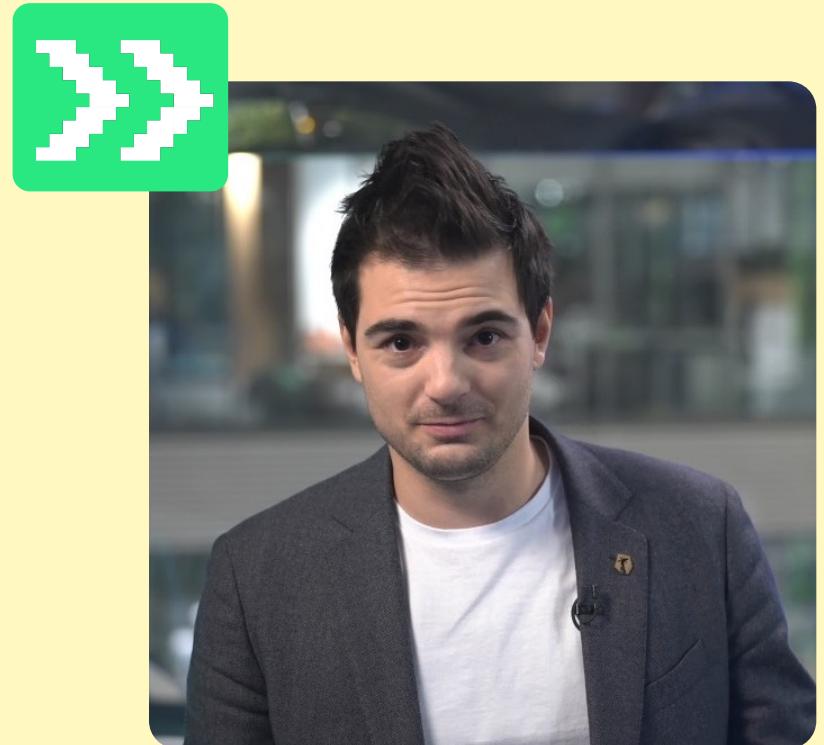


Авторегрессионные языковые модели; Механизм внимания (attention)

YOUNG & YANDEX

Радослав Нейчев
Выпускник и преподаватель ШАД и МФТИ,
руководитель группы ML-разработки в Яндексе,
основатель girafe-ai

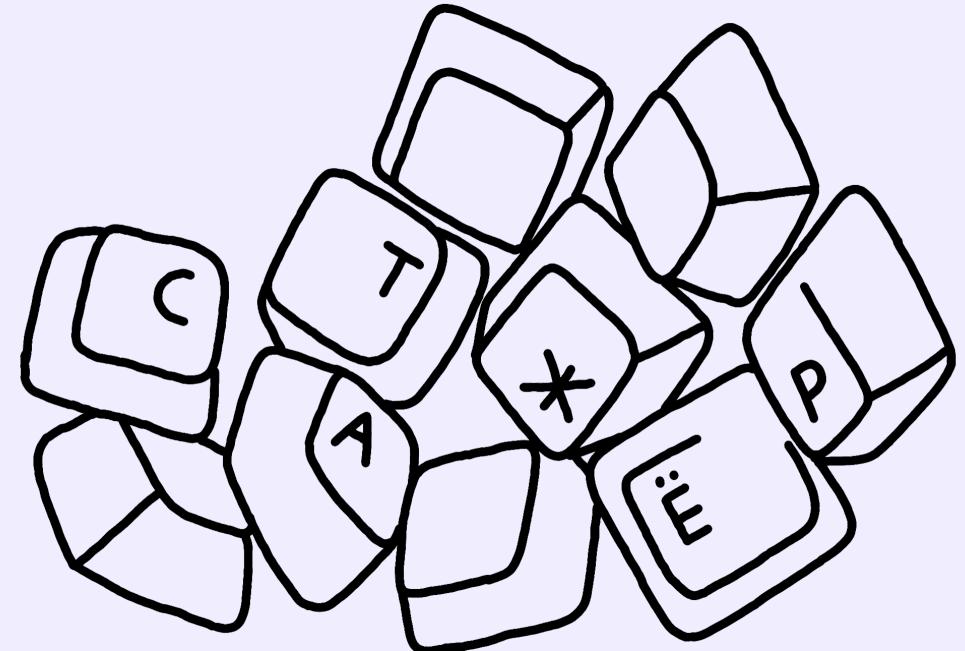


Содержание

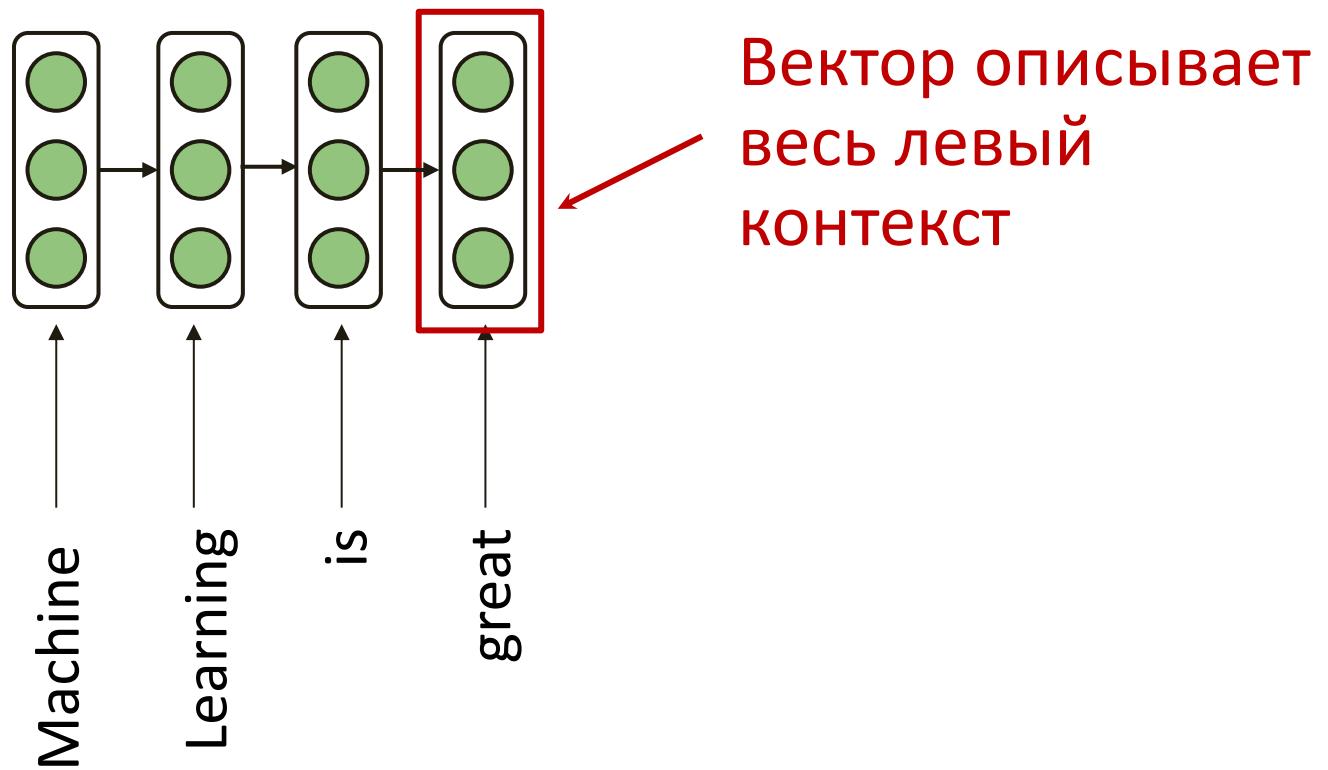
- 01** Авторегрессионные языковые модели
(на примере RNN)
- 02** Лучевой поиск (beam search)
- 03** (Self-)Attention – механизм внимания
- 04** Непосредственно Self-Attention
- 05** Outro про машинный перевод

Авторегрессионные языковые модели (на примере RNN)

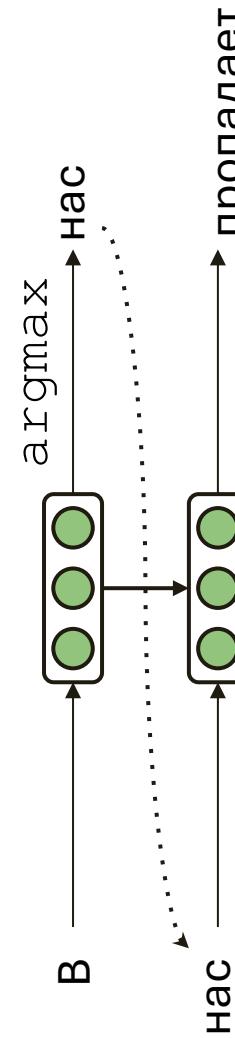
01



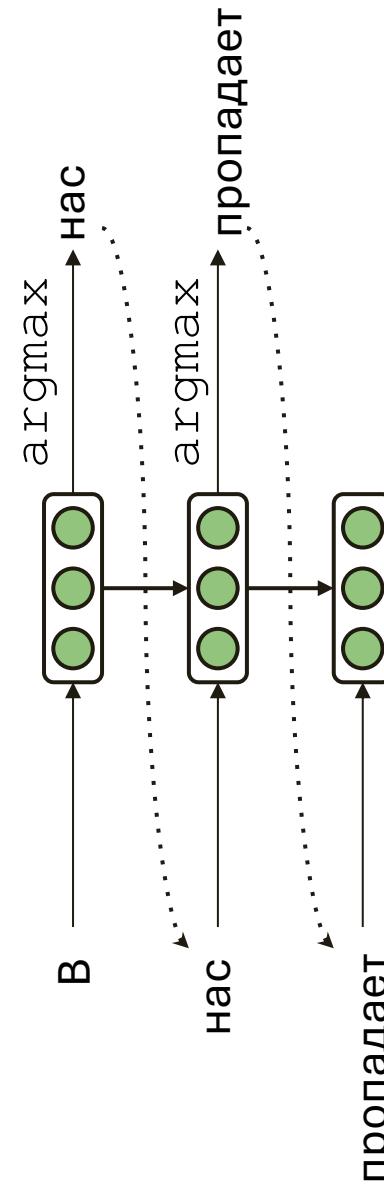
Кодирование последовательностей в RNN



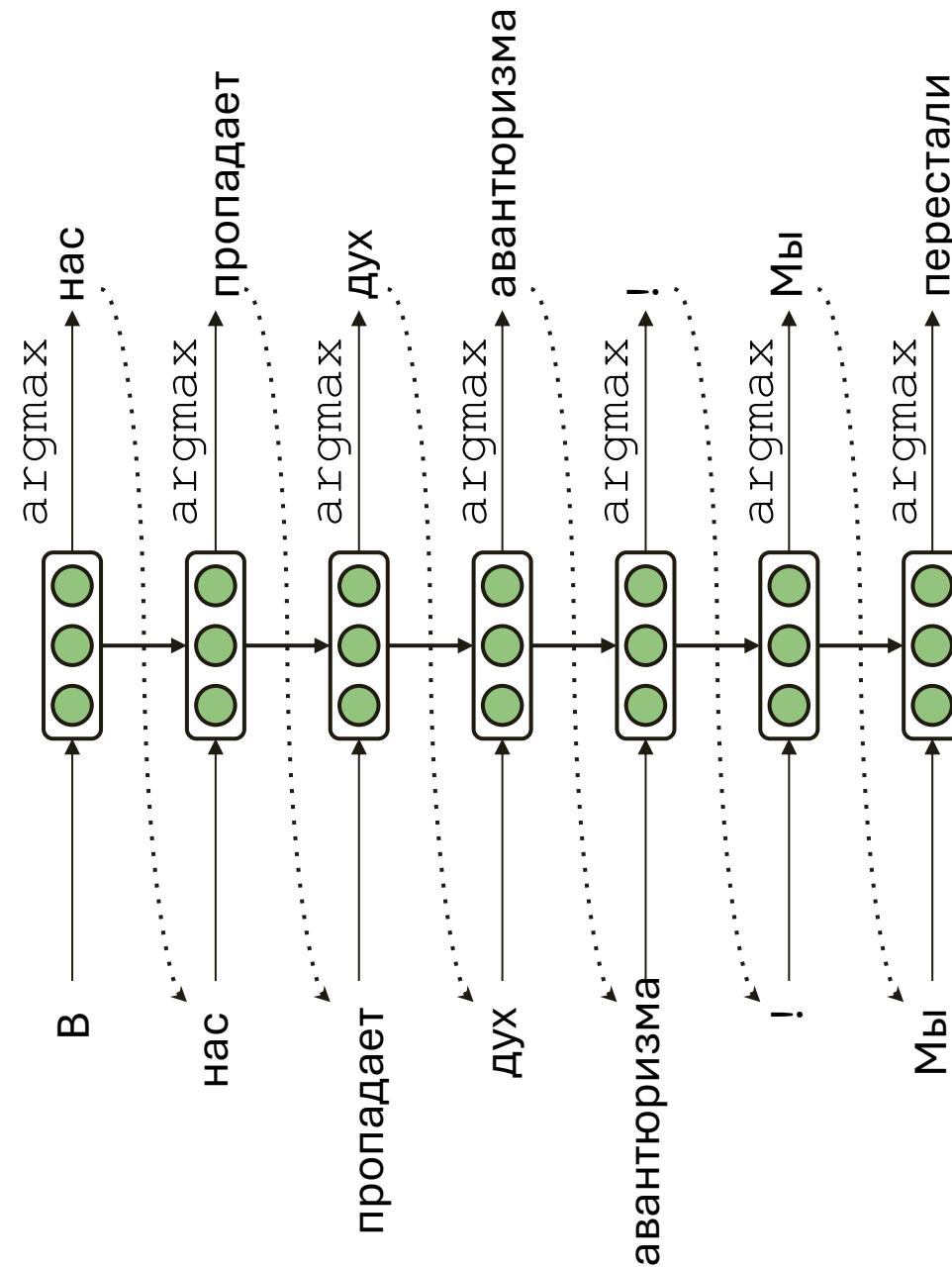
Генерация в RNN



Генерация в RNN



Генерация в RNN



Правдоподобие последовательности

- Мы можем оценить правдоподобие сгенерированной последовательности на основе самой модели:

