

# 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:

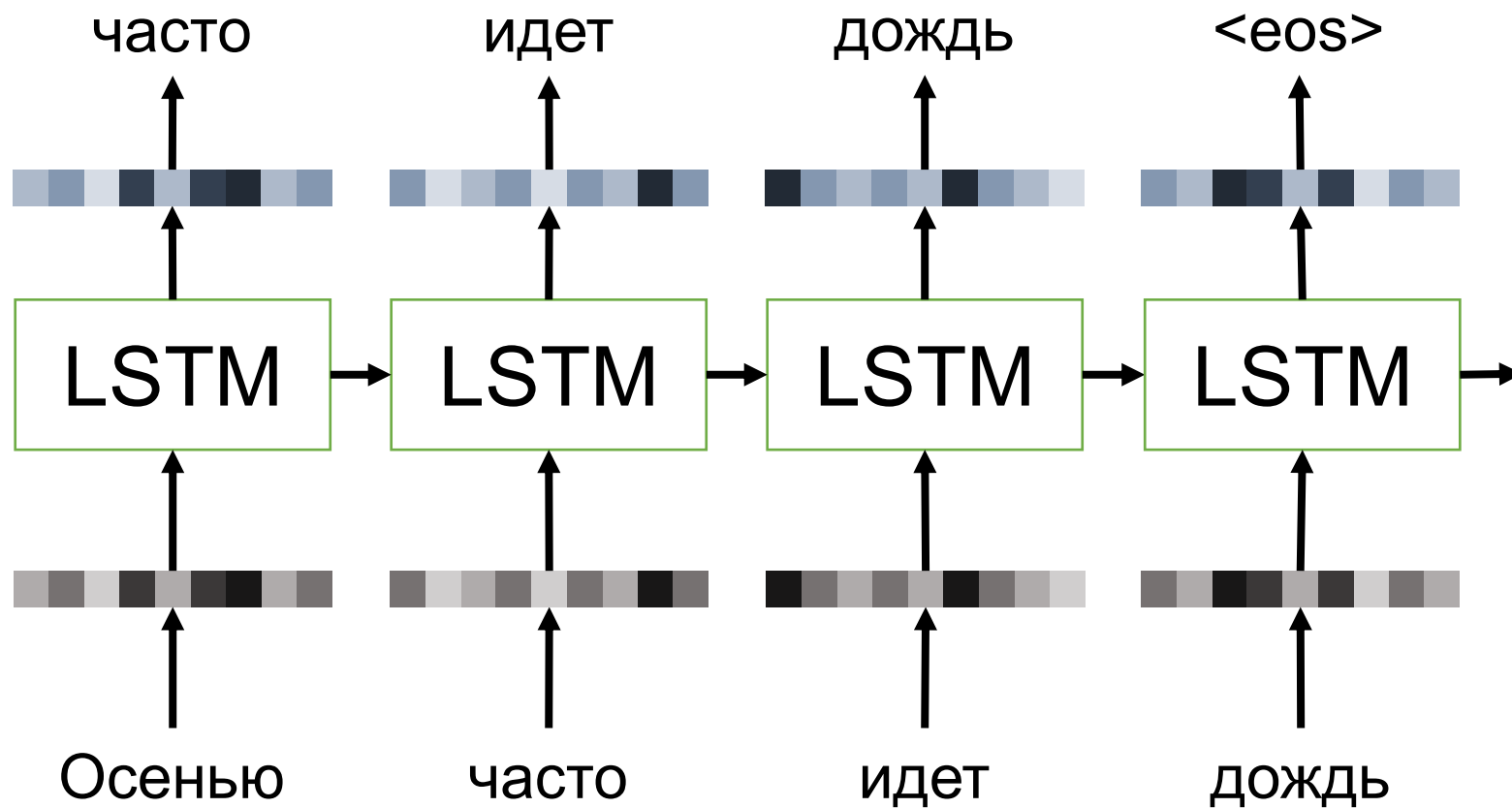
- $P(w_1, w_2, \dots, w_n)$

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

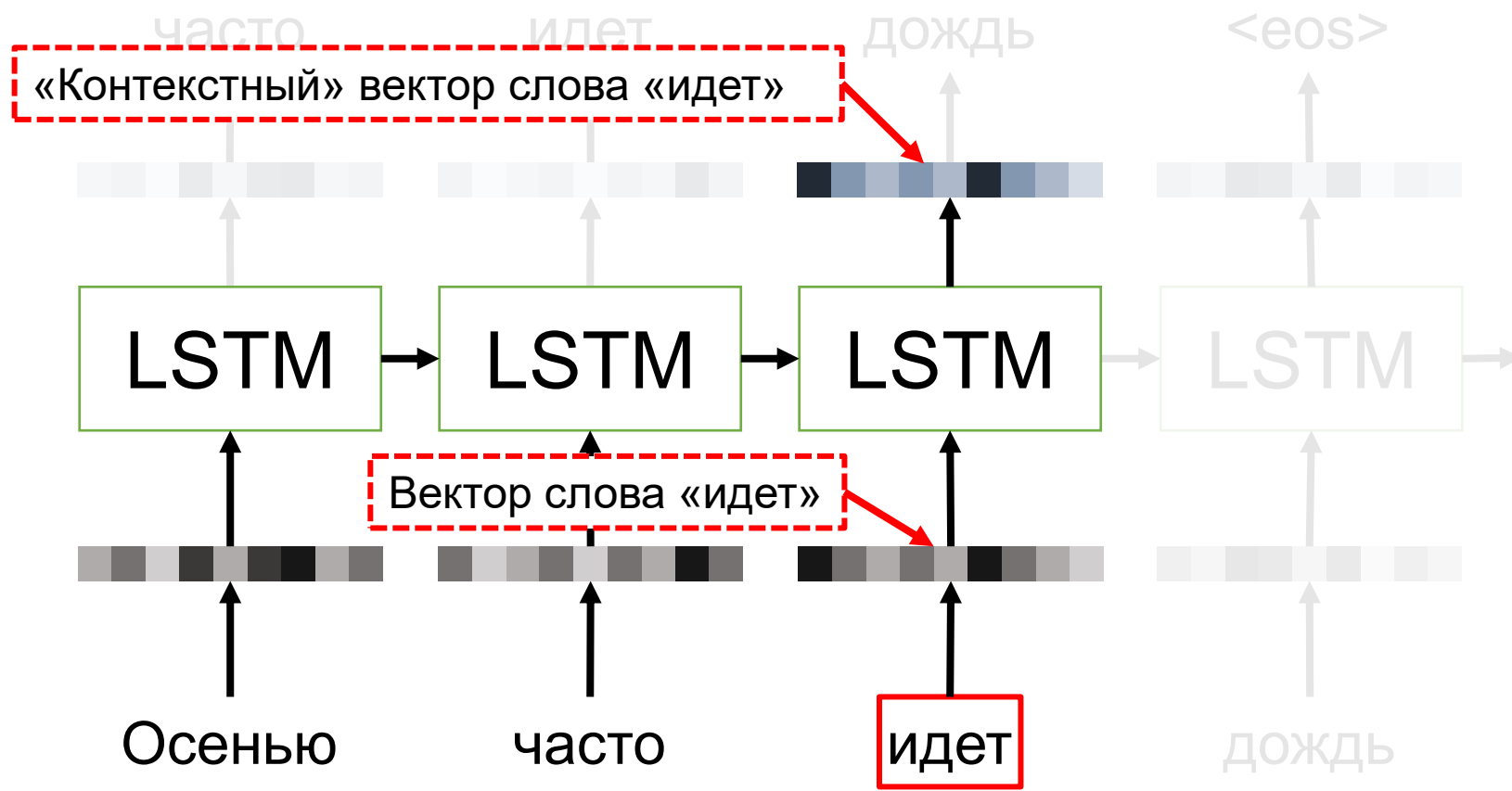
- $P(w_n | w_1, \dots, w_{n-1})$

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

# Языковые модели на RNN

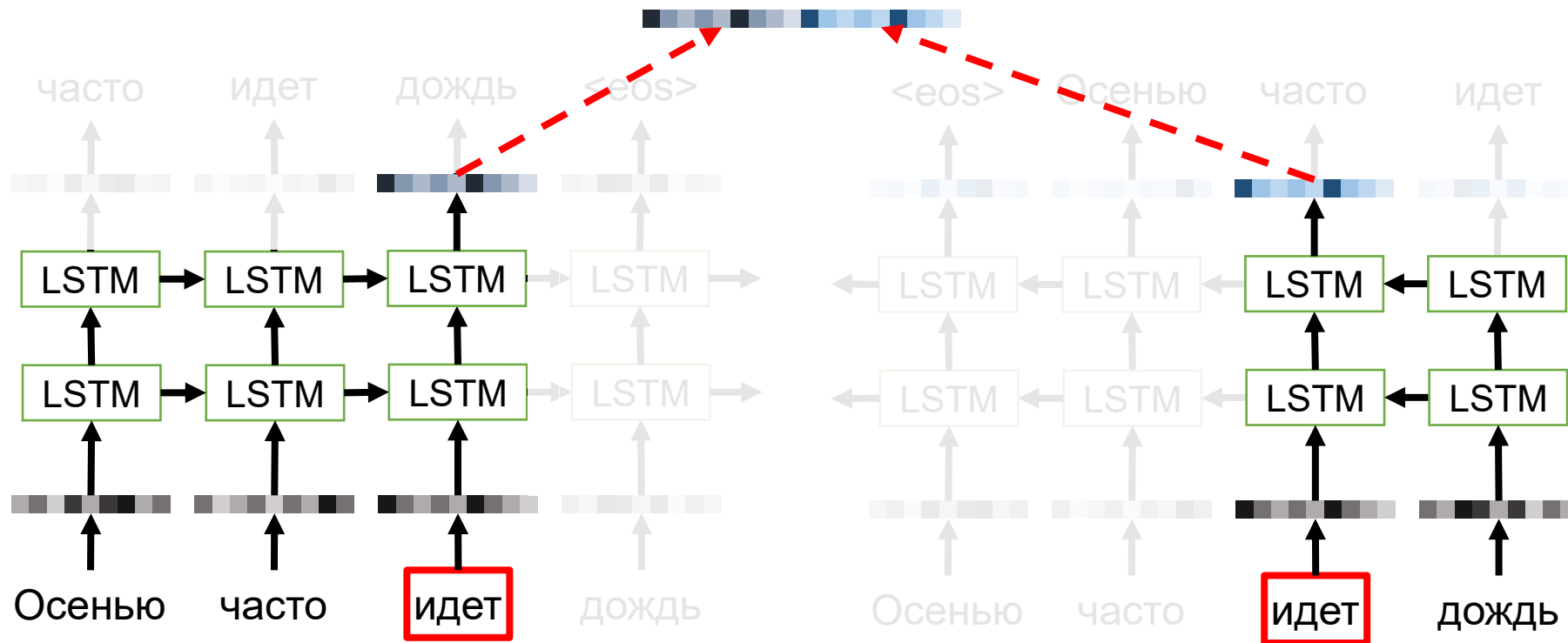


# Языковые модели на RNN

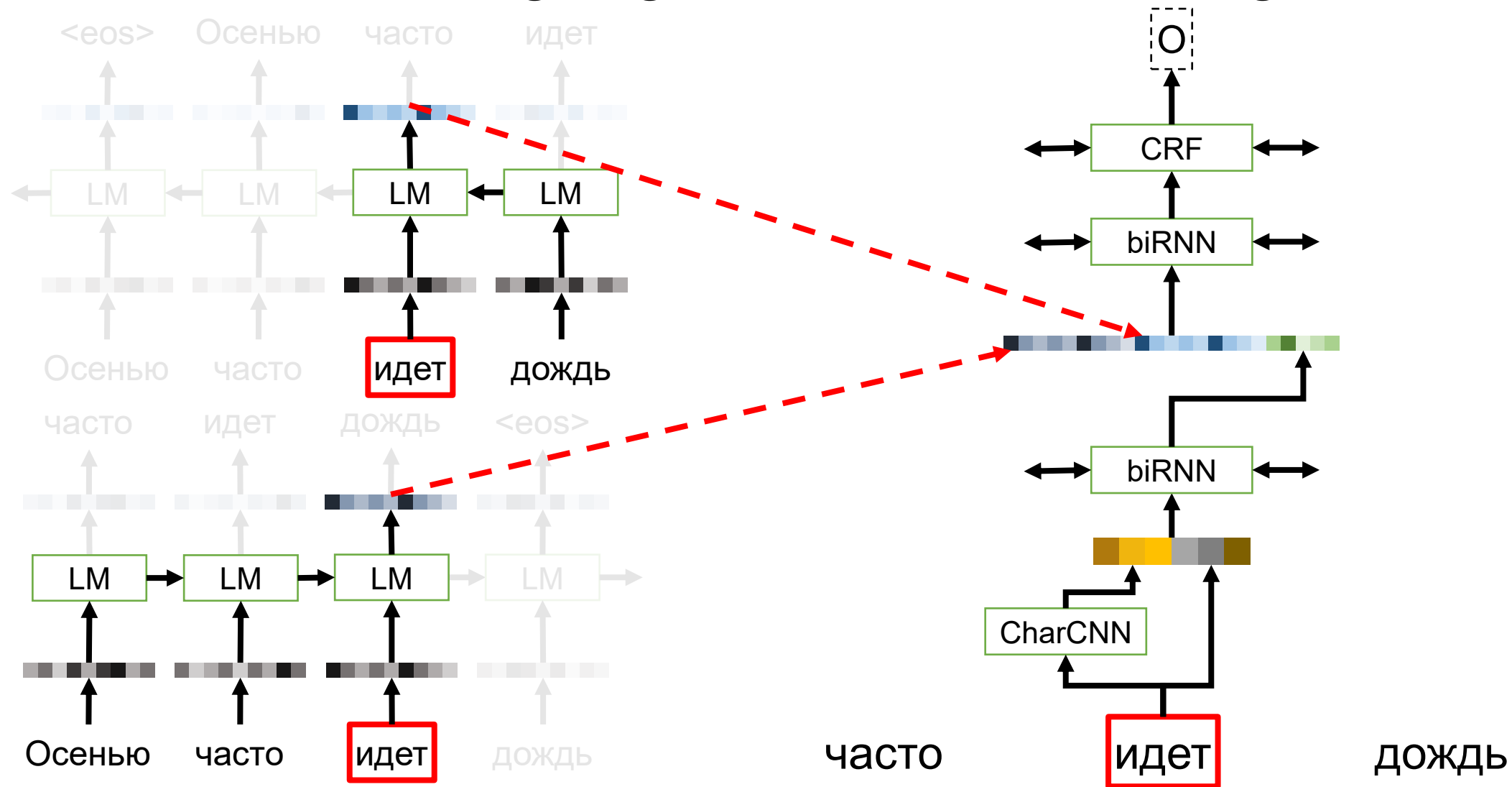


# Language Model Embedding

- 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:

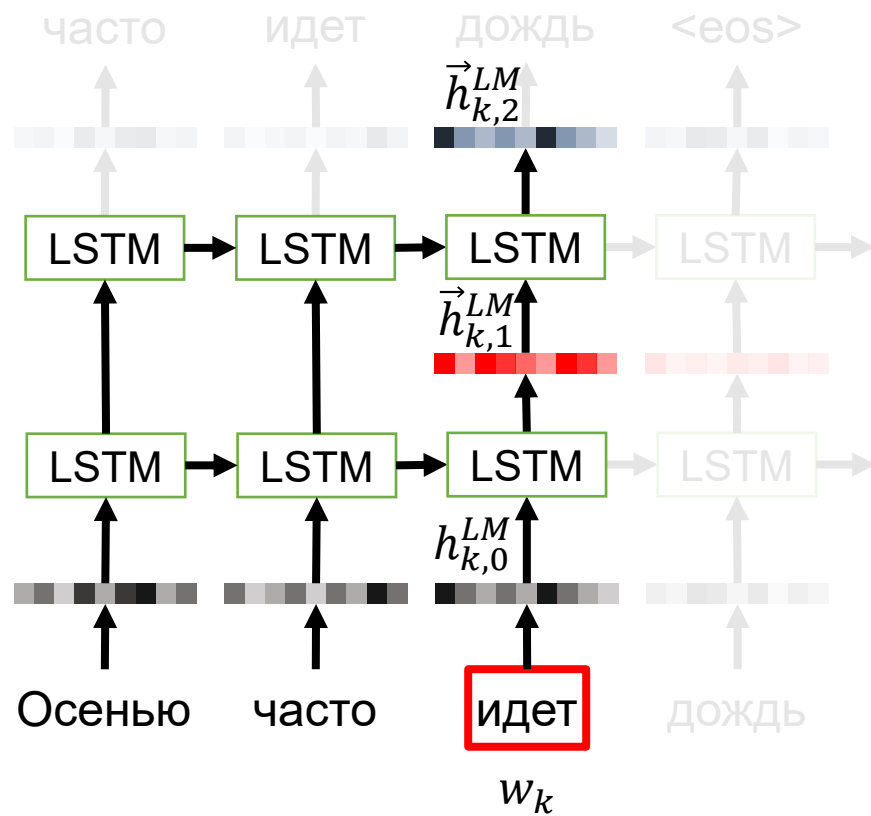
$$p(w_1, \dots, w_N) = \prod_{k=1}^N p(w_k | w_{k+1}, \dots, w_N)$$

