Рекуррентные нейронные сети

Ai Edu

План рассказа

- Вероятностные модели генерации текстов
- Рекуррентные нейронные сети
- Обучение RNN
- Типы RNN, многослойные RNN, примеры использования
- Модификации RNN: LSTM, GRU
- Seq2seq-архитектуры

Кодирование текста

При кодировании слов с помощью Word2Vec/FastText:

- Вектор текста - это средний вектор его слов
- Вектора текста - это средний вектор его слов с TfIDf-весами

Можно ли умнее?

Марковские модели

Предположение: наличие конкретного слова в тексте объясняется только k словами, стоящими перед ним

$$p(w_1, \ldots, w_n) = p(w_1)p(w_2 \mid w_1) \ldots p(w_n \mid w_{n-1}, \ldots, w_{n-k})$$

Марковские модели

Предположение: наличие конкретного слова в тексте объясняется только k словами, стоящими перед ним

$$p(w_1, \ldots, w_n) = p(w_1)p(w_2 \mid w_1) \ldots p(w_n \mid w_{n-1}, \ldots, w_{n-k})$$

Все вероятности (частоты) из правой части формулы можно посчитать по большому корпусу обучающих текстов.

Обычно делают со сглаживанием (чтобы избежать нулевых вероятностей).

Марковские модели

Подробно про марковские цепи: https://www.kdnuggets.com/2019/11/markov-chains-train-text-generation.html

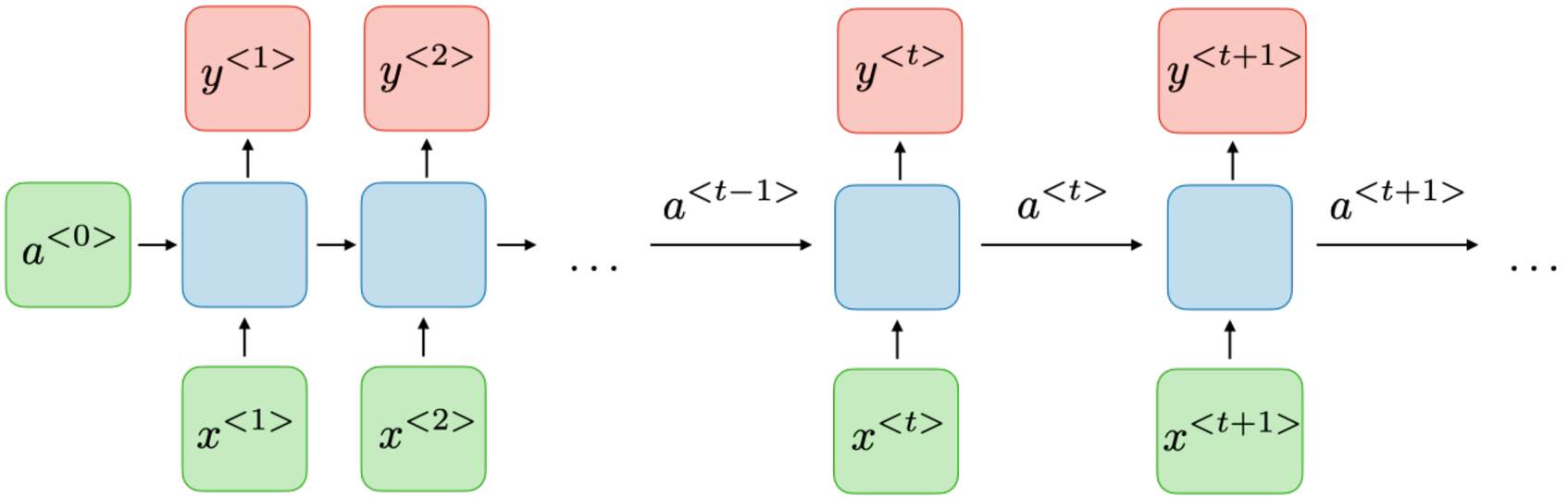
```
'I am a master armorer , lords of Westeros , sawing out each bay and peninsula
'Jon Snow is with the Night's Watch . I did not survive a broken hip , a leath
'Jon Snow is with the Hound in the woods . He won't do it . " Please don't'
'Where are the chains , and the Knight of Flowers to treat with you , Imp . "'
'Those were the same . Arianne demurred . " So the fishwives say , " It was Ty
'He thought that would be good or bad for their escape . If they can truly giv
'I thought that she was like to remember a young crow he'd met briefly years b
```

Идея

- Мы читаем текст *последовательно*
- Постепенно *все лучше понимаем, о чем он*

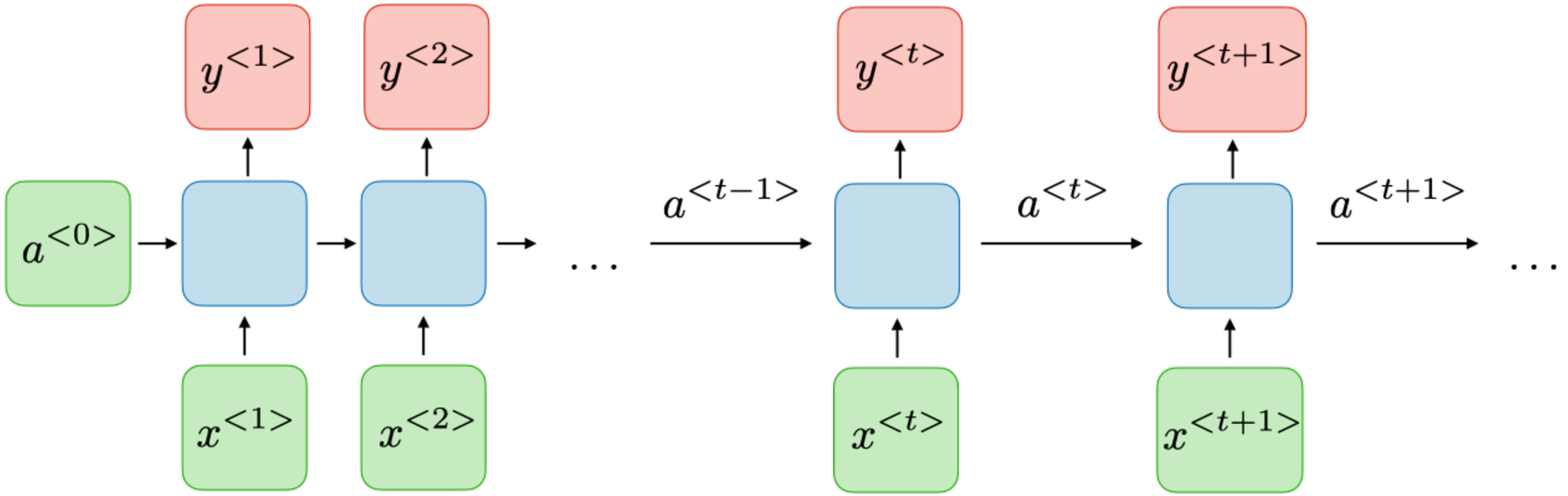
Рекуррентные нейронные сети

- На вход: слова (текст) $x_1, x_2, \ldots, x_n, \ldots$
- Читаем слева направо



Рекуррентные нейронные сети

- На вход: слова (текст) $x_1, x_2, \ldots, x_n, \ldots$
- Читаем слева направо
- a_t (вектор) - накопленная информация после прочтения t элементов