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

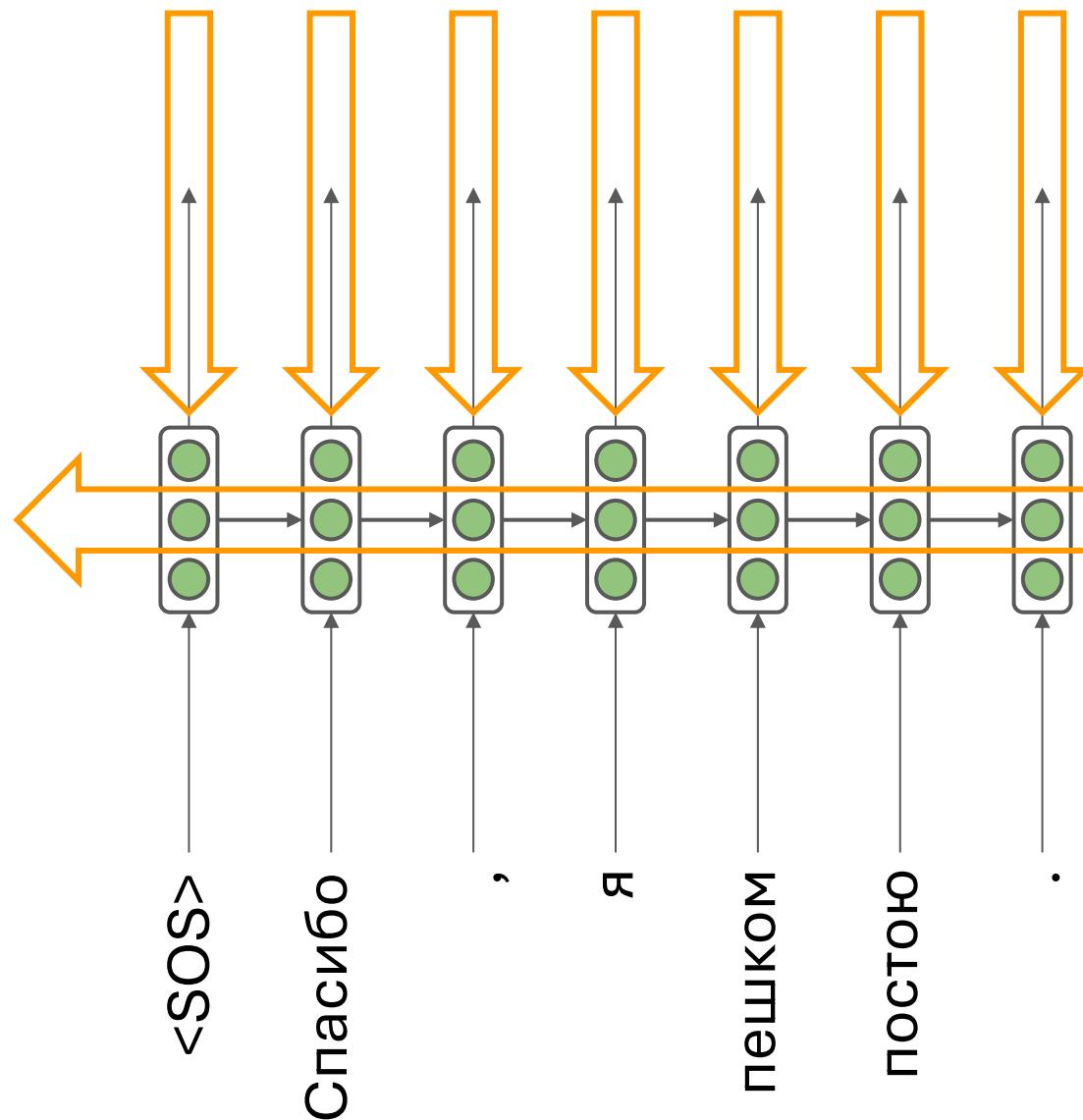
- у – сгенерированные элементы последовательности
- х – некоторая дополнительная информация
 - начальное состояние RNN, подводка и пр.

Вероятность слова на
шаге Т

Правдоподобие последовательности

При обучении языковой модели мы можем максимизировать правдоподобие сгенерированного текста.

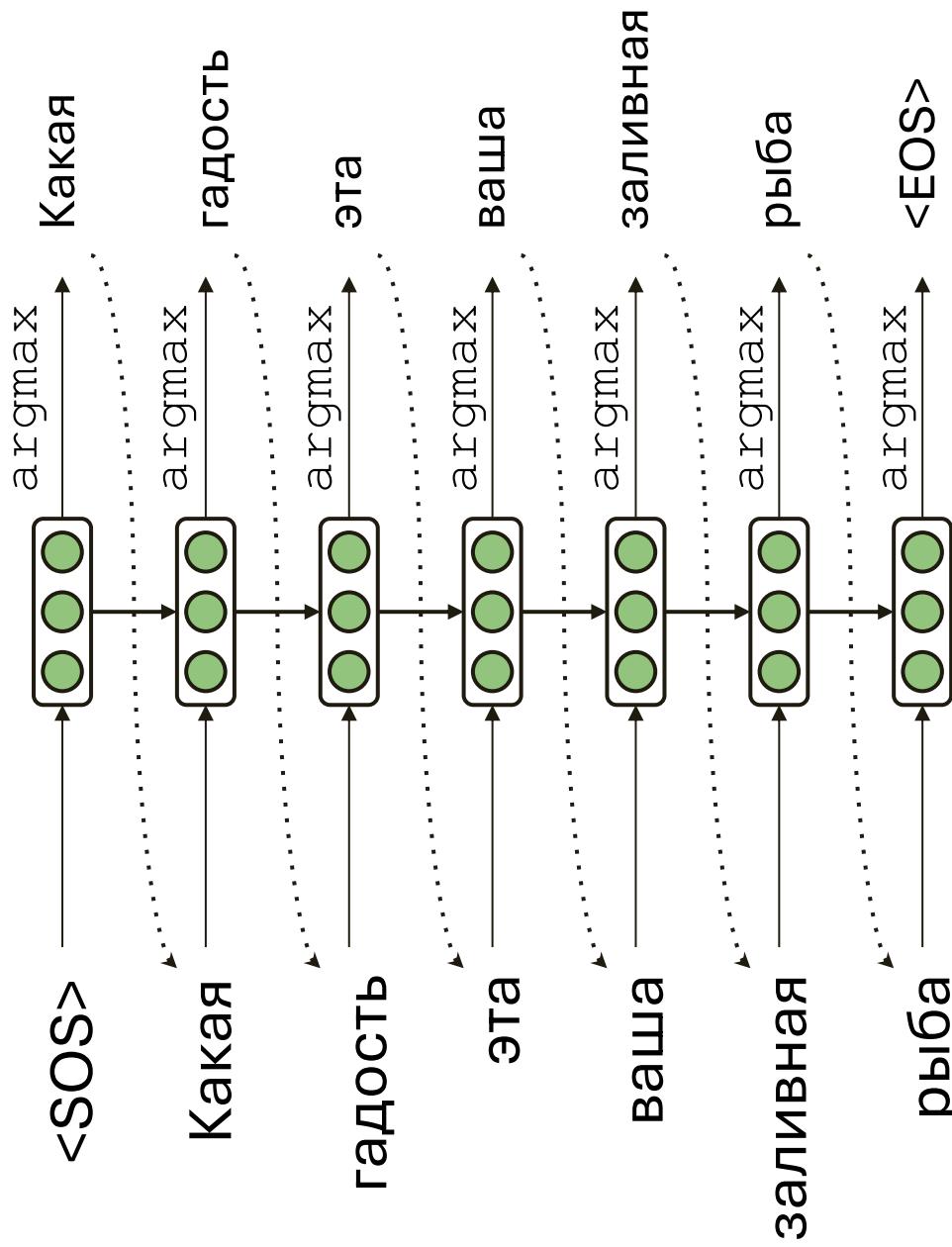
Это эквивалентно минимизации кросс-энтропии



Проблемы с генерацией

Что если на каком-то шаге модель предскажет ошибочный токен?

Все следующие шаги будут на него обусловлены – он станет частью входной последовательности!



Максимизация правдоподобия

- Наиболее правдоподобная последовательность максимизирует

$$P(y|x) = P(y_1|x) \prod_{t=2}^T P(y_t|y_1, \dots, y_{t-1}, x)$$

- Но мы не можем сгенерировать все возможные варианты

Лучевой поиск (beam search)

02



Лучевой поиск (Beam Search)

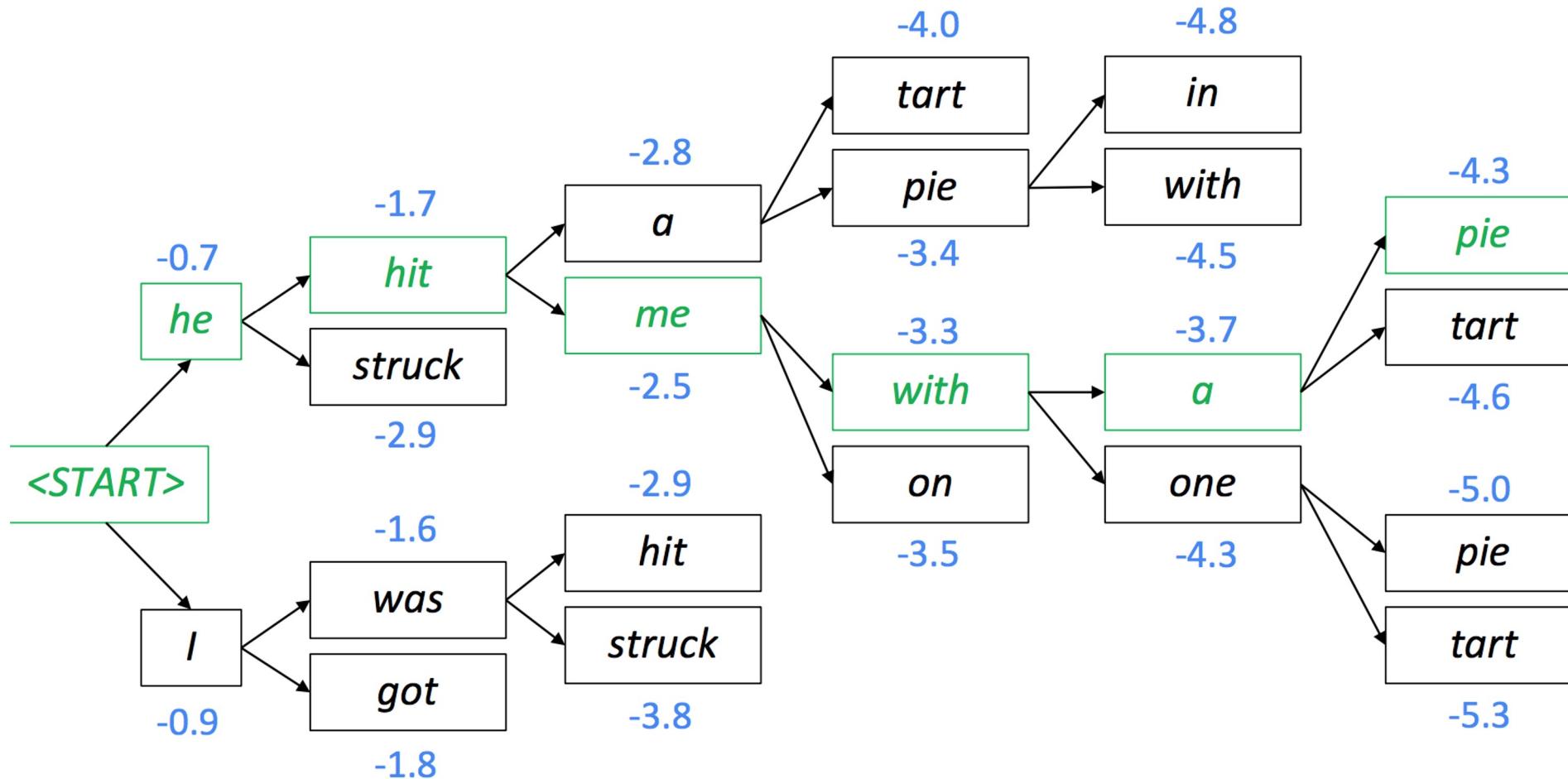
- На каждом шаге будем генерировать не один, а k продолжений для каждого из кандидатов
 - k – ширина луча (beam size)
- У каждой последовательности есть своя оценка:

$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

- На каждом шаге отбираем топ- k кандидатов

Лучевой поиск (Beam Search) – пример

Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Лучевой поиск (Beam Search) – детали

- Завершенная последовательность считается кандидатом, если был сгенерирован `<EOS>` токен (или достигнута максимальная длина)
- Критерий останова: найдено N (гиперпараметр) кандидатов
- У каждого кандидата есть своя оценка:

$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

... что может пойти не так?

Лучевой поиск (Beam Search) – детали

- Завершенная последовательность считается кандидатом, если был сгенерирован `<EOS>` токен (или достигнута максимальная длина)
- Критерий останова: найдено N (гиперпараметр) кандидатов
- У каждого кандидата есть своя оценка:

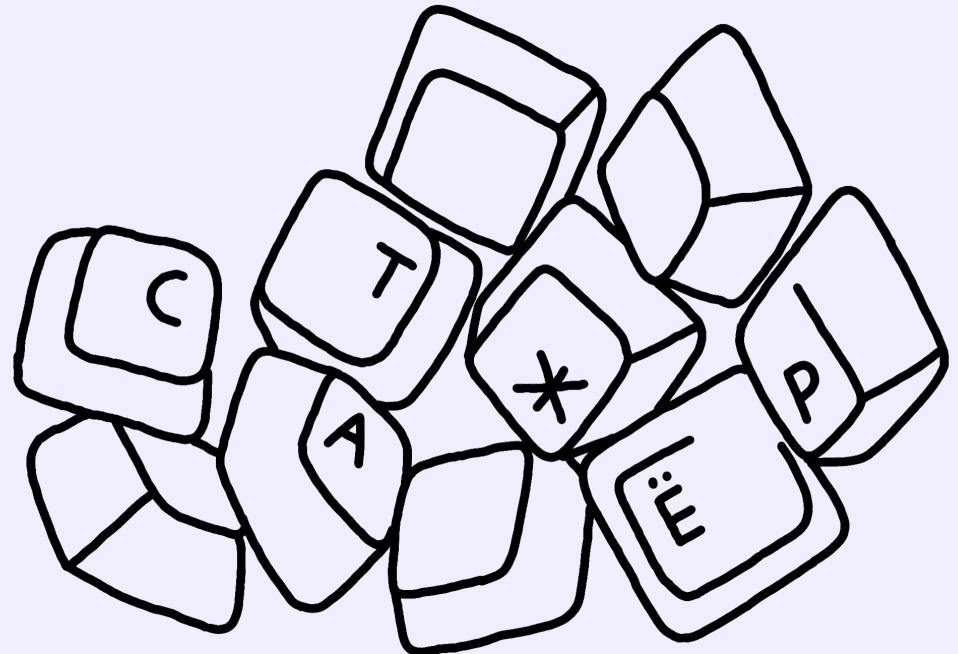
$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

- Чем длиннее кандидат, тем ниже итоговая оценка!
 - Применим нормировку на длину последовательности:

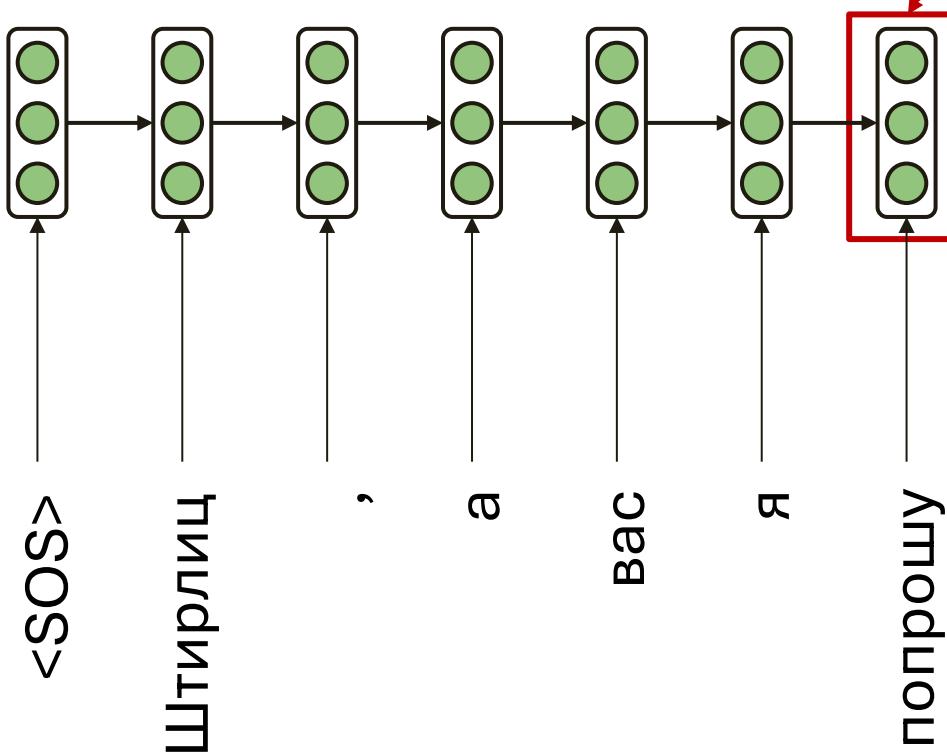
$$\frac{1}{t} \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

(Self-)Attention – механизм внимания

03



Вектор описывает
весь левый
контекст



Чем длиннее последовательность, тем меньше шанс
“вспомнить” данные, встреченные давно

Как работать с последовательностью переменной длины?

