

# Языковые модели на основе нейронных сетей

## Лекция 10

Майоров Владимир Дмитриевич

12 ноября 2025

# Задача языкового моделирования

Языковая модель – это вероятностное распределение на множестве последовательностей слов:

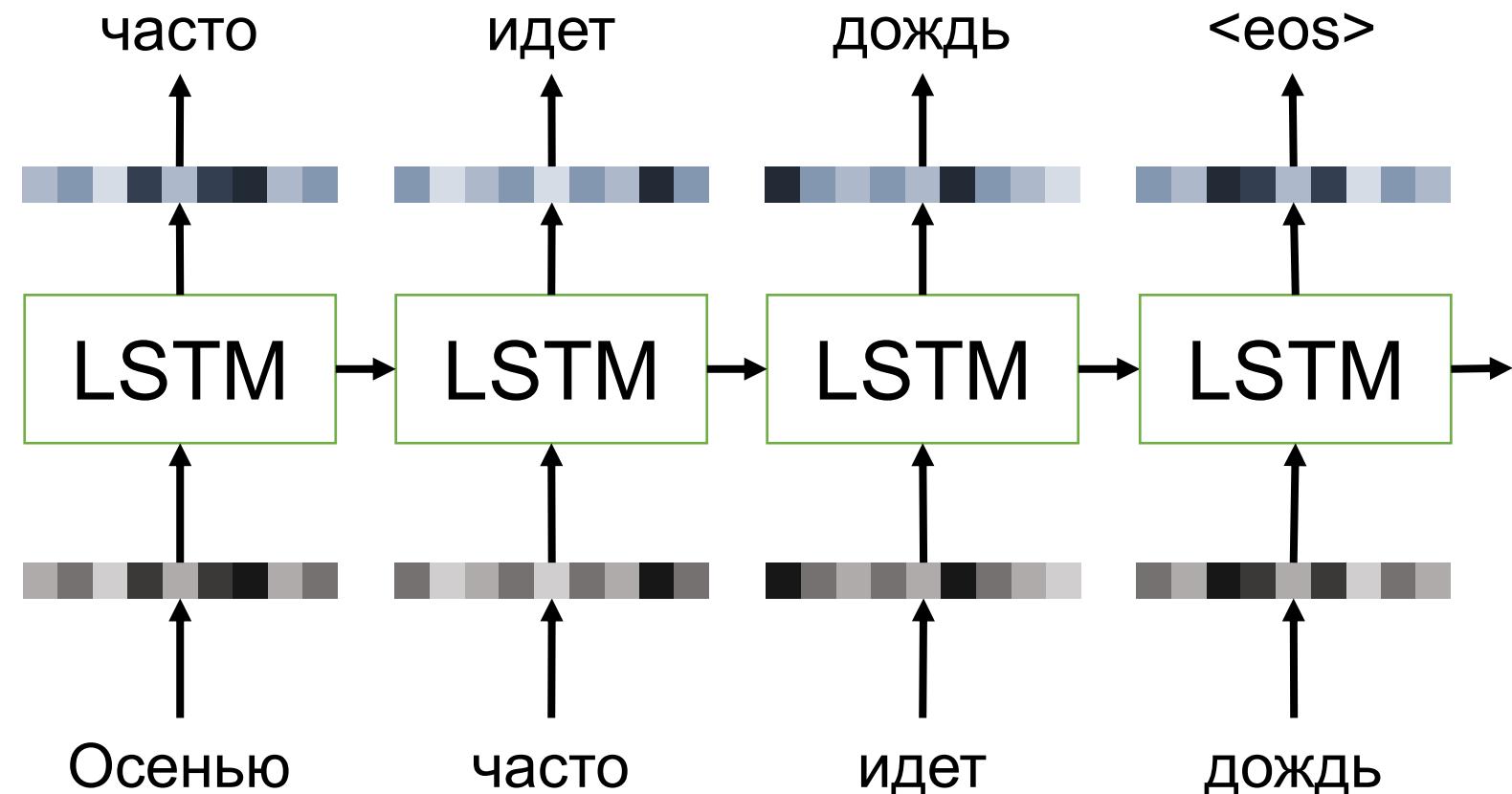
- $P(w_1, w_2, \dots, w_n)$

$P(\text{из окна сильно дуло}) \gg P(\text{дуло окна из сильно})$

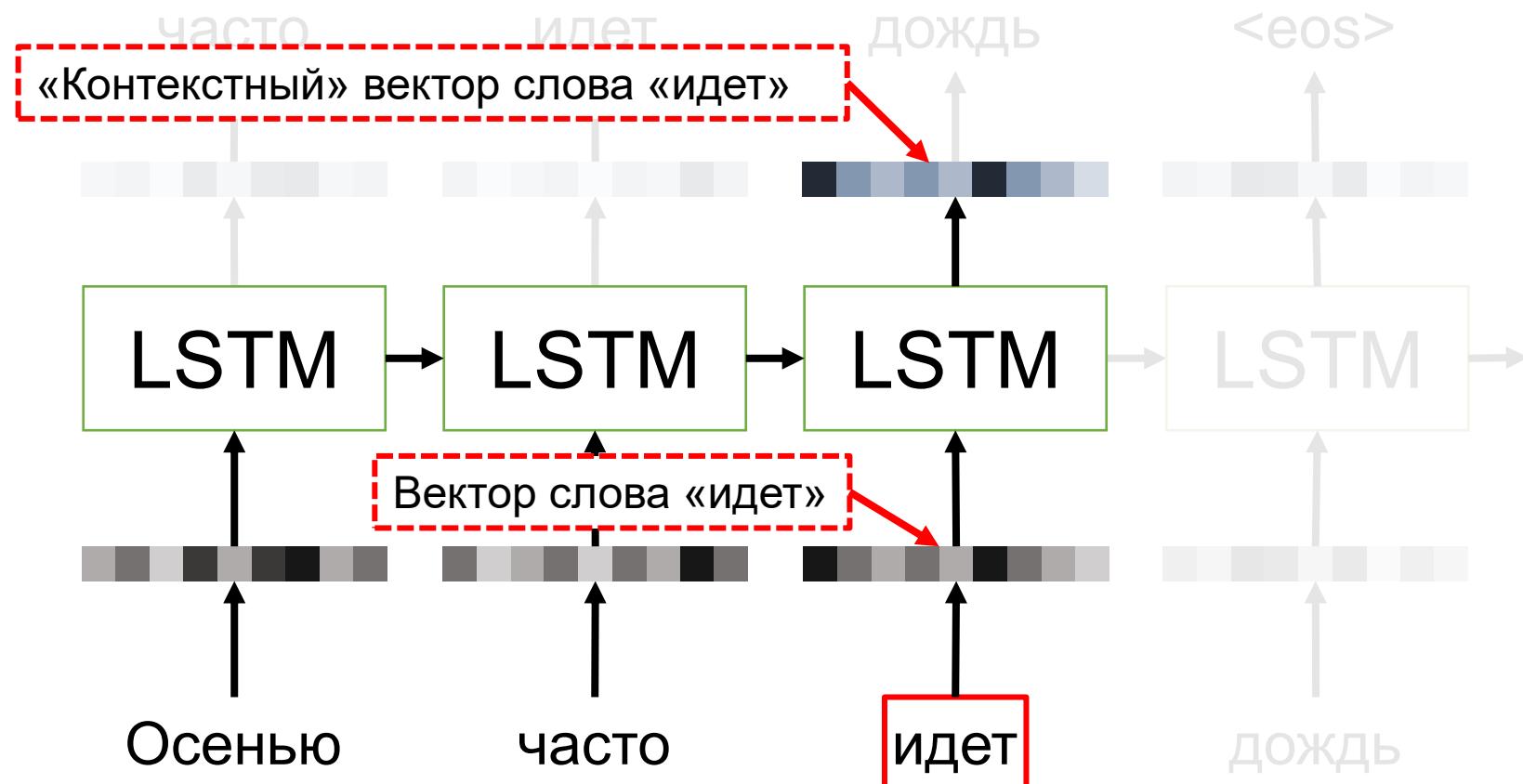
- $P(w_n | w_1, \dots, w_{n-1})$

- Осенью часто идет ...
- Село Коровка в качестве ...

# Языковые модели на RNN

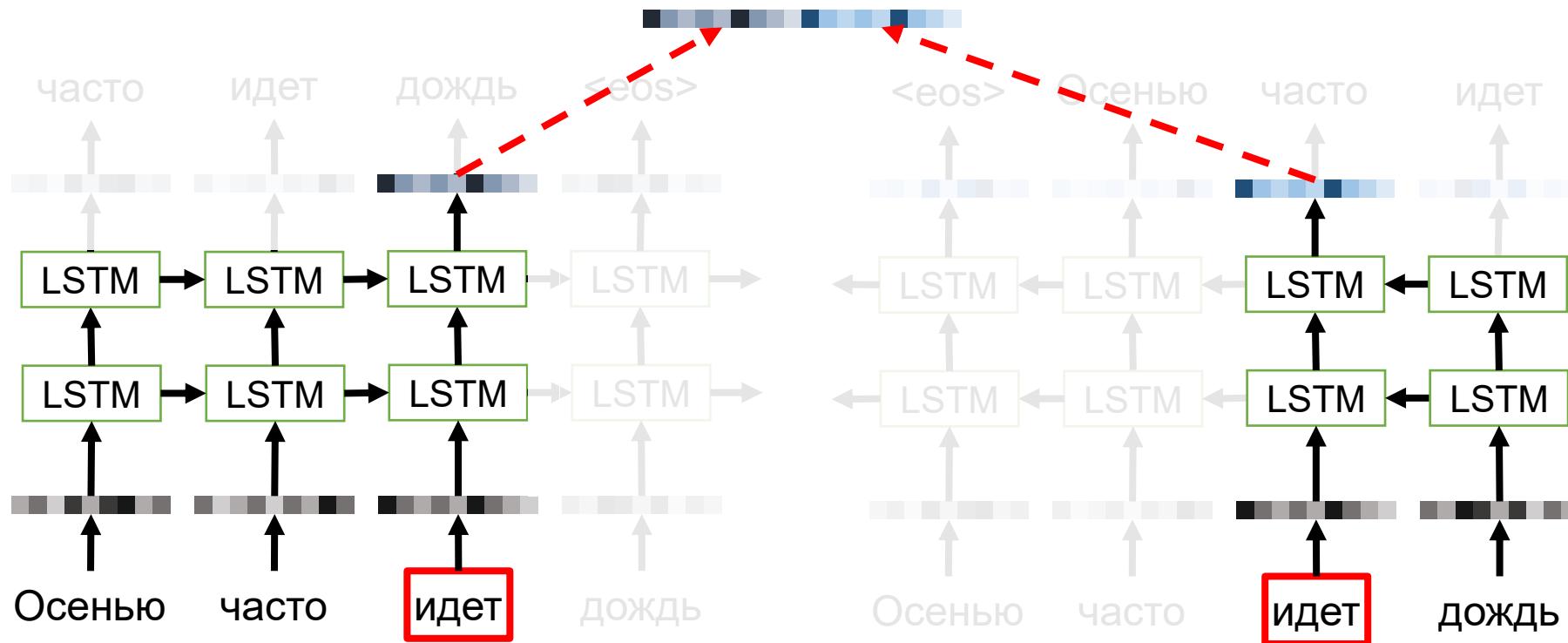


# Языковые модели на RNN

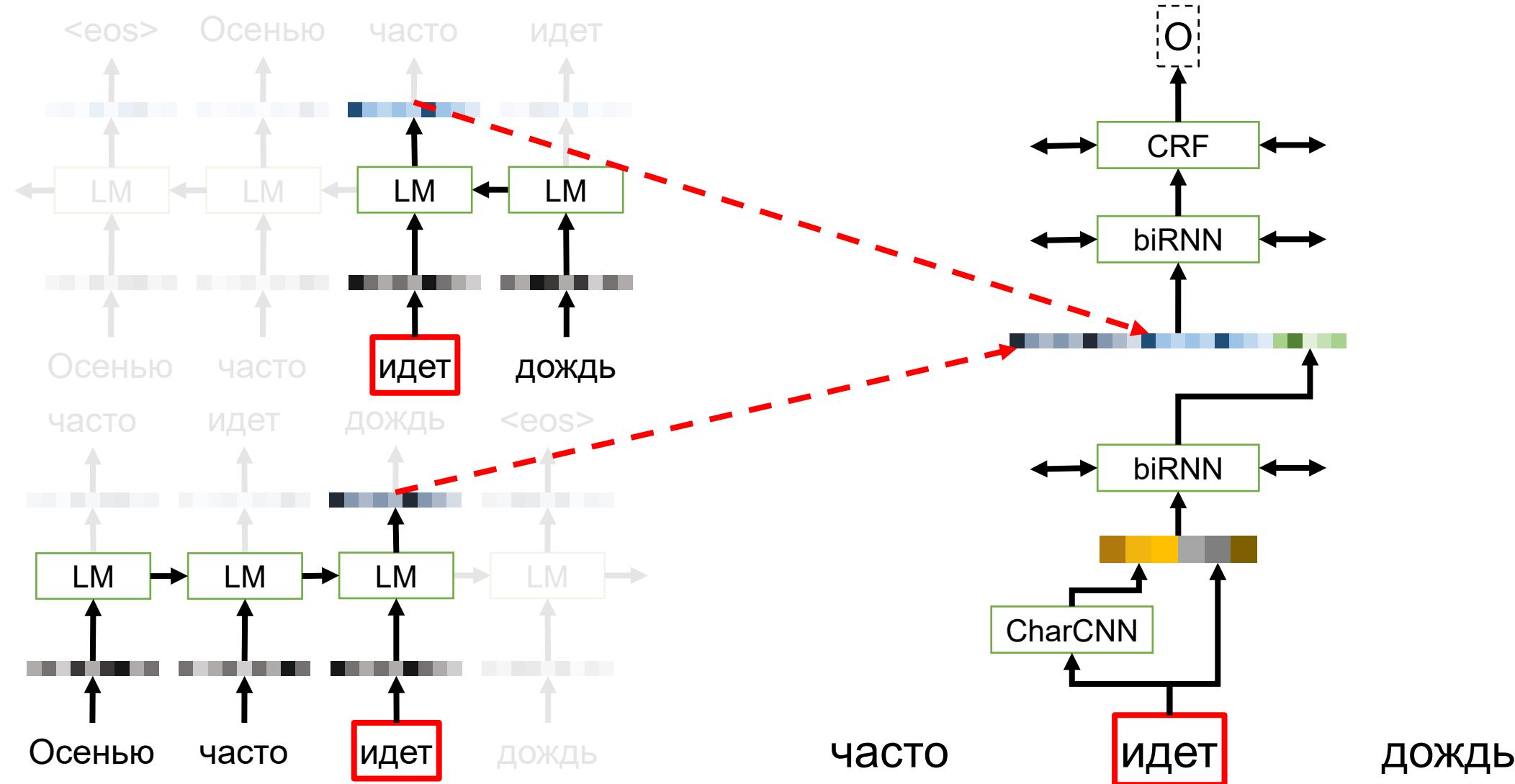


# Language Model Embedding

- Прямая модель:  $p(w_1, \dots, w_N) = \prod_{k=1}^N p(w_k | w_1, \dots, w_{k-1})$
- Обратная модель:  $p(w_1, \dots, w_N) = \prod_{k=1}^N p(w_k | w_{k+1}, \dots, w_N)$



# Language Model Embedding



# ELMo (Embeddings from Language Models)

- Прямая модель:

$$p(w_1, \dots, w_N) = \prod_{k=1}^N p(w_k | w_1, \dots, w_{k-1})$$

- Обратная модель:

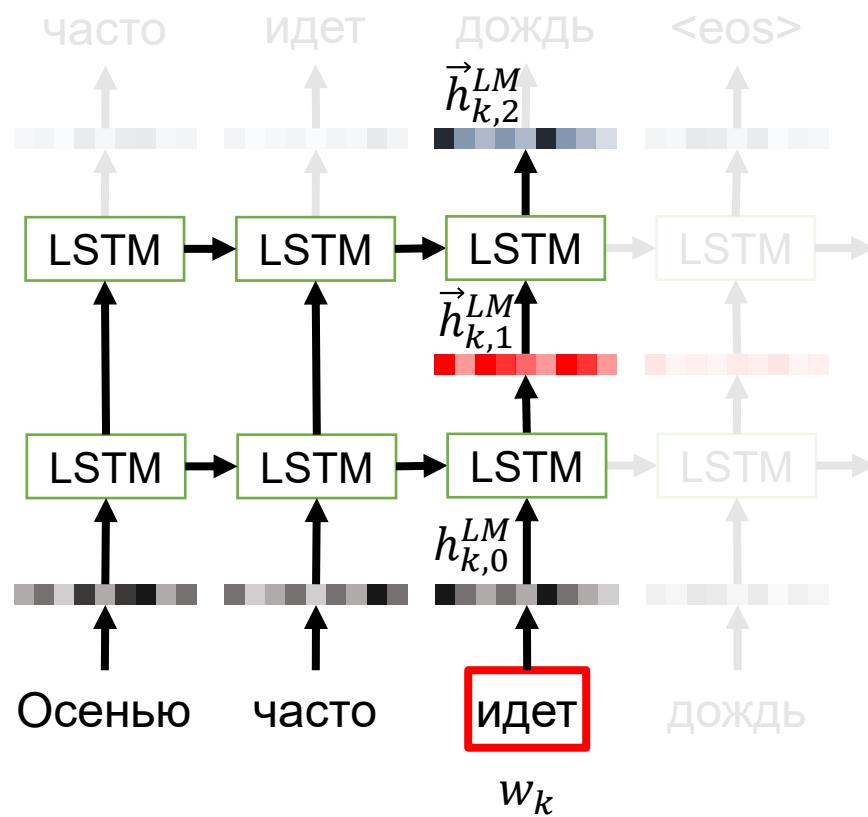
$$p(w_1, \dots, w_N) = \prod_{k=1}^N p(w_k | w_{k+1}, \dots, w_N)$$

