

Автоматическая обработка текстов Seq2Seq и Attention

Лекция 6

Емельянов А. А.
login-const@mail.ru

Задача машинного перевода (Seq2Seq)

- Исходное предложение: $x = (x_1, x_2, \dots, x_{T_x})$
- Целевое предложение: $y = (y_1, y_2, \dots, y_{T_y})$
- Любой тип системы машинного перевода может быть представлен как функция:

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

- Перевод эквивалентен поиску целевого предложения, которое максимизирует условную вероятность y при известном x , т. е.

$$\operatorname{argmax}_y p(y|x)$$

Задача машинного перевода (Seq2Seq)

- Системы машинного перевода создают вероятностную модель для вероятностей y при условии x ,

$$p(y|x, \theta)$$

- и ищут целевое предложение максимизируя вероятность:

$$\hat{y} = \operatorname{argmax}_y p(y|x, \theta)$$

- θ – это параметры модели, задающие за распределение вероятностей.

Задача машинного перевода (Seq2Seq)

- **Моделирование:**

Какую модель $p(y|x, \theta)$ мы ищем?

- **Обучение:**

Нам необходим метод для обучения подходящих параметров θ по тренировочным данным.

- **Поиск:**

Мы должны решить проблему поиска наиболее вероятного предложения (решение: «*argmax*»).

«Условные» языковые модели

- Seq2Seq – непосредственные модели вероятности $P(y|x)$:

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

«Условные» языковые модели

- Seq2Seq – непосредственные модели вероятности $P(y|x)$:

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

Вероятность следующих целевых слов до сих пор и исходное предложение x

Условные языковые модели

- Seq2Seq – непосредственные модели вероятности $P(y|x)$:

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

Вероятность следующих целевых слов до сих пор и исходное предложение x

- Почему условные языковые модели?

Условные языковые модели

- Seq2Seq – непосредственные модели вероятности $P(y|x)$:

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

Вероятность следующих целевых слов до сих пор и исходное предложение x

- Почему условные языковые модели?
 - Языковая модель предсказывает следующее слово?
 - Условная – предсказания обусловлены исходным предложением.

Encoder-Decoder архитектура

Recap: RNN LM

- Рекуррентные языковые модели (RNN LM)

- Итоговое распределение:

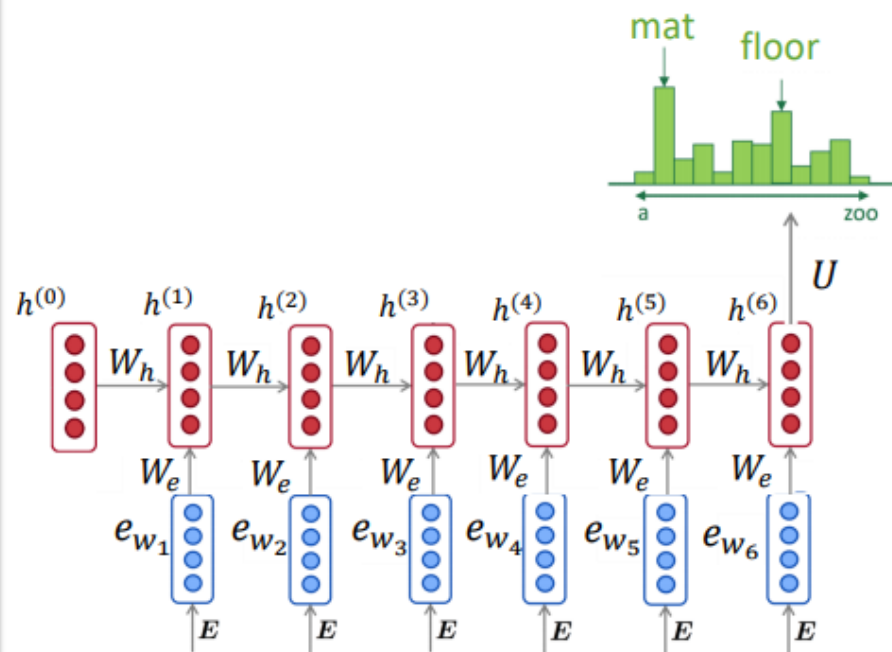
$$\hat{y}_t = \text{softmax}(Uh^{(t)} + b_2) \in R^{|V|}$$

- Скрытые состояния

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_x e^{(t-1)} + b_1)$$

- Эмбеддинги слов

- Слова или токены



$\langle bos \rangle$ Котик очень тихо спал на
 $x^{(i-6)} \quad x^{(i-5)} x^{(i-4)} \quad x^{(i-3)} \quad x^{(i-2)} \quad x^{(i-1)}$

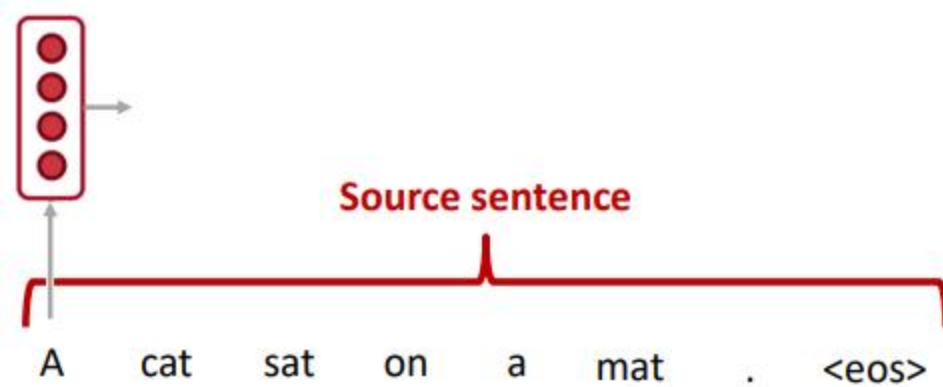
RNN Encoder-Decoder (Vanilla)



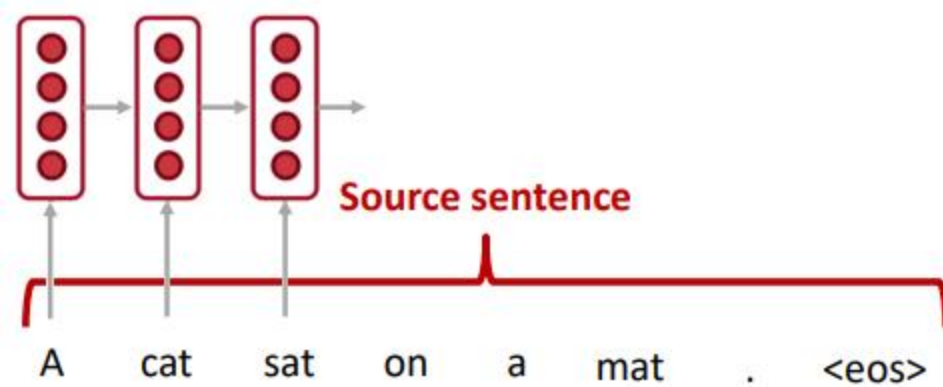
Source sentence

A cat sat on a mat . <eos>

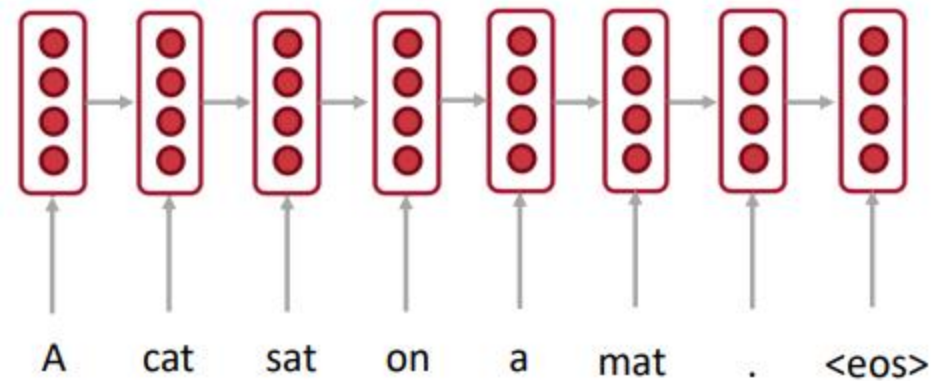
RNN Encoder-Decoder (Vanilla)



RNN Encoder-Decoder (Vanilla)



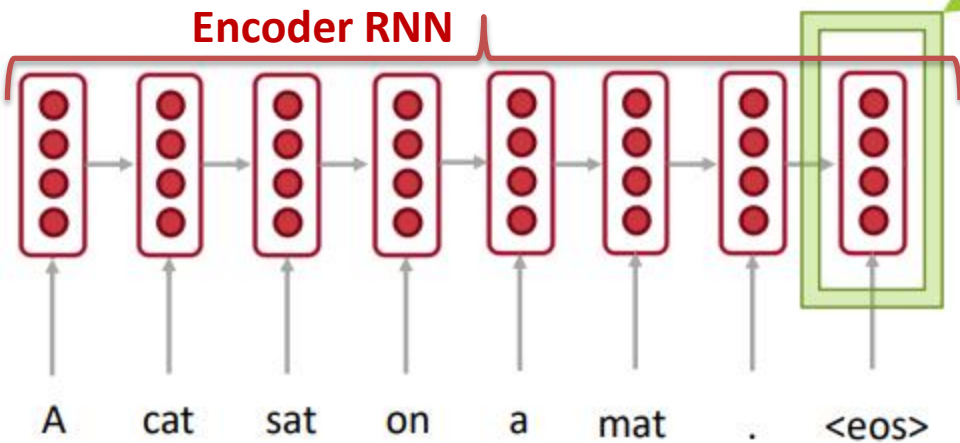
RNN Encoder-Decoder (Vanilla)



RNN Encoder-Decoder (Vanilla)

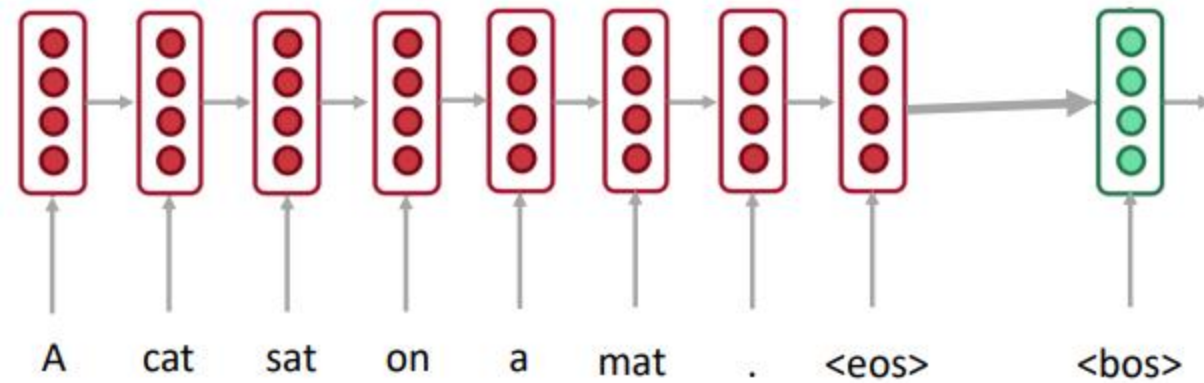


Encoder RNN

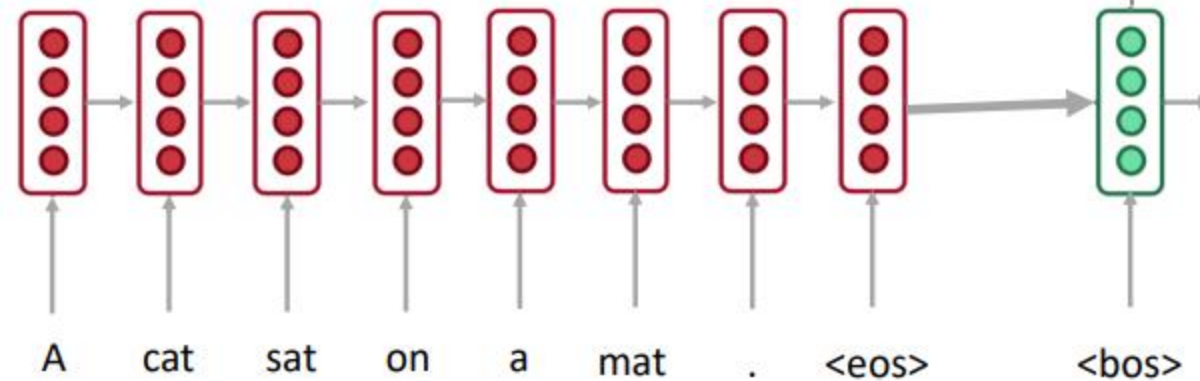
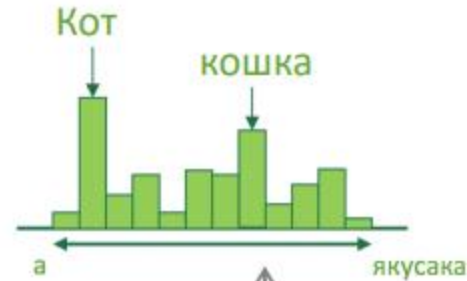


Закодированное исходное предложение.
Используем в качестве исходного
состояния для decoder RNN

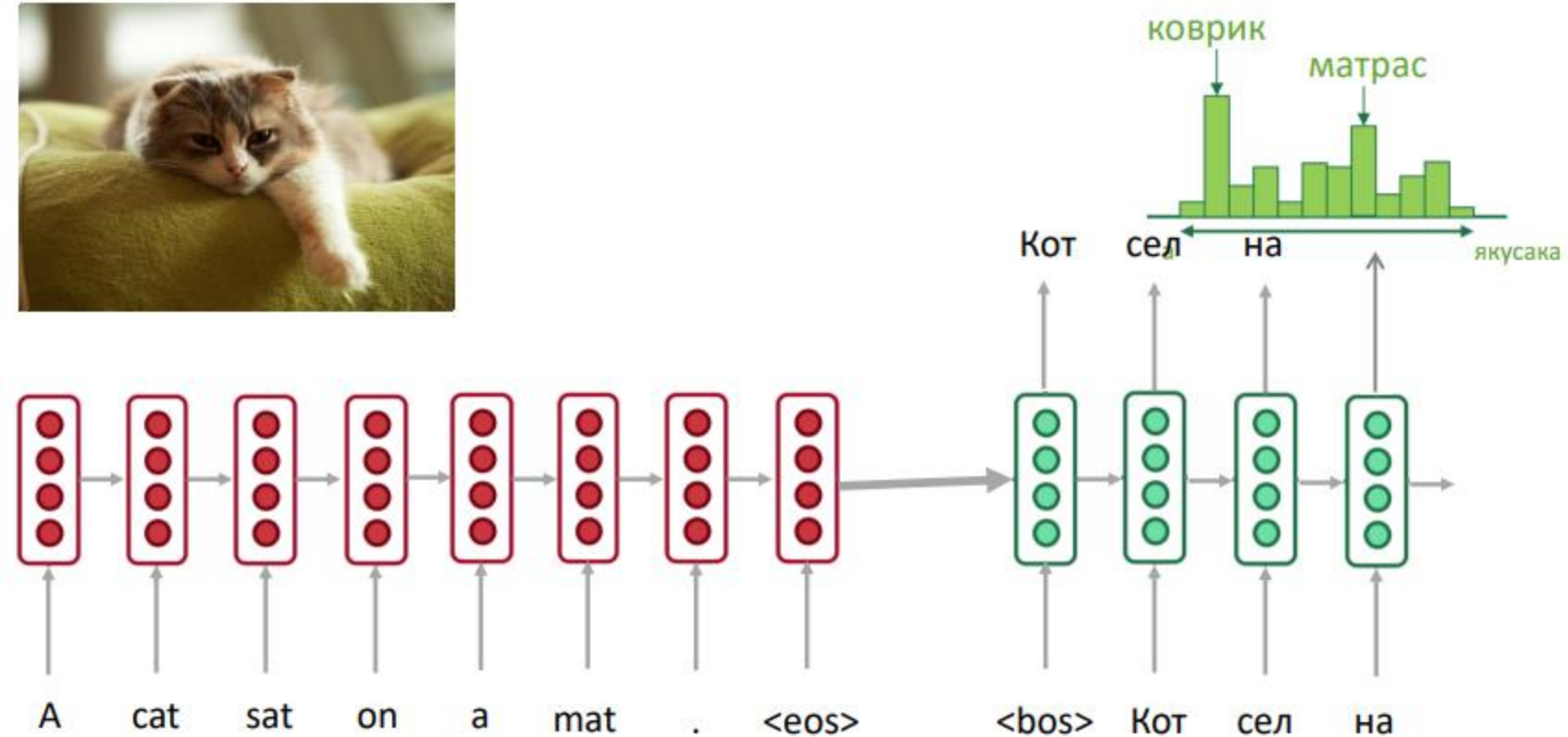
RNN Encoder-Decoder (Vanilla)



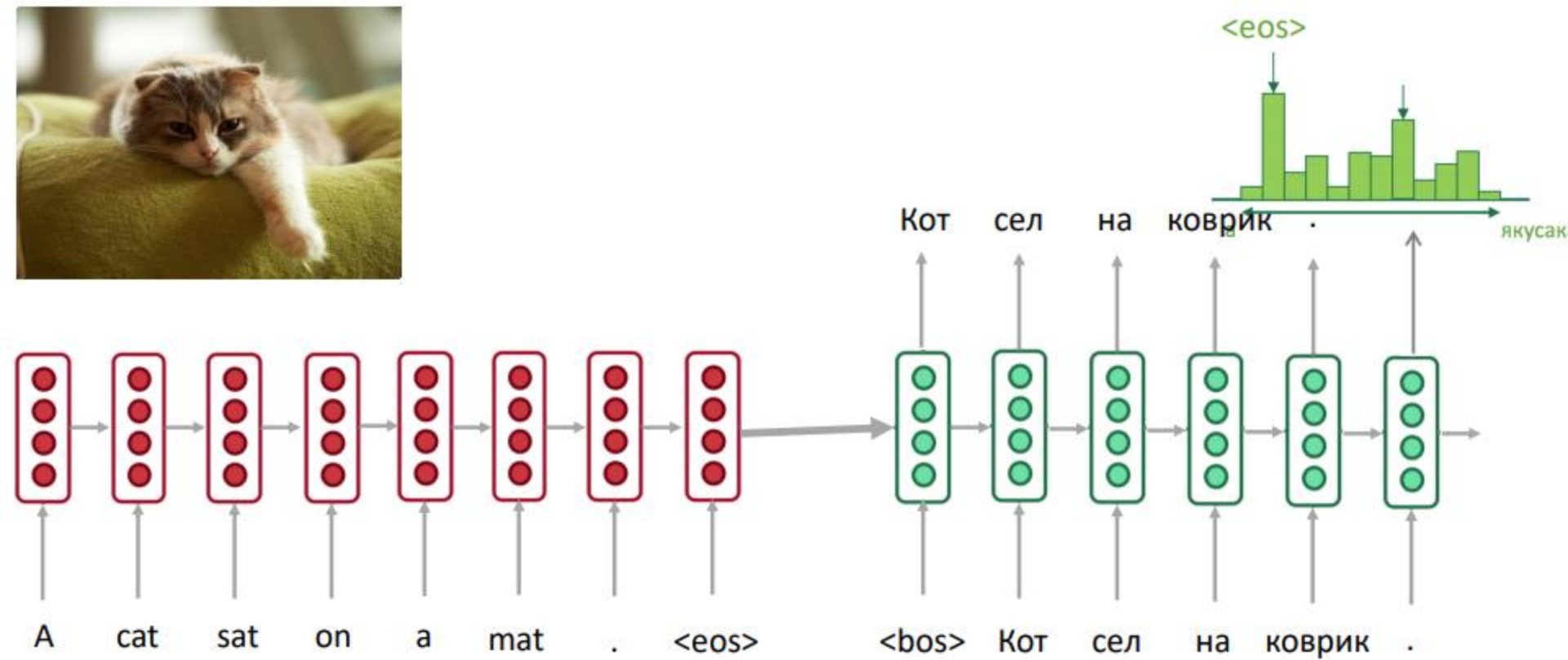
RNN Encoder-Decoder (Vanilla)



