

Основы глубинного обучения

Лекция 9

Рекуррентные модели

Евгений Соколов

esokolov@hse.ru

НИУ ВШЭ, 2024

Рекуррентные модели

Рекуррентные сети (RNN)

- Последовательность: $x_1, x_2, \dots, x_n, \dots$
- Читаем слева направо
- h_t — накопленная информация после чтения t элементов (вектор)

Рекуррентные сети (RNN)

- Последовательность: $x_1, x_2, \dots, x_n, \dots$
- x_i — либо one-hot вектор, либо векторное представление (word2vec, fasttext, ...)

Рекуррентные сети (RNN)

- Последовательность: $x_1, x_2, \dots, x_n, \dots$
- Читаем слева направо
- h_t — накопленная информация после чтения t элементов (вектор)
- $h_t = f(W_{xh}x_t + W_{hh}h_{t-1})$
- Если хотим что-то выдавать на каждом шаге: $o_t = f_o(W_{ho}h_t)$

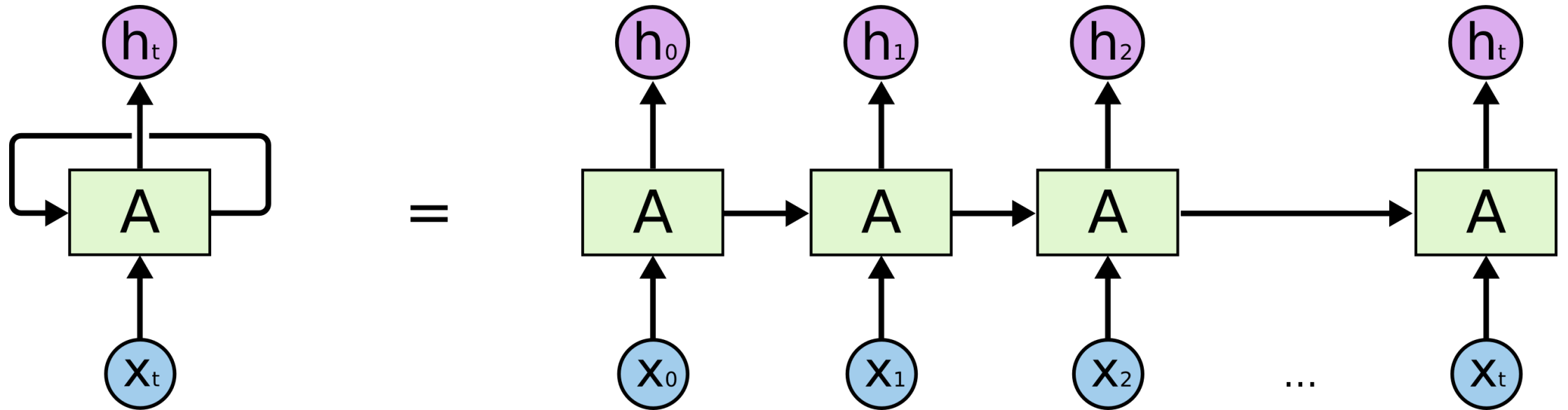
Рекуррентные сети (RNN)

- Типичный случай: $o_t \in \mathbb{R}^N$
- N — размер словаря
- То есть предсказываем вероятность того, что здесь стоит конкретное слово
- Предсказываем следующее слово

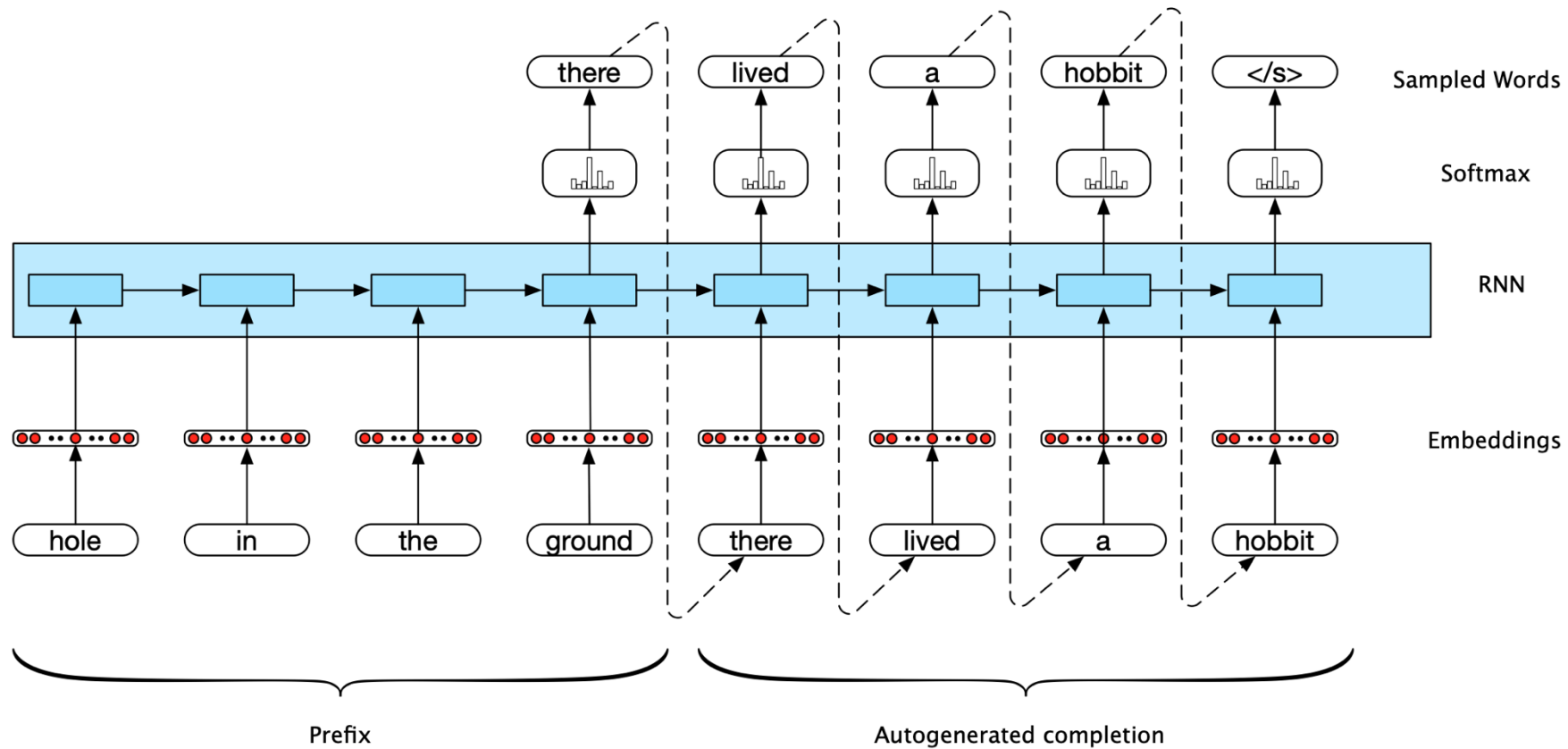
Рекуррентные сети (RNN)

- Типичный случай: $o_t \in \mathbb{R}^N$
- N — количество частей речи
- POS tagging

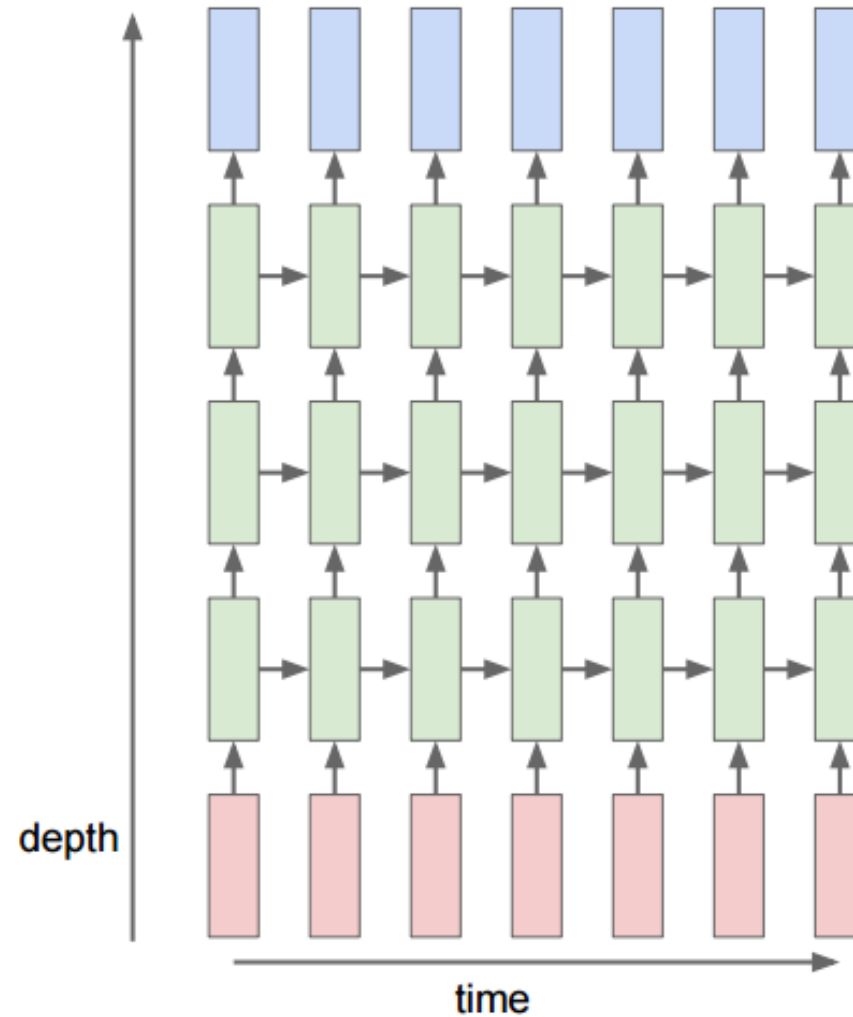
Рекуррентные сети (RNN)



Рекуррентные сети (RNN)



Можно делать многослойные RNN



Примеры

PANDARUS:

Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and
my fair nudes begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.

Примеры

For $\bigoplus_{n=1,\dots,m} \mathcal{L}_{m,n}$ where $\mathcal{L}_{m,n} = 0$, hence we can find a closed subset \mathcal{H} in \mathcal{H} and any sets \mathcal{F} on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ?? . Hence we obtain a scheme S and any open subset $W \subset U$ in $\text{Sh}(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of \mathcal{X}' , and \mathcal{T}_i is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and \mathcal{F}_p exists and let \mathcal{F}_i be a presheaf of \mathcal{O}_X -modules on \mathcal{C} as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf}^{opp}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \mapsto (U, \text{Spec}(A))$$

is an open subset of X . Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S .

Proof. See discussion of sheaves of sets. \square

The result for prove any open covering follows from the less of Example ?? . It may replace S by $X_{spaces, \acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar} , see Descent, Lemma ?? . Namely, by Lemma ?? we see that R is geometrically regular over S .

Lemma 0.1. Assume (3) and (3) by the construction in the description.

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{\text{Proj}}_X(\mathcal{A}) = \text{Spec}(B)$ over U compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X, \mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S . Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) f is locally of finite type. Since $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X . But given a scheme U and a surjective étale morphism $U \rightarrow X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ be the scheme X over S at the schemes $X_i \rightarrow X$ and $U = \lim_i X_i$. \square

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\dots,0}$.

Lemma 0.2. Let X be a locally Noetherian scheme over S , $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \mathcal{A}_2$ works.

Lemma 0.3. In Situation ?? . Hence we may assume $\mathfrak{q}' = 0$.