- Обучаем прямую и обратную языковую модель одновременно:

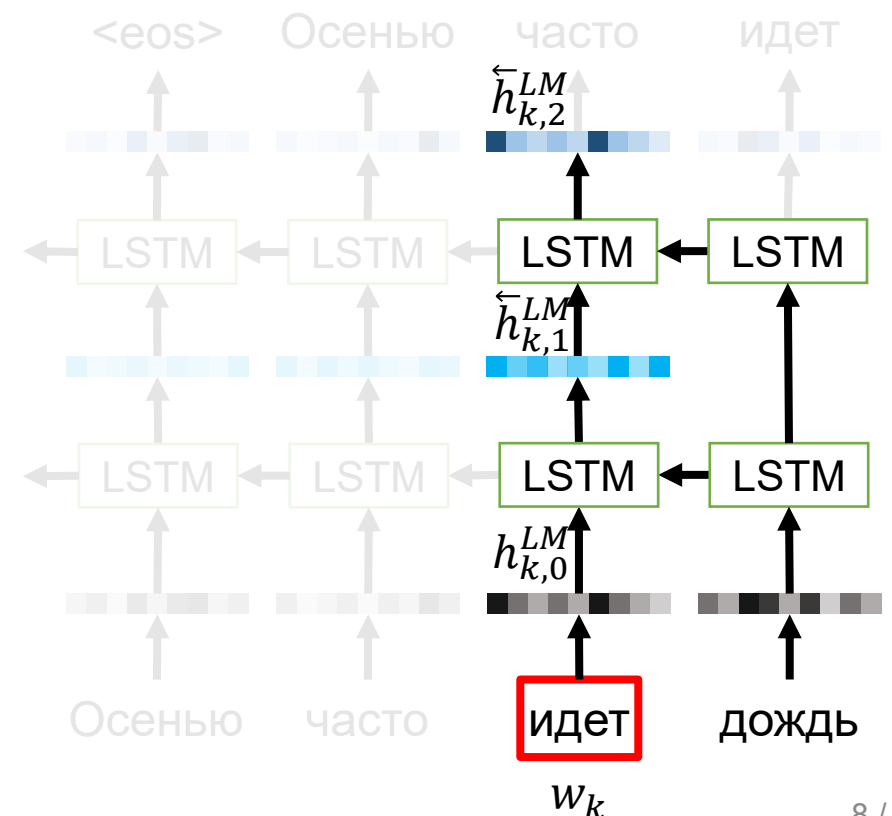
$$J = \sum_{k=1}^N [\log p(w_k | w_1, \dots, w_{k-1}; \Theta_x, \Theta_{\overrightarrow{LSTM}}, \Theta_s) + \log p(w_k | w_{k+1}, \dots, w_N; \Theta_x, \Theta_{\overleftarrow{LSTM}}, \Theta_s)]$$

# ELMo (Embeddings from Language Models)

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

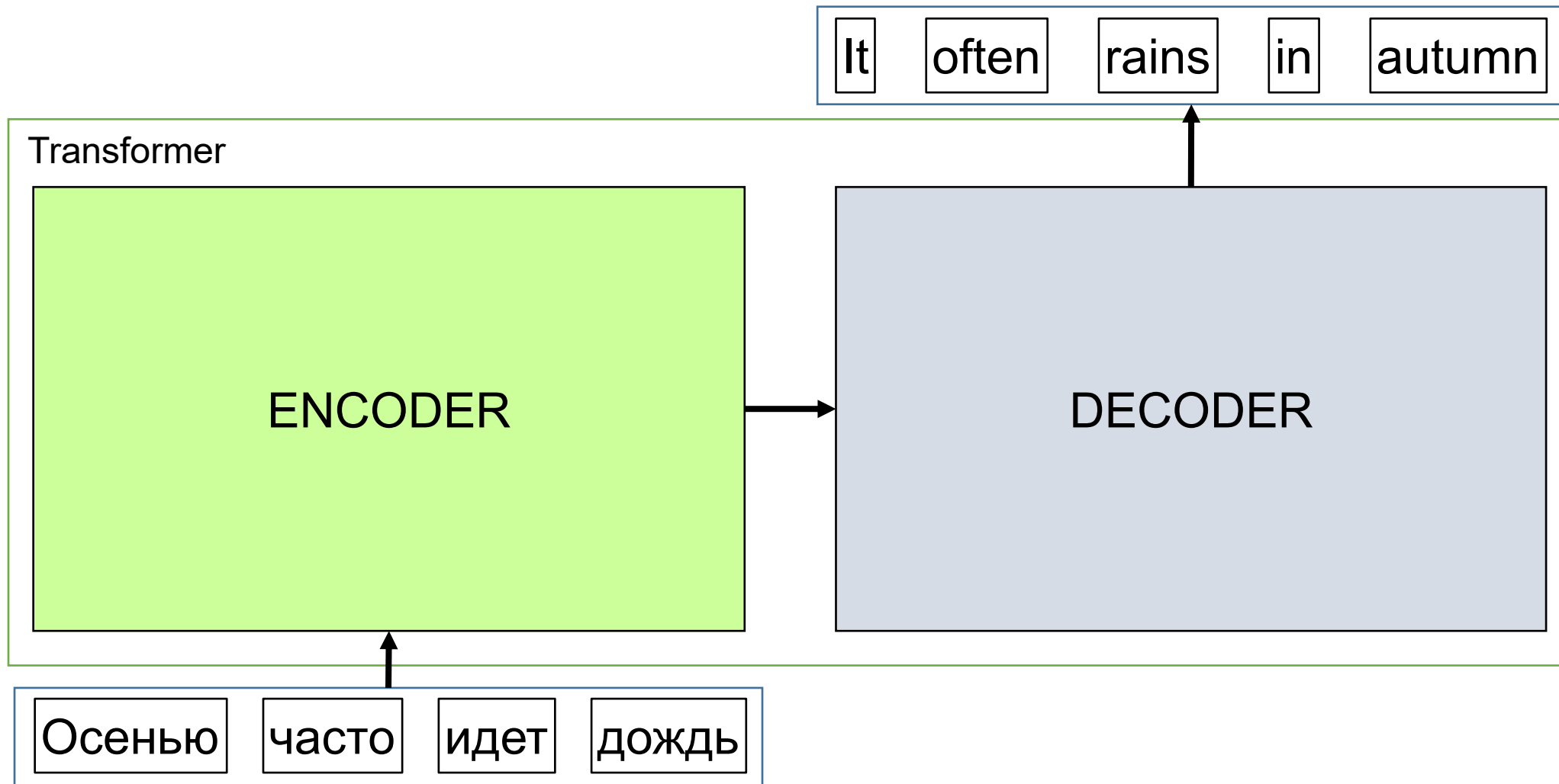


$$s_2 * \vec{h}_{k,2}^{LM} + s_1 * \vec{h}_{k,1}^{LM} + s_0 * \vec{h}_{k,0}^{LM} =$$



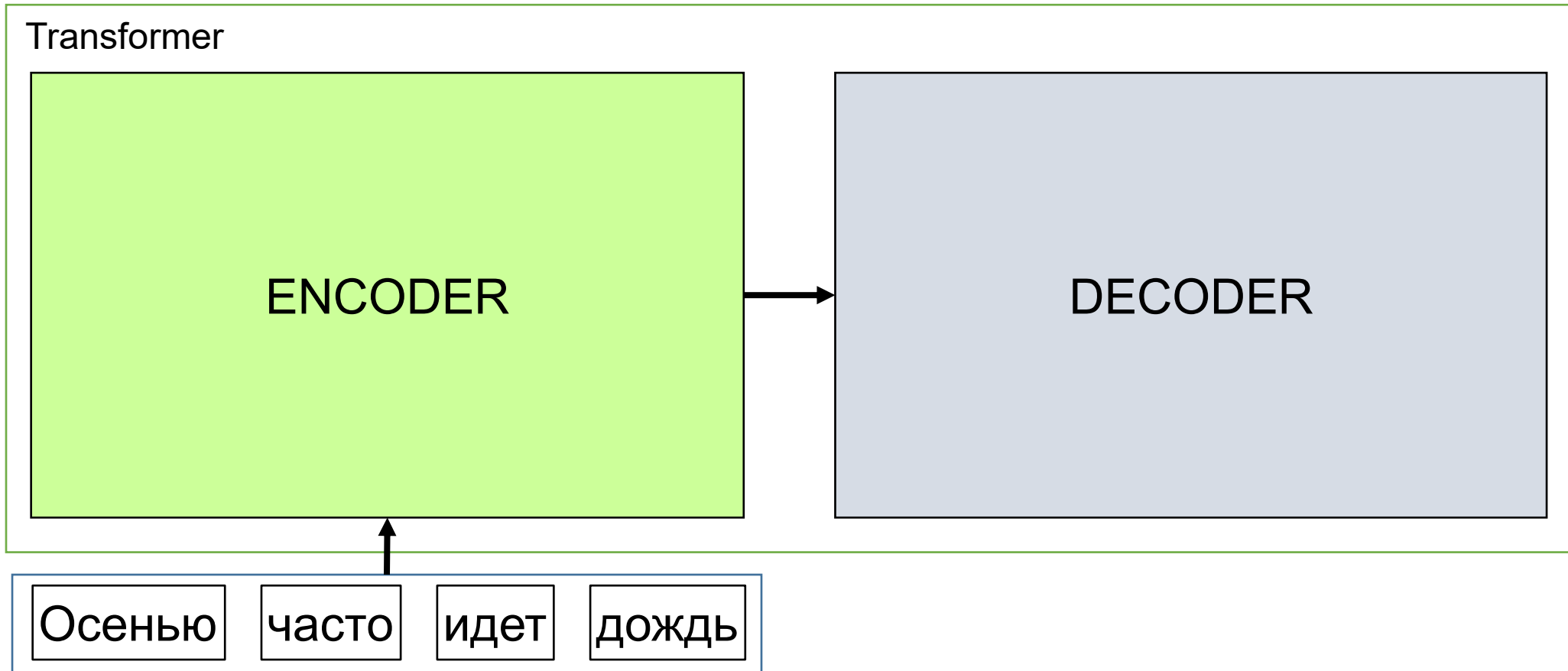


# Transformer\*

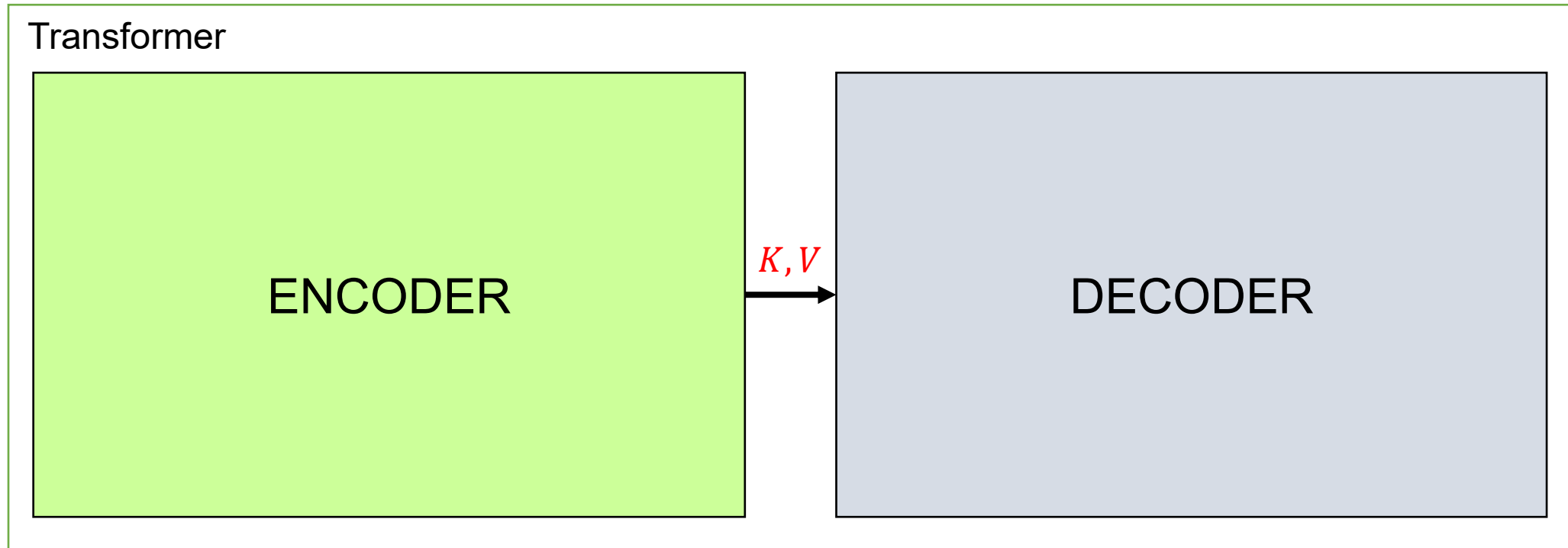


\*подробное объяснение про трансформер есть [здесь](#)

