



Лекция 2

Архитектура трансформера

20.02.24, ВМК МГУ
Владимир Макаренко

О чём сегодня поговорим

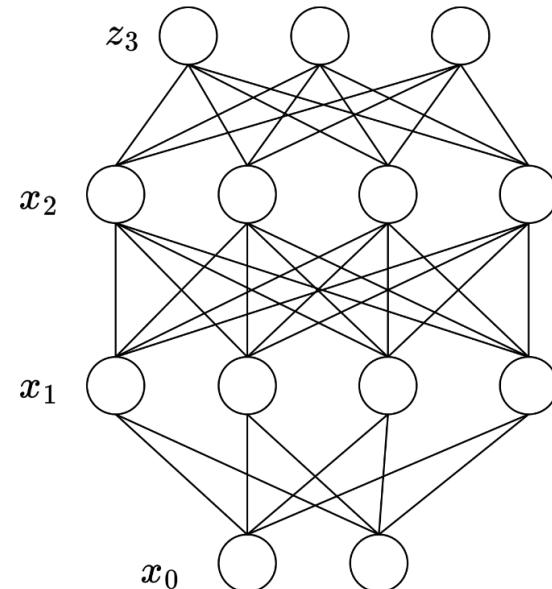
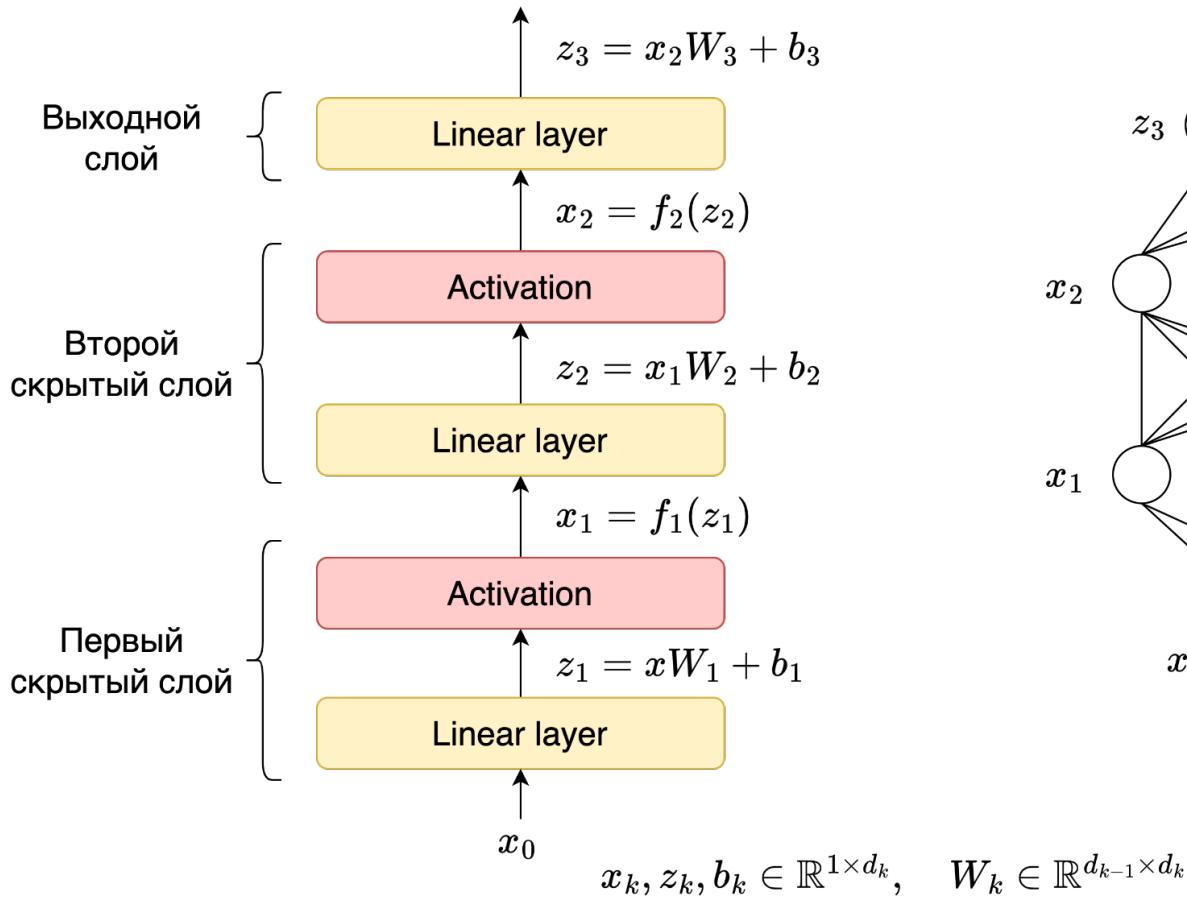
- Повторение
 - Полносвязные нейронные сети
 - Рекуррентные нейронные сети
 - Слой эмбеддингов
- Задача sequence to sequence
 - Постановка задачи
 - Как решали задачу с помощью рекуррентных нейросетей, и какие были проблемы
- Механизм внимания (attention) в рекуррентных нейросетях
 - Bahdanau attention
 - Luong attention
- На что похож механизм внимания?
 - Аналогия из информационного поиска
- Архитектура трансформера
 - Механизм self-attention
 - Архитектура кодировщика
 - Архитектура декодировщика



Повторение



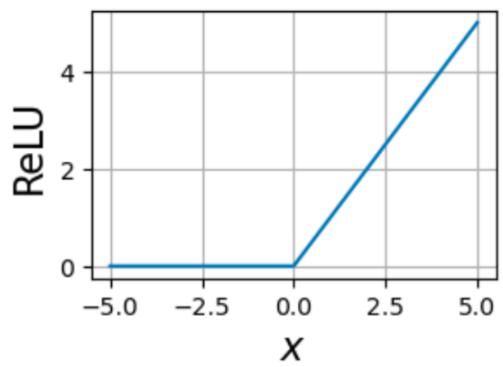
Fully-connected neural network (FCNN)



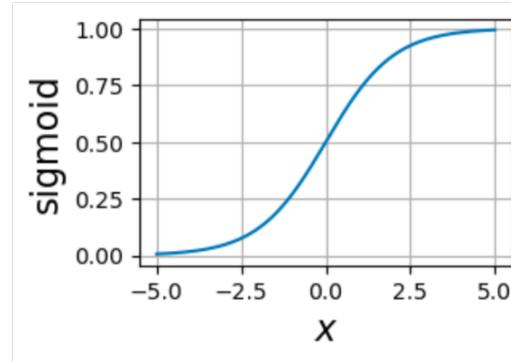
Fully-connected neural network (FCNN)

Популярные функции активации
(применяются покомпонентно):

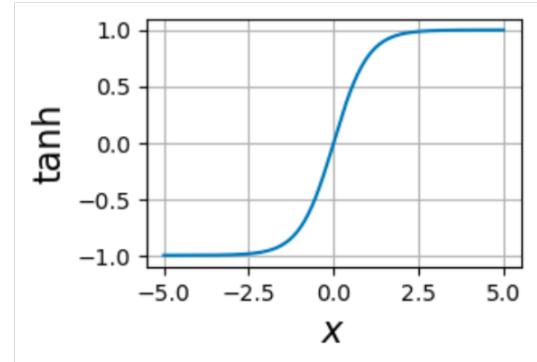
$$\text{ReLU}(x) = \max(0, x)$$



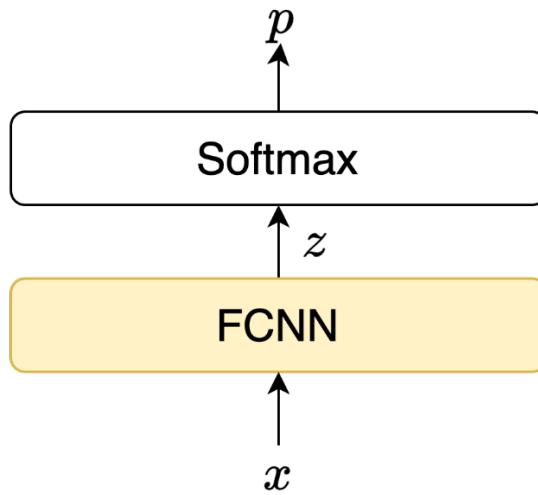
$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$



$$\tanh(x) = \frac{e^{-x} + e^x}{e^{-x} + e^x}$$



FCNN в задаче классификации



$$z = \text{fcnn}(x),$$

$$p = \text{softmax}(z), \quad p[j] = \frac{\exp(z[j])}{\sum_{i=1}^M \exp(z[i])}$$

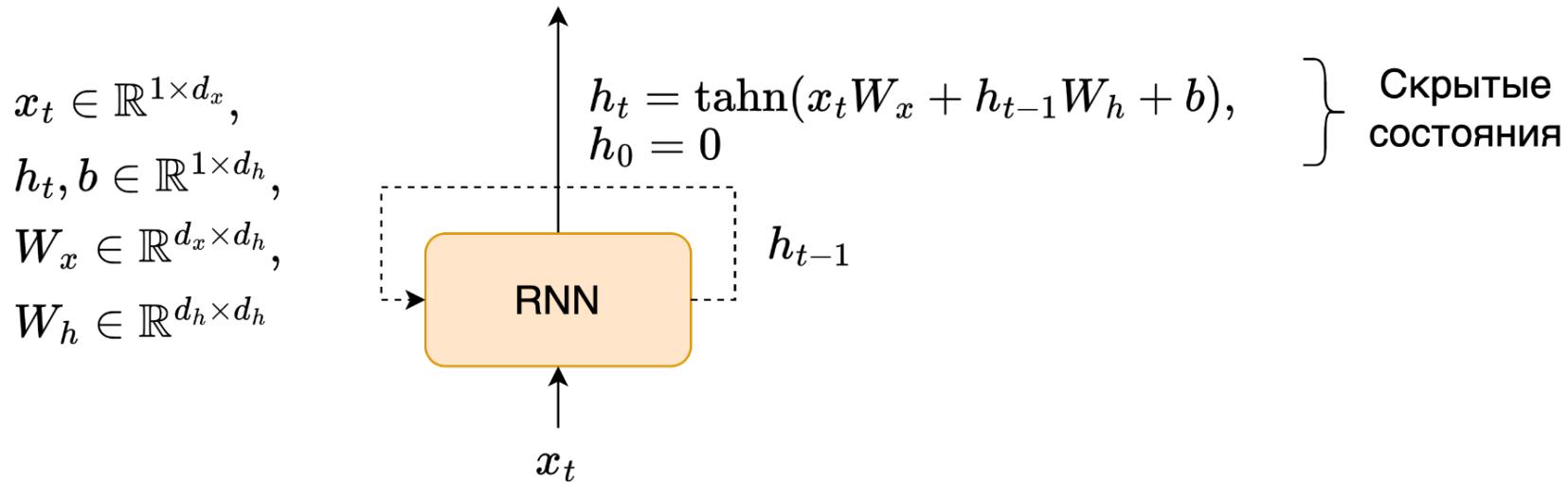
$$x \in \mathbb{R}^{1 \times d}, \quad z \in \mathbb{R}^{1 \times M}$$

Каждый объект x принадлежит одному из M классов.

Модель предсказывает, какому классу принадлежит объект x (распределение вероятностей p принадлежности каждому из классов).

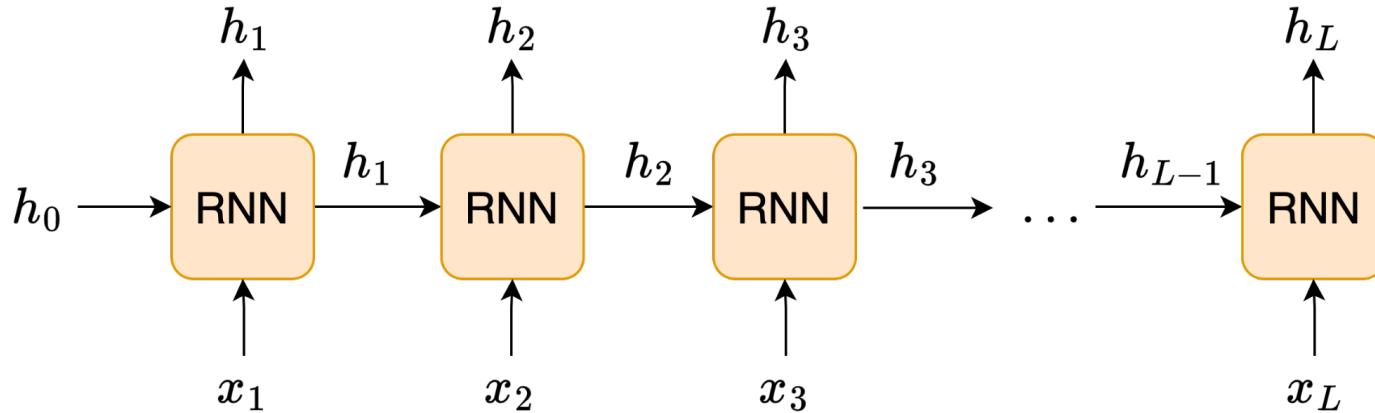
Recurrent neural network (RNN)

Цель: обработать последовательность x_1, x_2, \dots, x_L зависимых наблюдений **нефиксированной** длины.



- Состояние h_t можно рассматривать как внутреннюю память модели на шаге t .
- Состояние h_t неявно зависит от всех x_1, \dots, x_t , то есть хранит информацию о входах до шага t включительно.

Развернутый вид RNN



$$x_t \in \mathbb{R}^{1 \times d_x},$$

$$h_t, b \in \mathbb{R}^{1 \times d_h},$$

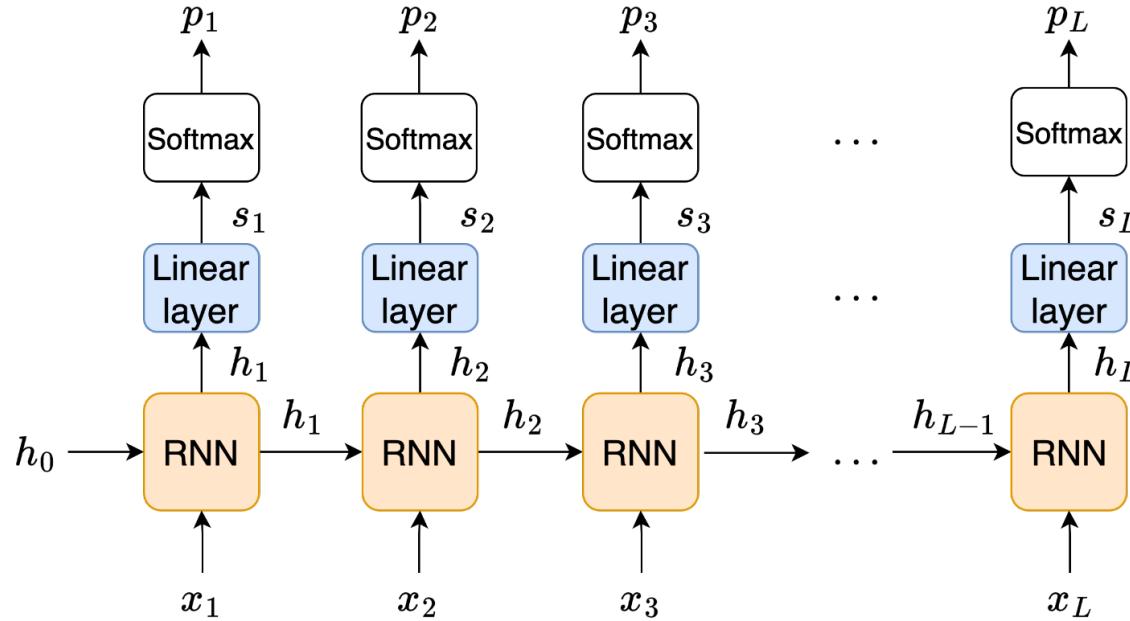
$$W_x \in \mathbb{R}^{d_x \times d_h},$$

$$W_h \in \mathbb{R}^{d_h \times d_h}$$

$$h_t = \tanh(x_t W_x + h_{t-1} W_h + b),$$

$$h_0 = 0$$

Классификация элементов последовательности с RNN



$$h_t = \tanh(x_t W_x + h_{t-1} W_h + b),$$

$$h_0 = 0,$$

$$s_t = h_t W_s + b_s,$$

$$p_t = \text{softmax}(s_t)$$

s_t - логиты на шаге t ,

p_t - предсказанное распределение на шаге t
(M классов)

$$x_t \in \mathbb{R}^{1 \times d_x},$$

$$h_t, b_h \in \mathbb{R}^{1 \times d_h},$$

$$s_t, p_t, b_s \in \mathbb{R}^{1 \times M},$$

$$W_x \in \mathbb{R}^{d_x \times d_h},$$

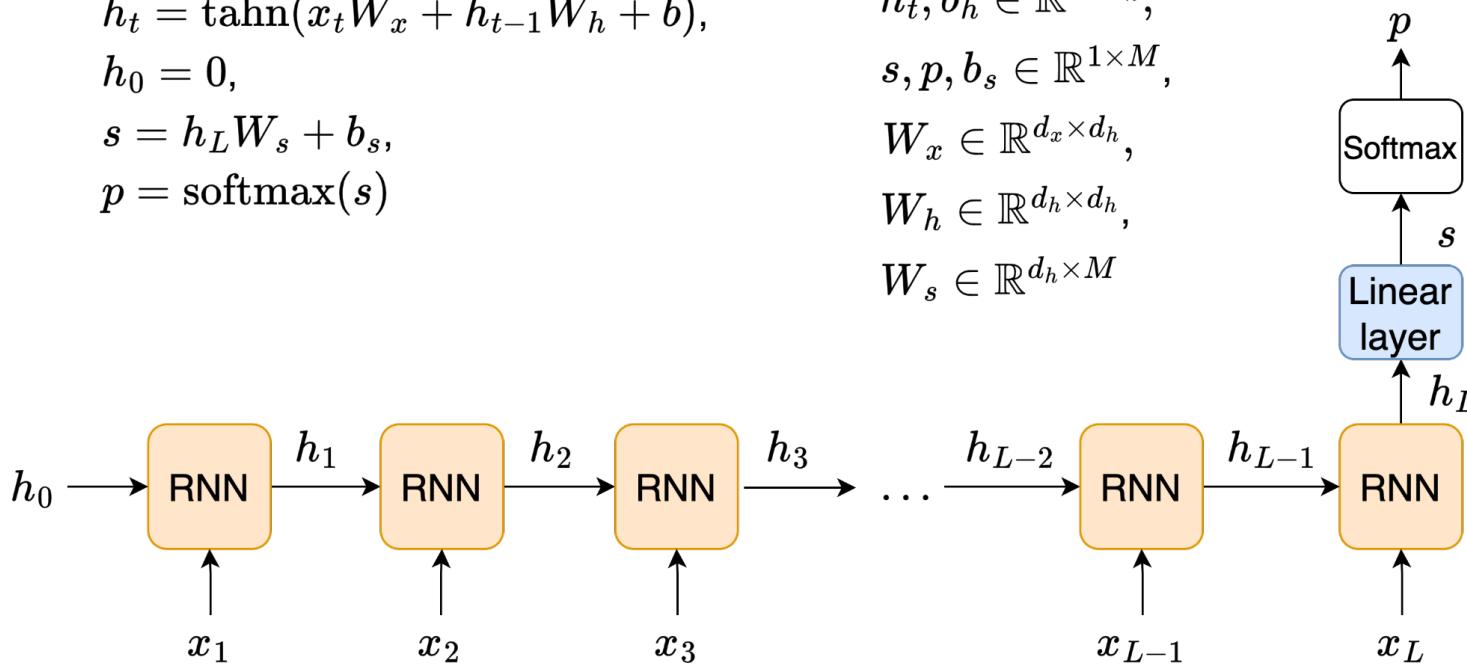
$$W_h \in \mathbb{R}^{d_h \times d_h},$$

$$W_s \in \mathbb{R}^{d_h \times M}$$

Классификация последовательности с RNN

$$\begin{aligned} h_t &= \tanh(x_t W_x + h_{t-1} W_h + b), \\ h_0 &= 0, \\ s &= h_L W_s + b_s, \\ p &= \text{softmax}(s) \end{aligned}$$