Proof. We will use the property we see that \mathfrak{p} is the next functor (??). On the other hand, by Lemma ?? we see that

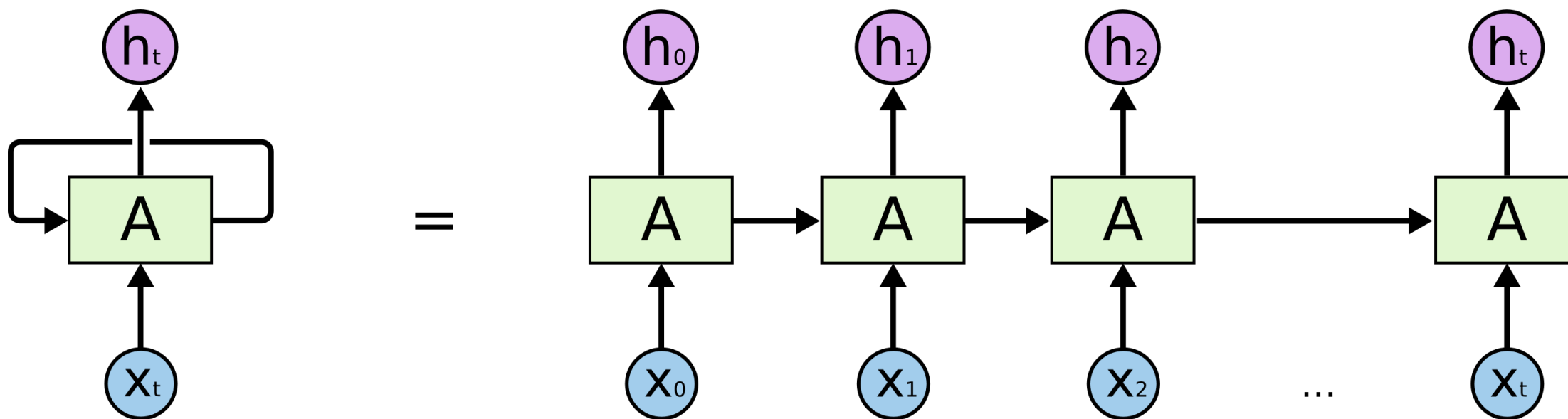
$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F -algebra where δ_{n+1} is a scheme over S . \square

Примеры

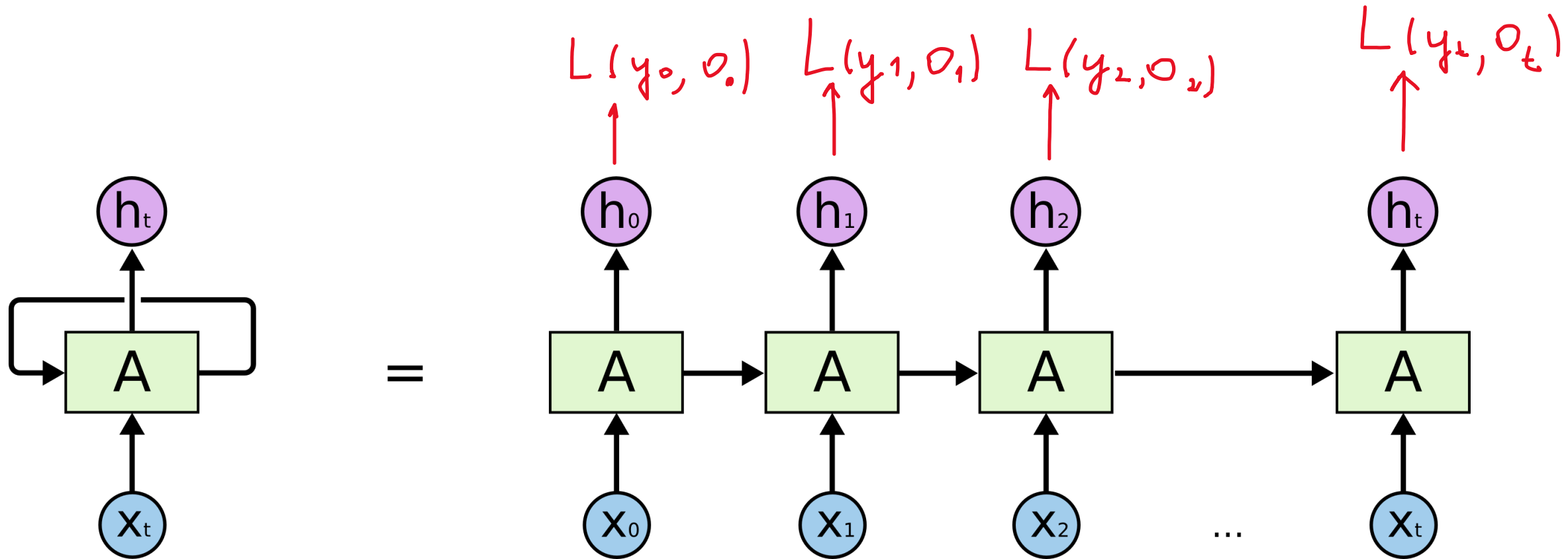
```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i++];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
        }
    }
    rw->name = "Getjbbregs";
    bprm_self_clearl(&iv->version);
    regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
    return segtable;
}
```

Развёртка RNN



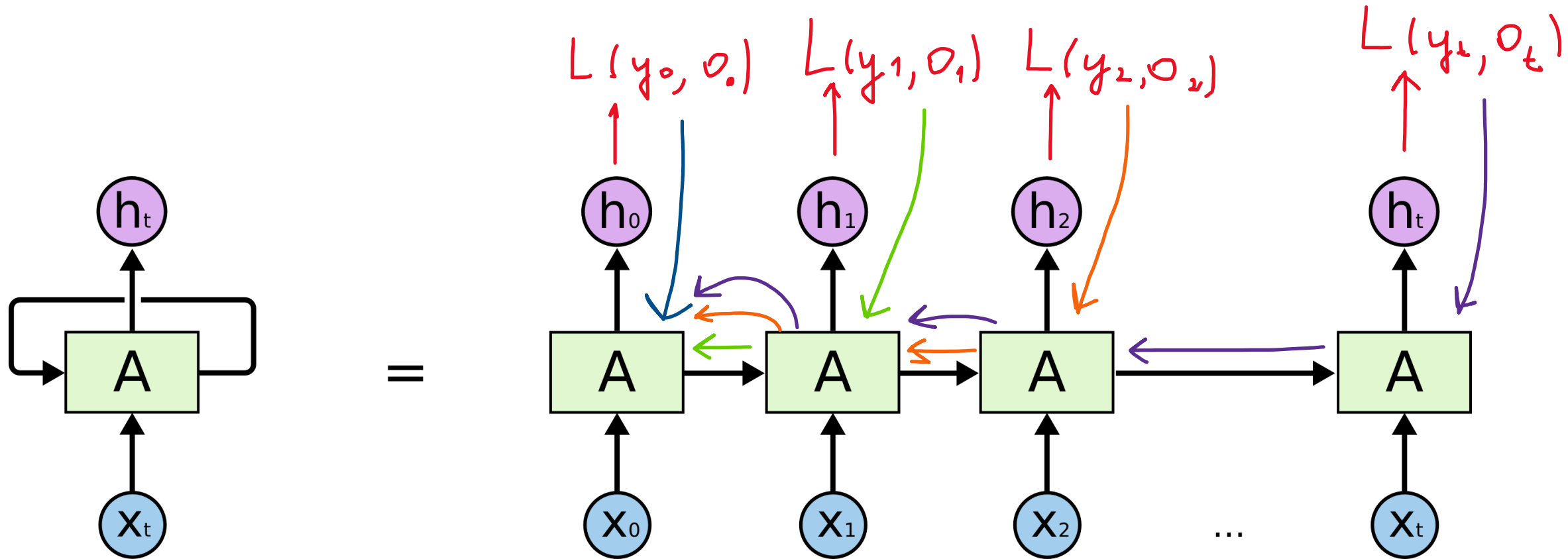
- $h_t = f(W_{xh}x_t + W_{hh}h_{t-1})$
- $o_t = f_o(W_{ho}h_t)$

Backpropagation Through Time (BPTT)



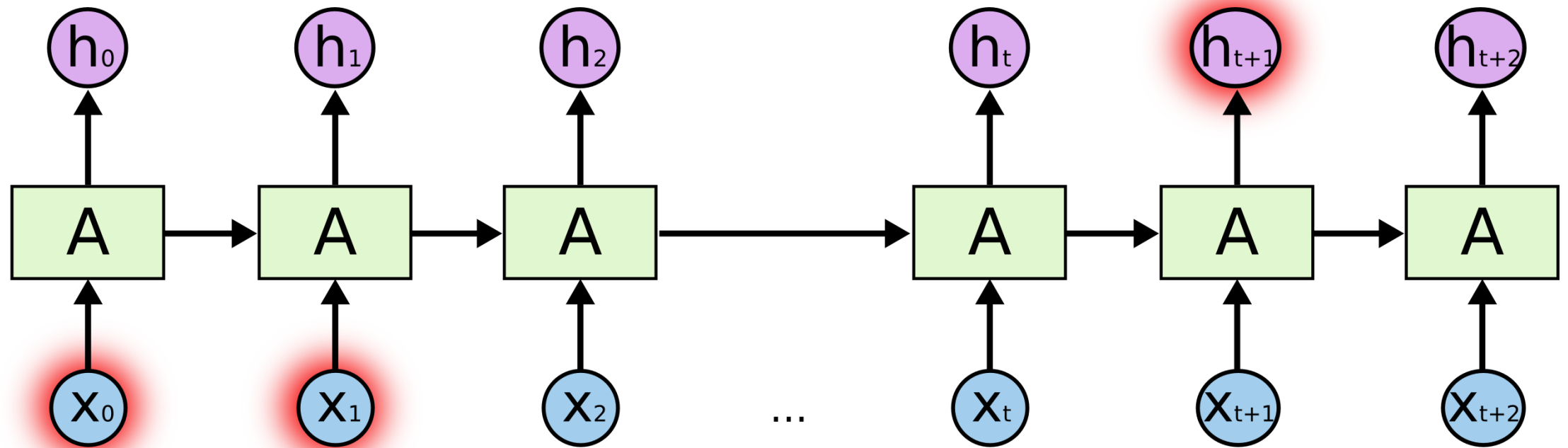
- $h_t = f(W_{xh}x_t + W_{hh}h_{t-1})$
- $o_t = f_o(W_{ho}h_t)$

Backpropagation Through Time (BPTT)



- $h_t = f(W_{xh}x_t + W_{hh}h_{t-1})$
- $o_t = f_o(W_{ho}h_t)$

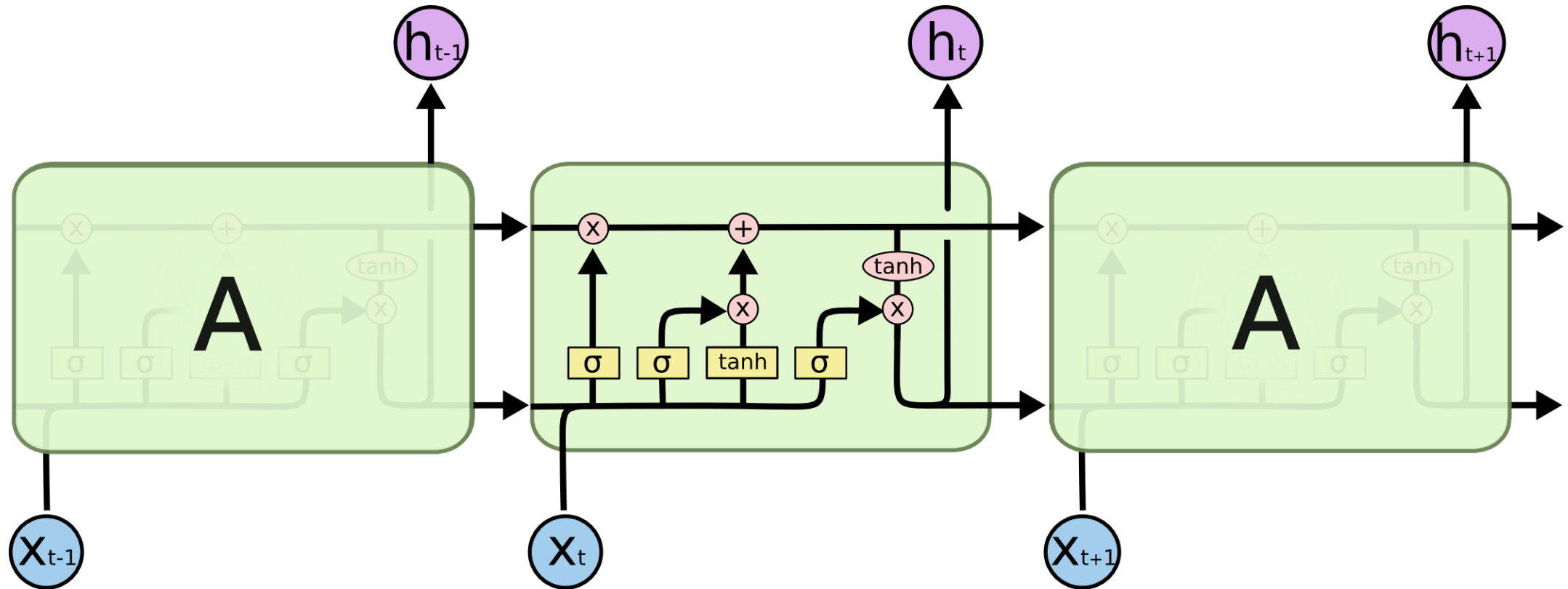
Backpropagation Through Time (BPTT)