RNN Encoder-Decoder (Vanilla)



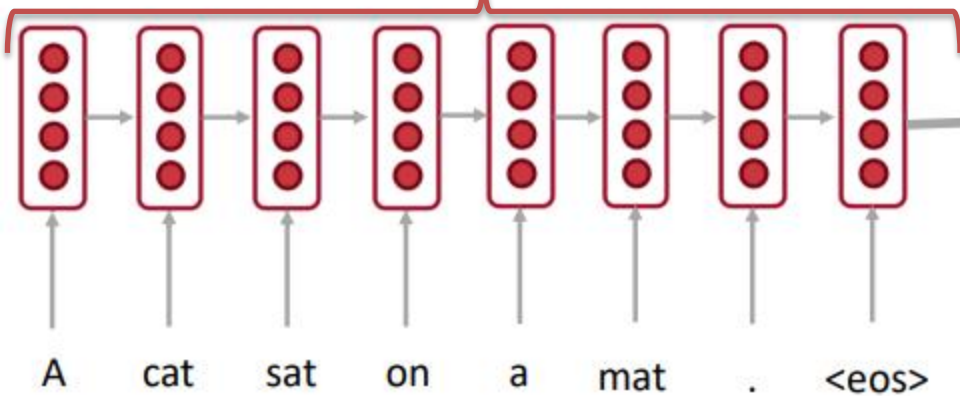
RNN Encoder-Decoder (Vanilla)



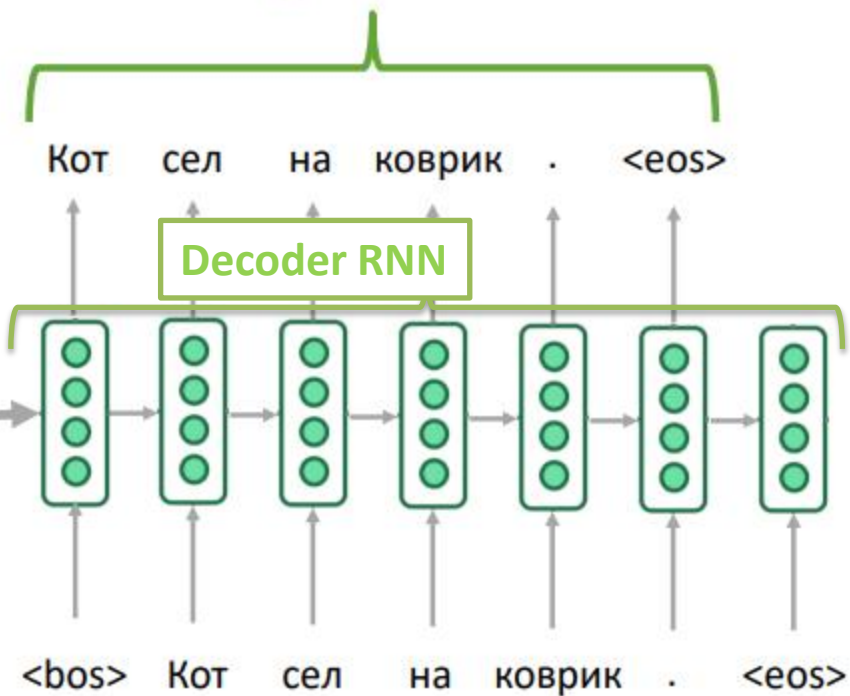
RNN Encoder-Decoder (Vanilla)



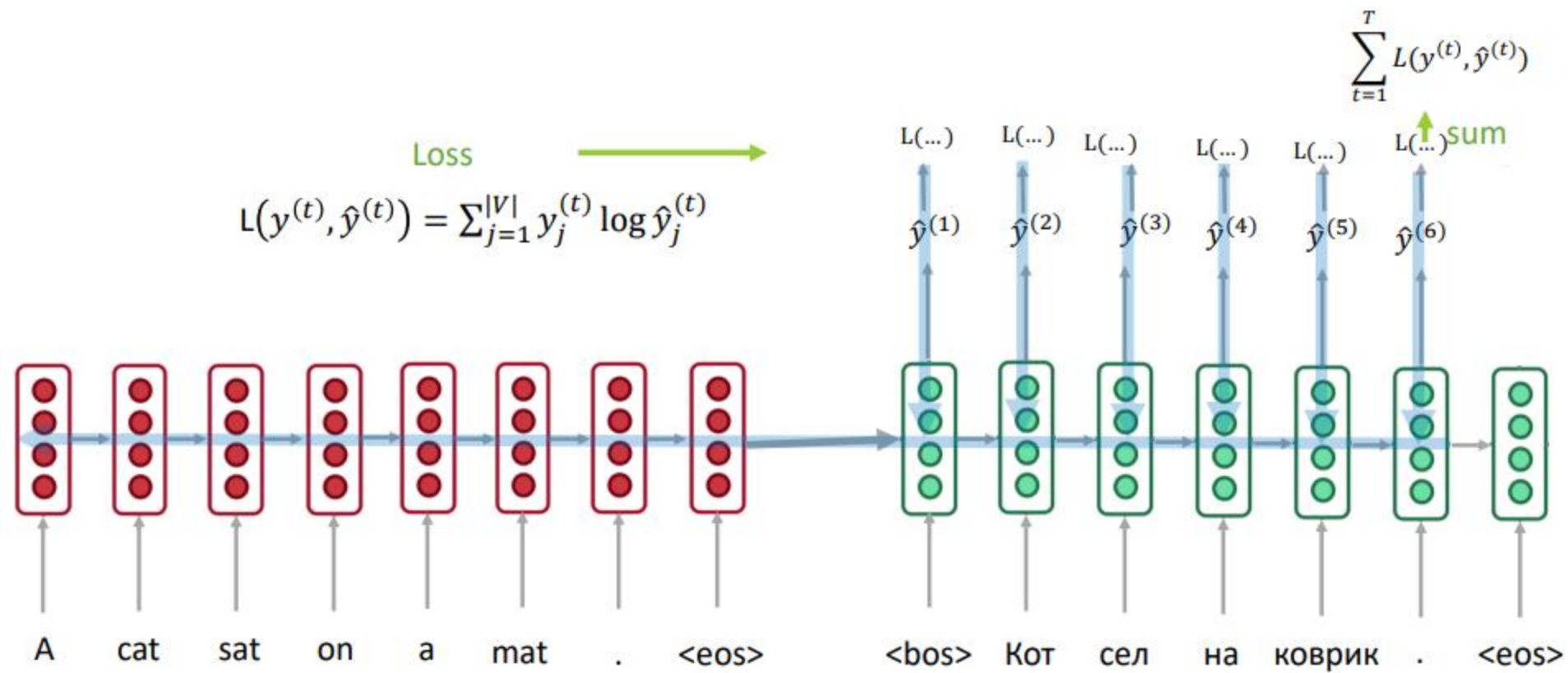
Encoder RNN



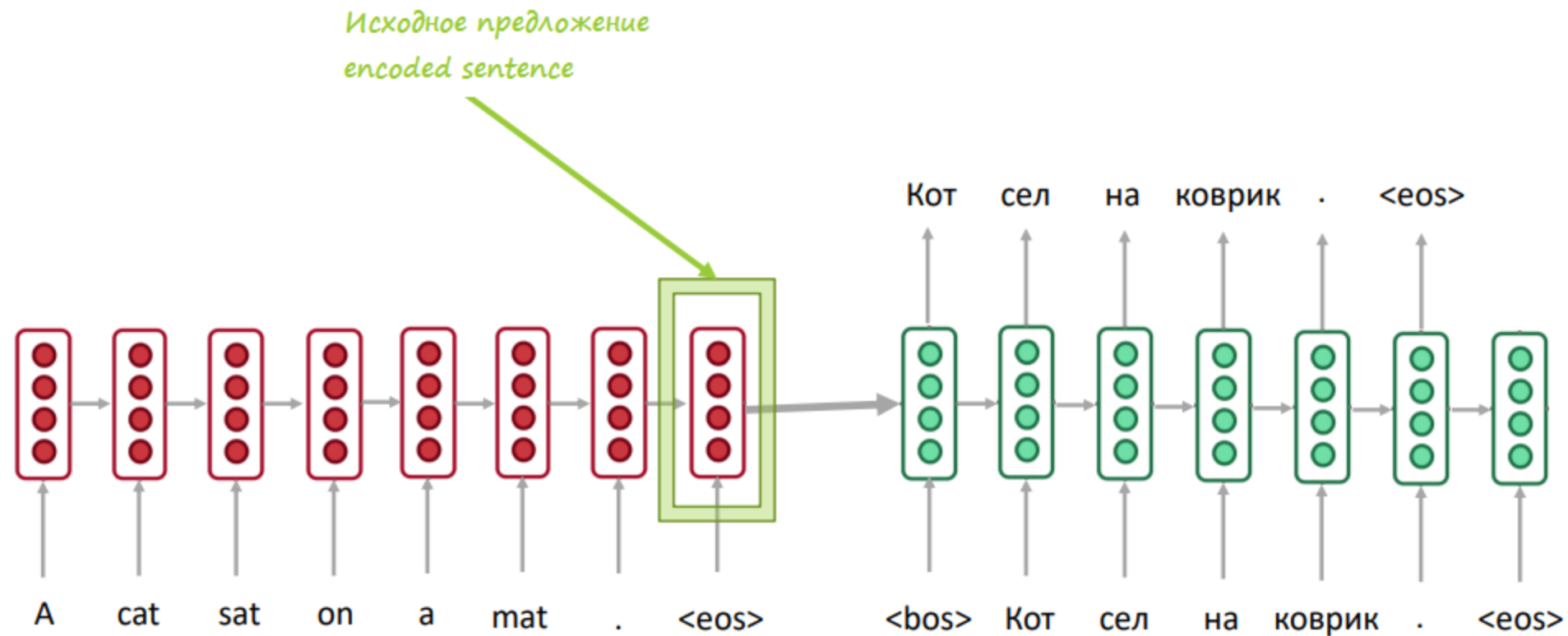
Target sentence



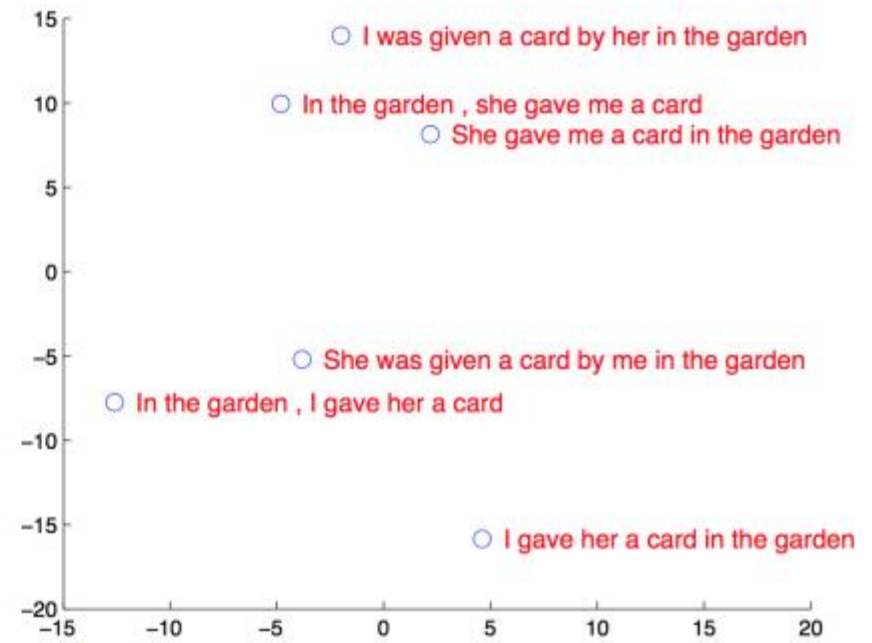
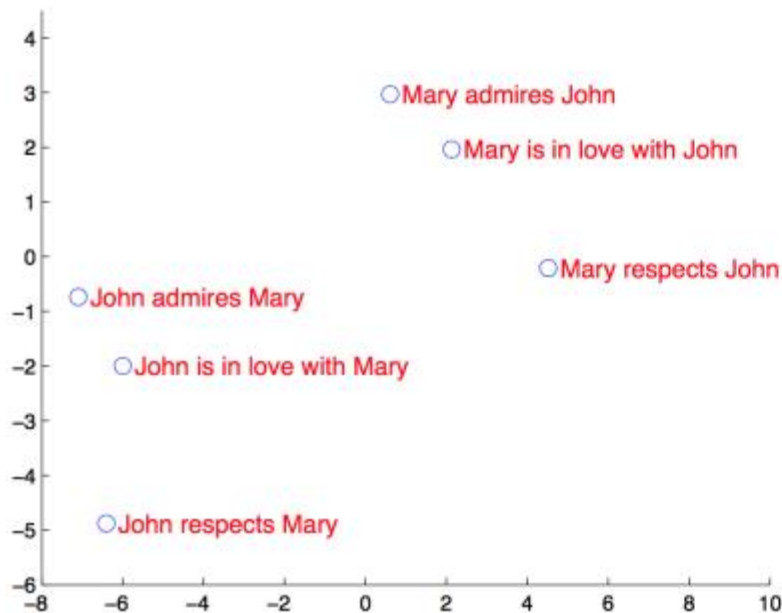
RNN Encoder-Decoder (Vanilla)



Посмотрим на эмбединги предложения после encoder



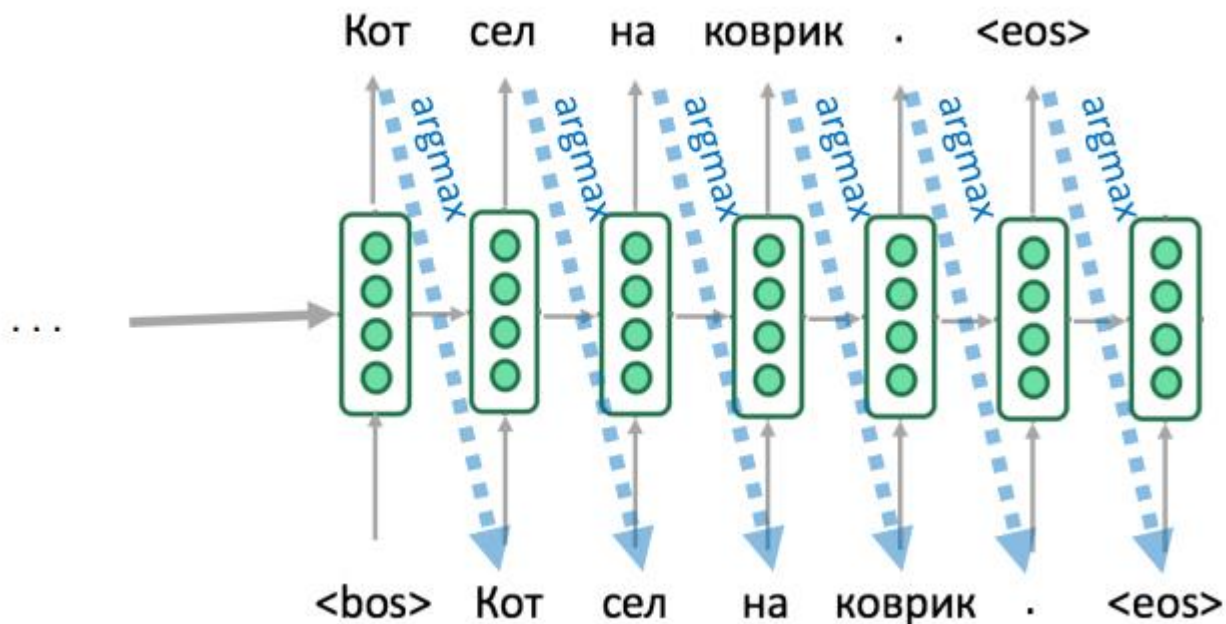
Посмотрим на эмбединги предложения после encoder



Decoding: Как генерировать последовательность?

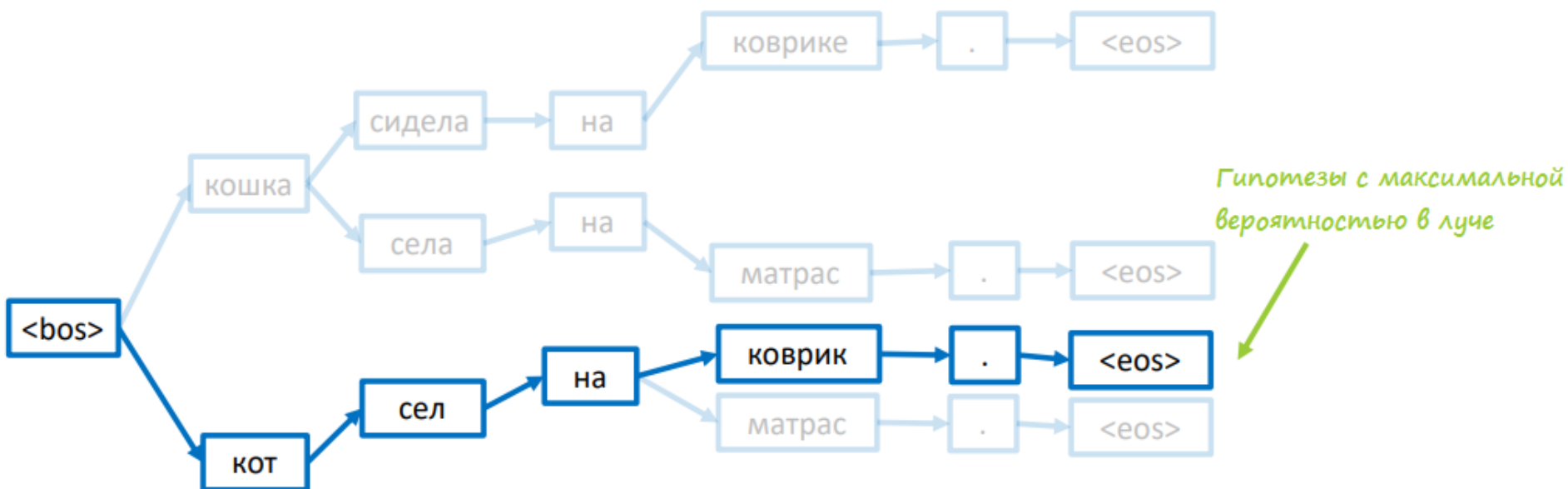
Decoding: Как генерировать последовательность?

