

# Генерация текста

# План

- Задачи генерации
- N-грамм генерация
- Рекуррентные нейронные сети

# Зачем это нужно?

## Автодополнение слова

В лесу родилась елочка

# Зачем это нужно?

## Автодополнение слова

В лесу родилась елочка

## Автодополнение фразы

Почему птицы летают  
поют  
летят клином

# Зачем это нужно?

## Автодополнение слова

В лесу родилась елочка

## Автодополнение фразы

Почему птицы летают  
поют  
летят клином

## Диалоговые системы

– Какая погода в Москве?

В Москве сейчас 8 градусов –

# Постановка задачи

Текст должен удовлетворять требованиям:

- Логическая связность
- Соблюдение языковых норм

Можно попытаться задать правила вручную.

Но правил слишком много, поэтому ничего хорошего не выйдет.

# Постановка задачи

Будем учиться имитировать человеческую речь.

Для этого научимся оценивать вероятность текстов.

$p(\text{В лесу родилась елочка})$        $p(\text{У лукоморья дуб зеленый})$

# Постановка задачи

Будем учиться имитировать человеческую речь.

Для этого научимся оценивать вероятность текстов.

$p(\text{В лесу родилась елочка}) \quad \vee \quad p(\text{У лукоморья дуб зеленый})$

$$\frac{\parallel 1}{|\text{dataset}|}$$

$$\frac{1 \parallel}{|\text{dataset}|}$$



# Постановка задачи

Будем учиться имитировать человеческую речь.

Для этого научимся оценивать вероятность текстов.

$p(\text{В лесу родилась елочка}) \quad \vee \quad p(\text{У лукоморья дуб зеленый})$

$$\frac{\parallel 1}{|\text{dataset}|}$$

$$\frac{1 \parallel}{|\text{dataset}|}$$

Работать с текстом как с одним целым невозможно!

# Постановка задачи

Разделим текст на слова.

$x$  – текст из  $m$  слов.

Обучим модель оценивать вероятность набора слов.

$$p(x) = p(x_1, \dots, x_m)$$

# Постановка задачи

Разделим текст на слова.

$x$  – текст из  $m$  слов.

Обучим модель оценивать вероятность набора слов.

$$p(x) = p(x_1, \dots, x_m)$$

Заменим совместную вероятность на произведение условных вероятностей.

$$p(x_1, \dots, x_m) = p(x_1) \cdot p(x_2 | x_1) \cdot p(x_3 | x_1, x_2) \cdot \dots \cdot p(x_m | x_1, \dots, x_{m-1}) = \prod_{i=1}^m p(x_i | x_{<i})$$

# Постановка задачи

$$p(x_1, \dots, x_m) = \prod_{i=1}^m p(x_i | x_{<i})$$

Достаточно обучить модель оценивать вероятность  $p(x_i | x_{<i})$ .

# Постановка задачи

$$p(x_1, \dots, x_m) = \prod_{i=1}^m p(x_i | x_{<i})$$

Достаточно обучить модель оценивать вероятность  $p(x_i | x_{<i})$ .

Все еще сложно, потому что надо учитывать все предыдущие слова.

Упростим задачу – будем смотреть только на предыдущие  $n$  слов.

$$p(x_1, \dots, x_m) \approx \prod_{i=1}^m p(x_i | x_{i-1}, \dots, x_{i-n})$$

# Постановка задачи

$$p(x_1, \dots, x_m) = \prod_{i=1}^m p(x_i | x_{<i})$$

Достаточно обучить модель оценивать вероятность  $p(x_i | x_{<i})$ .

Все еще сложно, потому что надо учитывать все предыдущие слова.

Упростим задачу – будем смотреть только на предыдущие  $n$  слов.

$$p(x_1, \dots, x_m) \approx \prod_{i=1}^m p(x_i | x_{i-1}, \dots, x_{i-n})$$

Такая модель называется  $n$ -граммной.

# N-грамм модель генерации

Предполагаем, что следующее слово зависит только от  $n$  предыдущих

$$p(x_1, \dots, x_m) \approx \prod_{i=1}^m p(x_i | x_{i-1}, \dots, x_{i-n})$$

# N-грамм модель генерации

Предполагаем, что следующее слово зависит только от  $n$  предыдущих

$$p(x_1, \dots, x_m) \approx \prod_{i=1}^m p(x_i | x_{i-1}, \dots, x_{i-n})$$

Если у слова меньше, чем  $n$  предыдущих, дополним пропуски символом <PAD>.

$$\begin{aligned} p(\text{В}, \text{лесу}, \text{родилась}, \text{елочка}) = & \\ & p(\text{В} \mid \text{<PAD>}, \text{<PAD>}) \\ & \cdot p(\text{лесу} \mid \text{<PAD>}, \text{В}) \\ & \cdot p(\text{родилась} \mid \text{В}, \text{лесу}) \\ & \cdot p(\text{елочка} \mid \text{лесу}, \text{родилась}) \end{aligned}$$



# N-грамм модель генерации: обучение

- Нужно оценить вероятность  $p(x_i | x_{i-1}, \dots, x_{i-n})$ .
- Посчитаем вручную, сколько раз  $x_i$  встретилось после  $x_{i-1}, \dots, x_{i-n}$ .

$$p(x_i | x_{i-1}, \dots, x_{i-n}) = \frac{p(x_i, x_{i-1}, \dots, x_{i-n})}{p(x_{i-1}, \dots, x_{i-n})} = \frac{N(x_i, x_{i-1}, \dots, x_{i-n})}{N(x_{i-1}, \dots, x_{i-n})}$$

# N-грамм модель генерации: обучение

- Нужно оценить вероятность  $p(x_i | x_{i-1}, \dots, x_{i-n})$ .
- Посчитаем вручную, сколько раз  $x_i$  встретилось после  $x_{i-1}, \dots, x_{i-n}$ .

$$p(x_i | x_{i-1}, \dots, x_{i-n}) = \frac{p(x_i, x_{i-1}, \dots, x_{i-n})}{p(x_{i-1}, \dots, x_{i-n})} = \frac{N(x_i, x_{i-1}, \dots, x_{i-n})}{N(x_{i-1}, \dots, x_{i-n})}$$

$$N(\text{В, лесу, родилась}) = 2$$

$$N(\text{В, лесу, обитает}) = 5$$

$$N(\text{В, лесу, было}) = 1$$

$$N(\text{В, лесу}) = 8$$

# N-грамм модель генерации: обучение

- Нужно оценить вероятность  $p(x_i | x_{i-1}, \dots, x_{i-n})$ .
- Посчитаем вручную, сколько раз  $x_i$  встретилось после  $x_{i-1}, \dots, x_{i-n}$ .

$$p(x_i | x_{i-1}, \dots, x_{i-n}) = \frac{p(x_i, x_{i-1}, \dots, x_{i-n})}{p(x_{i-1}, \dots, x_{i-n})} = \frac{N(x_i, x_{i-1}, \dots, x_{i-n})}{N(x_{i-1}, \dots, x_{i-n})}$$

$$N(\text{В, лесу, родилась}) = 2 \qquad p(\text{родилась} | \text{В, лесу}) = \frac{2}{8}$$

$$N(\text{В, лесу, обитает}) = 5 \qquad \longrightarrow \qquad p(\text{обитает} | \text{В, лесу}) = \frac{5}{8}$$

$$N(\text{В, лесу, было}) = 1 \qquad p(\text{было} | \text{В, лесу}) = \frac{1}{8}$$

$$N(\text{В, лесу}) = 8$$

# N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

# N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

<PAD> <PAD> \_

$$p(\text{В} \mid \text{<PAD> <PAD>}) = 0.3$$

$$p(\text{Я} \mid \text{<PAD> <PAD>}) = 0.2$$

$$p(\text{Из} \mid \text{<PAD> <PAD>}) = 0.15$$

# N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

<PAD> <PAD> B \_

$$p(\text{доме} \mid \text{<PAD> B}) = 0.34$$

$$p(\text{лесу} \mid \text{<PAD> B}) = 0.23$$

$$p(\text{машине} \mid \text{<PAD> B}) = 0.09$$

# N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

<PAD> <PAD> В лесу \_

$$p(\text{было} \mid \text{В лесу}) = 0.4$$

$$p(\text{воют} \mid \text{В лесу}) = 0.23$$

$$p(\text{родилась} \mid \text{В лесу}) = 0.09$$

# N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

<PAD> <PAD> В лесу было \_

$$p(\text{темно} \mid \text{лесу было}) = 0.22$$

$$p(\text{холодно} \mid \text{лесу было}) = 0.2$$

$$p(\text{свежо} \mid \text{лесу было}) = 0.13$$



# N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

<PAD> <PAD> В лесу было темно

# Преимущества n-грамм

- Тексты состоят из существующих n-грамм.
- Поэтому предложения грамматически верные.
- Модель проста в реализации и очень быстрая.

# Недостатки $n$ -грамм

- При генерации смотрит только на последние  $n$  слов.
- Из-за этого получаются логически несвязные тексты.
- При увеличении  $n$  вероятности слов оцениваются хуже.
- Из-за большого размера словаря многие  $n$ -граммы встречаются очень редко.

# Недостатки $n$ -грамм

- При генерации смотрит только на последние  $n$  слов.
- Из-за этого получаются логически несвязные тексты.
- При увеличении  $n$  вероятности слов оцениваются хуже.
- Из-за большого размера словаря многие  $n$ -граммы встречаются очень редко.

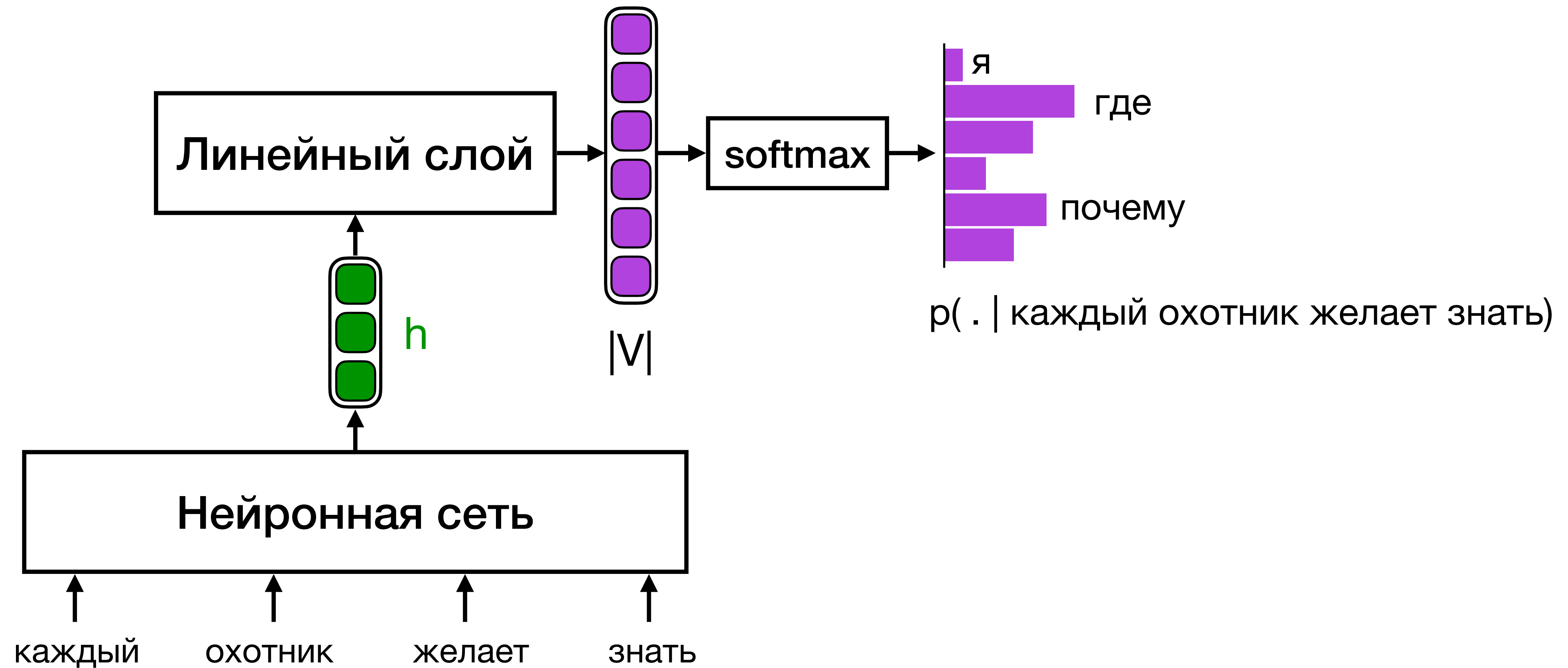
**Как уменьшить размер словаря?**

# Уменьшение размера словаря

- Удаление стоп-слов
- Лемматизация
- Стемминг
- Использование частей слов вместо целых слов

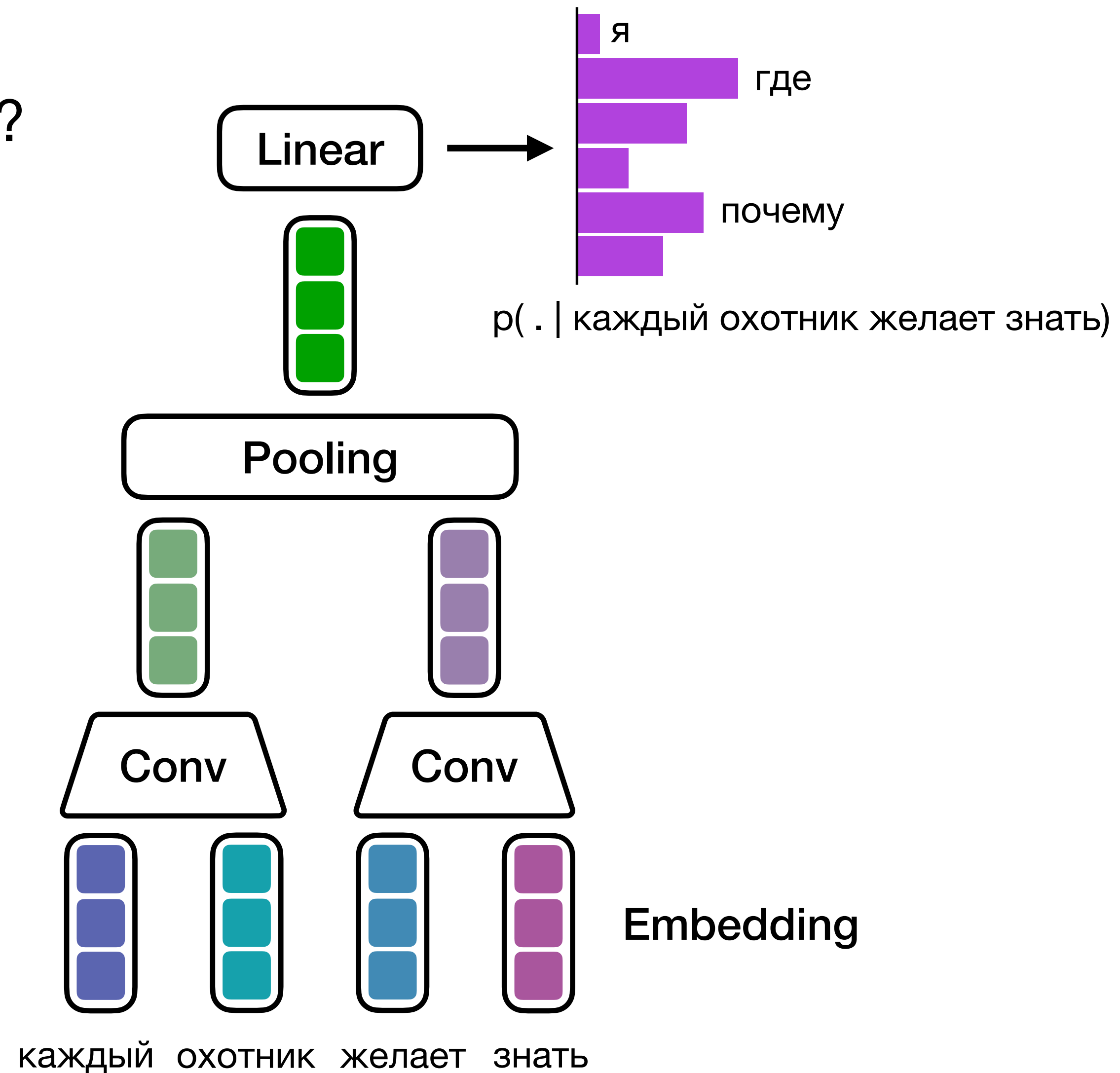
# **Рекуррентные нейронные сети (RNN)**

# Нейронные сети для генерации текста



# Сверточные нейронные сети

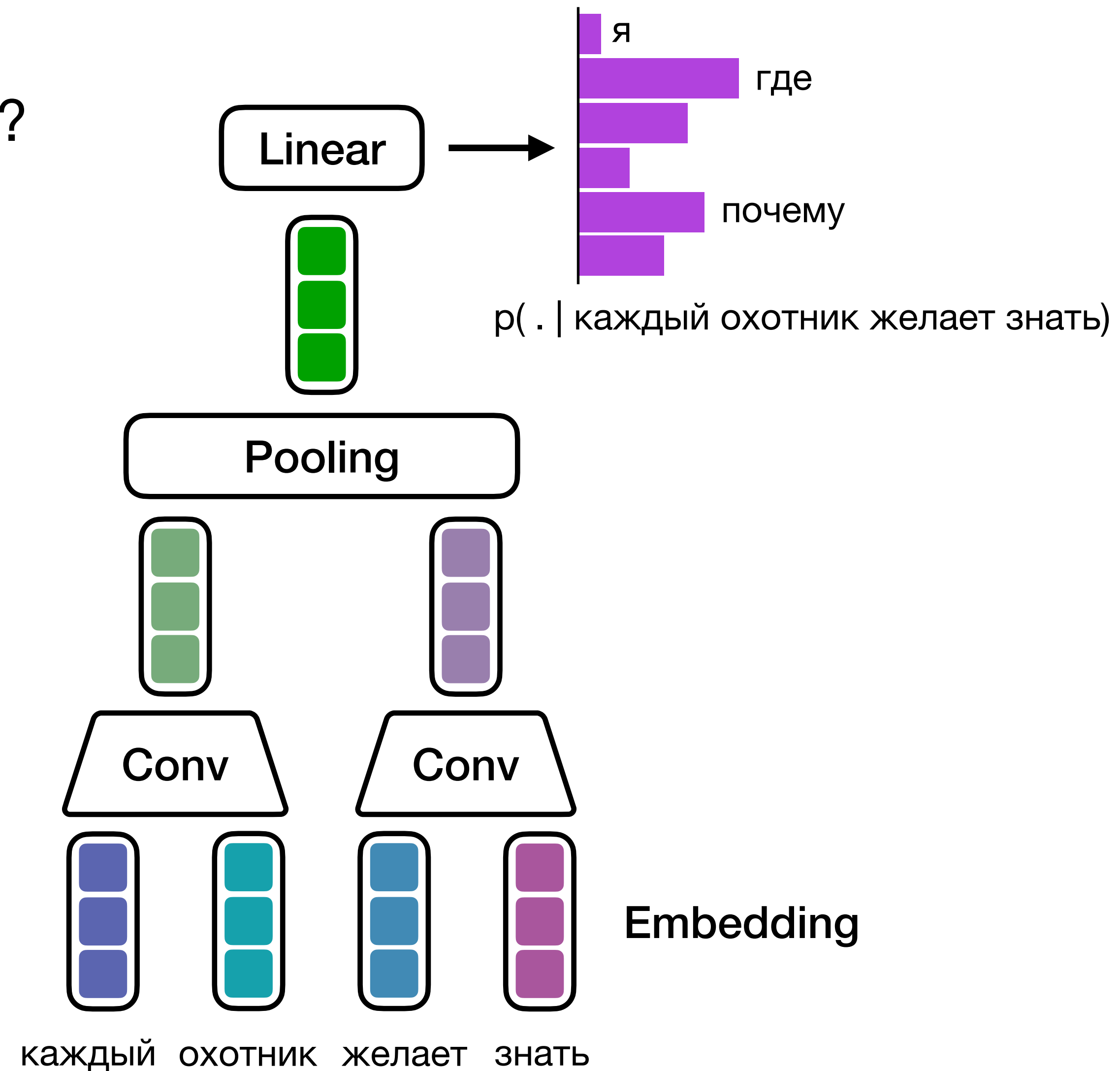
- Можно ли использовать CNN для генерации?





