Основы глубинного обучения

Лекция 9

Трансформеры

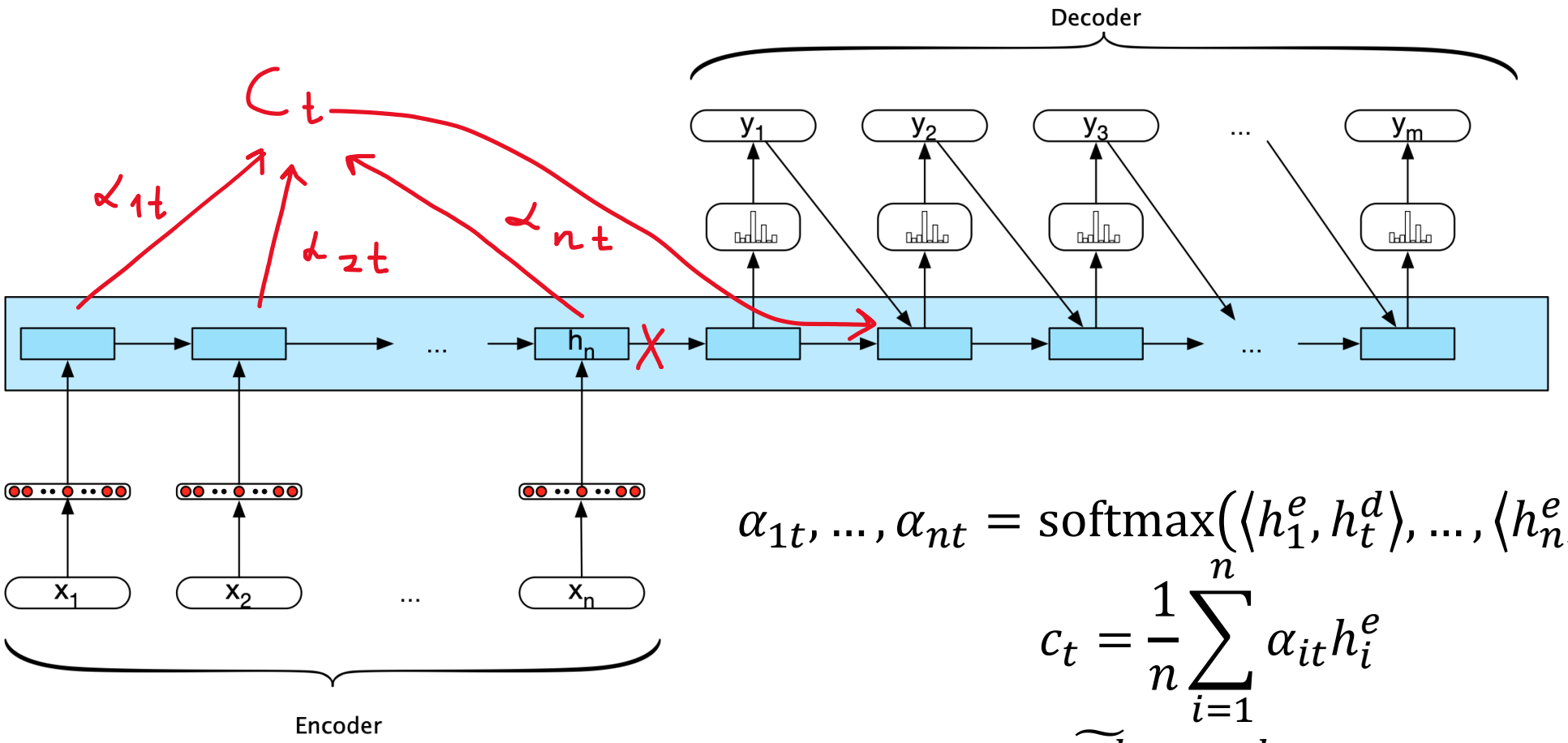
Евгений Соколов

esokolov@hse.ru

НИУ ВШЭ, 2025

Трансформер: основы self-attention

Механизм внимания



$$\alpha_{1t}, \dots, \alpha_{nt} = \text{softmax}\left(\left\langle h_1^e, h_t^d \right\rangle, \dots, \left\langle h_n^e, h_t^d \right\rangle\right)$$

$$c_t = \frac{1}{n}\sum_{i=1}^{n} \alpha_{it} h_i^e$$

$$\widetilde{h_t^d} = [h_t^d, c_t]$$

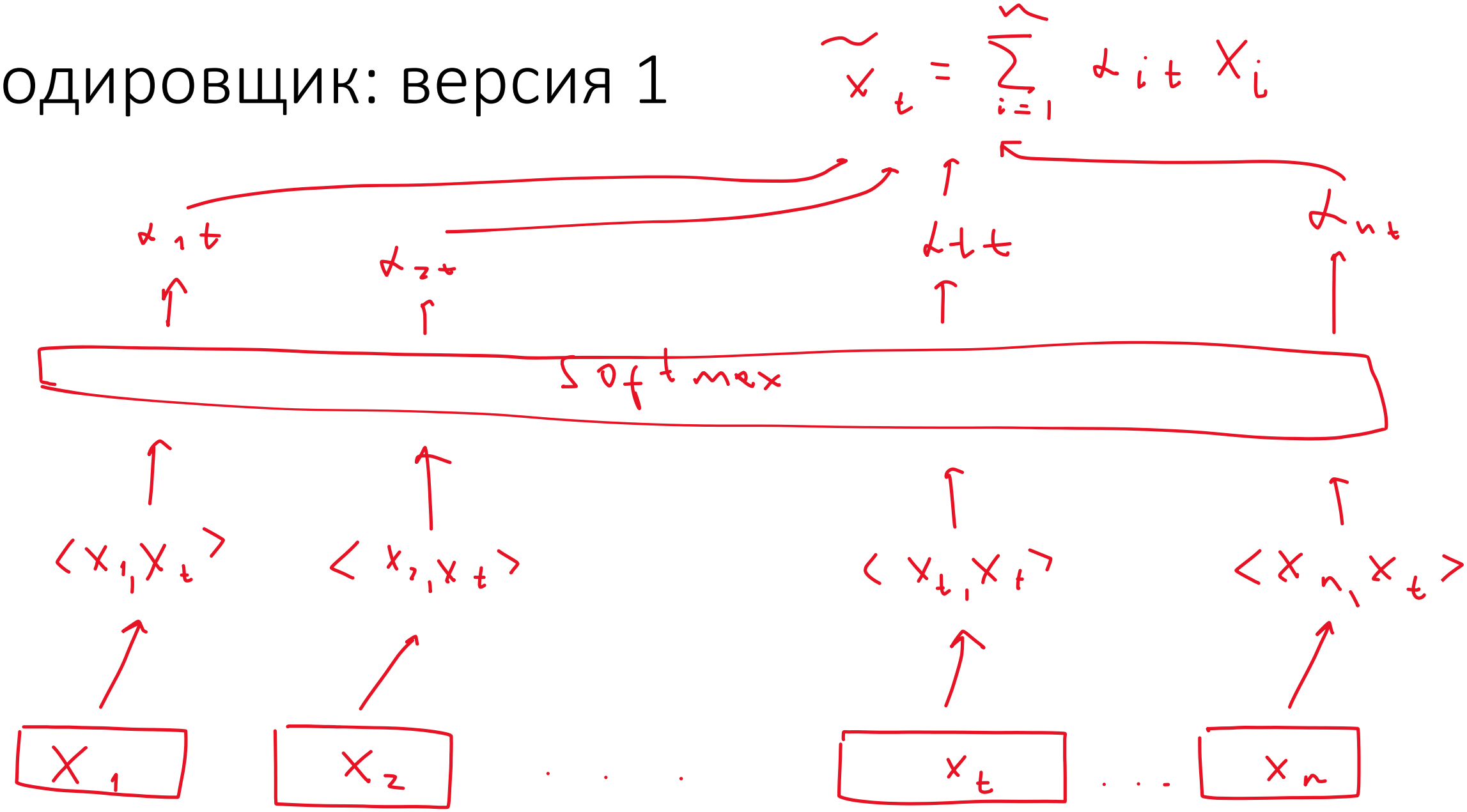
Как усилить архитектуру?

- Мы почему-то читаем входной текст слева направа
- Есть варианты с двунаправленными кодировщиками, но всё ещё мы пытаемся имитировать поведение людей
- Долой эти аналогии!

Кодировщик

- Начнём с качественного прочтения входного текста
- Попробуем обогатить каждое входное слово информацией обо всём тексте
- Назовём это «вниманием на себя» (self-attention)

Кодировщик: версия 1



Кодировщик: версия 1

- Мы подмешиваем к слову t информацию из слова i на основе сходства этих слов
- Наверное, мы хотим смешивать информацию более хитро — например, смотреть на слова той же части речи или той же части предложения

Кодировщик: версия 2

- Будем для каждого слова x_i обучать три вектора:
  - Запрос (query) $q_i = W_Q x_i$
  - Ключ (key) $k_i = W_K x_i$
  - Значение (value) $v_i = W_V x_i$
- «Важность» слова x_i для слова x_j: $\langle q_j, k_i \rangle$

Кодировщик: версия 2

- Вклад слова x_i в новое представление слова x_j:

$$\alpha_{ij} = \frac{\exp\left(\frac{\langle q_j, k_i \rangle}{\sqrt{d}}\right)}{\sum_{p=1}^{n} \exp\left(\frac{\langle q_j, k_p \rangle}{\sqrt{d}}\right)}$$

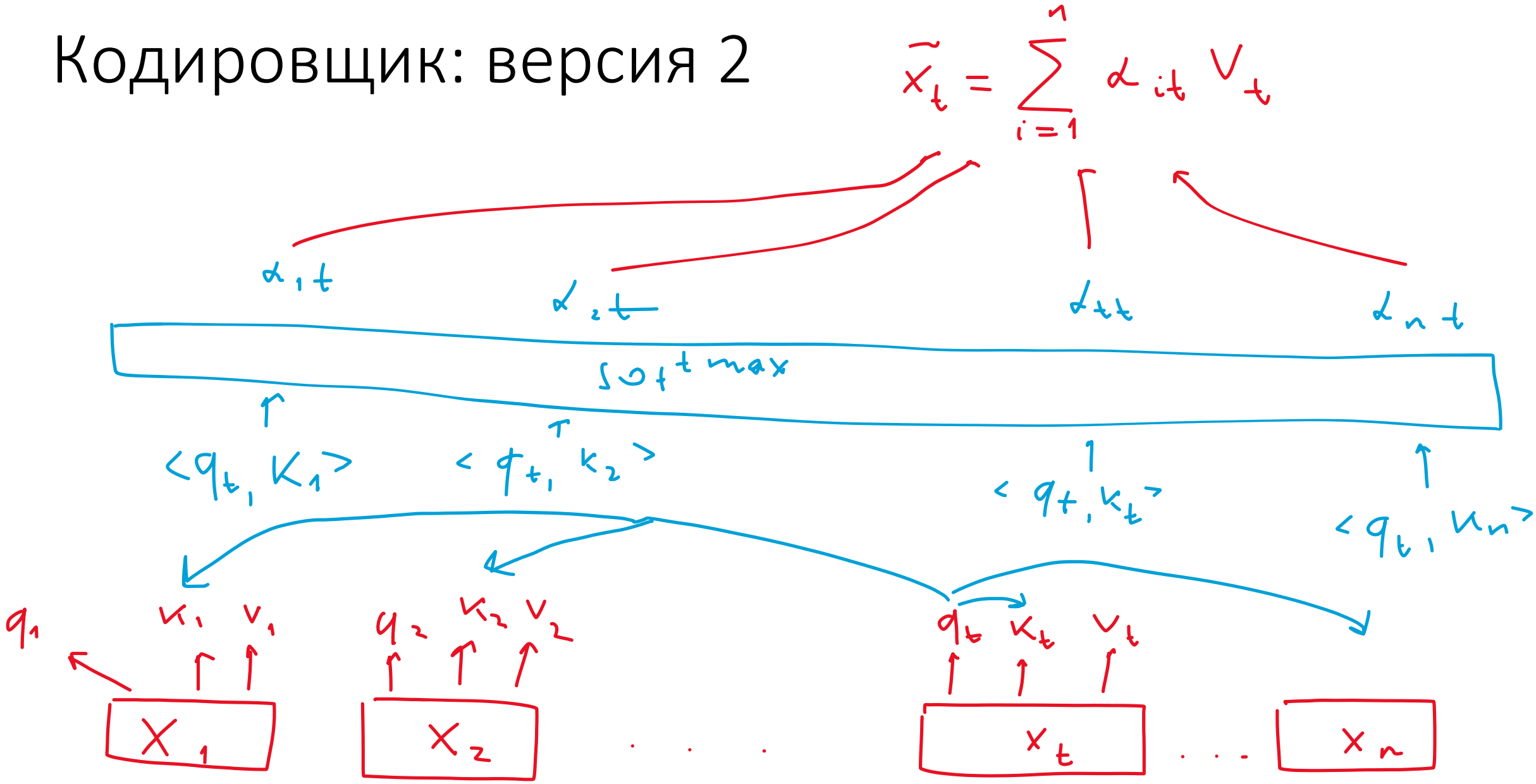
- d — размерность векторов q_j и k_i
- n — число слов во входной последовательности