- Greedy decoding

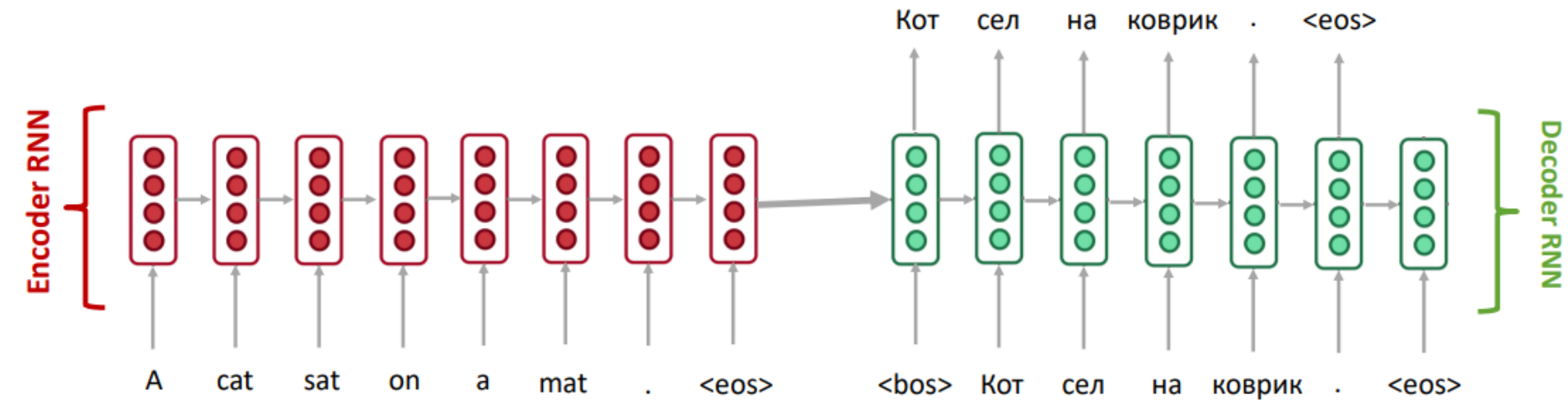


Decoding: Как генерировать последовательность?

- Beam search

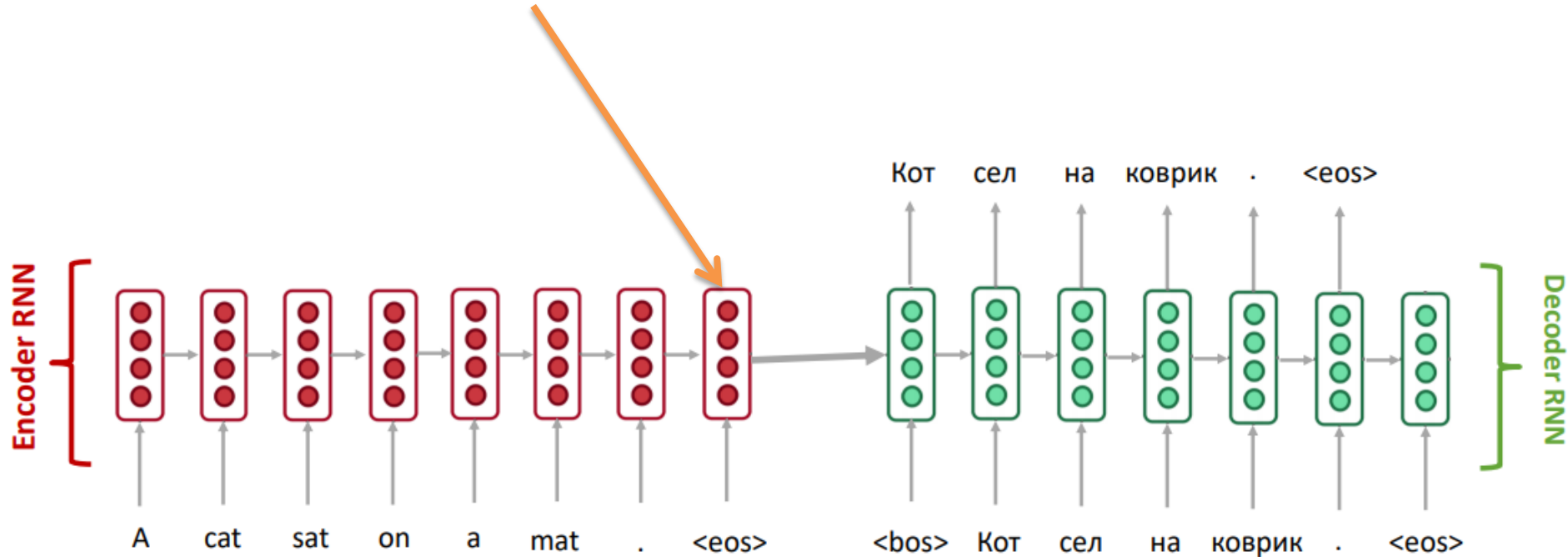


Какие проблемы с обычными RNN Encoder-Decoder моделями?

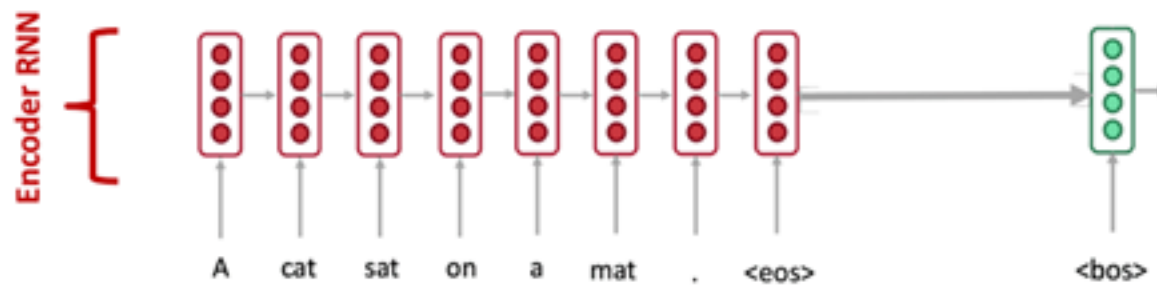


Какие проблемы с обычными RNN Encoder-Decoder моделями?

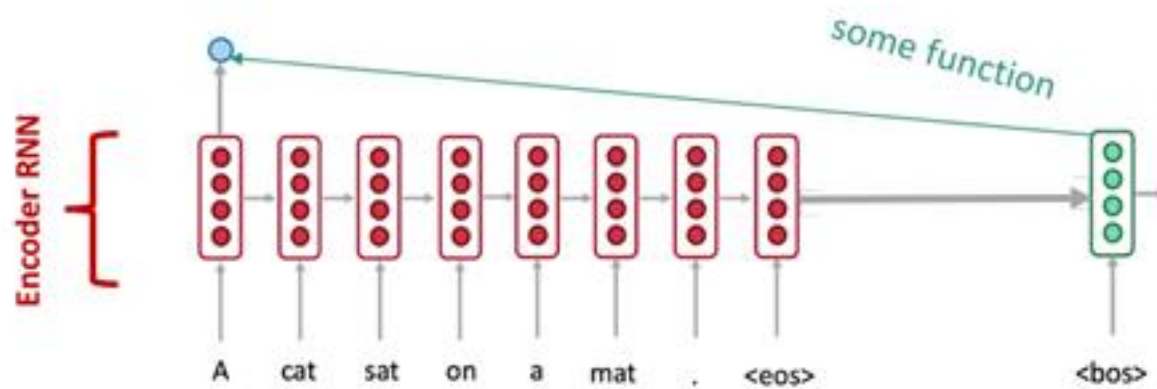
Information bottleneck:
модель должна помещать всю информацию о предложении в один вектор



