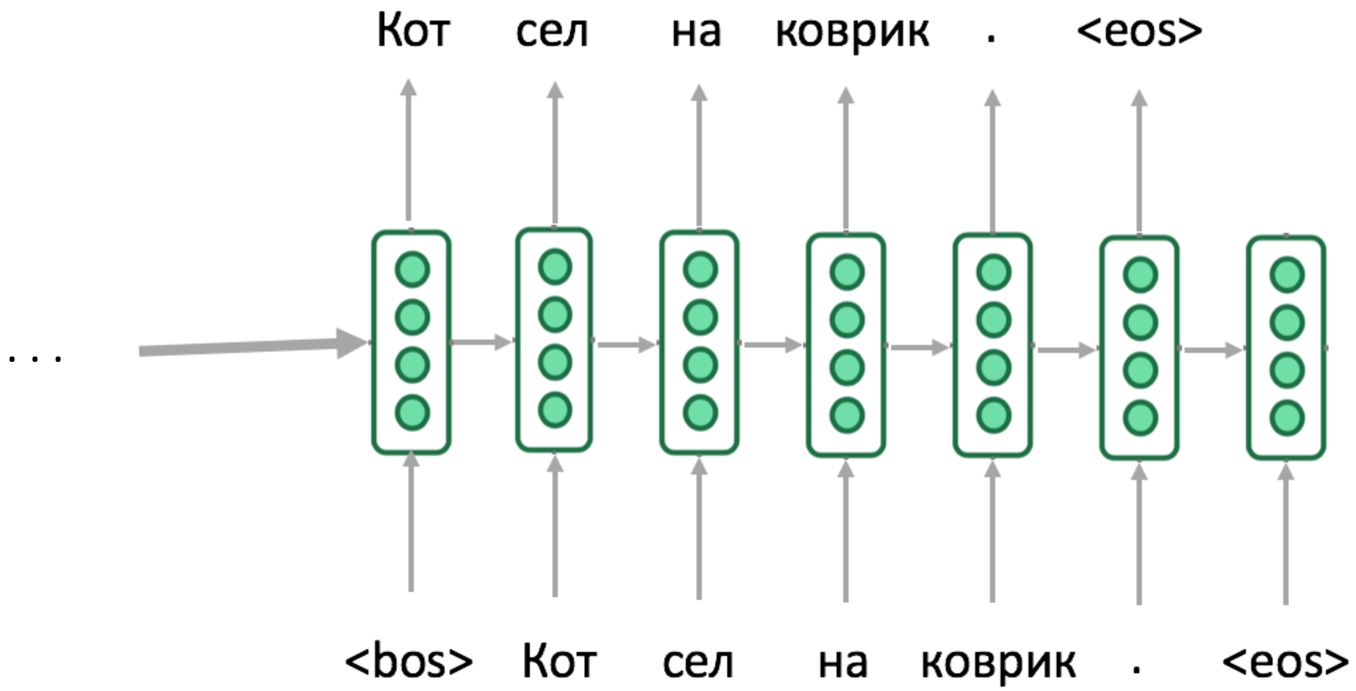




Sutskever et al, NIPS 2014, https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

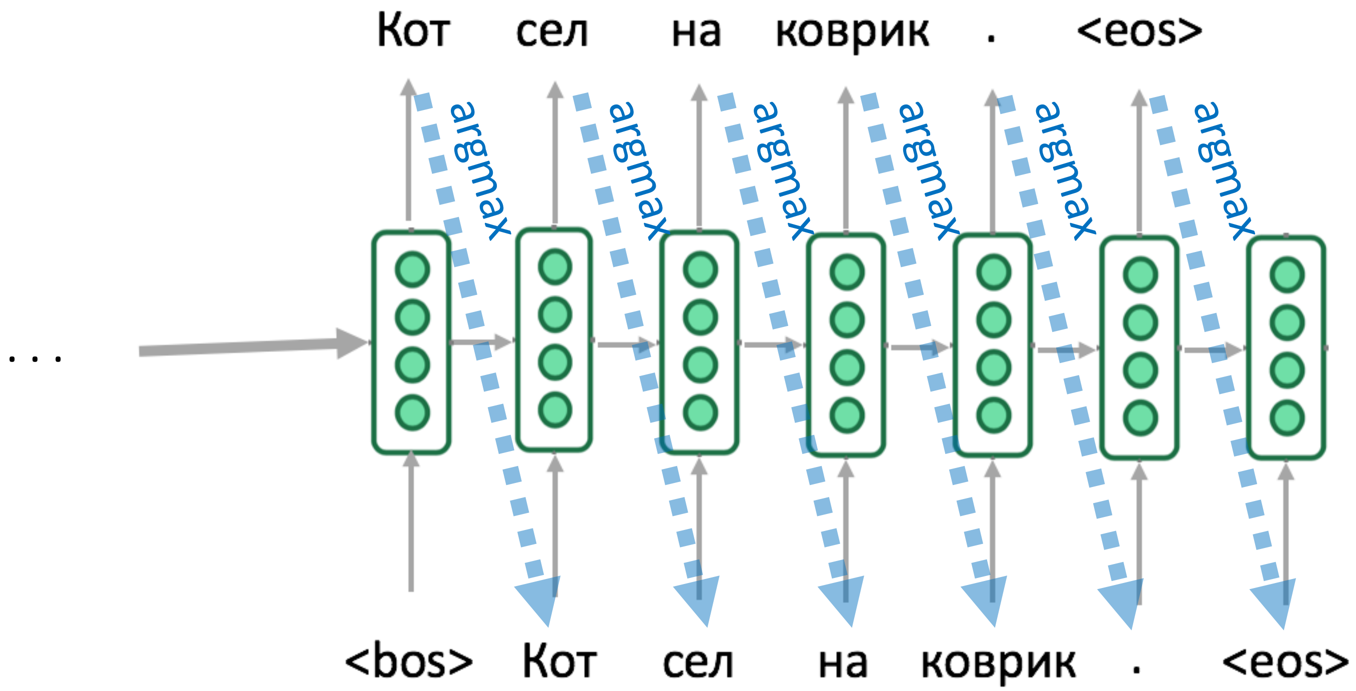
Decoding

Decoding: how to generate a sequence?



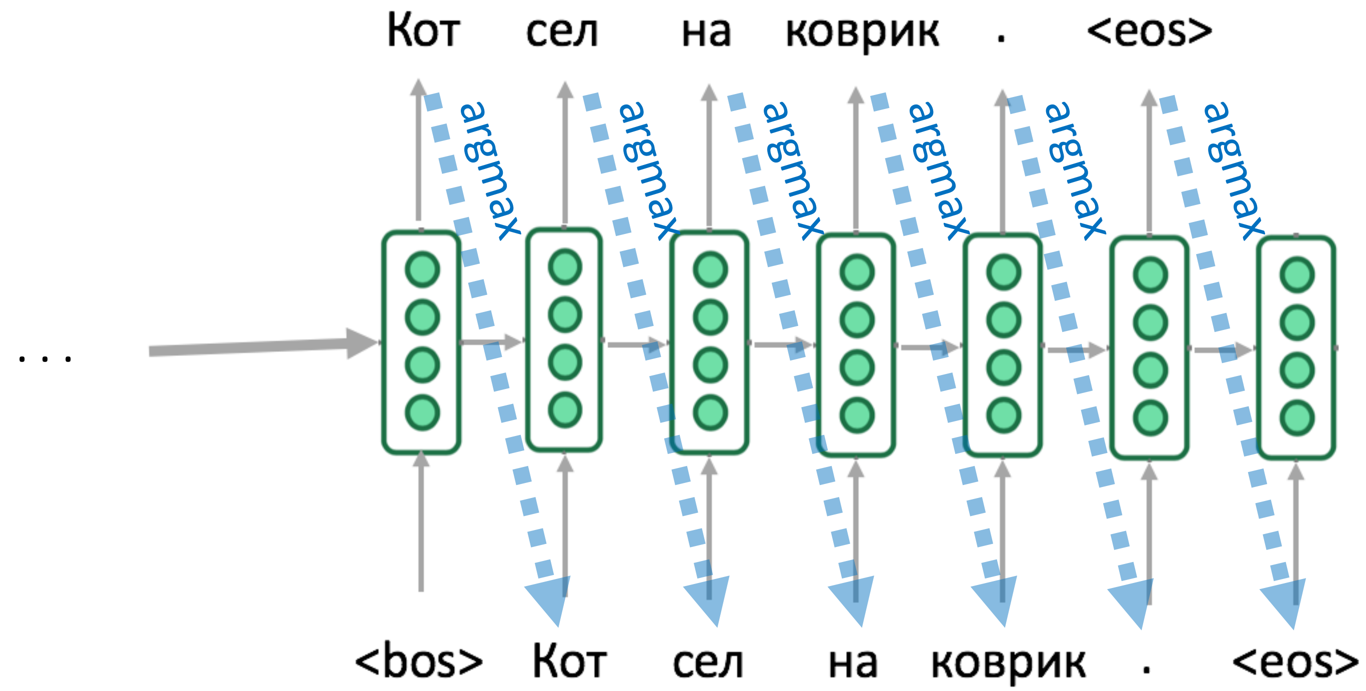
Decoding: how to generate a sequence?

1. Greedy decoding

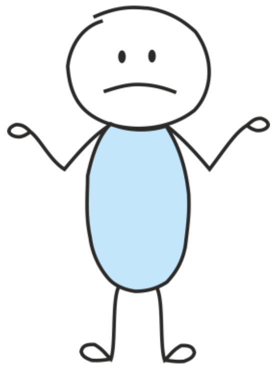


Decoding: how to generate a sequence?

1. Greedy decoding



Is it really good?



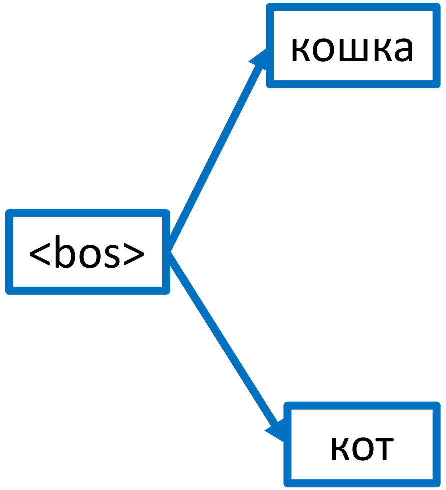
Decoding: how to generate a sequence?

2. Beam search

<bos>

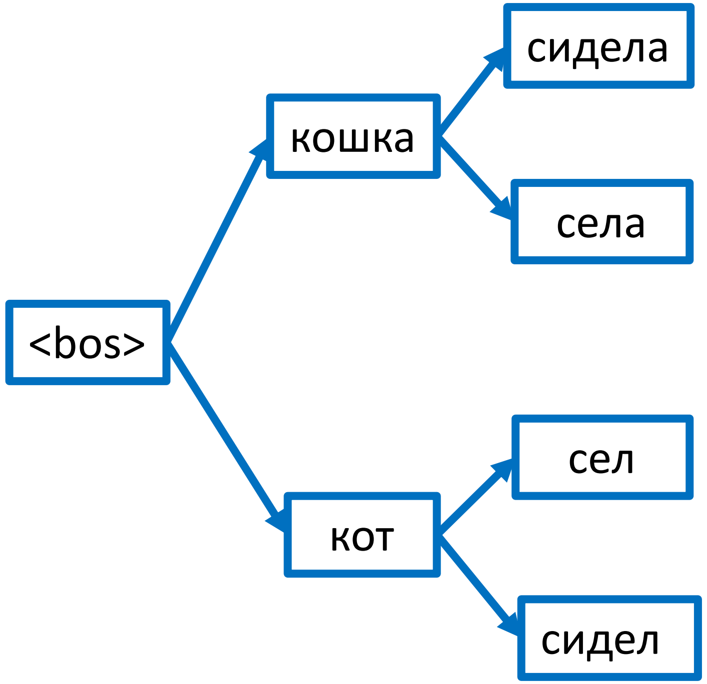
Decoding: how to generate a sequence?

2. Beam search



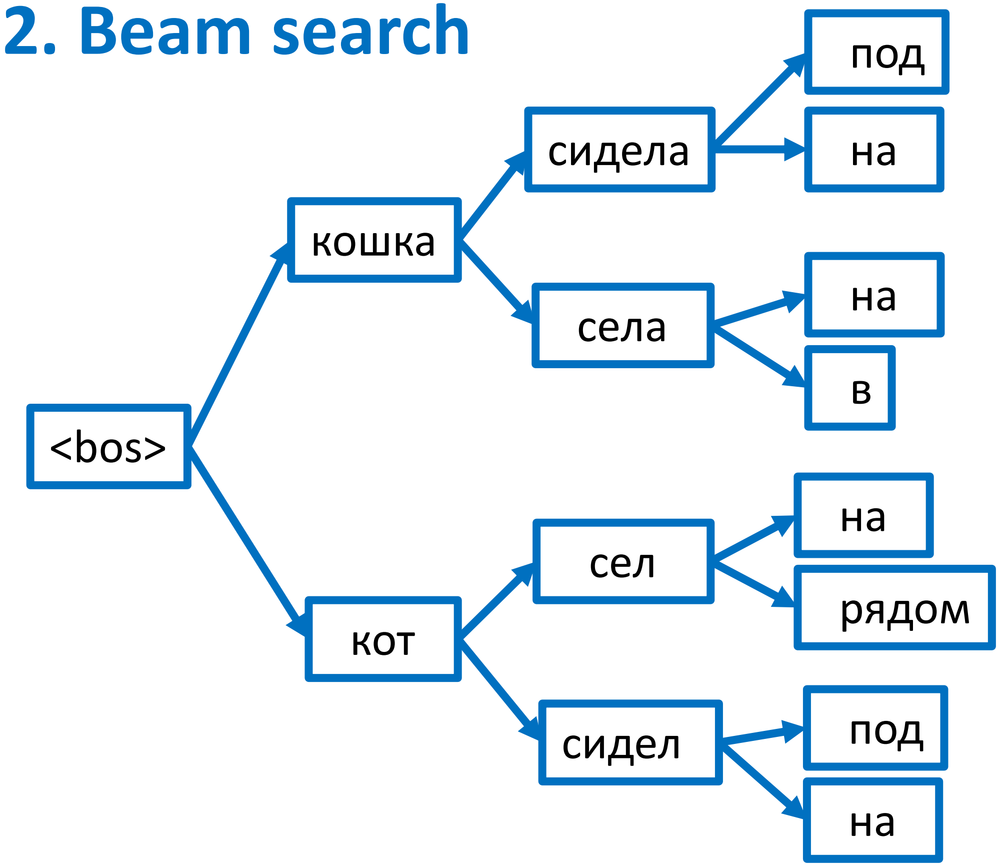
Decoding: how to generate a sequence?

2. Beam search



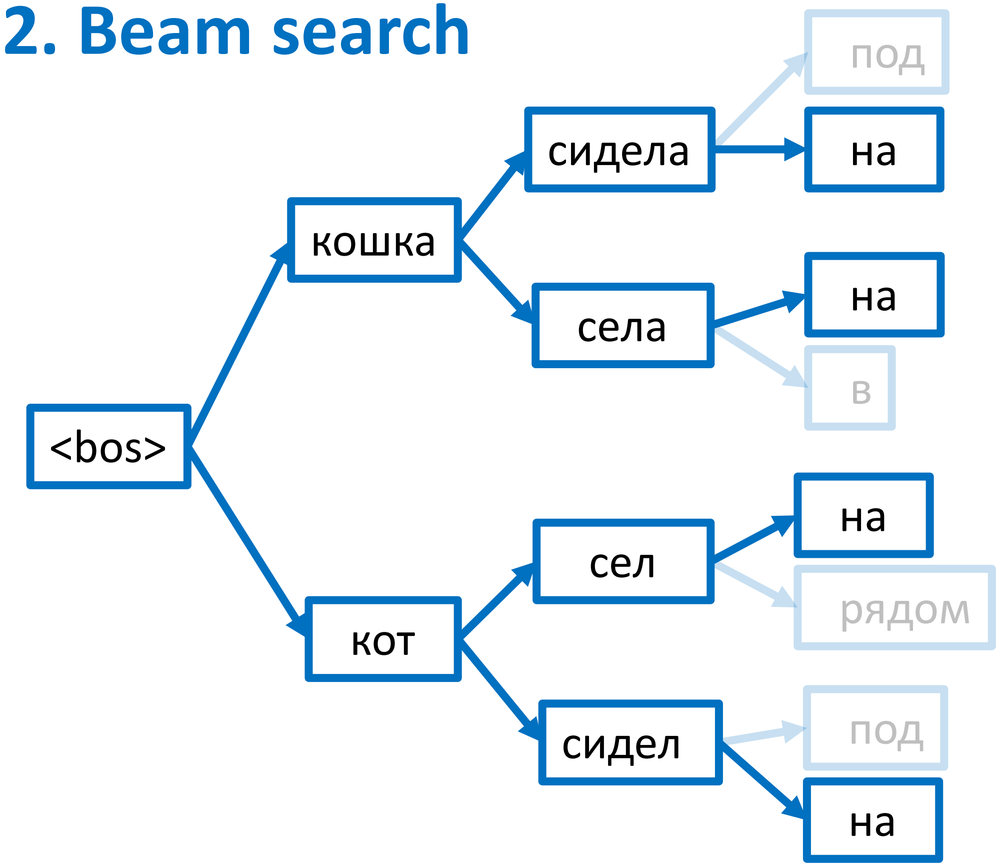
Decoding: how to generate a sequence?

2. Beam search



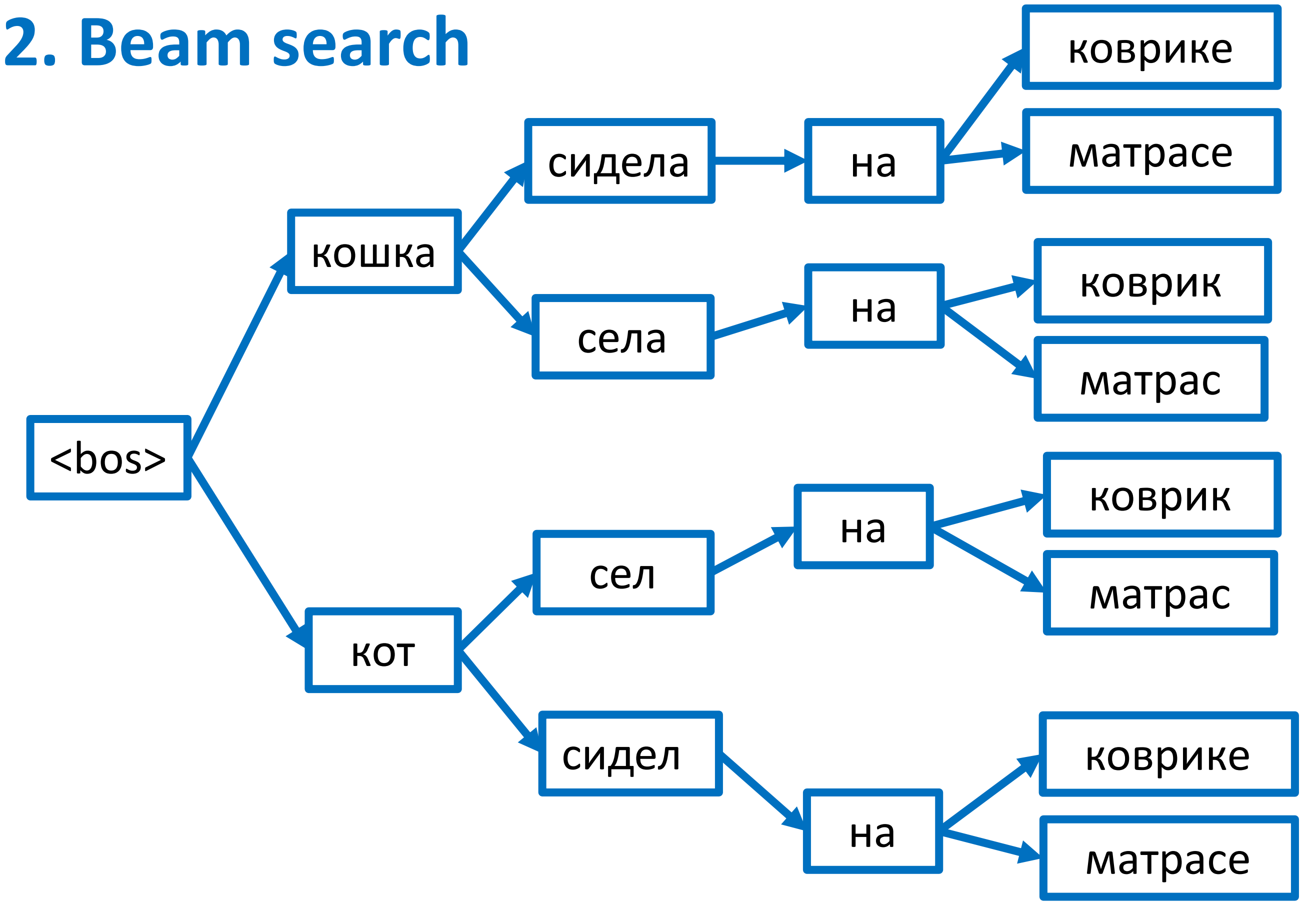
Decoding: how to generate a sequence?

2. Beam search



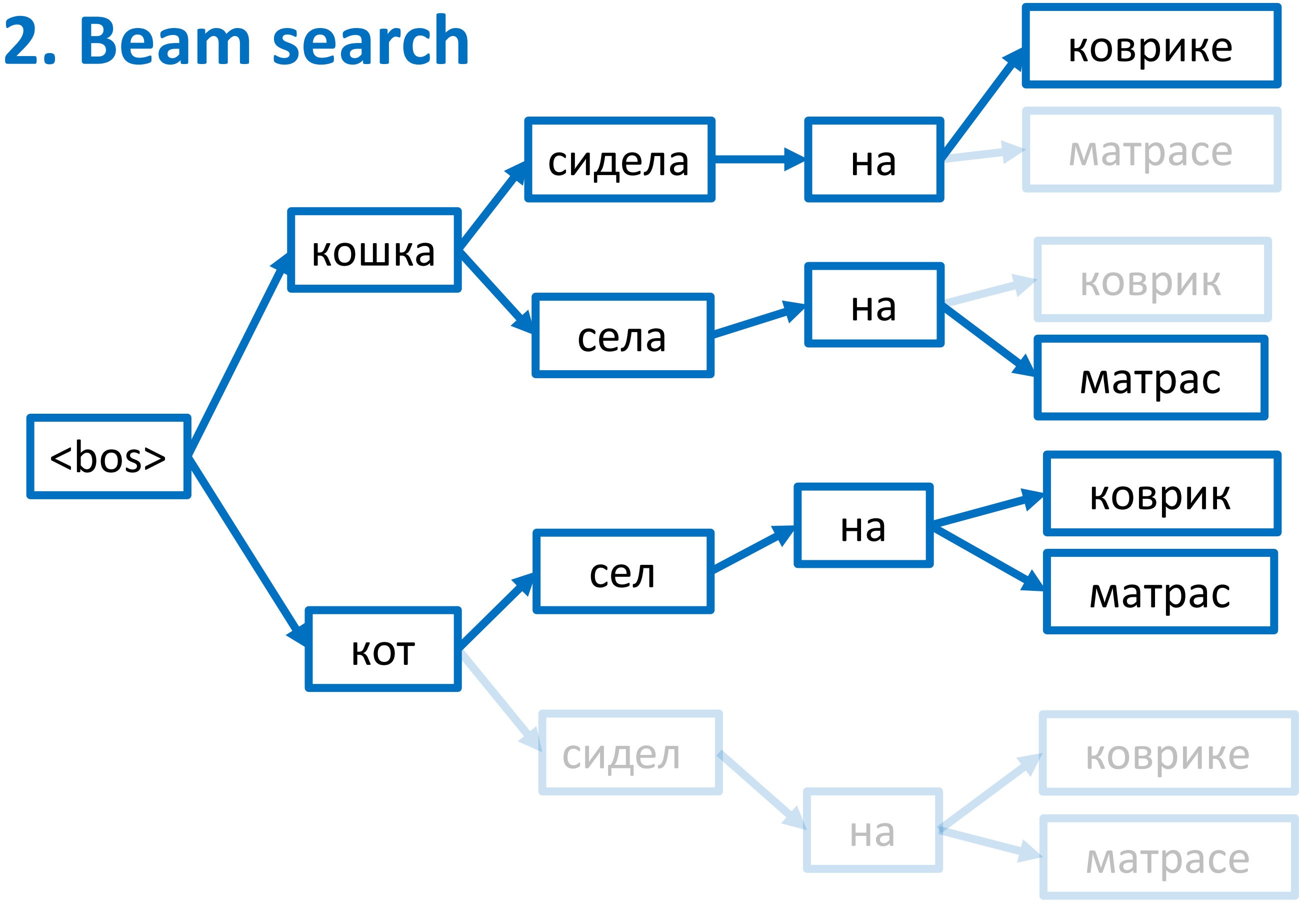
Decoding: how to generate a sequence?