- Обучаем прямую и обратную языковую модель одновременно:

$$\begin{aligned} J = & \sum_{k=1}^N [\log p(w_k | w_1, \dots, w_{k-1}; \Theta_x, \Theta_{\overrightarrow{LSTM}}, \Theta_s) \\ & + \log p(w_k | w_{k+1}, \dots, w_N; \Theta_x, \Theta_{\overleftarrow{LSTM}}, \Theta_s)] \end{aligned}$$

# ELMo (Embeddings from Language Models)

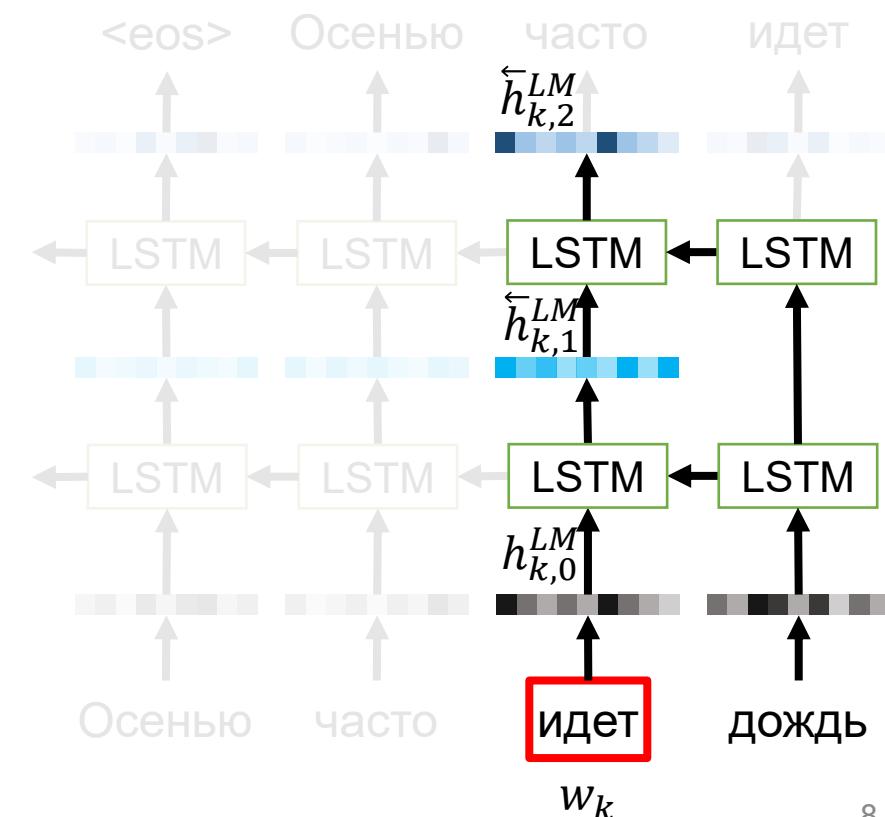
$$ELMo_k = \gamma^{task} \sum_{j=0}^L s_j^{task} h_{k,j}^{LM}, \text{ где } h_{k,i}^{LM} = [\vec{h}_{k,i}^{LM}; \hat{h}_{k,i}^{LM}], i = \overline{1, n}$$



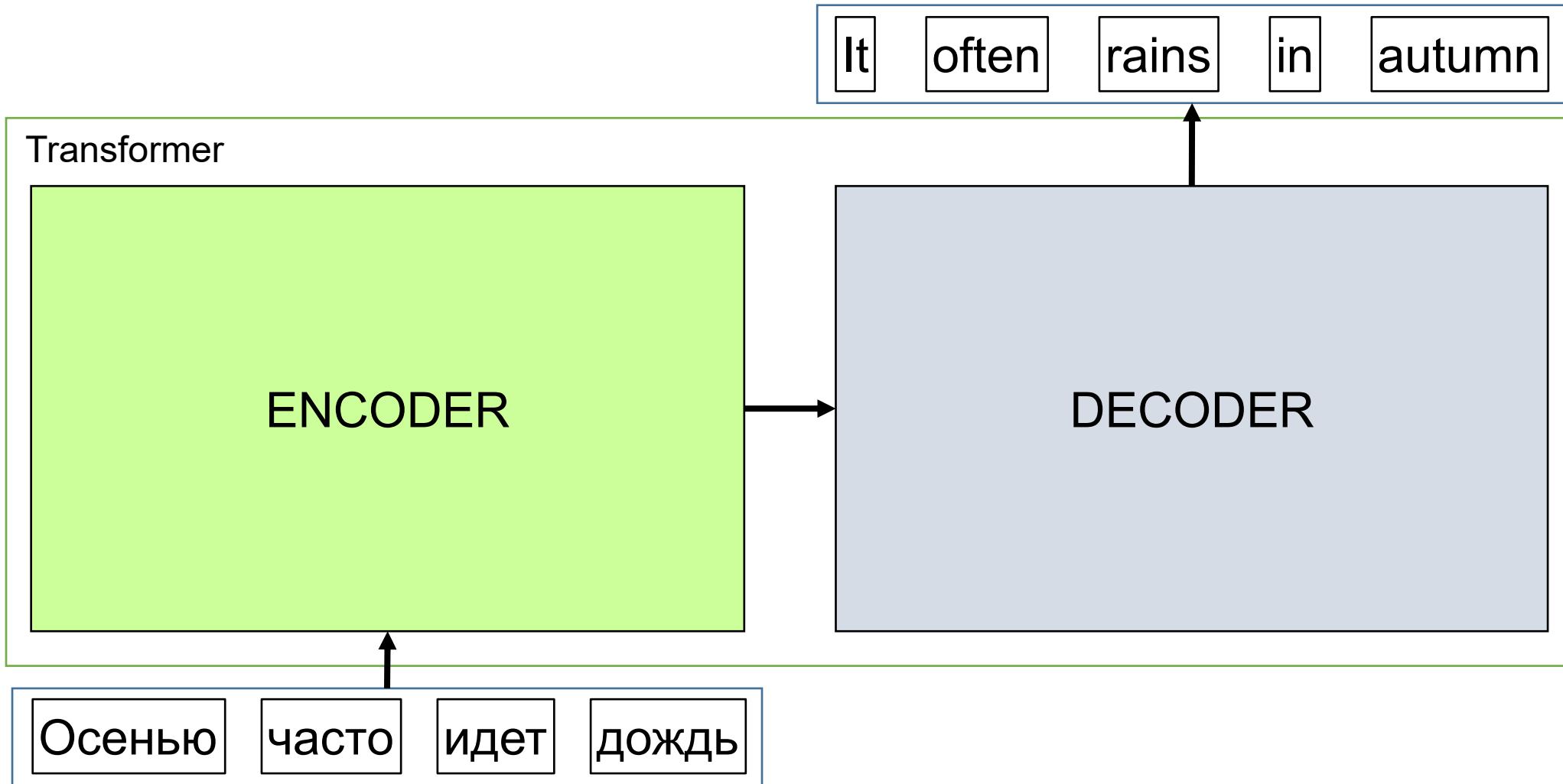
$$s_0 * \text{[grey]} = \text{[purple]}$$

$$+ \quad s_1 * \text{[red]} = \text{[blue]}$$

$$+ \quad s_2 * \text{[dark blue]} = \text{[light blue]}$$

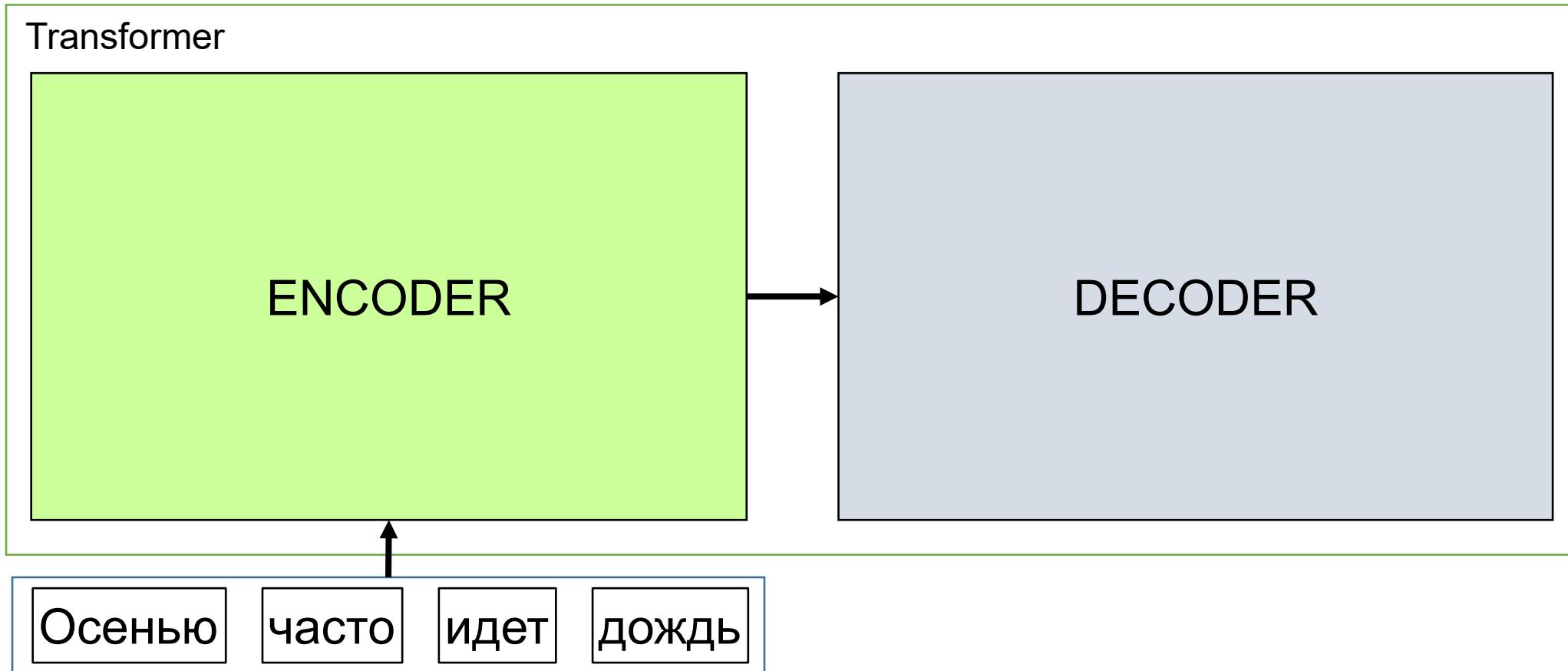


# Transformer\*

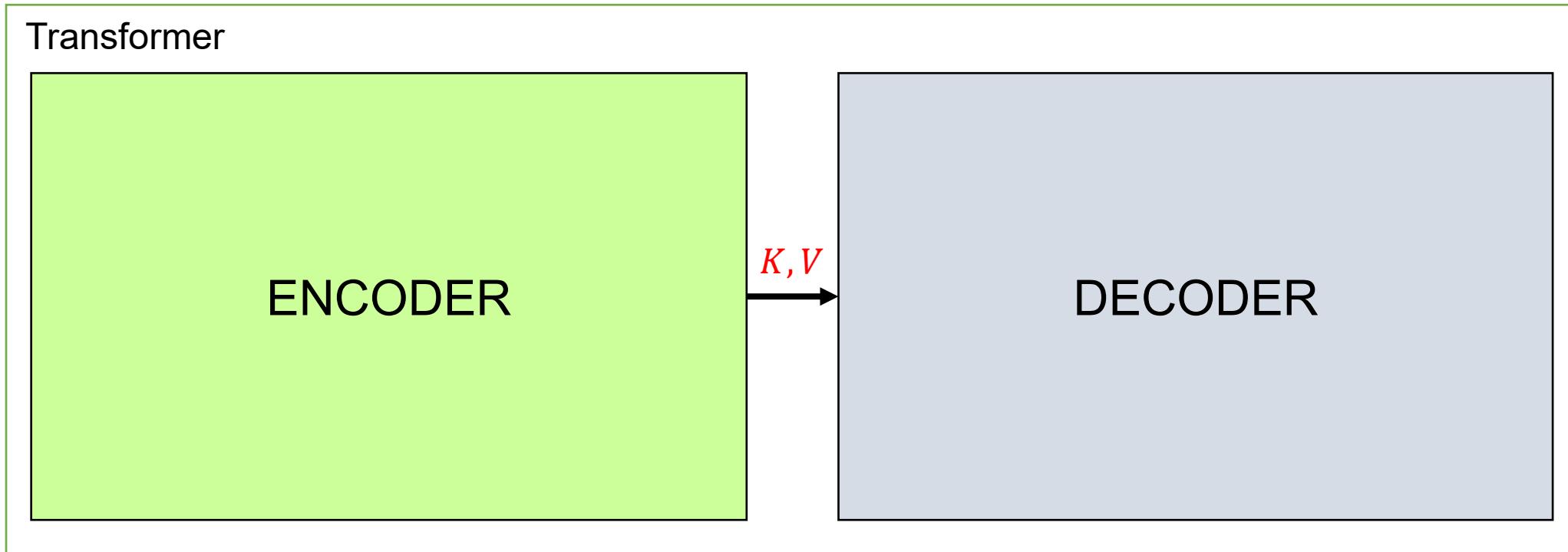


\*подробное объяснение про трансформер есть [здесь](#)