Кодировщик: версия 2

Новое представление слова x_j:

$$\widetilde{x_j} = \sum_{i=1}^{n} \alpha_{ij} v_i$$

Кодировщик: версия 2

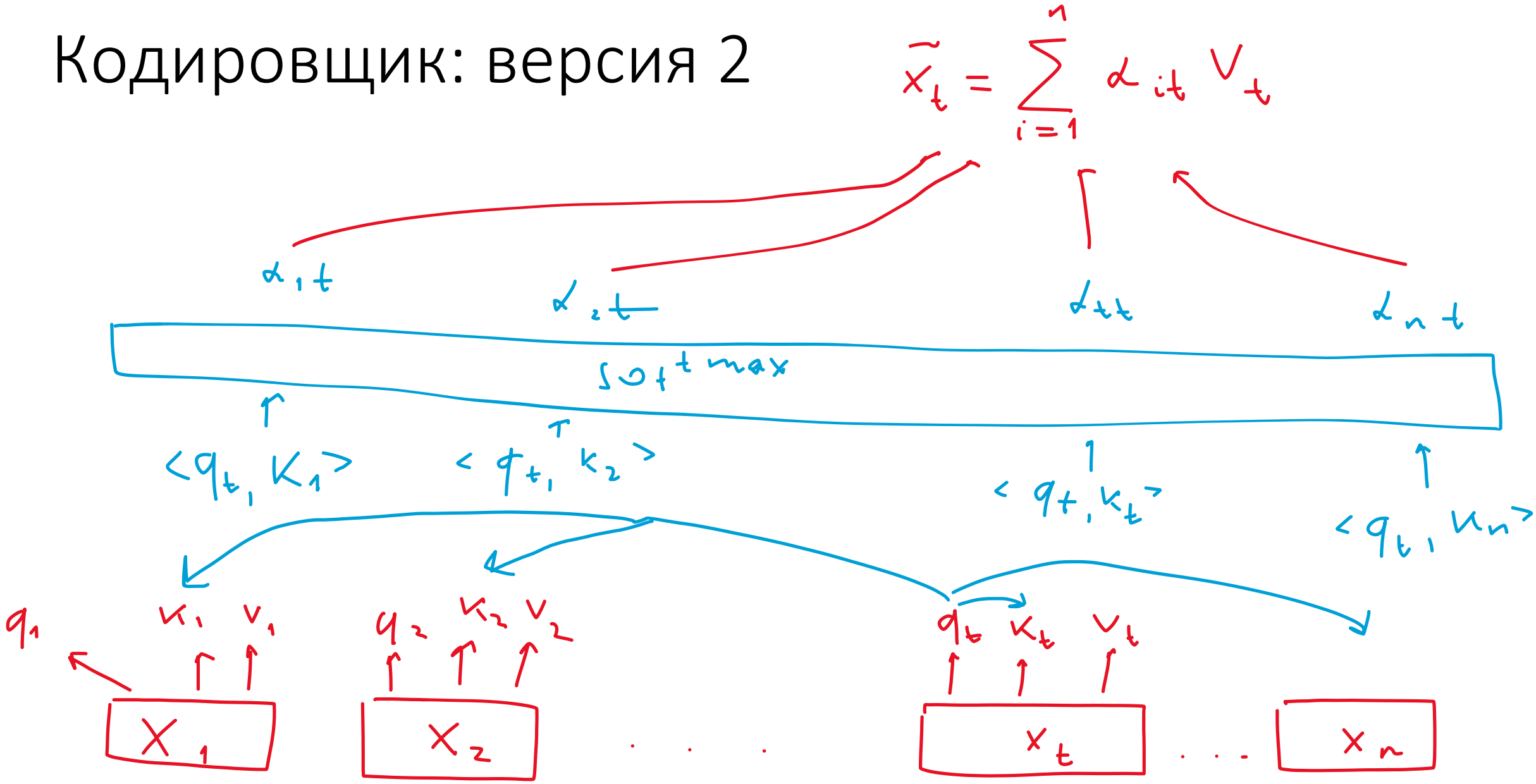


Кодировщик: версия 2

- Кодировщик задаётся тремя мини-моделями, вычисляющими по вектору слова векторы запроса, ключа и значения
- Каждая мини-модель — линейный слой

Трансформер: добавляем параметры

Кодировщик: версия 2

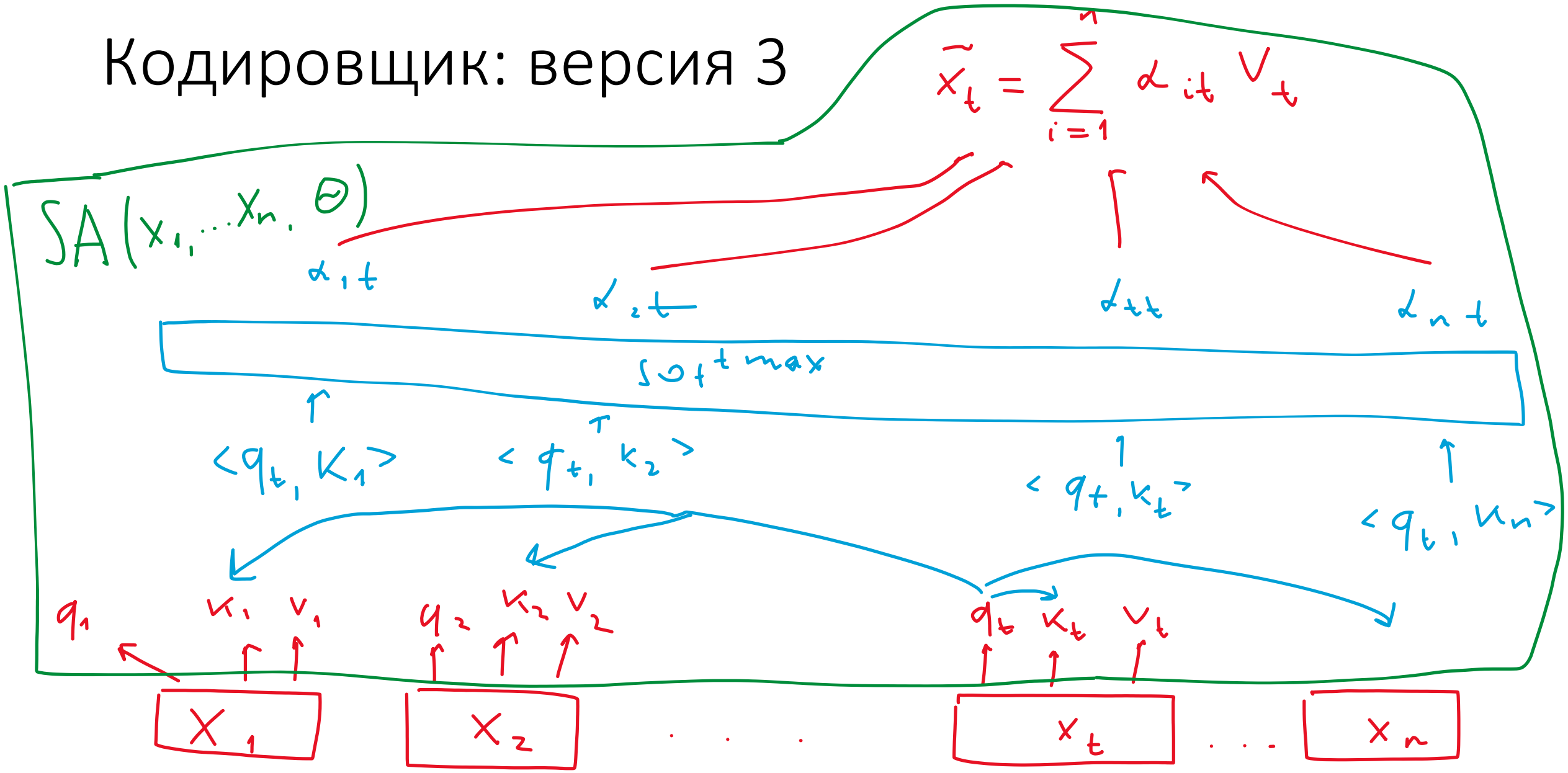


Кодировщик: версия 2

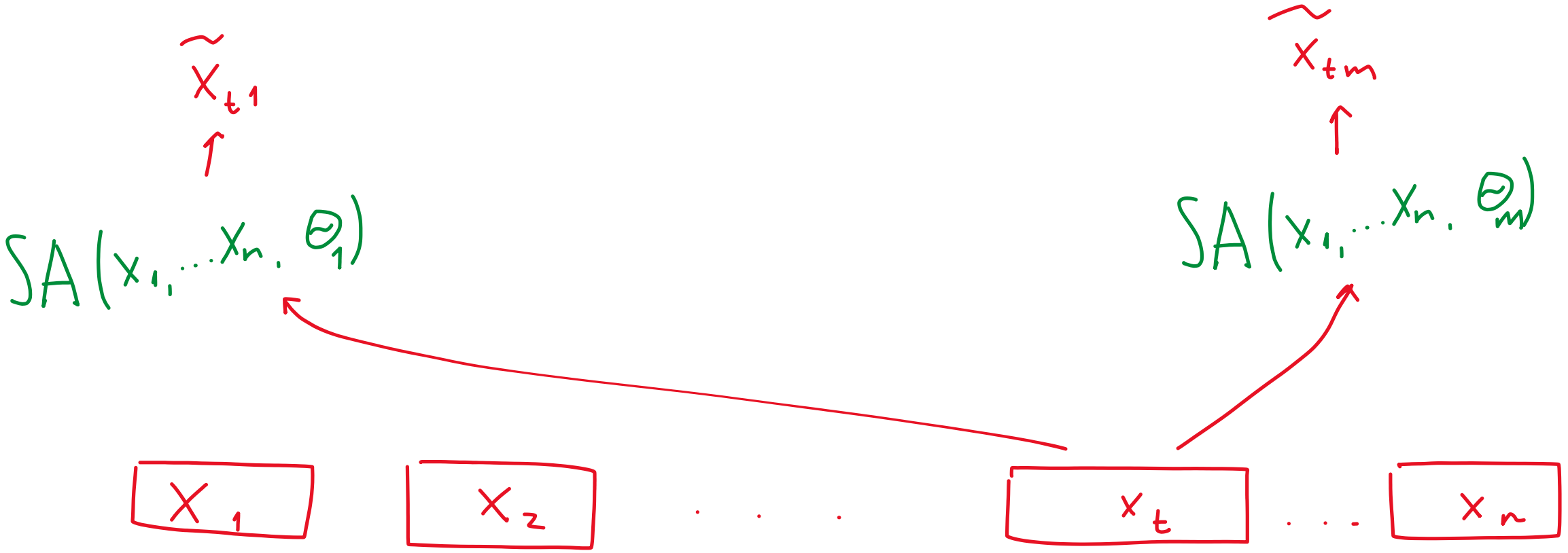
- Кодировщик задаётся тремя мини-моделями, вычисляющими по вектору слова векторы запроса, ключа и значения
- Каждая мини-модель — линейный слой

- Параметров мало
- Мы как будто добавляем в слово t информацию по одному критерию — но наверняка нас интересует много аспектов