2. Beam search



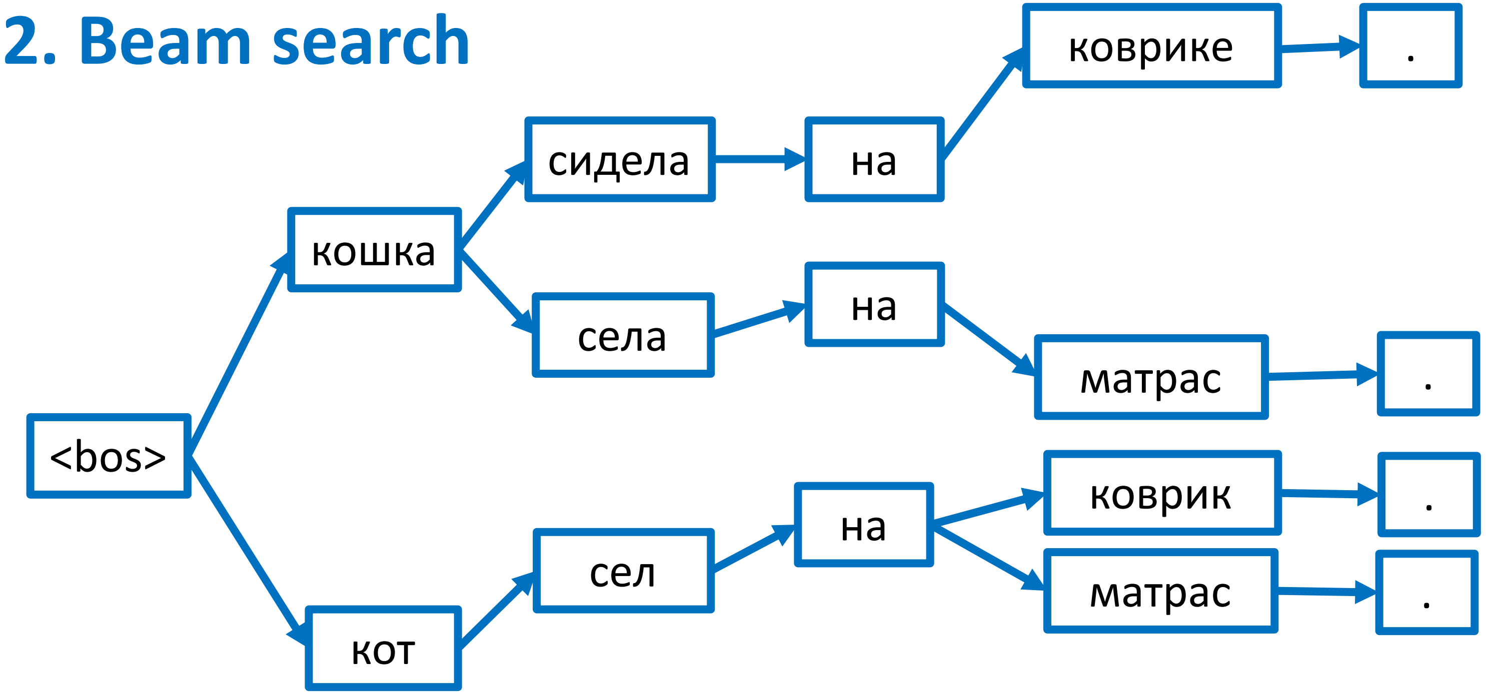
Decoding: how to generate a sequence?

2. Beam search



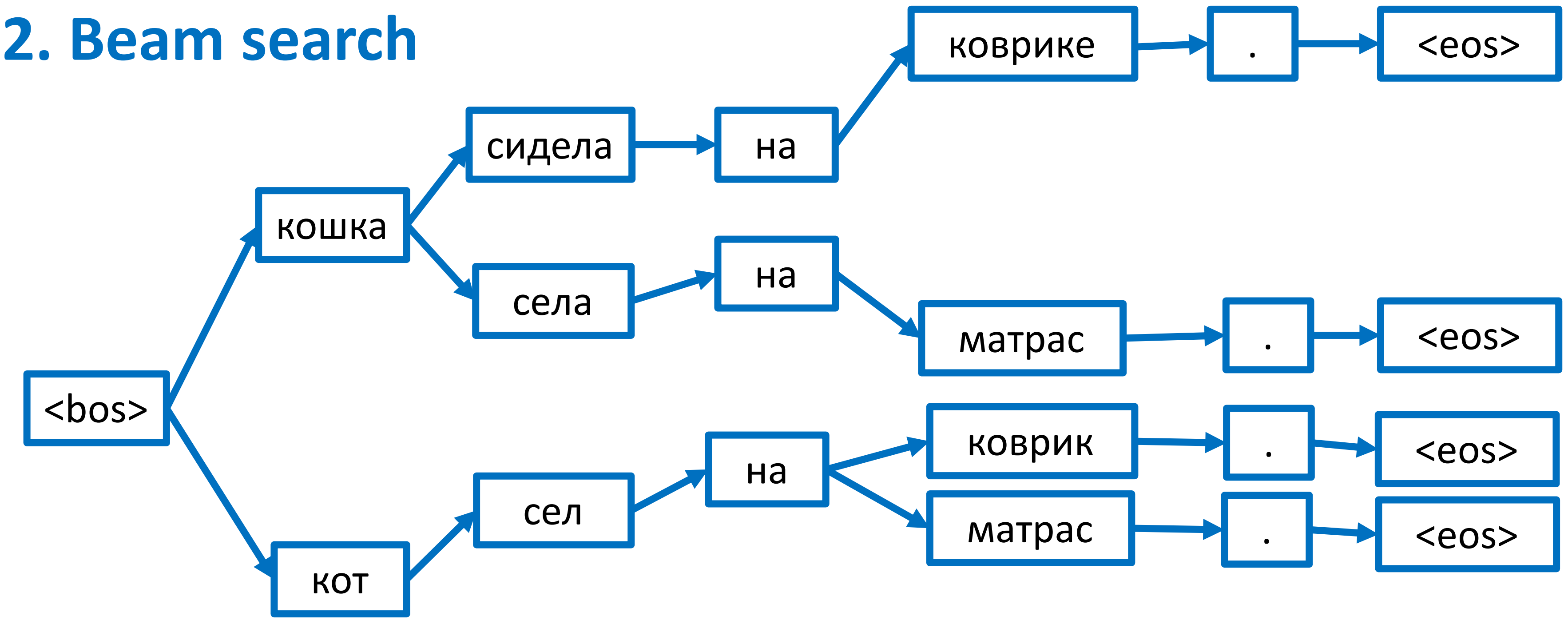
Decoding: how to generate a sequence?

2. Beam search



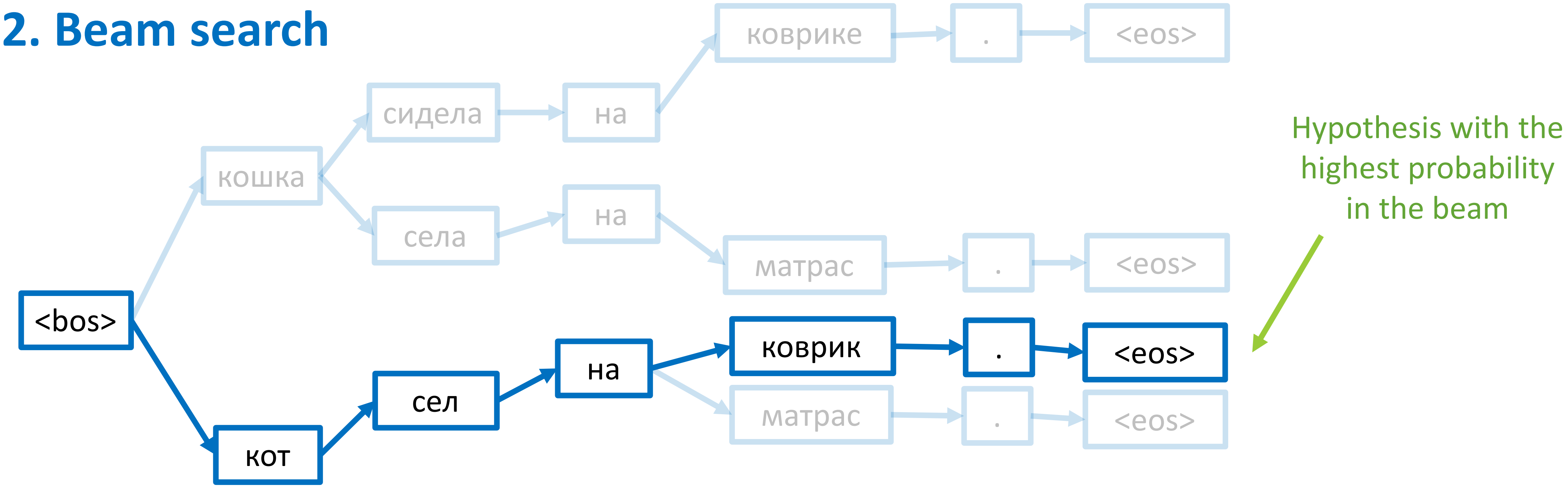
Decoding: how to generate a sequence?

2. Beam search



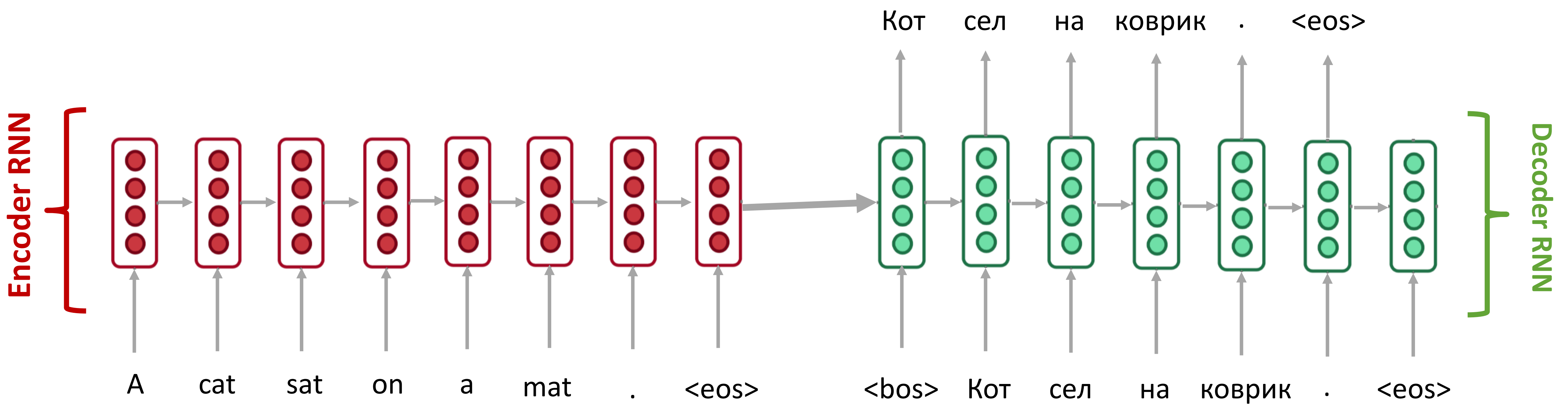
Decoding: how to generate a sequence?

2. Beam search

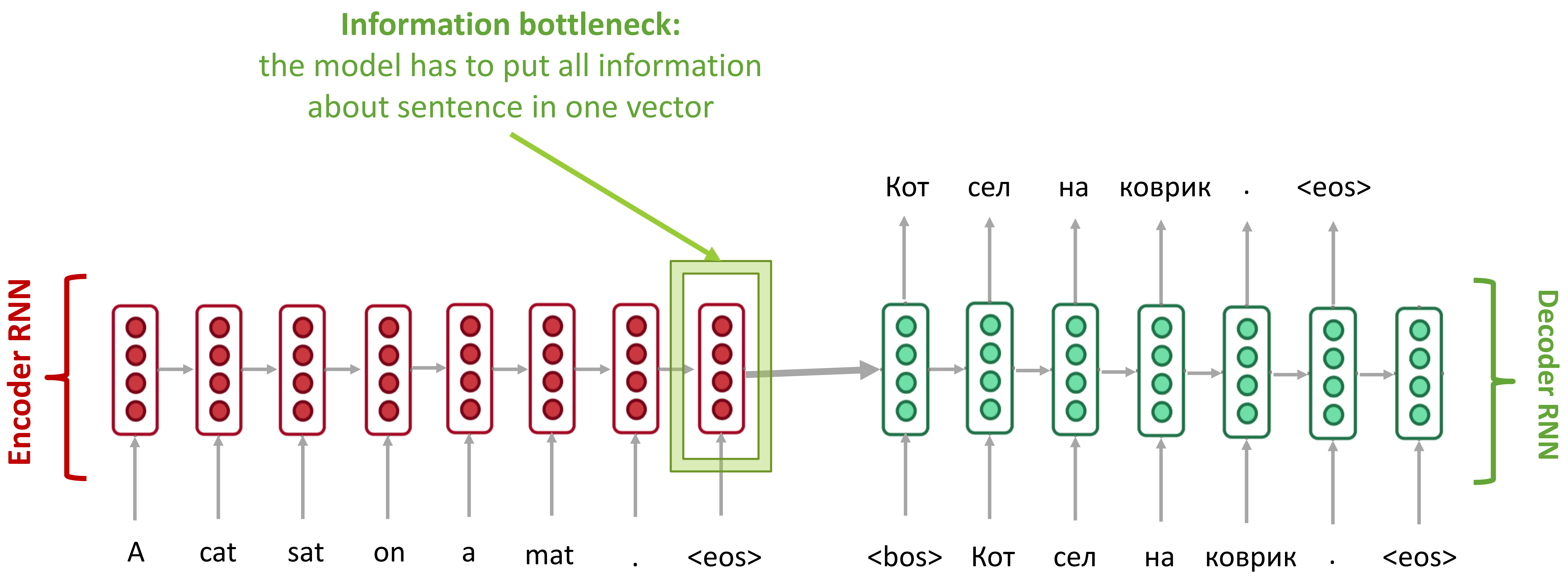


Attention-based models

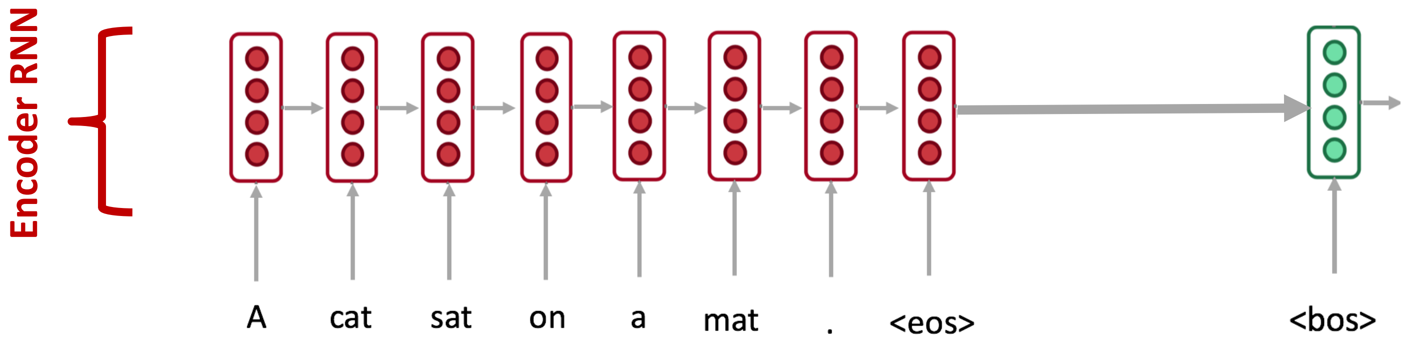
What is the problem with such models?



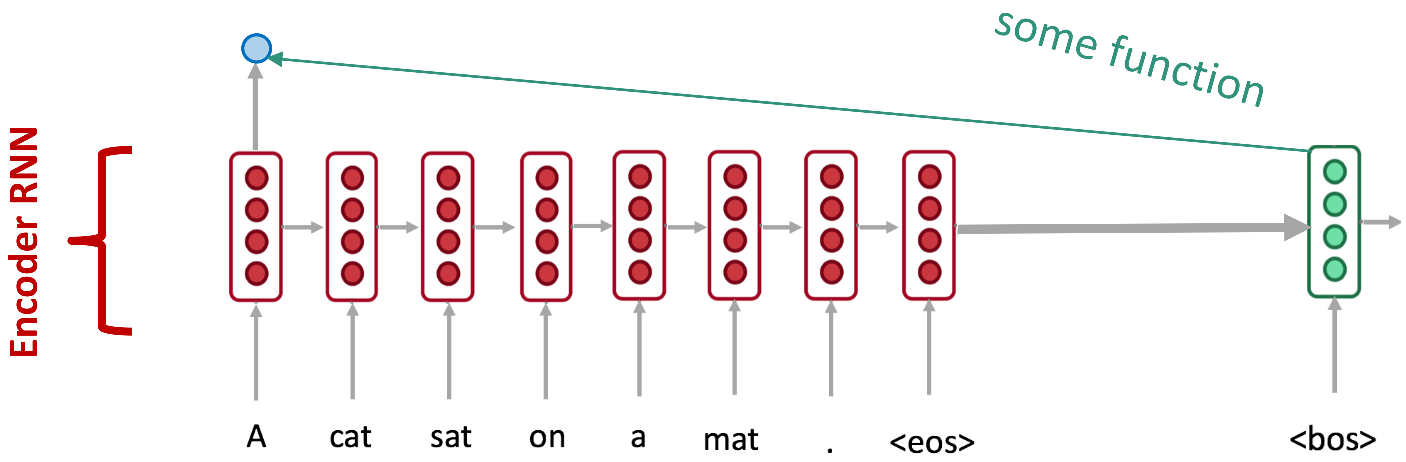
What is the problem with such models?