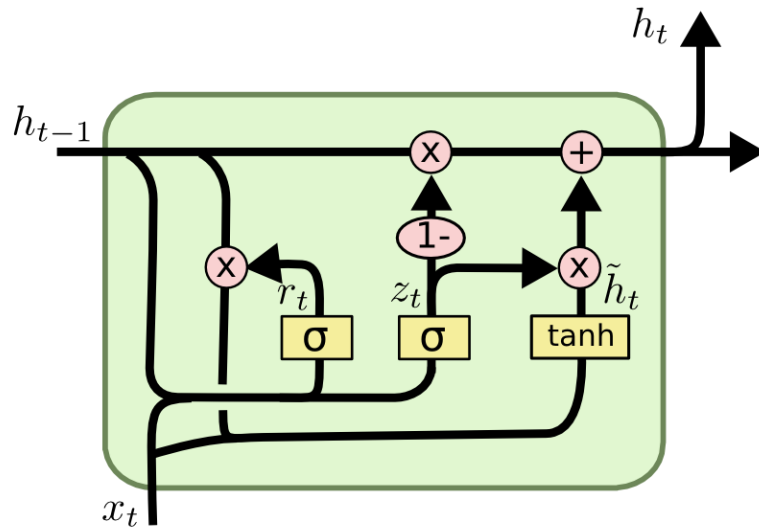
Проблемы с градиентами

- Сигнал теряется по мере прохождения
- Не факт, что получится обучить зависимость финального вектора h_n от первых слов в тексте

LSTM (Long Short-Term Memory)



LSTM (Long Short-Term Memory)



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

LSTM (Long Short-Term Memory)

- Позволяет предыдущему состоянию перейти в текущее без домножений на матрицы
- Модель сможет «протаскивать» информацию из начала текста в конец

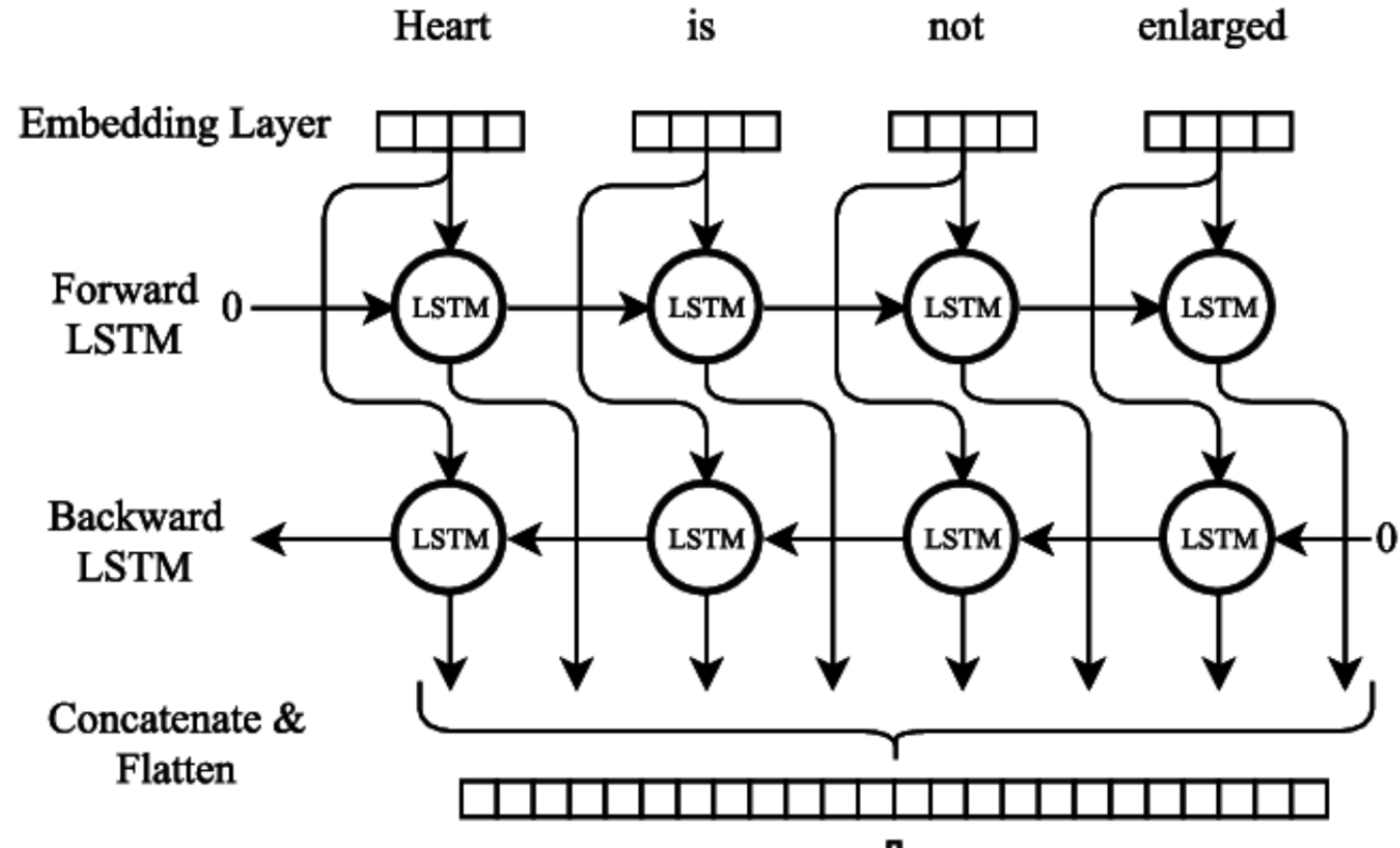
Рекуррентные сети (RNN)

- Типичный случай: $o_t \in \mathbb{R}^N$
- N — количество частей речи
- POS tagging

Bidirectional LSTM

- Почему мы определяем часть речи только по предыдущим словам?
- Будем смотреть и на следующие слова

Bidirectional LSTM



Bidirectional LSTM

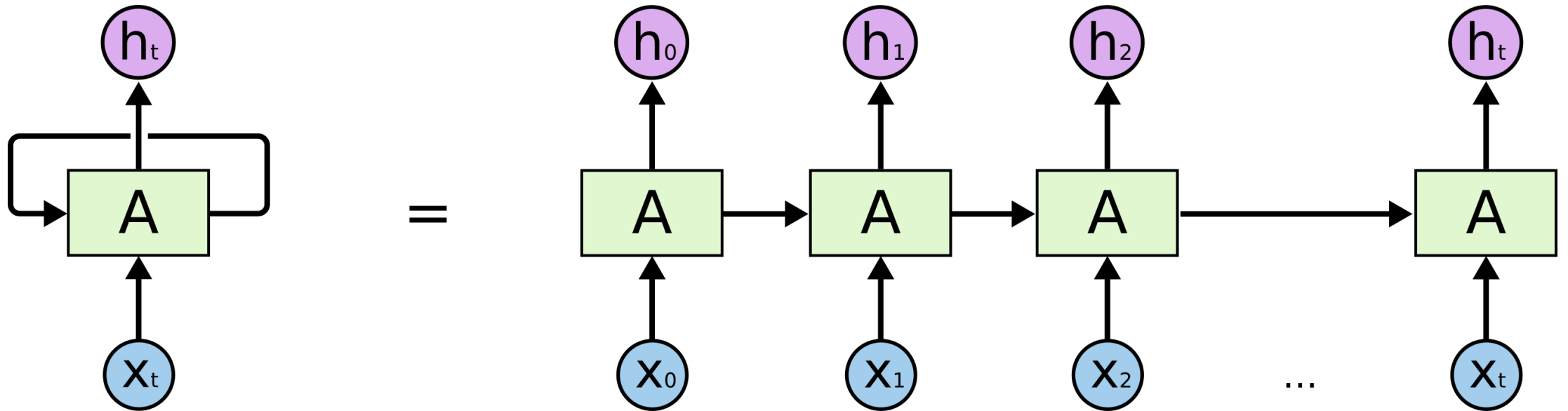
- Предсказание для слова строится по скрытым состояниям, учитывающим весь контекст

Seq2seq

Sequence to sequence

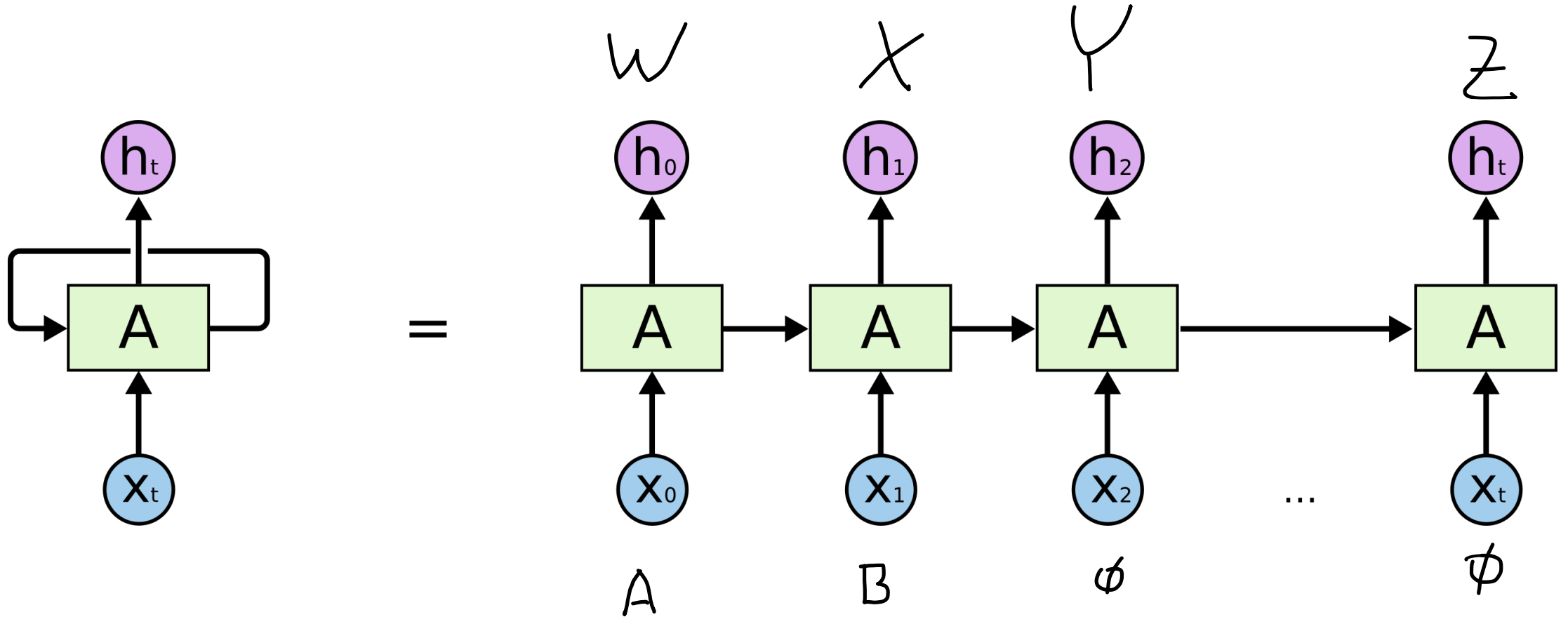
- Машинный перевод
- Суммаризация текста
- Генерация комментариев к коду
- Математические преобразования
- Смена стиля текста

