

# Рекуррентные нейронные сети

# Как предсказать следующее слово?

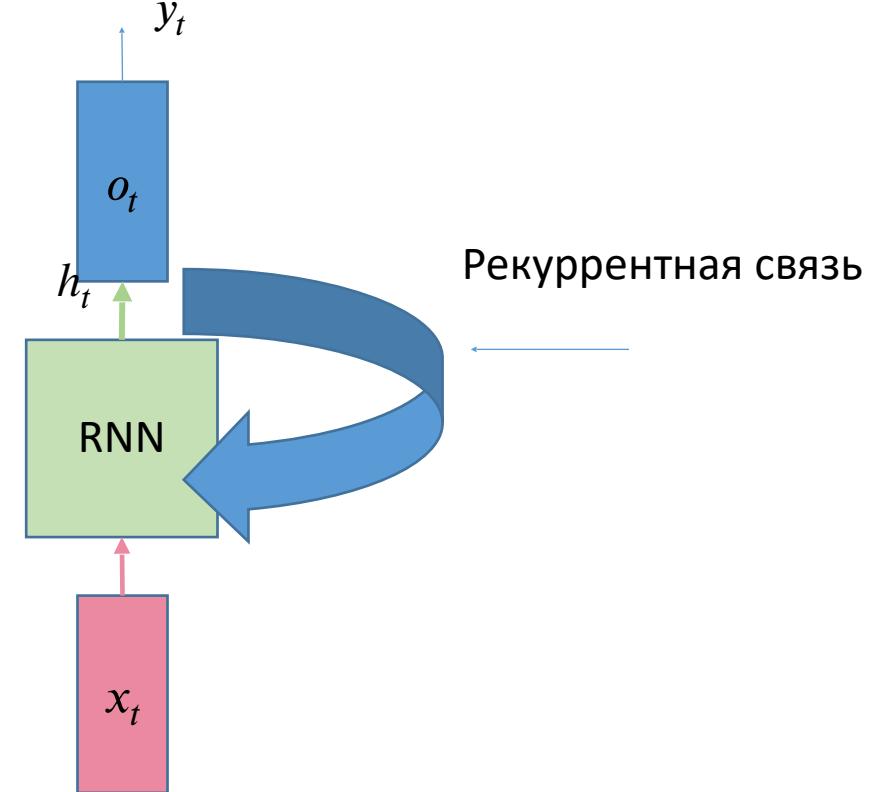
- Как обрабатывать длинные временные последовательности?
- Посмотреть на предыдущее слово?
- Посмотреть на предыдущие 3 слова?
- Можно ли учитывать контекст, и как это делать?

# Сеть прямого распространения vs рекуррентная сеть

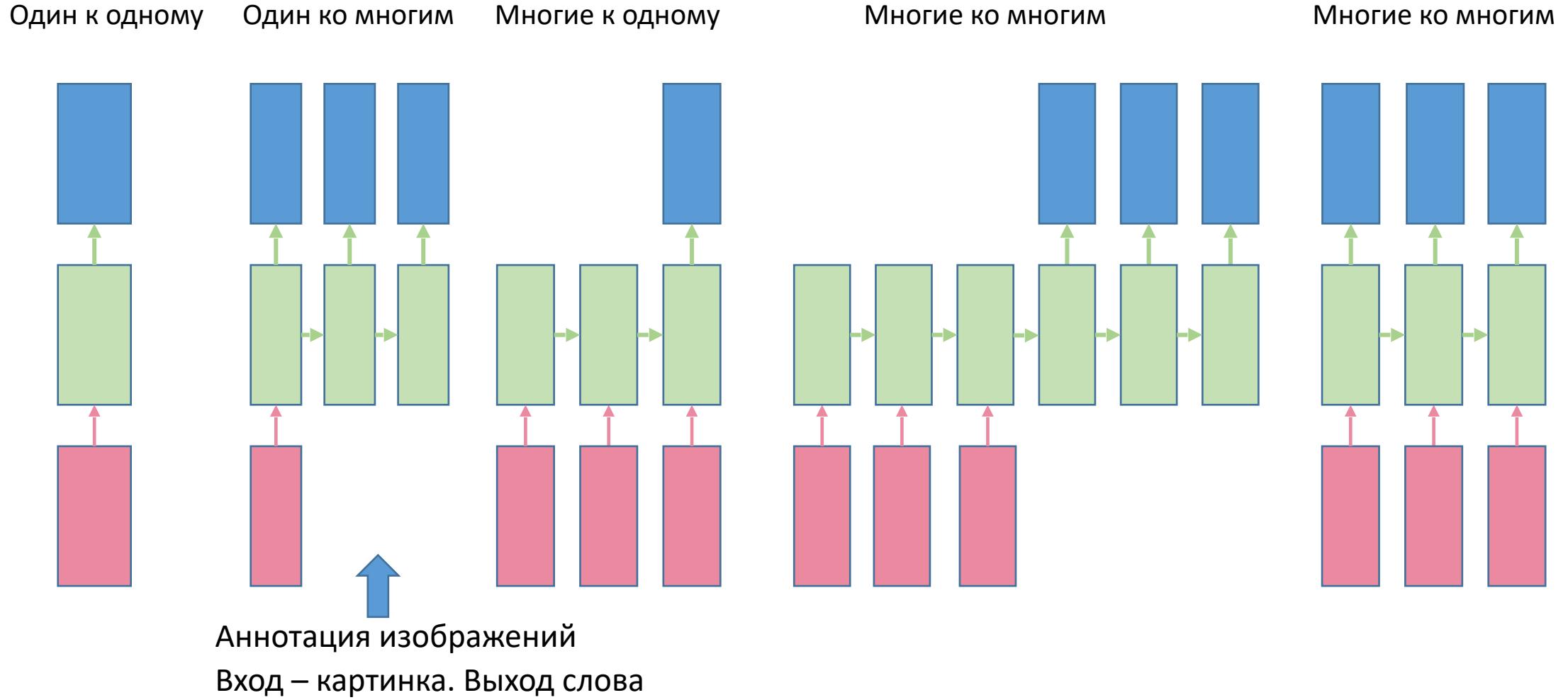
Один к одному



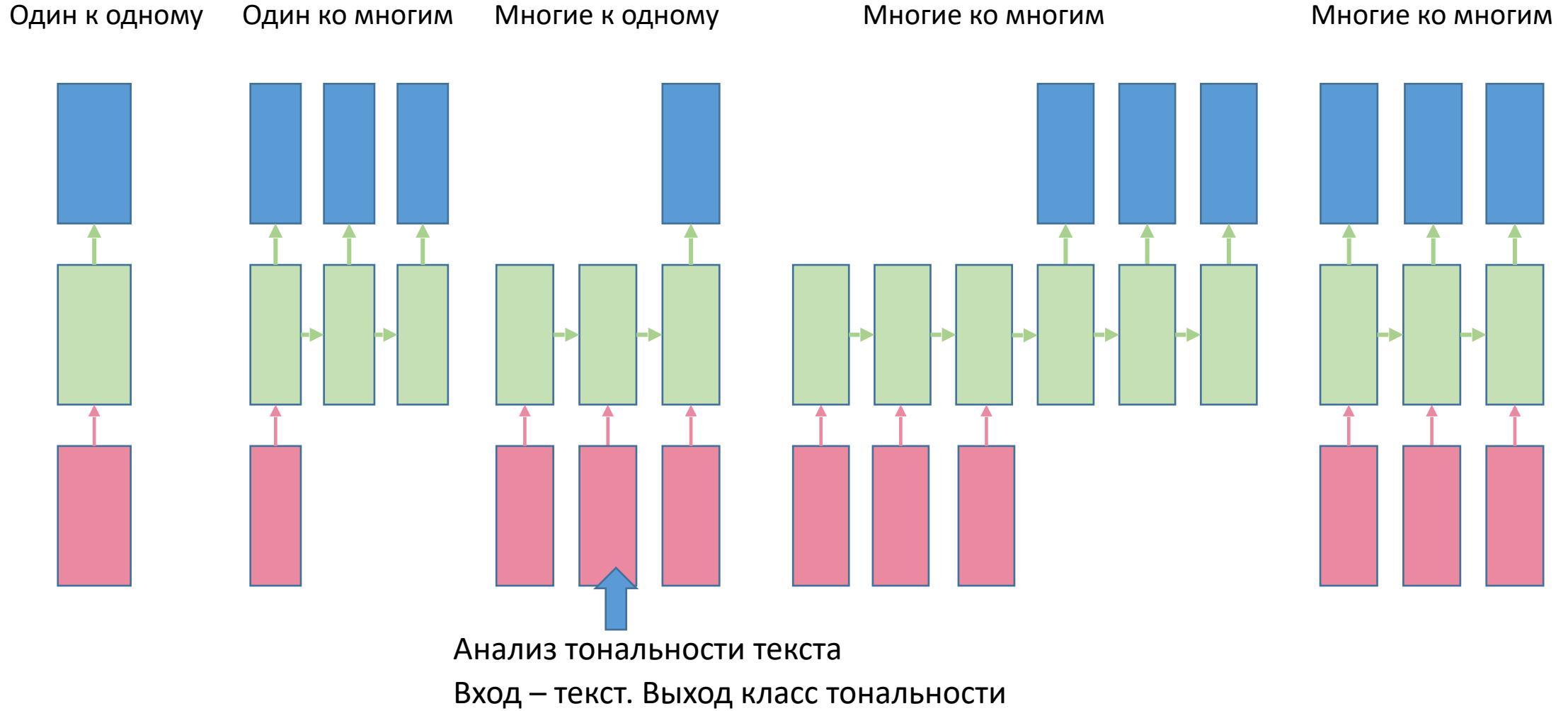
Для обработки последовательностей  
лучше использовать рекуррентную сеть



# Рекуррентная нейросеть

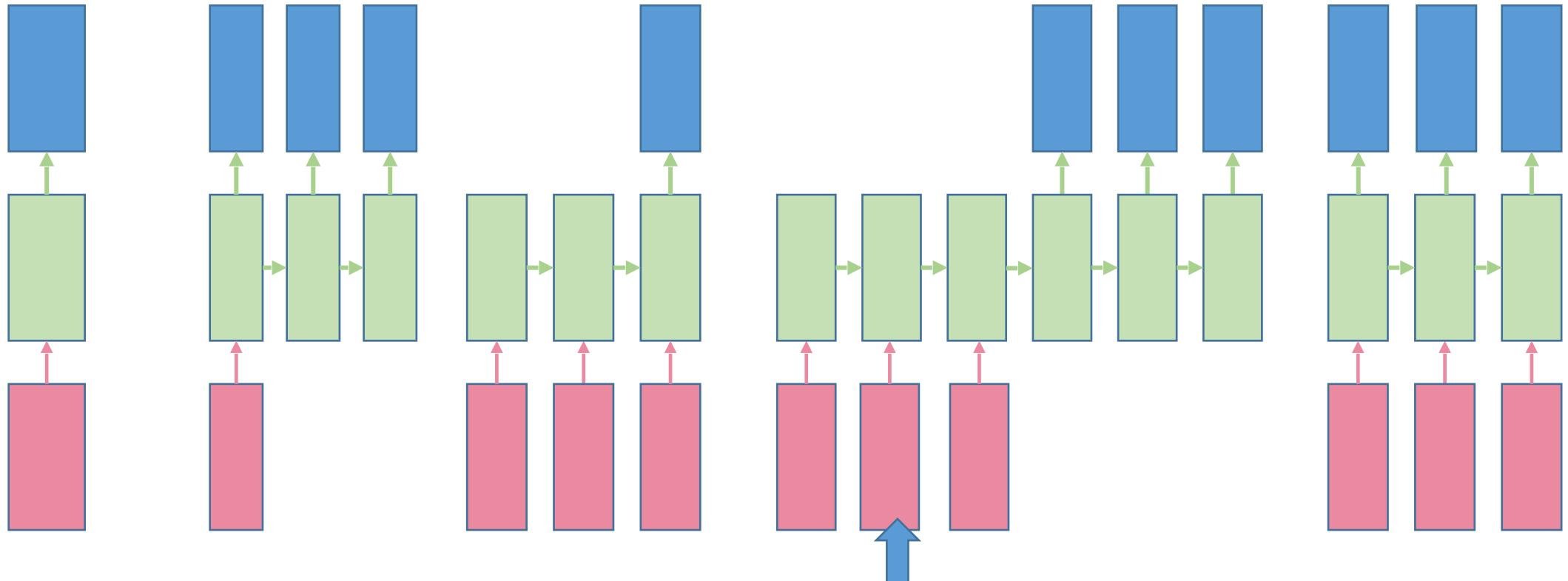


# Рекуррентная нейросеть



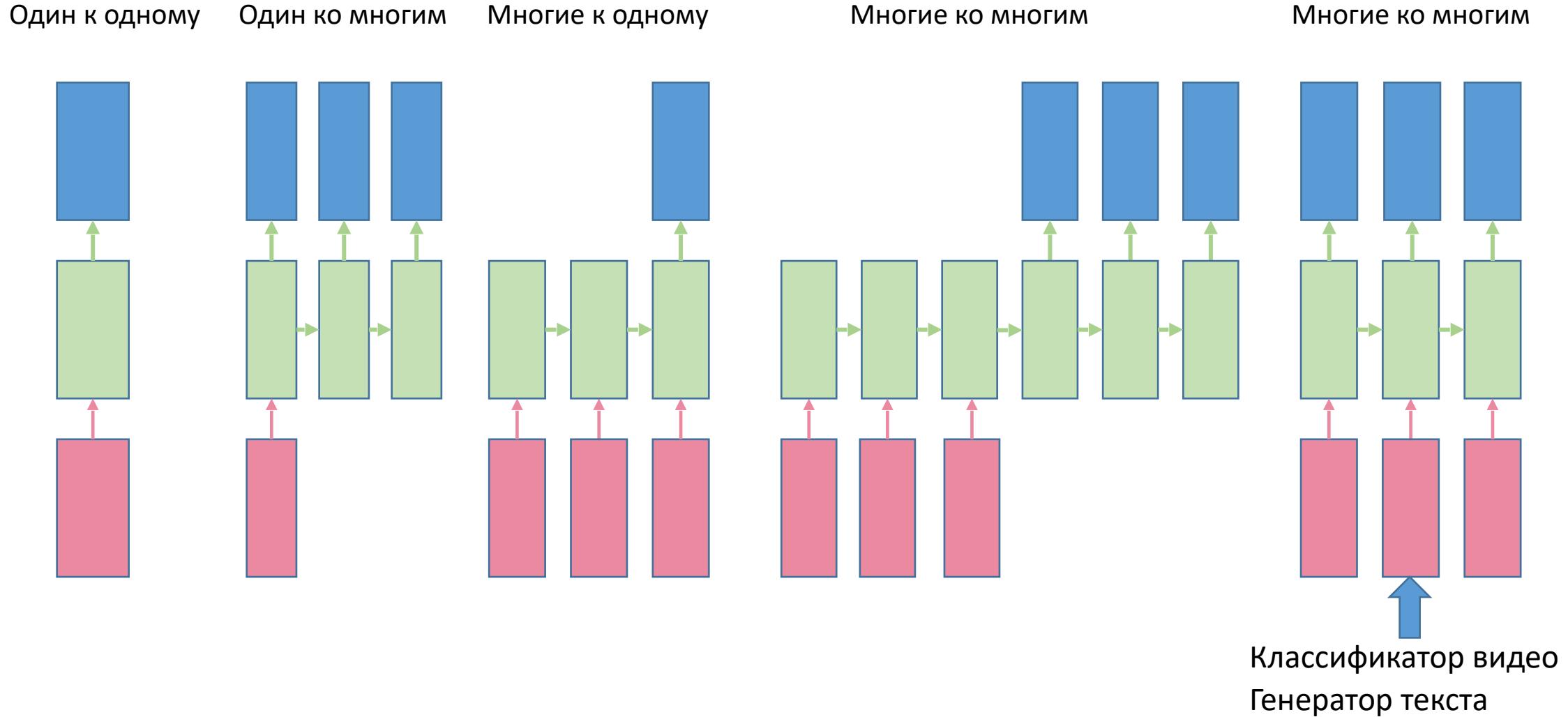
# Рекуррентная нейросеть

Один к одному    Один ко многим    Многие к одному    Многие ко многим    Многие ко многим



Переводчик, чат боты  
Вход – текст. Выход - текст

# Рекуррентная нейросеть

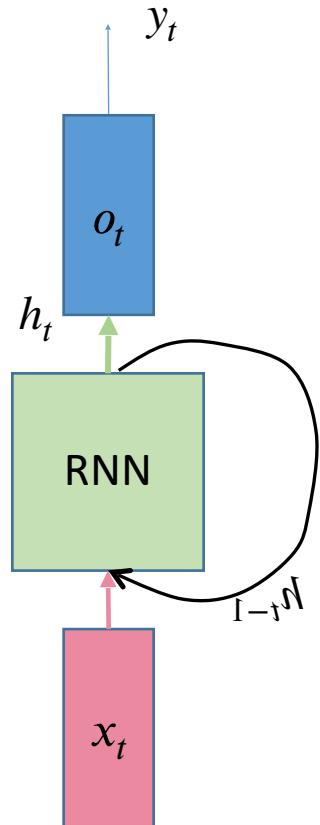


# Рекуррентная формула

Сеть подает на вход в момент времени  $t$  выход  $t-1$

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

Новое состояние  
Старое состояние  
Вход на шаге  $t$   
Функция с параметрами  $W$



# Рекуррентная формула

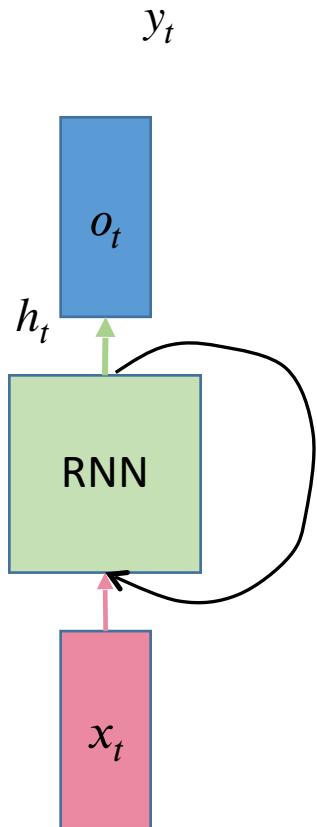
Сеть подает на вход в момент времени  $t$  выход  $t-1$

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

Новое состояние      Старое состояние      Вход на шаге  $t$

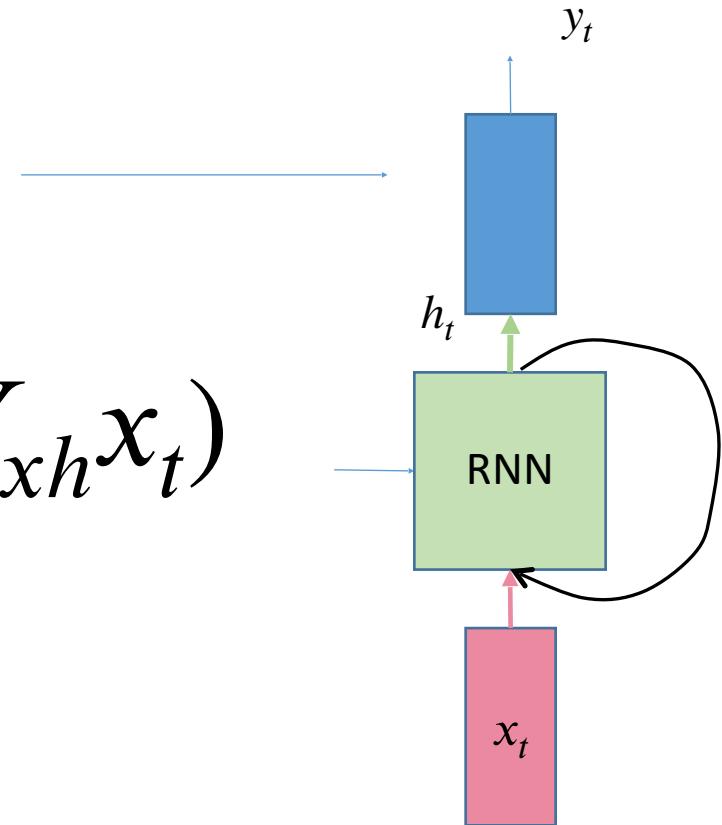
Функция с параметрами  $W$

Для каждого момента времени используется  
одна функция и одна матрица весов

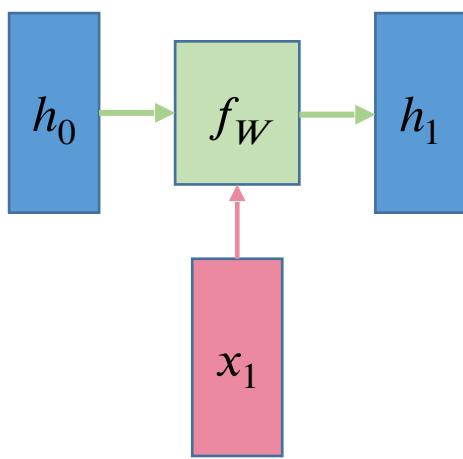


# Базовая рекуррентная сеть

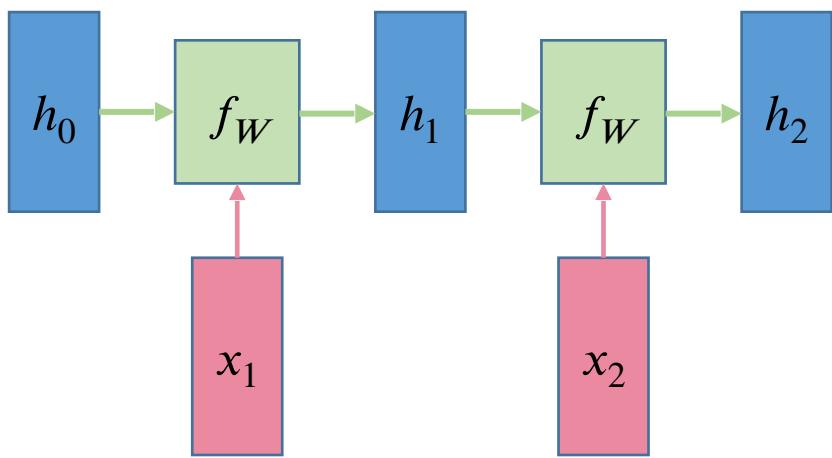
$$y_t = W_{hy} h_t$$
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$
$$h_t = f_W(h_{t-1}, x_t)$$