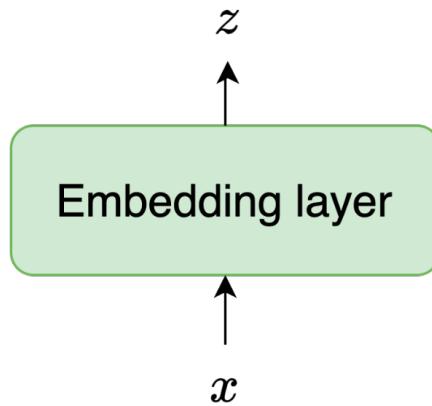
$$\begin{aligned} x_t &\in \mathbb{R}^{1 \times d_x}, \\ h_t, b_h &\in \mathbb{R}^{1 \times d_h}, \\ s, p, b_s &\in \mathbb{R}^{1 \times M}, \\ W_x &\in \mathbb{R}^{d_x \times d_h}, \\ W_h &\in \mathbb{R}^{d_h \times d_h}, \\ W_s &\in \mathbb{R}^{d_h \times M} \end{aligned}$$



s - логиты,

p - предсказанное распределение (M классов)

Слой эмбеддингов (Embedding layer)



$$z = (W_{x,1}, W_{x,2}, \dots, W_{x,d}),$$

$$x \in \{0, 1, 2, \dots, V - 1\},$$

$$z \in \mathbb{R}^{1 \times d},$$

$$W \in \mathbb{R}^{V \times d}$$

Пусть x может принимать одно из V значений (от 0 до $V - 1$).
Слой эмбеддингов превращает x в вектор.

В NLP слой эмбеддингов каждому слову (номер в словаре) сопоставляет вектор.

Слой эмбеддингов (Embedding layer)

Словарь

Слово	ID
мама	0
папа	1
банан	2
машина	3

W

-0.1	0.2	0.7
0.5	-1	1.1
-0.6	0.2	-0.3
0.1	-0.3	1

Какой эмбеддинг у слова "банан"?

Слой эмбеддингов (Embedding layer)

Словарь

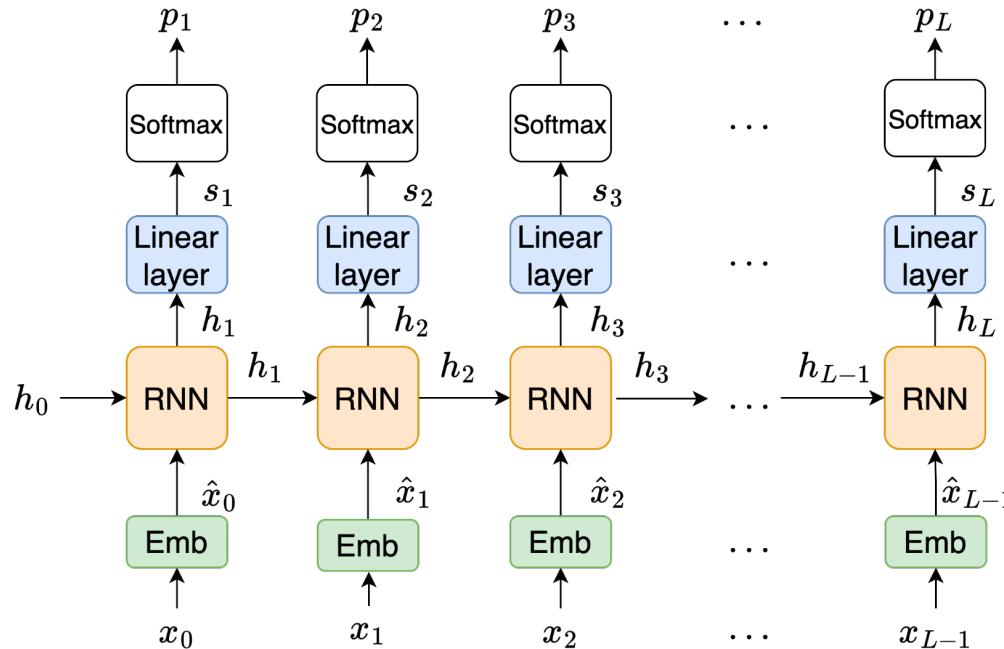
Слово	ID
мама	0
папа	1
банан	2
машина	3

W

-0.1	0.2	0.7
0.5	-1	1.1
-0.6	0.2	-0.3
0.1	-0.3	1

Ответ: (-0.6, 0.2, -0.3)

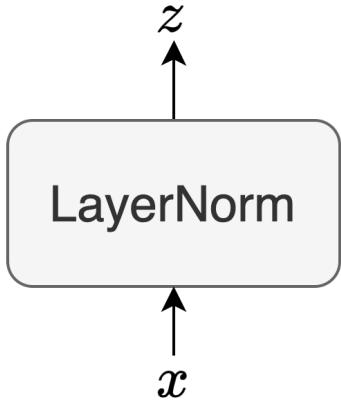
Генерация текстов с помощью RNN



На каждом шаге:

1. Подаем очередной токен на вход (ID в словаре).
2. Превращаем токен в эмбеддинг.
3. Прогоняем эмбеддинг через сеть и в конце получаем вероятность следующего токена.
4. Семплируем следующий токен из предсказанного распределения.

Слой LayerNorm



$$z_i = \alpha \cdot \frac{x_i - \hat{\mu}}{\hat{\sigma}} + \beta, \quad i \in \{1, 2, \dots, d\},$$

$$\hat{\mu} = \frac{\sum_{i=1}^d x_i}{d},$$

$$\hat{\sigma} = \sqrt{\frac{\sum_{i=1}^d (x_i - \hat{\mu})^2}{d}}$$

$$x, z \in \mathbb{R}^{1 \times d},$$

$$\alpha, \beta \in \mathbb{R}$$

Задача sequence to sequence

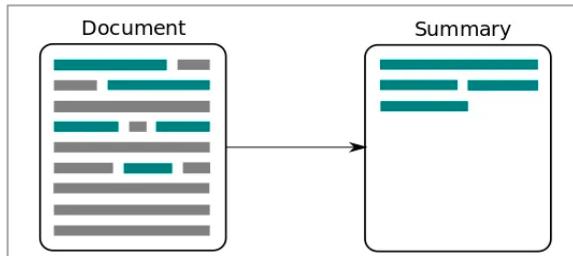


Задача sequence to sequence (seq2seq)

По входной последовательности токенов нужно сгенерировать другую последовательность токенов (возможно, из другого словаря и другой длины)

Примеры:

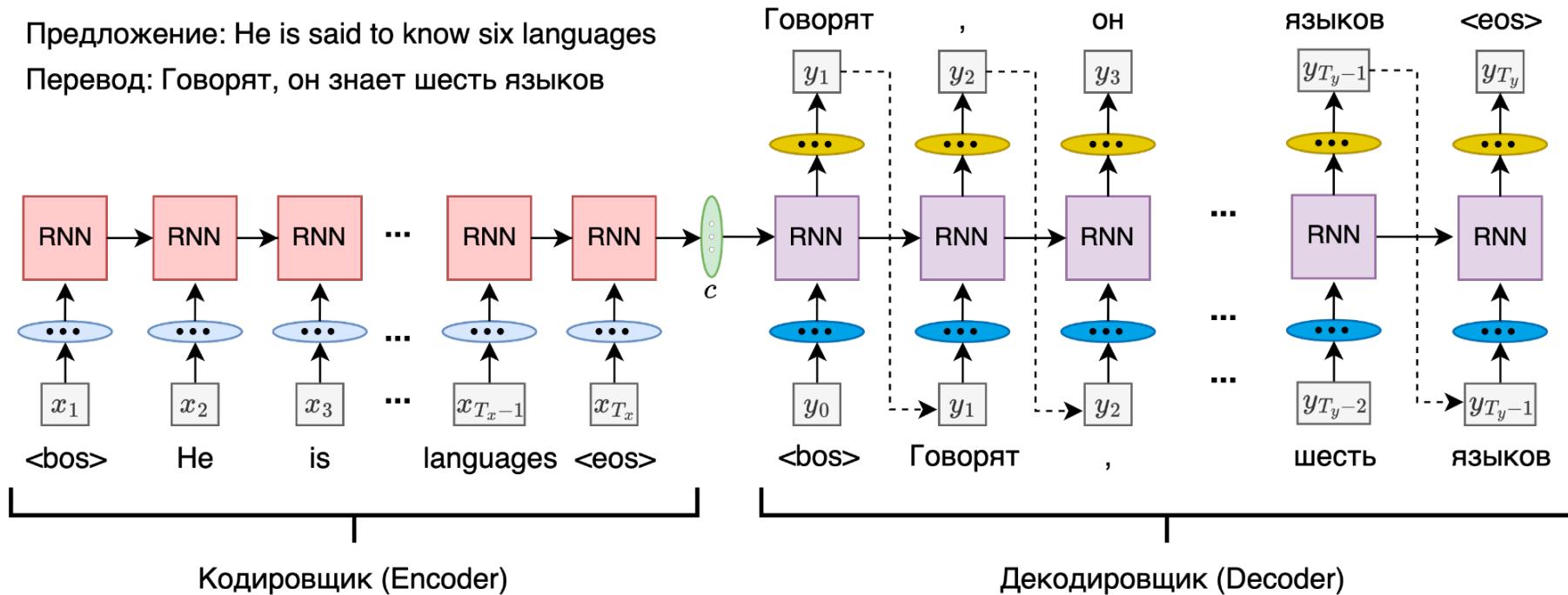
- Машинный перевод
- Суммаризация текста
- Генерация ответа на вопрос



Модель sequence-to-sequence на основе RNN

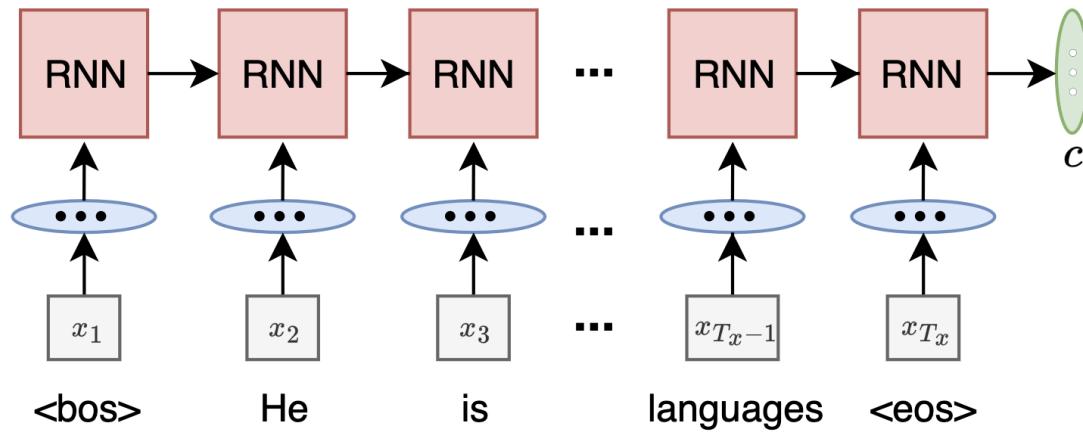
Предложение: He is said to know six languages

Перевод: Говорят, он знает шесть языков



Модель состоит из двух частей: кодировщик (encoder) и декодировщик (decoder).

Модель sequence-to-sequence на основе RNN



Кодировщик читает входную последовательность и формирует ее представление в виде вектора – финального скрытого состояния (hidden state).



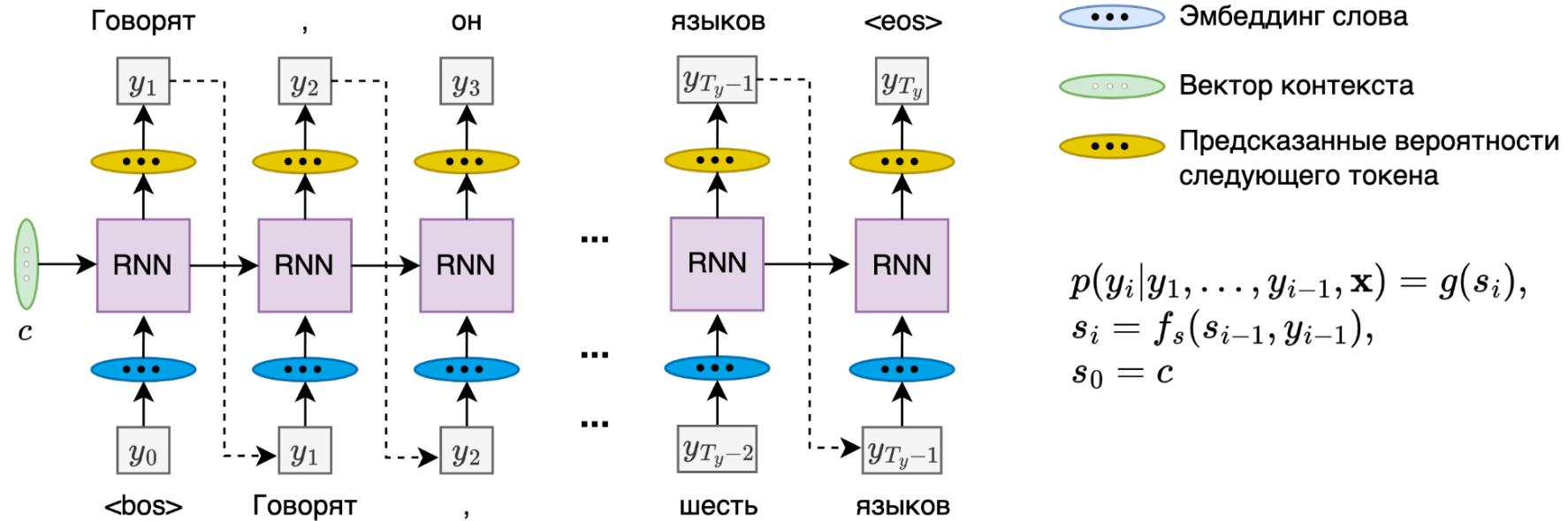
Эмбеддинг слова



Вектор контекста

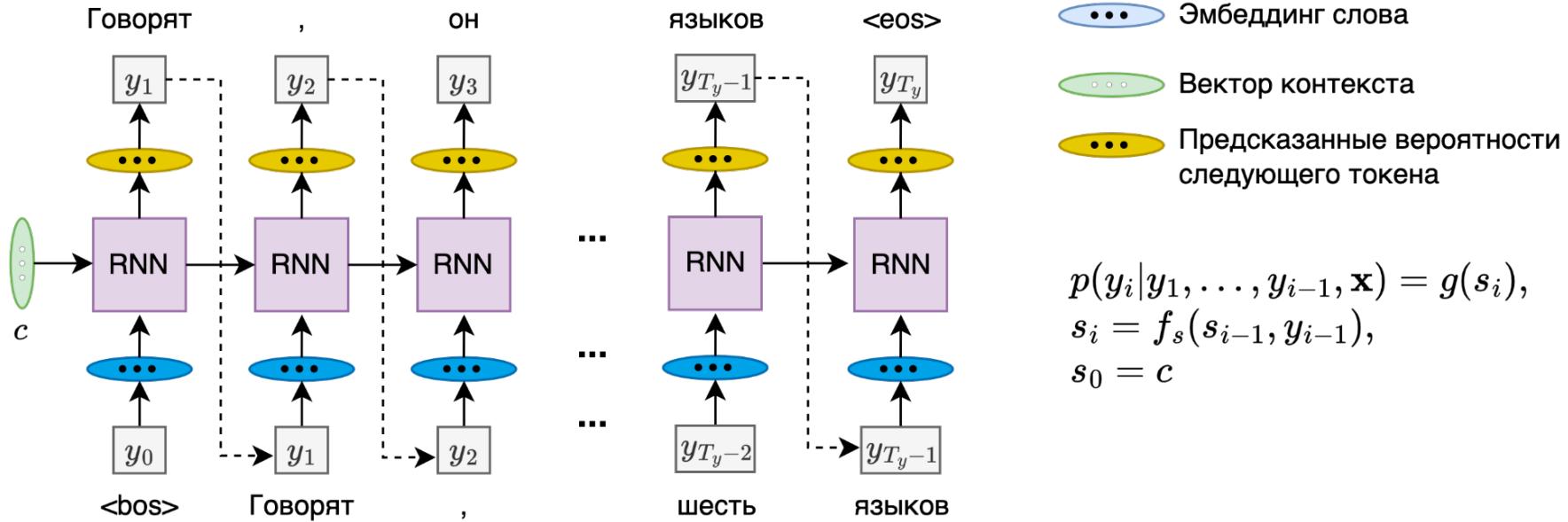
$$\begin{aligned} c &= h_{T_x}, \\ h_i &= f_h(h_{i-1}, x_i), \\ h_0 &= 0 \end{aligned}$$

