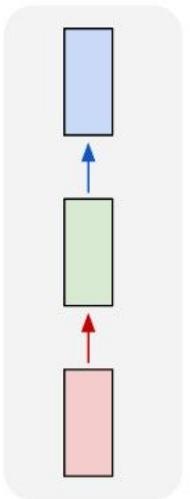


Обработка естественного языка: RNN, LSTM, Seq2seq

Языковые модели (LM)

“Vanilla” Neural Network

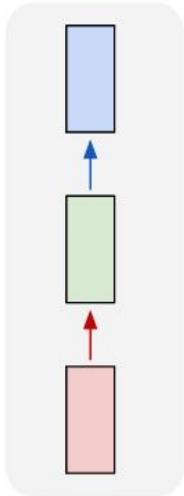
one to one



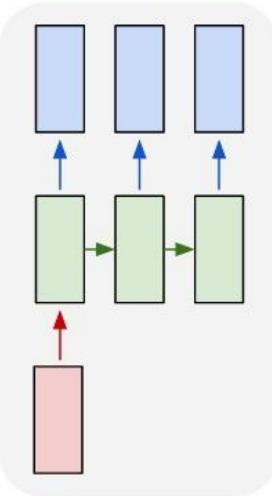
Vanilla Neural Networks

Recurrent Neural Networks: Process Sequences

one to one



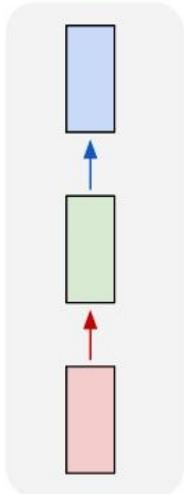
one to many



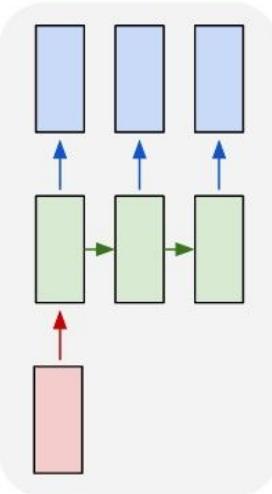
e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Networks: Process Sequences

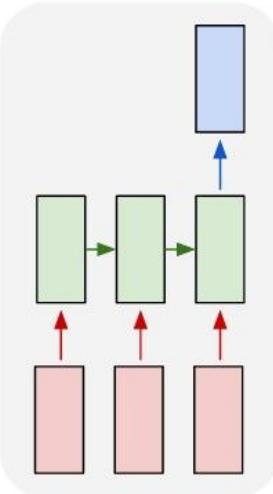
one to one



one to many



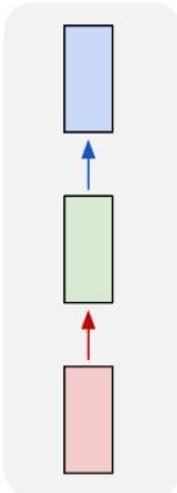
many to one



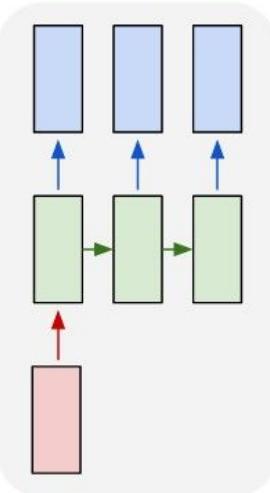
e.g. **action prediction**
sequence of video frames -> action class

Recurrent Neural Networks: Process Sequences

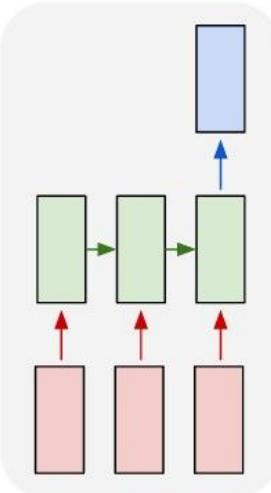
one to one



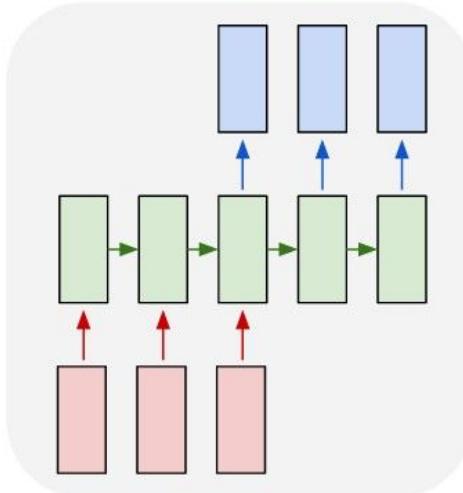
one to many



many to one



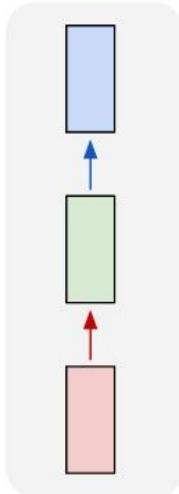
many to many



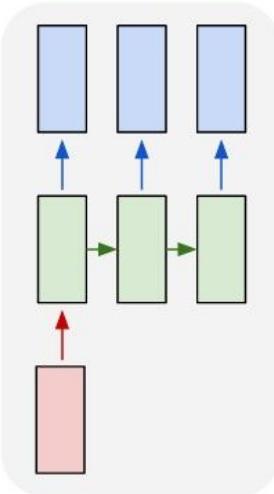
E.g. **Video Captioning**
Sequence of video frames ->
caption

Recurrent Neural Networks: Process Sequences

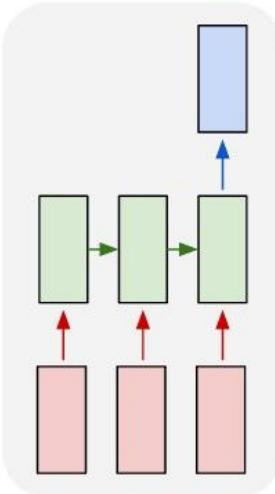
one to one



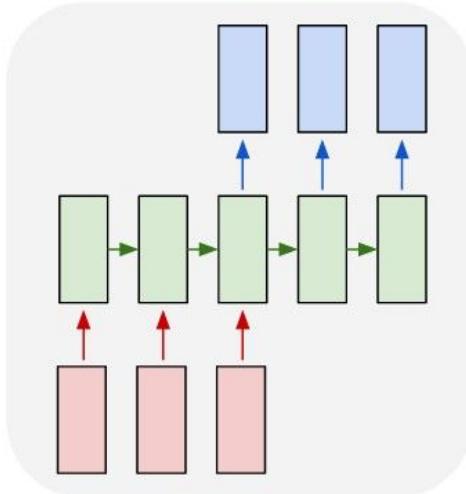
one to many



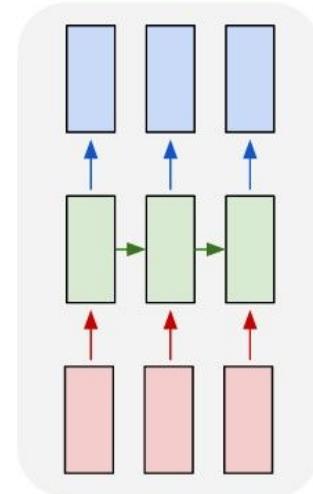
many to one



many to many



many to many



e.g. Video classification on frame level

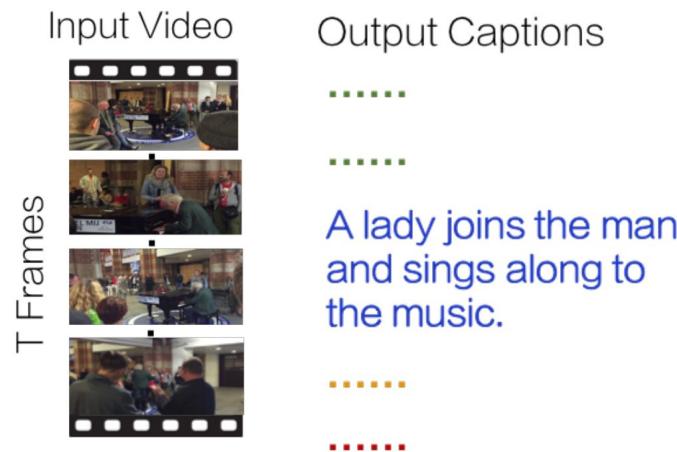
Why existing convnets are insufficient?

Variable sequence length inputs and outputs!

Example task: video captioning

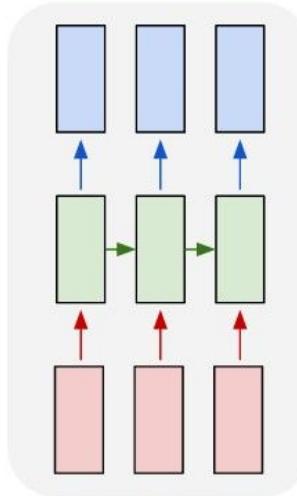
Input video can have variable number of frames

Output captions can be variable length.

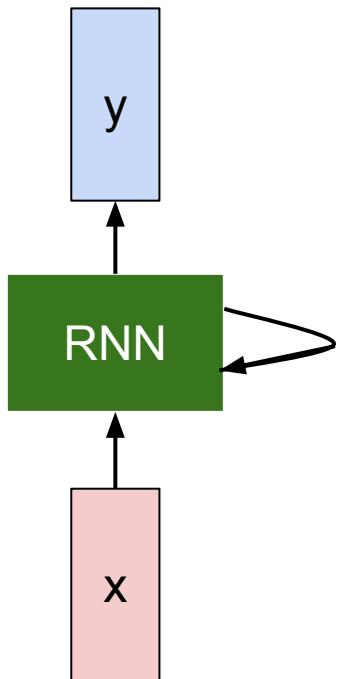


Let's start with a task that takes a variable input and produces an output at every step

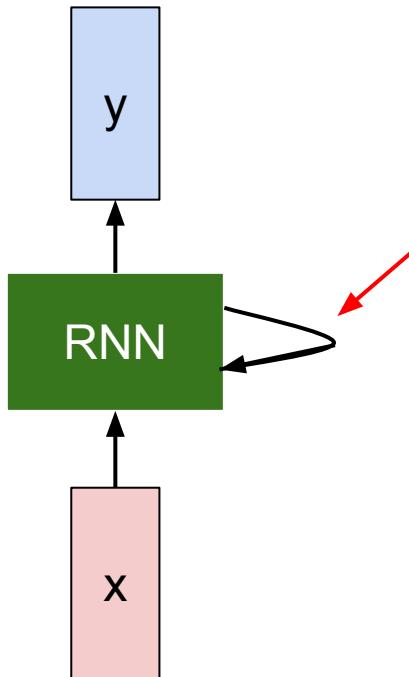
many to many



Recurrent Neural Network

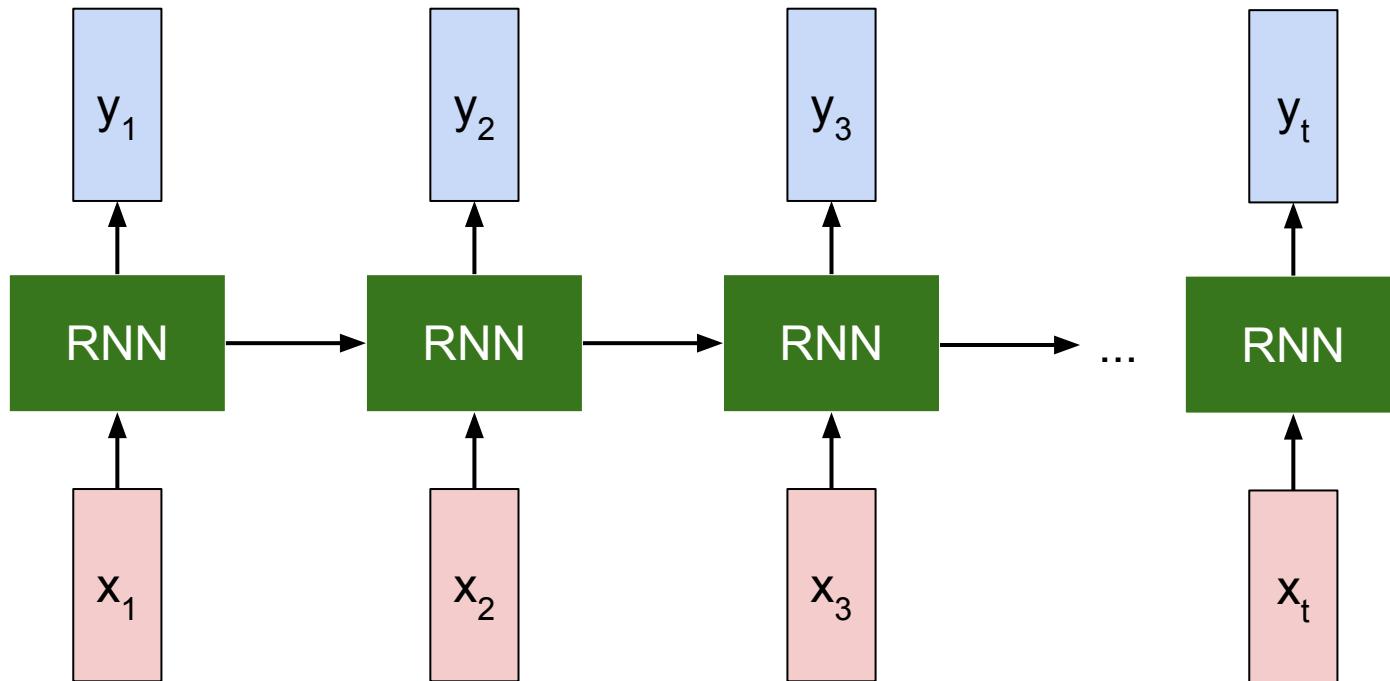


Recurrent Neural Network



Key idea: RNNs have an “internal state” that is updated as a sequence is processed

Unrolled RNN

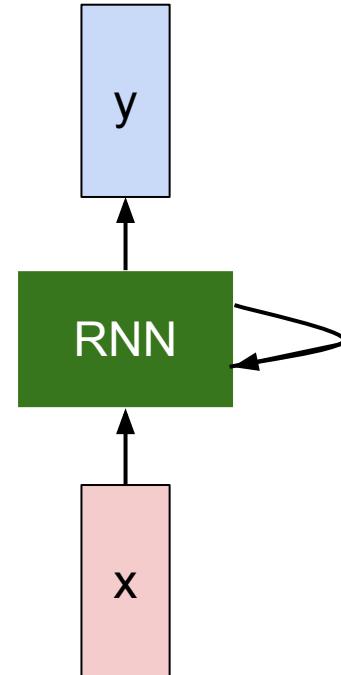


RNN hidden state update

We can process a sequence of vectors x by applying a **recurrence formula** at every time step:

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

new state / old state input vector at
some function | some time step
with parameters W

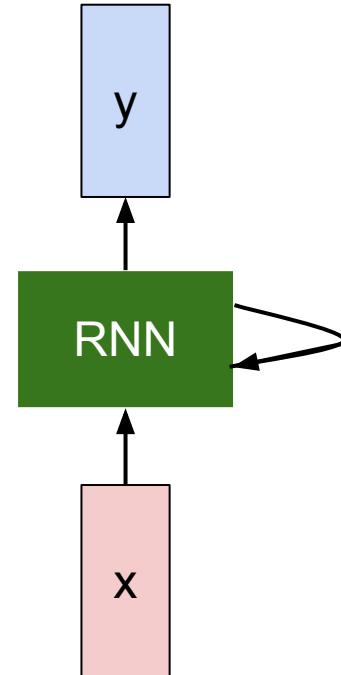


RNN output generation

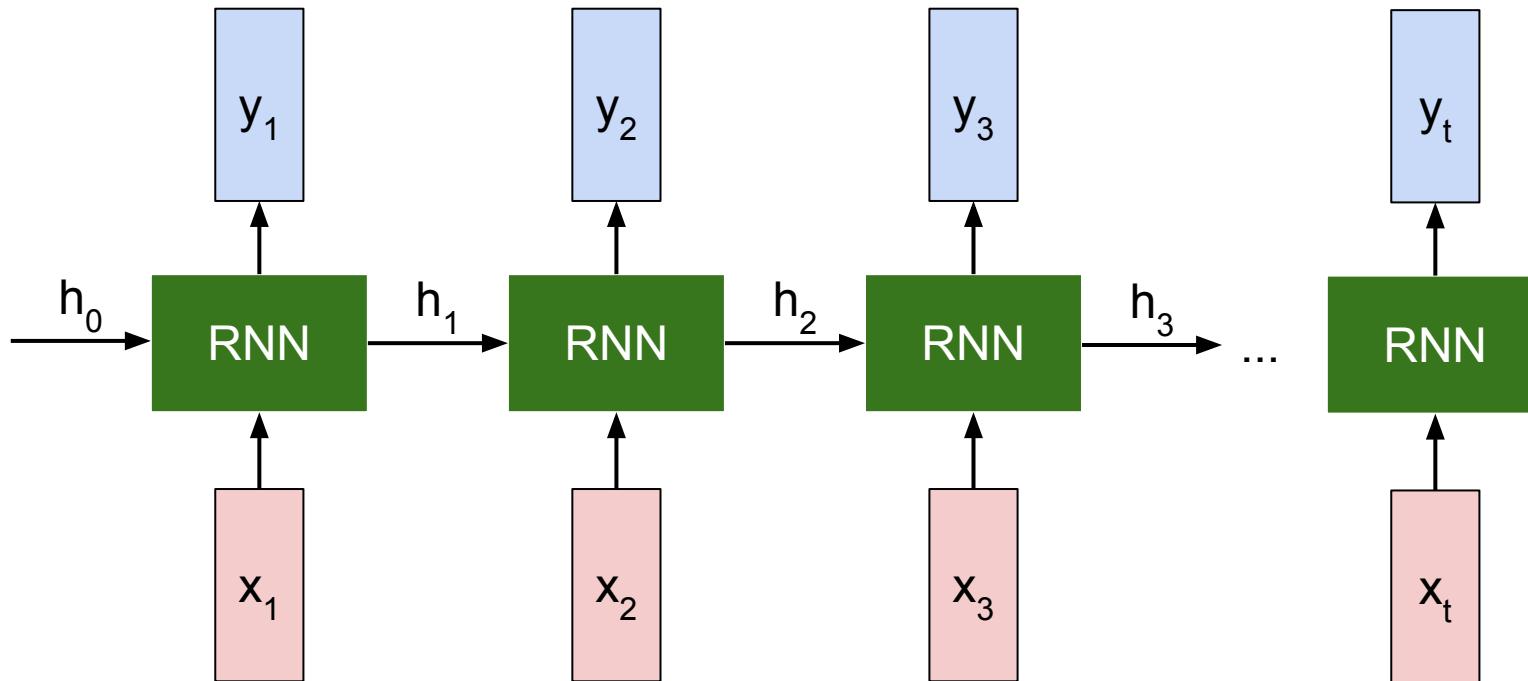
We can process a sequence of vectors x by applying a **recurrence formula** at every time step:

$$y_t = f_{W_{hy}}(h_t)$$

output new state
another function
with parameters W_o



Recurrent Neural Network

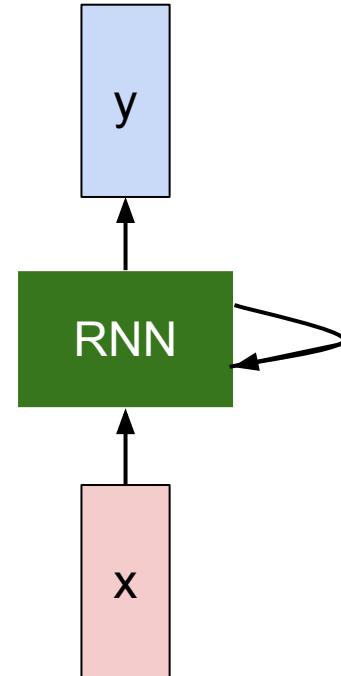


Recurrent Neural Network

We can process a sequence of vectors \mathbf{x} by applying a **recurrence formula** at every time step:

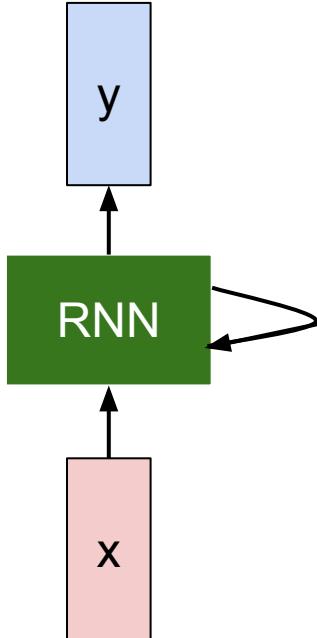
$$h_t = f_W(h_{t-1}, x_t)$$

Notice: the same function and the same set of parameters are used at every time step.



(Simple) Recurrent Neural Network

The state consists of a single “*hidden*” vector \mathbf{h} :



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



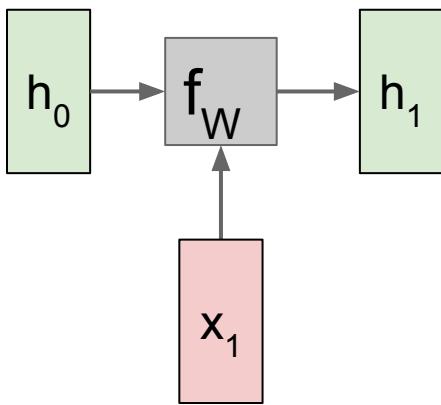
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Sometimes called a “Vanilla RNN” or an
“Elman RNN” after Prof. Jeffrey Elman

RNN: Computational Graph

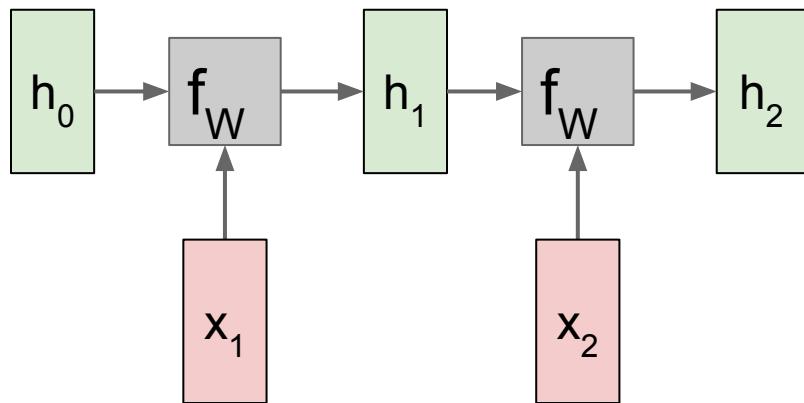
Tanh (гиперболический тангенс) — это функция активации, часто используемая в нейронных сетях, особенно в рекуррентных нейронных сетях (RNN). Она принимает значения на входе и "сжимает" их в диапазон от -1 до 1.



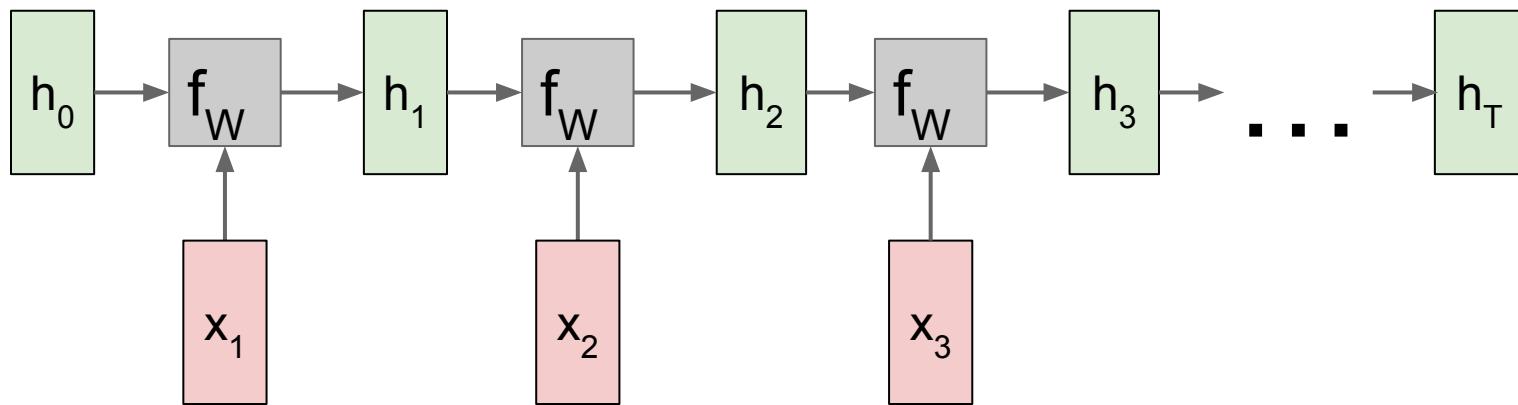
Whx — матрица весов для входа, связывает текущий вход x_{txt} с текущим скрытым состоянием h_{ht} .

Whh — матрица весов для скрытого состояния, связывает скрытое состояние на предыдущем временном шаге h_{t-1} с текущим скрытым состоянием h_{ht} .