Seq2seq Machine Translation

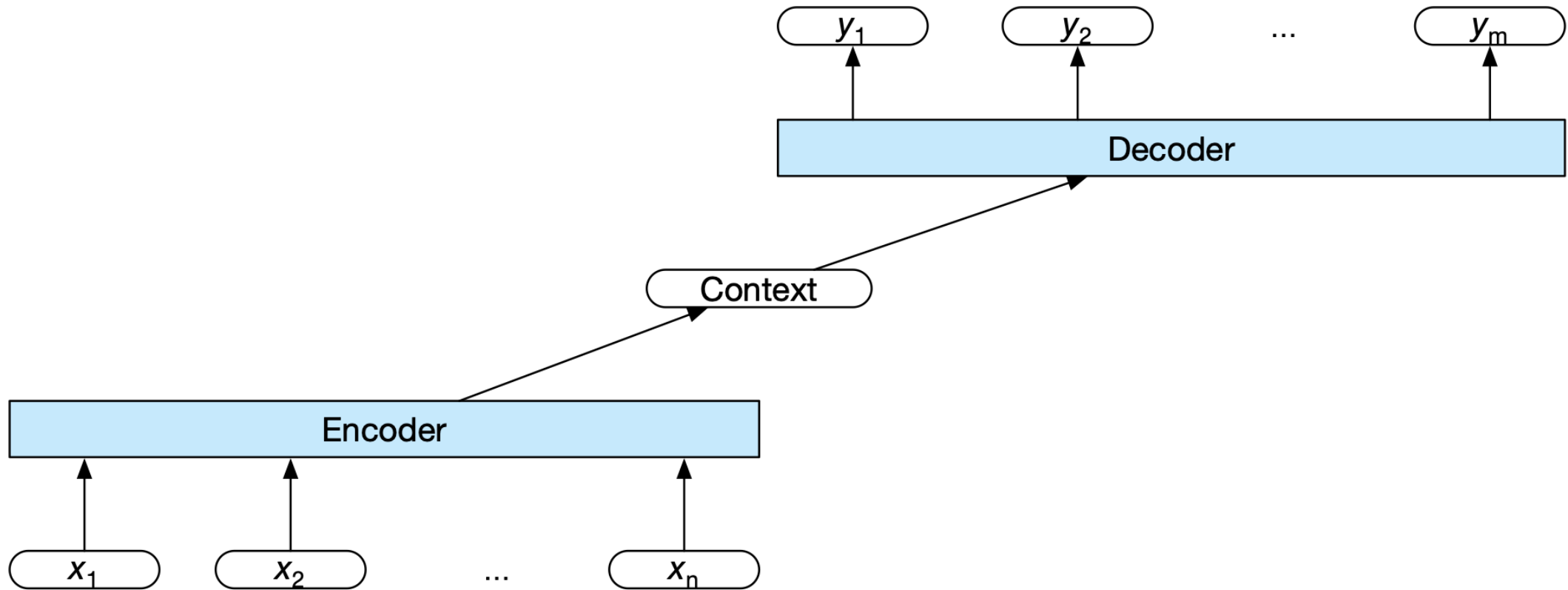


Что делать, если длины входного и выходного текстов разные?

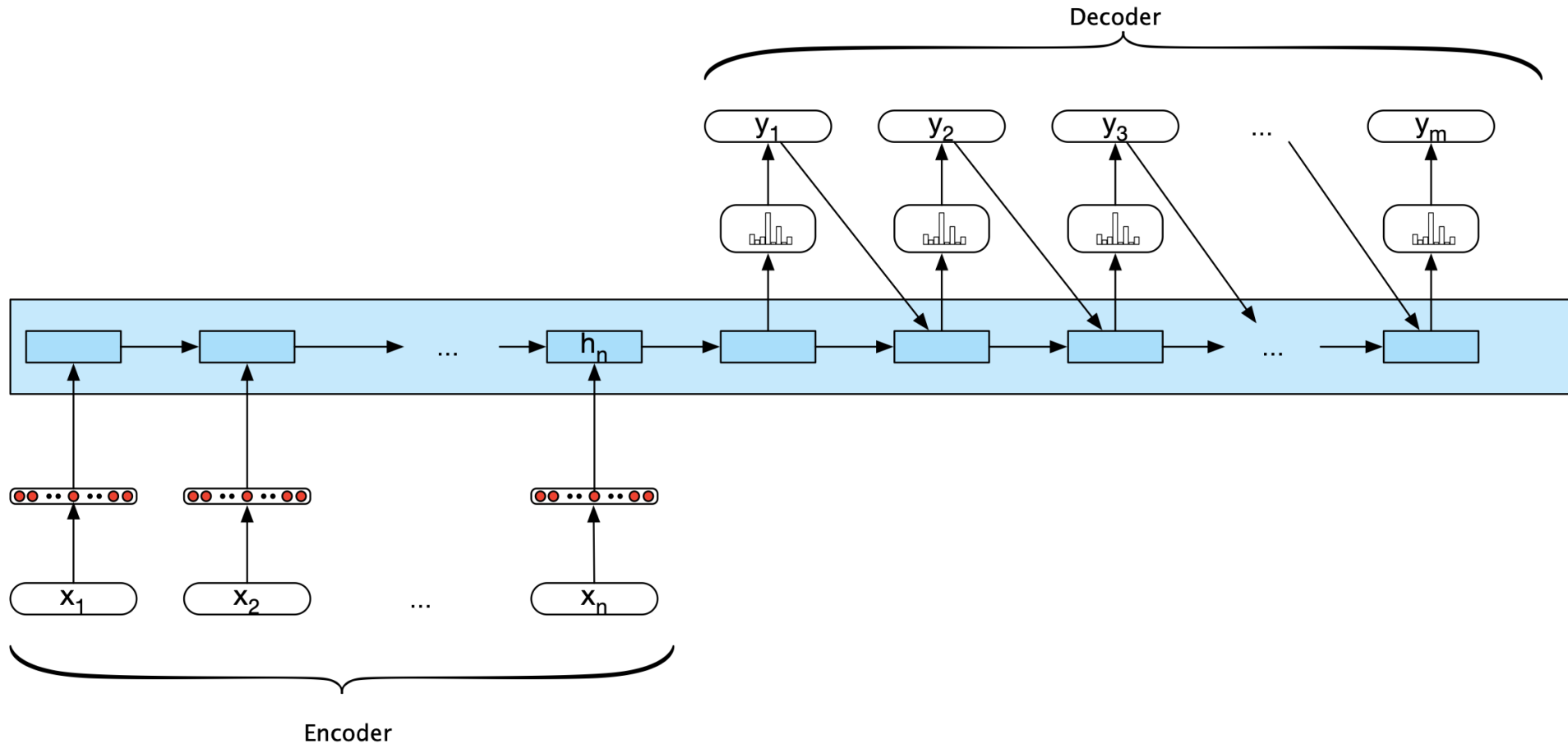
Seq2seq Machine Translation



Seq2seq Machine Translation



Seq2seq Machine Translation



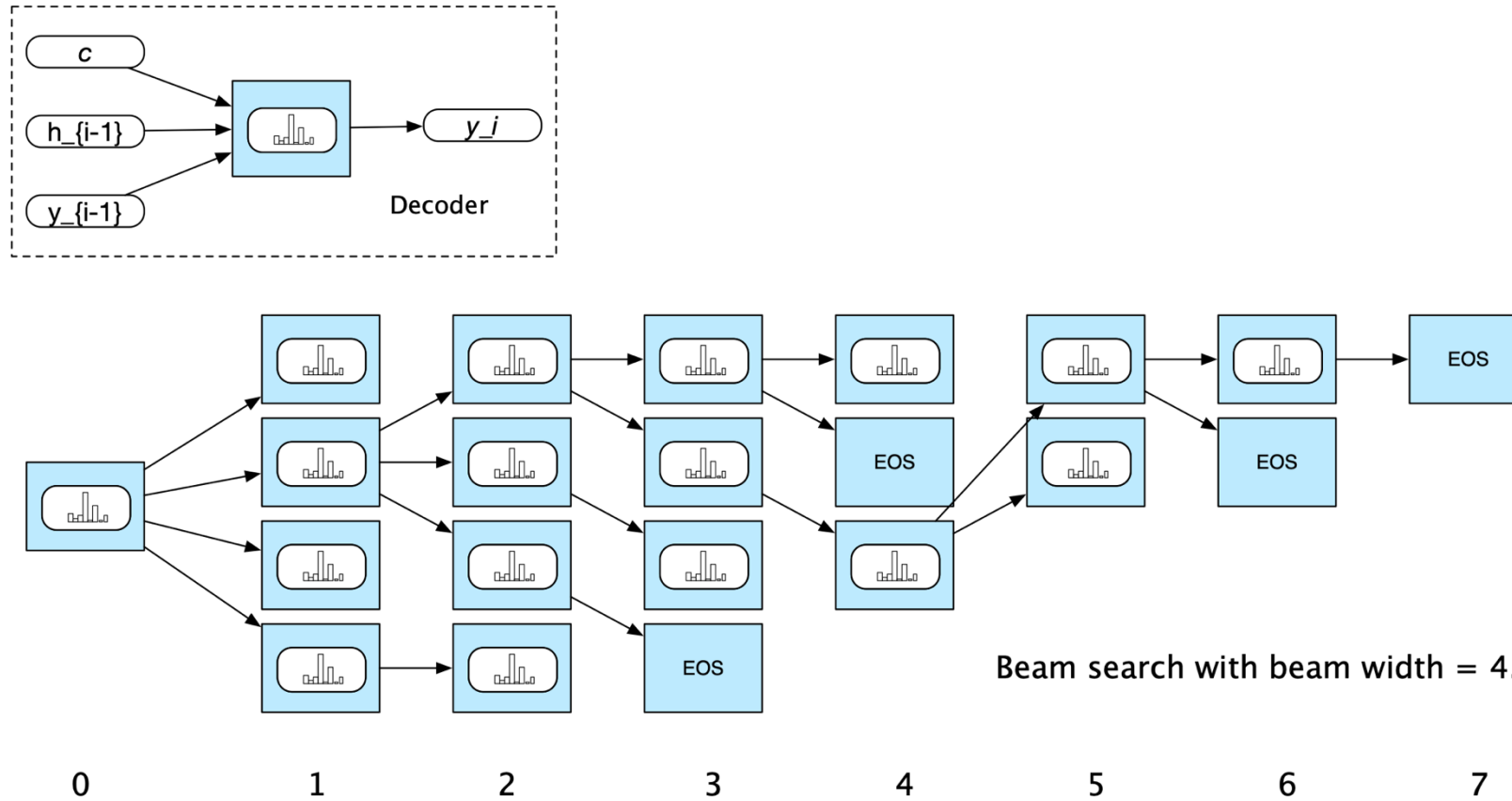
Seq2seq Machine Translation

- В конце входного текста ставим специальный токен <EOS>
- Прогоняем входной текст через RNN
- Скрытое состояние после всего текста — «контекст»
- Контекст передаётся в RNN, которая генерирует выходной текст
- Используется Beam Search

Beam Search

- Выбираем B вариантов для первого слова по максимальной вероятности
- Для каждого рассматриваем все возможные варианты для следующего слова, оставляем B наиболее вероятных вариантов
- И так далее