Модель sequence-to-sequence на основе RNN



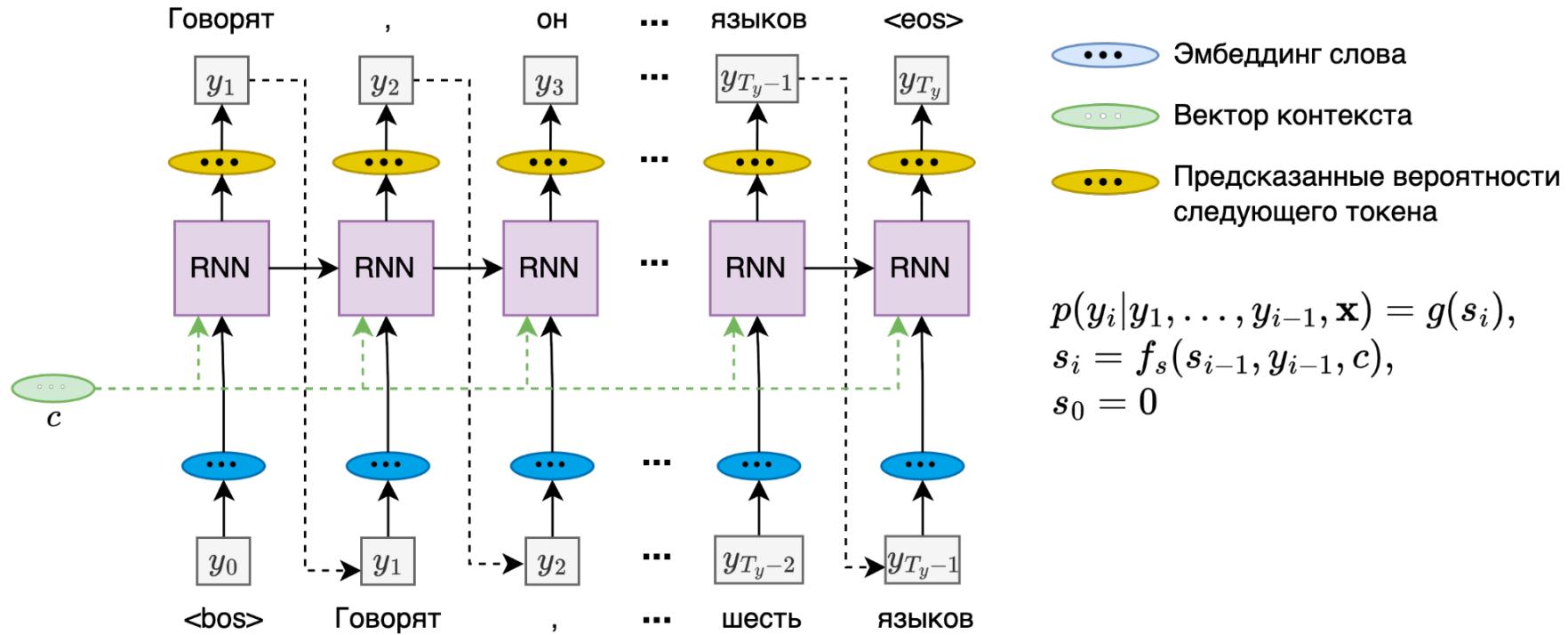
Декодировщик принимает полученное представление последовательности в виде своего начального скрытого состояния и генерирует выходную последовательность.

Модель sequence-to-sequence на основе RNN



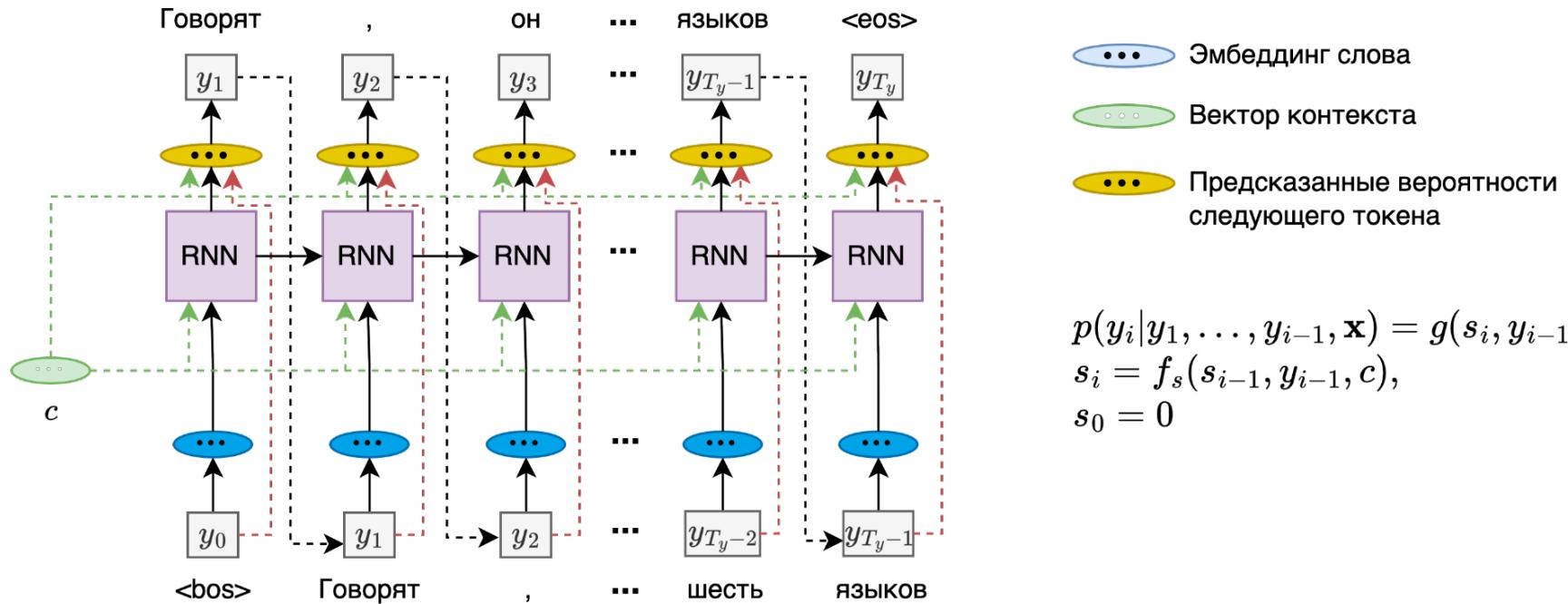
Проблема: на каждом шаге не имеем прямого доступа к вектору контекста, и декодировщик быстро его забывает.

Модель sequence-to-sequence на основе RNN



Решение: на каждом шаге будем явно передавать вектор контекста в нейросеть.

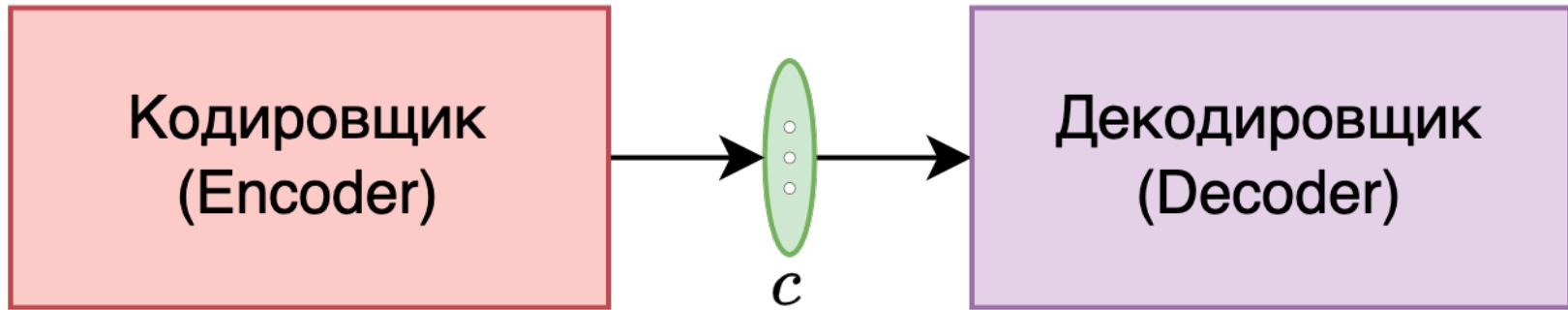
Модель sequence-to-sequence на основе RNN



$$p(y_i | y_1, \dots, y_{i-1}, \mathbf{x}) = g(s_i, y_{i-1}, c),$$
$$s_i = f_s(s_{i-1}, y_{i-1}, c),$$
$$s_0 = 0$$

Можно еще и явно передавать вектор контекста и предыдущий токен для предсказания вероятностей.

Модель sequence-to-sequence на основе RNN



Проблемы:

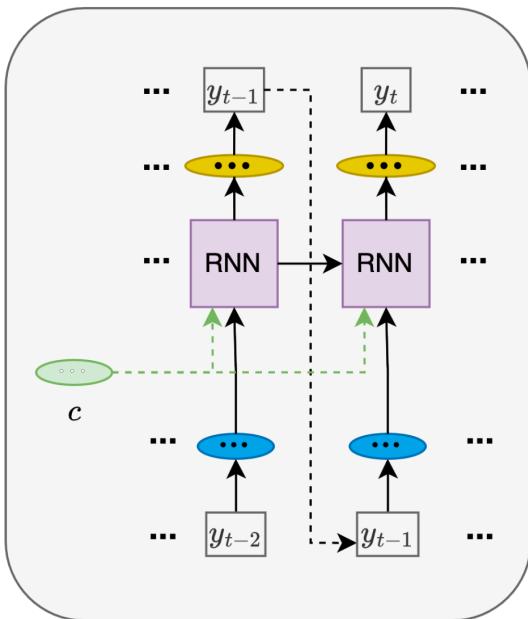
- Узкое место (bottleneck) в виде вектора контекста между encoder и decoder. Для длинных последовательностей теряется слишком много информации при попытке сжать ее в вектор.
- Плохо параллелируется при обучении и применении.

Механизм внимания в рекуррентных нейросетях



Механизм внимания

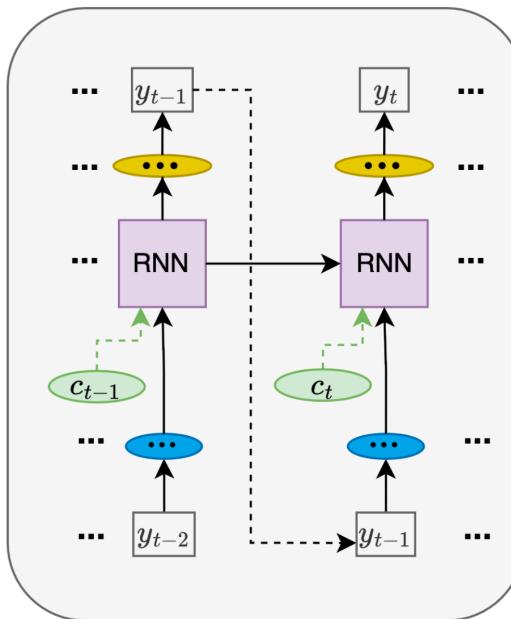
Было



Один и тот же вектор контекста
на каждом шаге

$$c = h_{T_x}$$

Стало

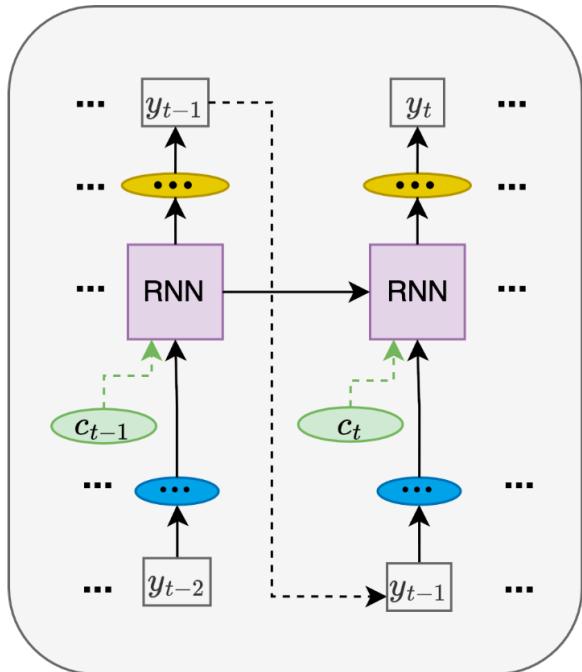


Свой вектор контекста
на каждом шаге

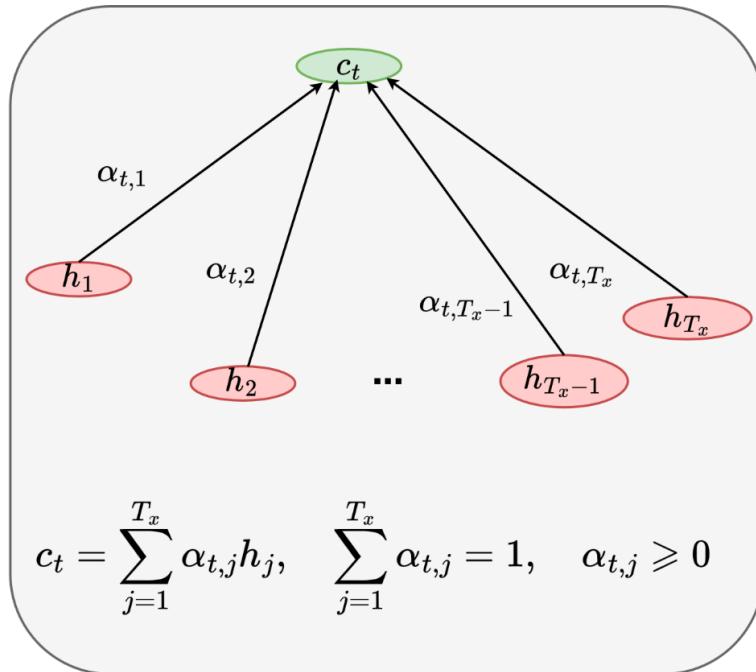
$$c_t = ?$$

Механизм внимания

Свой вектор контекста на каждом шаге



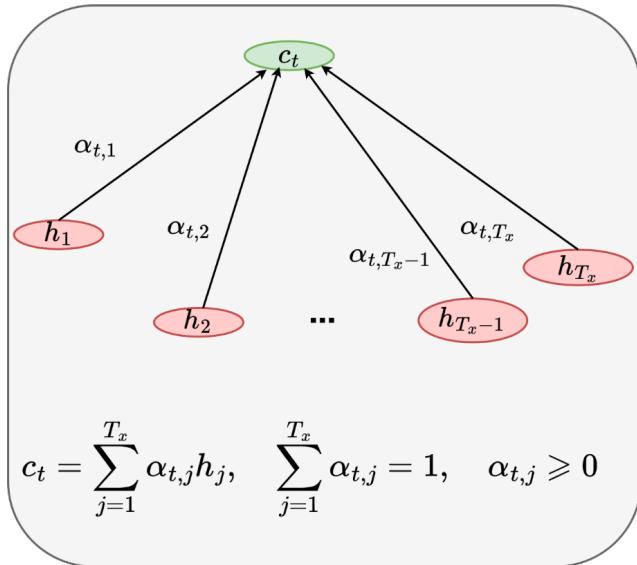
Расчет контекстного вектора



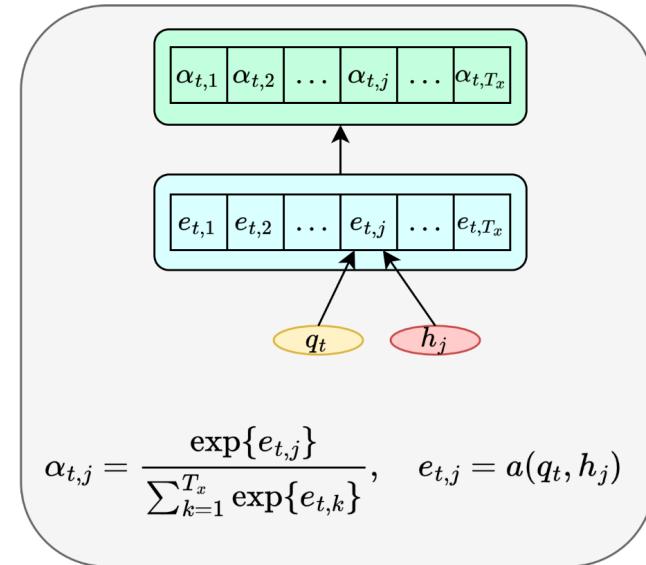
Расчитываем, с какой вероятностью нужно взять h_t в качестве контекстного вектора.
В качестве контекстного вектора берем математическое ожидание. Как считать $\alpha_{t,j}$?

Механизм внимания

Расчет контекстного вектора



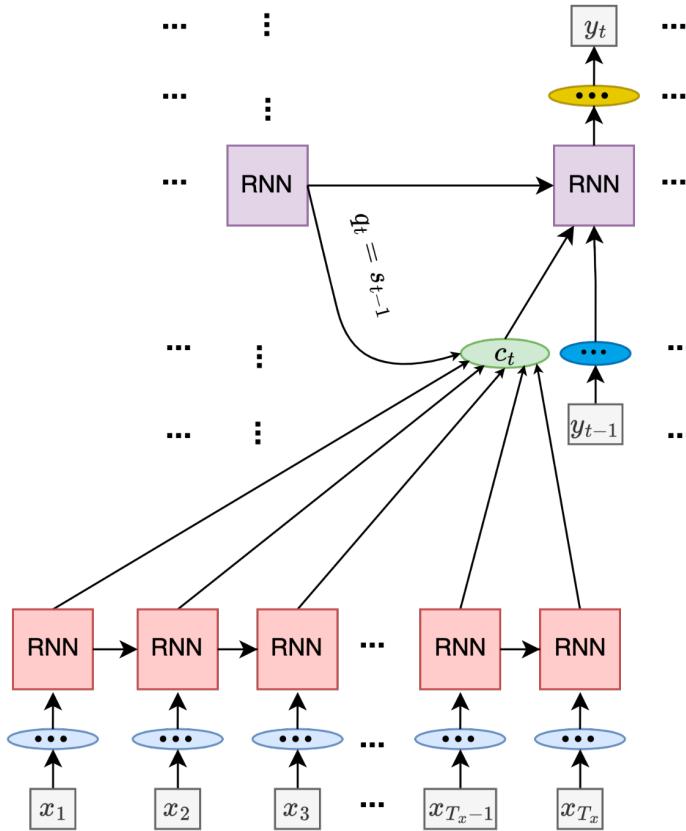
Расчет коэффициентов



Вектор q_t - запрос на шаге t .
Функция a - модель выравнивания.

Осталось только выбрать модель внимания и способ получения запросов.