Кодировщик: версия 3



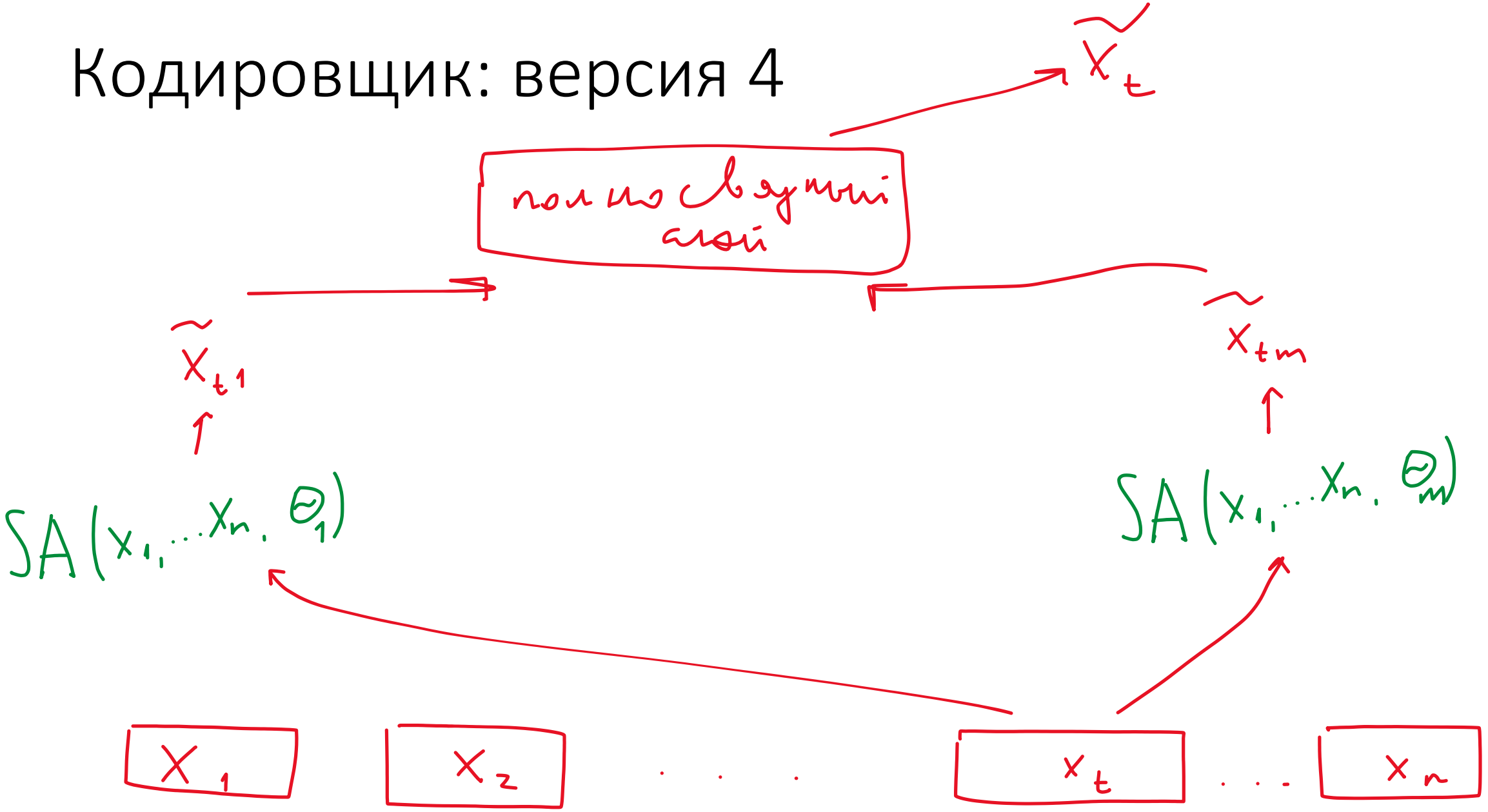
Кодировщик: версия 3



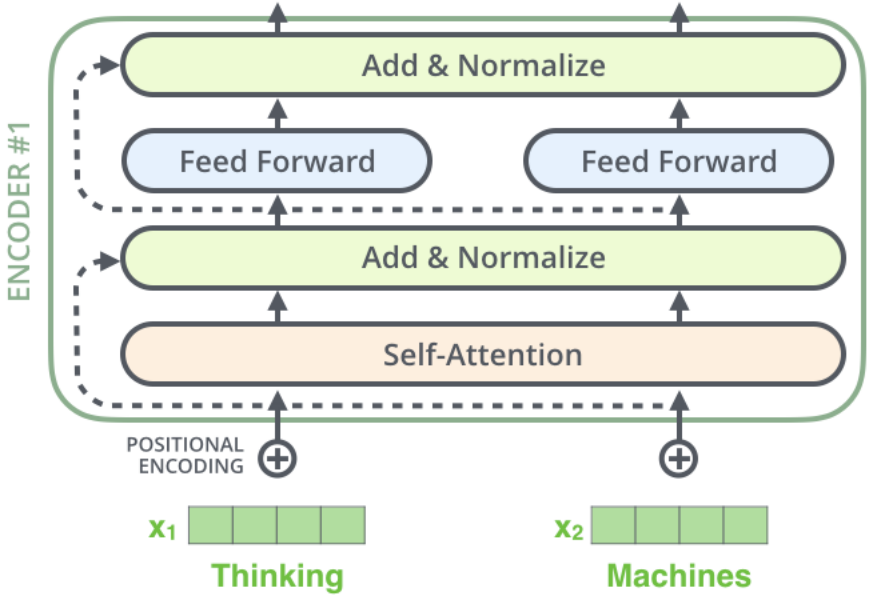
Кодировщик: версия 3

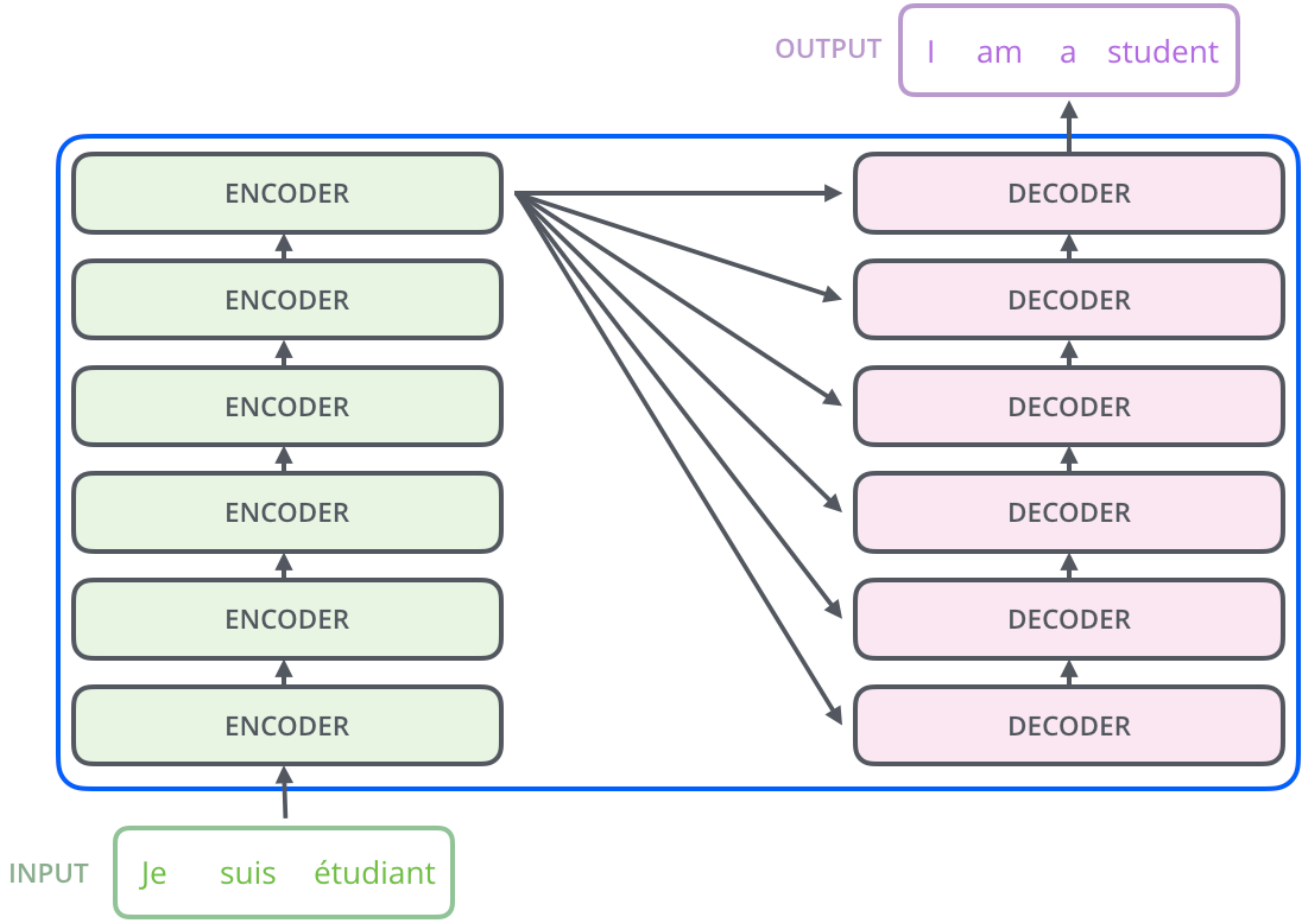
- Мы теперь к каждому слову примешиваем разные аспекты информации из других слов
- Получаем много версий вектора для слова t

Кодировщик: версия 4



Кодировщик: версия 5



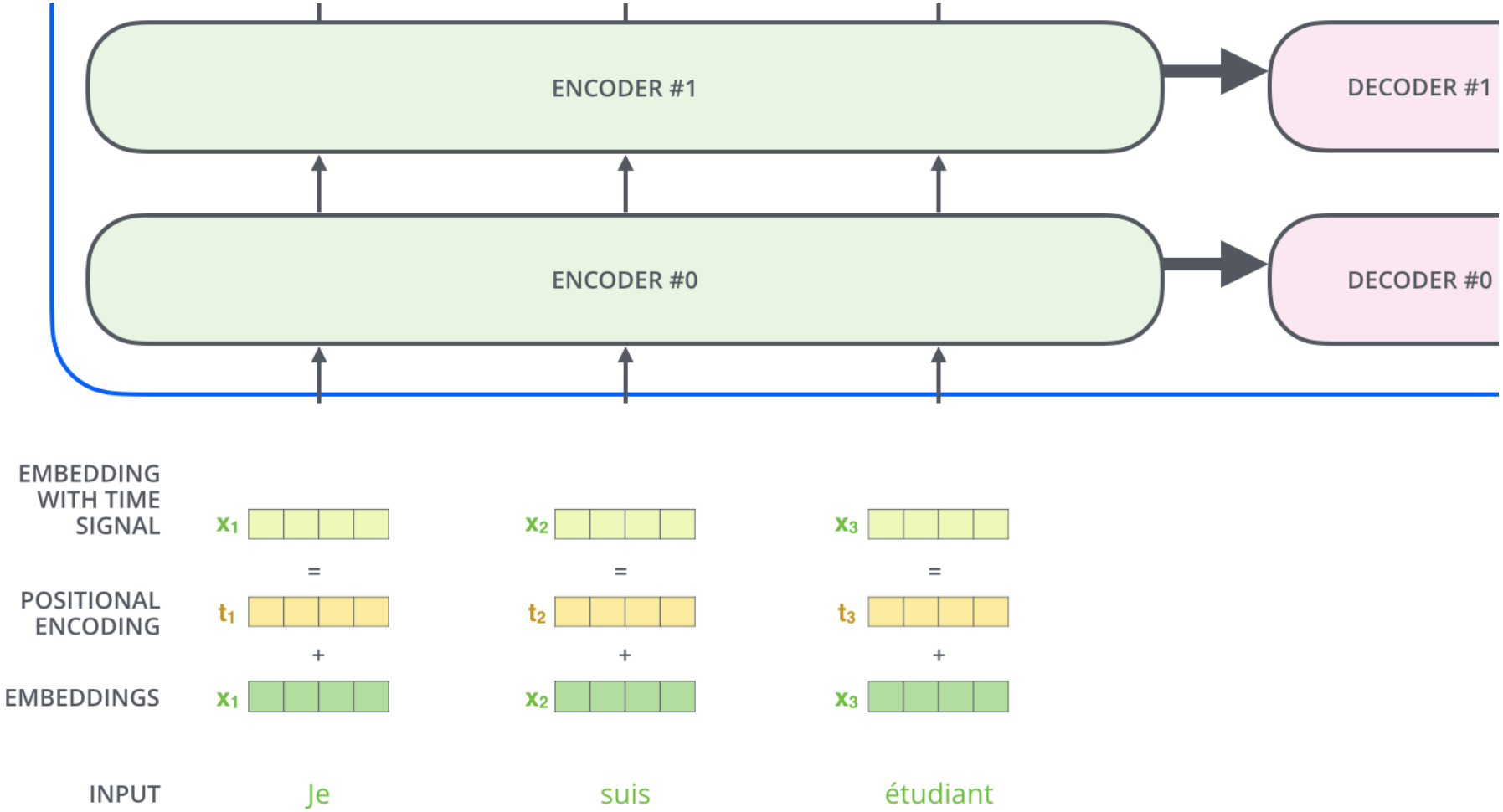


Трансформер: учитываем позиции слов

Positional encoding

- Сейчас у модели нет информации о порядке слов
- Попробуем её добавить путём прибавки чего-нибудь ко входным векторам

Positional encoding

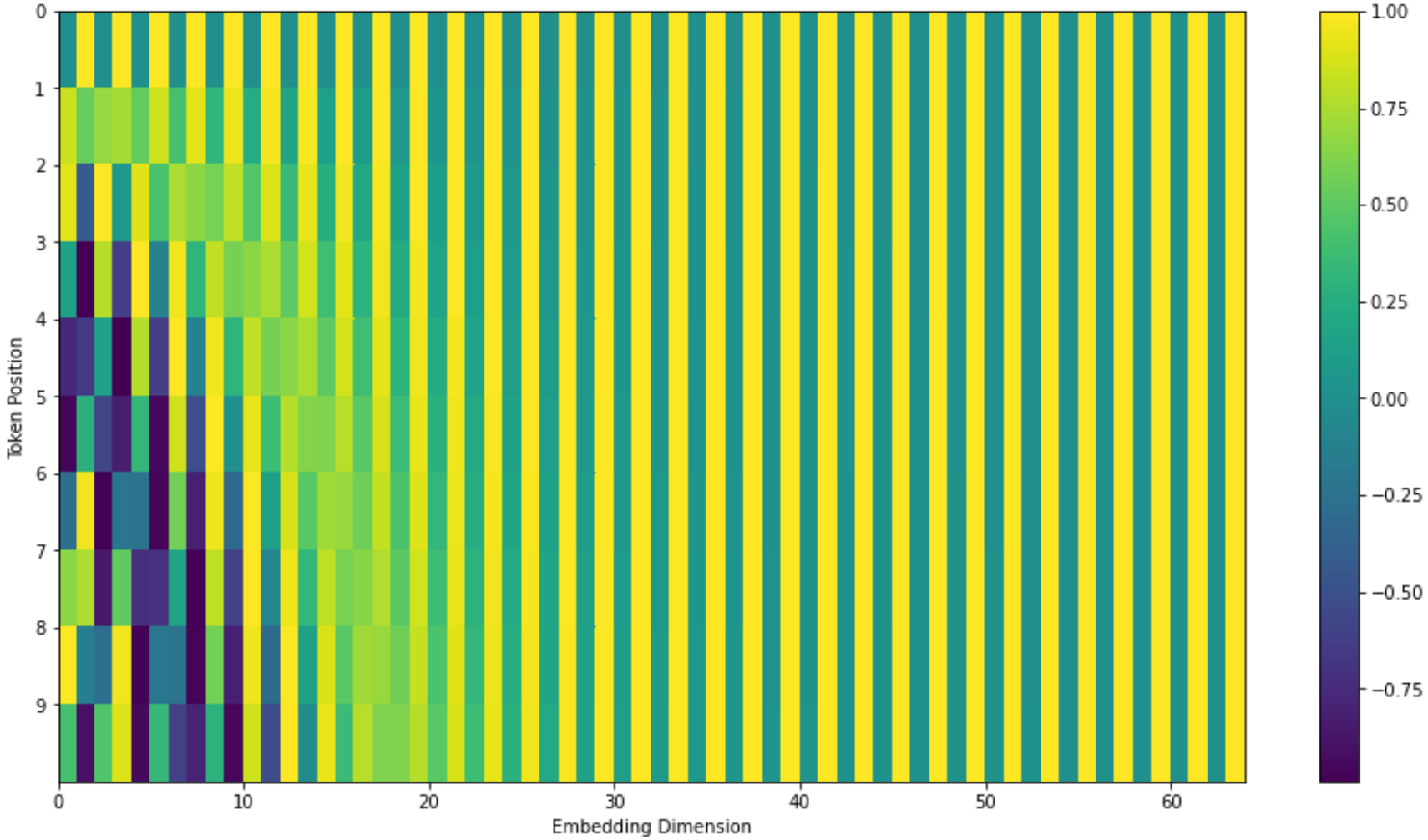


Positional encoding

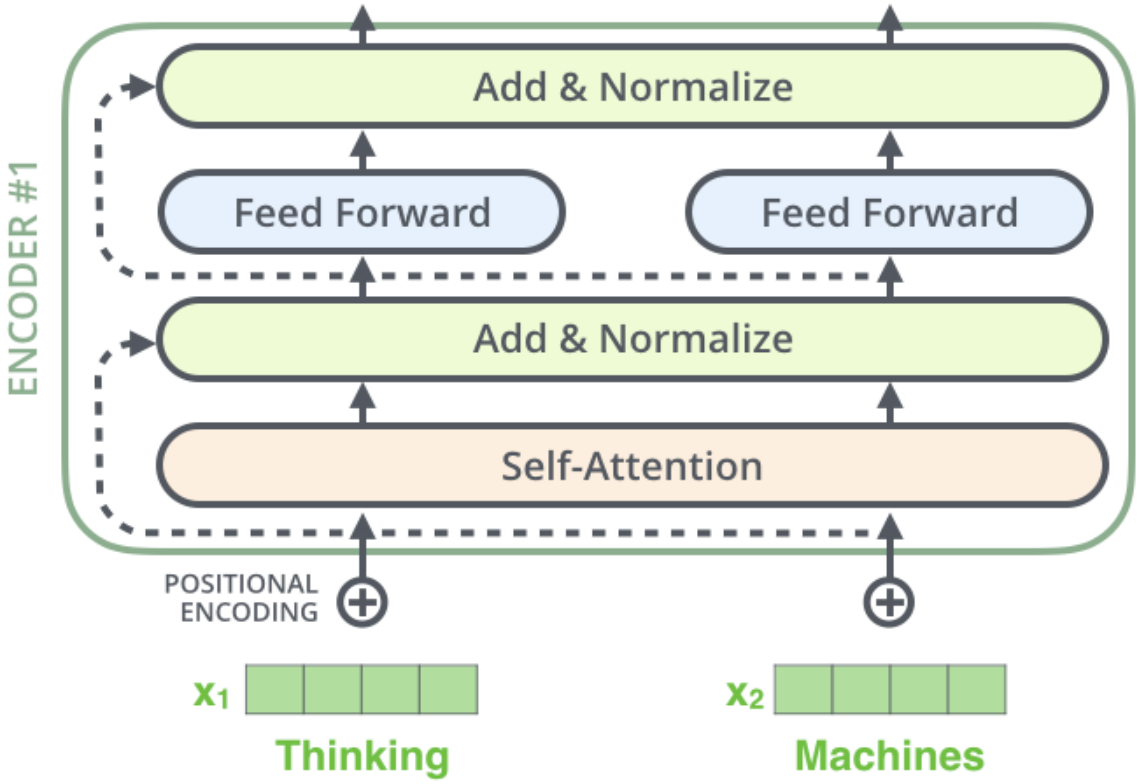
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Или можно просто обучить эти векторы!