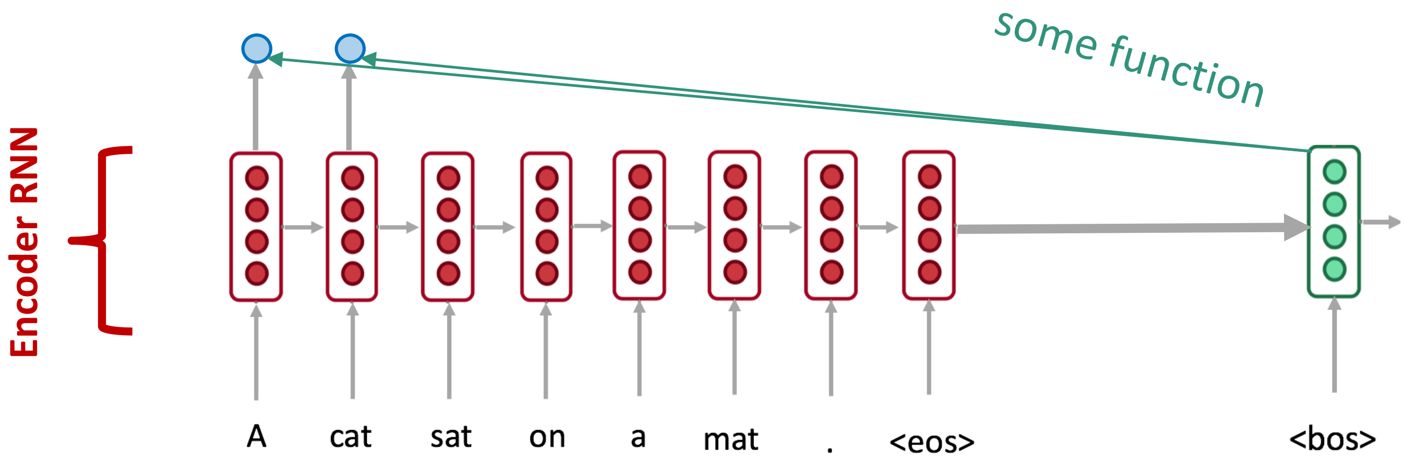
Attention



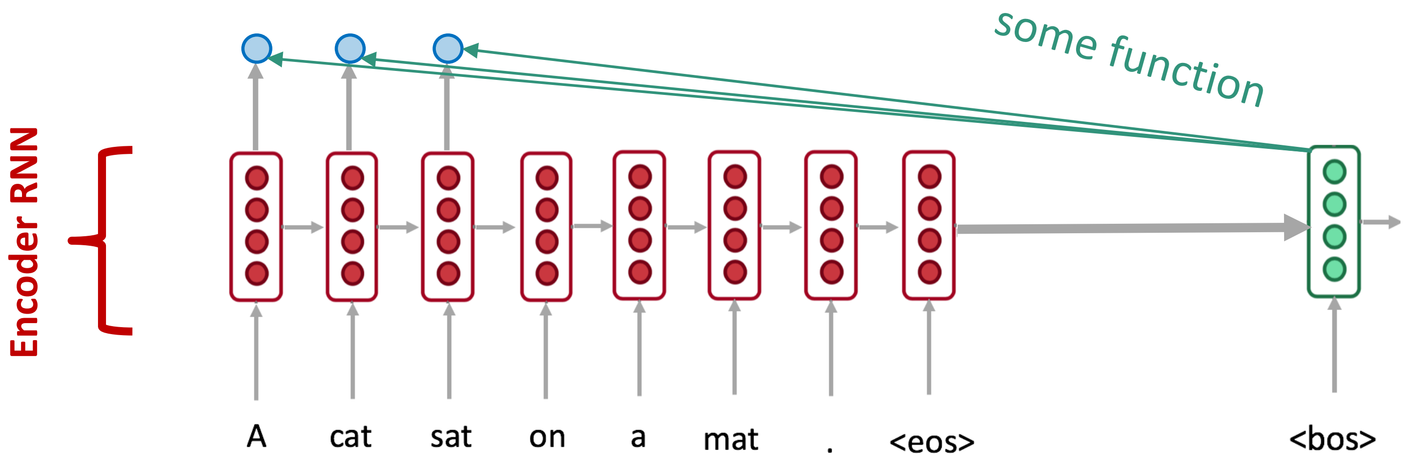
Attention



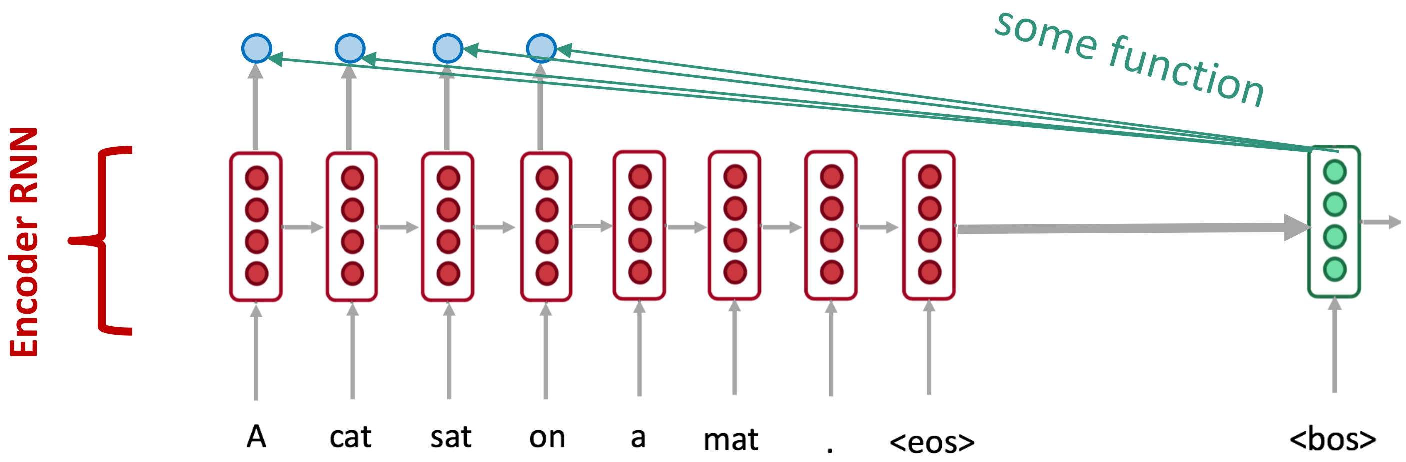
Attention



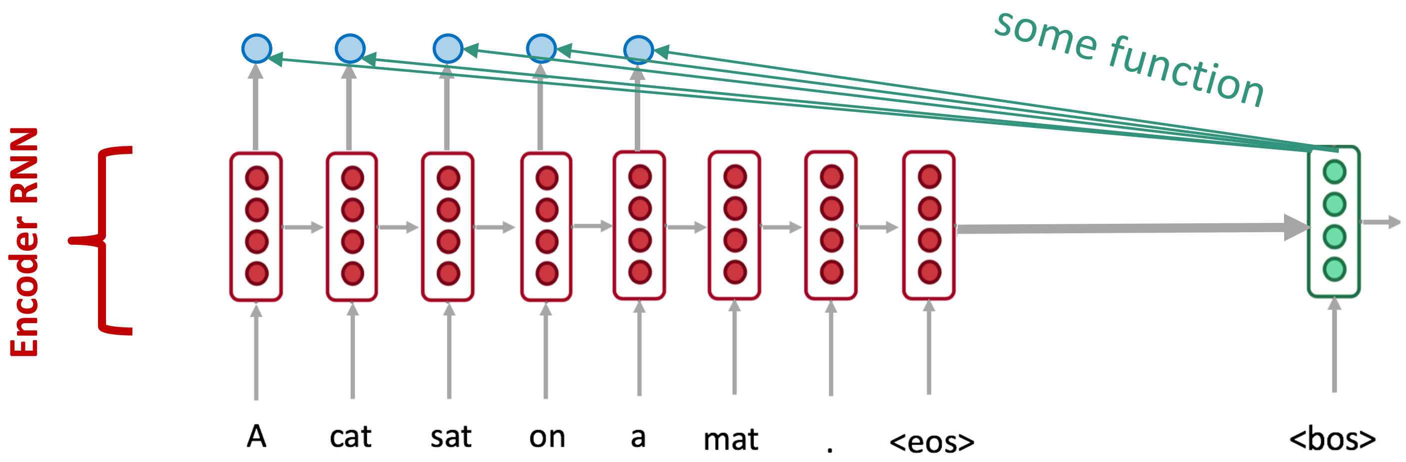
Attention



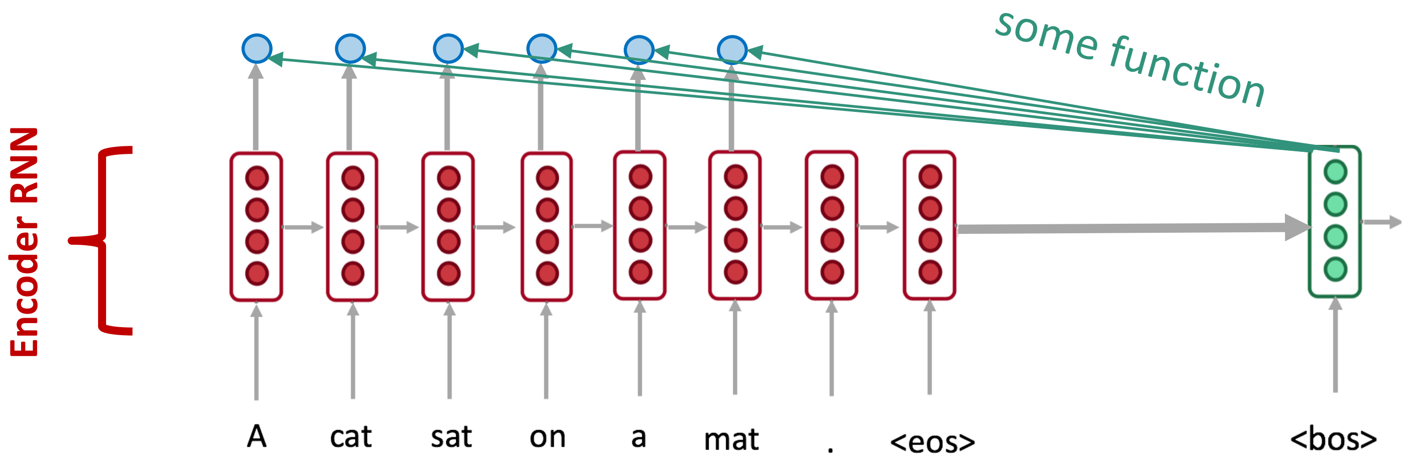
Attention



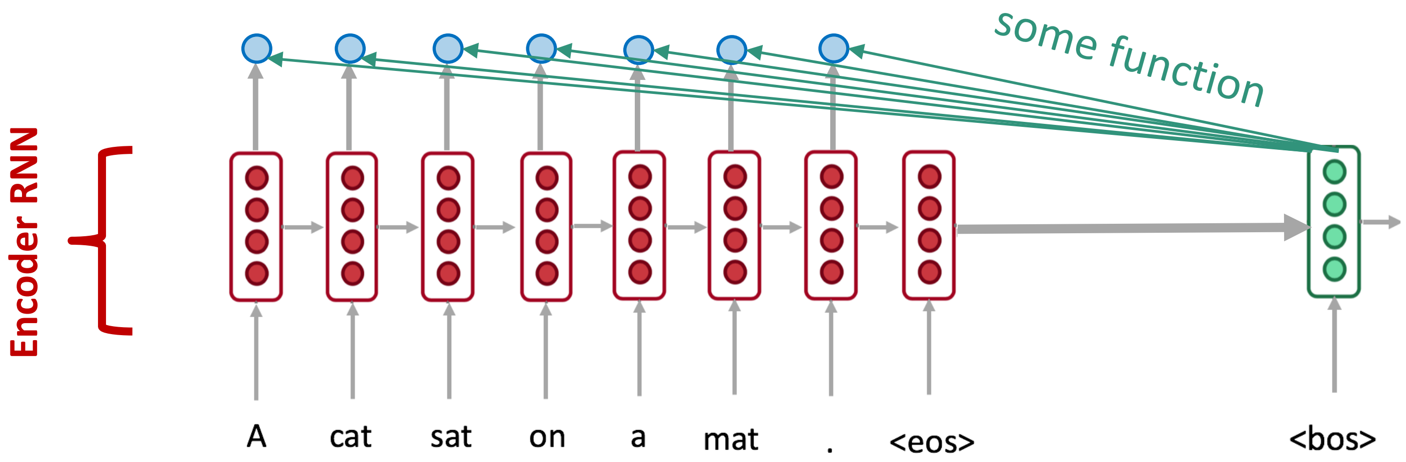
Attention



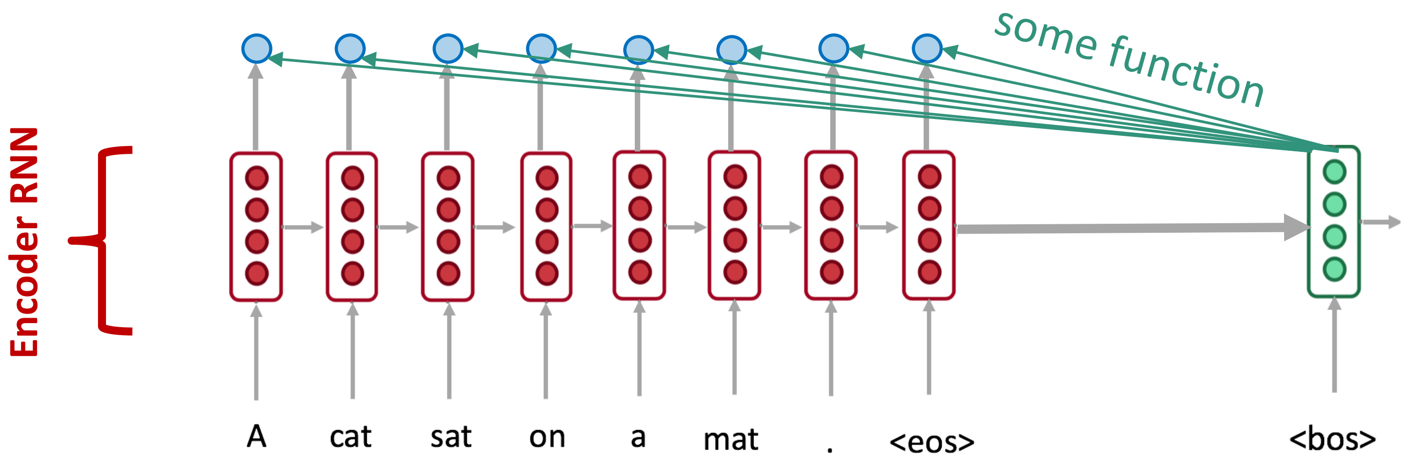
Attention



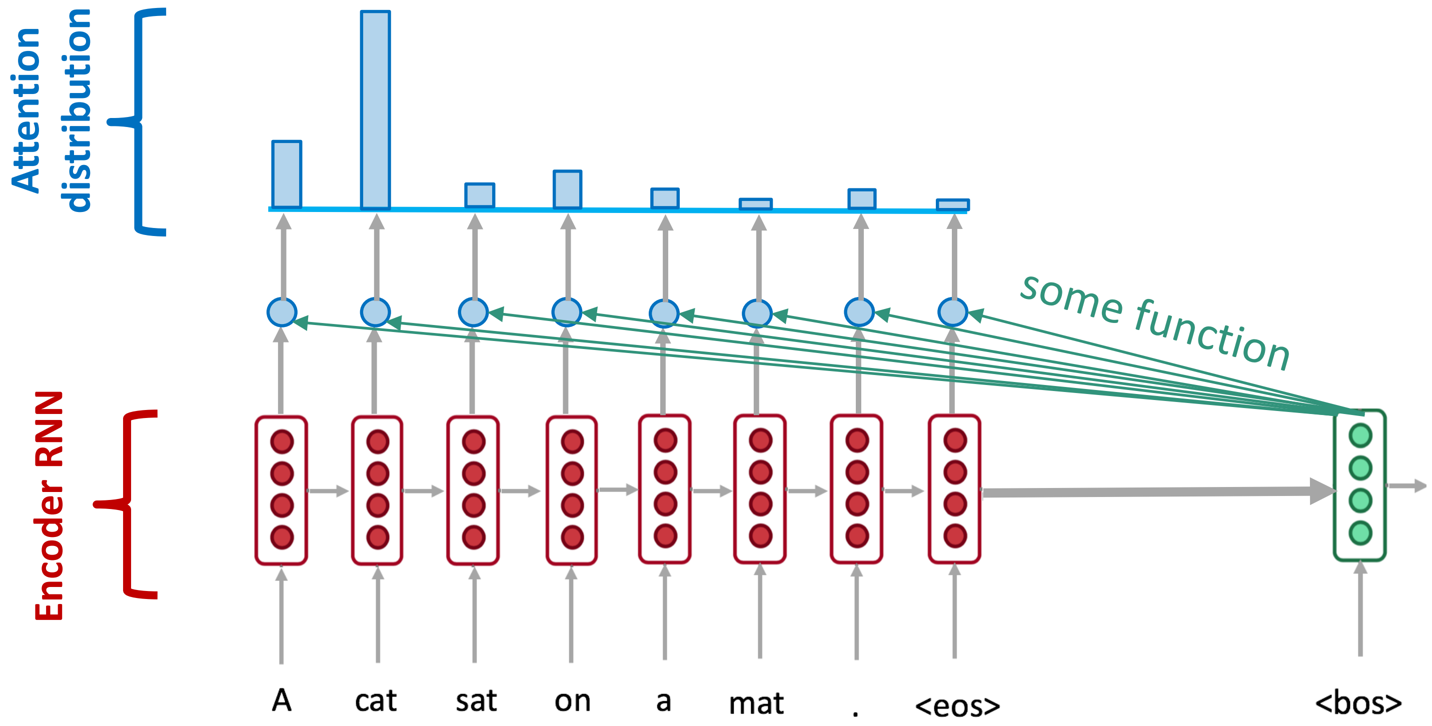
Attention



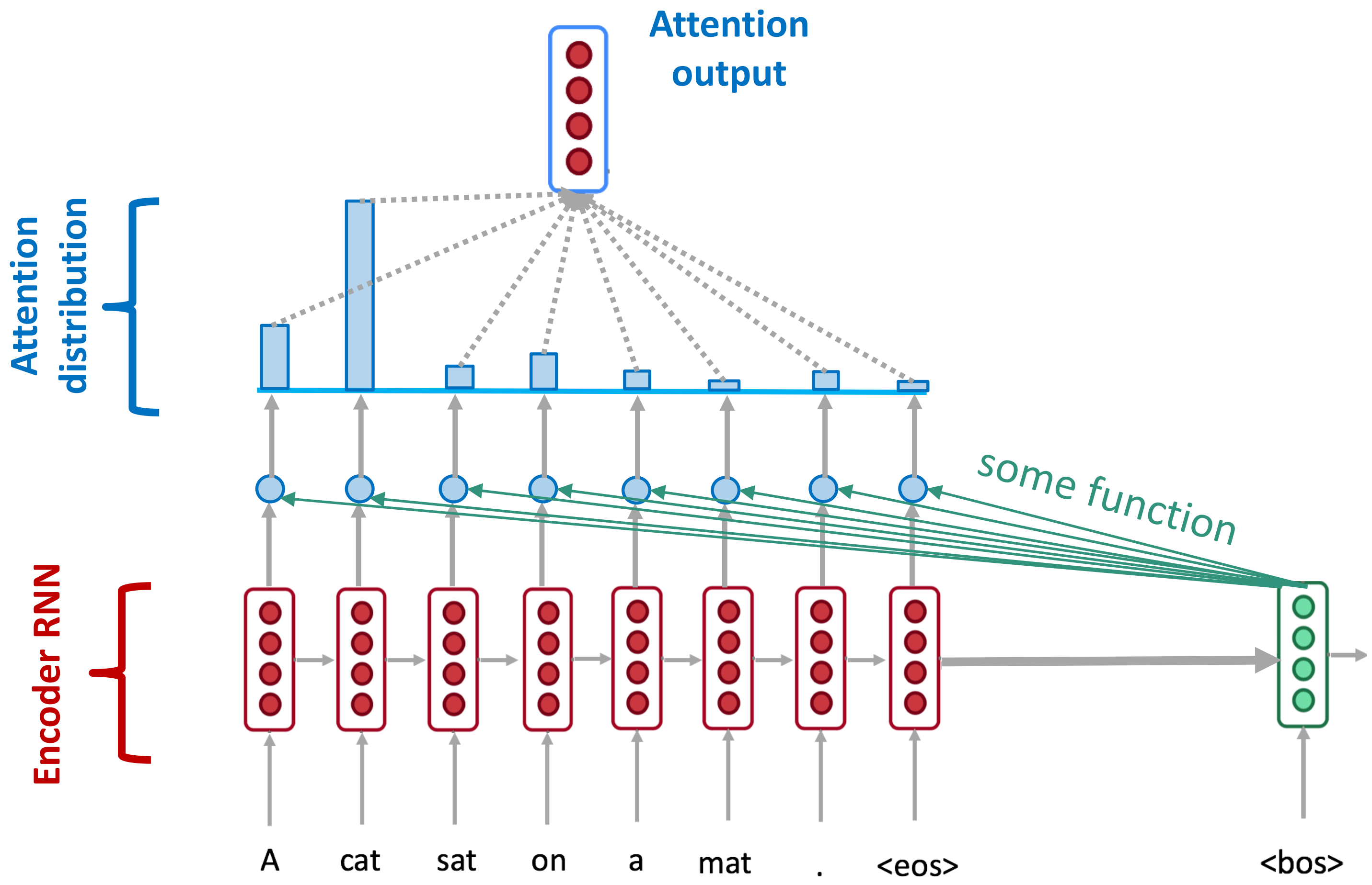
Attention



Attention

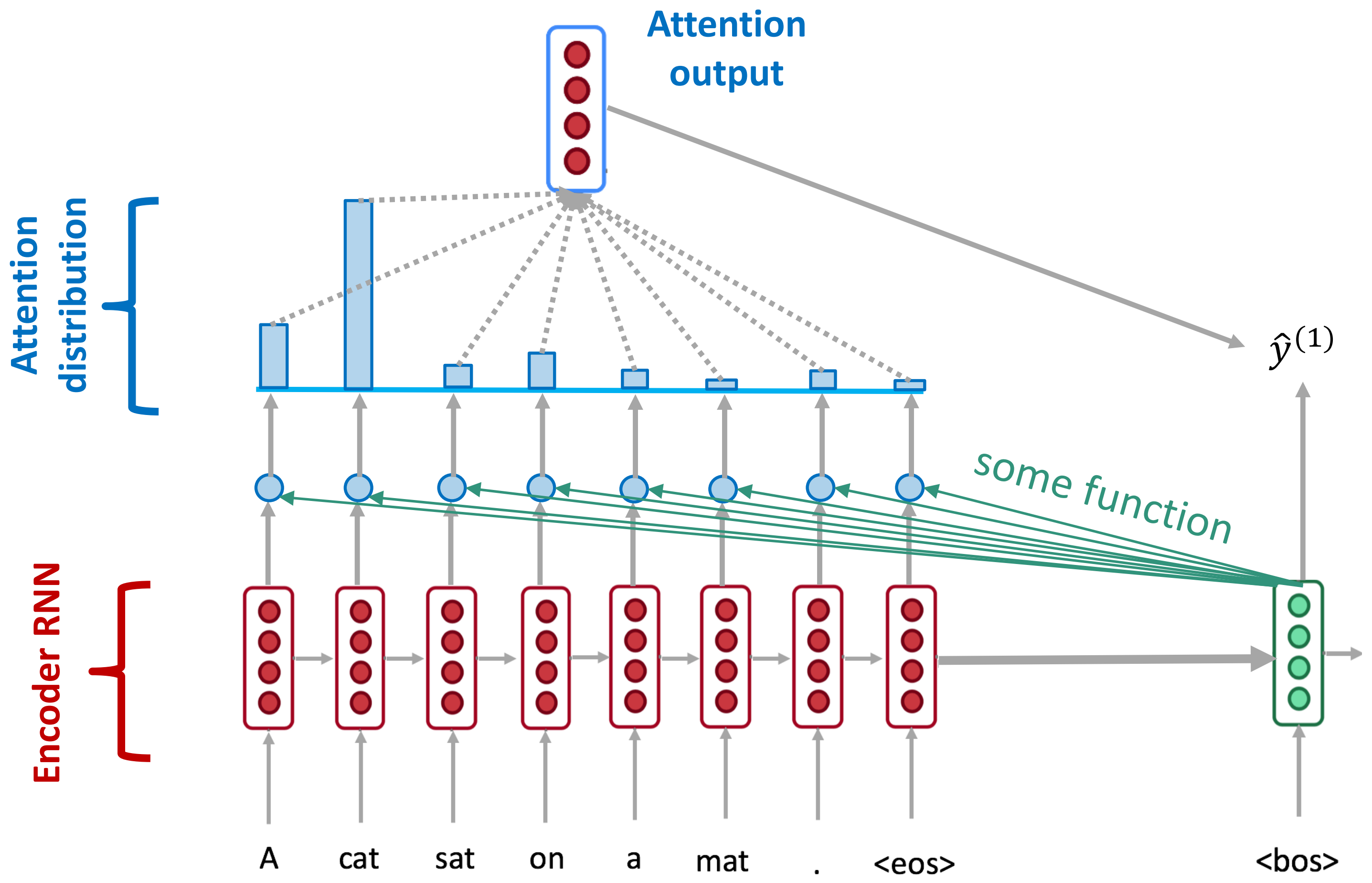

Attention distribution
some function
Encoder RNN
A
cat
sat
on
a
mat
.
<eos>
<bos>

Attention



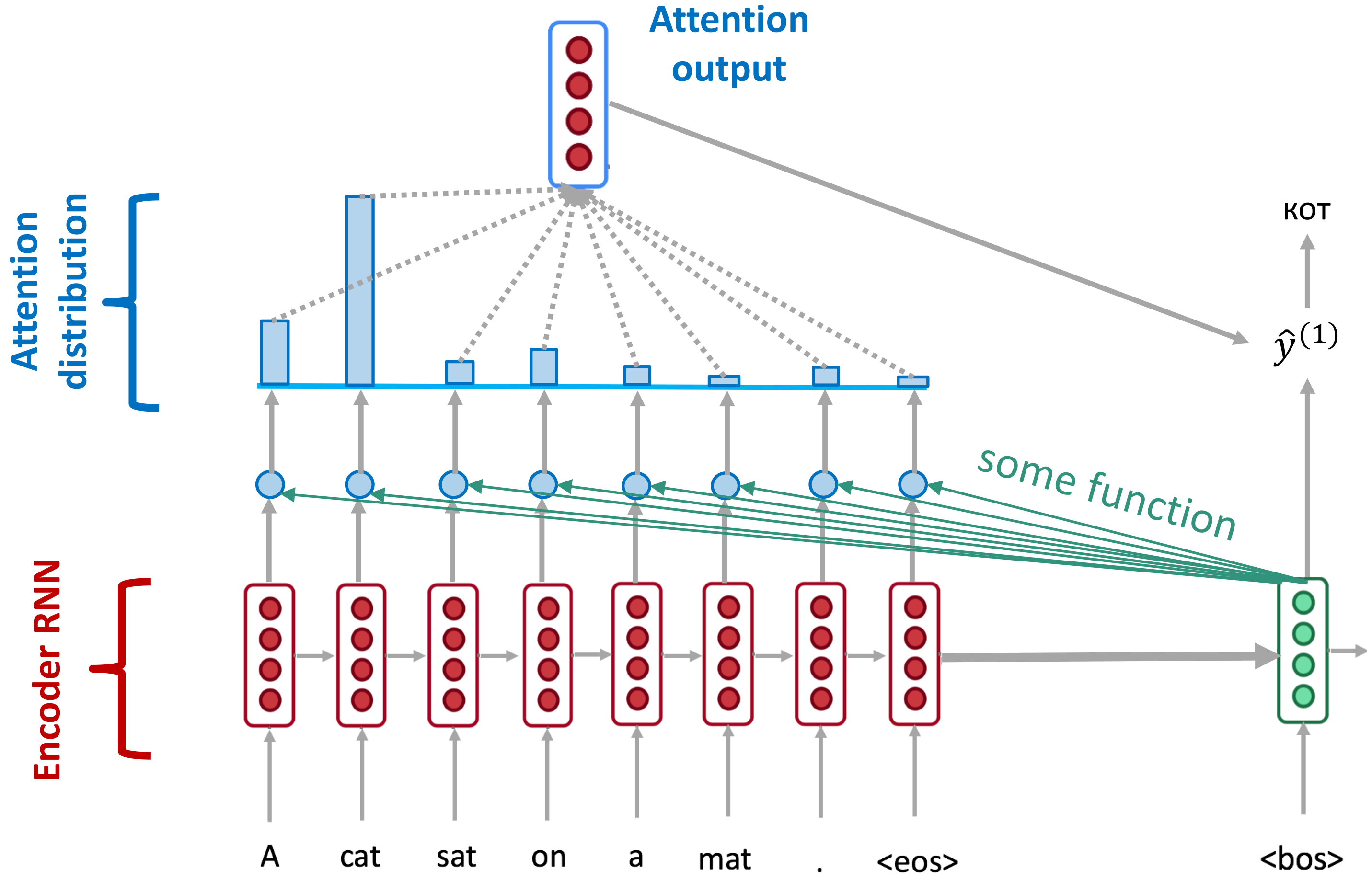
Attention output
Attention distribution
some function
Encoder RNN
A
cat
sat
on
a
mat
.
<eos>
<bos>

Attention



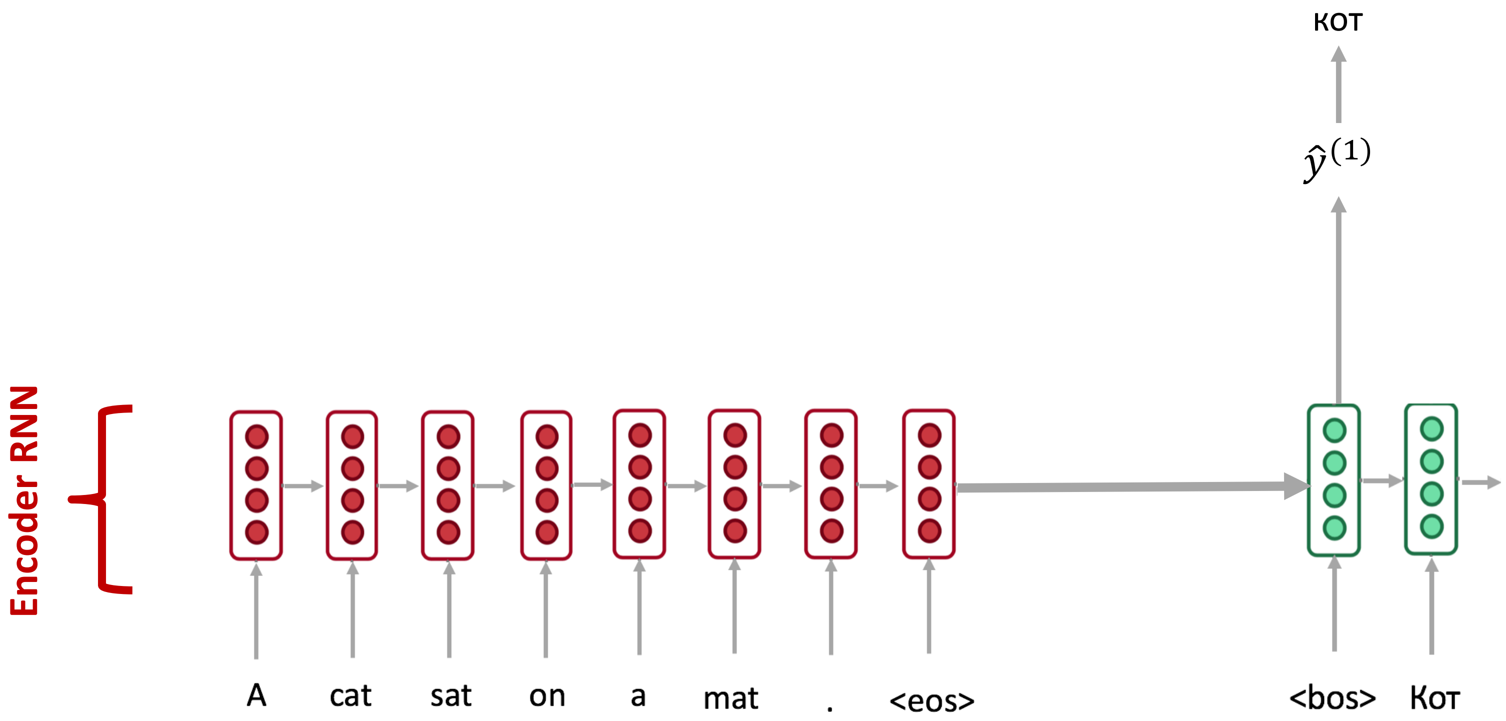
Attention output
Attention distribution
some function
$\hat{y}^{(1)}$
Encoder RNN
A
cat
sat
on
a
mat
.
<eos>
<bos>