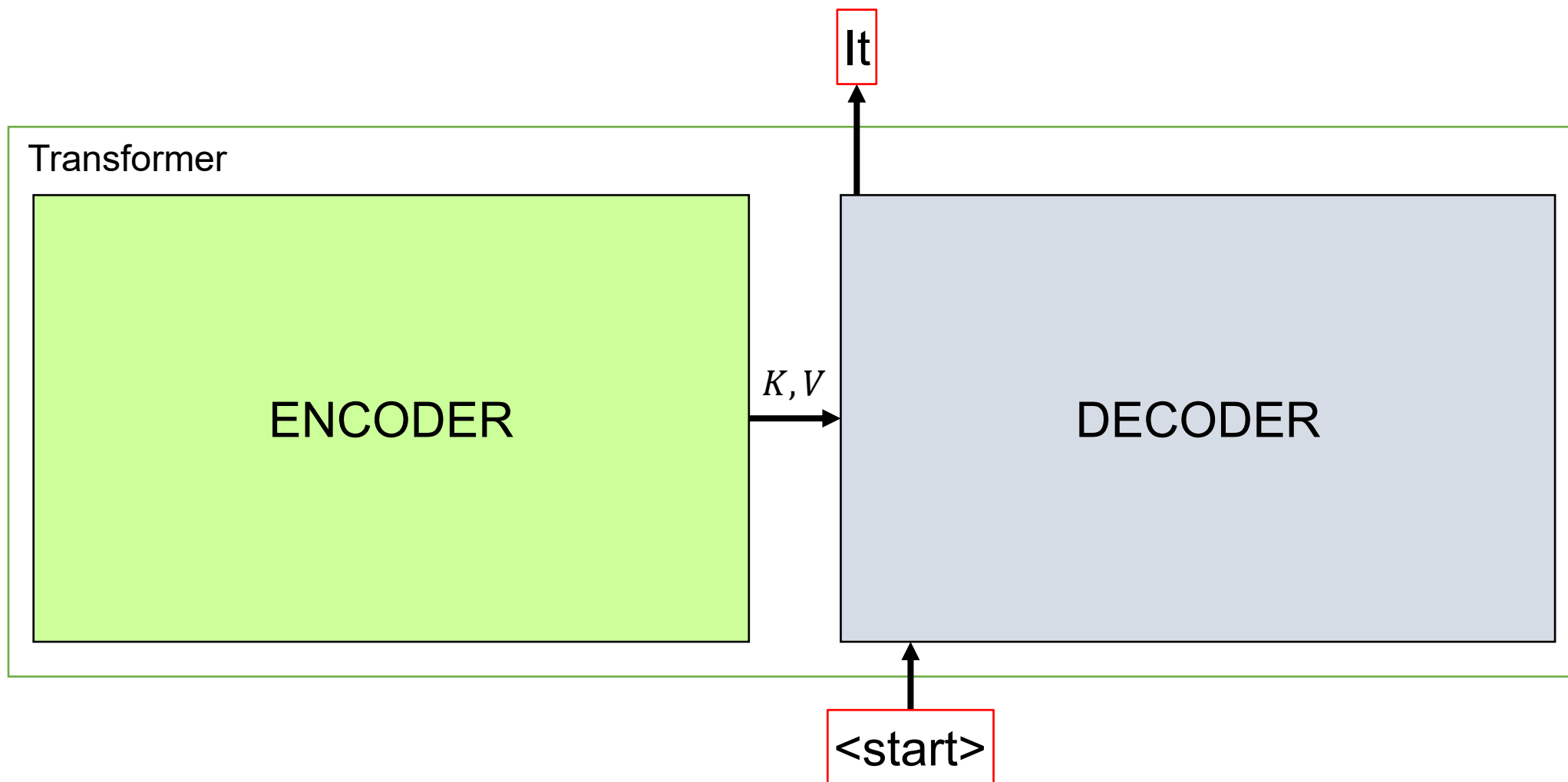
# Transformer



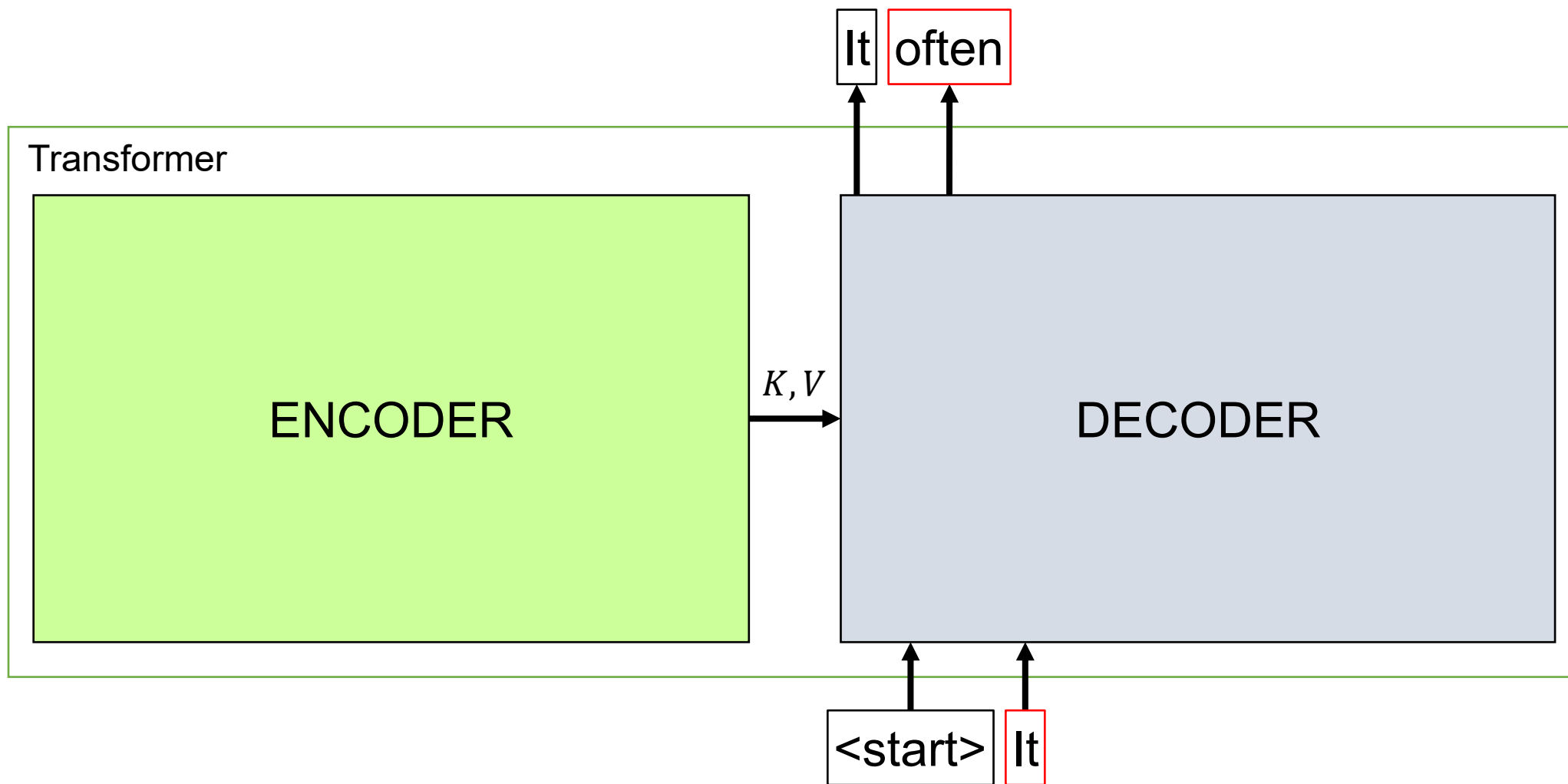
# Transformer



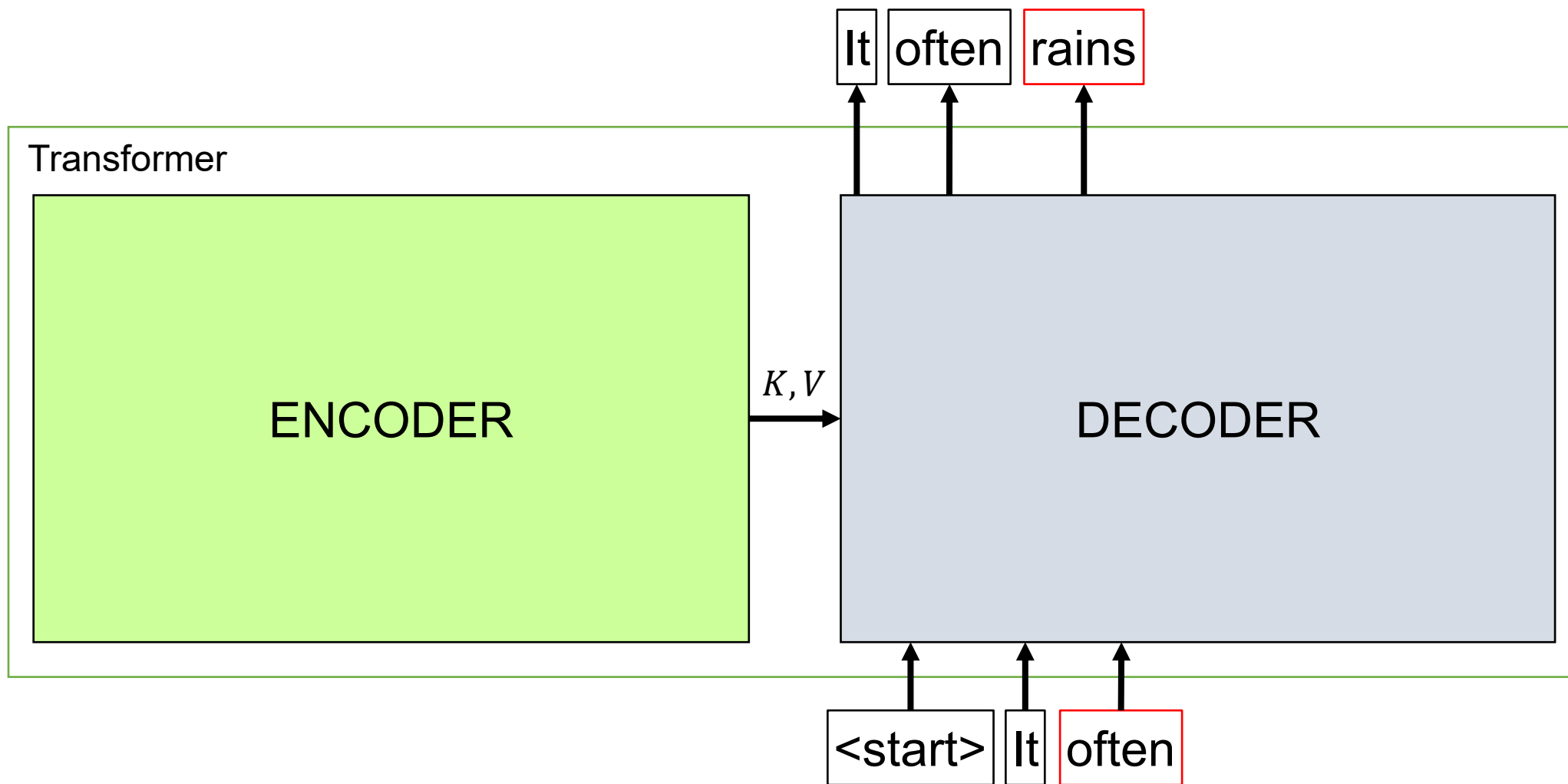
# Transformer



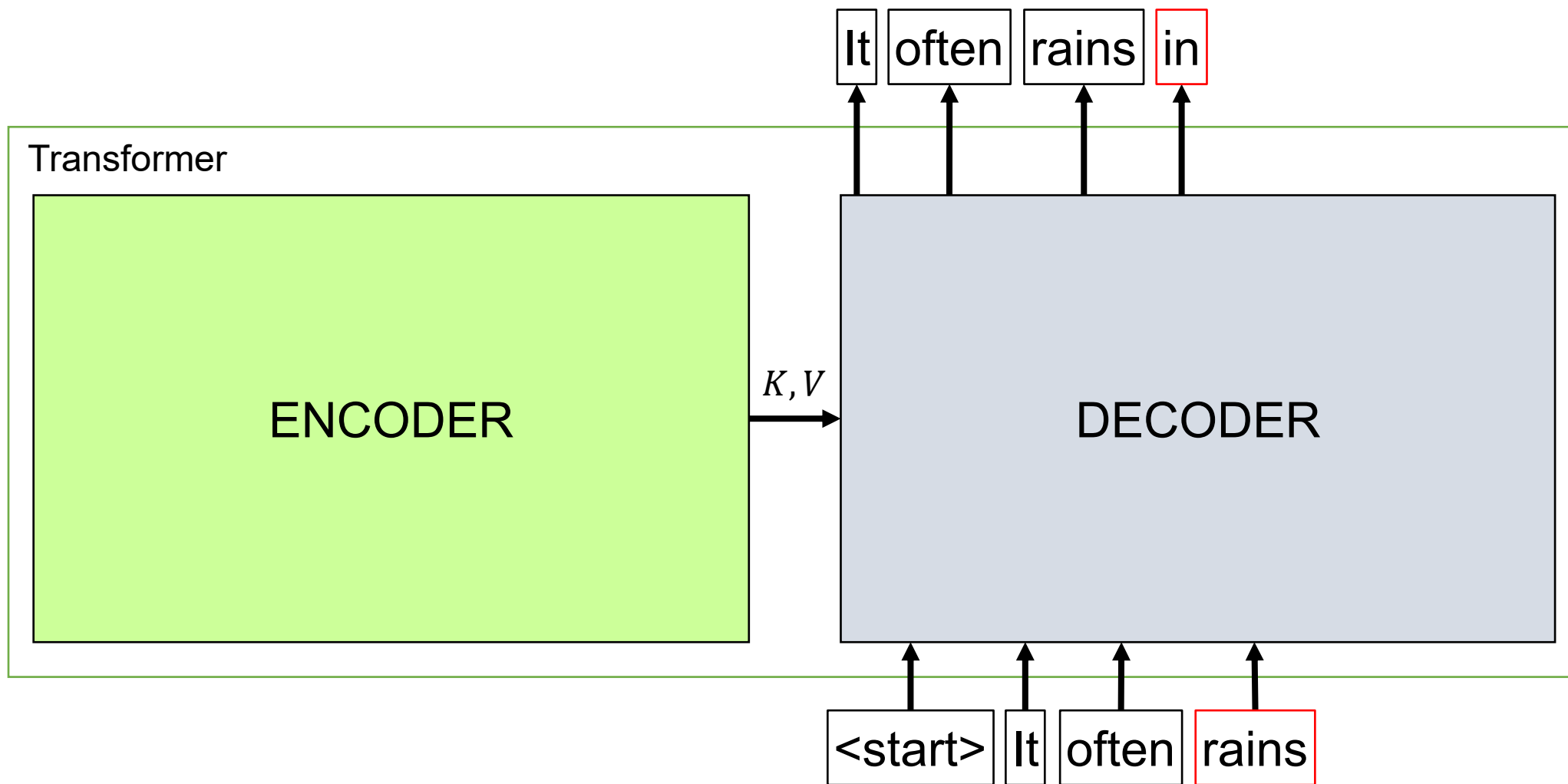
# Transformer



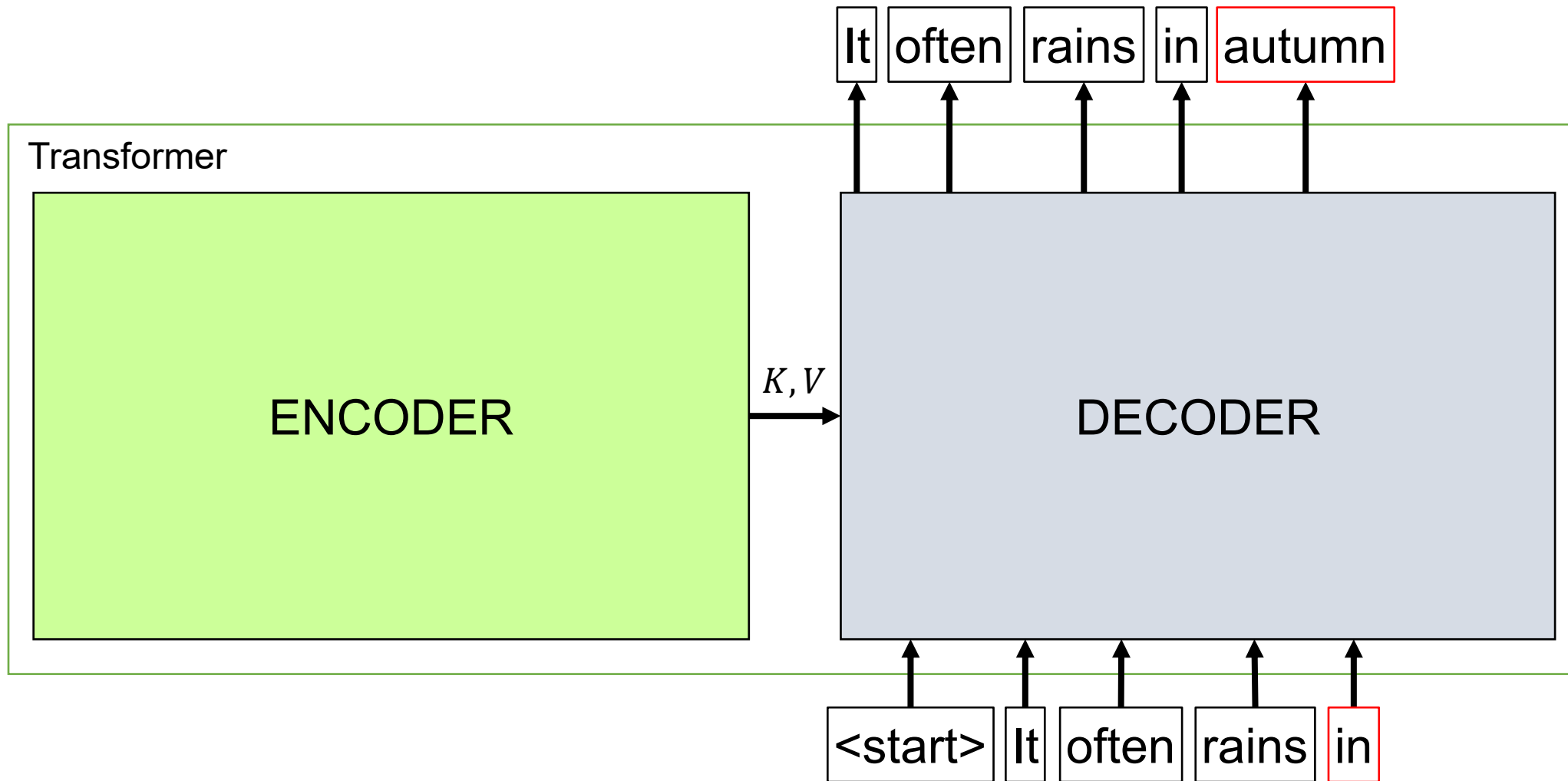