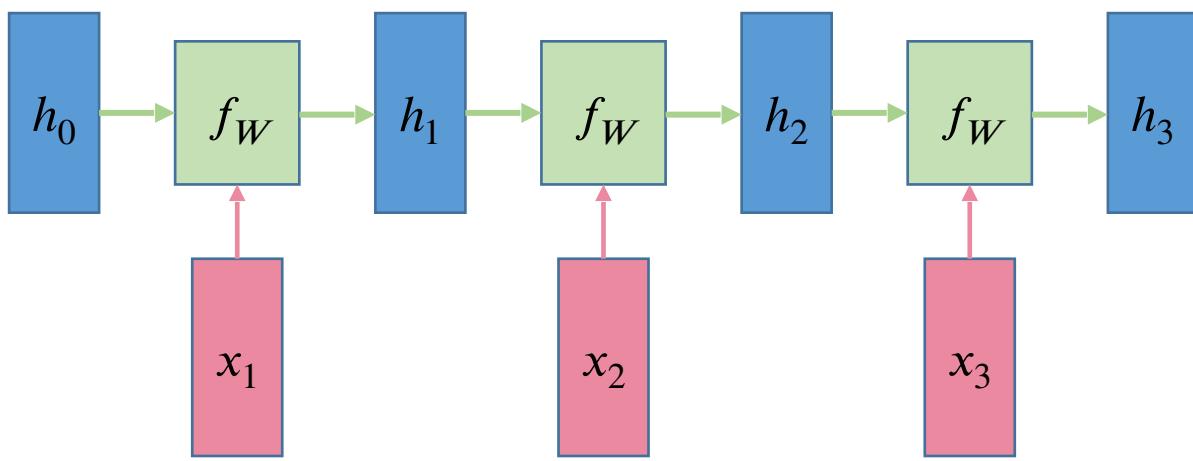
# RNN вычислительный граф



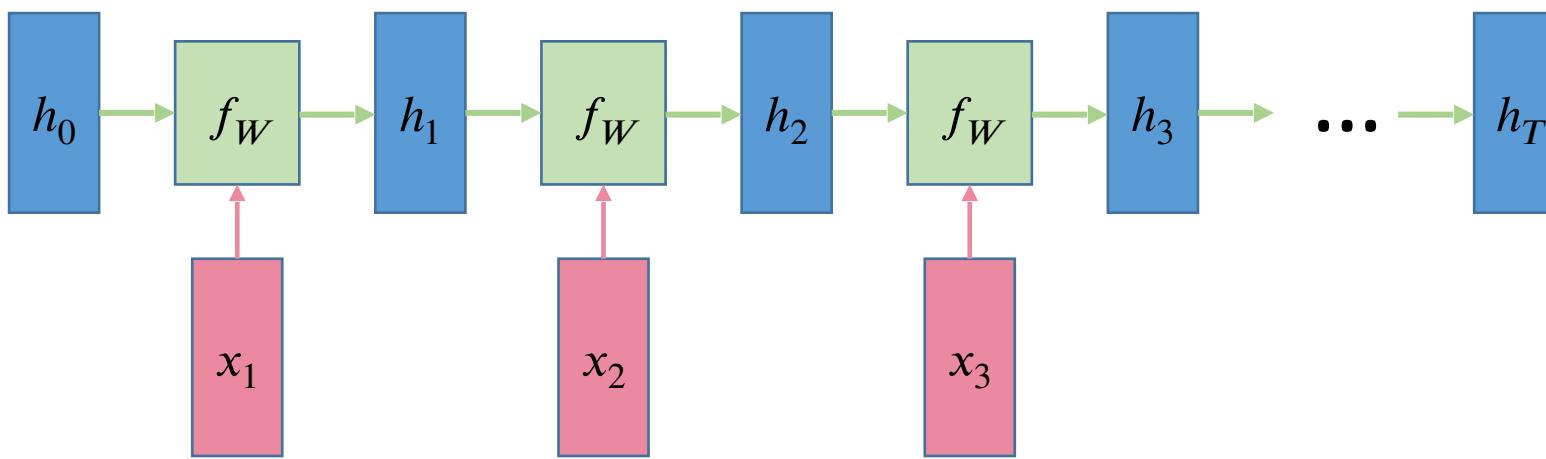
# RNN вычислительный граф



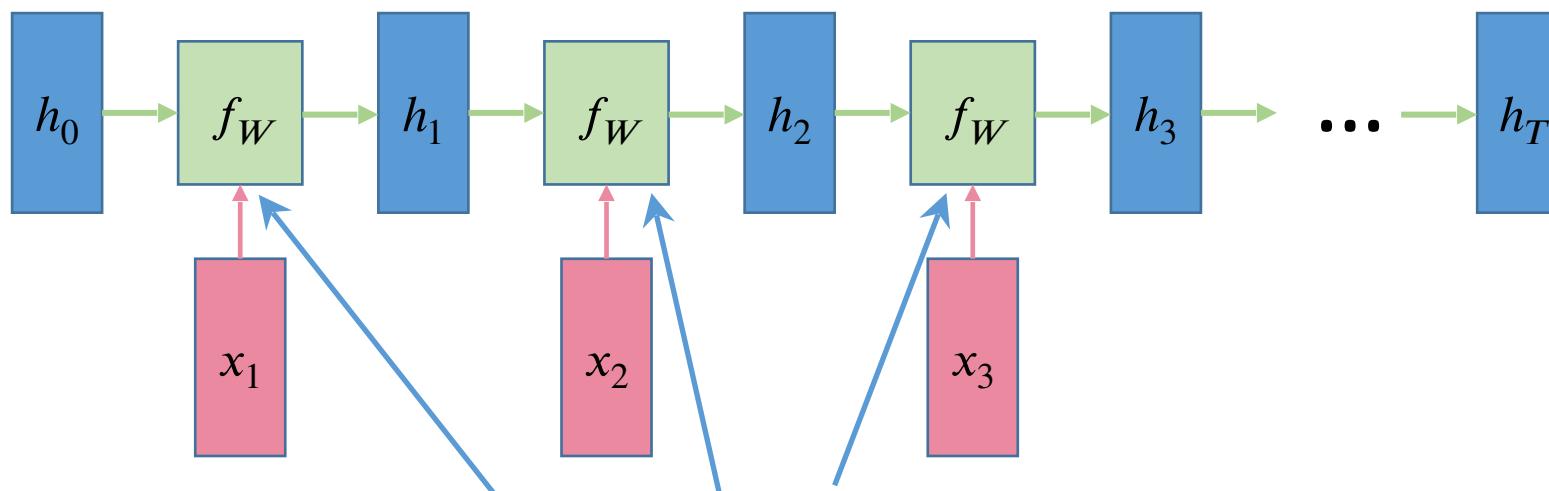
# RNN вычислительный граф



# RNN вычислительный граф

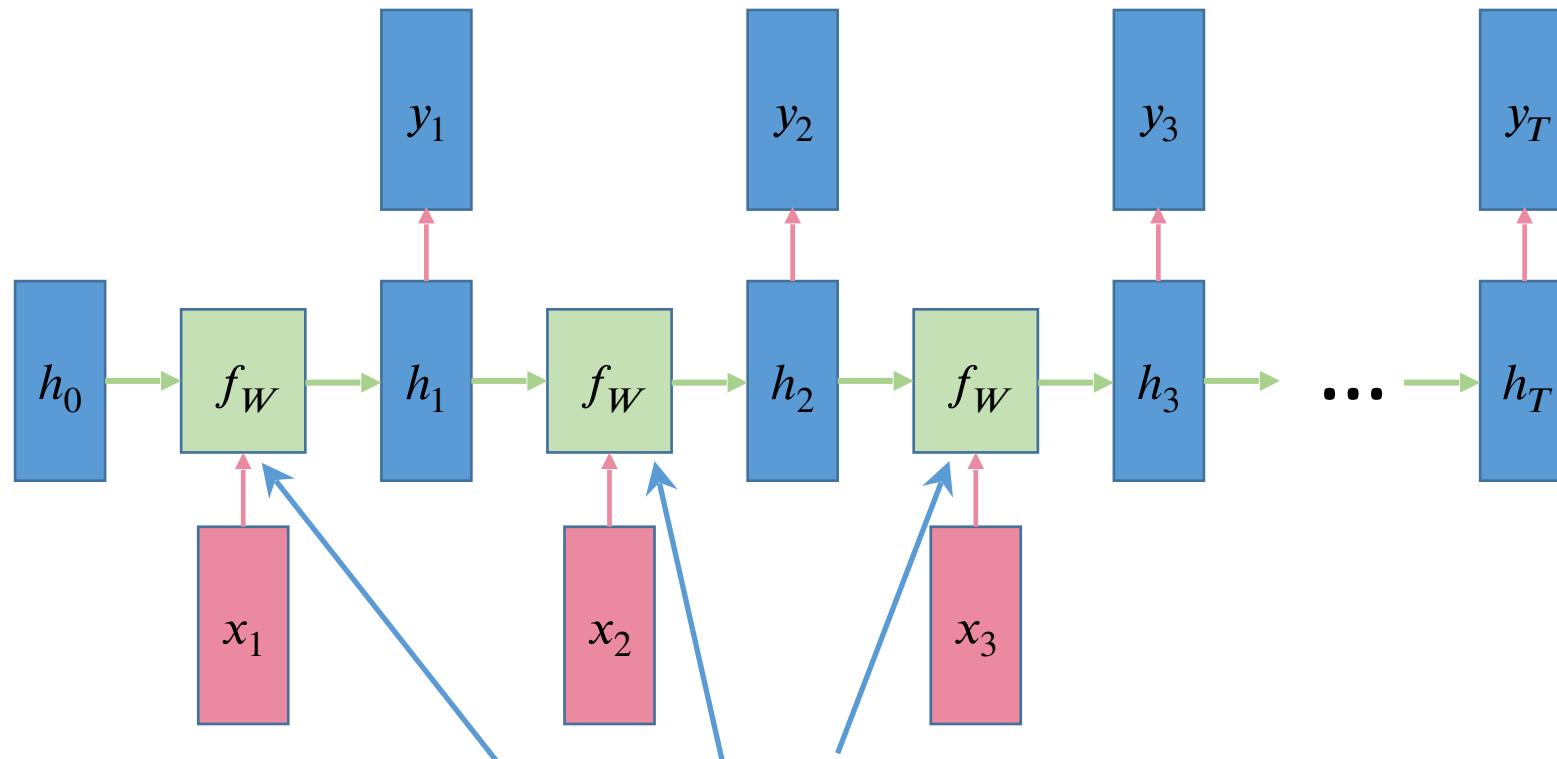


# RNN вычислительный граф



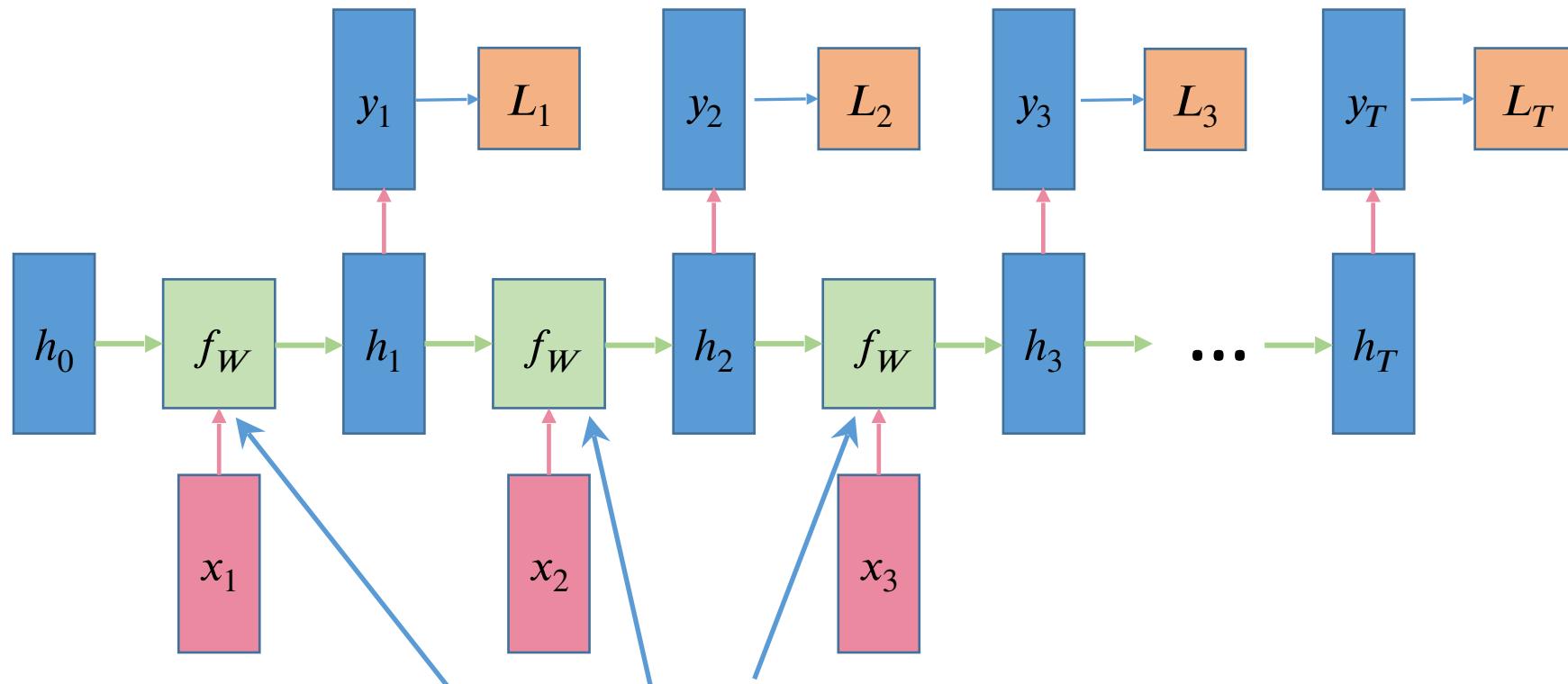
Одна матрица  $W$  на каждом шаге

# Вычислительный граф. Многие ко многим



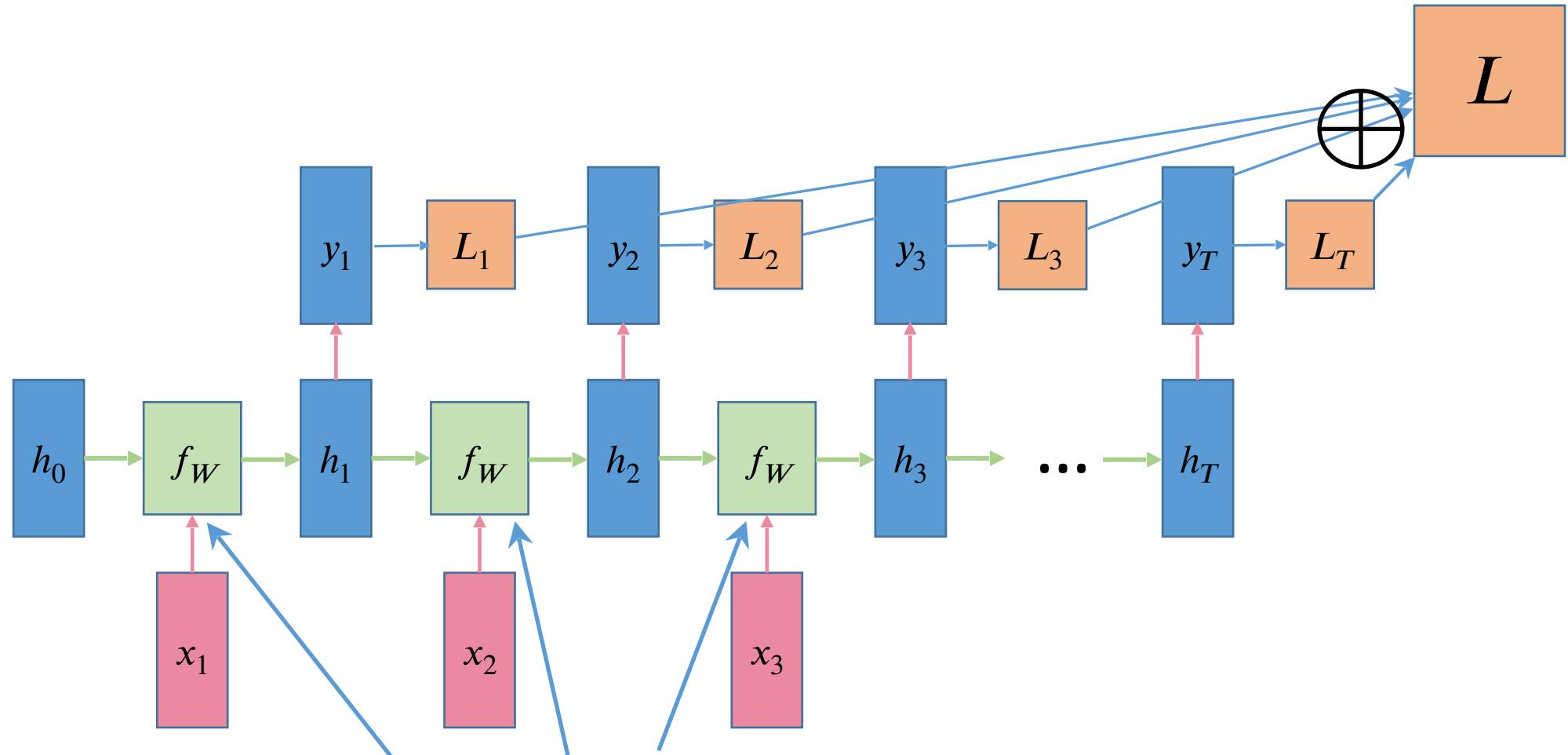
Одна матрица  $W$  на каждом шаге

# Вычислительный граф. Многие ко многим



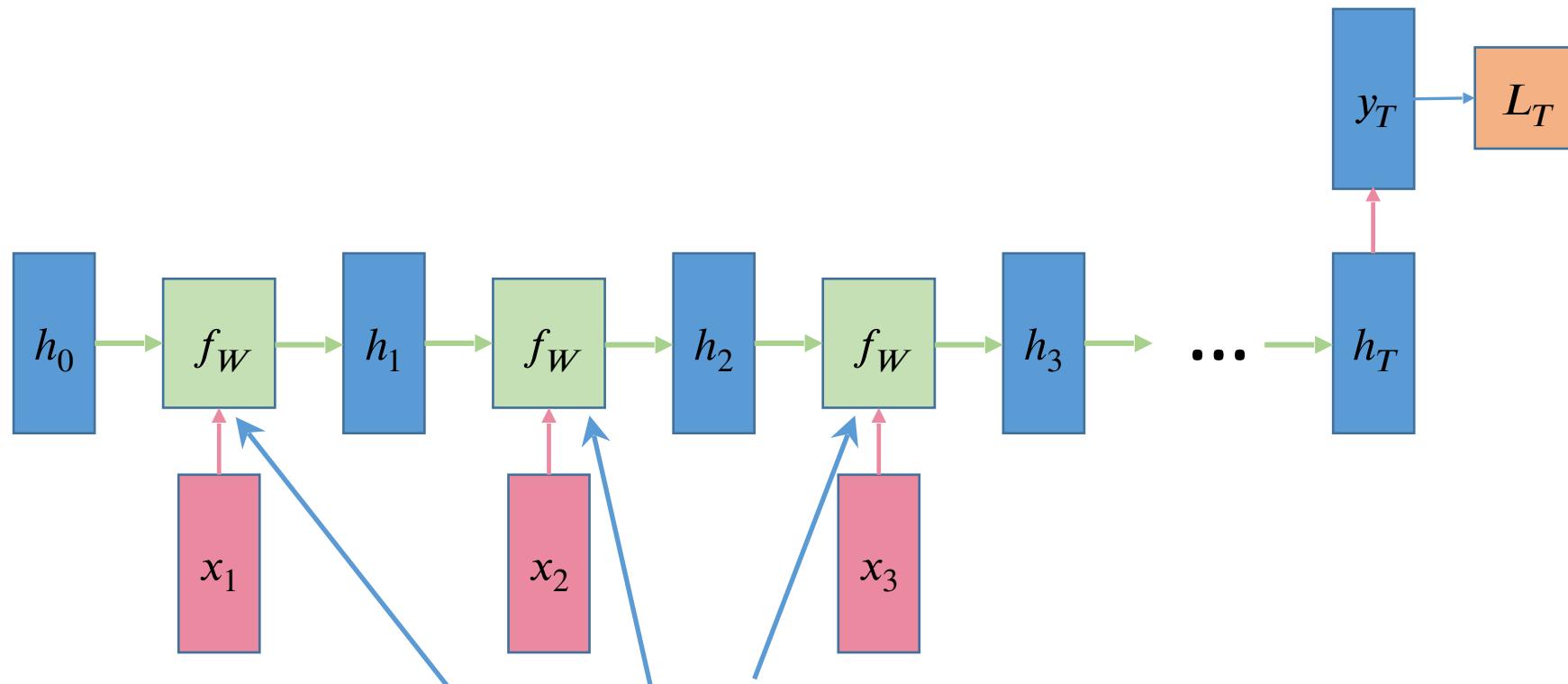
Одна матрица  $W$  на каждом шаге

# Вычислительный граф. Многие ко многим



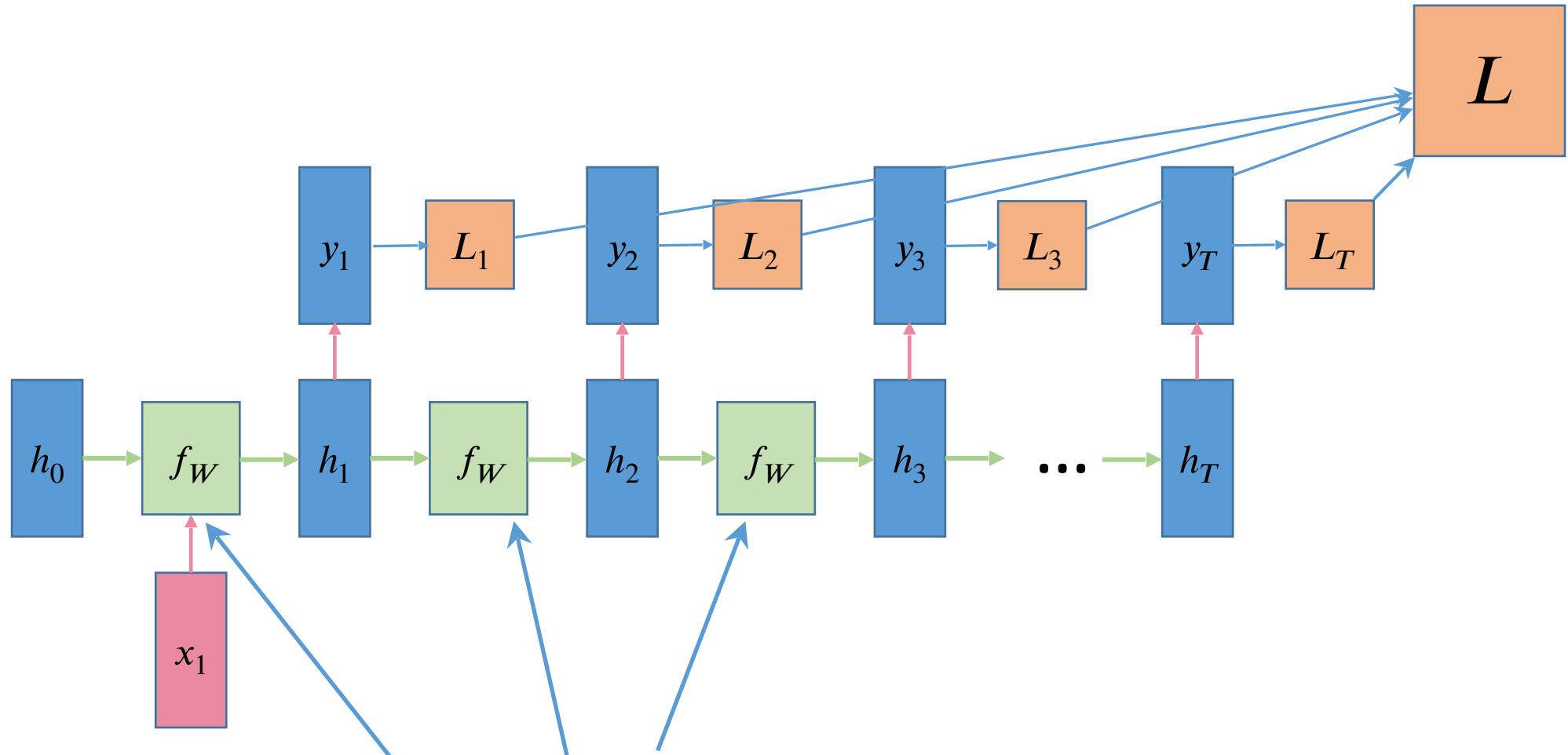