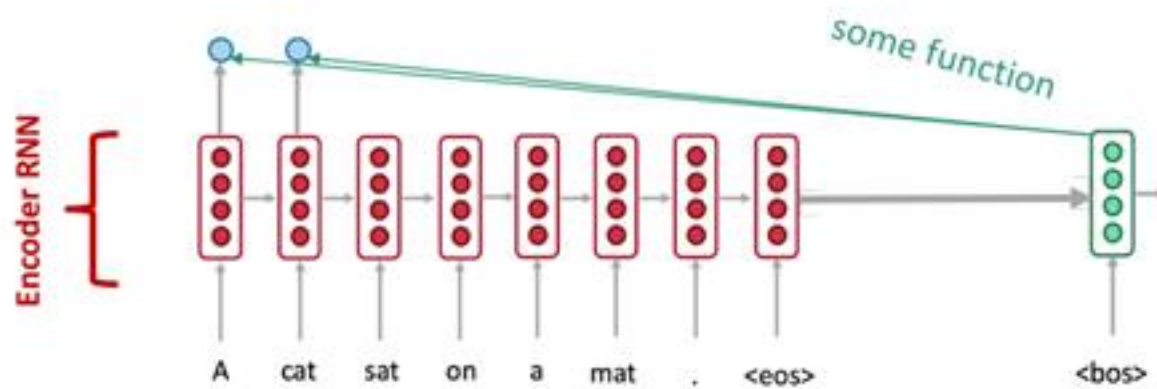
Attention



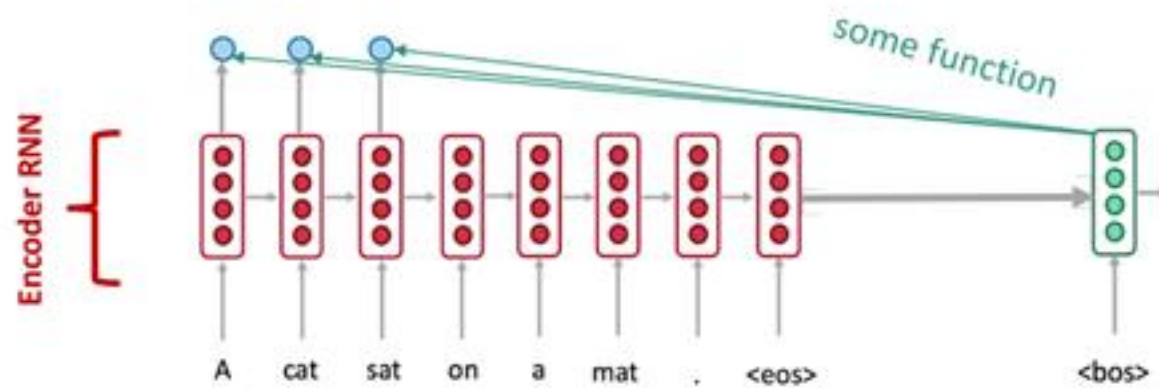
Attention



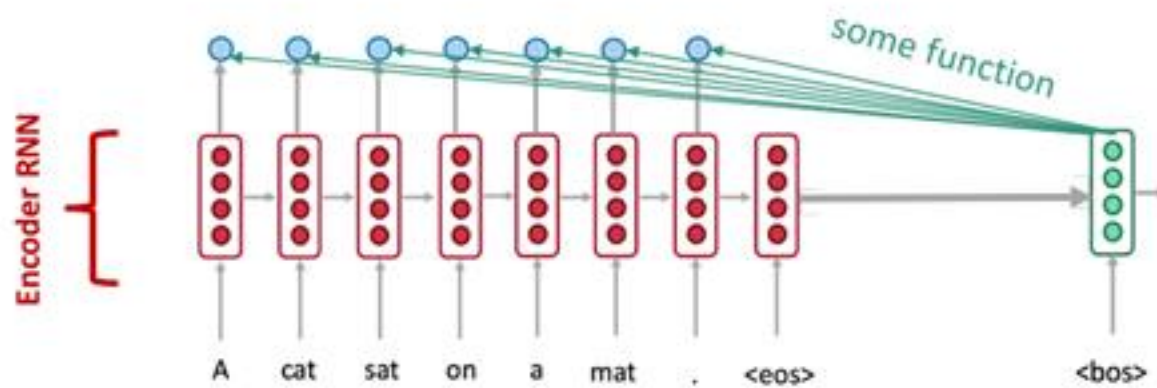
Attention



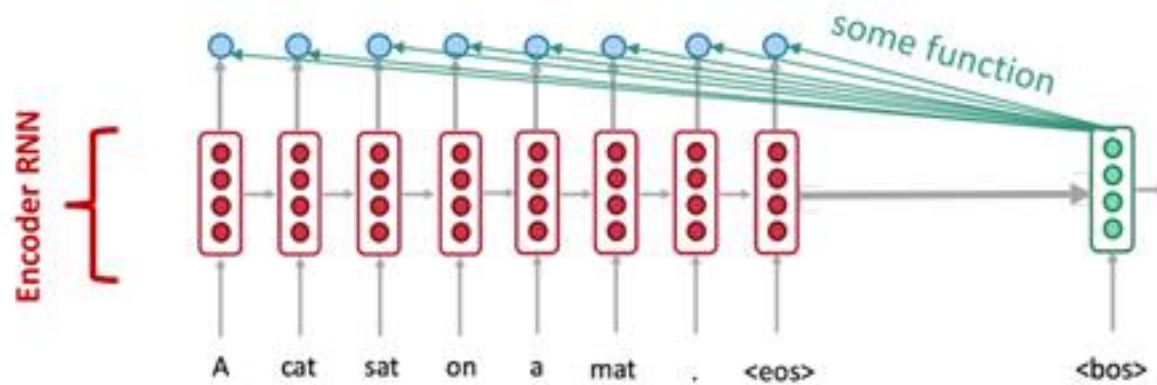
Attention



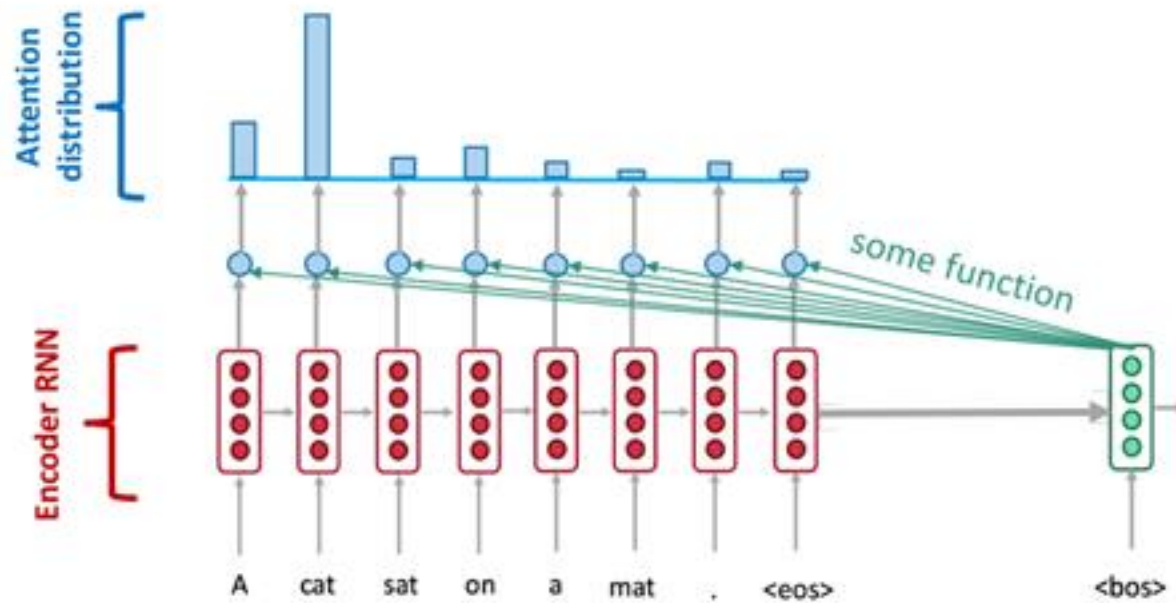
Attention



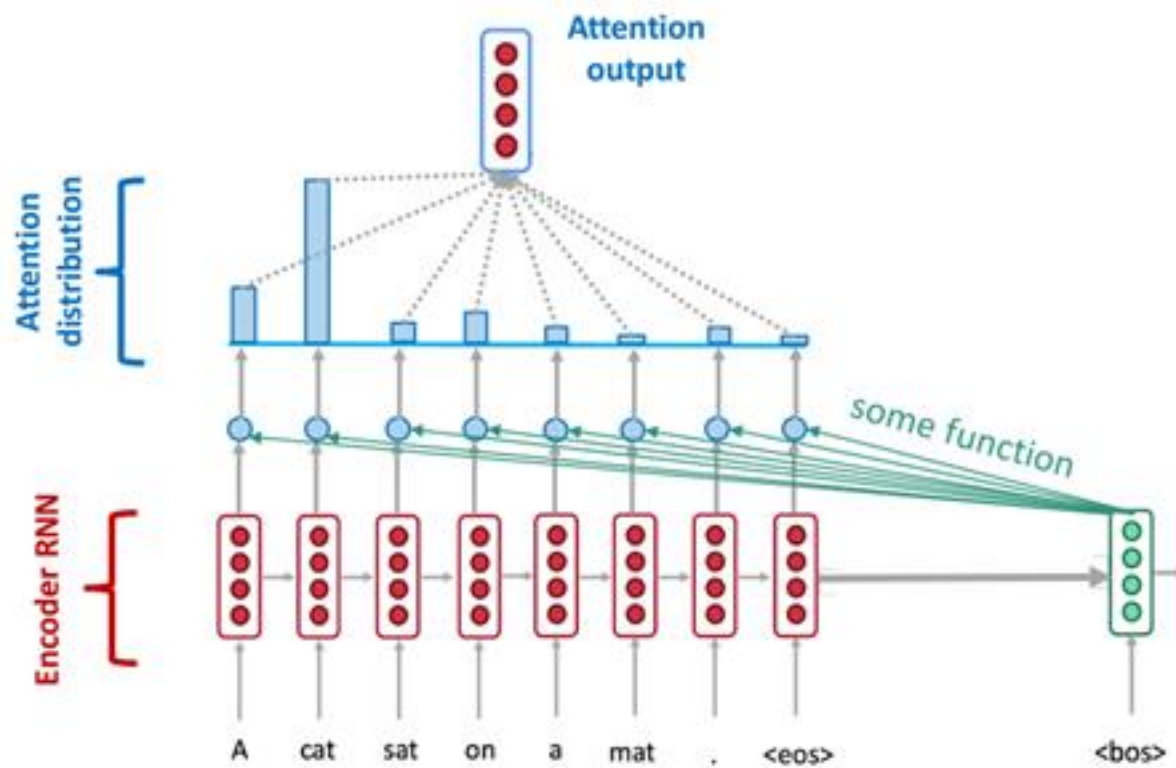
Attention



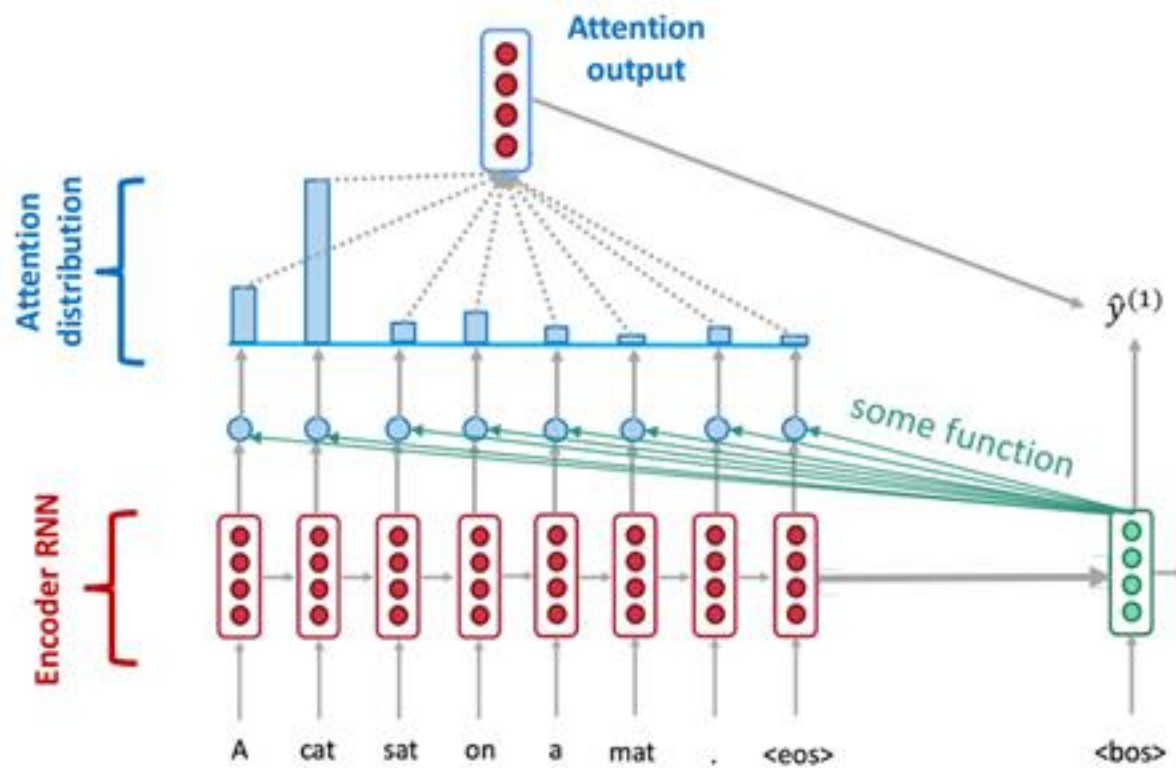
Attention



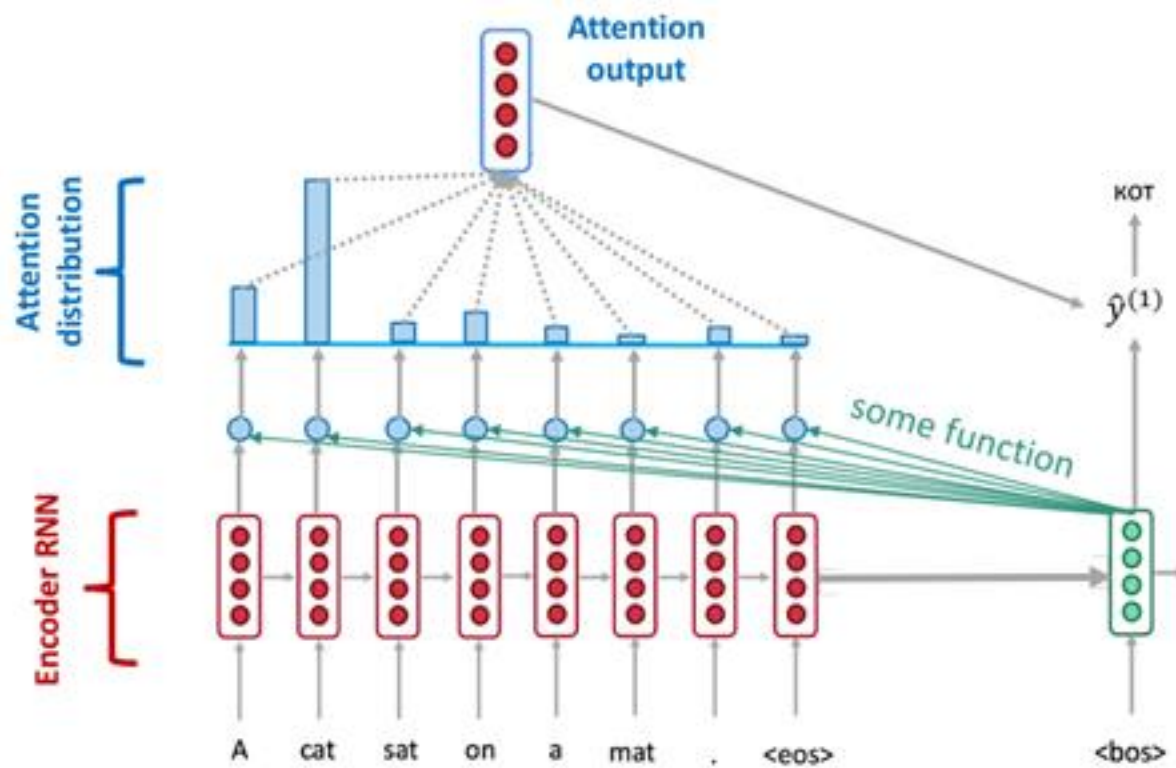
Attention



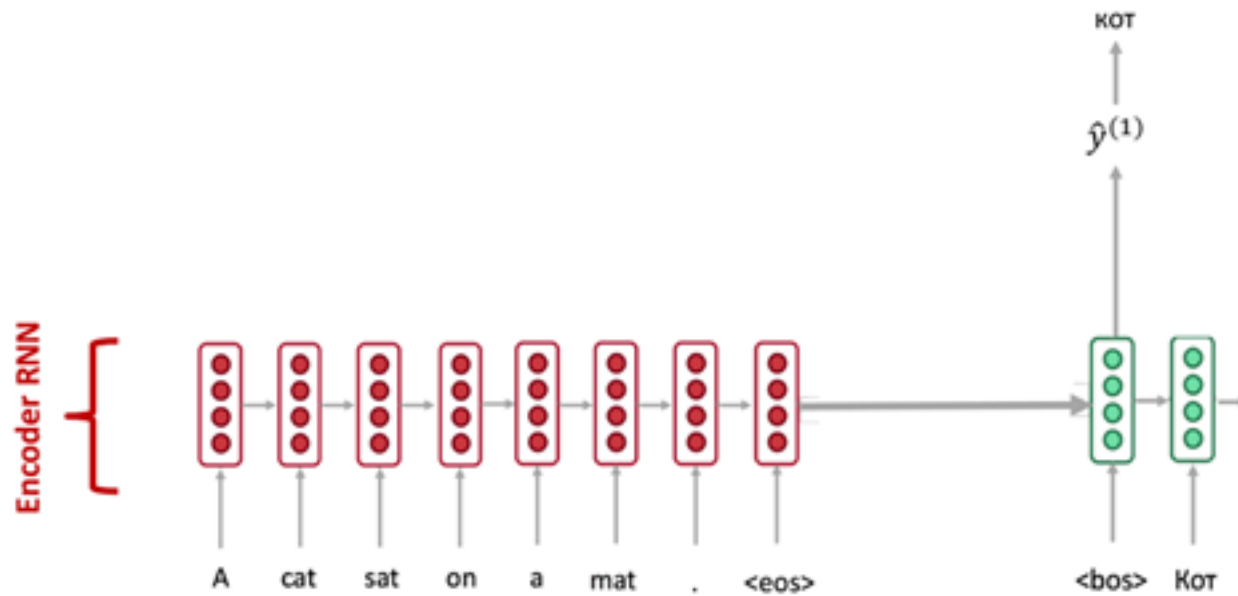
Attention



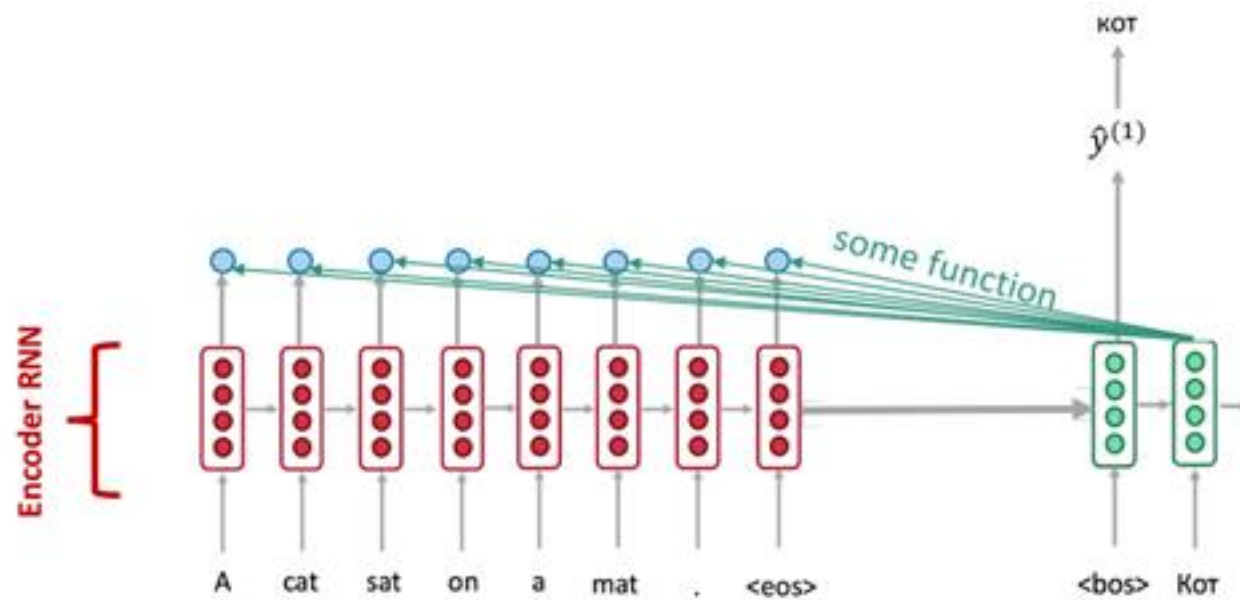
Attention



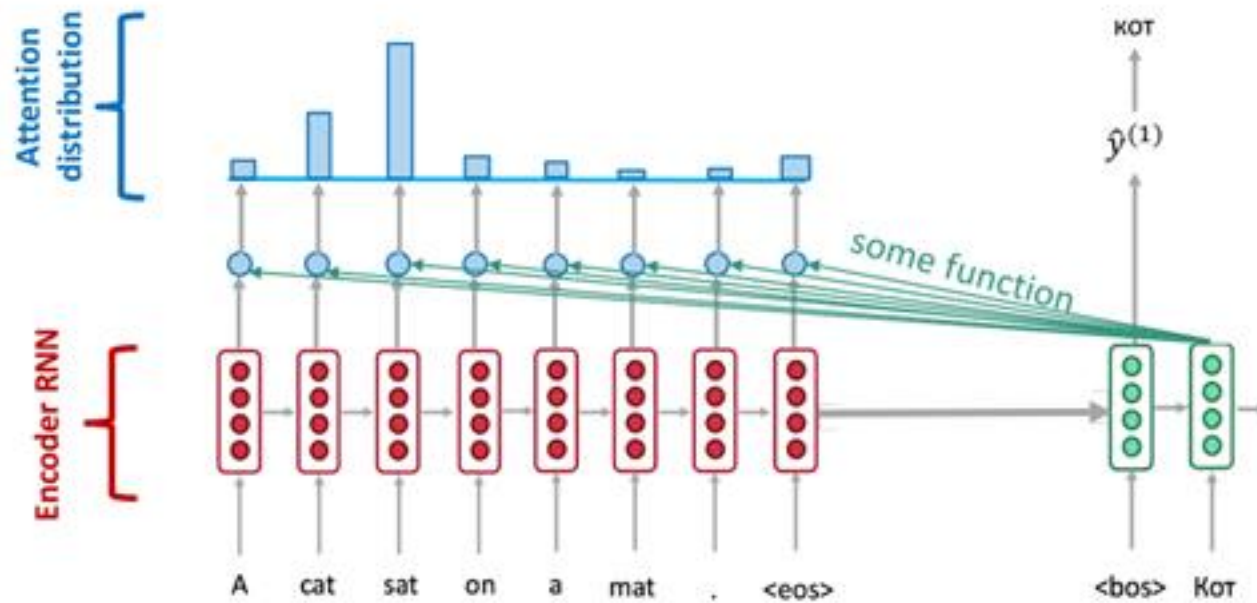
Attention



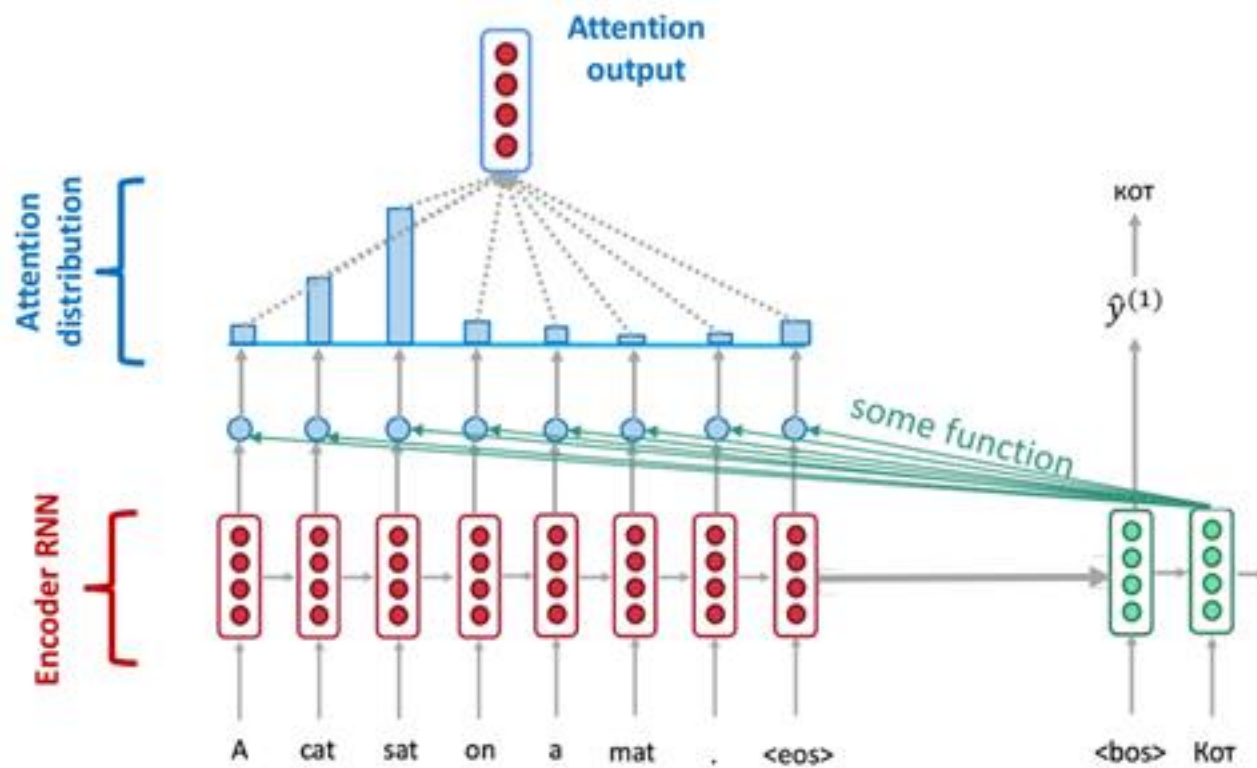
Attention



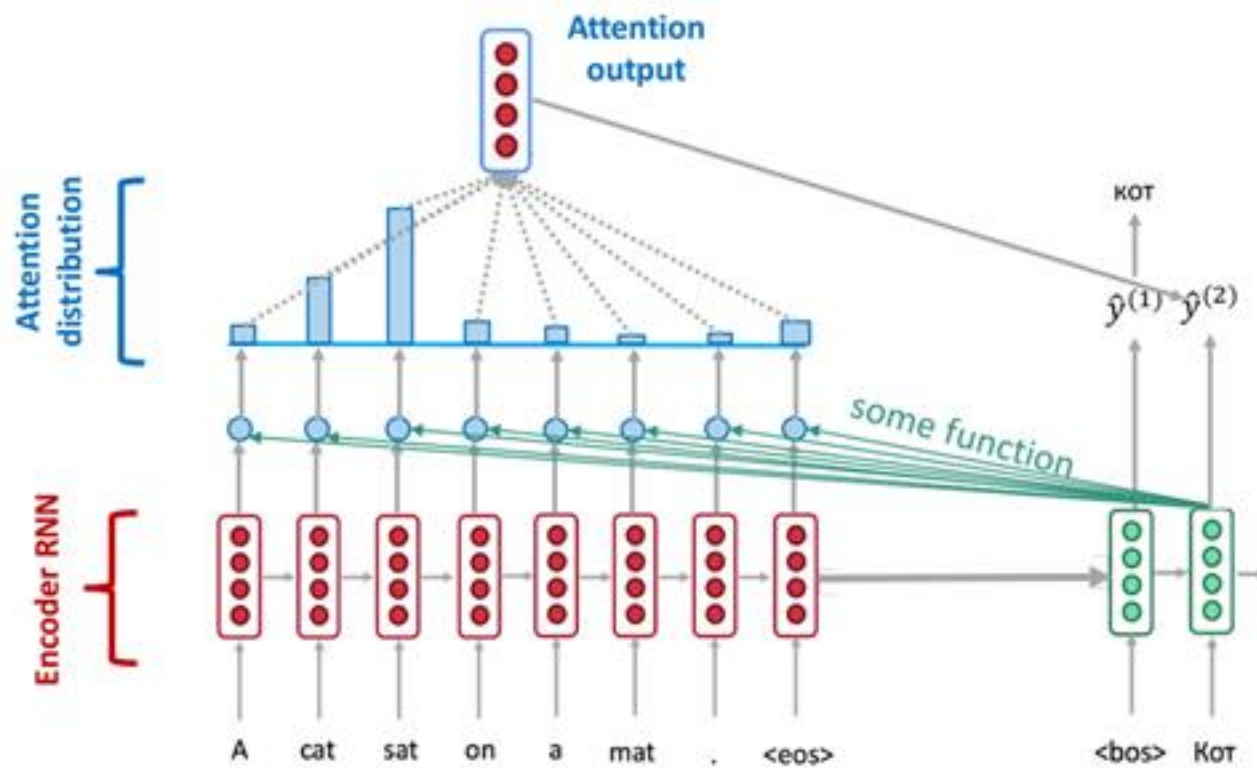
Attention



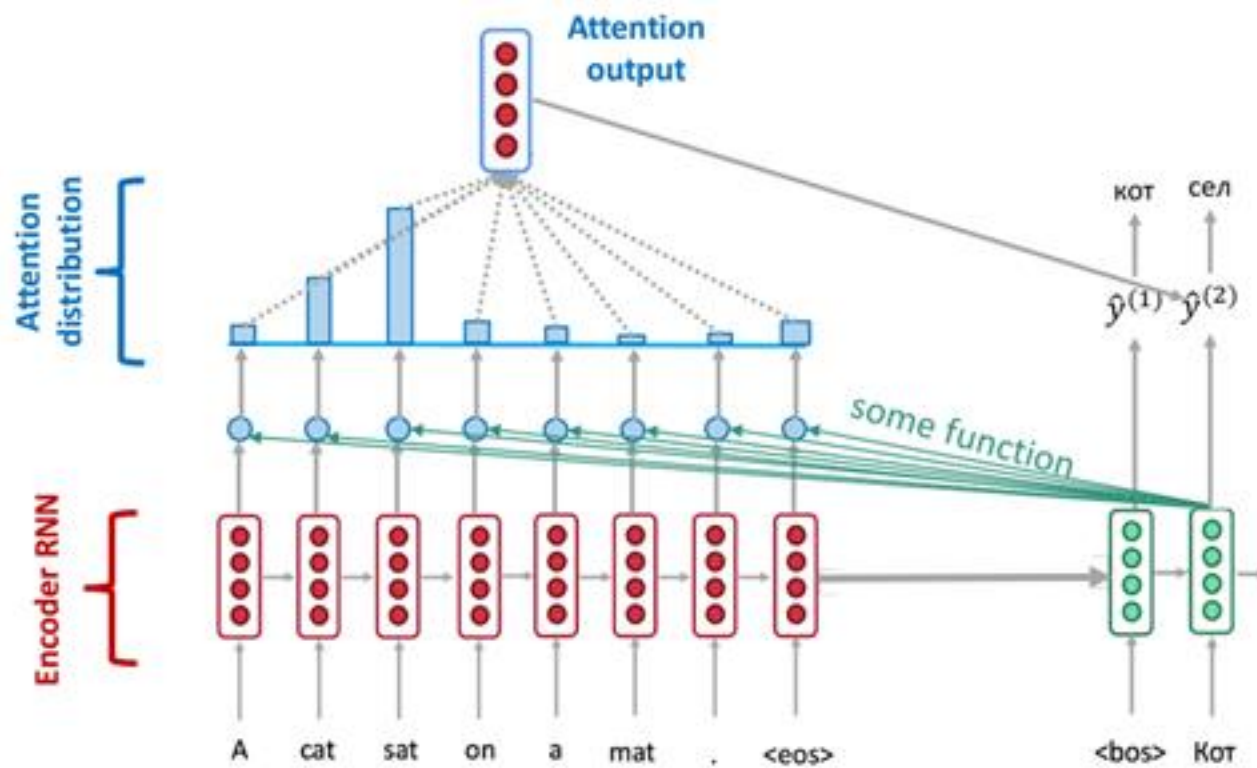
Attention



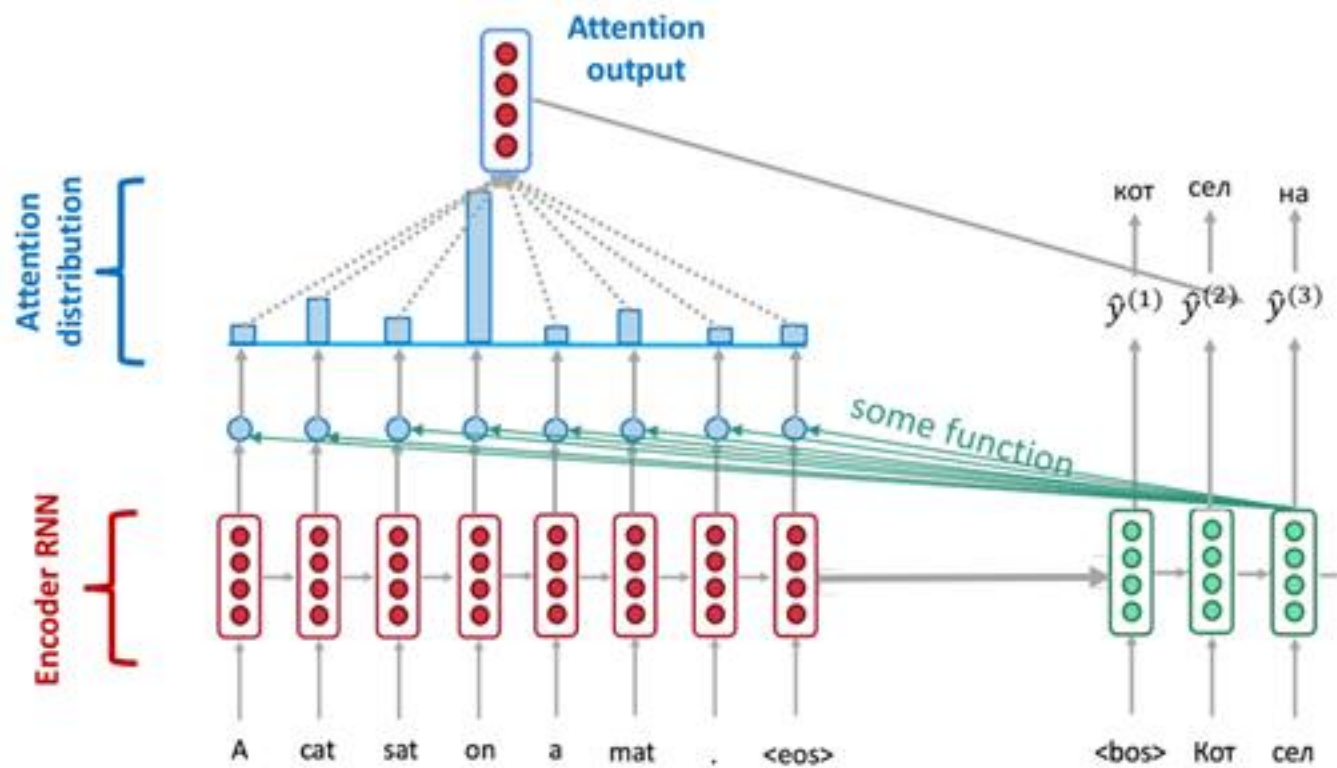
Attention



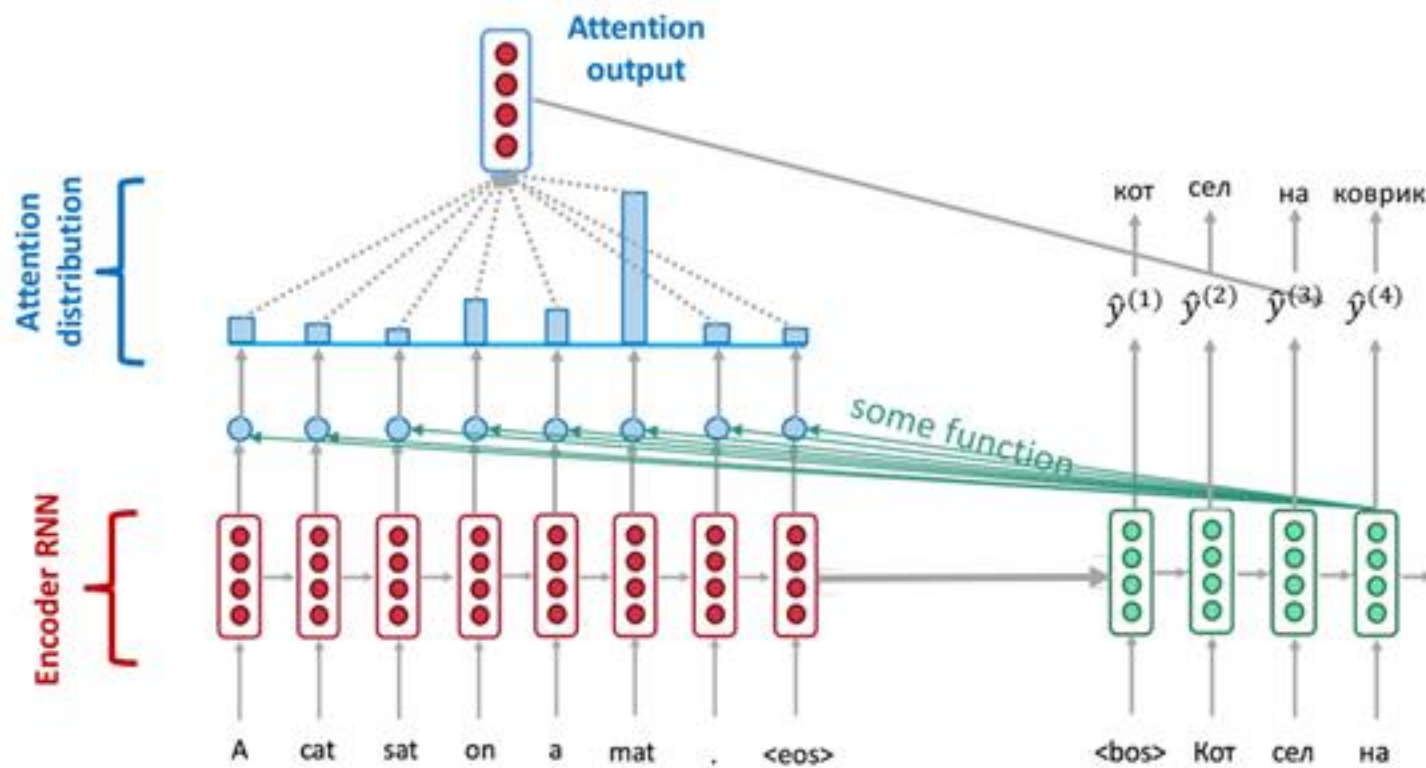
Attention



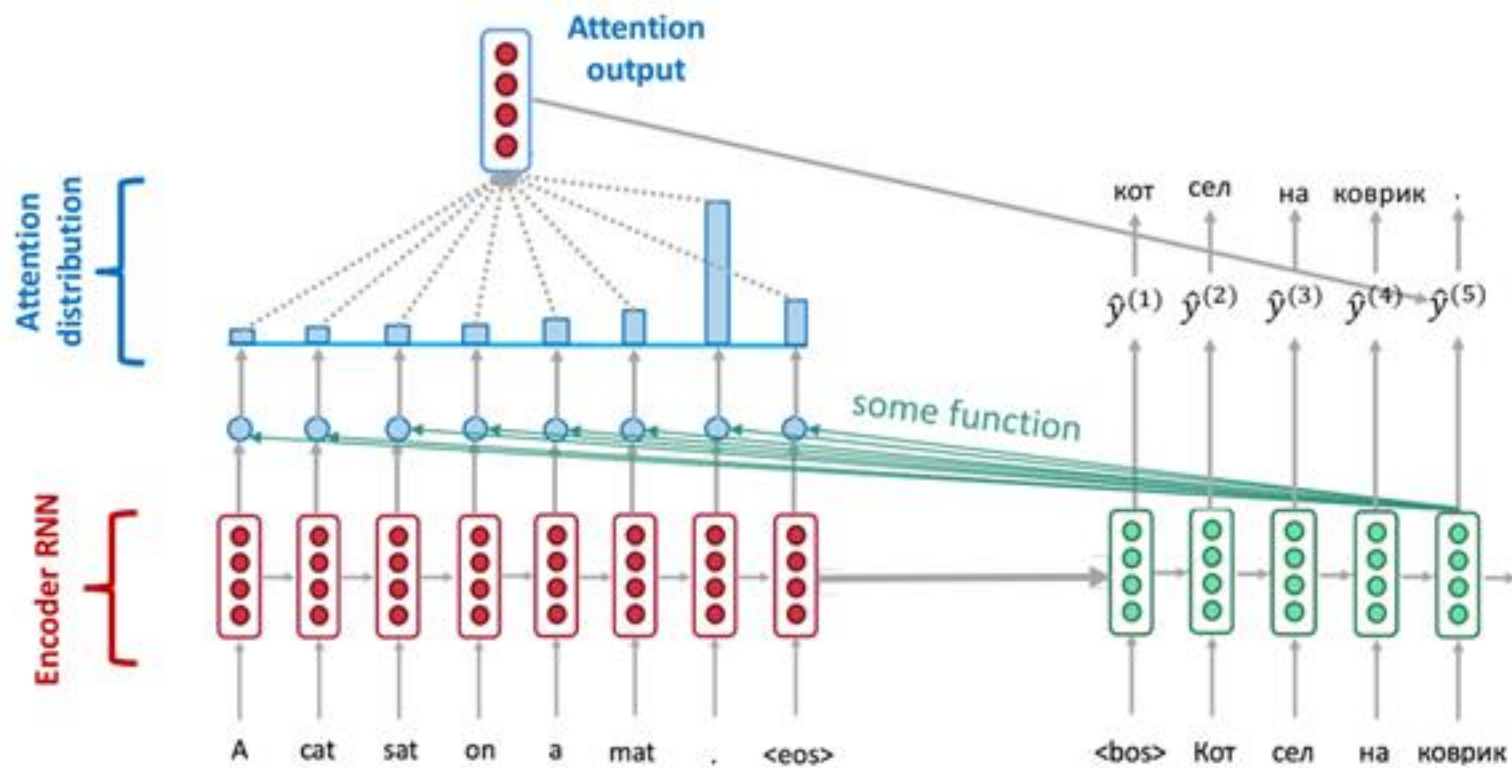
Attention



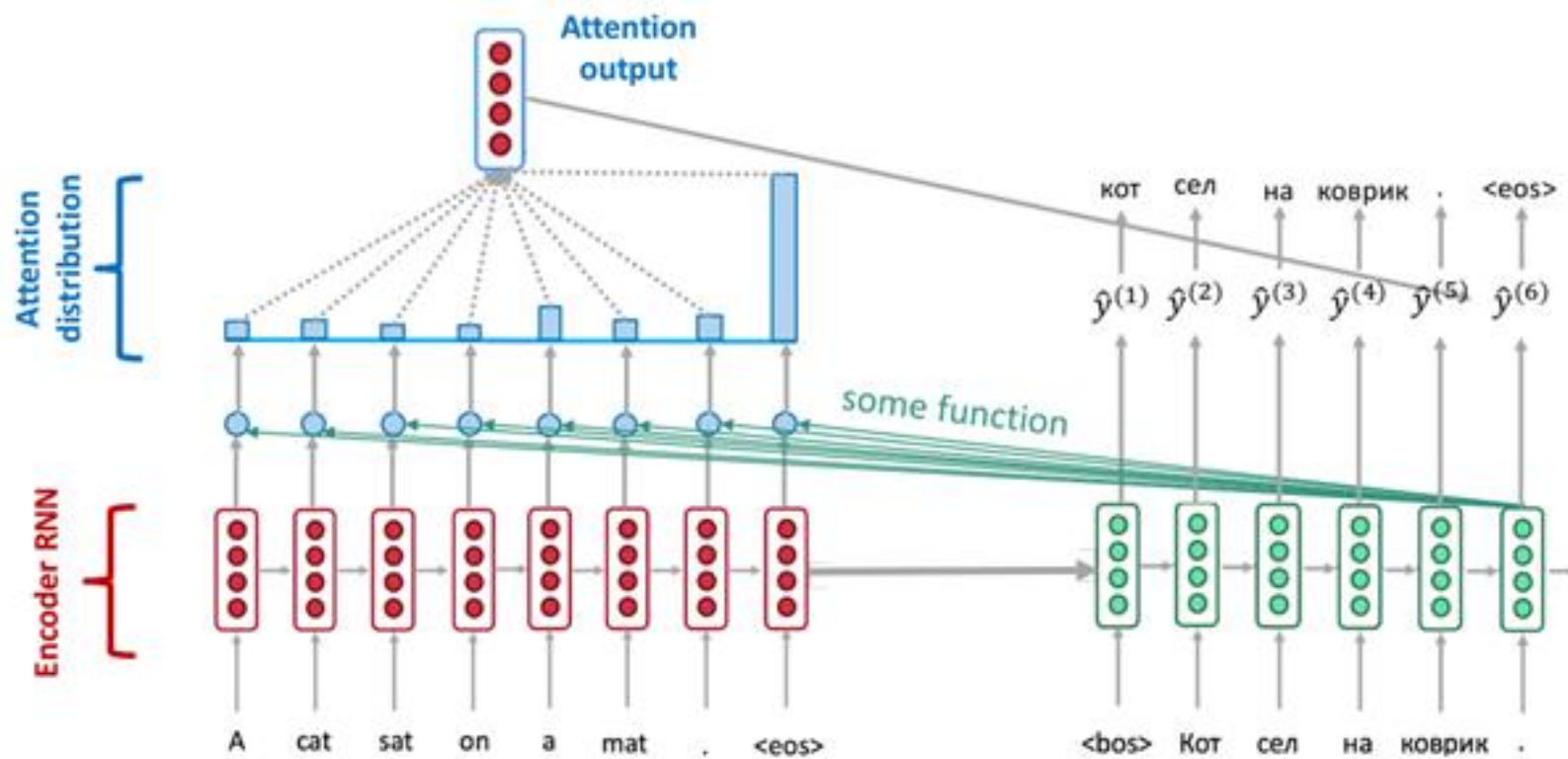
Attention



Attention

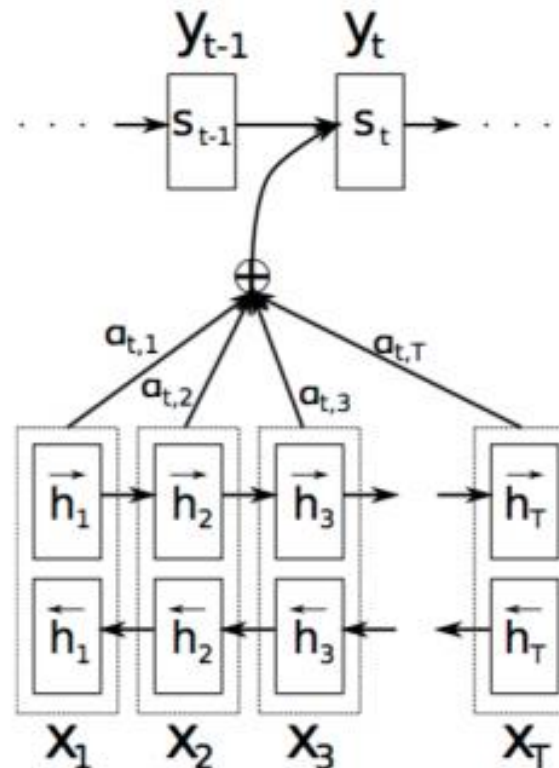