Среднее

Пошаговая обработка

$$z = \frac{1}{T} \sum_{i=1}^T x_i \quad h_{t+1} = f_\theta(h_t, x_t)$$

Как работать с последовательностью переменной длины?

Взвешенное среднее

$$z = \sum_{i=1}^T \alpha_i h_i$$

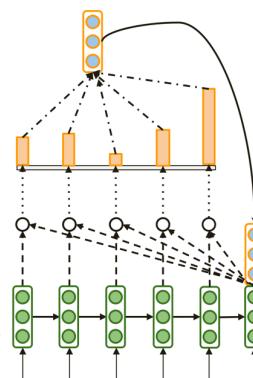
$$\sum_{i=1}^T \alpha_i = 1$$

$$\forall i \quad \alpha_i \in [0; 1]$$

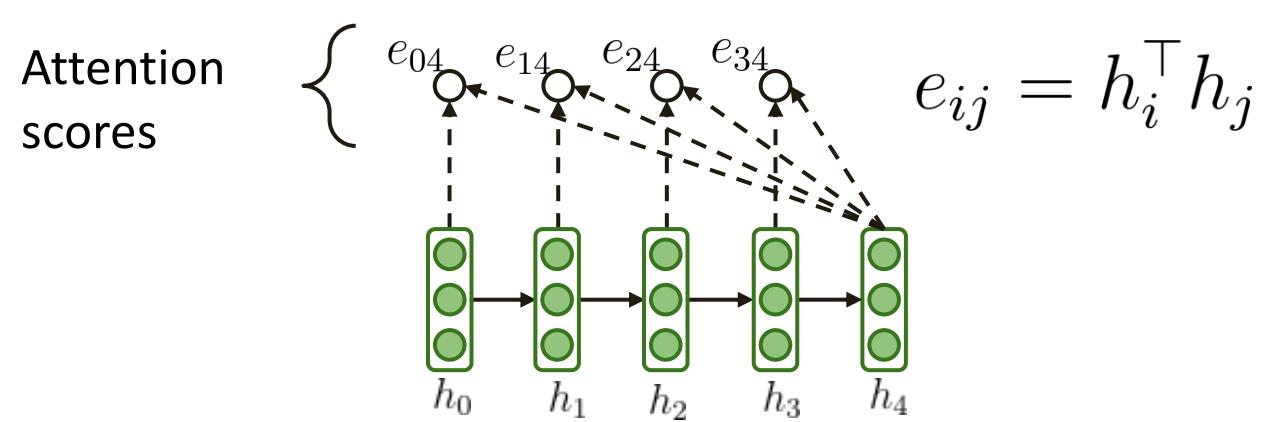
Как работать с последовательностью переменной длины?



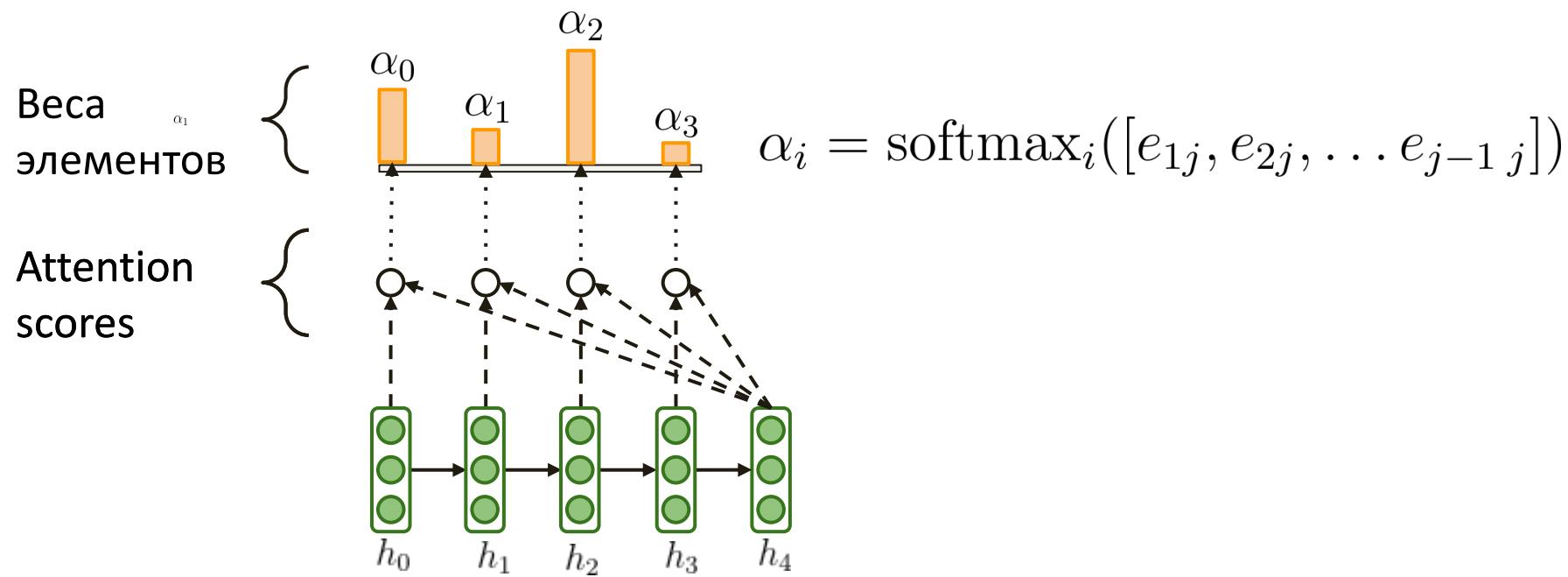
$$h_{t+1} = f_\theta(h_t, x_t)$$



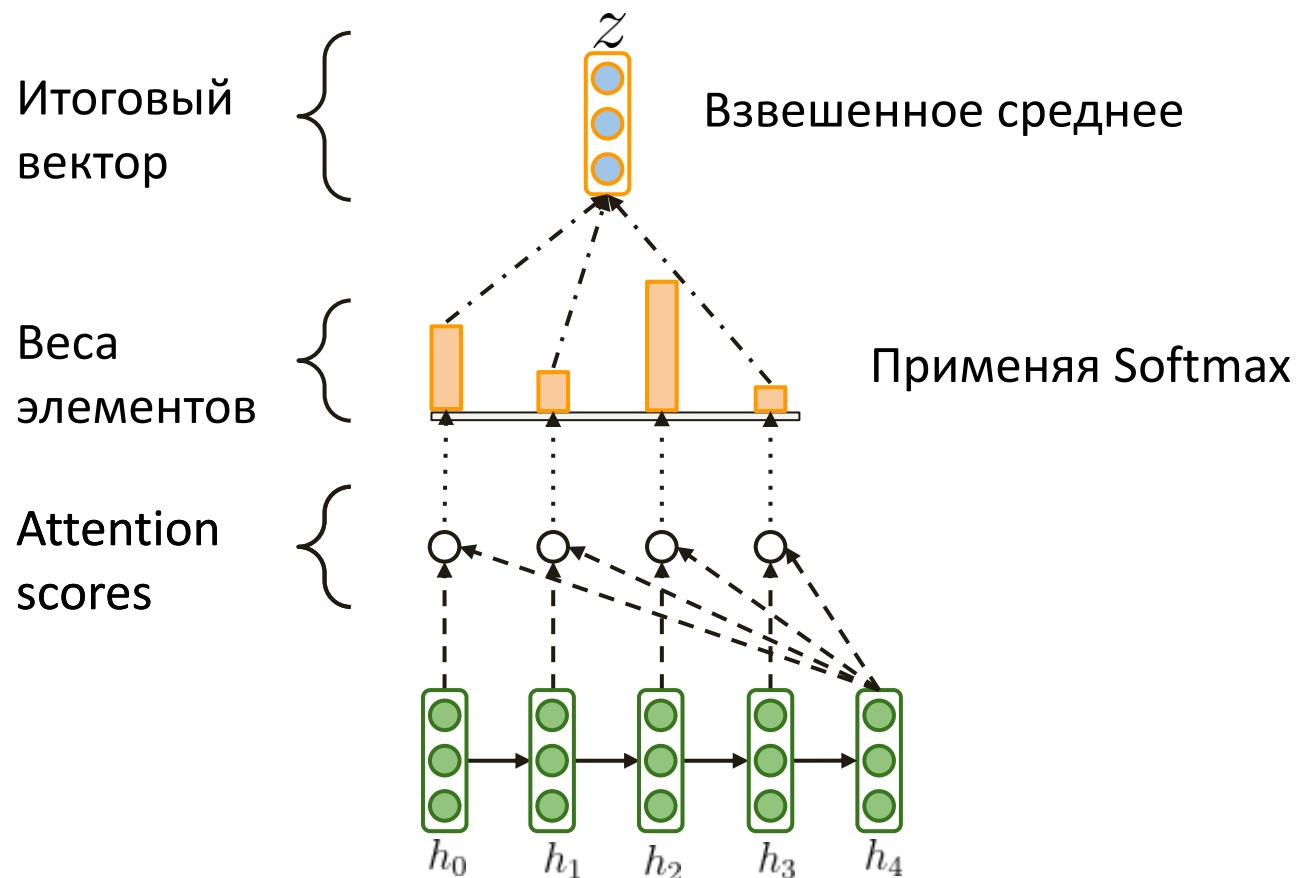
Seq2seq with attention



RNN + Attention (почти)



RNN + Attention (почти)

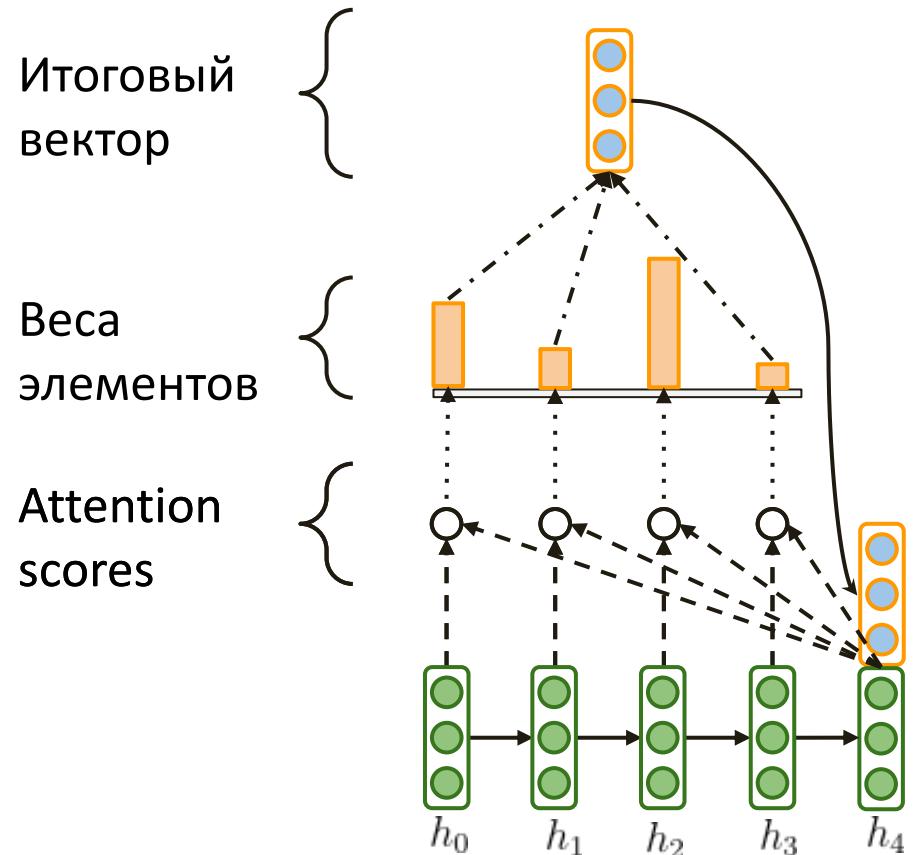


$$z = \sum_{i=1}^T \alpha_i h_i$$

$$\sum_{i=1}^T \alpha_i = 1$$

$$\forall i \quad \alpha_i \in [0; 1]$$

RNN + Attention (почти)

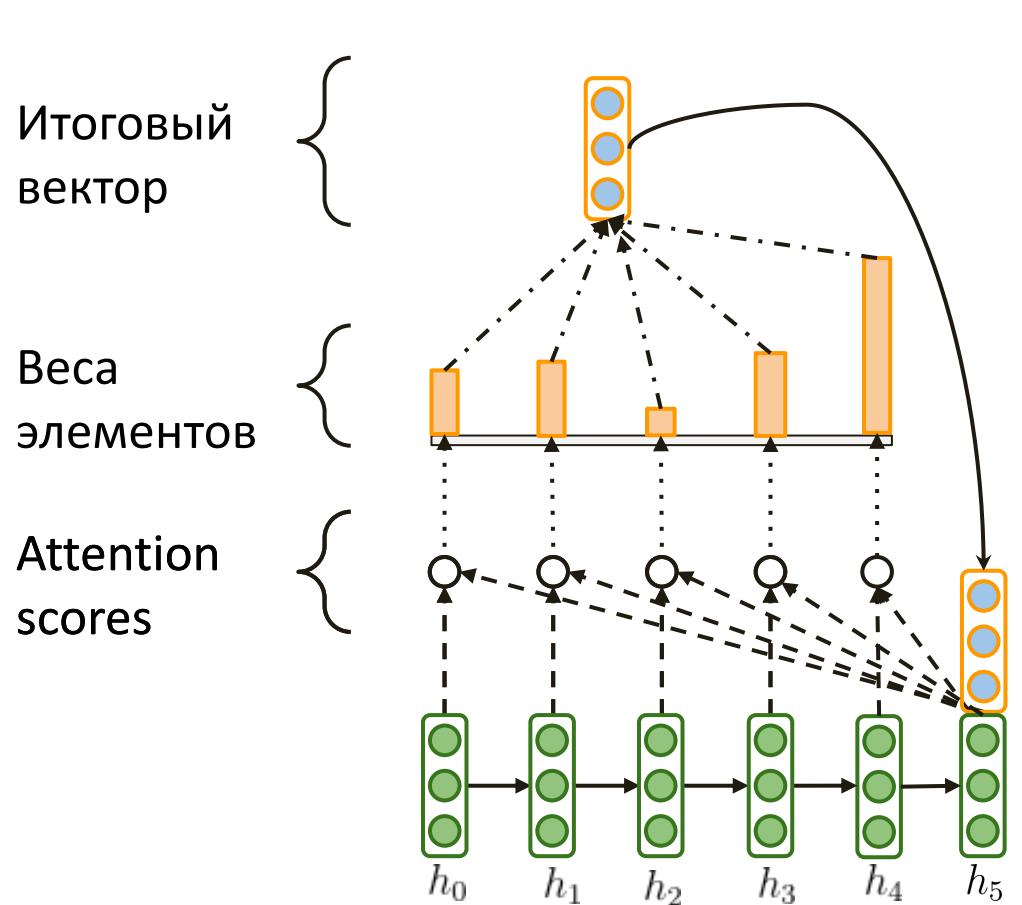


$$z = \sum_{i=1}^T \alpha_i h_i$$

$$\sum_{i=1}^T \alpha_i = 1$$

$$\forall i \quad \alpha_i \in [0; 1]$$

RNN + Attention (почти)