Одна матрица  $W$  на каждом шаге

# Вычислительный граф. Многие к одному



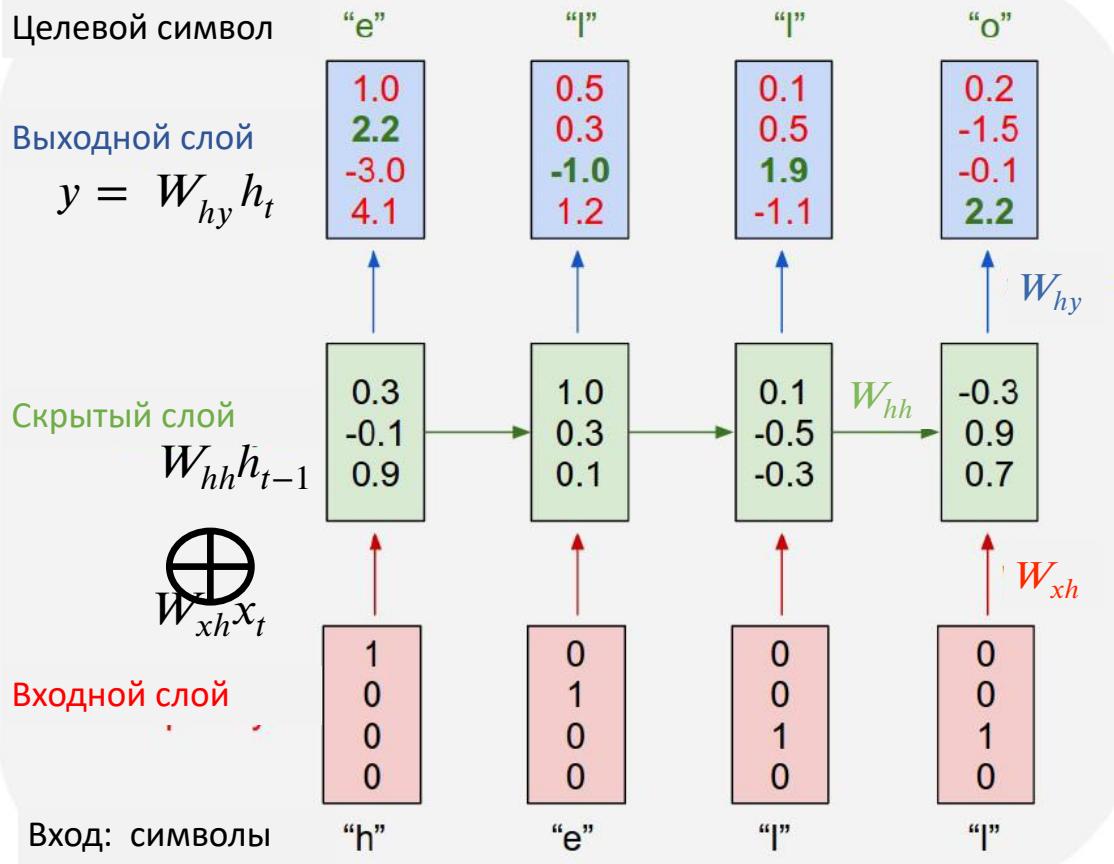
Одна матрица  $W$  на каждом шаге

# Вычислительный граф. Один ко многим



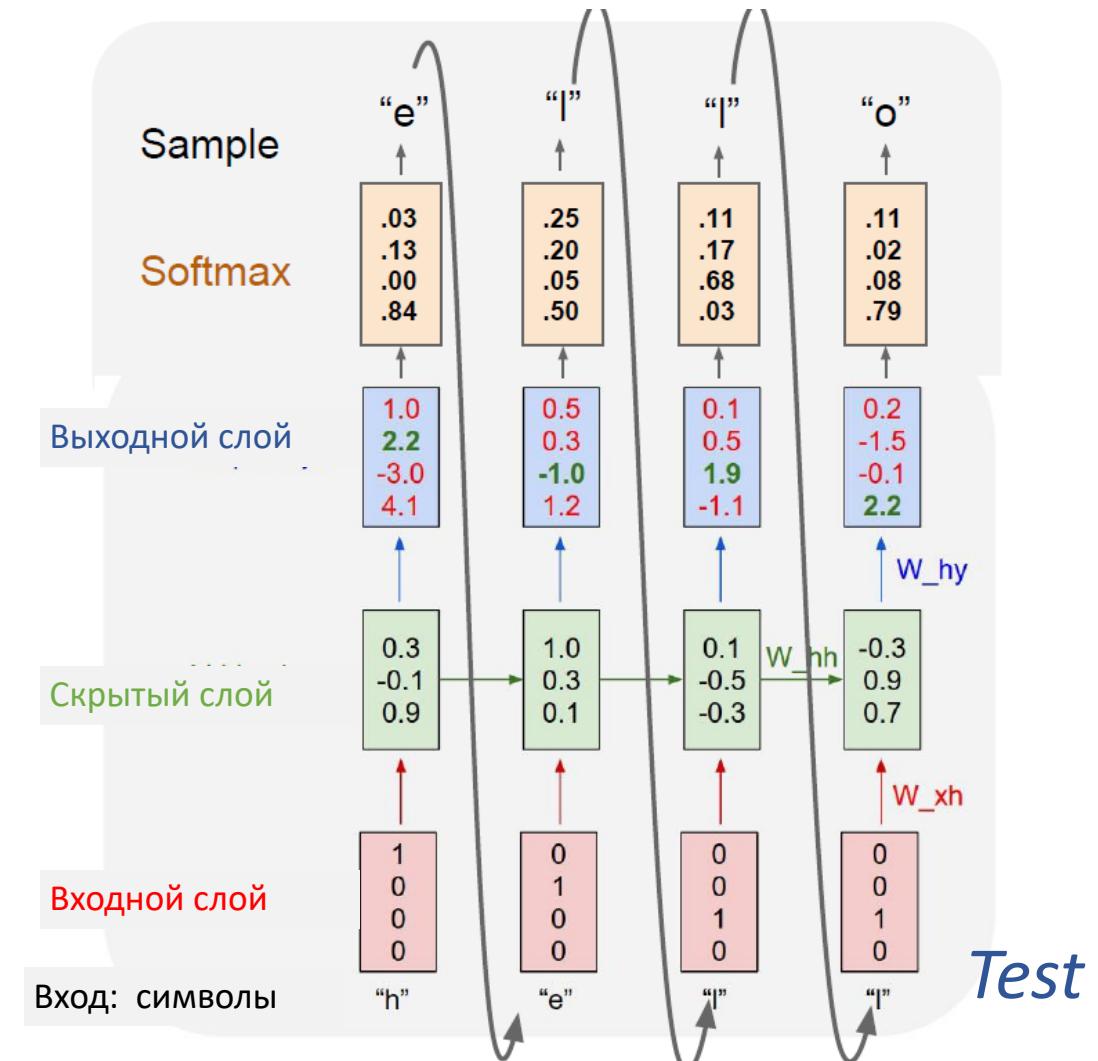
Одна матрица  $W$  на каждом шаге

# Character-level. Языковая модель

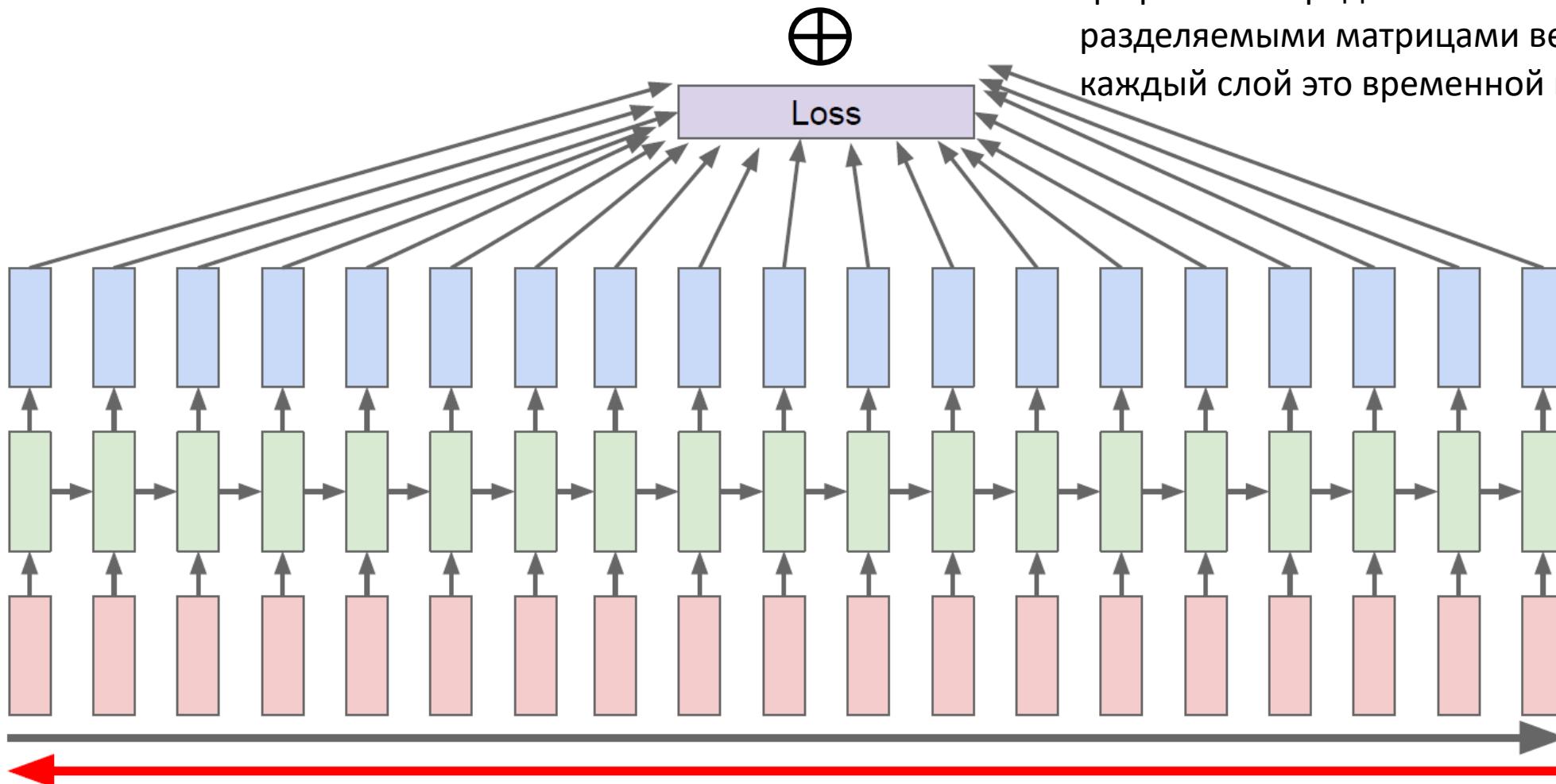


*Train*  
Teacher forcing

Словарь: [h,e,I,o]



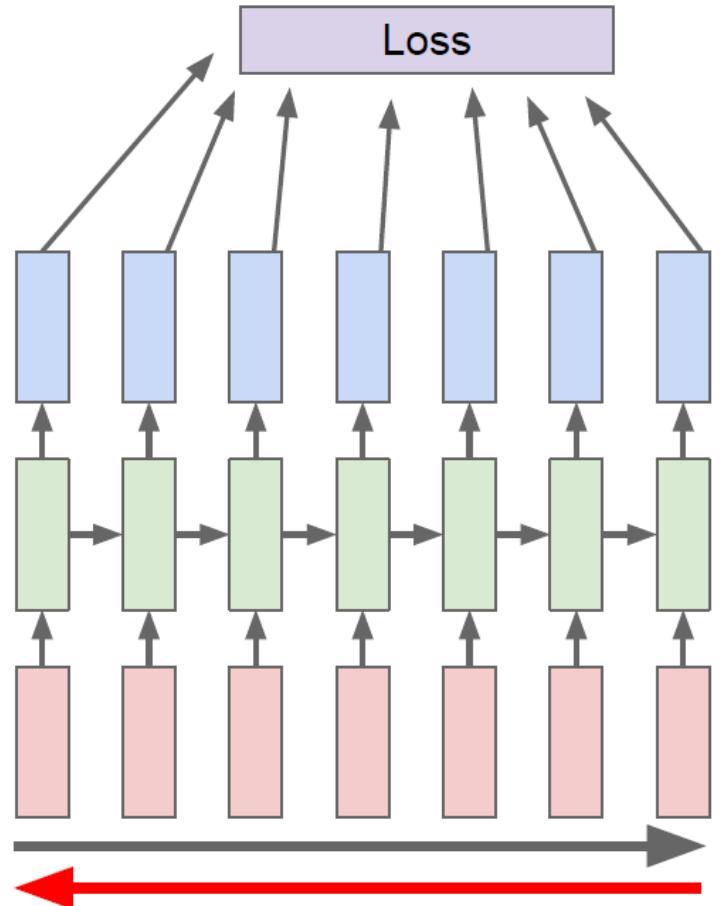
# Backpropagation по времени



Граф можно представить как глубокую сеть с разделяемыми матрицами весов, где каждый слой это временной шаг.

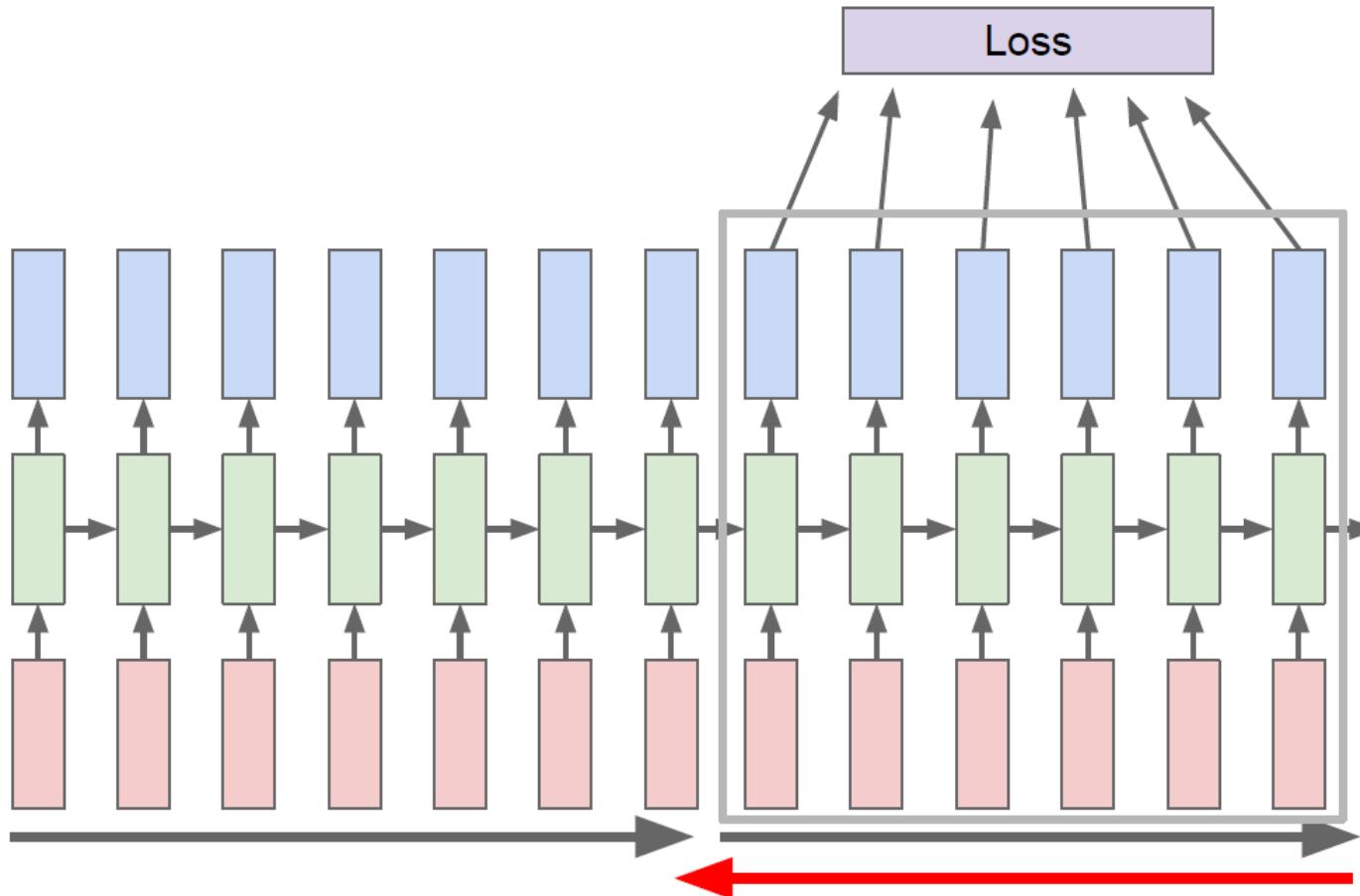
Прямой и обратный проход по всей последовательности