Механизм внимания Bahdanau et al. (2014)

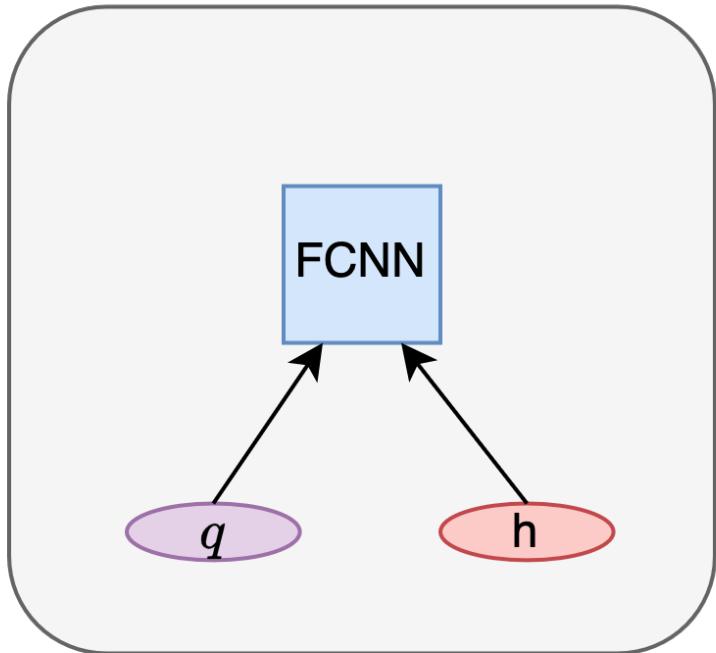


$$\begin{aligned} p(y_t | y_1, \dots, y_{t-1}, \mathbf{x}) &= g(s_t), \\ s_t &= f_s(s_{t-1}, y_{t-1}, c_t), \\ c_t &= \sum_{j=1}^{T_x} \alpha_{t,j} h_j, \\ \alpha_{t,j} &= \frac{\exp\{e_{t,j}\}}{\sum_{k=1}^{T_x} \exp\{e_{t,k}\}}, \\ e_{t,j} &= a(q_t, h_j), \quad q_t = s_{t-1} \end{aligned}$$

В качестве запроса
выступает предыдущее
состояние
декодировщика

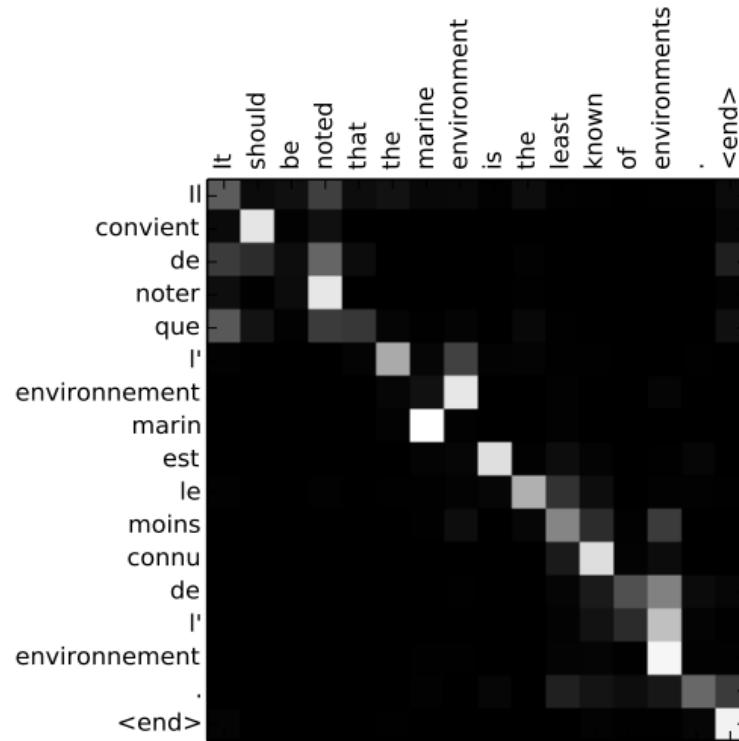
Механизм внимания Bahdanau et al. (2014)

$$a(q, h)$$

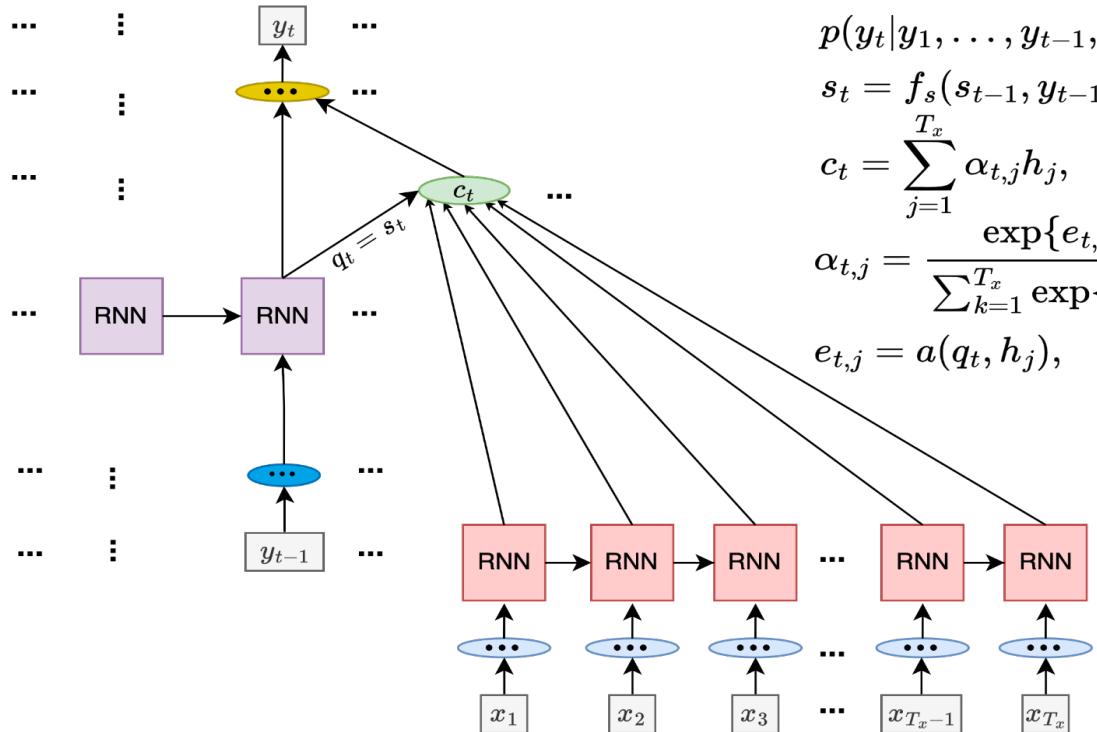


В качестве модели выравнивания
выступает полносвязная нейросеть
(fcnn) с одним скрытым слоем

Механизм внимания Bahdanau et al. (2014)



Механизм внимания Luong et al. (2015)



$$\begin{aligned} p(y_t | y_1, \dots, y_{t-1}, \mathbf{x}) &= g(s_t, c_t), \\ s_t &= f_s(s_{t-1}, y_{t-1}), \\ c_t &= \sum_{j=1}^{T_x} \alpha_{t,j} h_j, \\ \alpha_{t,j} &= \frac{\exp\{e_{t,j}\}}{\sum_{k=1}^{T_x} \exp\{e_{t,k}\}}, \\ e_{t,j} &= a(q_t, h_j), \quad q_t = s_t \end{aligned}$$

В качестве запроса
выступает текущее
состояние
декодировщика

Механизм внимания Luong et al. (2015)

Варианты модели выравнивания:

$$a(q, h) = q^T h,$$

$$a(q, h) = q^T W_a h,$$

$$a(q, h) = \text{fcnn}([q, h])$$

На что похож механизм внимания?



Поиск информации по ключу

```
key1, value1 = 'Вячеслав', 17
key2, value2 = 'Святослав', 22
key3, value3 = 'Иван', 25
key4, value4 = 'Сергей', 20

age = {
    key1: value1,
    key2: value2,
    key3: value3,
    key4: value4
}

query = 'Святослав'
value = age[query]
print(value) # 22

query = 'Ваня'
value = age[query] # Ошибка
```

- Пусть у нас некоторая база данных в которых хранятся пары ключ-значение (key-value).
- Нам поступает запрос (query) и мы хотим по нему найти значение в базе данных.
- Проблема: для запроса не всегда существует ключ, который с ним совпадает, хотя могут быть похожие ключи ("Иван", "Ваня")

Мягкий поиск информации по запросу

Запрос (query): Какой возраст у программиста?

Фамилия	Должность (key)	Возраст (value)
Иванов	Бухгалтер	50
Петров	Старший Python-разработчик	30
Сидоров	Системный администратор	35
Смирнова	Менеджер	27
Кузнецова	Продуктовый аналитик	33
Васильева	Младший C++-разработчик	22

Каждой записи присвоим неотрицательный вес в соответствии с релевантностью ключа запросу. Сумма весов должна быть равна единице.

Мягкий поиск информации по запросу

Запрос (query): Какой возраст у программиста?

Фамилия	Должность (key)	Возраст (value)	
Иванов	Бухгалтер	50	0
Петров	Старший Python-разработчик	30	0.4
Сидоров	Системный администратор	35	0.2
Смирнова	Менеджер	27	0
Кузнецова	Продуктовый аналитик	33	0
Васильева	Младший C++-разработчик	22	0.4

Далее суммируем values в соответствии с весами, получая 27.8.

Если рассматривать веса как вероятности, то нашли матожидание value.

Что из себя представляет механизм внимания?

- Запросы
 - Предыдущие состояния декодировщика (Bahdanau)
 - Текущие состояния декодировщика (Luong)
- Ключи
 - Состояния кодировщика
- Значения
 - Состояния кодировщика

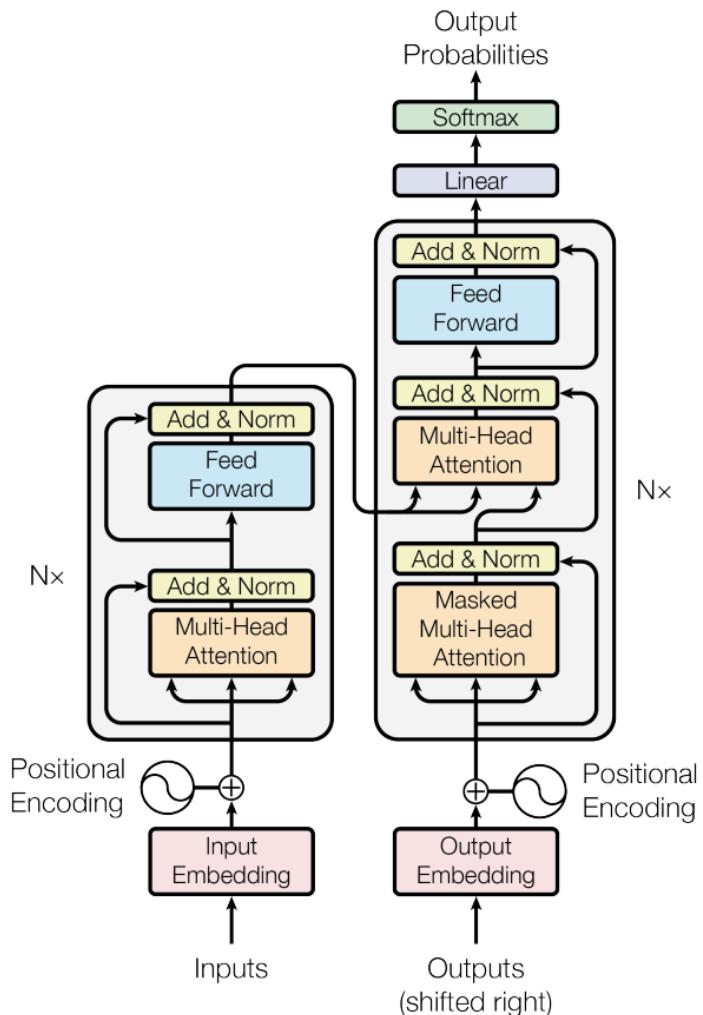
Архитектура трансформера



Transformer (Attention is all you need, 2017)

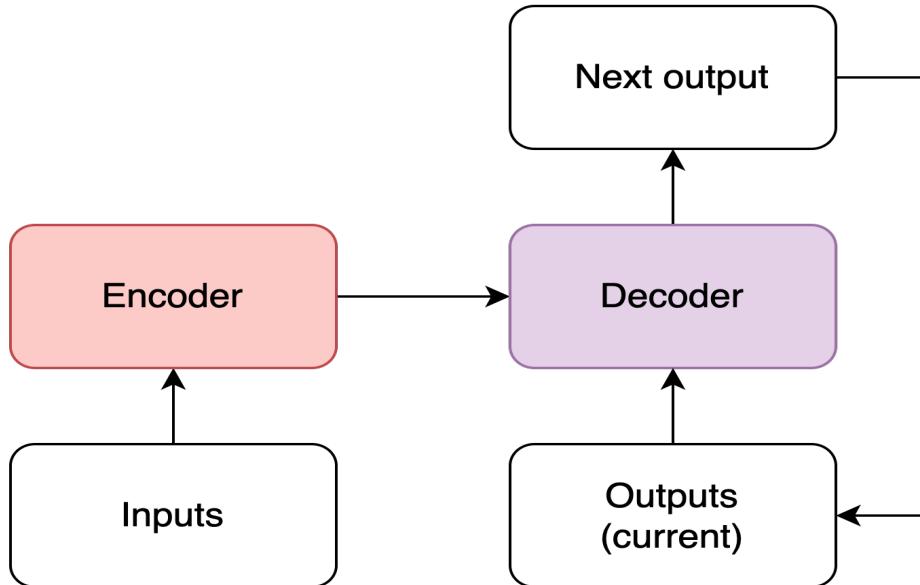
Архитектура, пришедшая на смену RNN:

- Хорошо параллелируется
- Внутри нет рекуррентности и связанной с ней проблем, используется только механизм внимания
- Имеем доступ ко всей информации (нет узкого места, как в RNN)



Верхнеуровневое устройство трансформера

- Как и раньше, есть две части: encoder и decoder
- Encoder кодирует входную последовательность
- Decoder предсказывает следующий токен выходной последовательности, имея выходы кодировщика и предыдущие токены выходной последовательности
- Генерация начинается с токена `<bos>` (его передаем сами, а модель генерирует последующие токены)



Устройство кодировщика

Encoder h_1, \dots, h_L

Encoder layer N

Encoder layer $N - 1$

Encoder layer 2

Encoder layer 1

Embeddings

x_1, \dots, x_L

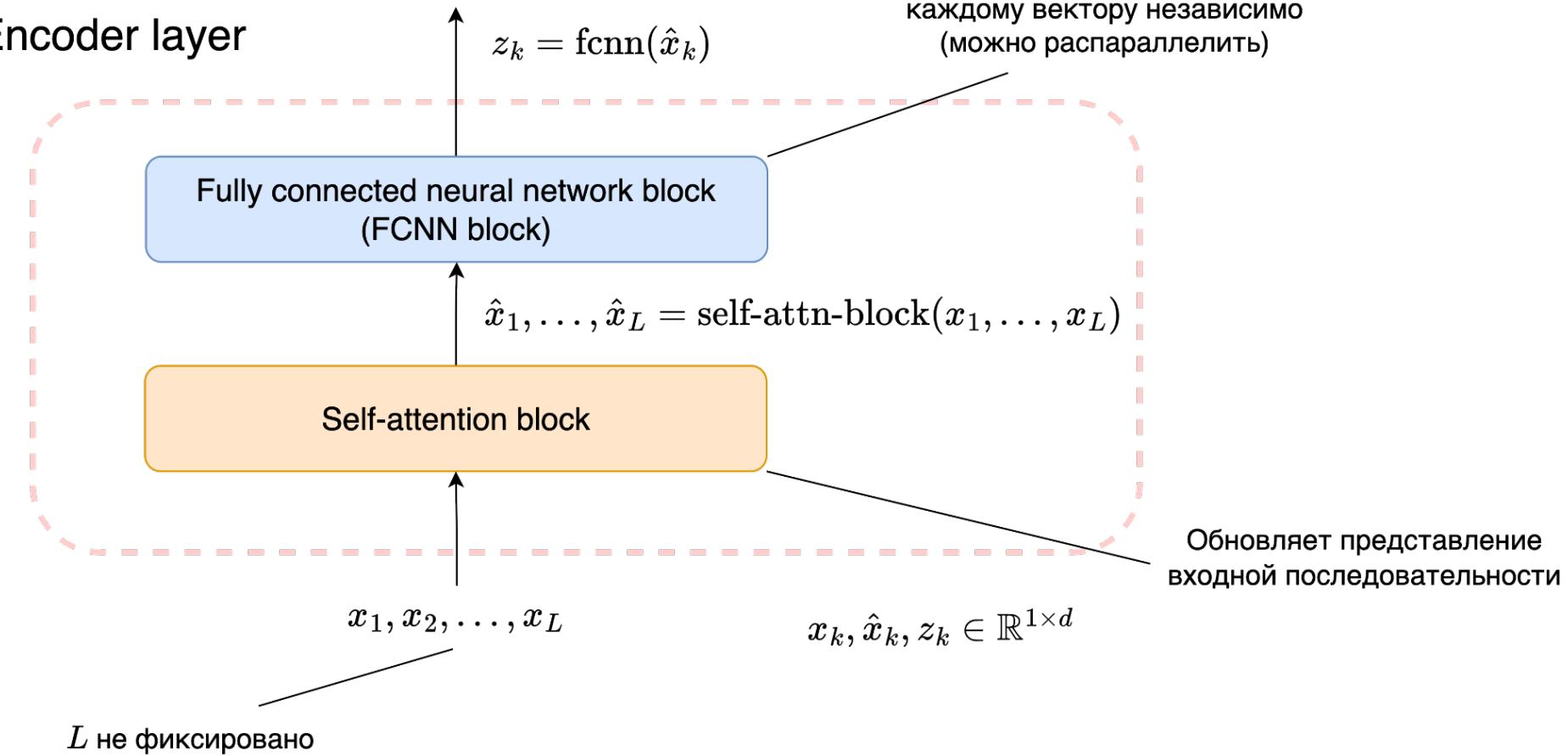
Кодировщик:

- Принимает на вход последовательность токенов
- Вычисляет эмбеддинги входных токенов (слой embeddings)
- Применяет последовательно N однотипных преобразований (encoder layer)