Beam Search



Seq2seq Machine Translation

- Четырёхслойные LSTM в качестве кодировщика и декодировщика
- В каждом слое — скрытые векторы размерности 1000
- Каждое слово описывается векторным представлением размерности 1000
- Входной текст подаётся «наоборот» — тогда первое слово входного текста оказывается ближе к первому слову выходного в нашей архитектуре

Проблемы seq2seq-архитектуры

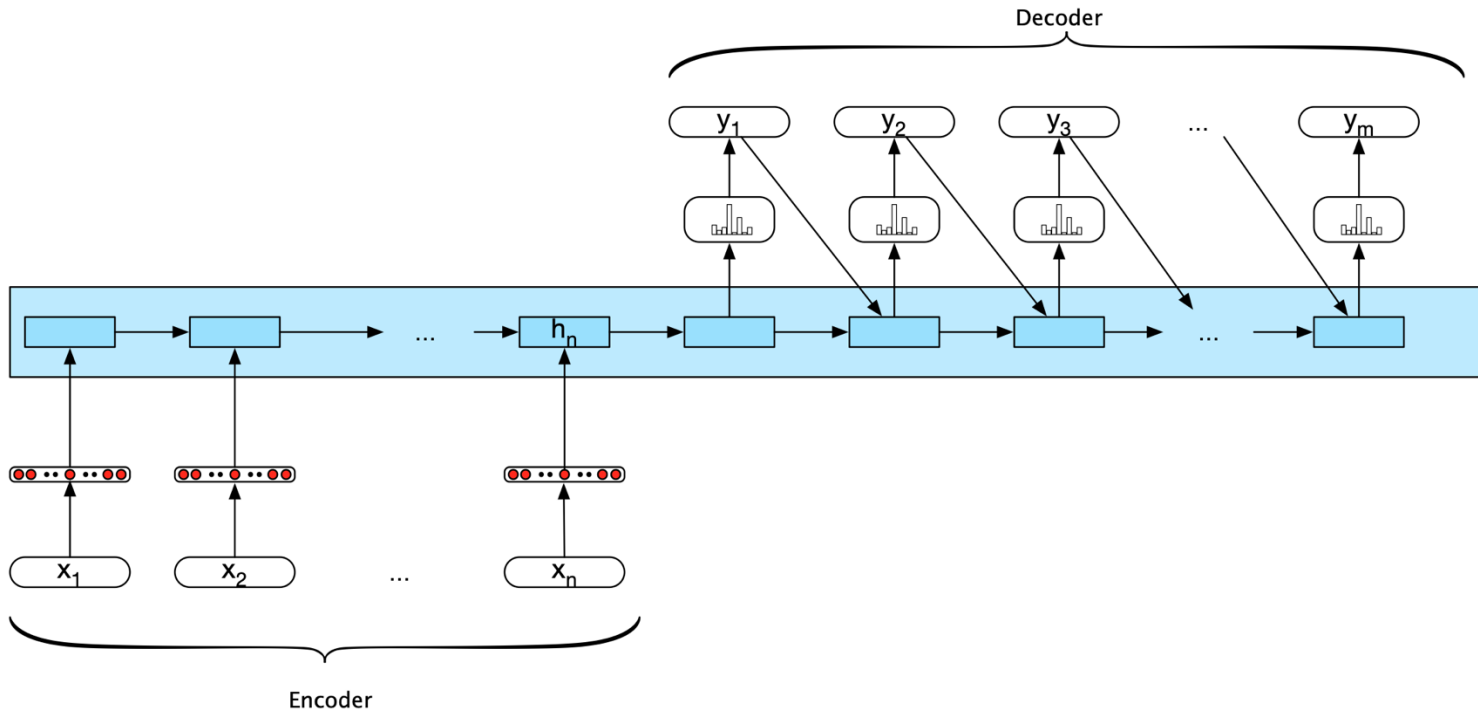
- Нужно сжать весь текст в один вектор
- Теряется информация о первых словах
- Декодер тоже может терять информацию по мере генерации последовательности
- Может, нам поможет BiLSTM?

Проблемы seq2seq-архитектуры

- Нужно сжать весь текст в один вектор
 - Теряется информация о первых словах
 - Декодер тоже может терять информацию по мере генерации последовательности
-
- Можно использовать BiLSTM, но тогда будет теряться информация о словах в середине
 - И непонятно, как им декодировать

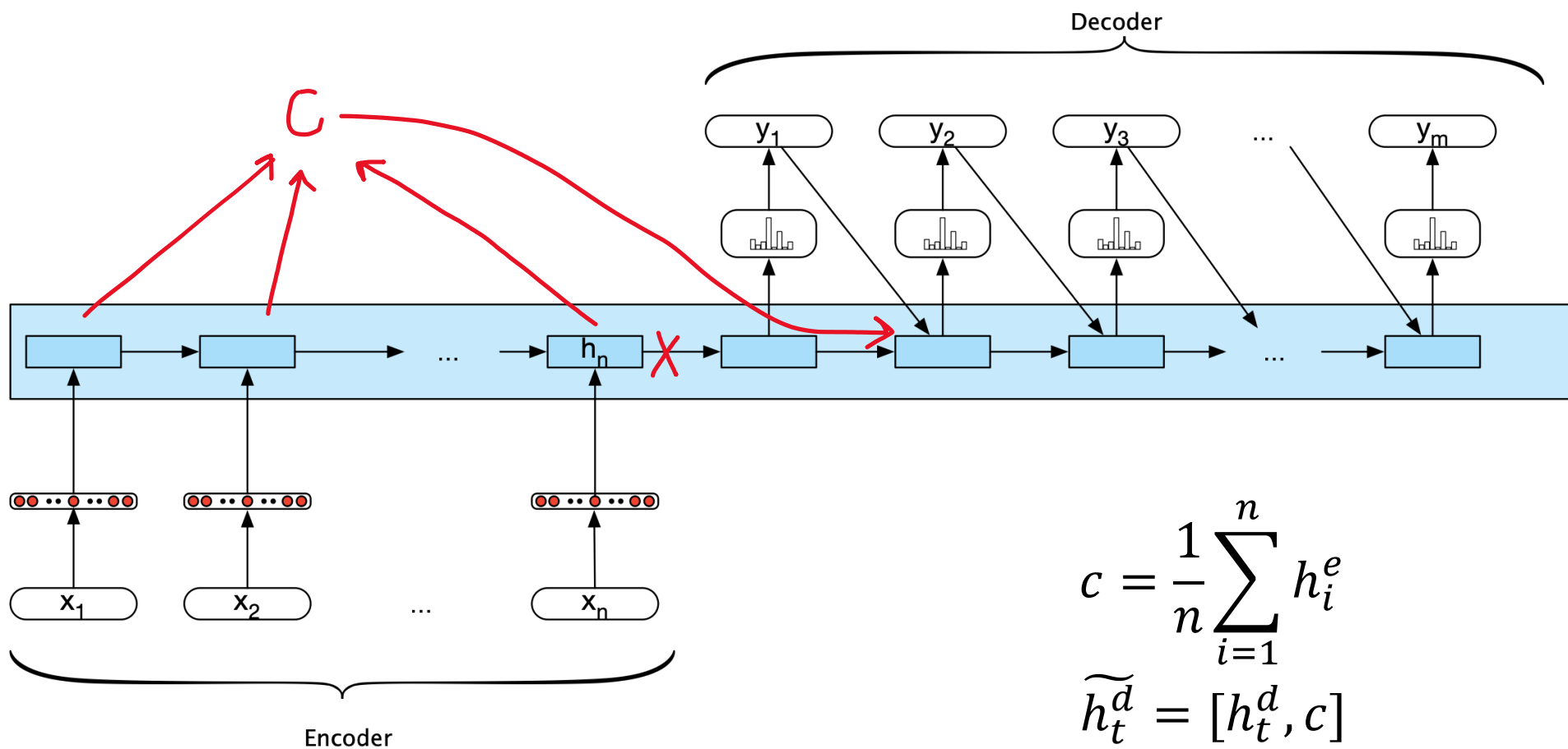
Механизм внимания

Seq2seq



- Читаем текст слева направо, собираем информацию о словах внутри скрытого вектора h_t
- Вряд ли можно уместить полный смысл текста в одном векторе
- Скрытый вектор всего входного текста — «бутылочное горлышко» в такой архитектуре

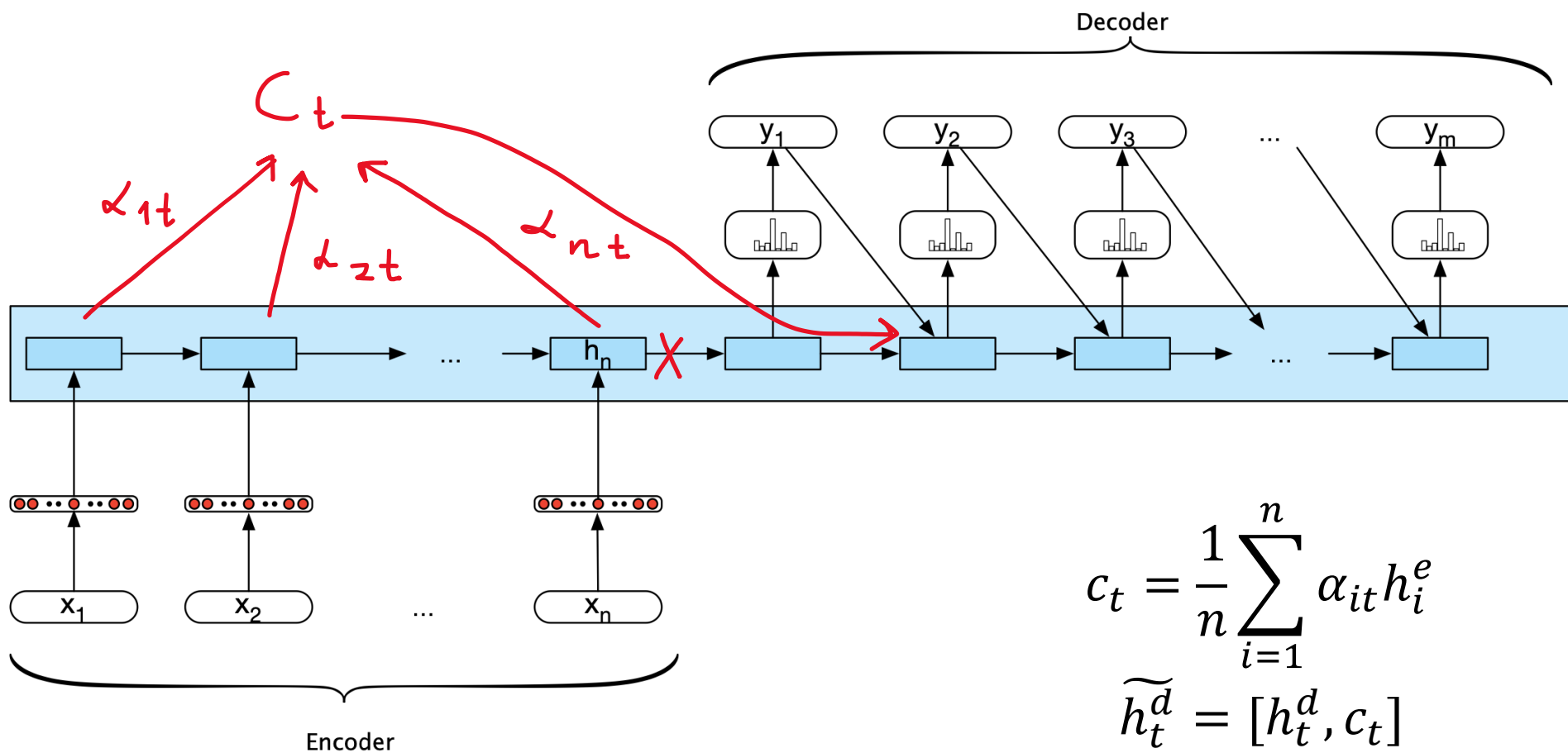
Механизм внимания: версия 1



Механизм внимания: версия 1

- При генерации всех выходов мы используем одну и ту же информацию о входном тексте
- Это мало чем отличается от seq2seq-архитектуры
- Хочется, чтобы каждое выходное слово могло «обращать внимание» на «свои» входные слова