# Transformer



# Transformer

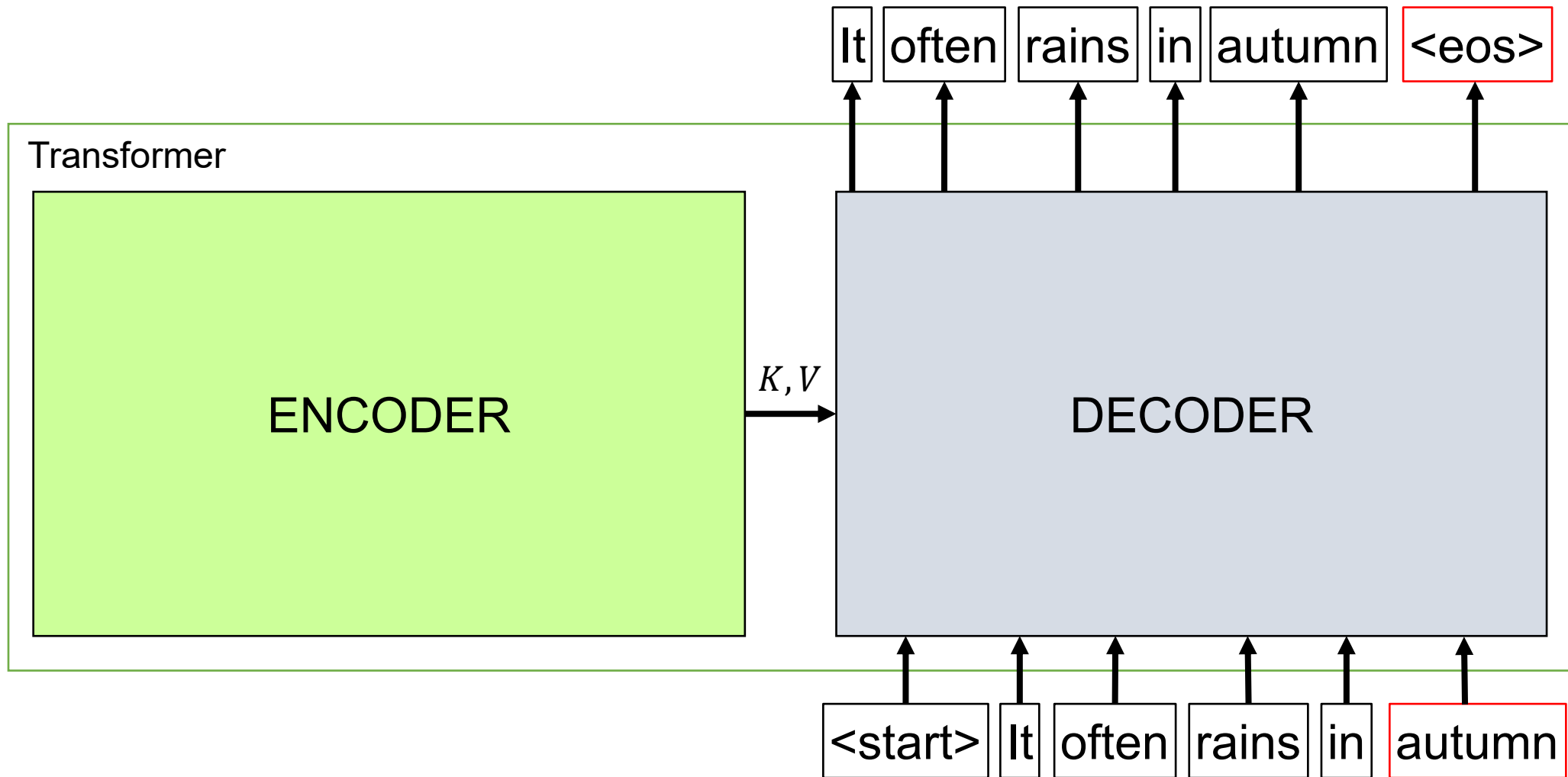


# Transformer





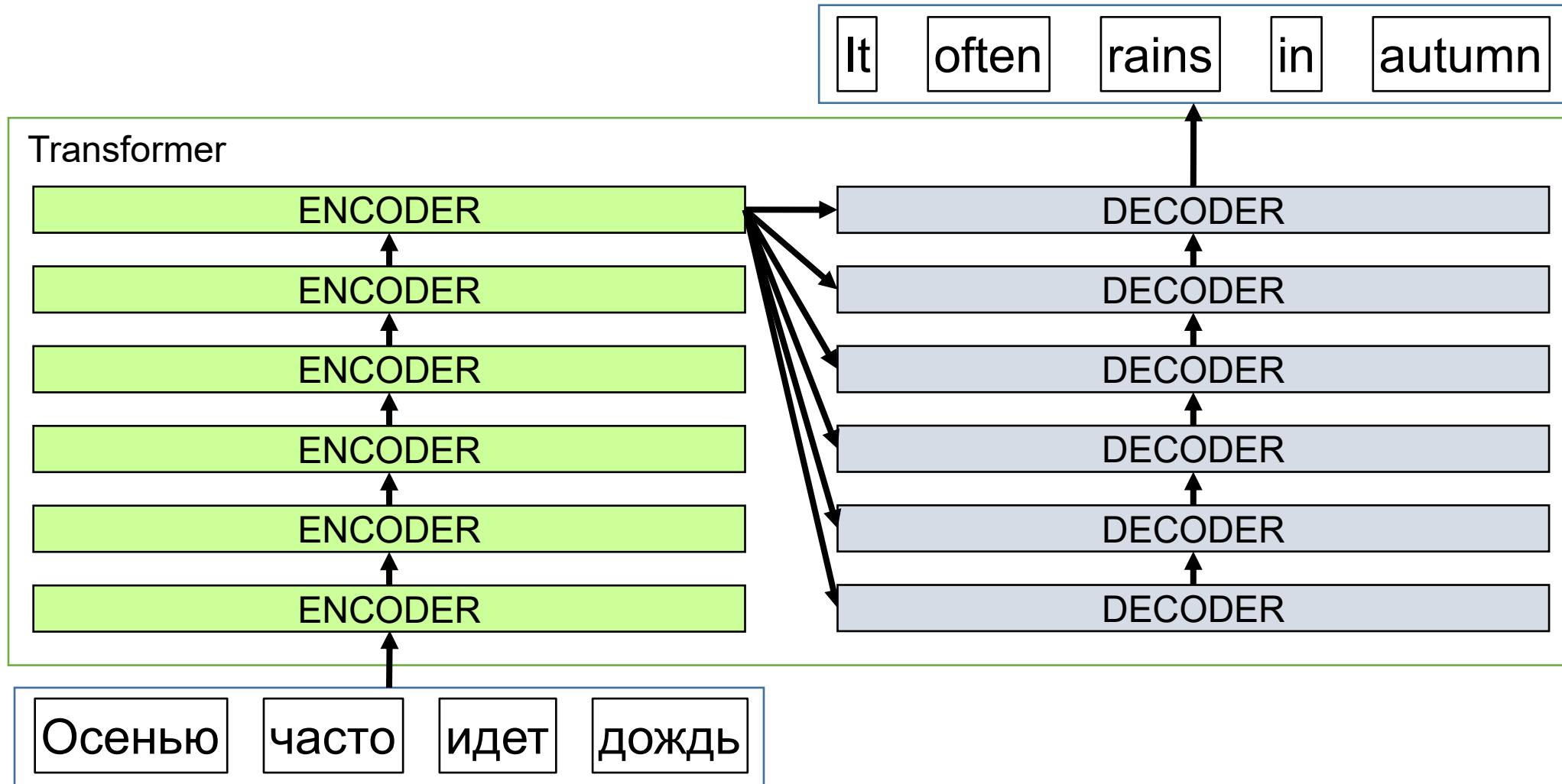
# Transformer



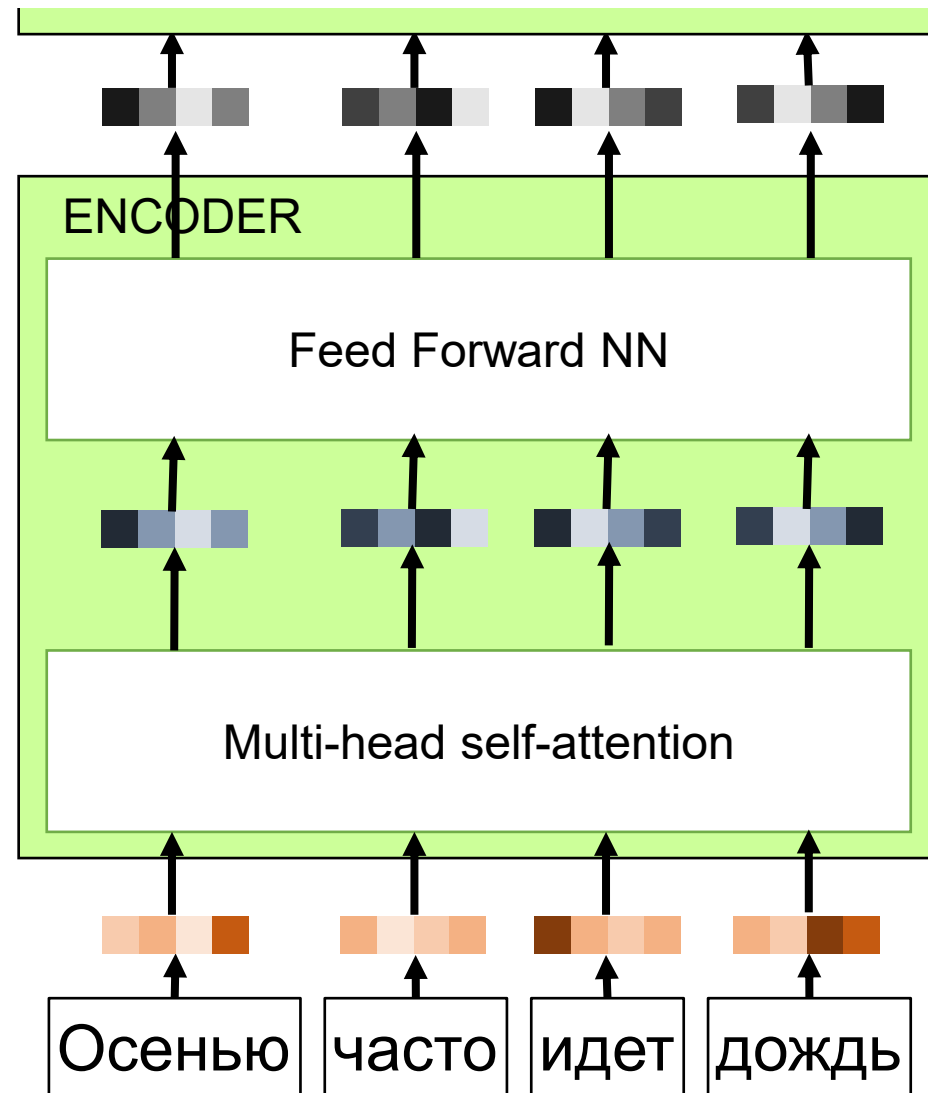
# BPE (Byte Pair Encoding)