Positional encoding

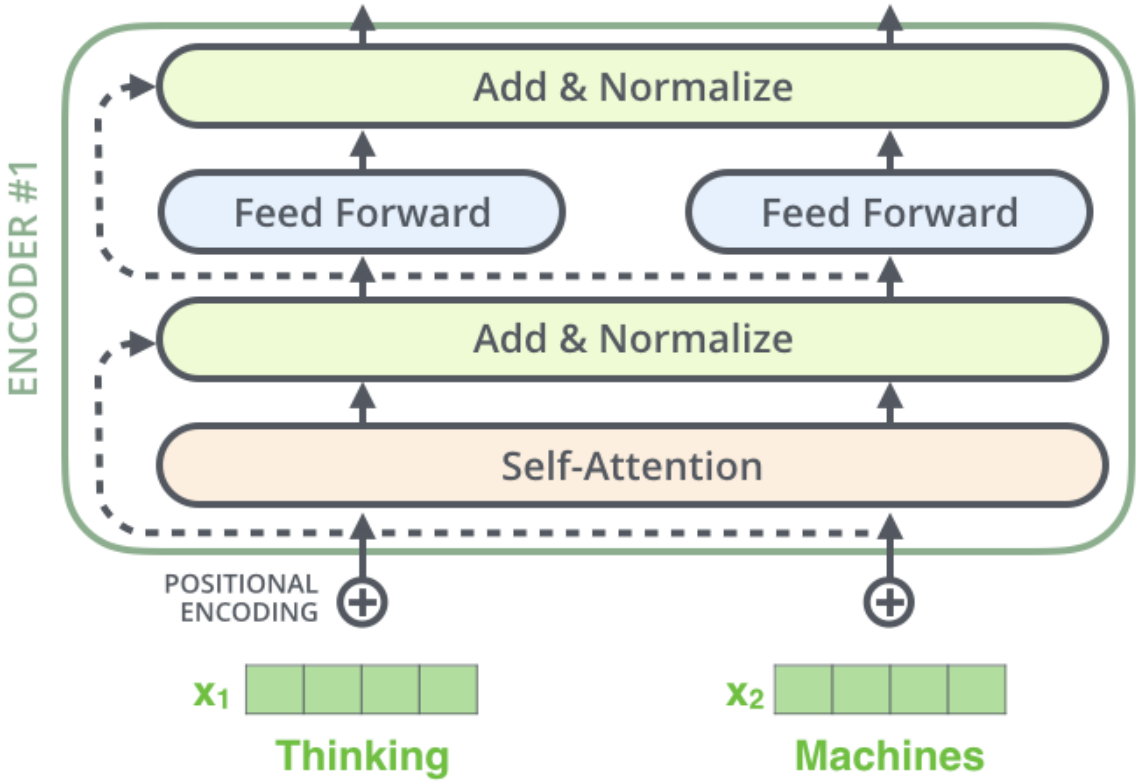


Кодировщик: версия 6

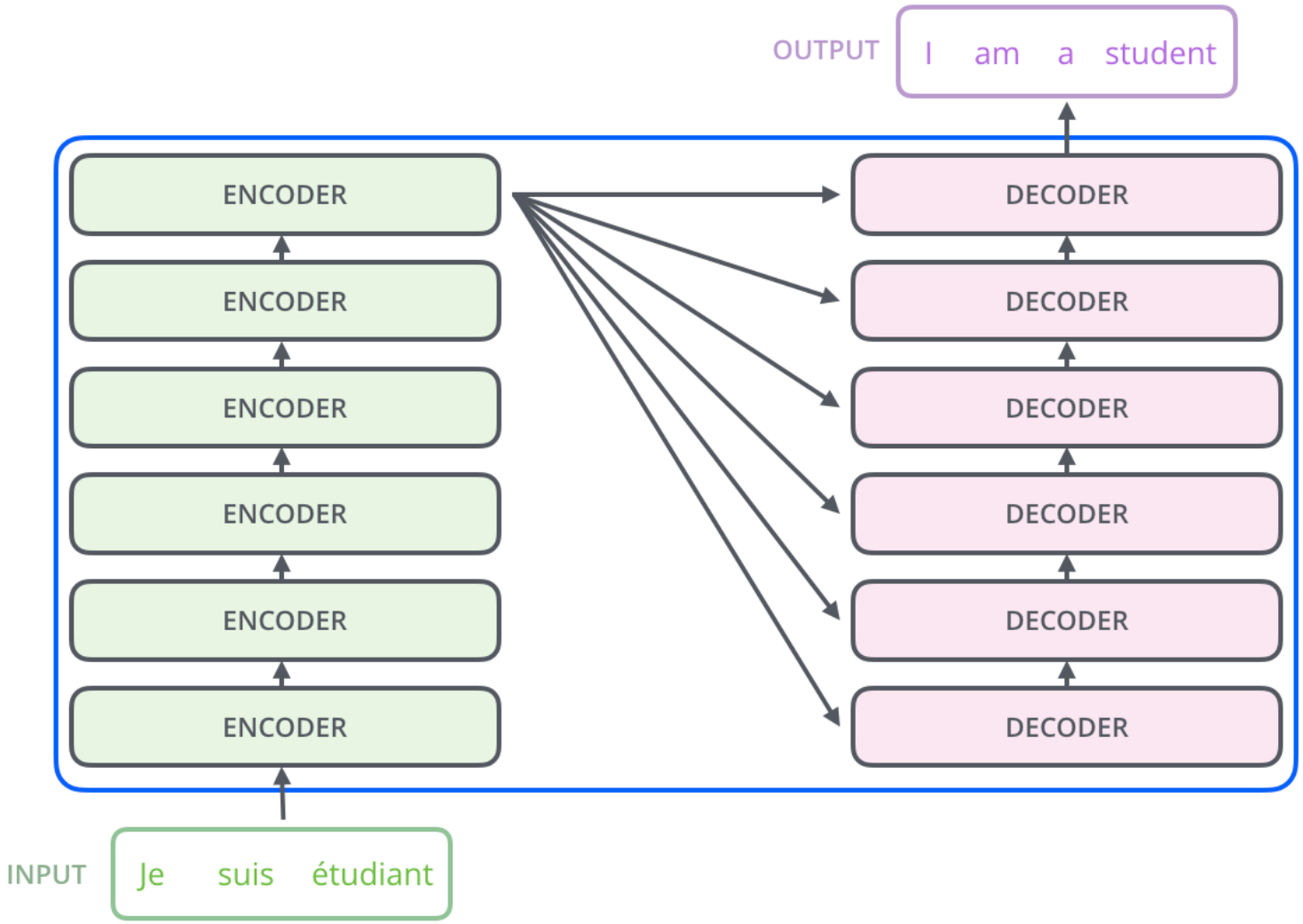


Трансформер: декодировщик

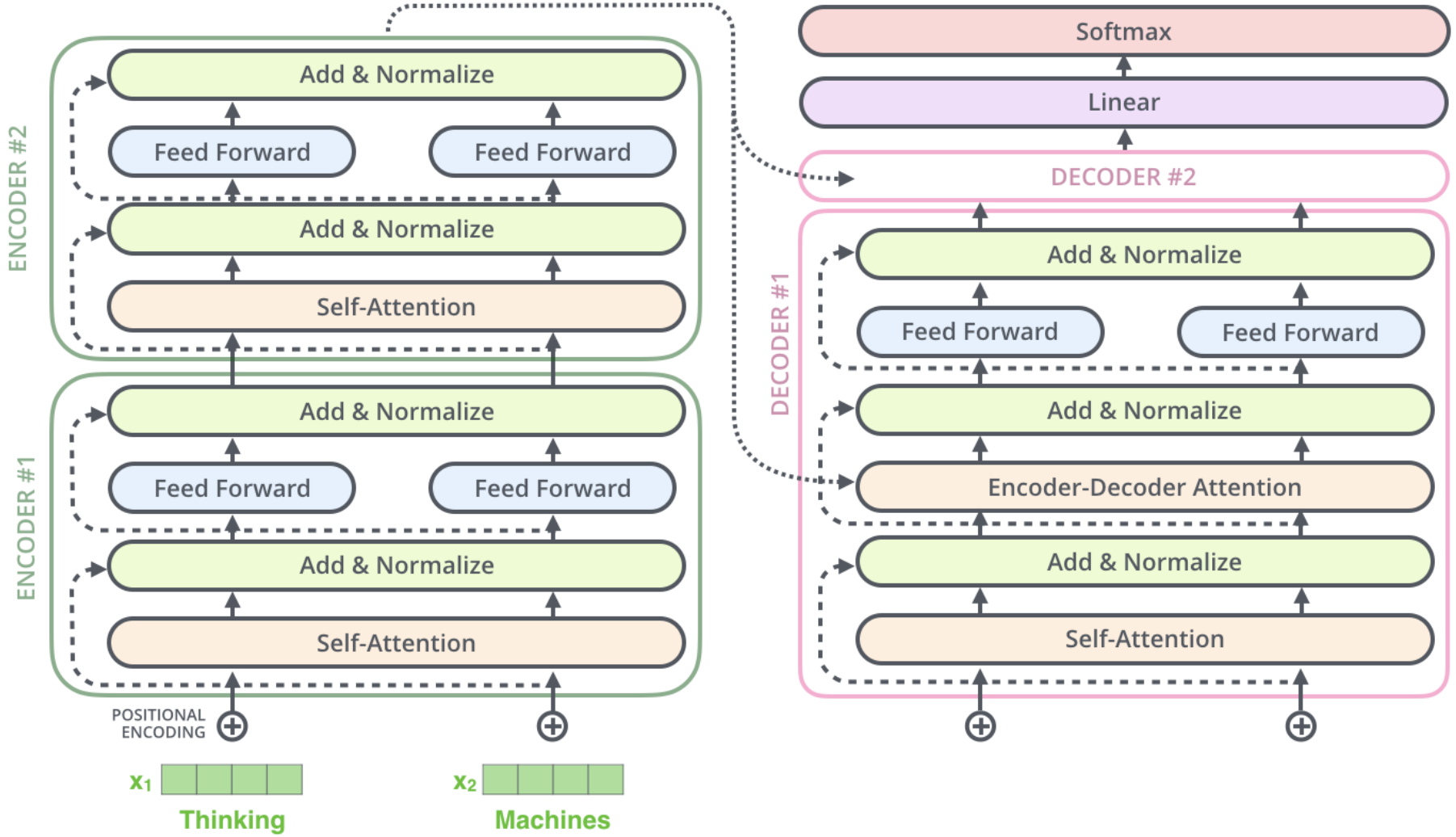
Кодировщик: версия 6



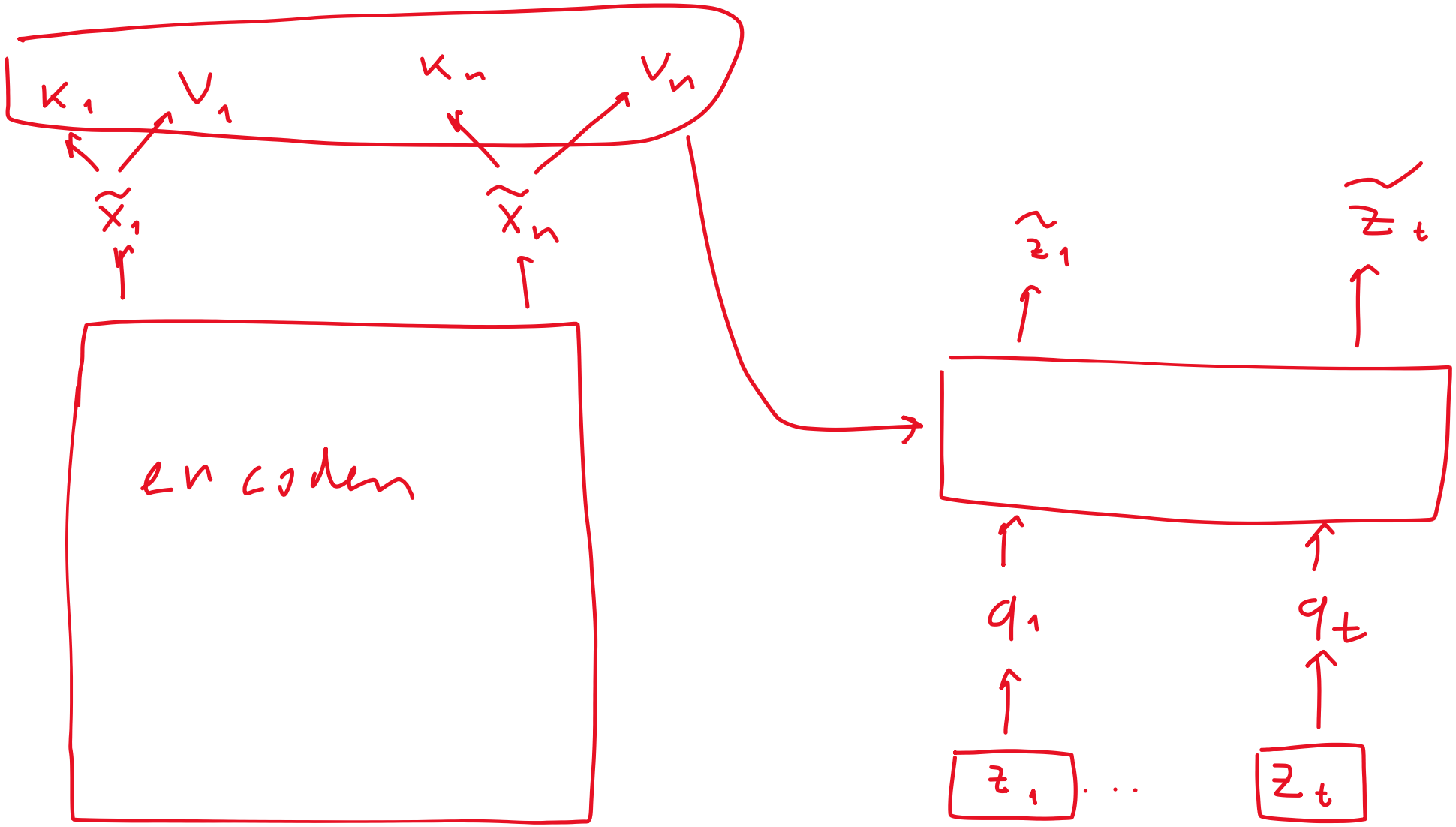
Трансформер



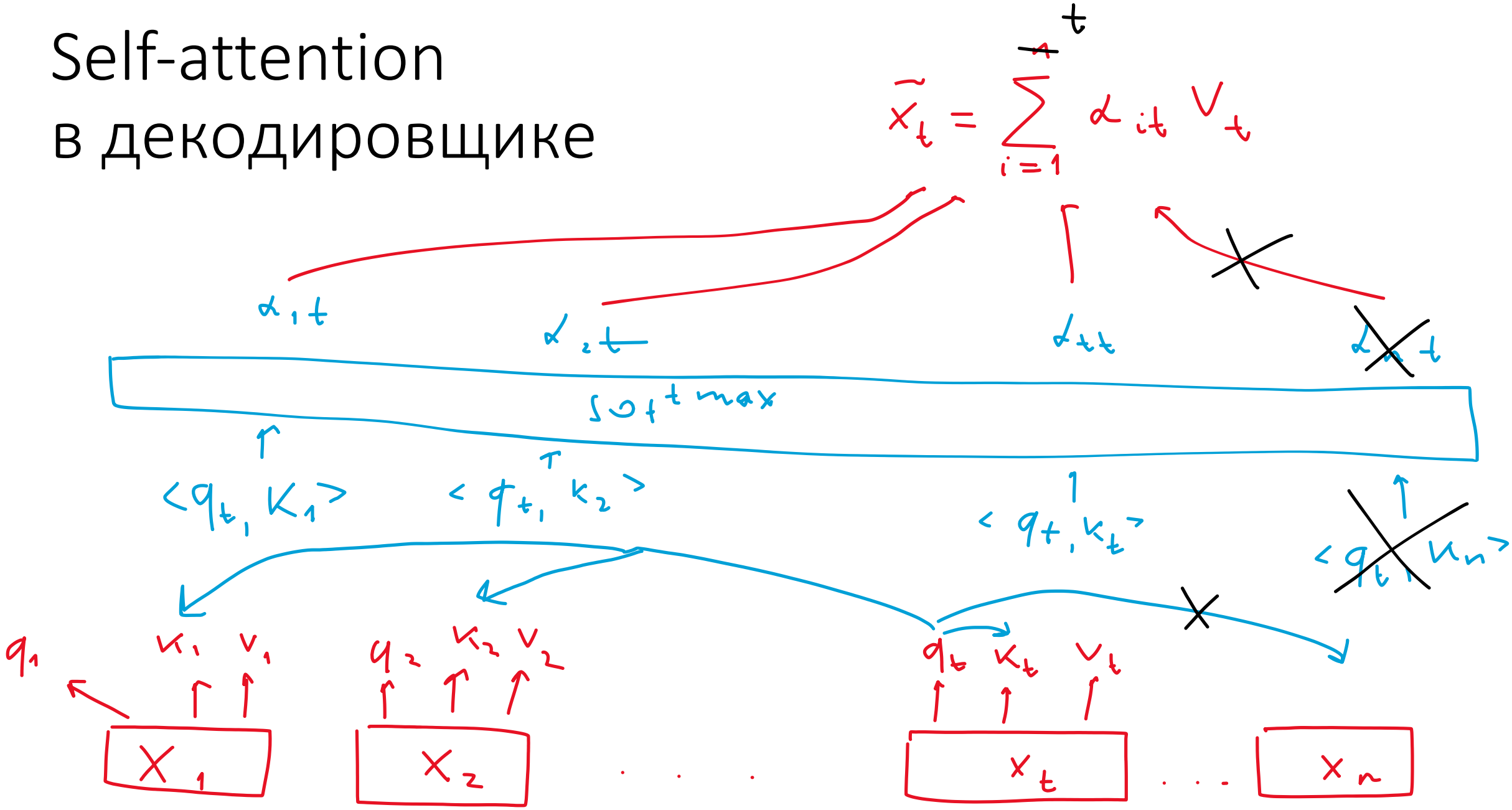
Деодировщик в трансформере



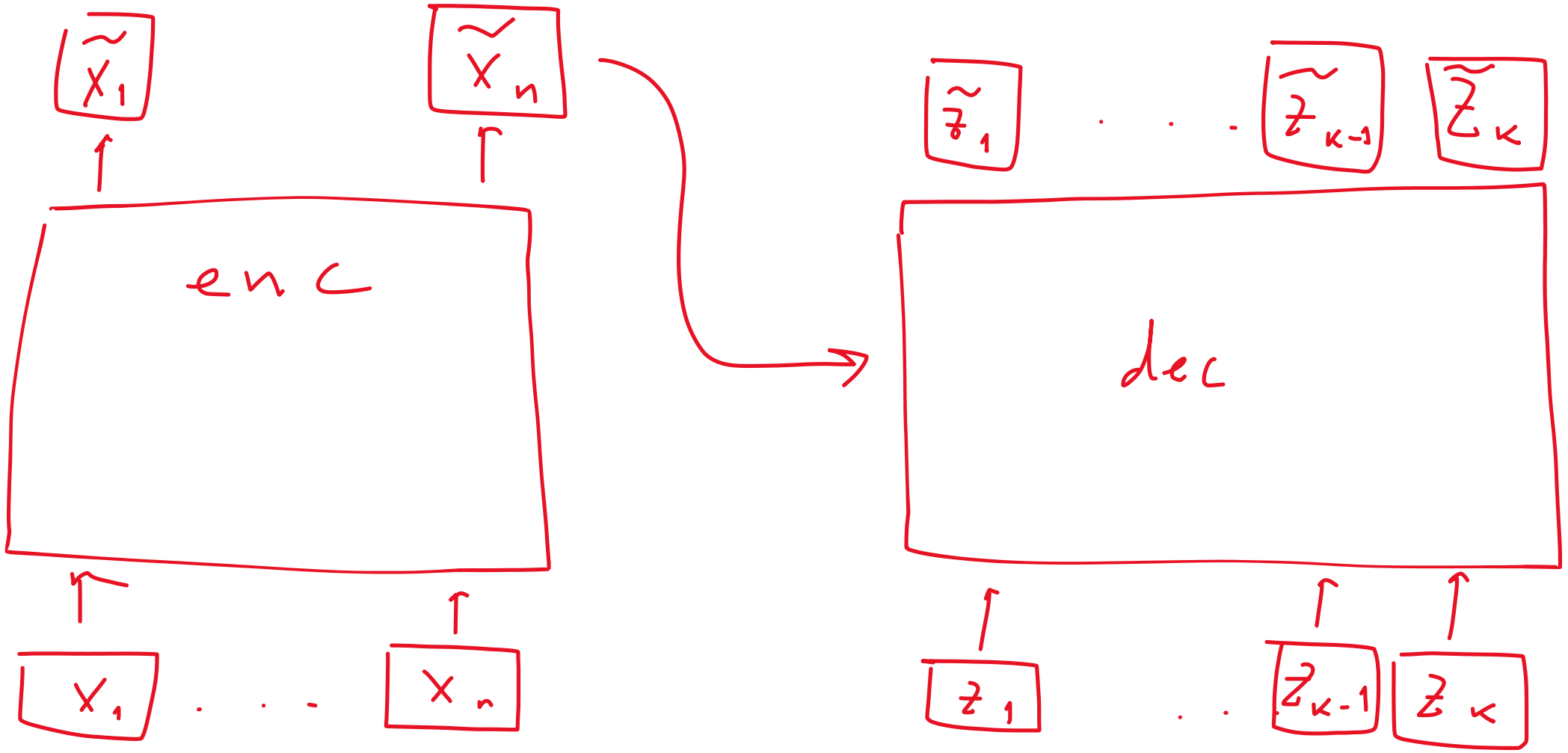
Encoder-Decoder Attention



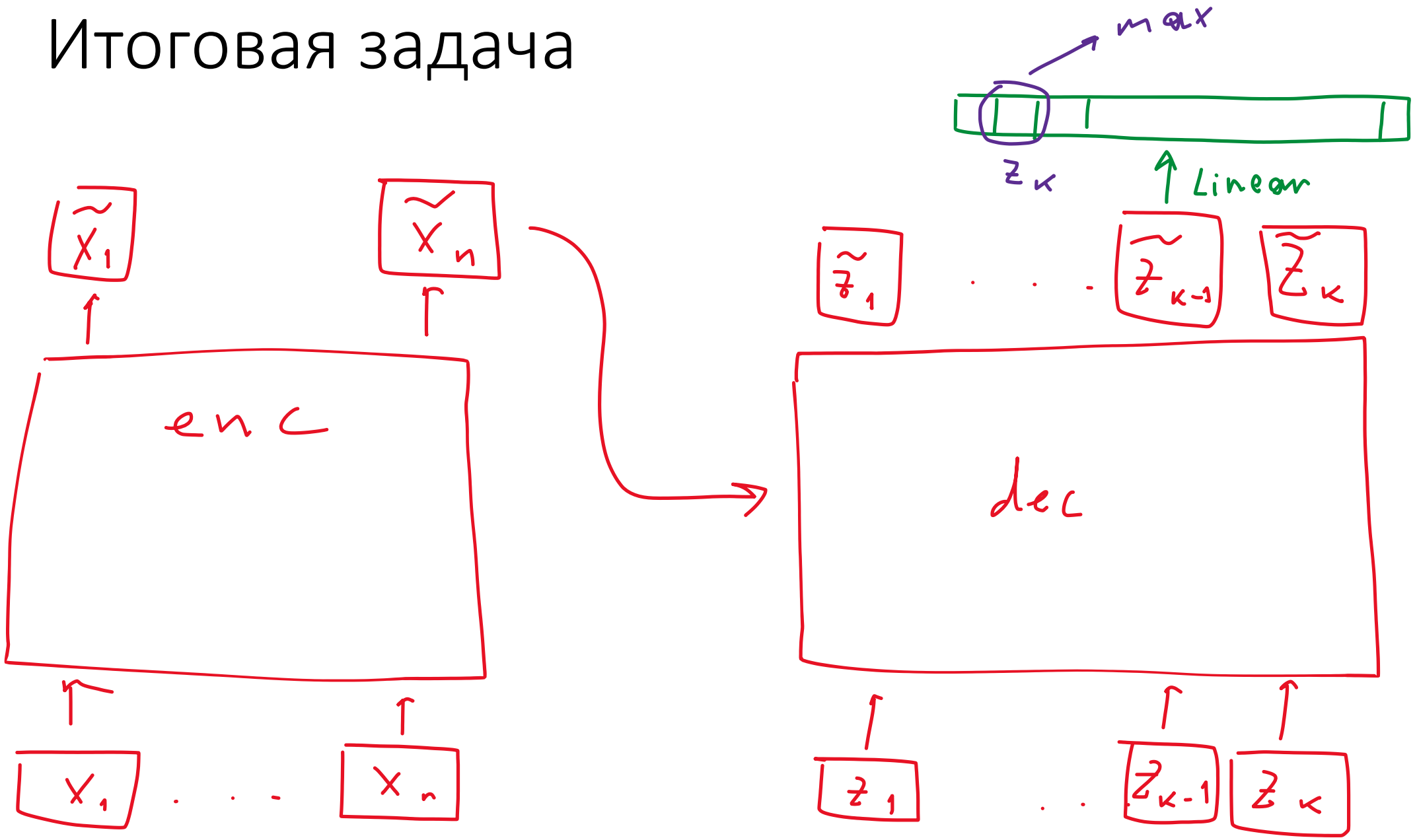