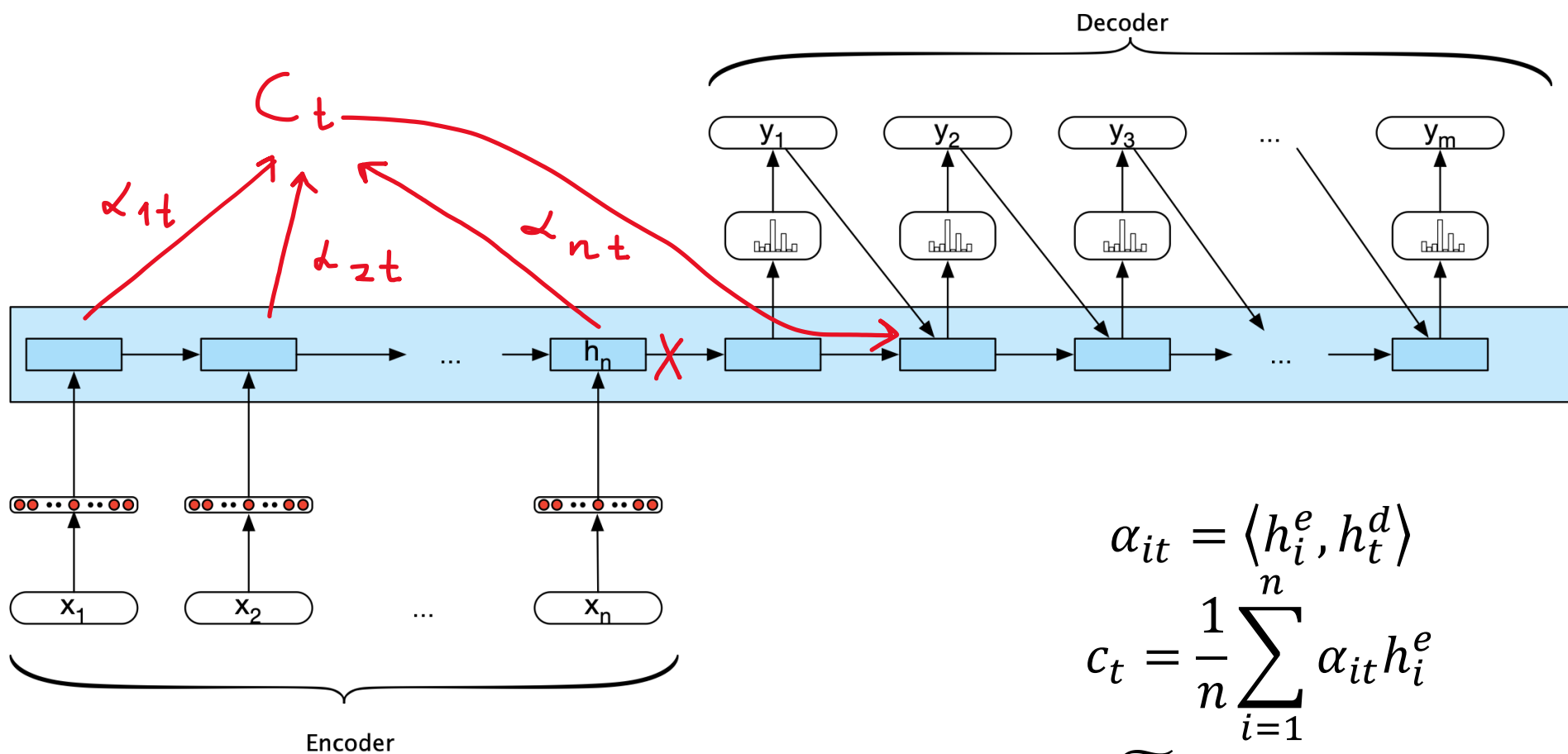
Механизм внимания: версия 2



Механизм внимания: версия 2

- Надо зафиксировать раз и навсегда, как i -е входное слово влияет на t -е выходное слово
- Эти влияния очень зависят от конкретных текстов
- Надо вычислять α_{it} в зависимости от данных

Механизм внимания: версия 3



$$\alpha_{it} = \langle h_i^e, h_t^d \rangle$$

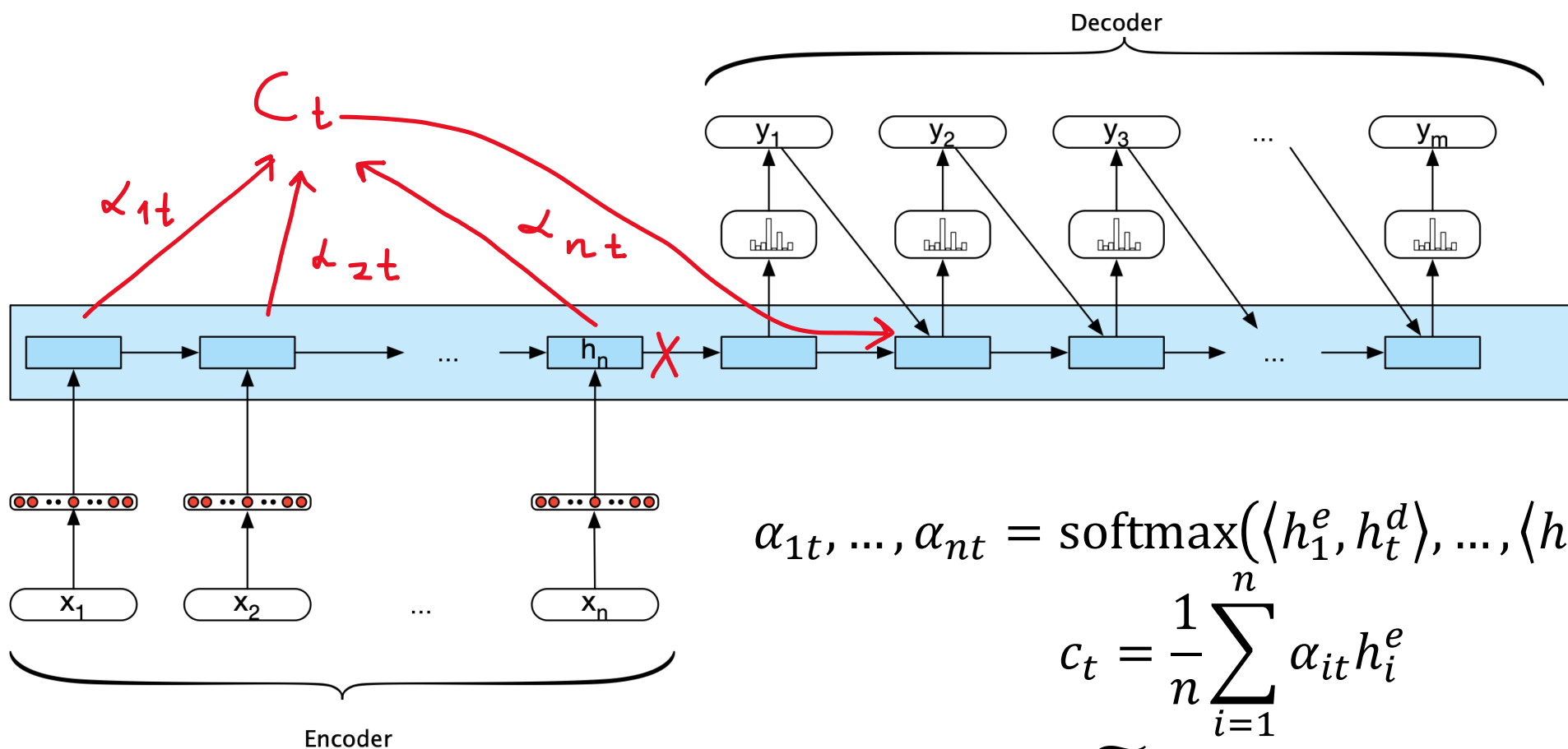
$$c_t = \frac{1}{n} \sum_{i=1}^n \alpha_{it} h_i^e$$

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

Механизм внимания: версия 3

- Есть проблема с тем, что веса α_{it} могут принимать совершенно любые значения
- Почему это проблема?
 - Масштаб c_t может зависеть от длины входной последовательности
 - Скорее всего, все веса будут сильно ненулевыми, то есть все входные слова будут влиять на все выходные слова

Механизм внимания: версия 4



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

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

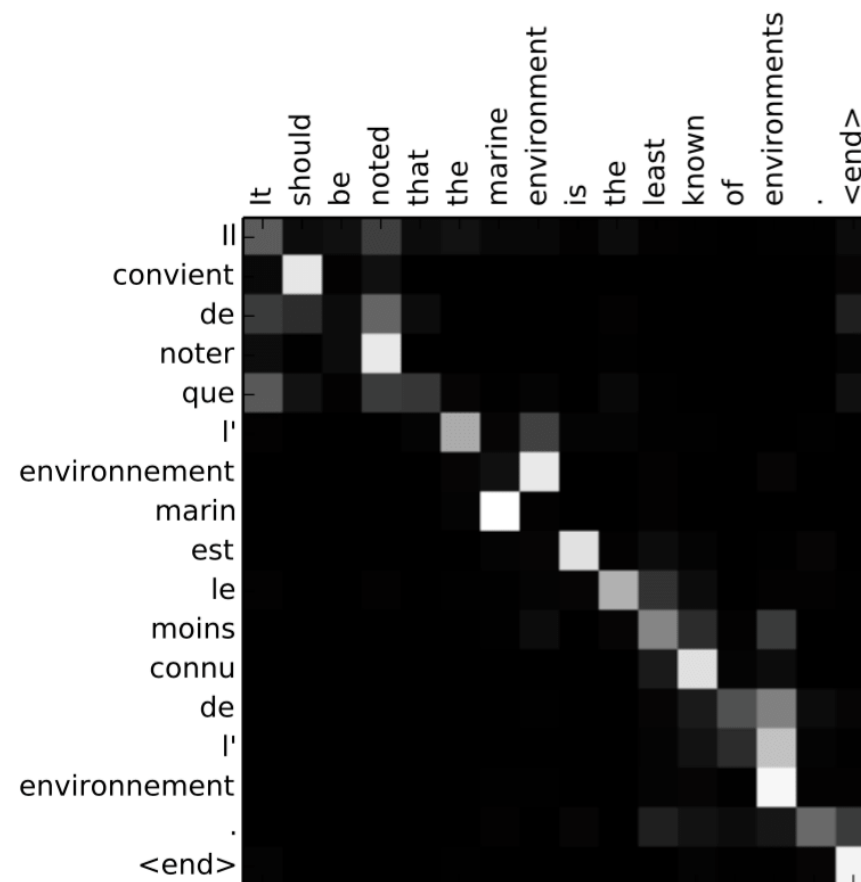
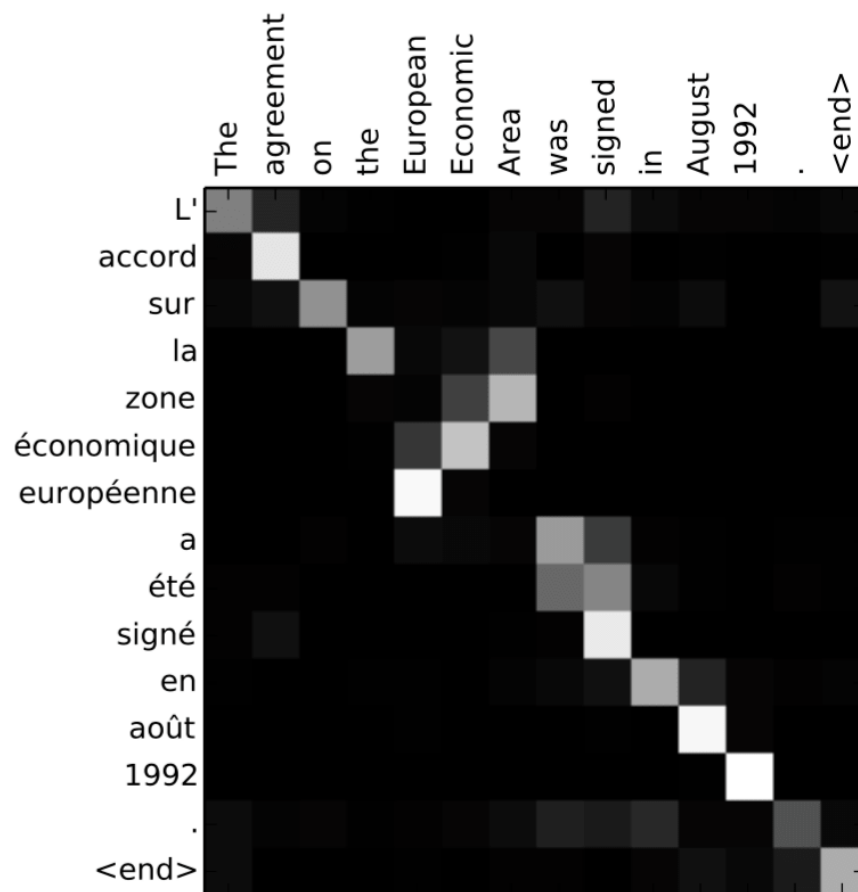
$$\widetilde{h}_t^d = [h_t^d, c_t]$$