- 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`

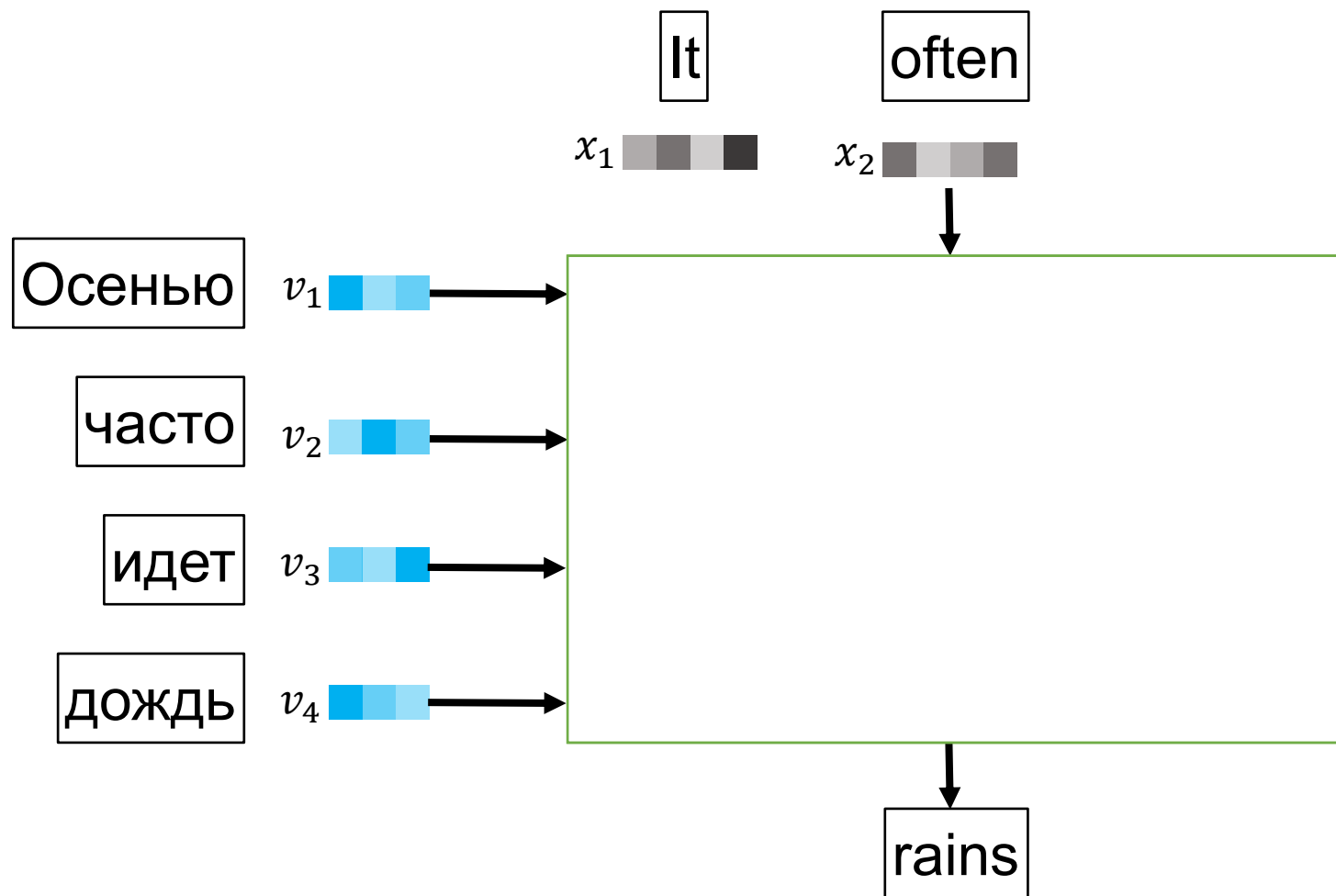
# Transformer



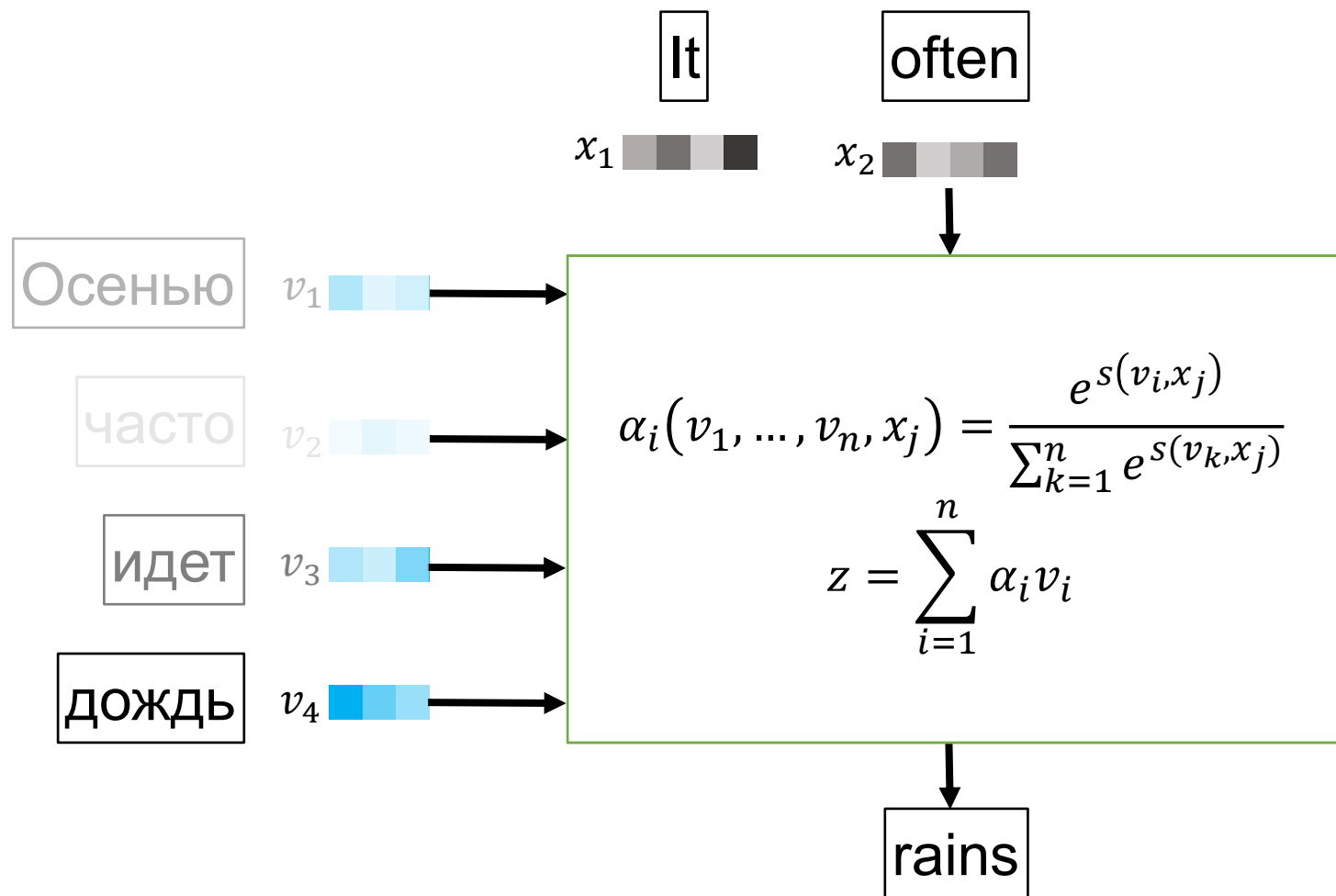
# Transformer Encoder



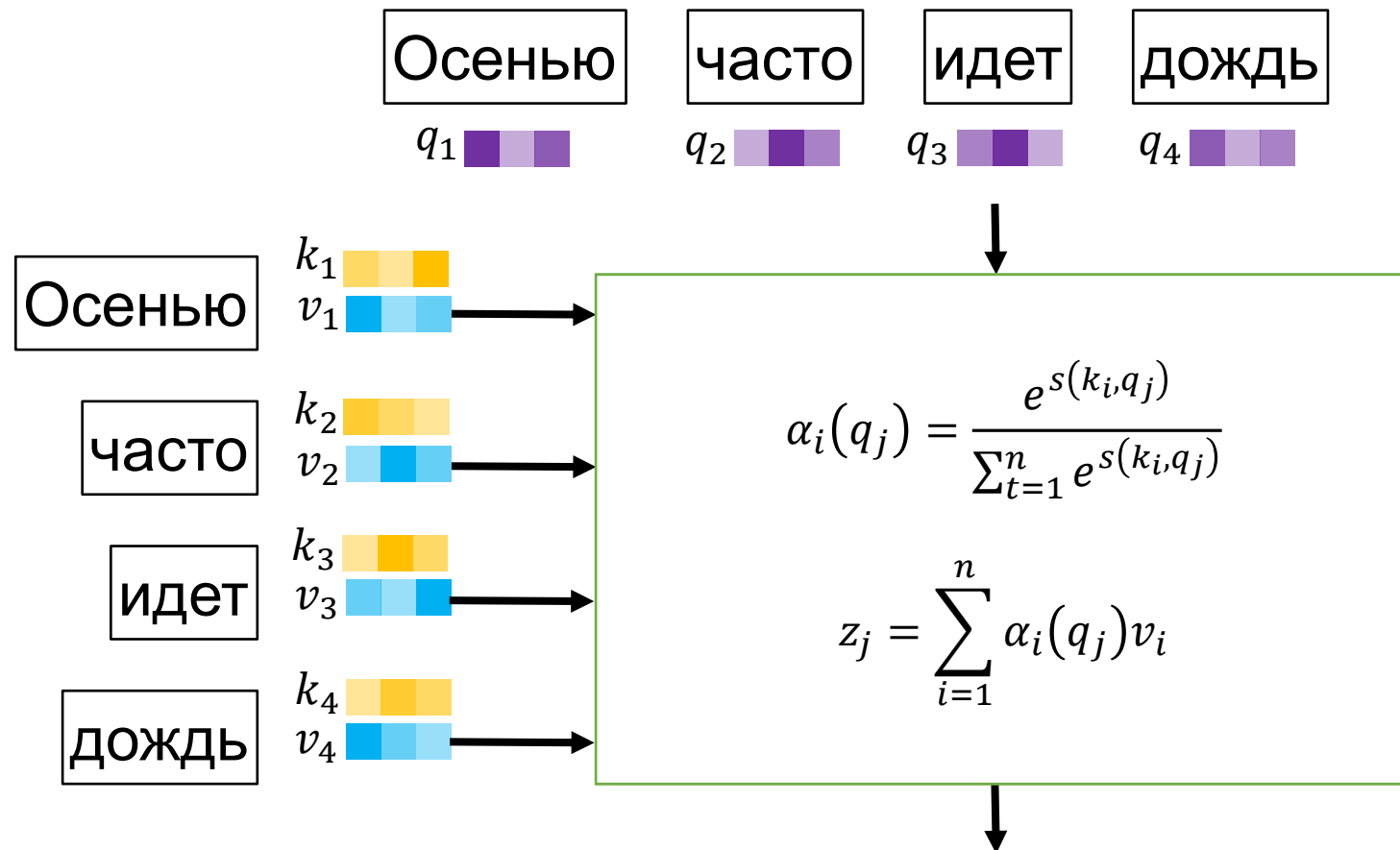
# Механизм внимания



# 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

Вход:

Осенью

часто

идет

дождь

Векторы:

$x_1$  

$x_2$  

$x_3$  

$x_4$  

Запросы:


$q_1$  


$q_2$  

$q_3$  

$q_4$  

Ключи:

$k_1$  

$k_2$  

$k_3$  

$k_4$  

Значения:

$v_1$  

$v_2$  

$v_3$  

$v_4$  

# Scaled Dot-Product Self-attention

Вход:

Осенью

часто

идет

дождь

Векторы:

$x_1$  

$x_2$  

$x_3$  

$x_4$  

Запросы:

$q_1$  

$q_2$  

$q_3$  

$q_4$  

Ключи:

$k_1$  

$k_2$  

$k_3$  

$k_4$  

Оценки:

$q_3 k_1 / \sqrt{d}$

$q_3 k_2 / \sqrt{d}$

$q_3 k_3 / \sqrt{d}$

$q_3 k_4 / \sqrt{d}$

softmax:

0.1

0.08

0.7

0.12

Значения:

$v_1$  

$v_2$  

$v_3$  

$v_4$  

# Scaled Dot-Product Self-attention

Вход:

Осенью

часто

идет

дождь

Векторы:

$x_1$  

$x_2$  

$x_3$  

$x_4$  

Запросы:

$q_1$  

$q_2$  

$q_3$  

$q_4$  

Ключи:

$k_1$  

$k_2$  

$k_3$  

$k_4$  

Оценки:

$q_3 k_1 / \sqrt{d}$

$q_3 k_2 / \sqrt{d}$

$q_3 k_3 / \sqrt{d}$

$q_3 k_4 / \sqrt{d}$

softmax:

0.1

0.08

0.7

0.12

Значения:

$v_1$  

$v_2$  

$v_3$  

$v_4$  

Сумма:

$z_1$  

$z_2$  

$z_3$  

$z_4$  

# Scaled Dot-Product Self-attention

Вход:

Осенью    часто    идет    дождь

Векторы:

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

$$Q = X * W^Q$$

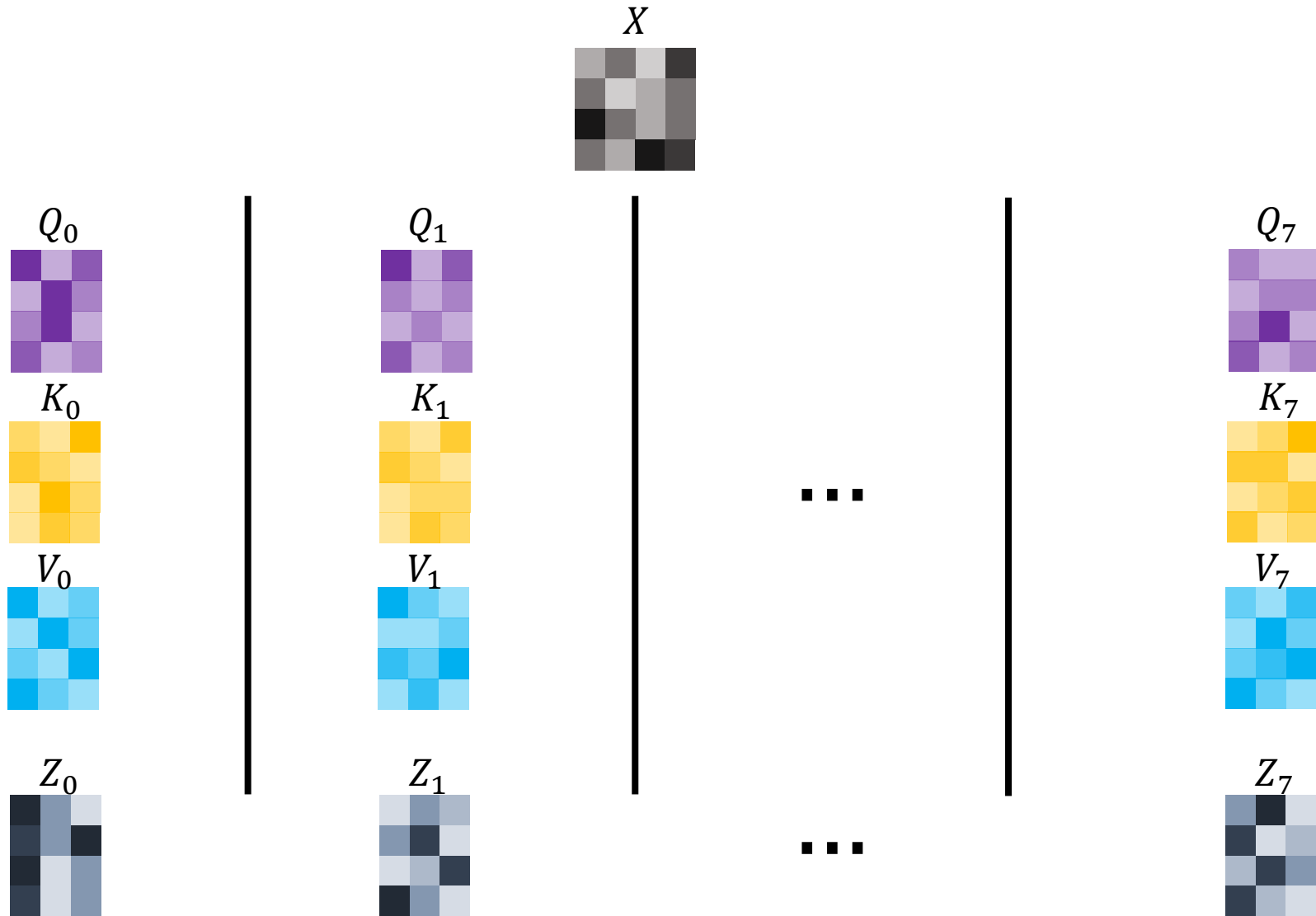
$$K = X * W^K$$

$$V = X * W^V$$

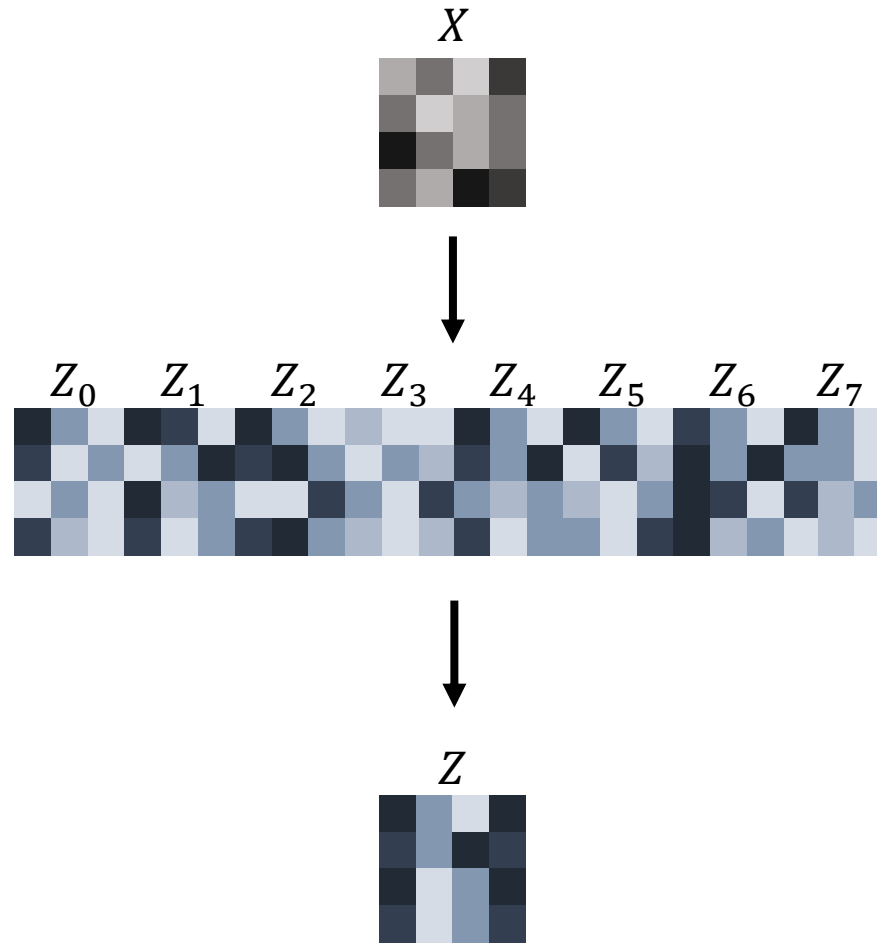
$$Z = \text{softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right)V$$