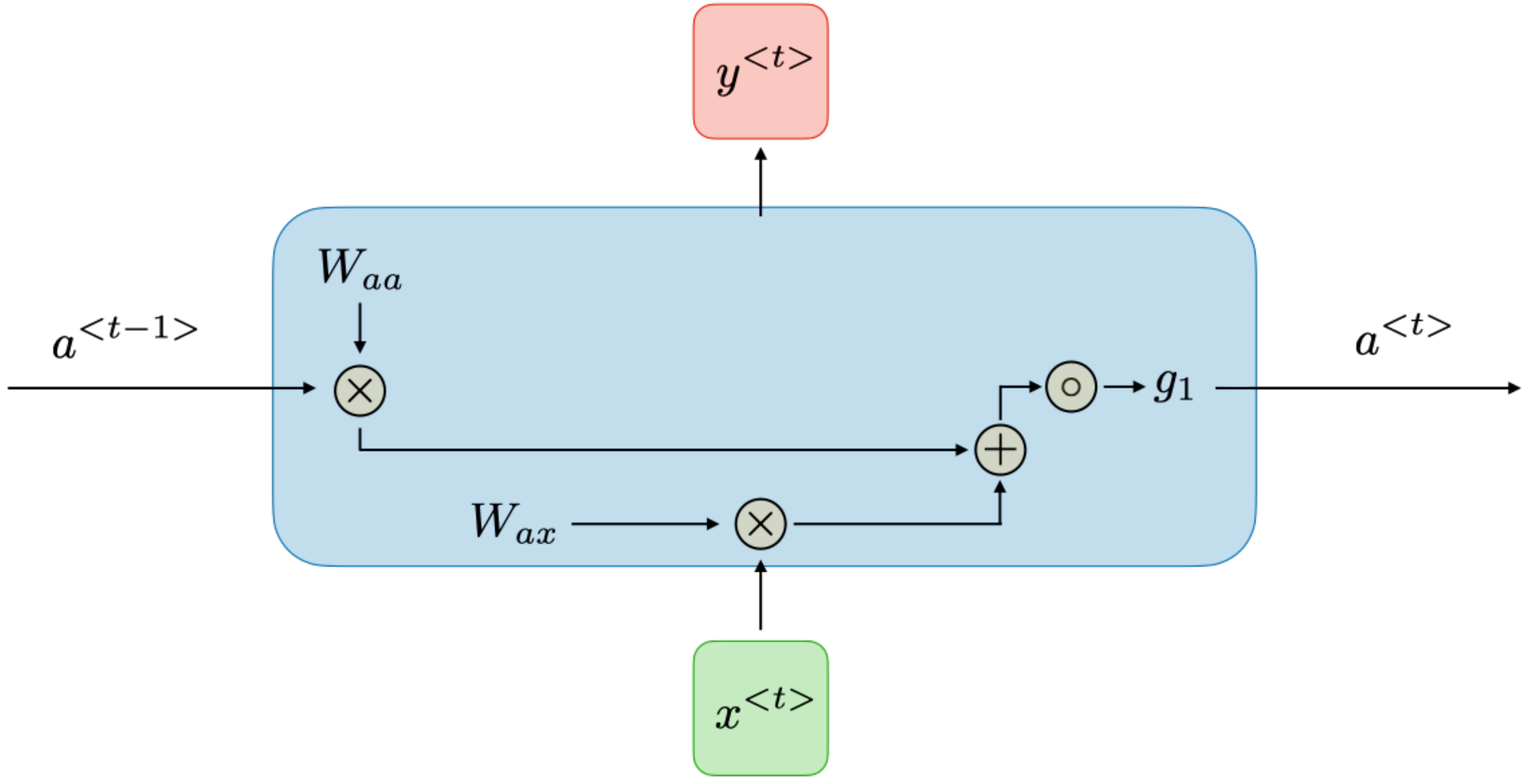
Рекуррентные нейронные сети

- На вход: слова (текст) $x_1, x_2, \ldots, x_n, \ldots$
- Читаем слева направо
- a_t (вектор) - накопленная информация после прочтения t элементов
- x_t - или one-hot вектор, или векторное представление слова (w2v, fasttext)

Обновление состояния ячейки

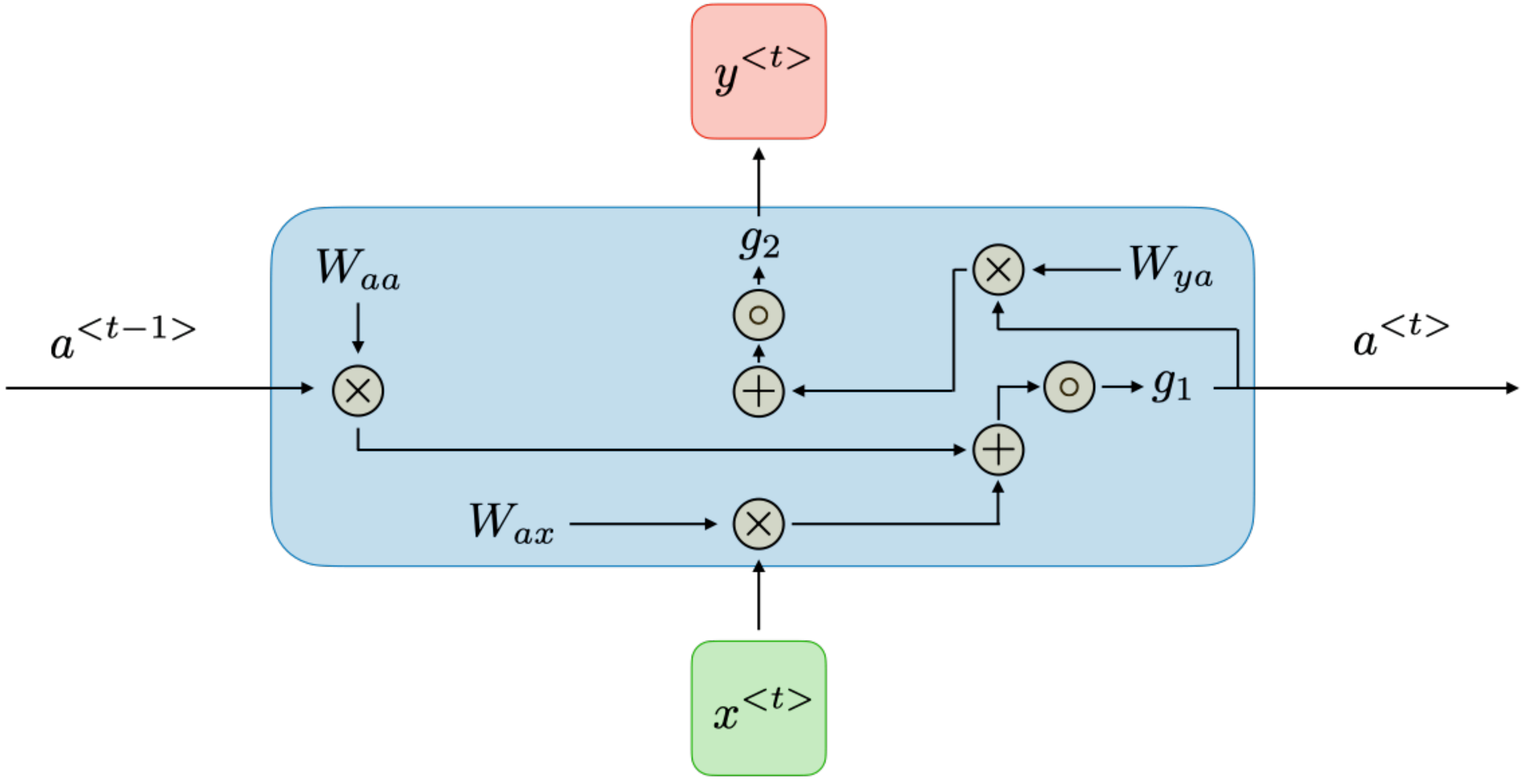
$$a_t = g_1(W_{ax}x_t + W_{aa}a_{t-1})$$