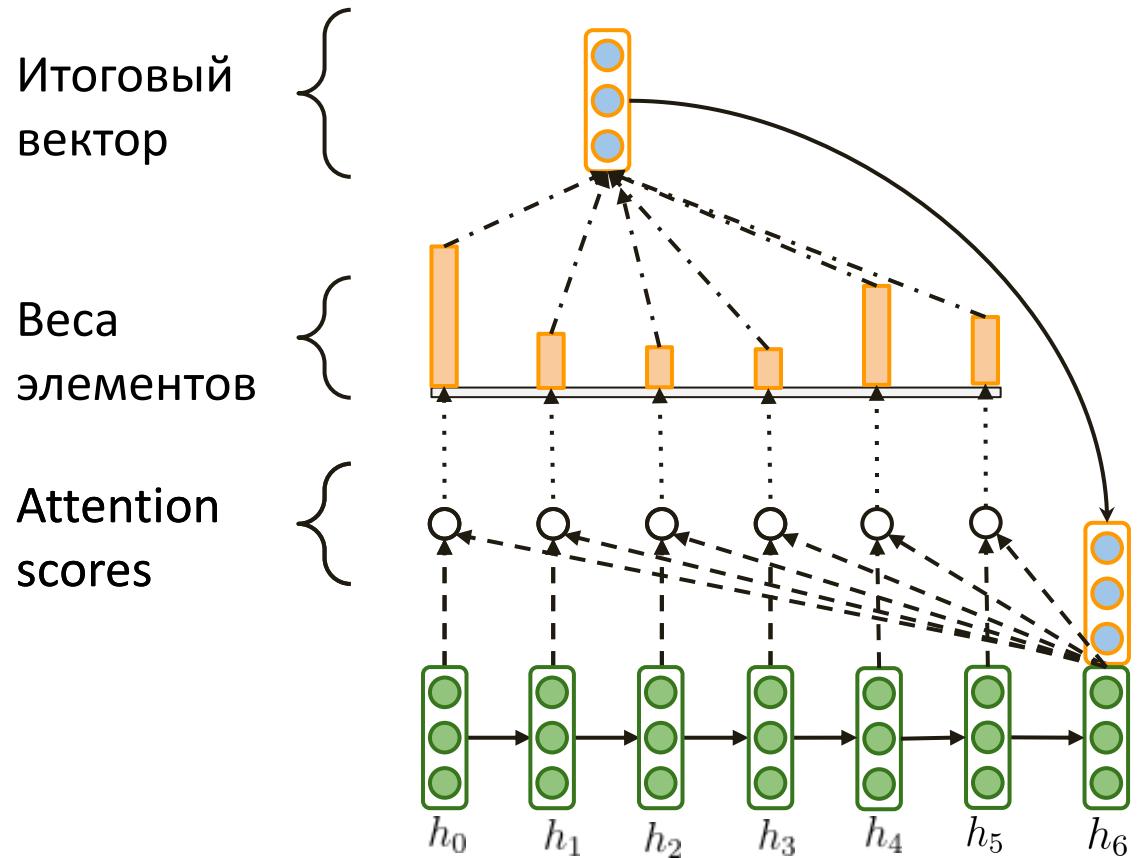


$$z = \sum_{i=1}^T \alpha_i h_i$$

$$\sum_{i=1}^T \alpha_i = 1$$

$$\forall i \quad \alpha_i \in [0; 1]$$

RNN + Attention (почти)



$$z = \sum_{i=1}^T \alpha_i h_i$$

$$\sum_{i=1}^T \alpha_i = 1$$

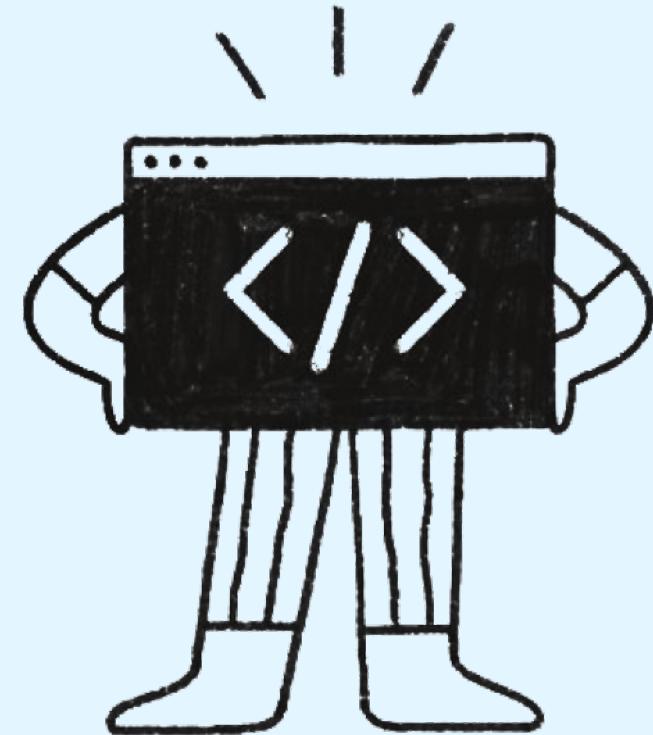
$$\forall i \quad \alpha_i \in [0; 1]$$

А нужна ли RNN при наличии Attention?

Непосредственно

Self-Attention

04



Self-Attention at a High Level

“The animal didn't cross the street because it was too tired”

- What does “it” in this sentence refer to?

Self-Attention at a High Level

“The animal didn't cross the street because it was too tired”

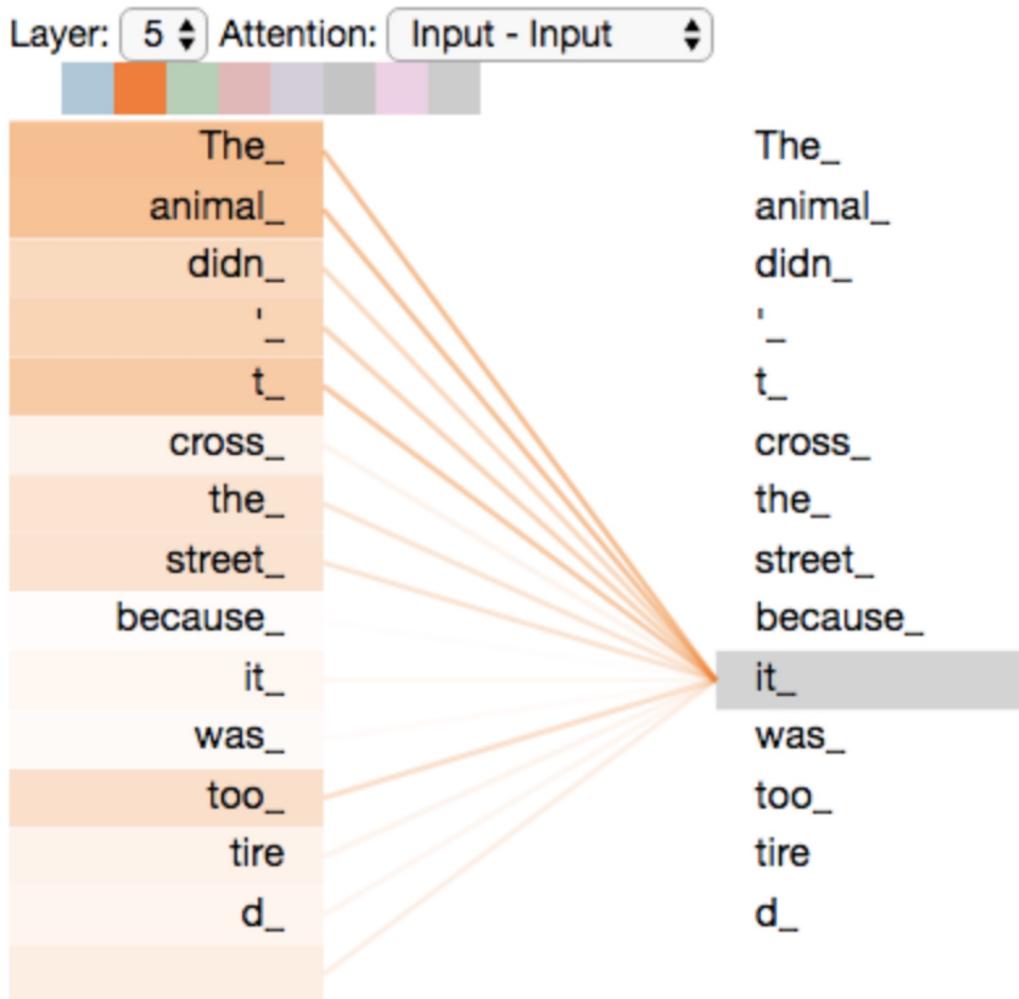
- What does “it” in this sentence refer to?
- We want self-attention to associate “**it**” with “**animal**”

Self-Attention at a High Level

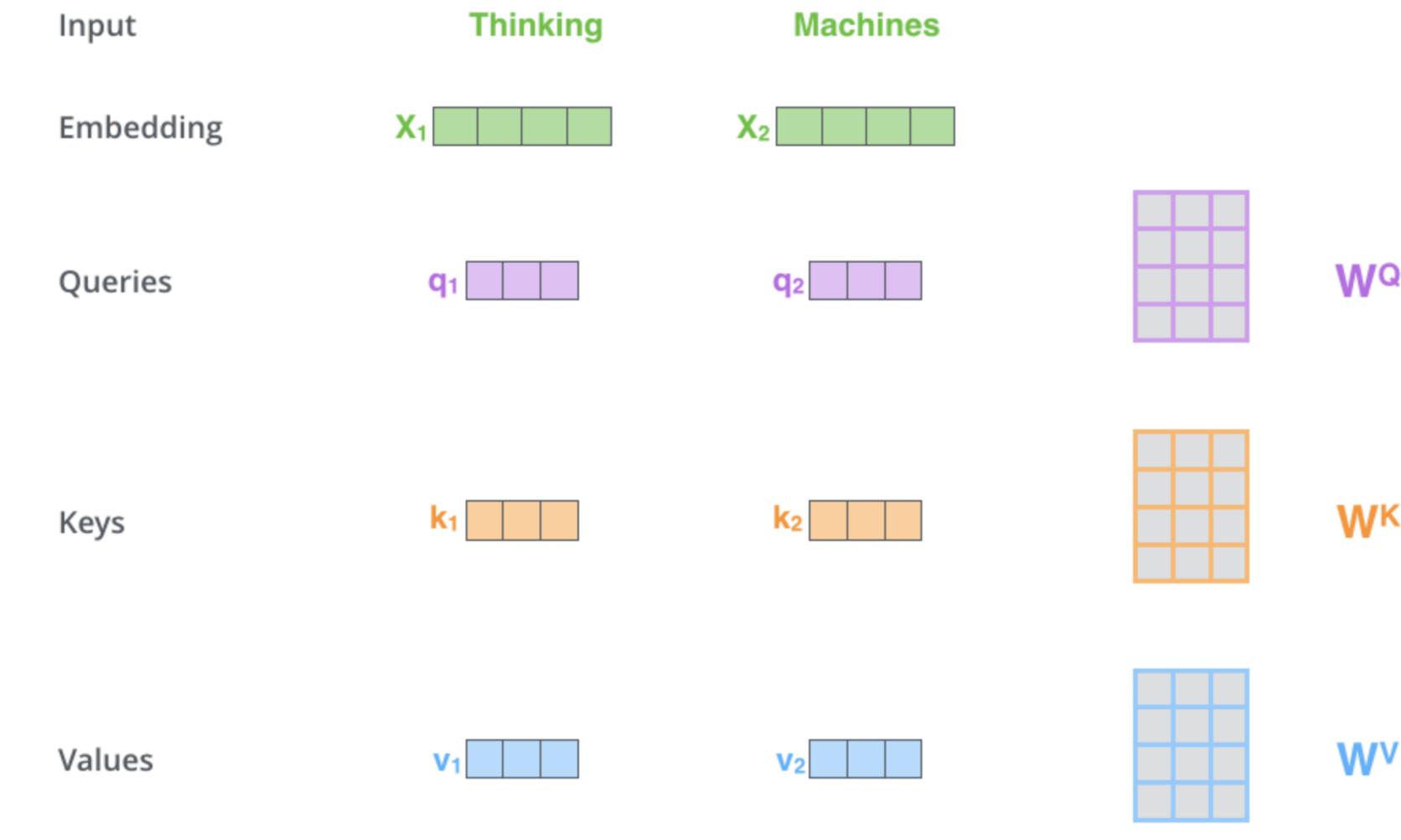
“The animal didn't cross the street because it was too tired”

- What does “it” in this sentence refer to?
- We want self-attention to associate “**it**” with “**animal**”
- Self-attention is the method the Transformer uses to bake the “understanding” of other relevant words into the one we’re currently processing

Self-Attention at a High Level



Self-Attention: detailed explanation

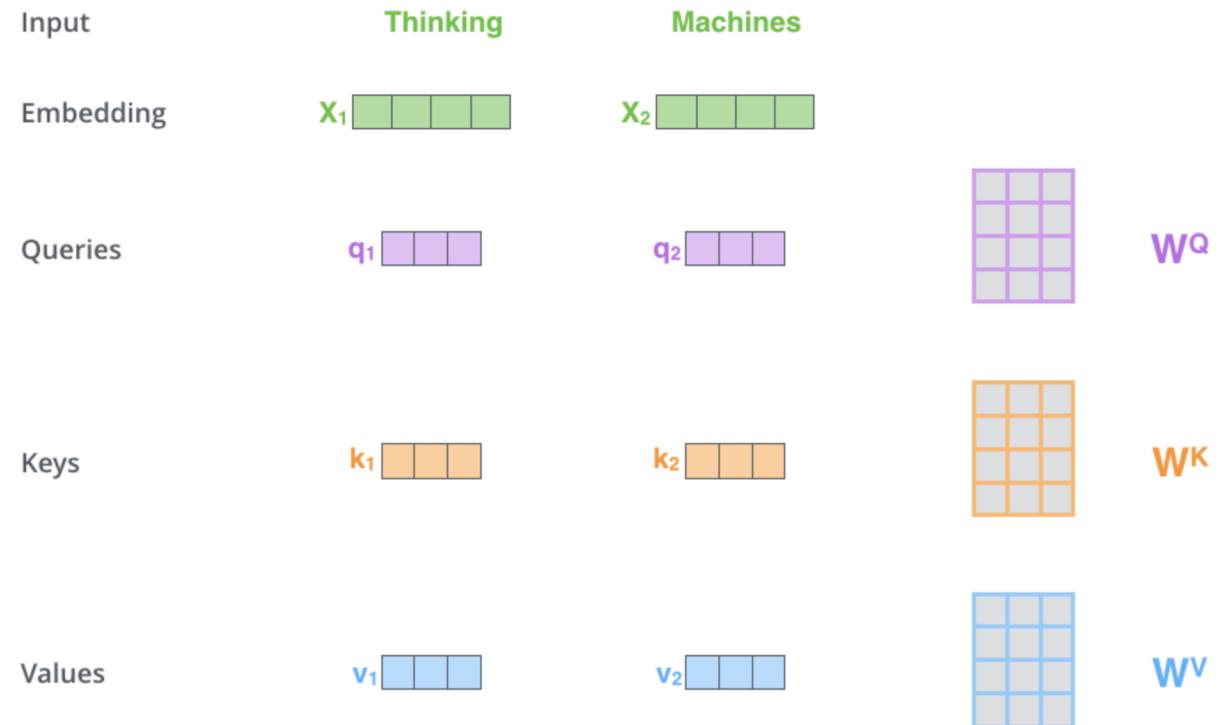


Self-Attention: detailed explanation

STEP 1:

create 3 vectors
(Query, Key, Value)

from each of the
encoder's input vectors



Self-Attention: detailed explanation

What are the “query”, “key”, and “value” vectors?