RNN: Computational Graph

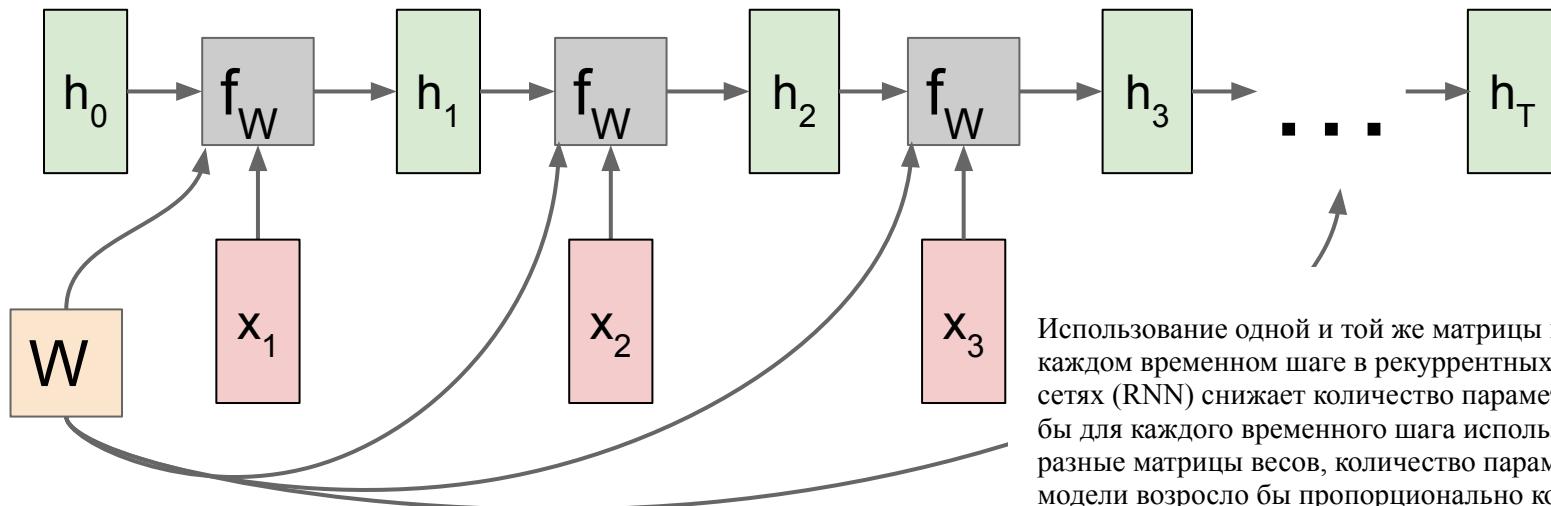


RNN: Computational Graph



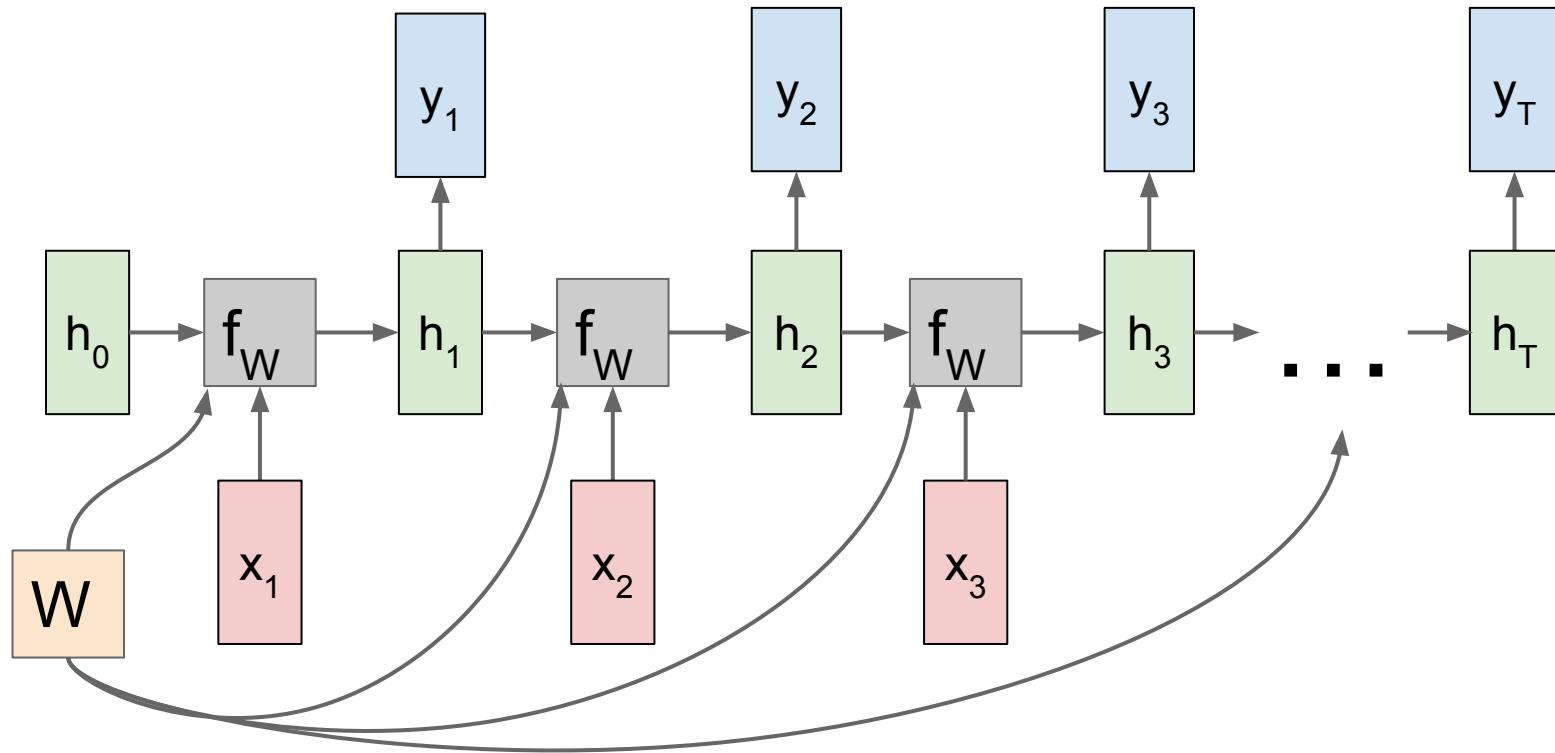
RNN: Computational Graph

Re-use the same weight matrix at every time-step

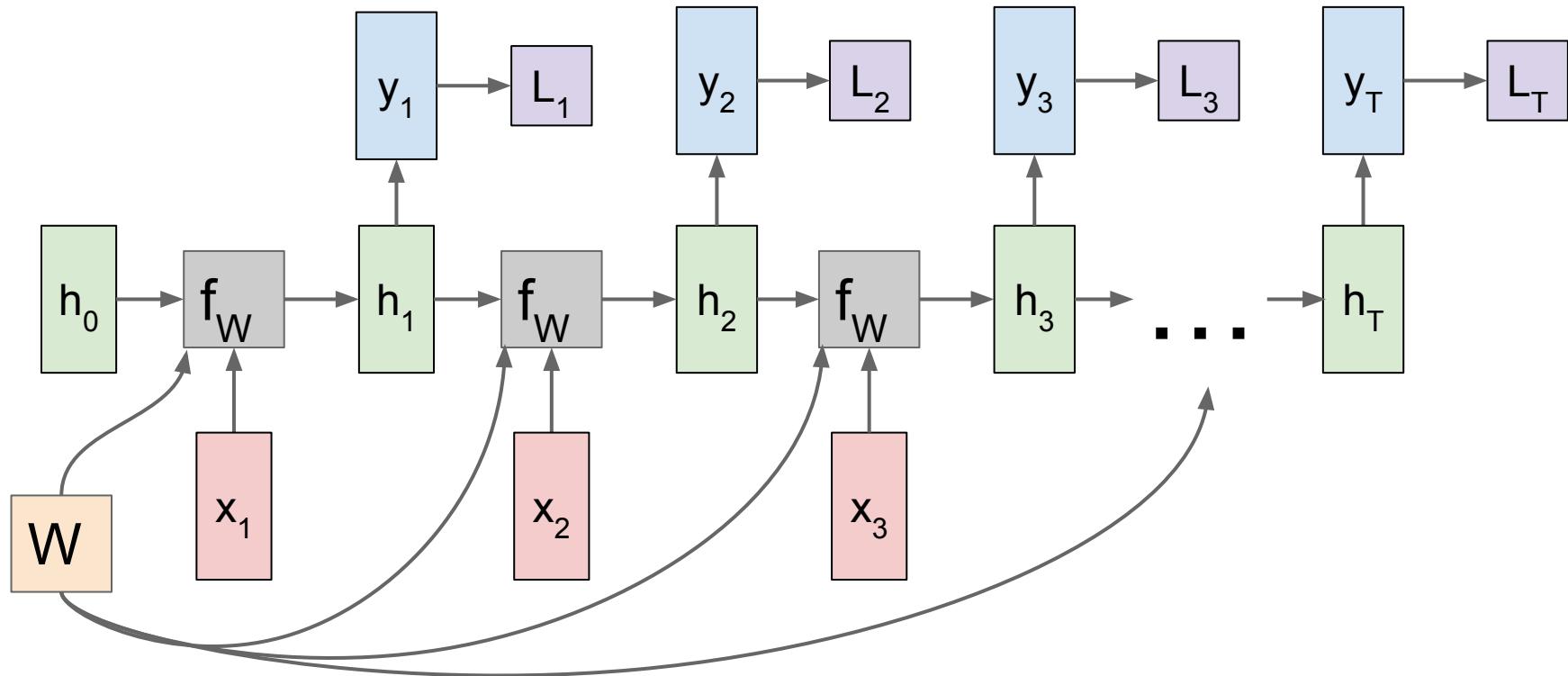


Использование одной и той же матрицы весов на каждом временном шаге в рекуррентных нейронных сетях (RNN) снижает количество параметров: Если бы для каждого временного шага использовались разные матрицы весов, количество параметров в модели возросло бы пропорционально количеству временных шагов. Это сделало бы обучение модели более ресурсоемким и сложным.