Обычно $g_1 = tanh$

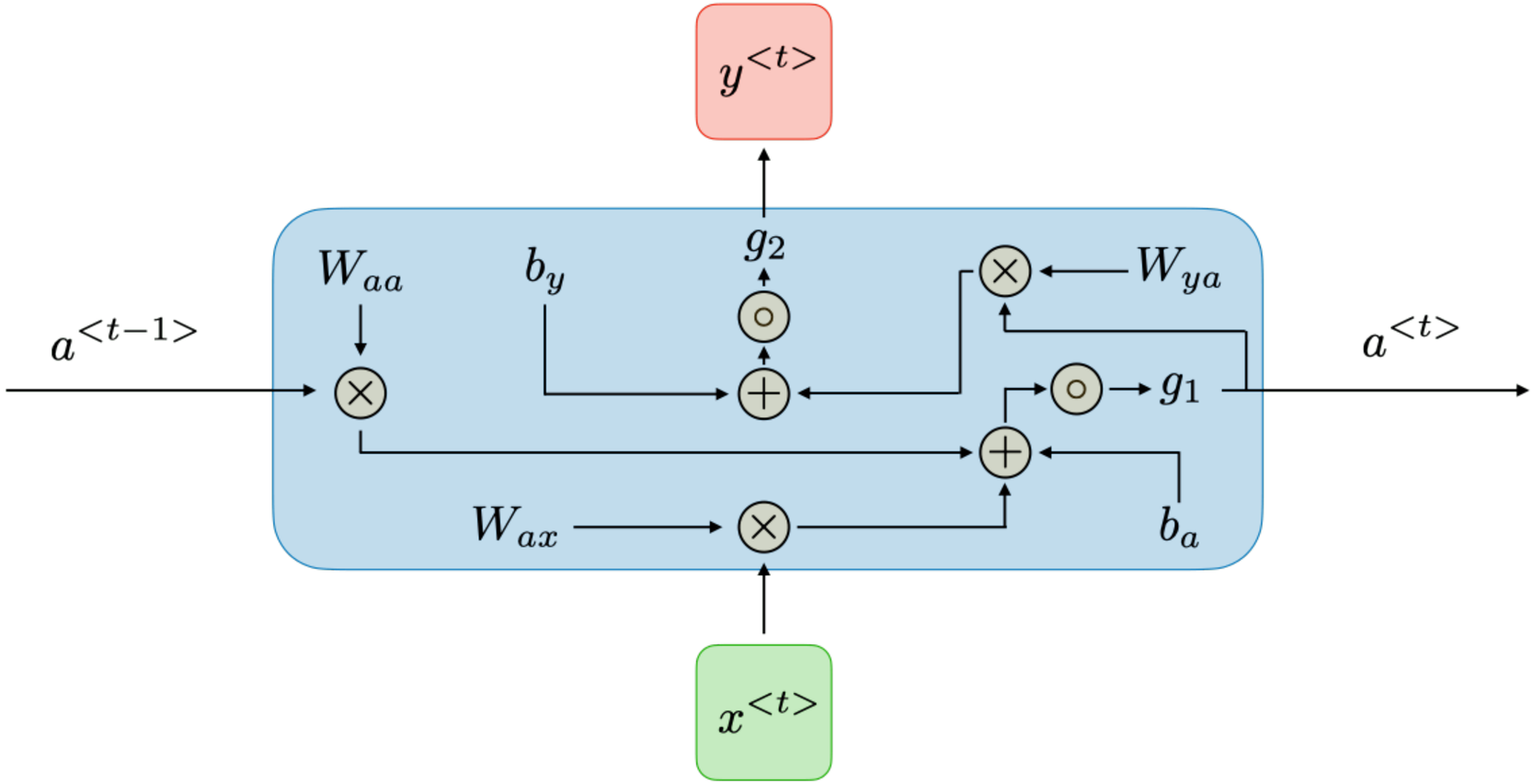


Получение прогноза

$$y_t = g_2(W_{ya} a_t)$$

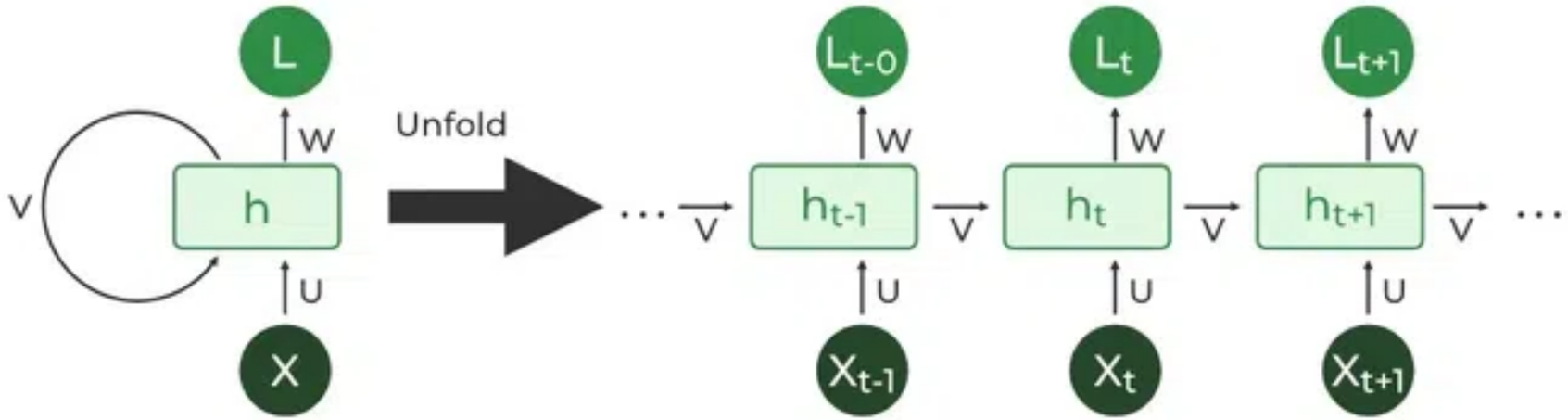


Общая картинка (с bias-term)



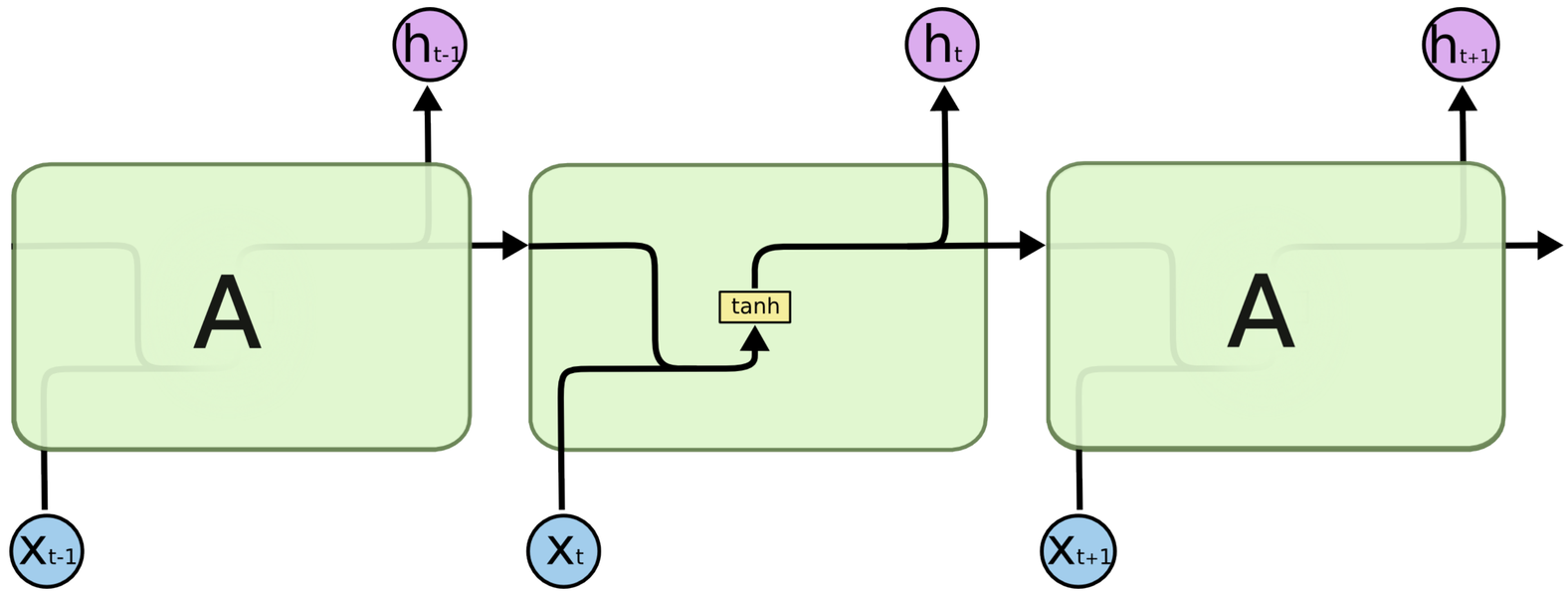
Рекуррентные нейронные сети

Общая схема RNN (здесь скрытые состояния обозначены как h_t):



Рекуррентные нейронные сети

Общая схема RNN (здесь скрытые состояния обозначены как h_t):



Обучение RNN

- Нейронные сети обучаются градиентным спуском (или его модификацией)
- Формула для градиентного спуска:

$$w_{k+1} = w_k - \eta \nabla L(w_k)$$

- Здесь w_k - вектор весов модели на k-й итерации метода, η - learning rate,

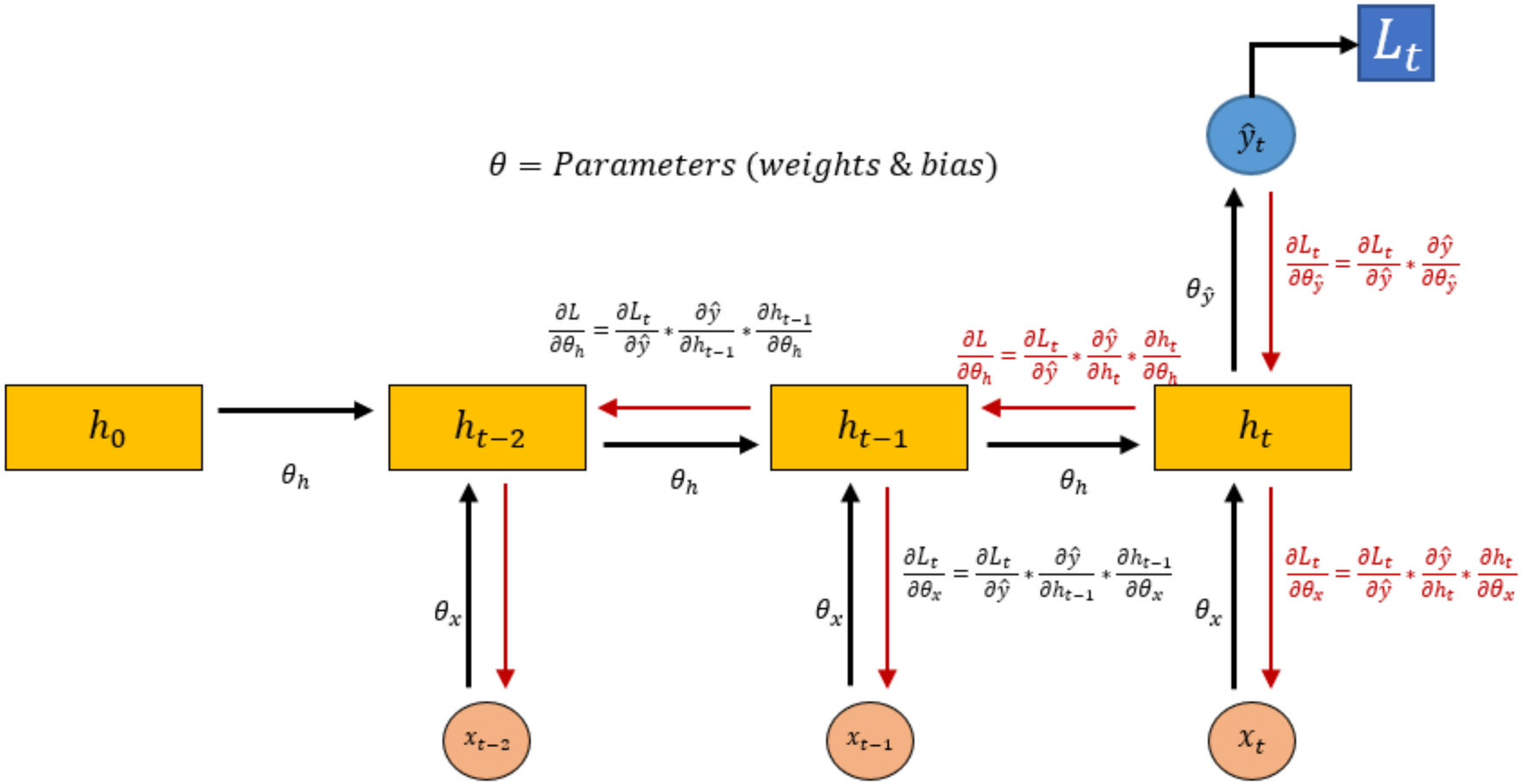
$\nabla L(w_k)$ - градиент функции потерь на k-й итерации.

Вычисление градиента

Нам нужно уметь вычислять $\nabla L(w) = \{\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \dots\}$, то есть производные по каждому весу.

Это делается при помощи метода обратного распространения ошибок.