Self-Attention: detailed explanation

What are the “query”, “key”, and “value” vectors?

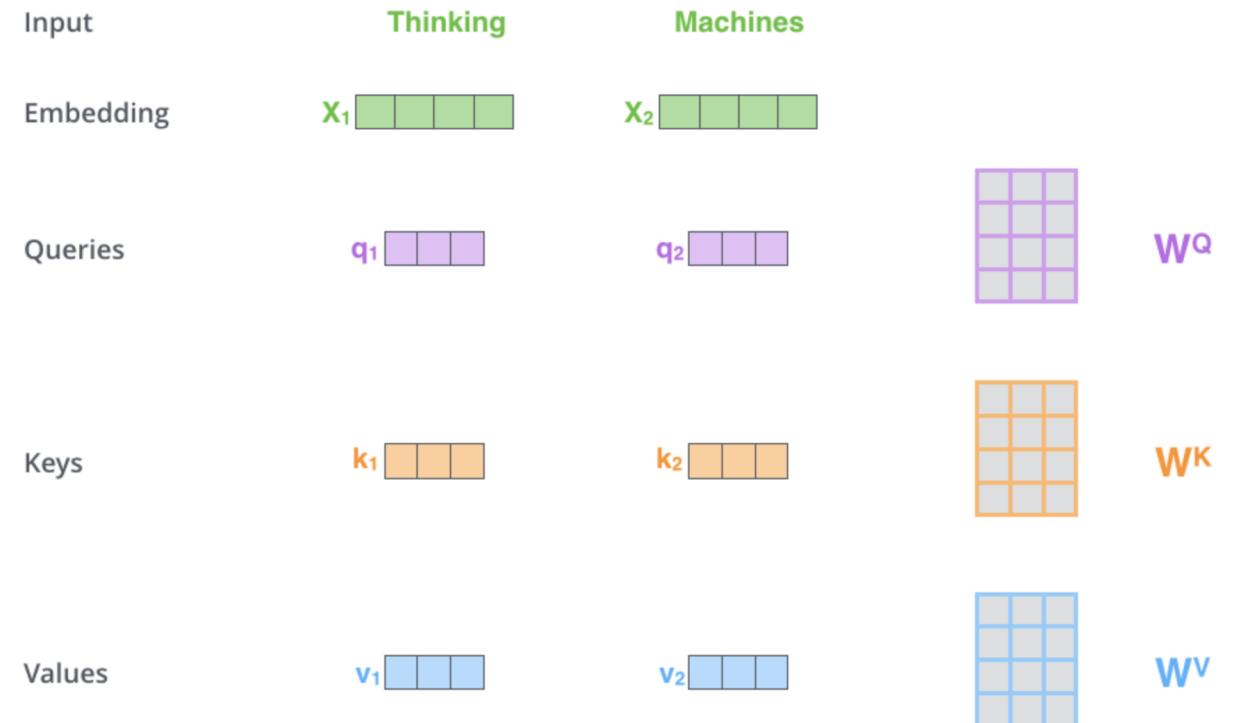
They’re abstractions that are useful for
calculating and thinking about attention.

Self-Attention: detailed explanation

STEP 2:

calculate a score

(score each word of the
input sentence against the
current word)



Self-Attention: detailed explanation

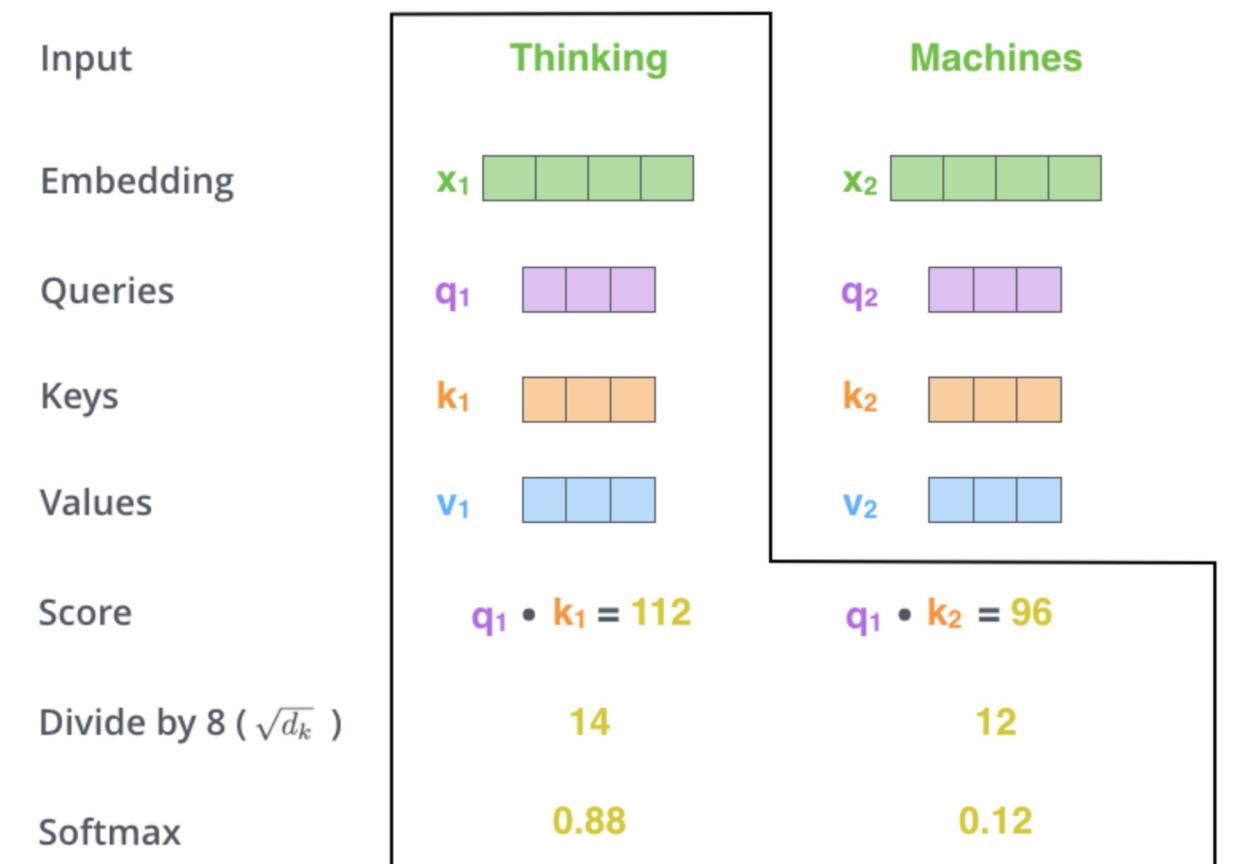
STEP 3:

divide the scores by 8

(the square root of the dimension of the key vectors)

STEP 4:

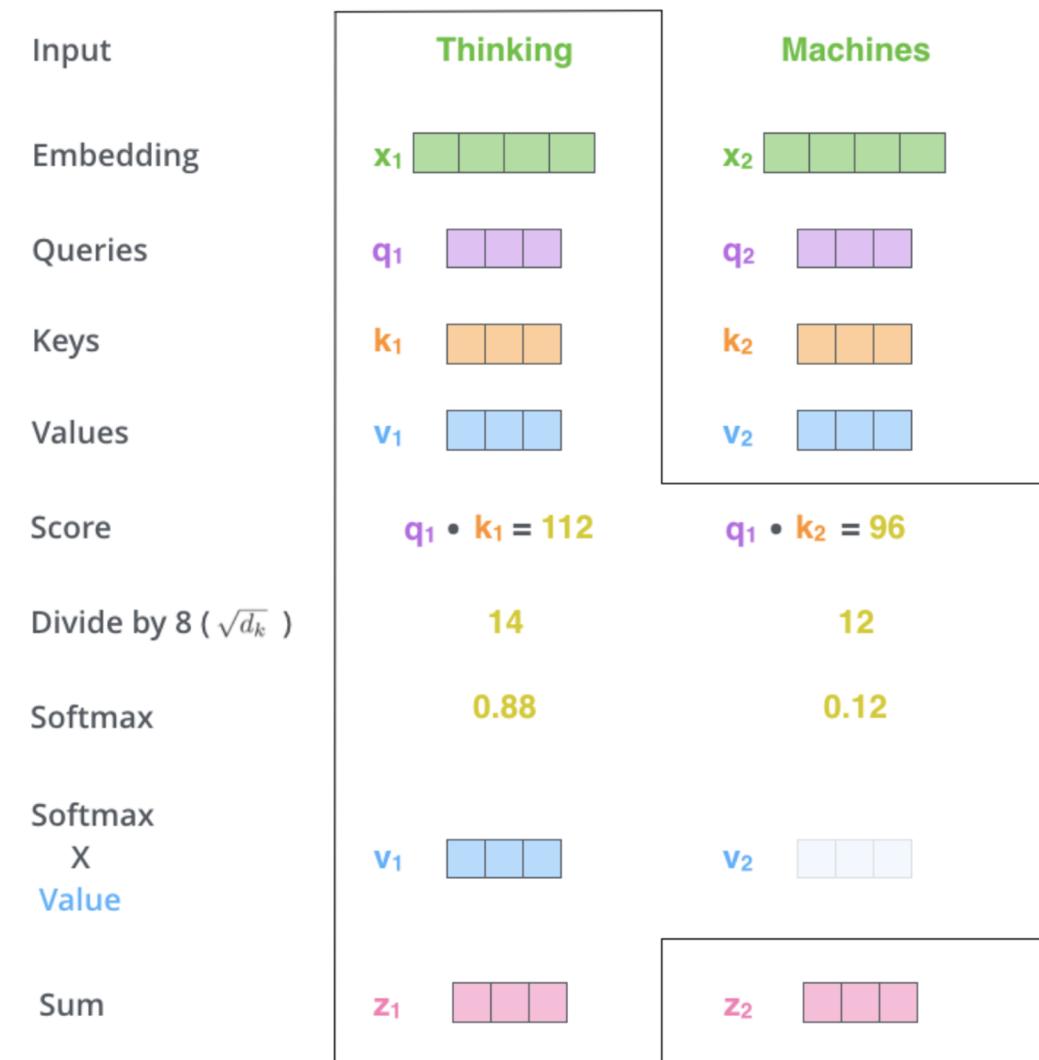
softmax



Self-Attention: detailed explanation

STEP 5:

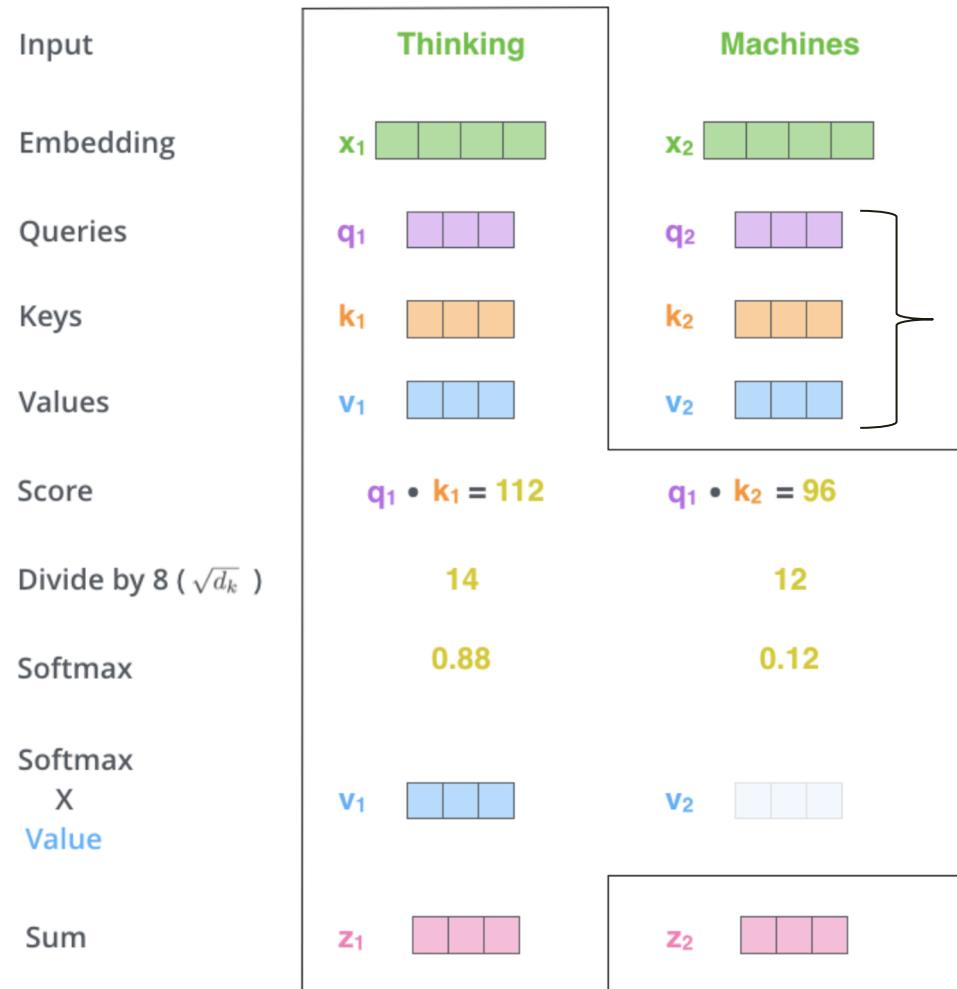
multiply each value vector
by the softmax score