softmax

$$(a_1, \dots, a_n) \rightarrow \left(\frac{\exp(a_1)}{\sum_{i=1}^n \exp(a_i)}, \dots, \frac{\exp(a_n)}{\sum_{i=1}^n \exp(a_i)} \right)$$

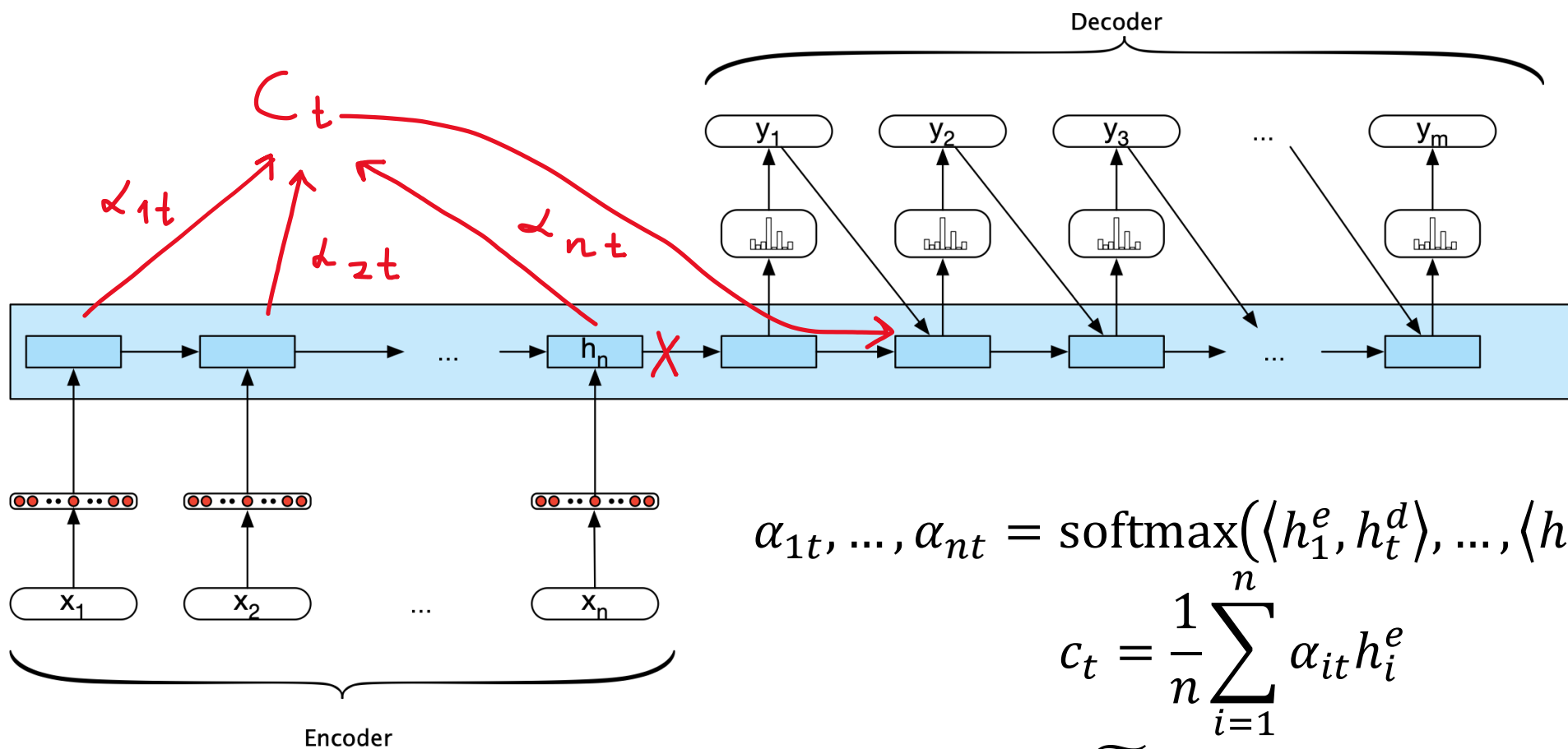
- $[-5, 1, 10] \rightarrow [0, 0, 1]$
- $[1, 1, 1] \rightarrow [0.33, 0.33, 0.33]$
- $[1, 2, 0] \rightarrow [0.24, 0.67, 0.09]$

Механизм внимания



Трансформер: основы self-attention

Механизм внимания



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

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

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

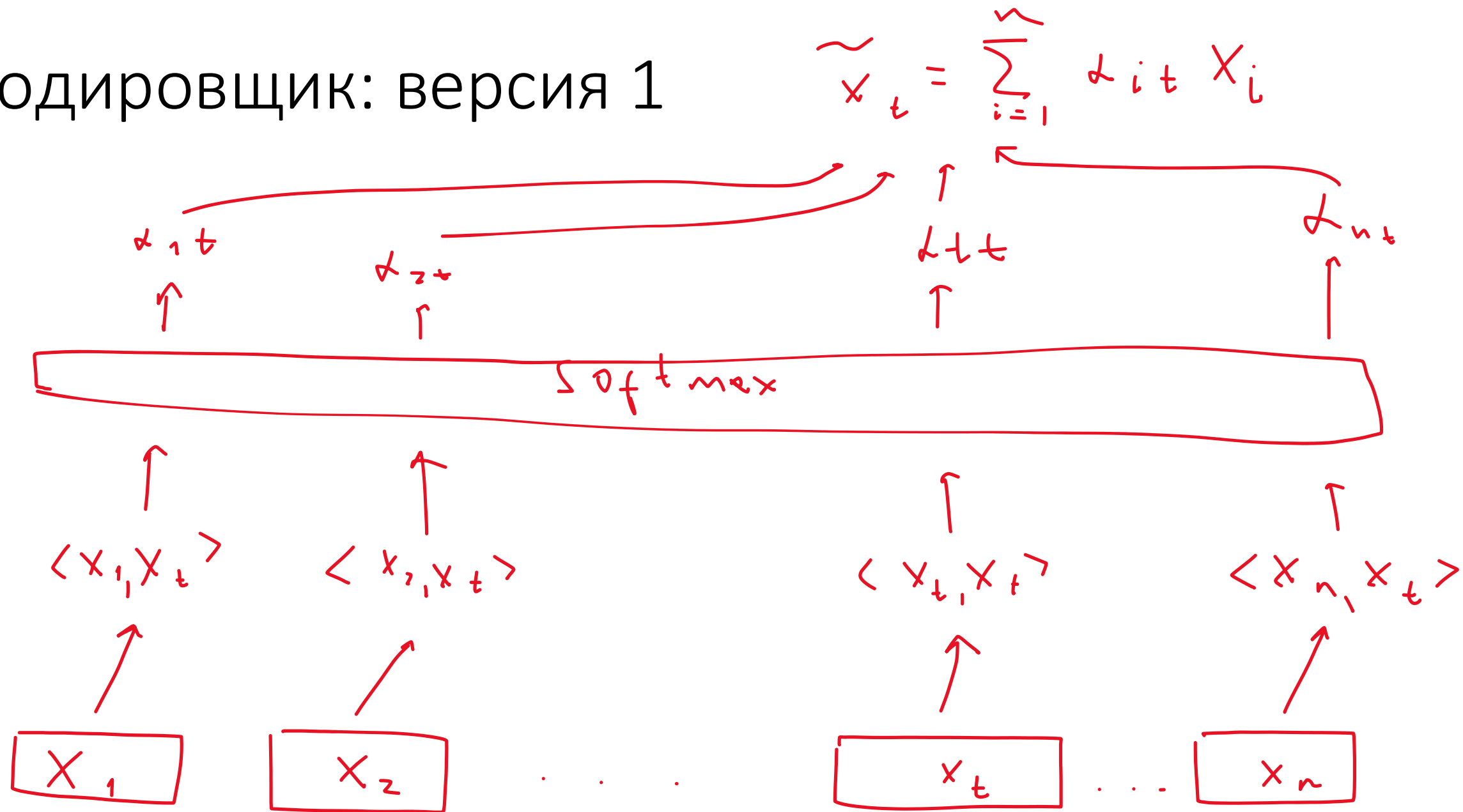
Как усилить архитектуру?

- 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



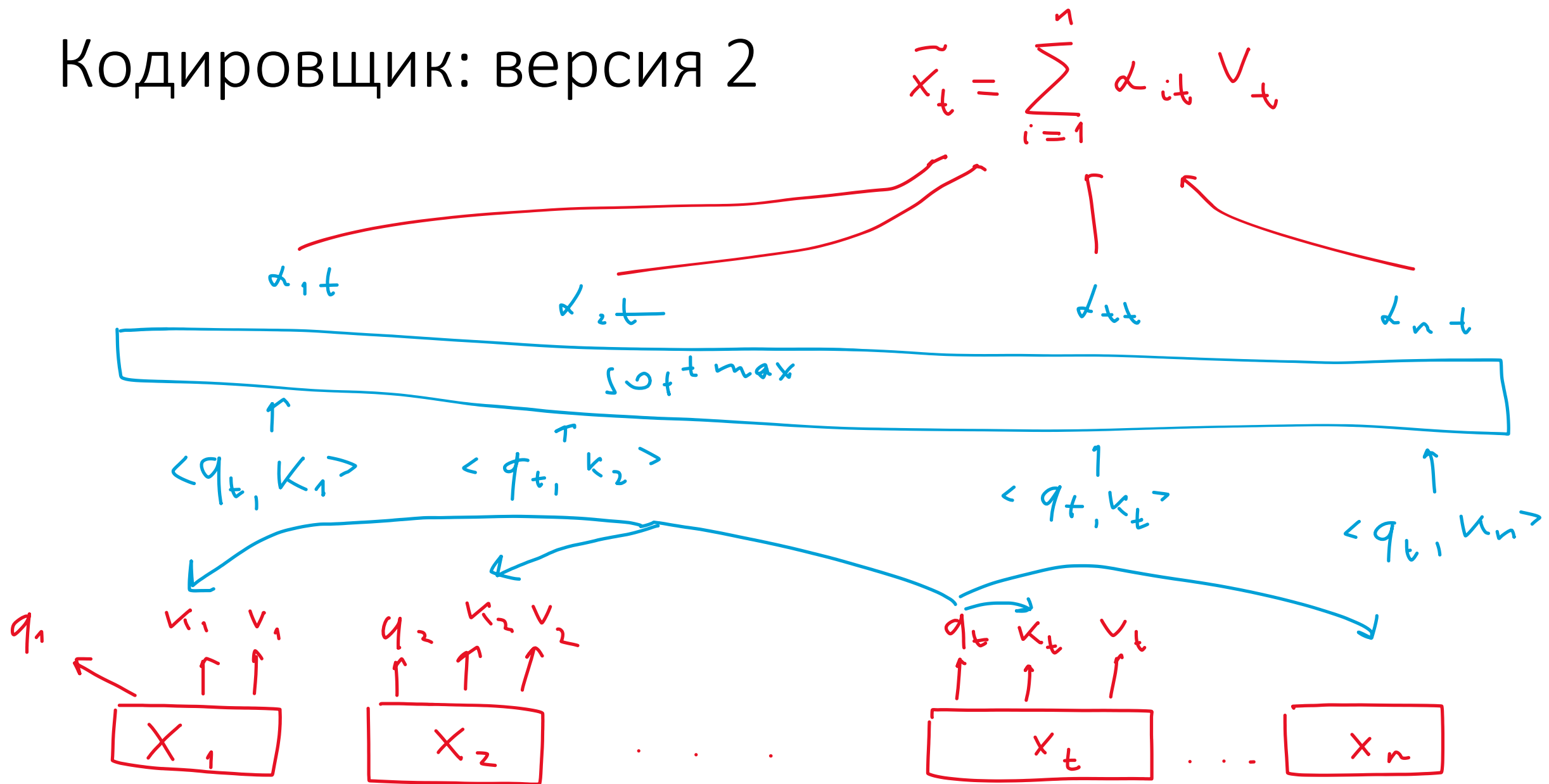
Кодировщик: версия 1

- 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



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 :