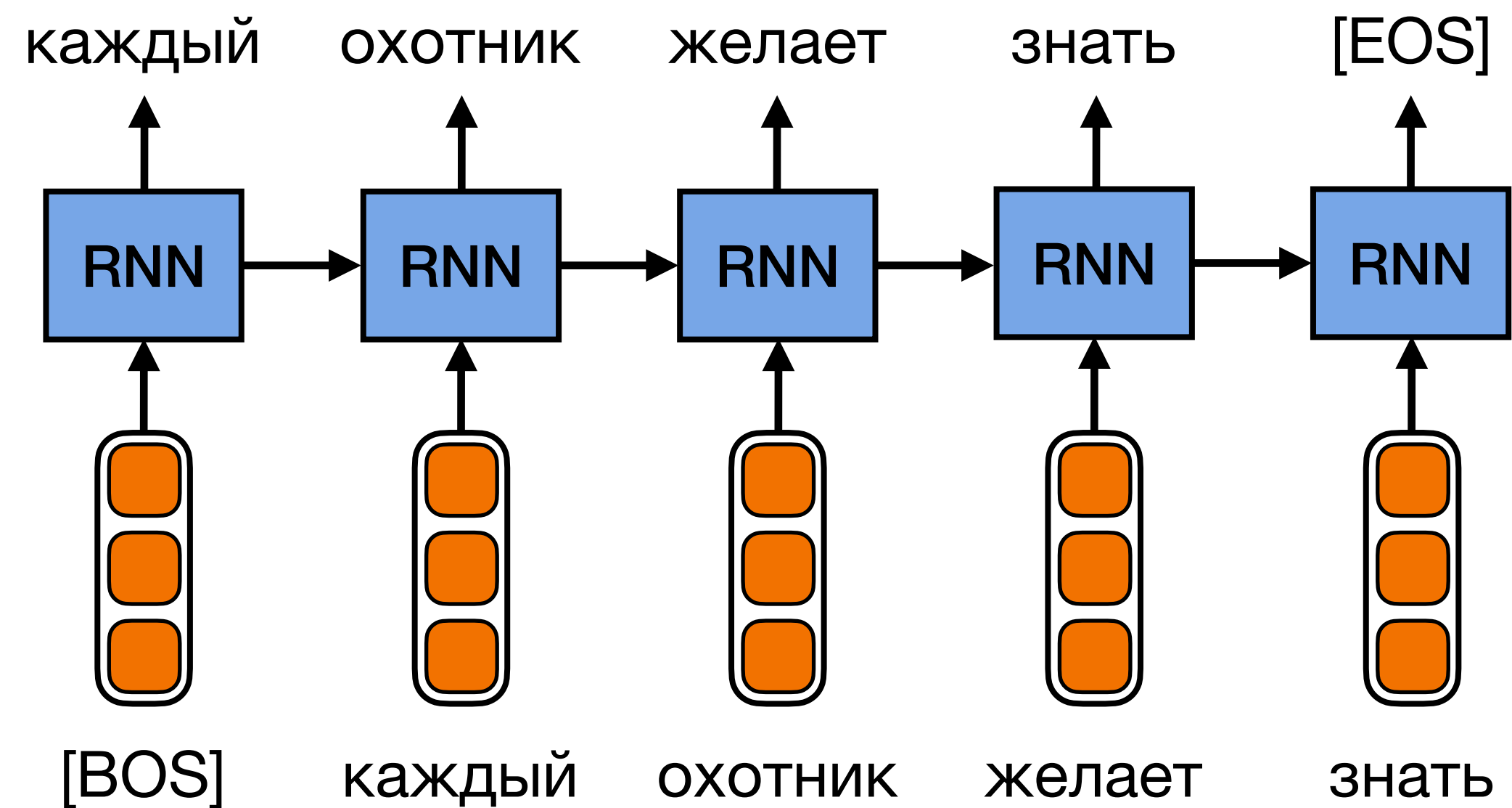
# Сверточные нейронные сети

- Можно ли использовать CNN для генерации?
- Можно, но не нужно
- Информация будет размываться при увеличении длины текста
- Обучение не эффективно



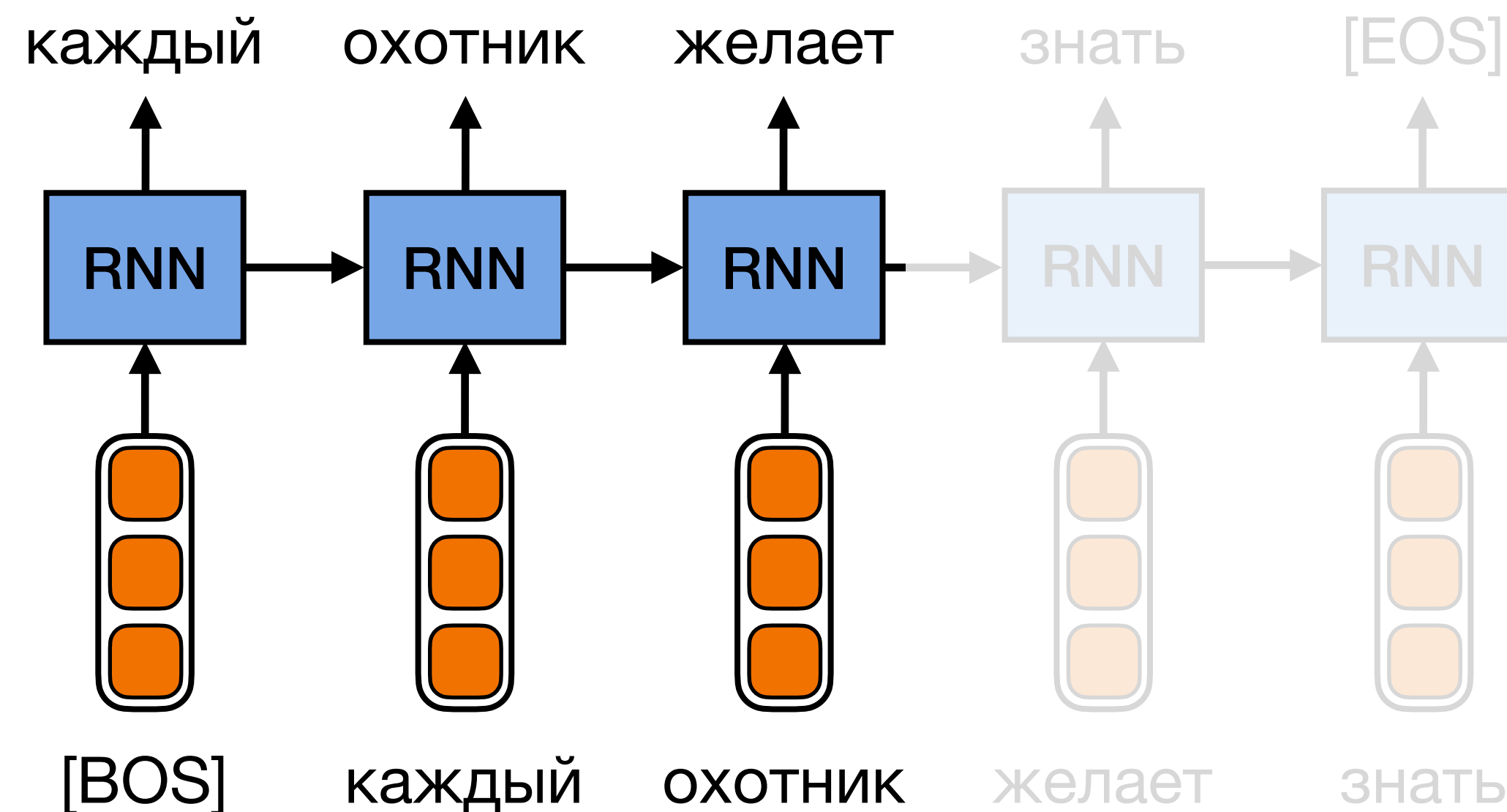
# Recurrent neural networks (RNN)

- Разработаны для работы с последовательными данными.
- Каждый блок предсказывает следующий токен.



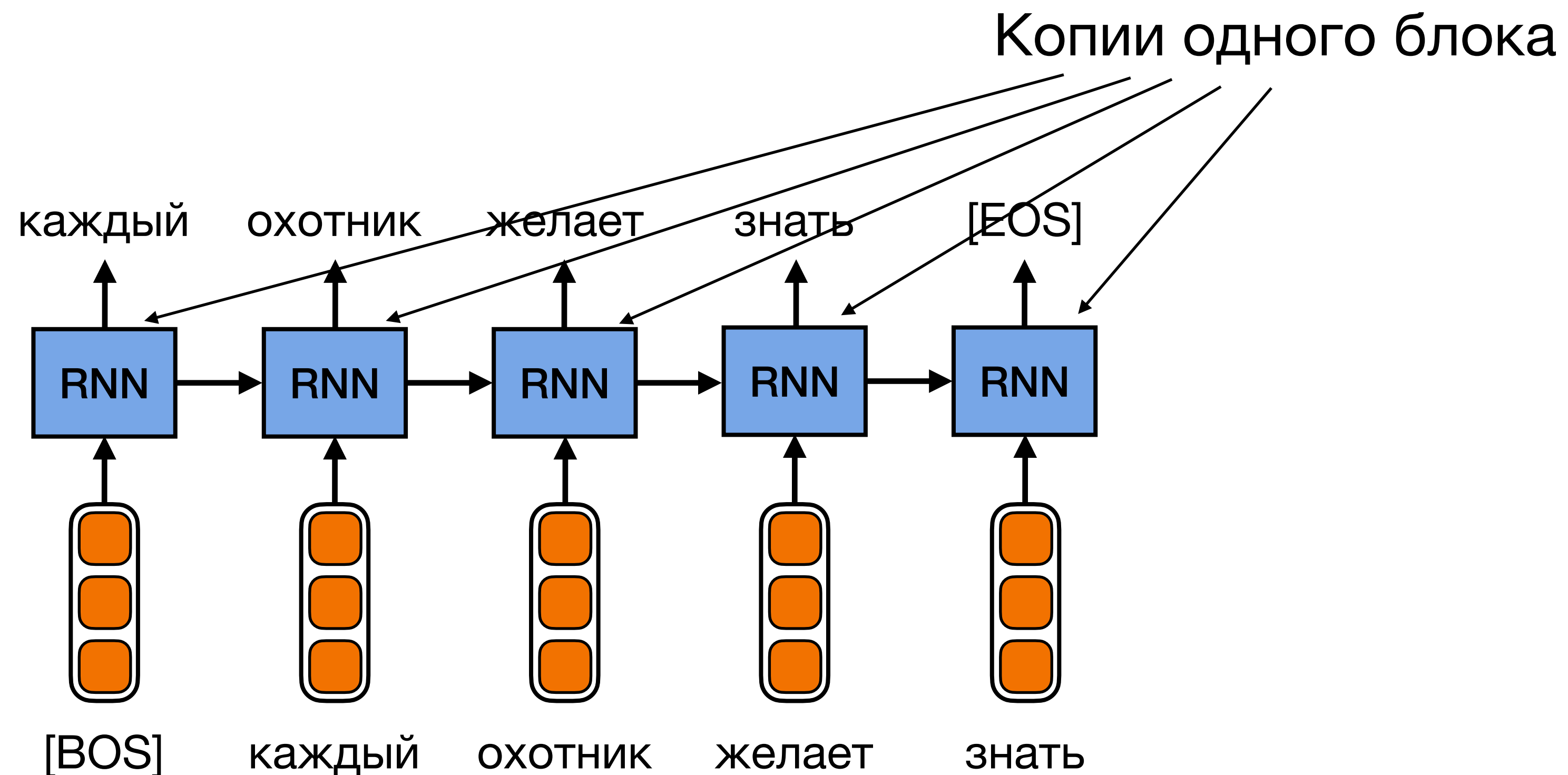
# Recurrent neural networks (RNN)

- Разработаны для работы с последовательными данными.
- Каждый блок предсказывает следующий токен.
- Процесс генерации интуитивно понятен.

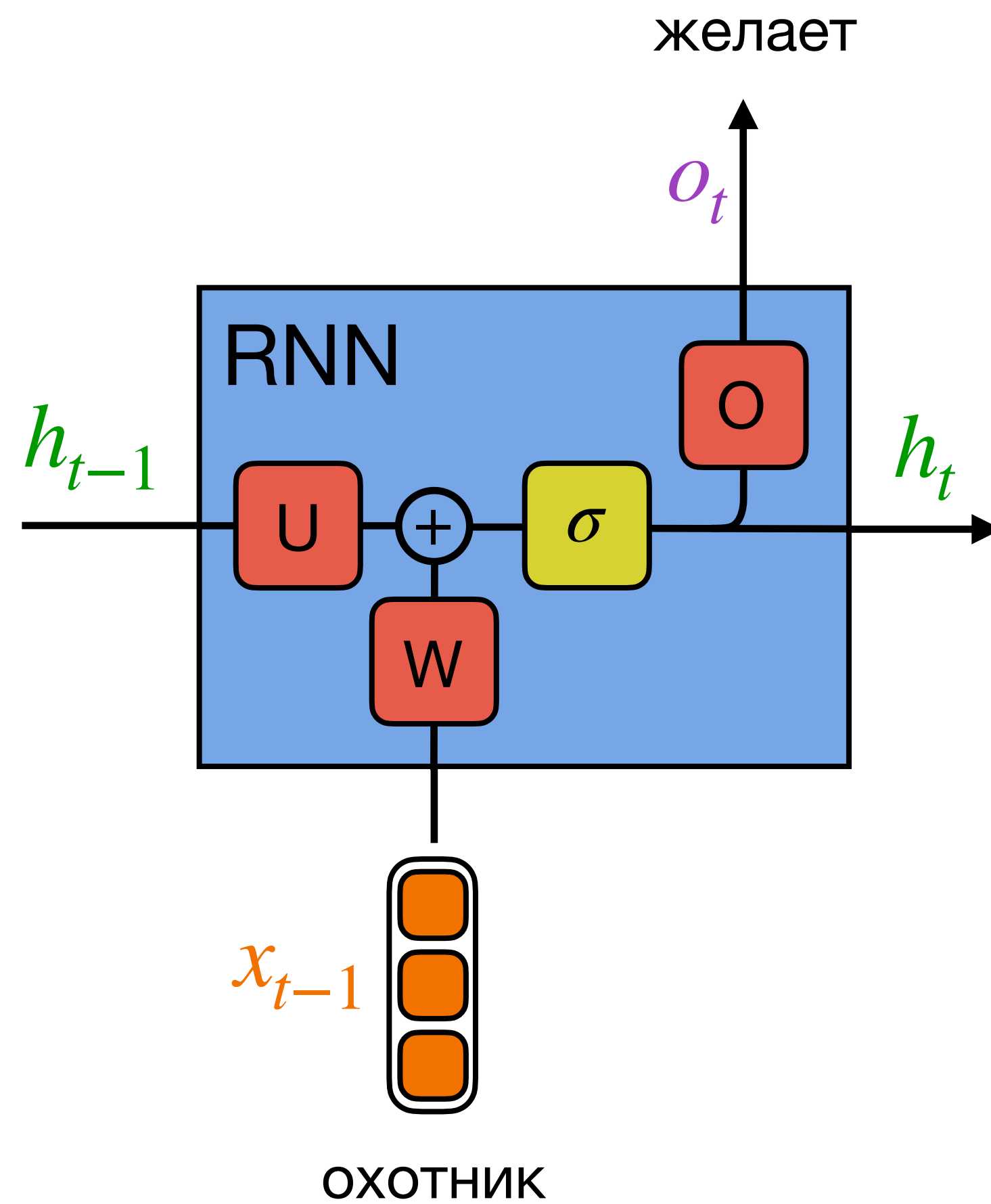


# Recurrent neural networks (RNN)

- Разработаны для работы с последовательными данными.
- Каждый блок предсказывает следующий токен.



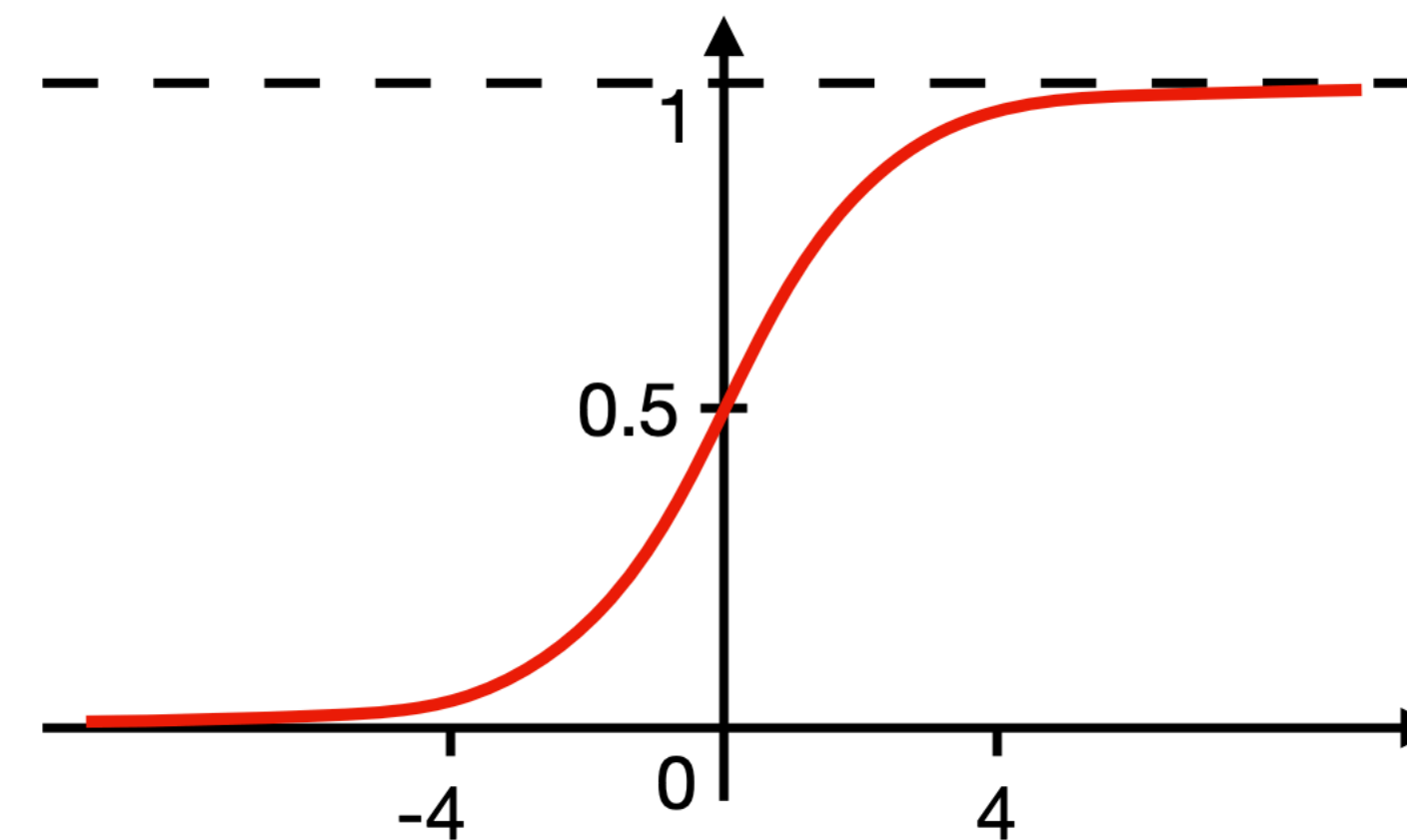
# Блок RNN



$$h_t = \sigma(Wx_{t-1} + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

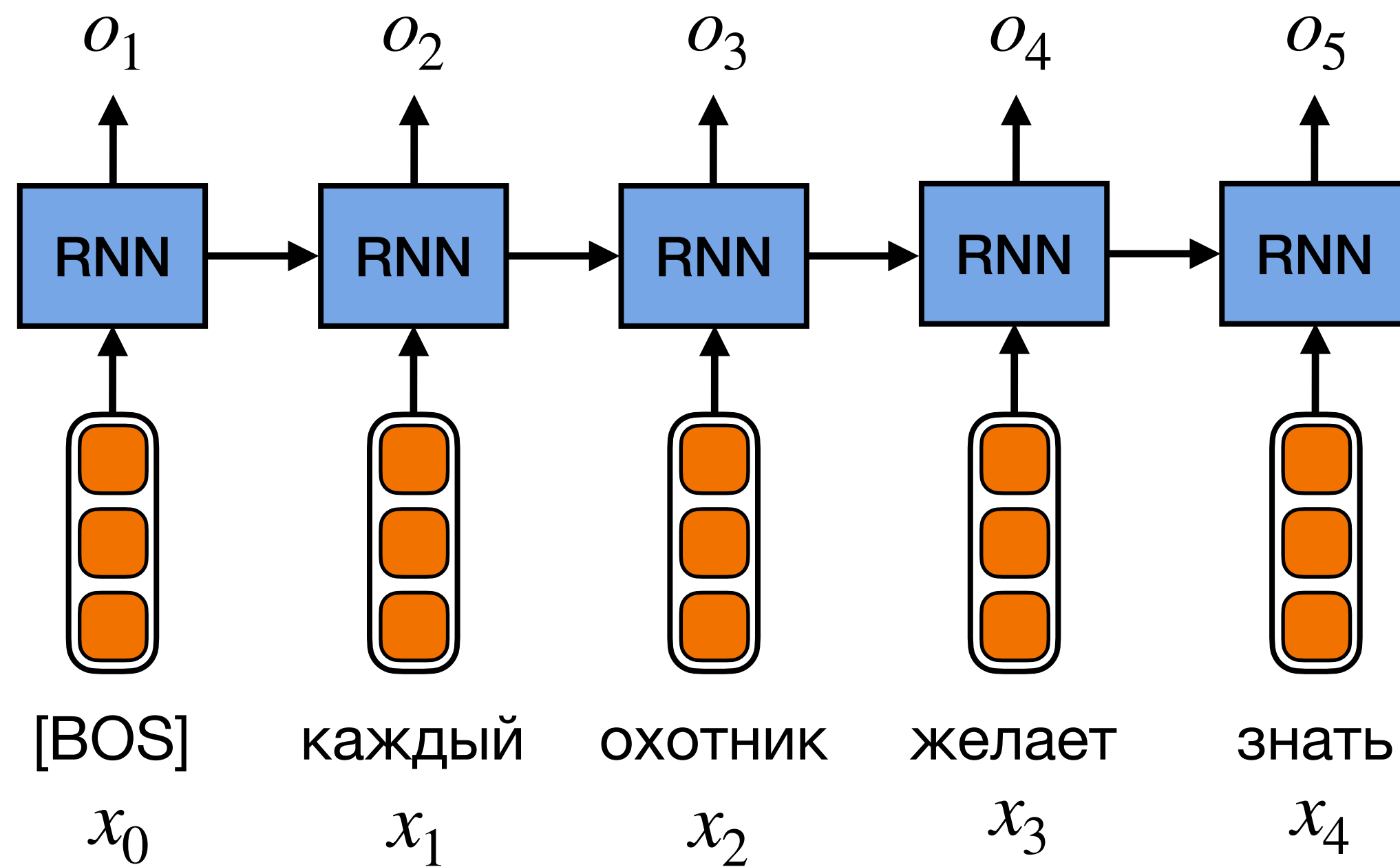
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