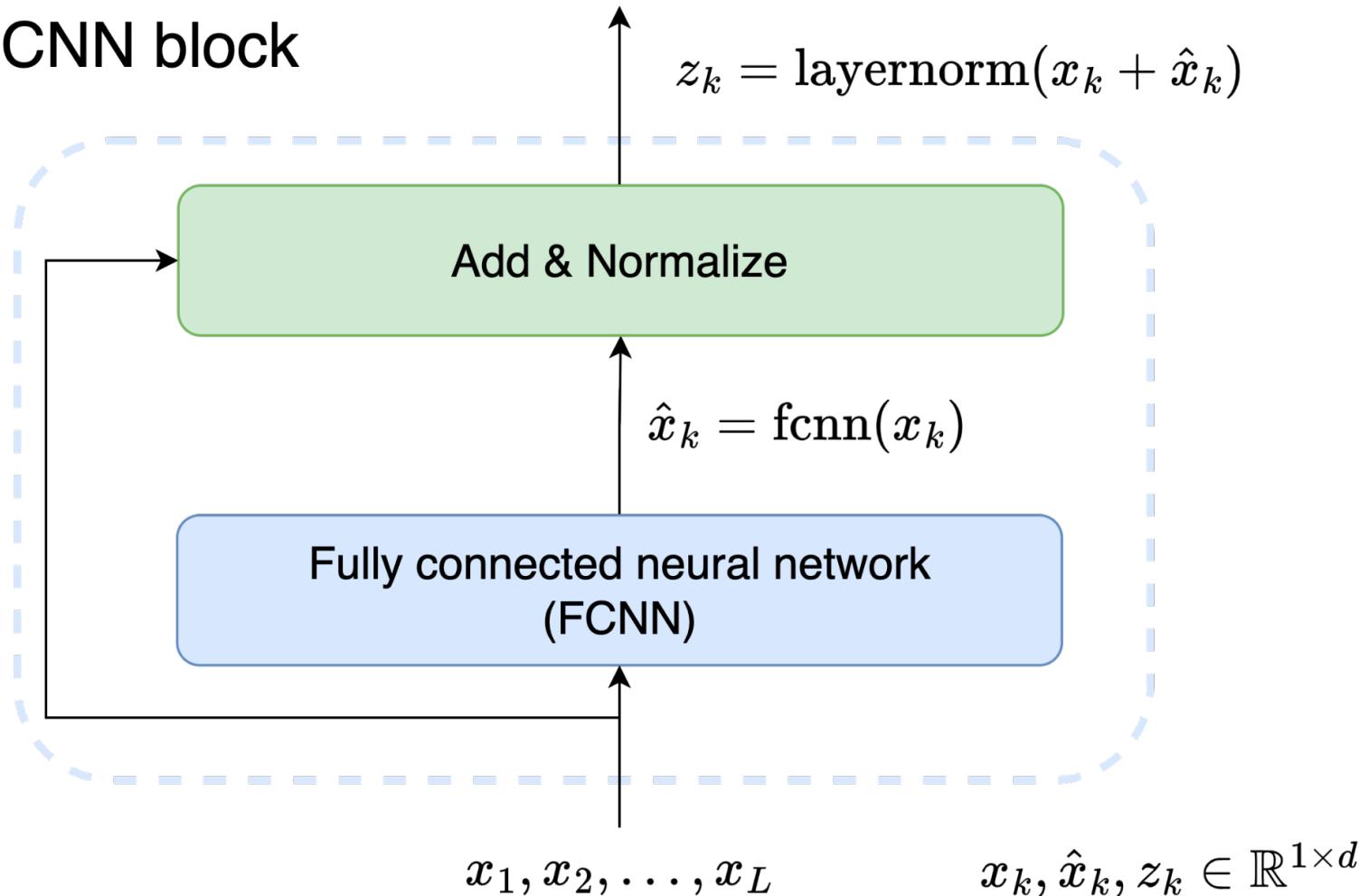
$$x_k \in \mathbb{N},$$

$$h_k \in \mathbb{R}^{1 \times d}$$

Encoder layer



FCNN block



Self-attention block

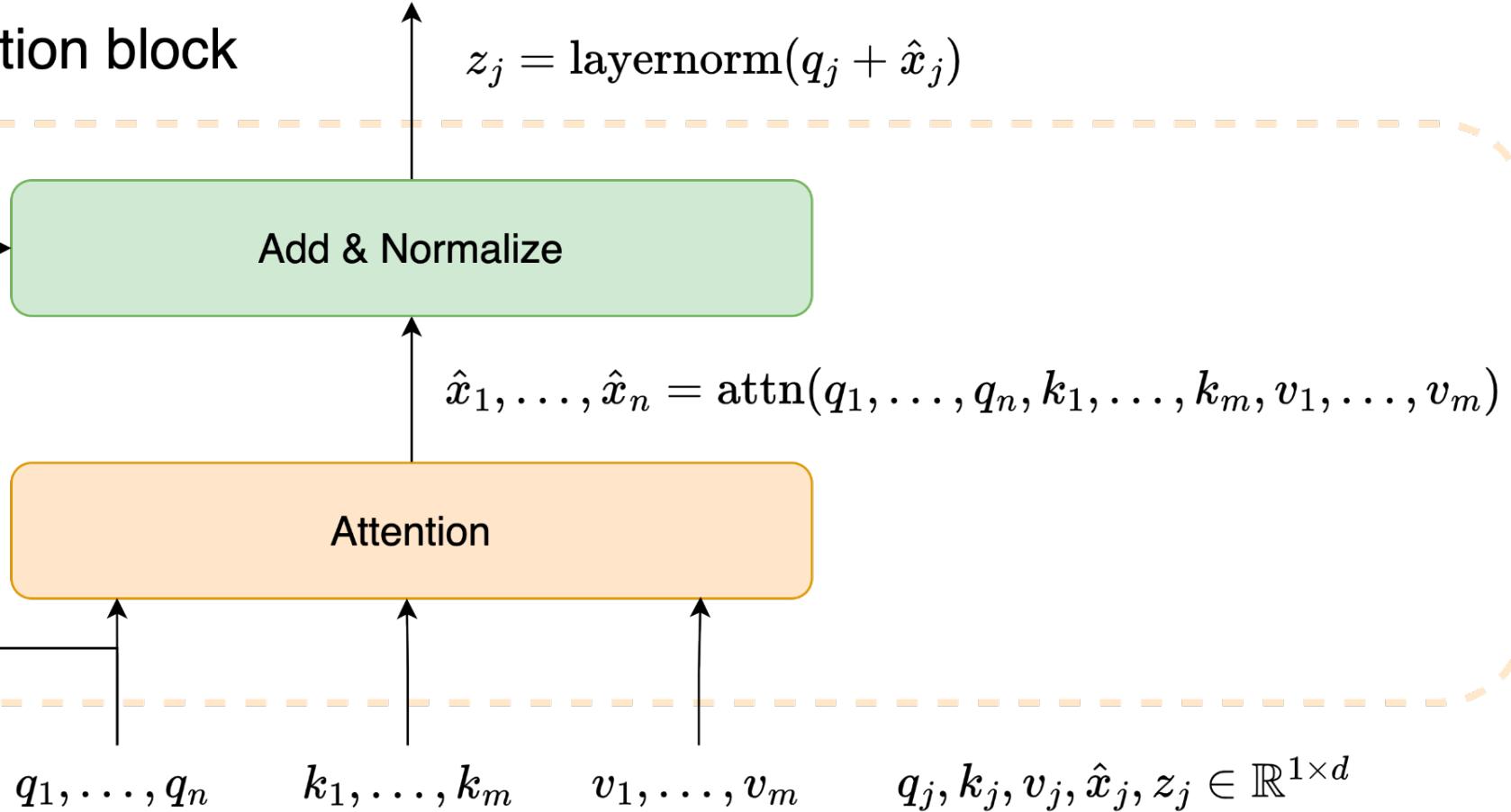
$$z_1, \dots, z_L = \text{attn-block}(x_1, \dots, x_L, x_1, \dots, x_L, x_1, \dots, x_L)$$

Attention block

$$x_1, \dots, x_L$$

$$x_j, z_j \in \mathbb{R}^{1 \times d}$$

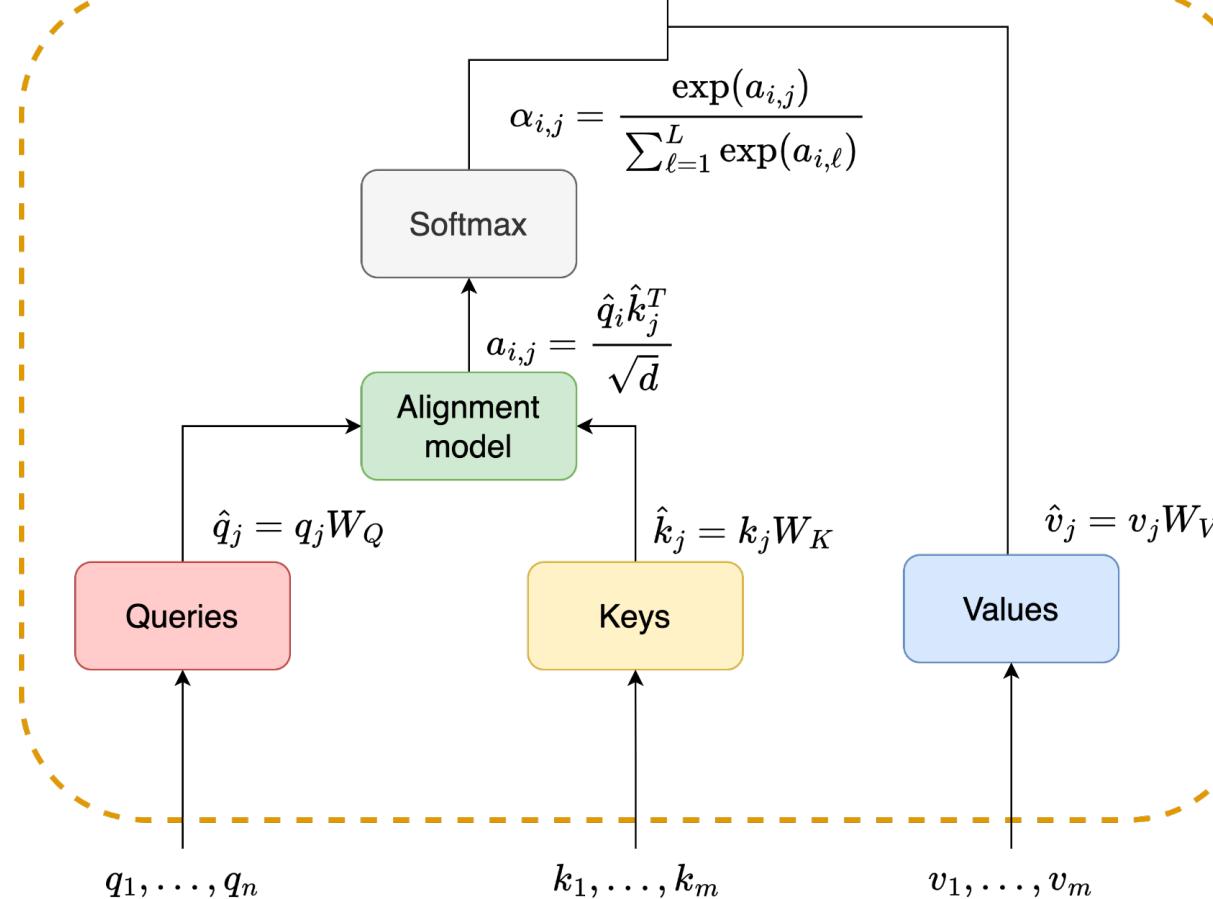
Attention block



Attention

$$z_i = \sum_{j=1}^L \alpha_{i,j} \hat{v}_j$$

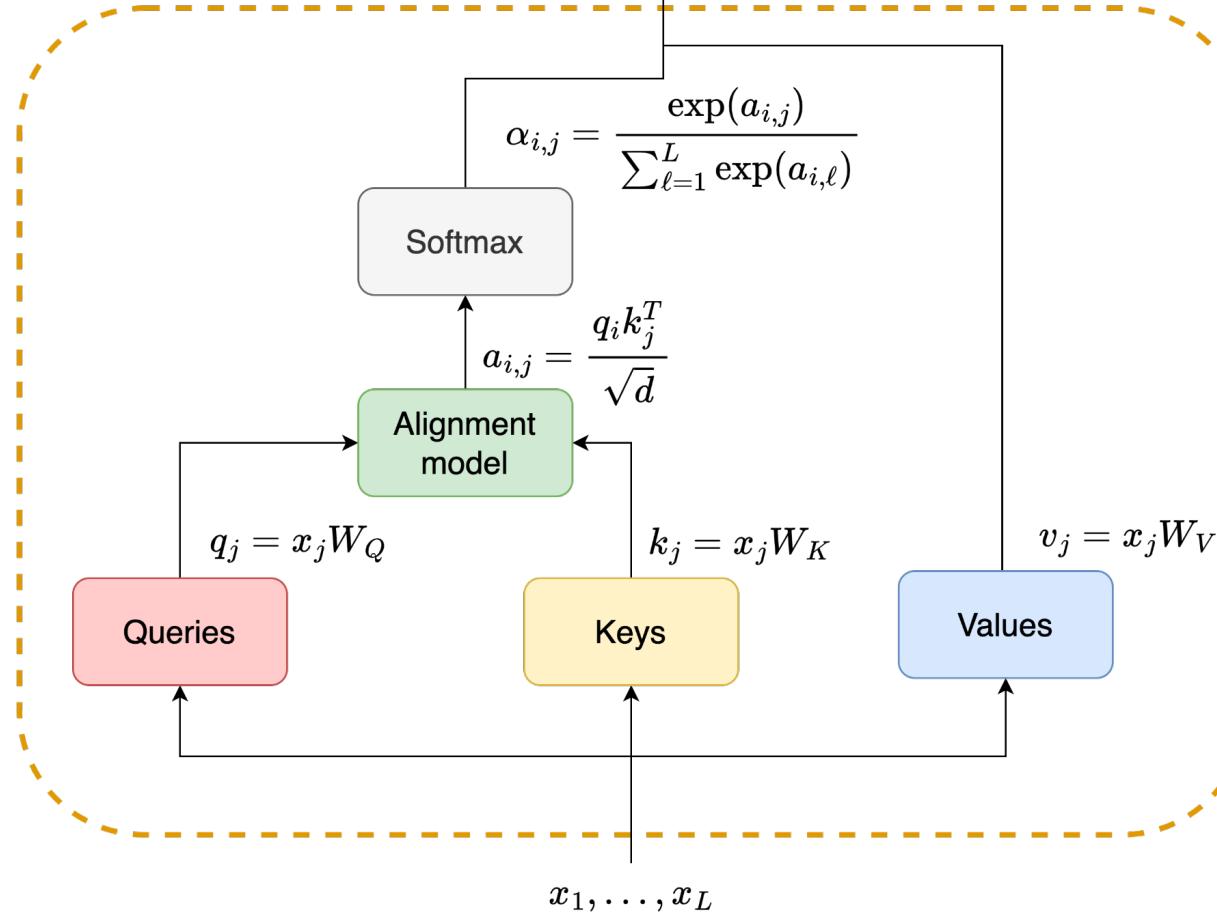
$$x_j, q_j, k_j, v_j, \hat{q}_j, \hat{k}_j, \hat{v}_j, z_j \in \mathbb{R}^{1 \times d}, \\ W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$$



Self-attention

$$z_i = \sum_{j=1}^L \alpha_{i,j} v_j$$

$$x_j, q_j, k_j, v_j, z_j \in \mathbb{R}^{1 \times d}, \\ W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$$



Зачем нужна номировка скалярных произведений?

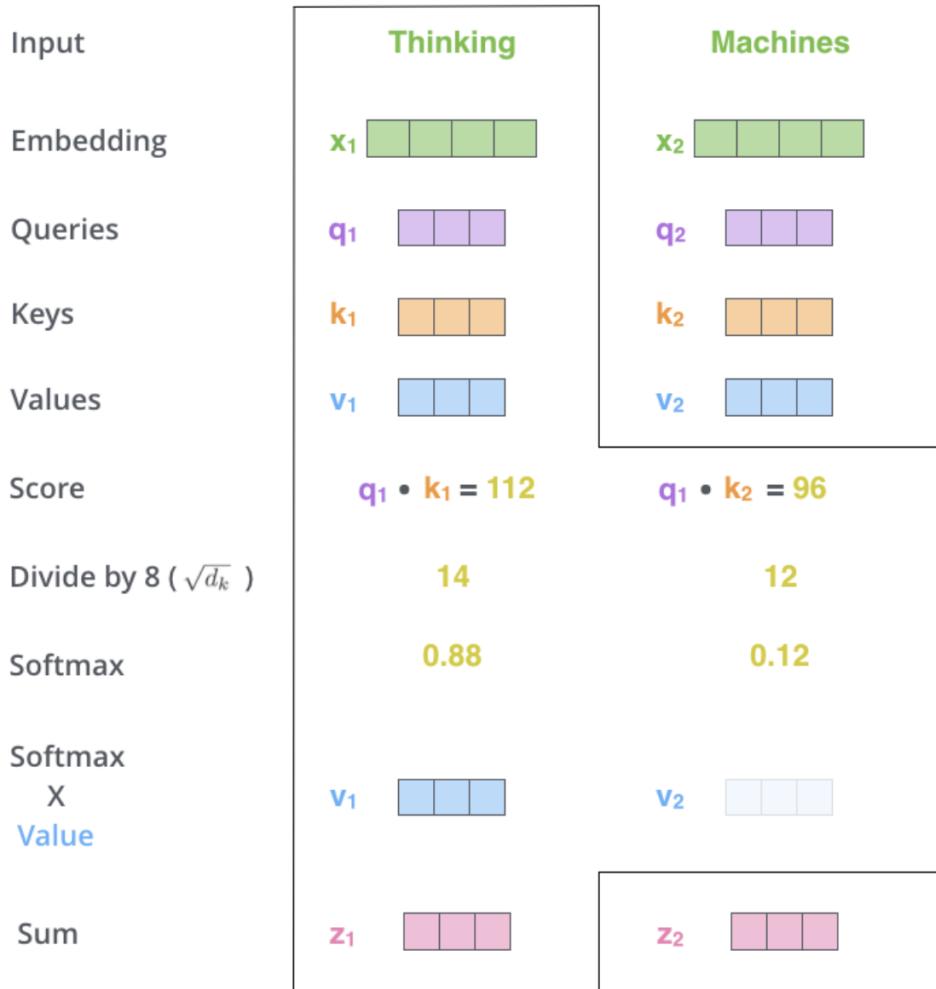
Пусть $q_1, \dots, q_d, k_1, \dots, k_d$ - независимые случайные величины с нулевыми средними и единичными дисперсиями, тогда

$$\mathbb{D} \left(\sum_{i=1}^d q_i k_i \right) = \sum_{i=1}^d \mathbb{D}(q_i k_i) = \sum_{i=1}^d \mathbb{E}(q_i^2 k_i^2) = \sum_{i=1}^d \mathbb{E}q_i^2 \mathbb{E}k_i^2 = \sum_{i=1}^d \mathbb{D}q_i \mathbb{D}k_i = d.$$

Тогда, очевидно, что

$$\mathbb{E} \left(\frac{\sum_{i=1}^d q_i k_i}{\sqrt{d}} \right) = 0, \quad \mathbb{D} \left(\frac{\sum_{i=1}^d q_i k_i}{\sqrt{d}} \right) = 1.$$

То есть при инициализации нейросети аргументы softmax не будут сильно отличаться друг от друга, и не будет проблем с насыщением.