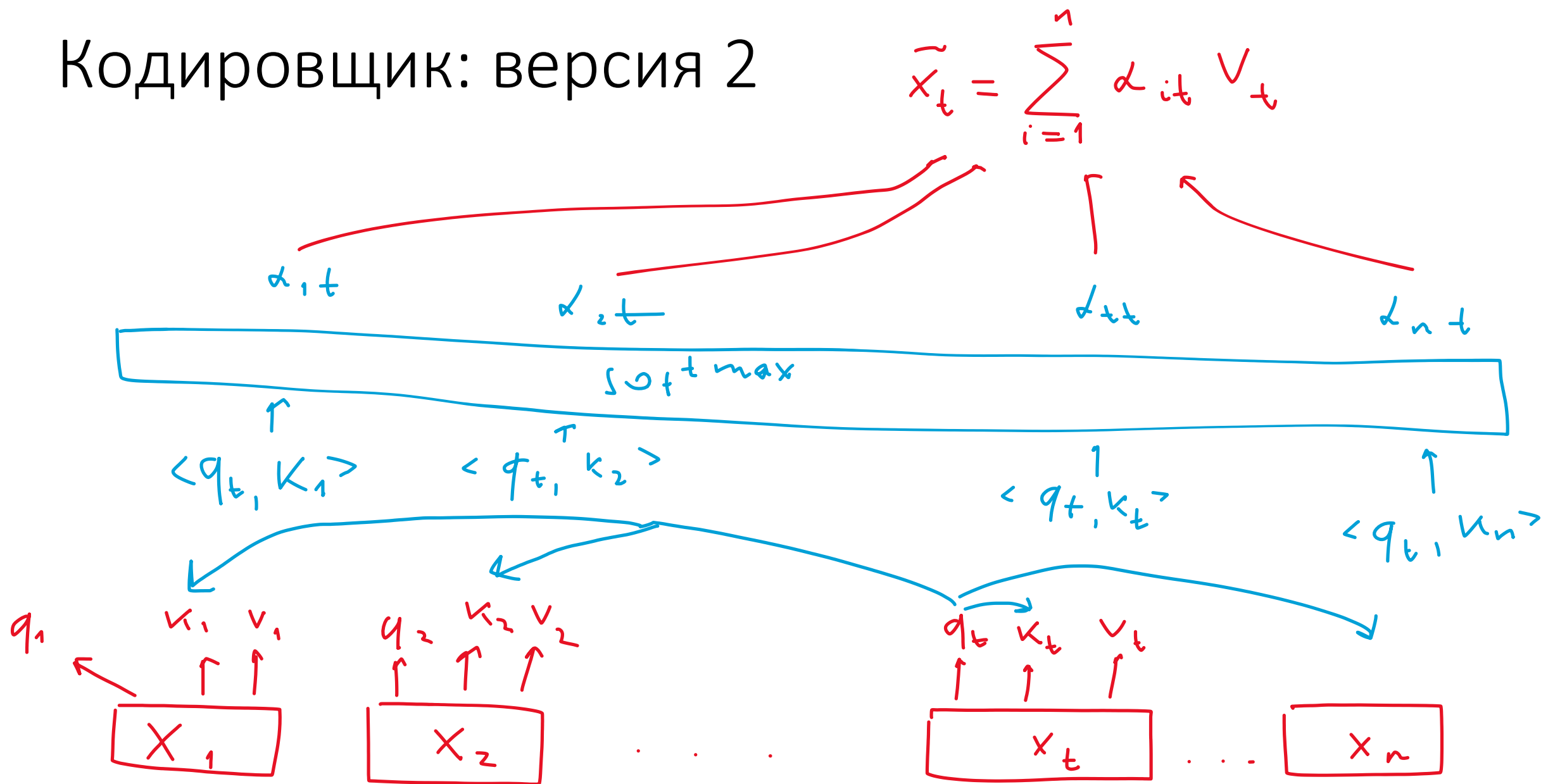
$$\tilde{x}_j = \sum_{i=1}^n \alpha_{ij} v_i$$

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



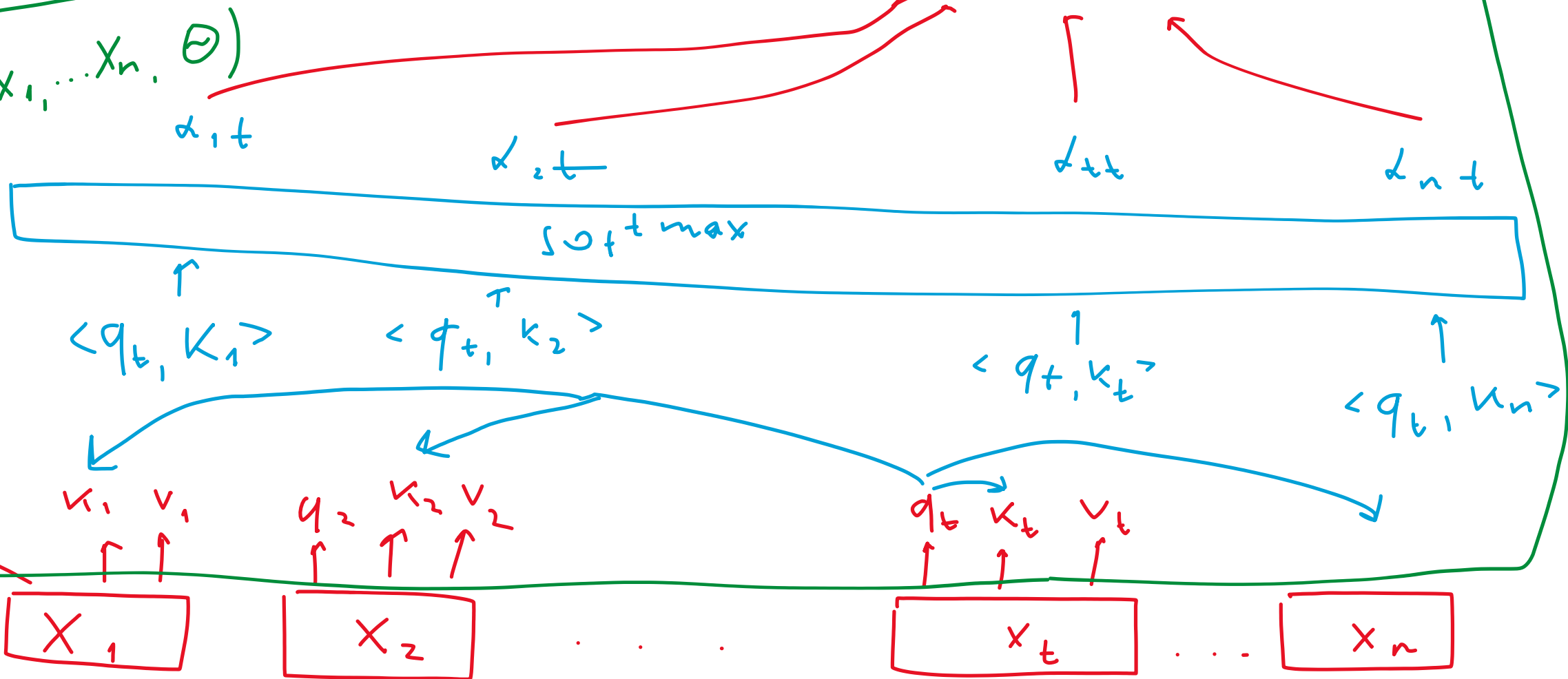
Кодировщик: версия 2

- 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

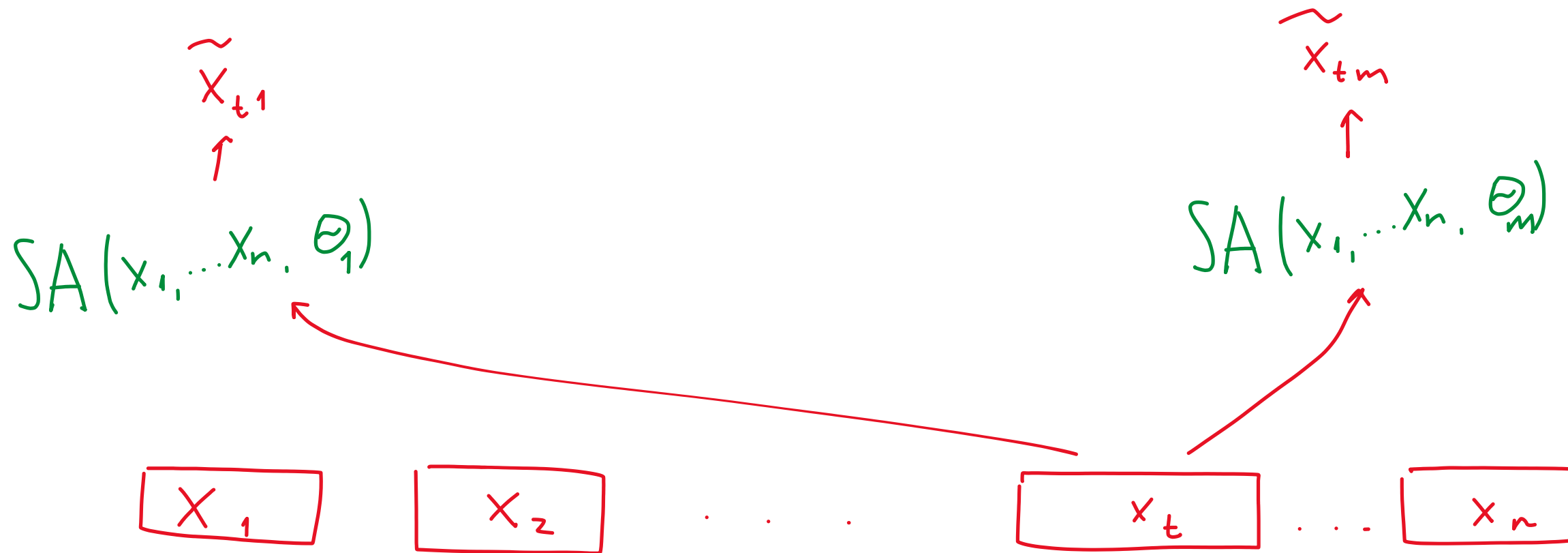
Кодировщик: версия 3

$SA(x_1, \dots, x_n, \Theta)$

$$\bar{x}_t = \sum_{i=1}^n \alpha_{it} V_t$$



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