# RNN: Обучение

$$p(x_1, \dots, x_m) = \prod_{t=1}^m p(x_t | x_{<t}) \rightarrow \max$$

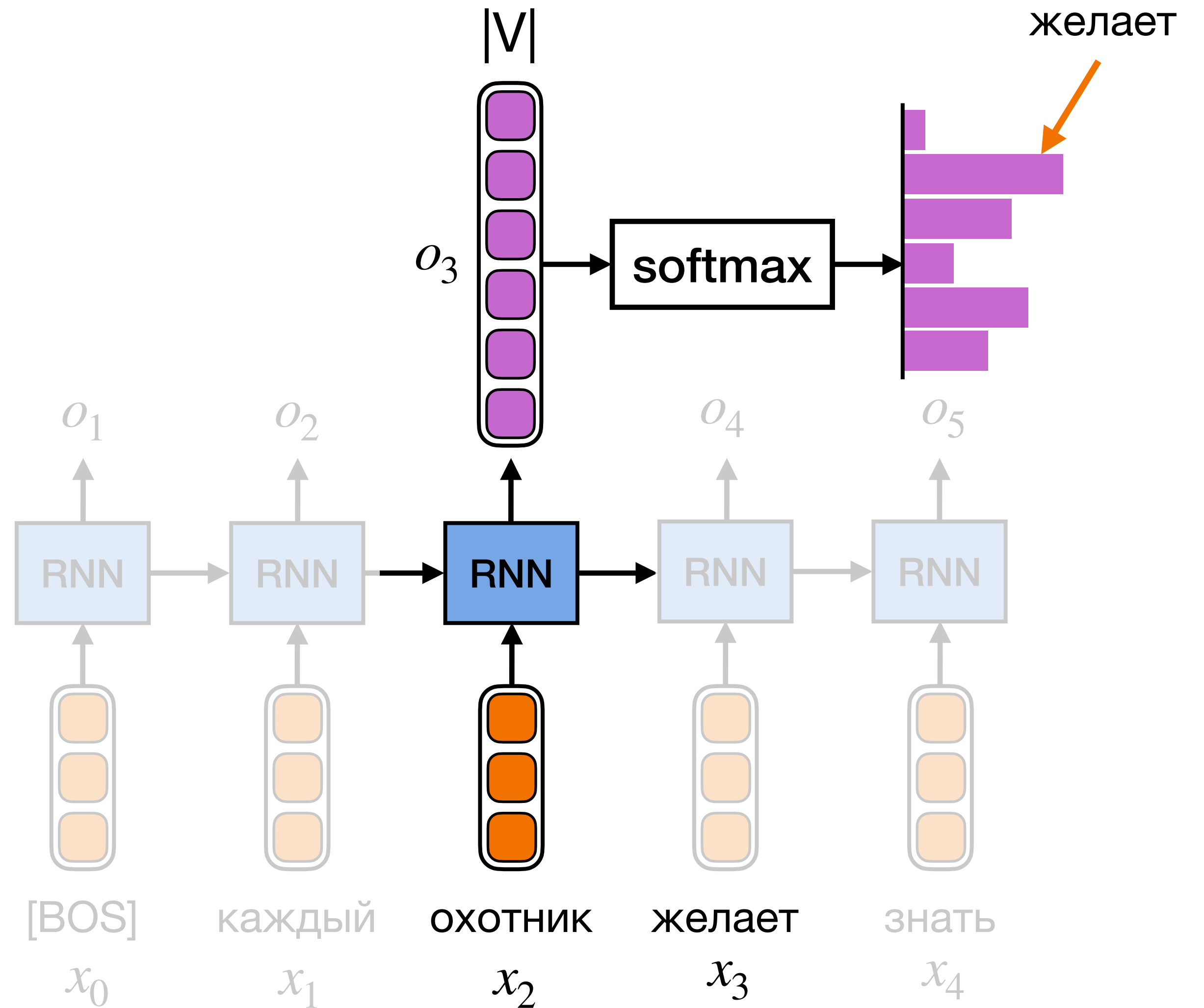
$$p(\cdot | x_{<t}) = \text{softmax}(o_t)$$



# RNN: Обучение

$$p(x_1, \dots, x_m) = \prod_{t=1}^m p(x_t | x_{<t}) \rightarrow \max$$

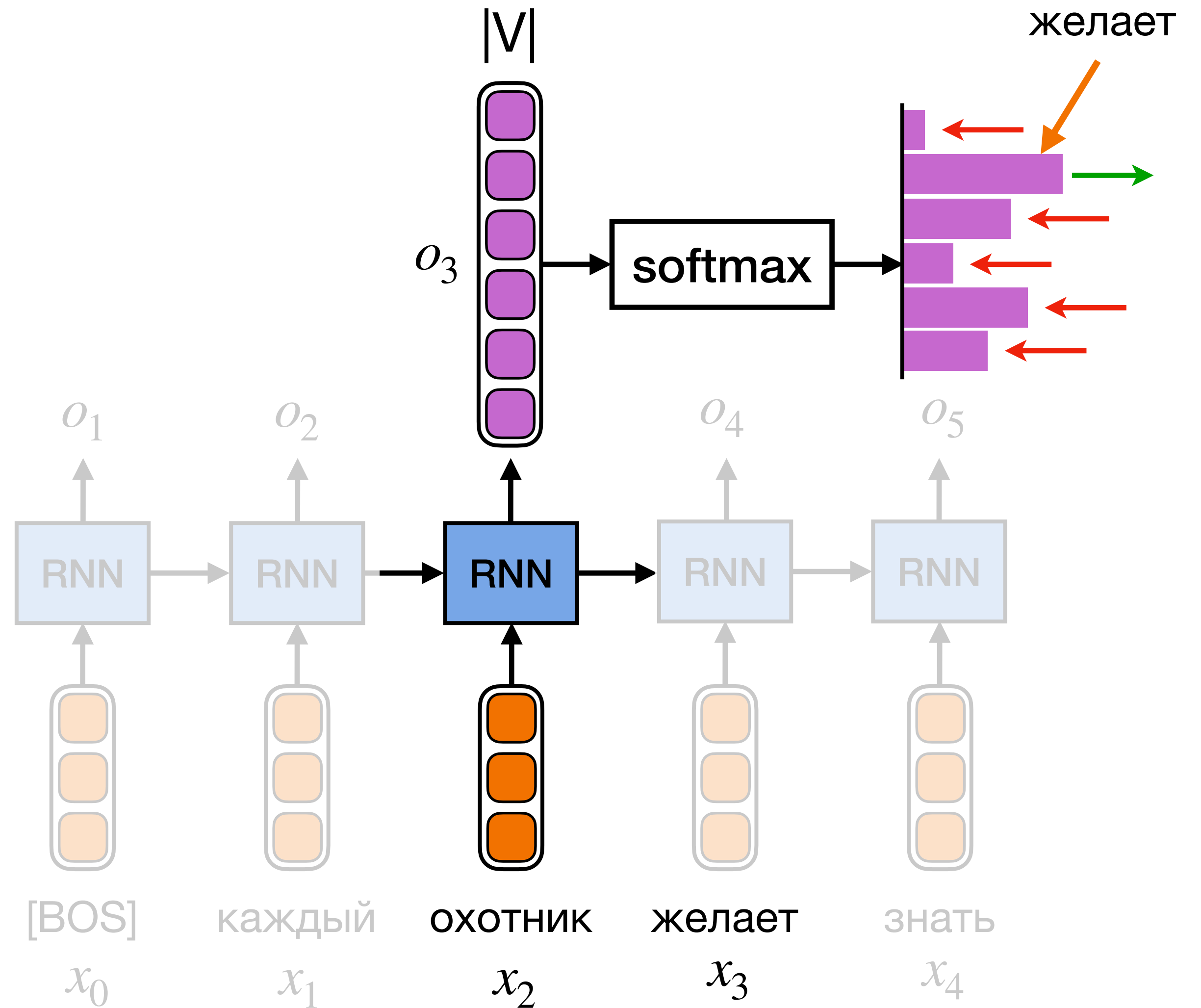
$$p(\cdot | x_{<t}) = \text{softmax}(o_t)$$



# RNN: Обучение

$$p(x_1, \dots, x_m) = \prod_{t=1}^m p(x_t | x_{<t}) \rightarrow \max$$

$$p(\cdot | x_{<t}) = \text{softmax}(o_t)$$





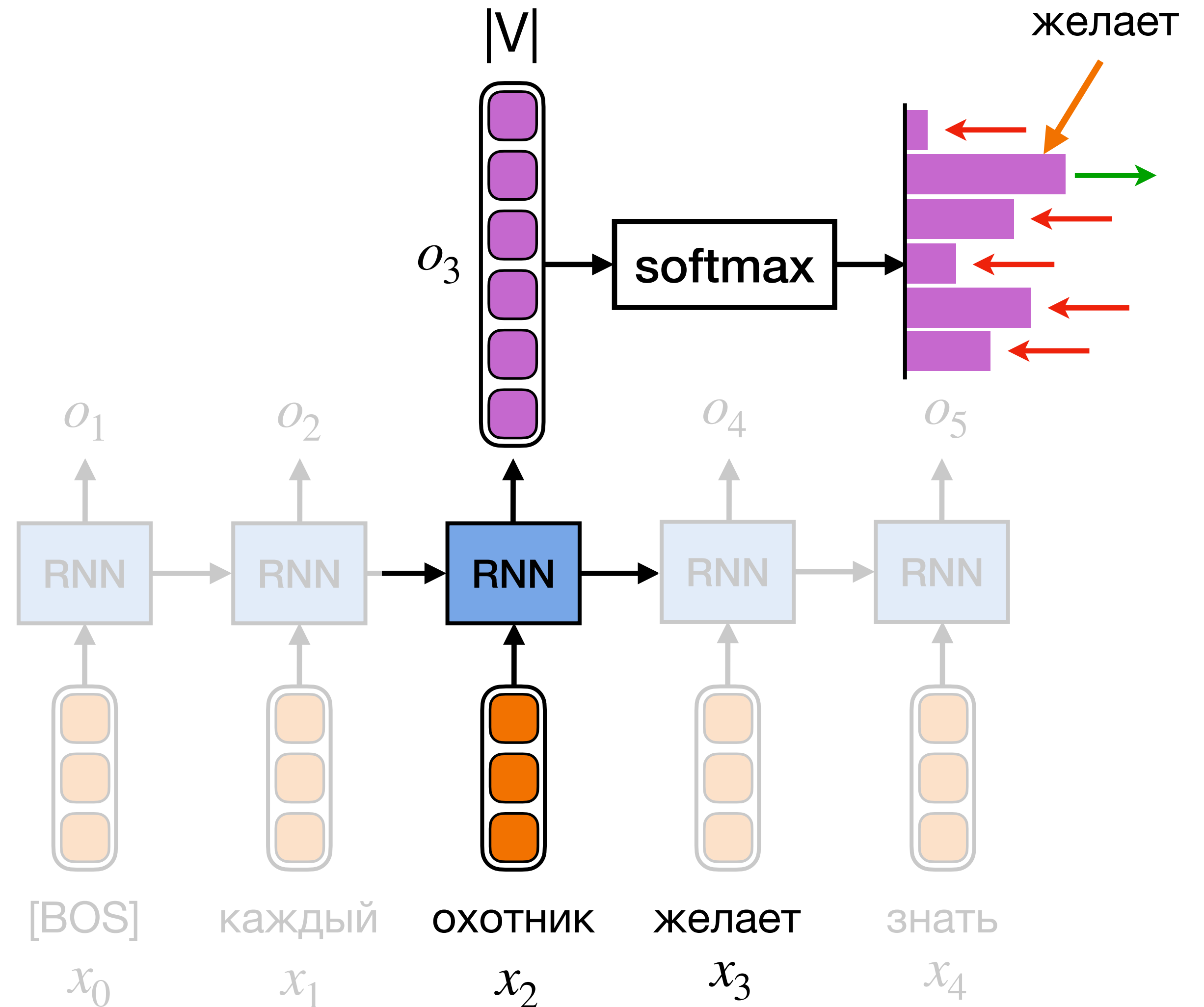
# RNN: Обучение

$$p(x_1, \dots, x_m) = \prod_{t=1}^m p(x_t | x_{<t}) \rightarrow \max$$

$$p(\cdot | x_{<t}) = \text{softmax}(o_t)$$

Накладывая логарифм и отрицание, получаем кросс-энтропию.

$$L(x) = - \sum_{t=1}^m \log p(x_t | x_{<t}) \rightarrow \min$$



# RNN: Обучение

$$p(x_1, \dots, x_m) = \prod_{t=1}^m p(x_t | x_{<t}) \rightarrow \max$$

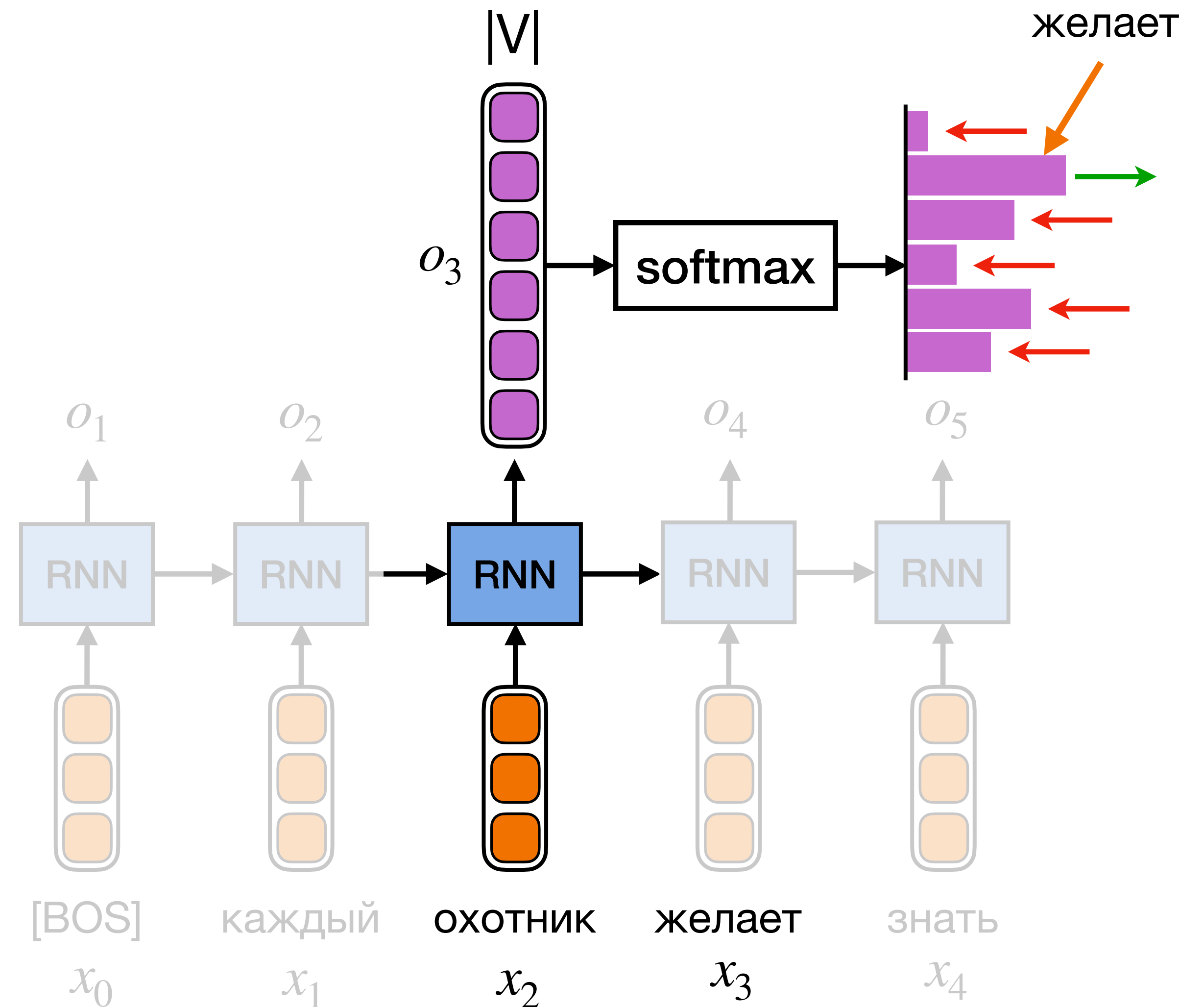
$$p(\cdot | x_{<t}) = \text{softmax}(o_t)$$

Накладывая логарифм и отрицание, получаем кросс-энтропию.

$$L(x) = - \sum_{t=1}^m \log p(x_t | x_{<t}) \rightarrow \min$$

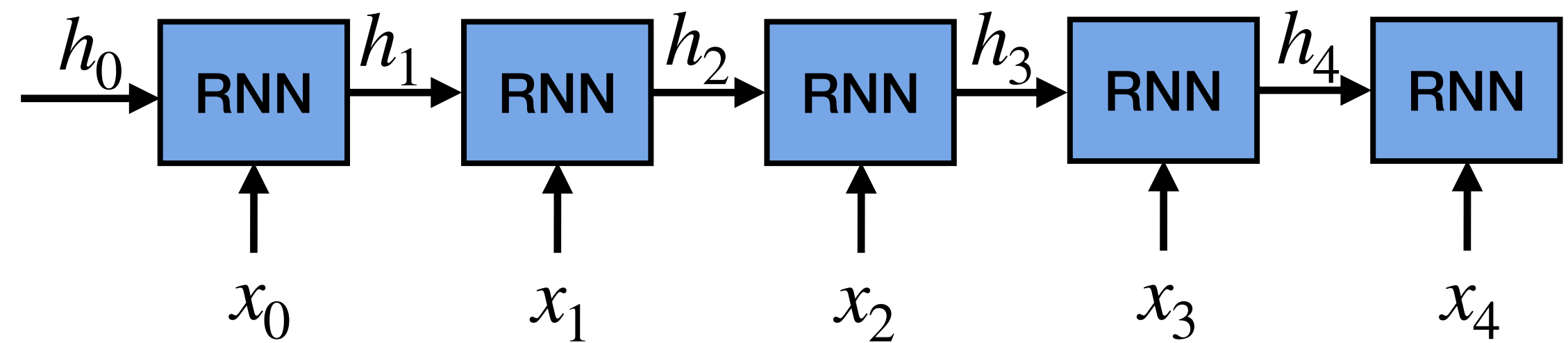
Ошибка для всего корпуса.

$$L(X) = - \frac{1}{|X|} \sum_{x \in X} \sum_{t=1}^m \log p(x_t | x_{<t}) \rightarrow \min$$



# Градиенты RNN

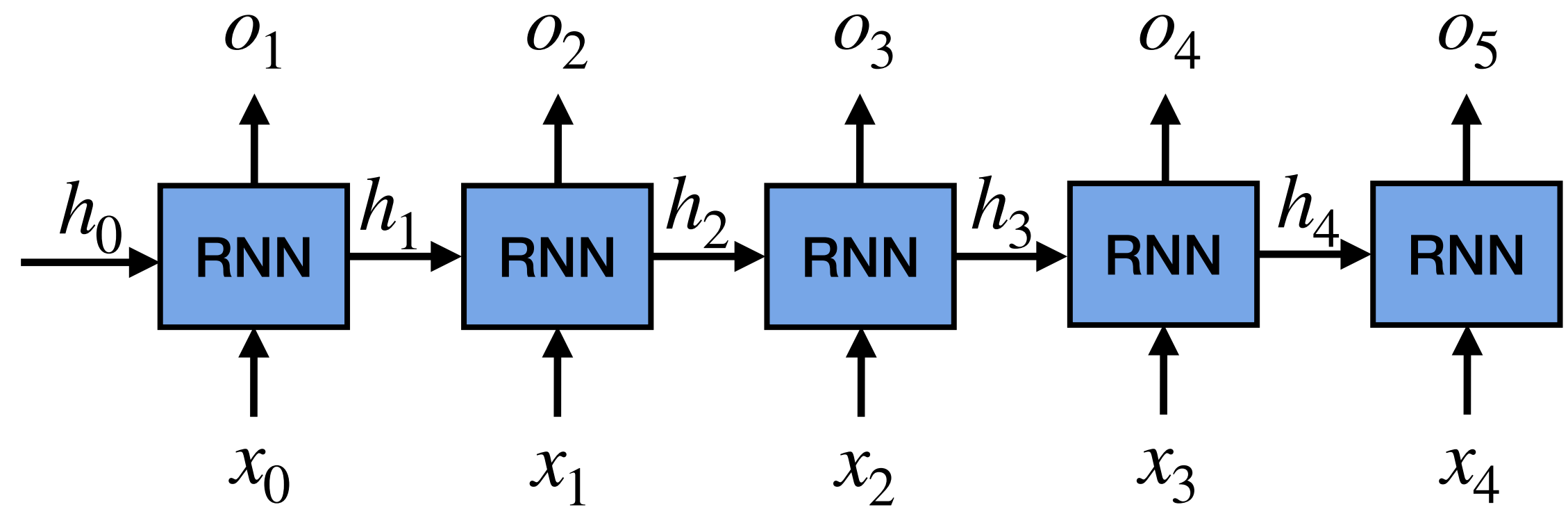
$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$



# Градиенты RNN

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

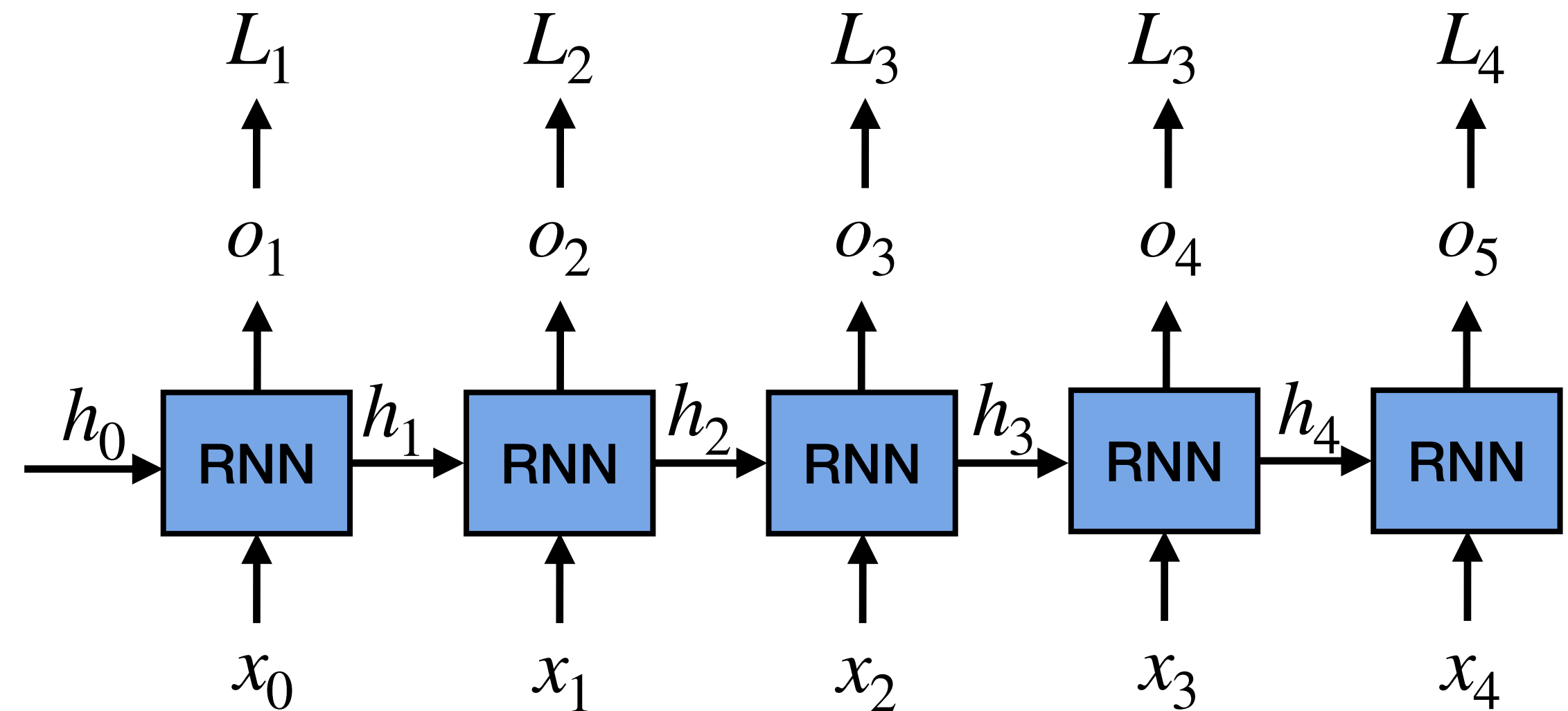


# Градиенты RNN

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t | x_{<t}) = -\log \text{softmax}(o_t)_{x_t}$$