# Truncated Backpropagation по времени



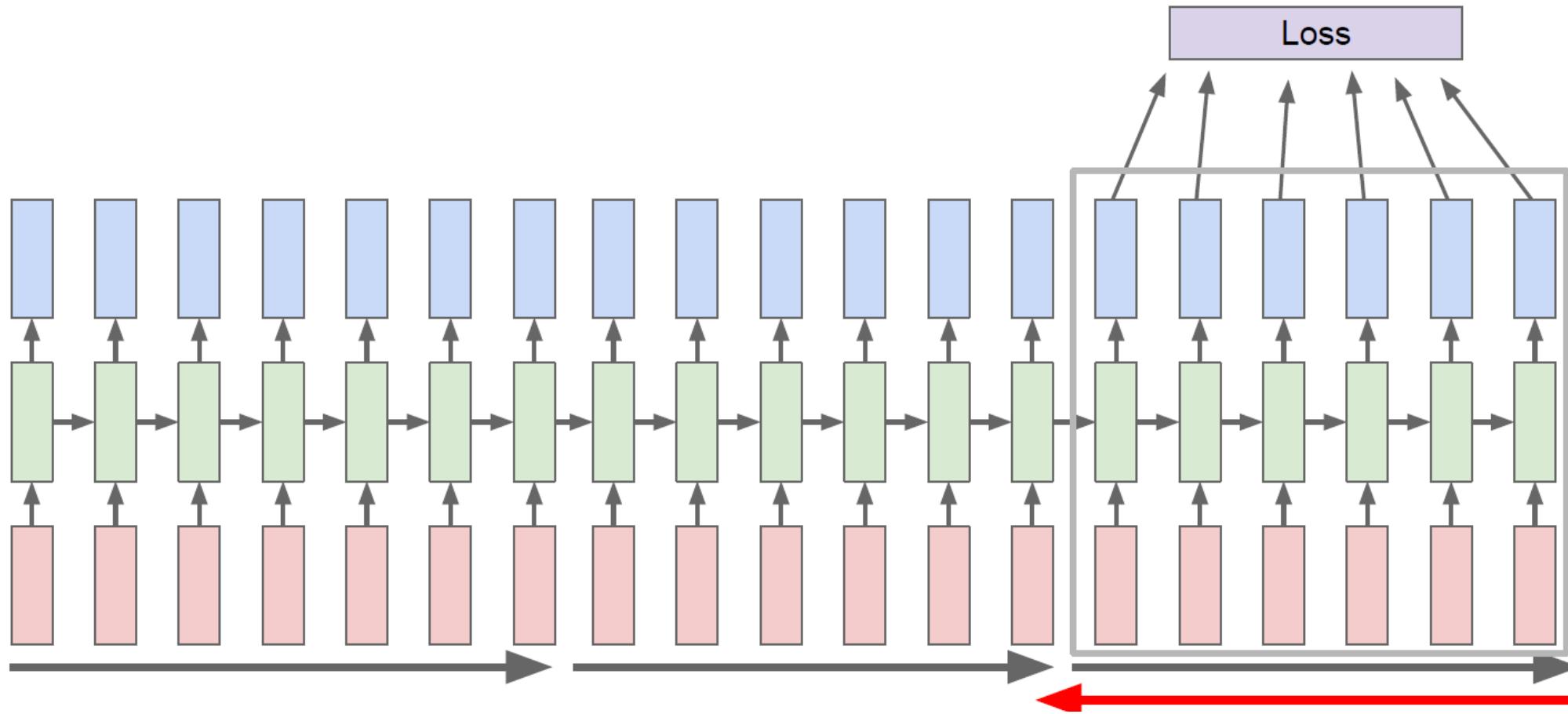
Прямой и обратный проход  
только по кусочку всей  
последовательности

# Truncated Backpropagation по времени



Сохраняем состояния  
предыдущего  
кусочка и делаем  
прямой и обратный  
проход по  
следующему

# Truncated Backpropagation по времени



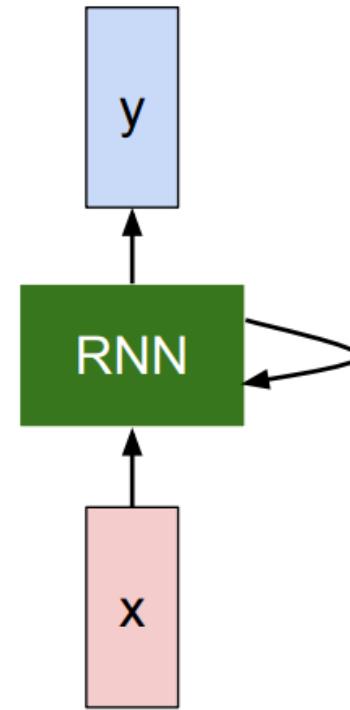
# Примеры: генерация текста

## THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,  
That thereby beauty's rose might never die,  
But as the riper should by time decease,  
His tender heir might bear his memory:  
But thou, contracted to thine own bright eyes,  
Feed'st thy light's flame with self-substantial fuel,  
Making a famine where abundance lies,  
Thyself thy foe, to thy sweet self too cruel:  
Thou that art now the world's fresh ornament,  
And only herald to the gaudy spring,  
Within thine own bud buriest thy content,  
And tender churl mak'st waste in niggarding:  
Pity the world, or else this glutton be,  
To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,  
And dig deep trenches in thy beauty's field,  
Thy youth's proud livery so gazed on now,  
Will be a tatter'd weed of small worth held:  
Then being asked, where all thy beauty lies,  
Where all the treasure of thy lusty days;  
To say, within thine own deep sunken eyes,  
Were an all-eating shame, and thriftless praise.  
How much more praise desp'rd thy beauty's use,  
If thou couldst answer 'This fair child of mine  
Shall sum my count, and make my old excuse,'  
Proving his beauty by succession thine!  
This were to be new made when thou art old,  
And see thy blood warm when thou feel'st it cold.



# Примеры: генерация текста

at first:

tyntd-iafhatawiaoahrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e  
plia tkldrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

↓ train more

"Tmont thithey" fomesscerliund  
Keushey. Thom here  
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome  
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

↓ train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of  
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort  
how, and Gogition is so overelical and ofter.

↓ train more

"Why do what that day," replied Natasha, and wishing to himself the fact the  
princess, Princess Mary was easier, fed in had oftened him.  
Pierre aking his soul came to the packs and drove up his father-in-law women.

# Визуализация скрытого состояния

Text color corresponds to  $\tanh(c)$ , where -1 is red and +1 is blue.

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

Cell that turns on inside quotes:

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.
```

```
Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
    siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!!(current->notifier)(current->notifier_data)) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
            collect_signal(sig, pending, info);
        }
    }
    return sig;
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX
     * defines the longest valid length.
    */
```

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
    struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
        (void *)df->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("audit rule for LSM \\\"%s\\\" is invalid\n",
            df->lsm_str);
        ret = 0;
    }
    return ret;
}
```

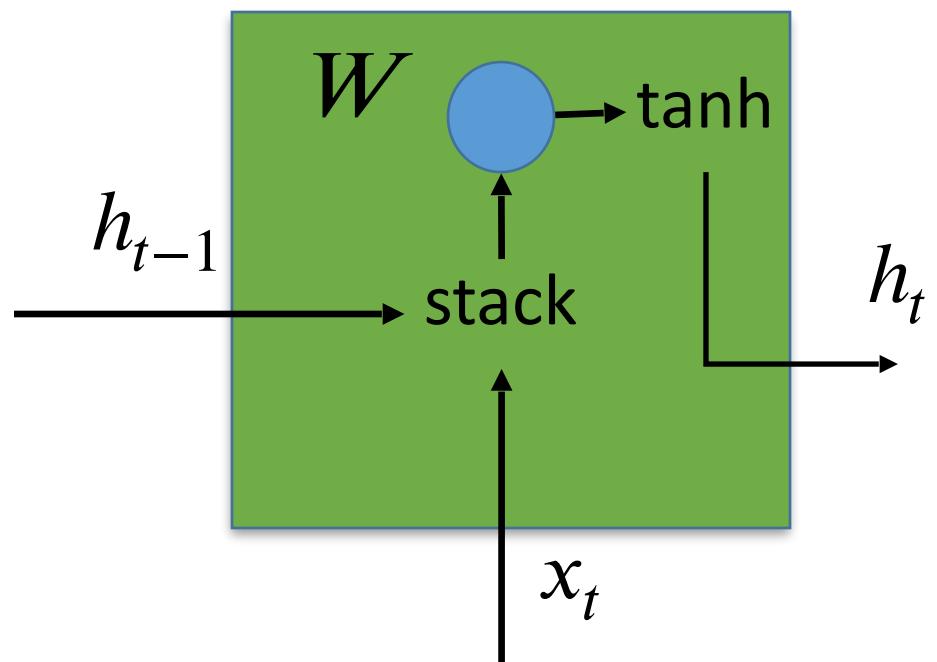
Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
    int i;
    if (classes[class]) {
        for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
            if (mask[i] & classes[class][i])
                return 0;
    }
    return 1;
}
```

Cell that might be helpful in predicting a new line. Note that it only turns on for some ")":

```
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX
     * defines the longest valid length.
    */
    if (len > PATH_MAX)
        return ERR_PTR(-ENAMETOOLONG);
    str = kmalloc(len + 1, GFP_KERNEL);
    if (unlikely(!str))
        return ERR_PTR(-ENOMEM);
    memcpy(str, *bufp, len);
    str[len] = 0;
    *bufp += len;
    *remain -= len;
    return str;
}
```

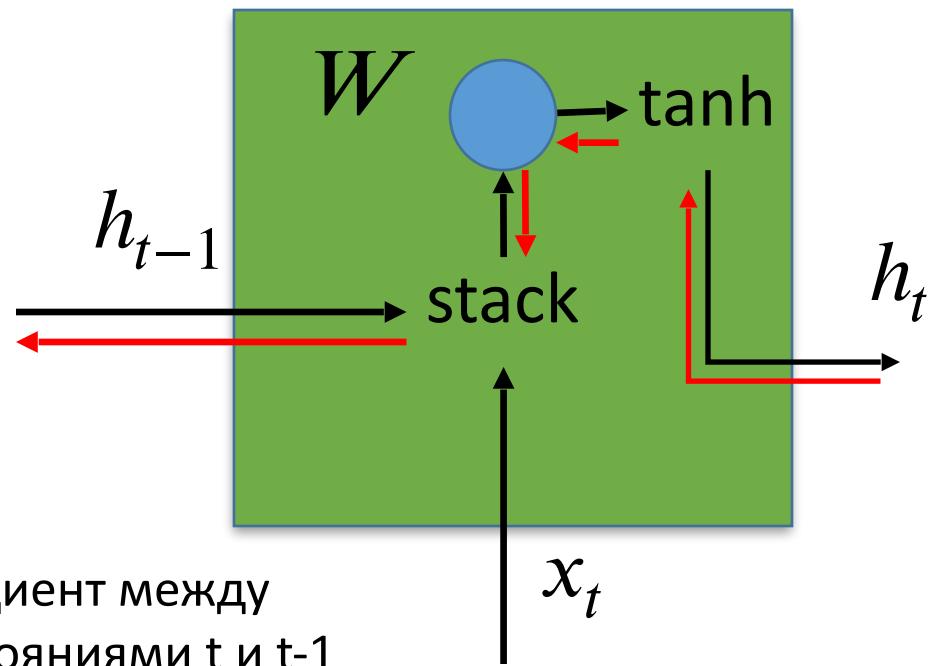
# Градиент через RNN



$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ h_t &= \tanh((W_{hh} \quad W_{xh}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}) \\ &= \tanh( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}) \end{aligned}$$

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994  
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013

# Градиент через RNN

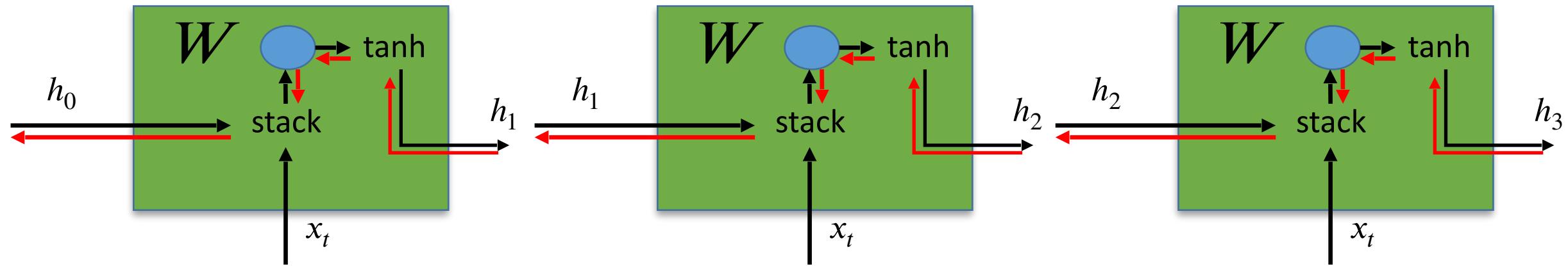


Градиент между  
состояниями  $t$  и  $t-1$   
умножается на матрицу  
весов  $W^T$

$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ &= \tanh((W_{hh} \quad W_{xh}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}) \\ &= \tanh( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}) \end{aligned}$$

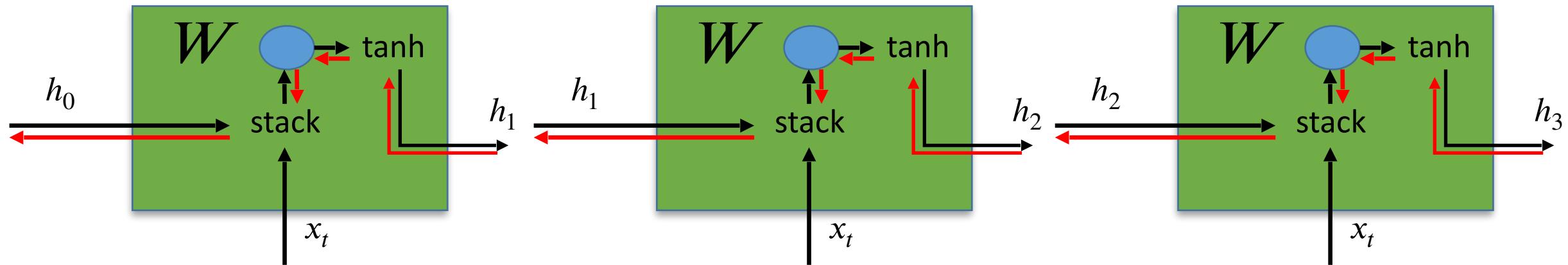
Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994  
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

# Градиент через RNN



Вычисление градиента для состояния 0 будет приводить к многократному умножению на матрицу весов  $W$

# Градиент через RNN

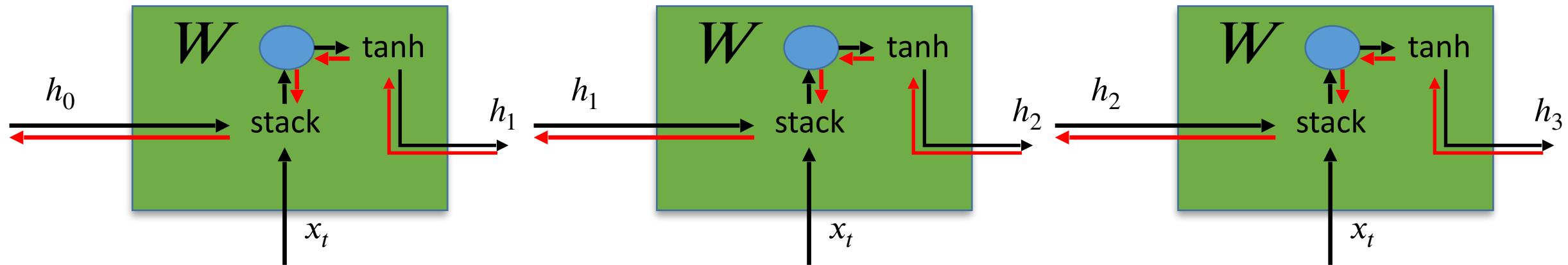


Вычисление градиента для состояния 0 будет приводить к многократному умножению на матрицу весов  $W$

Значения матрицы больше 1:  
**Exploding gradient**

Значения матрицы меньше 1:  
**Vanishing gradient**

# Градиент через RNN



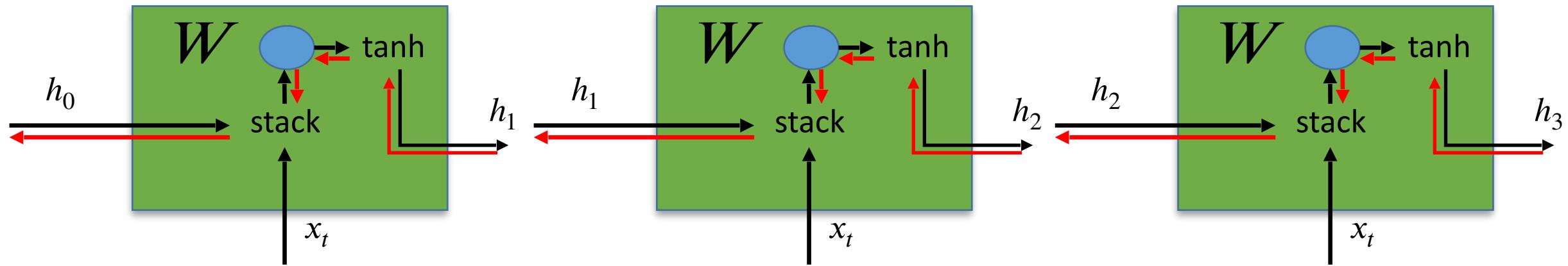
Вычисление градиента для состояния 0 будет приводить к многократному умножению на матрицу весов  $W$

Значения матрицы больше 1:  
**Exploding gradient**