Self-attention в матричном виде

Получаем матрицы Q, K, V:

$$\begin{matrix} \mathbf{X} \\ \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \mathbf{W^Q} \\ \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \mathbf{Q} \\ \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

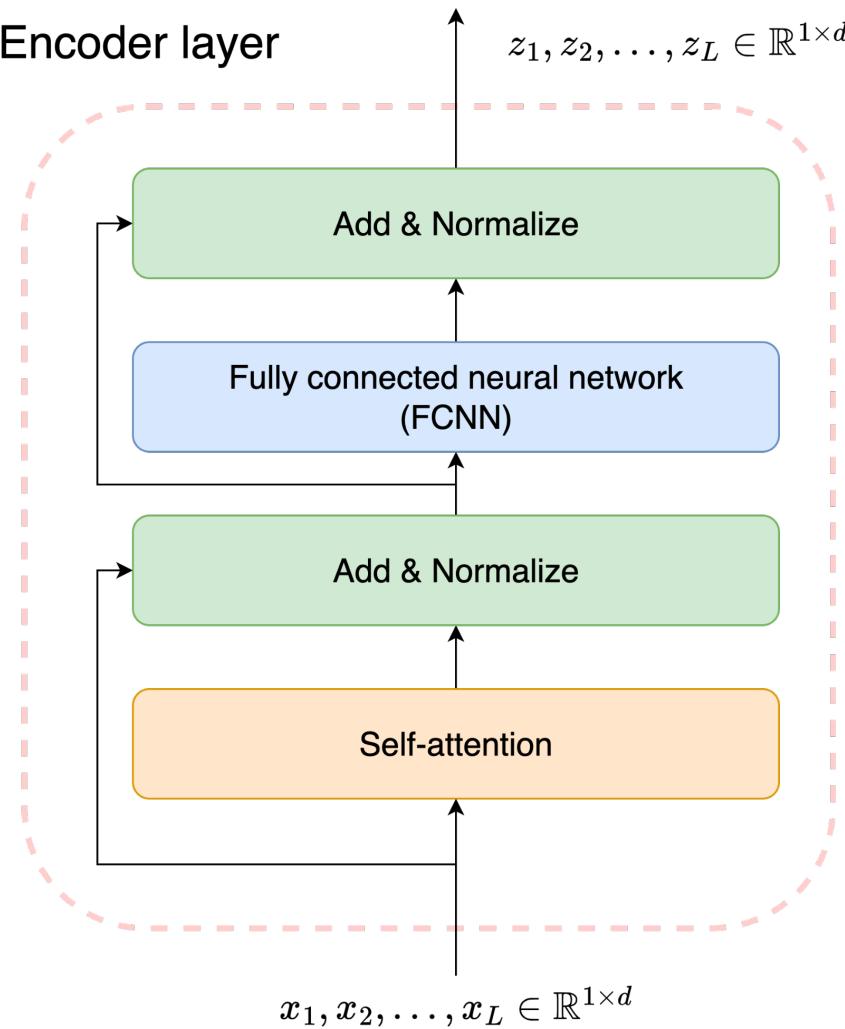
$$\begin{matrix} \mathbf{X} \\ \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \mathbf{W^K} \\ \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \mathbf{K} \\ \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} \mathbf{X} \\ \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \mathbf{W^V} \\ \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \mathbf{V} \\ \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

Считаем self-attention в матричной форме:

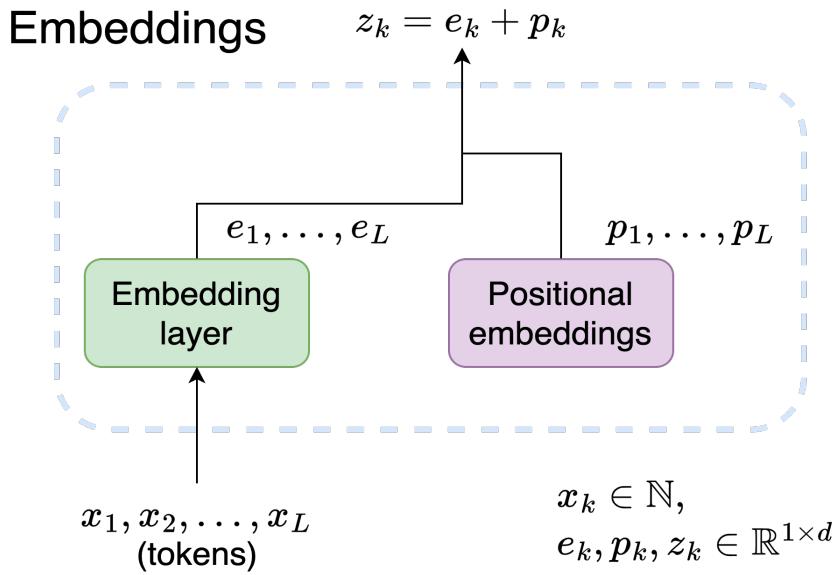
$$\text{softmax} \left(\frac{\mathbf{Q} \times \mathbf{K^T}}{\sqrt{d_k}} \right) \mathbf{V} = \mathbf{Z}$$

Encoder layer



Вычисление эмбеддингов

- Embedding layer – эмбеддинги токенов (учатся вместе с моделью). В статье размерность 512
- Positional embeddings – эмбеддинги позиций. Учатся вместе с моделью, либо задаются аналитически
- Без positional embeddings никак не будет учитываться порядок токенов (как в мешке слов)



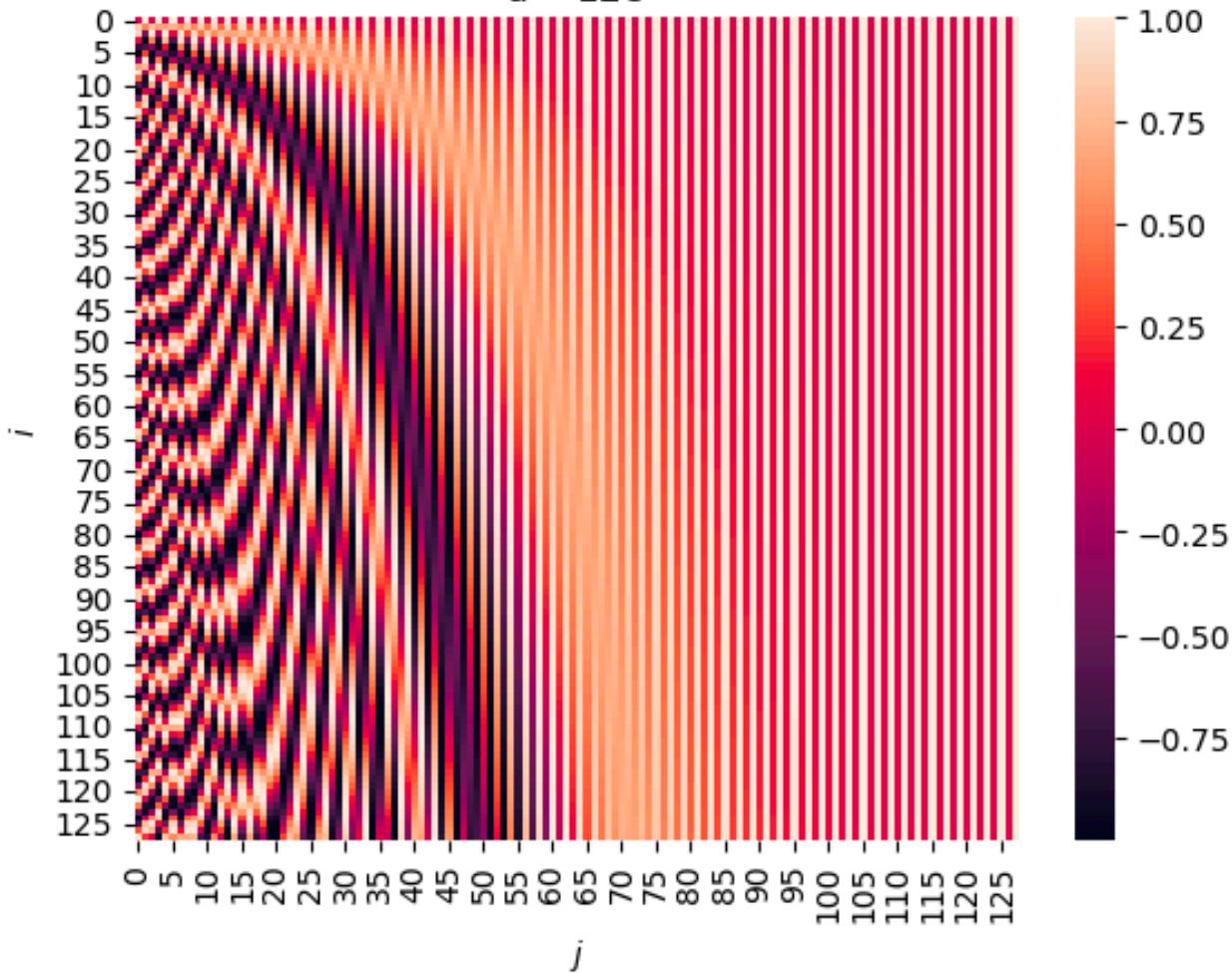
Аналитическое задание positional embeddings

Обозначения:

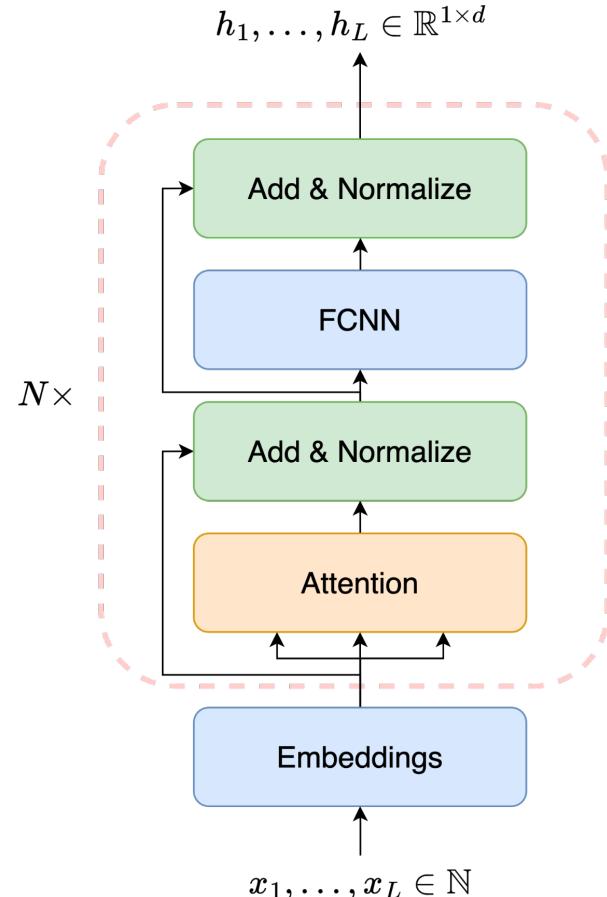
- d - размерность вектора эмбеддингов,
- $p_{i,k}$ - k -я компонента i -го вектора начиная с нуля).

В статье "Attention is all you need":

- $d = 512$,
- $p_{i,2j} = \sin(i/10000^{2j/d})$,
- $p_{i,2j+1} = \cos(i/10000^{2j/d})$.

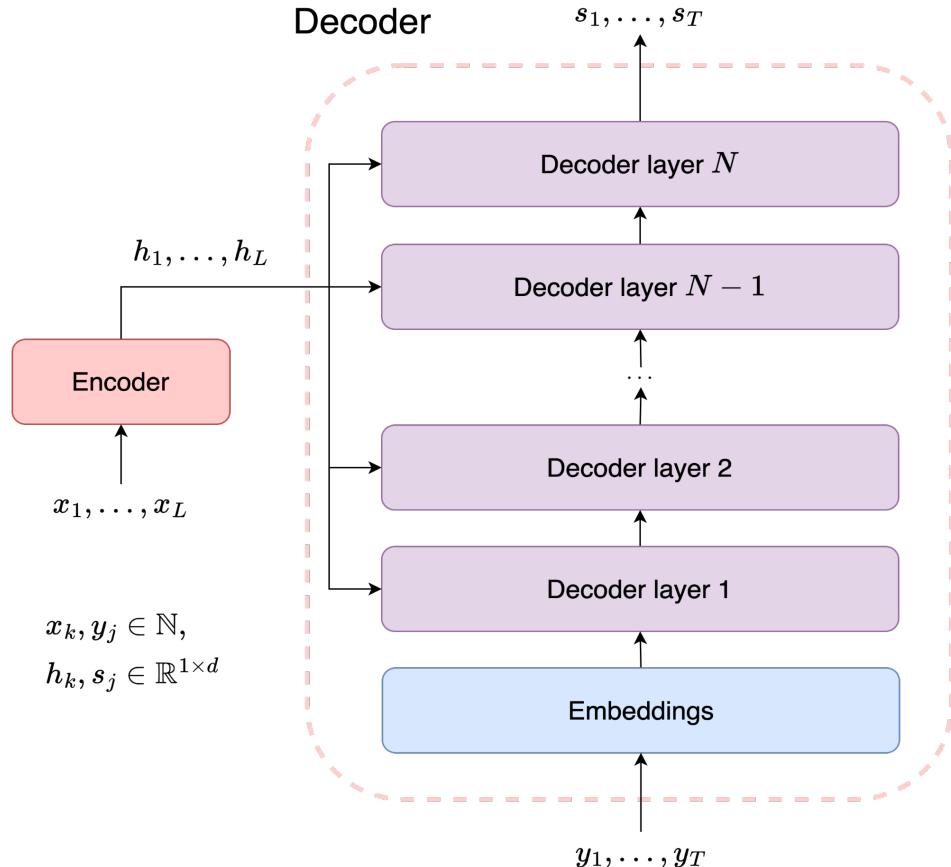
$d = 128$ 

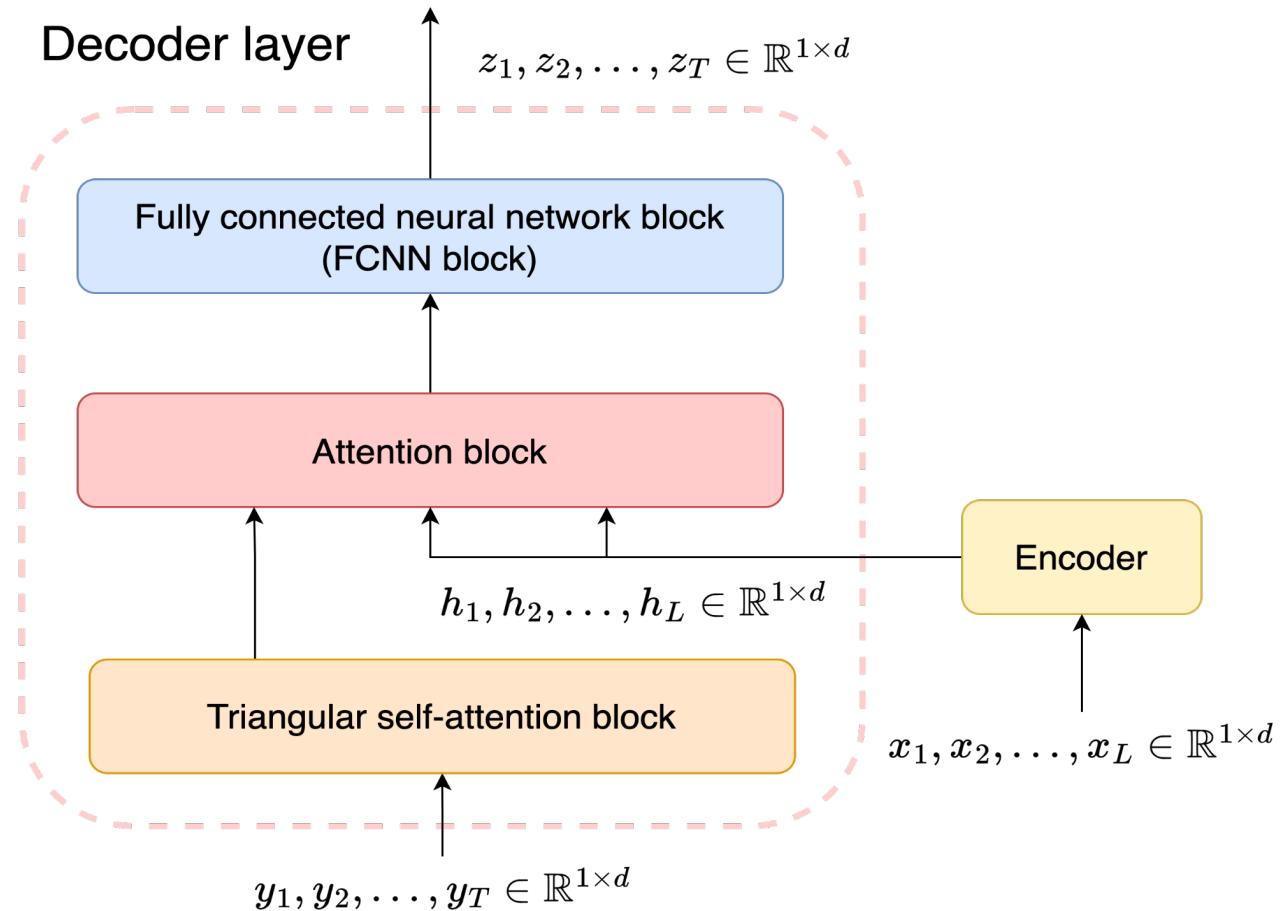
Encoder



Устройство декодировщика

- Принимает на вход уже сгенерированные токены (или только токен `<bos>` изначально) и выходы (полученные векторы) кодировщика
- Превращает токены в эмбеддинги (слой `Embeddings`)
- Применяет последовательно N однотипных преобразований (decoder layer)





$$M \in \{0, 1\}^{L \times L}$$

j

i

1	0	0	1	0	1	1	0	0	1	1	0
1	1	0	0	1	0	0	1	0	0	0	1
1	0	1	0	1	0	0	1	0	0	1	0
1	1	1	1	0	0	1	0	0	1	1	0
0	0	1	0	1	0	0	0	0	0	1	0
1	1	1	1	0	1	0	0	1	0	0	0
0	0	1	0	1	1	1	1	0	1	1	1
1	1	0	1	1	0	1	1	0	0	0	0
0	1	1	1	0	1	0	1	1	0	1	1
1	1	0	0	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	0	1	1	1	0
1	0	1	1	0	1	1	0	0	1	1	1