Attention

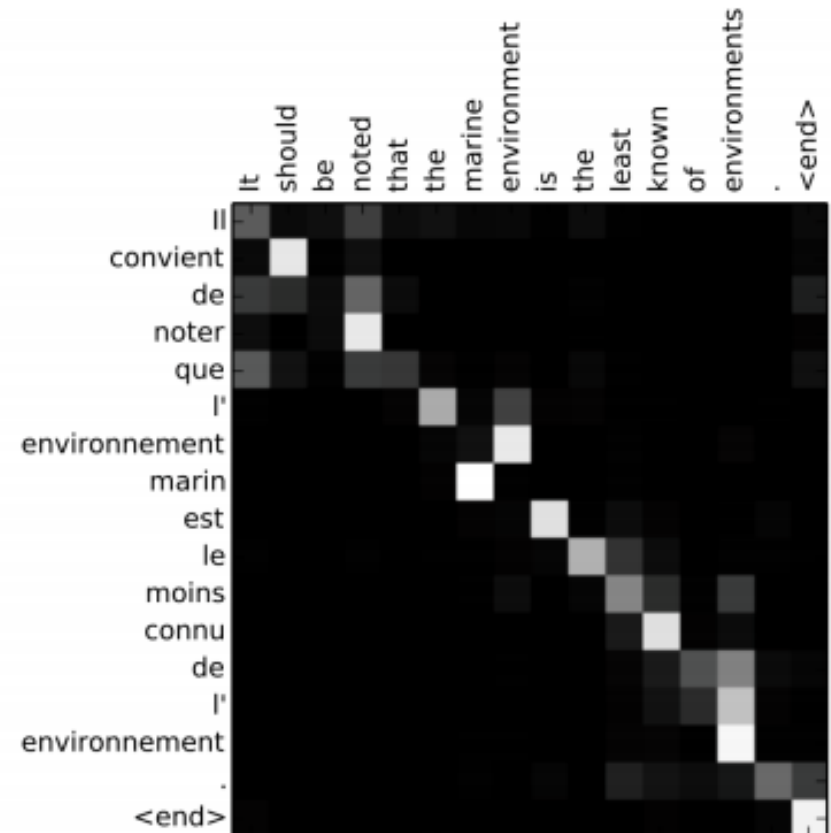
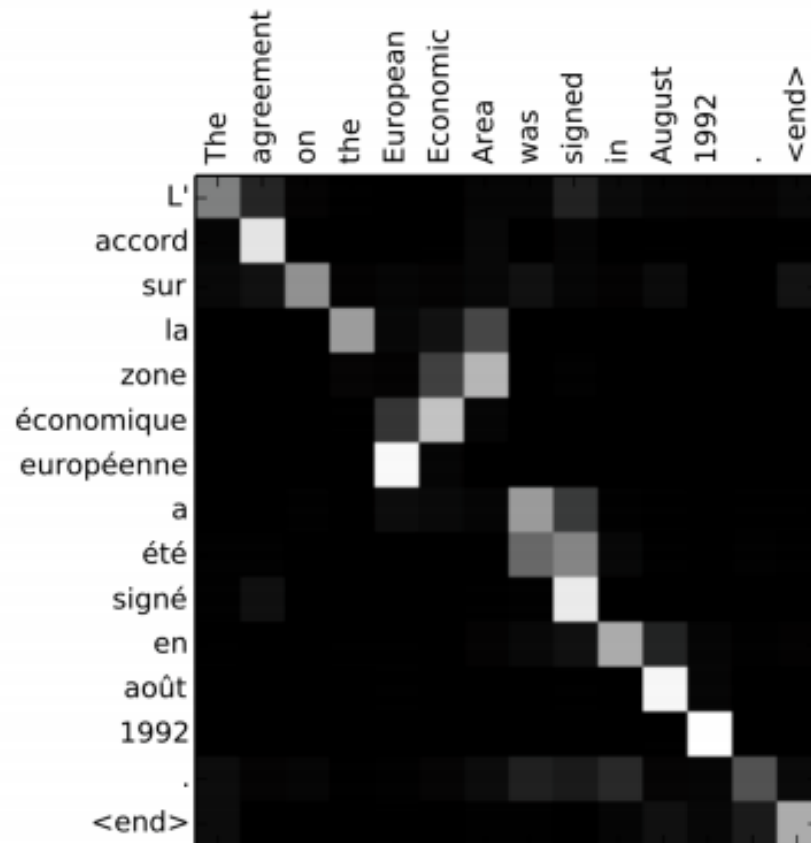


Neural Machine Translation by Jointly Learn to Align and Translate

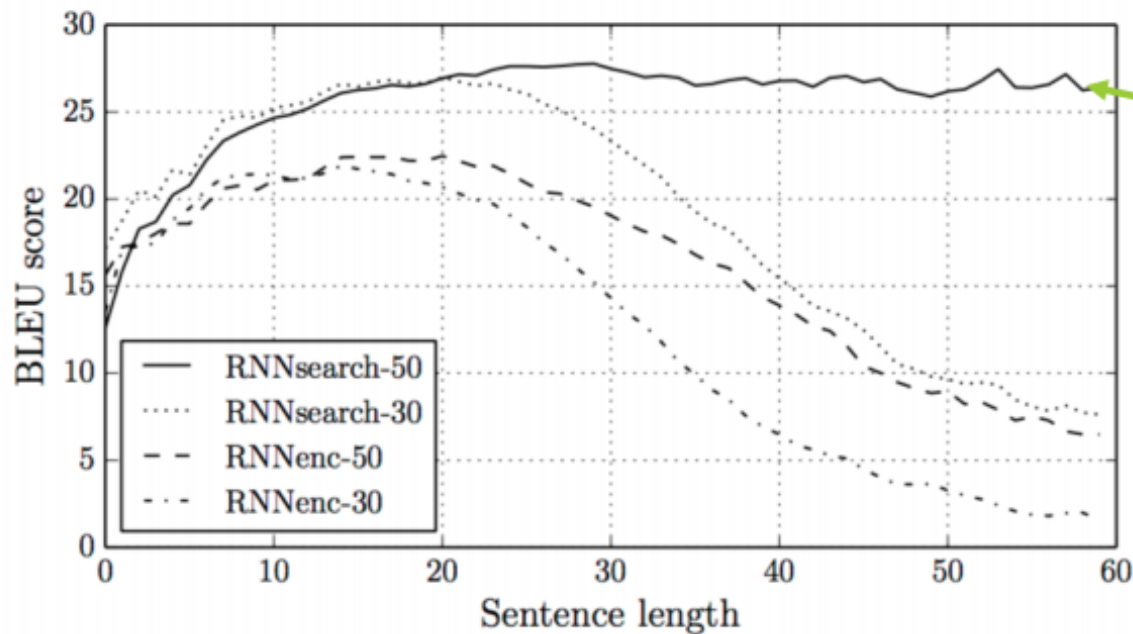
- Преимущества внимания:
 - Не нужно помнить все предложение
 - Нет фиксированного размера представления
 - «Похоже на людей»



Neural Machine Translation by Jointly Learn to Align and Translate



Neural Machine Translation by Jointly Learn to Align and Translate



Нет потери качества для
длинных предложений

Как вычислять Attention?

- dot product: $f_{att}(h_i, s_j) = h_i^T s_j$
- bilinear attention: $f_{att}(h_i, s_j) = h_i^T W_a s_j$
- multi-layer perceptron: $f_{att}(h_i, s_j) = v_a \tanh(W_a [s_j; h_i^T])$
- любая другая функция, которую можете себе представить ☺:

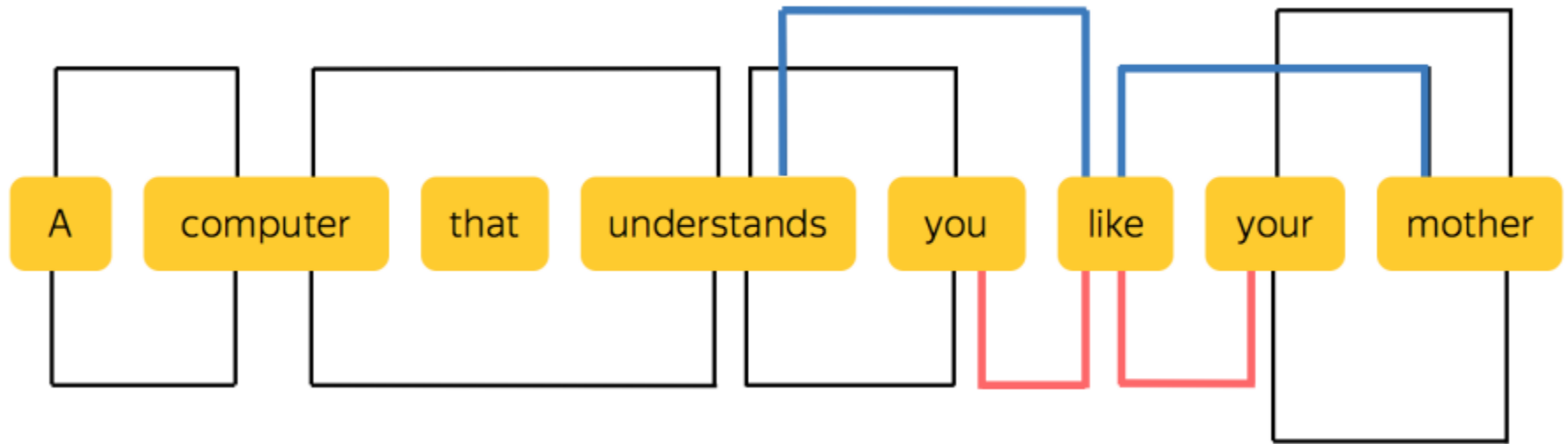
$$z(c, m, q) = [c, m, q, c \circ q, c \circ m, |c - q|, |c - m|, c^T W^{(b)} q, c^T W^{(b)} m]$$

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

A bit crazy, huh?