Backpropagation through time



Backpropagation through time

- Запишем формулы в следующих обозначениях:

$$h_t = f(x_t, h_{t-1}, w_h)$$

$$o_t = g(h_t, w_o)$$

- Функция потерь вычисляется как сумма потерь по всем временным шагам:

$$L(x_1, \ldots, x_T, y_1, \ldots, y_T, w_h, w_o) = \frac{1}{T} \sum_{t=1}^{T} l(y_t, o_t)$$

Backpropagation through time

- Посчитаем градиент по весу:

$$\frac{\partial L}{\partial w_h} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial l(y_t, o_t)}{\partial w_h} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial l(y_t, o_t)}{\partial o_t}\frac{\partial g(h_t, w_o)}{h_t}\frac{\partial h_t}{\partial w_h}$$

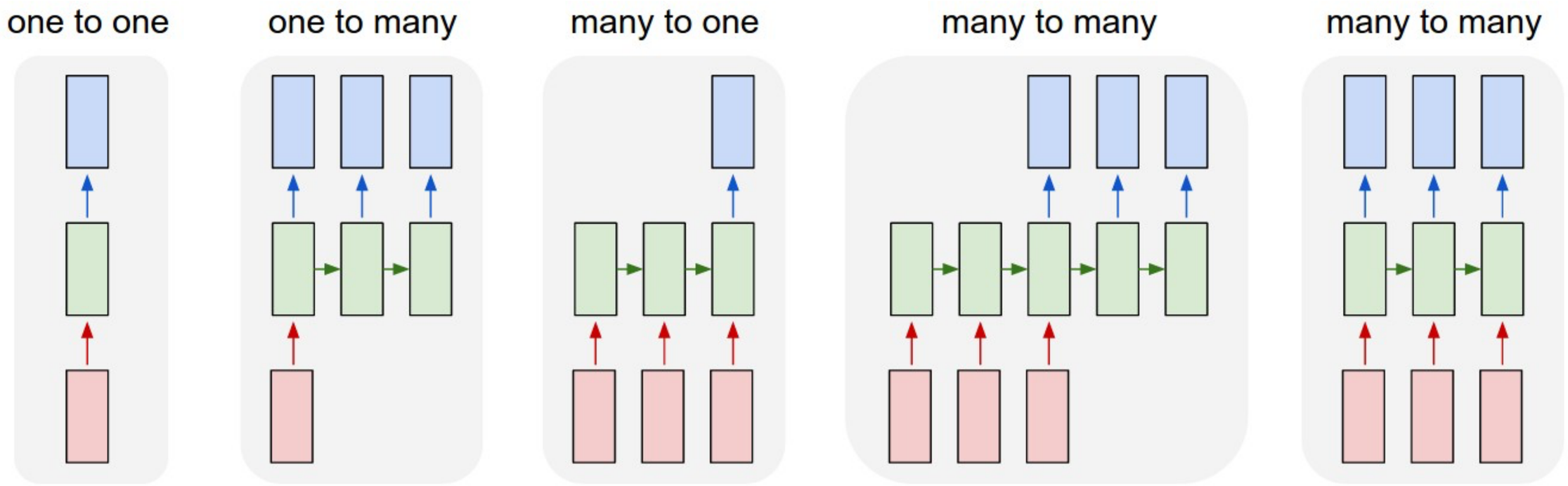
- Первый и второй множитель считаются сразу, а третий придется вычислить по формуле (уходим назад во время):

$$\frac{\partial h_t}{\partial w_h} = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h} + \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial w_h}$$

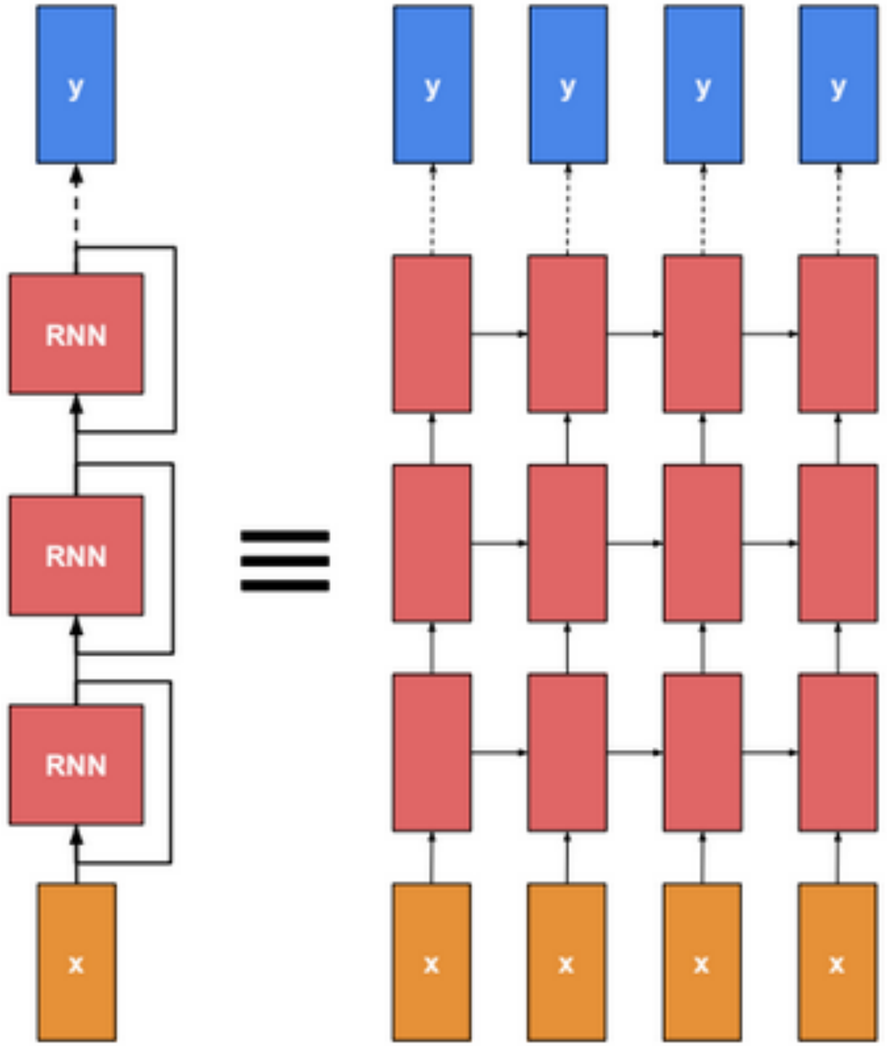
Backpropagation through time

Подробности вычисления производных в матричном виде: https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