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

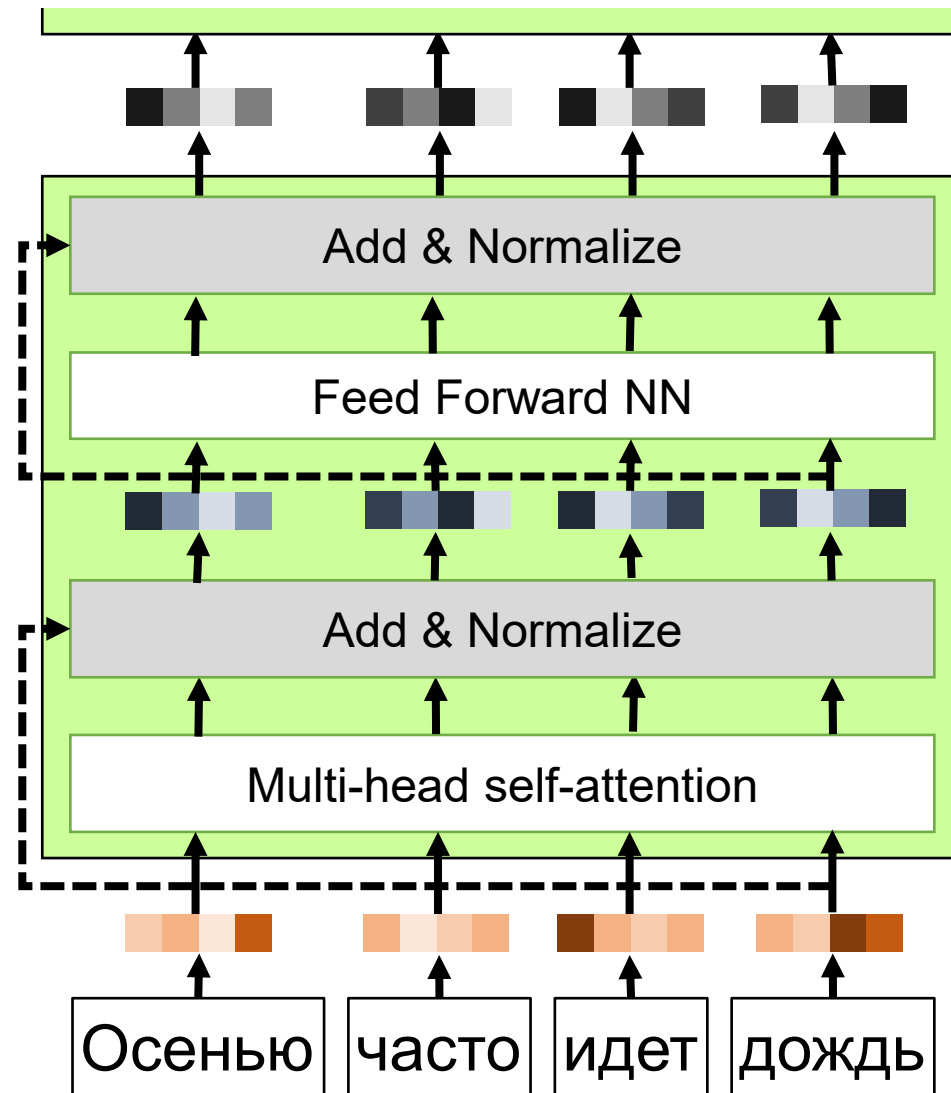
# Multi-head attention



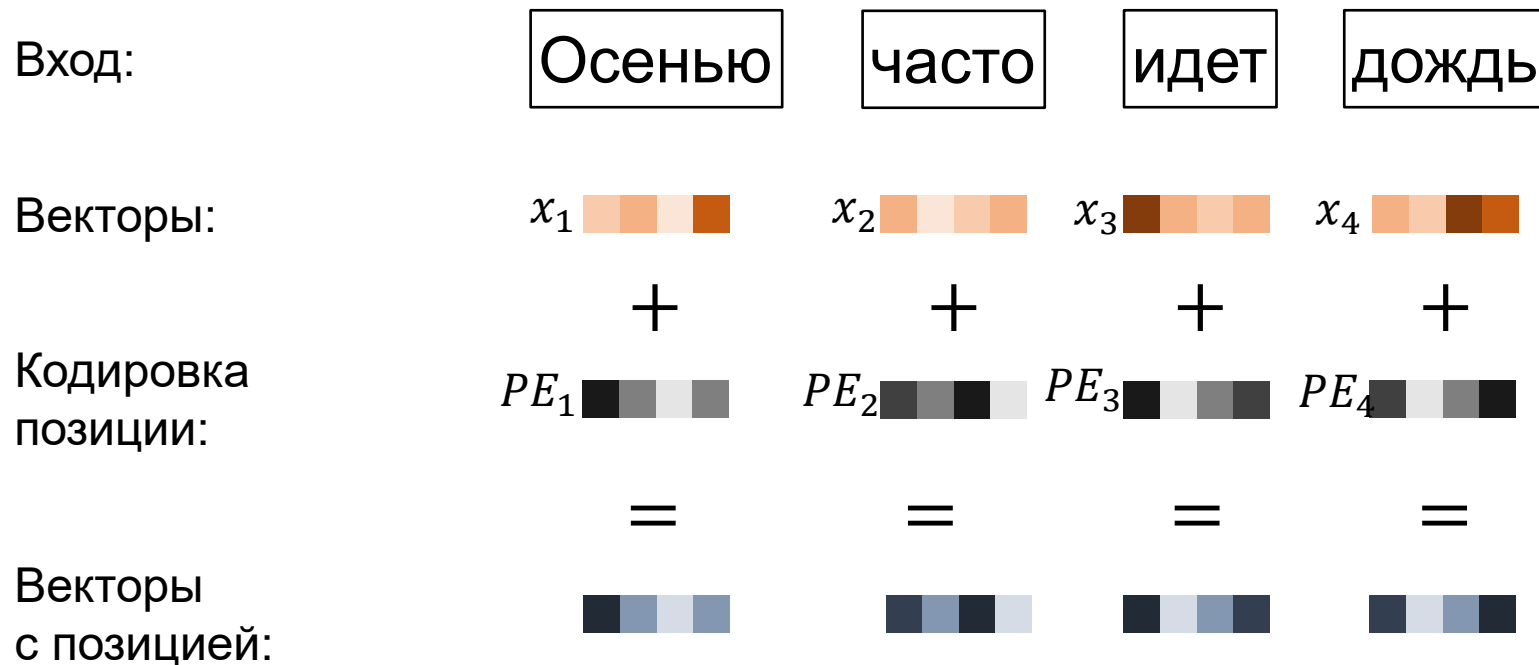
# Multi-head attention



# Transformer Encoder



# Позиционное кодирование



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

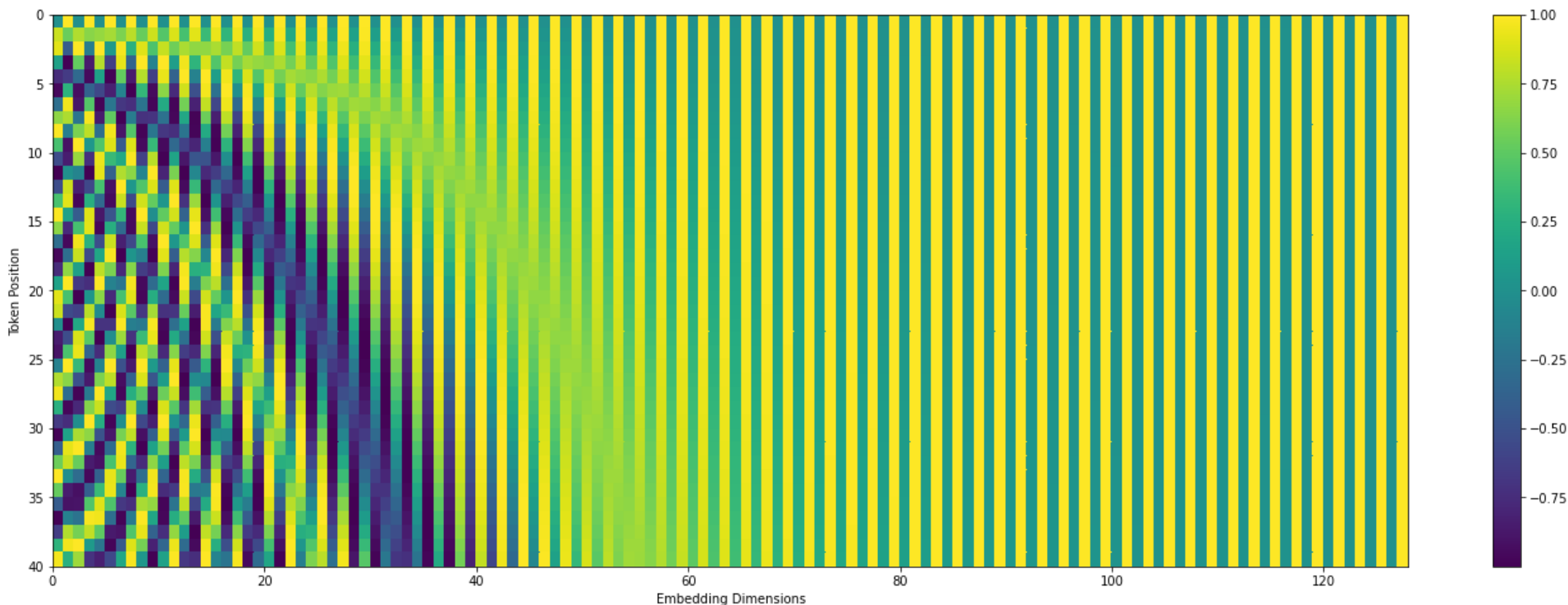
$$PE_{pos,2i+1} = \cos(pos / 10000^{2i/d_{model}})$$



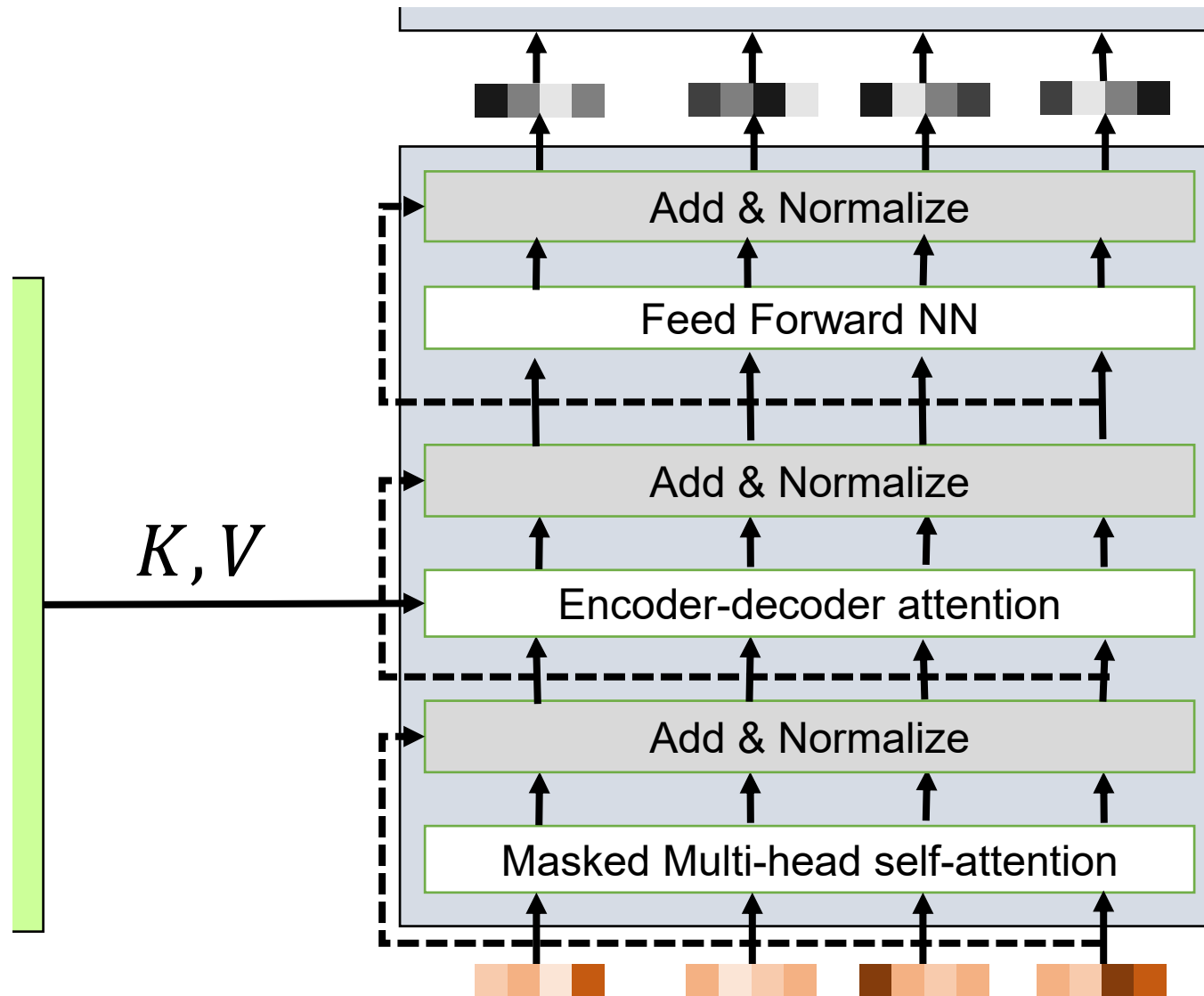
# Позиционное кодирование

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

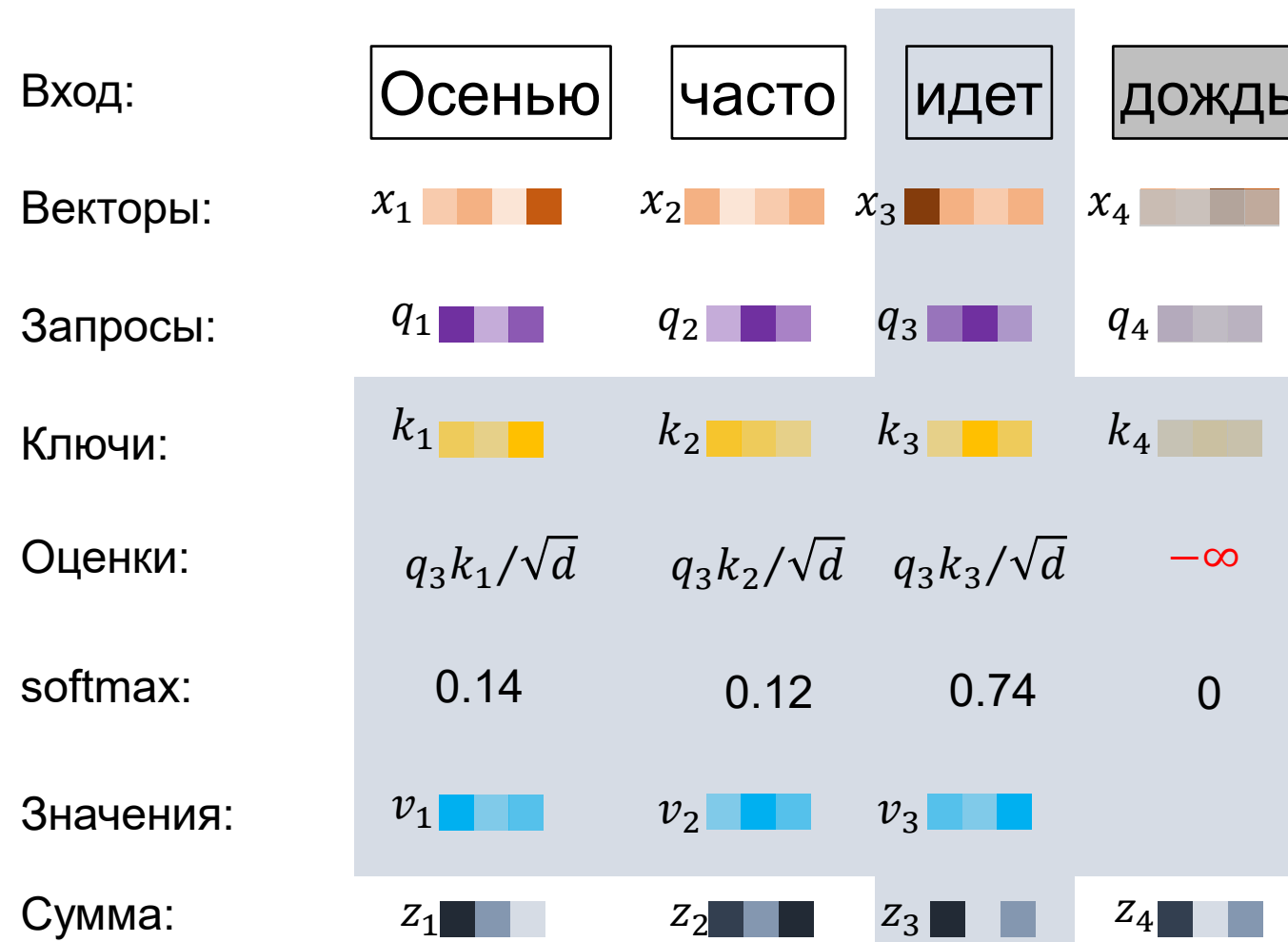
$$PE_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}})$$