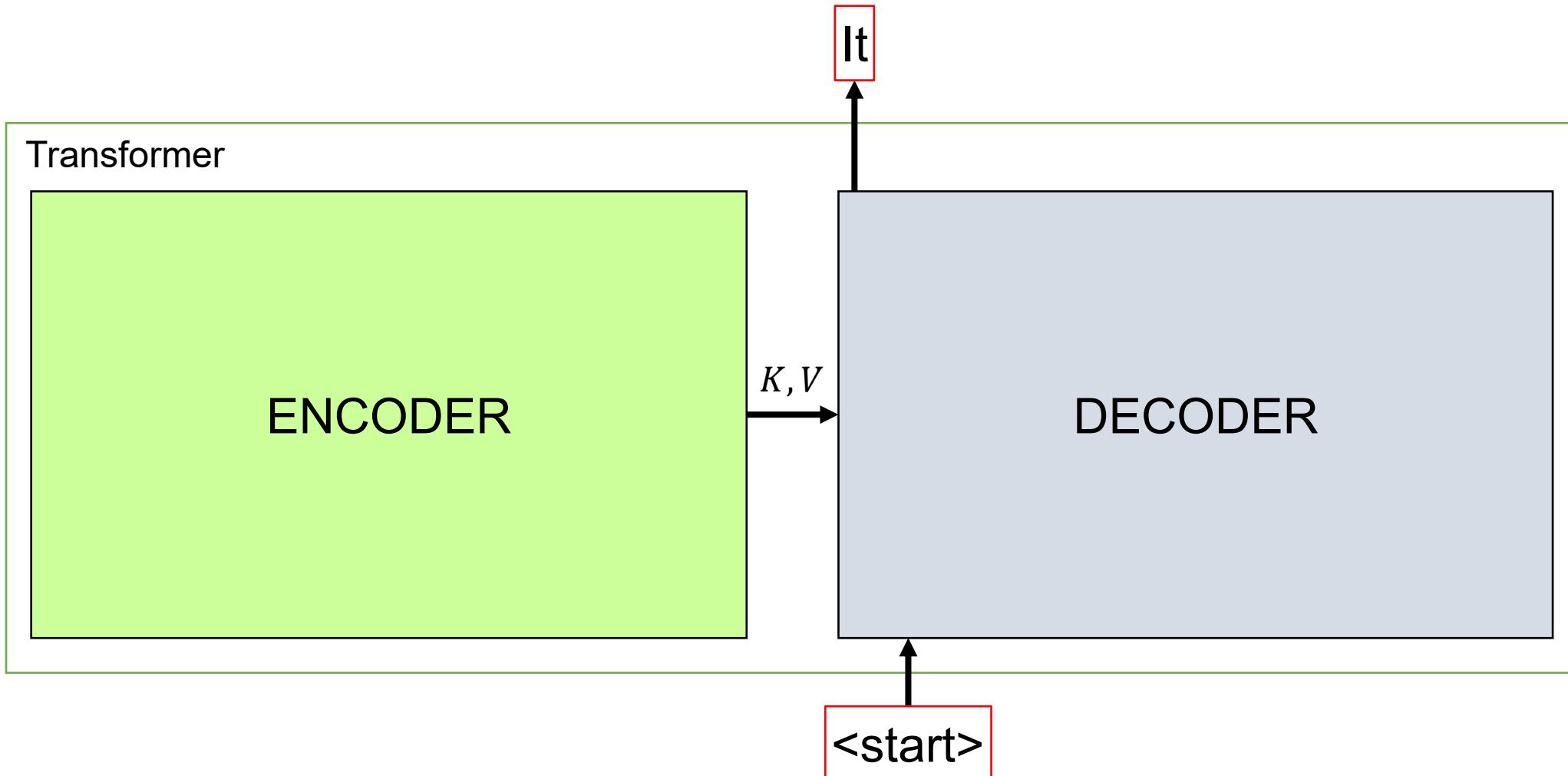
# Transformer



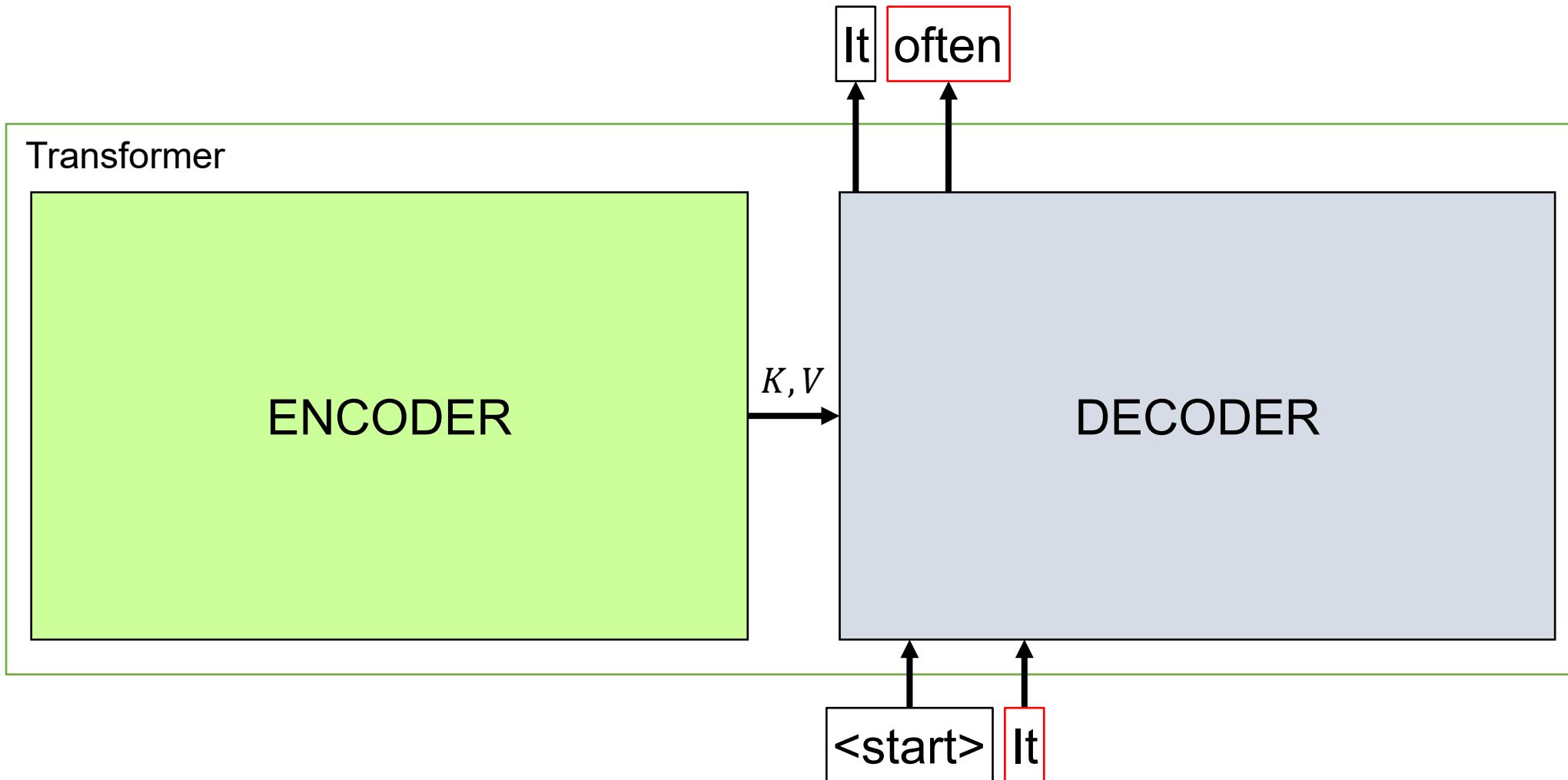
# Transformer



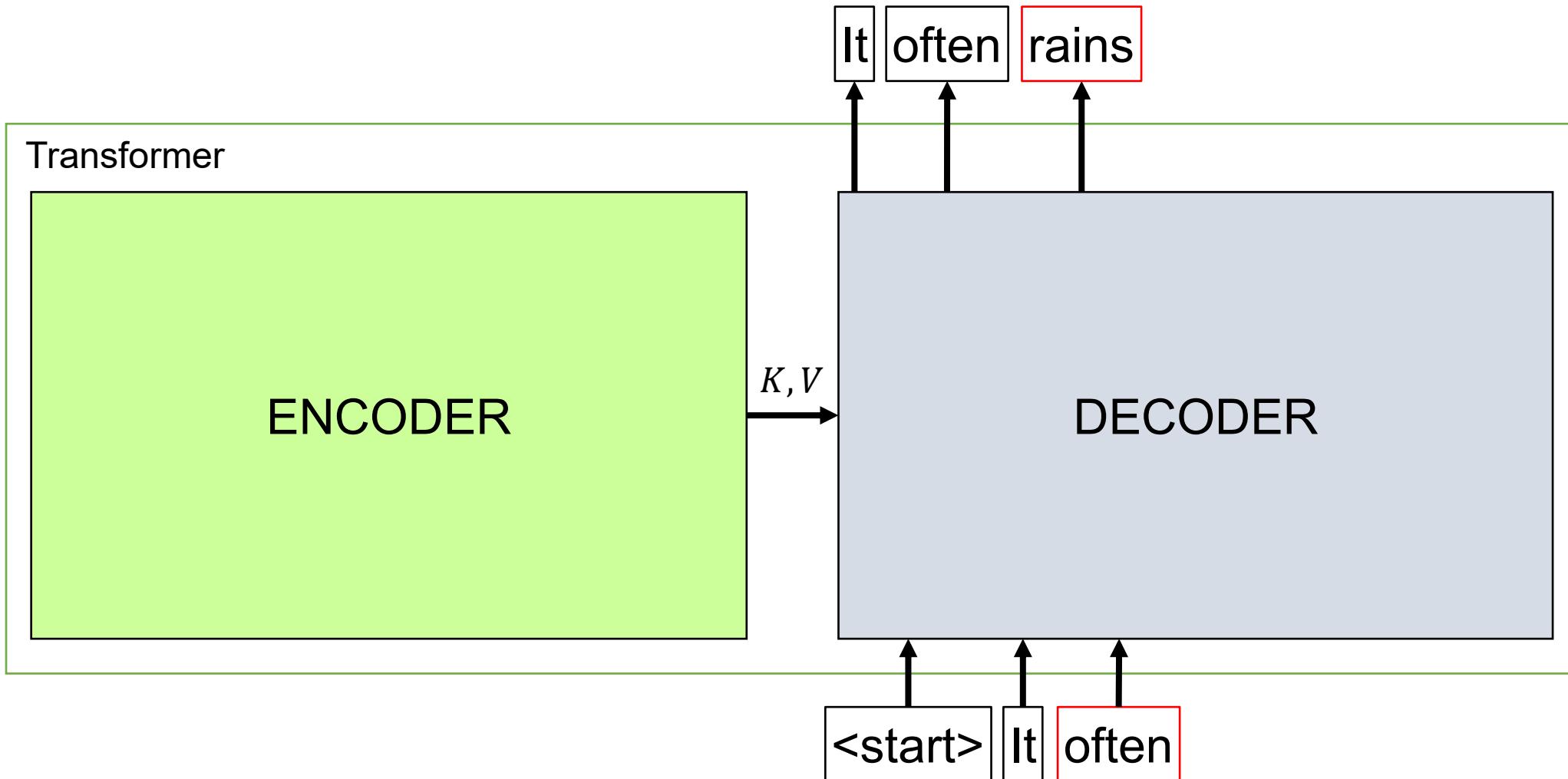
# Transformer



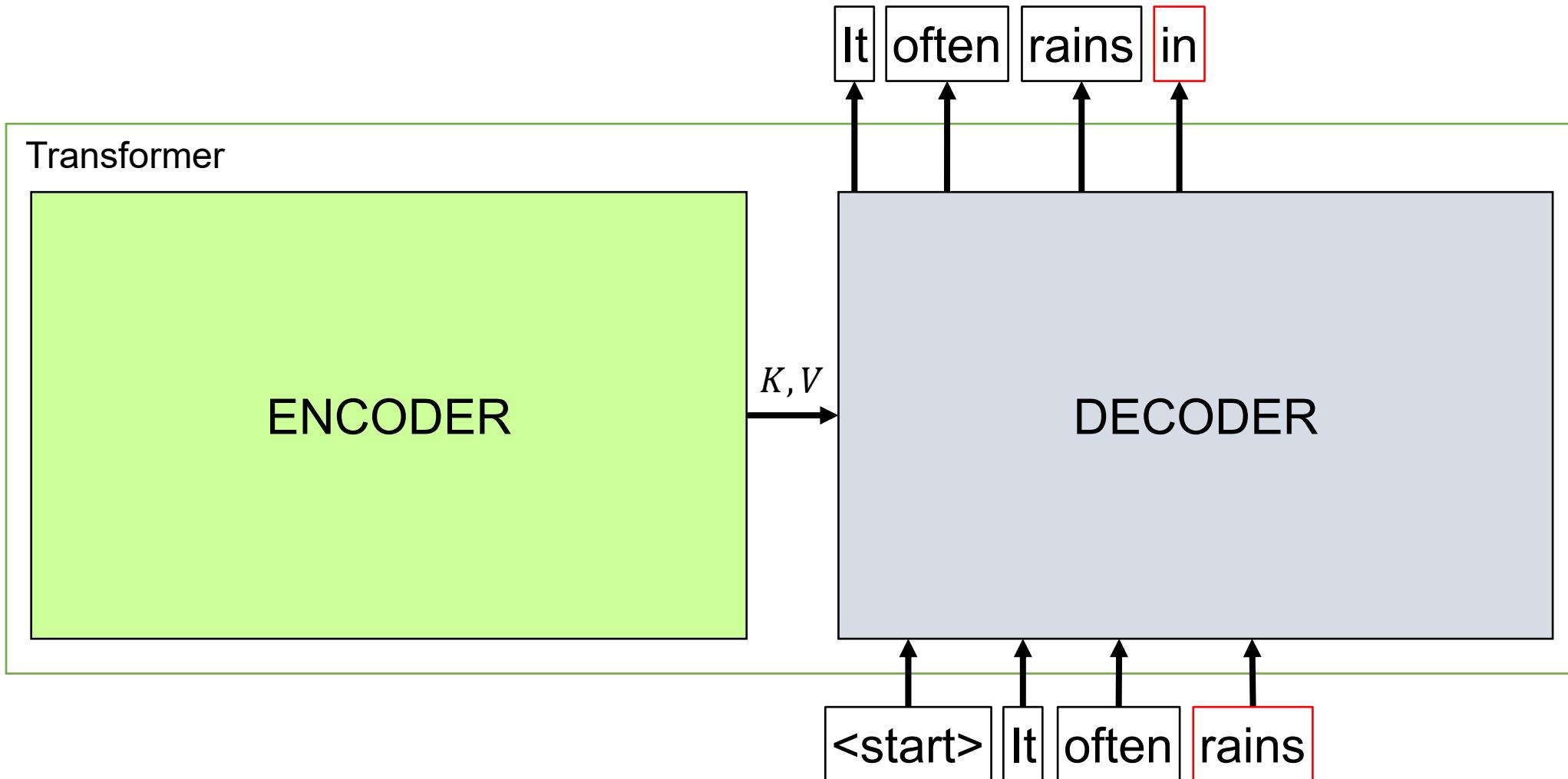
# Transformer



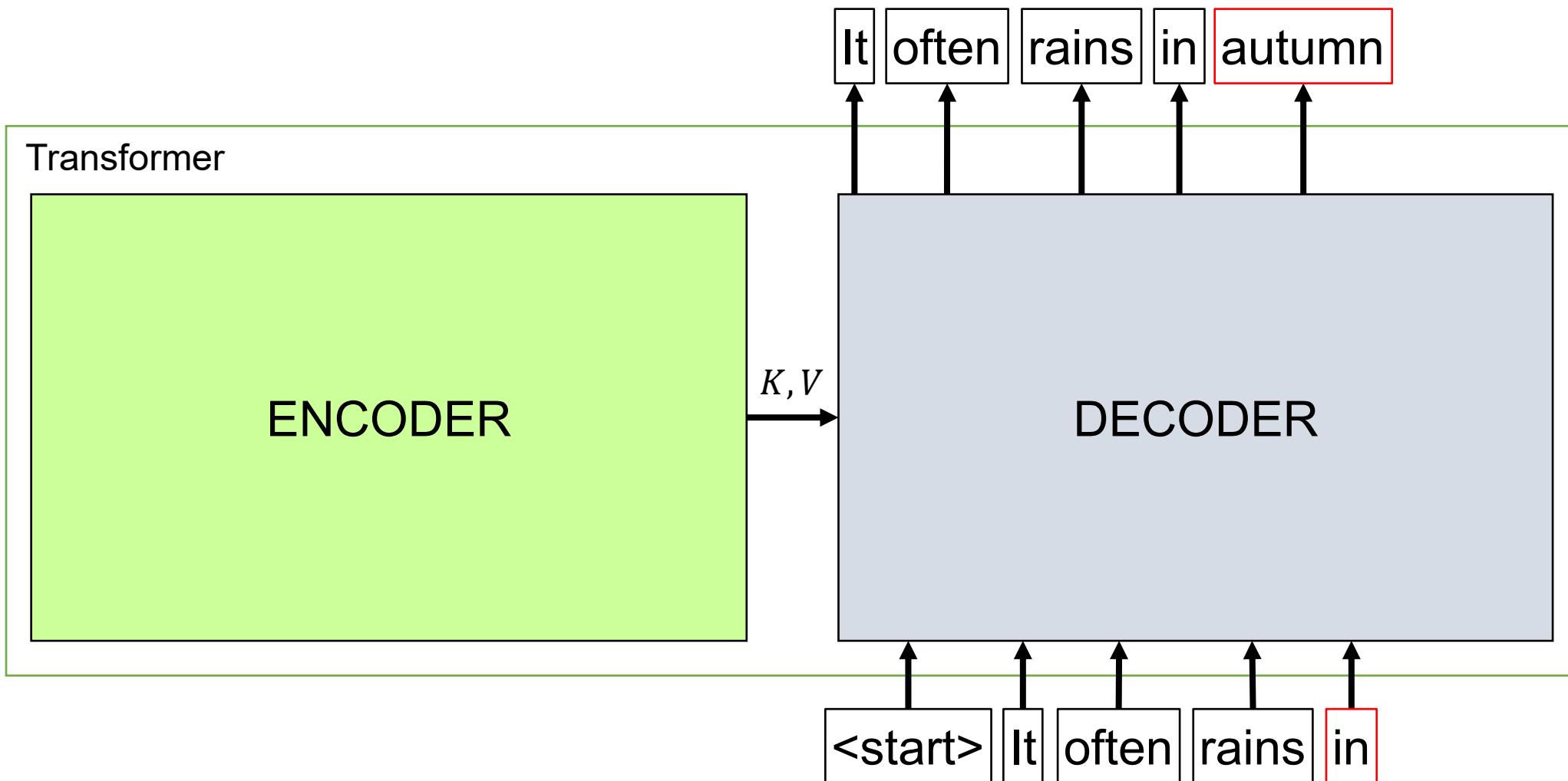
# Transformer



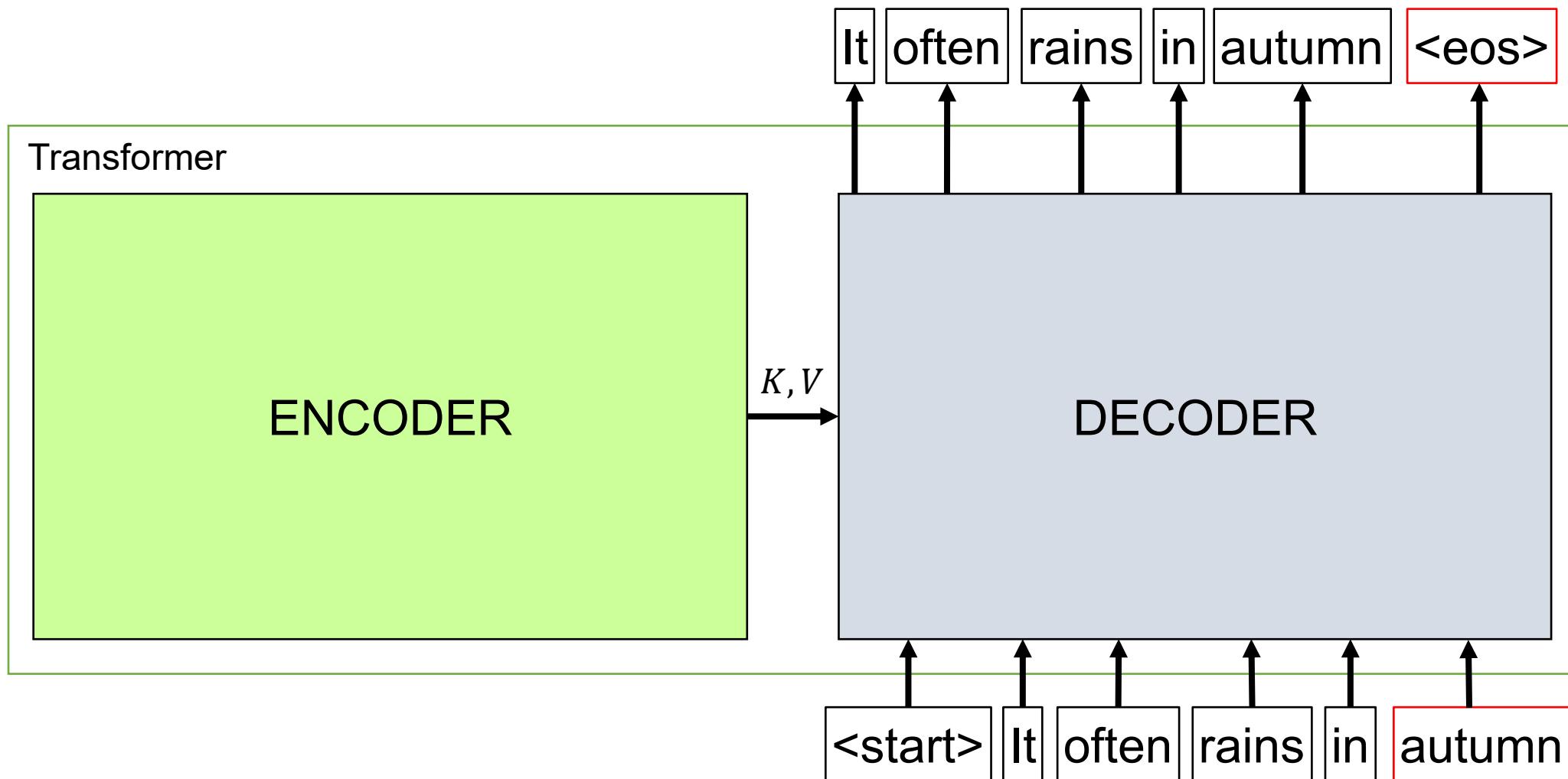
# Transformer



# Transformer



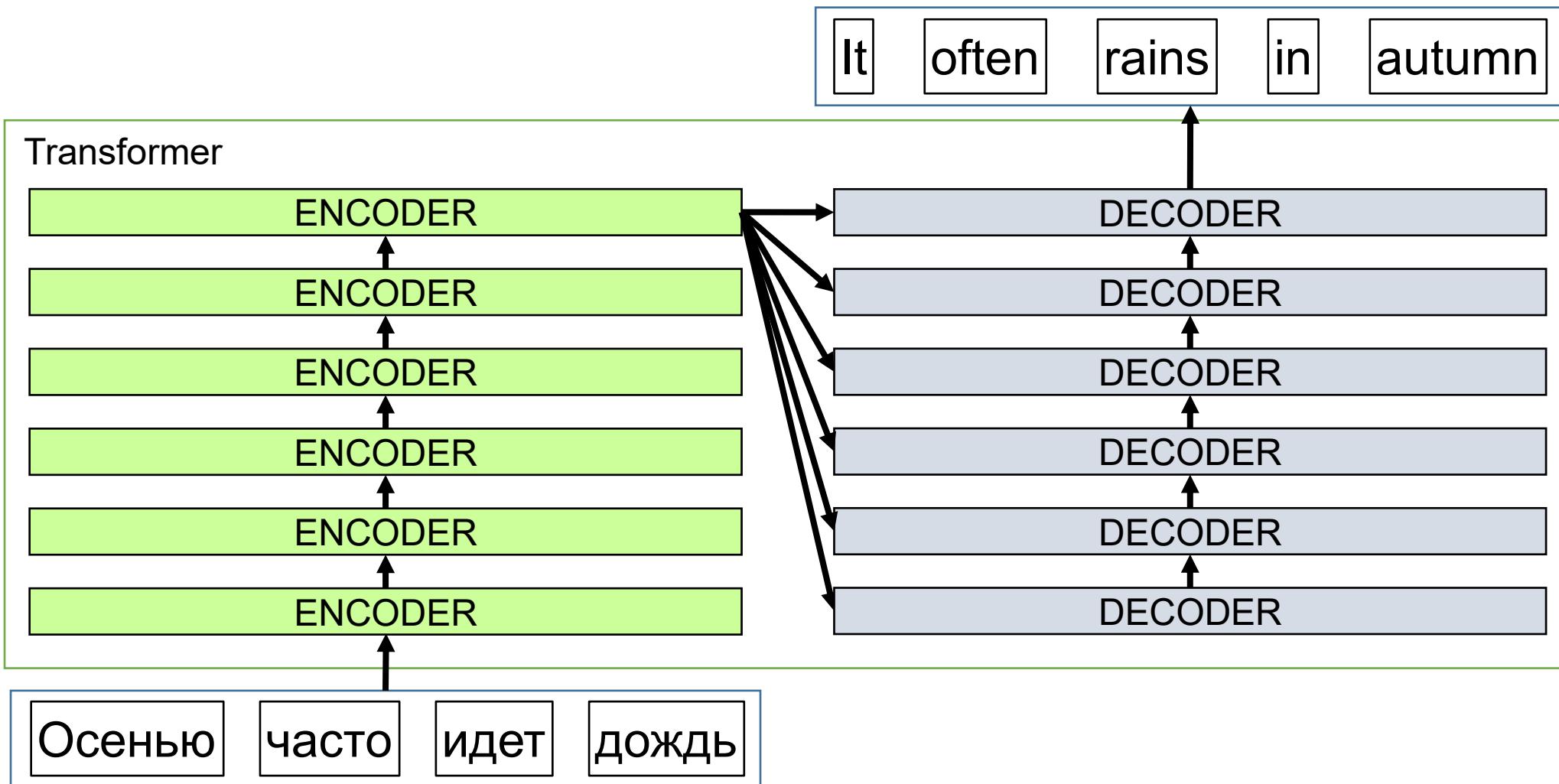