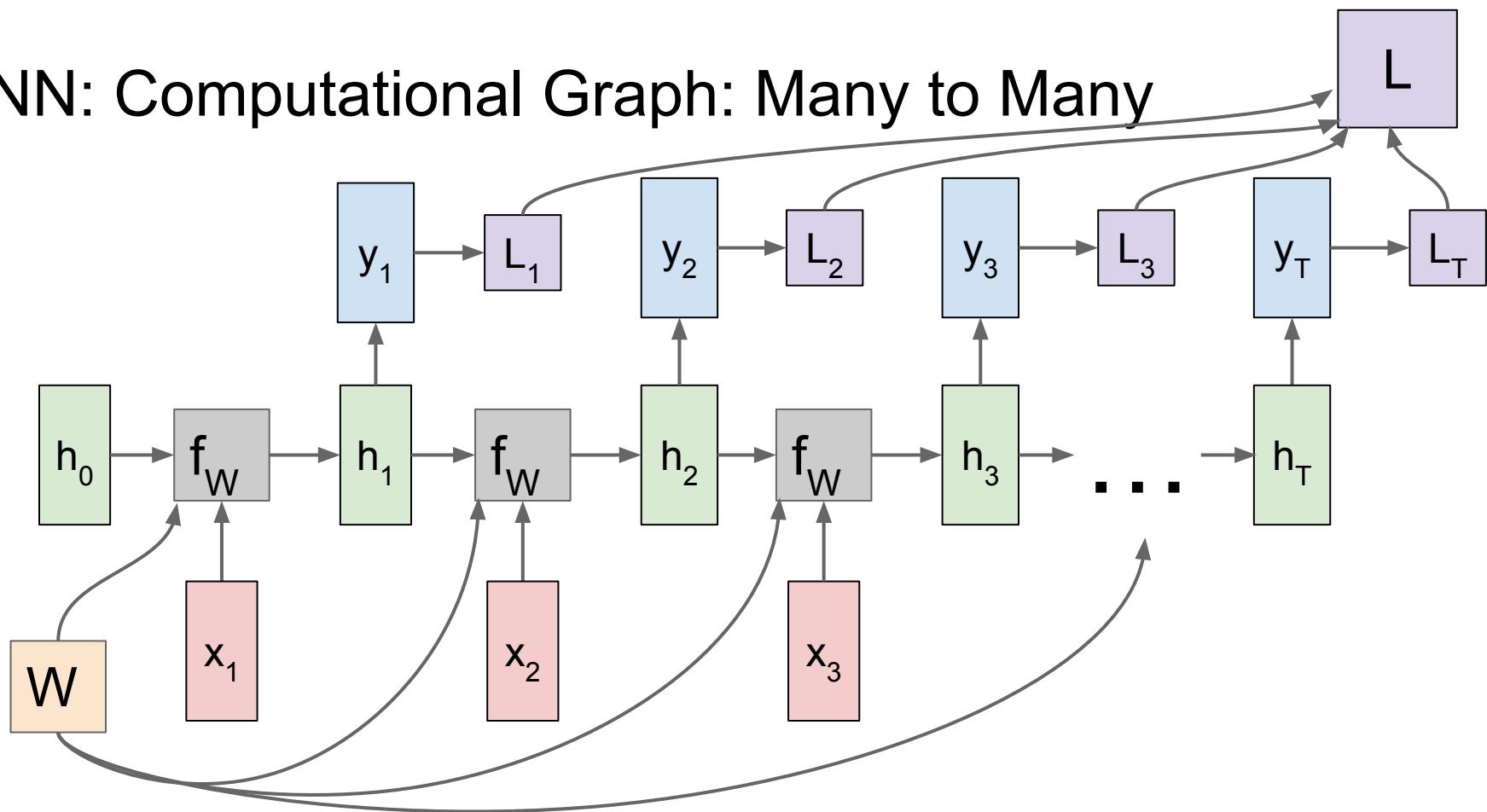
RNN: Computational Graph: Many to Many



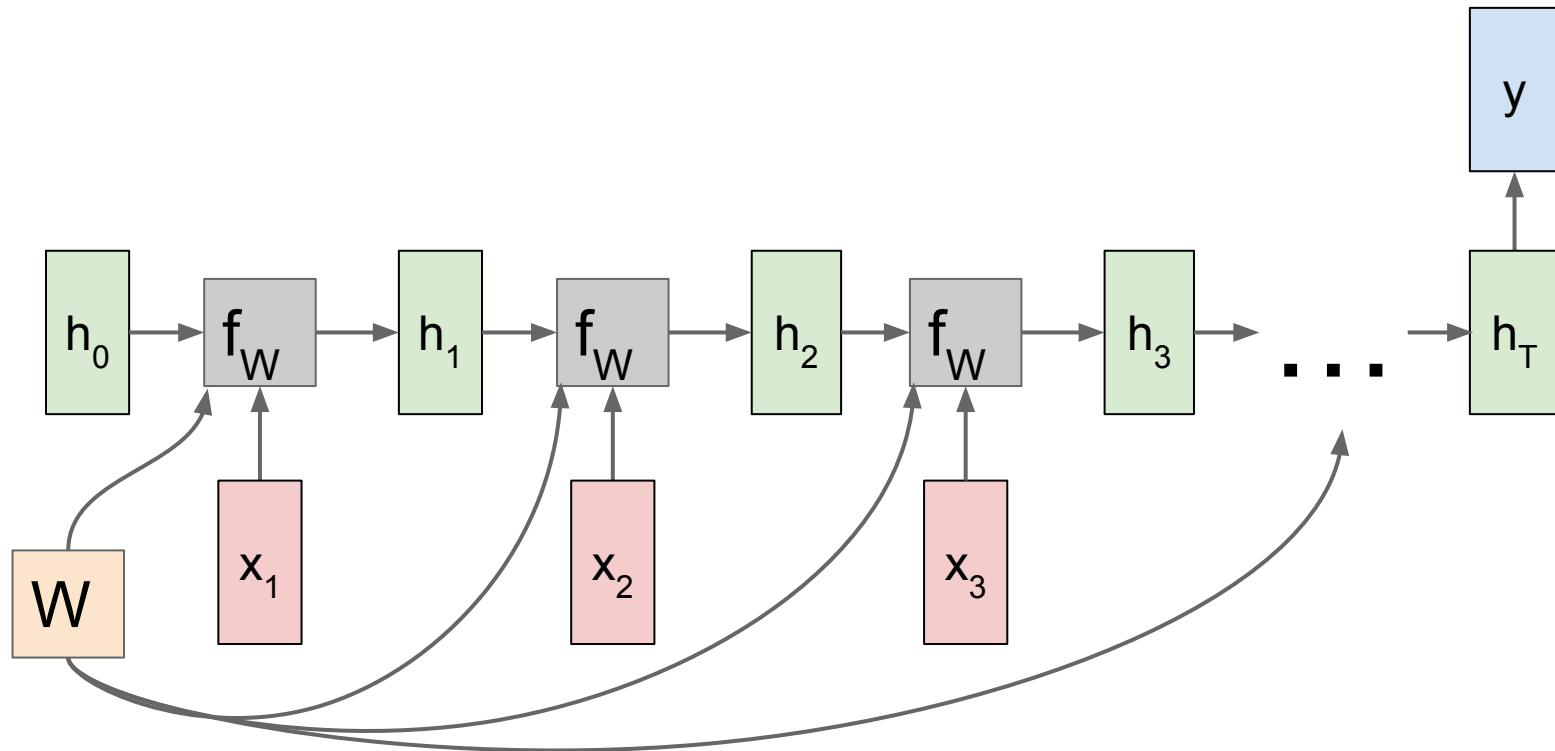
RNN: Computational Graph: Many to Many



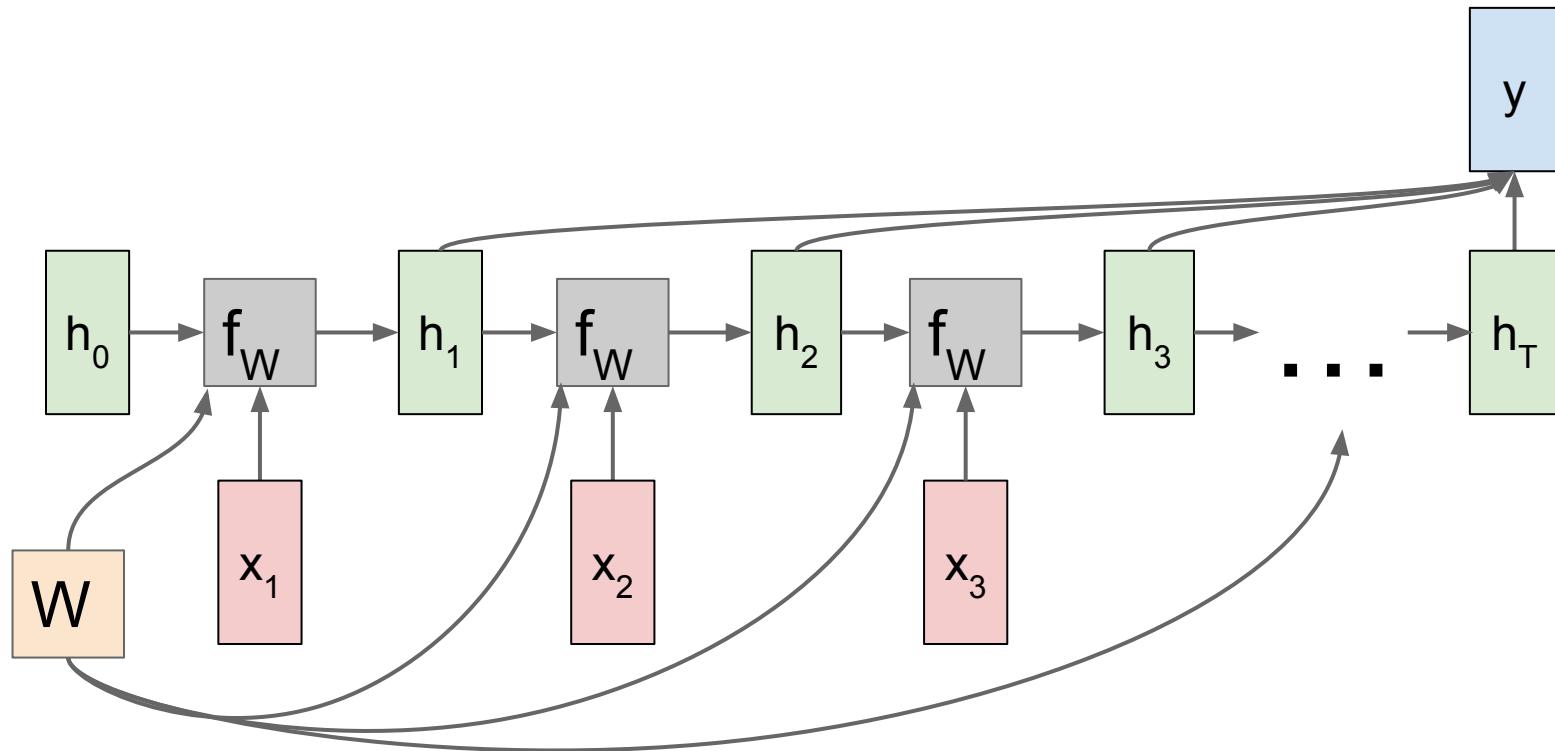
RNN: Computational Graph: Many to Many



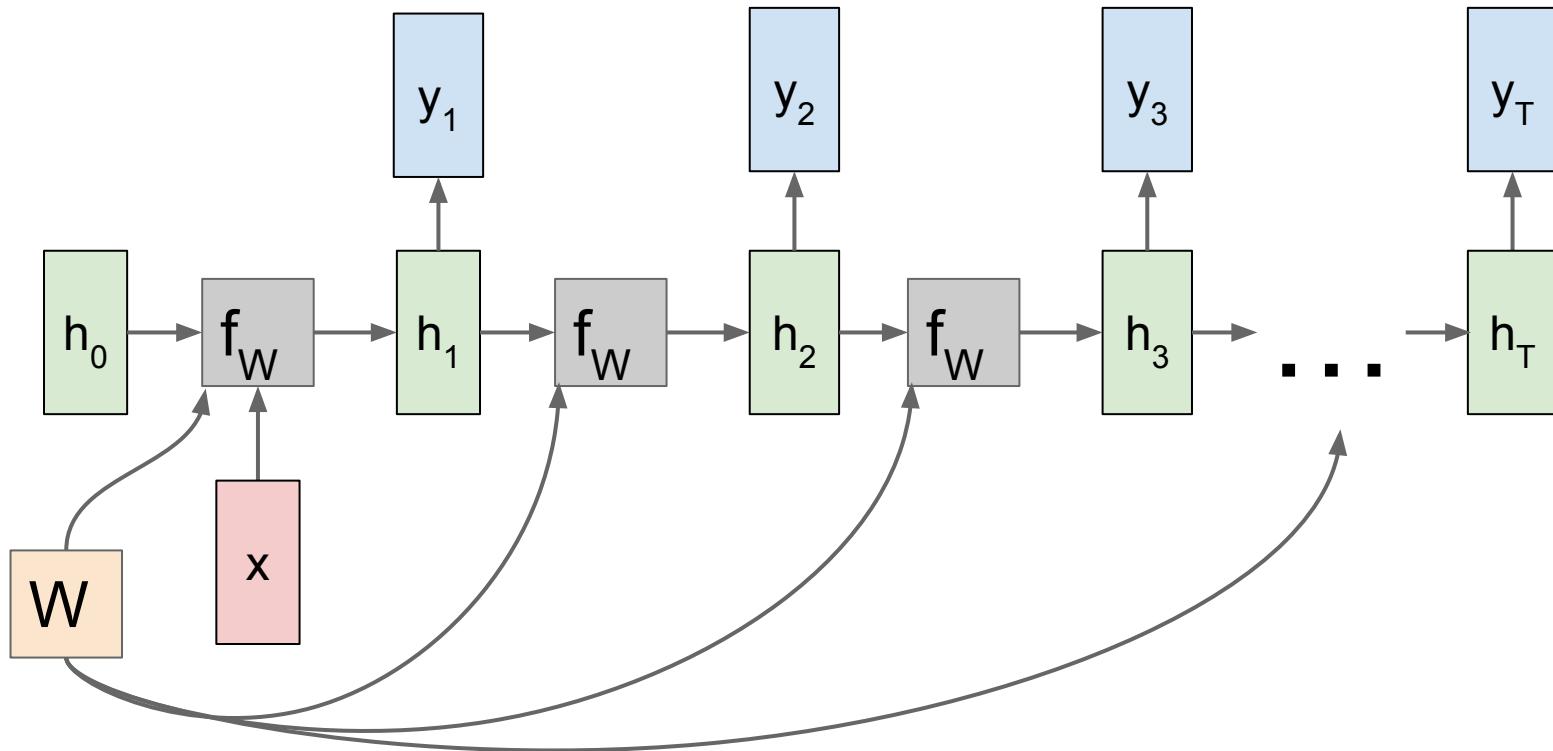
RNN: Computational Graph: Many to One



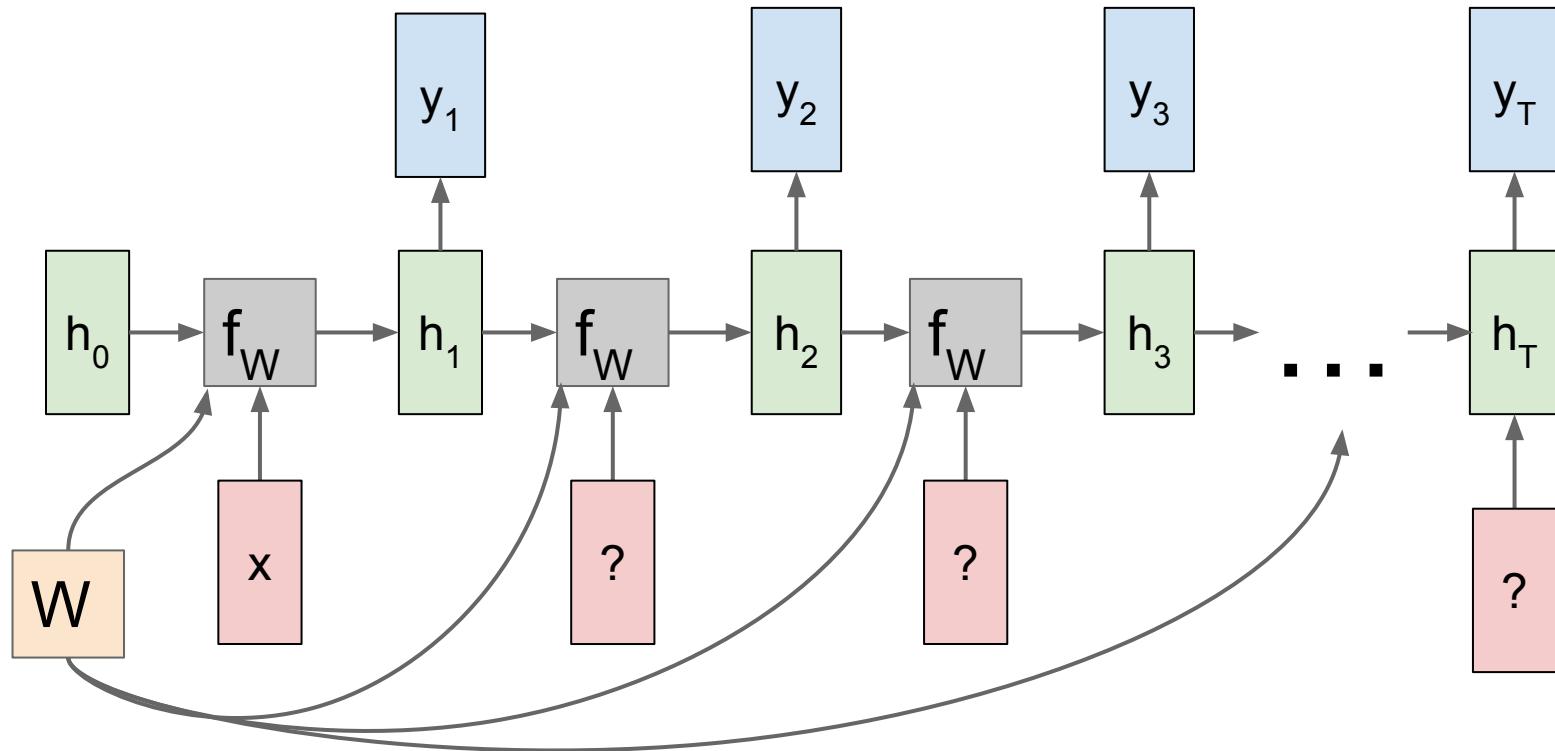
RNN: Computational Graph: Many to One



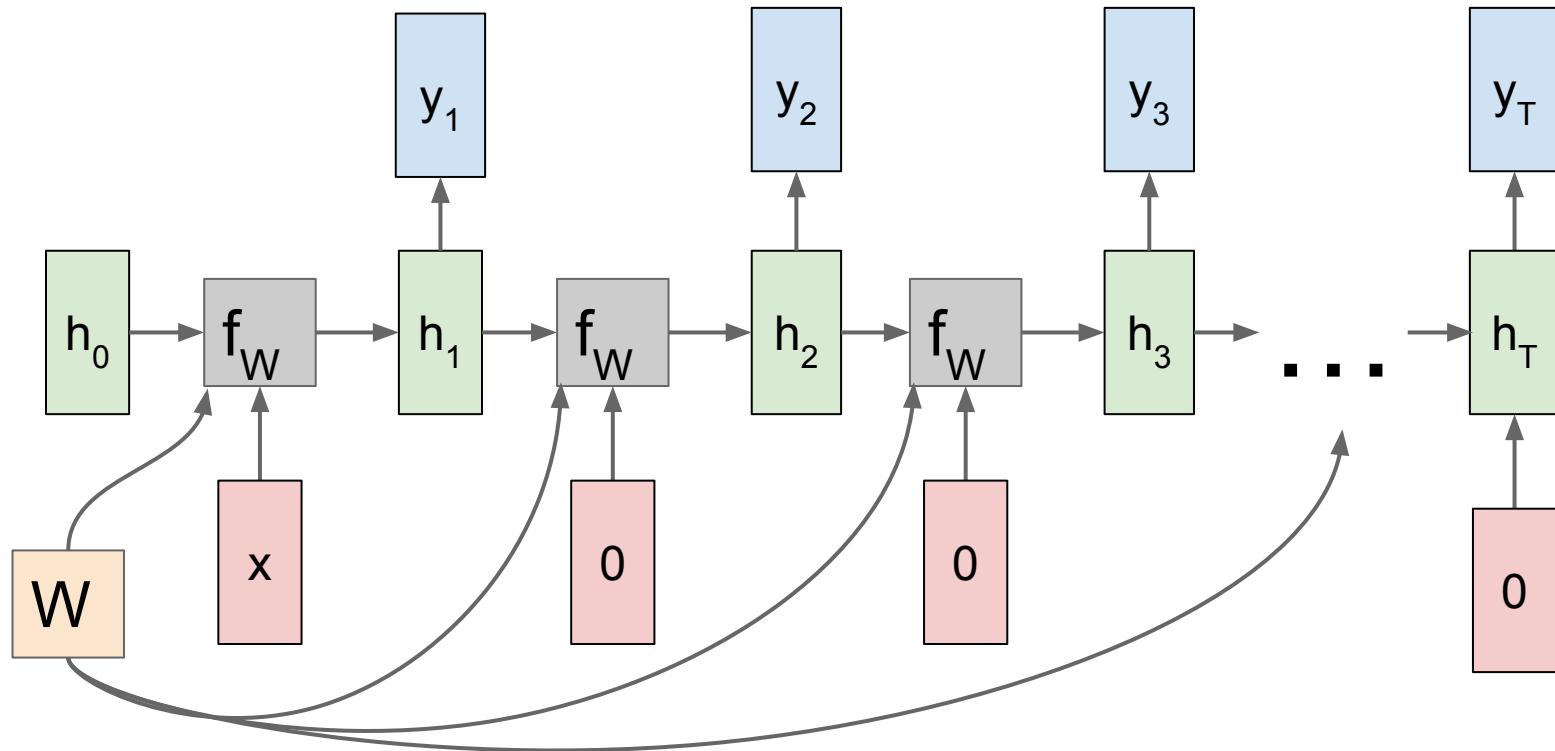
RNN: Computational Graph: One to Many



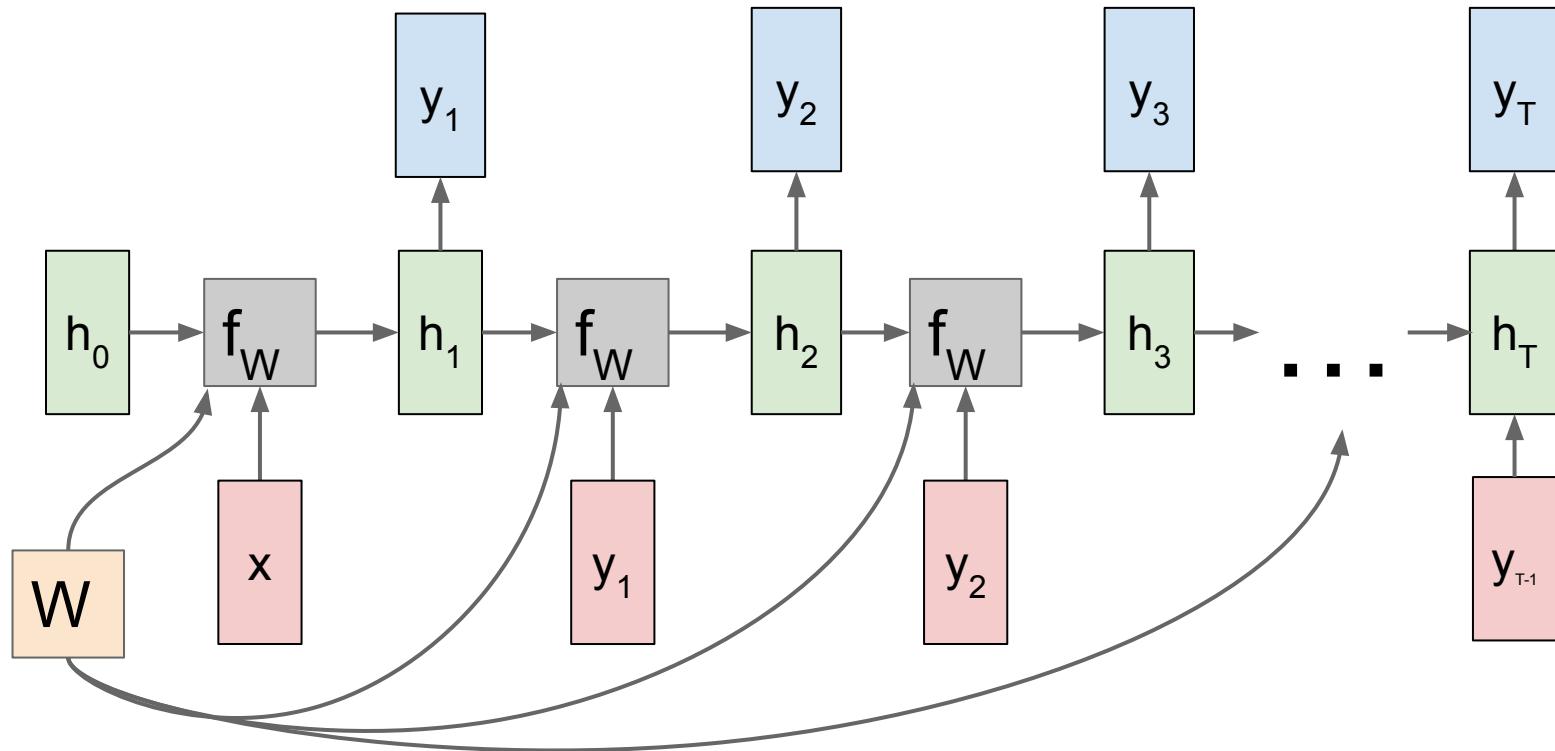
RNN: Computational Graph: One to Many



RNN: Computational Graph: One to Many



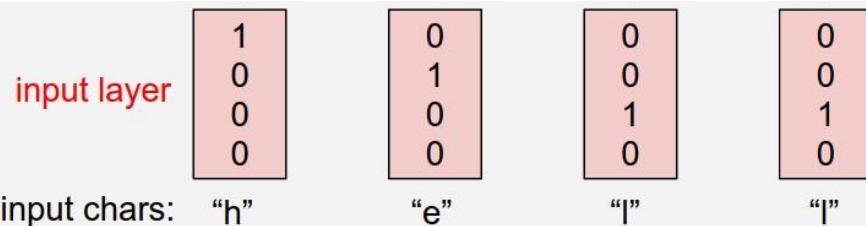
RNN: Computational Graph: One to Many



Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

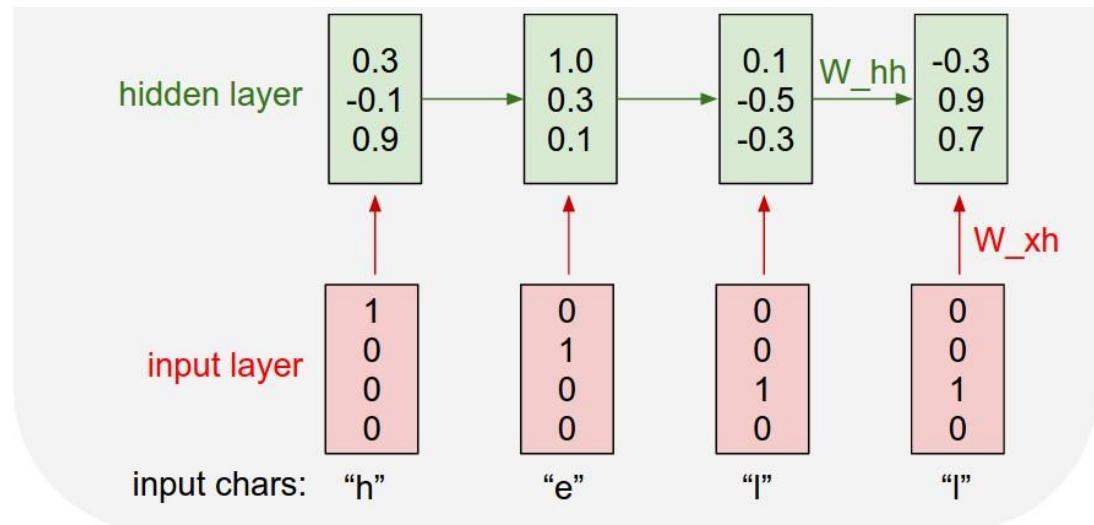


Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

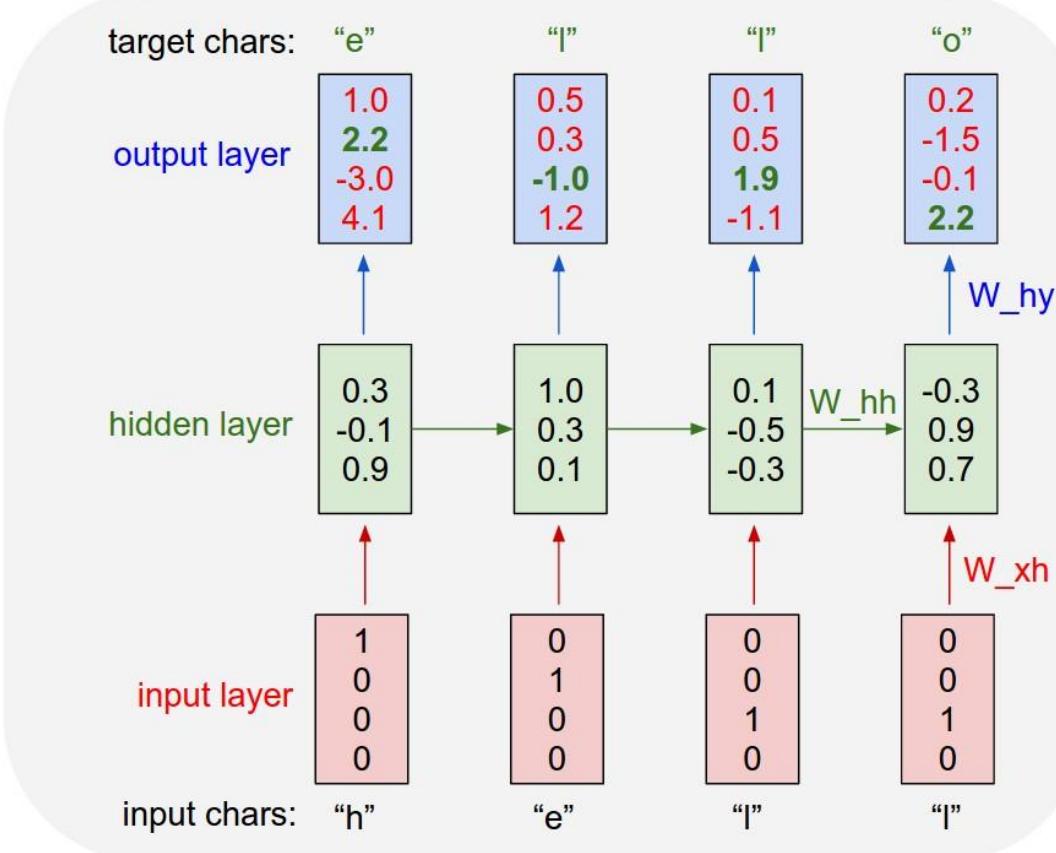
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

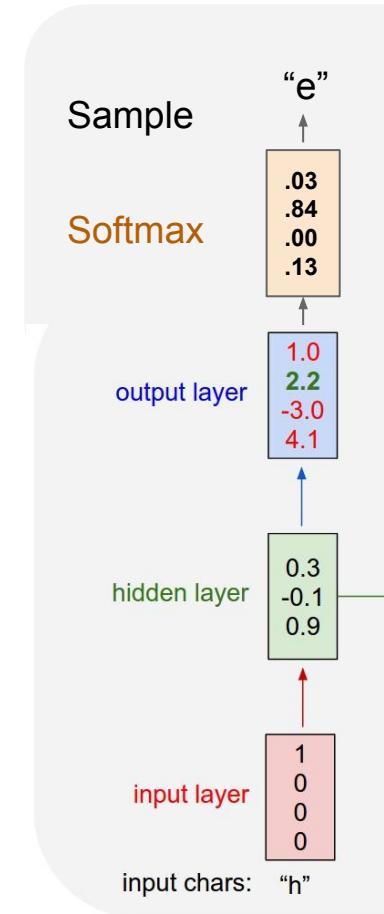
Example training
sequence:
“hello”



Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

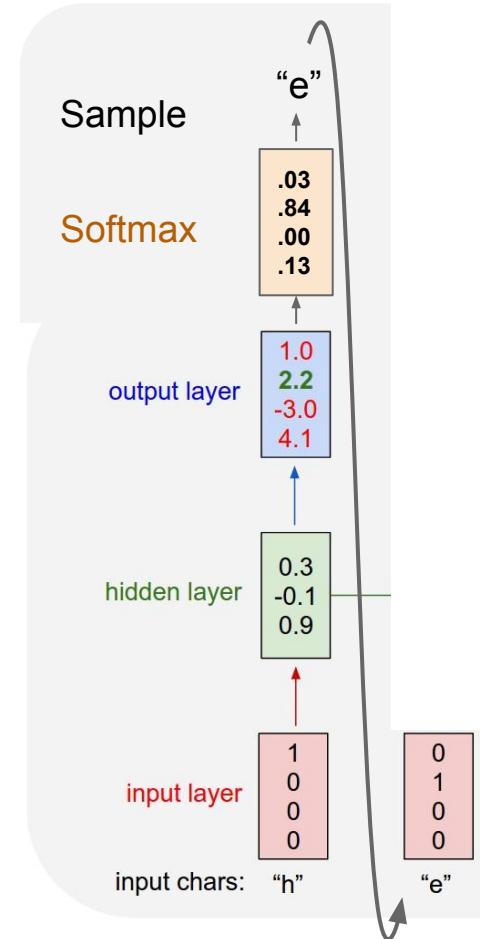
At test-time sample
characters one at a time,
feed back to model



Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

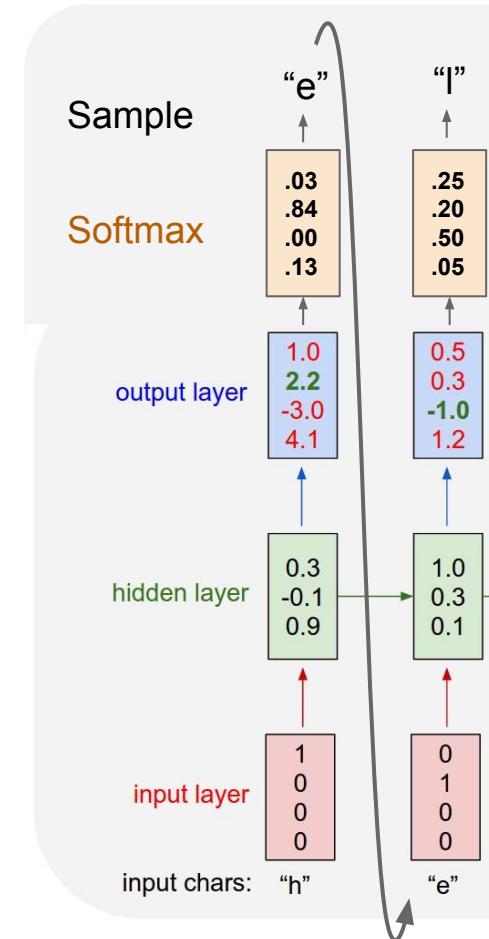
At test-time sample
characters one at a time,
feed back to model



Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

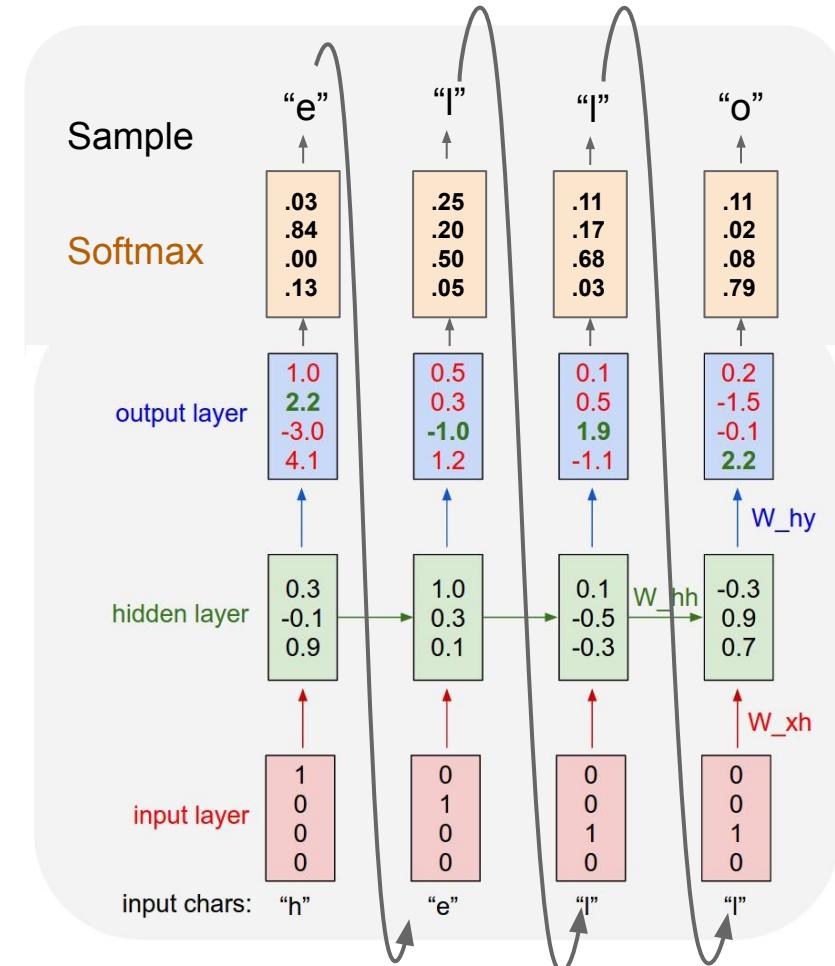
At test-time sample
characters one at a time,
feed back to model



Example: Character-level Language Model Sampling

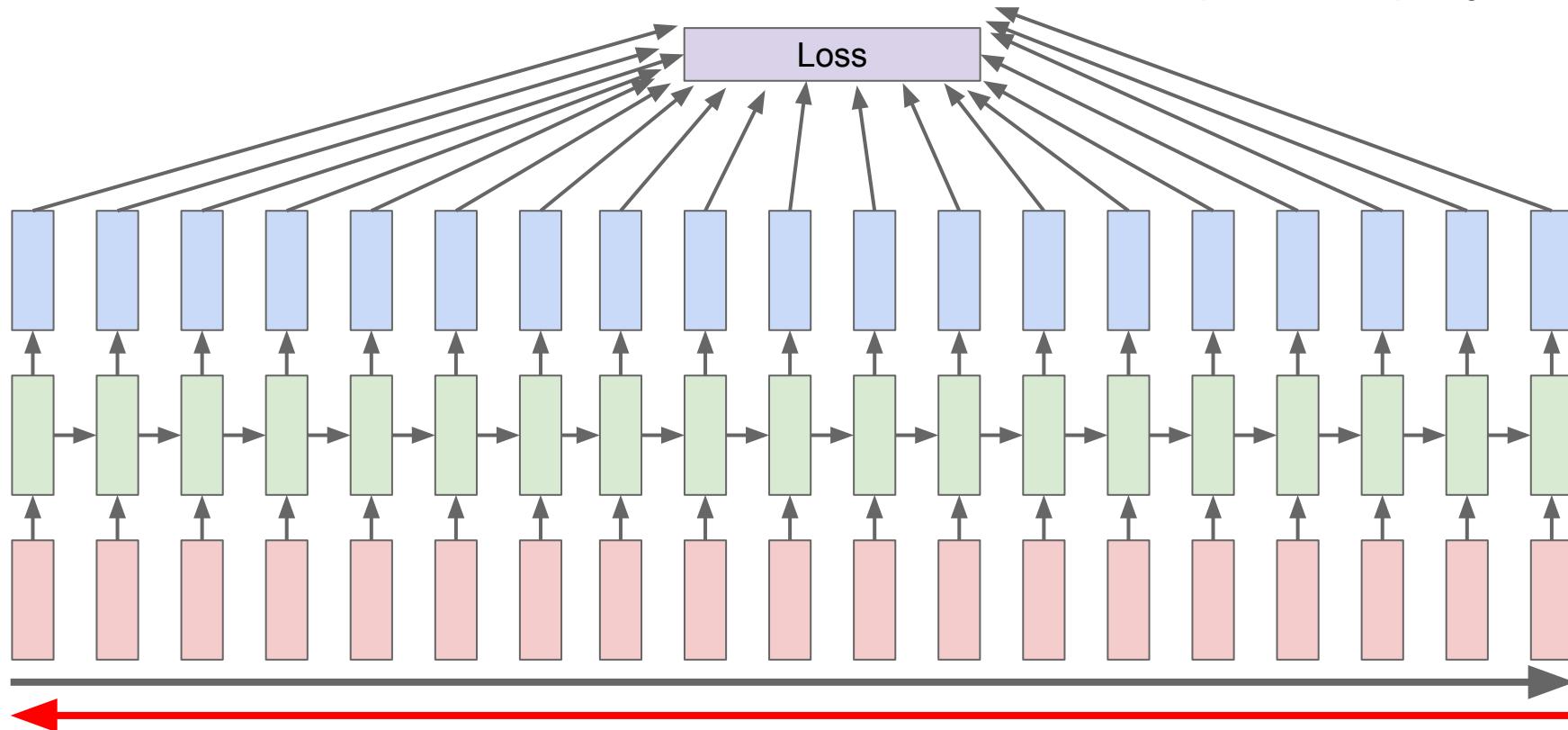
Vocabulary:
[h,e,l,o]

At test-time sample
characters one at a time,
feed back to model

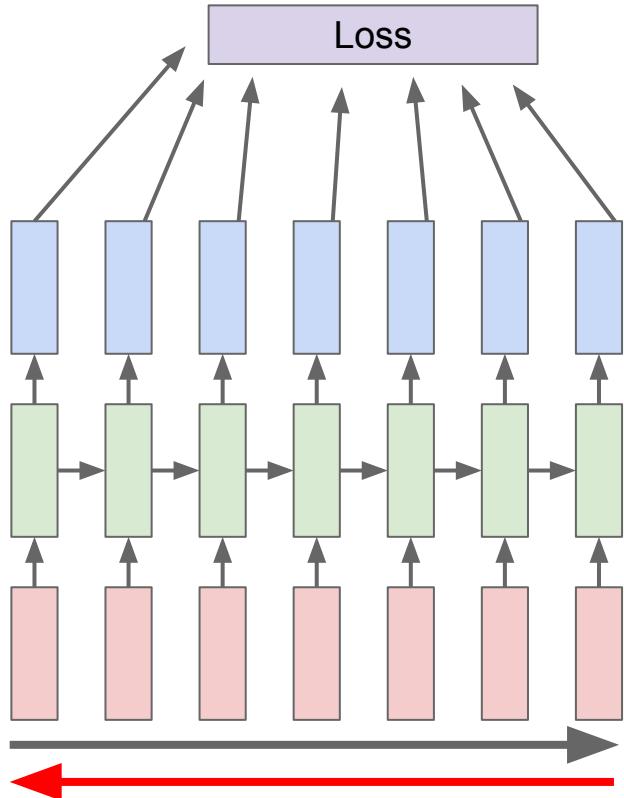


Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

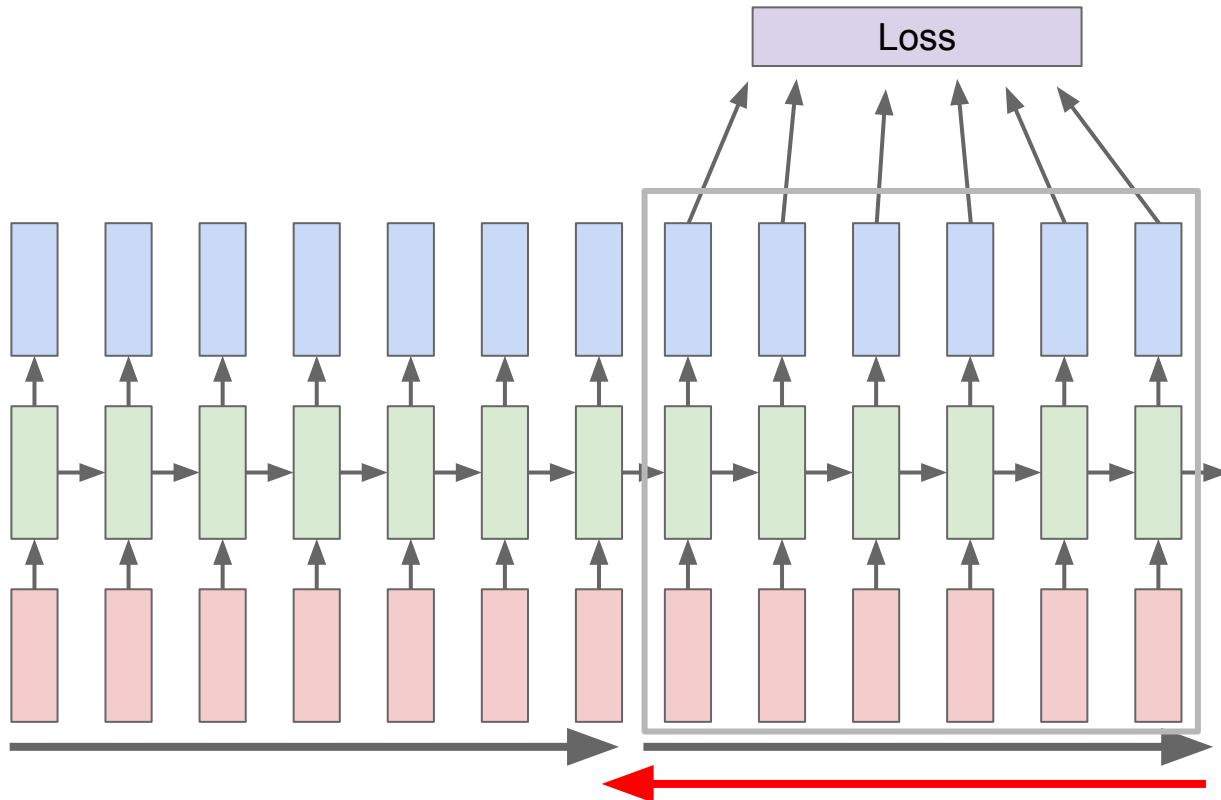


Truncated Backpropagation through time



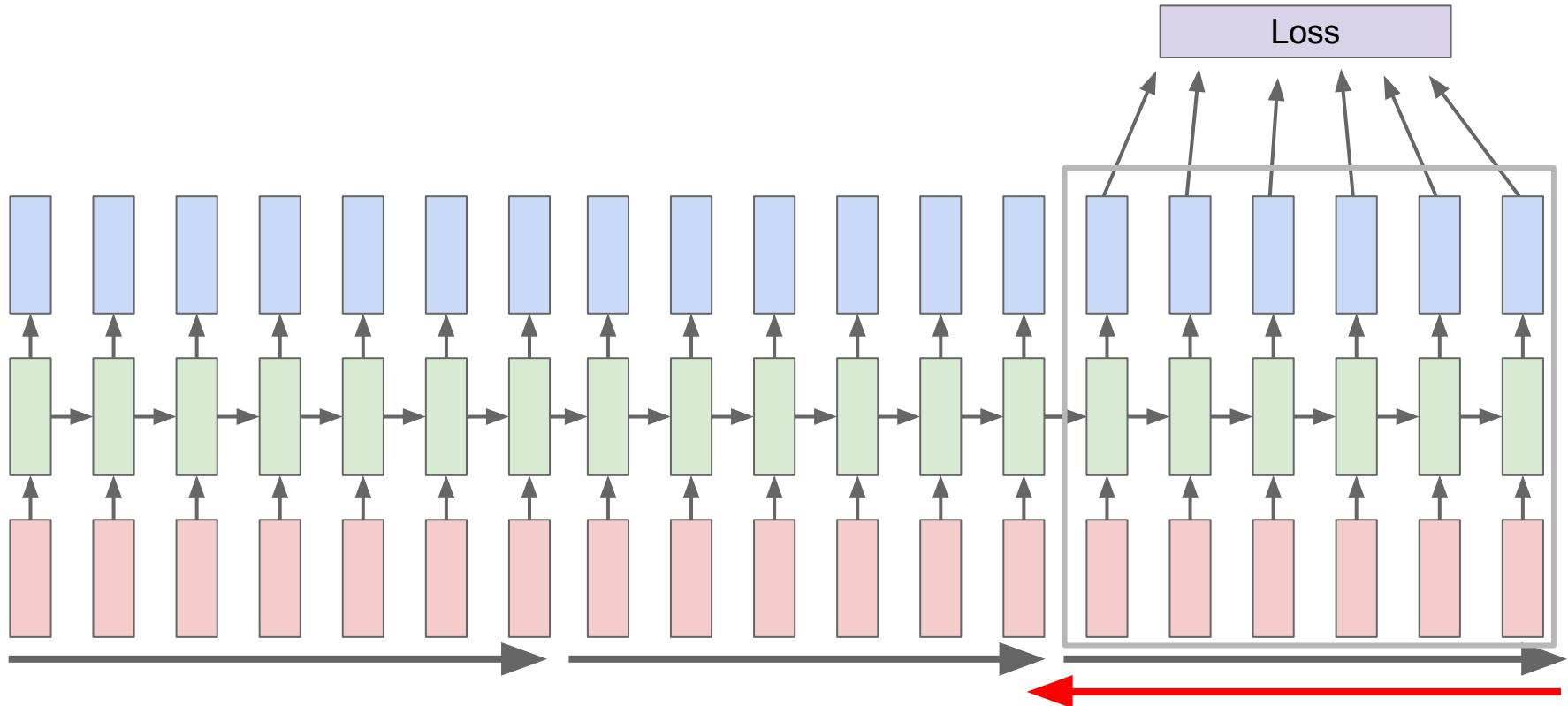
Run forward and backward
through chunks of the
sequence instead of whole
sequence

Truncated Backpropagation through time



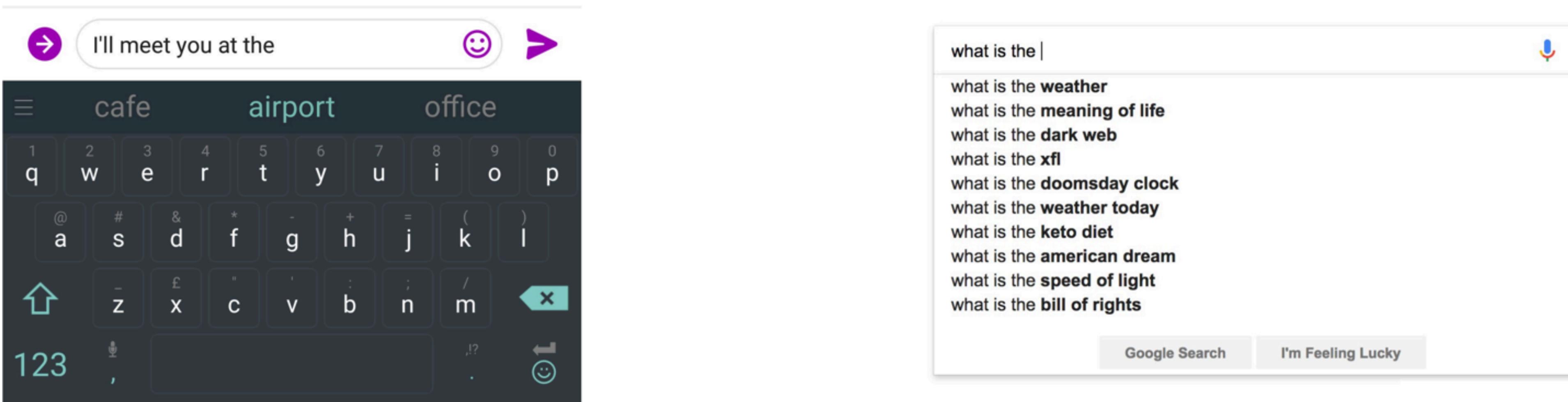
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Truncated Backpropagation through time



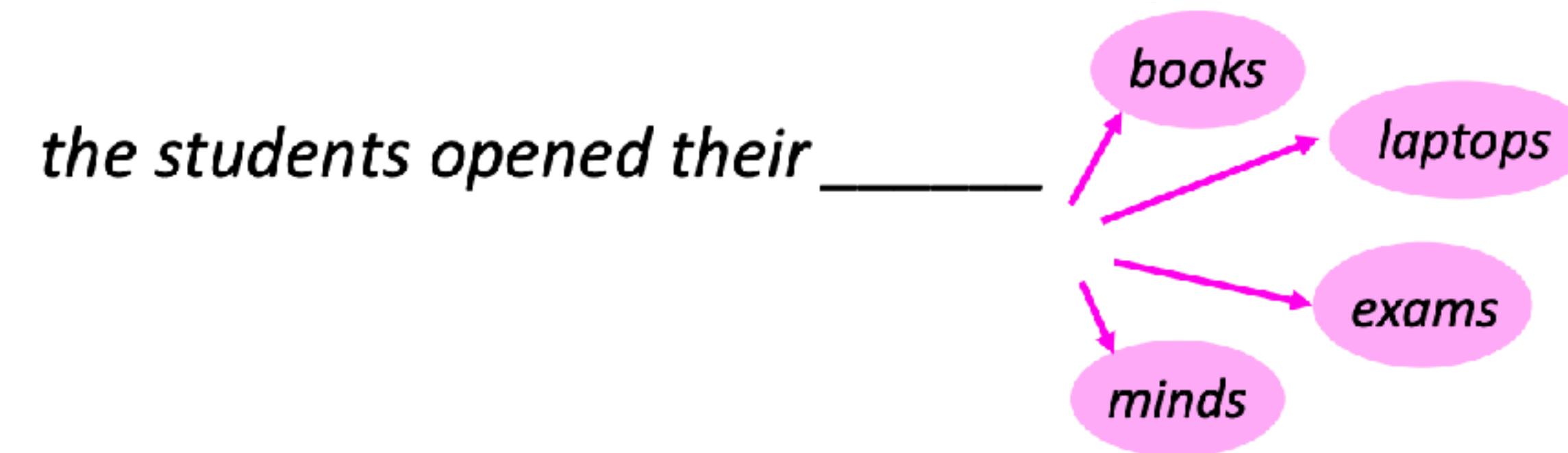
Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим



Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим

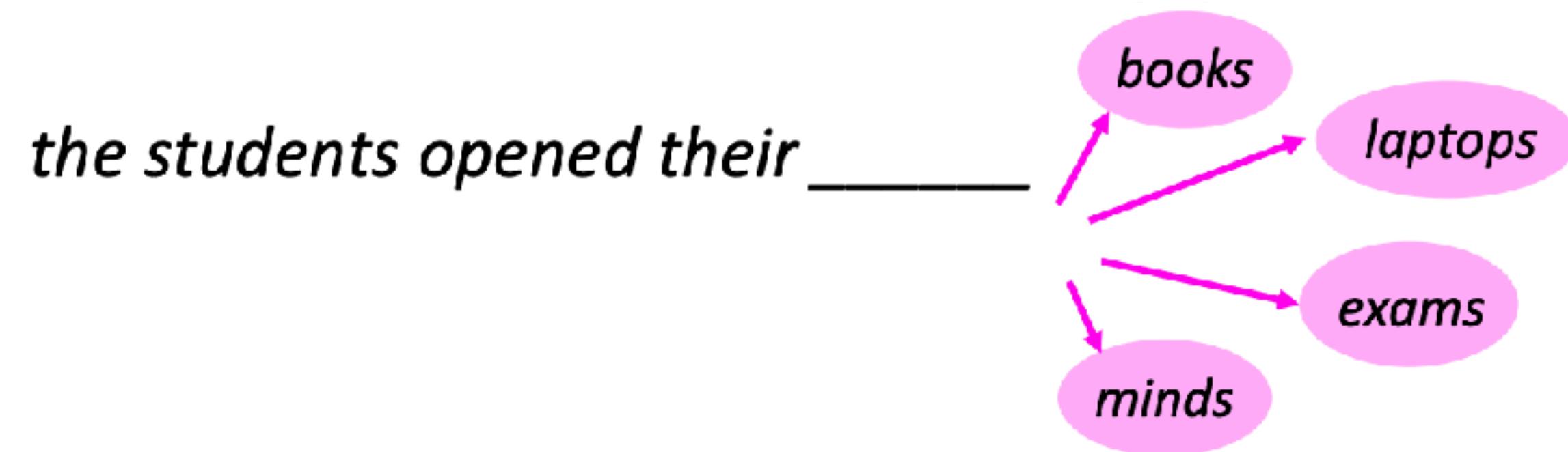


Авторегрессионная модель

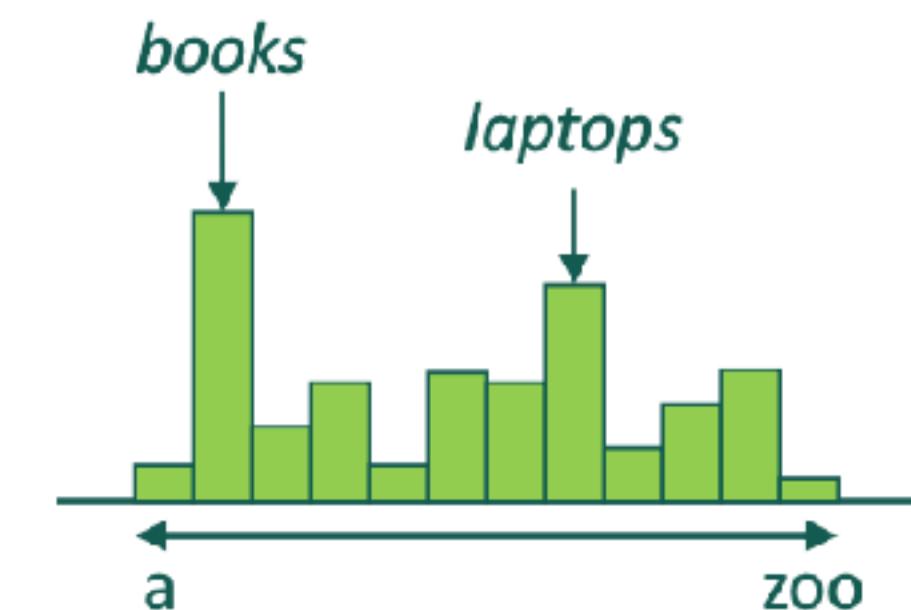
$$p(x) = p(x_n | x_1, \dots, x_{n-1}) \cdot p(x_2 | x_1) \cdot p(x_1) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$

Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим

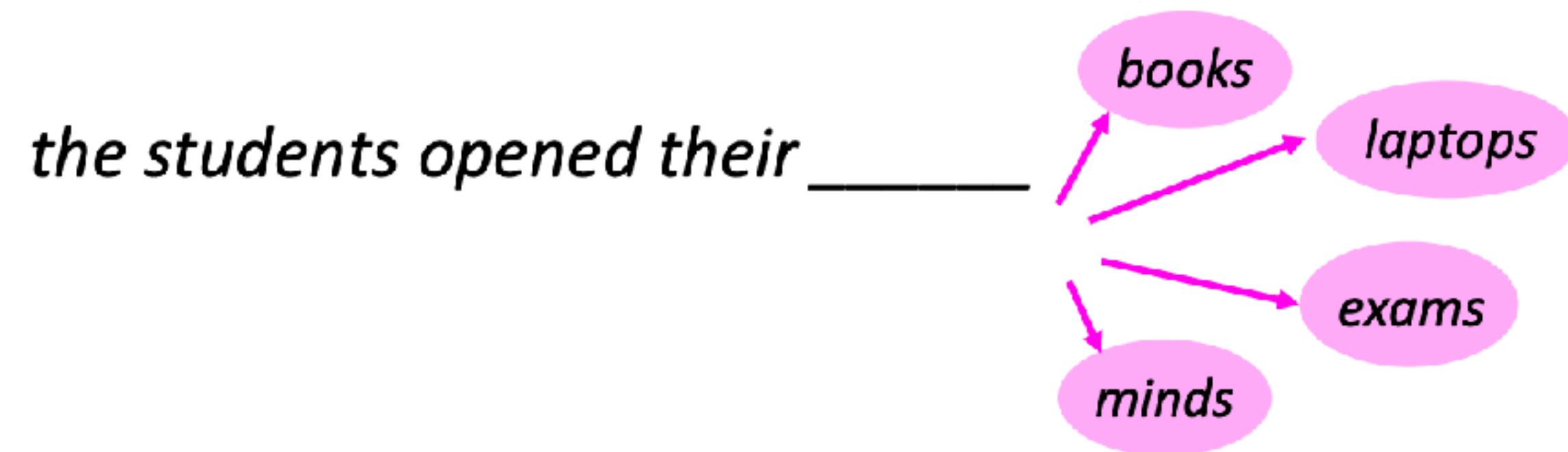


На выходе 5го шага: $p(x_5 | \text{the}, \text{students}, \text{opened}, \text{their}), x_5 \in \text{Vocab}$



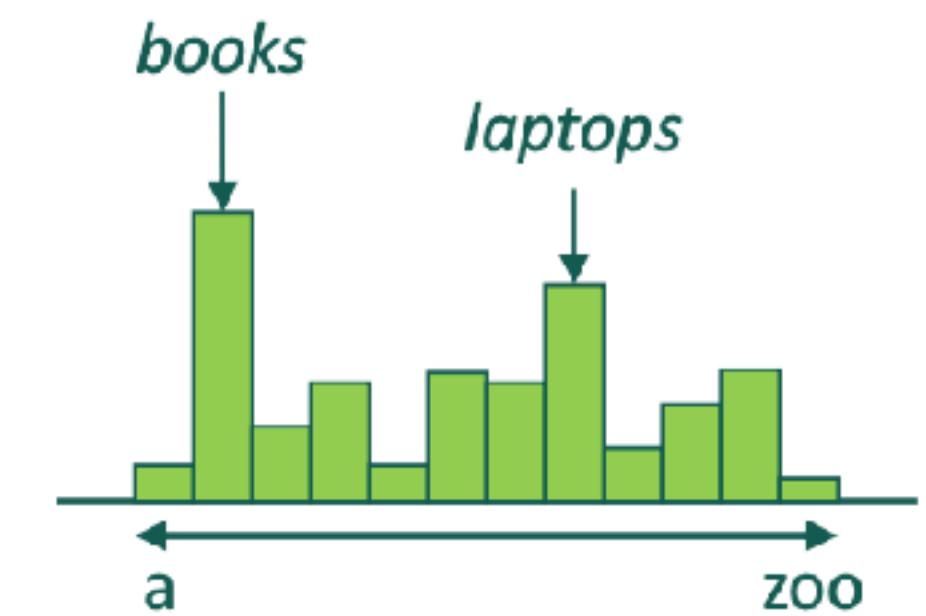
Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим



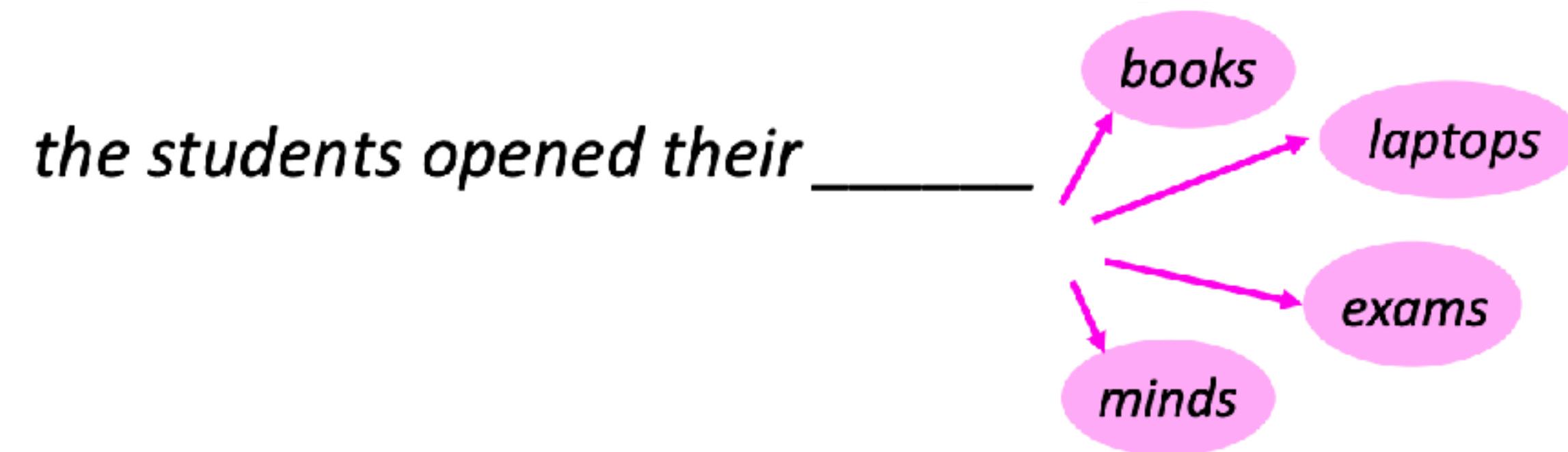
На выходе 5го шага: $p(x_5 | \text{the}, \text{students}, \text{opened}, \text{their}), x_5 \in \text{Vocab}$

Классификация картинок: $p(\text{class} | \text{image}), \text{class} \in \{\text{cat}, \text{dog}, \text{etc.}\}$



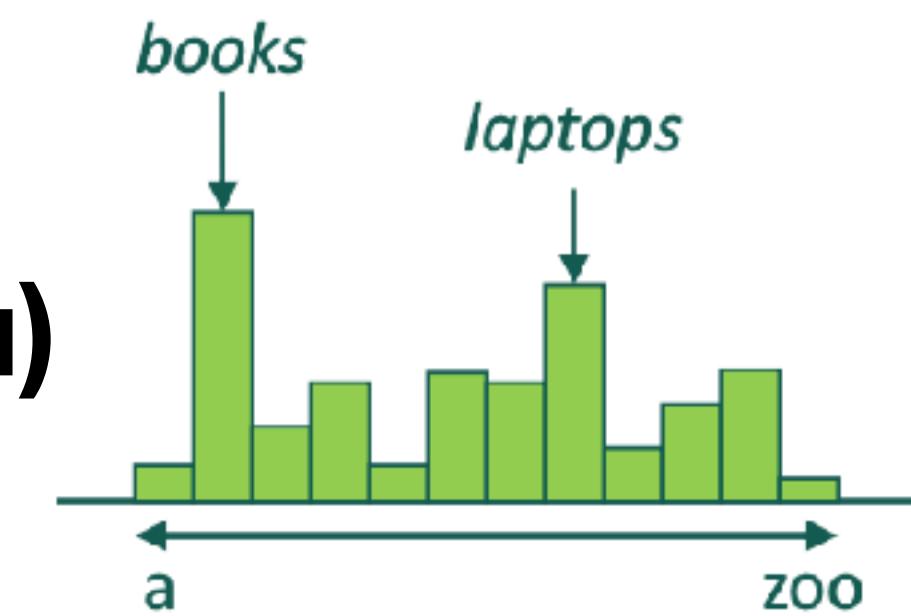
Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим



На выходе 5го шага: $p(x_5 | \text{the}, \text{students}, \text{opened}, \text{their}), x_5 \in \text{Vocab}$

Каждый шаг - классификация (выбираем слово из словаря)



Рекуррентные нейросети (RNN)

$$p(x_5 | \text{the}, \text{students}, \text{opened}, \text{their}) = \text{Softmax}(\text{Linear}(h^t))$$

h^t - представление (энкодинг) *the, students, opened, their*

$$\hat{y}^{(4)} = P(\mathbf{x}^{(5)} | \text{the students opened their})$$

books

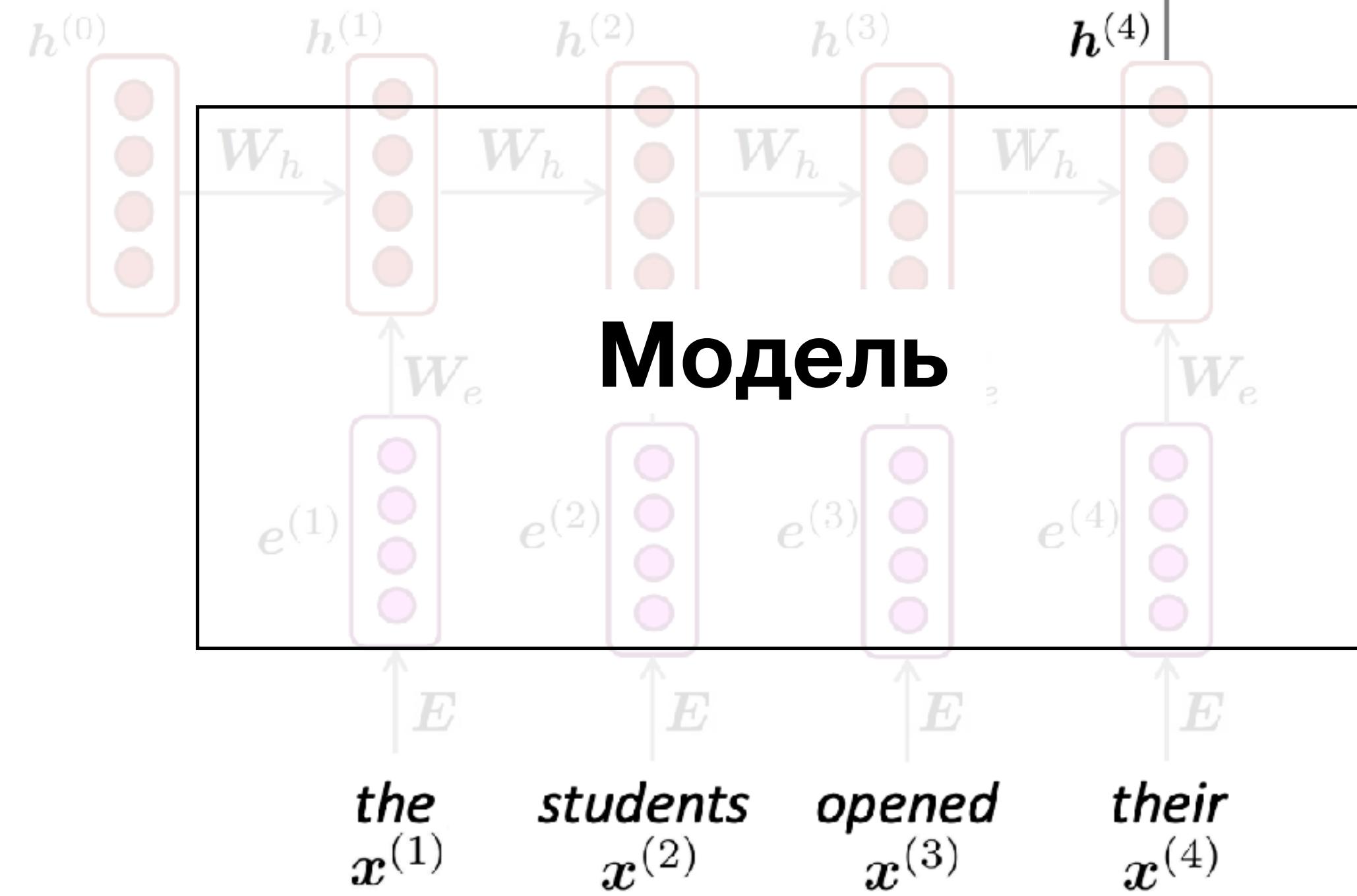
laptops

zoo

a

zoo

$$U$$

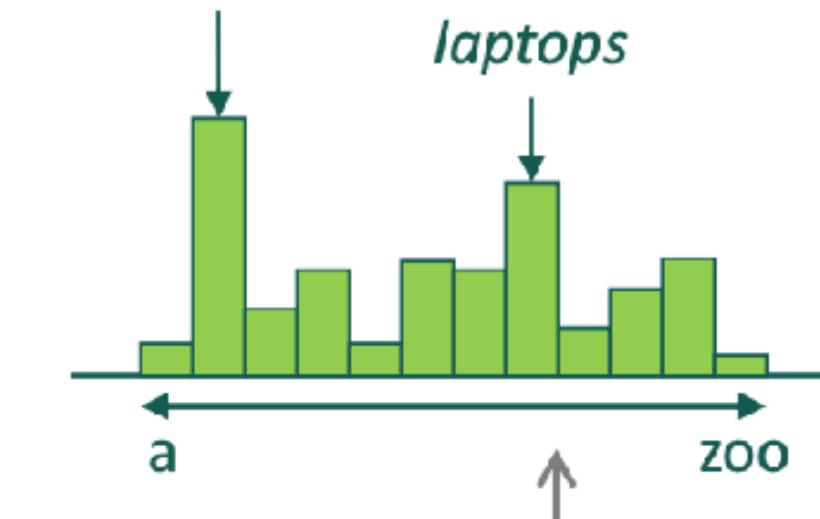


Рекуррентные нейросети (RNN)

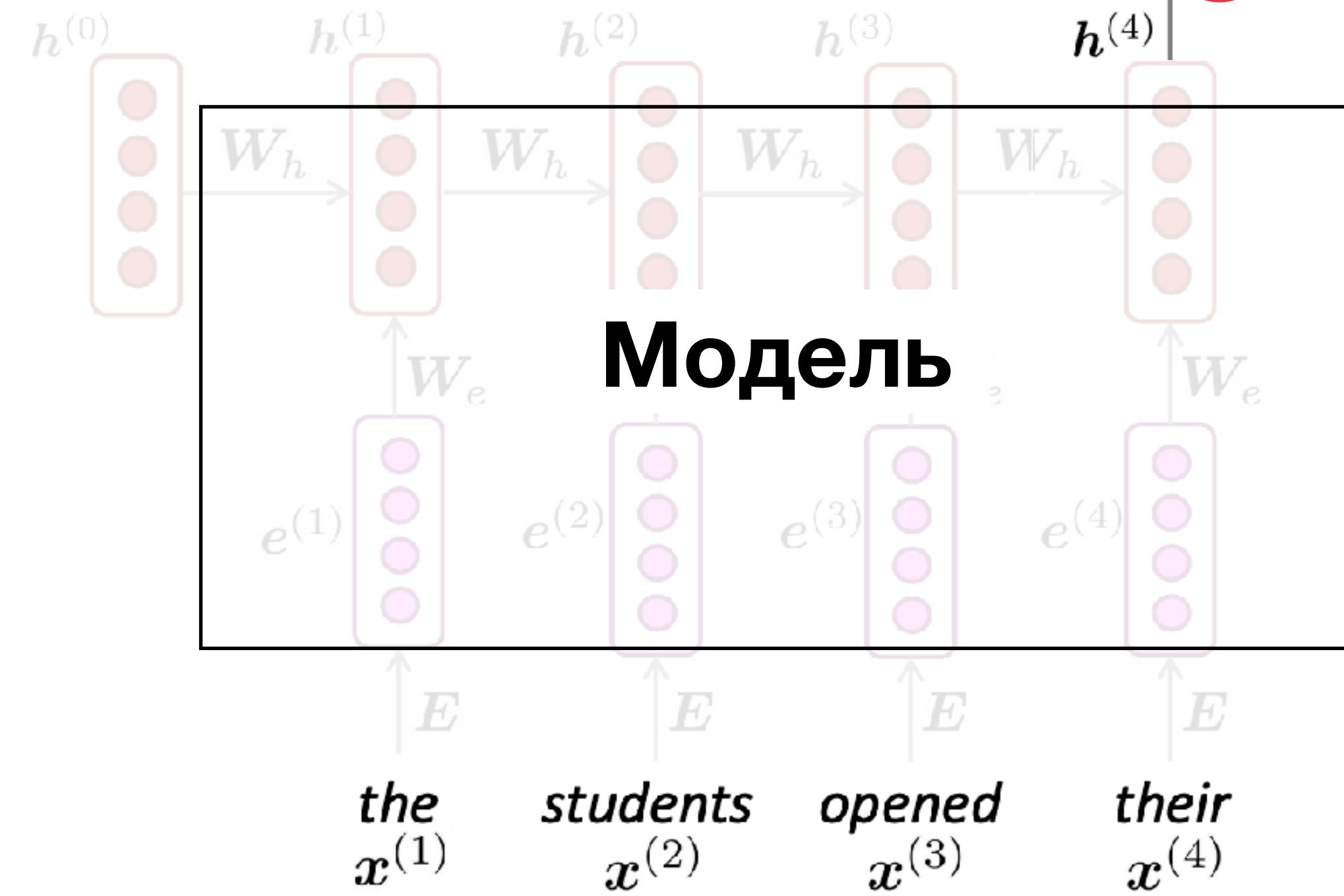
$$p(x_5 | \text{the}, \text{students}, \text{opened}, \text{their}) = \text{Softmax}(Uh^t)$$

h^t - представление (энкодинг) *the, students, opened, their*

$$\hat{y}^{(4)} = P(\mathbf{x}^{(5)} | \text{the students opened their})$$



\mathbf{U} - веса Linear



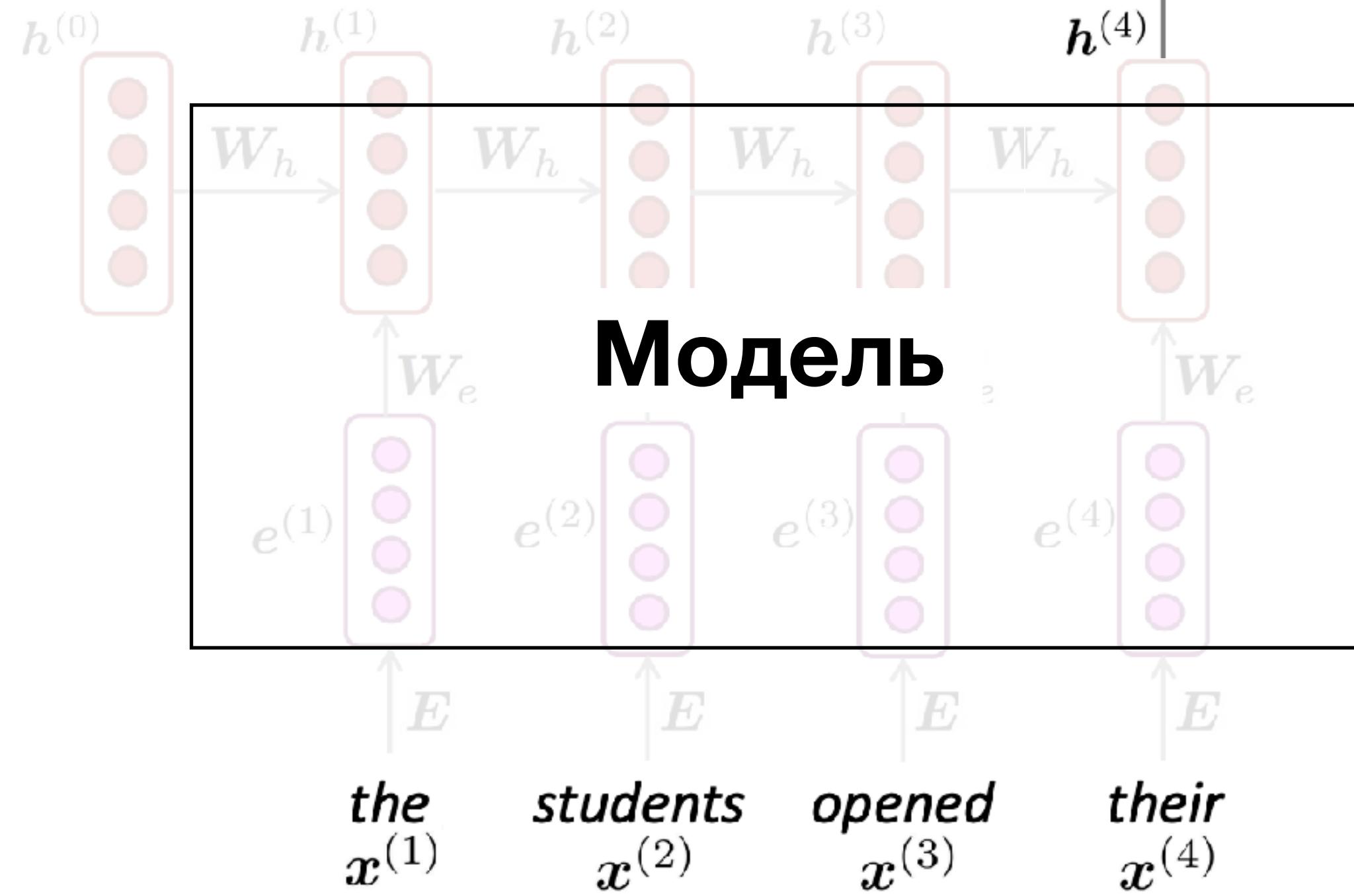
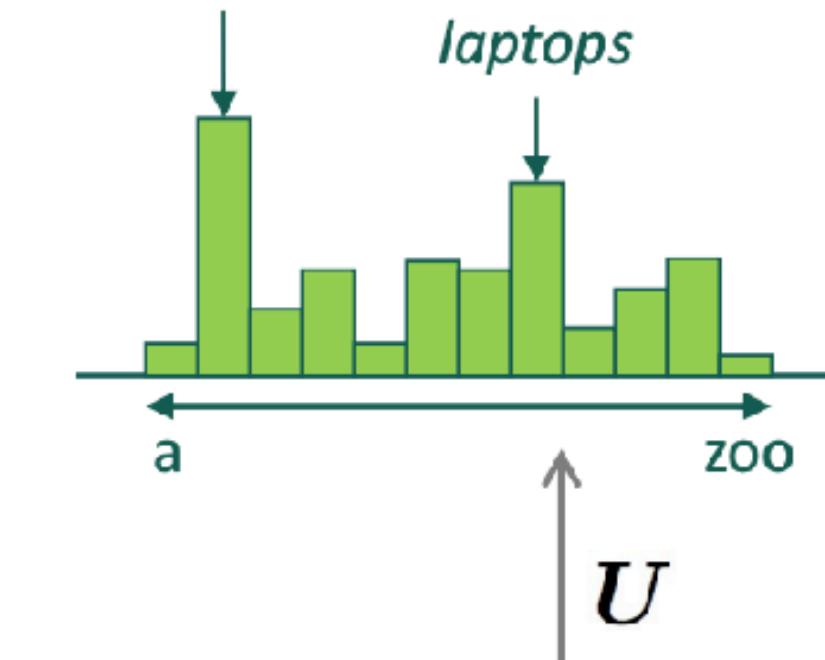
Рекуррентные нейросети (RNN)

$$p(x_5 | \text{the}, \text{students}, \text{opened}, \text{their}) = \text{Softmax}(Uh^t)$$

h^t - представление (энкодинг) *the, students, opened, their*

Как получить представления?

$$\hat{y}^{(4)} = P(\mathbf{x}^{(5)} | \text{the students opened their})$$



[Image credit](#)

Рекуррентные нейросети (RNN)

$$p(x_5 | \text{the}, \text{students}, \text{opened}, \text{their}) = \text{Softmax}(Uh^t)$$

h^t - представление (энкодинг) *the, students, opened, their*

Как получить представления?

Контекст (предыдущие слова) - может быть очень длинным!

Модель должна уметь обрабатывать и 1, и 100 слов

“As he crossed toward the pharmacy at the corner he involuntarily turned his head because of a burst of light that had ricocheted from his temple, and saw, with that quick smile with which we greet a rainbow or a rose, a blindingly white parallelogram of sky being unloaded from the van —a dresser with mirrors across which, as across a cinema screen, passed a flawlessly clear reflection of boughs sliding and swaying not arboreally, but with a human vacillation, produced by the nature of those who were carrying this sky, these boughs, this gliding façade.”

Рекуррентные нейросети (RNN)

$$p(x_5 | \text{the}, \text{students}, \text{opened}, \text{their}) = \text{Softmax}(Uh^t)$$

h^t - представление (энкодинг) *the, students, opened, their*

Как получить представления?

Контекст (предыдущие слова) - может быть очень длинным!