```
torch.nn.utils.clip_grad_norm_(model.parameters(), 1)
```

# Градиент через RNN



Вычисление градиента для состояния 0 будет приводить к многократному умножению на матрицу весов  $W$

Значения матрицы меньше 1:  
**Vanishing gradient**

Нужно менять архитектуру RNN

# Проблемы RNN сетей

Не может видеть далеко назад - при прохождении градиента обратно – происходит экспоненциальное затухание градиента, чем дальше назад – тем больше затухает.

Решение - **LSTM**

# Long Short Term Memory (LSTM)

Vanilla RNN

$$h_t = \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

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

Hochreiter and Schmidhuber, “Long Short Term Memory”,  
Neural Computation  
1997

# Long Short Term Memory (LSTM)

$$\begin{aligned}\mathbf{c}'_t &= \tanh(W_{xc} \mathbf{x}_t + W_{hc} \mathbf{h}_{t-1} + \mathbf{b}_{c'}) && \textit{candidate cell state} \\ \mathbf{i}_t &= \sigma(W_{xi} \mathbf{x}_t + W_{hi} \mathbf{h}_{t-1} + \mathbf{b}_i) && \textit{input gate} \\ \mathbf{f}_t &= \sigma(W_{xf} \mathbf{x}_t + W_{hf} \mathbf{h}_{t-1} + \mathbf{b}_f) && \textit{forget gate} \\ \mathbf{o}_t &= \sigma(W_{xo} \mathbf{x}_t + W_{ho} \mathbf{h}_{t-1} + \mathbf{b}_o) && \textit{output gate} \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{c}'_t, && \textit{cell state} \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) && \textit{block output}\end{aligned}$$

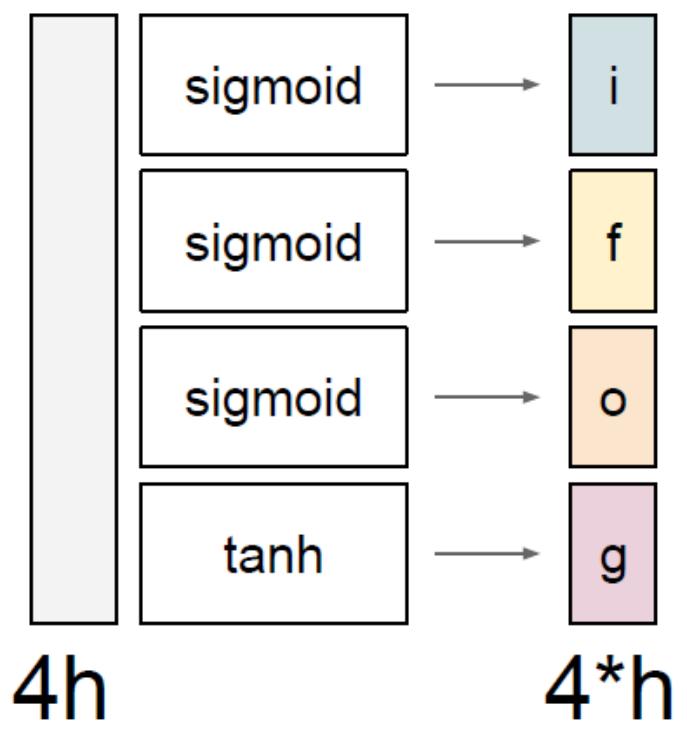
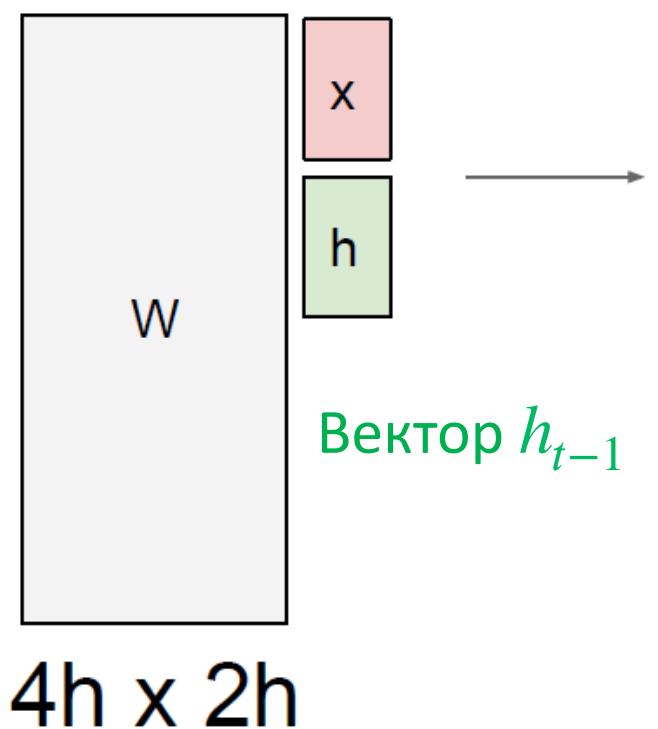
$$y_t = W_{hy} h_t$$

\* Вместо  $\mathbf{c}'_t$  - могут использовать  $g_t$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

# Long Short Term Memory (LSTM)

Вектор  $x$



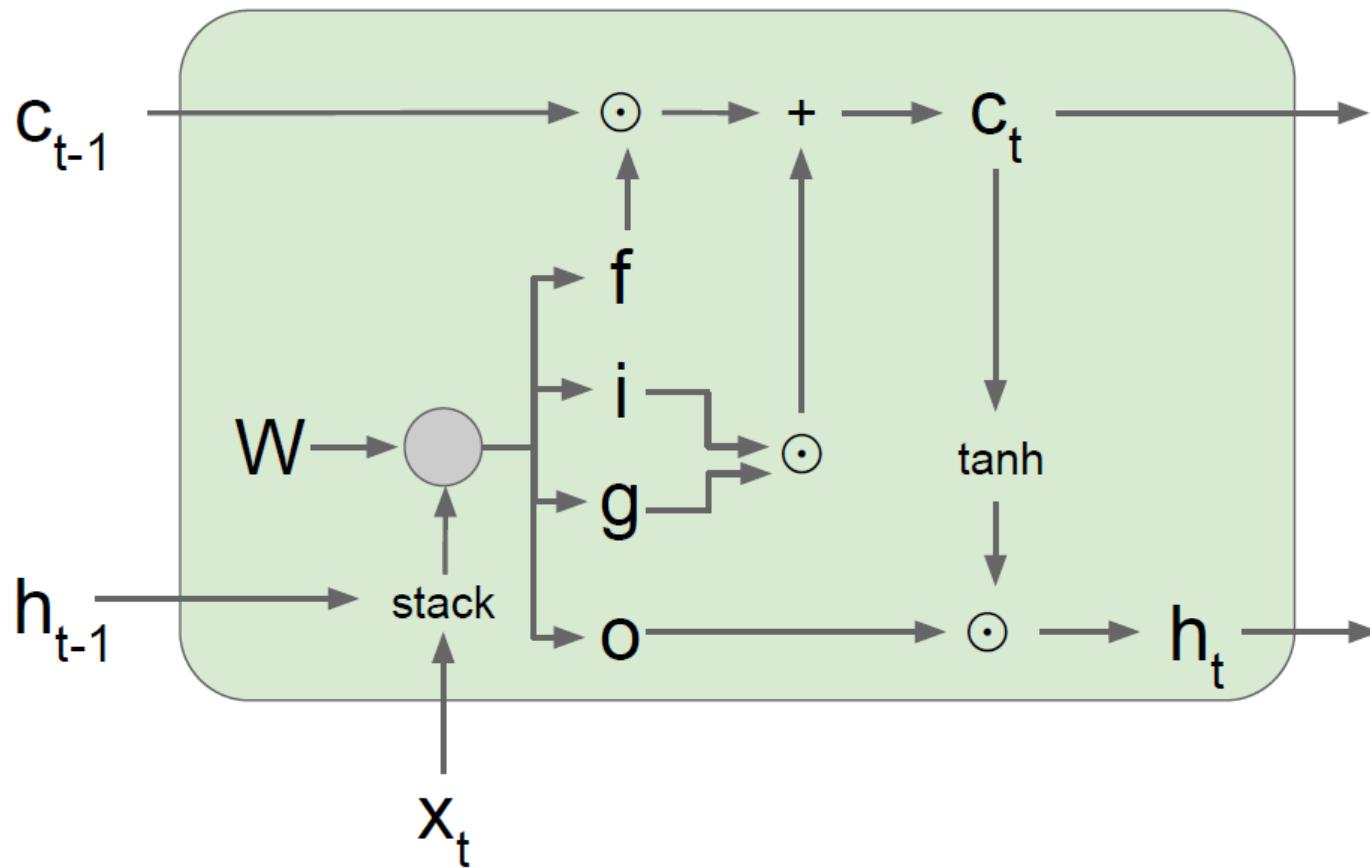
- $f$ : Forget gate, Забывать ли состояние элемента
- $i$ : Input gate, Писать ли состояние элемента
- $g$ : Gate gate, Сколько писать в элемент
- $o$ : Output gate, Сколько выводить из элемента

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

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

# Long Short Term Memory (LSTM)



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

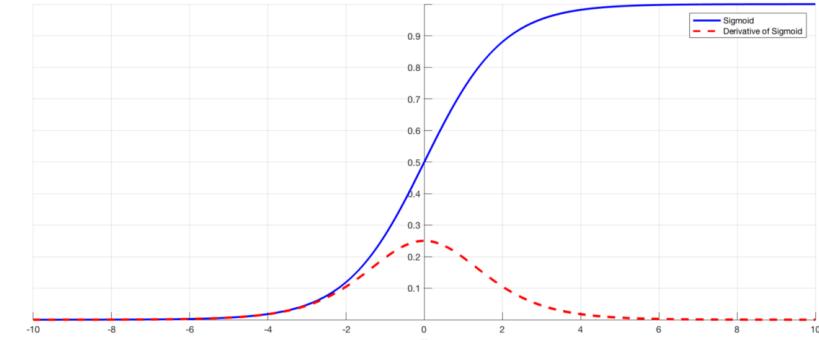
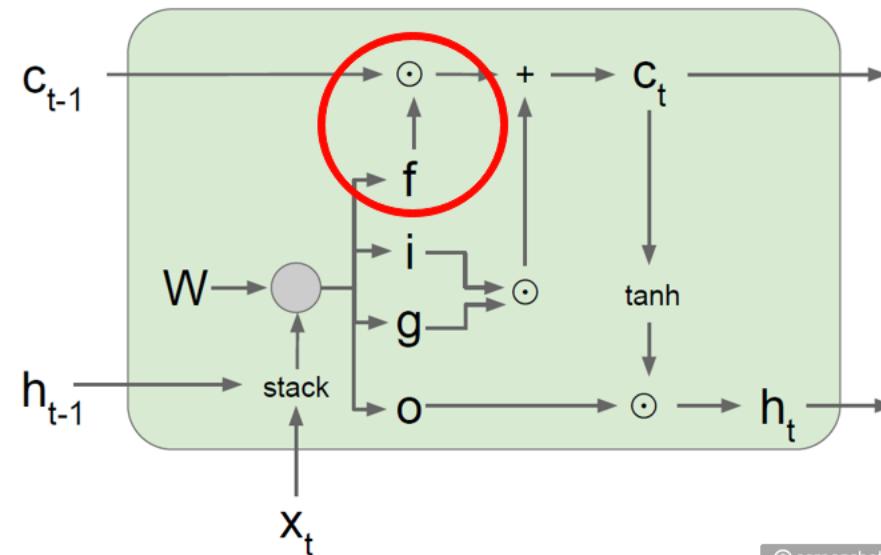
# LSTM. Gradient Flow

- Что будет если  $f_t=1$  всегда?