# Transformer



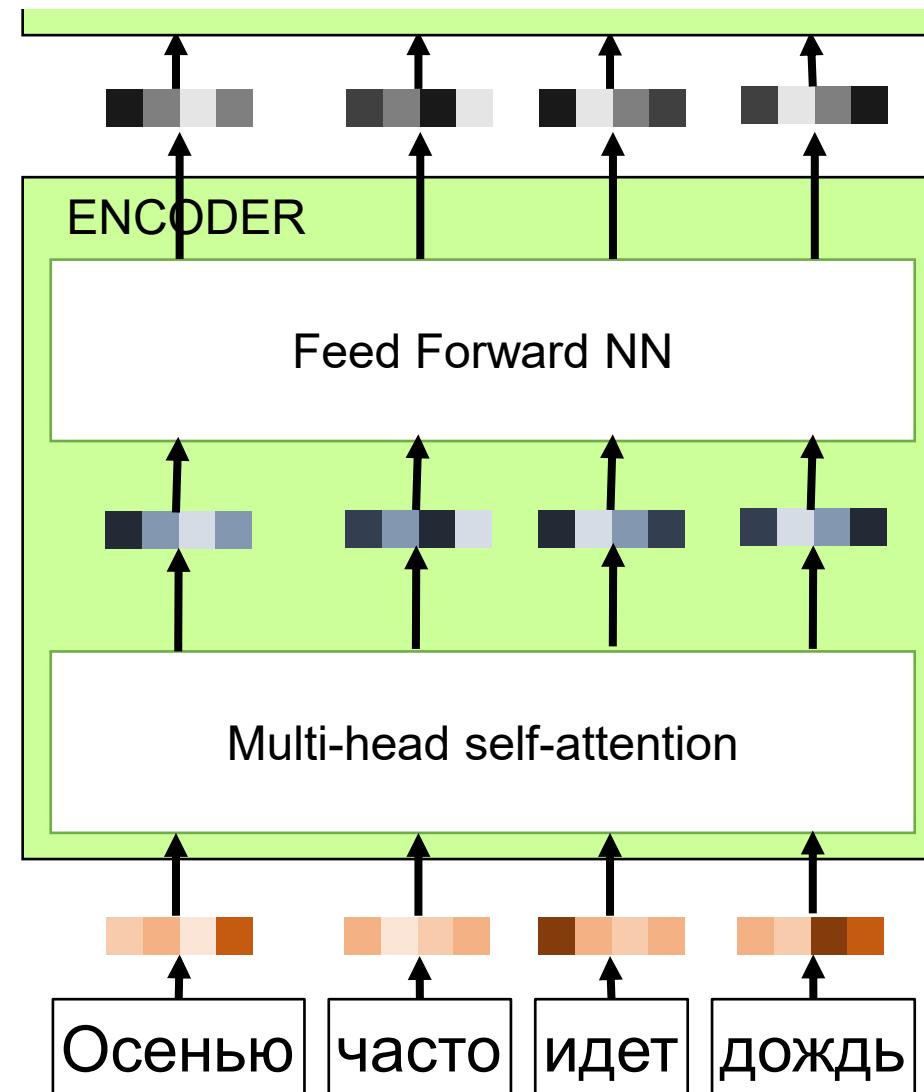
# BPE (Byte Pair Encoding)

- Метод сегментации текста с фиксированным (небольшим) размером словаря
  1. Словарь инициализируется всеми символами из train (добавляется символ конца слова ◊)
  2. В цикле n раз:
    1. Выбрать самую частую пару символов ('A', 'B')
    2. Добавить в словарь символ 'AB'