STEP 6:

sum up the weighted value
vectors

Self-Attention



STEP 1: create Query, Key, Value

STEP 2: calculate scores

STEP 3: divide by $\sqrt{d_k}$

STEP 4: softmax

STEP 5: multiply each value vector by the softmax score

STEP 6: sum up the weighted value vectors

Self-Attention: Matrix Calculation

Pack embeddings into matrix X

$$X \times W^Q = Q$$

Multiply X by weight matrices we've trained (W^K , W^Q , W^V)

$$X \times W^K = K$$

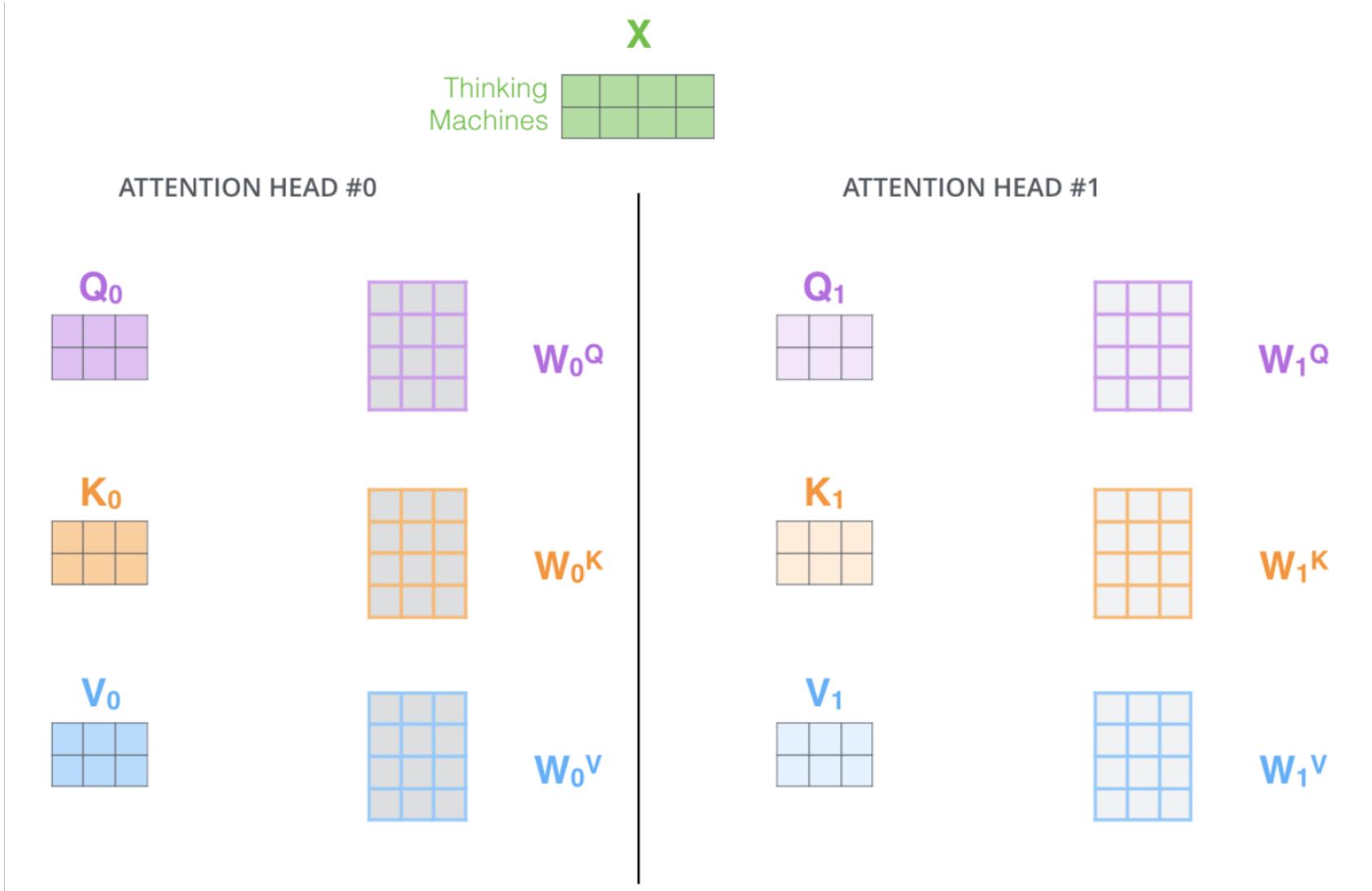
$$X \times W^V = V$$

Self-Attention: Matrix Calculation

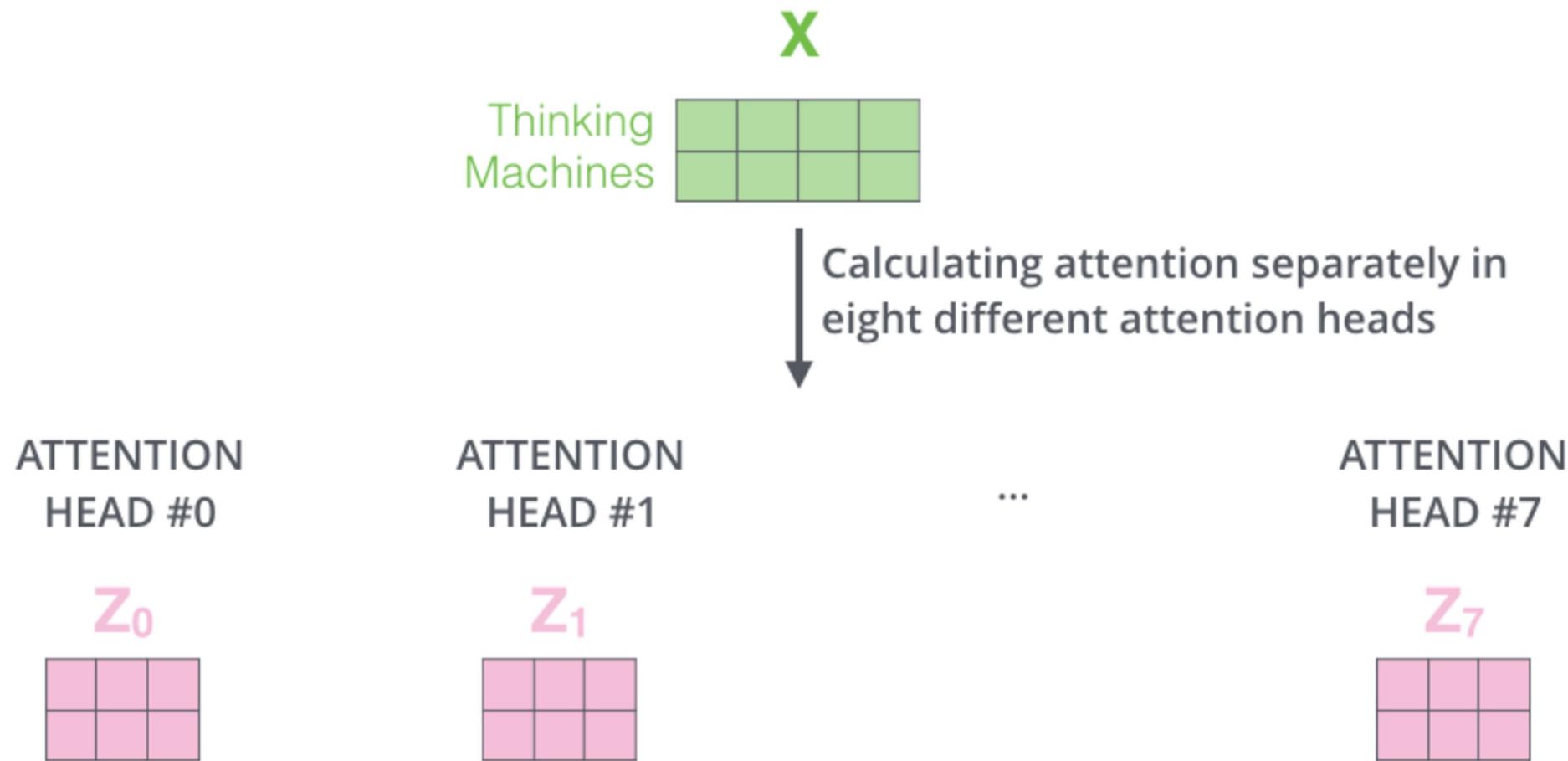
$$\text{softmax} \left(\frac{\begin{matrix} \mathbf{Q} & \mathbf{K^T} \\ \begin{matrix} \text{purple grid} & \times & \begin{matrix} \text{orange grid} \\ \sqrt{d_k} \end{matrix} \end{matrix} \end{matrix}}{\mathbf{V}} \right) = \mathbf{Z}$$

The diagram illustrates the computation of self-attention. It shows the softmax function applied to the product of two matrices, \mathbf{Q} and \mathbf{K}^T , scaled by $\sqrt{d_k}$. The result is then multiplied by matrix \mathbf{V} . The matrices are represented as 2x4 grids of colored squares: purple for \mathbf{Q} , orange for \mathbf{K}^T , and blue for \mathbf{V} . The final output \mathbf{Z} is a 2x4 grid of pink squares.

Multi-Head Attention



Multi-Head Attention



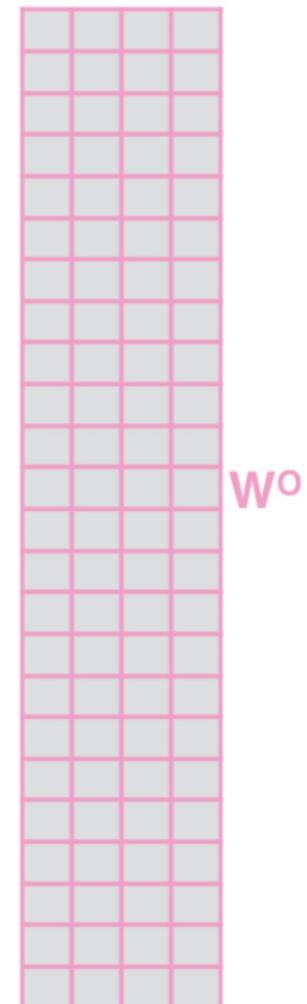
Multi-Head Attention

1) Concatenate all the attention heads



2) Multiply with a weight matrix W^o that was trained jointly with the model

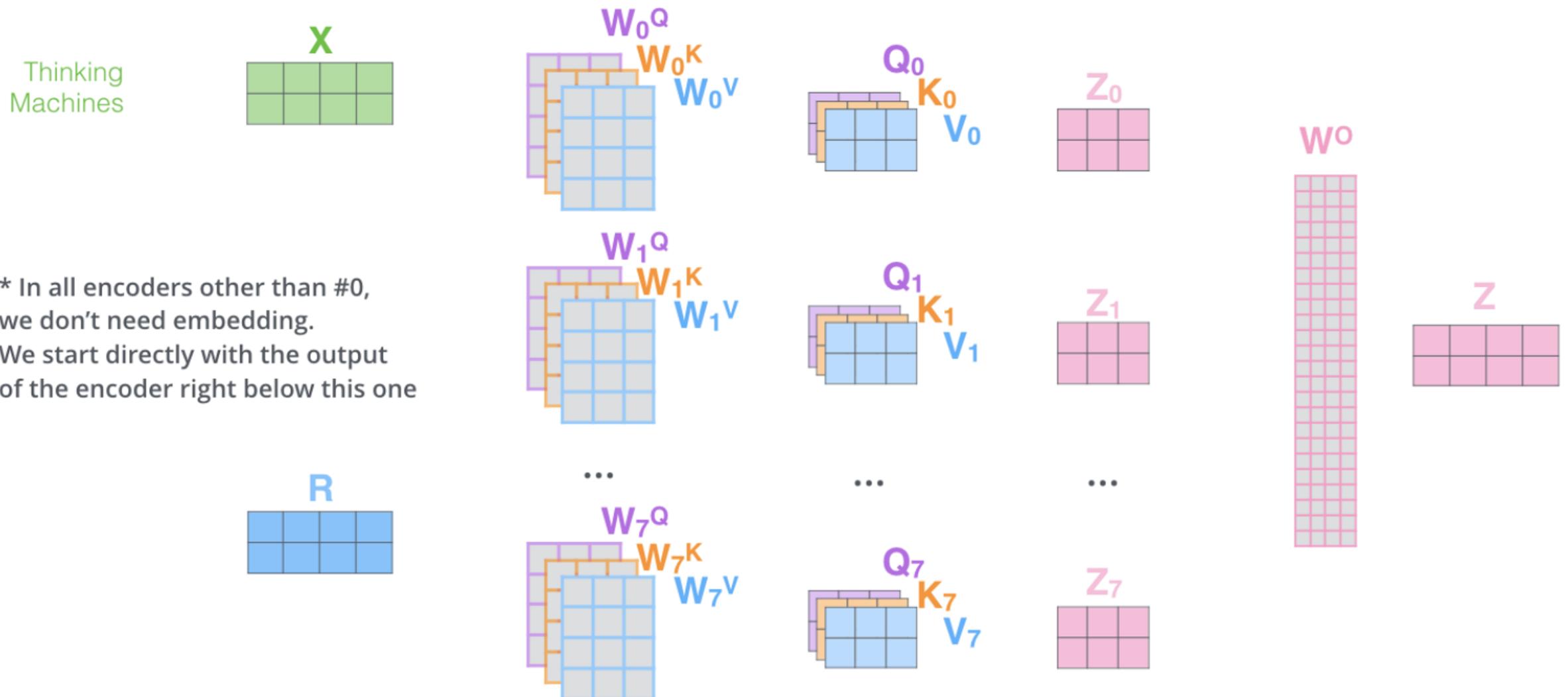
X



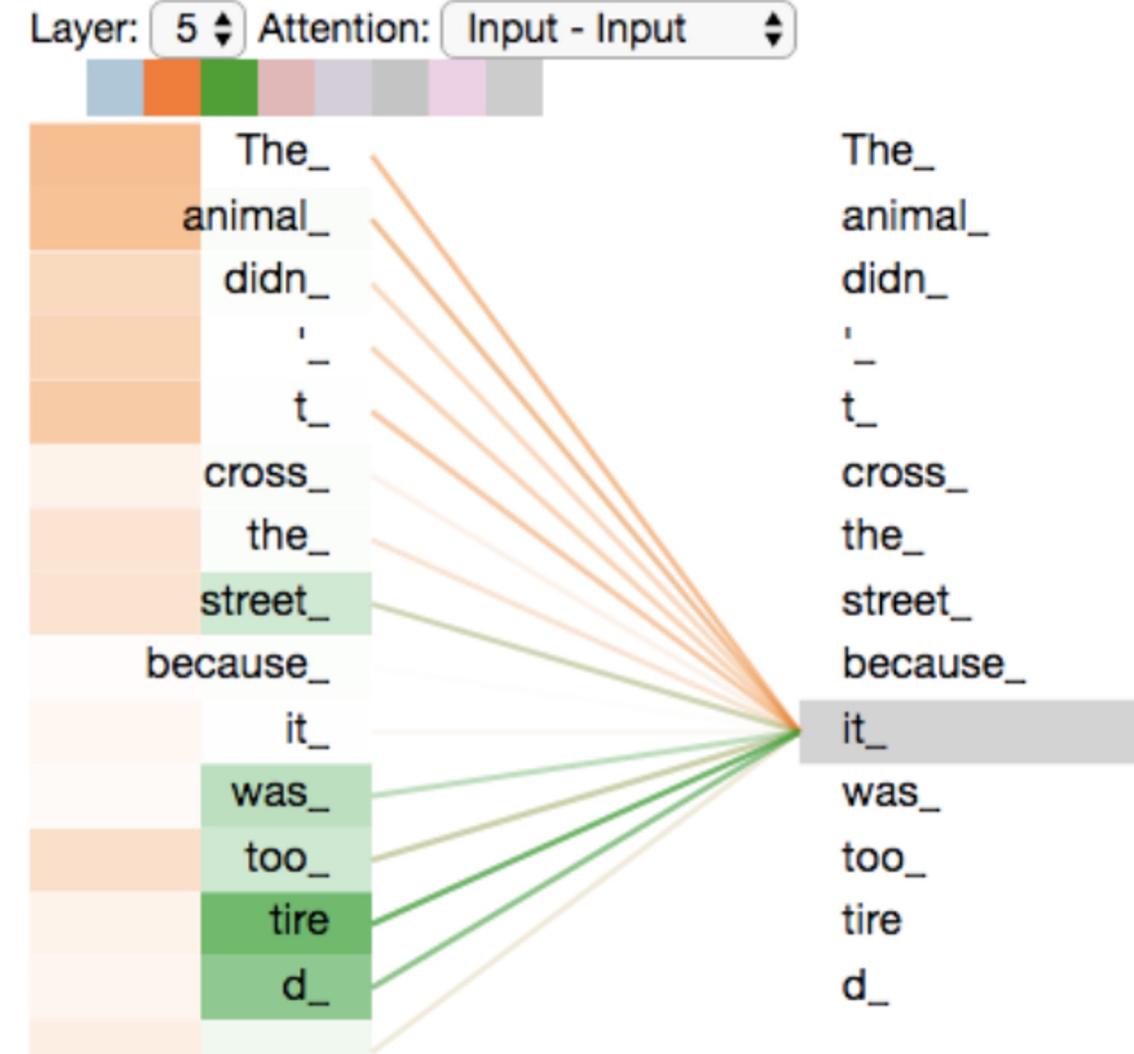
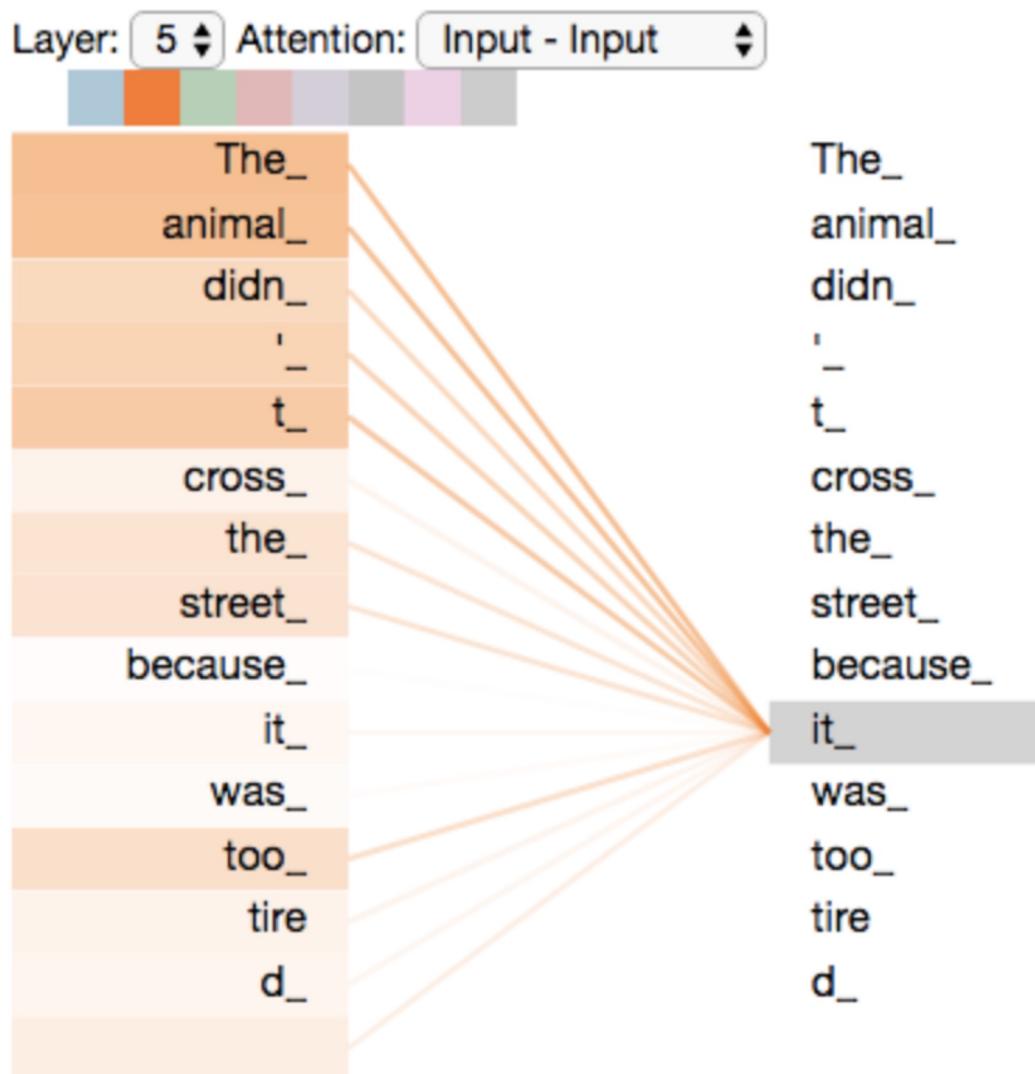
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