- Как надо инициализировать bias forget gate?

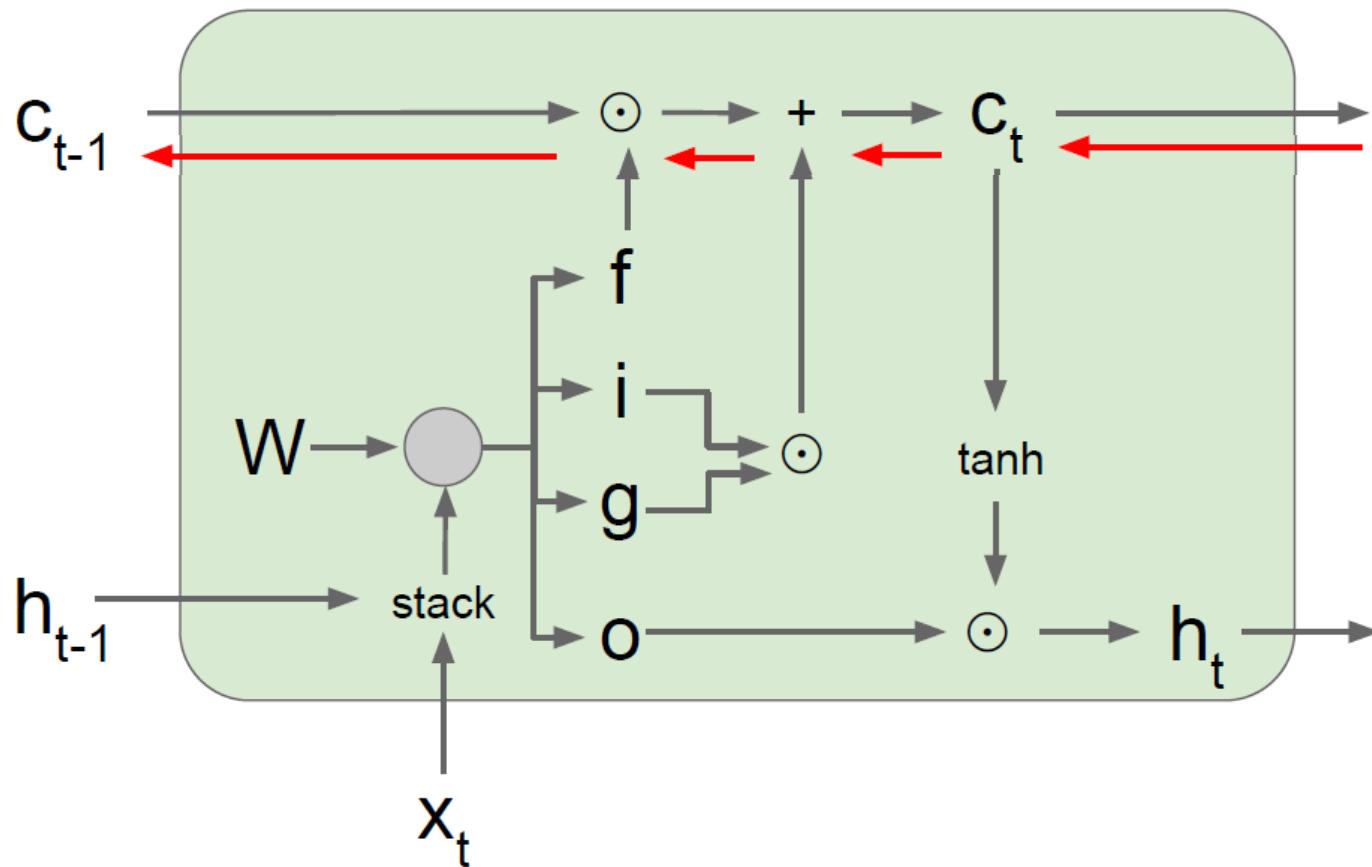
$$c_t = c_{t-1} + i_t \odot c'_t.$$

$$\frac{\partial c_t}{\partial c_{t-1}} = 1.$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$



# LSTM. Gradient Flow



Обратный проход от  $c_t$  до  
 $c_{t-1}$  перемножаются  
только с  $f$  без матрицы  $W$

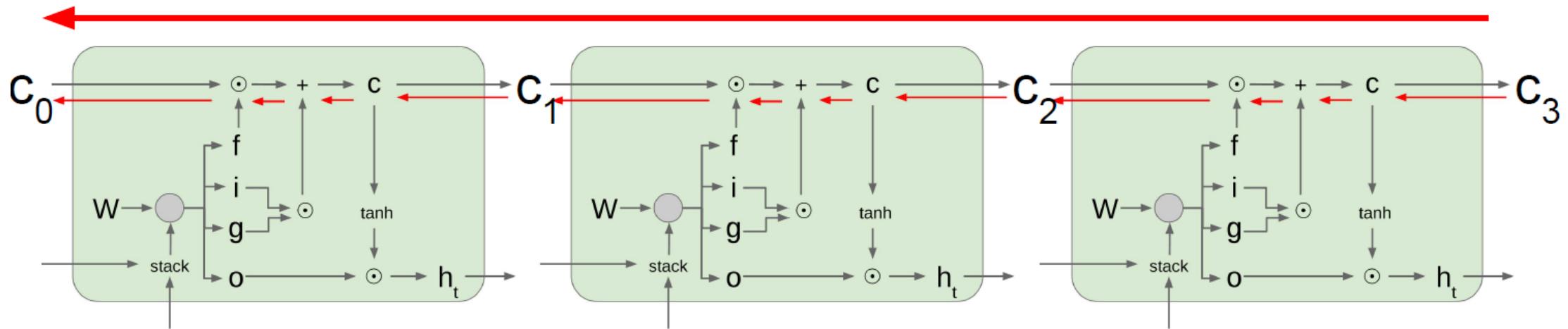
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

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

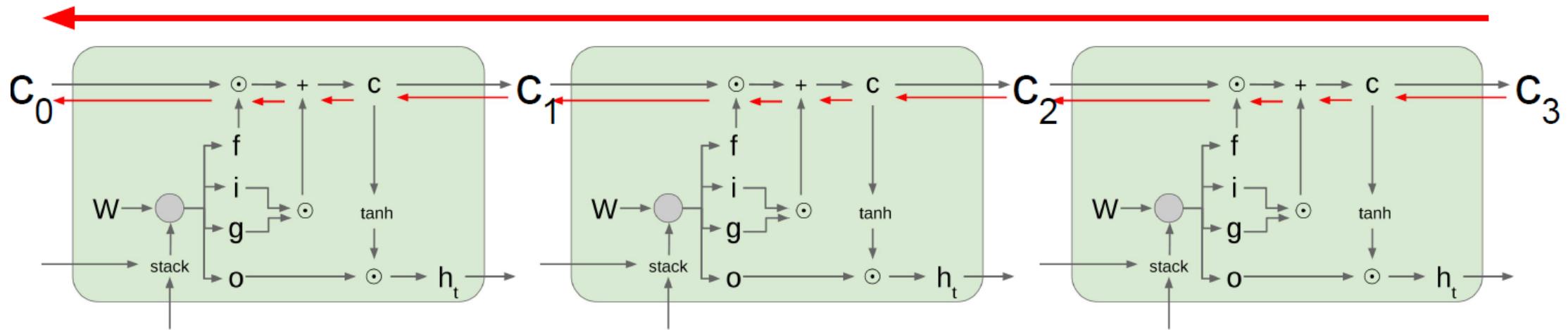
# LSTM. Gradient Flow

Незатухающий градиент

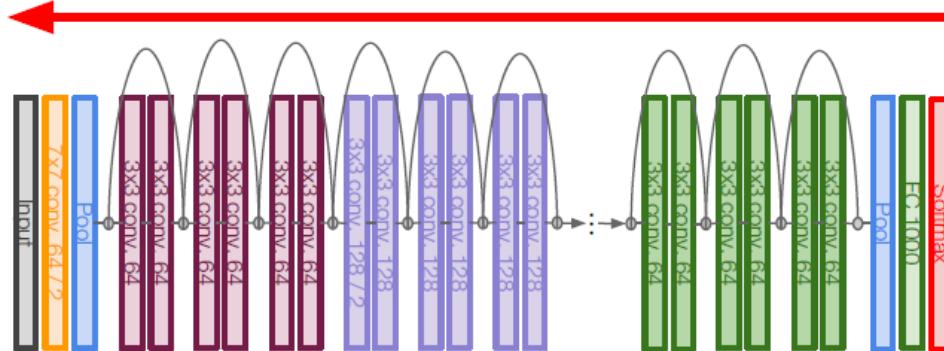


# LSTM. Gradient Flow

# Незатухающий градиент



## ResNet



# LSTM - peepholes («замочные скважины»)

$$\begin{aligned} c'_t &= \tanh(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_{c'}) && \text{candidate cell state} \\ i_t &= \sigma(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) && \text{input gate} \\ f_t &= \sigma(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) && \text{forget gate} \\ o_t &= \sigma(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) && \text{output gate} \\ c_t &= f_t \odot c_{t-1} + i_t \odot c'_t, && \text{cell state} \\ h_t &= o_t \odot \tanh(c_t) && \text{block output} \end{aligned}$$

- Гейты не зависят от  $c_t$  поэтому если  $o_t \Rightarrow 0$  то их поведение не зависит от состояния памяти.
- Но мы хотим все равно управлять памятью, даже в этом случае, добавляя peepholes

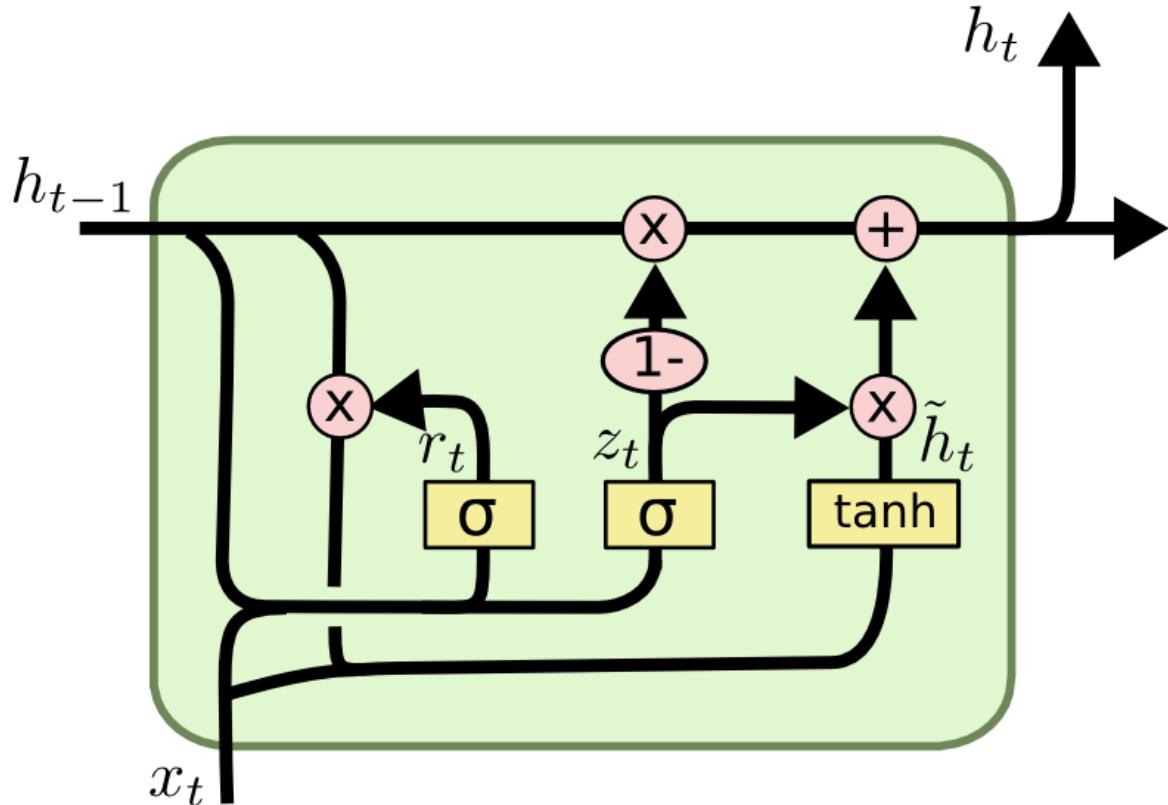
$$\begin{aligned} i_t &= \sigma(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + W_{pi}c_{t-1} + \mathbf{b}_i) \\ f_t &= \sigma(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + W_{pf}c_{t-1} + \mathbf{b}_f) \\ o_t &= \sigma(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + W_{po}c_{t-1} + \mathbf{b}_o) \end{aligned}$$

# GRU – gated recurrent unit

- Совмещаем выходные и забывающий гейты – их заменяет `update_gate (u)`
- Совмещаем `hidden_state(h)` и `cell_state(c)` – объединяем их
- Добавляем `reset_gate`

# GRU RNN (разобрать самим)

**GRU** [*Learning phrase representations using rnn encoder-decoder for statistical machine translation*, Cho et al. 2014]



$$\mathbf{u}_t = \sigma(W_{xu}x_t + W_{hu}h_{t-1} + b_u),$$