Self-attention в декодировщике



Итоговая задача

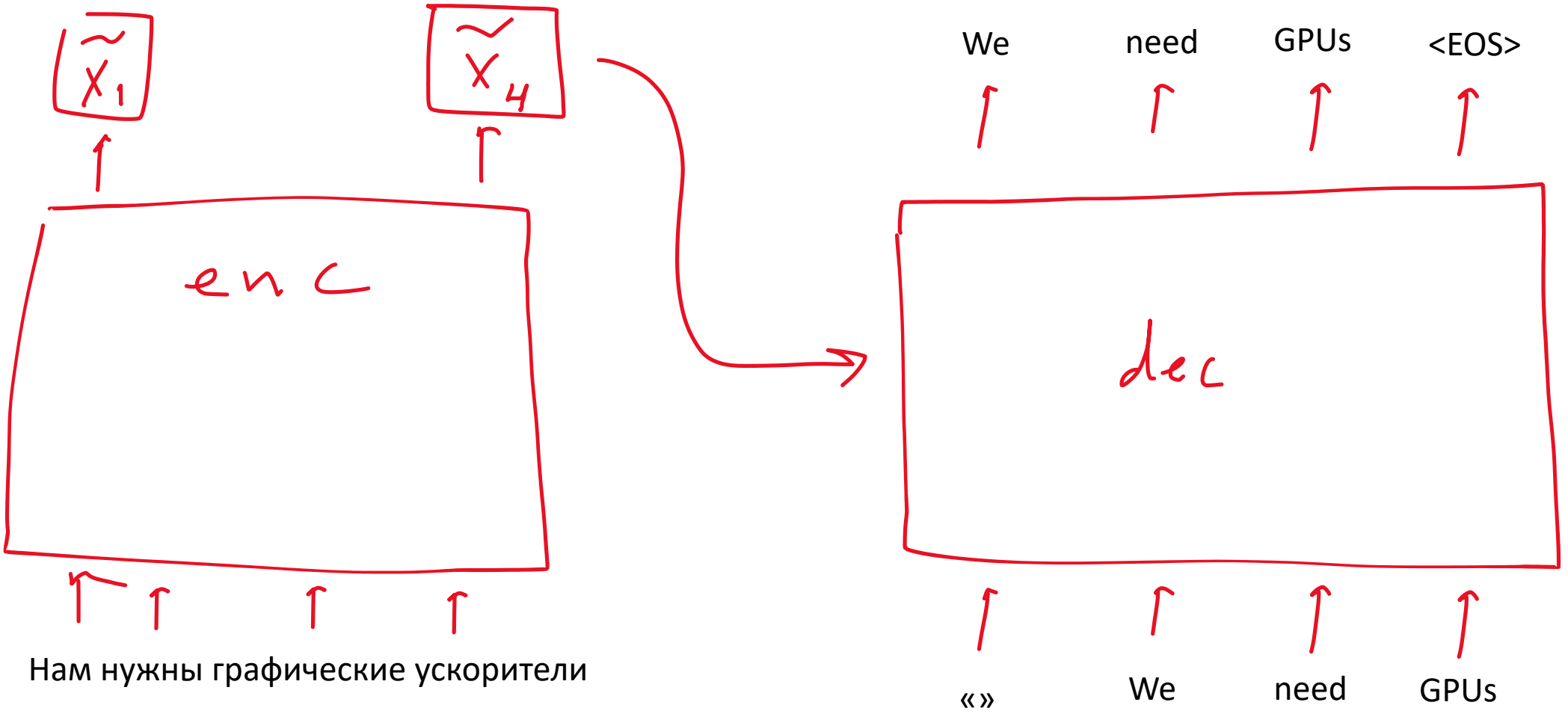


Итоговая задача



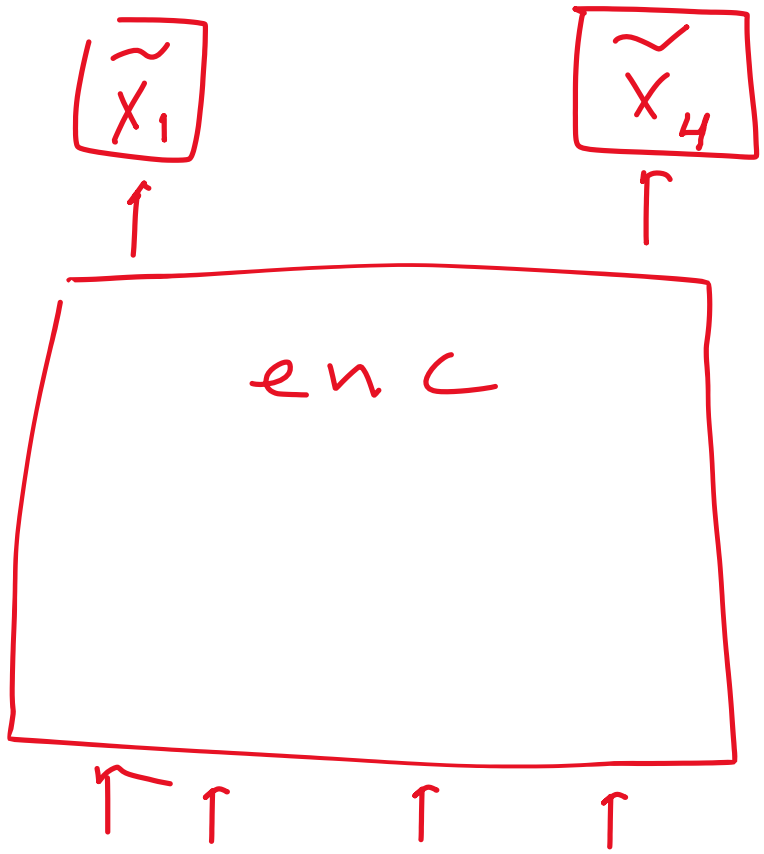
Трансформер: режимы работы

Режим 1: seq2seq



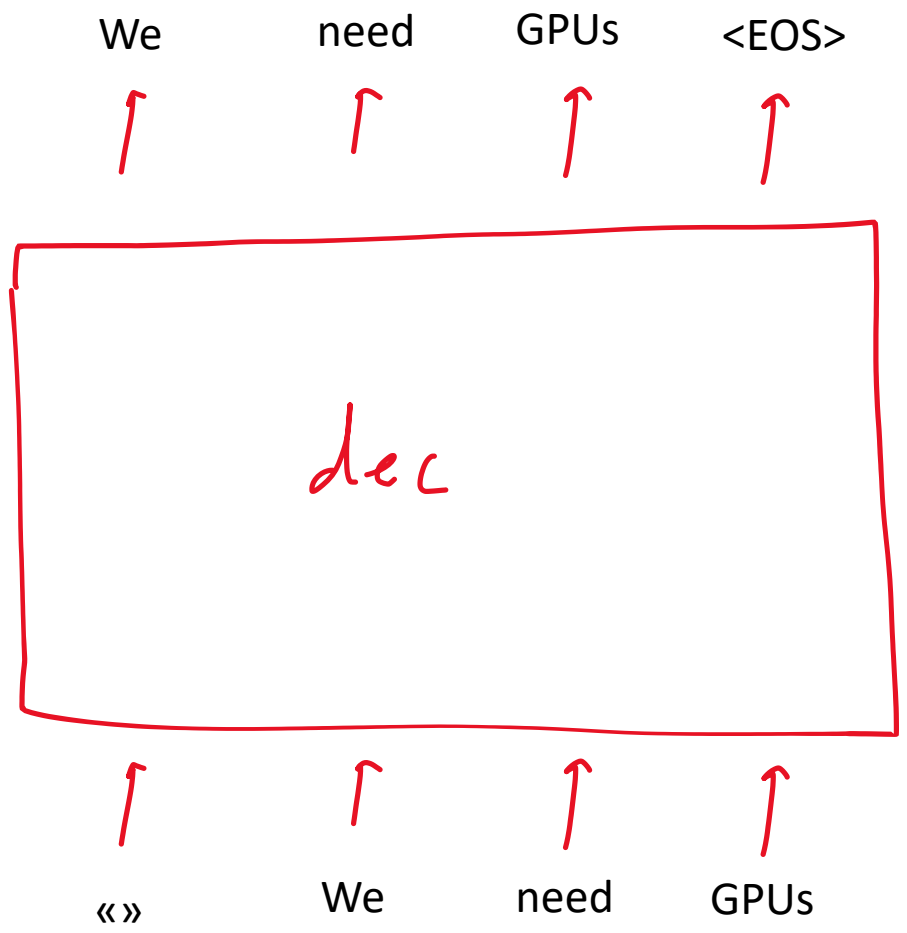
Нам нужны графические ускорители

Режим 2: только кодировщик



Нам нужны графические ускорители

Режим 3: только декодировщик



Обучение трансформеров с нуля

- Только кодировщик:
  - Классификация текстов