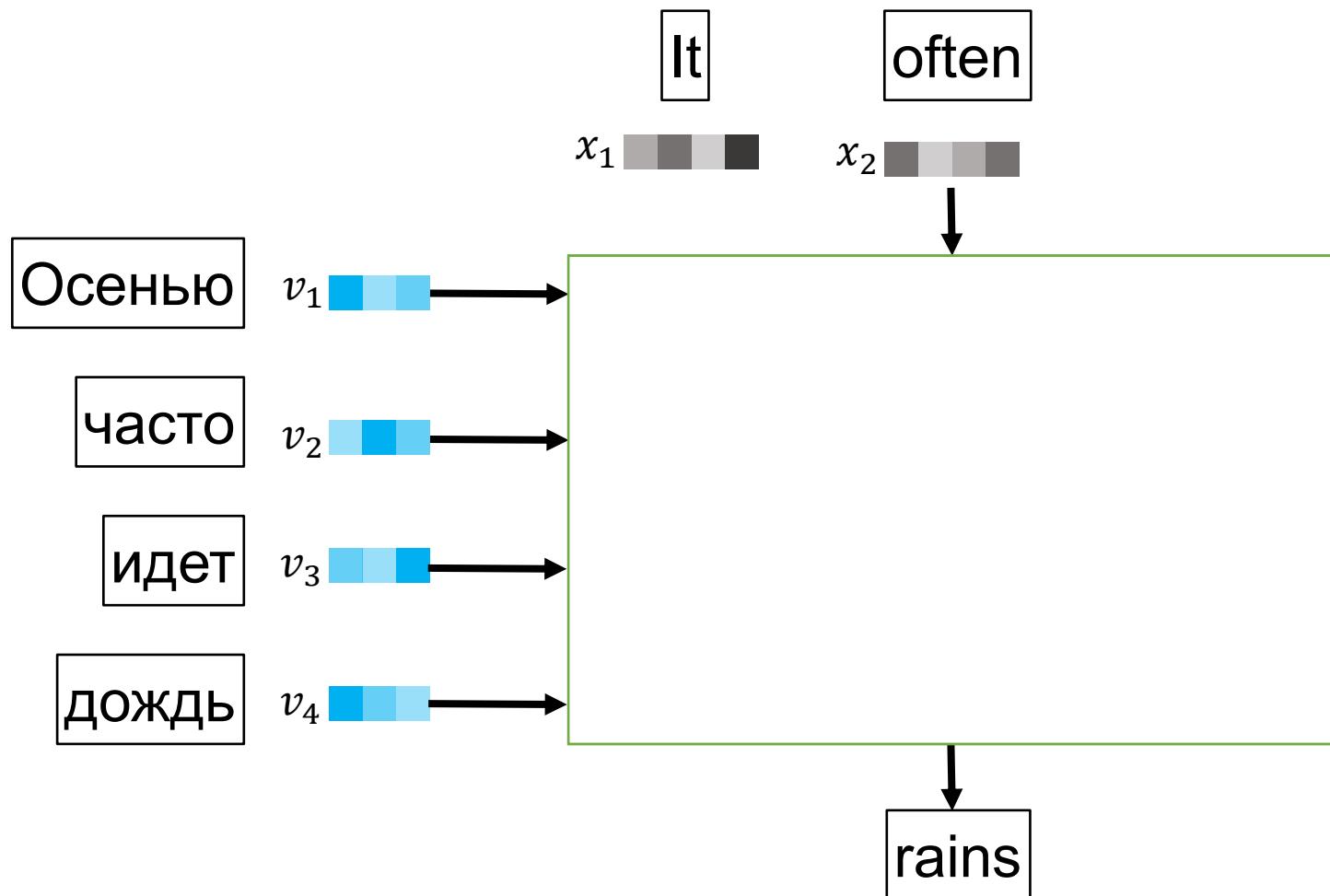
# Transformer



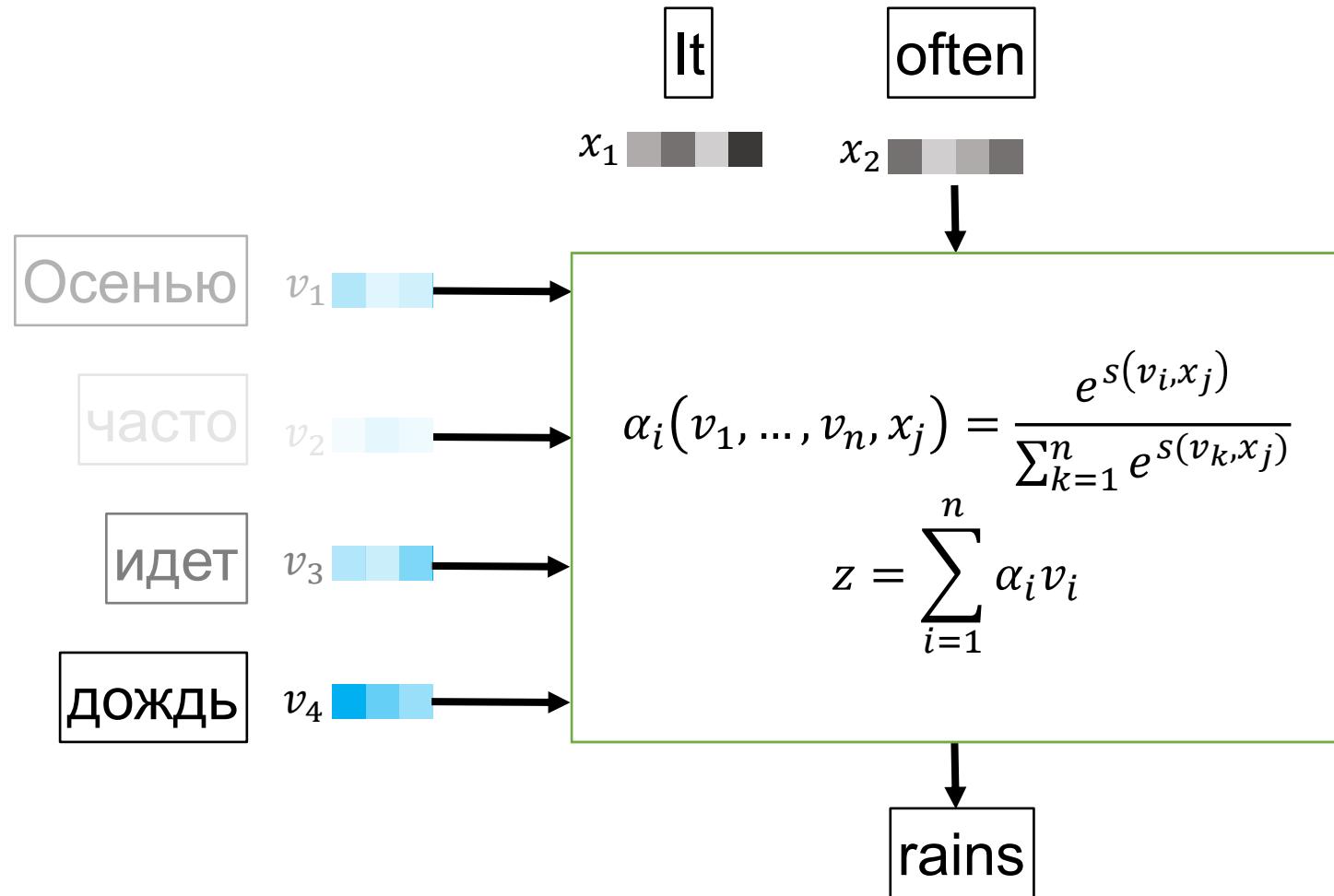
# Transformer Encoder



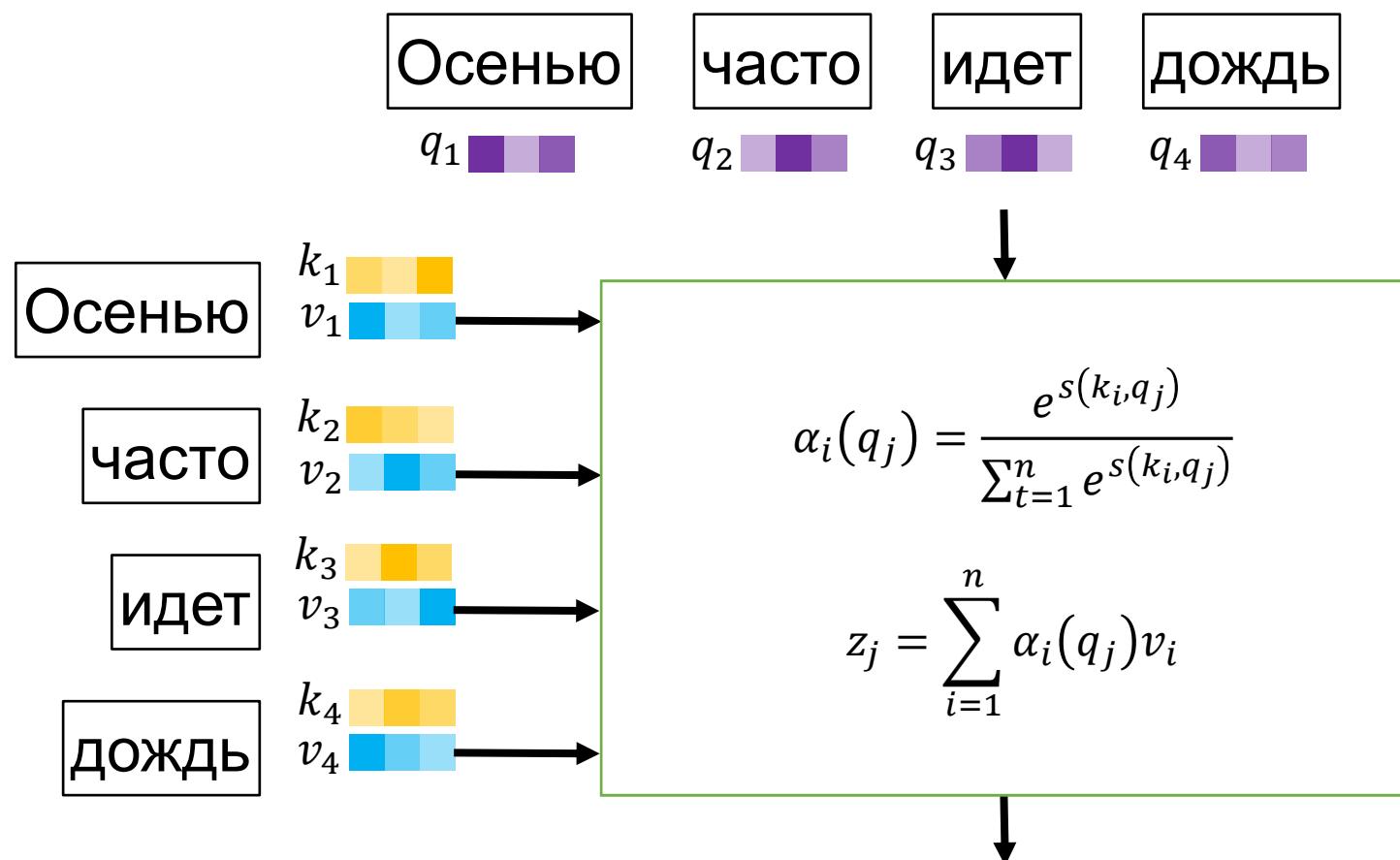
# Механизм внимания



# Механизм внимания



# Self-Attention



# Scaled Dot-Product Self-attention

Вход:

Осенью

часто

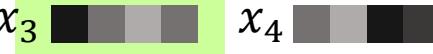
идет

дождь

Векторы:

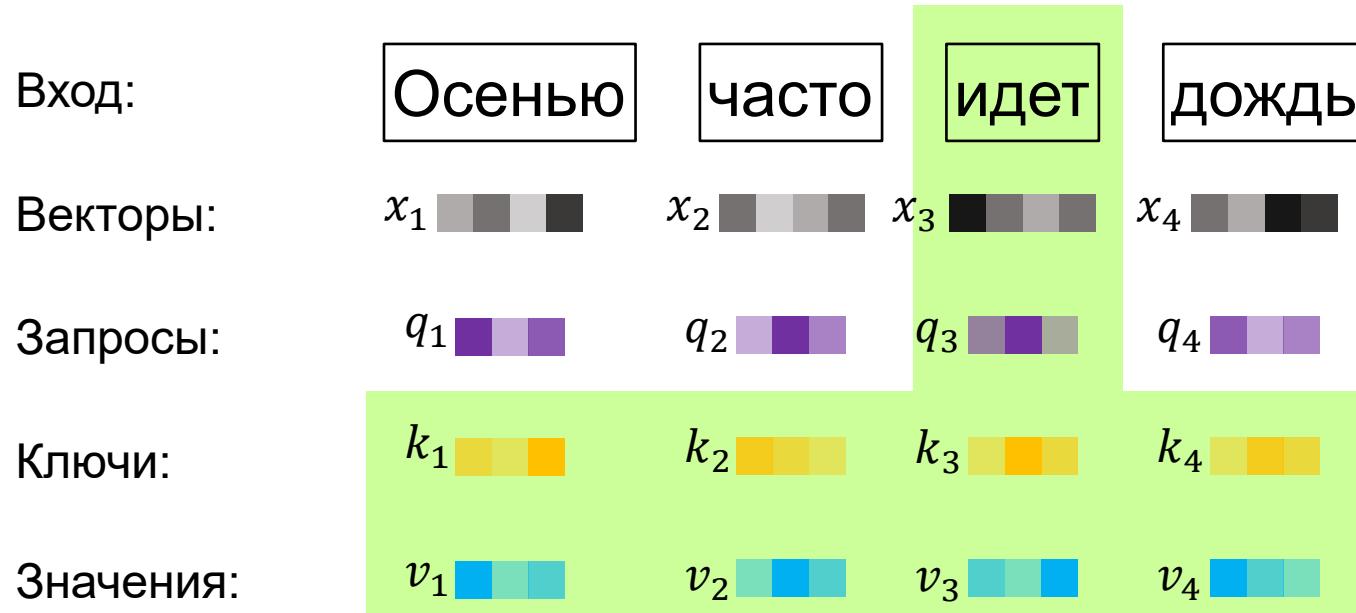
$x_1$  

$x_2$  

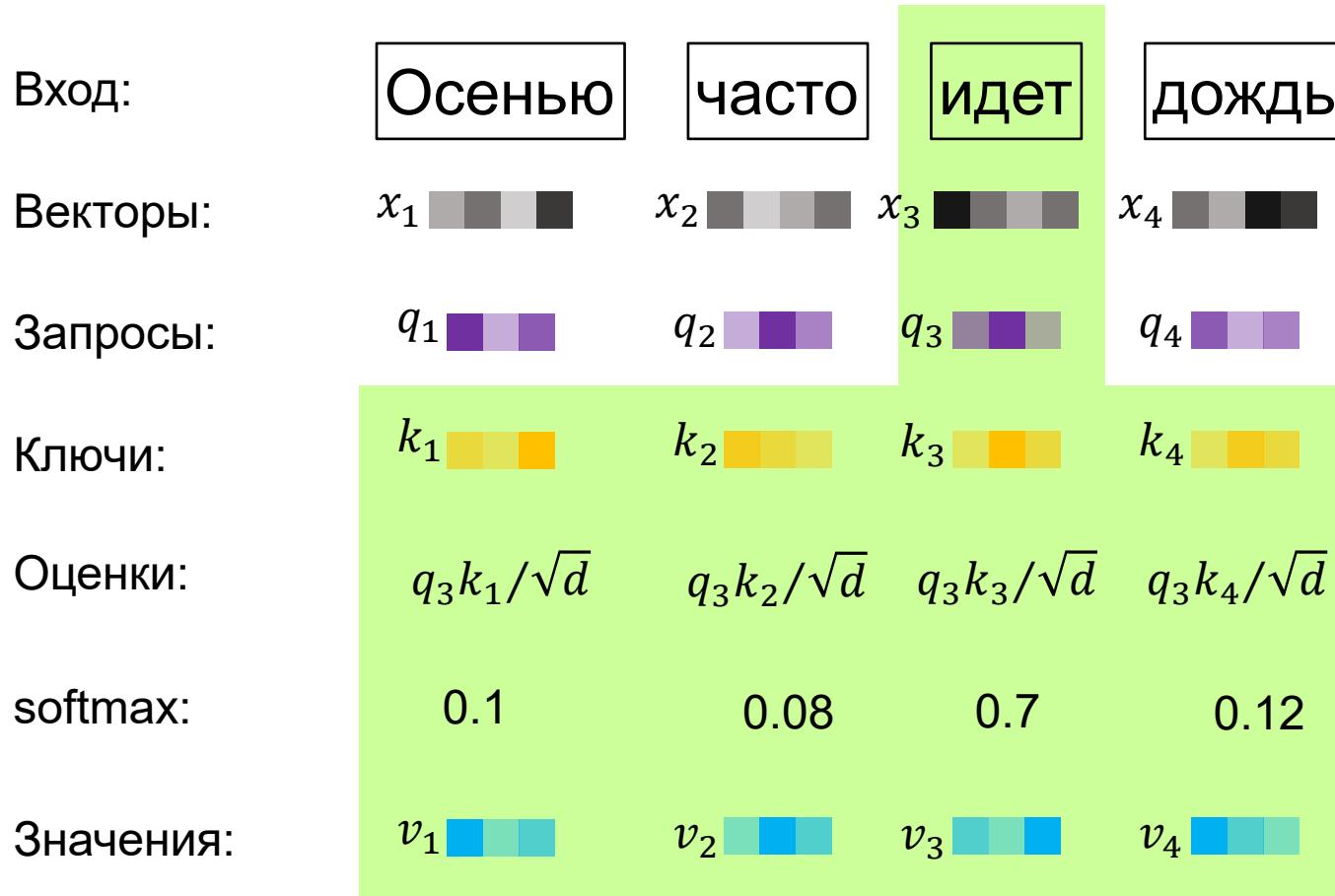
$x_3$  

$x_4$  

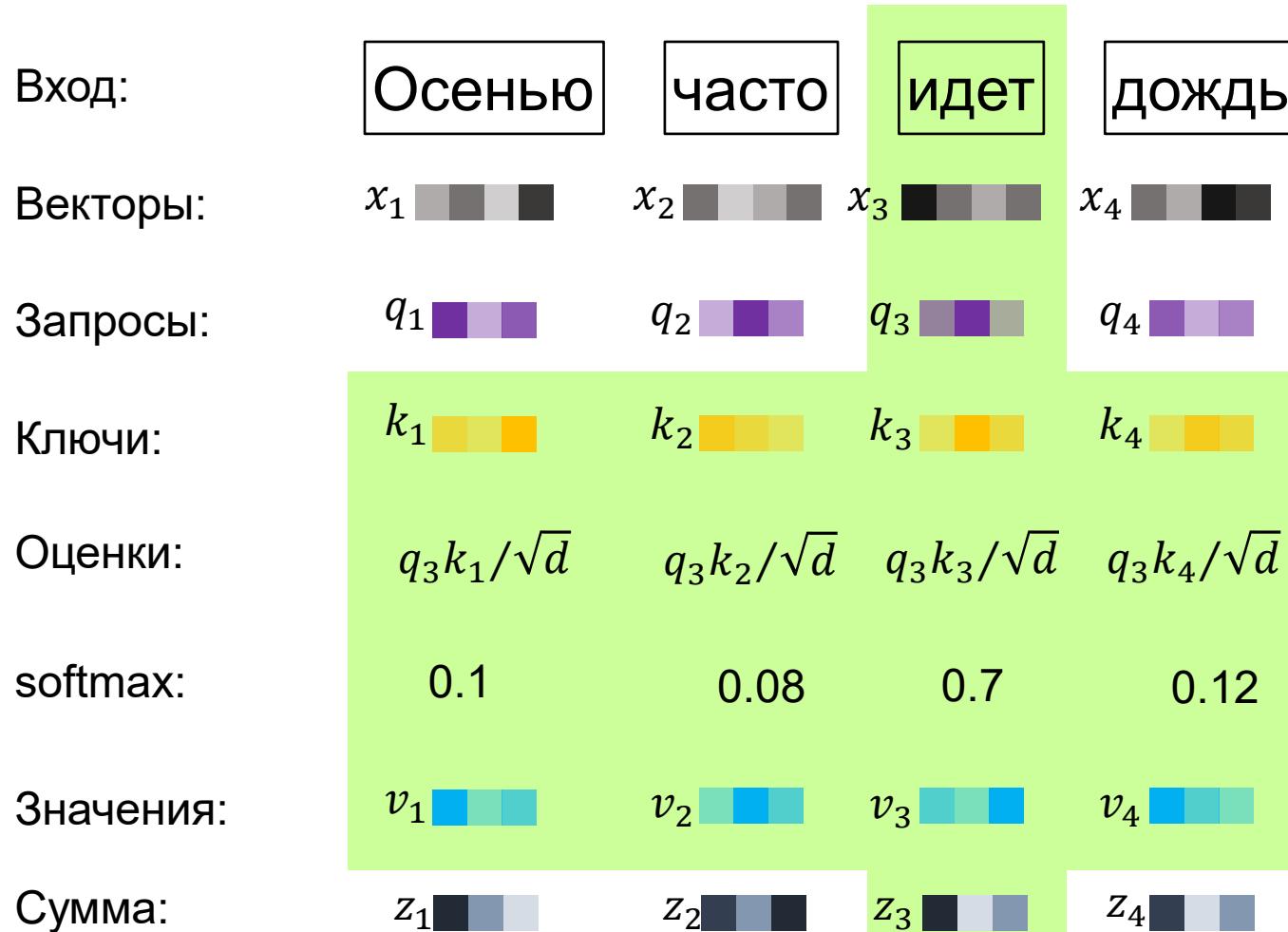
# Scaled Dot-Product Self-attention



# Scaled Dot-Product Self-attention



# Scaled Dot-Product Self-attention



# Scaled Dot-Product Self-attention

Вход:

Осенью

часто

идет

дождь

Векторы:

 $x_1$   $x_2$   $x_3$   $x_4$  

$$Q = X * W^Q$$

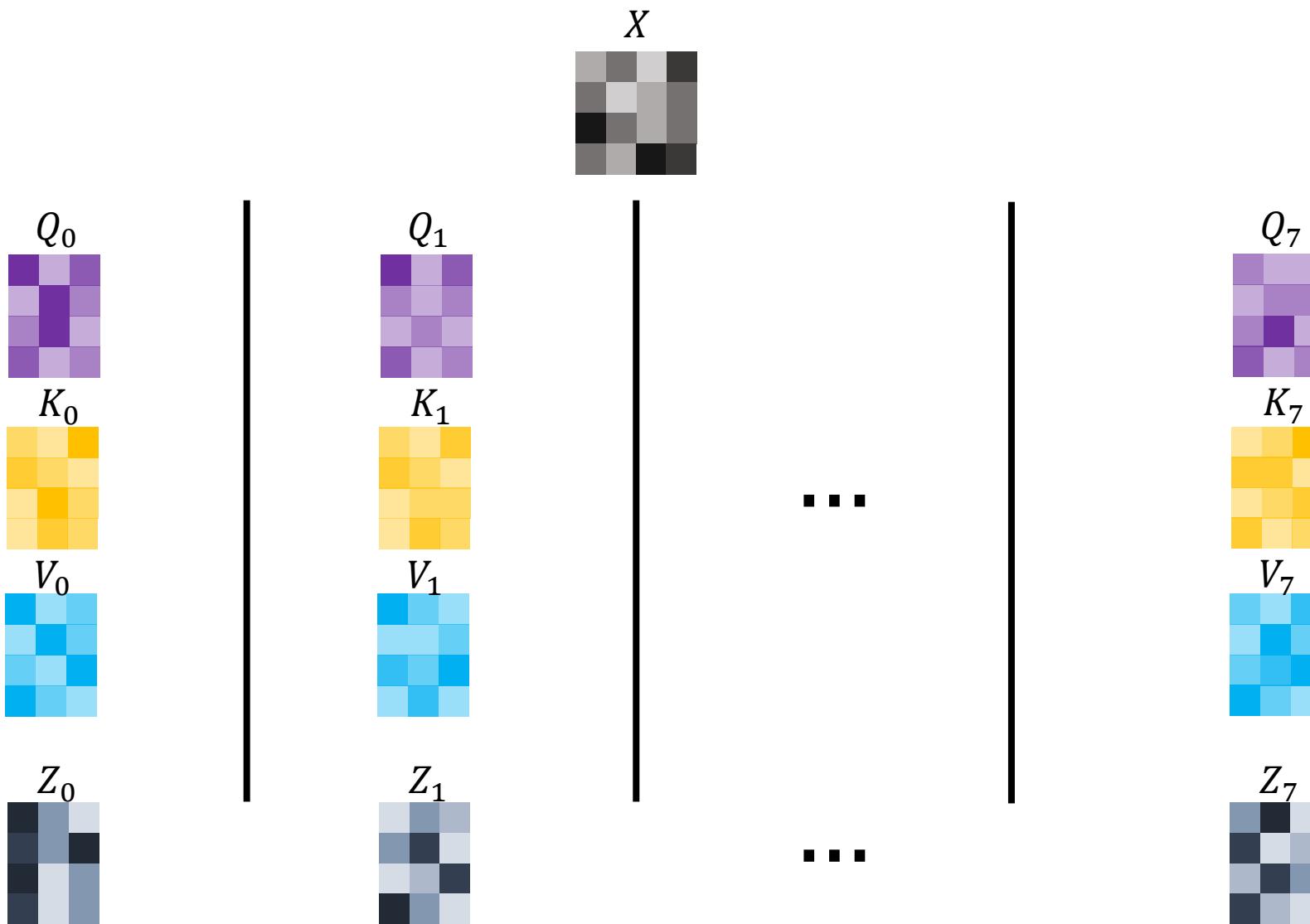
$$K = X * W^K$$

$$V = X * W^V$$

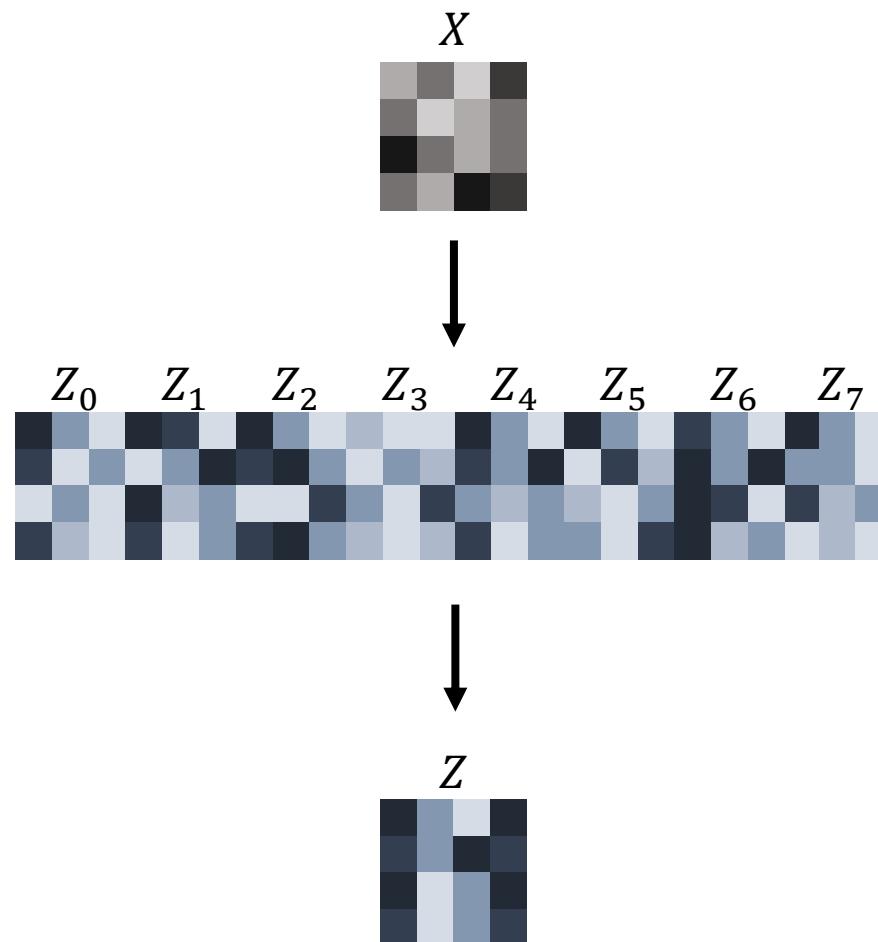
$$Z = \text{softmax} \left( \frac{QK^T}{\sqrt{d_K}} \right) V$$