$$\mathbf{r}_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r),$$

$$\mathbf{h}'_t = \tanh(W_{xh'}x_t + W_{hh'}(\mathbf{r}_t \odot h_{t-1})),$$

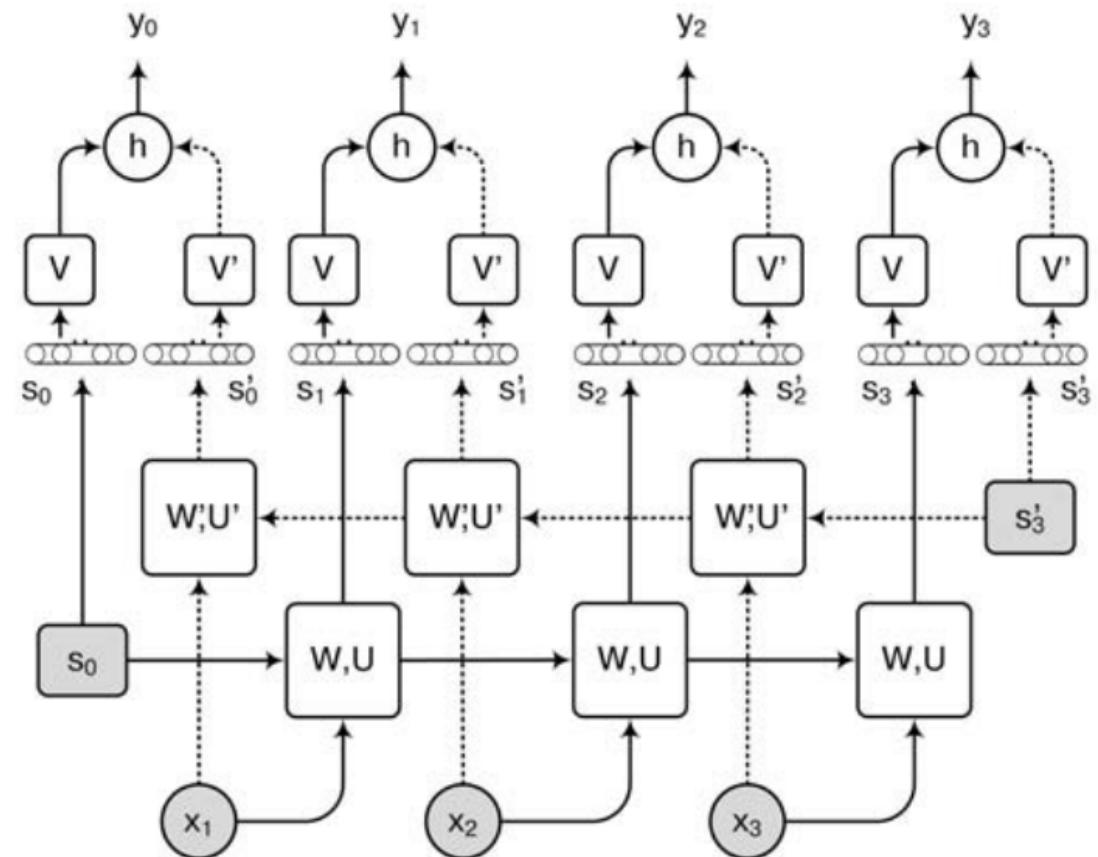
$$\mathbf{h}_t = (1 - \mathbf{u}_t) \odot \mathbf{h}'_t + \mathbf{u}_t \odot h_{t-1}.$$

# Как улучшить RNN?

Можно использовать будущий контекст  
(future context)

Используем двунаправленную RNN.

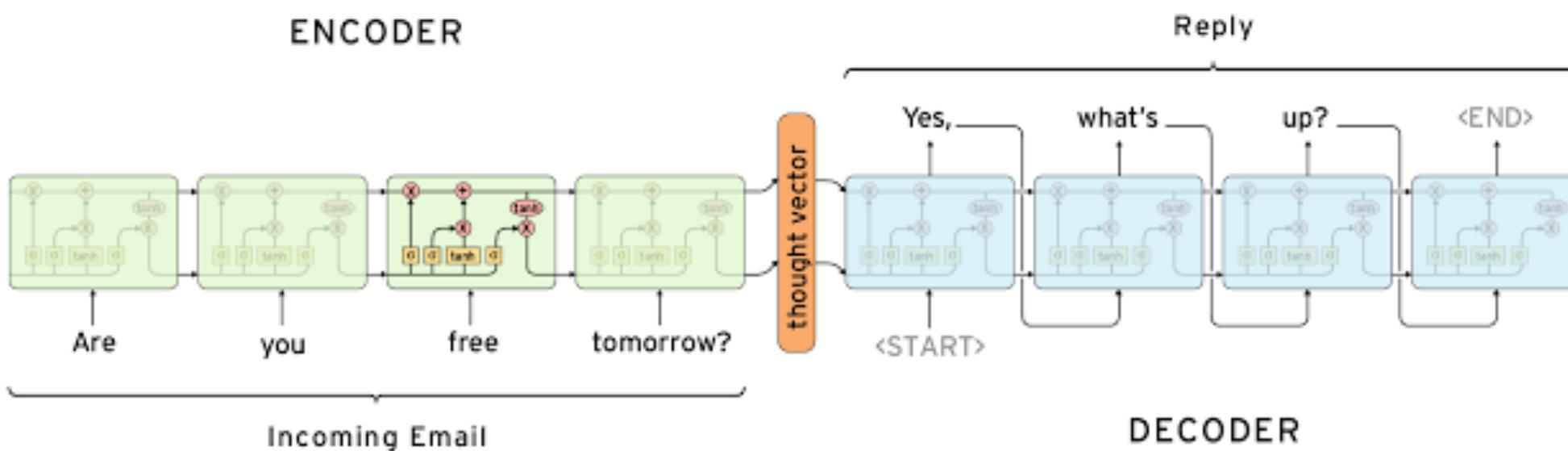
Считаем два скрытых состояния  
слева-направо и справа-налево



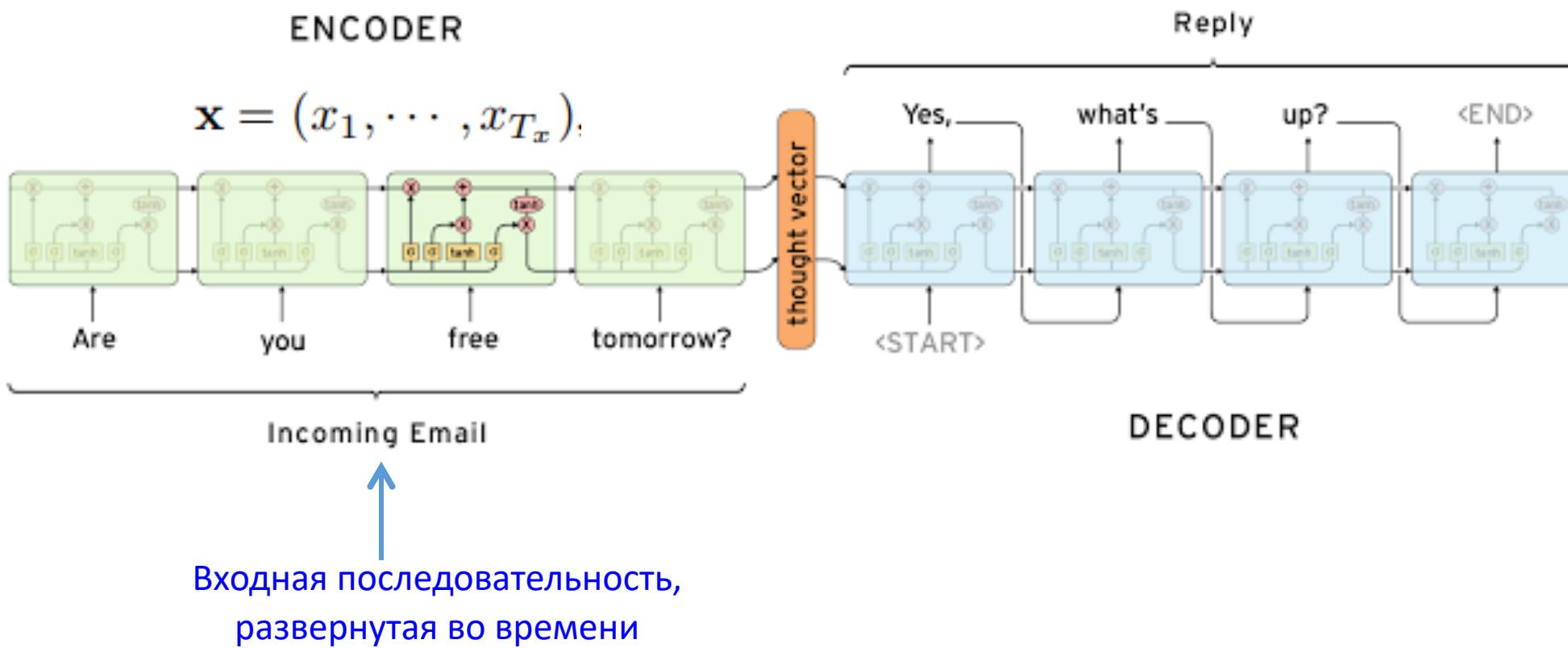
$$\begin{aligned} s_t &= \sigma(\mathbf{b} + Ws_{t-1} + Ux_t), & s'_t &= \sigma(\mathbf{b}' + W's'_{t+1} + U'x_t), \\ o_t &= \mathbf{c} + Vs_t + V's'_t, & y_t &= h(o_t). \end{aligned}$$

## Механизмы внимания

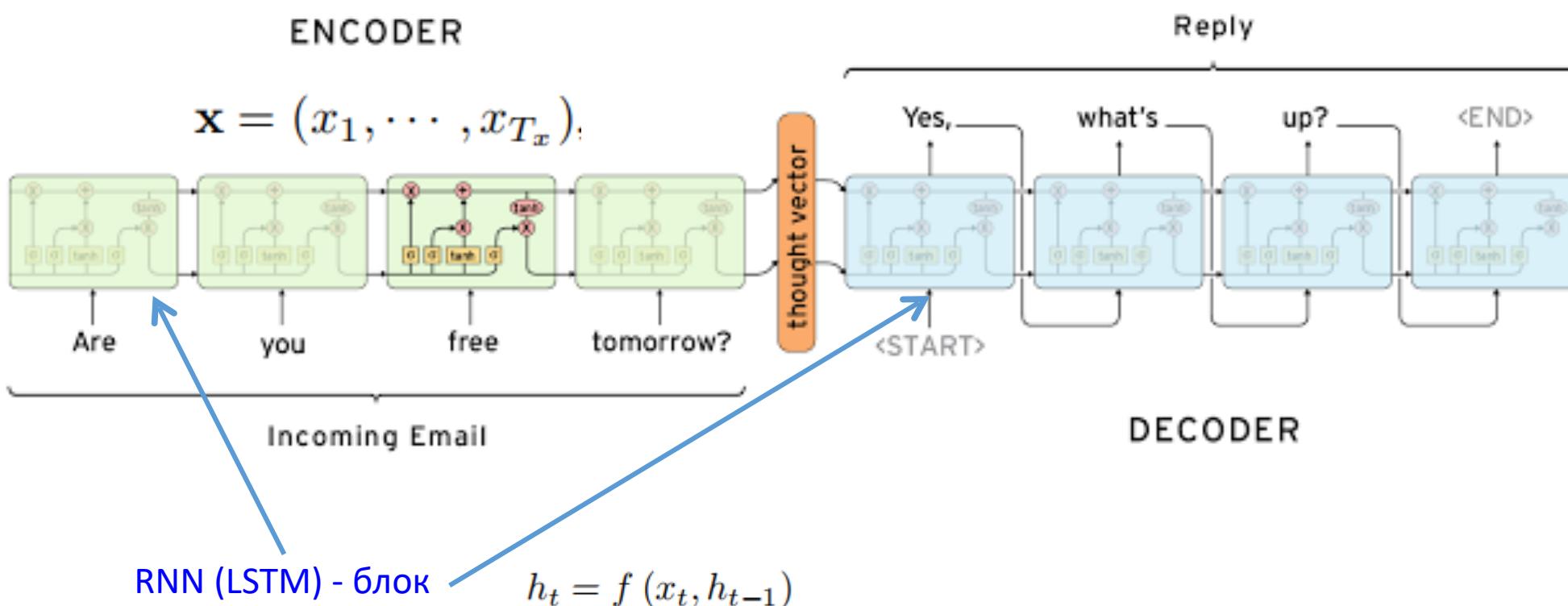
# Чат бот



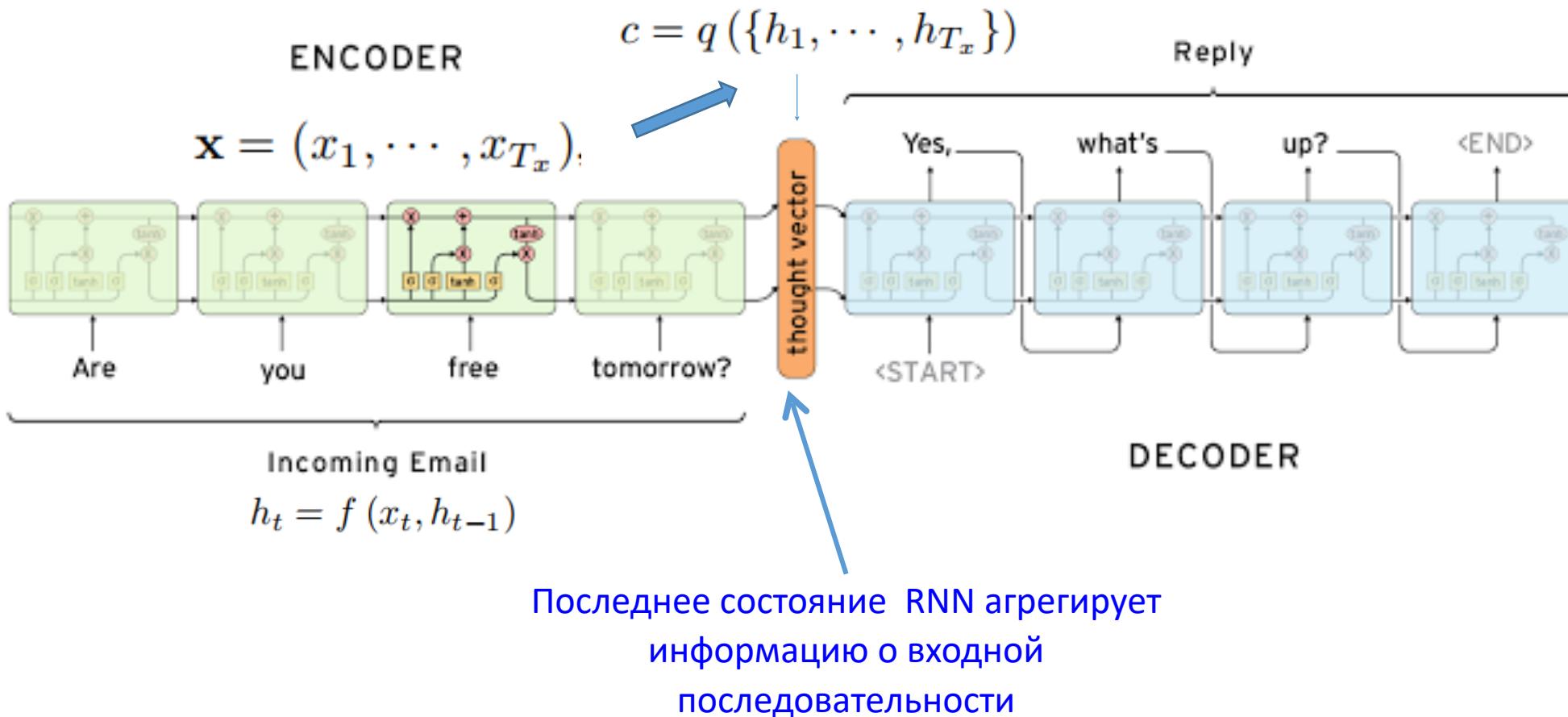
# Чат бот



# Чат бот

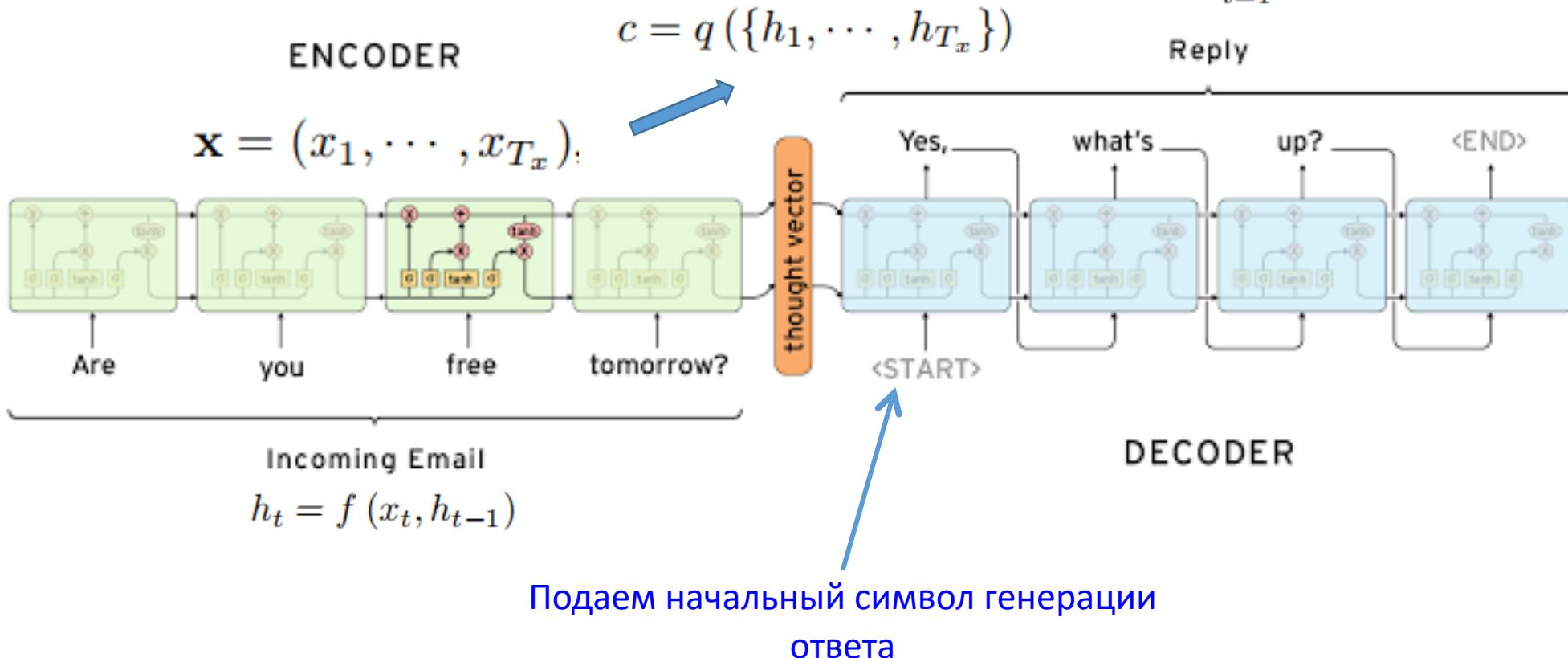


# Чат бот

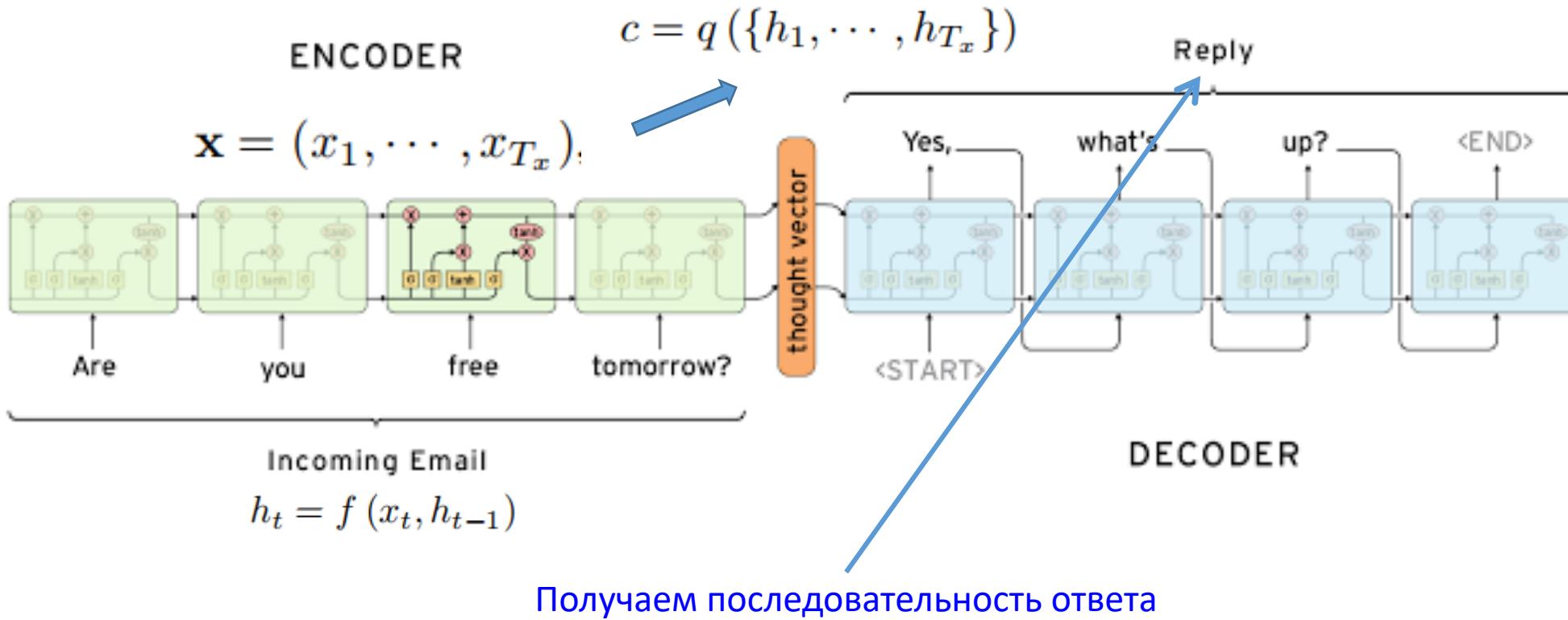


# Чат бот

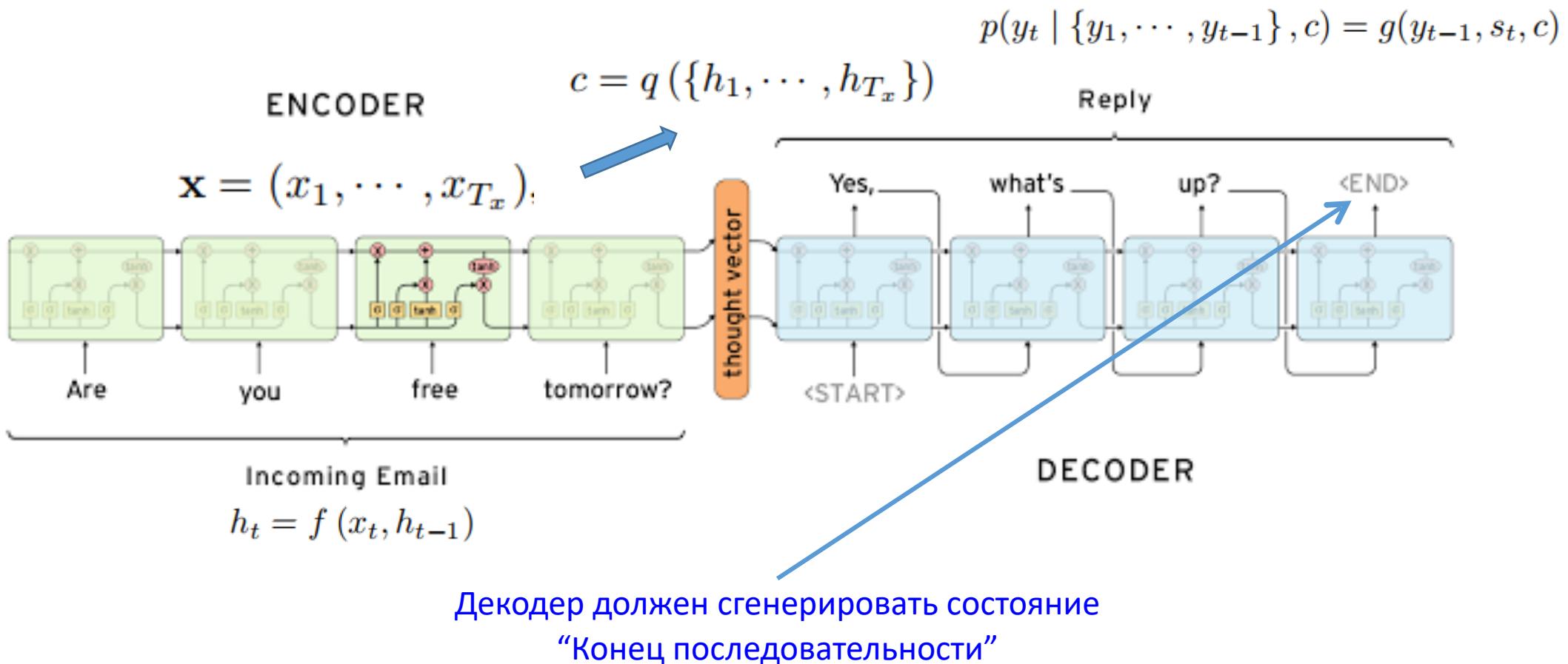
$$p(\mathbf{y}) = \prod_{t=1}^T p(y_t | \{y_1, \dots, y_{t-1}\}, c).$$



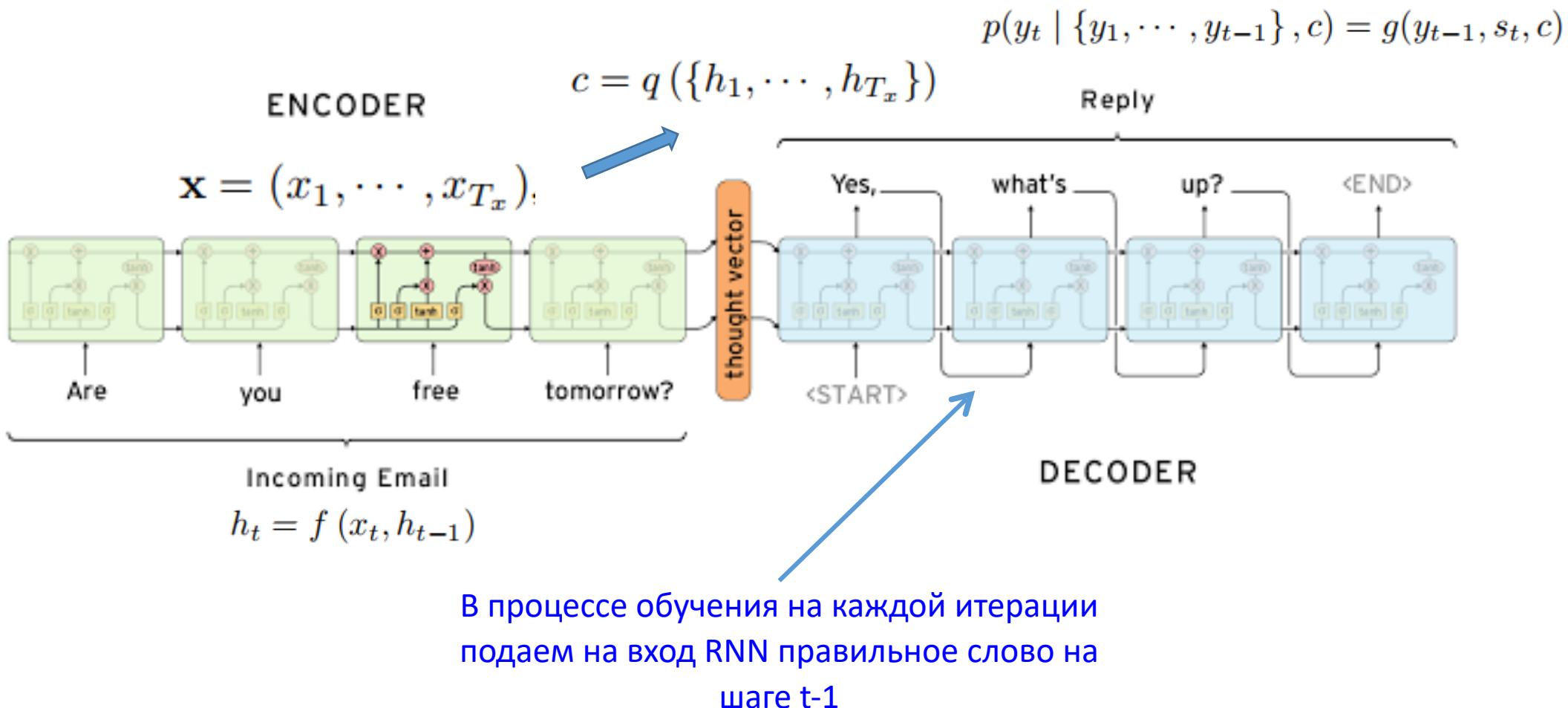
# Чат бот



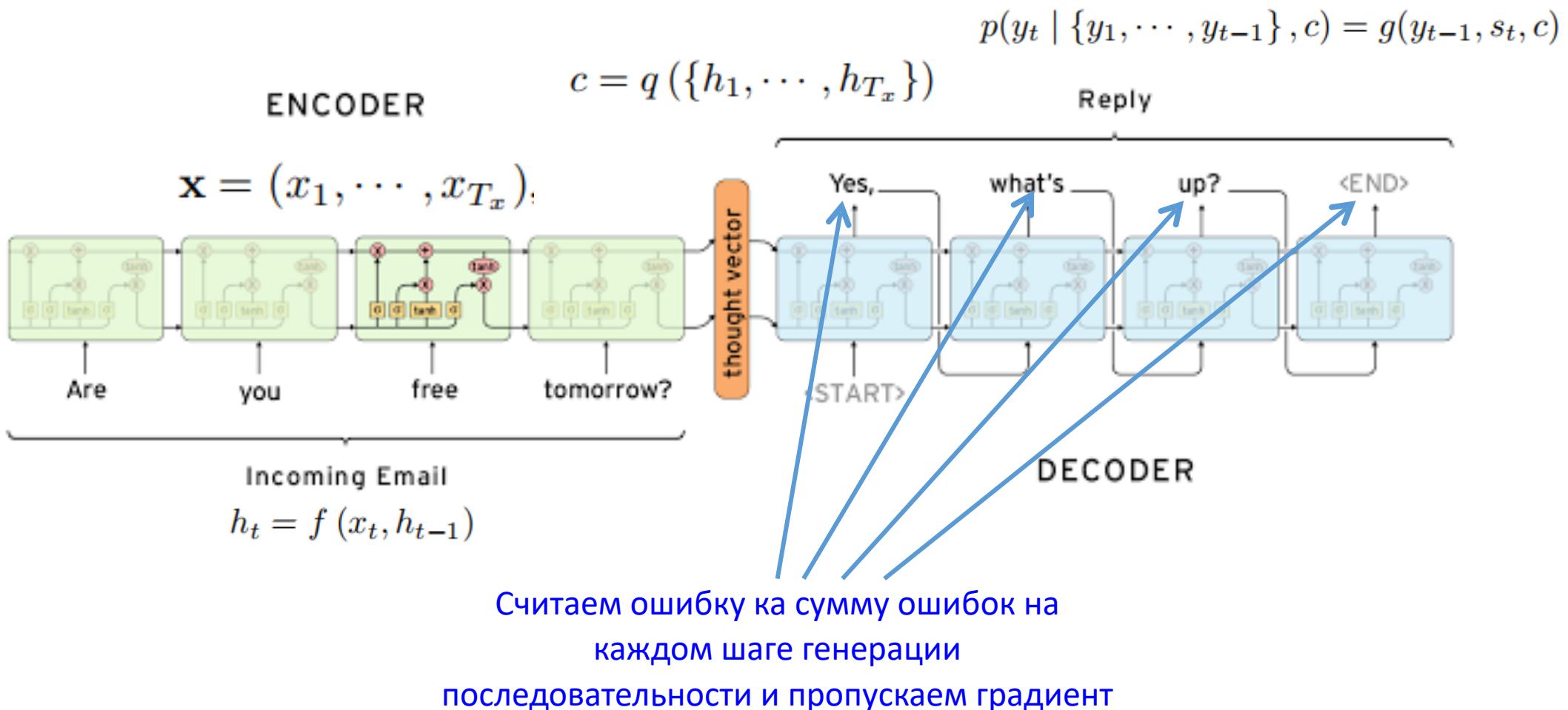
# Чат бот



# Чат бот



# Чат бот



# Чат бот

- Проблема: При генерации ответа нужна не только скрытая информация, но и контекстная информация запроса.
- Выход: при генерации ответа смотреть на слова, которые необходимы для создания ответа.
  - Куда мы пойдем завтра -> нужно смотреть на слово “завтра”
  - .....
- Можно заставить сеть выучить вероятностное распределение над входной последовательностью и использовать скрытые состояния энкодера через взвешенную сумму

# Механизм внимания (Attention)

- Расширенная модель внимания представляет декодер как:

$$p(y_i | y_1, \dots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

где  $s_i$  - это скрытое состояние декодера RNN  $s_i = f(s_{i-1}, y_{i-1}, c_i)$

- В отличие от модели энкодер-декодер вероятность обуславливается контекстным вектором  $c_i$  для каждого целевого слова  $y_{i-1}$
- Контекстный вектор  $c_i$  зависит от аннотаций  $(h_1 \dots h_T)$  на которые энкодер мапирует входную последовательность

# Механизм внимания (Attention)

- Каждая аннотация  $h_i$  содержит информацию о всей последовательности и фокус на часть слов, окружающих  $i$ -е слово