  - Сравнение текстов
  - Классификация слов в тексте
  - Отбор ответов на вопрос
  - …
- Только декодировщик:
  - Генерация текста
  - Генерация ответов на задания

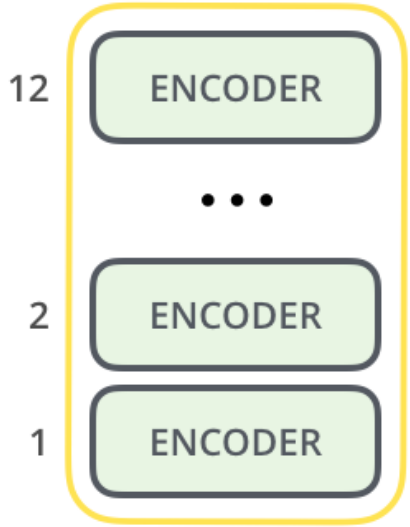
Обучение трансформеров с нуля

- Требует огромного объёма данных
- Требует колоссальных вычислительных мощностей

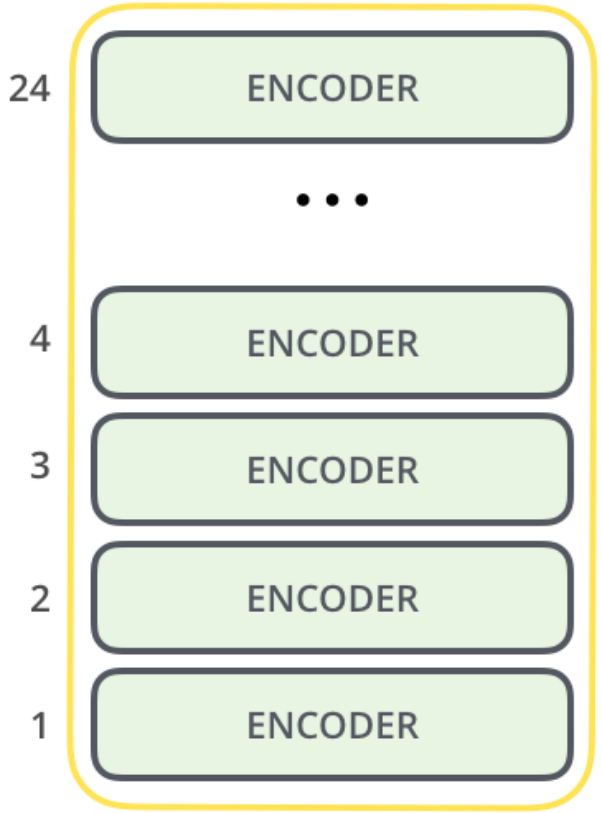
- Можно ли заранее обучить большую модель так, чтобы она легко донастраивалась на новые задачи?