Transformer: Attention is all you need!

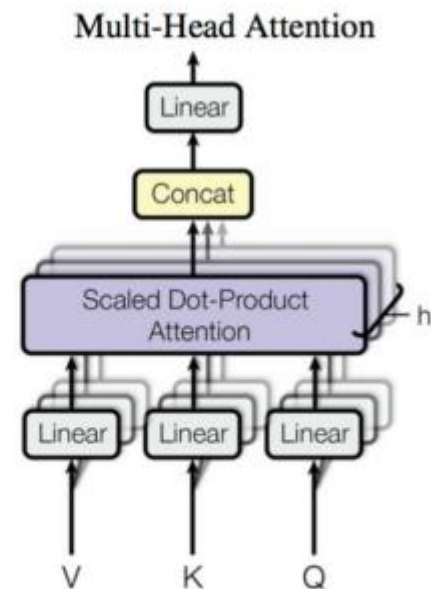
Self Attention



Multi-Head Attention

Она руководит **новым** проектом

- Gender agreement
- Case government
- Lexical preferences
- ...



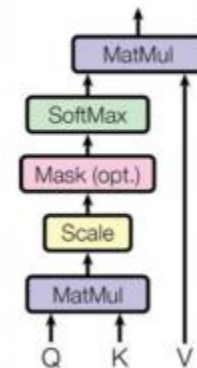
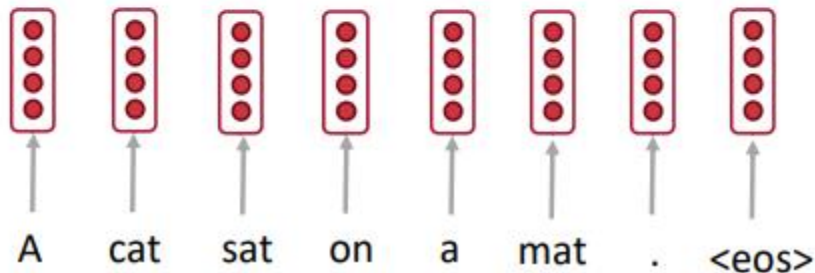
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

Self Attention

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

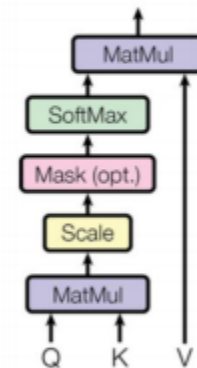
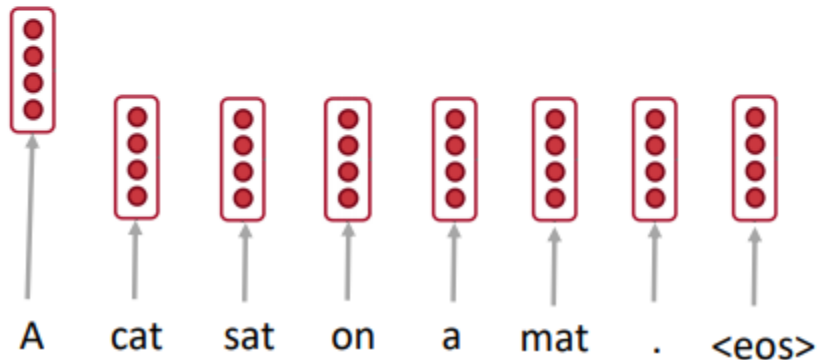
Scaled Dot-Product Attention



Self Attention

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

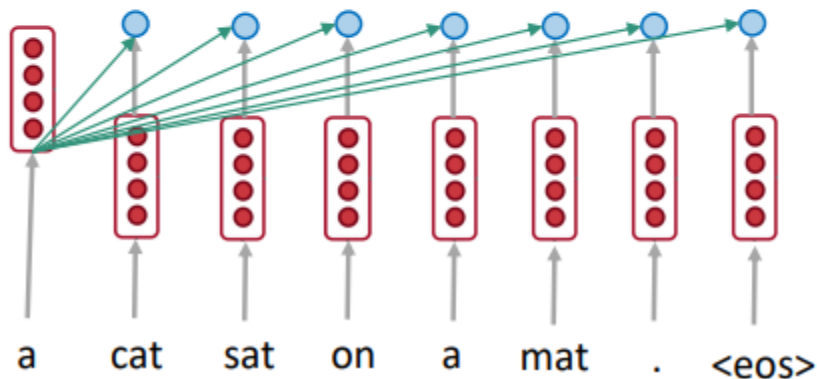
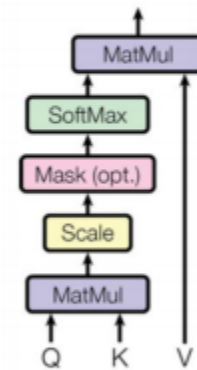
Scaled Dot-Product Attention



Self Attention

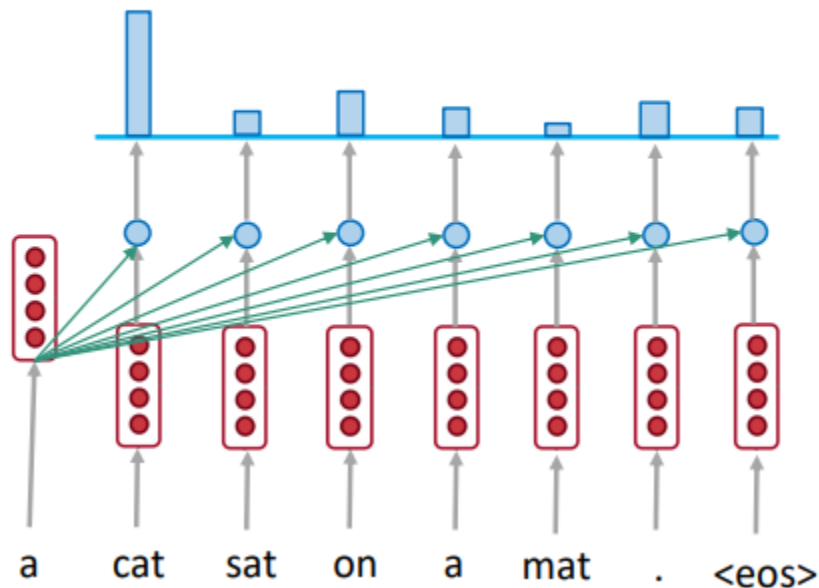
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention

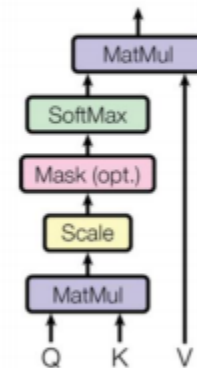


Self Attention

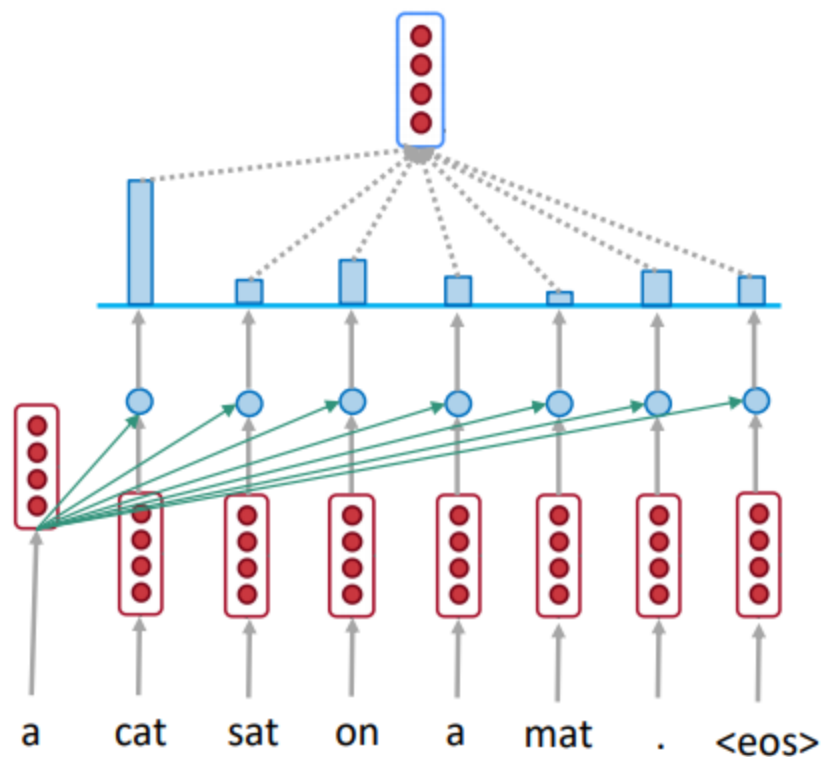
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Scaled Dot-Product Attention