 $z_1$   $z_2$   $z_3$   $z_4$  

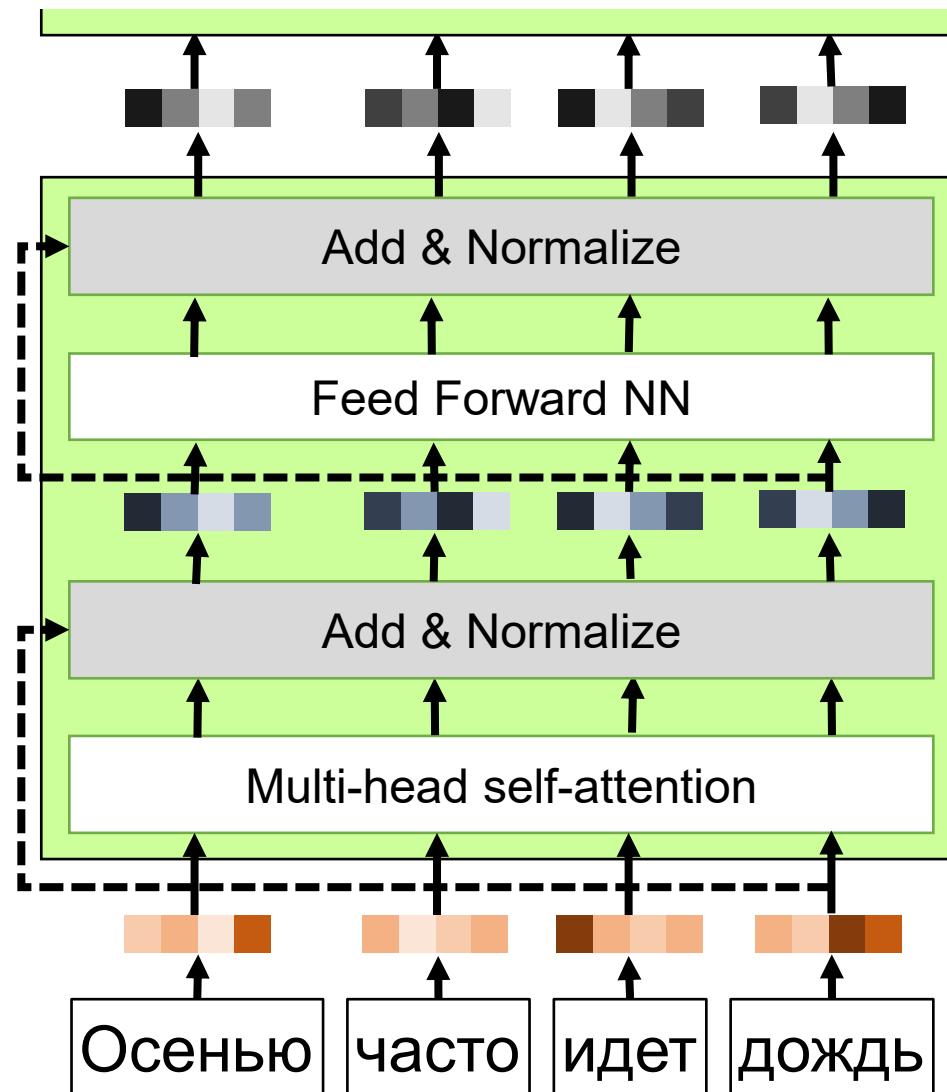
# Multi-head attention



# Multi-head attention



# Transformer Encoder



# Позиционное кодирование

Вход:

Осенью

часто

идет

дождь

Векторы:

 $x_1$  $x_2$  $x_3$  $x_4$ 

+

+

+

+

Кодировка  
позиции: $PE_1$  $PE_2$  $PE_3$  $PE_4$ 

=

=

=

=

Векторы  
с позицией:

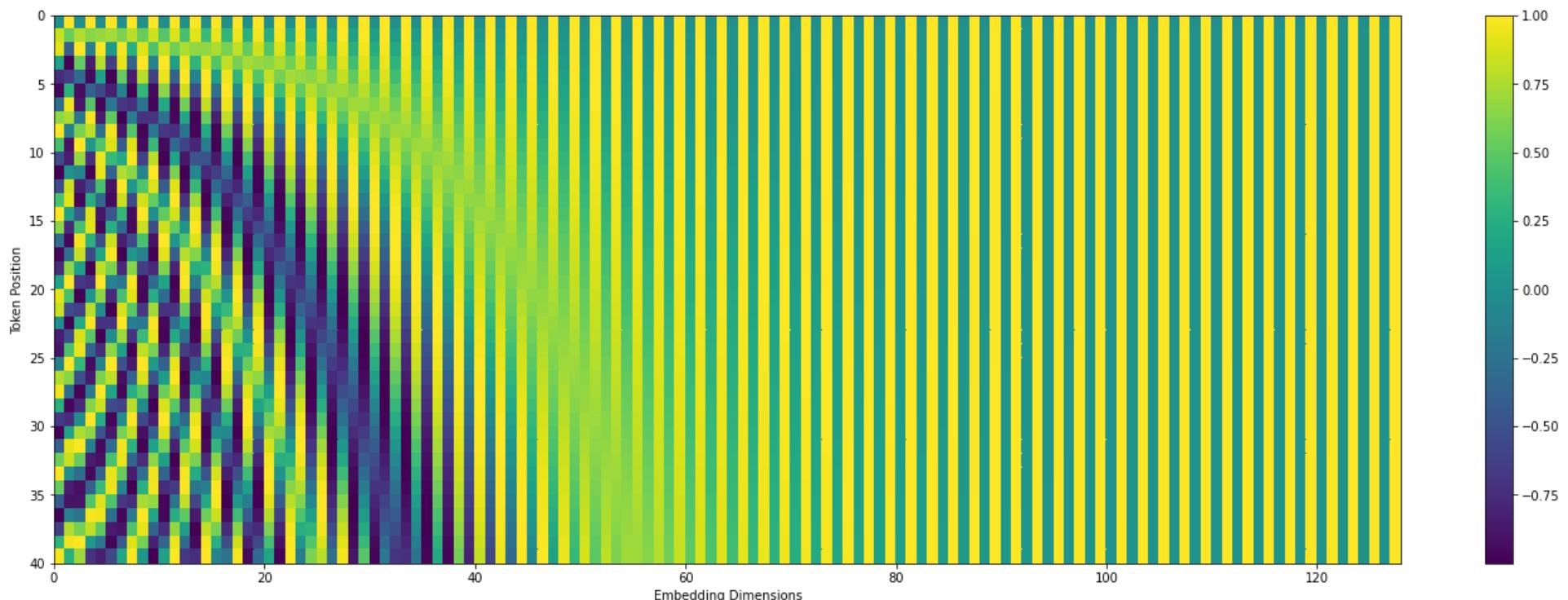
$$PE_{pos,2i} = \sin(\frac{pos}{10000^{2i/d_{model}}})$$

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

# Позиционное кодирование

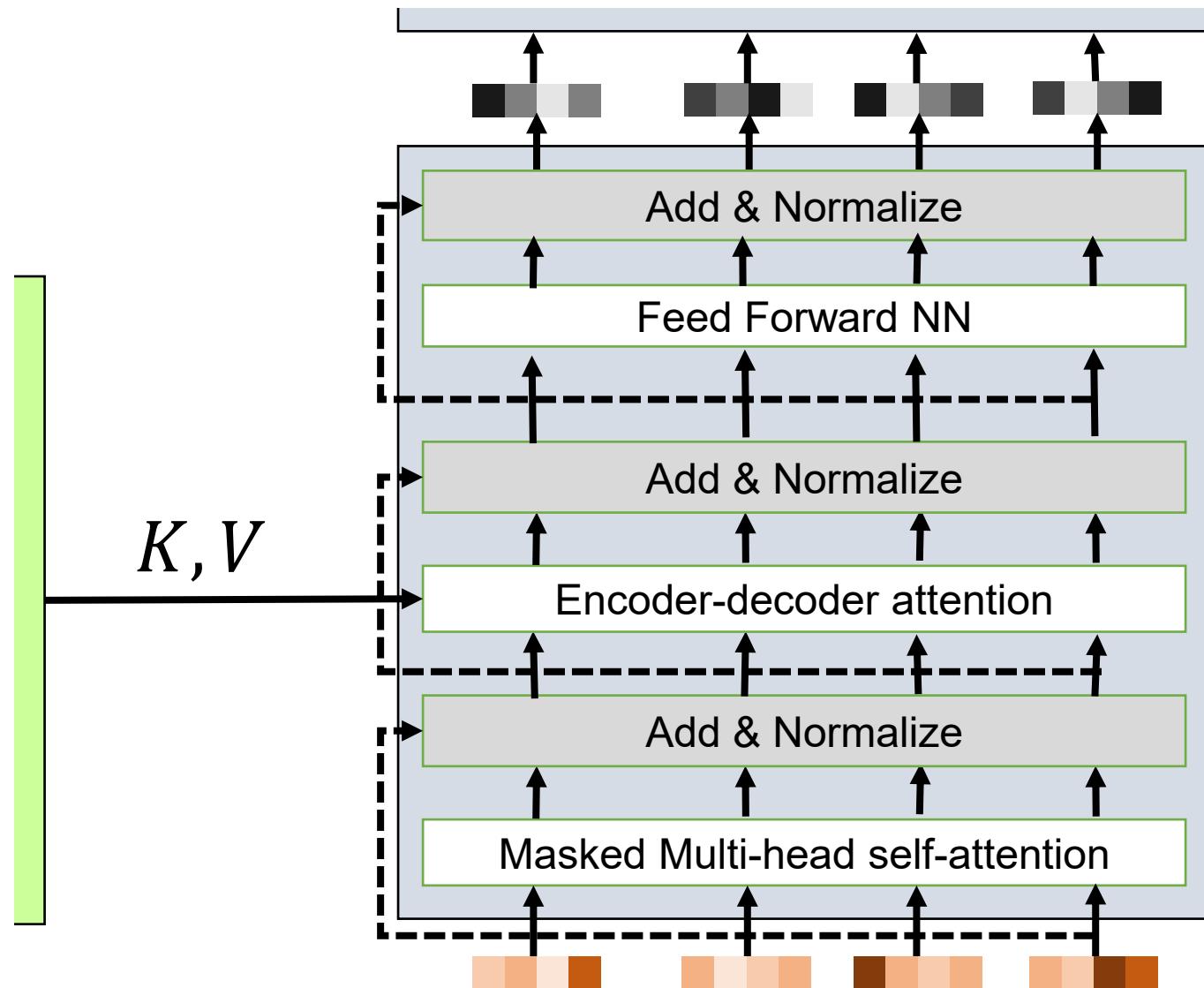
$$PE_{pos,2i} = \sin(\frac{pos}{10000^{2i/d_{model}}})$$

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

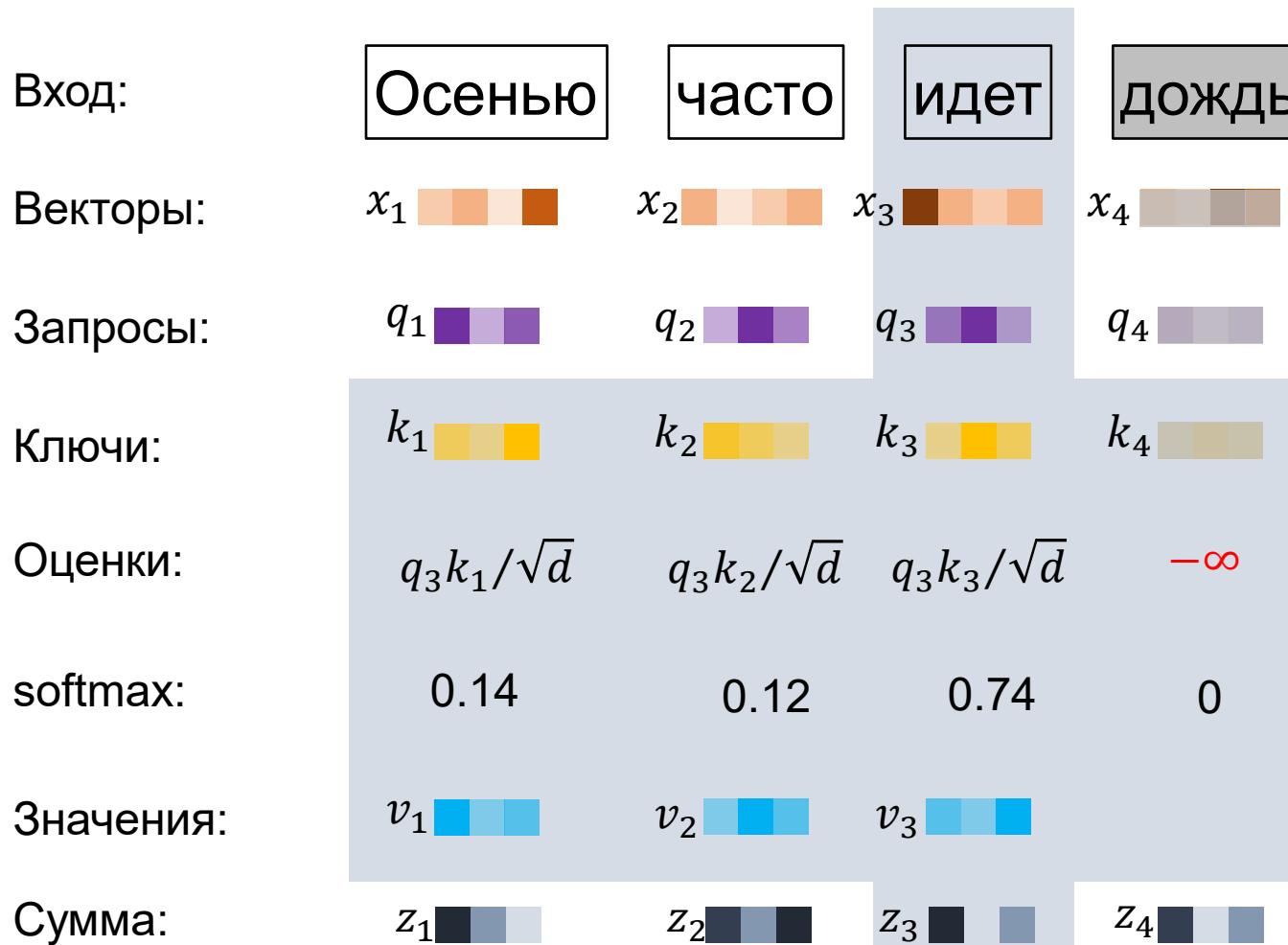


Код для генерации картинки [здесь](#)