Модель должна уметь обрабатывать и 1, и 100 слов

Идея: будем сохранять историю в “кэше” (специальный вектор, ***hidden state***)

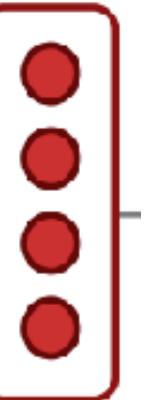
Рекуррентные нейросети (RNN)

Шаг 0: инициализируем “кэш” - $h^0 = 0$

Рекуррентные нейросети (RNN)

Шаг 1: пусть первое слово известно - *the*

Хотим предсказать второе - *students*

$$h^0 = 0$$


$$\begin{matrix} \textit{the} \\ x^{(1)} \end{matrix}$$

[Image credit](#)

Рекуррентные нейросети (RNN)

Шаг 1: пусть первое слово известно - *the*

Хотим предсказать второе - *students*

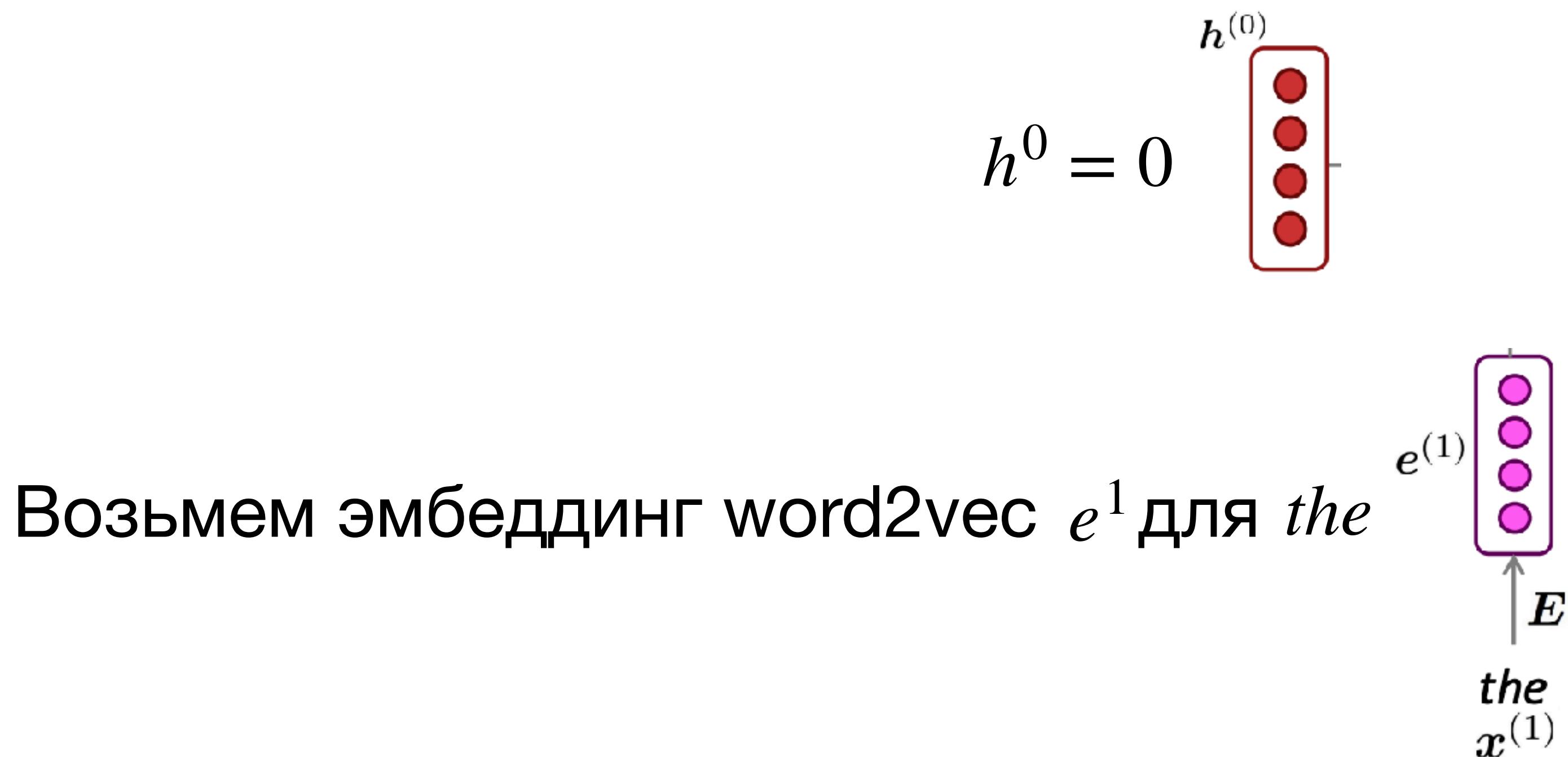


Image credit

Рекуррентные нейросети (RNN)

Шаг 1: пусть первое слово известно - *the*

Хотим предсказать второе - *students*

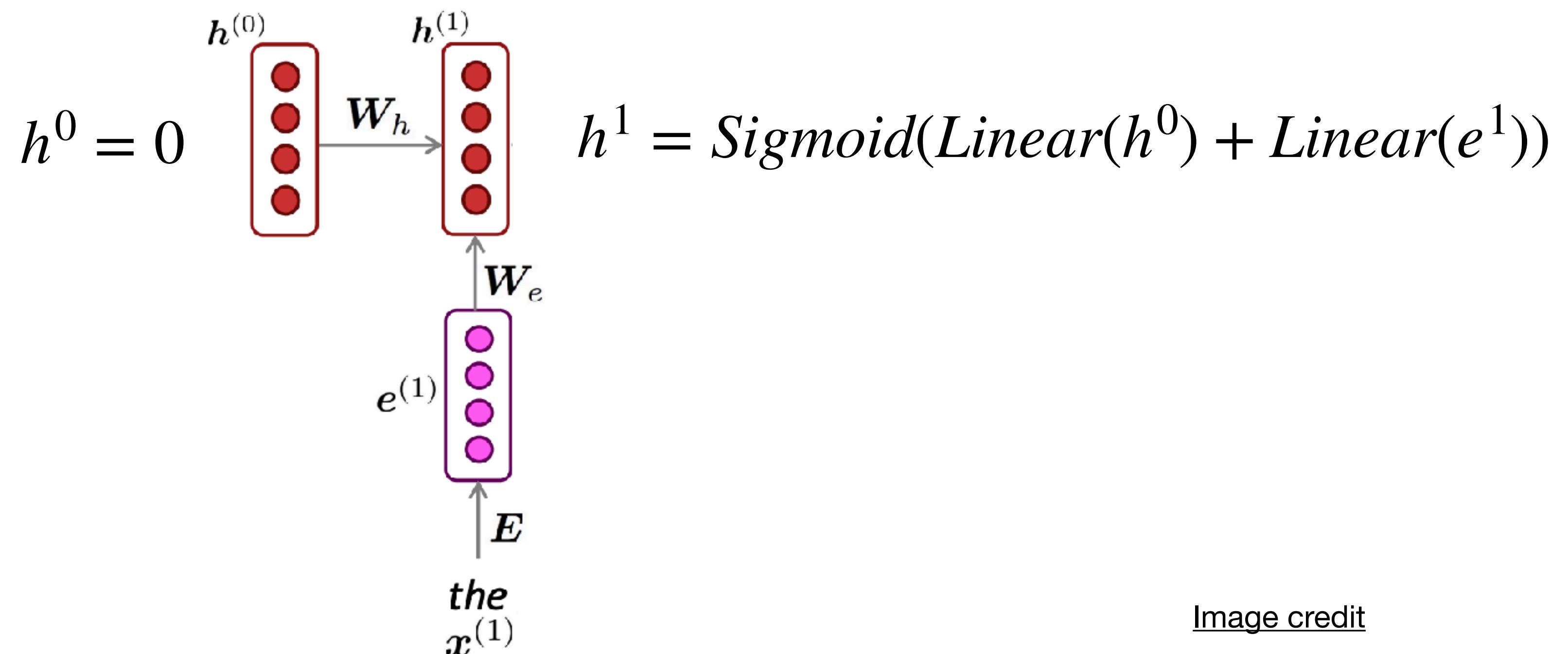
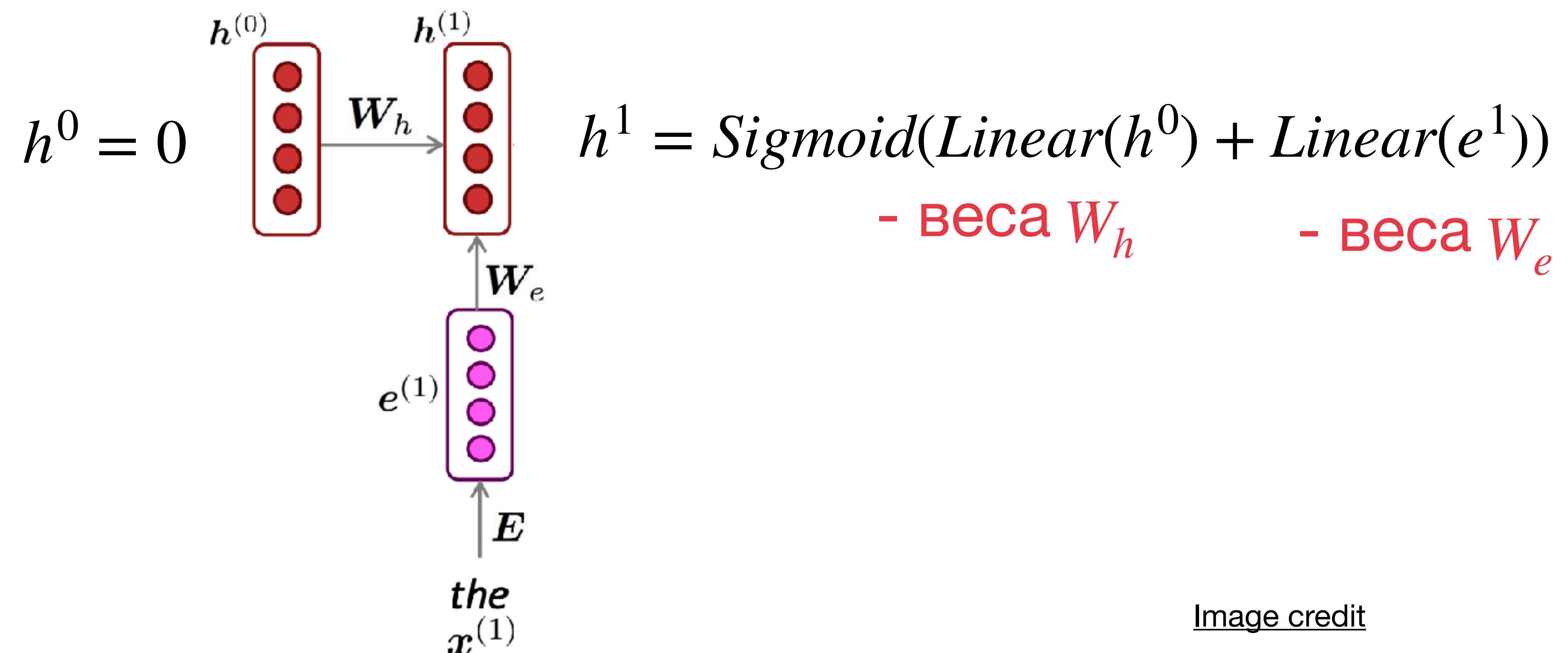


Image credit

Рекуррентные нейросети (RNN)

Шаг 1: пусть первое слово известно - *the*

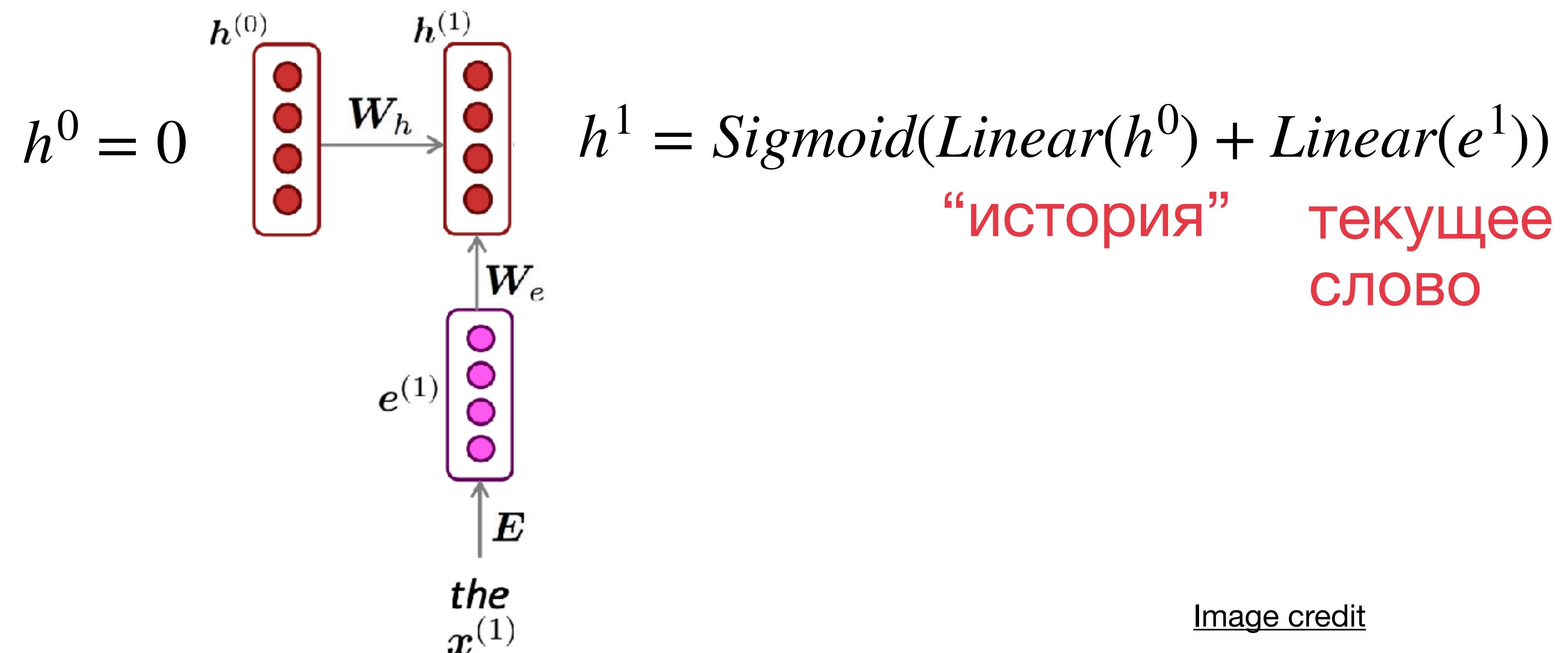
Хотим предсказать второе - *students*



Рекуррентные нейросети (RNN)

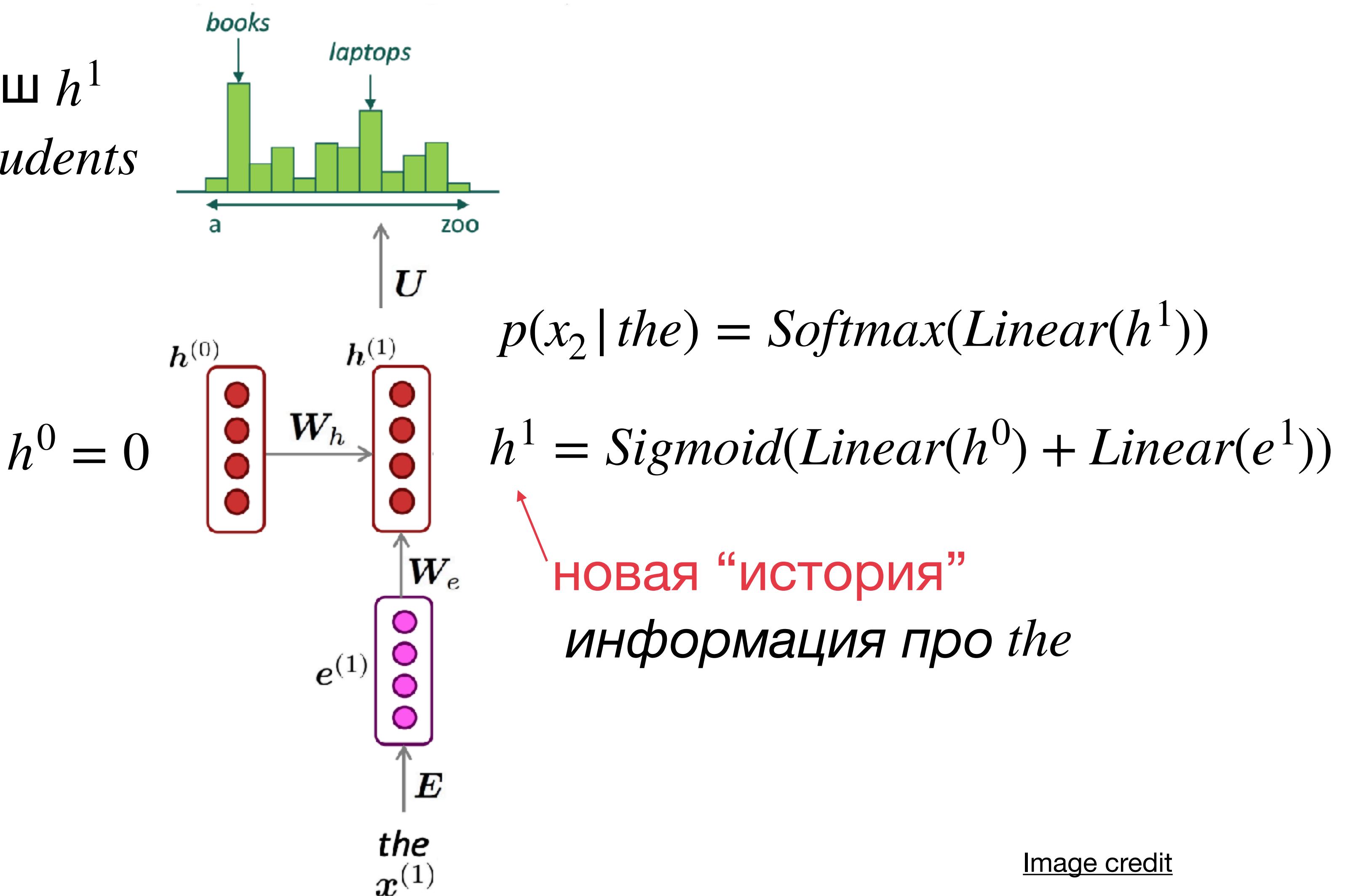
Шаг 1: пусть первое слово известно - *the*

Хотим предсказать второе - *students*



Рекуррентные нейросети (RNN)

Шаг 1: пересчитываем кэш h^1
предсказываем $students$

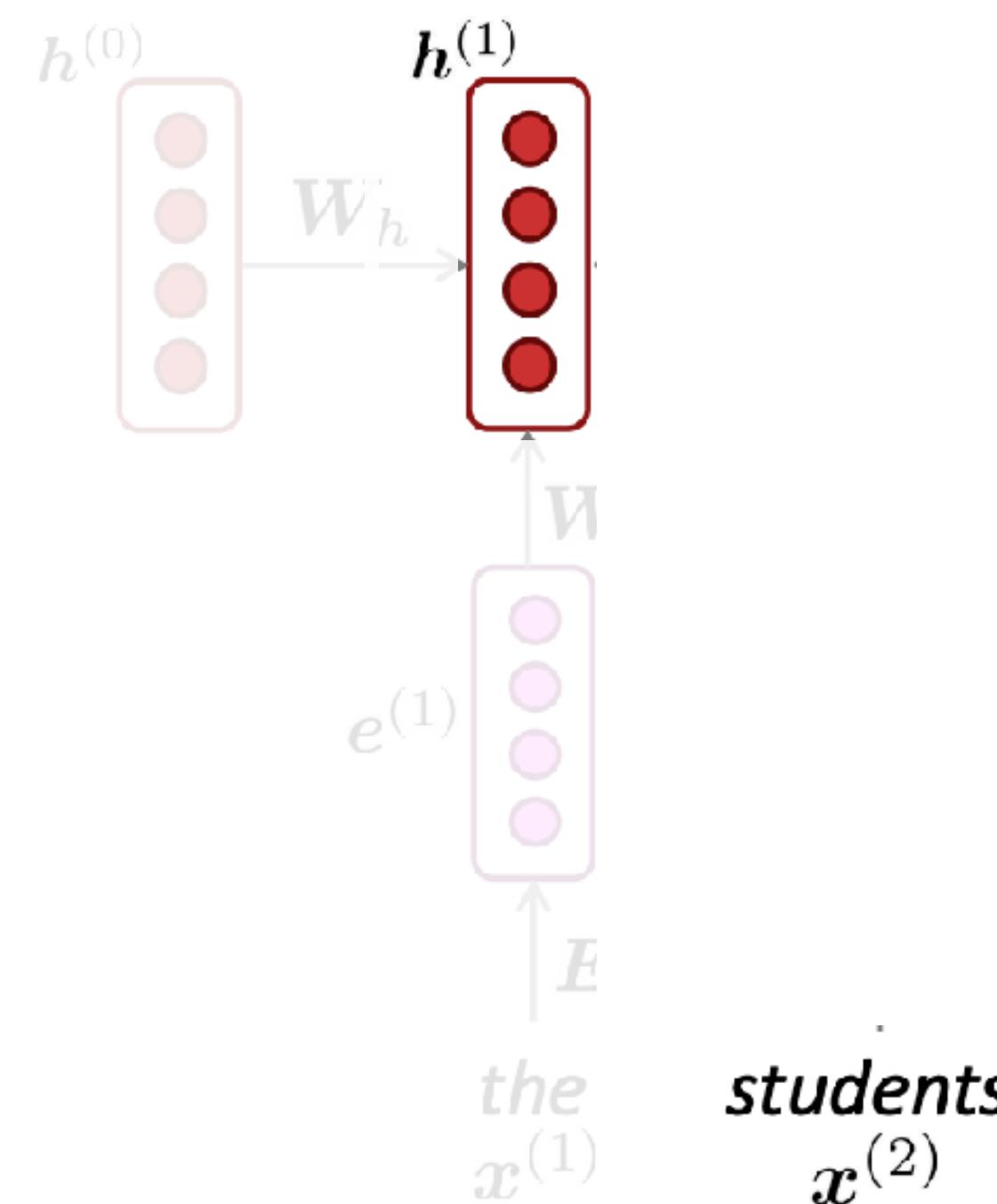


Рекуррентные нейросети (RNN)

Шаг 2: есть h^1 (информация про *the*)

текущее слово - *students*

хотим предсказать *opened*



[Image credit](#)

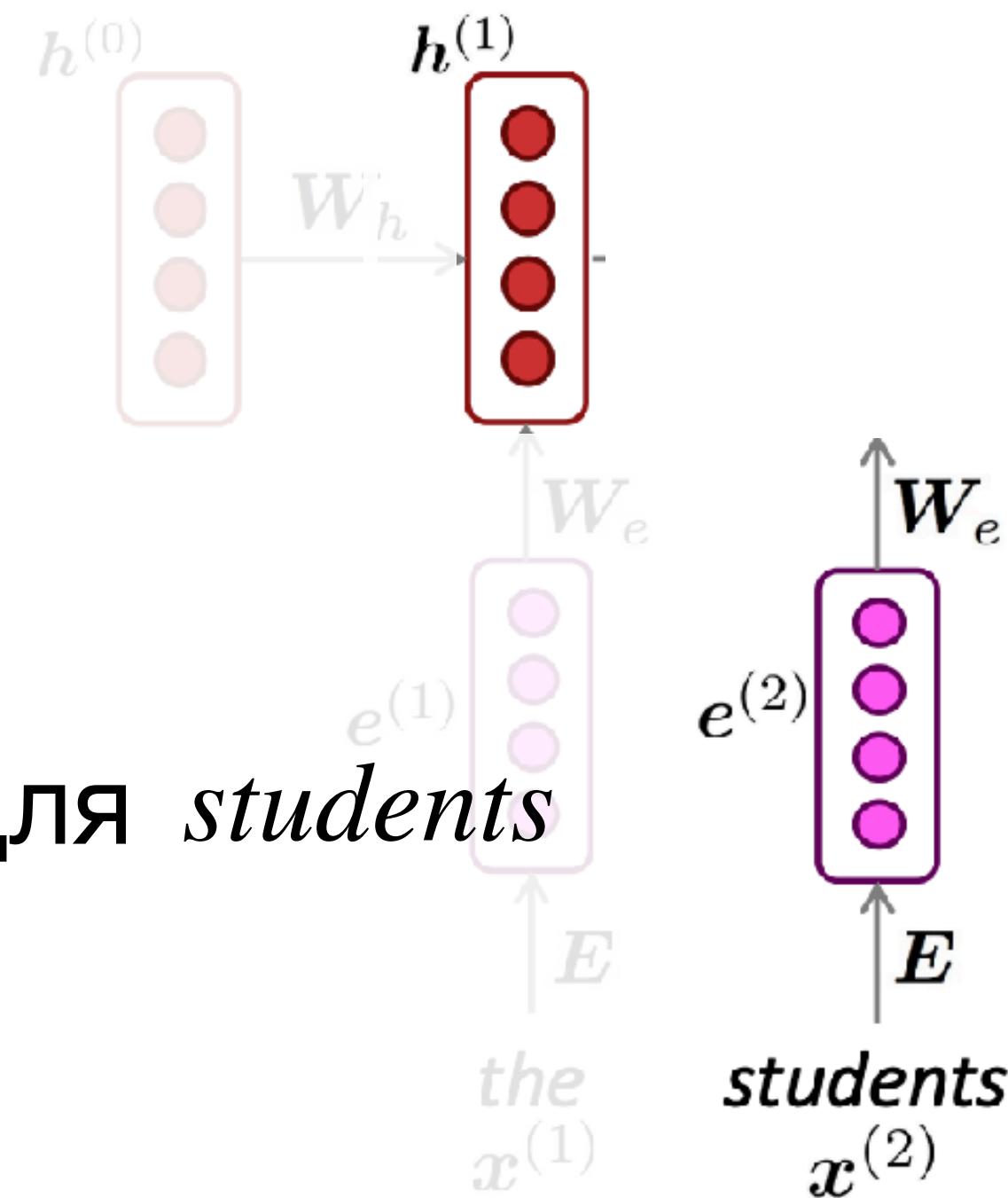
Рекуррентные нейросети (RNN)

Шаг 2: есть h^1 (информация про *the*)

текущее слово - *students*

хотим предсказать *opened*

Возьмем эмбеддинг word2vec e^2 для *students*



[Image credit](#)

Рекуррентные нейросети (RNN)

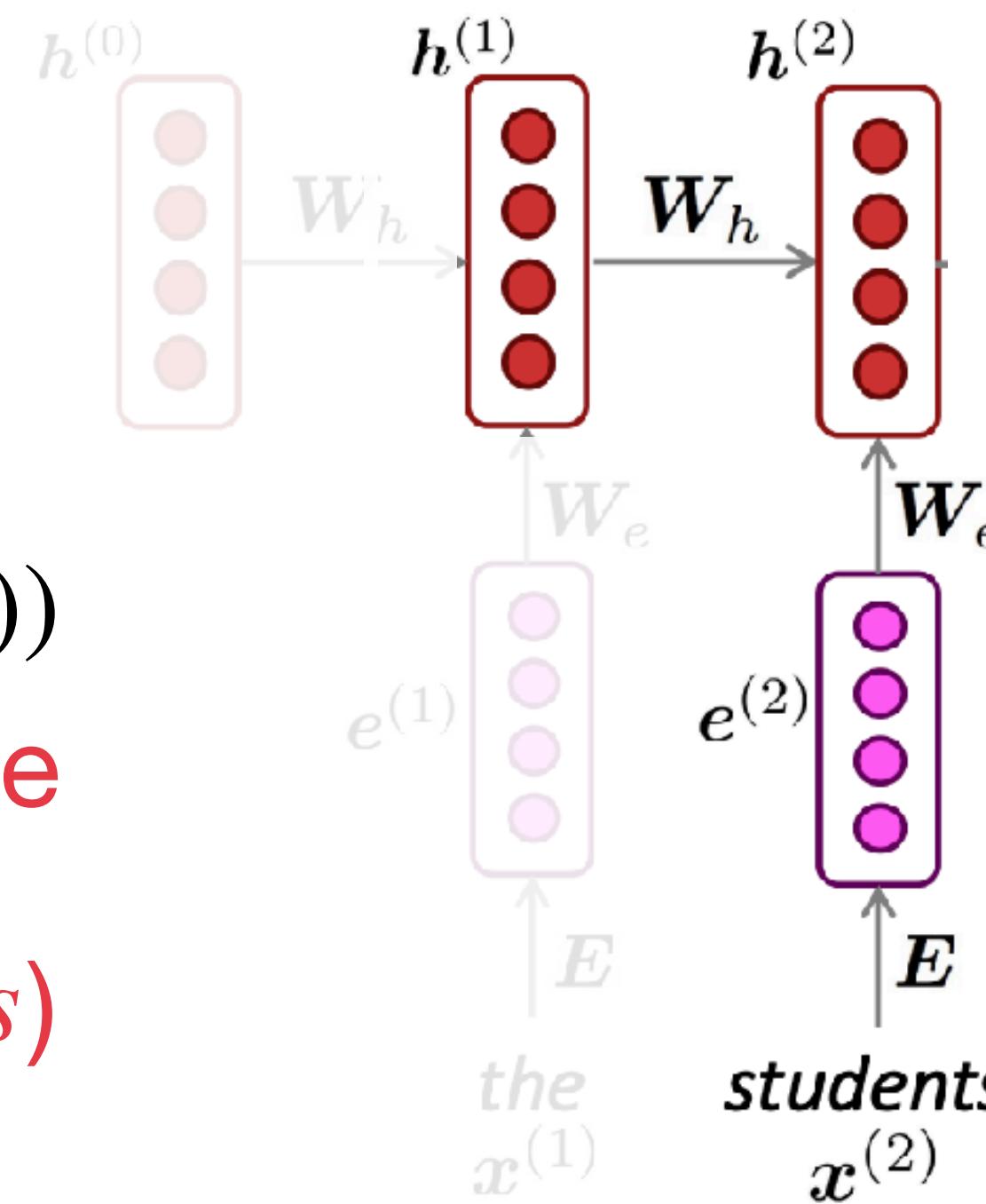
Шаг 2: есть h^1 (информация про *the*)

текущее слово - *students*

хотим предсказать *opened*

$$h^2 = \text{Sigmoid}(\text{Linear}(h^1) + \text{Linear}(e^2))$$

“история” текущее
(the) слово
 (students)



[Image credit](#)

Рекуррентные нейросети (RNN)

Шаг 2: есть h^1 (информация про *the*)