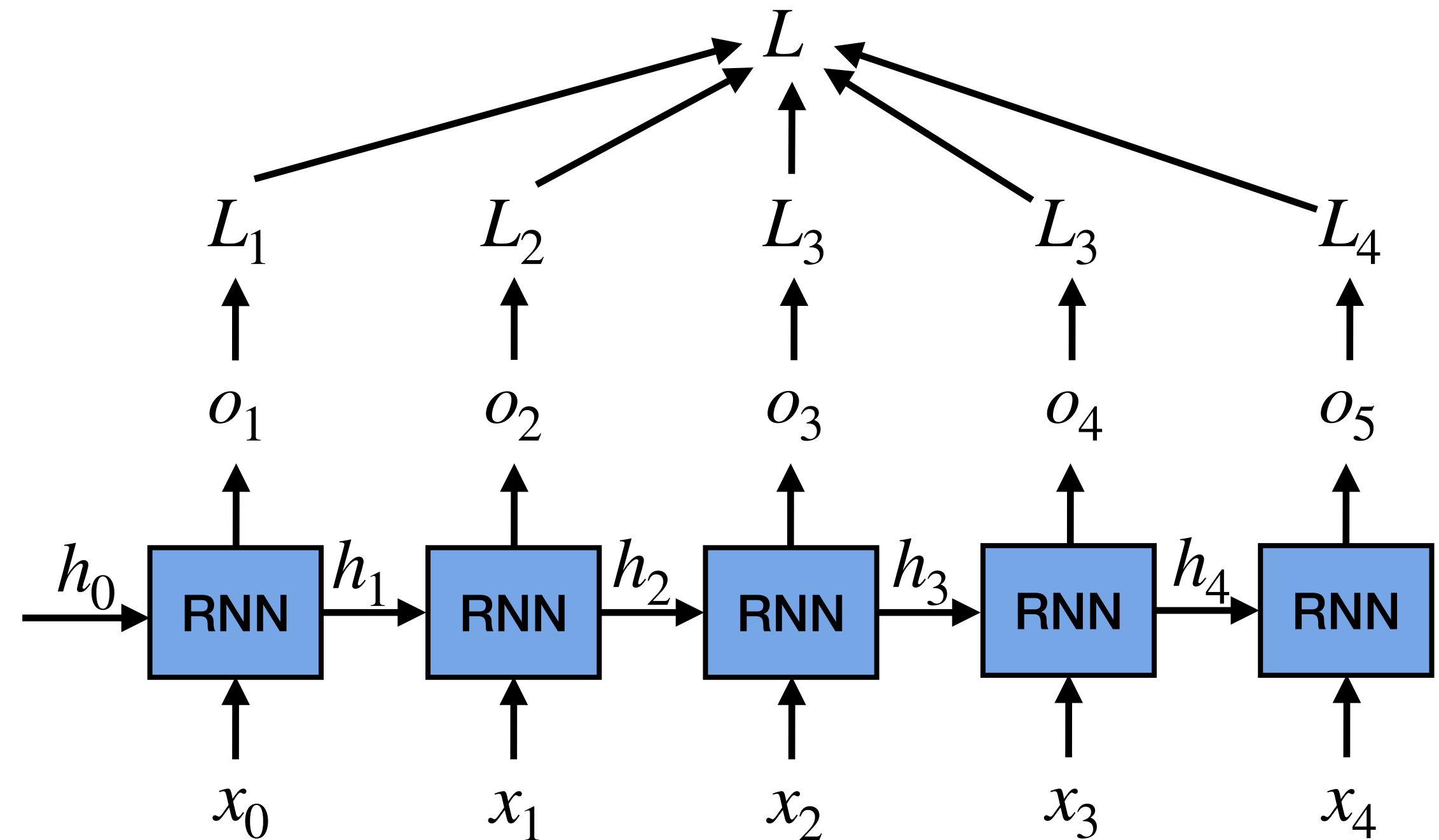
# Градиенты RNN

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t | x_{<t}) = -\log \text{softmax}(o_t)_{x_t}$$

$$L = \sum_{t=1}^m L_t$$



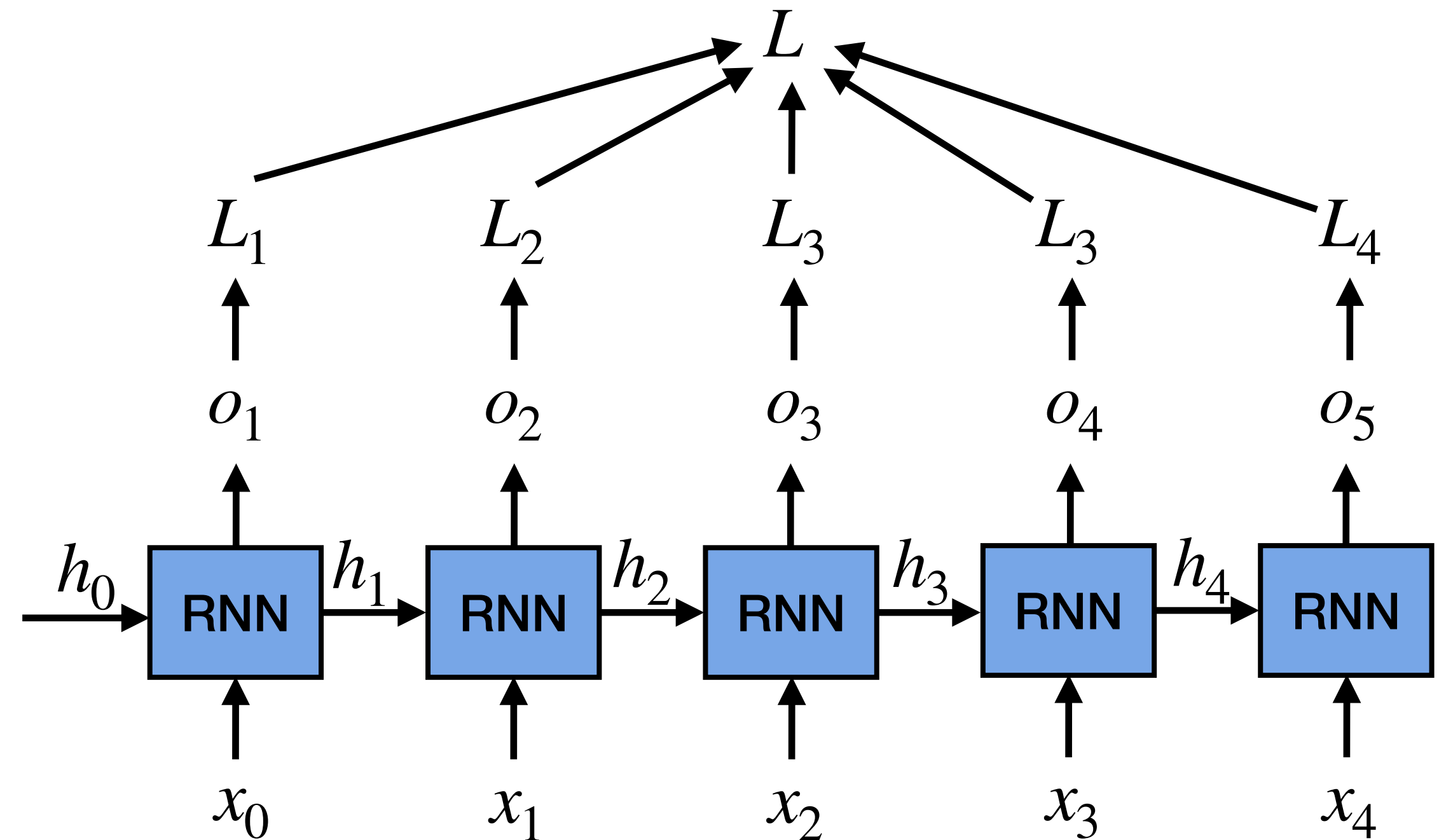
# Градиенты RNN

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t | x_{<t}) = -\log \text{softmax}(o_t)_{x_t}$$

$$L = \sum_{t=1}^m L_t$$



Посмотрим на то, как ведут себя градиенты.  
Будет немного больно.

# Градиенты RNN

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t | x_{<t}) = -\log \text{softmax}(o_t)_{x_t}$$

$$L = \sum_{t=1}^m L_t$$

chain rule

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \frac{dh_t}{dU}$$



# Градиенты RNN

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t | x_{<t}) = -\log \text{softmax}(o_t)_{x_t}$$

$$L = \sum_{t=1}^m L_t$$

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \frac{dh_t}{dU}$$

$$\frac{dh_t}{dU} = \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \frac{dh_{t-1}}{dU}$$

Переходим от производных к частным производным

# Градиенты RNN

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t | x_{<t}) = -\log \text{softmax}(o_t)_{x_t}$$

$$L = \sum_{t=1}^m L_t$$

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \frac{dh_t}{dU}$$

$$\begin{aligned} \frac{dh_t}{dU} &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \boxed{\frac{dh_{t-1}}{dU}} = \\ &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \boxed{\left( \frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{dh_{t-2}}{dU} \right)} \end{aligned}$$

# Градиенты RNN

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t | x_{<t}) = -\log \text{softmax}(o_t)_{x_t}$$

$$L = \sum_{t=1}^m L_t$$

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \frac{dh_t}{dU}$$

$$\begin{aligned} \frac{dh_t}{dU} &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \frac{dh_{t-1}}{dU} = \\ &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \left( \frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{dh_{t-2}}{dU} \right) = \\ &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{dh_{t-2}}{dU} \end{aligned}$$

Раскрываем скобки

# Градиенты RNN

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t | x_{<t}) = -\log \text{softmax}(o_t)_{x_t}$$

$$L = \sum_{t=1}^m L_t$$

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \frac{dh_t}{dU}$$

$$\begin{aligned} \frac{dh_t}{dU} &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \frac{dh_{t-1}}{dU} = \\ &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \left( \frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{dh_{t-2}}{dU} \right) = \\ &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{dh_{t-2}}{dU} = \\ &= \sum_{k=1}^t \left( \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial U} \end{aligned}$$

Собираем все в одну сумму

# Градиенты RNN

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t | x_{<t}) = -\log \text{softmax}(o_t)_{x_t}$$

$$L = \sum_{t=1}^m L_t$$

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \frac{dh_t}{dU}$$

$$\begin{aligned} \frac{dh_t}{dU} &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \frac{dh_{t-1}}{dU} = \\ &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \left( \frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{dh_{t-2}}{dU} \right) = \\ &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{dh_{t-2}}{dU} = \\ &= \sum_{k=1}^t \left( \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial U} \end{aligned}$$

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \left[ \sum_{k=1}^t \left( \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial U} \right]$$

# Взрыв градиентов

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \left[ \sum_{k=1}^t \left( \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial U} \right]$$


Серия умножений производных

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| > 1$$

- Происходит **взрыв градиента**  $\frac{dL}{dU}$
- Модель расходится, NaN в весах

# Взрыв градиентов

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \left[ \sum_{k=1}^t \left( \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial U} \right]$$


Серия умножений производных

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| > 1$$

- Происходит **взрыв градиента**  $\frac{dL}{dU}$
- Модель расходится, NaN в весах

## Решения:

- Регуляризация
- Уменьшение learning rate
- Gradient clipping

# Взрыв градиентов

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \left[ \sum_{k=1}^t \left( \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial U} \right]$$

Серия умножений производных

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| > 1$$

- Происходит **взрыв градиента**  $\frac{dL}{dU}$
- Модель расходится, NaN в весах

## Решения:

- Регуляризация
- Уменьшение learning rate
- Gradient clipping

$$1. g \leftarrow \min \left( 1, \frac{\text{max norm}}{\|g\|} \right) \cdot g$$

Правильный способ

$$2. g \leftarrow \text{clip}(g, -C, C)$$

Ленивый способ (меняет направление градиента)



# Затухание градиентов

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \left[ \sum_{k=1}^t \left( \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial U} \right]$$
$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| < 1$$

- Происходит **затухание градиента**
- Модель перестает учиться
- Модель не улавливает далекие зависимости!

# Затухание градиентов

$$\frac{dL}{dU} = \sum_{t=1}^m \frac{dL_t}{do_t} \frac{do_t}{dh_t} \left[ \sum_{k=1}^t \left( \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial U} \right] \quad \left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| < 1$$

- Происходит **затухание градиента**
- Модель перестает учиться
- Модель не улавливает далекие зависимости!

Затухание градиентов – частая проблема RNN.

Ее нельзя починить трюками.

# Затухание градиентов: почему возникает?

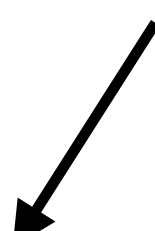
$$h_j = \sigma(\underbrace{Wx_j + Uh_{j-1} + b_h}_{z_j})$$

$$\frac{\partial h_j}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial z_j} \frac{\partial z_j}{\partial h_{j-1}}$$

# Затухание градиентов: почему возникает?

$$h_j = \sigma(\underbrace{Wx_j + Uh_{j-1} + b_h}_{z_j})$$

Поэлементное умножение

$$\frac{\partial h_j}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial z_j} \frac{\partial z_j}{\partial h_{j-1}} = \left( \sigma(z_j) \odot (1 - \sigma(z_j)) \right) U$$


# Затухание градиентов: почему возникает?

$$h_j = \sigma(\underbrace{Wx_j + Uh_{j-1} + b_h}_{z_j})$$

Поэлементное умножение

$$\frac{\partial h_j}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial z_j} \frac{\partial z_j}{\partial h_{j-1}} = \left( \sigma(z_j) \odot (1 - \sigma(z_j)) \right) U$$

Посмотрим на спектральную норму

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \underbrace{\| \sigma(z_j) \odot (1 - \sigma(z_j)) \|}_{< 1} \cdot \|U\|$$

т. к.  $\sigma(z) \in [0, 1]$

# Затухание градиентов: почему возникает?

$$h_j = \sigma(\underbrace{Wx_j + Uh_{j-1} + b_h}_{z_j})$$

Поэлементное умножение

$$\frac{\partial h_j}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial z_j} \frac{\partial z_j}{\partial h_{j-1}} = \left( \sigma(z_j) \odot (1 - \sigma(z_j)) \right) U$$

Посмотрим на спектральную норму

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \underbrace{\| \sigma(z_j) \odot (1 - \sigma(z_j)) \|}_{< 1} \cdot \|U\|$$

т. к.  $\sigma(z) \in [0, 1]$

Если  $U$  – ортогональная, то

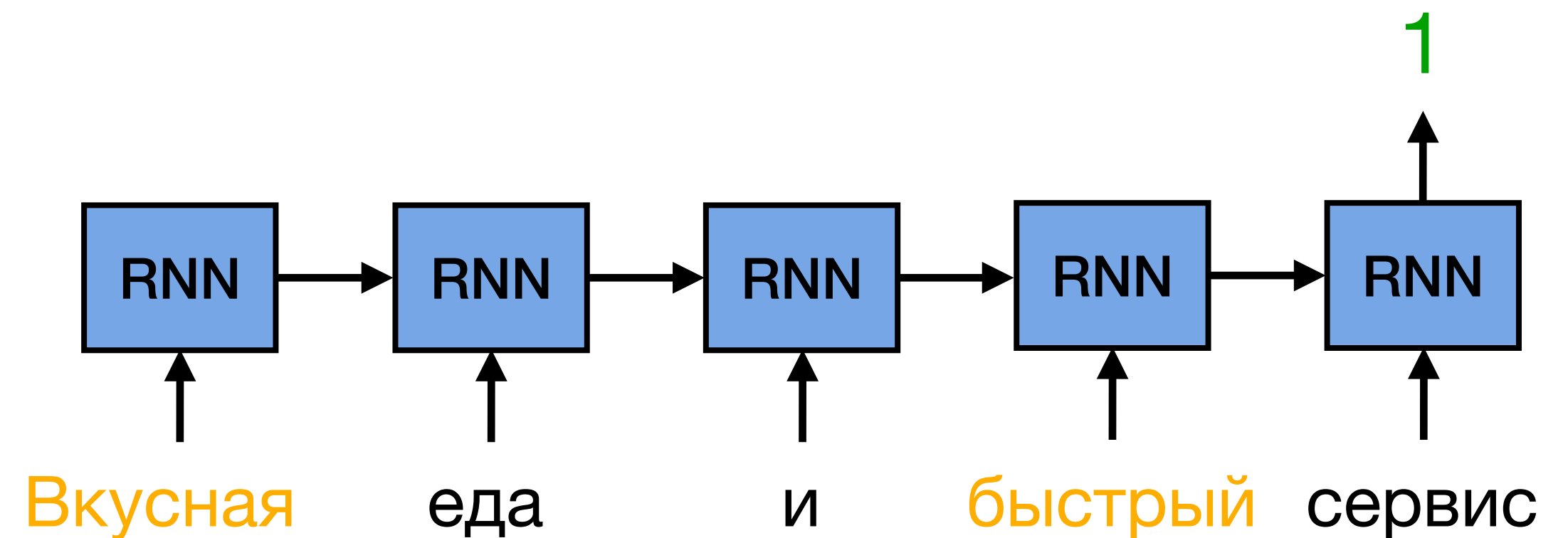
$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \| \sigma(z_j) \odot (1 - \sigma(z_j)) \| < 1$$

# RNN для классификации

- В качестве предсказания берем выход **последнего** блока.

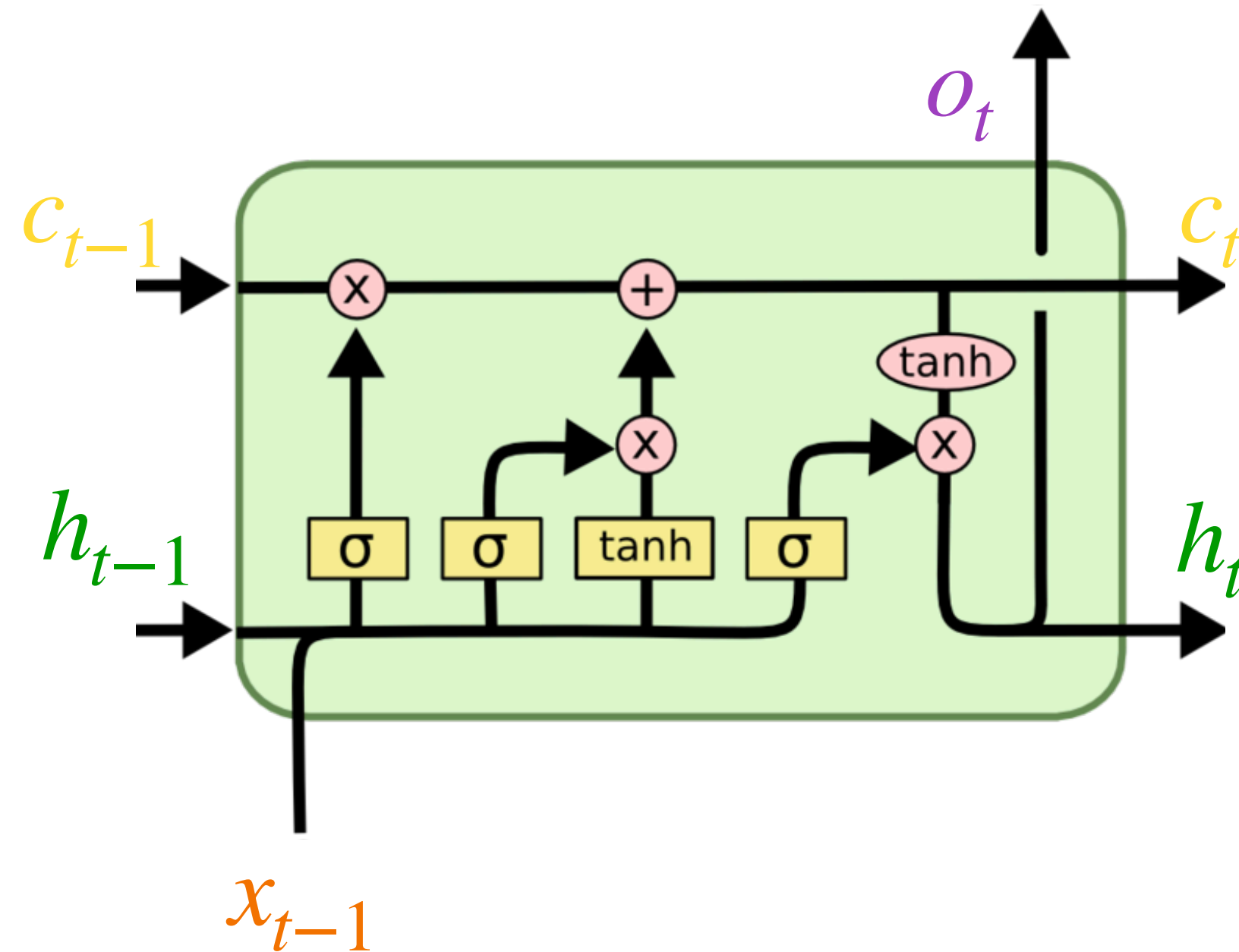
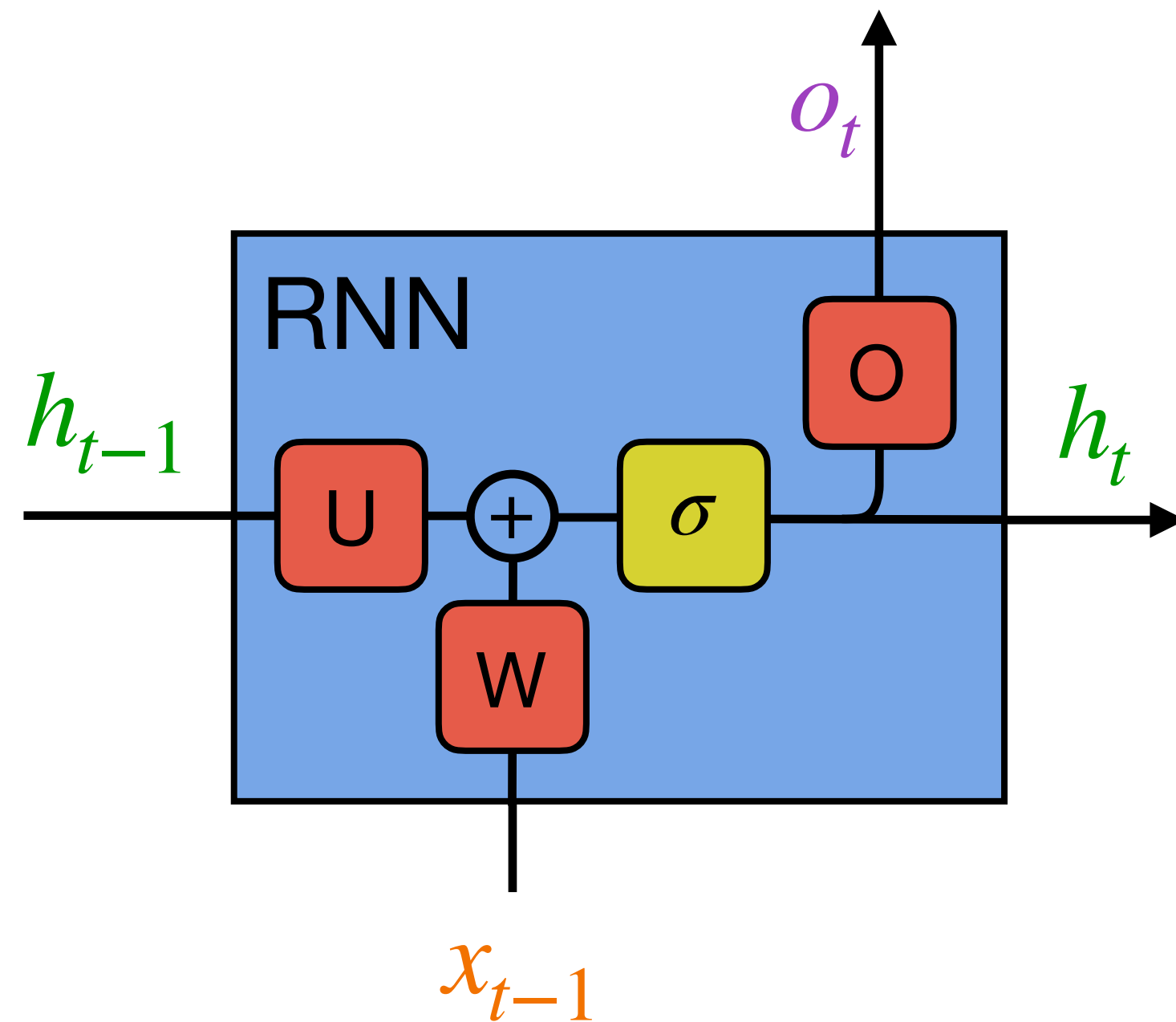
Классификация тональности