BERT: предобучение

BERT: архитектура

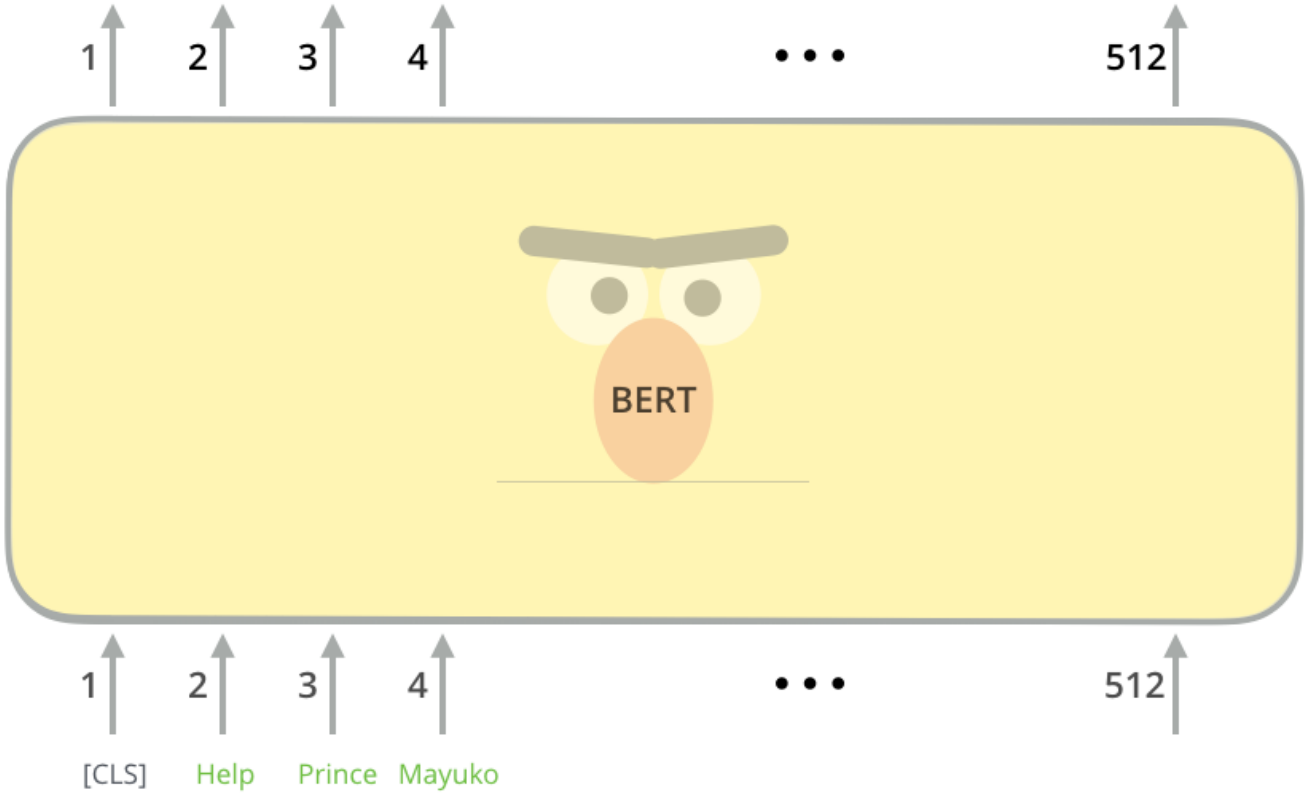


BERT$_{BASE}$

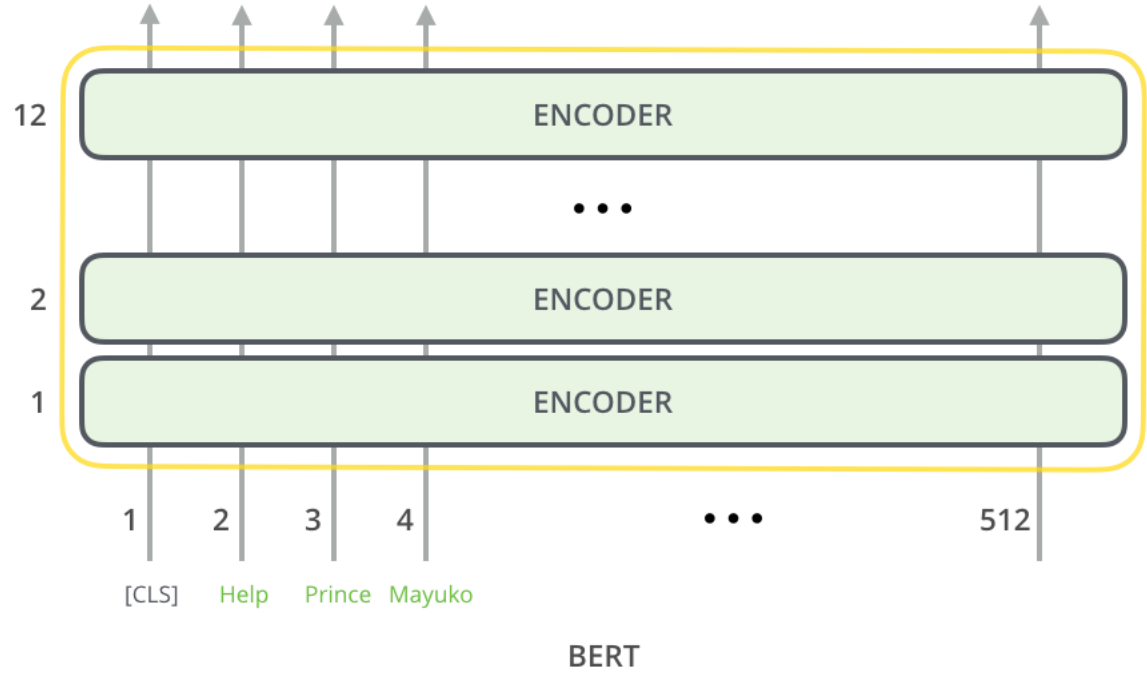


BERT$_{LARGE}$

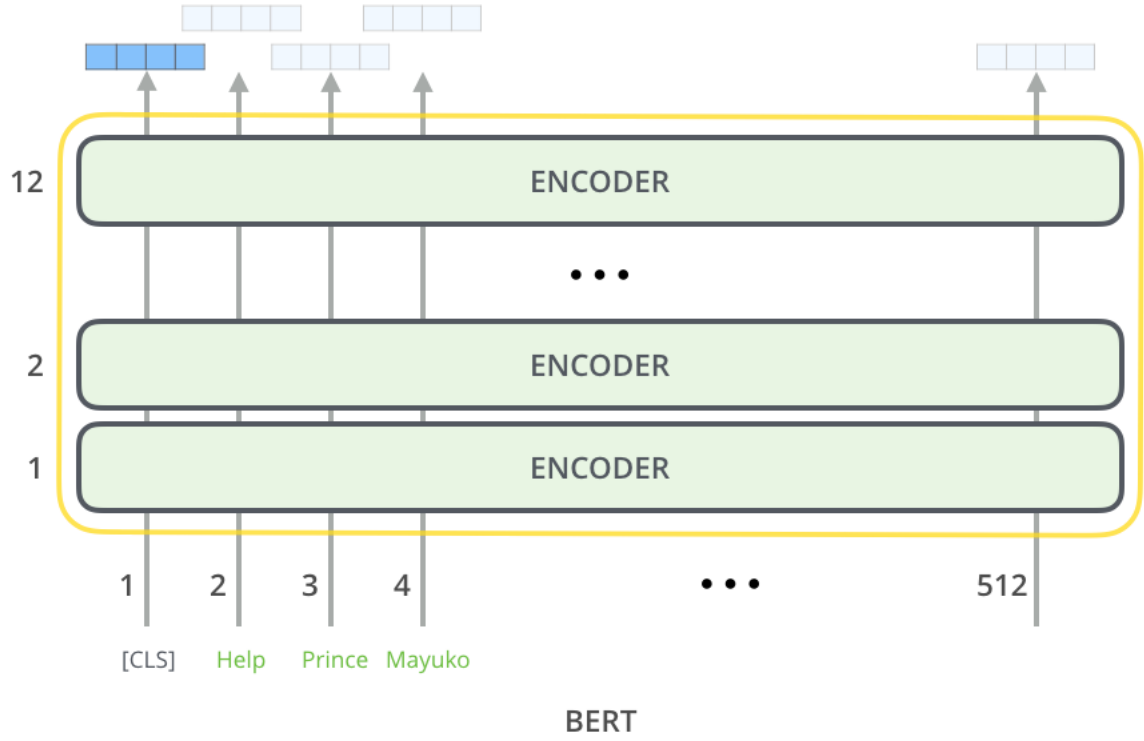
BERT: архитектура



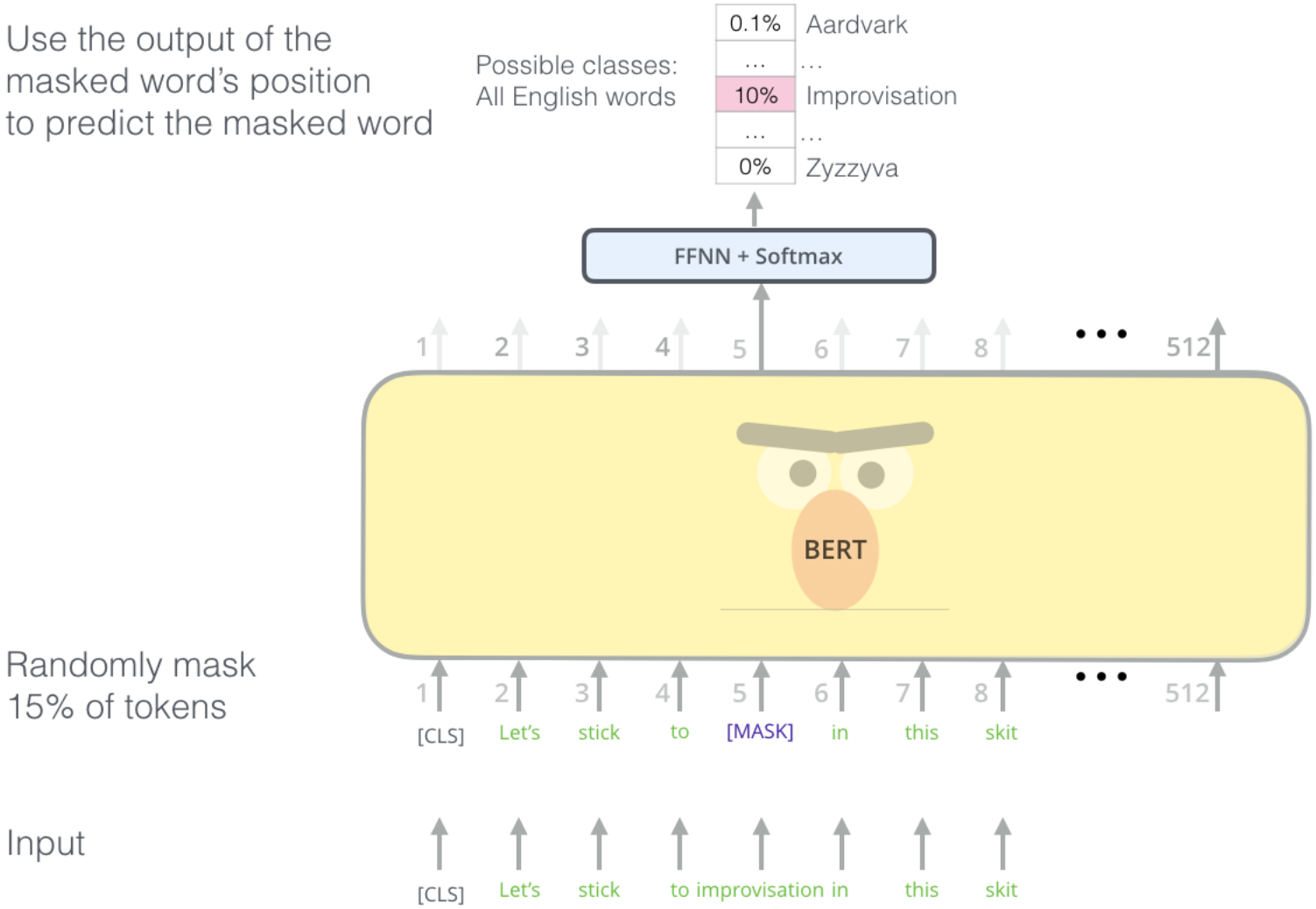
BERT: архитектура



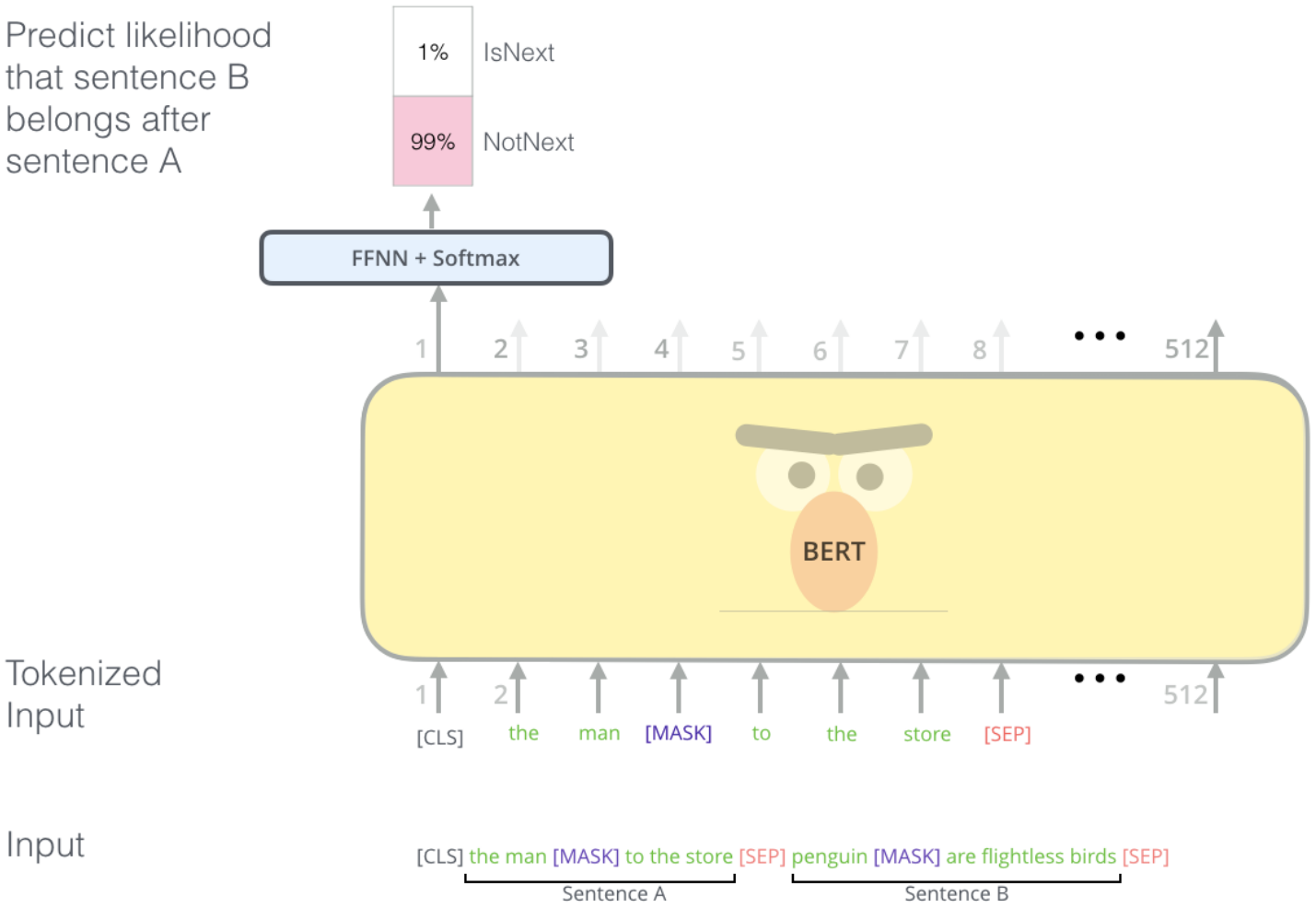
BERT: архитектура



BERT: предобучение

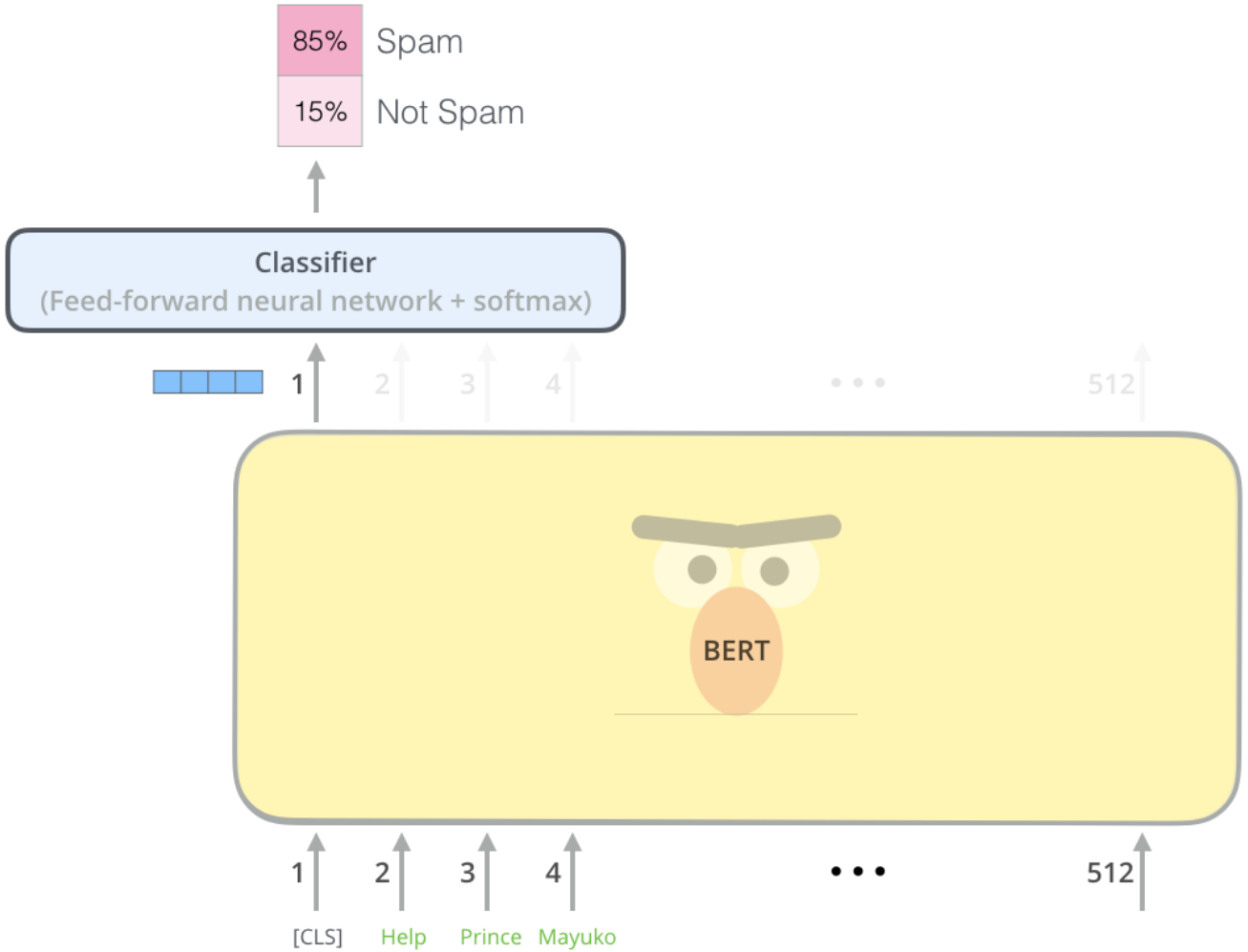


BERT: предобучение

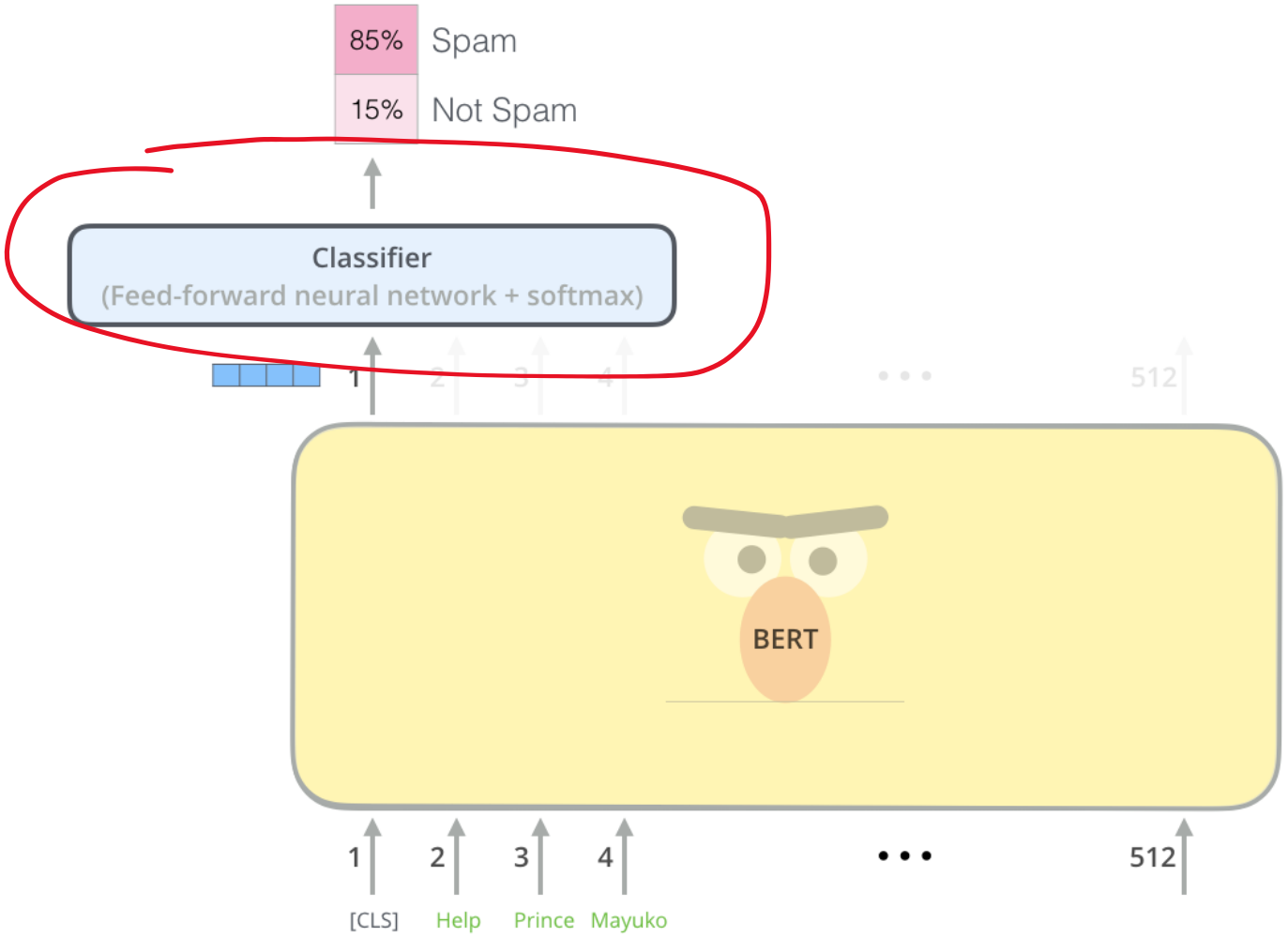


BERT: применение

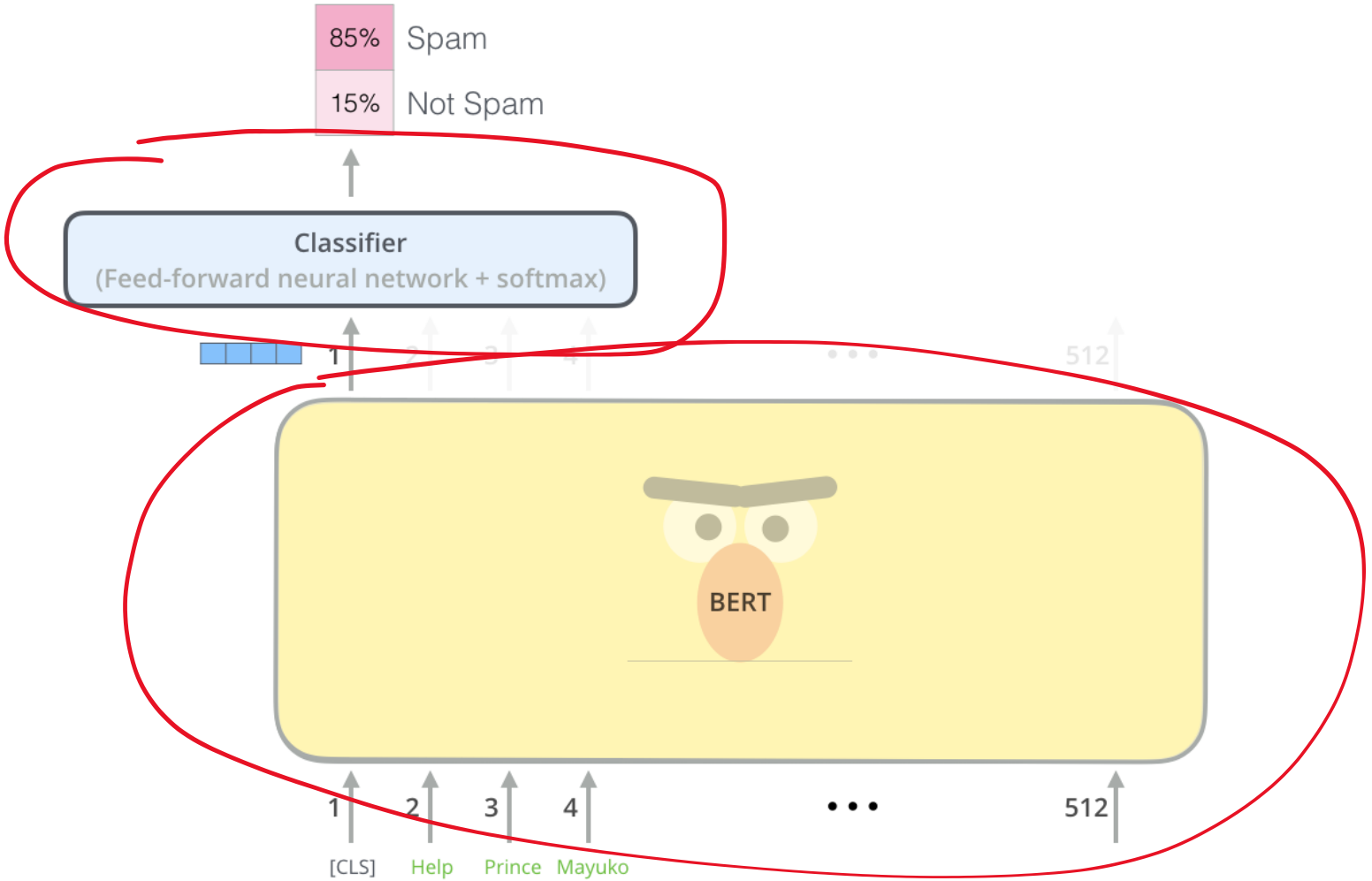
BERT: архитектура



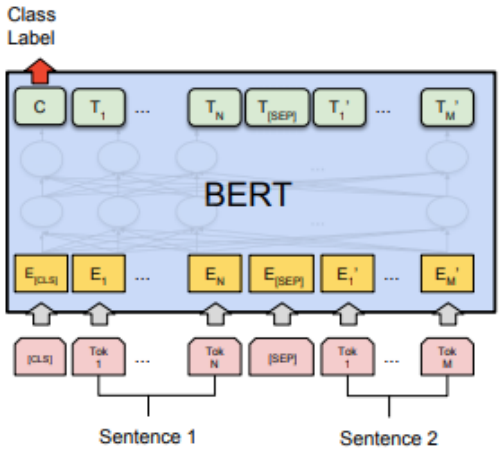