Attention

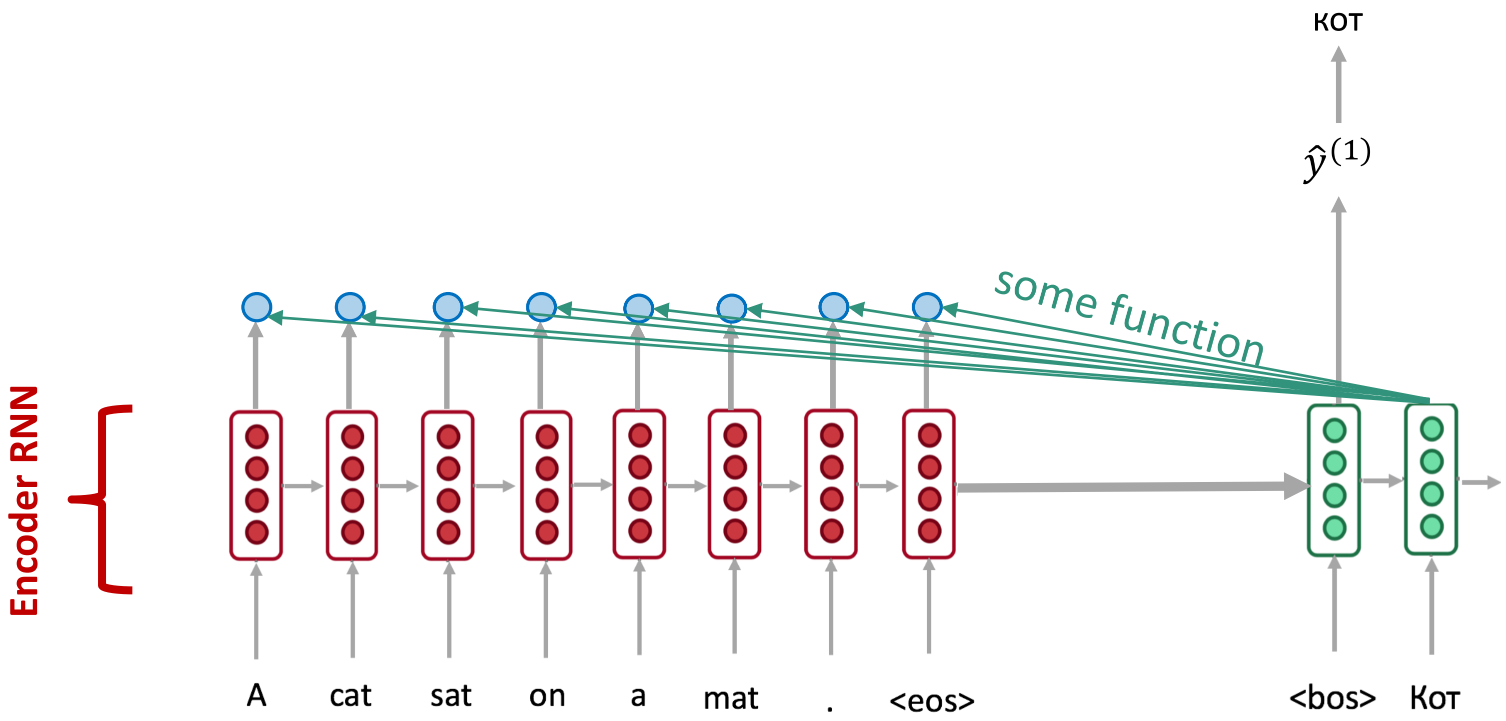


Attention output
Attention distribution
кот
$\hat{y}^{(1)}$
some function
Encoder RNN
A
cat
sat
on
a
mat
.
<eos>
<bos>

Attention

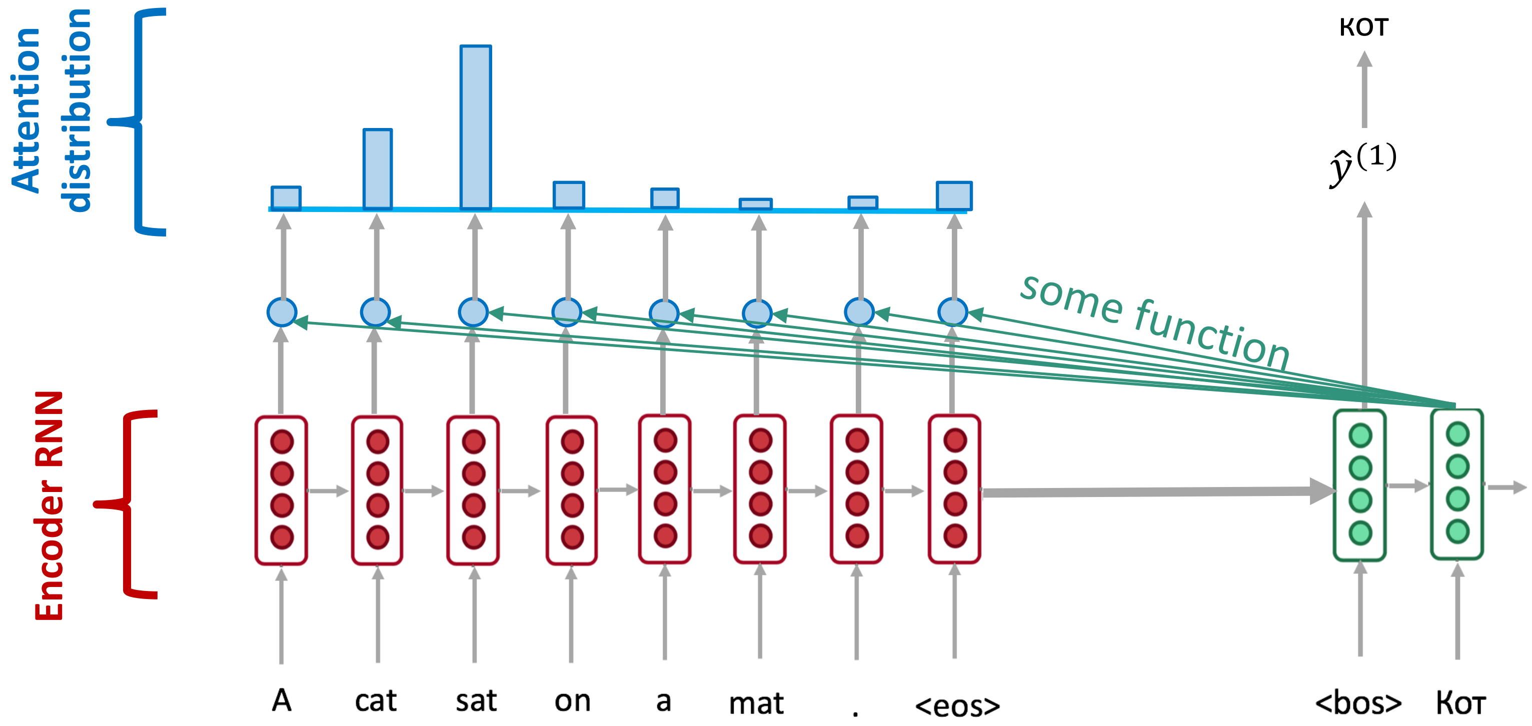


Attention



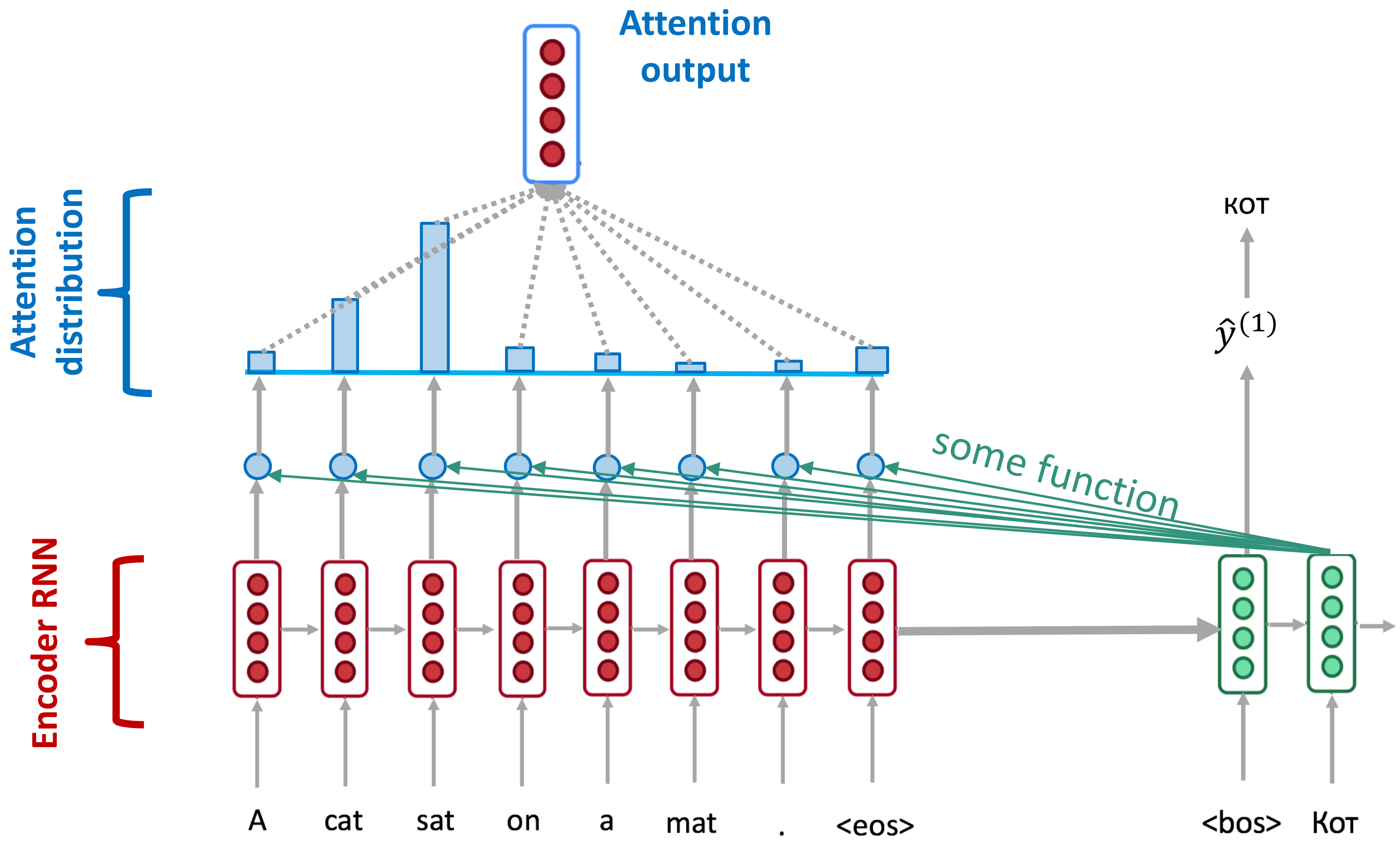
кот
$\hat{y}^{(1)}$
some function
Encoder RNN
A
cat
sat
on
a
mat
.
<eos>
<bos>
Кот

Attention



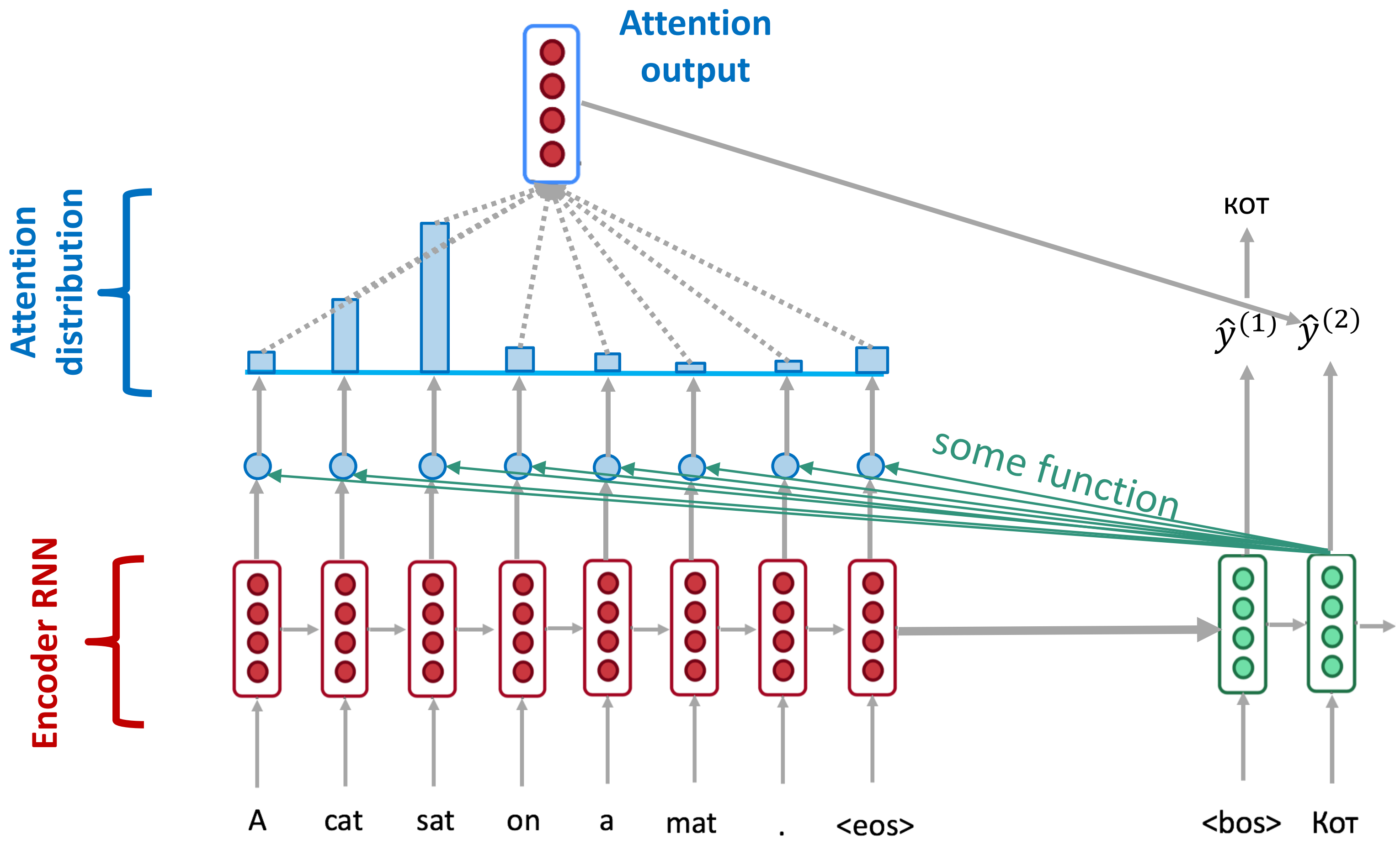
Attention distribution
Encoder RNN
some function
кот
$\hat{y}^{(1)}$
A
cat
sat
on
a
mat
.
<eos>
<bos>
Кот

Attention



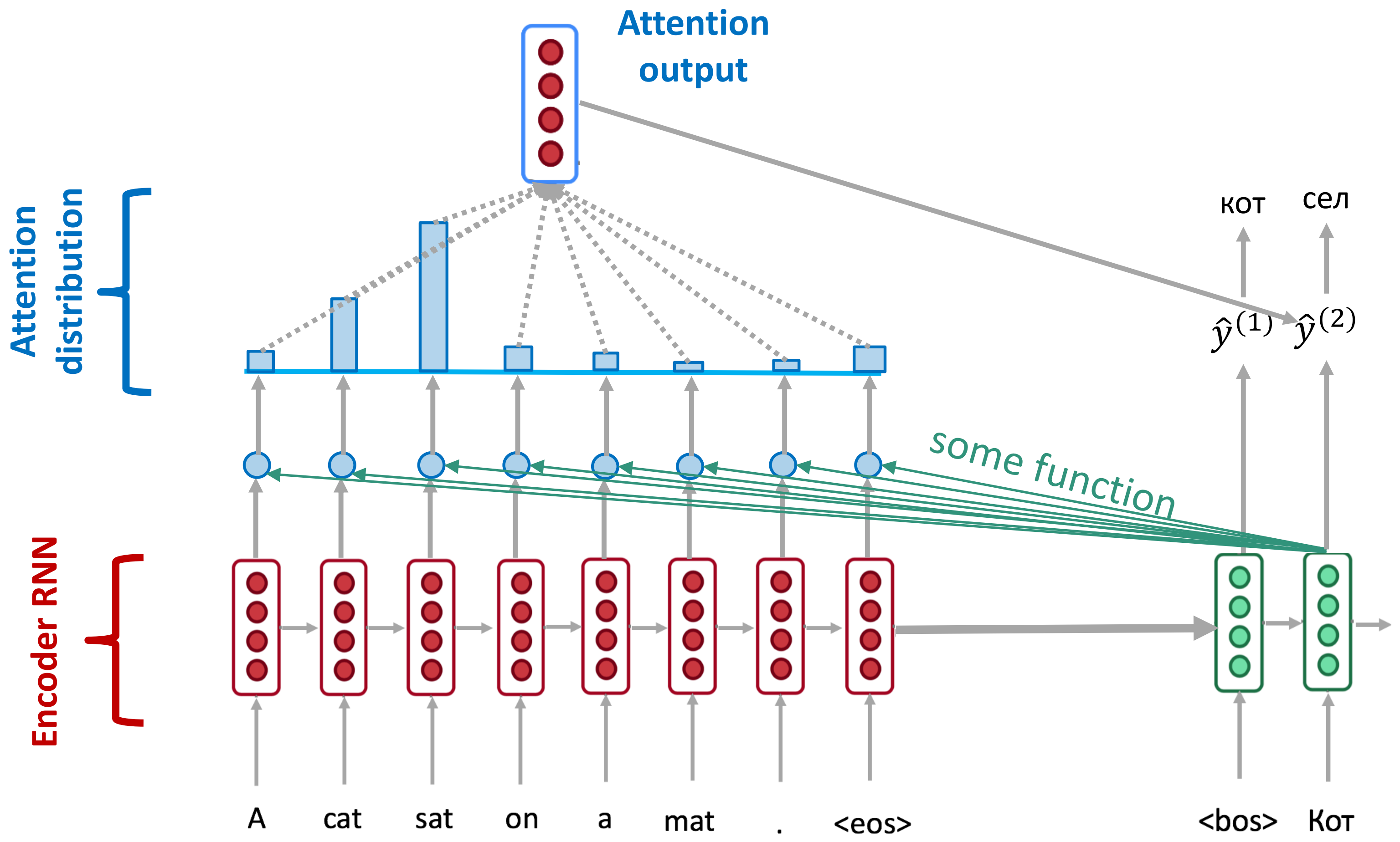
Attention output
Attention distribution
Encoder RNN
some function
кот
$\hat{y}^{(1)}$
A
cat
sat
on
a
mat
.
<eos>
<bos>
Кот

Attention



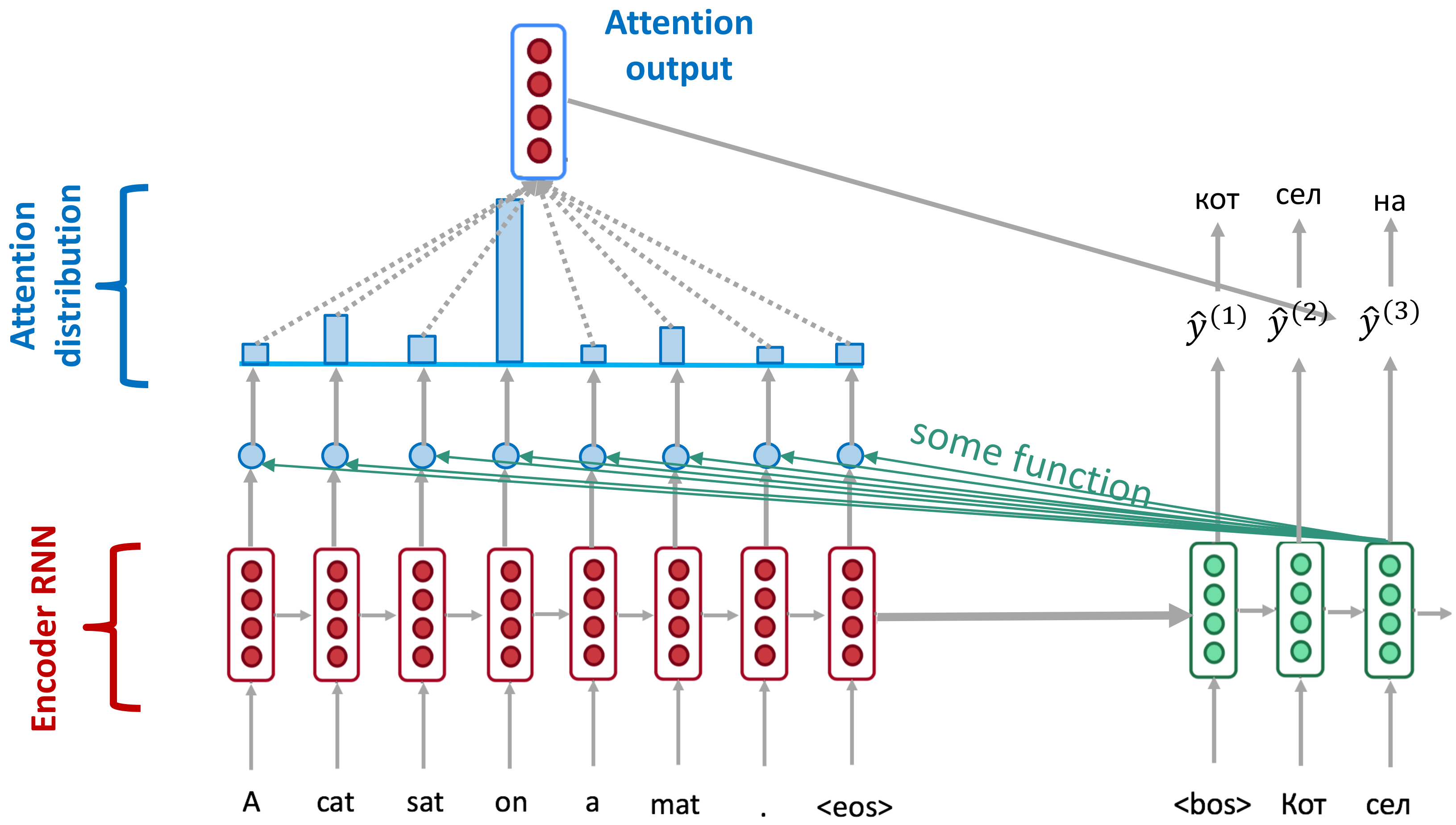
Attention output
Attention distribution
Encoder RNN
some function
кот
$\hat{y}^{(1)}$ $\hat{y}^{(2)}$
A
cat
sat
on
a
mat
.
<eos>
<bos>
Кот

Attention



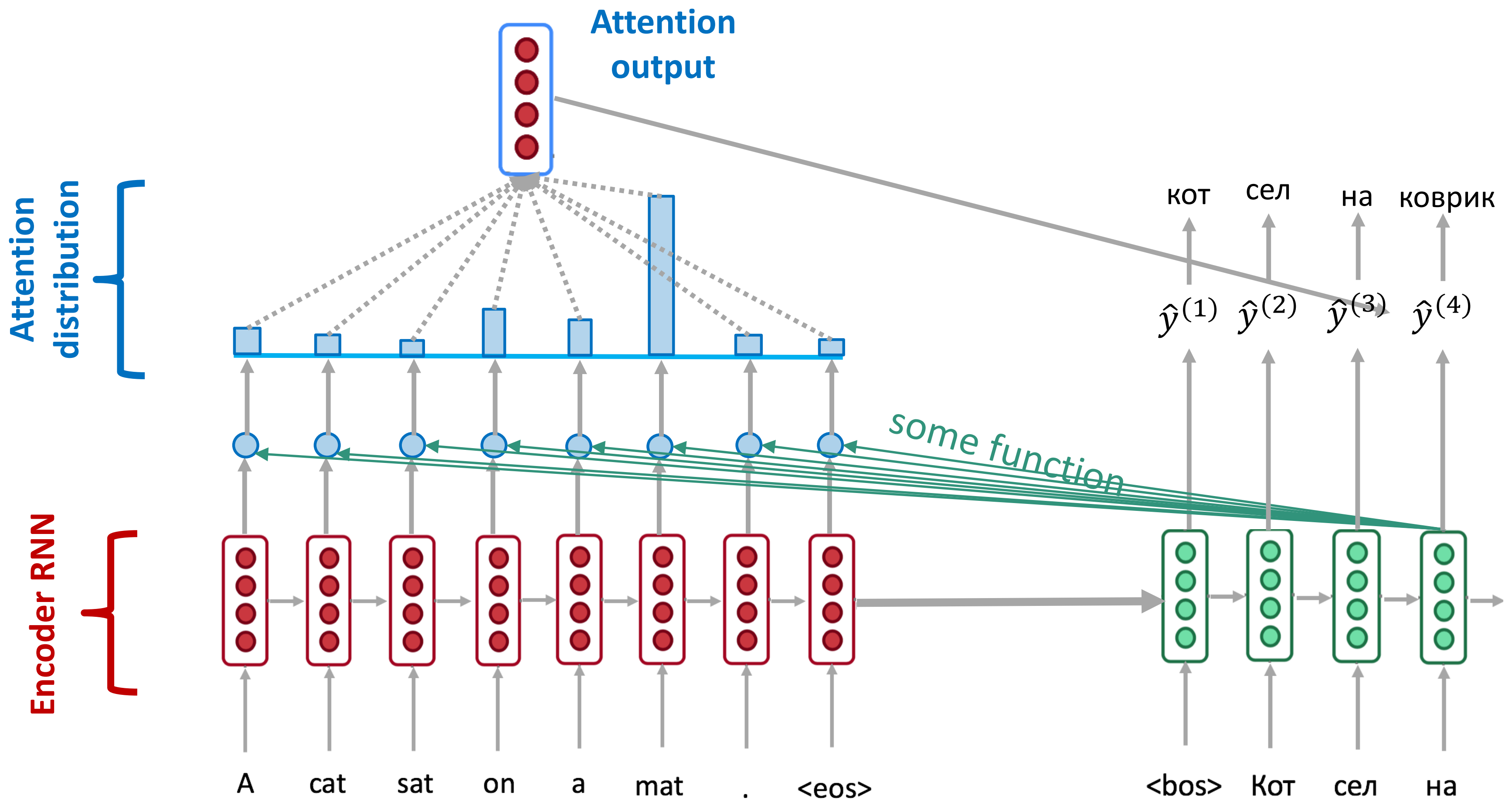
Attention output
Attention distribution
some function
Encoder RNN
кот
сел
$\hat{y}^{(1)}$
$\hat{y}^{(2)}$
A
cat
sat
on
a
mat
.
<eos>
<bos>
Кот