- 1** – ПОЗИТИВНЫЙ
- 0** – НЕГАТИВНЫЙ



# Long shot-term memory (LSTM)

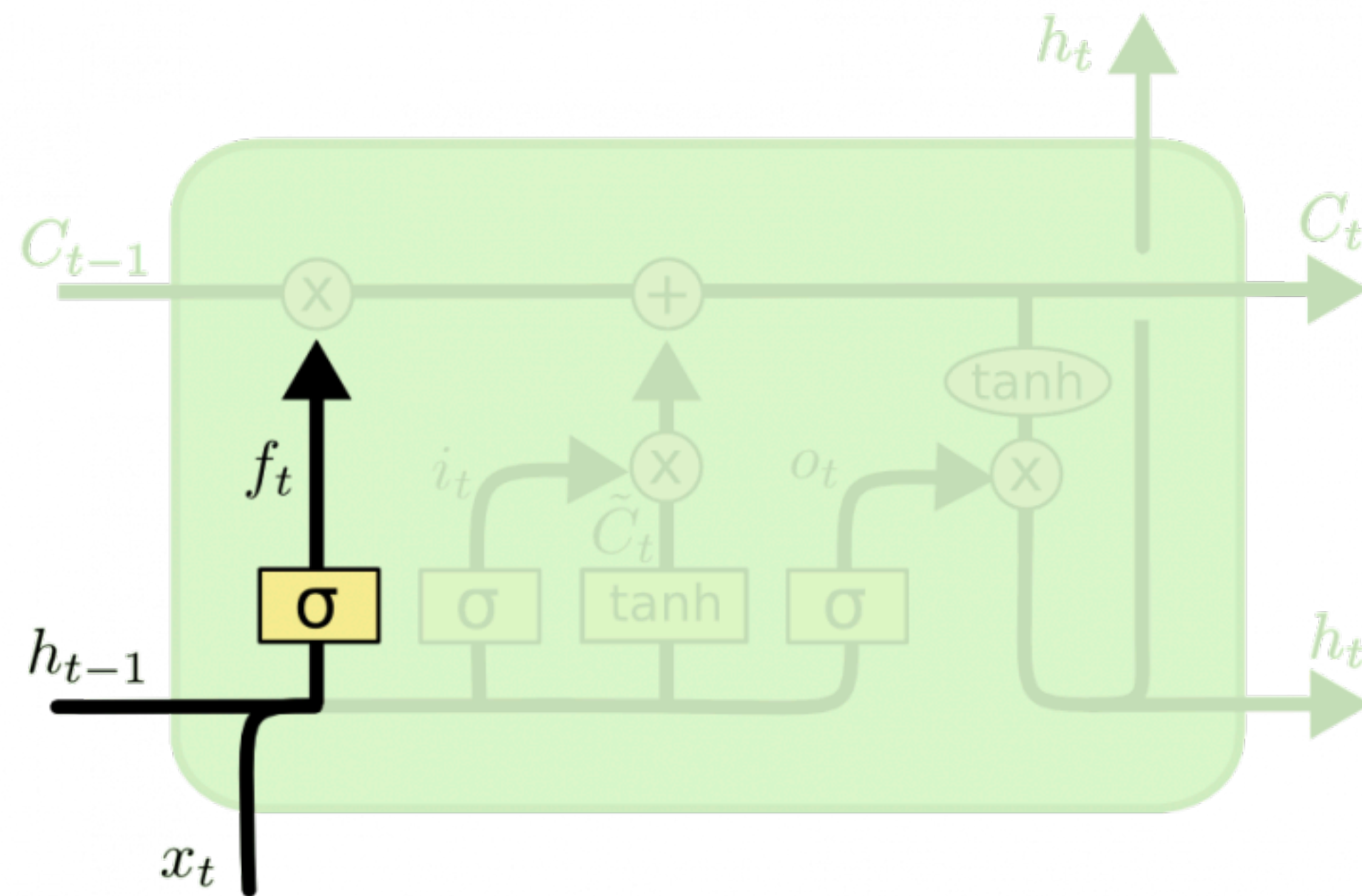
В LSTM добавляется вектор памяти  $c_t$   
Благодаря ему модель не забывает старую информацию





# LSTM: фильтр забывания

Контролирует, какую информацию надо забыть, а какую оставить.

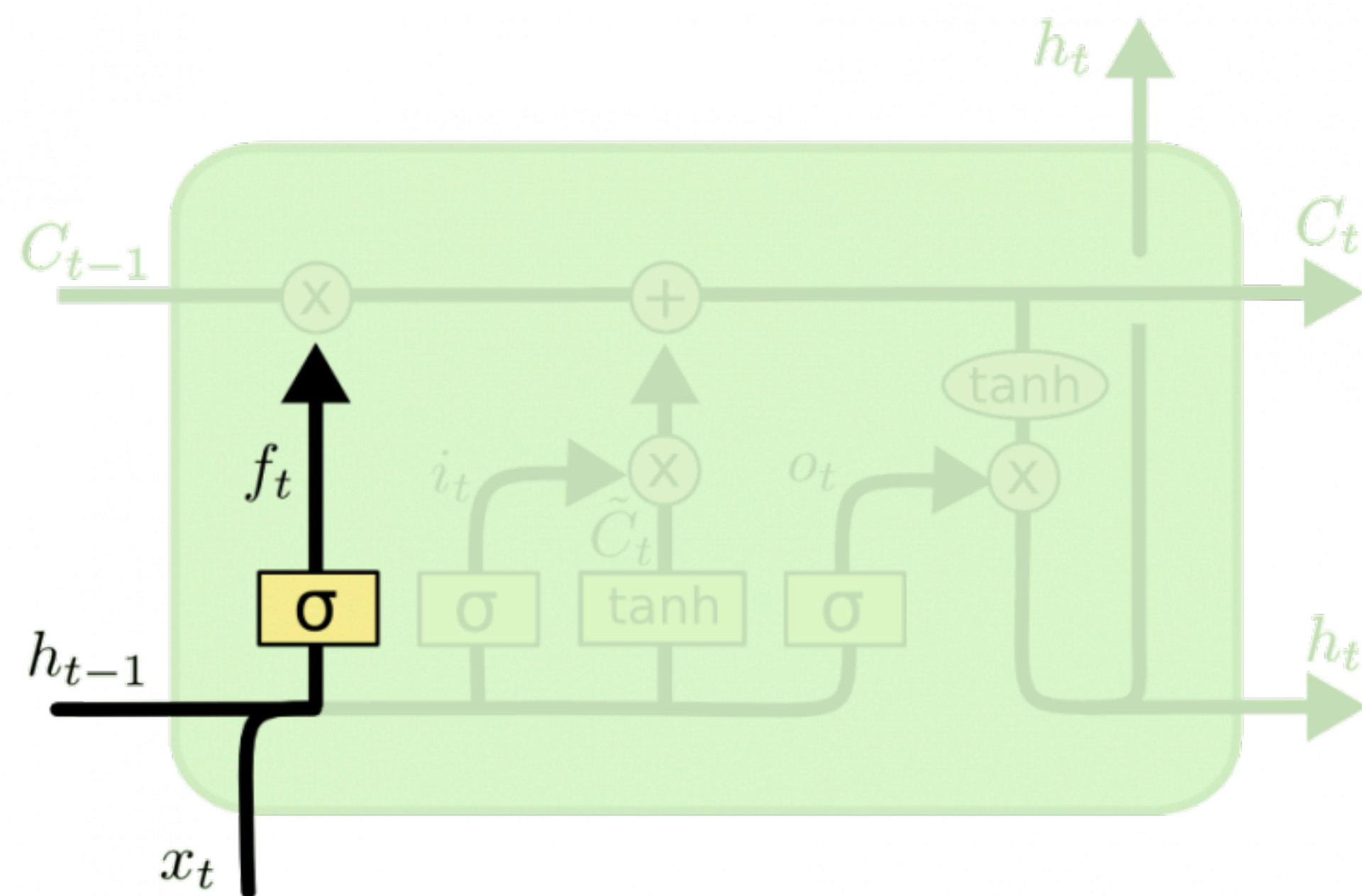


$$f_t = \sigma(W_f x_{t-1} + U_f h_{t-1} + b_f)$$

$$f_t \in [0,1]$$

# LSTM: фильтр забывания

Контролирует, какую информацию надо забыть, а какую оставить.



$$f_t = \sigma(W_f x_{t-1} + U_f h_{t-1} + b_f)$$

$$f_t \in [0, 1]$$

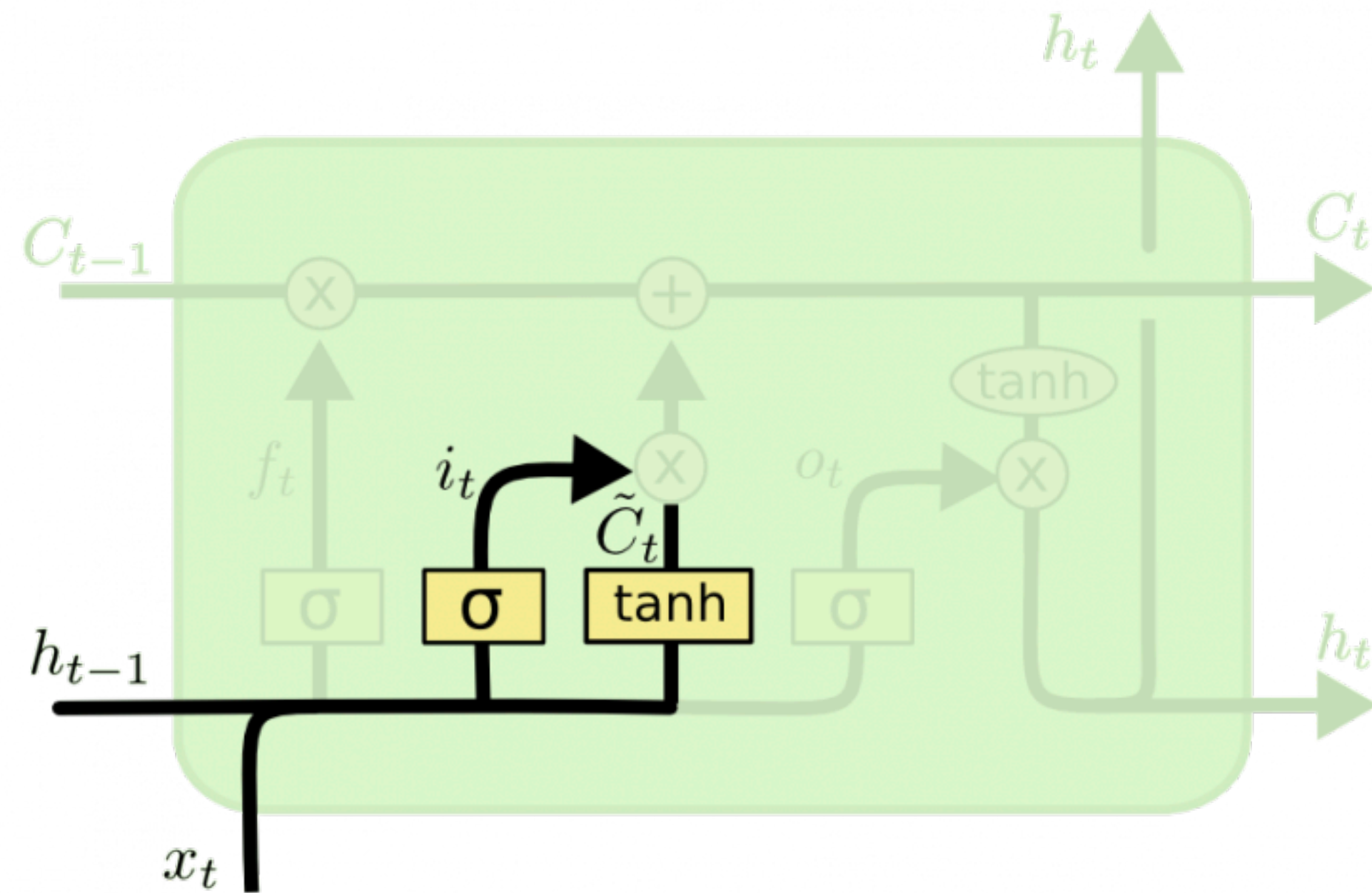
Еда была **вкусная** и мы **не разочаровались**.

$x_3$  – слово-маркер, можно забыть все до него.

$x_7$  – негативный маркер, но до него идет "**не**".  
Знаем об этом из  $h_7$ .

# LSTM: фильтр входа

Контролирует, какую информацию надо добавить в вектор памяти  $C_t$



Из-за  $i_t$  можем не добавлять ничего.

$$i_t = \sigma(W_i x_{t-1} + U_i h_{t-1} + b_i)$$

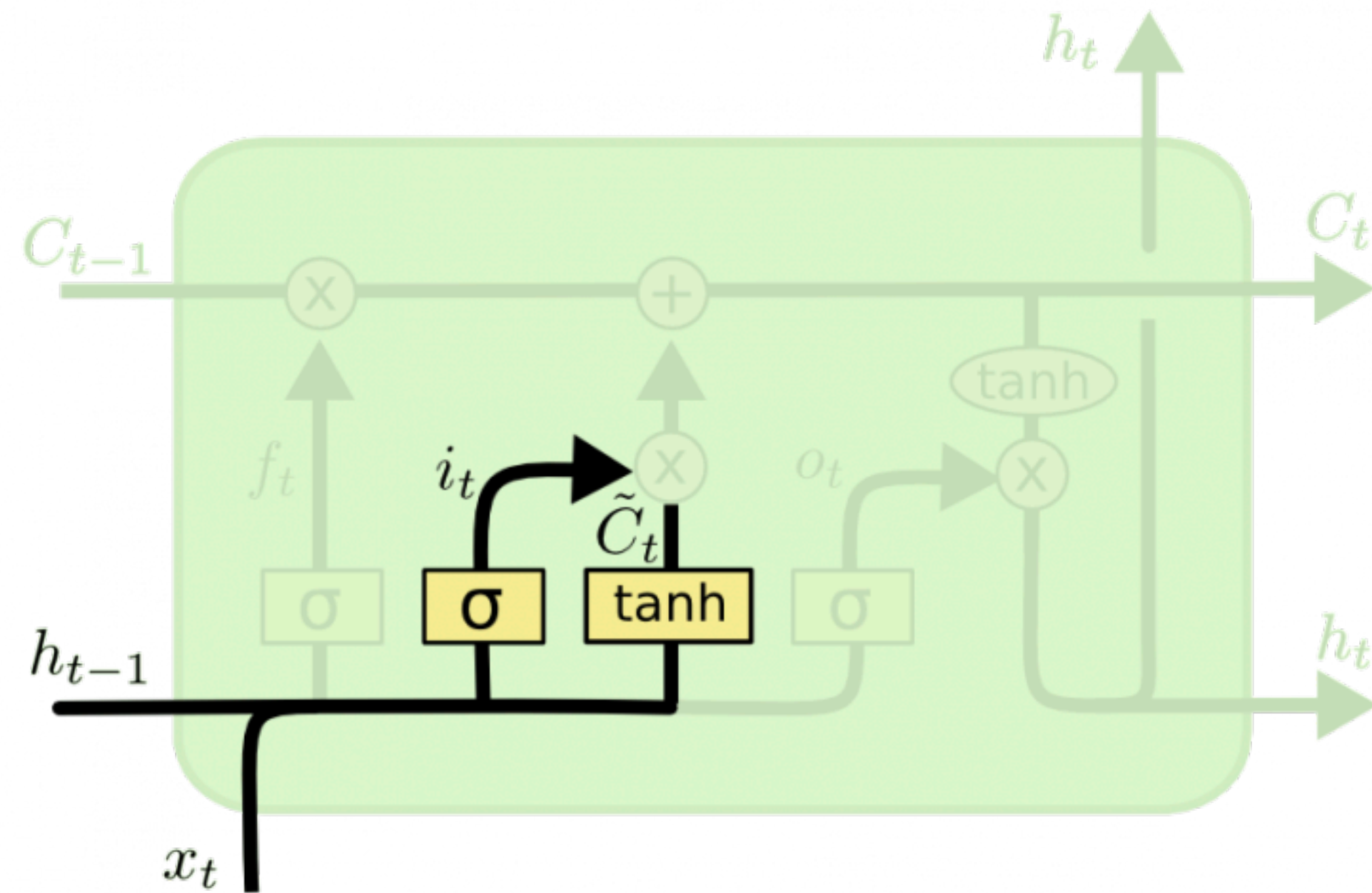
$$i_t \in [0, 1]$$

$$\tilde{C}_t = \tanh(W_i x_{t-1} + U_i h_{t-1} + b_i)$$



# LSTM: фильтр входа

Контролирует, какую информацию надо добавить в вектор памяти  $C_t$



Из-за  $i_t$  можем не добавлять ничего.

$$i_t = \sigma(W_i x_{t-1} + U_i h_{t-1} + b_i)$$

$$i_t \in [0, 1]$$

$$\tilde{C}_t = \tanh(W_i x_{t-1} + U_i h_{t-1} + b_i)$$

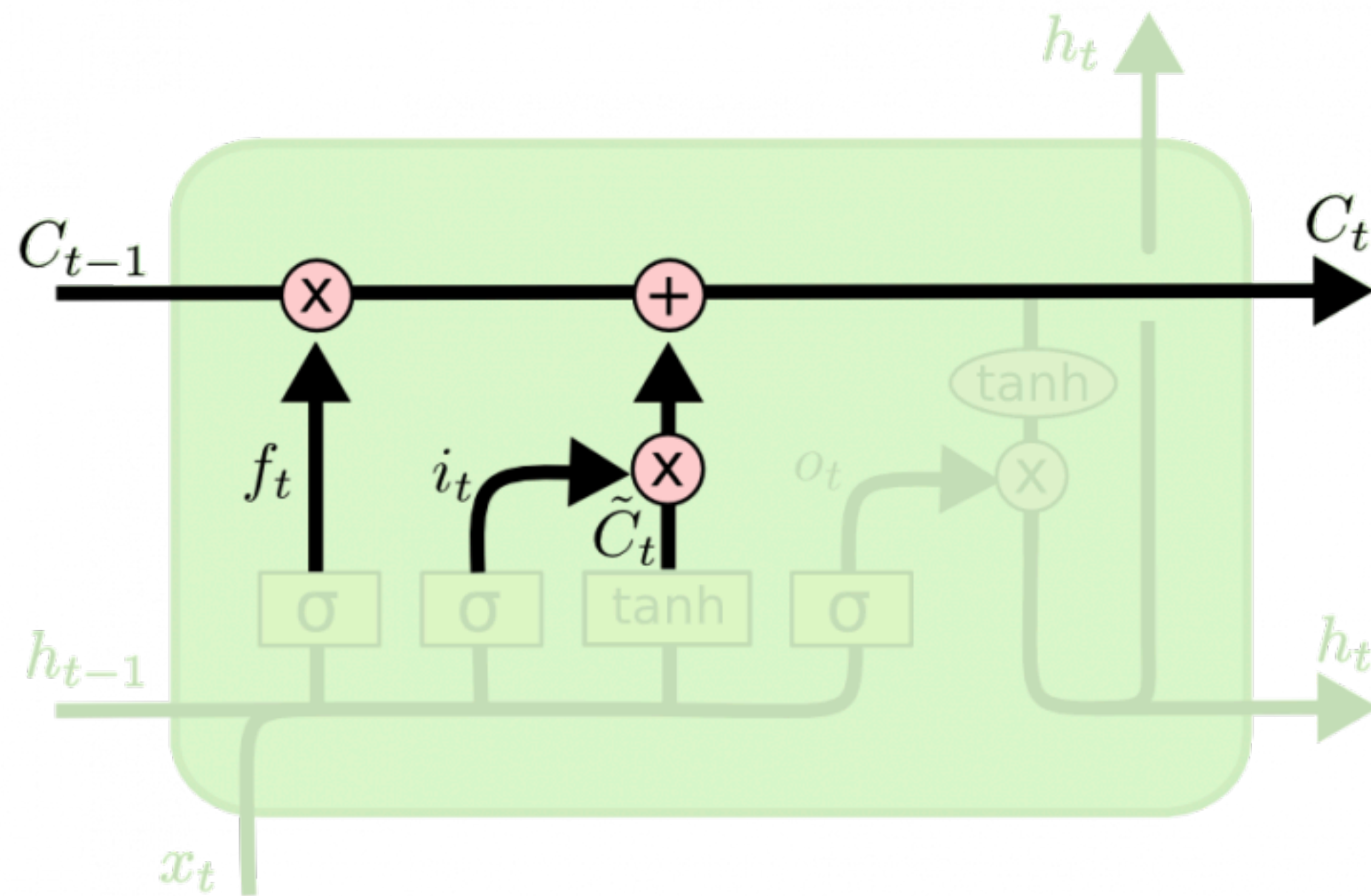
Еда была **вкусная** и **мы** не разочаровались.