# Transformer Decoder



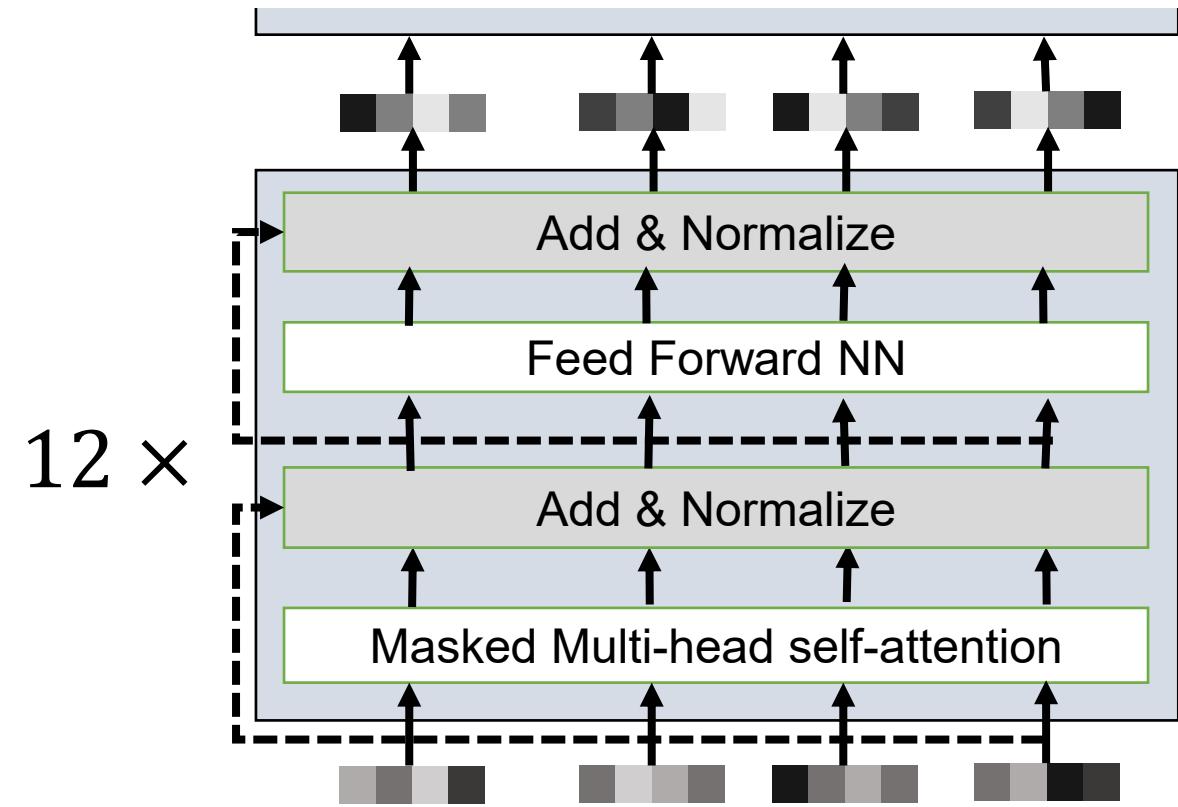
# Masked Scaled Dot-Product Self-attention



# GPT (Generative Pre-Training)

Языковая модель, основанная на декодере трансформера

- Encoder-decoder attention выкидывается



# GPT (Generative Pre-Training)

- Предобучение на корпусе  $\mathcal{U}$  (без учителя)

$$\begin{aligned} h_0 &= U \mathbf{W}_e + W_p; \\ h_i &= \text{transformer}(h_{i-1}), i = \overline{1, n} \\ P(u) &= \text{softmax}(h_n \mathbf{W}_e^T) \end{aligned}$$

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

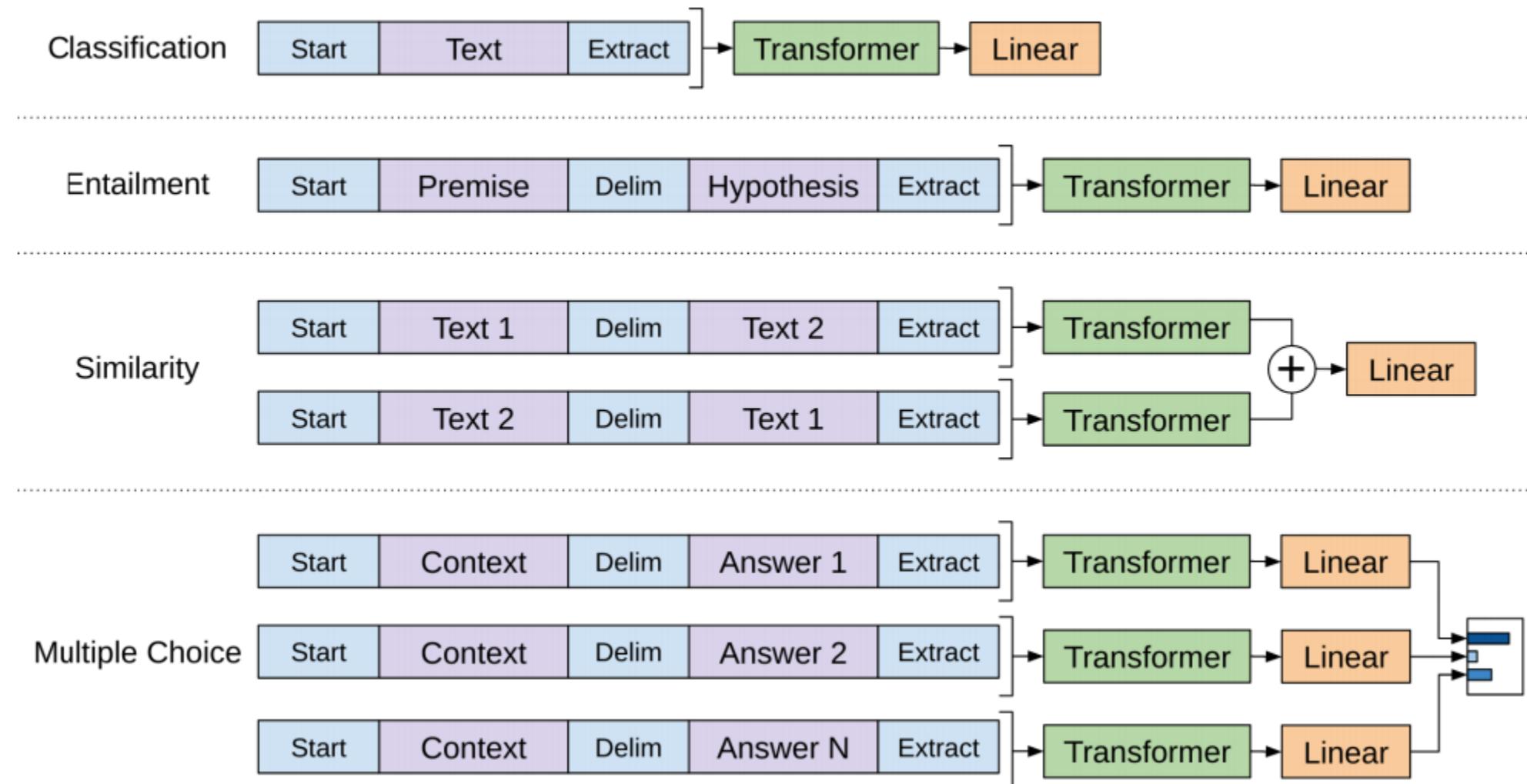
- Донастройка под целевую задачу на корпусе  $\mathcal{C}$  (с учителем)

$$P(y|x^1, \dots, x^m) = \text{softmax}(h_l^m W_y)$$

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \dots, x^m)$$

$$L(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C});$$

# GPT (Generative Pre-Training)



\*Картина из [оригинальной статьи](#)

## GPT-2

- Большинство задач решается оценкой распределения:

$$p(\text{output}|\text{input})$$

- Хочется построить модель, которая будет решать множество задач. То есть оценивать:

$$p(\text{output}|\text{input}, \text{task})$$

- Задачу можно формулировать словами и подавать языковой модели вместе с *input*

## GPT-2

Очень похожа на GPT, но :

- Больше параметров (1.5 миллиарда параметров [48 слоев трансформера, размерность векторов 1600] vs 117 миллионов в GPT)
- Больше корпус для обучения (40 GB текста)
  - Внешние ссылки reddit (> 2 голосов)
- Другая токенизация: ВРЕ по байтам, а не по символам
- Небольшие изменения архитектуры сети:
  - Передвинут Layer normalization
  - Изменена инициализация

## GPT-2

Хорошо подходит для задач, похожих на языковое моделирование:

- Понимание текста (ответы на вопросы по тексту)
- Автоматическое рефериование
- Машинный перевод
- Вопрос-ответ

## GPT-3

Практически ничем не отличается от GPT-2, но

- Еще больше параметров (175 миллиардов [96 слоев трансформера, размерность векторов – 12288] vs 1.5 миллиарда в GPT-2)
- Еще больше корпус для обучения (570 GB текста)
- Еще больше контекст (2048 токенов)

# GPT-3

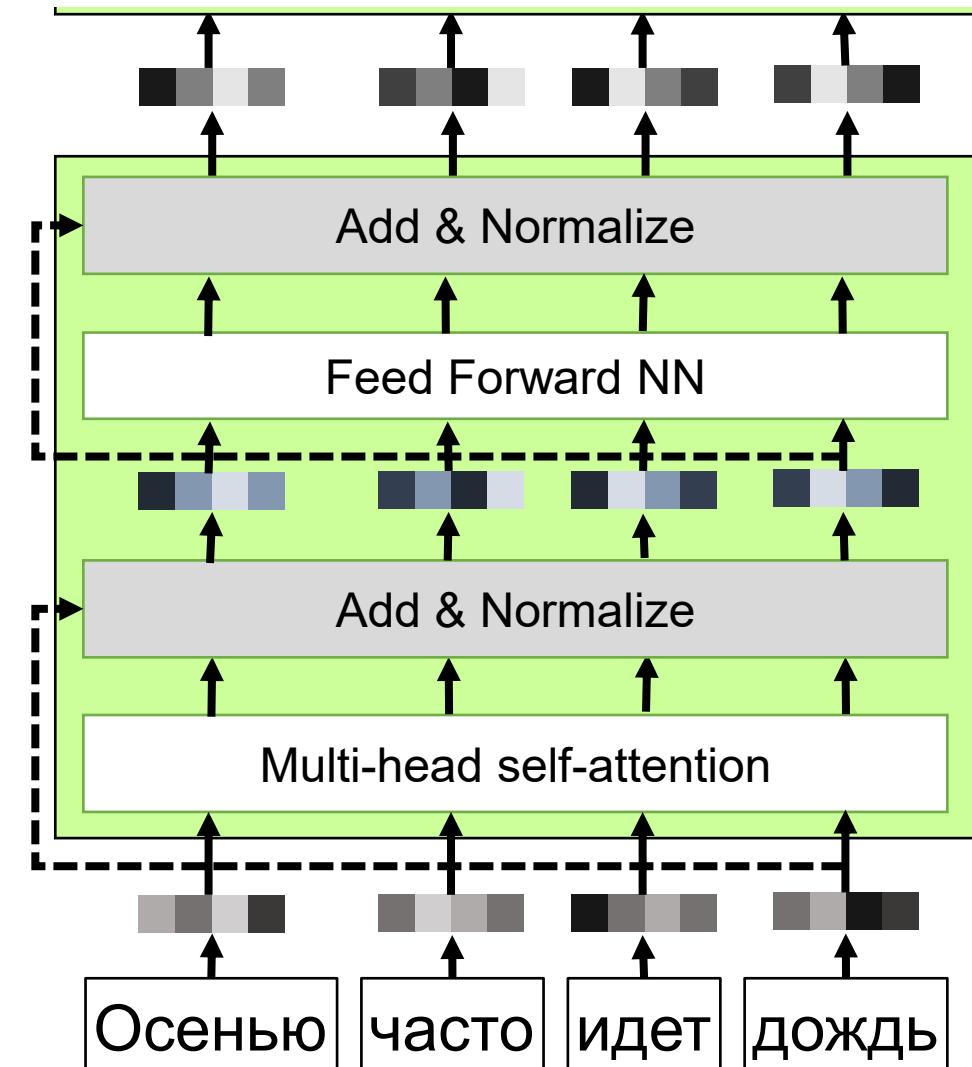
Можно применять в 3 вариантах:

- Zero-shot:
  - формулировка задачи
  - вопрос
- One-shot:
  - формулировка задачи
  - пример в supervised стиле
  - вопрос
- Few-shot:
  - формулировка задачи
  - несколько примеров в supervised стиле
  - вопрос