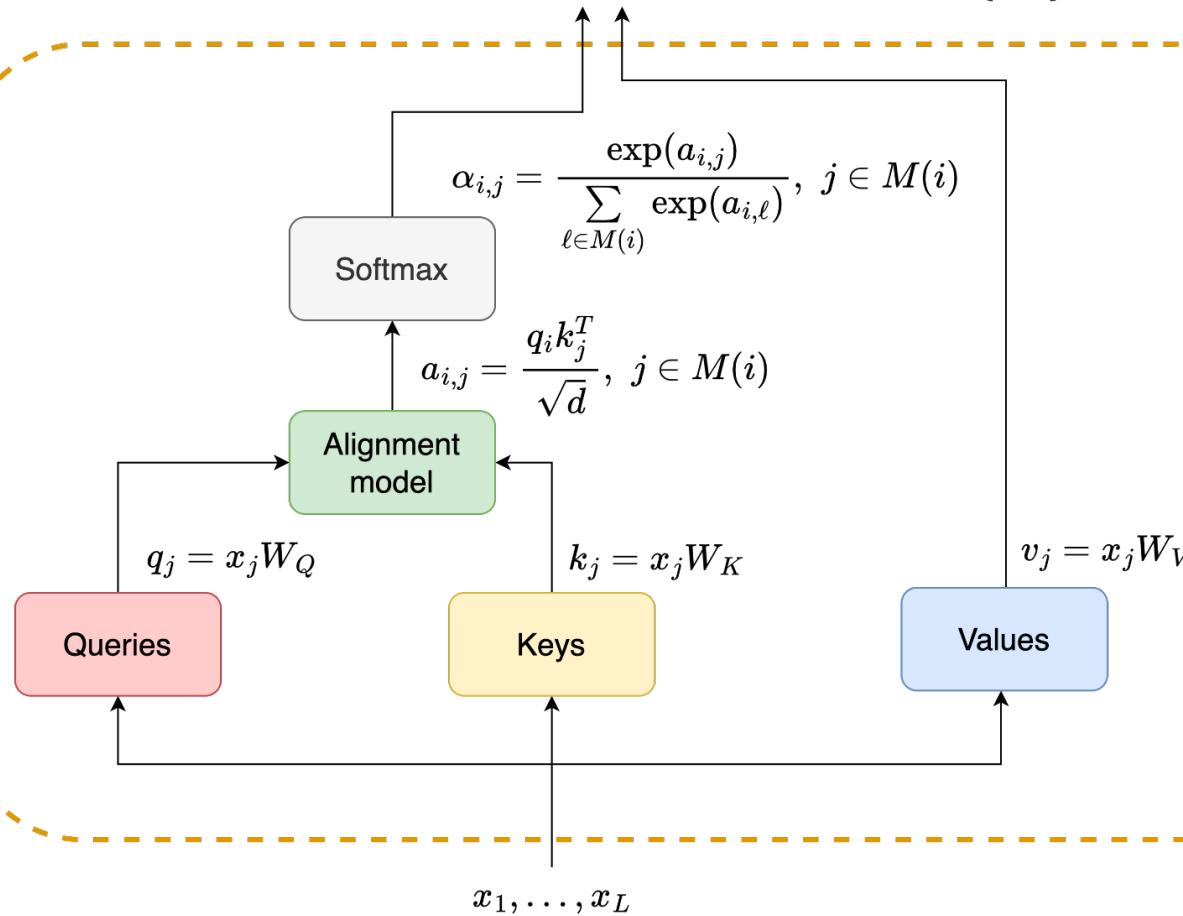
Маска внимания (Attention mask)

i-ый элемент последовательности может "смотреть" не на все элементы, а только на те, у которых $M_{i,j} = 1$

Masked self-attention

$$z_i = \sum_{j \in M(i)} \alpha_{i,j} v_j$$

$$x_j, q_j, k_j, v_j, z_j \in \mathbb{R}^{1 \times d}, \quad W_Q, W_K, W_V \in \mathbb{R}^{d \times d}, \\ M \in \{0, 1\}^{L \times L}, \quad M(i) = \{j | M_{i,j} = 1\}$$

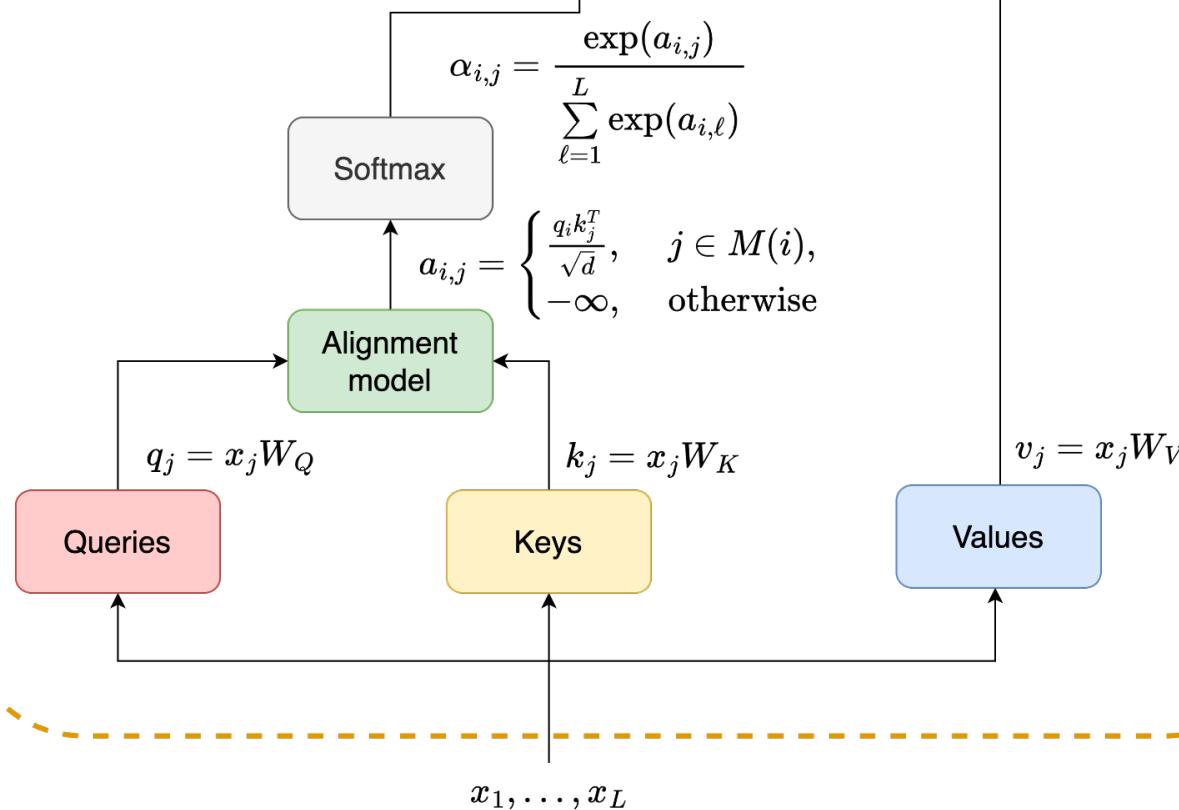


Masked self-attention (simplified view)

$$z_i = \sum_{j=1}^L \alpha_{i,j} v_j$$

$$x_j, q_j, k_j, v_j, z_j \in \mathbb{R}^{1 \times d}, \quad W_Q, W_K, W_V \in \mathbb{R}^{d \times d},$$