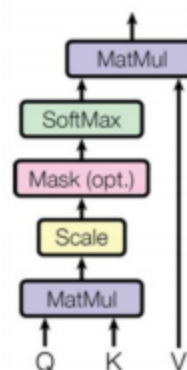


Self Attention



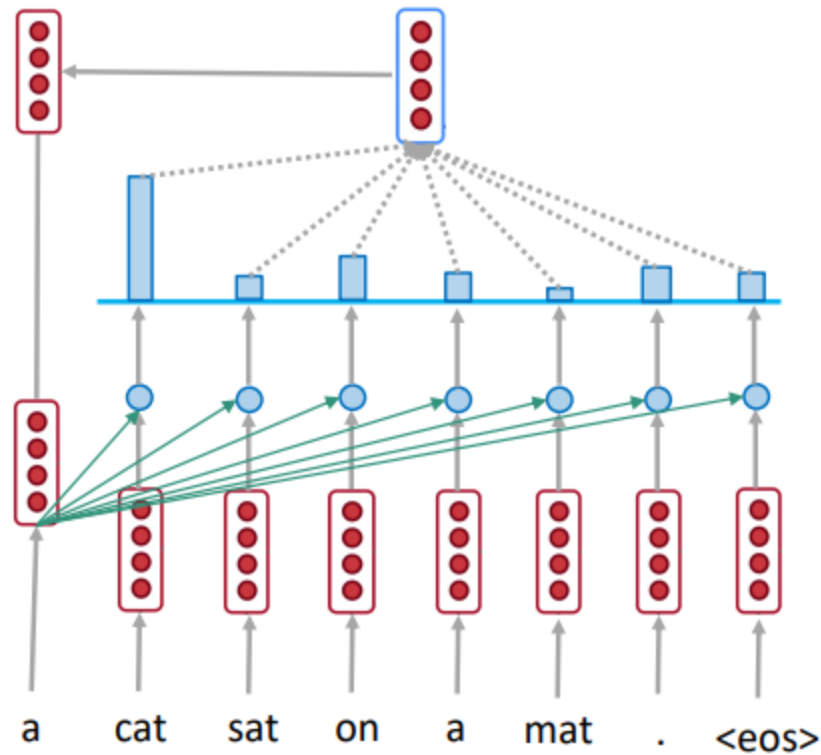
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention



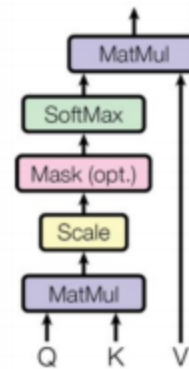
Self Attention

- Обновляем представление слова «a»



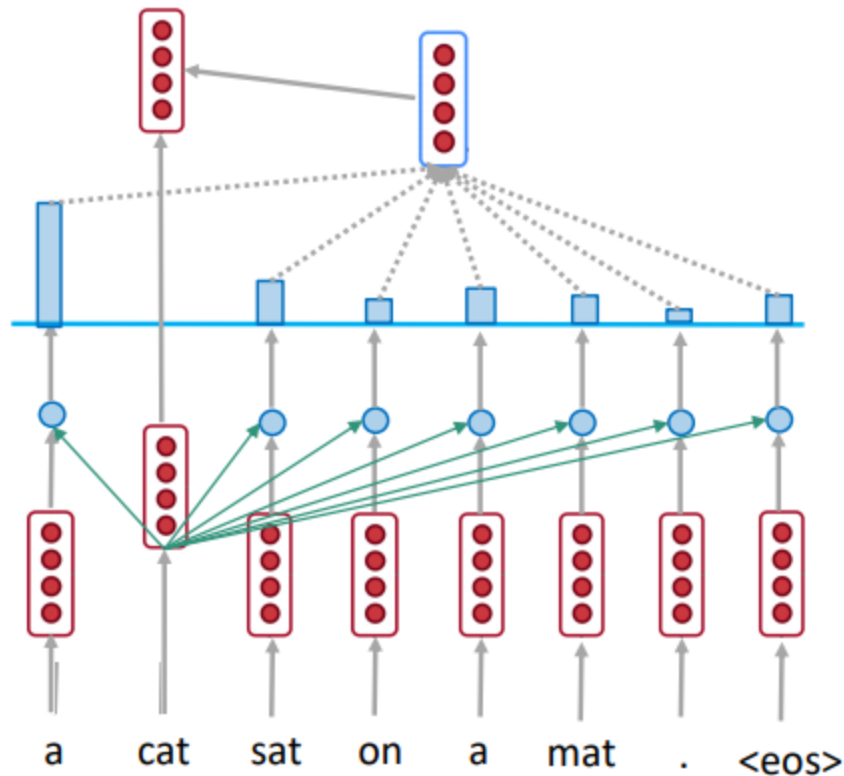
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention



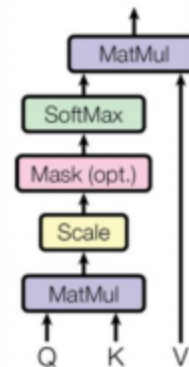
Self Attention

- Обновляем представление слова «cat»



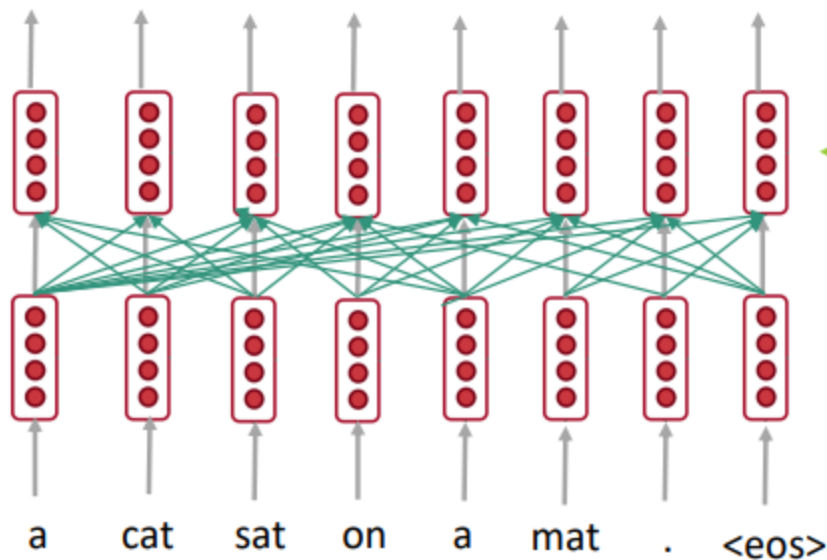
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention



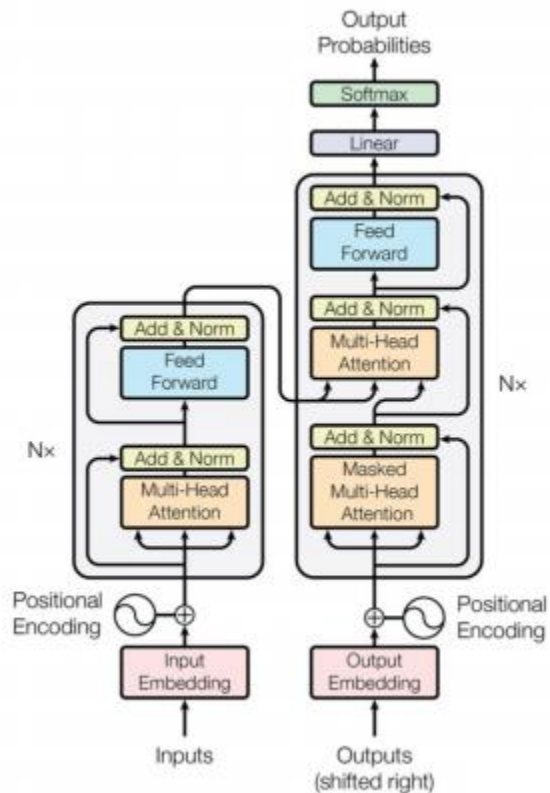
Self Attention

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

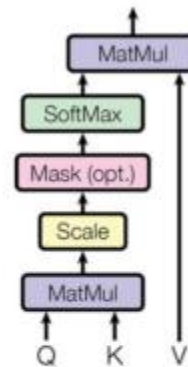


Updated representations
for each word
(one layer)

Transformer

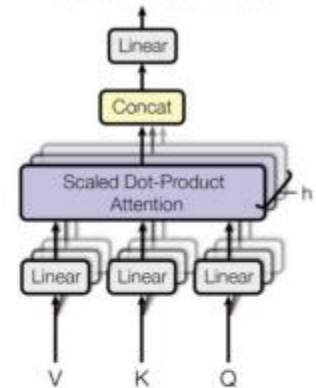


Scaled Dot-Product Attention



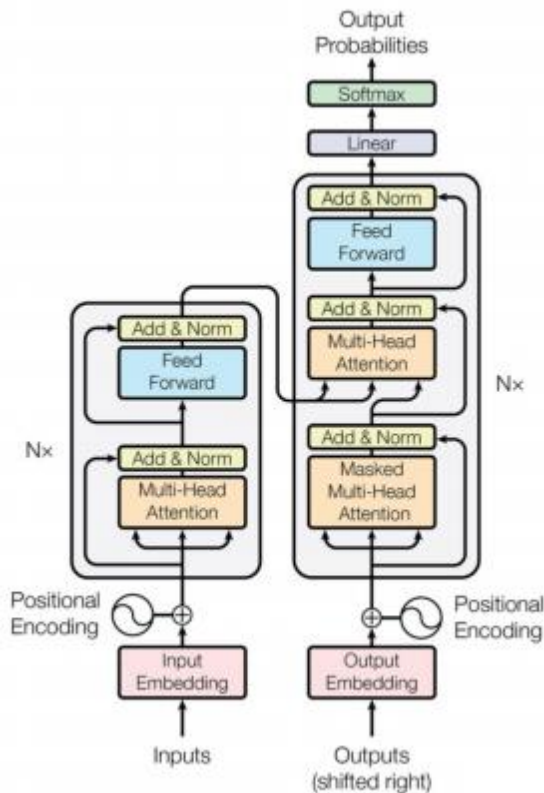
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-Head Attention



Transformer

- Преимущества:
 - Нет рекуррентности, параллельное кодирование
 - Быстрое обучение: encoder и decoder могут быть параллельны
 - Нет длинных связей: $O(1)$ для любых токенов
 - Три внимания: модель не помнит слишком много
 - Multi-head attention позволяет обращать внимание на разные аспекты



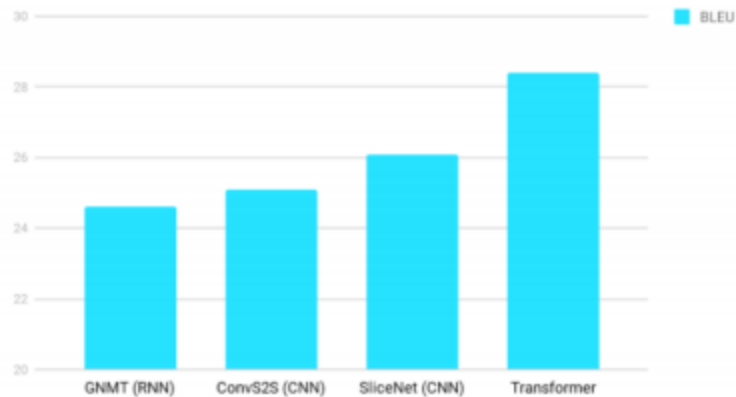
Transformer

*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.*

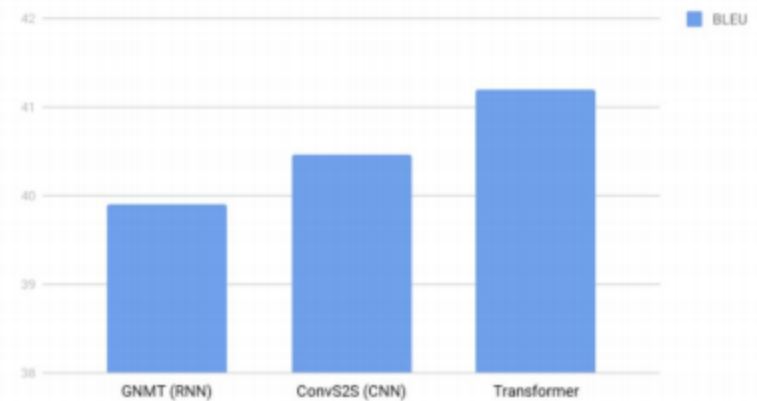
Transformer

English German Translation quality



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

English French Translation Quality



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

to French translation (one or eight attention heads).

Google BERT (2018)

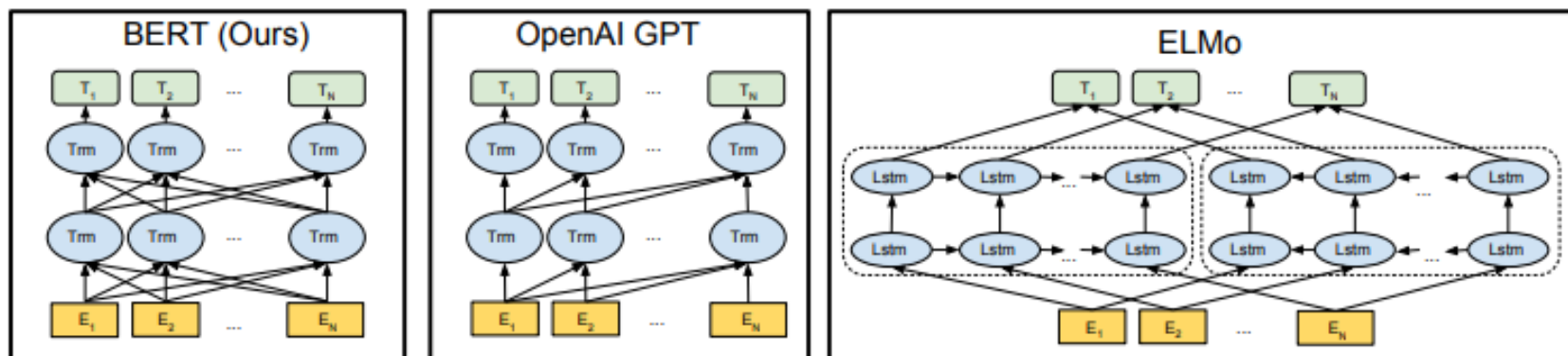
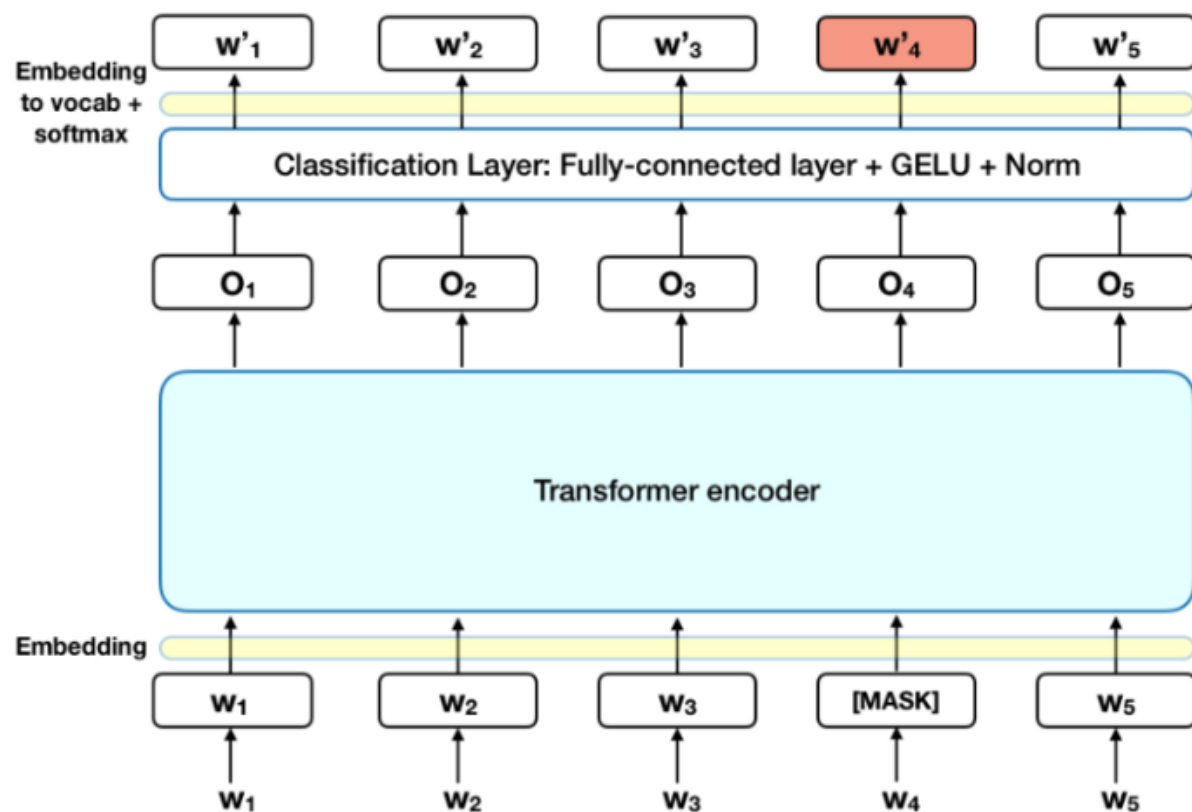


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

Google BERT (2018)

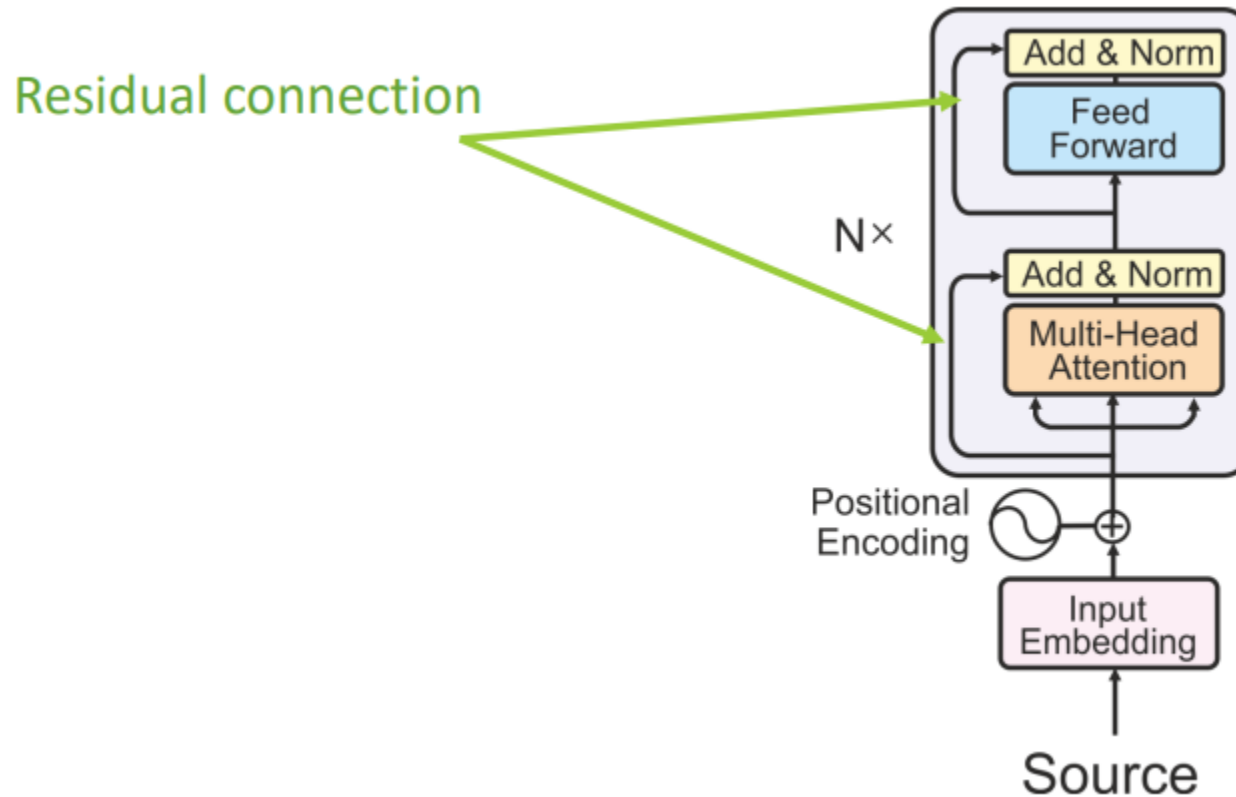
- **Masked LM (MLM)**

- Прежде чем последовательности слов поступают в BERT, 15% слов в каждой последовательности заменяются токеном [MASK]. Затем модель пытается предсказать исходное значение замаскированных слов на основе контекста, предоставленного другими, не маскируемыми словами в последовательности.