Attention



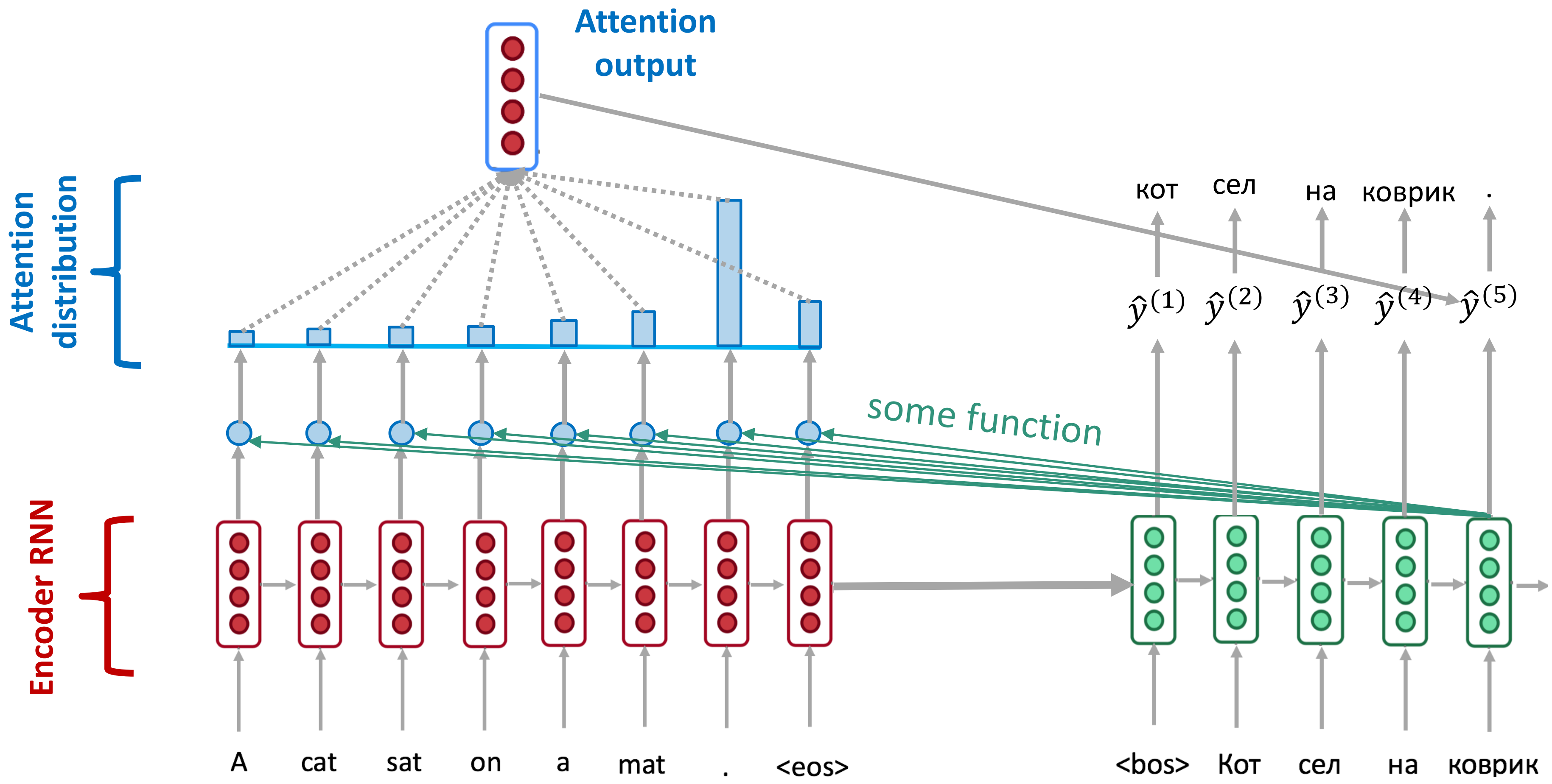
Attention output
Attention distribution
Encoder RNN
some function
кот
сел
на
$\hat{y}^{(1)}$ $\hat{y}^{(2)}$ $\hat{y}^{(3)}$
A
cat
sat
on
a
mat
.
<eos>
<bos>
Кот
сел

Attention



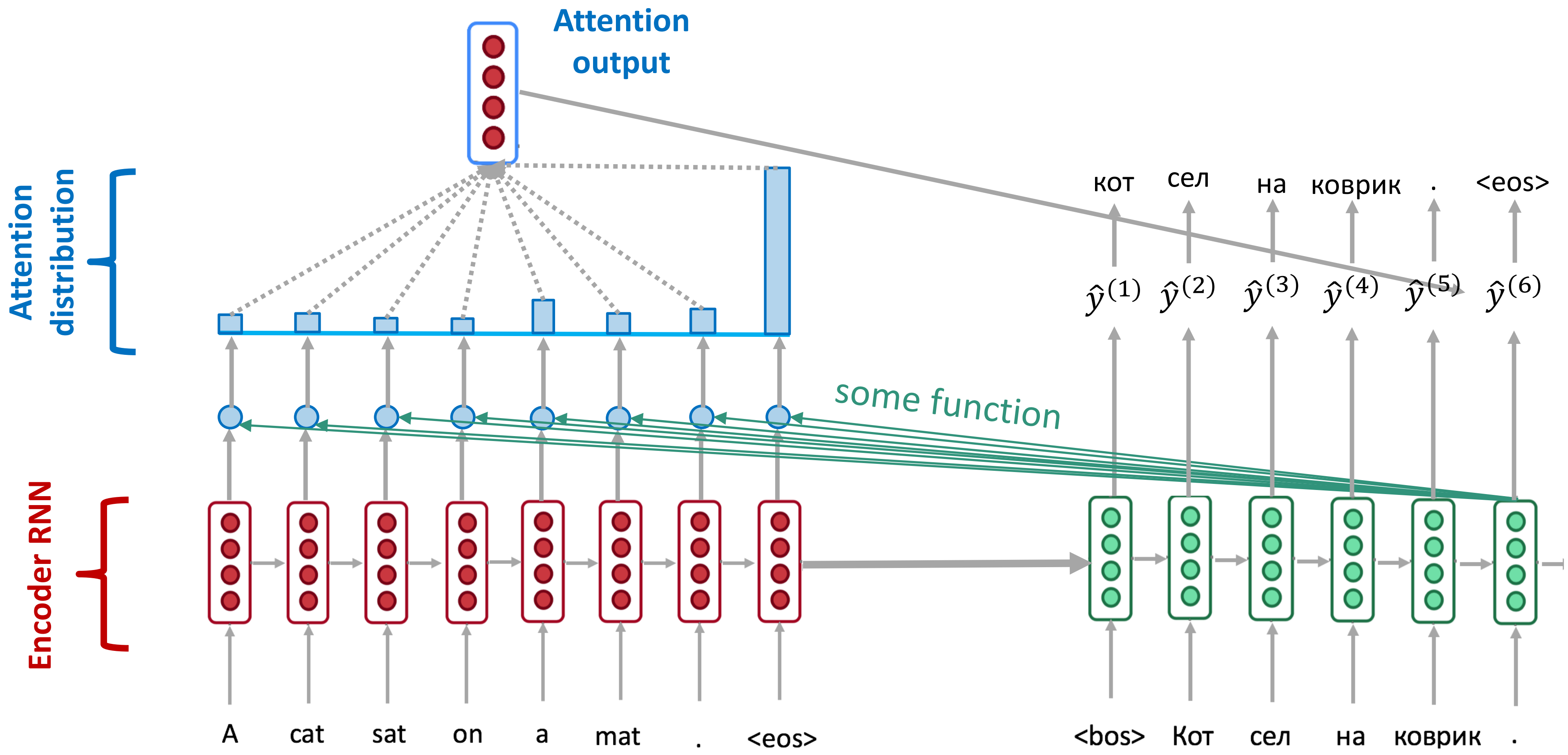
Attention output
Attention distribution
Encoder RNN
some function
кот
сел
на
коврик
$\hat{y}^{(1)}$
$\hat{y}^{(2)}$
$\hat{y}^{(3)}$
$\hat{y}^{(4)}$
A
cat
sat
on
a
mat
.
<eos>
<bos>
Кот
сел
на

Attention



Attention output
Attention distribution
some function
Encoder RNN
кот сел на коврик .
$\hat{y}^{(1)}$ $\hat{y}^{(2)}$ $\hat{y}^{(3)}$ $\hat{y}^{(4)}$ $\hat{y}^{(5)}$
A cat sat on a mat . <eos>
<bos> Кот сел на коврик

Attention

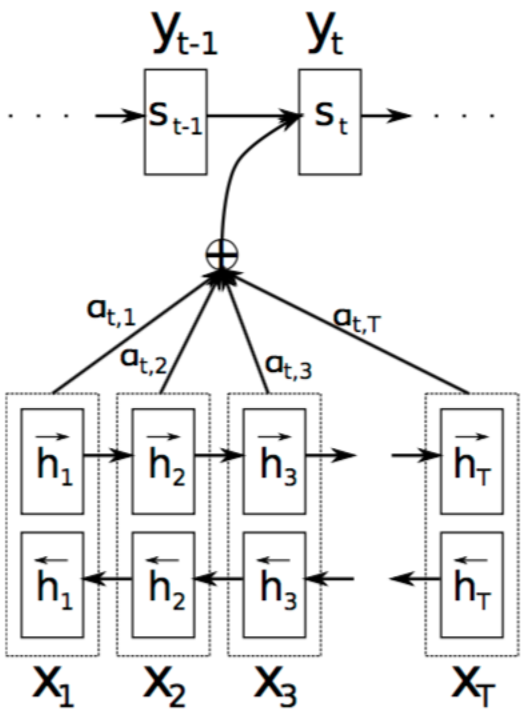


Attention output
Attention distribution
some function
Encoder RNN
кот
сел
на
коврик
.
<eos>
$\hat{y}^{(1)}$ $\hat{y}^{(2)}$ $\hat{y}^{(3)}$ $\hat{y}^{(4)}$ $\hat{y}^{(5)}$ $\hat{y}^{(6)}$
A cat sat on a mat . <eos>
<bos> Кот сел на коврик .

Neural Machine Translation by Jointly Learn to Align and Translate

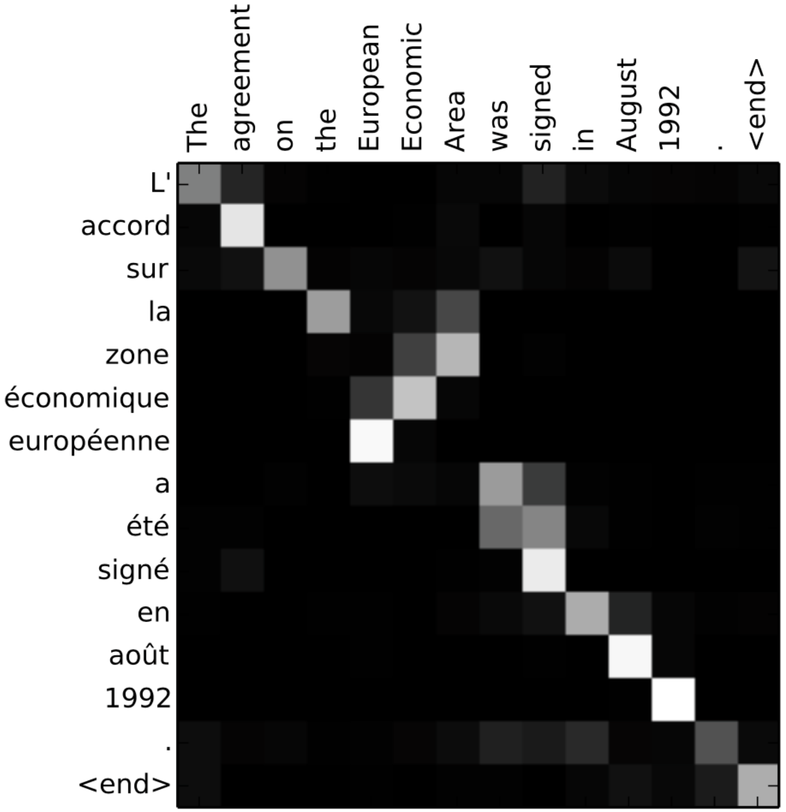
Attention advantages:

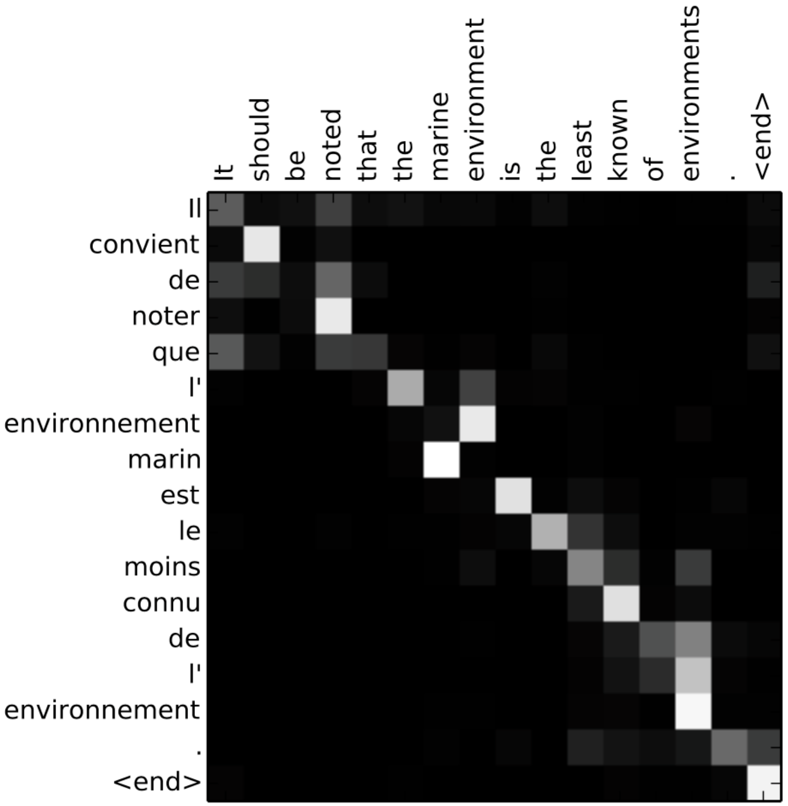
- do not have to remember the whole sentence
- no fixed size representation
- similar to humans



Bahdanau et al, ICLR 2015, https://arxiv.org/pdf/1409.0473.pdf

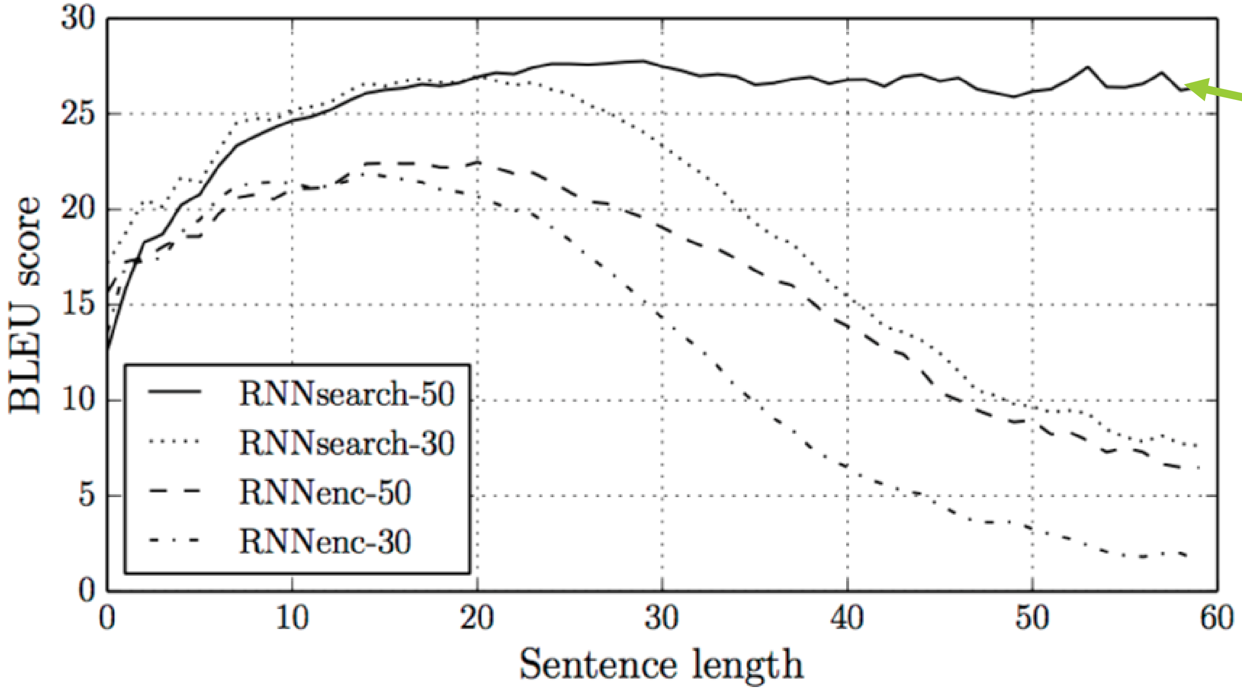
Neural Machine Translation by Jointly Learn to Align and Translate





Bahdanau et al, ICLR 2015, https://arxiv.org/pdf/1409.0473.pdf

Neural Machine Translation by Jointly Learn to Align and Translate



No performance drop for long sentences!

Bahdanau et al, ICLR 2015, https://arxiv.org/pdf/1409.0473.pdf

Attention score: how to compute it?

- dot product
- bilinear function
- multi-layer perceptron
- any, literally, ANY function you can imagine

$$f_{att}(h_i, s_j) = h_i^\top s_j$$

$$f_{att}(h_i, s_j) = h_i^\top \mathbf{W}_a s_j$$

$$f_{att}(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{v}_a{}^\top \tanh(\mathbf{W}_a[\mathbf{h}_i; \mathbf{s}_j])$$

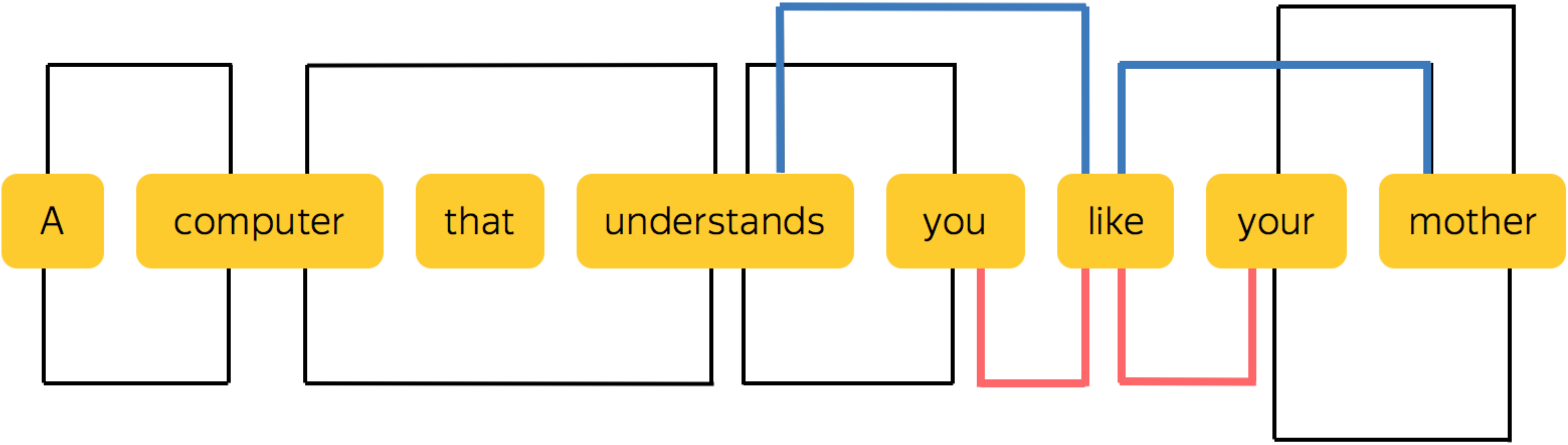
$$z(c, m, q) = \left[c, m, q, c \circ q, c \circ m, |c - q|, |c - m|, c^T W^{(b)} q, c^T W^{(b)} m\right]$$

$$G(c, m, q) = \sigma\left(W^{(2)} \tanh\left(W^{(1)} z(c, m, q) + b^{(1)}\right) + b^{(2)}\right)$$

A bit crazy, huh?

Transformer:
Attention is all you need!

Self-Attention



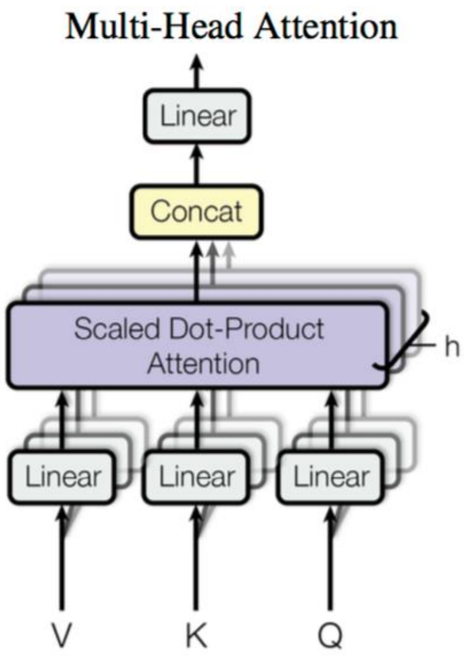
(example and picture from David Talbot)

Multi-Head Attention

Она руководит **новым** проектом

- Gender agreement
- Case government
- Lexical preferences
- …

(example from David Talbot)

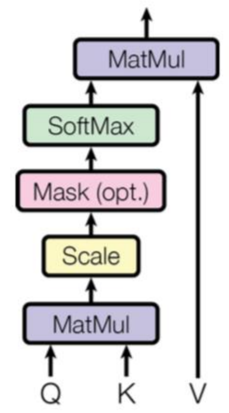


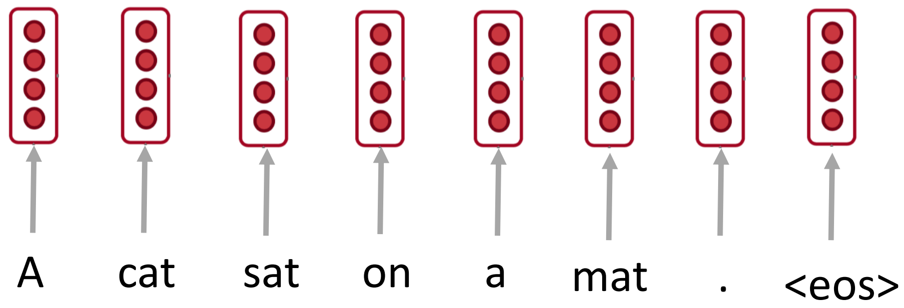
$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},\ldots,\mathrm{head_h})W^O$$
$$\text{where}\ \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Self-Attention

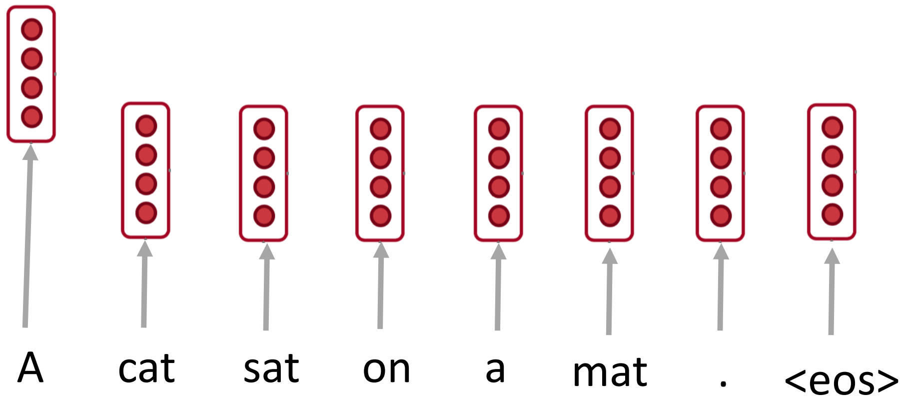
$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention



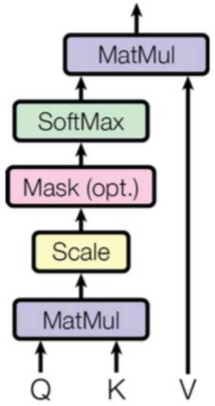


Self-Attention



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

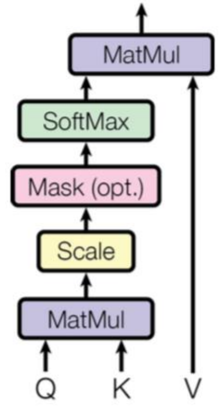
Scaled Dot-Product Attention

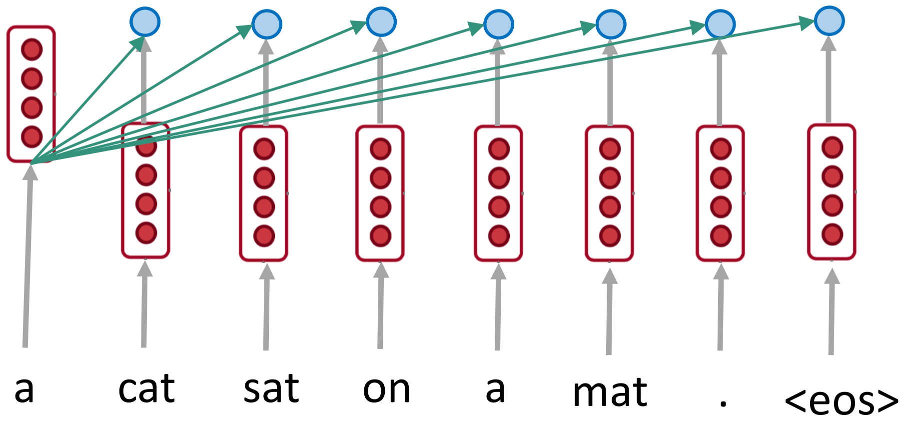


Self-Attention

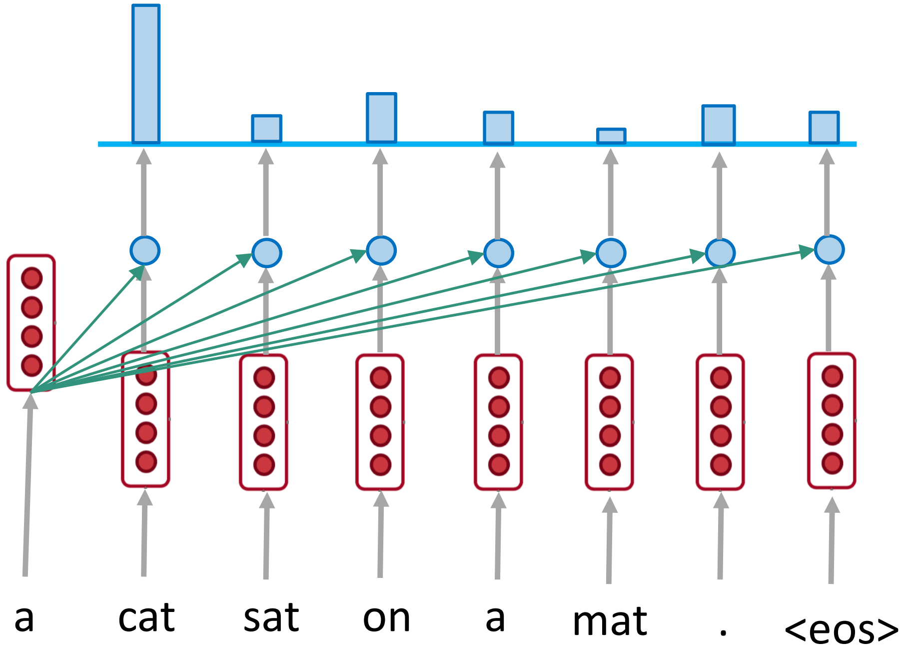
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention

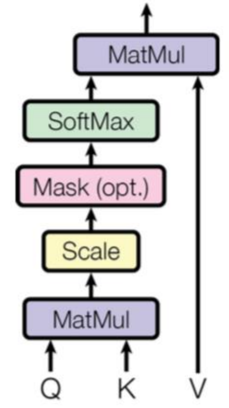




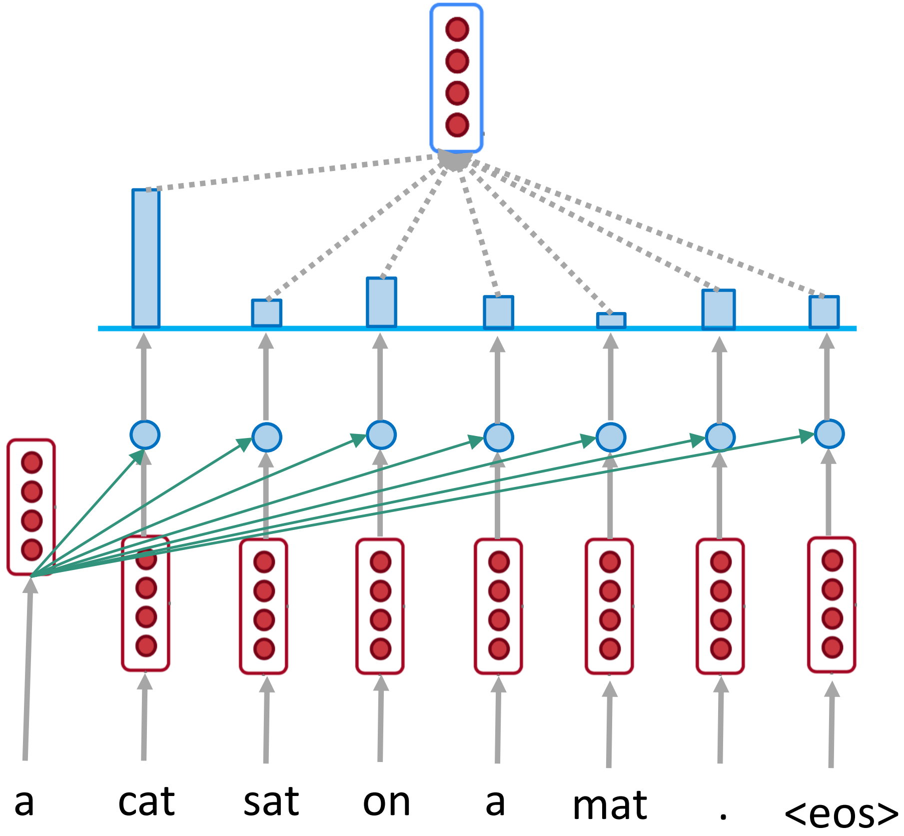
Self-Attention



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

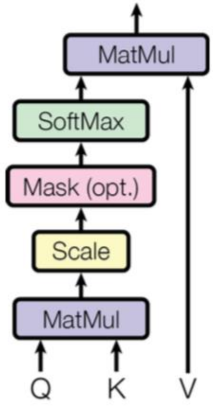


Self-Attention



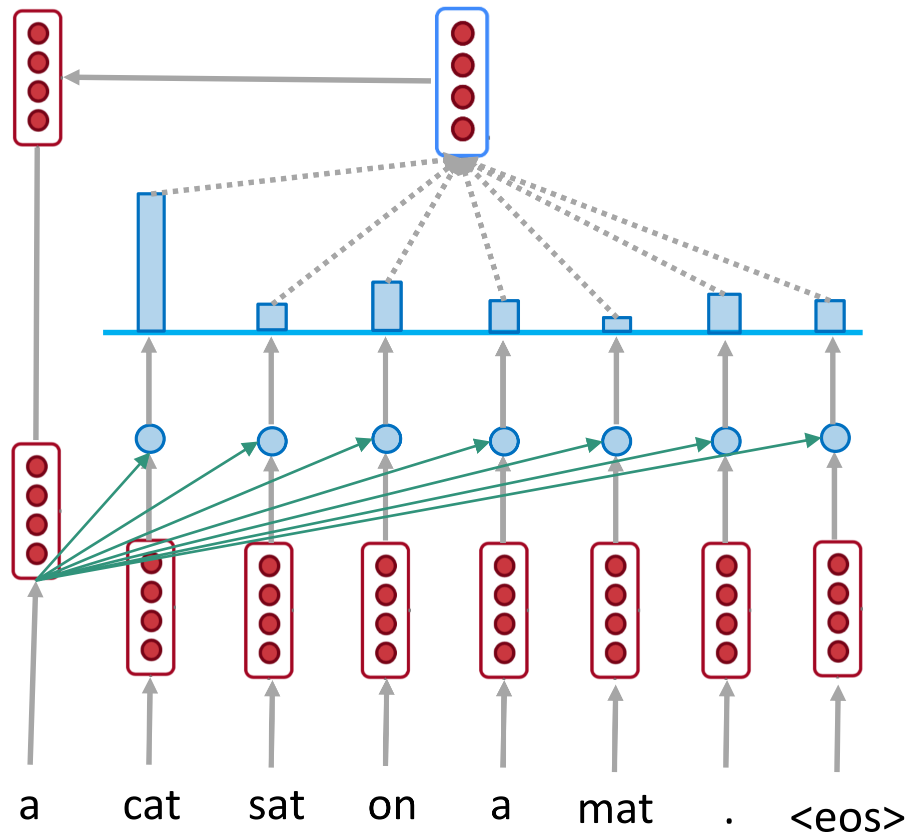
$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention



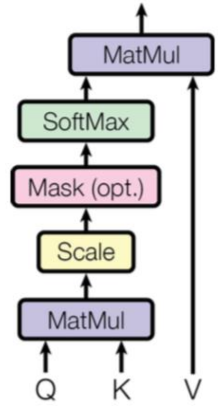
Self-Attention

update representation for the word ”a”

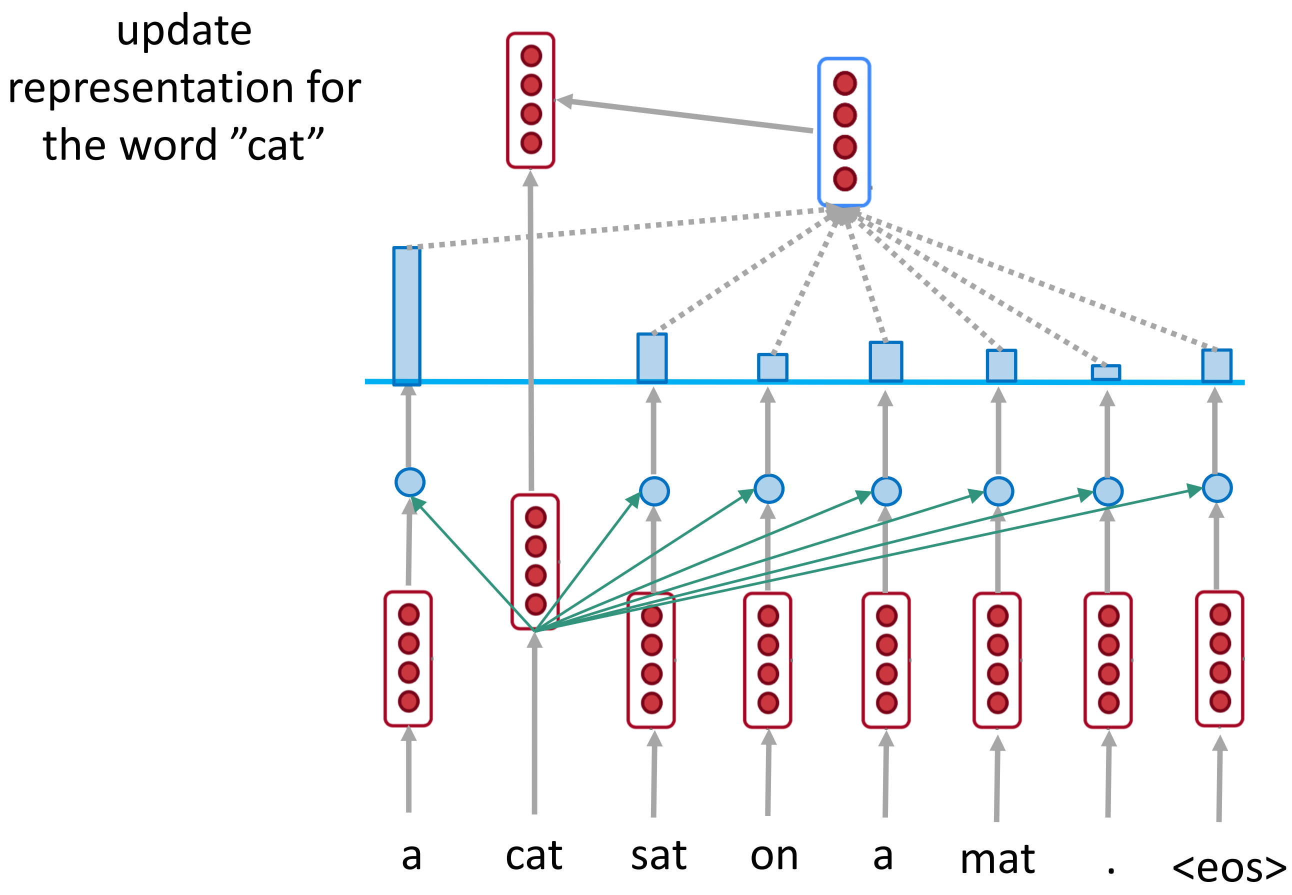


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention

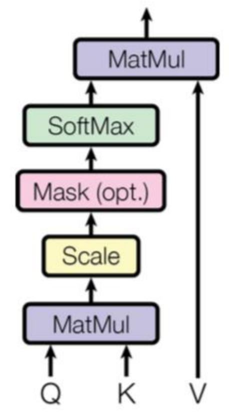


Self-Attention



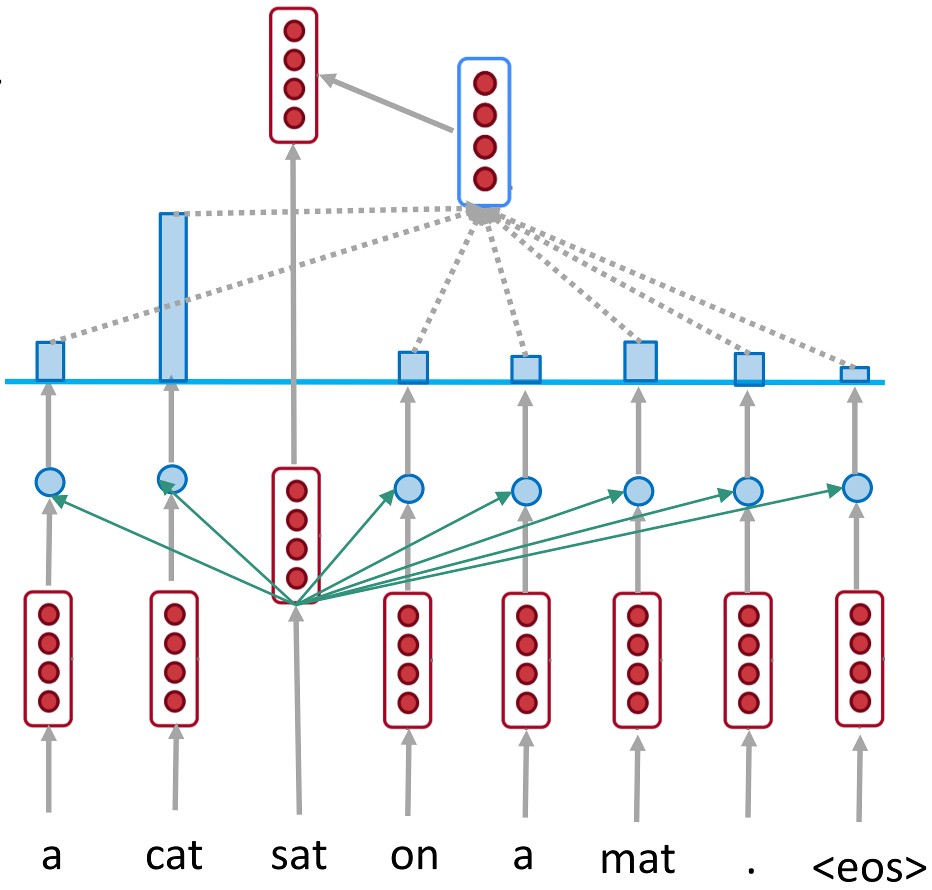
$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Scaled Dot-Product Attention



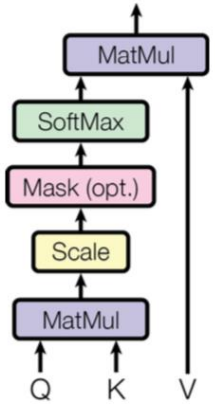
Self-Attention

update representation for the word ”sat”



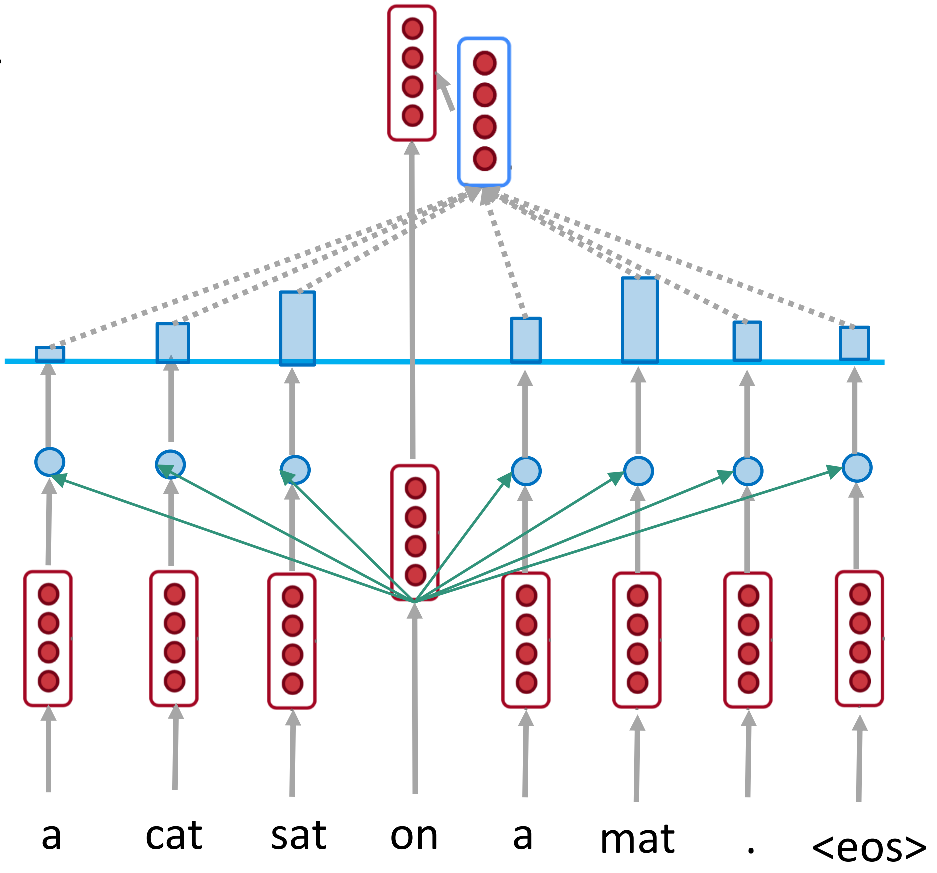
$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention



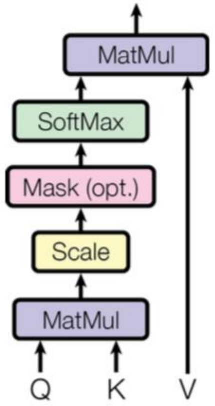
Self-Attention

update
representation for
the word ”on”



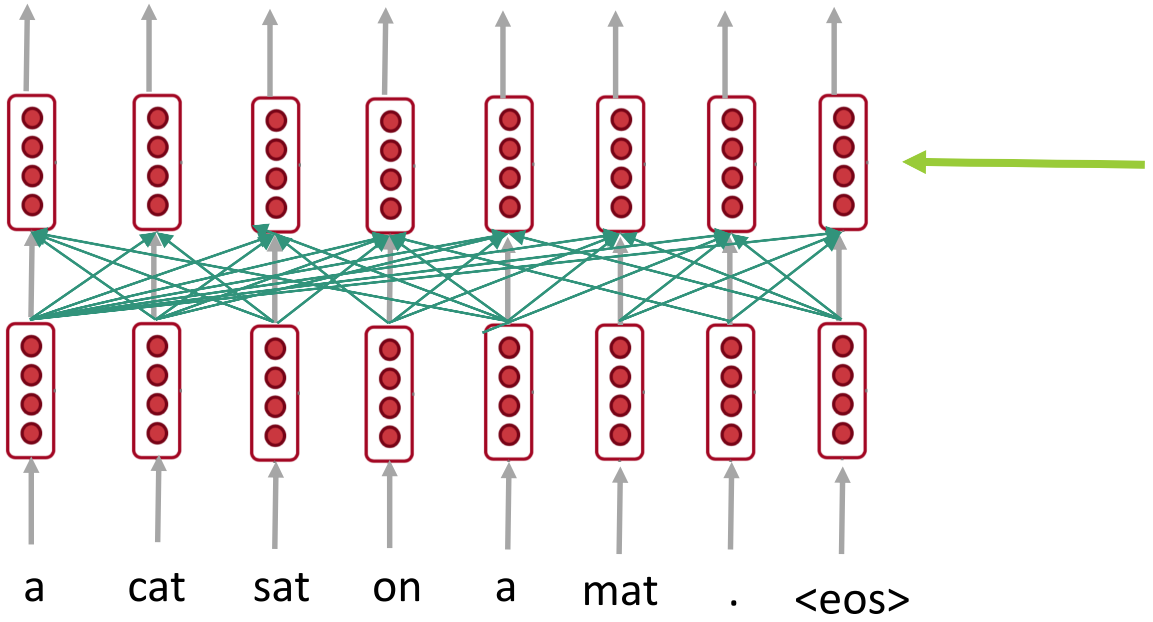
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention



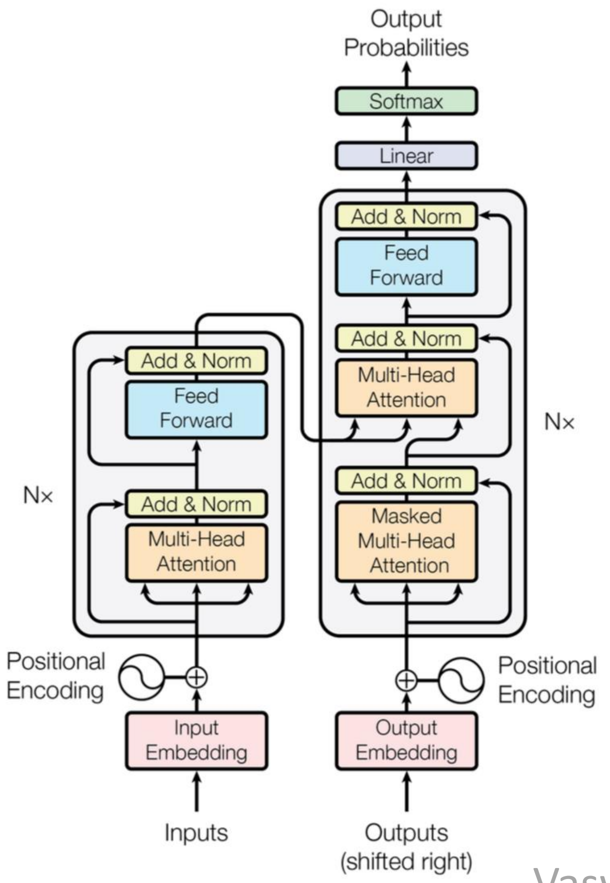
Self-Attention

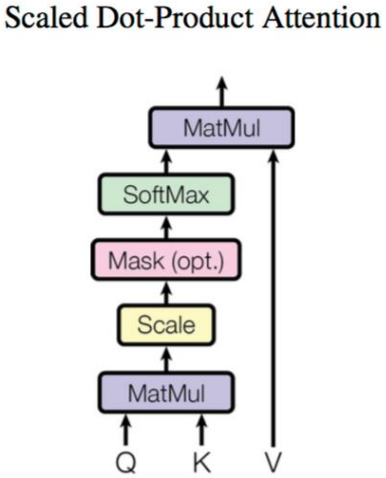
$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

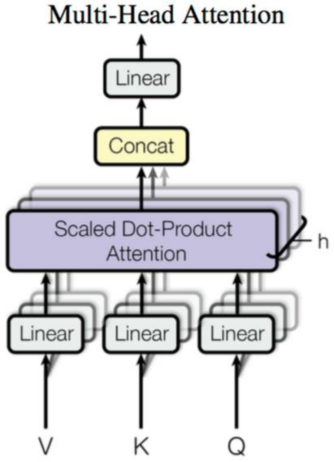


Updated representations for each word (one layer)

Transformer



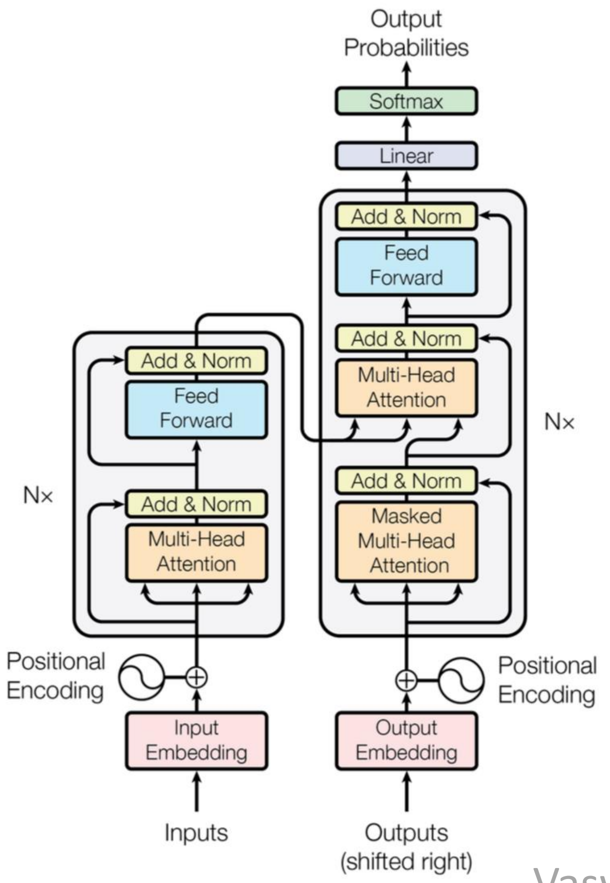




$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Vaswani et al, NIPS 2017, https://papers.nips.cc/paper/7181-attention-is-all-you-need

Transformer

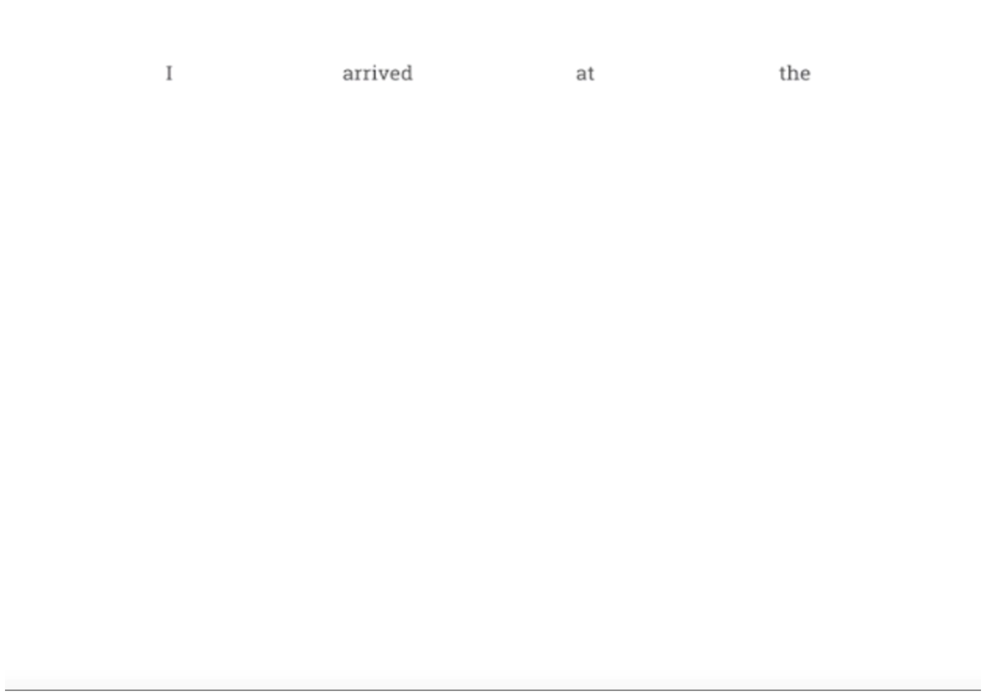


Advantages:

- No recurrence: parallel encoding
- Fast training: both encoder and decoder are parallel
- No long connections: O(1) for all tokens
- Three attentions: the model does not have to remember too much
- Multi-head attention allows to pay attention to different aspects

Vaswani et al, NIPS 2017, https://papers.nips.cc/paper/7181-attention-is-all-you-need

Transformer



I arrived at the bank after crossing the…

… river? …road?