Типы RNN



Многослойные RNN



Примеры

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain 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 nues 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.
```

http://karpathy.github.io/2015/05/21/rnn -effectiveness/

Примеры

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, 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 $U \to T$ is a separated algebraic space.

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

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

and the compariciely in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ 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 $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ 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 $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

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

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

is a unique morphism of algebraic stacks. Note that

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

and

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

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. □

The result for prove any open covering follows from the less of Example **??**. It may replace S by $X_{spaces,étale}$ 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* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

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

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *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* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

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

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

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* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *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 mext 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. □

http://karpathy.github.io/2015/05/21/rnn -effectiveness/

Примеры

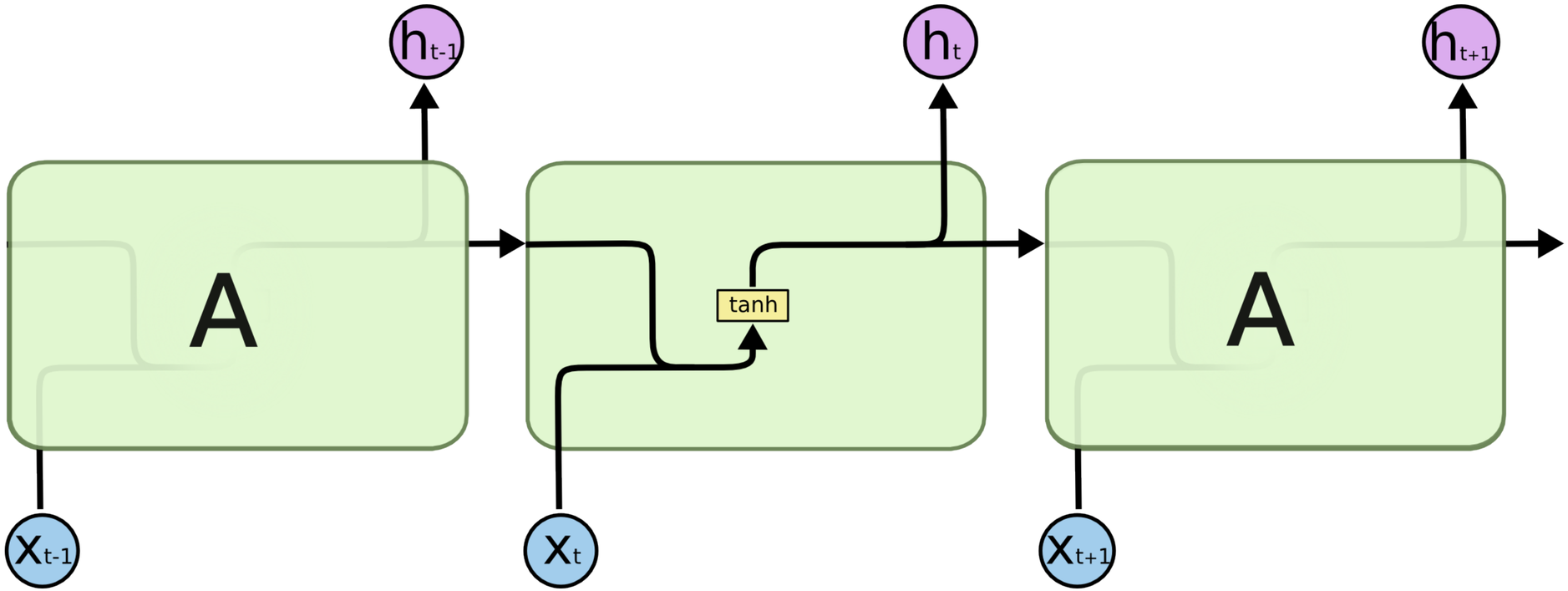
```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    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;
}
```

http://karpathy.github.io/2015/05/21/rnn -effectiveness/

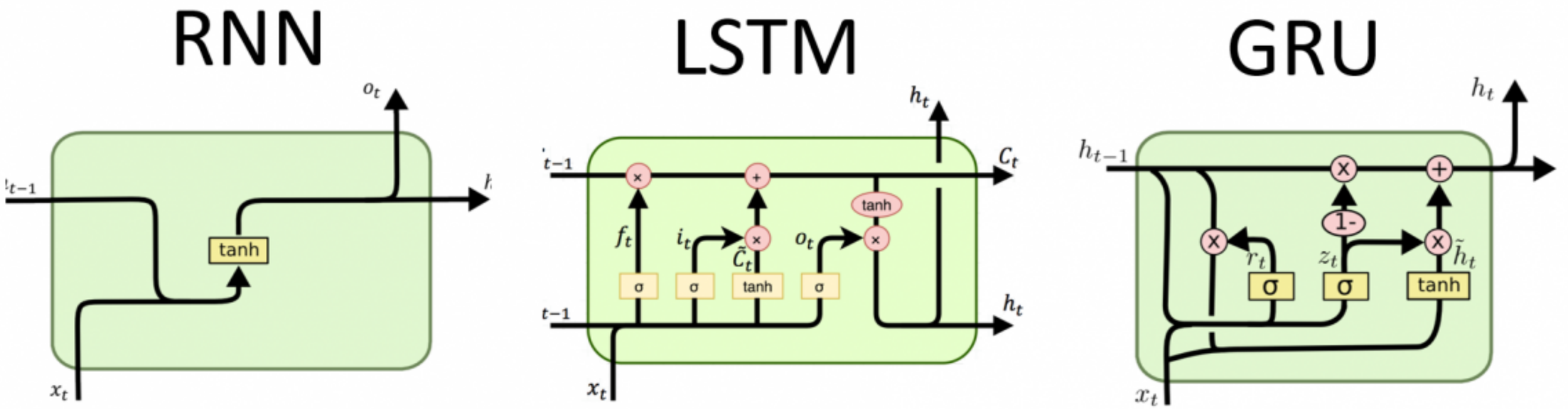
Проблемы с градиентами

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

Классическая RNN

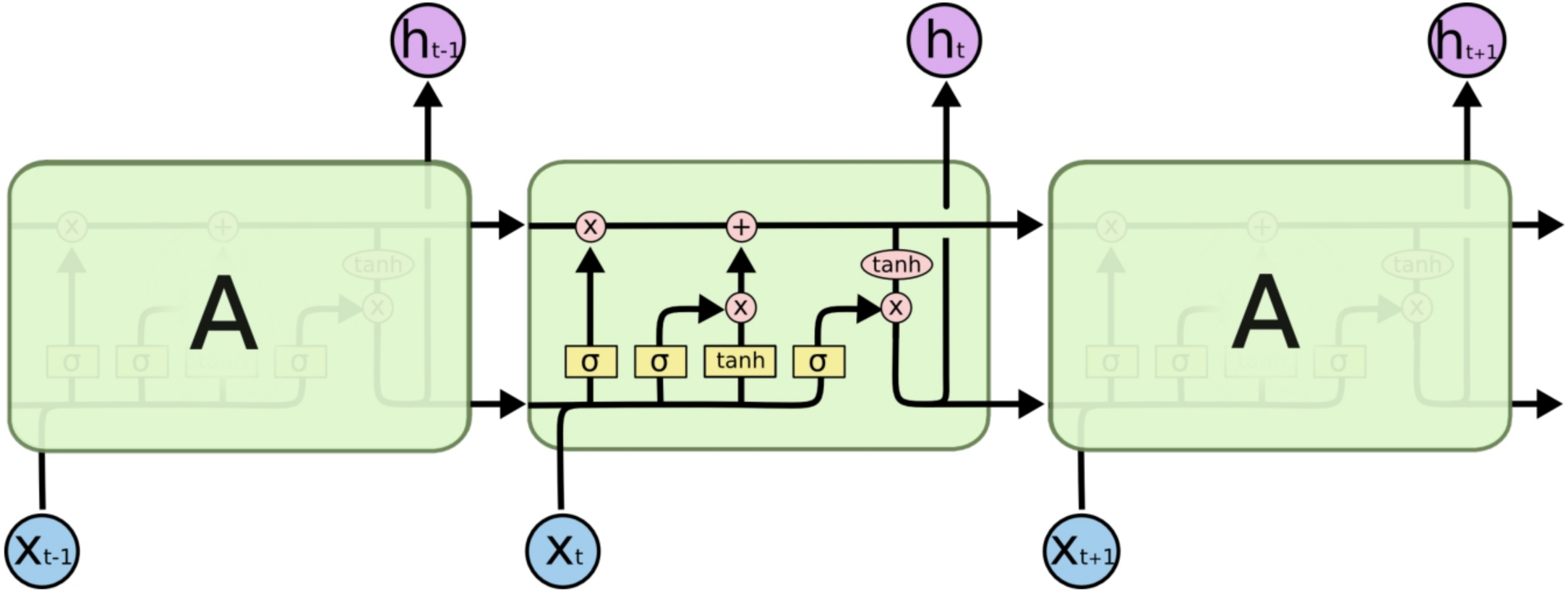


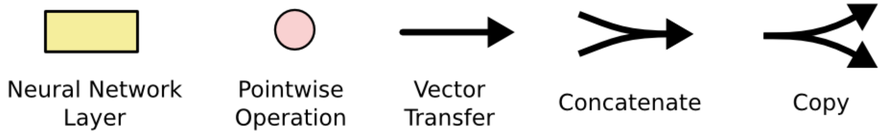
Виды рекуррентных сетей



- В LSTM 4 матрицы весов (= 4 слоя) вместо одной (в отличие от RNN)
- Матрицы весов обозначены желтыми прямоугольниками

LSTM (Long Short-Term Memory)



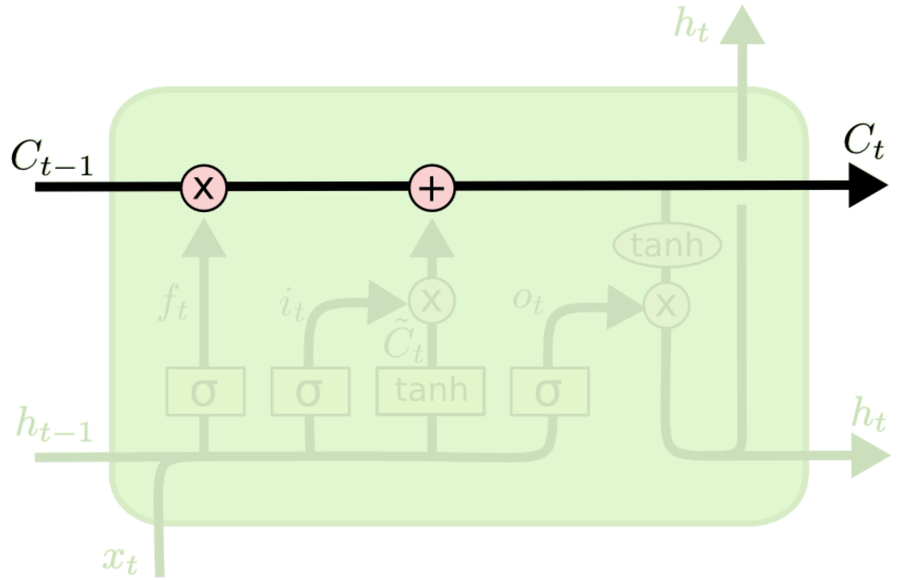


https://colah.github.io/posts/2015 -08 -Understanding -LSTMs/

LSTM: C_t - состояние ячейки

- C_t - глобальное состояние ячейки = долговременная память

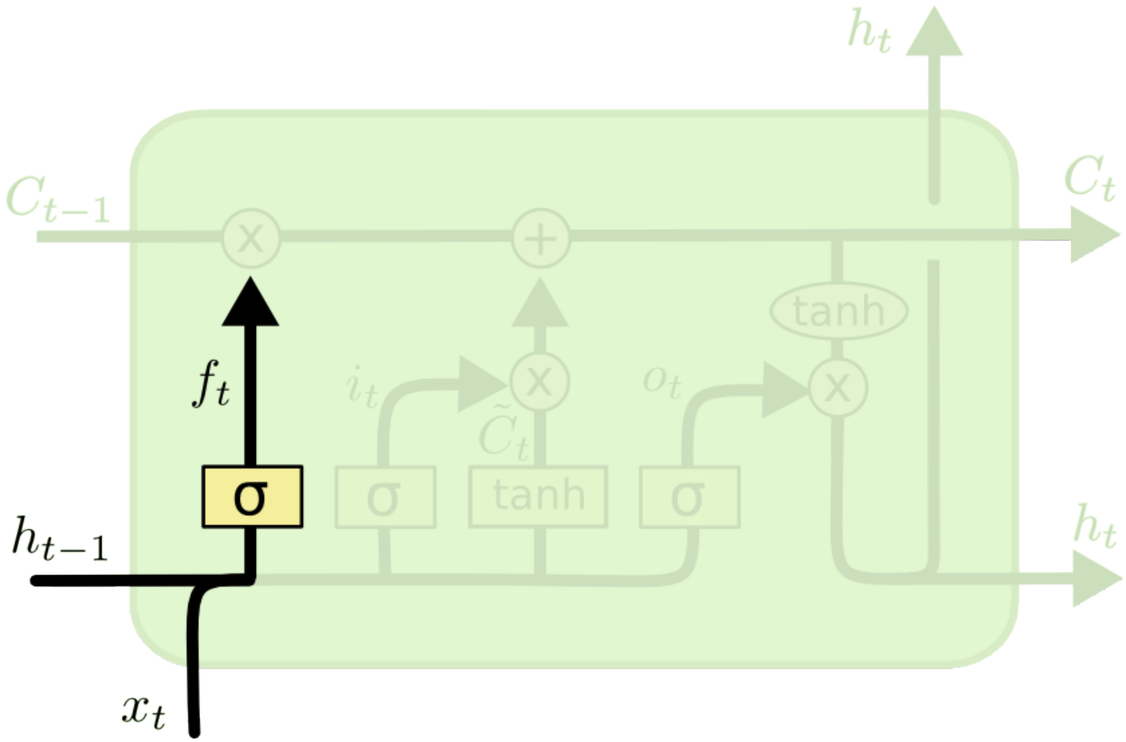
- h_t - локальное состояние ячейки = кратковременная память



В него с каждым временным шагом добавляется некоторая информация, а некоторая забывается

LSTM: f_t - forget layer

- f_t - информация с предыдущих шагов, которую хотим забыть

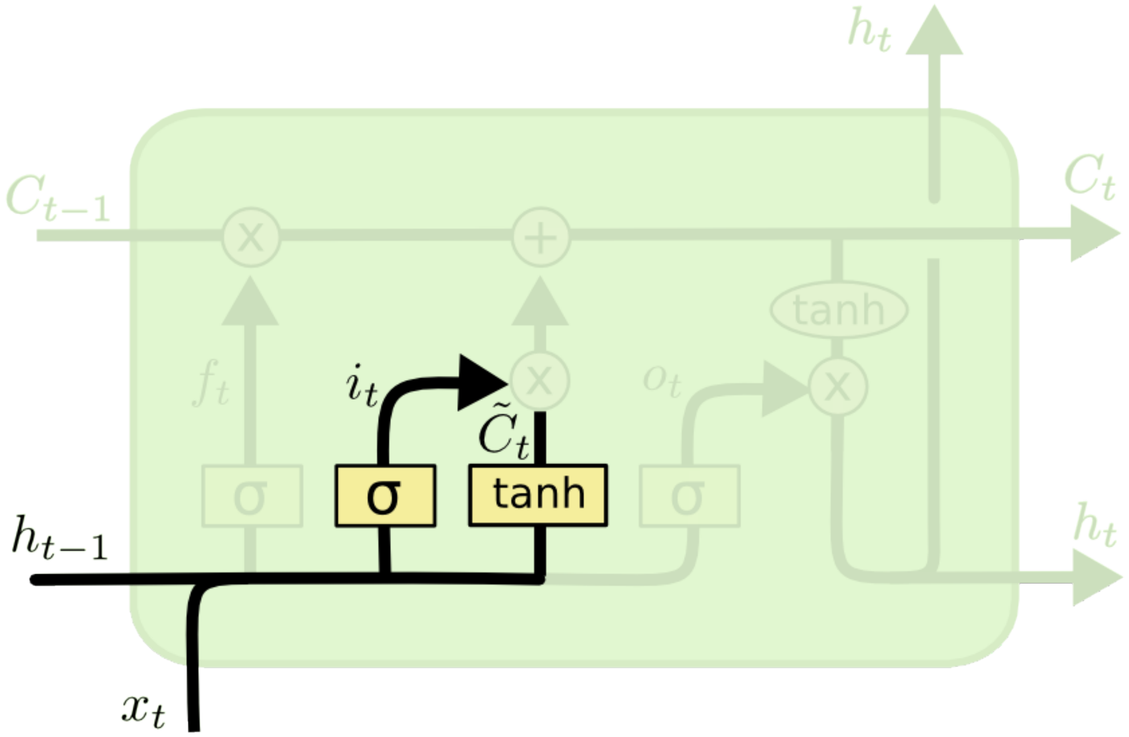