текущее слово - *students*

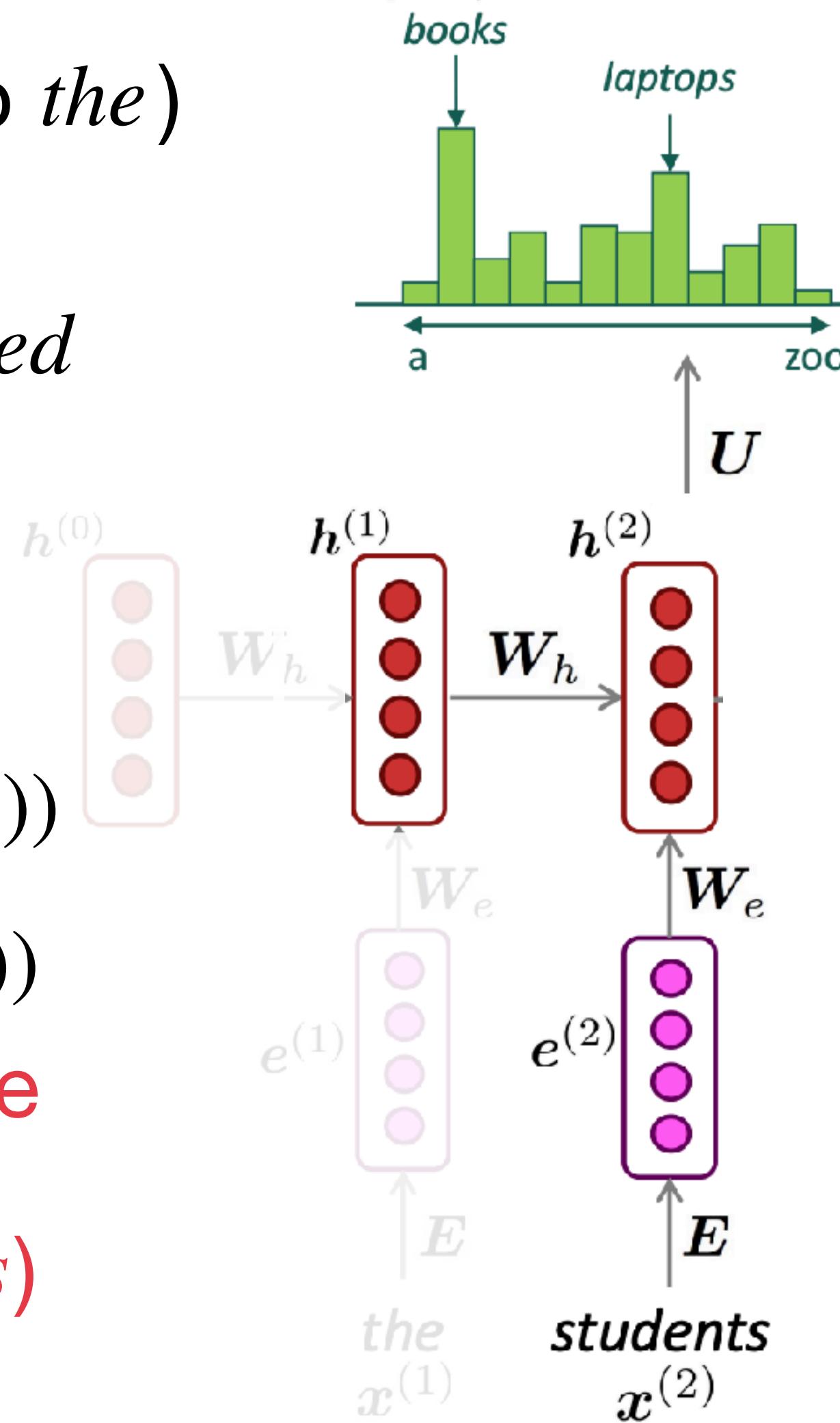
хотим предсказать *opened*

$$p(x_3 | \text{the}, \text{students}) = \text{Softmax}(\text{Linear}(h^2))$$

$$h^2 = \text{Sigmoid}(\text{Linear}(h^1) + \text{Linear}(e^2))$$

“история”
(*the*)

текущее
слово
(*students*)



[Image credit](#)

Рекуррентные нейросети (RNN)

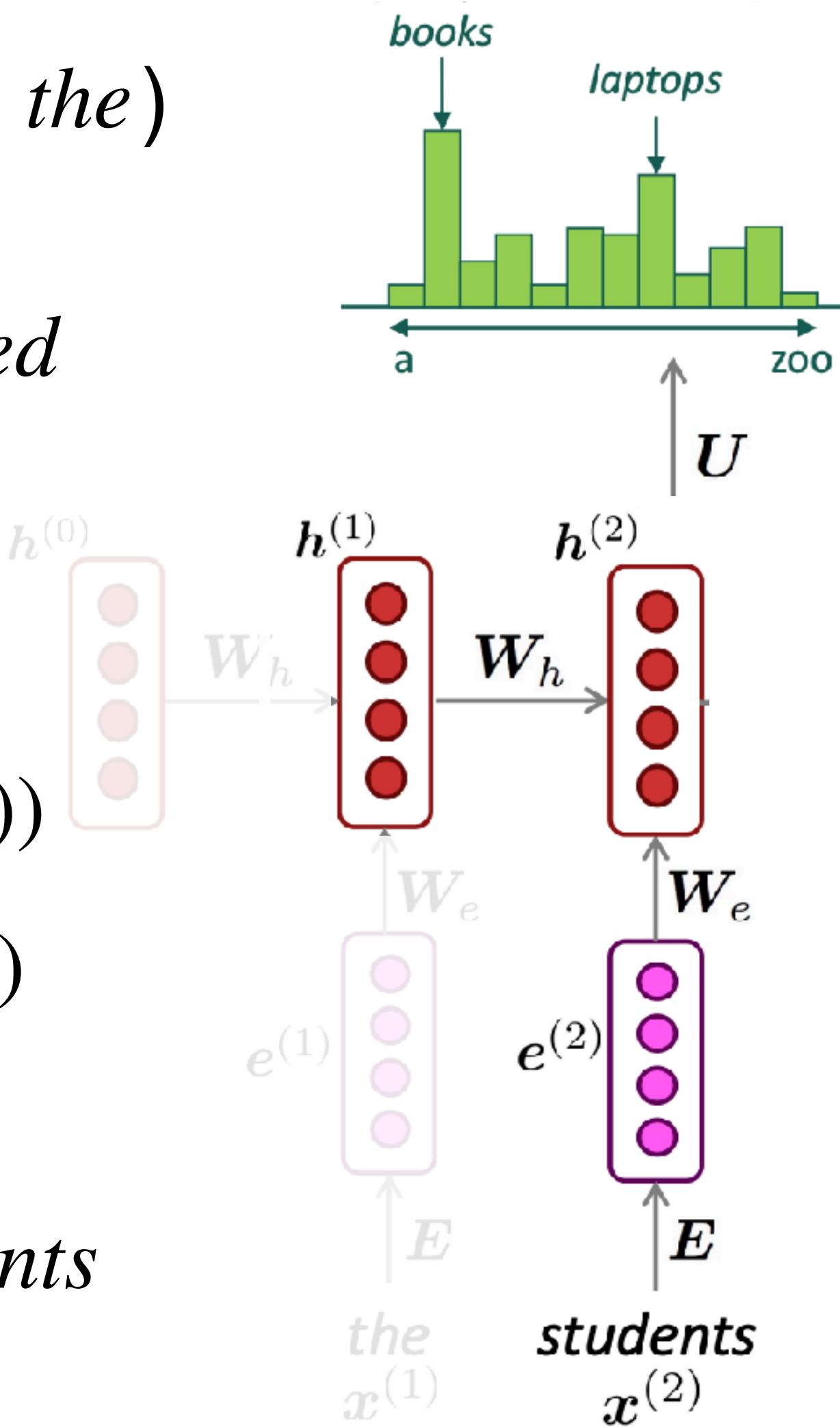
Шаг 2: есть h^1 (информация про *the*)
текущее слово - *students*
хотим предсказать *opened*

$$p(x_3 | \text{the}, \text{students}) = \text{Softmax}(\text{Linear}(h^2))$$

$$h^2 = \text{Sigmoid}(\text{Linear}(h^1) + \text{Linear}(e^2))$$

новая “история”

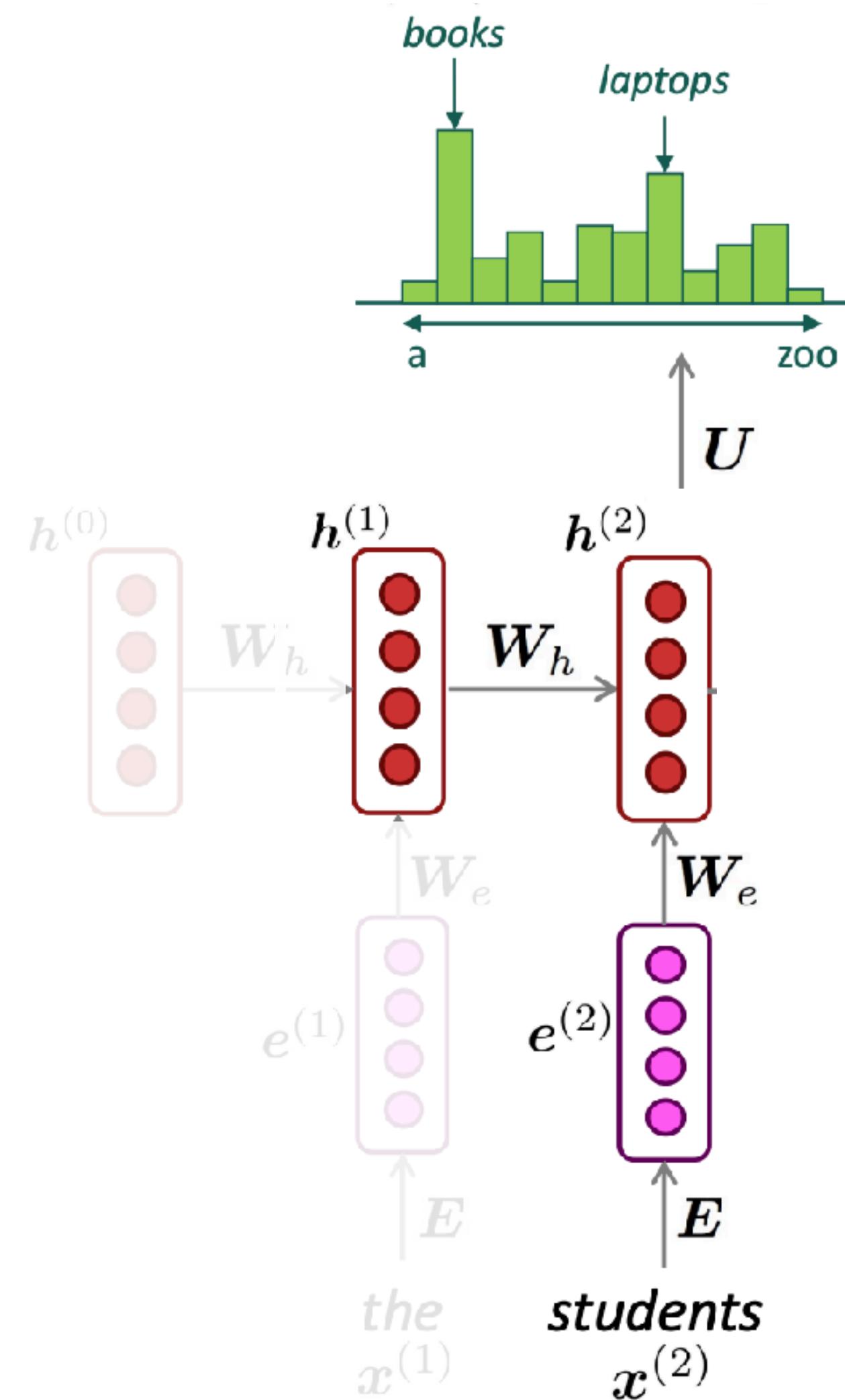
информация про *the* (h^1), *students*



[Image credit](#)

Рекуррентные нейросети (RNN)

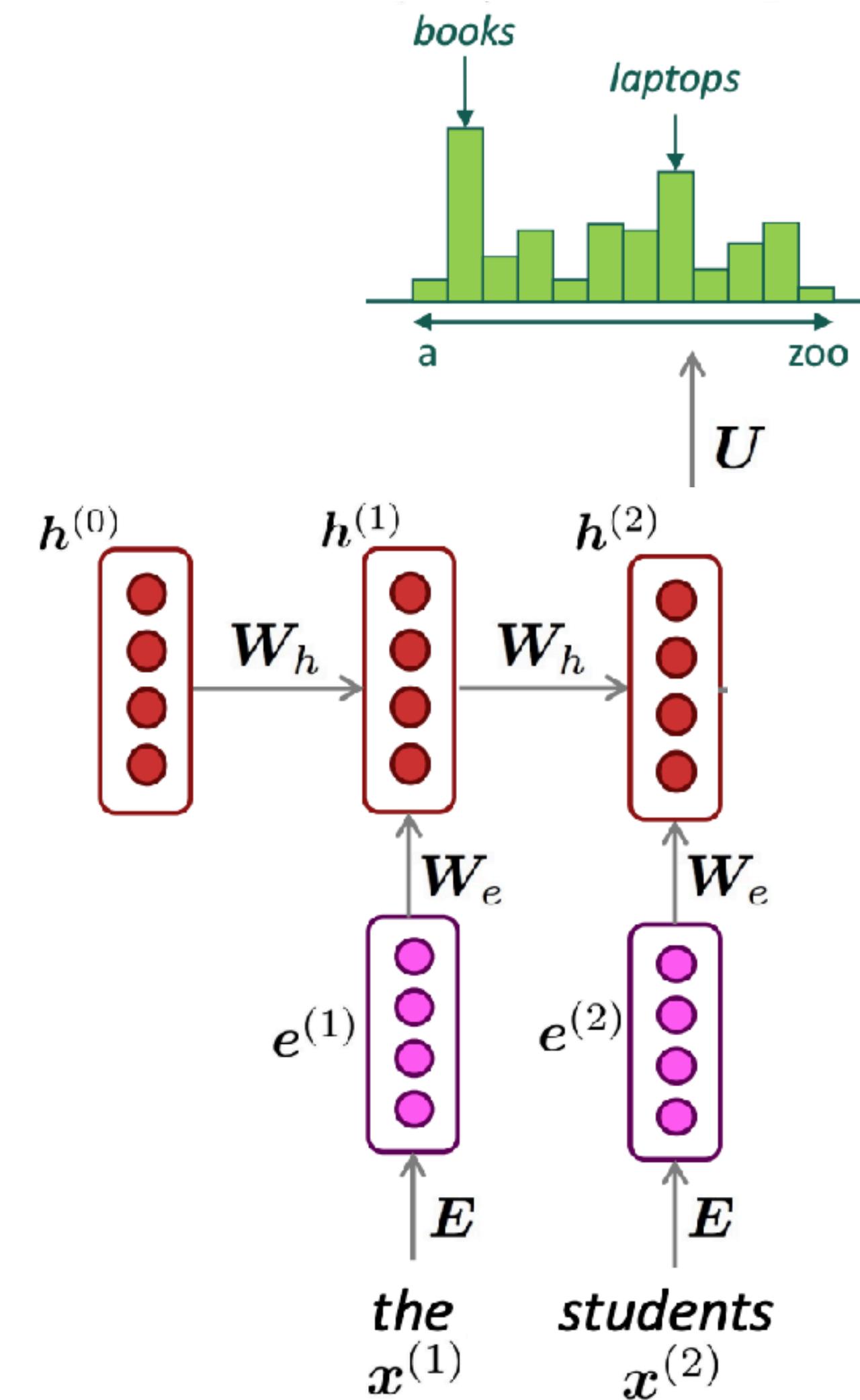
Шаг 2: пересчитываем кэш h^2
предсказываем *opened*



[Image credit](#)

Рекуррентные нейросети (RNN)

Два шага RNN



[Image credit](#)

Рекуррентные нейросети (RNN)

output distribution

$$\hat{y}^{(t)} = \text{softmax}(\mathbf{U}\mathbf{h}^{(t)} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden states

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_e \mathbf{e}^{(t)} + \mathbf{b}_1)$$

$\mathbf{h}^{(0)}$ is the initial hidden state

word embeddings

$$\mathbf{e}^{(t)} = \mathbf{E}\mathbf{x}^{(t)}$$

words / one-hot vectors

$$\mathbf{x}^{(t)} \in \mathbb{R}^{|V|}$$

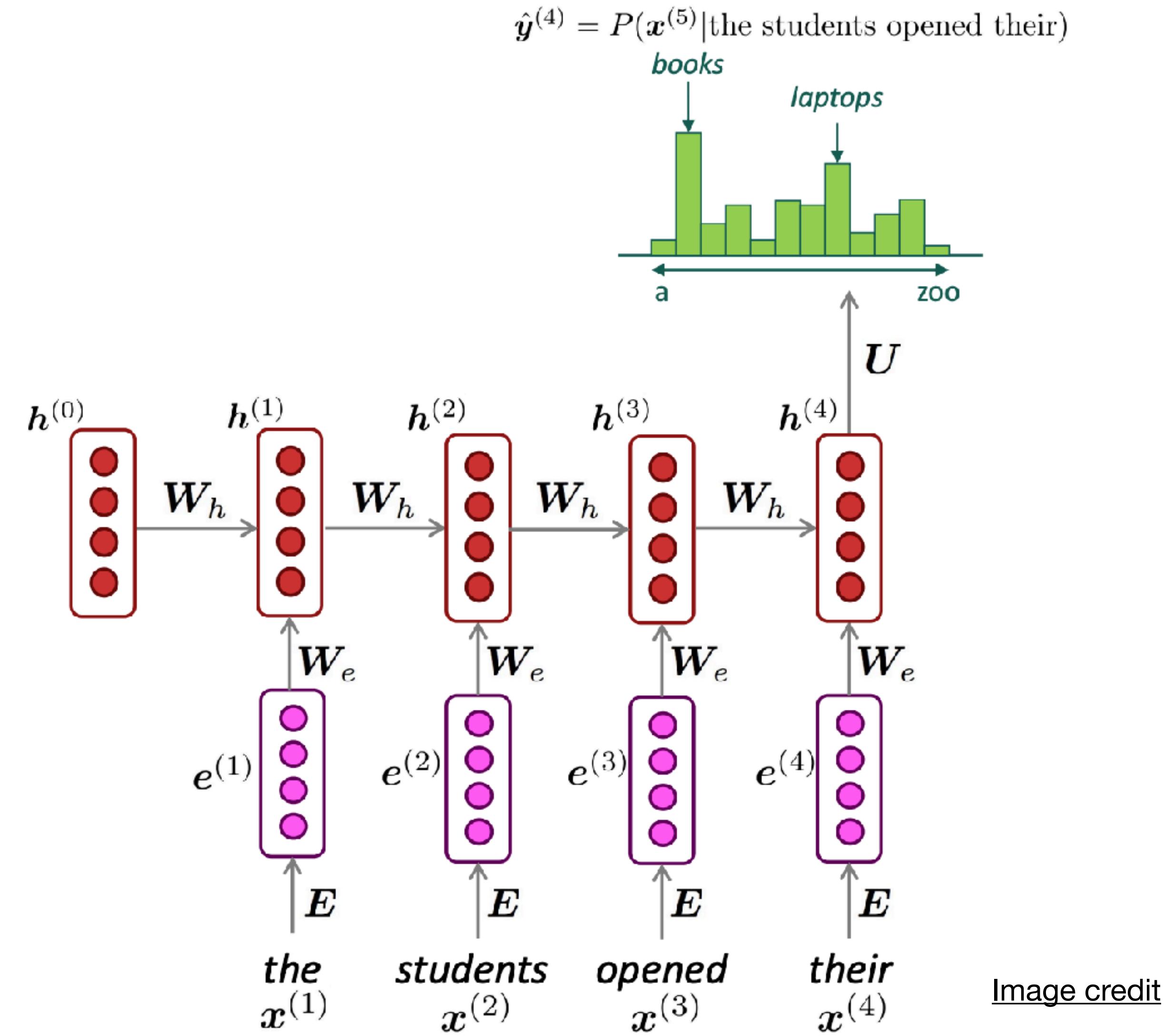
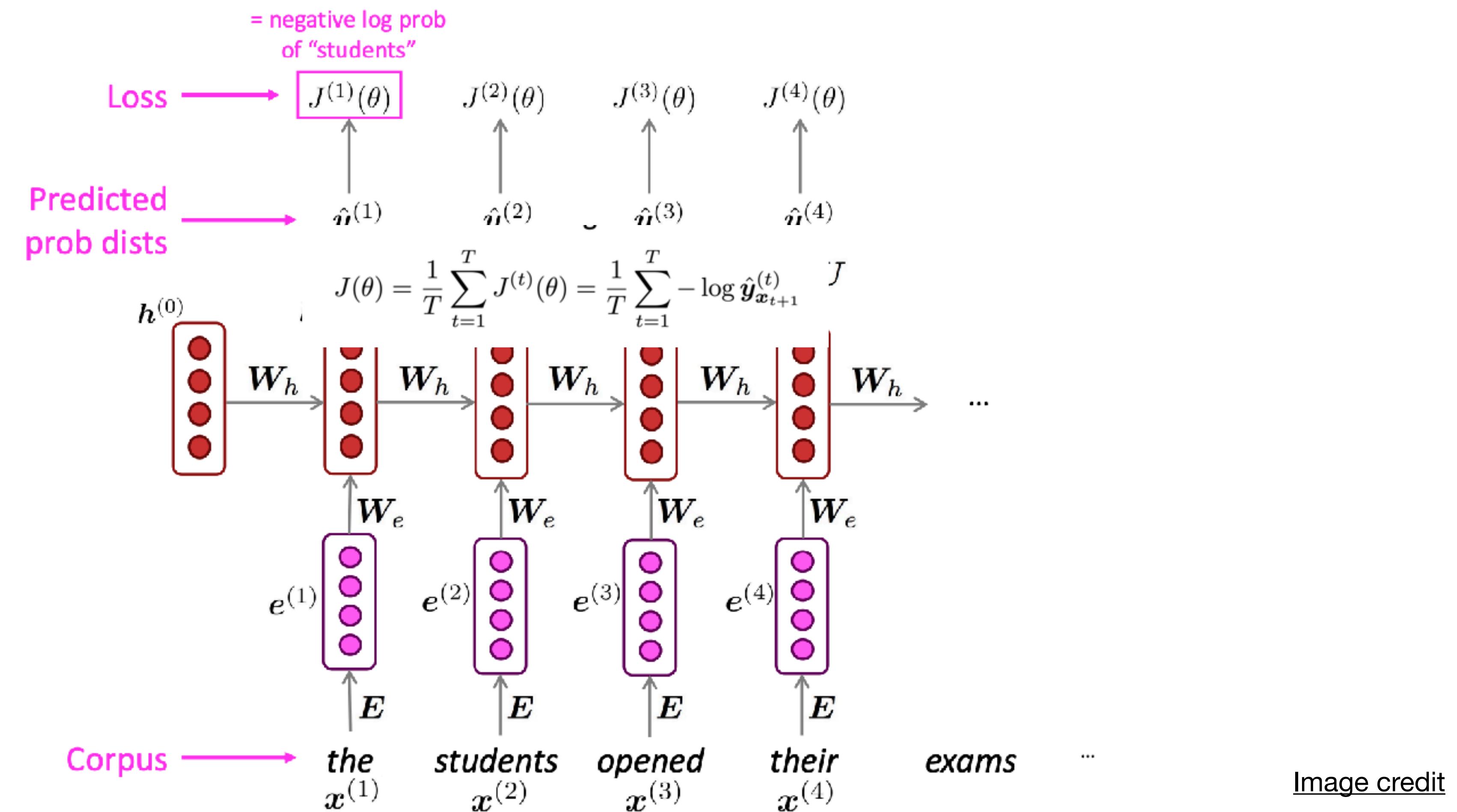


Image credit

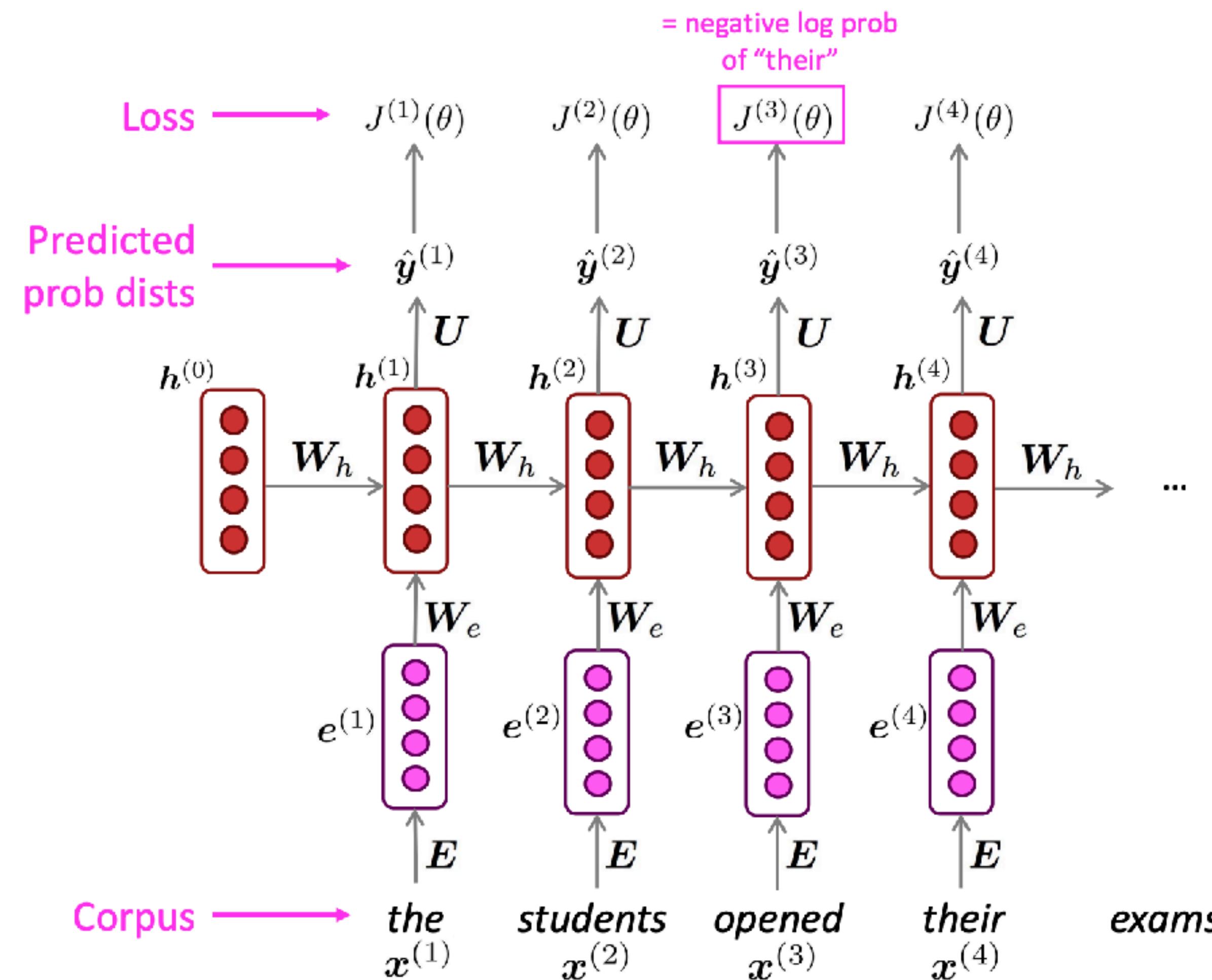
Рекуррентные нейросети (RNN)

Обучение: лосс для классификации на каждом шаге



Рекуррентные нейросети (RNN)

Обучение: лосс для классификации на каждом шаге



[Image credit](#)

Рекуррентные нейросети (RNN)

Обучение: лосс для классификации на каждом шаге

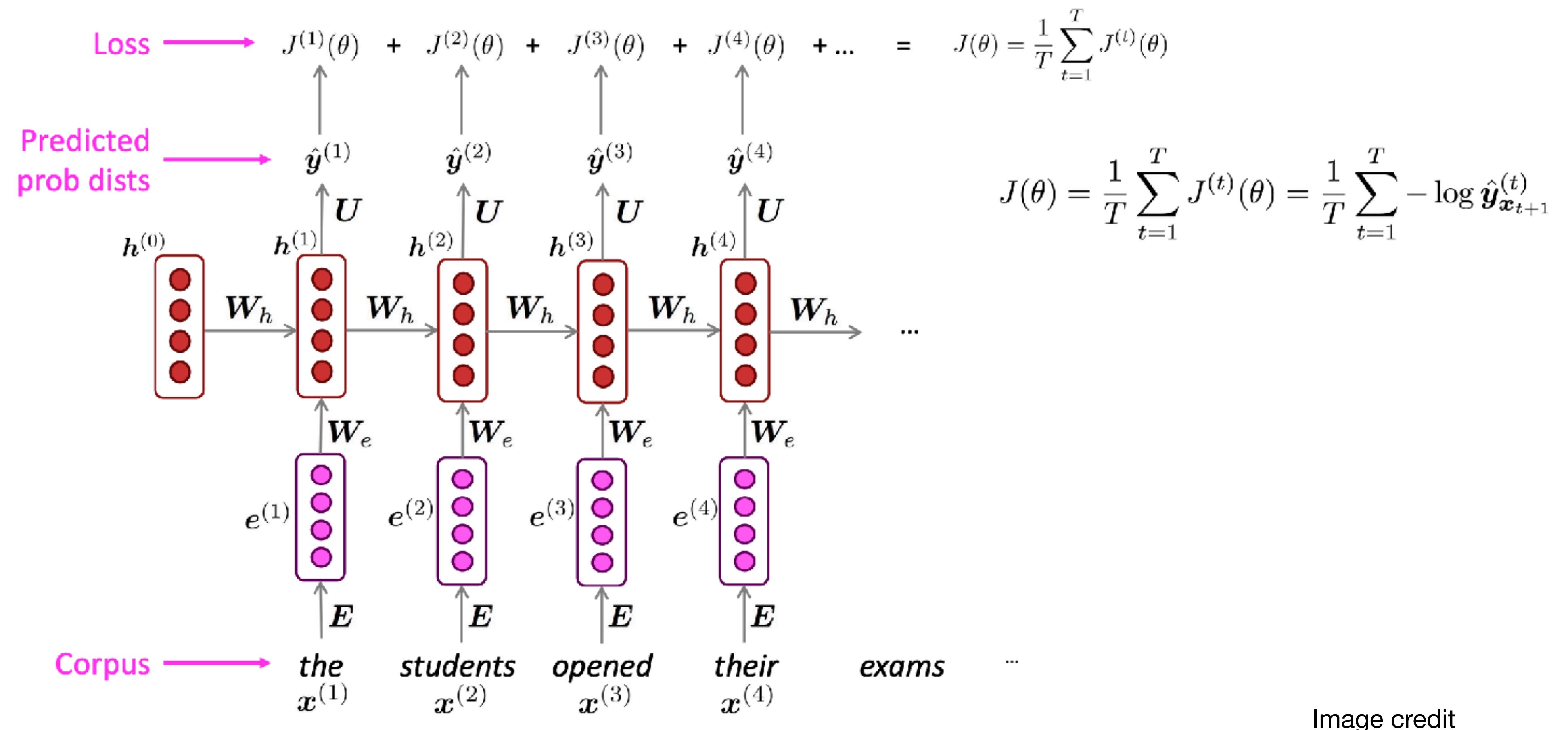
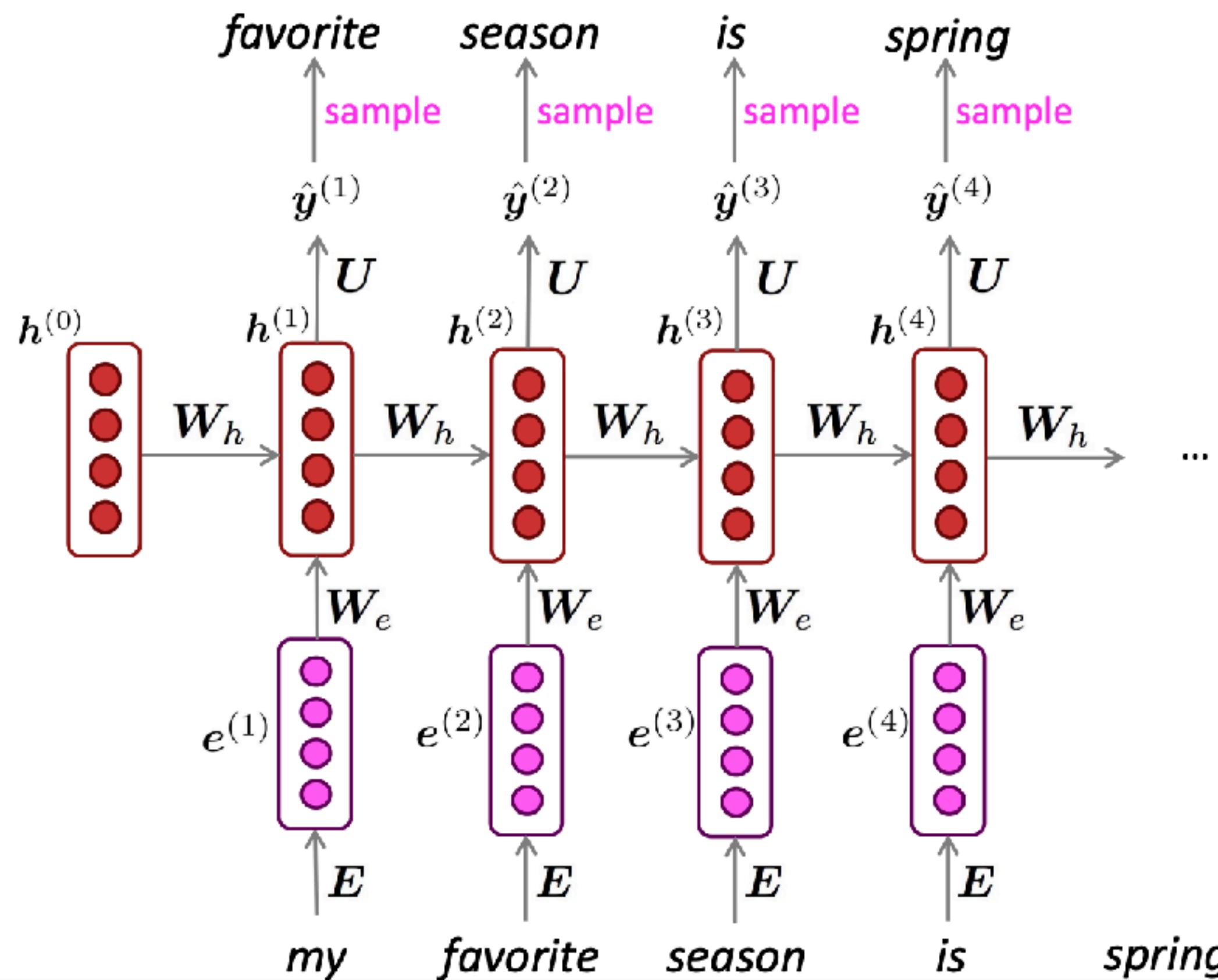


Image credit

Рекуррентные нейросети (RNN)

Генерация текста (language modeling)



“Sorry,” Harry shouted, panicking—“I’ll leave those brooms in London, are they?”