Vaswani et al, NIPS 2017, https://papers.nips.cc/paper/7181-attention-is-all-you-need

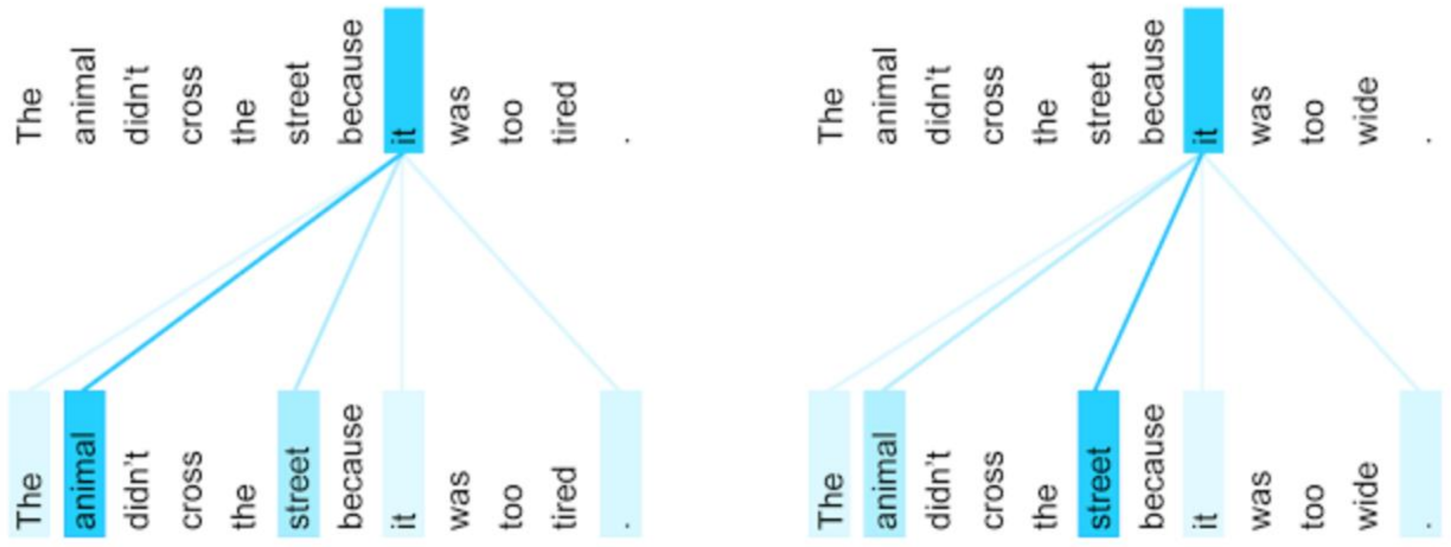
Transformer

The animal didn't cross the street because it was too tired.
L'animal n'a pas traversé la rue parce qu'il était trop fatigué.

The animal didn't cross the street because it was too wide.
L'animal n'a pas traversé la rue parce qu'elle était trop large.

Vaswani et al, NIPS 2017, https://papers.nips.cc/paper/7181-attention-is-all-you-need

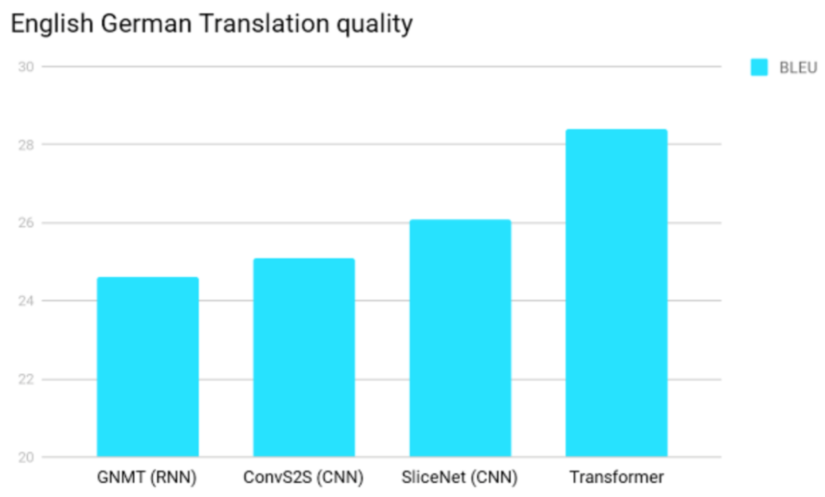
Transformer



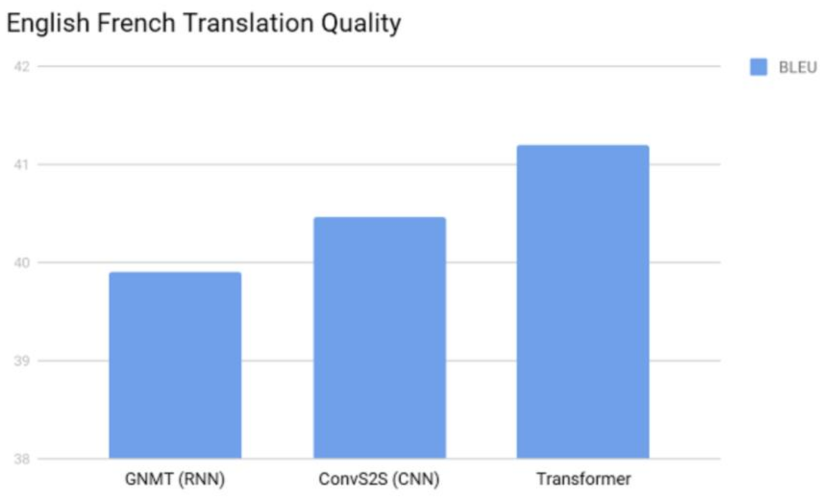
The encoder self-attention distribution for the word "it" from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

Vaswani et al, NIPS 2017, https://papers.nips.cc/paper/7181-attention-is-all-you-need

Transformer



BLEU scores (higher is better) of single models on the standard WMT newstest2014 English to German translation benchmark.



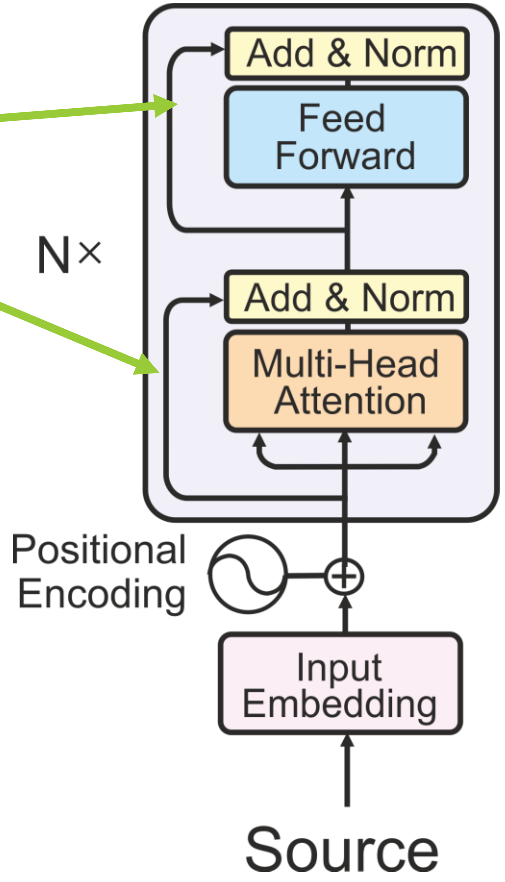
BLEU scores (higher is better) of single models on the standard WMT newstest2014 English to French translation benchmark.

Vaswani et al, NIPS 2017, https://papers.nips.cc/paper/7181-attention-is-all-you-need

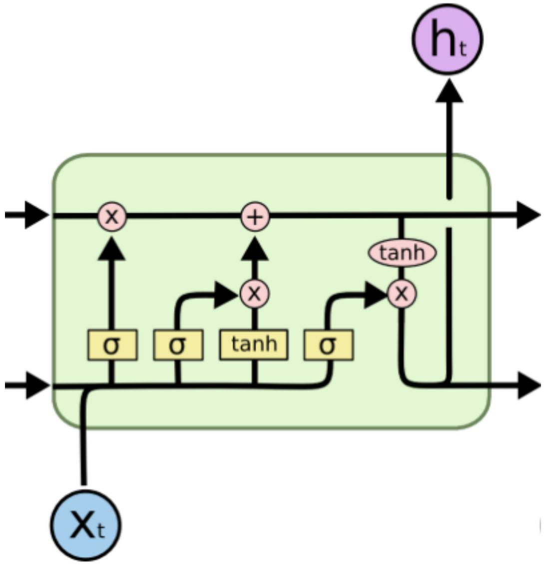
A piece of understanding: Why residual connection?

Why residual connection?

Residual connection



Example 1:
LSTM and long-term dependences



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$h_t = o_t * \tanh\left(C_t\right)$$

or

Original version:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$$C_t = C_{t-1} + i_t * \tilde{C}_t$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Why the problem of vanishing/exploding gradient here is not such urgent?

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Example 2:
Deep Residual Learning for Image Recognition

Deep NNs are hard to train!

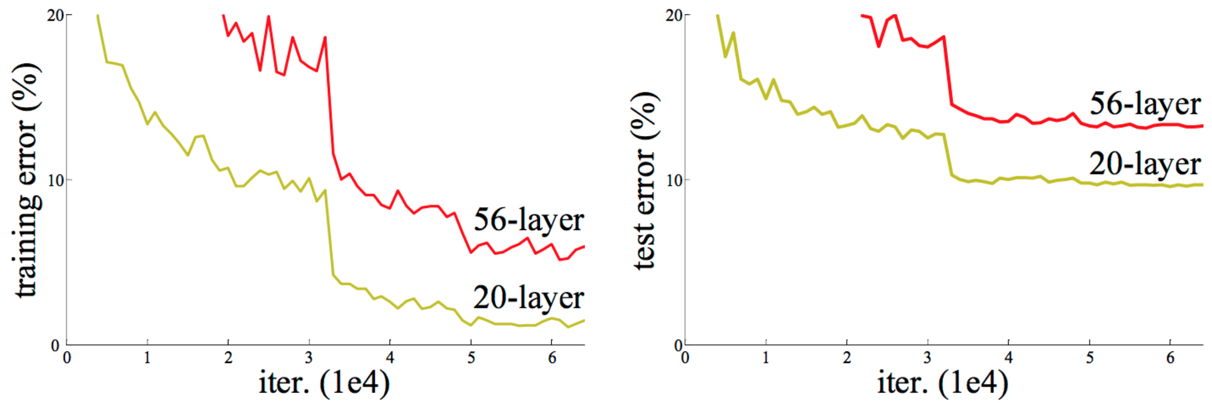


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

He et al, CVPR 2016, https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

Example 2:
Deep Residual Learning for Image Recognition

Solution: add residual connection, then gradients can flow!

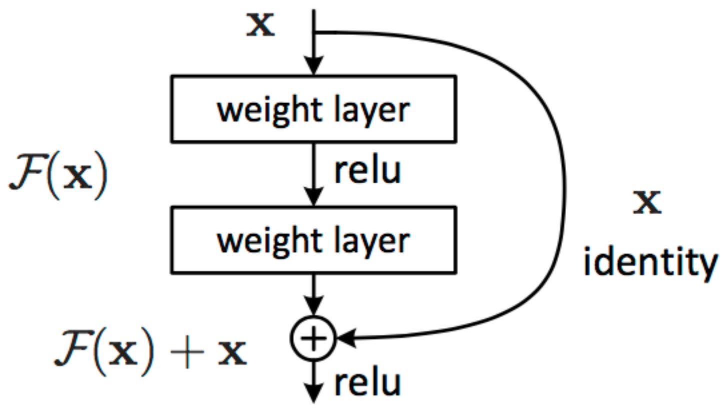


Figure 2. Residual learning: a building block.

He et al, CVPR 2016,
https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

Example 2:
Deep Residual Learning for Image Recognition

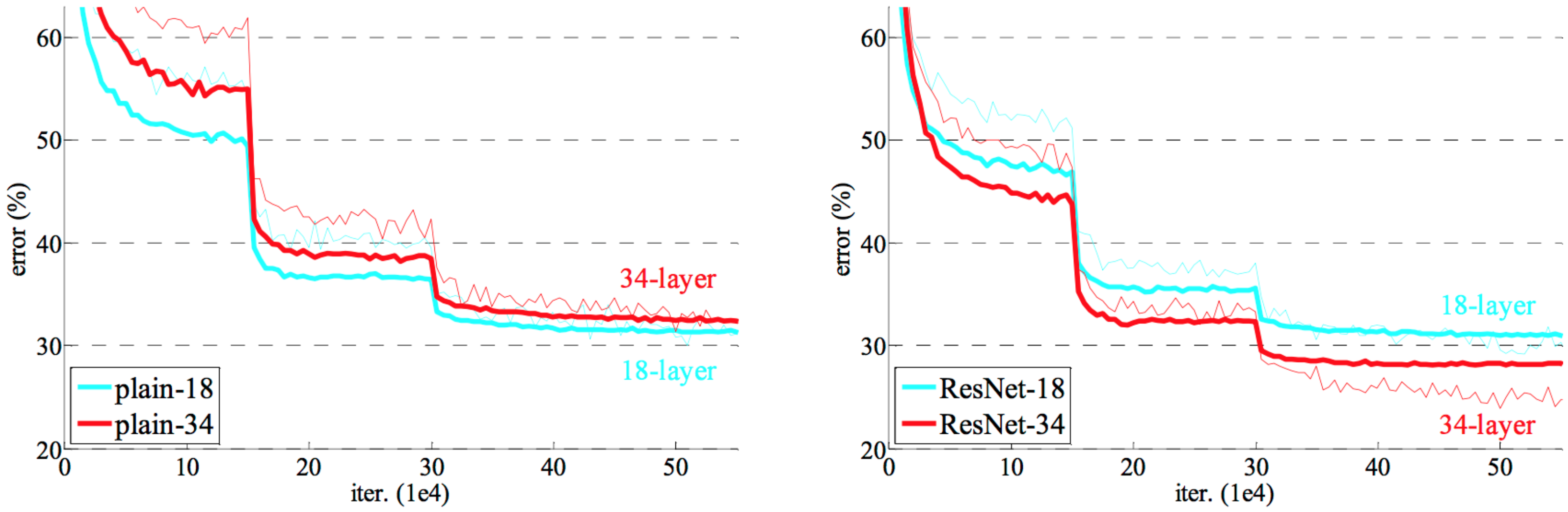


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

He et al, CVPR 2016, https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

Example 2':
Highway networks

gate

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot (1 - T(\mathbf{x}, \mathbf{W_T})).$$

Layer pre-output

Layer input

Srivastava et al, 2015, https://arxiv.org/pdf/1507.06228.pdf

What if…
We don’t know our voc size?

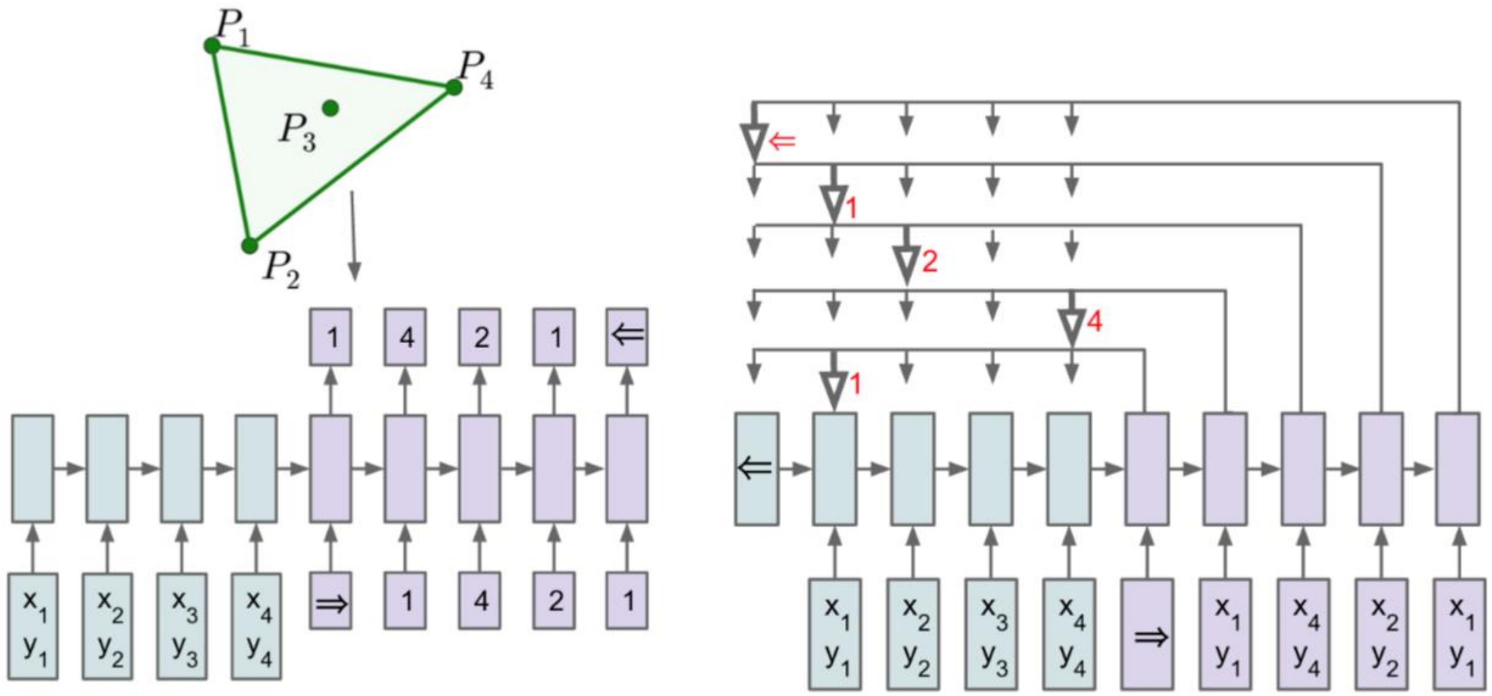
SUPPOSE THE TASK IS TO PUT SOME INPUTS IN A PARTICULAR ORDER

Pointer Networks

Attention weights

=

output distribution



(a) Sequence-to-Sequence

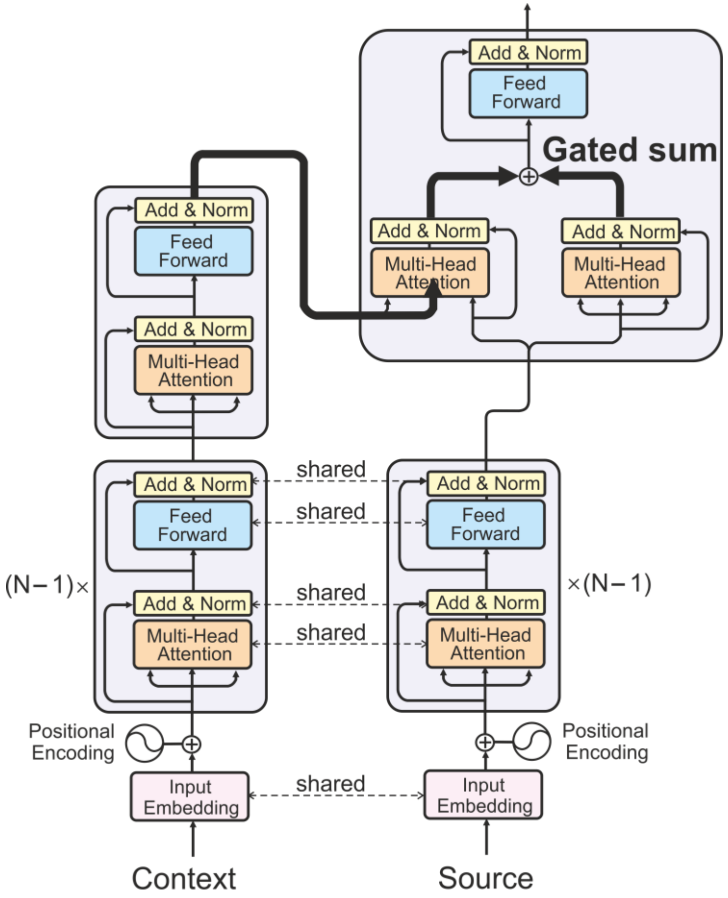
(b) Ptr-Net

Vinyals et al, NIPS 2015, https://papers.nips.cc/paper/5866-pointer-networks.pdf

What if…
We give context to an NMT system?