- Контекст-вектор считается как взвешенная сумма аннотаций

$$c_i = \sum_{j=0}^T a_{ij} h_j$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=0}^T \exp(e_{ik})}$$

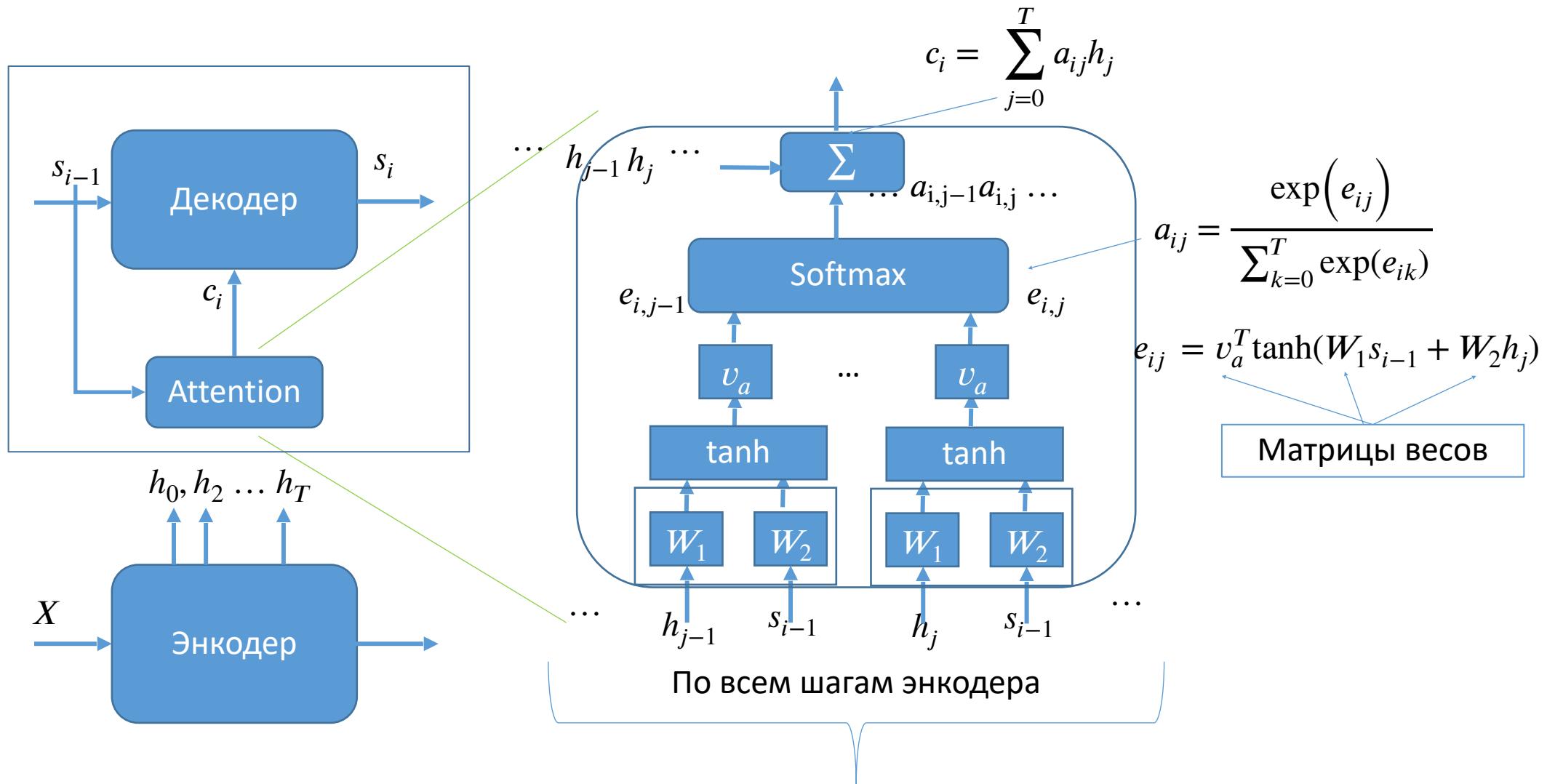
- вес

$$e_{ij} = a(s_{i-1}, h_j)$$

- модель выравнивания

- Выравнивание показывает на сколько подходит вход с позиции  $j$  к выходу позиции  $i$ .

# Механизм внимания (Attention)



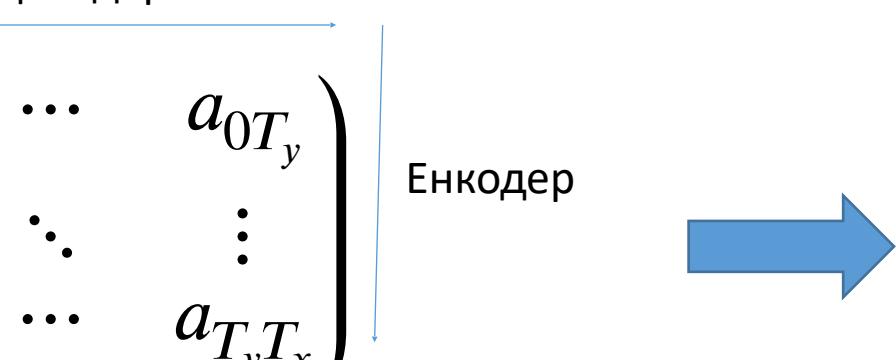
# Механизм внимания (Attention)

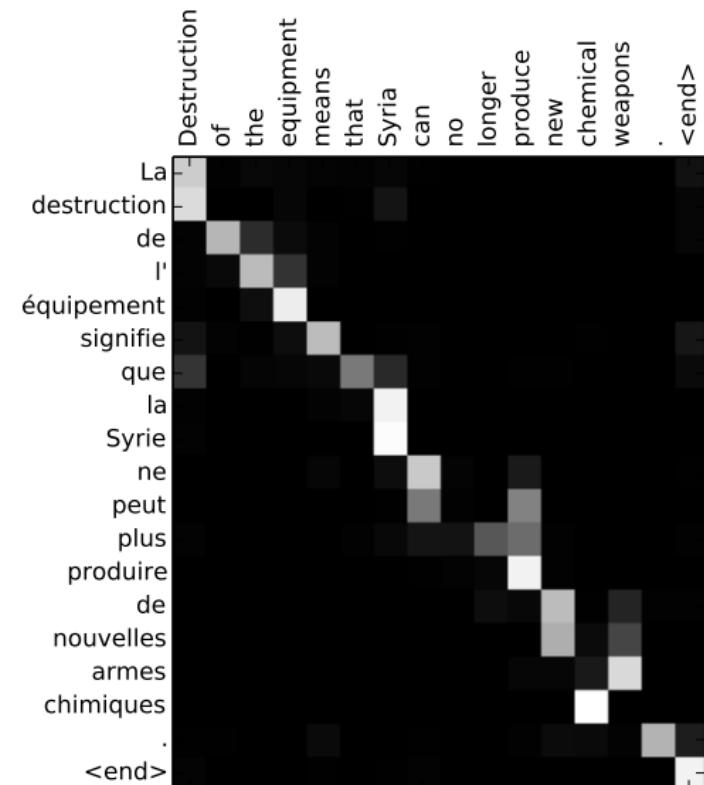
- Веса а от softmax можно объединить в матрицу А

$$A = \begin{pmatrix} a_{00} & \cdots & a_{0T_y} \\ \vdots & \ddots & \vdots \\ a_{0T_x} & \cdots & a_{T_y T_x} \end{pmatrix}$$

Декодер

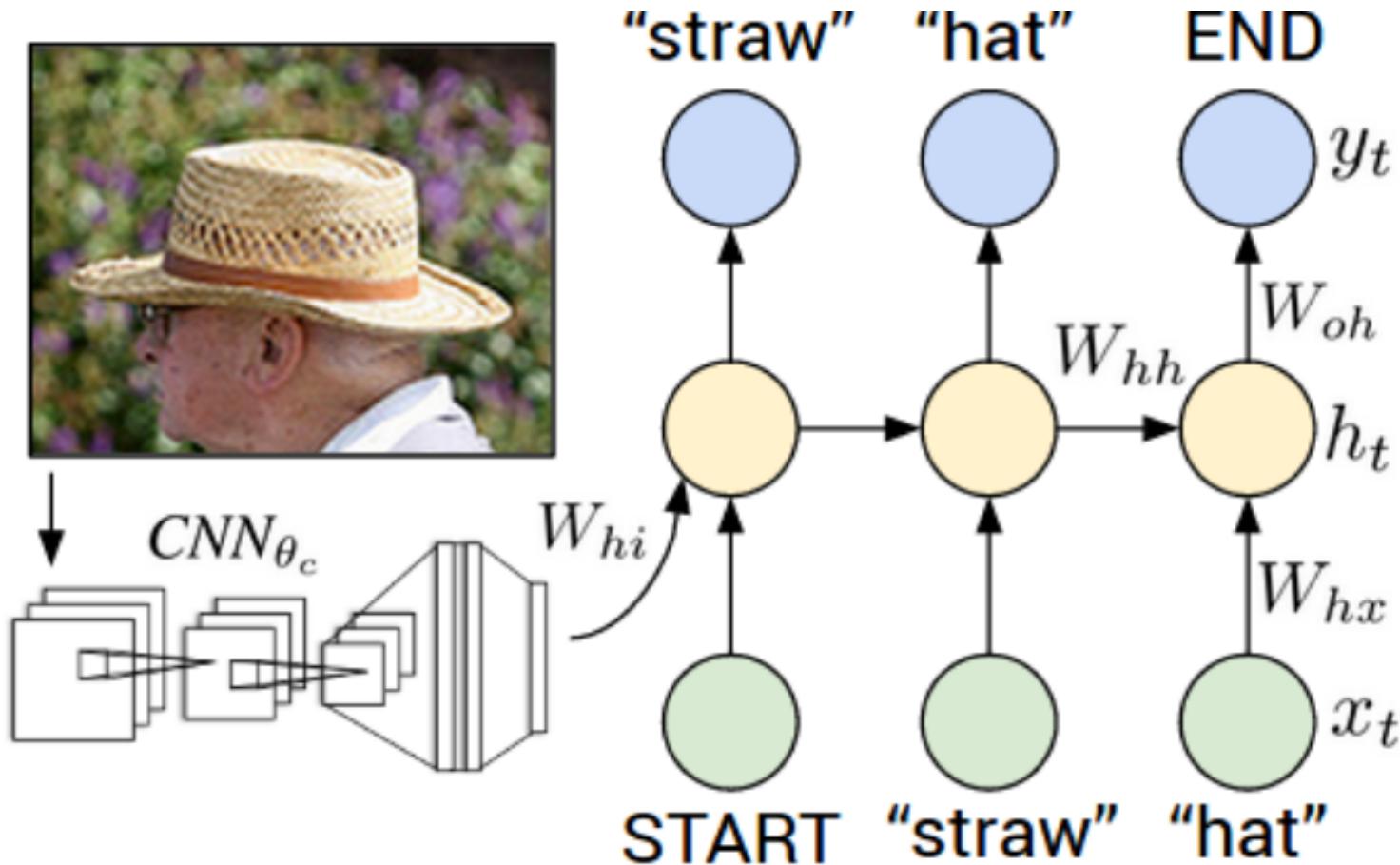
Енкодер



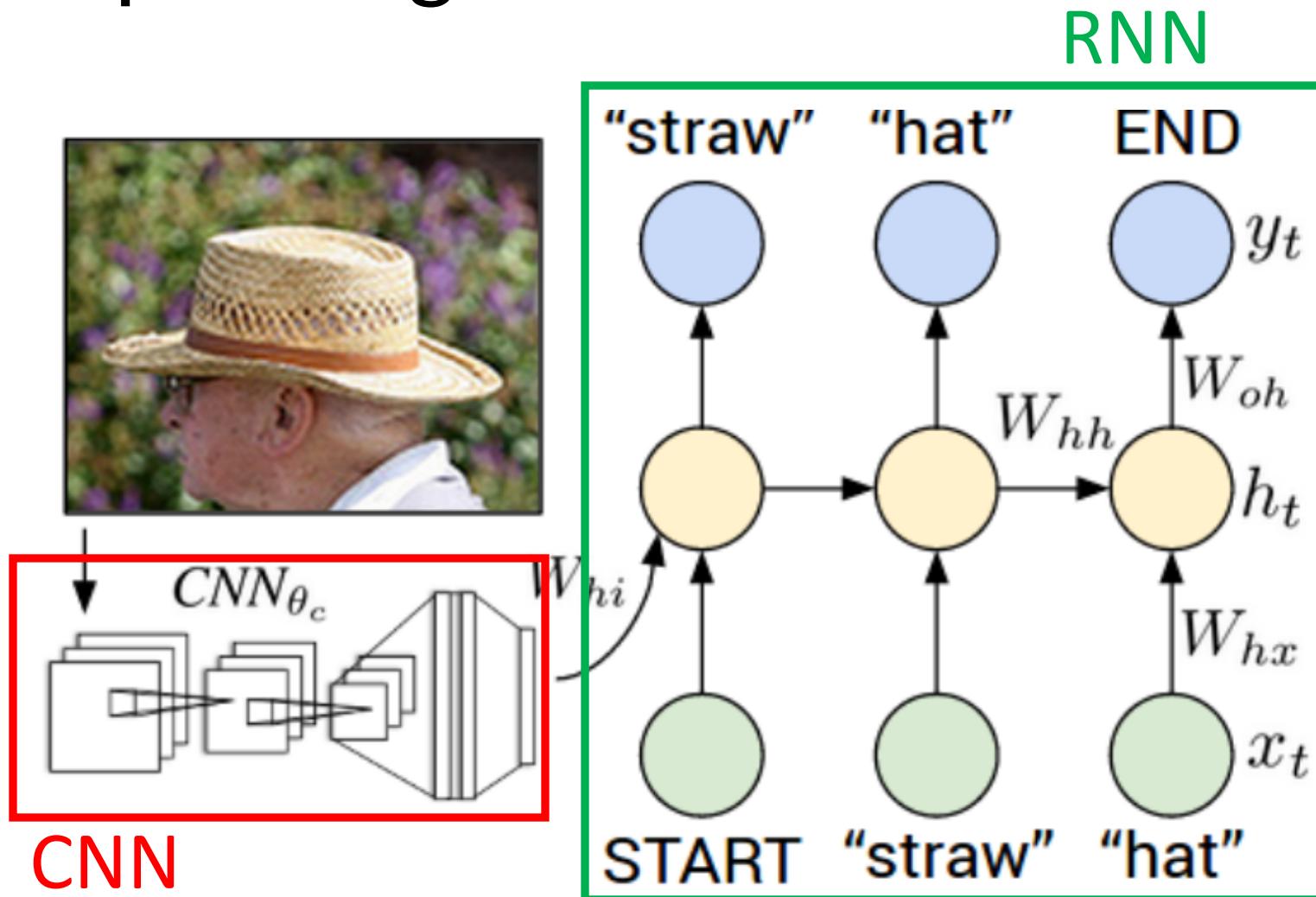


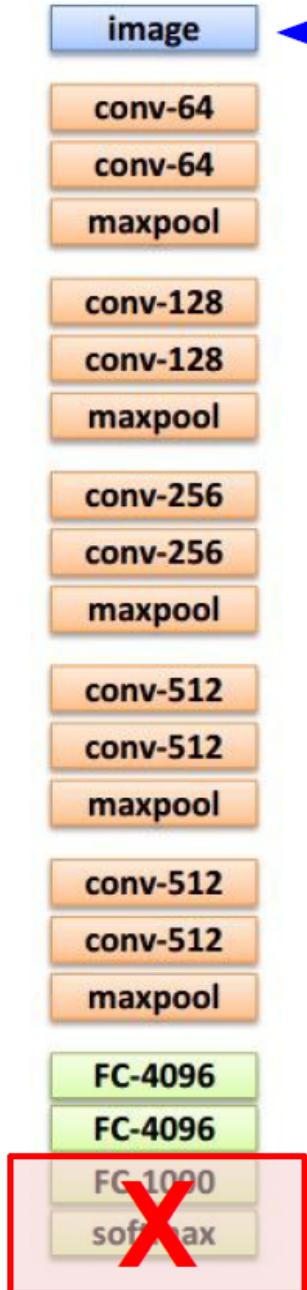
# Немного о применении RNN сетей

# Image Captioning



# Image Captioning

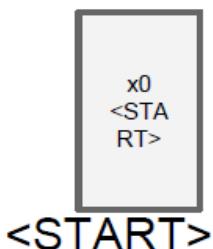




test image



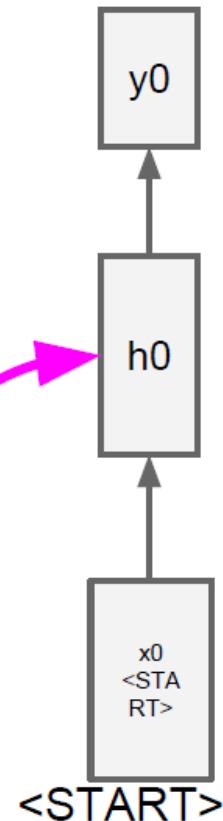
test image





v

Wih



test image

Рекуррентная функция

$$h = \tanh(W_{xh} * x + W_{hh} * h + W_{ih} * v)$$



test image

conv-64  
conv-64  
maxpool

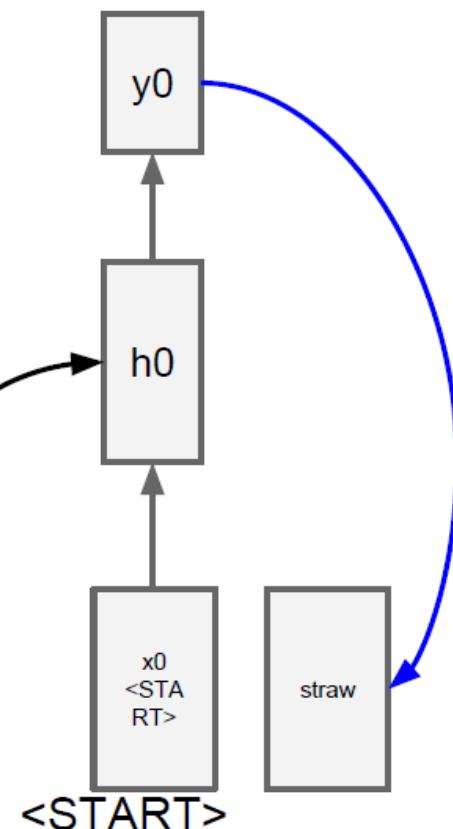
conv-128  
conv-128  
maxpool

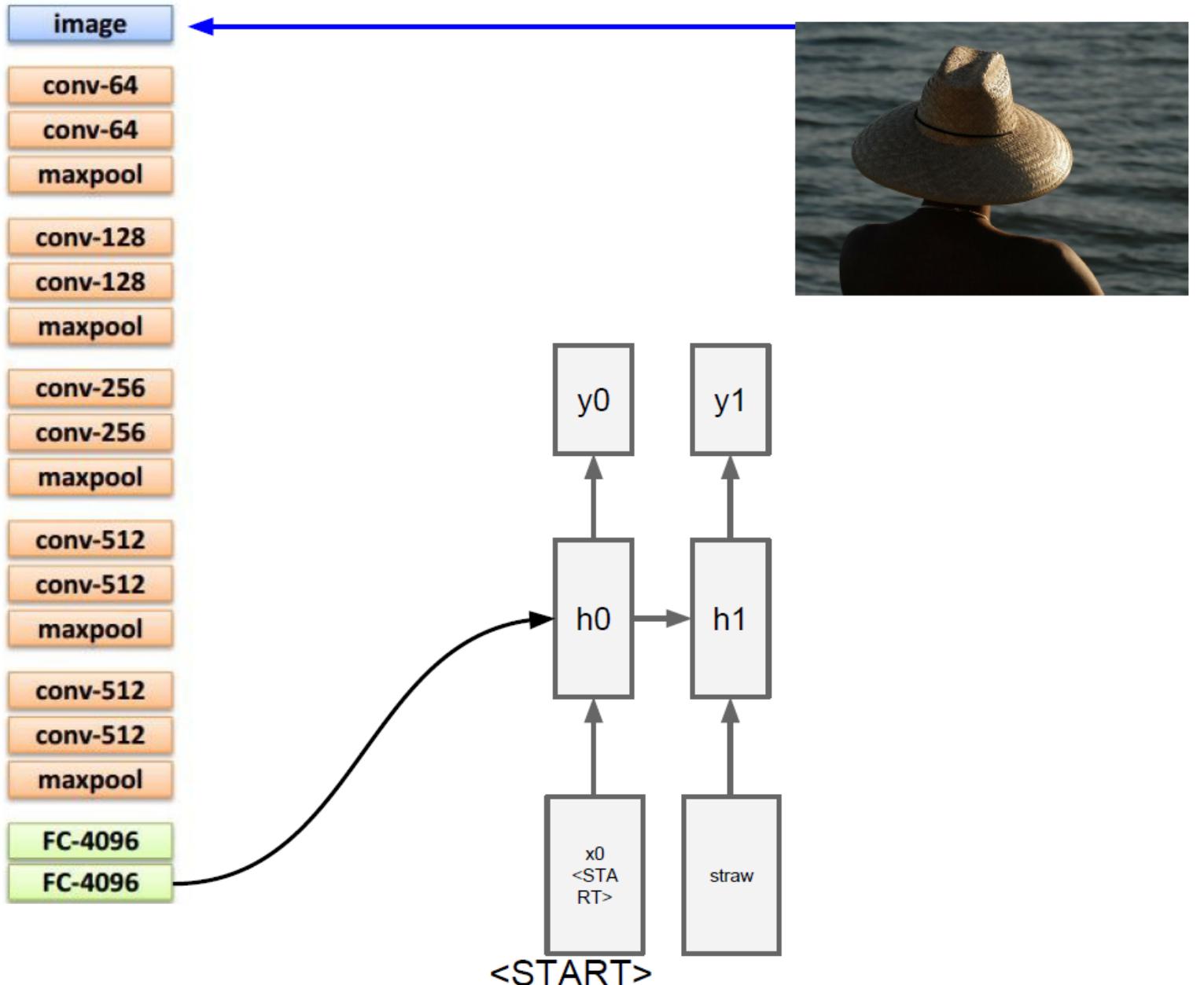
conv-256  
conv-256  
maxpool

conv-512  
conv-512  
maxpool

conv-512  
conv-512  
maxpool

FC-4096  
FC-4096





test image



image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

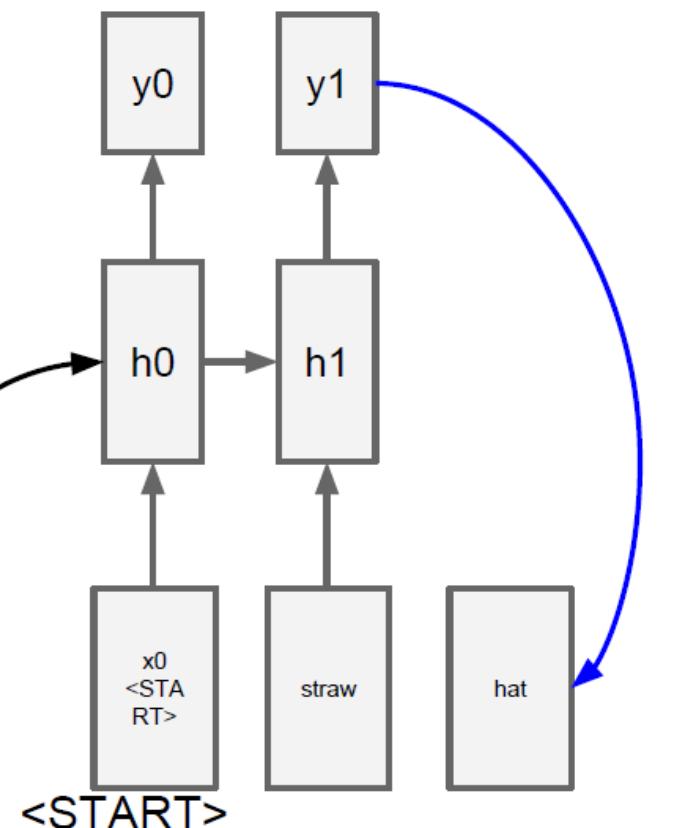
conv-512

conv-512

maxpool

FC-4096

FC-4096

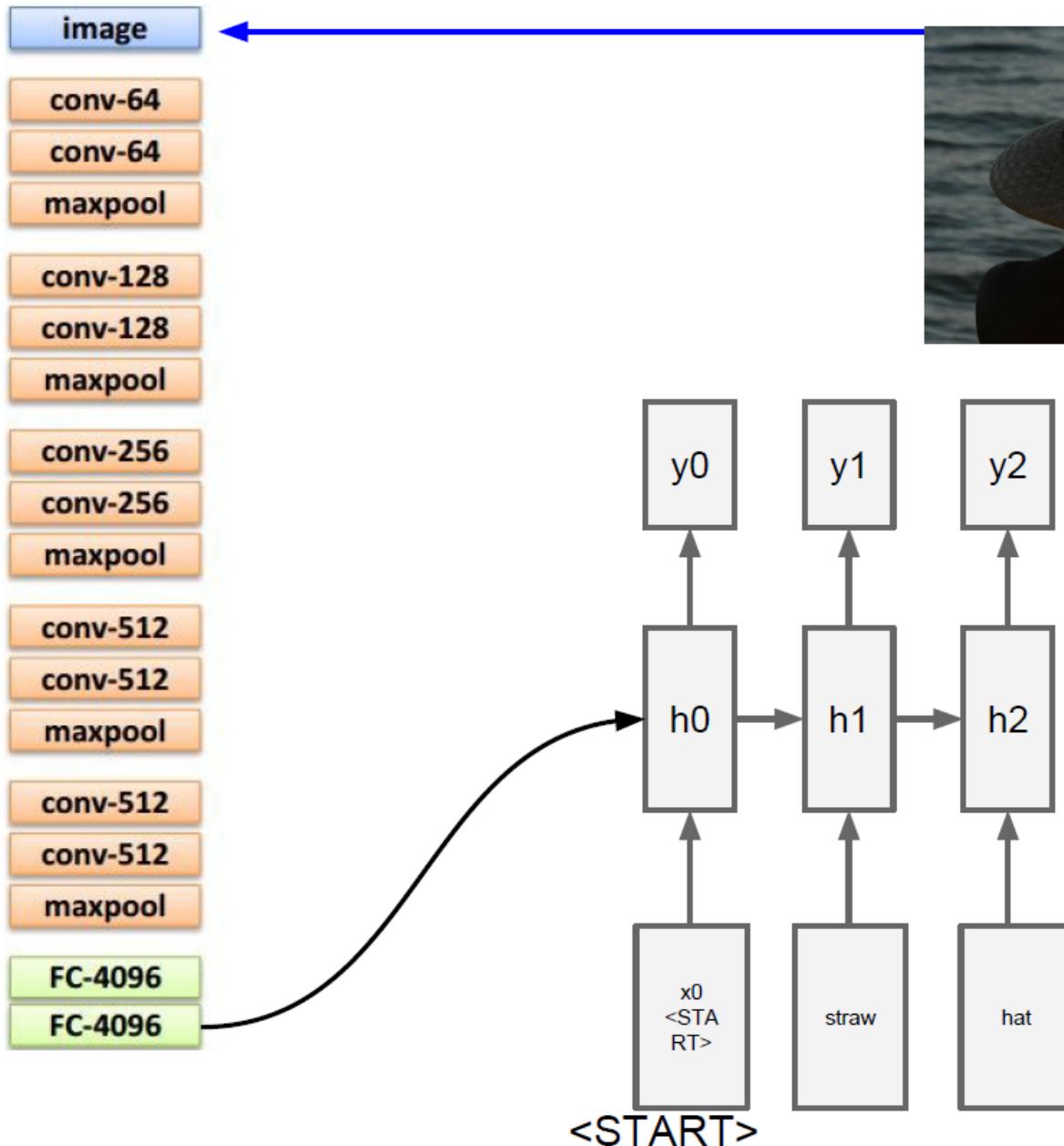


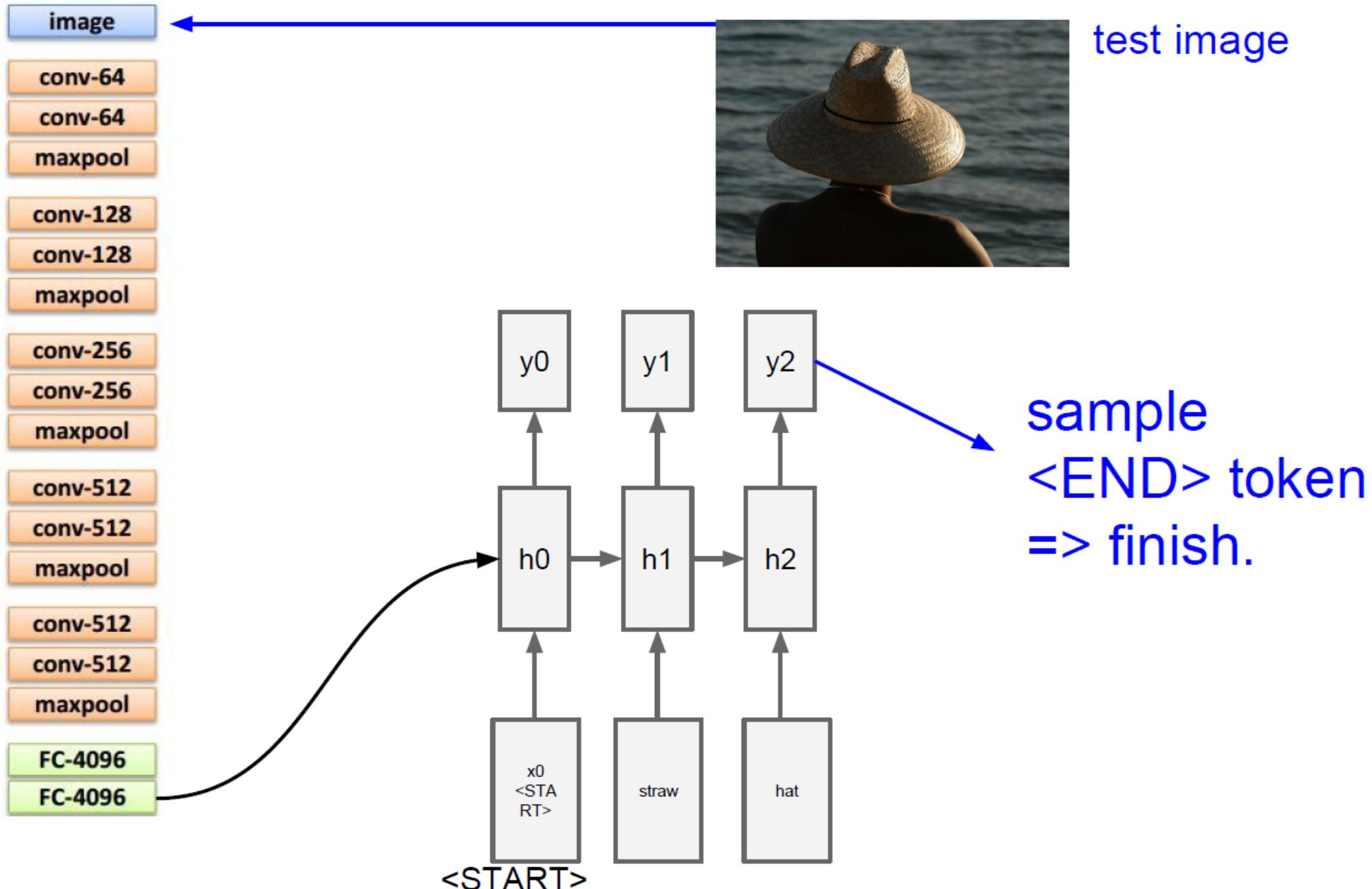
test image

sample!



test image





# Пример работы. Image Captioning



*A cat sitting on a suitcase on the floor*



*A cat is sitting on a tree branch*



*A dog is running in the grass with a frisbee*



*A white teddy bear sitting in the grass*



*Two people walking on the beach with surfboards*



*A tennis player in action on the court*



*Two giraffes standing in a grassy field*



*A man riding a dirt bike on a dirt track*

# Использованные материалы

- Книга «**Николенко, Кадурин, Архангельская: Глубокое обучение. Погружение в мир нейронных сетей**»