“No idea,” said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry’s shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn’t felt it seemed. He reached the teams too.

Image credit

Рекуррентные нейросети (RNN)

Можно использовать для других задач!

output distribution

$$\hat{y}^{(t)} = \text{softmax}(\mathbf{U}\mathbf{h}^{(t)} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden states

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_e \mathbf{e}^{(t)} + \mathbf{b}_1)$$

$\mathbf{h}^{(0)}$ is the initial hidden state

word embeddings

$$\mathbf{e}^{(t)} = \mathbf{E}\mathbf{x}^{(t)}$$

words / one-hot vectors

$$\mathbf{x}^{(t)} \in \mathbb{R}^{|V|}$$

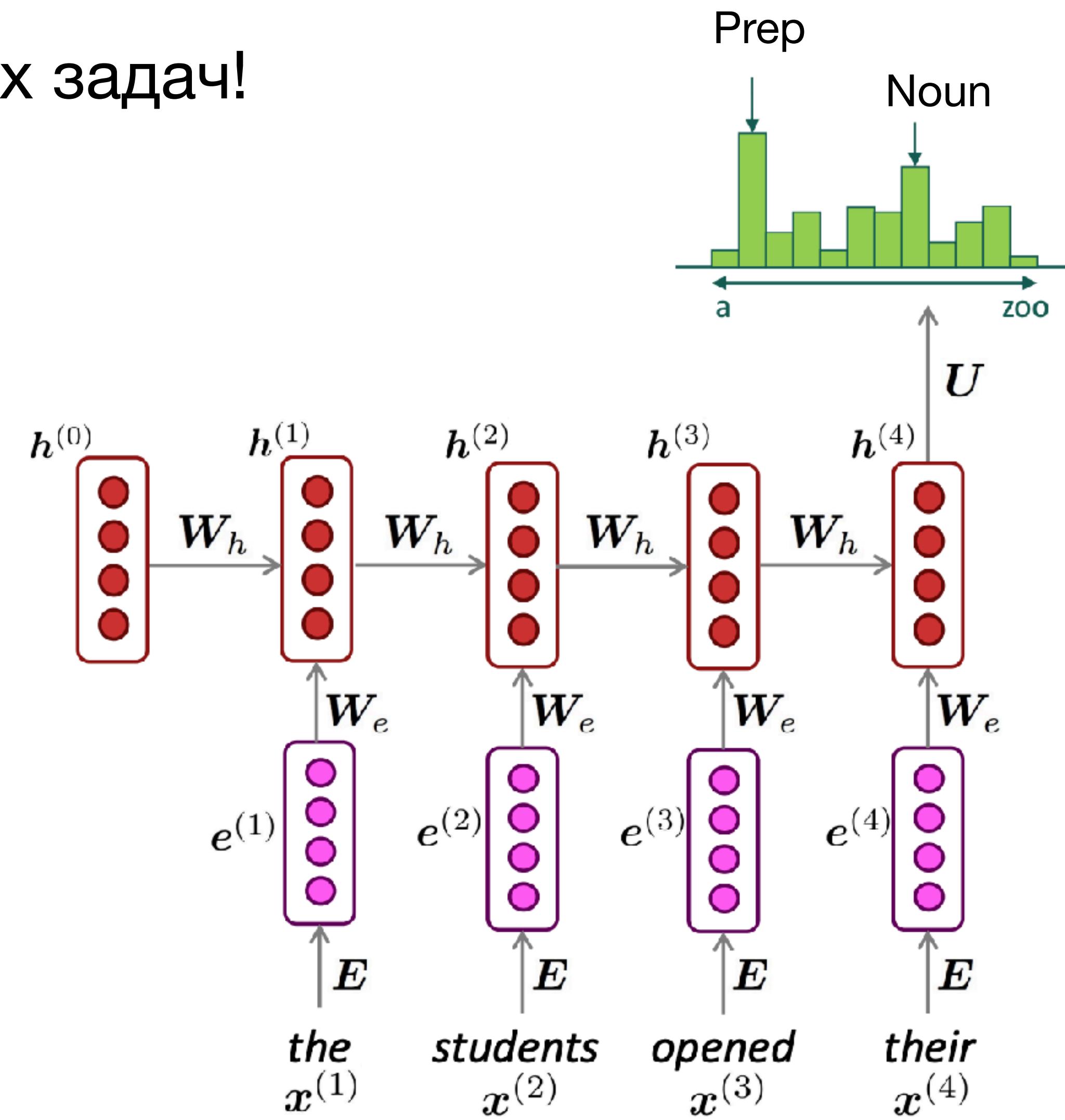


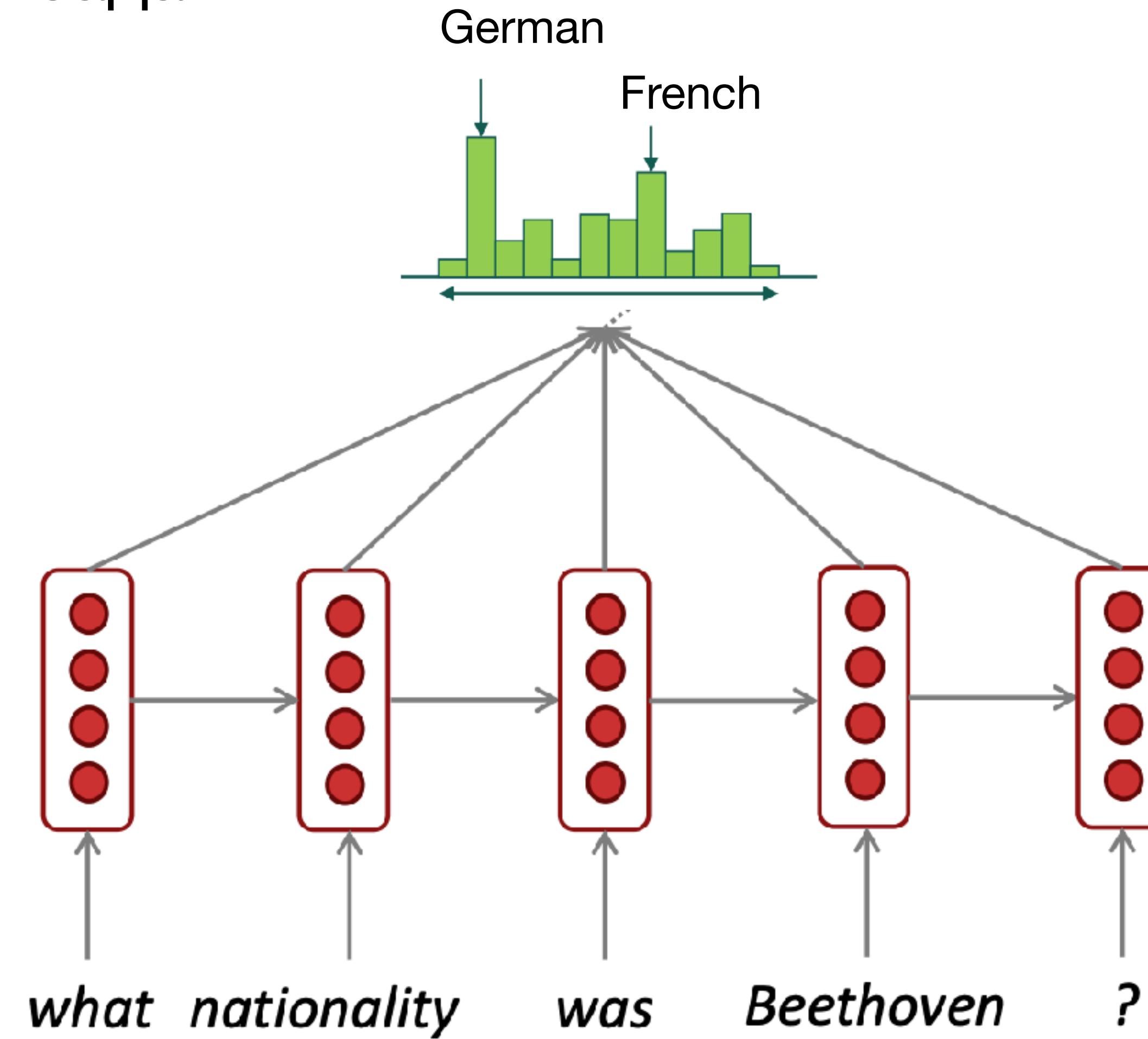
Image credit

Рекуррентные нейросети (RNN)

Можно использовать для других задач!

Sentence classification

Классификация предложений — это задача в области обработки естественного языка (NLP), заключающаяся в определении категории или метки для данного предложения. Это может быть полезно в различных приложениях, таких как анализ тональности, фильтрация спама, автоматическая категоризация текстов и многое другое.



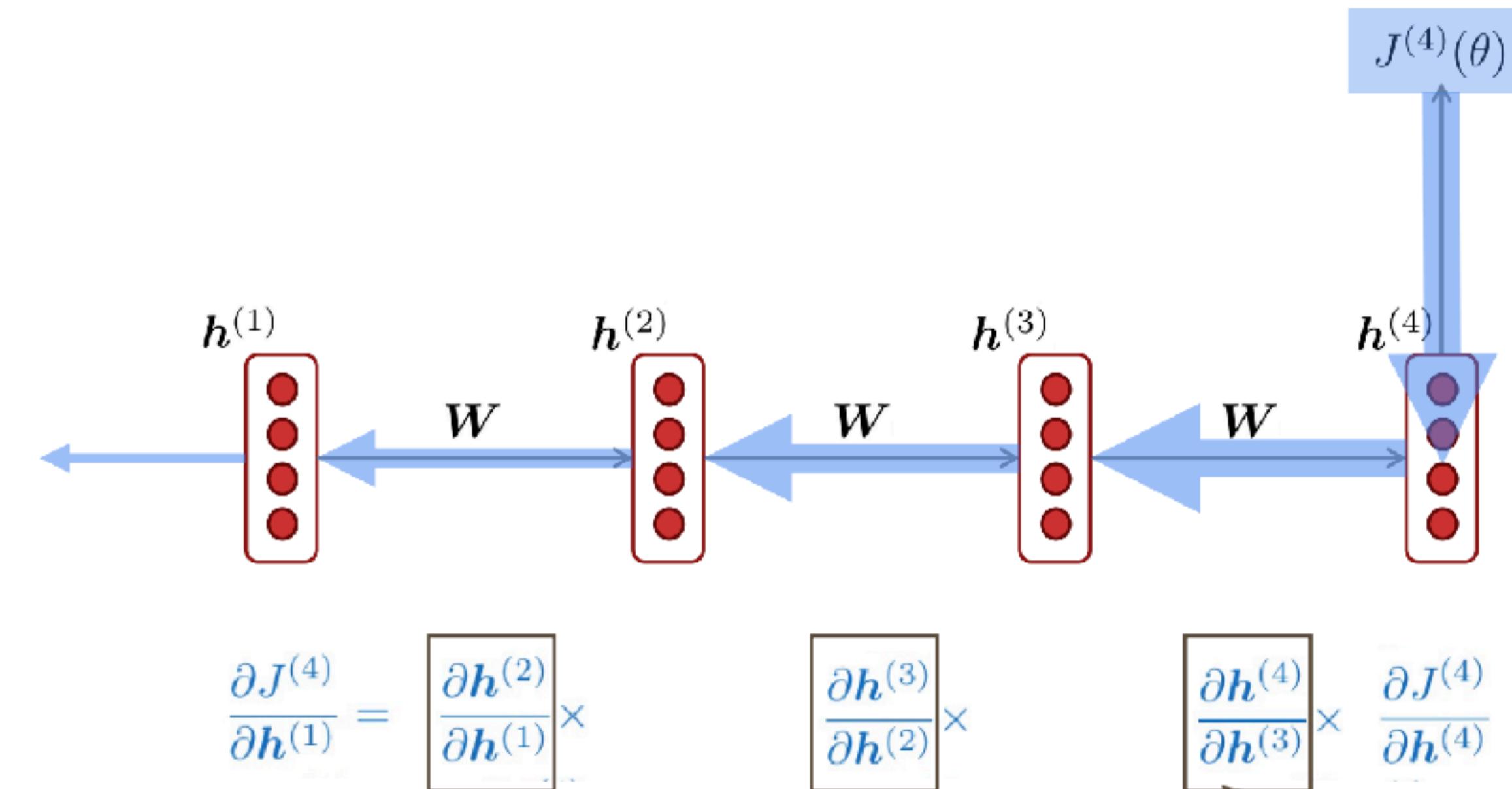
[Image credit](#)

Рекуррентные нейросети (RNN)

Проблемы?

- Медленно
- Vanishing gradients

Vanishing gradients (затухающие градиенты) — это проблема, которая возникает при обучении глубоких нейронных сетей, особенно в рекуррентных нейронных сетях (RNN) и многослойных полносвязных сетях. Суть этой проблемы заключается в том, что градиенты, используемые для обновления весов во время обратного распространения, становятся слишком малыми на ранних слоях сети, и веса перестают обновляться.



LM task: When she tried to print her tickets, she found that the printer was out of toner.
She went to the stationery store to buy more toner. It was very overpriced. After
installing the toner into the printer, she finally printed her _____

[Image credit](#)

Рекуррентные нейросети (RNN)

Проблемы?

- Медленно
- Vanishing gradients
- Exploding gradients? Solution: gradients clipping

Exploding gradients (взрывающиеся градиенты) — это проблема, противоположная затухающим градиентам, возникающая при обучении глубоких нейронных сетей. Взрывающиеся градиенты появляются, когда градиенты становятся слишком большими во время обратного распространения, что приводит к резкому увеличению весов и неустойчивому поведению сети. Эта проблема чаще всего встречается в рекуррентных нейронных сетях (RNN).

Ограничение градиентов (Gradient Clipping): Метод ограничения градиентов заключается в установке максимального порога для значений градиентов. Если градиент превышает заданный порог, он "обрезается" до этого порога. Это позволяет держать градиенты под контролем и избегать их взрывного роста. Например, в PyTorch это можно сделать так:

```
torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
```

Algorithm 1 Pseudo-code for norm clipping

```
 $\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$ 
if  $\|\hat{\mathbf{g}}\| \geq \text{threshold}$  then
     $\hat{\mathbf{g}} \leftarrow \frac{\text{threshold}}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$ 
end if
```

min-char-rnn.py gist: 112 lines of Python

```
1  """
2  Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
3  BSD License
4  """
5  import numpy as np
6
7  # data I/O
8  data = open('input.txt', 'r').read() # should be simple plain text file
9  chars = list(set(data))
10 data_size, vocab_size = len(data), len(chars)
11 print('data has %d characters, %d unique.' % (data_size, vocab_size))
12 char_to_ix = { ch:i for i,ch in enumerate(chars) }
13 ix_to_char = { i:ch for i,ch in enumerate(chars) }
14
15 # hyperparameters
16 hidden_size = 100 # size of hidden layer of neurons
17 seq_length = 25 # number of steps to unroll the RNN for
18 learning_rate = 1e-1
19
20 # model parameters
21 wkh = np.random.rand(hidden_size, vocab_size)*0.01 # input to hidden
22 whh = np.random.rand(hidden_size, hidden_size)*0.01 # hidden to hidden
23 why = np.random.rand(vocab_size, hidden_size)*0.01 # hidden to output
24 bh = np.zeros((hidden_size, 1)) # hidden bias
25 by = np.zeros((vocab_size, 1)) # output bias
26
27 def lossFun(inputs, targets, hprev):
28     """
29     inputs,targets are both list of integers.
30     hprev is Hx1 array of initial hidden state
31     returns the loss, gradients on model parameters, and last hidden state
32     """
33     xs, hs, ys, ps = {}, {}, {}, {}
34     hs[-1] = np.copy(hprev)
35     loss = 0
36     # forward pass
37     for t in xrange(len(inputs)):
38         xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation
39         xs[t][inputs[t]] = 1
40         hs[t] = np.tanh(np.dot(wkh, xs[t]) + np.dot(whh, hs[t-1]) + bh) # hidden state
41         ys[t] = np.dot(why, hs[t]) + by # unnormalized log probabilities for next chars
42         ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars
43         loss += -np.log(ps[t][targets[t],0]) # softmax (cross-entropy loss)
44
45         # backward pass: compute gradients going backwards
46         dwhx, dwhy = np.zeros_like(wkh), np.zeros_like(whh), np.zeros_like(why)
47         dbh, dby = np.zeros_like(bh), np.zeros_like(by)
48         dhnext = np.zeros_like(hs[0])
49         for t in reversed(xrange(len(inputs))):
50             dy = np.copy(ps[t])
51             dy[targets[t]] -= 1 # backprop into y
52             dhy = np.dot(dy, hs[t].T)
53             dhy -= dwhy
54             dhy = np.dot(why.T, dhy) + dhnext # backprop into h
55             ddraw = (i - hs[t].T) * dhy # dh backprop through tanh nonlinearity
56             dwhx += np.dot(ddraw, xs[t].T)
57             dwhy += np.dot(ddraw, hs[t-1].T)
58             dhnext = np.dot(whh.T, ddraw)
59             for dparam in [dwhx, dwhy, dbh, dby]:
60                 np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
61
62     return loss, dwhx, dwhy, dbh, dby, hs[len(inputs)-1]
```



```
63 def sample(h, seed_ix, n):
64     """
65     sample a sequence of integers from the model
66     h is memory state, seed_ix is seed letter for first time step
67     """
68     x = np.zeros((vocab_size, 1))
69     x[seed_ix] = 1
70     ixes = []
71     for t in xrange(n):
72         h = np.tanh(np.dot(wkh, x) + np.dot(whh, h) + bh)
73         y = np.dot(why, h) + by
74         p = np.exp(y) / np.sum(np.exp(y))
75         ix = np.random.choice(range(vocab_size), p=p.ravel())
76         x = np.zeros((vocab_size, 1))
77         x[ix] = 1
78         ixes.append(ix)
79
80     return ixes
81
82 n, p = 0, 0
83 mxwh, mwhh, mwhy = np.zeros_like(wkh), np.zeros_like(whh), np.zeros_like(why)
84 mbh, mbhy = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
85 smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
86 while True:
87     # prepare inputs (we're sweeping from left to right in steps seq_length long)
88     if p+seq_length >= len(data) or n == 0:
89         hprev = np.zeros((hidden_size,1)) # reset RNN memory
90         p = 0 # go from start of data
91     inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
92     targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
93
94     # sample from the model now and then
95     if n % 100 == 0:
96         sample_ix = sample(hprev, inputs[0], 200)
97         txt = ''.join(ix_to_char[ix] for ix in sample_ix)
98         print '----\n%s\n----' % (txt, )
99
100     # forward seq_length characters through the net and fetch gradient
101     loss, dwhx, dwhh, dwhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
102     smooth_loss = smooth_loss * .999 + loss * .001
103     if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth_loss) # print progress
104
105     # perform parameter update with Adagrad
106     for param, dparam, mem in zip([wkh, whh, why, bh, by],
107                                   [dwhx, dwhh, dwhy, dbh, dby],
108                                   [mxwh, mwhh, mwhy, mbh, mbhy]):
109         mem += dparam * dparam
110         param += -learning_rate * param / np.sqrt(mem + 1e-8) # adagrad update
111
112     p += seq_length # move data pointer
113     n += 1 # iteration counter
```

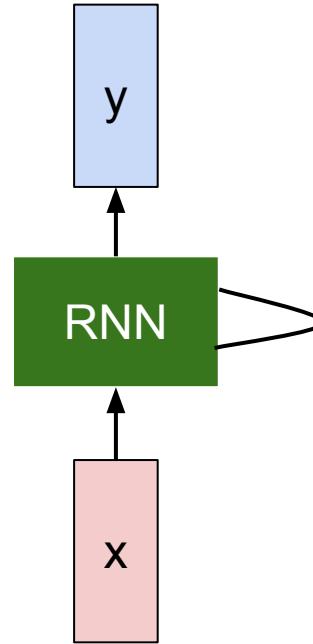
(<https://gist.github.com/karpathy/d4dee566867f8291f086>)

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 deserv'd 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.



at first:

tyntd-iafhatawiaoihrdemot 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 shuwyl 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.

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 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.

VIOLA:

Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:

O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.

The Stacks Project: open source algebraic geometry textbook

The Stacks Project

[home](#) [about](#) [tags explained](#) [tag lookup](#) [browse](#) [search](#) [bibliography](#) [recent comments](#) [blog](#) [add slogans](#)

Browse chapters

Part	Chapter	online	TeX source	view pdf
Preliminaries	1. Introduction	online	tex	pdf
	2. Conventions	online	tex	pdf
	3. Set Theory	online	tex	pdf
	4. Categories	online	tex	pdf
	5. Topology	online	tex	pdf
	6. Sheaves on Spaces	online	tex	pdf
	7. Sites and Sheaves	online	tex	pdf
	8. Stacks	online	tex	pdf
	9. Fields	online	tex	pdf
	10. Commutative Algebra	online	tex	pdf

Parts

1. [Preliminaries](#)
2. [Schemes](#)
3. [Topics in Scheme Theory](#)
4. [Algebraic Spaces](#)
5. [Topics in Geometry](#)
6. [Deformation Theory](#)
7. [Algebraic Stacks](#)
8. [Miscellany](#)

Statistics

The Stacks project now consists of

- o 455910 lines of code
- o 14221 tags (56 inactive tags)
- o 2366 sections

Latex source



<http://stacks.math.columbia.edu/>

The stacks project is licensed under the [GNU Free Documentation License](#)

For $\bigoplus_{n=1,\dots,m} \mathcal{L}_{m,n} = 0$, 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

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

and

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

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 $X_{\text{spaces},\text{étale}}$ 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 $\text{Proj}_X(\mathcal{A}) = \text{Spec}(B)$ 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 $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{x,\dots,x_0}$.

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 $i_0 \leq p$ is a subset of $\mathcal{J}_{n,0} \circ \mathcal{A}_2$ works.

Lemma 0.3. In Situation ???. Hence we may assume $q' = 0$.

Proof. We will use the property we see that 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

Proof. Omitted. \square

Lemma 0.1. Let \mathcal{C} be a set of the construction.

Let \mathcal{C} be a gerber covering. Let \mathcal{F} be a quasi-coherent sheaves of \mathcal{O} -modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

Proof. This is an algebraic space with the composition of sheaves \mathcal{F} on $X_{\text{étale}}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where \mathcal{G} defines an isomorphism $\mathcal{F} \rightarrow \mathcal{F}$ of \mathcal{O} -modules. \square

Lemma 0.2. This is an integer \mathcal{Z} is injective.

Proof. See Spaces, Lemma ??.

Lemma 0.3. Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \rightarrow Y' \rightarrow Y \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X.$$

be a morphism of algebraic spaces over S and Y .

Proof. Let X be a nonzero scheme of X . Let X be an algebraic space. Let \mathcal{F} be a quasi-coherent sheaf of \mathcal{O}_X -modules. The following are equivalent

- (1) \mathcal{F} is an algebraic space over S .
- (2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. \square

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{array}{ccccc}
 S & \xrightarrow{\quad} & & & \\
 \downarrow & & & & \\
 \xi & \longrightarrow & \mathcal{O}_{X'} & \nearrow & \\
 \text{gor}_s & & & & \\
 & & = \alpha' \longrightarrow & & X \\
 & & \downarrow & & \downarrow \\
 & & = \alpha' \longrightarrow \alpha & & \text{Spec}(K_\psi) \qquad \text{Mor}_{\text{Sets}} \qquad d(\mathcal{O}_{X_{X/k}}, \mathcal{G})
 \end{array}$$

is a limit. Then \mathcal{G} is a finite type and assume S is a flat and \mathcal{F} and \mathcal{G} is a finite type f_* . This is of finite type diagrams, and

- the composition of \mathcal{G} is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

\square

Proof. We have see that $X = \text{Spec}(R)$ and \mathcal{F} is a finite type representable by algebraic space. The property \mathcal{F} is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U . \square

Proof. This is clear that \mathcal{G} is a finite presentation, see Lemmas ??.

A reduced above we conclude that U is an open covering of \mathcal{C} . The functor \mathcal{F} is a “field”

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\bar{x}} \xrightarrow{-1} (\mathcal{O}_{X_{\text{étale}}}) \longrightarrow \mathcal{O}_{X_{\bar{x}}}^{-1} \mathcal{O}_{X_{\lambda}}(\mathcal{O}_{X_{\eta}}^{\bar{v}})$$

is an isomorphism of covering of \mathcal{O}_{X_i} . If \mathcal{F} is the unique element of \mathcal{F} such that X is an isomorphism.

The property \mathcal{F} is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme \mathcal{O}_X -algebra with \mathcal{F} are opens of finite type over S .

If \mathcal{F} is a scheme theoretic image points. \square

If \mathcal{F} is a finite direct sum \mathcal{O}_{X_k} is a closed immersion, see Lemma ???. This is a sequence of \mathcal{F} is a similar morphism.