$x_3$  – слово-маркер.  
Запоминаем, что класс  
положительный.

$x_5$  – не влияет на класс.  
Ничего не добавляем.

# LSTM: обновление памяти

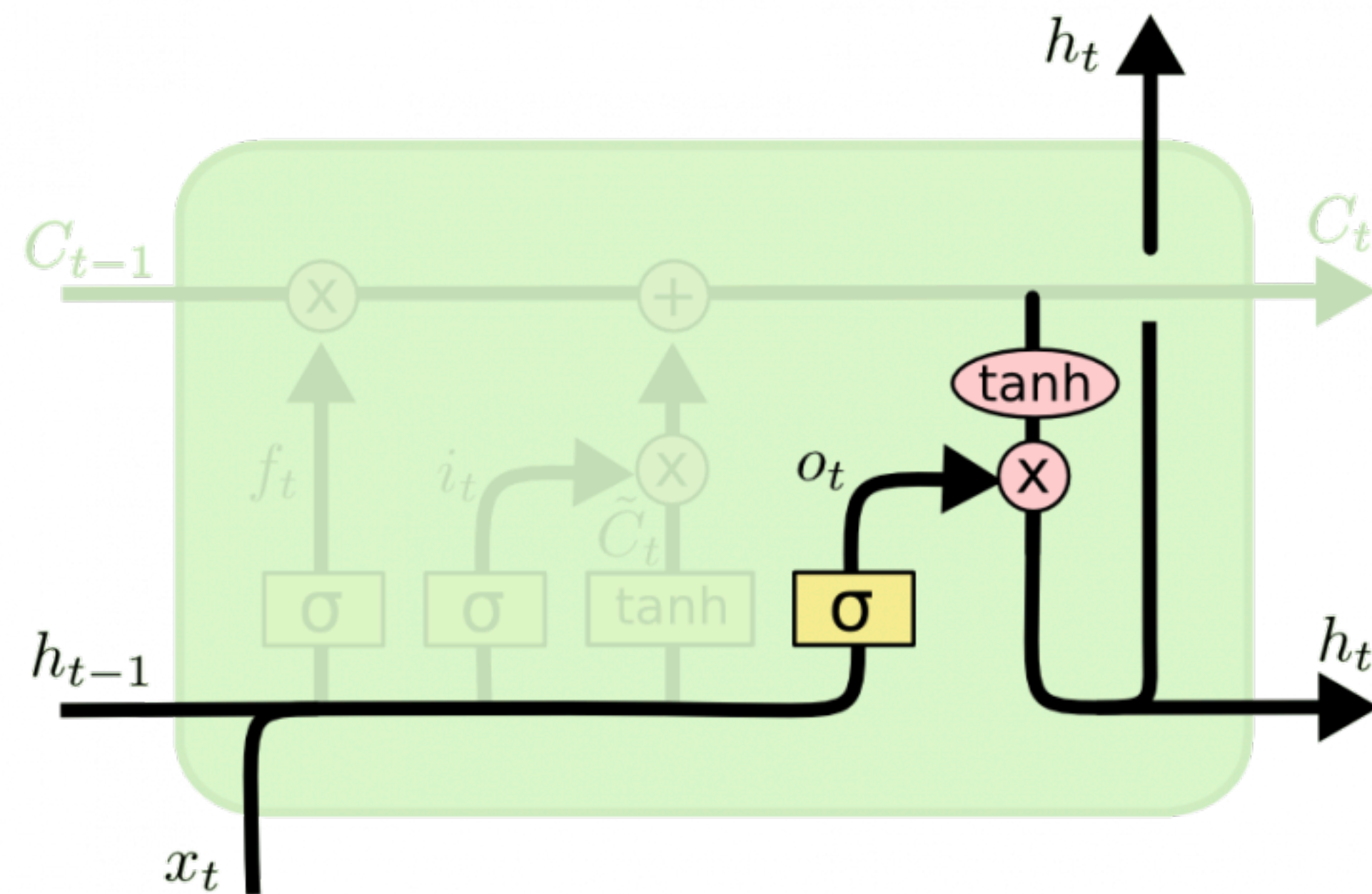
Удаляем ненужную информацию и добавляем новую.



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

# LSTM: фильтр выхода

Контролирует выход текущего шага.

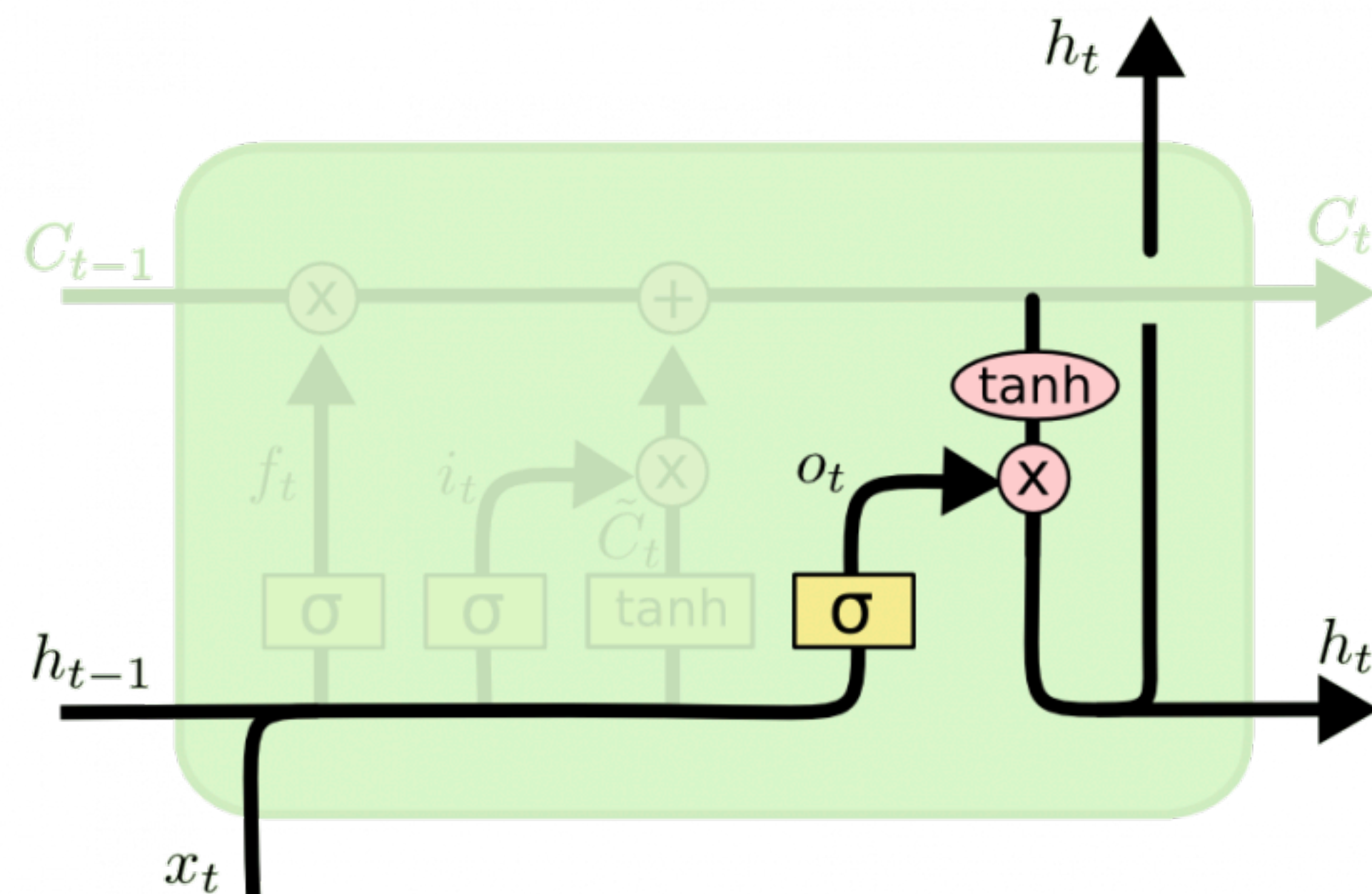


$$o_t = \sigma(W_t x_{t-1} + U_t h_{t-1} + b_t)$$

$$h_t = o_t \odot \tanh(c_t)$$

# LSTM: фильтр выхода

Контролирует выход текущего шага.



$$o_t = \sigma(W_t x_{t-1} + U_t h_{t-1} + b_t)$$

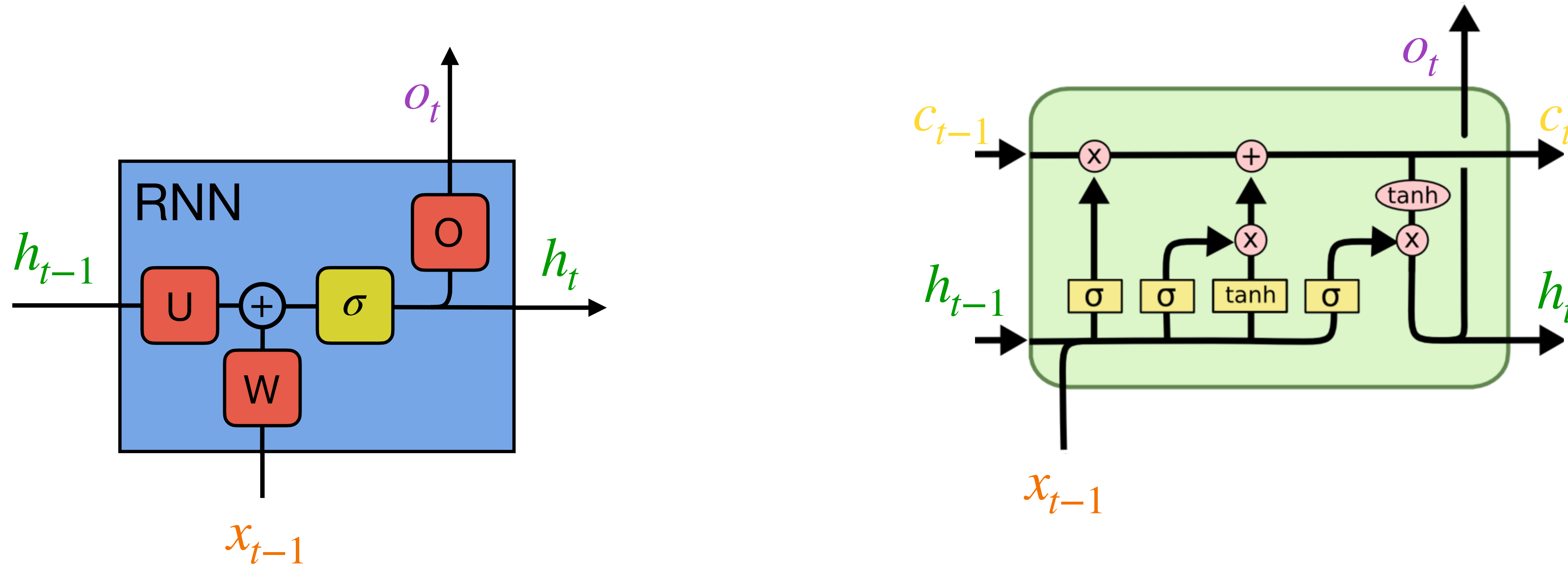
$$h_t = o_t \odot \tanh(c_t)$$

Учитель ведет урок, посвященный рекуррентным моделям. \_

Начало нового предложения.  
Надо вспомнить, что речь идет об учителе и уроке.



# Сравнение RNN и LSTM

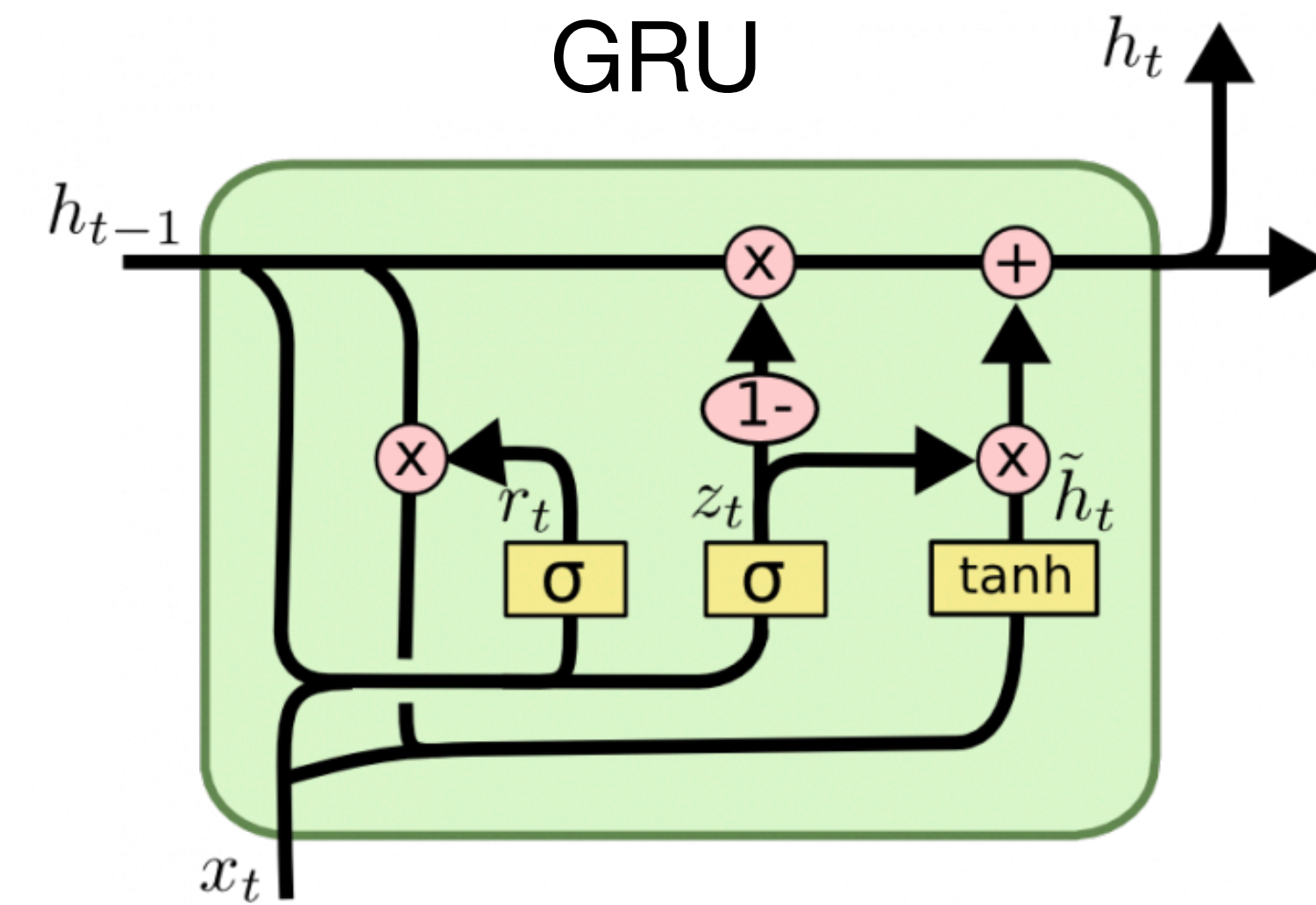
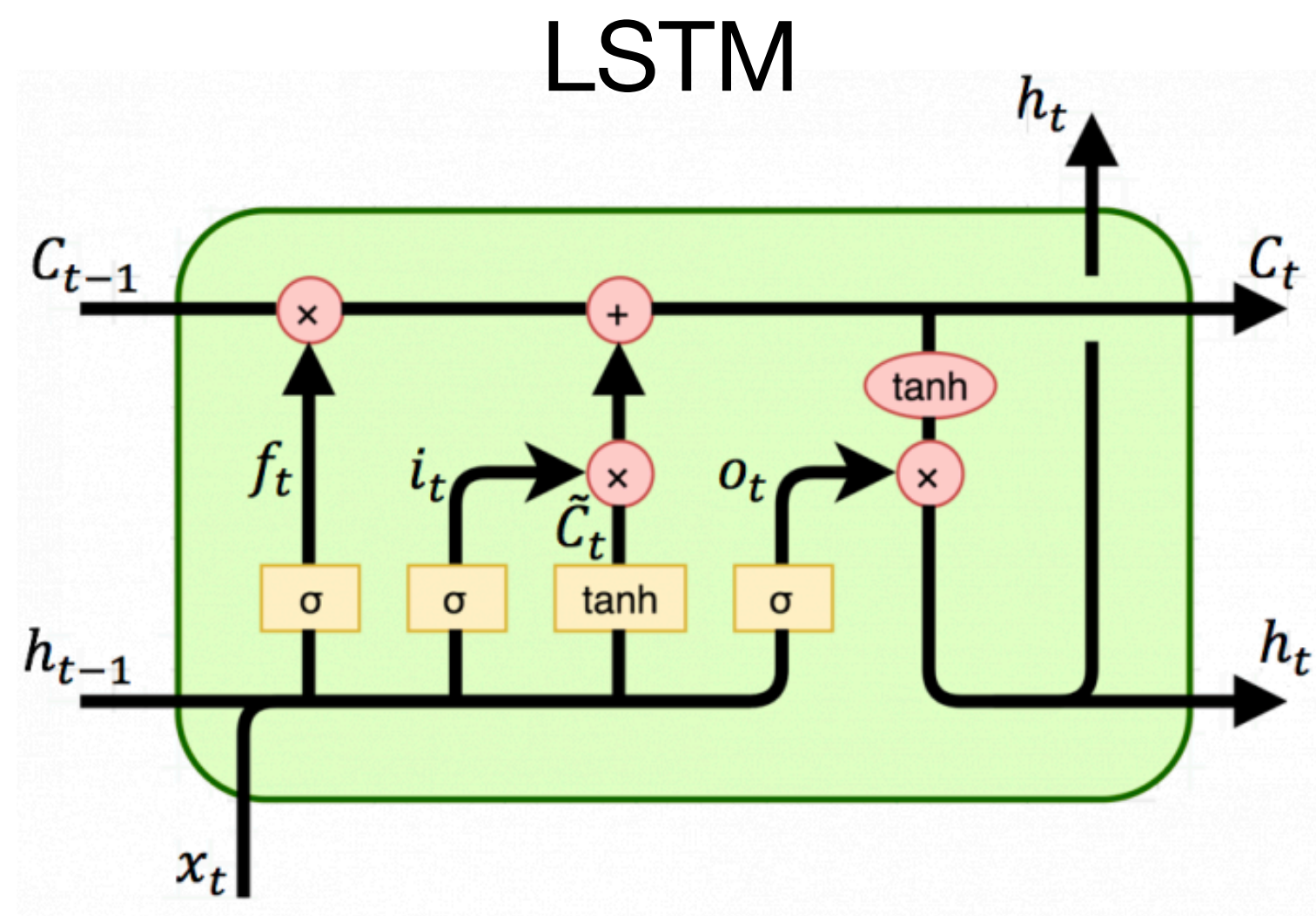


- LSTM намного лучше улавливает удаленные зависимости.
- В LSTM 4 слоя вместо 1 у RNN.  
=> В 4 раза больше параметров!

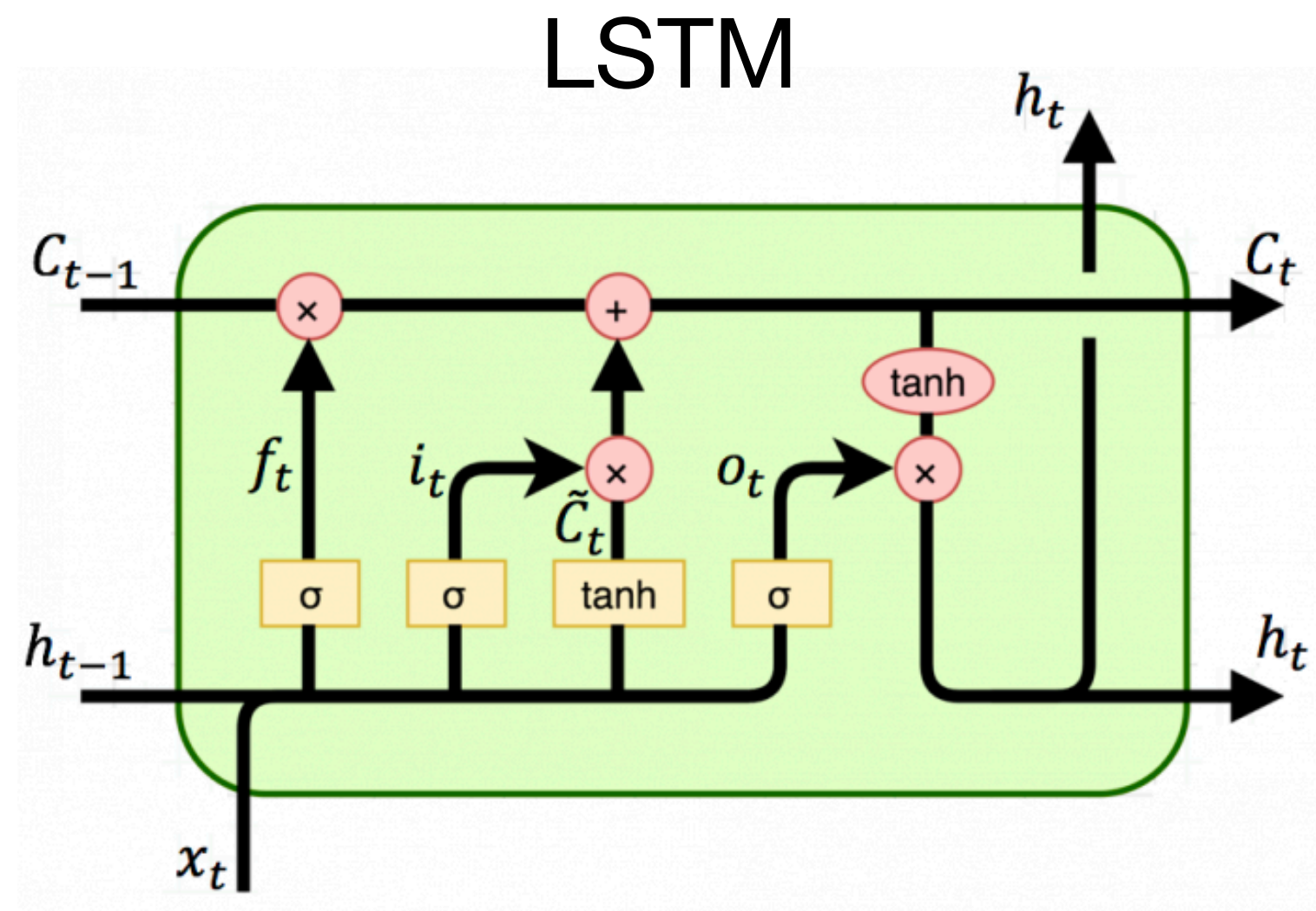


# Gated recurrent units (GRU)

Наиболее успешная вариация LSTM для уменьшения числа параметров.