$$M \in \{0, 1\}^{L \times L}, \quad M(i) = \{j | M_{i,j} = 1\}$$



Популярные маски внимания

Full attention mask

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

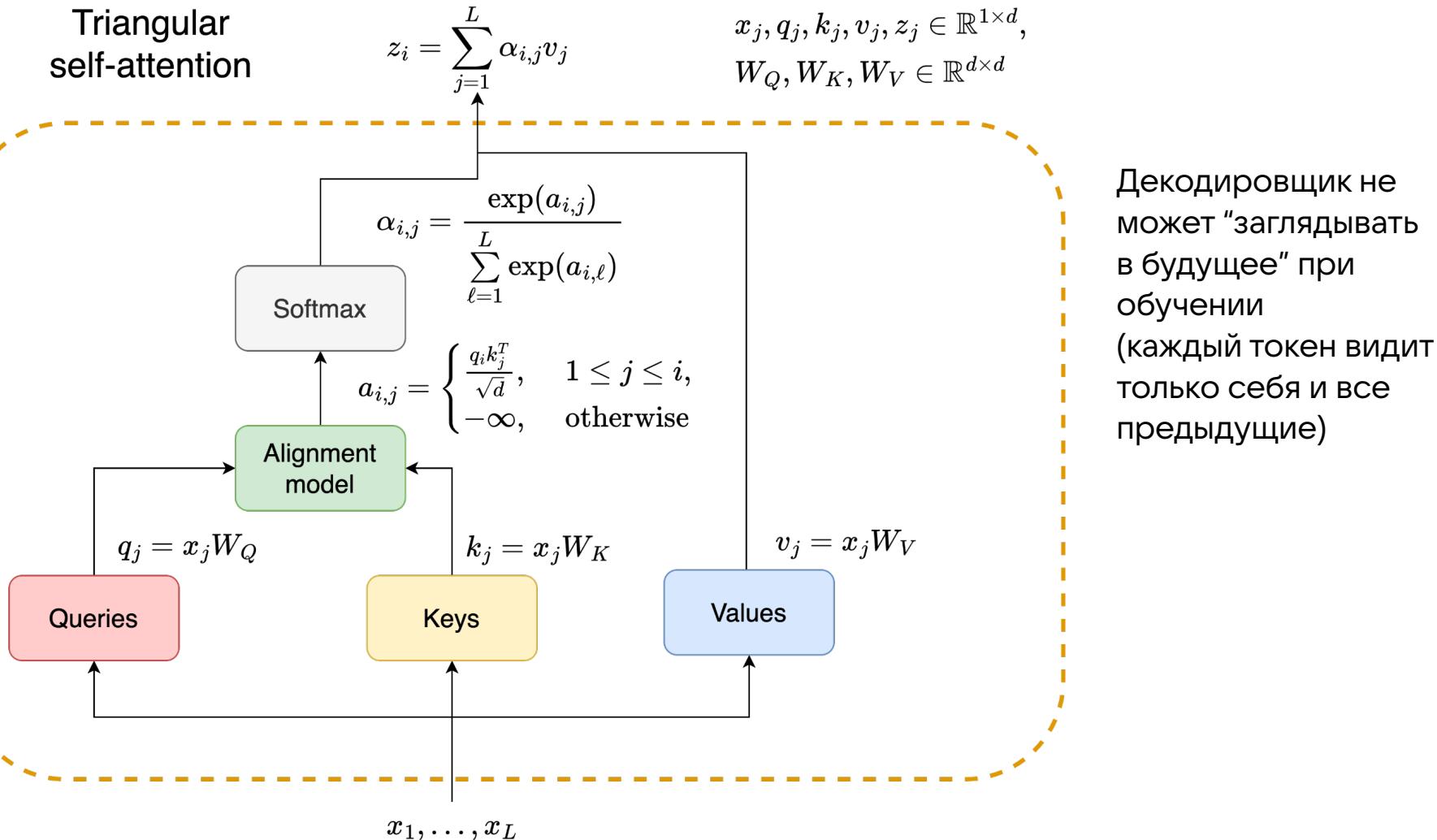
Sliding window
attention mask

1	1			
1	1	1		
	1	1	1	
		1	1	1
			1	1

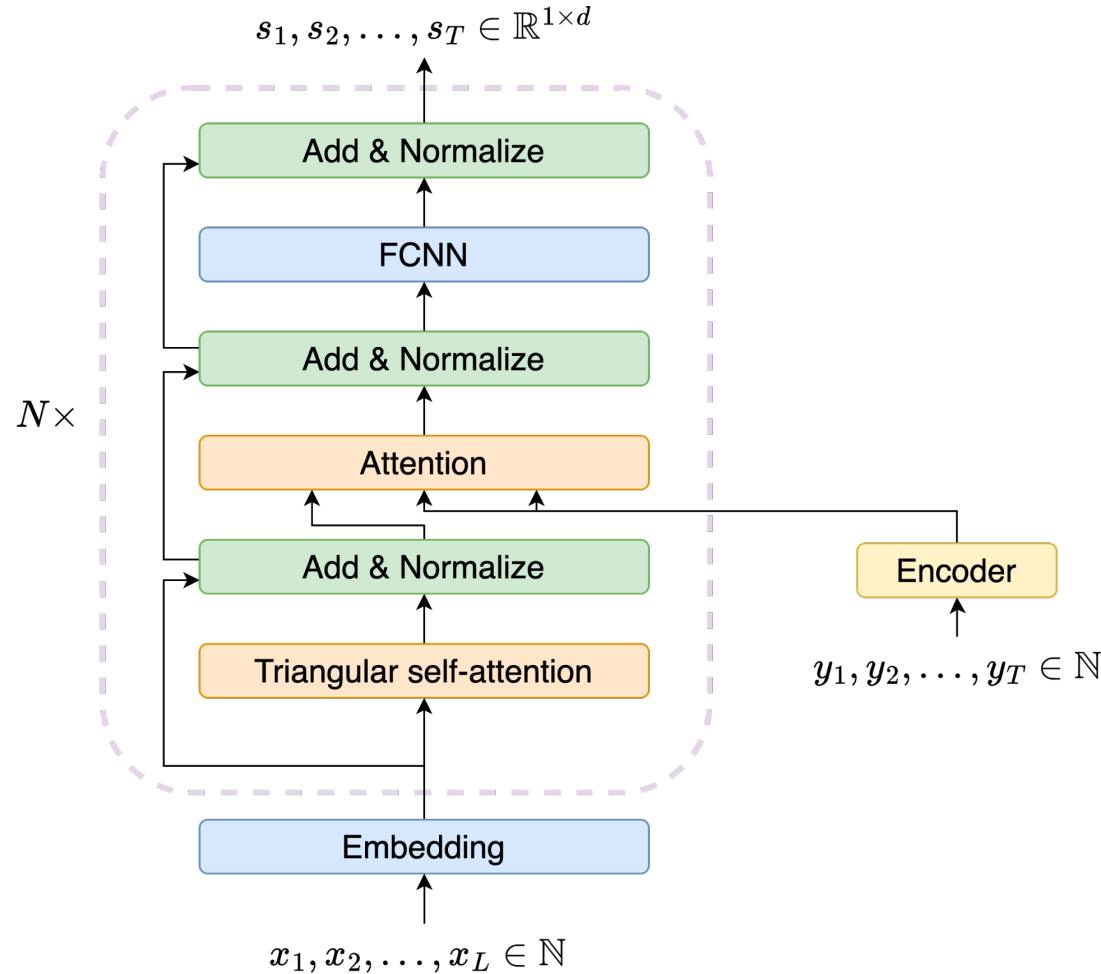
Triangular
attention mask

1				
1	1			
1	1	1		
1	1	1	1	
1	1	1	1	1

Triangular self-attention



Decoder



Предсказание следующего токена

$$\mathbb{P}(y_{k+1} = j | y_1, \dots, y_k) = p_k[j]$$

$$p_k = \text{softmax}(\hat{s}_k)$$

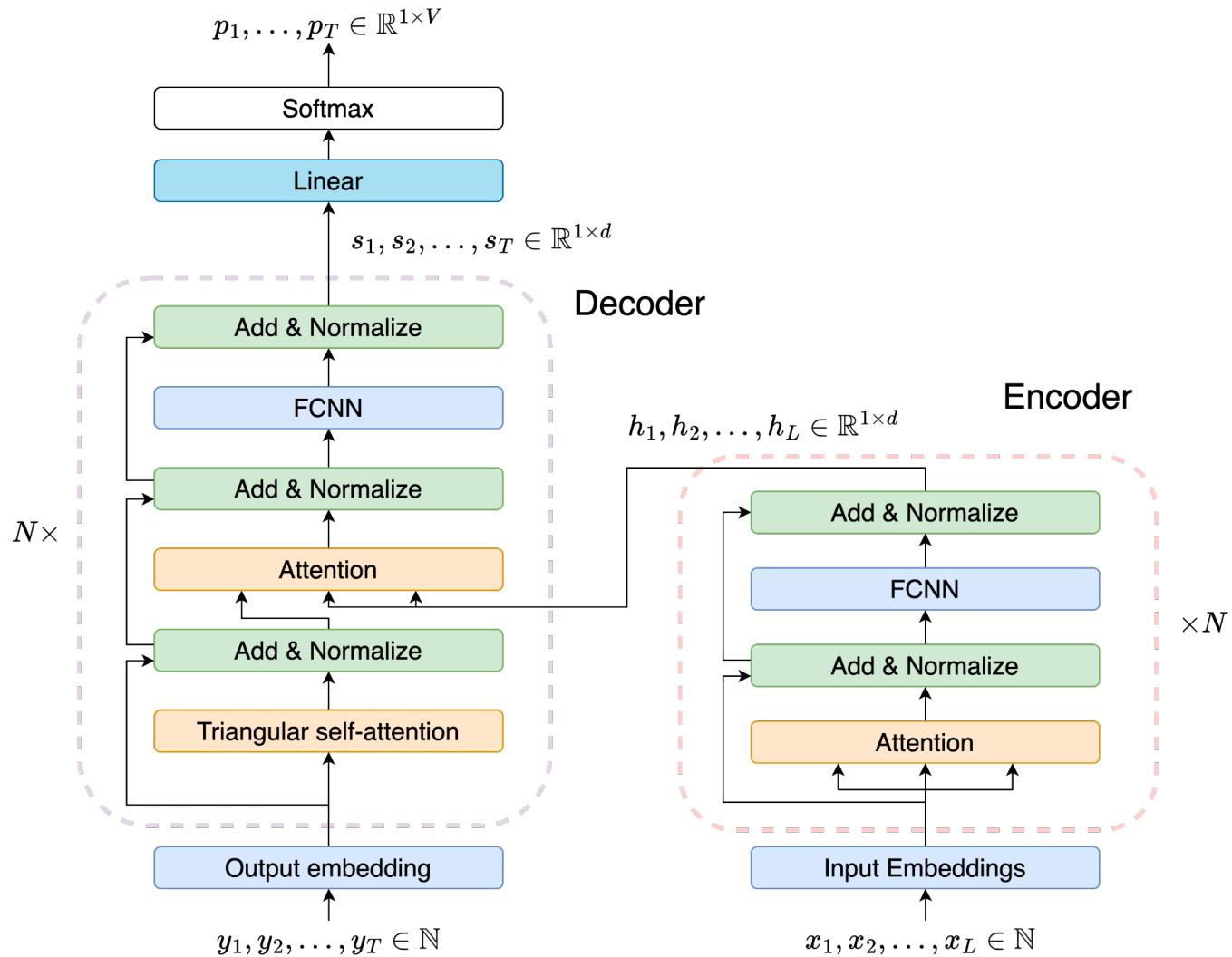
Softmax

$$\hat{s}_k = s_k W, \quad W \in \mathbb{R}^{d \times V}$$

Linear

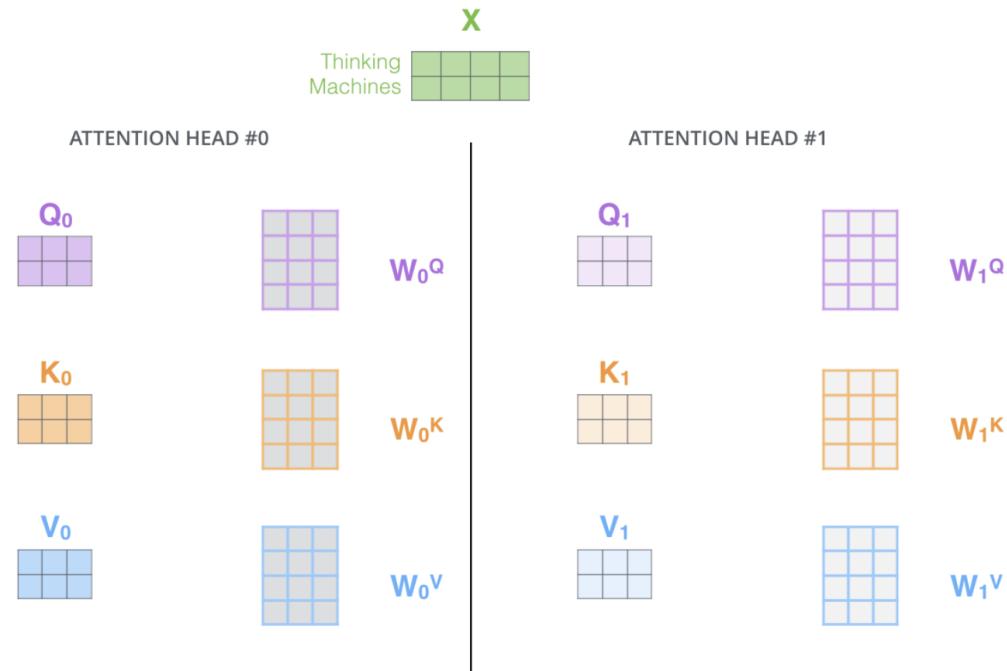
V - размер словаря

$$s_1, \dots, s_T \in \mathbb{R}^{1 \times d}$$

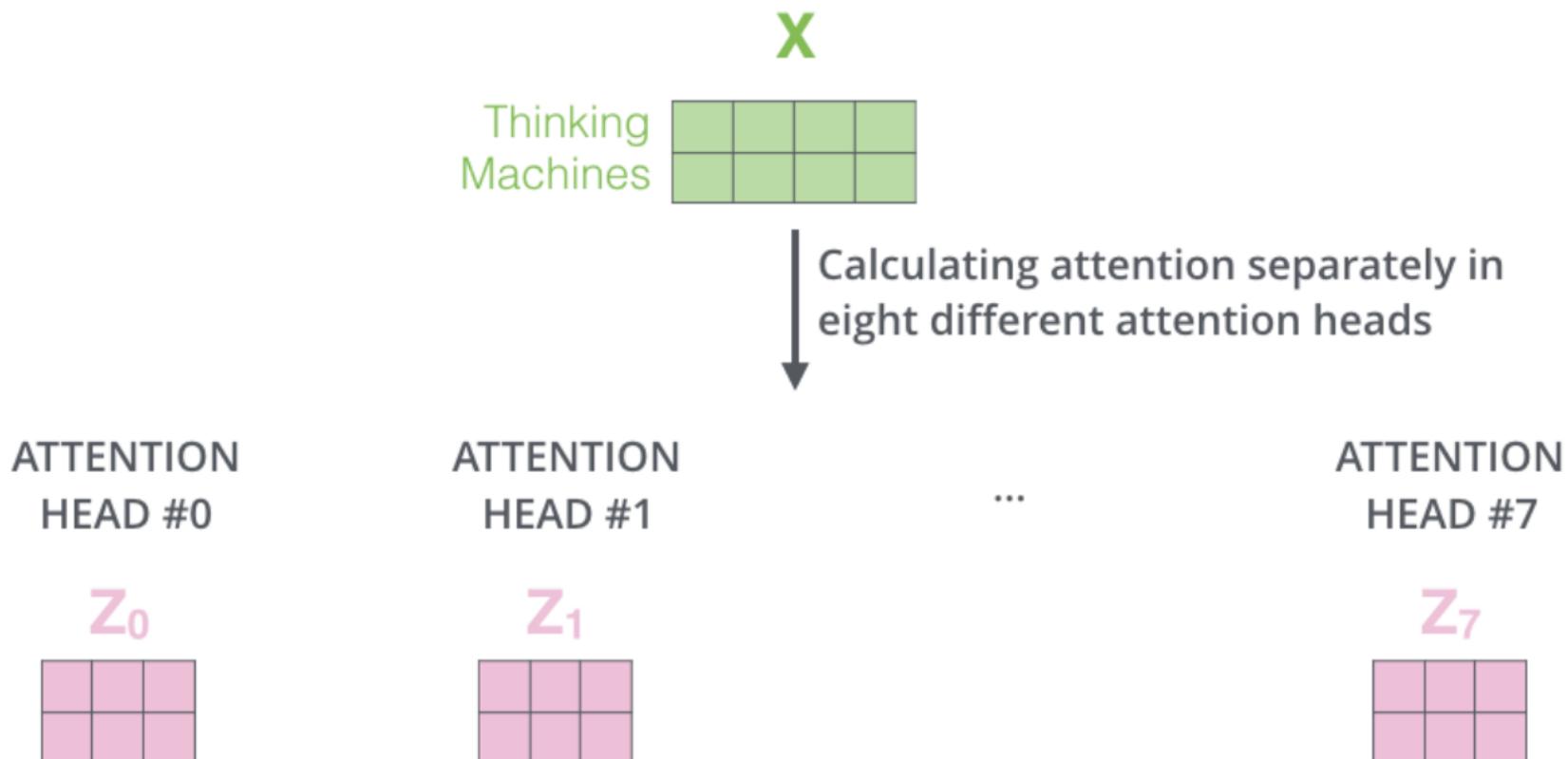


Multi-head attention

- Можно учить сразу несколько механизмов self-attention с разными линейными преобразованиями, а потом их объединять.
- Мотивация: каждый из механизмов за счет случайной инициализации матриц будет учиться фокусироваться на различных смысловых аспектах.



Multi-head attention



Multi-head attention

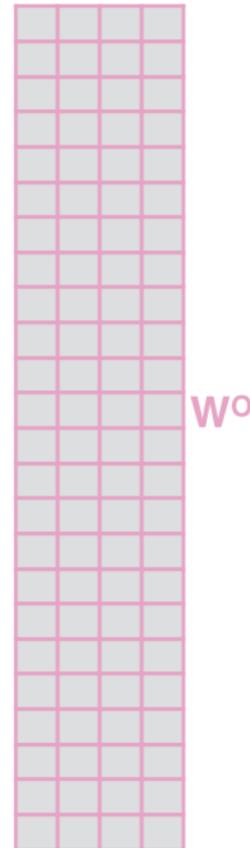
1) Concatenate all the attention heads



Multi-head attention

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

X



Multi-head attention

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

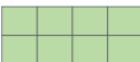
$$= \begin{matrix} Z \\ \hline \text{---} \\ \begin{matrix} \boxed{} & \boxed{} & \boxed{} & \boxed{} \\ \boxed{} & \boxed{} & \boxed{} & \boxed{} \end{matrix} \end{matrix}$$

Multi-head attention

- 1) This is our input sentence* each word*
- 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

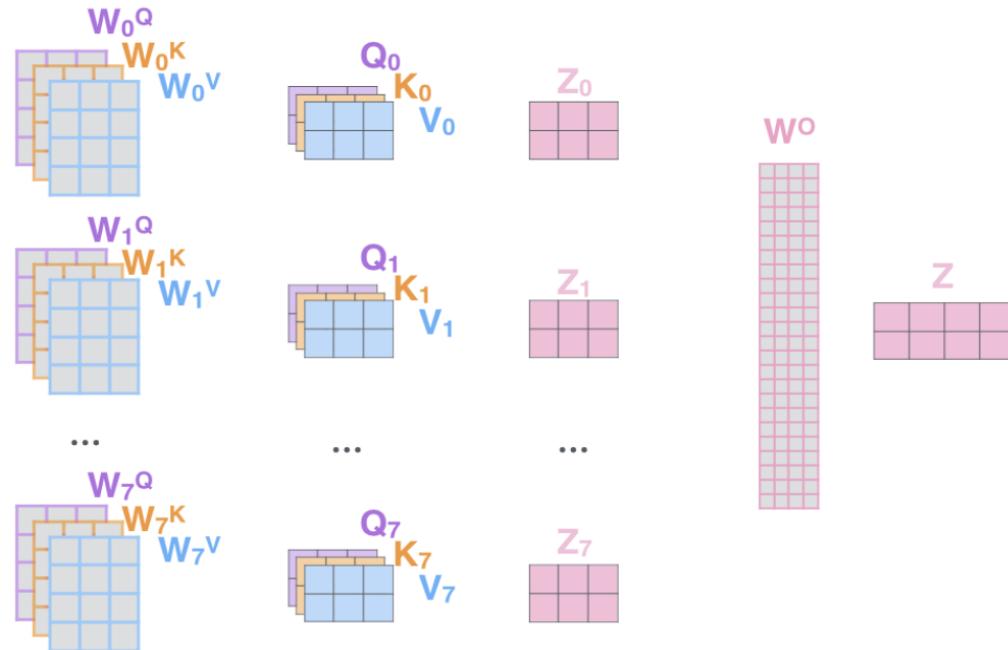
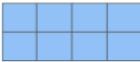
Thinking
Machines

X



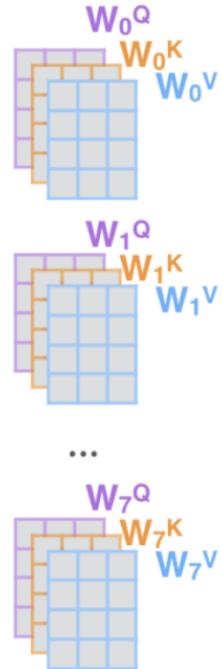
* In all encoders other than #0, we don't need embedding.
We start directly with the output of the encoder right below this one

R

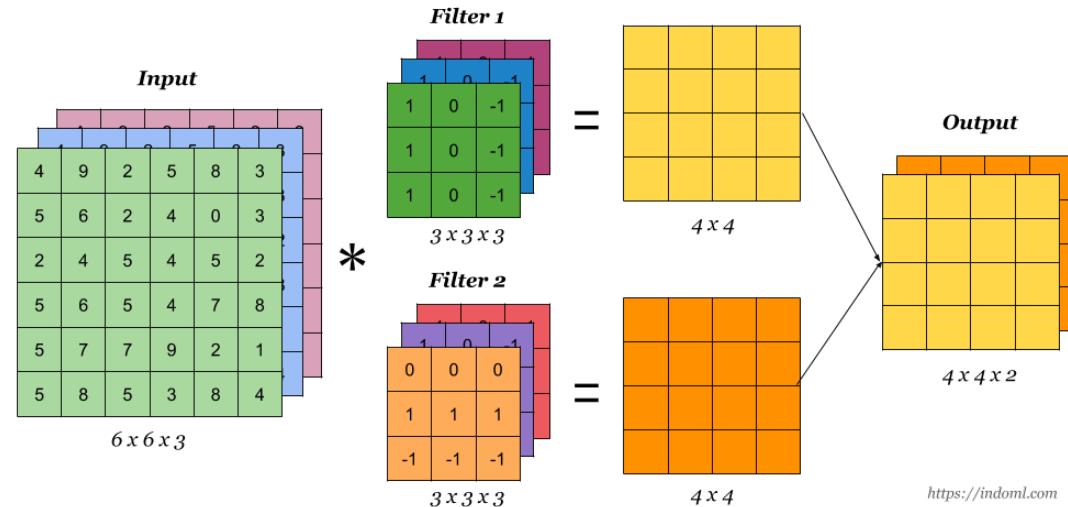


Аналогия с несколькими каналами в CNN

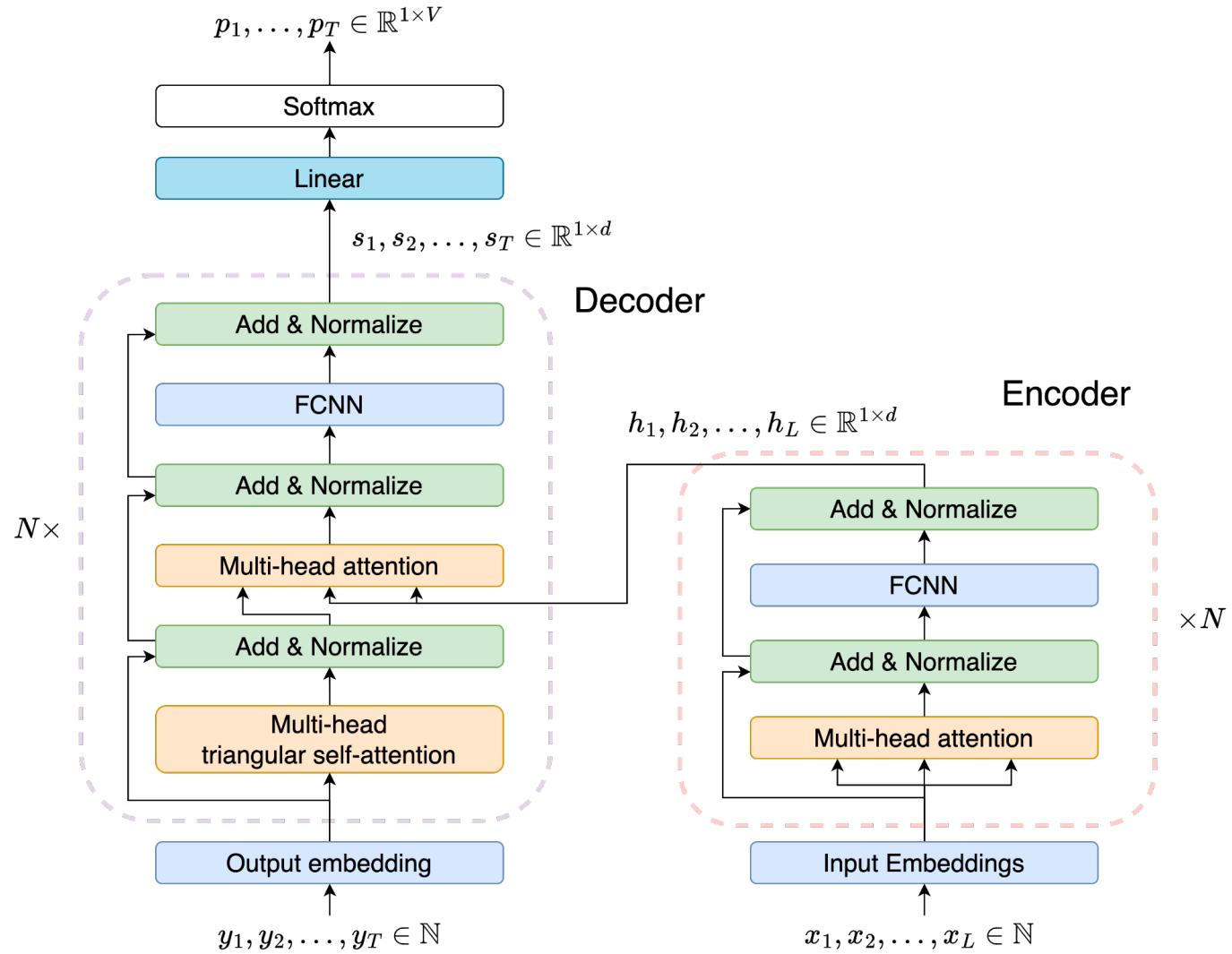
Несколько голов в МНА-слое:



Несколько фильтров в сверточном слое:



<https://indoml.com>



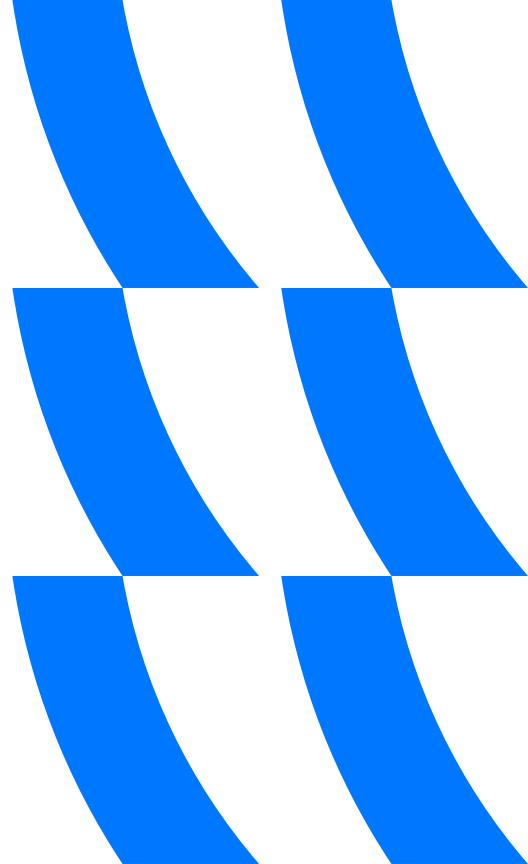
Что сегодня изучили?

- Повторили
 - Полносвязные нейронные сети
 - Рекуррентные нейронные сети
 - Слой эмбеддингов
- Задача sequence to sequence
 - Постановка задачи
 - Как решали задачу с помощью рекуррентных нейросетей, и какие были проблемы
- Механизм внимания (attention) в рекуррентных нейросетях
 - Bahdanau attention
 - Luong attention
- На что похож механизм внимания?
 - Аналогия из информационного поиска
- Архитектура трансформера
 - Механизм self-attention
 - Архитектура кодировщика
 - Архитектура декодировщика



Что еще почитать?

- Статья [Attention is all you need](#)
- [The illustrated transformer](#)
- [The annotated transformer](#)



Спасибо за внимание!

Владимир Макаренко, старший разработчик-исследователь