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM: учет новой информации

- i_t - вектор с “весами” значений, которые будем обновлять (исходя из новой информации на шаге t)

- $\tilde{C}_t$ - вектор с новой информацией

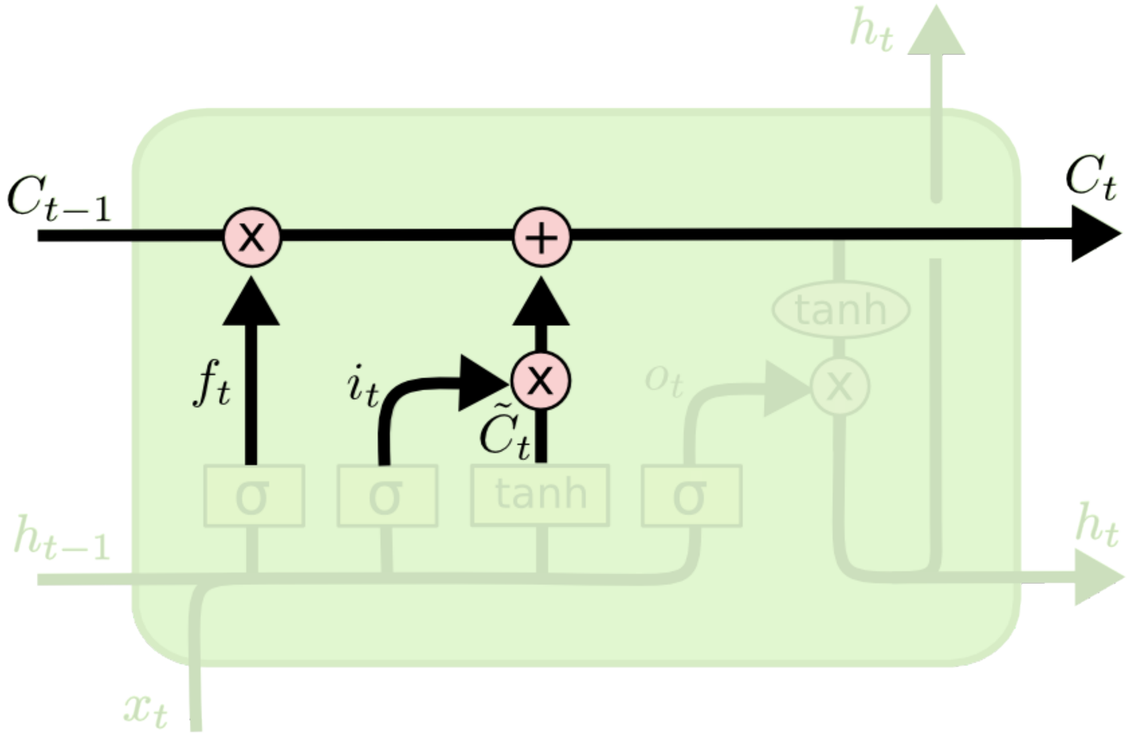


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM: обновление состояния ячейки

Обновляем состояние ячейки: часть забываем (первое слагаемое), часть добавляем (второе слагаемое)

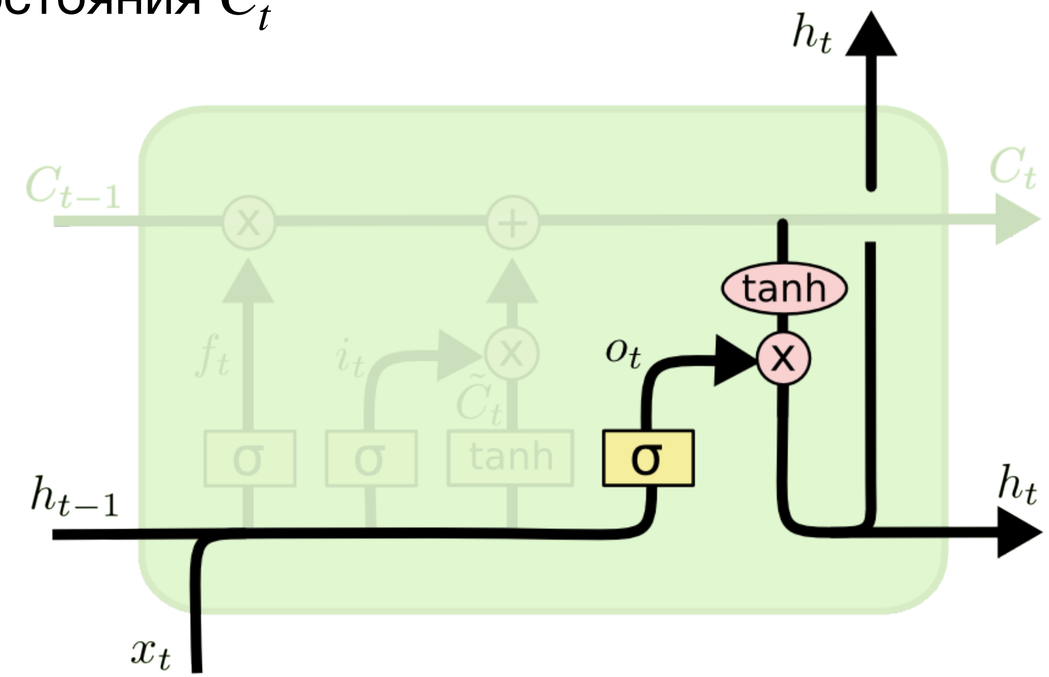


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM: прогноз и обновление h_t

- Чтобы сделать прогноз o_t на текущем шаге (например, предсказываем часть речи на каждом шаге), используем поступившую на шаге t информацию и локальное состояние h_{t-1}

- Обновляем локальное состояние h_t с учетом прогноза o_t и обновившегося глобального состояния C_t

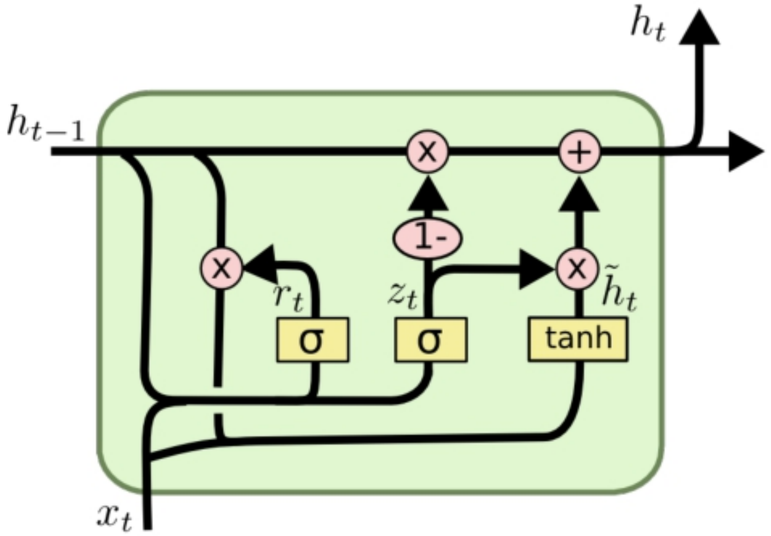


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

GRU (Gated Recurrent Unit), 2014

- Три слоя (3 матрицы весов) - логика немного отличается от логики LSTM
- Быстрее обучается, так как меньше параметров
- По качеству в большинстве задач не хуже, чем LSTM



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

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

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

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

https://colah.github.io/posts/2015 -08 -Understanding -LSTMs/

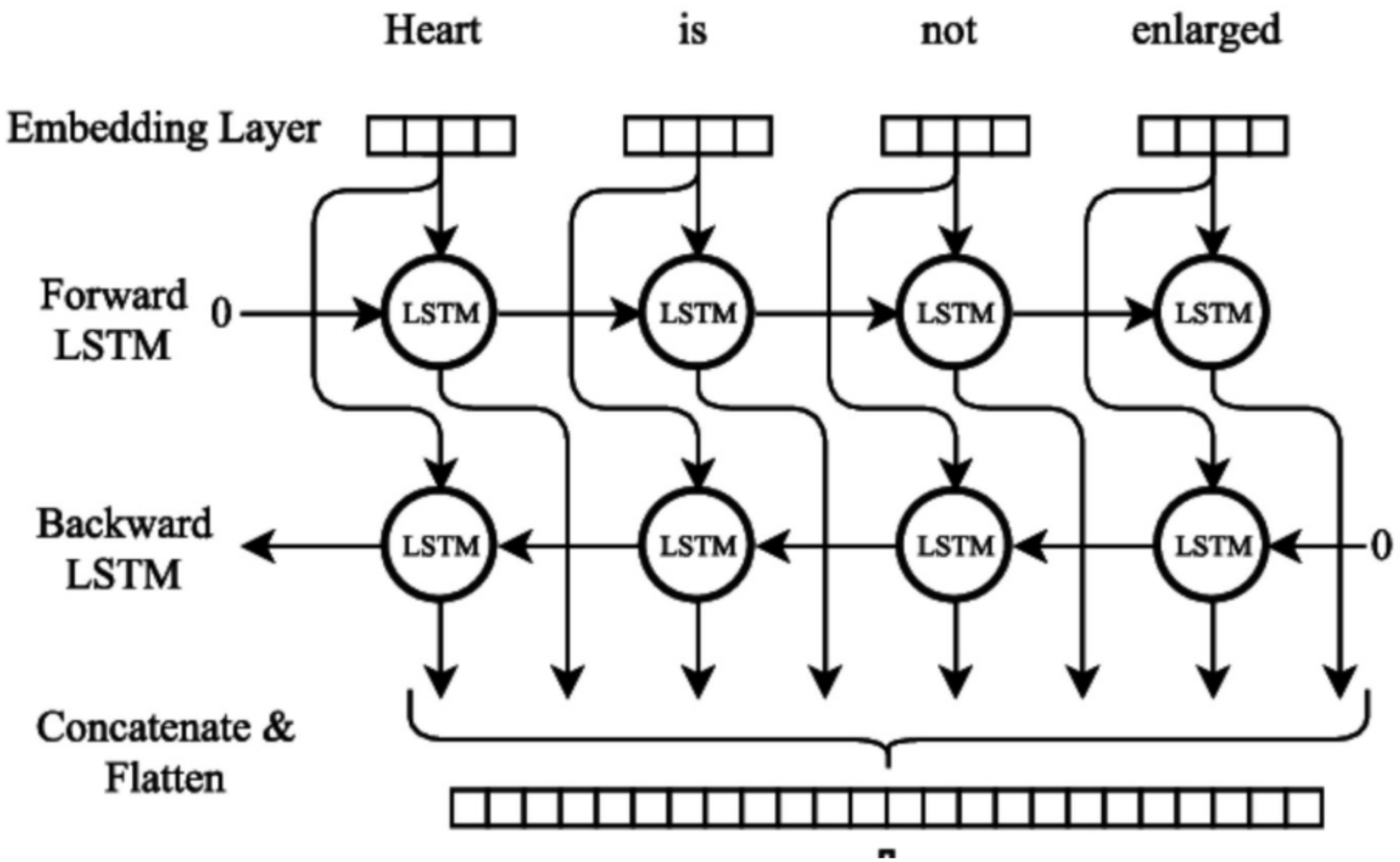
Какой вариант рекуррентных сетей лучше?