$$= \begin{matrix} Z \\ \hline \text{---} \\ \text{---} \end{matrix}$$

- 1) This is our input sentence*
- 2) We embed each word*
- 3) Split into 8 heads. We multiply X or R with weight matrices
- 4) Calculate attention using the resulting $Q/K/V$ matrices
- 5) Concatenate the resulting Z matrices, then multiply with weight matrix W^o to produce the output of the layer

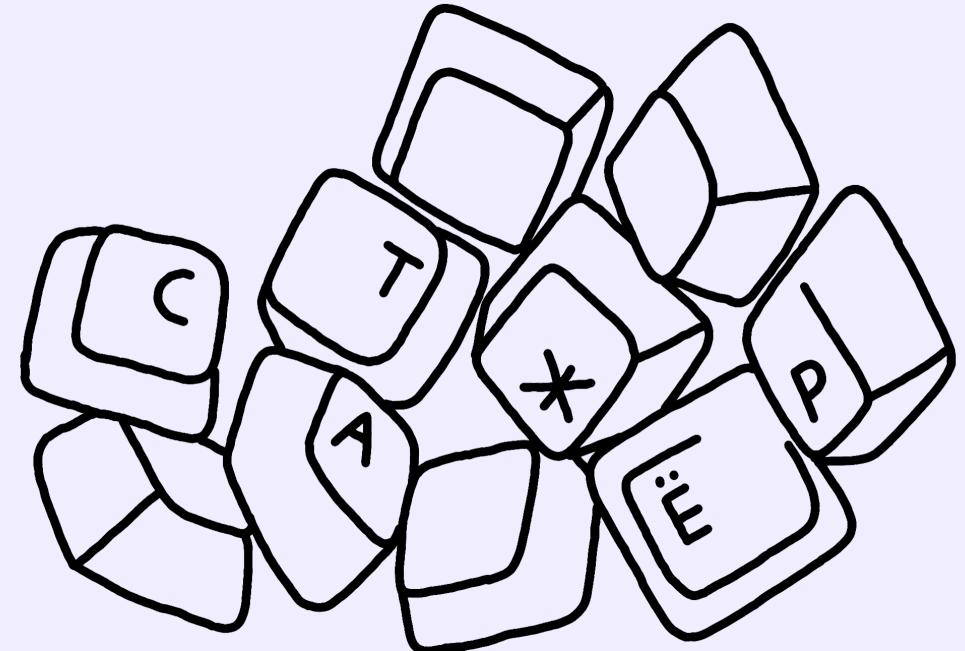


Multi-Head Attention



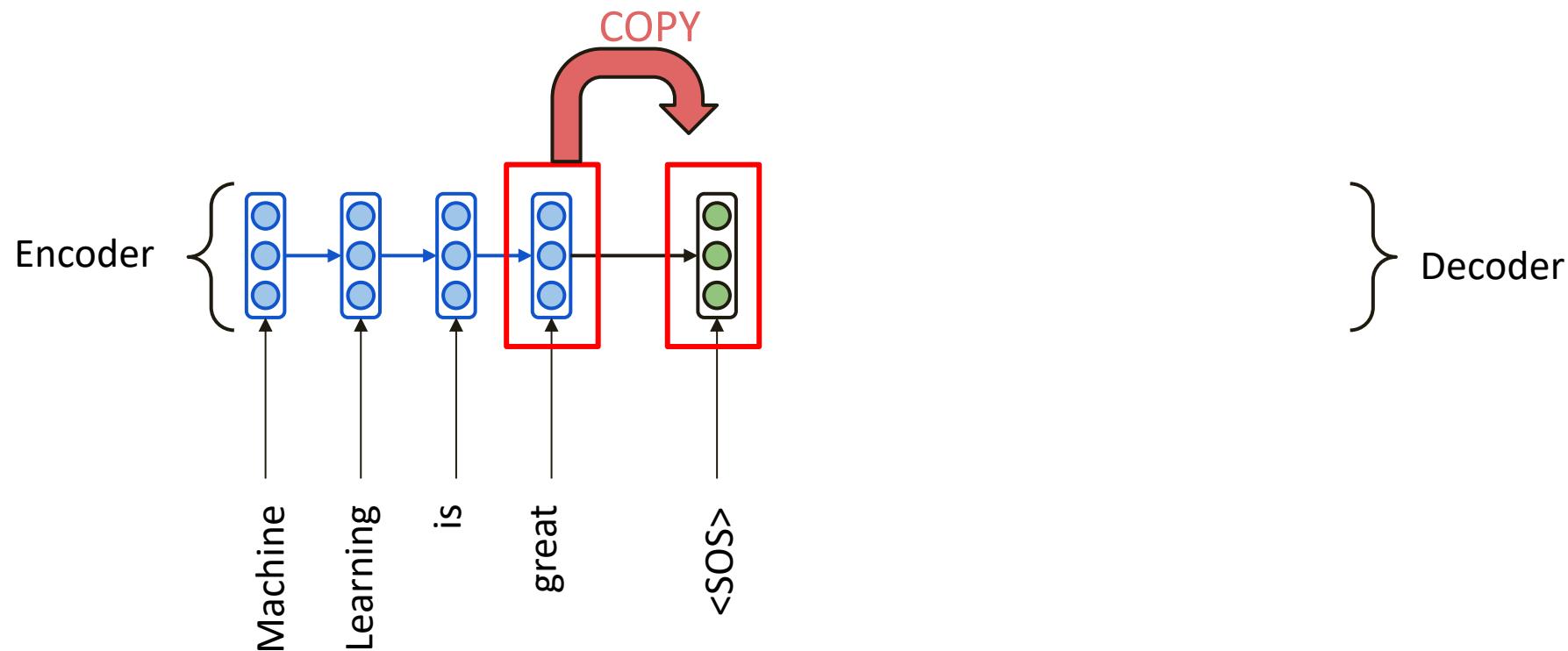
Outro про машинный перевод

05

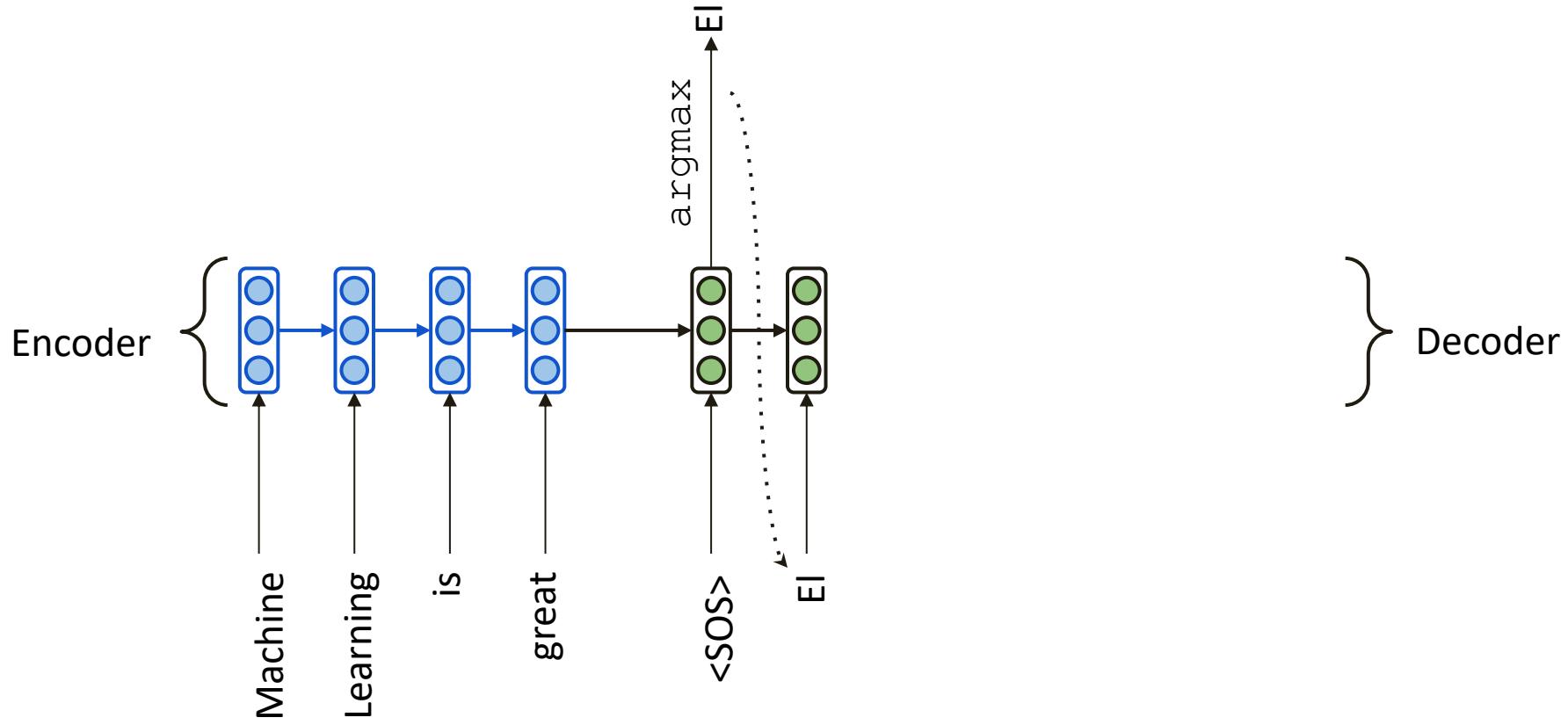


Seq2seq NMT

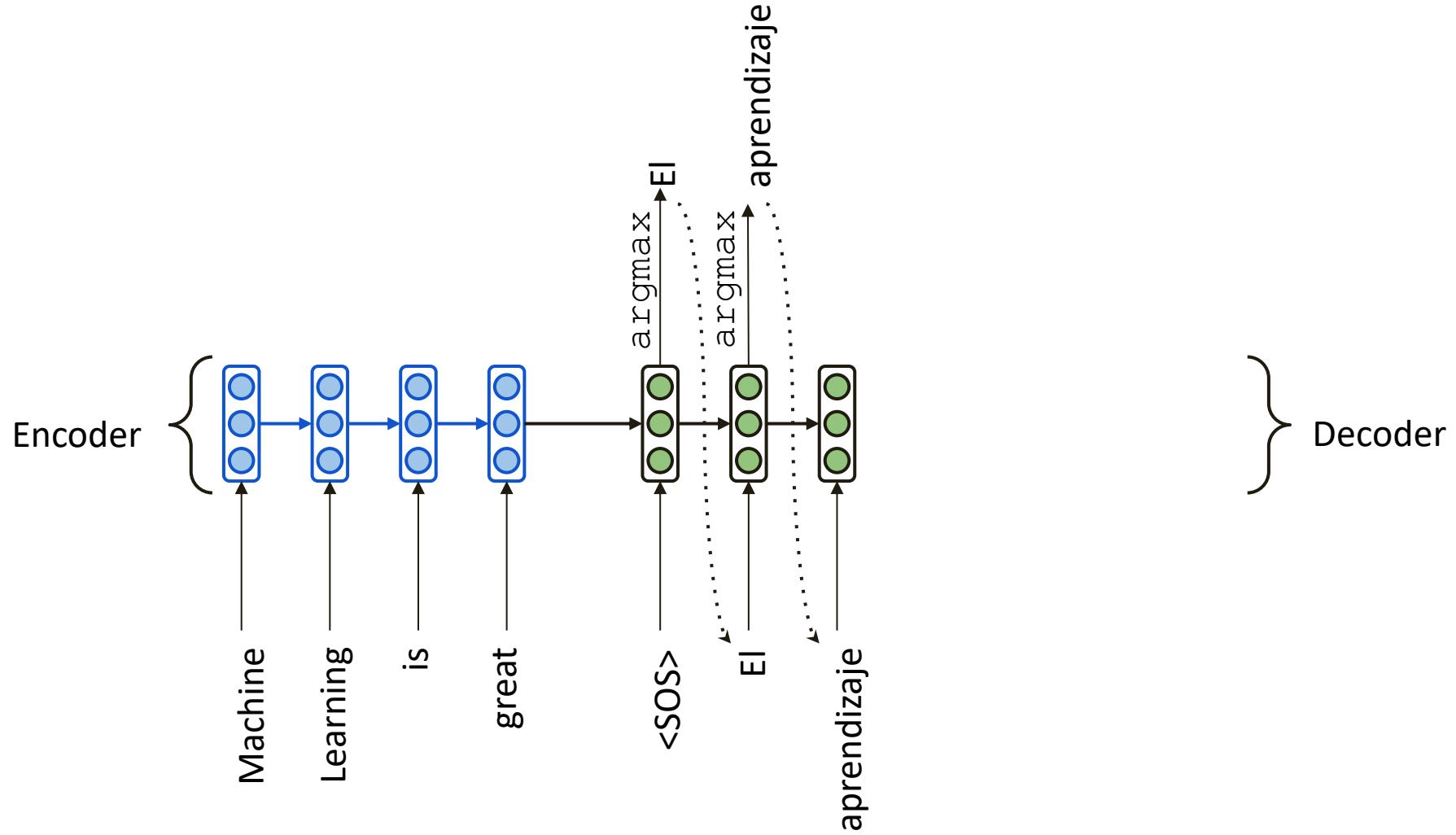
Две языковые модели: для
исходного и целевого языков