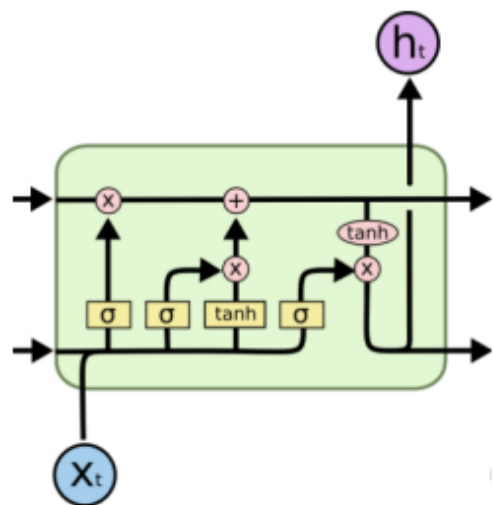
- Rather than *always* replacing the chosen words with [MASK], the data generator will do the following:
- 80% of the time: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK]
- 10% of the time: Replace the word with a random word, e.g., my dog is hairy → my dog is apple
- 10% of the time: Keep the word unchanged, e.g., my dog is hairy → my dog is hairy. The purpose of this is to bias the representation towards the actual observed word.

Почему residual connection?



Почему residual connection? Пример

- LSTM and long-term dependencies



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

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

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

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

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

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

Original version:

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

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

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

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

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

- Почему проблема vanishing/exploding здесь не настолько существенна?

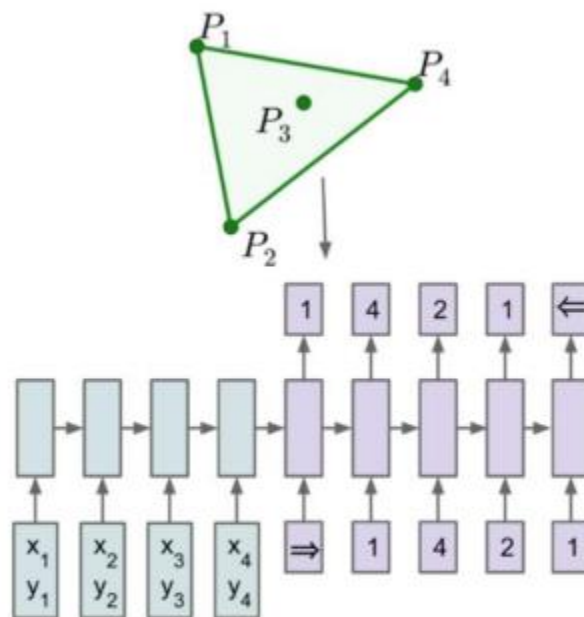
Что если мы не знаем размер словаря?

- Pointer Networks

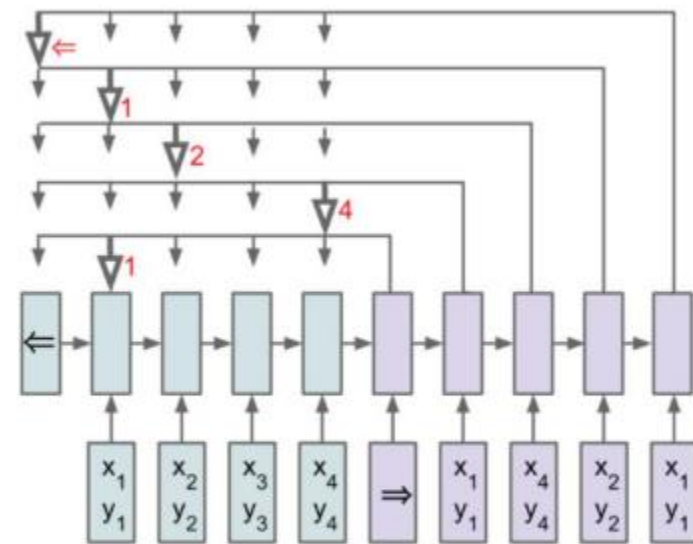
Attention weights



output distribution



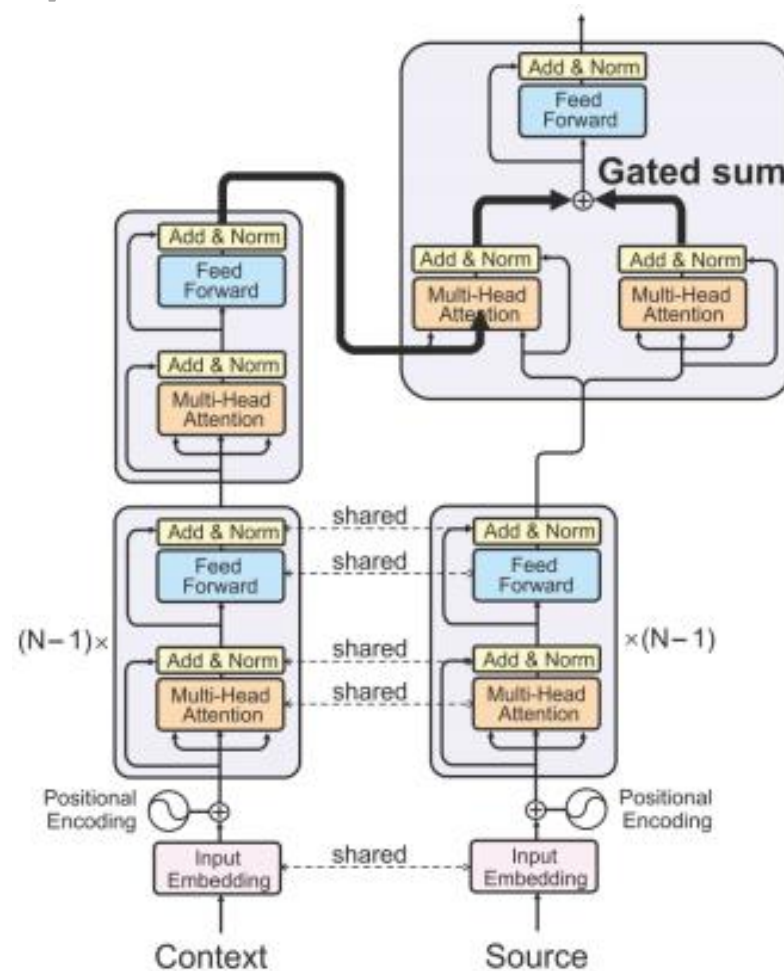
(a) Sequence-to-Sequence



(b) Ptr-Net

Что если ввести контекст в NMT систему?

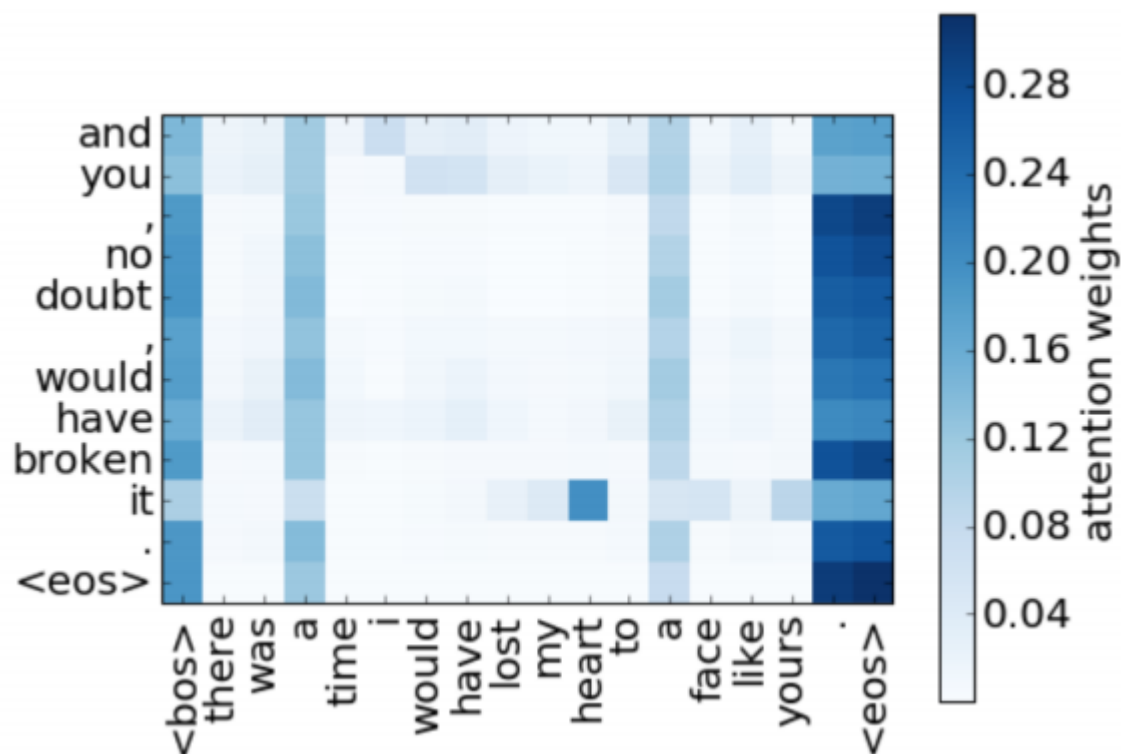
- Context-aware NMT для обнаружения анафоры
- Начинаем с Transformer [Vaswani et al, 2018]
- Включаем информацию о context на стороне encoder
- Используем разделение encoder и context
- Используем первые $N-1$ слоев source и context
- Последний слой включает в себя контекстную информацию



Что если ввести контекст в NMT систему?

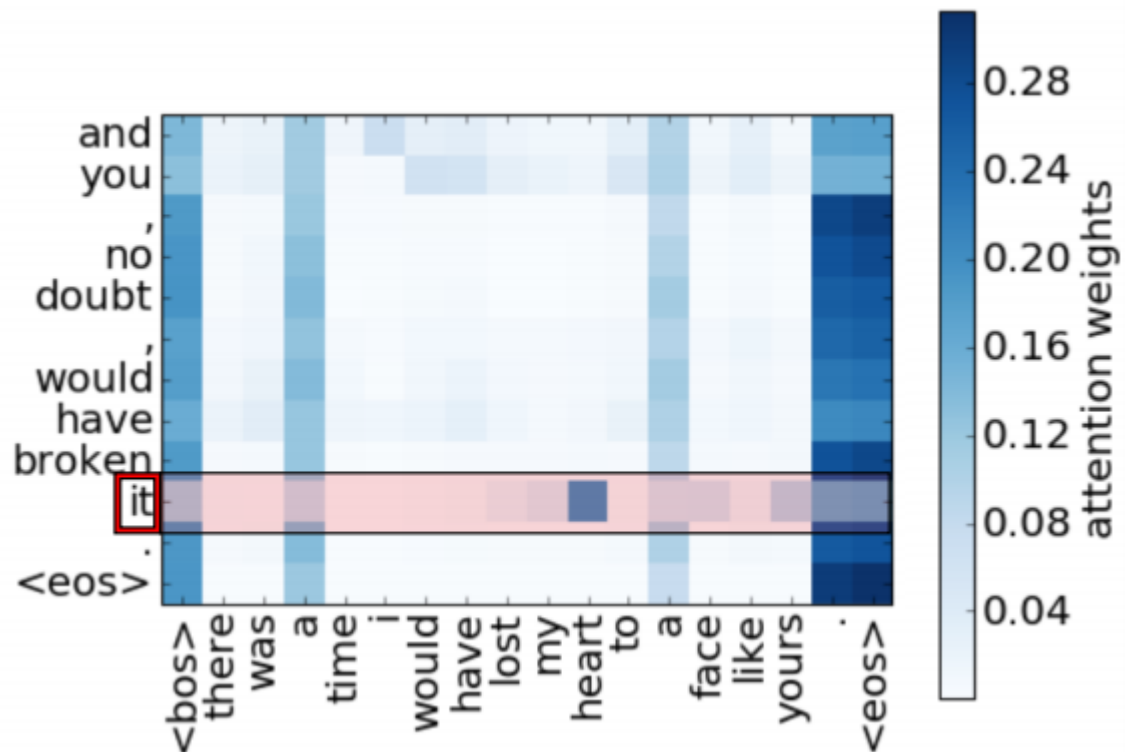
- Context:
 - There was a time I would have lost my heart to a face like yours.

- Source:
 - And you, no doubt, would have broken **it**.



Что если ввести контекст в NMT систему?

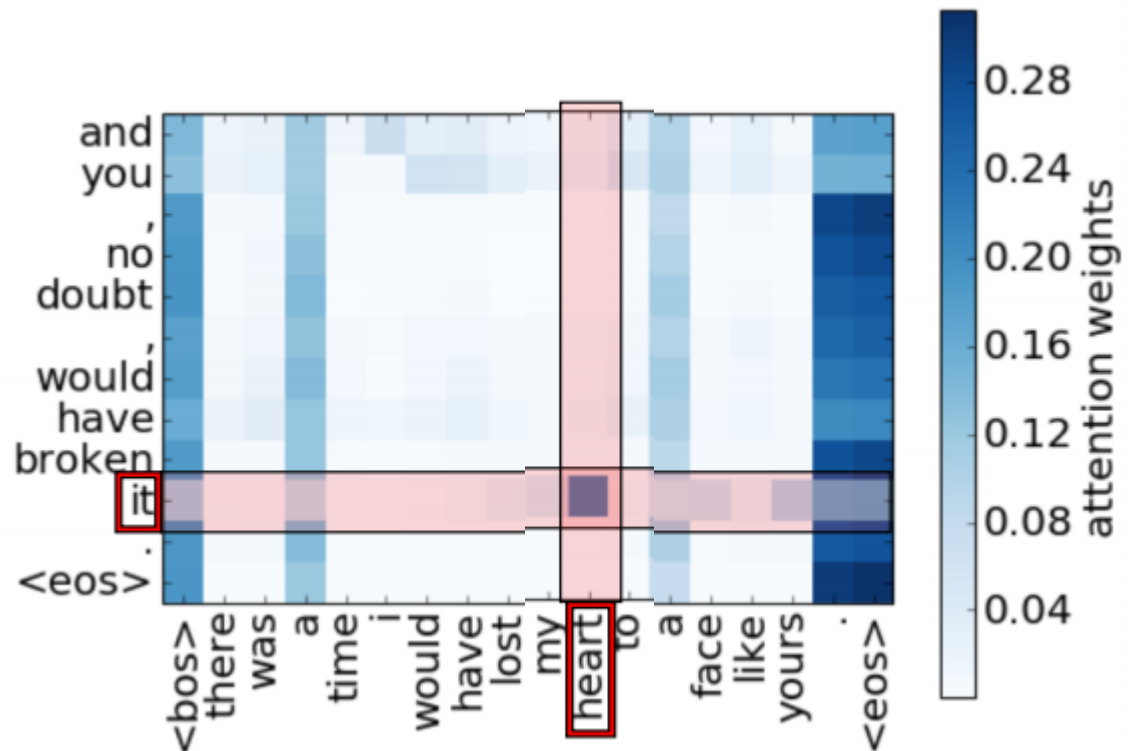
- Context:
 - There was a time I would have lost my heart to a face like yours.
- Source:
 - And you, no doubt, would have broken **it**.



Что если ввести контекст в NMT систему?

- 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**.



Layer norm

Batch normalization

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

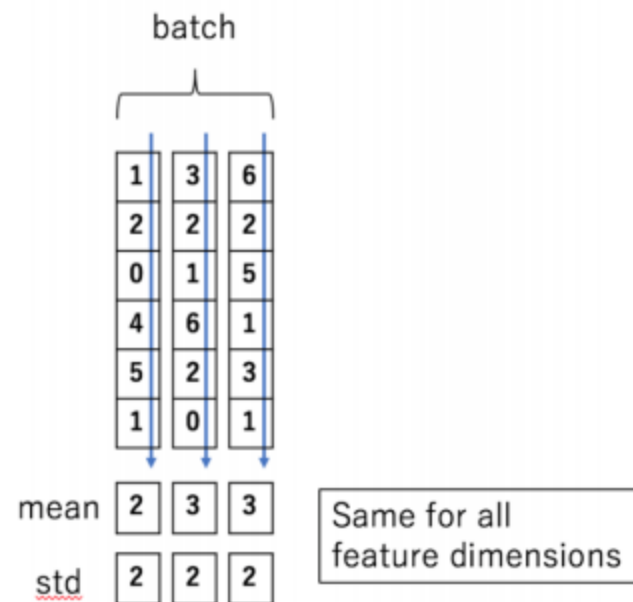
Layer normalization:

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

Batch Normalization



Layer Normalization



1. Paper: Ba et al, 2016, <https://arxiv.org/abs/1607.06450>

2. Pictures: <http://mlexplained.com/2018/01/13/weight-normalization-and-layer-normalization-explained-normalization-in-deep-learning-part-2/>

Применения Attention

- Classification
- Natural Language Inference
- NER
- Question Answering
- Sentence parse trees
- Image captioning
- ...

СПАСИБО ЗА ВНИМАНИЕ

Литература

- NLP курс в яндексе https://github.com/yandexdataschool/nlp_course
- Курс в Stanford CS224N <http://cs224n.stanford.edu/>
- CMU lectures [seq2seq](#) and [attention](#)
- [BLEU](#) and [CIDEr](#)
- Image captioning
 - MSCOCO captioning [challenge](#)
 - Captioning baseline [notebook](#)
- Lecture on attention mechanisms - [video](#) (RUSSIAN)
- Illustrated transformer [post](#)