# Transformer Decoder



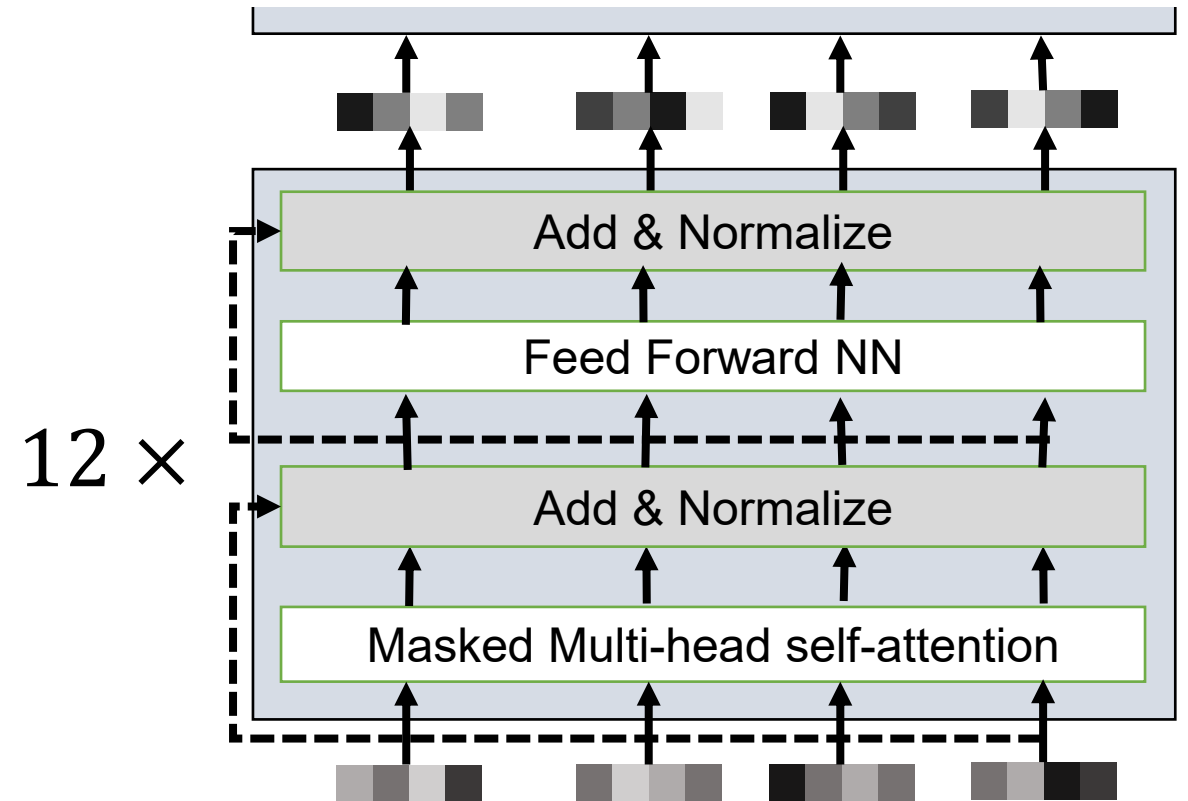
# Masked Scaled Dot-Product Self-attention



# 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)

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

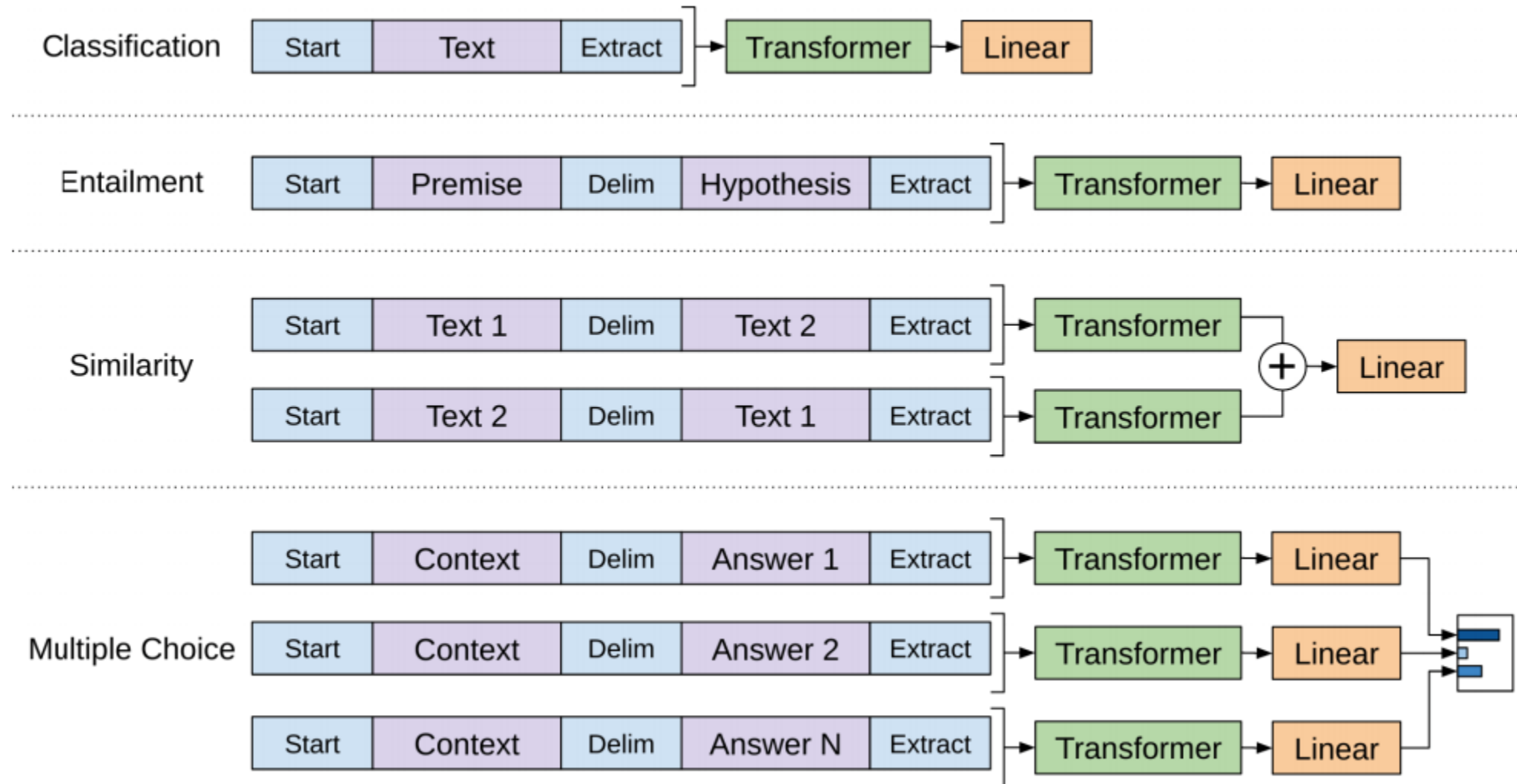
$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

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

$$\begin{aligned}P(y | x^1, \dots, x^m) &= \text{softmax}(h_l^m \mathbf{W}_y) \\L_2(\mathcal{C}) &= \sum_{(x,y)} \log P(y | x^1, \dots, x^m)\end{aligned}$$

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

# GPT (Generative Pre-Training)



\*Картинка из [оригинальной статьи](#)

# GPT-2

- Большинство задач решается оценкой распределения:

$$p(output|input)$$

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

$$p(output|input, task)$$

- 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: BPE 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 реферирование
- 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

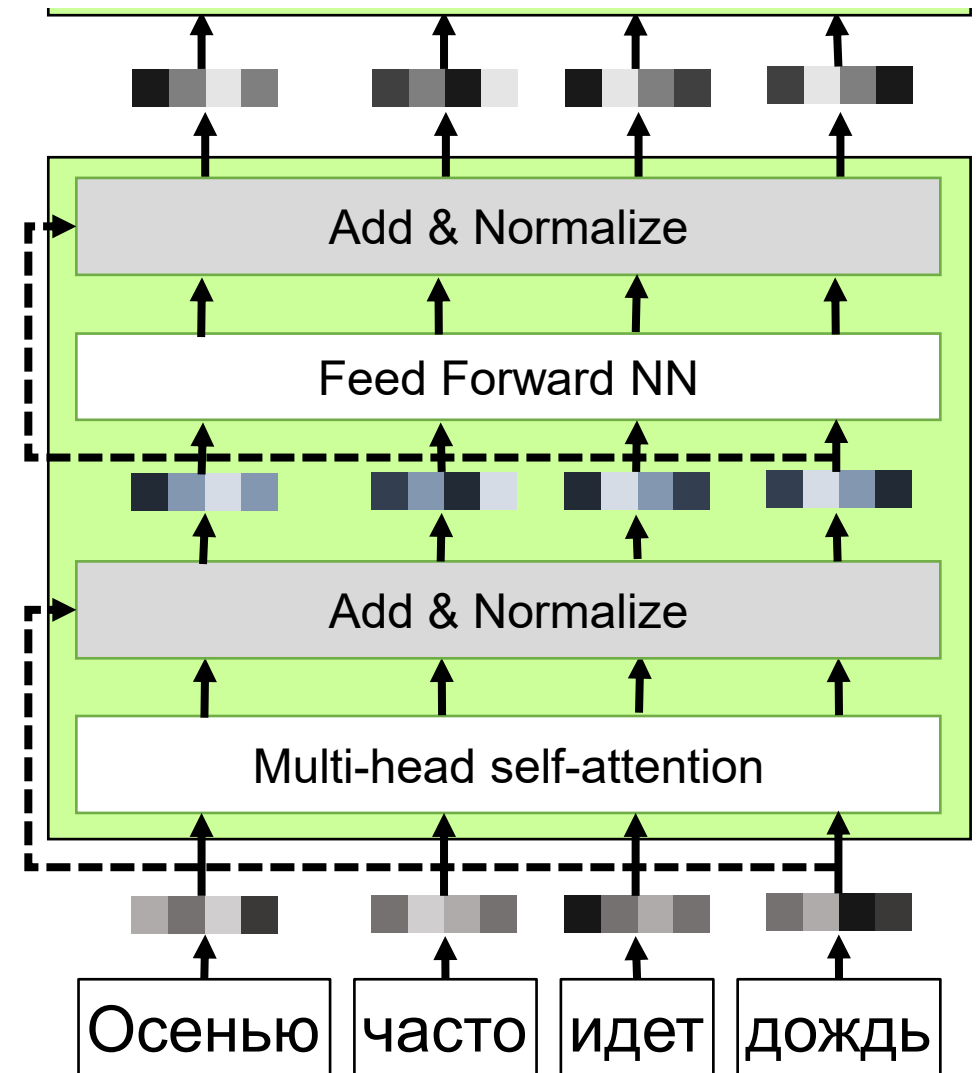
Можно применять в 3 вариантах:

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

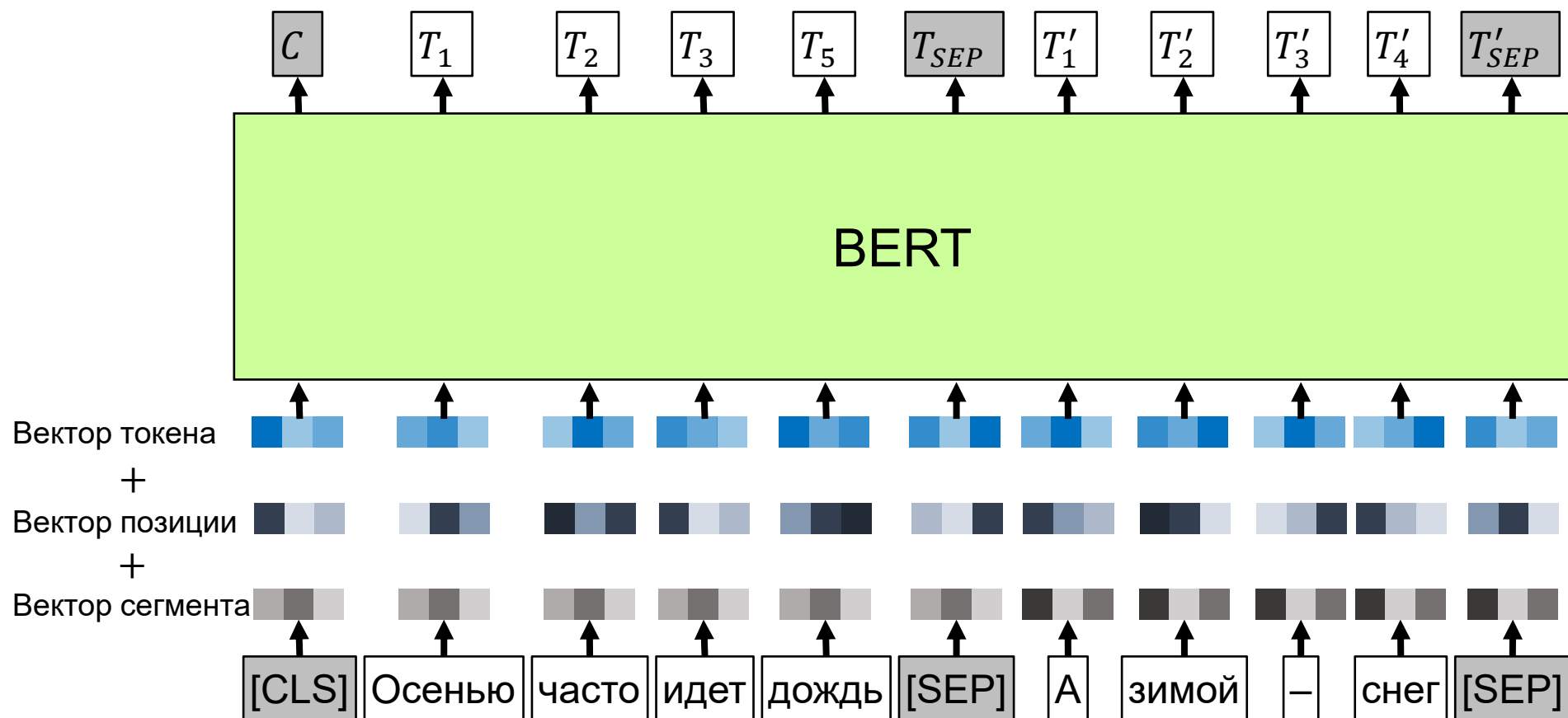
# BERT

Языковая модель, основанная на энкодере трансформера

$12(24) \times$

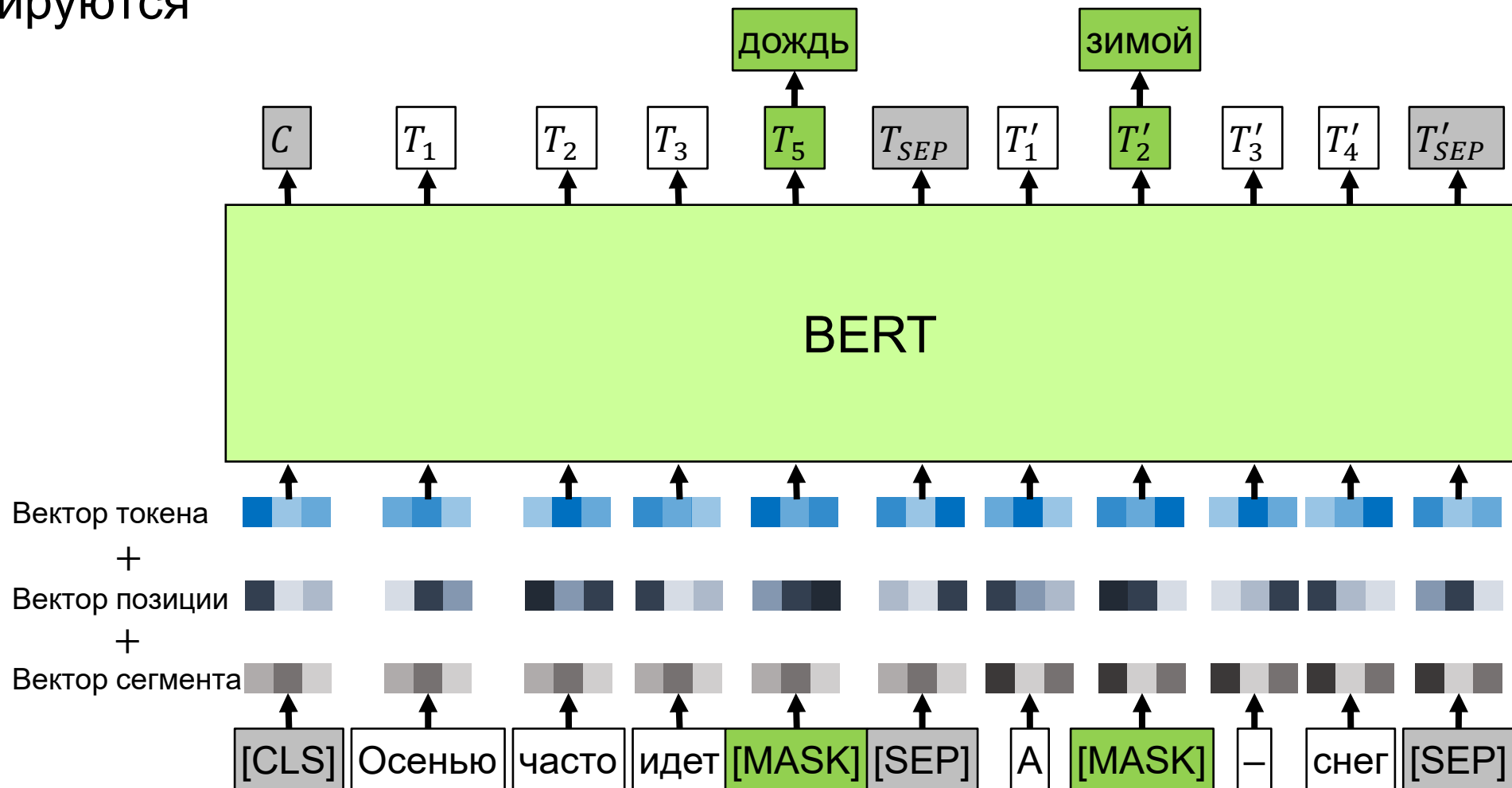


# BERT



# BERT pretraining (masked LM)

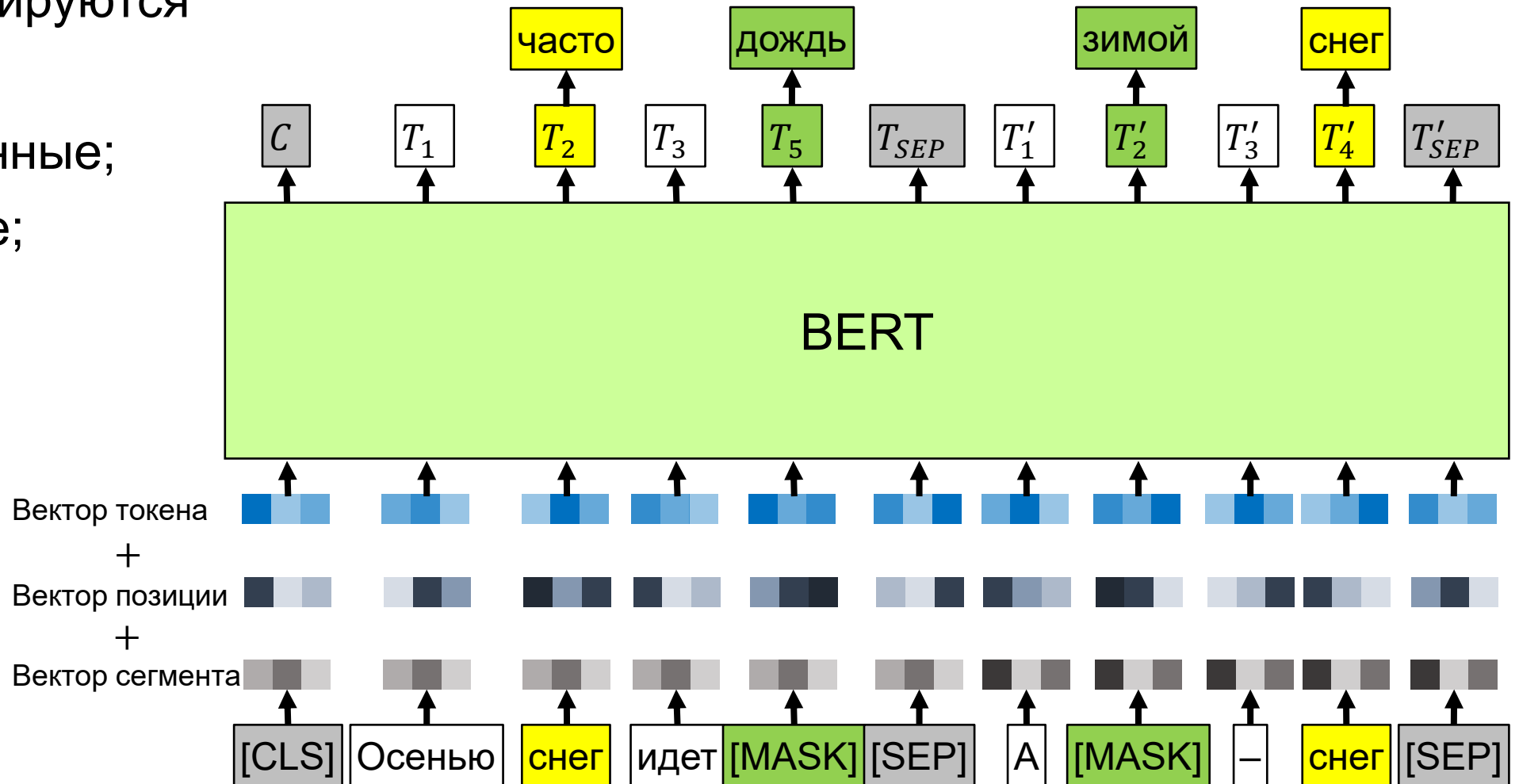
15% токенов маскируются



# BERT pretraining (masked LM)

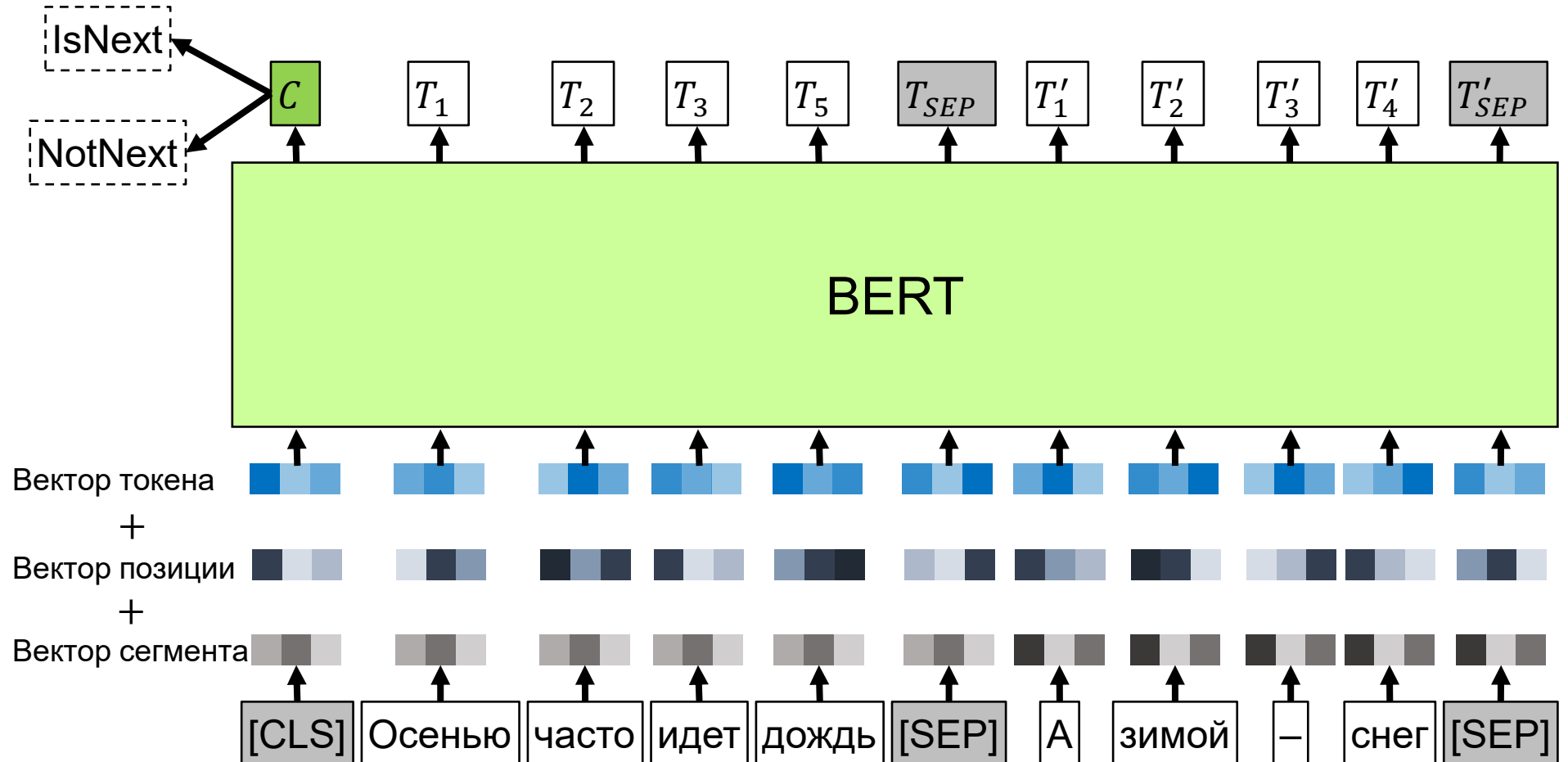
15% токенов маскируются

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



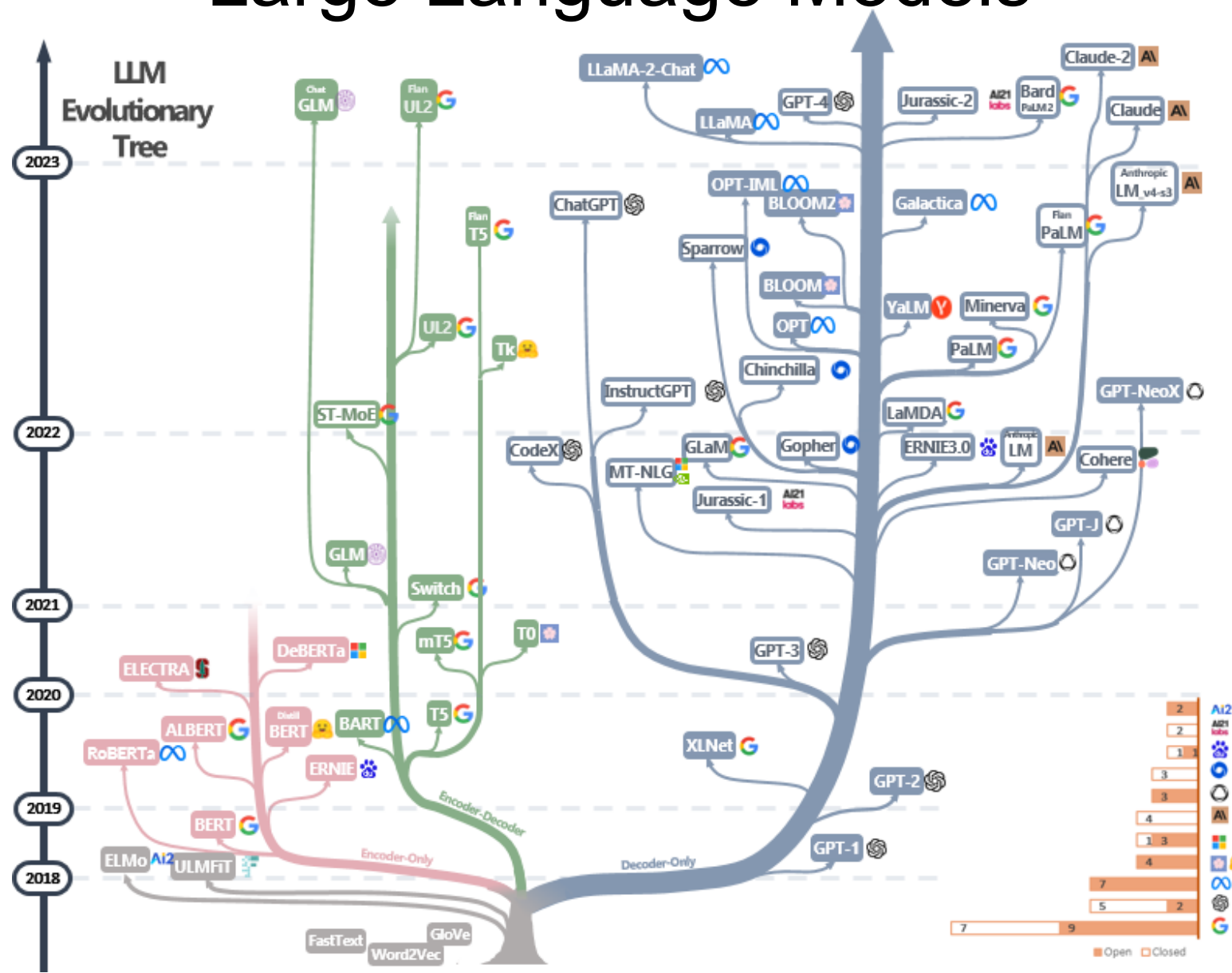
# BERT pretraining (NSP)

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

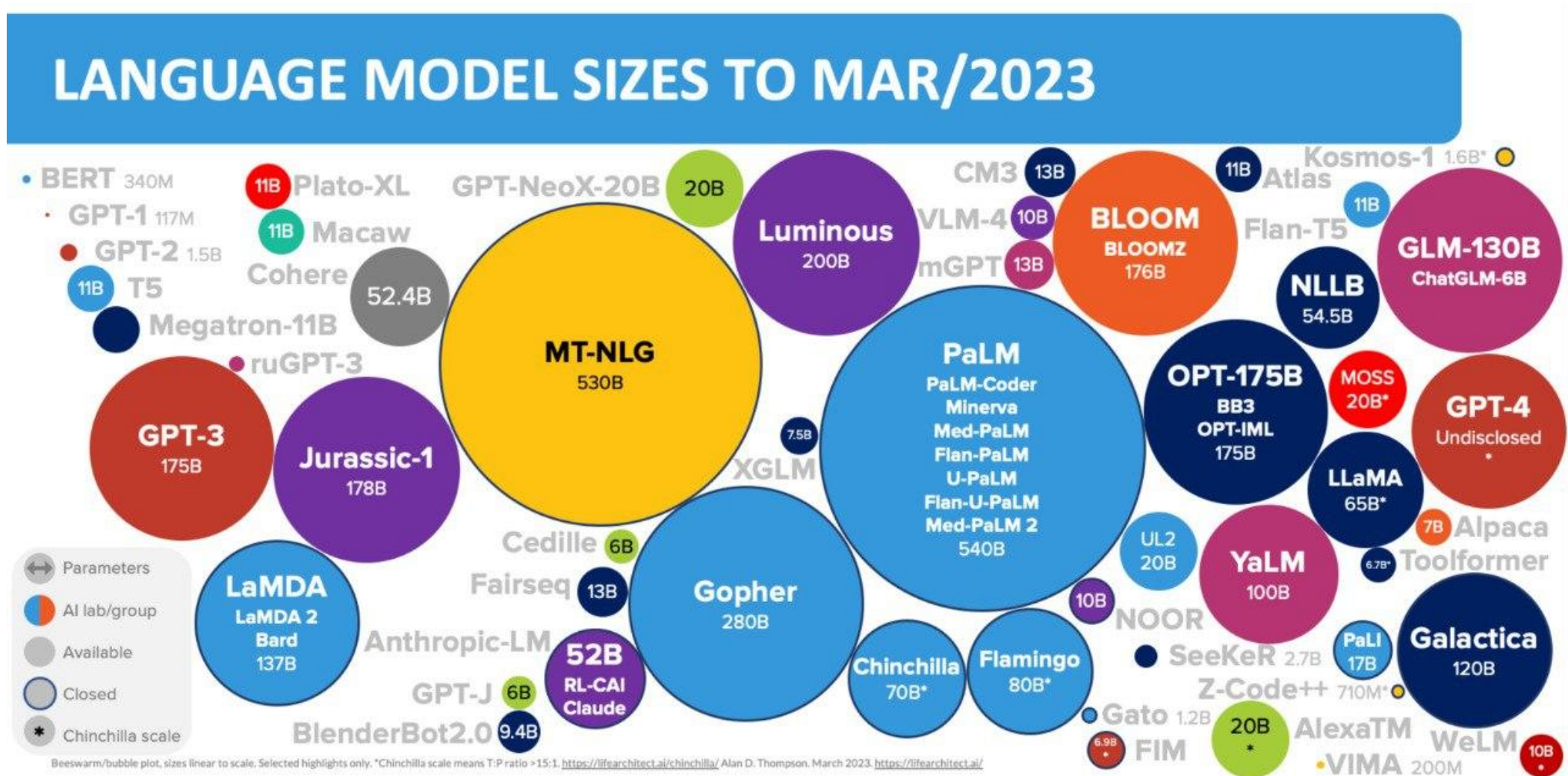




# Large Language Models



# Large Language Models



# Следующая лекция

Информационный поиск