В 2015 исследователи проводили эксперименты - по качеству все модификации LSTM / RNN дают приблизительно одинаковые результаты.

Bidirectional LSTM

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

Bidirectional LSTM



https://arxiv.org/abs/1609.08409

Bidirectional LSTM

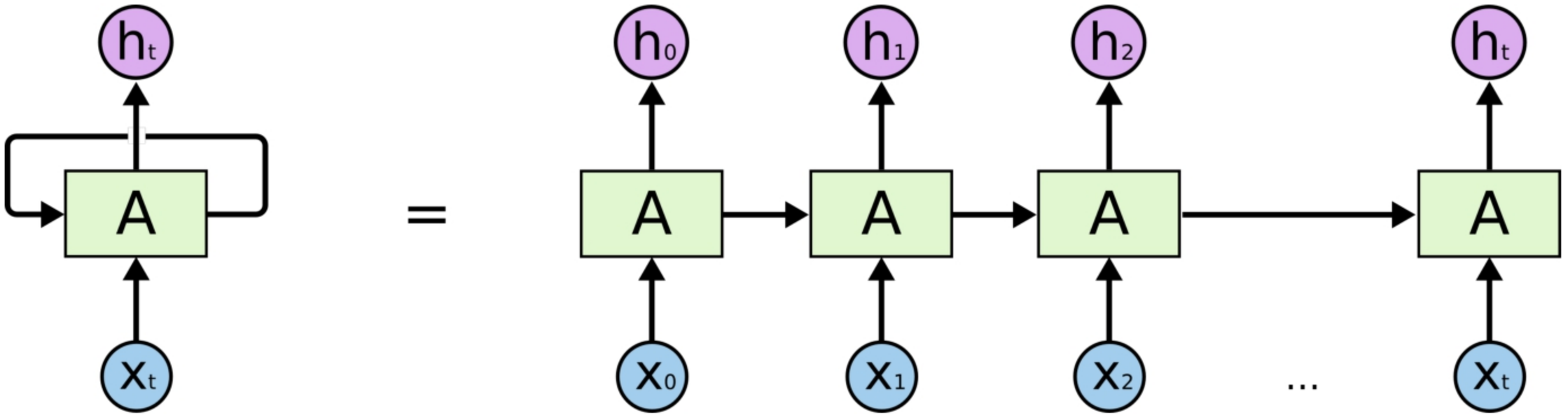
· Предсказание для слова строится по скрытым состояниям, учитывающим весь контекст

Seq2seq-архитектуры

Sequence to sequence

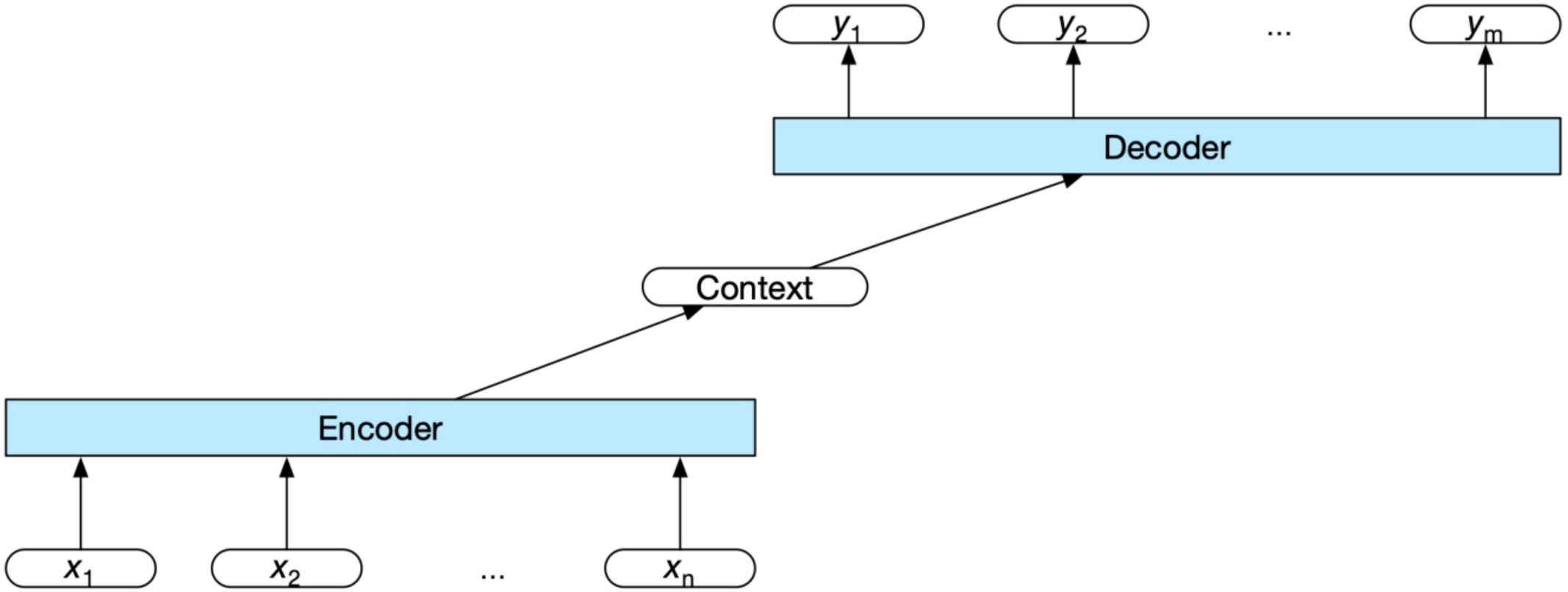
- Машинный перевод
- Суммаризация текста
- Генерация комментариев к коду
- Математические преобразования
- Смена стиля текста

Seq2seq Machine Translation



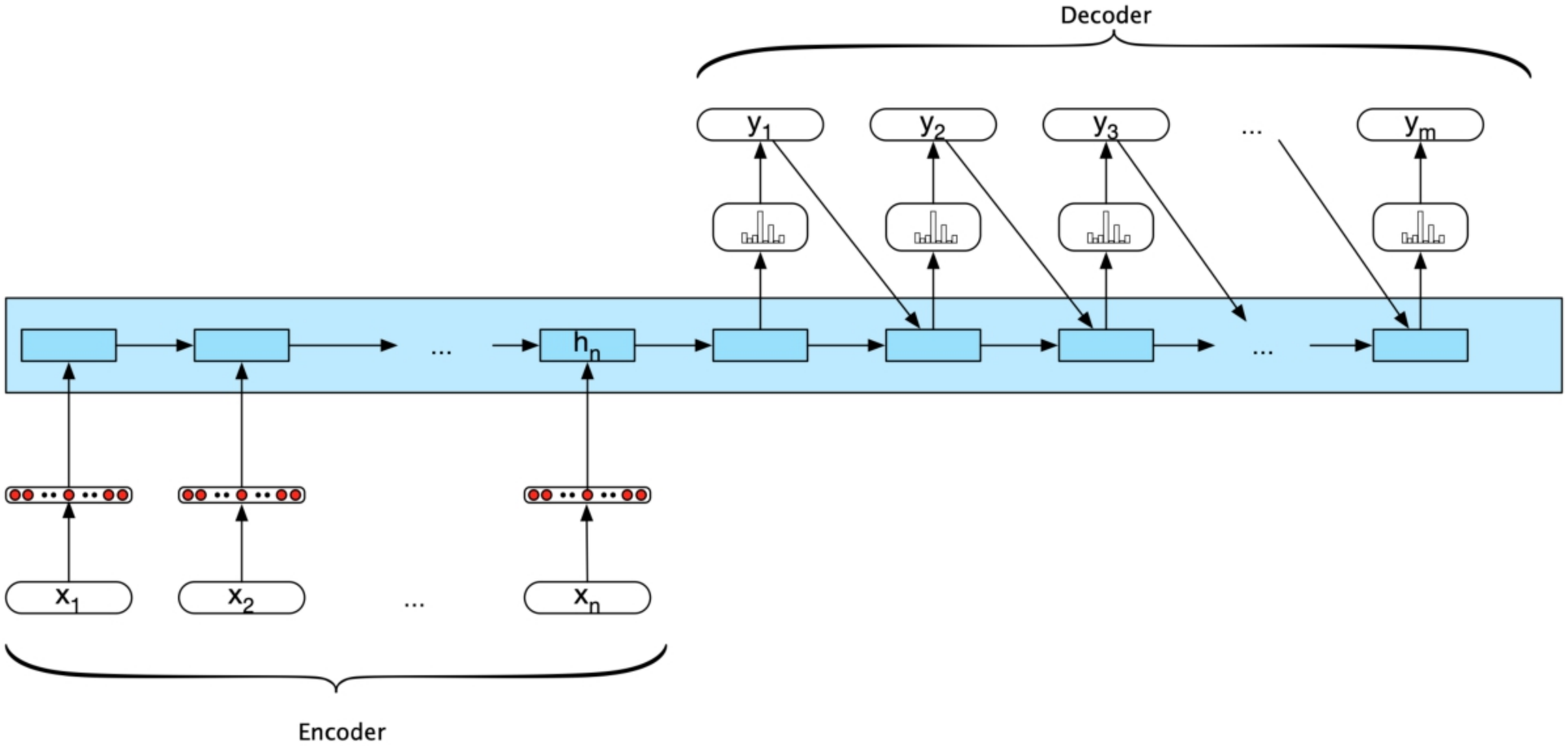
Что делать, если длины входного и выходного текстов разные?

Seq2seq Machine Translation



https://web.stanford.edu/~jurafsky/slp3/10.pdf

Seq2seq Machine Translation



https://web.stanford.edu/~jurafsky/slp3/10.pdf

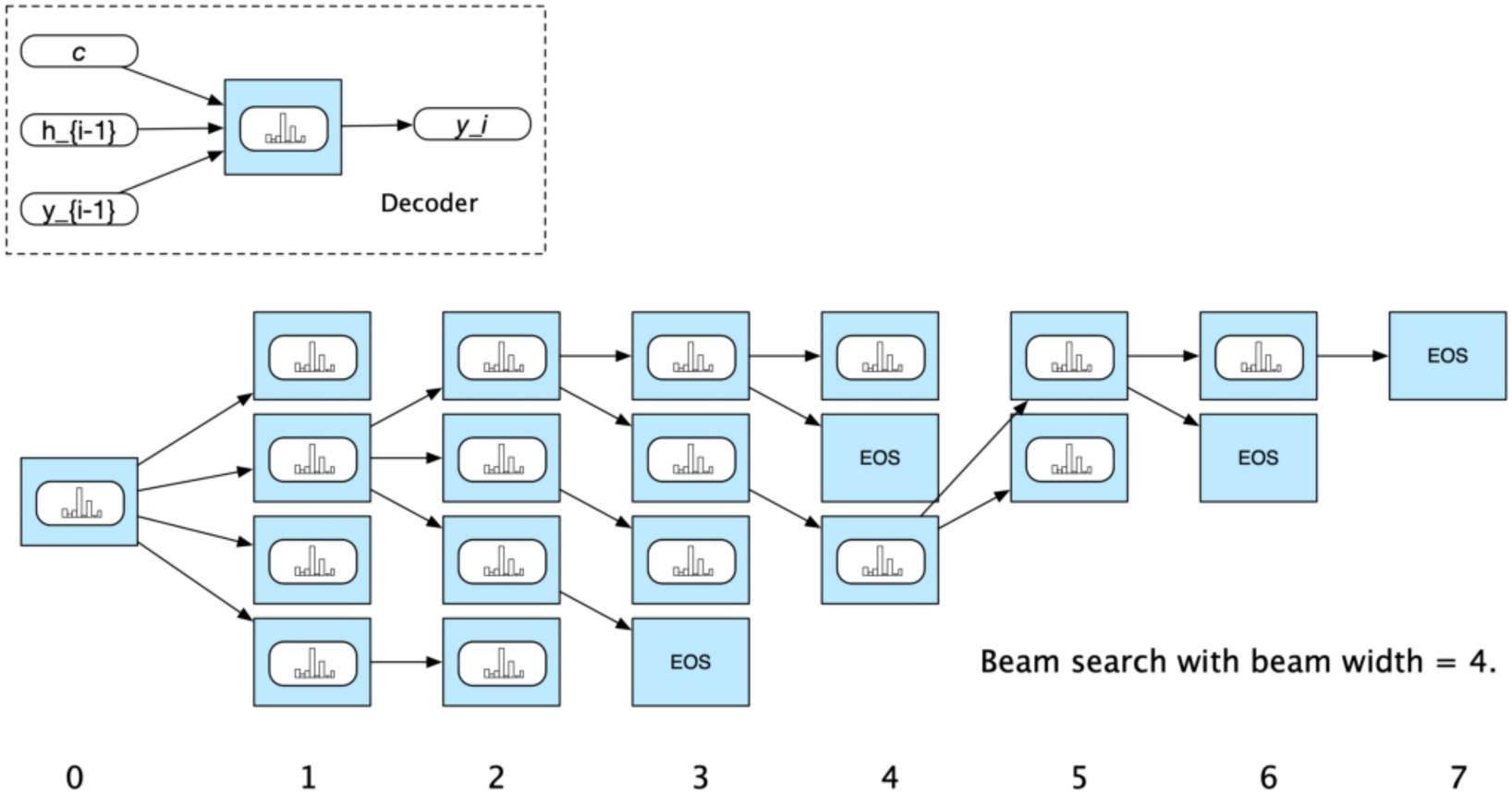
Seq2seq Machine Translation

- 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



https://web.stanford.edu/~jurafsky/slp3/10.pdf

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

https://arxiv.org/abs/1409.3215

Проблемы seq2seq- архитектуры

- Нужно сжать весь текст в один вектор
- Теряется информация о первых словах
- Декодер тоже может терять информацию по мере генерации последовательности

Проблемы seq2seq- архитектуры

- Нужно сжать весь текст в один вектор
- Теряется информация о первых словах
- Декодер тоже может терять информацию по мере генерации последовательности

- Немного улучшает ситуацию BiLSTM
- Для очень длинных текстов уже лучше применять Attention.