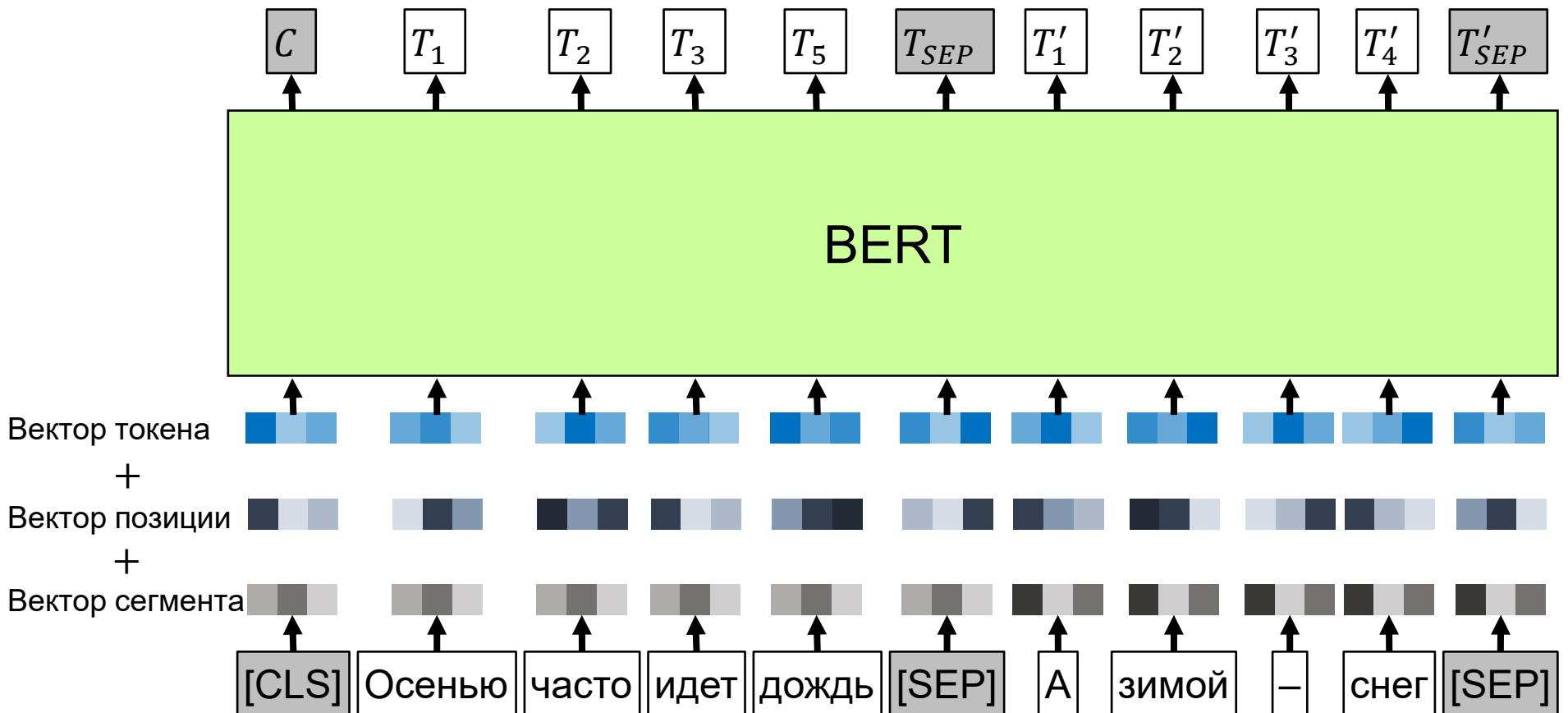
## BERT

Языковая модель, основанная на  
энкодере трансформера

12(24) ×

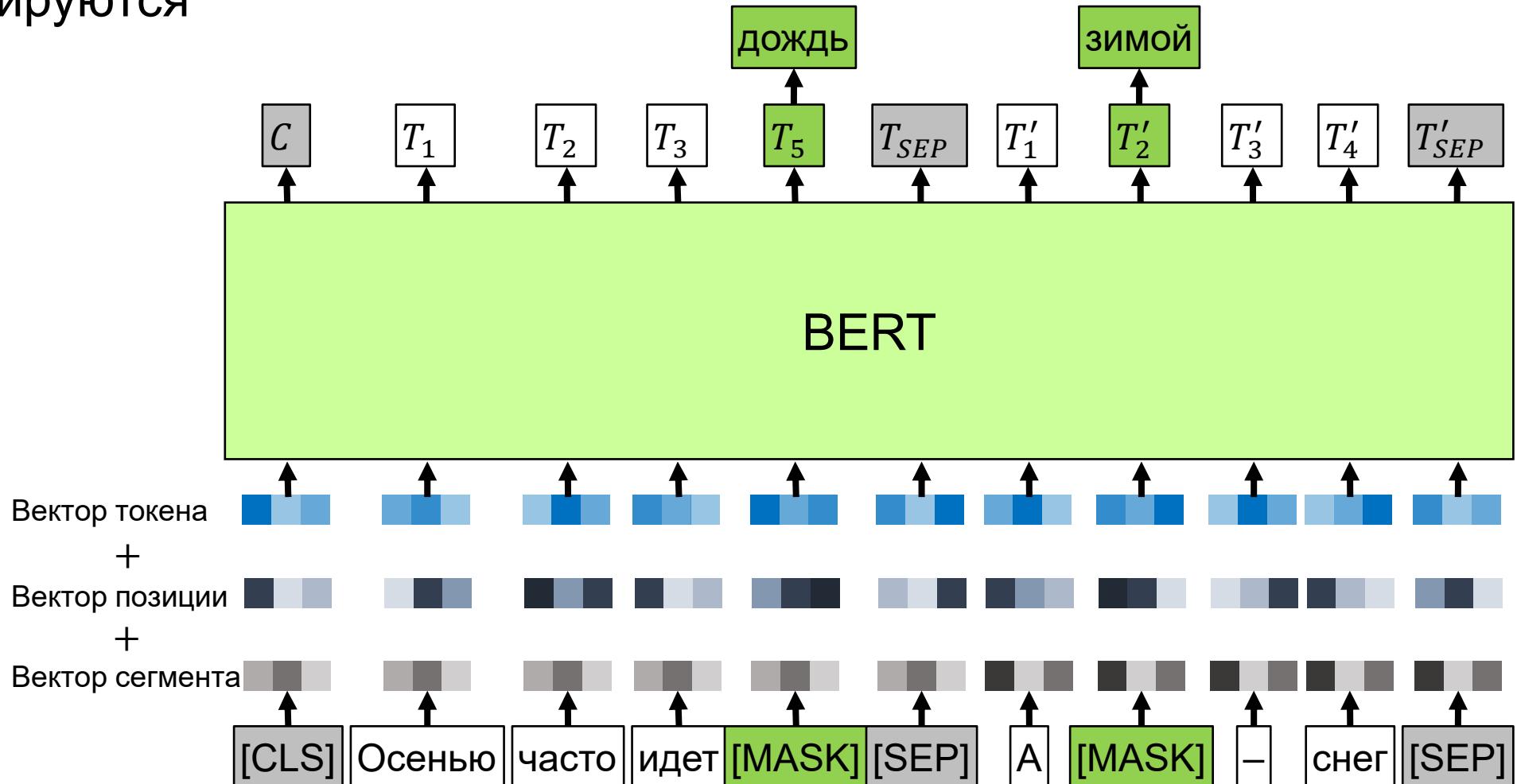


## BERT



# BERT pretraining (masked LM)

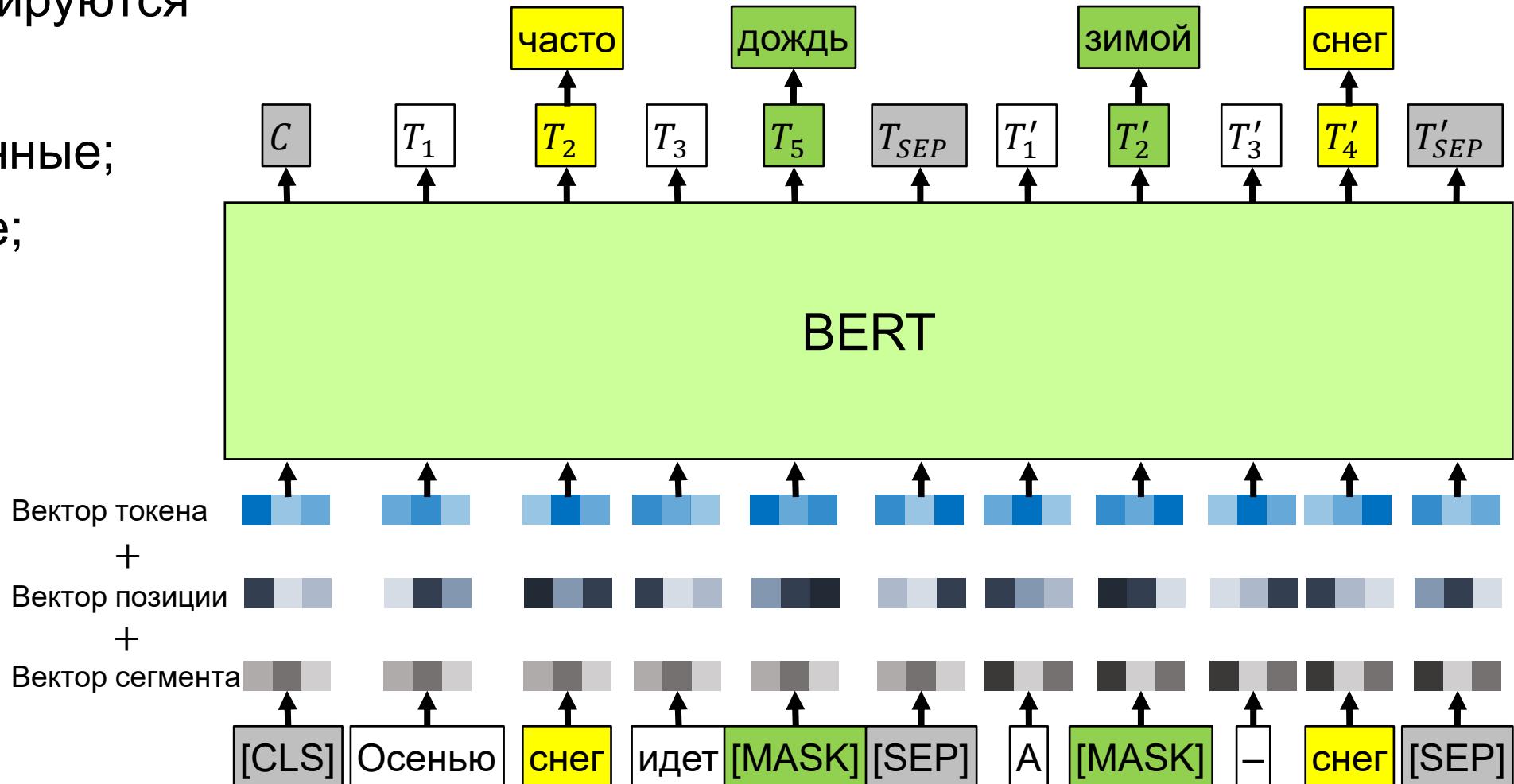
15% токенов маскируются



# BERT pretraining (masked LM)

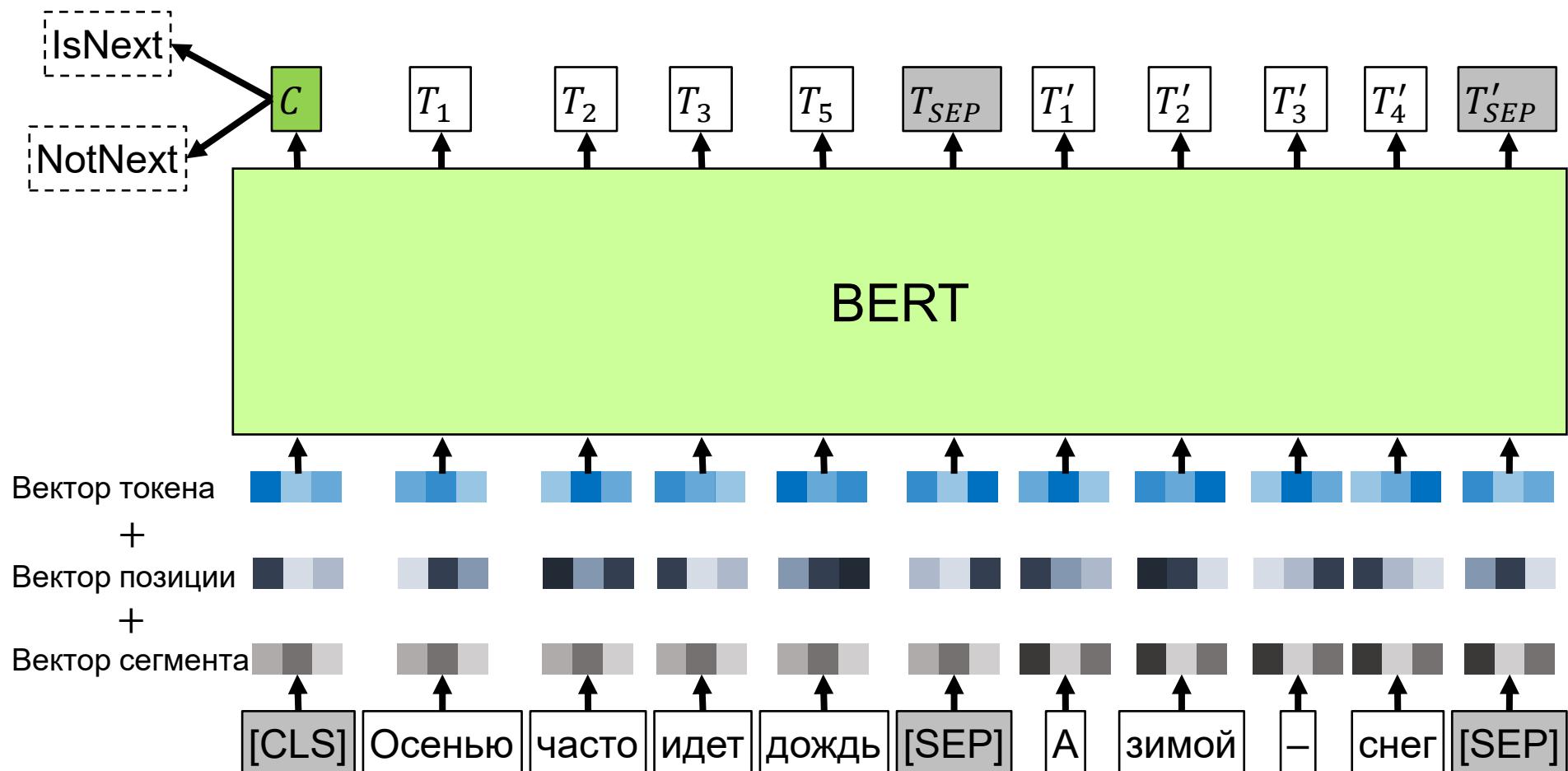
15% токенов маскируются

- 80% - [MASK];
- 10% - неизмененные;
- 10% - случайные;

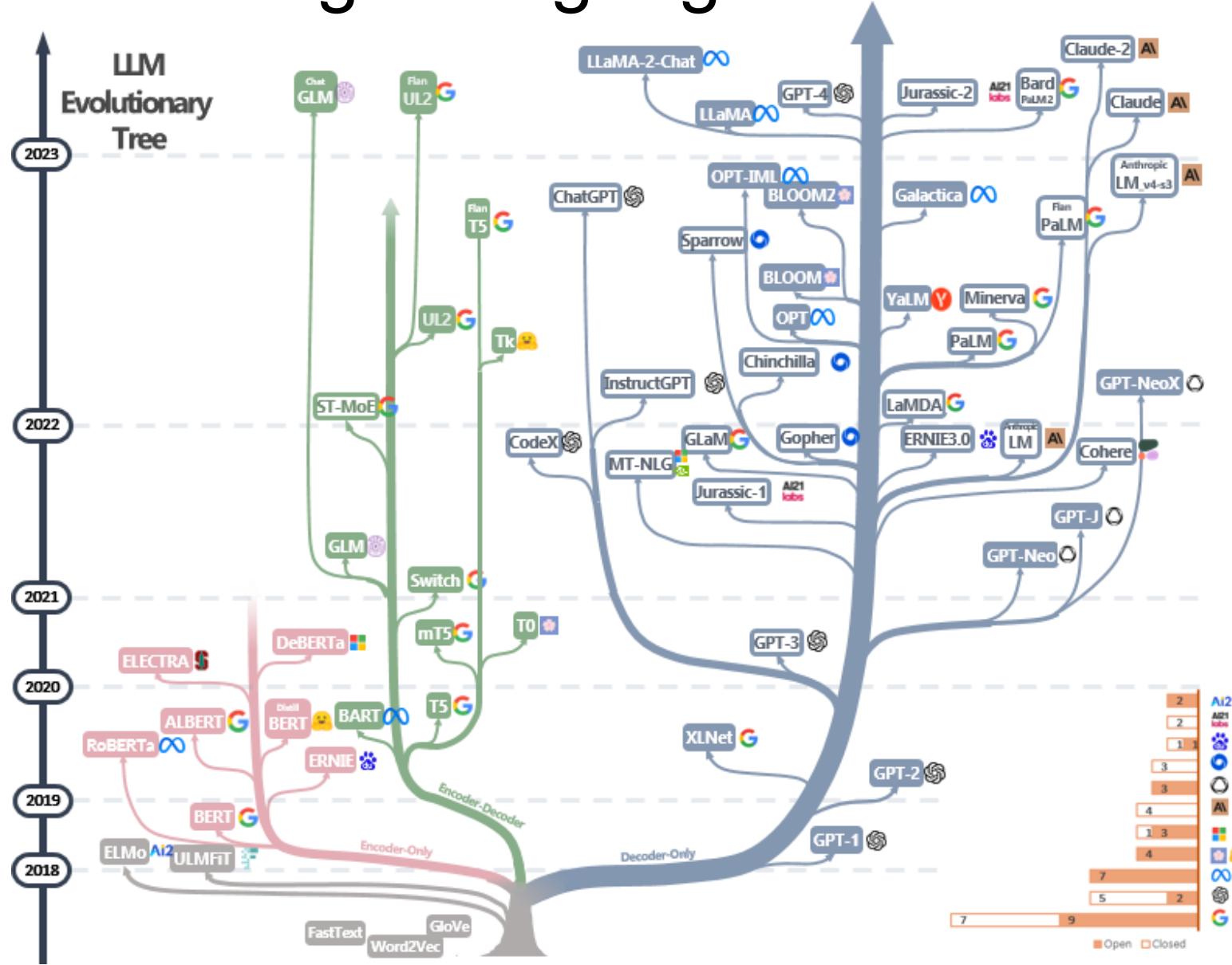


# BERT pretraining (NSP)

Предсказывается, является ли фрагмент  $B$  следующим за фрагментом  $A$

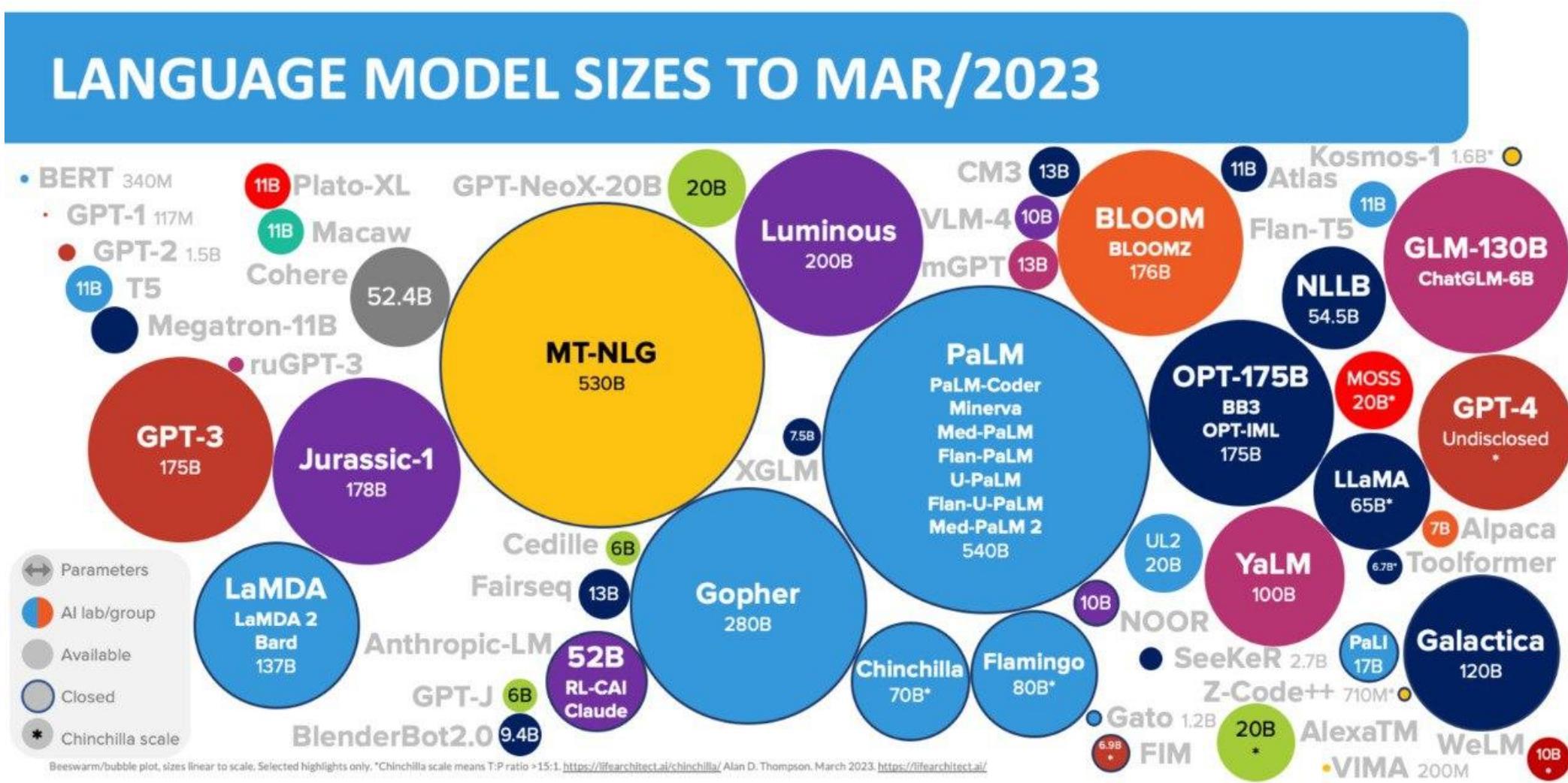


# Large Language Models



Источник картинки

# Large Language Models



# Следующая лекция

Информационный поиск