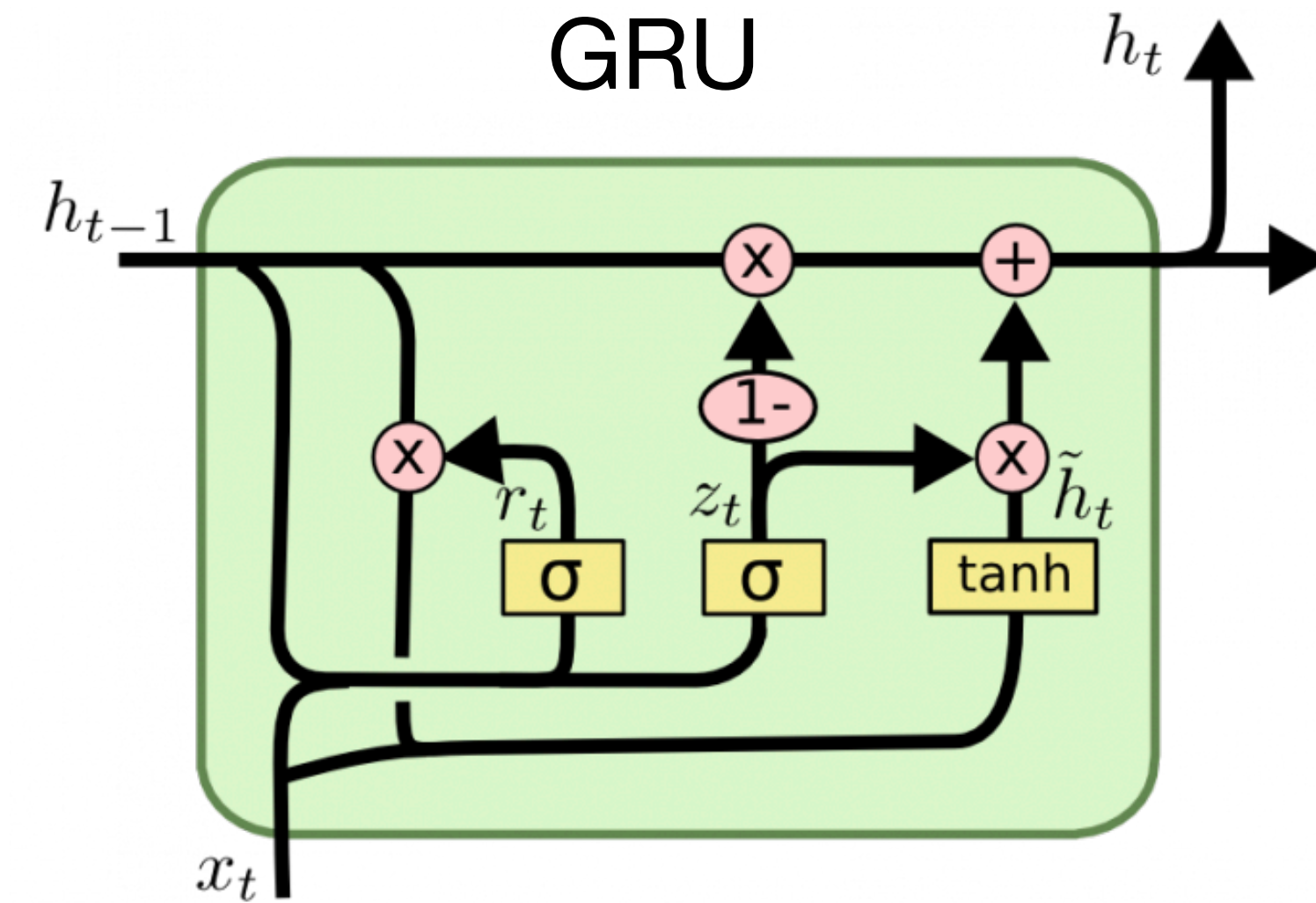


# Gated recurrent units (GRU)



$$f_t = \sigma(W_f x_{t-1} + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_{t-1} + U_i h_{t-1} + b_i)$$

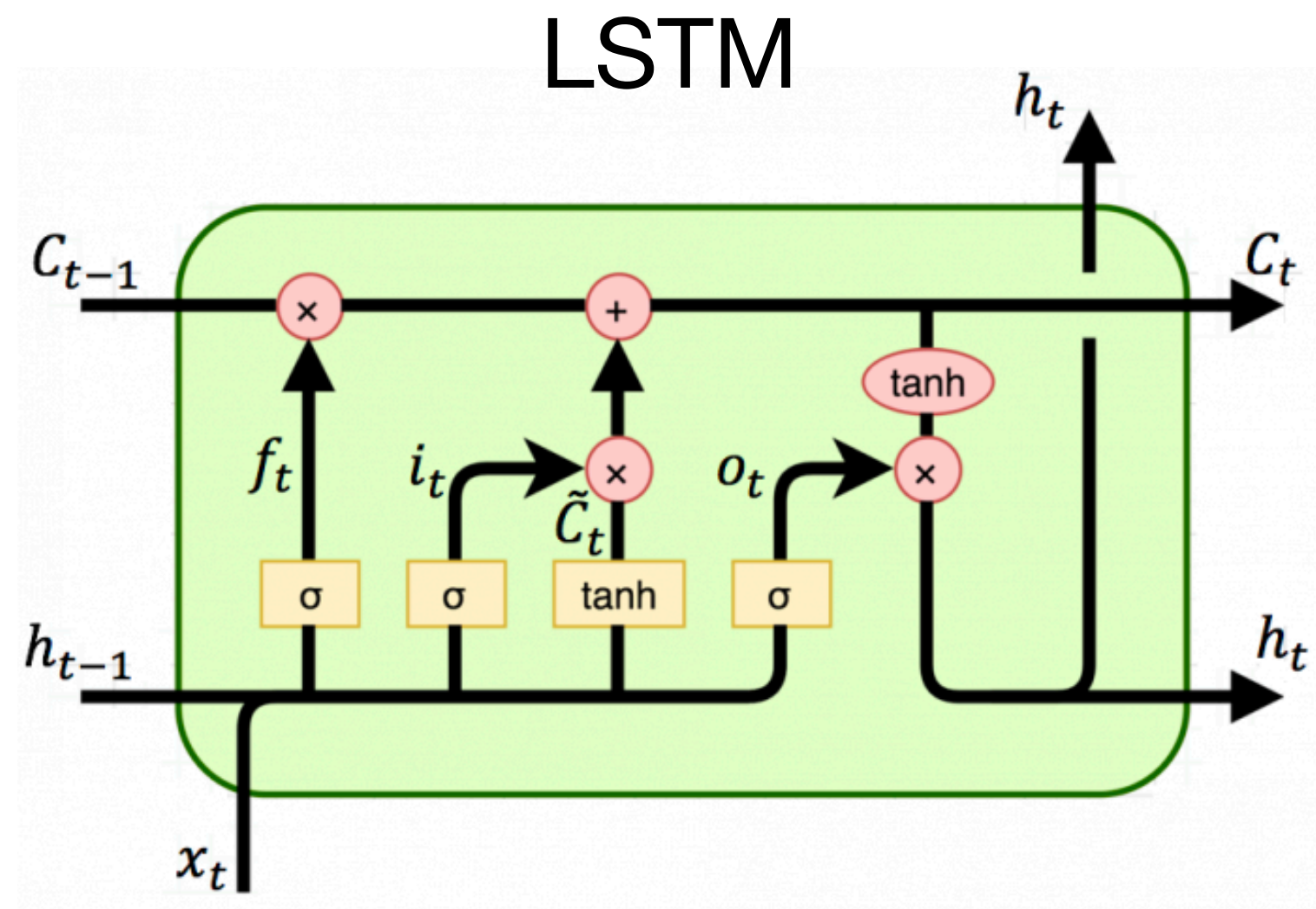


$$z_t = \sigma(W_z x_{t-1} + U_z h_{t-1} + b_z)$$

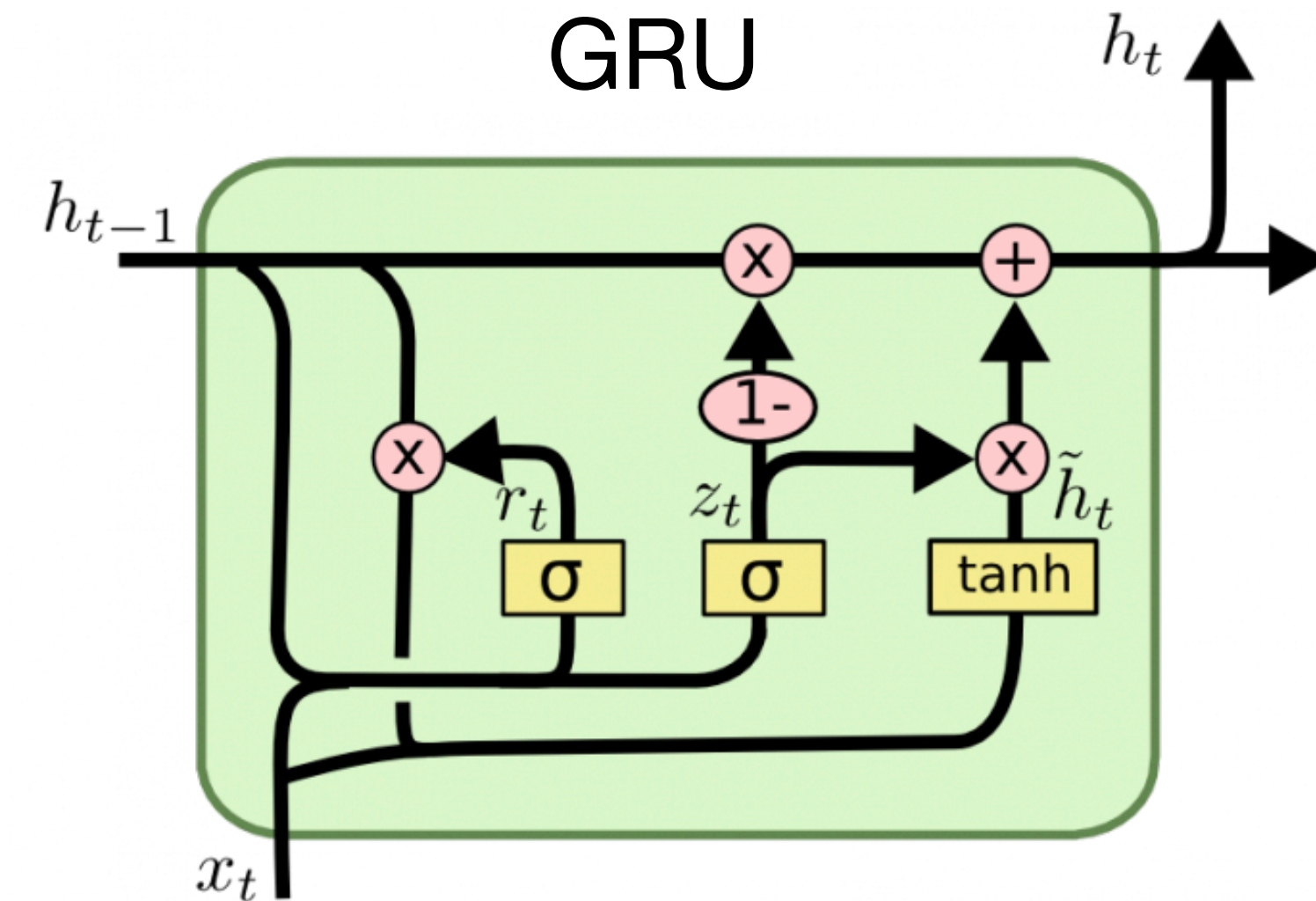
- Фильтры забывания и входа объединены.
- Если мы забыли часть информации, то эту же часть надо дополнить новой.



# Gated recurrent units (GRU)



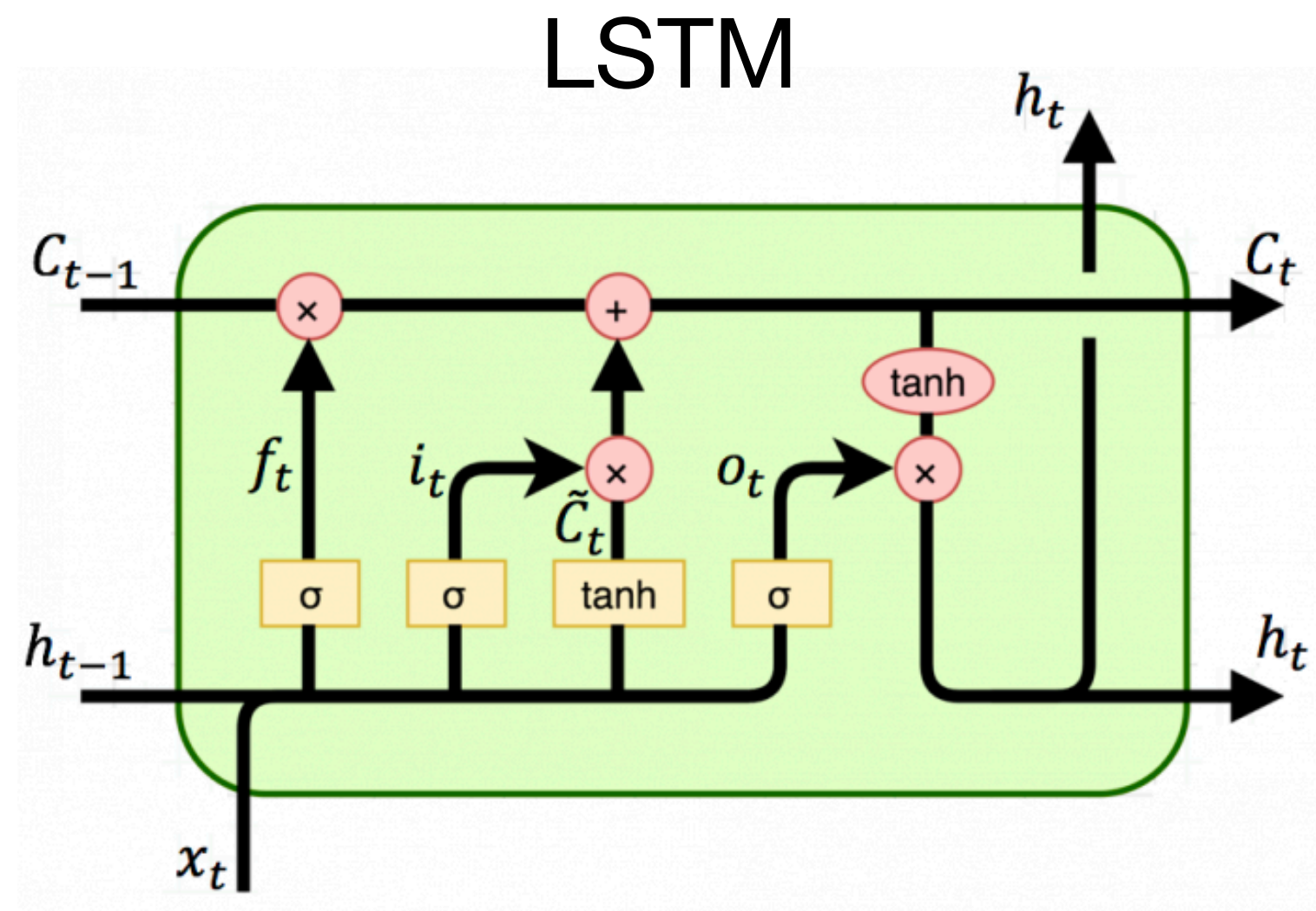
$$o_t = \sigma(W_t x_{t-1} + U_t h_{t-1} + b_t)$$



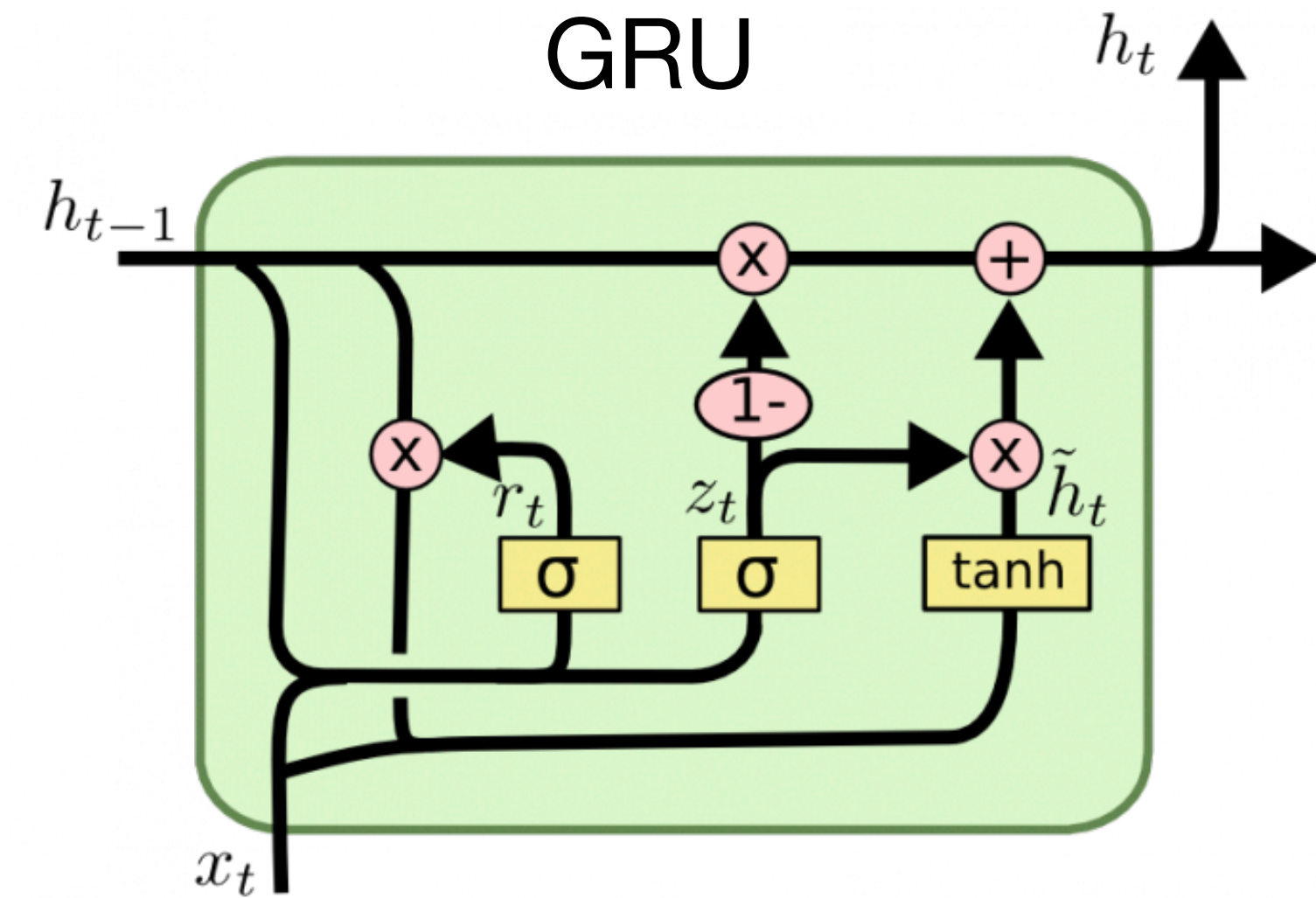
$$r_t = \sigma(W_r x_{t-1} + U_r h_{t-1} + b_r)$$

Контролируют, какая информация нужна для текущего выхода

# Gated recurrent units (GRU)



$$\tilde{c}_t = \tanh(W_i x_{t-1} + U_i h_{t-1} + b_i)$$

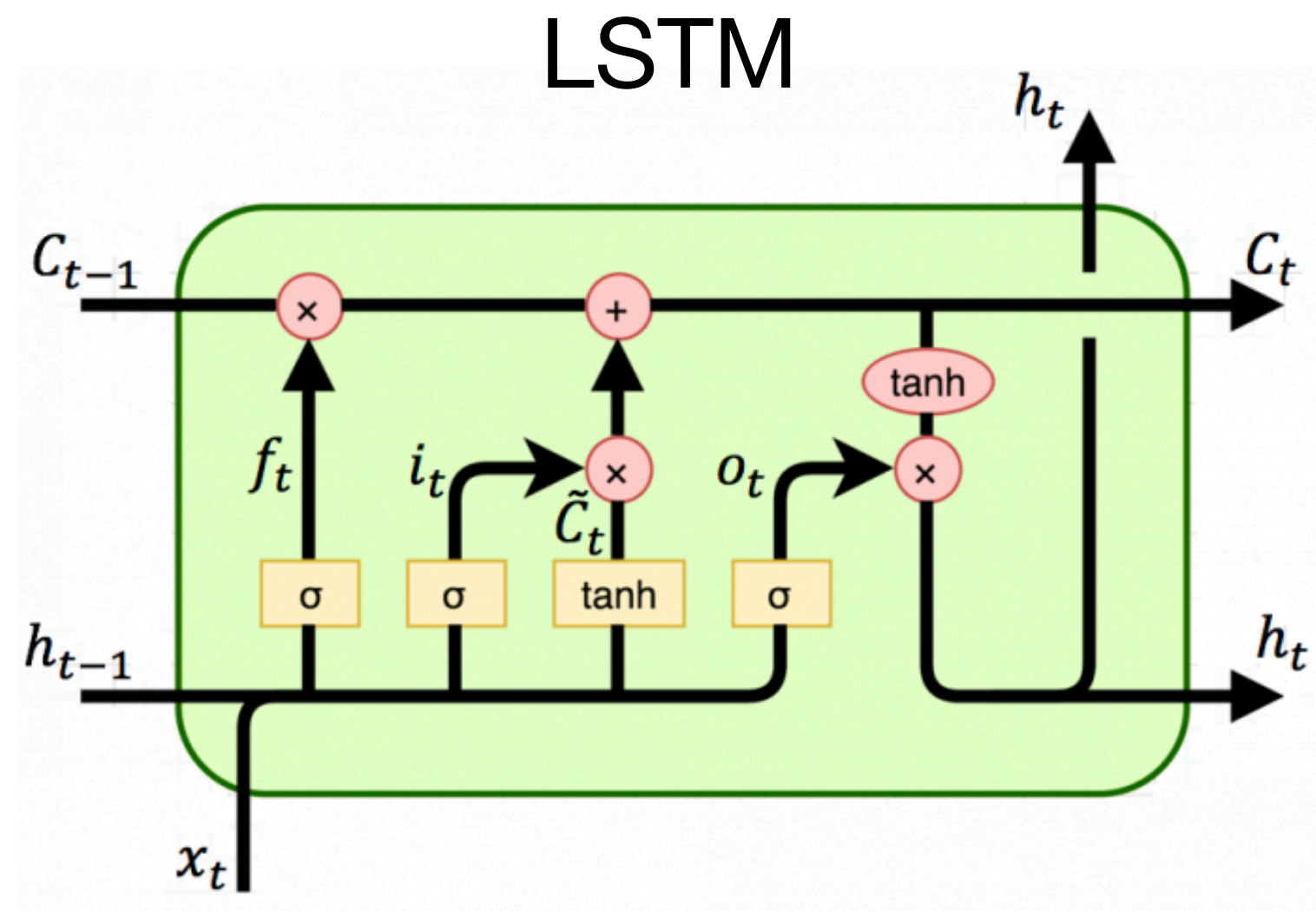


$$\tilde{h}_t = \tanh(W_h x_{t-1} + U_h (r_t \odot h_{t-1}) + b_h)$$

Выбирают, какую информацию надо сохранить.