LET’S SAY THAT CONTEXT IS ONE PREVIOUS SENTENCE

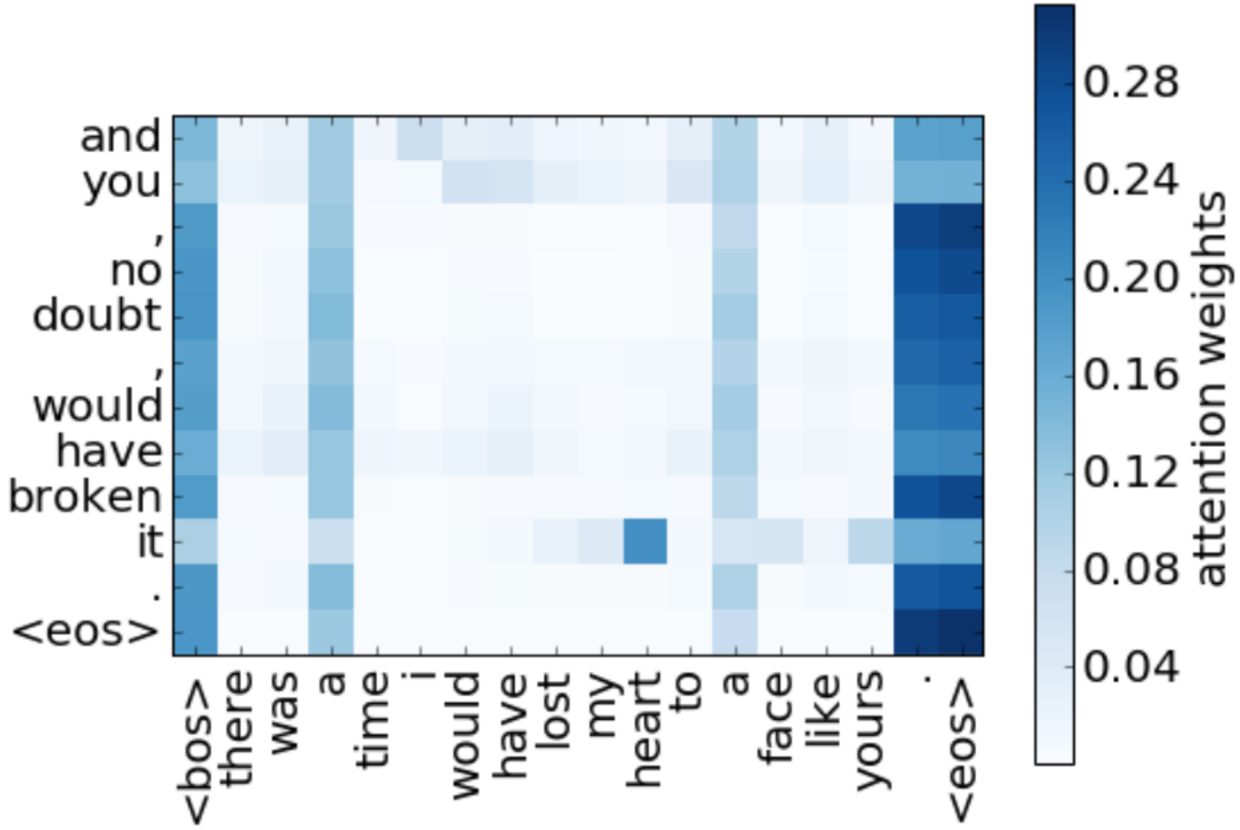
Context-aware NMT learns anaphora resolution



- start with the Transformer
  [Vaswani et al, 2018]
- incorporate context information
  on the encoder side
- use a separate encoder for context
- share first N-1 layers of source and context encoders
- the last layer incorporates contextual information

Voita et al, ACL 2018, http://www.aclweb.org/anthology/P18-1117

Context-aware NMT learns anaphora resolution



Context:

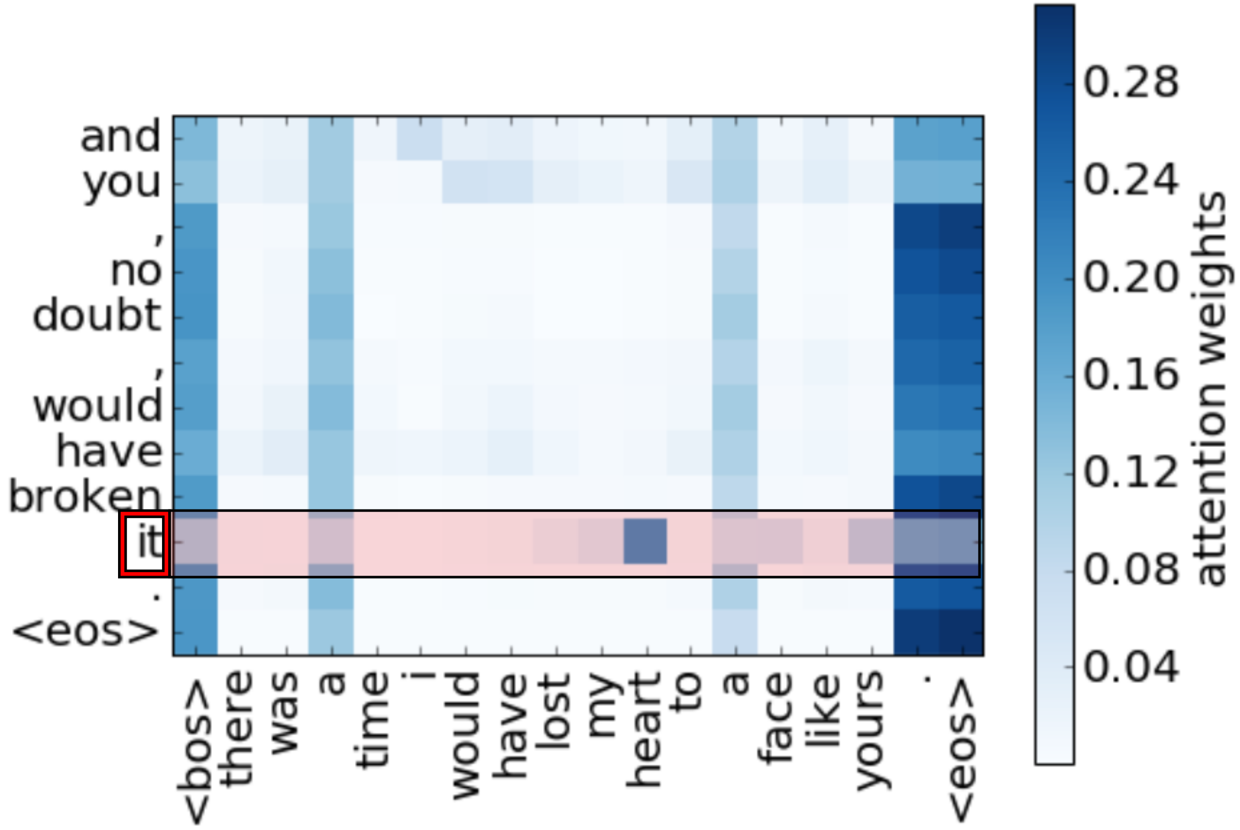
› There was a time I would have lost my heart to a face like yours.

Source:

› And you, no doubt, would have broken it.

Voita et al, ACL 2018, http://www.aclweb.org/anthology/P18-1117

Context-aware NMT learns anaphora resolution



Context:

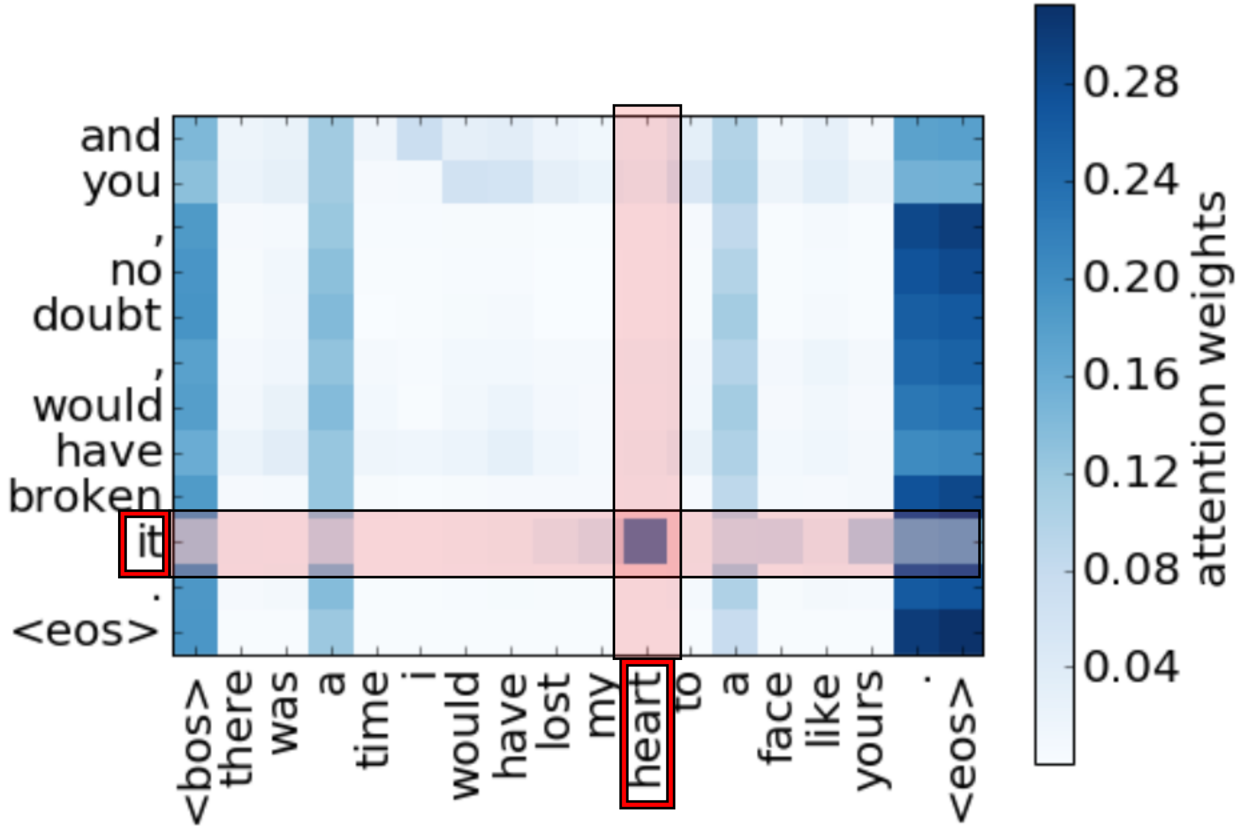
- There was a time I would have lost my heart to a face like yours.

Source:

- And you, no doubt, would have broken it.

Voita et al, ACL 2018, http://www.aclweb.org/anthology/P18-1117

Context-aware NMT learns anaphora resolution



Context:

› There was a time I would have lost my heart to a face like yours.

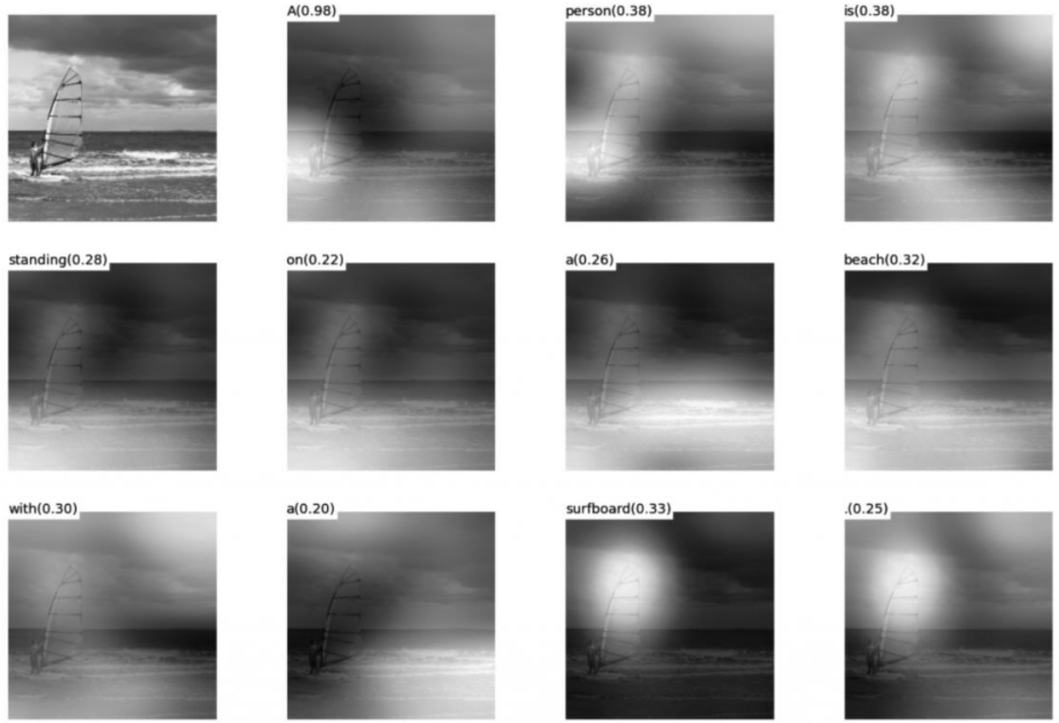
Source:

› And you, no doubt, would have broken it.

Voita et al, ACL 2018, http://www.aclweb.org/anthology/P18-1117

Attention:
Other use cases

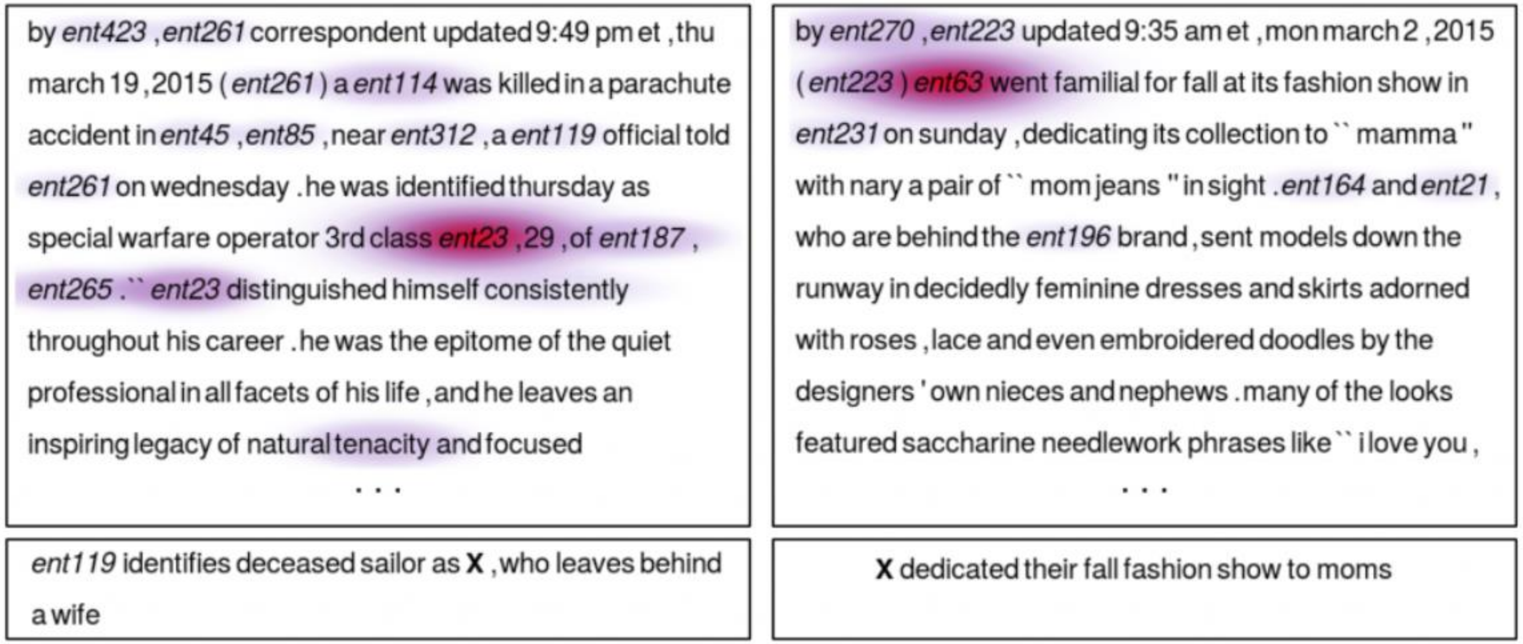
Image caption generation



(b) A person is standing on a beach with a surfboard.

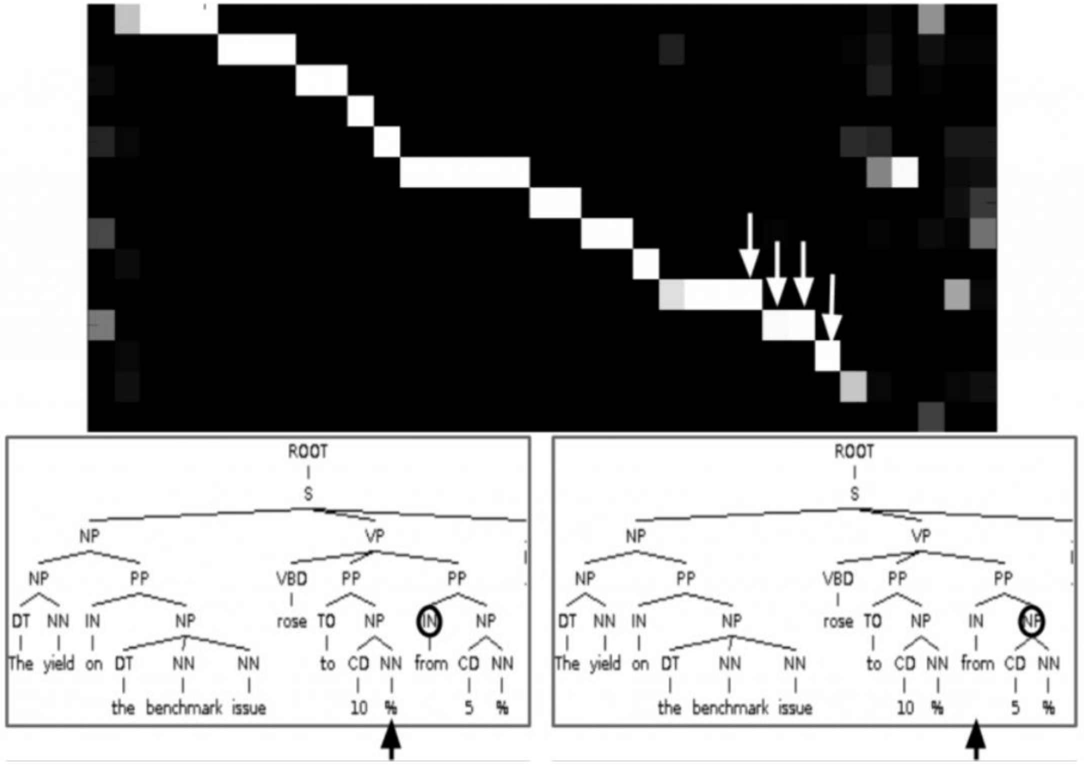
Use a Convolutional Neural Network to “encode” the image, and a Recurrent Neural Network with attention mechanisms to generate a description.

Question answering task



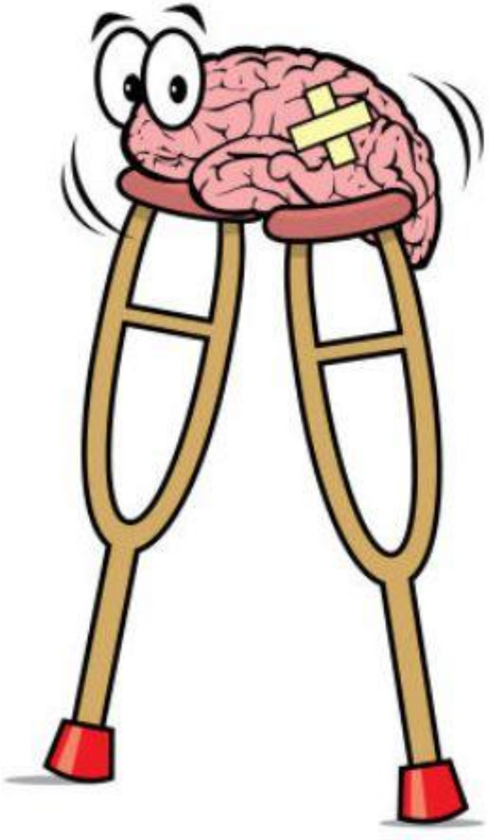
Use a RNN to read a text, read a (synthetically generated) question, and then produce an answer.

Generate sentence parse trees

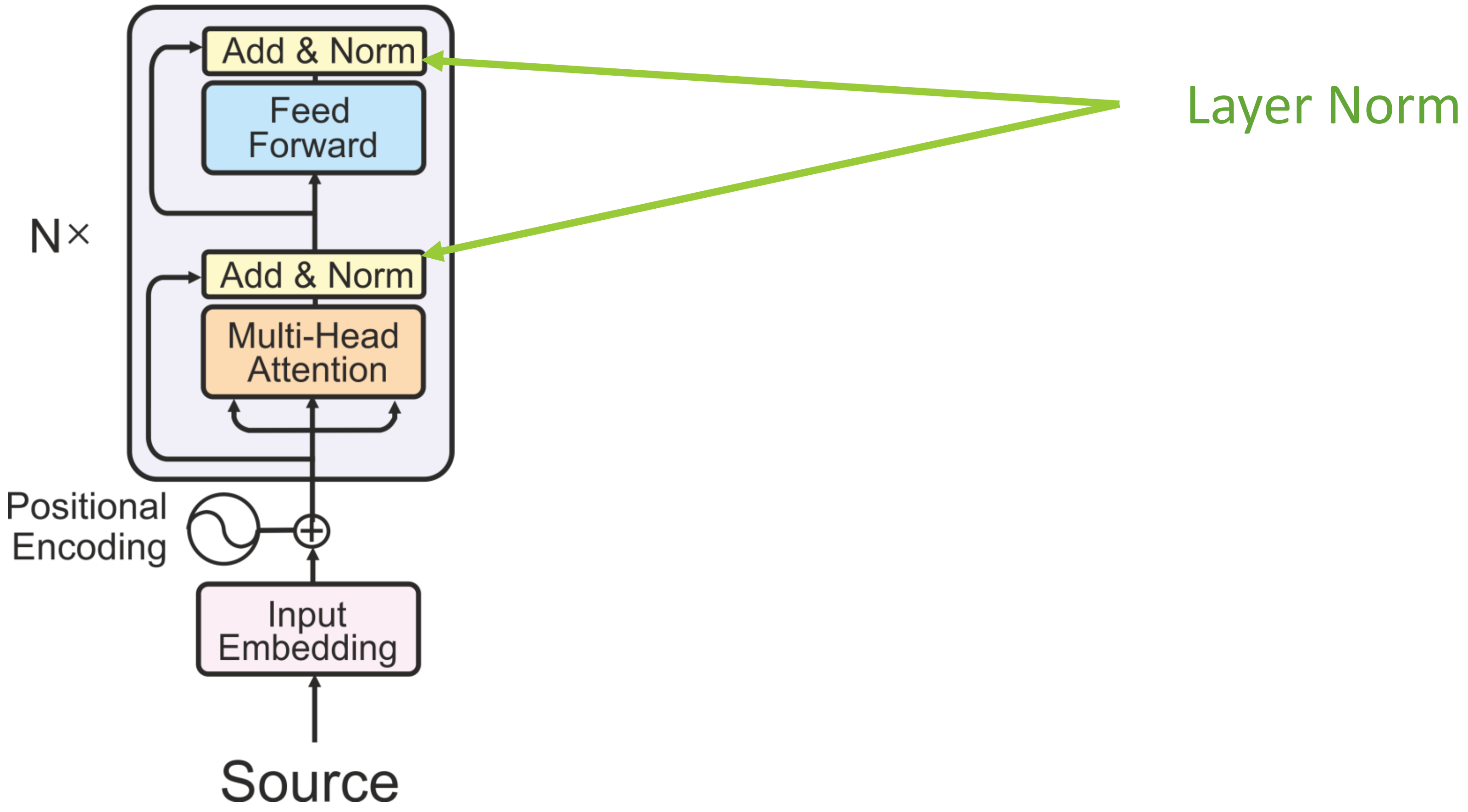


Use a Recurrent Neural Network with attention mechanism to generate sentence parse trees

Hack of the day



Layer Norm



Add & Norm
Feed Forward
N×
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Source
Layer Norm

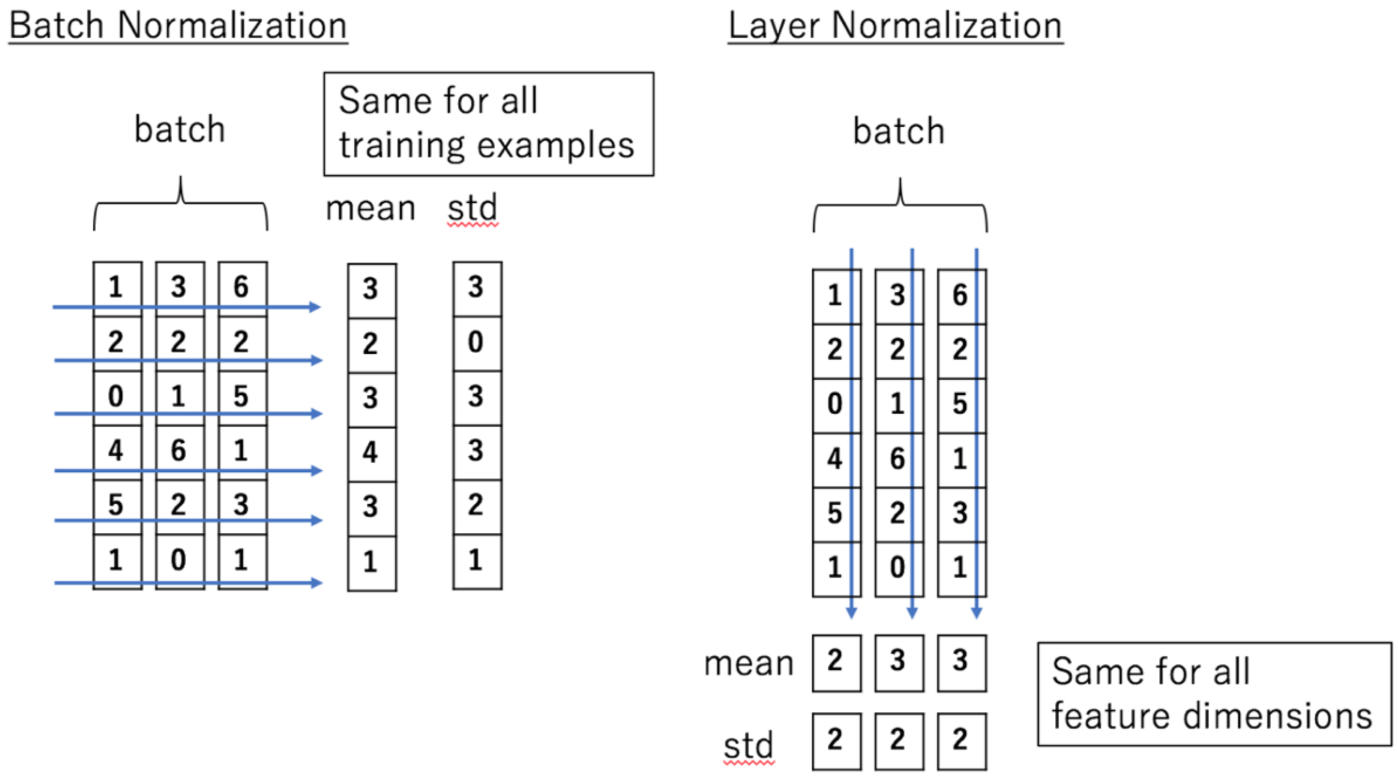
Layer norm

Batch normalization

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}$$
$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_{ij} - \mu_j)^2$$
$$\hat{x_{ij}} = \frac{x_{ij}-\mu_j}{\sqrt{\sigma_j^2+\epsilon}}$$

Layer normalization:

$$\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij}$$
$$\sigma_i^2 = \frac{1}{m}\sum_{j=1}^{m}(x_{ij} - \mu_i)^2$$
$$\hat{x_{ij}} = \frac{x_{ij}-\mu_i}{\sqrt{\sigma_i^2+\epsilon}}$$



Paper: Ba et al, 2016, https://arxiv.org/abs/1607.06450
Pictures: http://mlexplained.com/2018/01/13/weight-normalization-and-layer-normalization-explained-normalization-in-deep-learning-part-2/

That’s all for today!

Coming soon:

- Structured learning
- EM, Machine Translation (specific)
- Dialogue systems
- and much, much more

Sincerely yours,
Yandex Research