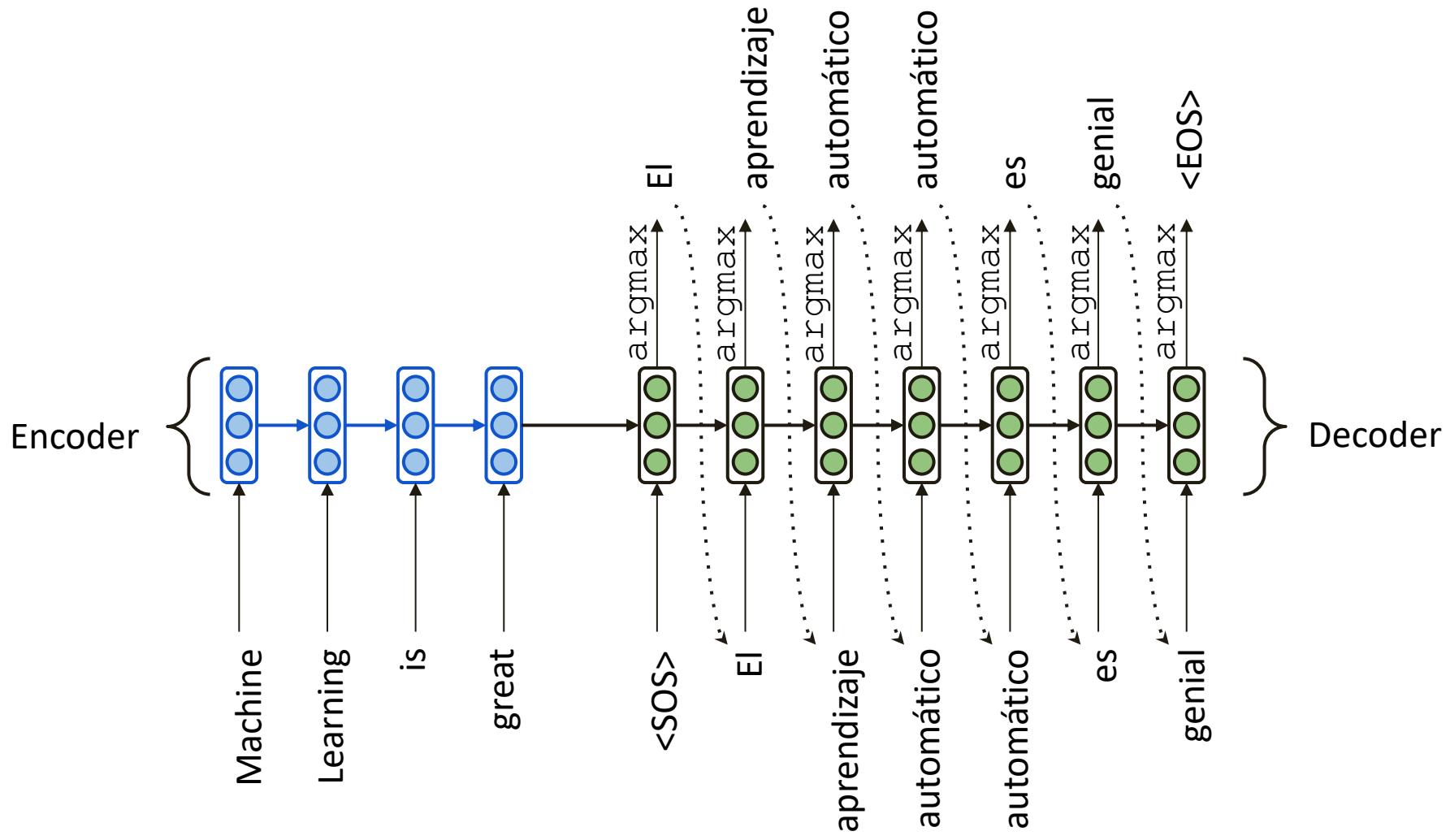
Seq2seq машинный перевод



Seq2seq NMT



Seq2seq NMT



Правдоподобие последовательности

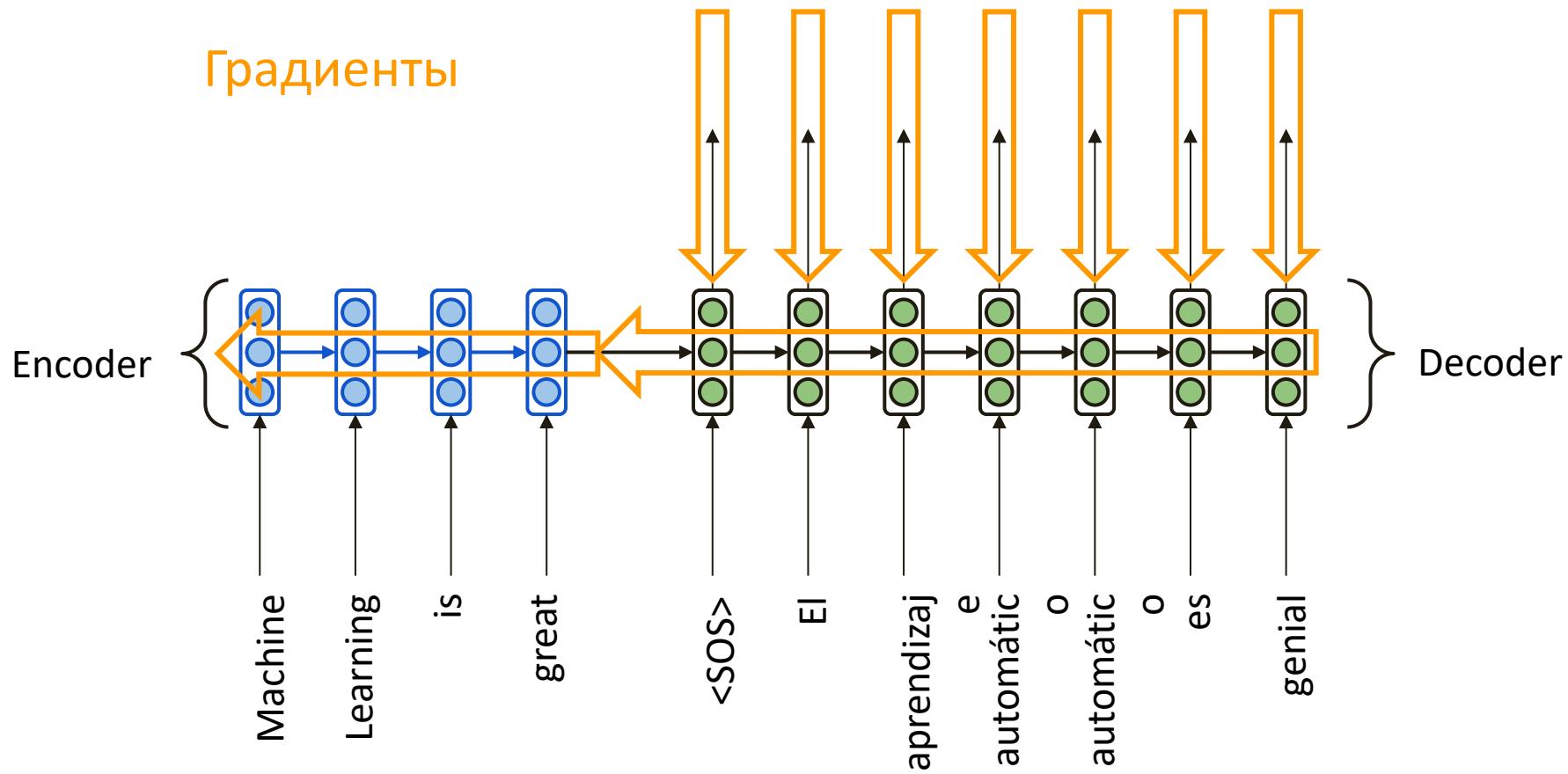
- Мы можем оценить правдоподобие сгенерированной последовательности на основе самой модели:

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

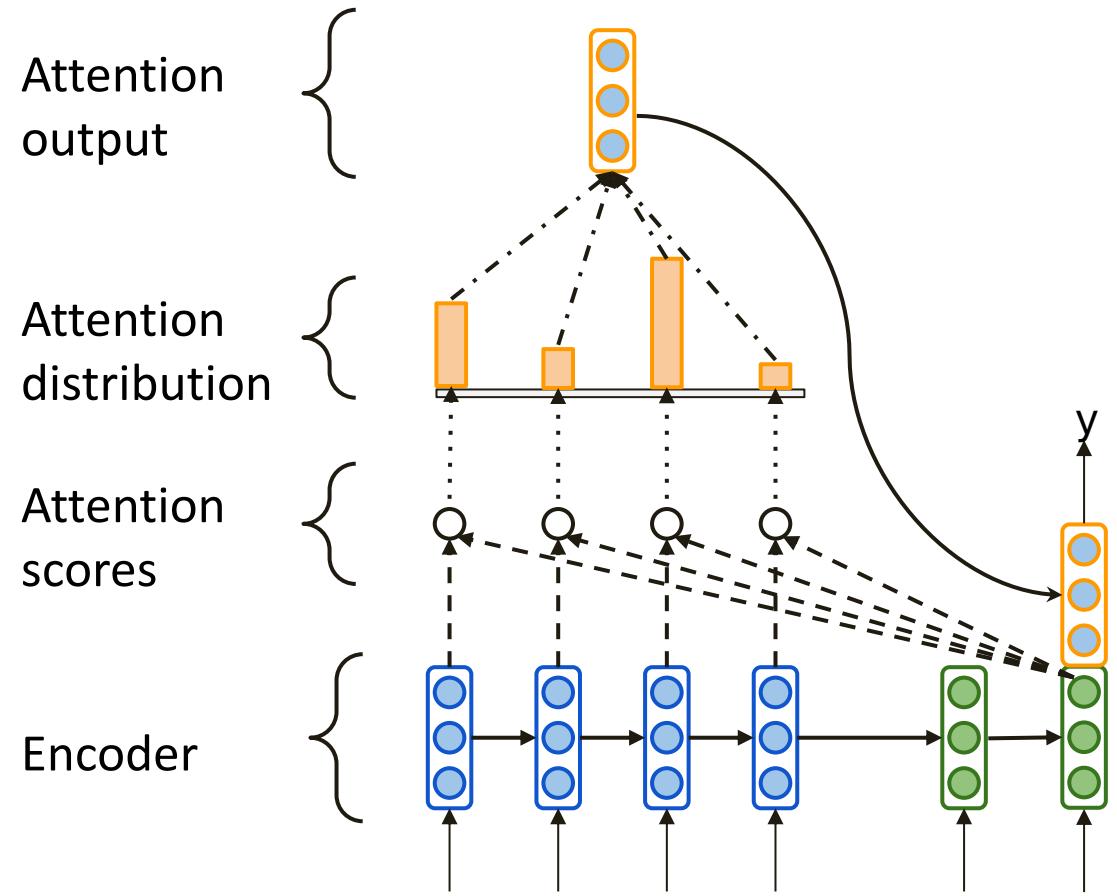
- у – сгенерированные элементы последовательности
- х – некоторая дополнительная информация:
 - начальное состояние RNN, подводка и пр.
 - или фраза на исходном языке**

Вероятность слова на
шаге Т

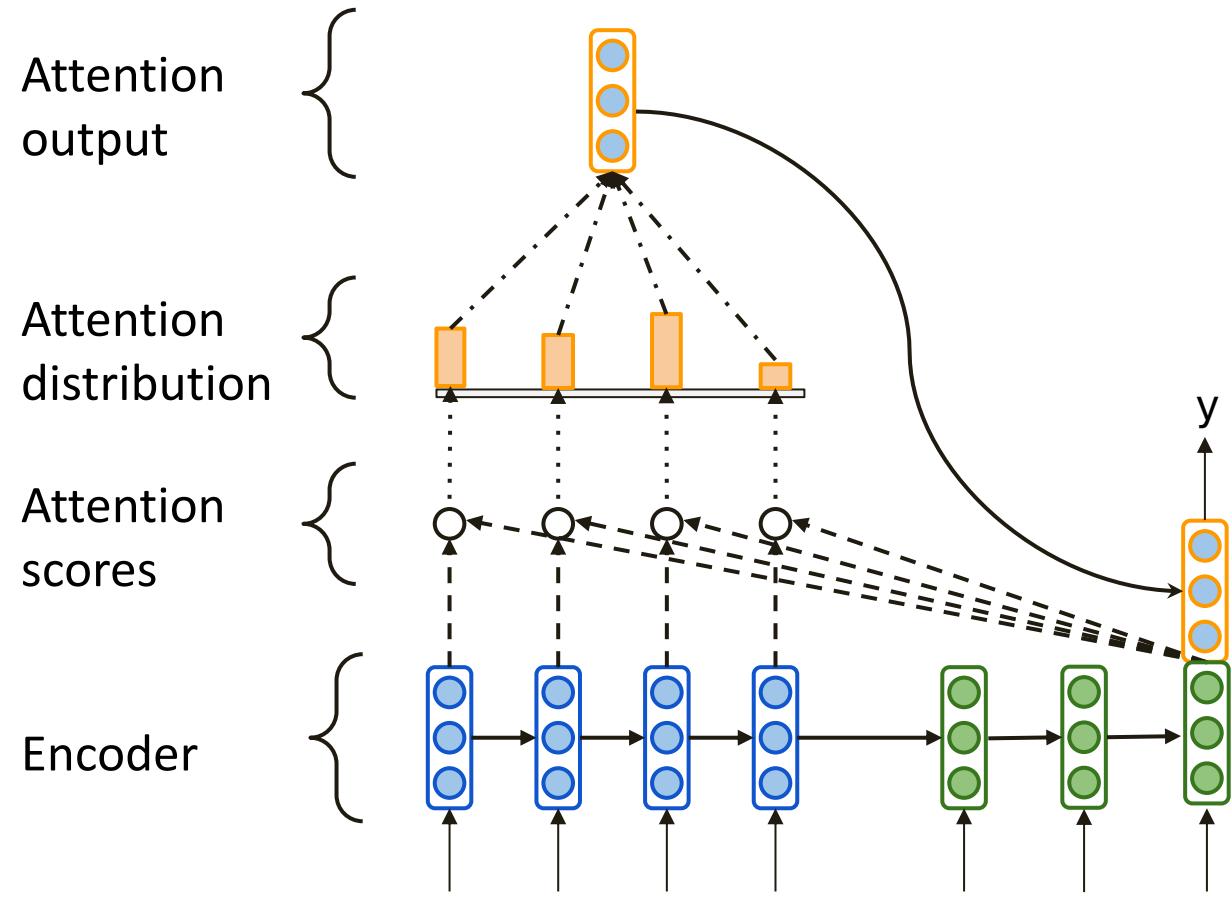
Можно обучать end-to-end



Seq2seq with attention



Seq2seq with attention



Спасибо за внимание



Y&B