BERT: дообучение



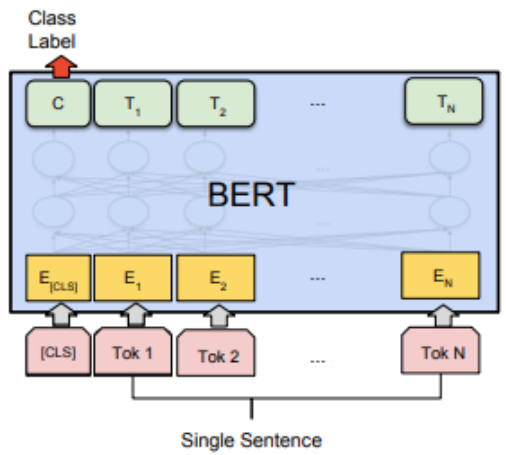
BERT: дообучение



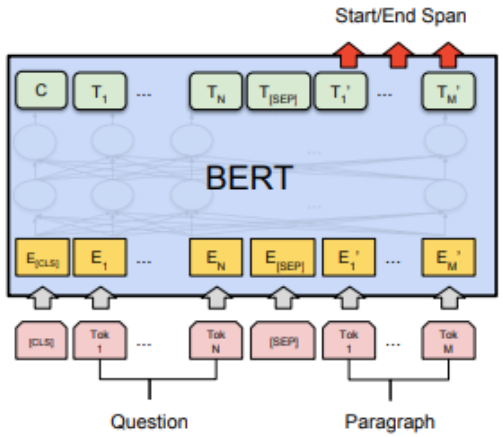
BERT: применение



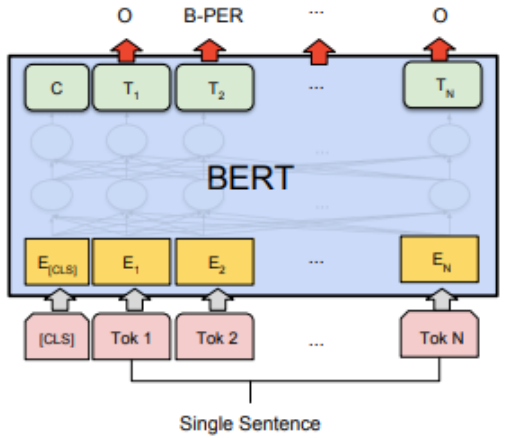
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA

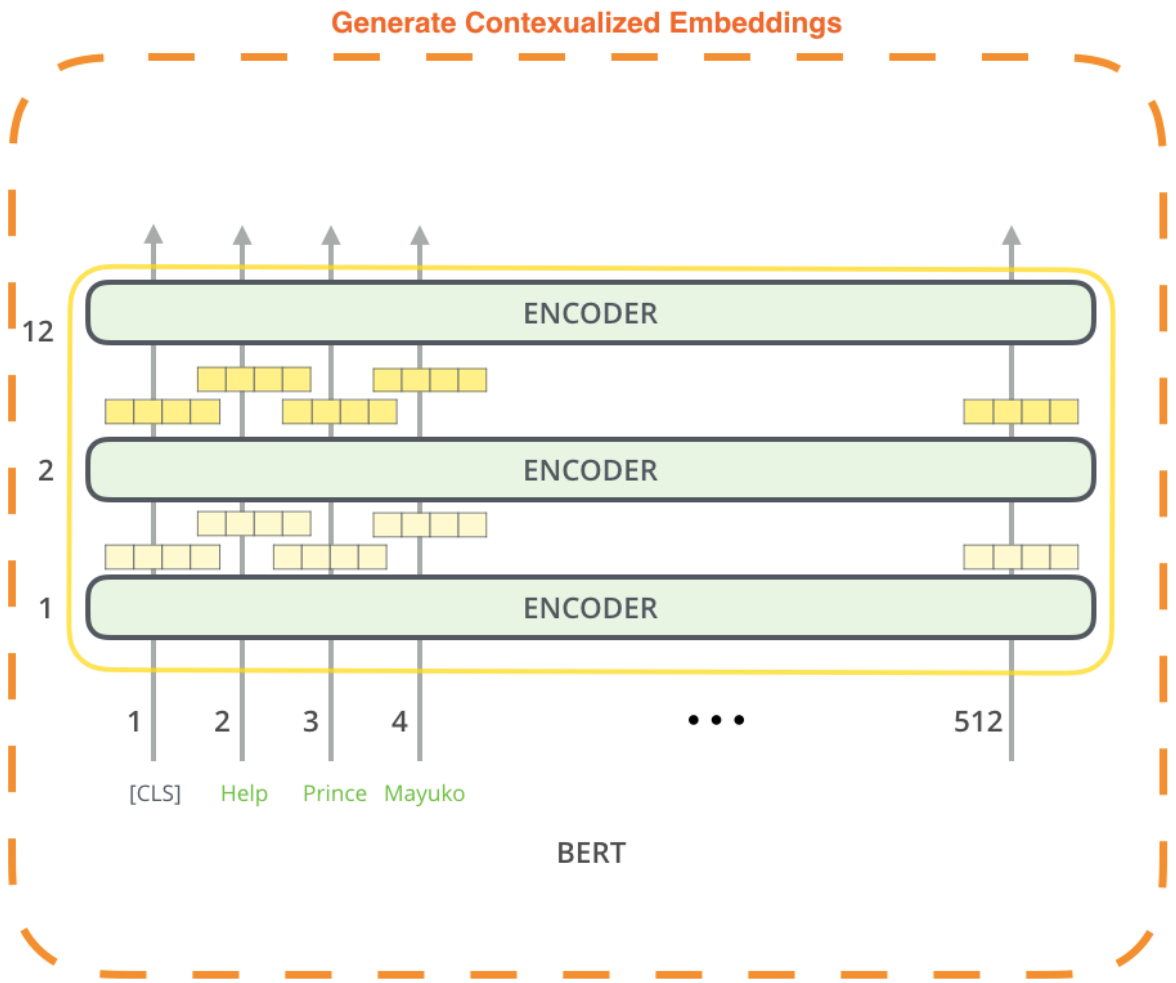


(c) Question Answering Tasks: SQuAD v1.1

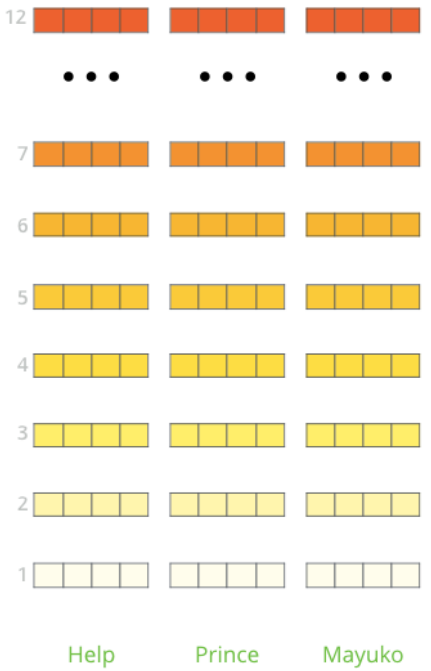


(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

BERT: применение



The output of each encoder layer along each token's path can be used as a feature representing that token.



But which one should we use?