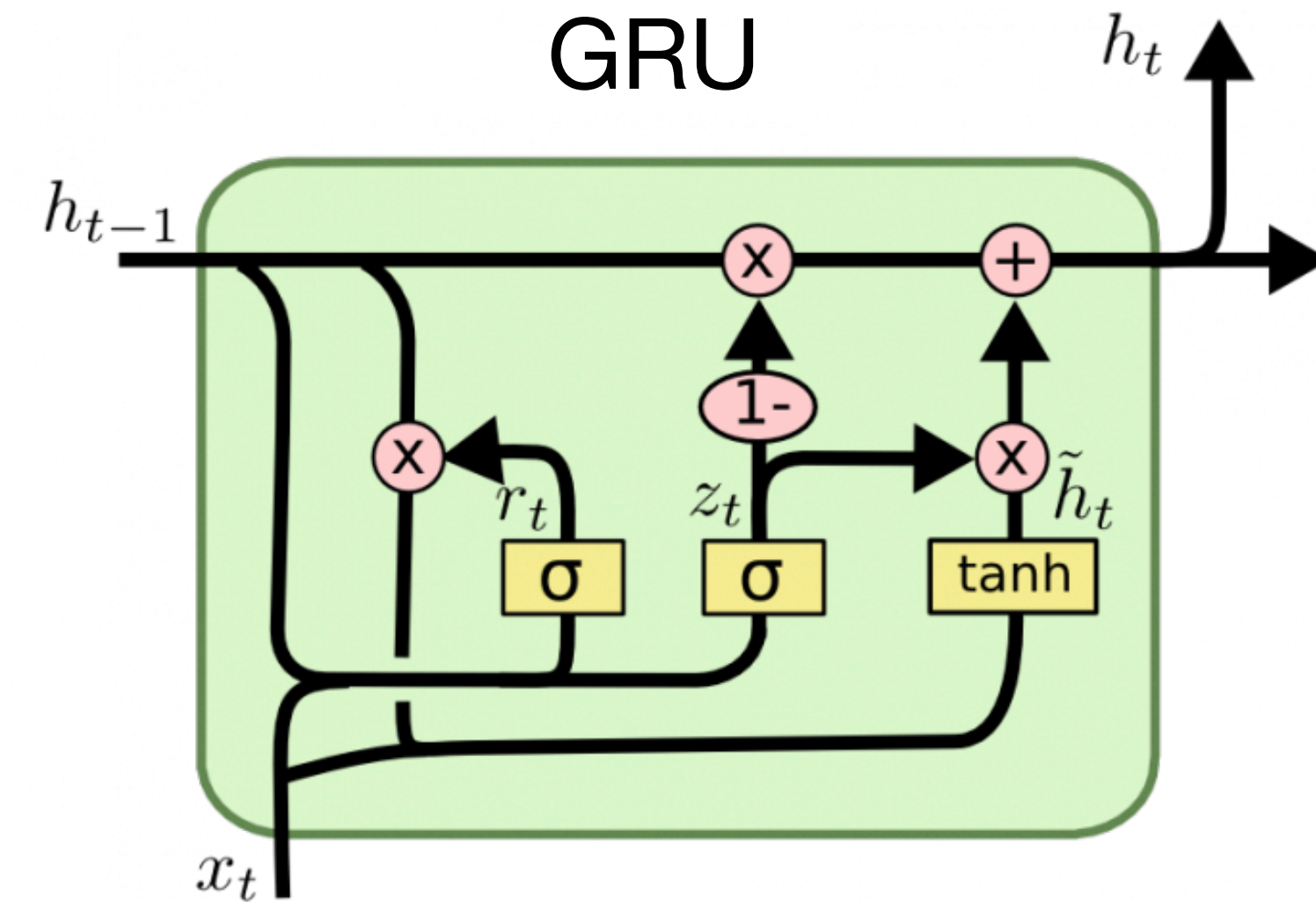


# Gated recurrent units (GRU)



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

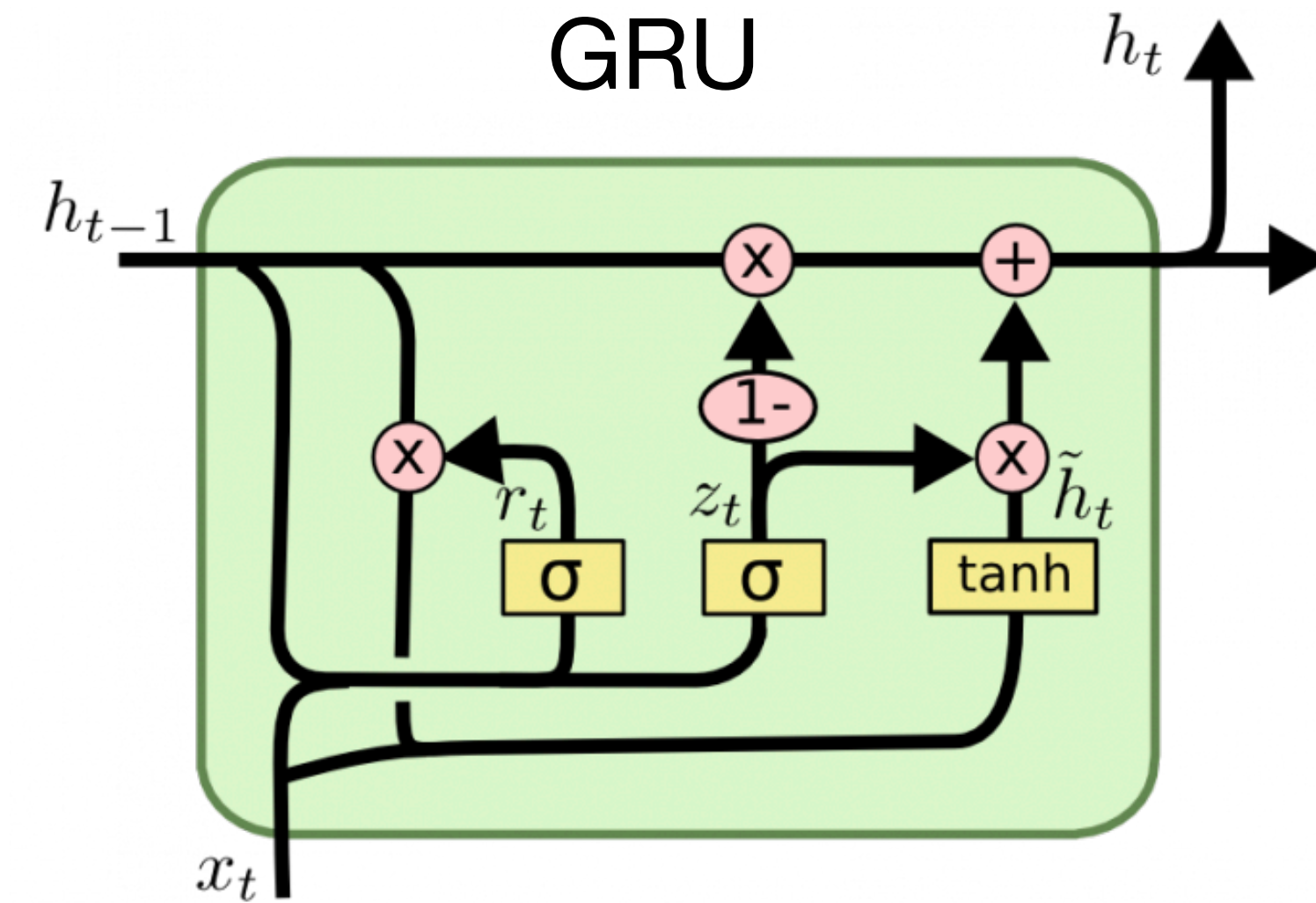
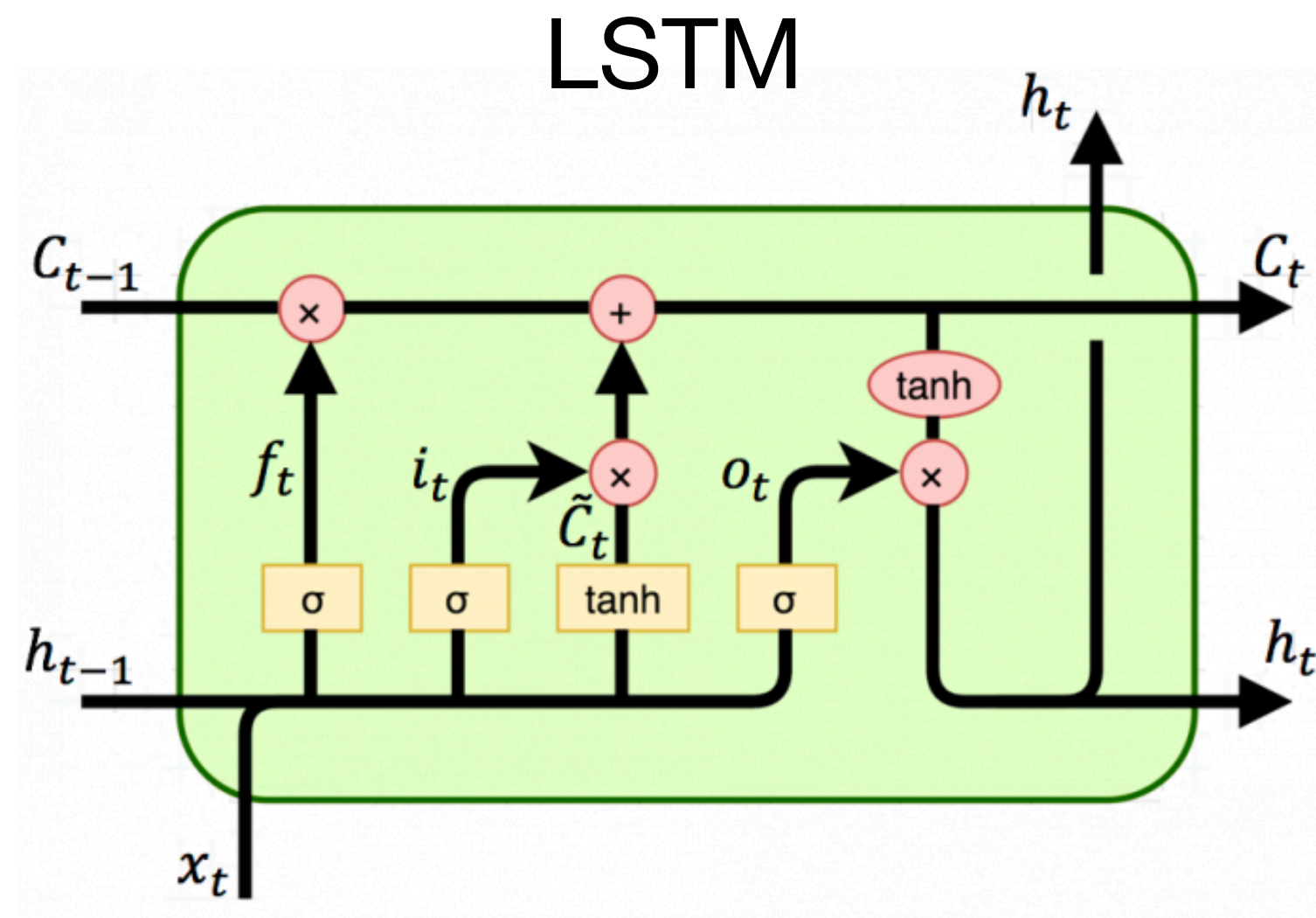
$$h_t = o_t \odot \tanh(c_t)$$



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

Обновление скрытого состояния.

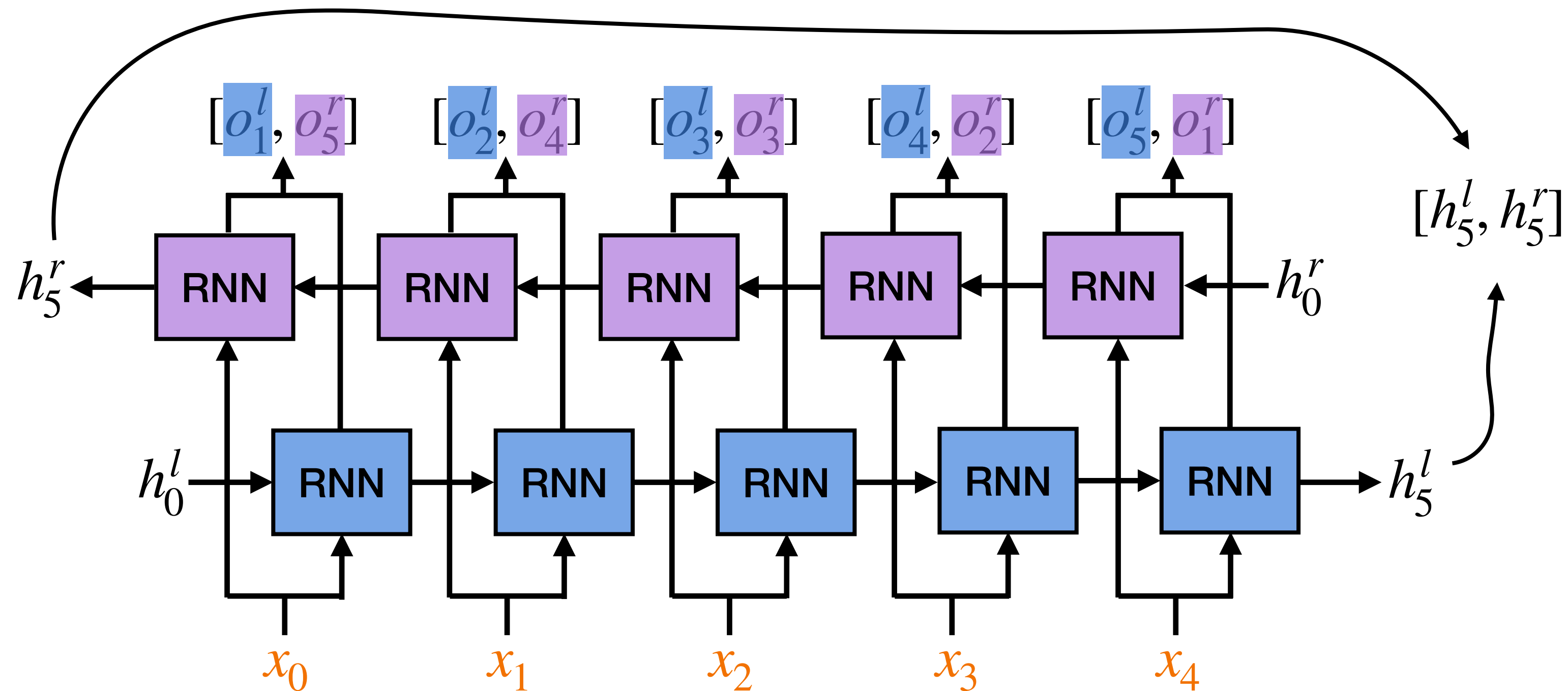
# Сравнение GRU и LSTM



- Обе модели хорошо работают с удаленными зависимостями.
- В GRU 3 слоя, у LSTM – 4.
- GRU лучше учится, когда данных мало. LSTM лучше, когда данных много.

# Двунаправленные рекуррентные сети (biRNN)

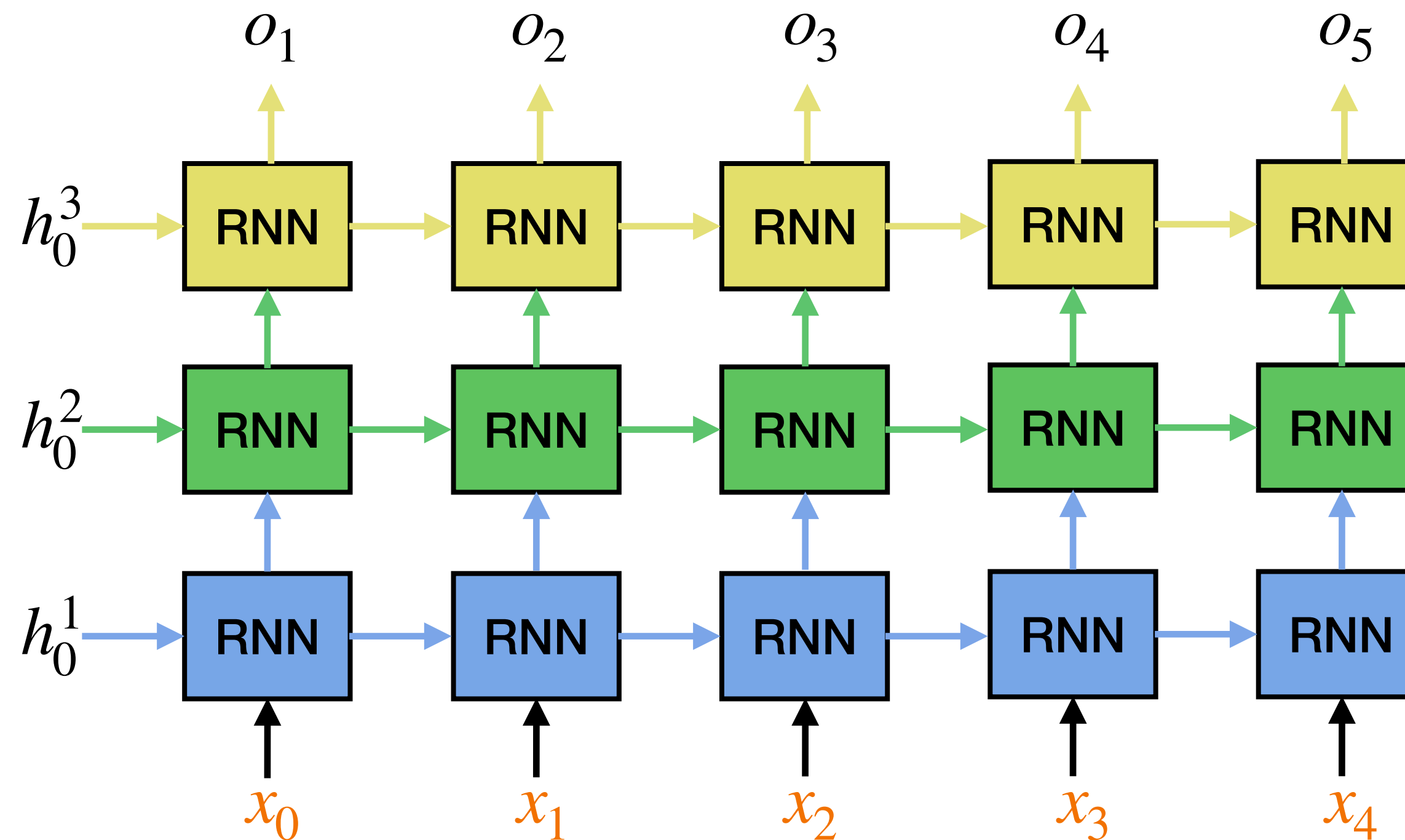
- Читает текст **слева направо** и **справа налево**.



- Имеет в два раза больше параметров.
- Бесполезна для задачи генерации.

# Многослойные рекуррентные сети (biRNN)

- Выходы текущего слоя являются входами следующего.
- Извлекаются более сложные признаки.



- Число параметров увеличивается в число слоев раз.
- Обычно ограничиваются **двумя** слоями.



# Итог

- Рекуррентная нейронная сеть (**RNN**) предназначена для **обработки последовательностей**.
- Она может использоваться для **любых** задач NLP.
- Для борьбы с затуханием градиентов придумали **LSTM** и **GRU**.
- Обе модели используют набор фильтров, позволяющих забывать неактуальную информацию и запоминать важную.
- Для улучшения производительности модели можно:
  - Добавить сеть, читающую текст в обратном направлении.
  - Увеличить число слоев сети.