[This repository](#) [Search](#)[Explore](#) [Gist](#) [Blog](#) [Help](#)

karpathy



torvalds / linux

[Watch](#) 3,711[Star](#) 23,054[Fork](#) 9,141

Linux kernel source tree

520,037 commits

1 branch

420 releases

5,039 contributors

branch: master ➔ [linux](#) / +

Merge branch 'drm-fixes' of git://people.freedesktop.org/~airlied/linux ...

torvalds authored 9 hours ago latest commit 4b1706927d ↗

Documentation	Merge git://git.kernel.org/pub/scm/linux/kernel/git/nab/target-pending	6 days ago
arch	Merge branch 'x86-urgent-for-linus' of git://git.kernel.org/pub/scm/l...	a day ago
block	block: discard bdi_unregister() in favour of bdi_destroy()	9 days ago
crypto	Merge git://git.kernel.org/pub/scm/linux/kernel/git/herbert/crypto-2.6	10 days ago
drivers	Merge branch 'drm-fixes' of git://people.freedesktop.org/~airlied/linux	9 hours ago
firmware	firmware/ihex2fw.c: restore missing default in switch statement	2 months ago
fs	vfs: read file_handle only once in handle_to_path	4 days ago
include	Merge branch 'perl-urgent-for-linus' of git://git.kernel.org/pub/scm/...	a day ago
init	init: fix regression by supporting devices with major:minor:offset fo...	a month ago
iio	iio: sensors: New driver and initial framework for industrial I/O sensors	a month ago



Code



74 Pull requests



Pulse



Graphs

HTTPS clone URL

<https://github.com/torvalds/linux.git>You can clone with **HTTPS**,
SSH, or **Subversion**. ⓘ[Clone in Desktop](#)[Download ZIP](#)

```
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << 1))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000fffffff8) & 0x0000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &soffset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}
```

Generated C code

```
/*
 * Copyright (c) 2006-2010, Intel Mobile Communications. All rights reserved.
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 *
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software Foundation,
 * Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

#include <linux/kexec.h>
#include <linux/errno.h>
#include <linux/io.h>
#include <linux/platform_device.h>
#include <linux/multi.h>
#include <linux/ckevent.h>

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/seteew.h>
#include <asm/pgproto.h>
```

```
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/seteew.h>
#include <asm/pgproto.h>

#define REG_PG      vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)      (func)

#define SWAP_ALLOCATE(nr)      (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %esp, %0, %3" : : "r" (0)); \
    if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
    pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
    PUT_PARAM_RAID(2, sel) = get_state_state();
    set_pid_sum((unsigned long)state, current_state_str(),
                (unsigned long)-1->lr_full, low;
}

```

OpenAI GPT-2 generated text

[source](#)

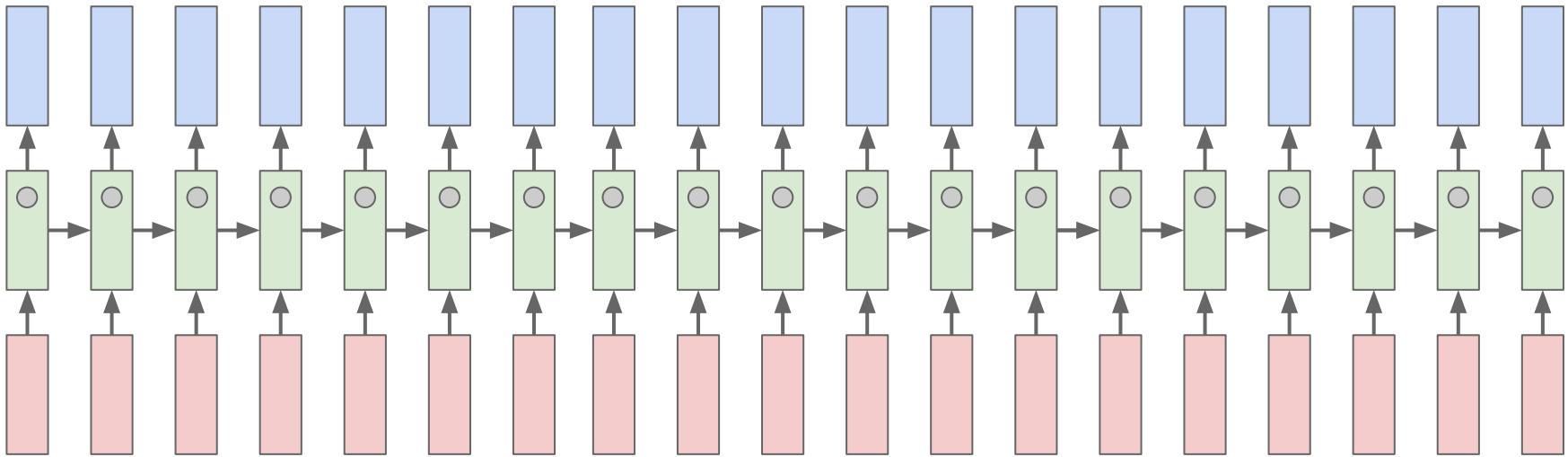
Input: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Output: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Searching for interpretable cells



Searching for interpretable cells

```
/* 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.
    */
}
```

Searching for interpretable cells

"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."

quote detection cell

Обнаружение цитат (quote detection) — это задача в обработке естественного языка (NLP), направленная на автоматическое определение и извлечение текстовых фрагментов, представляющих собой цитаты, из больших объемов текстовых данных

Searching for interpretable cells

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.

line length tracking cell

Searching for interpretable cells

```
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;
}
```

if statement cell

Searching for interpretable cells

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;
}
```

quote/comment cell

Searching for interpretable cells

```
#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;
}
```

code depth cell

RNN tradeoffs

RNN Advantages:

- Can process any length input
- Computation for step t can (in theory) use information from many steps back
- Model size doesn't increase for longer input
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

RNN Disadvantages:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back

LSTM

Vanilla RNN постоянно переписывает $\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b})$

Идея: добавить “память” $\mathbf{c}^{(t)}$

LSTM

Vanilla RNN постоянно переписывает $\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b})$

Идея: добавить “память” $\mathbf{c}^{(t)}$

Новая информация

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)} + \mathbf{b}_c)$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{\mathbf{c}}^{(t)}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \circ \tanh \mathbf{c}^{(t)}$$

All these are vectors of same length n

LSTM

Vanilla RNN постоянно переписывает $\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b})$

Идея: добавить “память” $\mathbf{c}^{(t)}$

Новая информация

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)} + \mathbf{b}_c)$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{\mathbf{c}}^{(t)}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \circ \tanh \mathbf{c}^{(t)}$$

All these are vectors of same length n

LSTM

Vanilla RNN постоянно переписывает $\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b})$

Идея: добавить “память” $\mathbf{c}^{(t)}$

Новая информация

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)} + \mathbf{b}_c)$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{\mathbf{c}}^{(t)}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \circ \tanh \mathbf{c}^{(t)}$$

All these are vectors of same length n

LSTM

Vanilla RNN постоянно переписывает $\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b})$

Идея: добавить “память” $\mathbf{c}^{(t)}$

Forget gate: контролируем сколько забыть

Input gate: контролируем сколько записать

$$\begin{aligned} f^{(t)} &= \sigma(\mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_f \mathbf{x}^{(t)} + \mathbf{b}_f) \\ i^{(t)} &= \sigma(\mathbf{W}_i \mathbf{h}^{(t-1)} + \mathbf{U}_i \mathbf{x}^{(t)} + \mathbf{b}_i) \end{aligned}$$

Sigmoid function: all gate values are between 0 and 1

Новая информация

Можно записать новую и стереть старую

$$\begin{aligned} \tilde{\mathbf{c}}^{(t)} &= \tanh(\mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)} + \mathbf{b}_c) \\ \mathbf{c}^{(t)} &= f^{(t)} \circ \mathbf{c}^{(t-1)} + i^{(t)} \circ \tilde{\mathbf{c}}^{(t)} \\ \mathbf{h}^{(t)} &= o^{(t)} \circ \tanh \mathbf{c}^{(t)} \end{aligned}$$

All these are vectors of same length n

LSTM

Vanilla RNN постоянно переписывает $\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b})$

Идея: добавить “память” $\mathbf{c}^{(t)}$

Forget gate: контролируем сколько забыть

Sigmoid function: all gate values are between 0 and 1

$$f^{(t)} = \sigma(\mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_f \mathbf{x}^{(t)} + \mathbf{b}_f)$$

Input gate: контролируем сколько записать

$$i^{(t)} = \sigma(\mathbf{W}_i \mathbf{h}^{(t-1)} + \mathbf{U}_i \mathbf{x}^{(t)} + \mathbf{b}_i)$$

Output gate: контролируем сколько считать

$$o^{(t)} = \sigma(\mathbf{W}_o \mathbf{h}^{(t-1)} + \mathbf{U}_o \mathbf{x}^{(t)} + \mathbf{b}_o)$$

Новая информация

Можно записать новую и стереть старую

Считывание

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)} + \mathbf{b}_c)$$

$$\mathbf{c}^{(t)} = f^{(t)} \circ \mathbf{c}^{(t-1)} + i^{(t)} \circ \tilde{\mathbf{c}}^{(t)}$$

$$\mathbf{h}^{(t)} = o^{(t)} \circ \tanh \mathbf{c}^{(t)}$$

All these are vectors of same length n

Long Short Term Memory (LSTM)

Vanilla RNN

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

LSTM

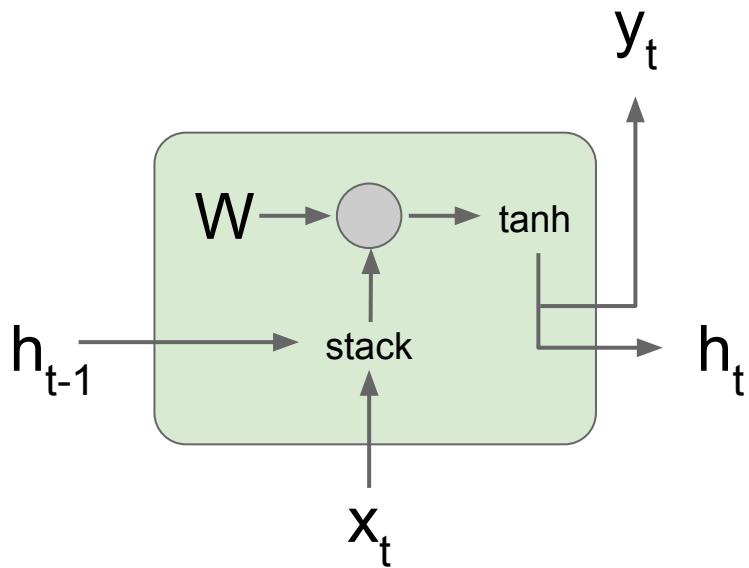
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

Vanilla RNN Gradient Flow

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



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

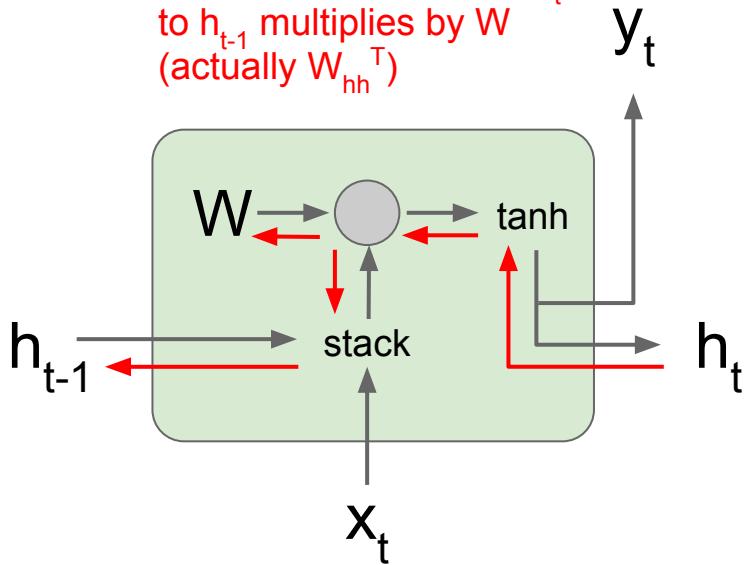
Vanilla RNN Gradient Flow

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

Backpropagation from h_t

to h_{t-1} multiplies by W

(actually W_{hh}^T)

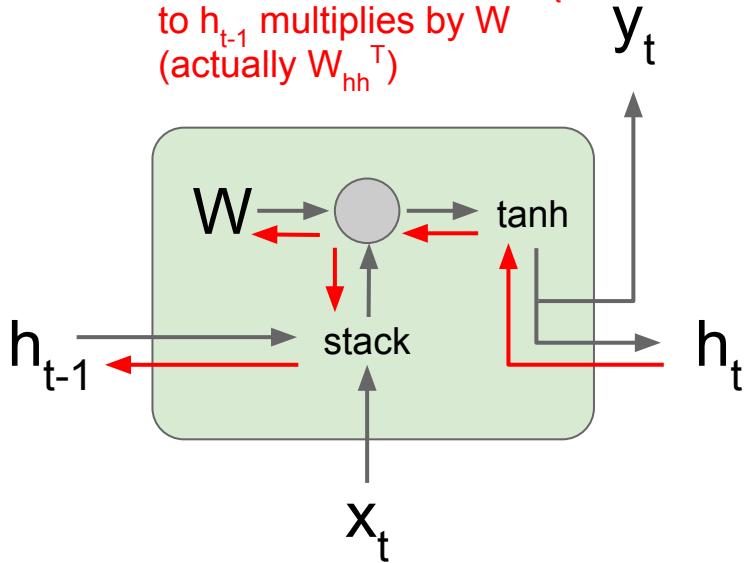


$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ &= \tanh \left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\ &= \tanh \left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \end{aligned}$$

Vanilla RNN Gradient Flow

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

Backpropagation from h_t
to h_{t-1} multiplies by W
(actually W_{hh}^T)

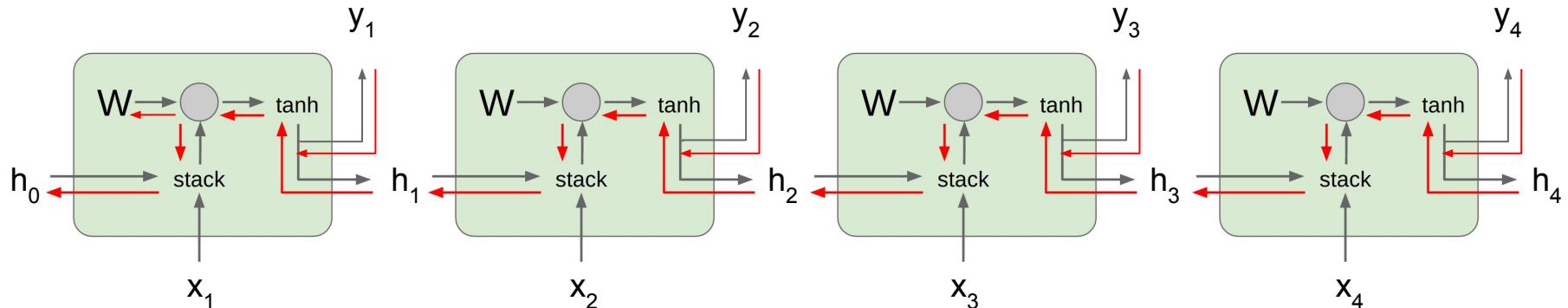


$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ &= \tanh \left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\ &= \tanh \left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \end{aligned}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \tanh'(W_{hh}h_{t-1} + W_{xh}x_t)W_{hh}$$

Vanilla RNN Gradient Flow

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

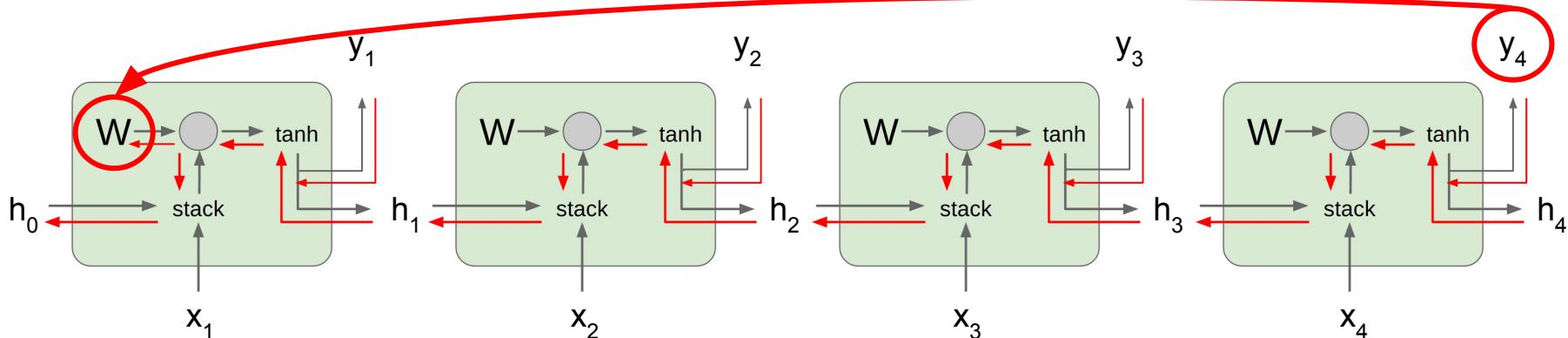


$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

Vanilla RNN Gradient Flow

Gradients over multiple time steps:

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



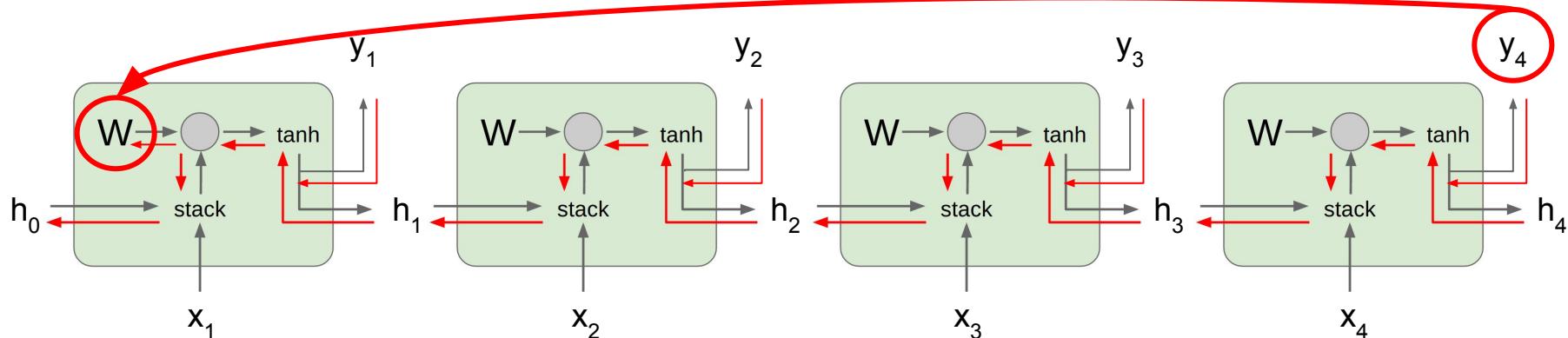
$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W}$$

Vanilla RNN Gradient Flow

Gradients over multiple time steps:

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



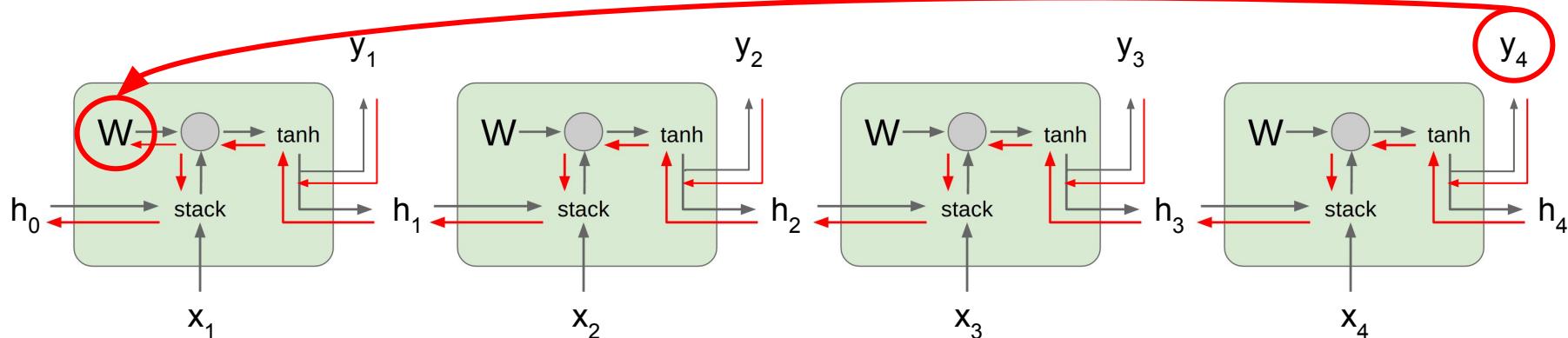
$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W} = \frac{\partial L_T}{\partial h_T} \left(\prod_{t=2}^T \frac{\partial h_t}{\partial h_{t-1}} \right) \frac{\partial h_1}{\partial W}$$

Vanilla RNN Gradient Flow

Gradients over multiple time steps:

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



$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

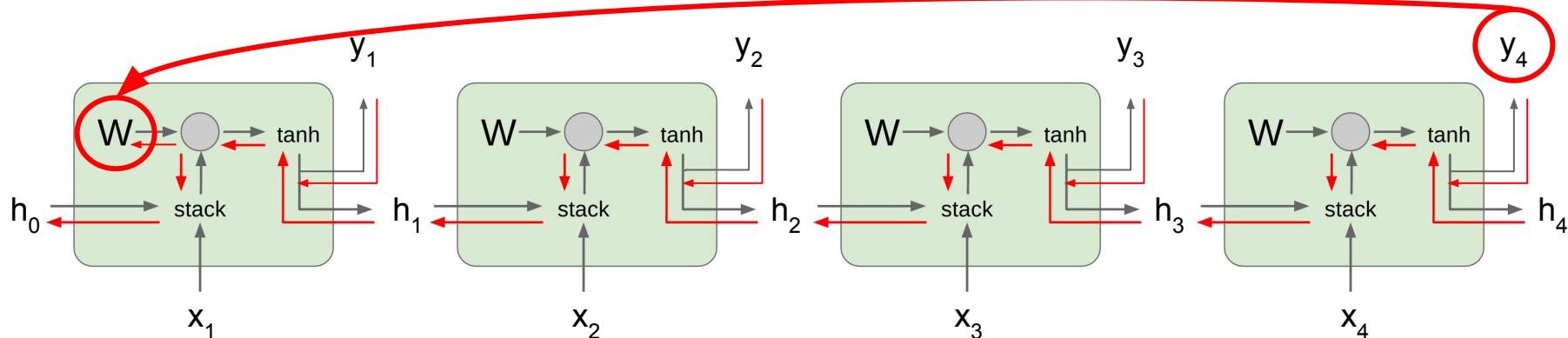
$$\boxed{\frac{\partial h_t}{\partial h_{t-1}} = \tanh'(W_{hh}h_{t-1} + W_{xh}x_t)W_{hh}}$$

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W} = \frac{\partial L_T}{\partial h_T} \left(\prod_{t=2}^T \boxed{\frac{\partial h_t}{\partial h_{t-1}}} \right) \frac{\partial h_1}{\partial W}$$

Vanilla RNN Gradient Flow

Gradients over multiple time steps:

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



$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

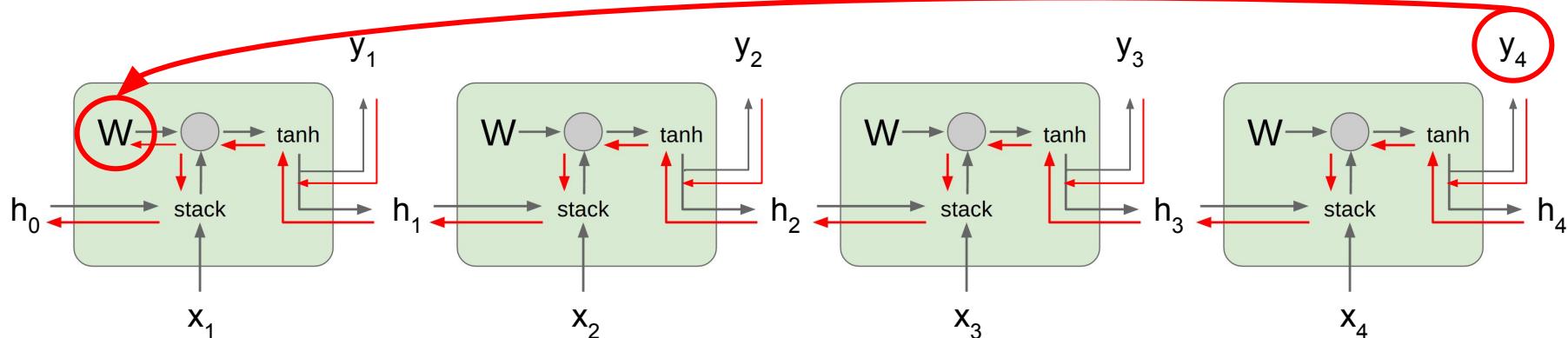
Almost always < 1
Vanishing gradients

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \left(\prod_{t=2}^T \boxed{\tanh'(W_{hh} h_{t-1} + W_{xh} x_t)} \right) W_{hh}^{T-1} \frac{\partial h_1}{\partial W}$$

Vanilla RNN Gradient Flow

Gradients over multiple time steps:

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



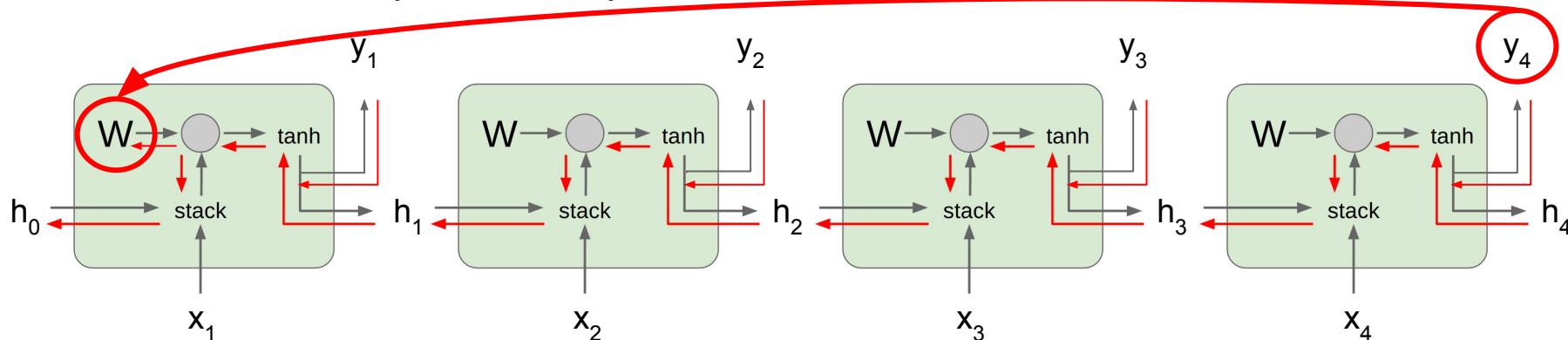
$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

What if we assumed no non-linearity?

Vanilla RNN Gradient Flow

Gradients over multiple time steps:

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



What if we assumed no non-linearity?

$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

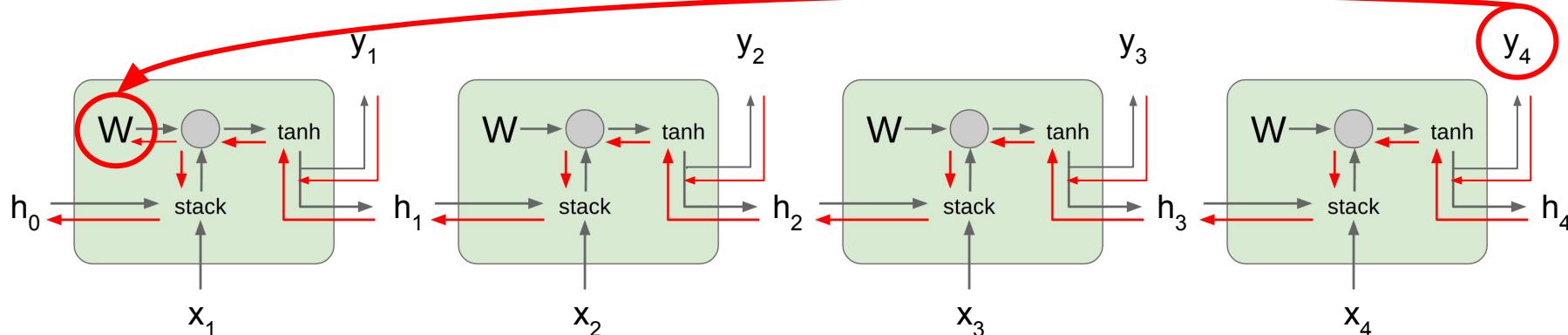
Largest singular value > 1 :
Exploding gradients

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \boxed{W_{hh}^{T-1}} \frac{\partial h_1}{\partial W}$$

Largest singular value < 1 :
Vanishing gradients

Vanilla RNN Gradient Flow

Gradients over multiple time steps:



What if we assumed no non-linearity?

$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \boxed{W_{hh}^{T-1}} \frac{\partial h_1}{\partial W}$$

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

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

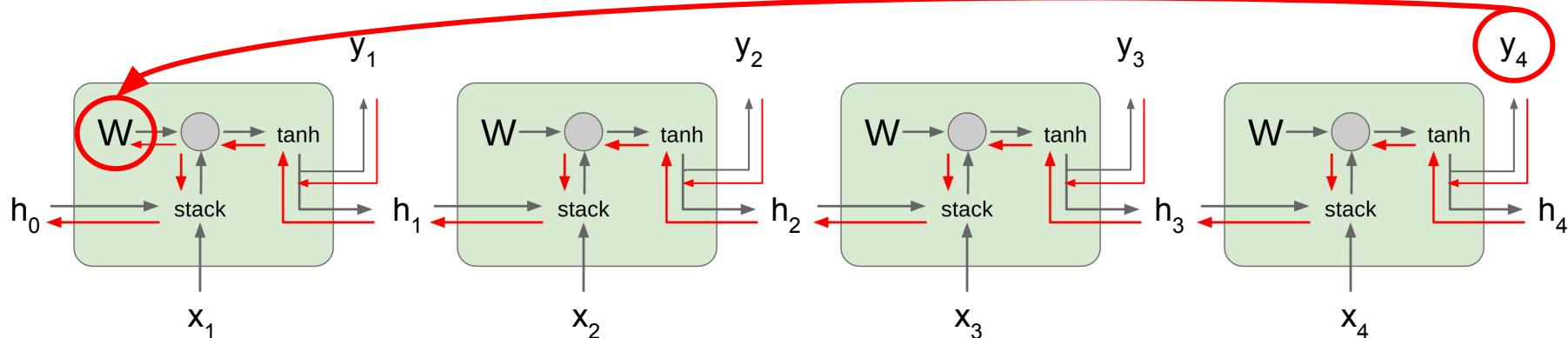
→ **Gradient clipping:**
Scale gradient if its norm is too big

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
    grad *= (threshold / grad_norm)
```

Vanilla RNN Gradient Flow

Gradients over multiple time steps:

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



What if we assumed no non-linearity?

$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

Largest singular value > 1 :
Exploding gradients

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \boxed{W_{hh}^{T-1}} \frac{\partial h_1}{\partial W}$$

Largest singular value < 1 :
Vanishing gradients

→ Change RNN architecture

Long Short Term Memory (LSTM)

Vanilla RNN

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

LSTM

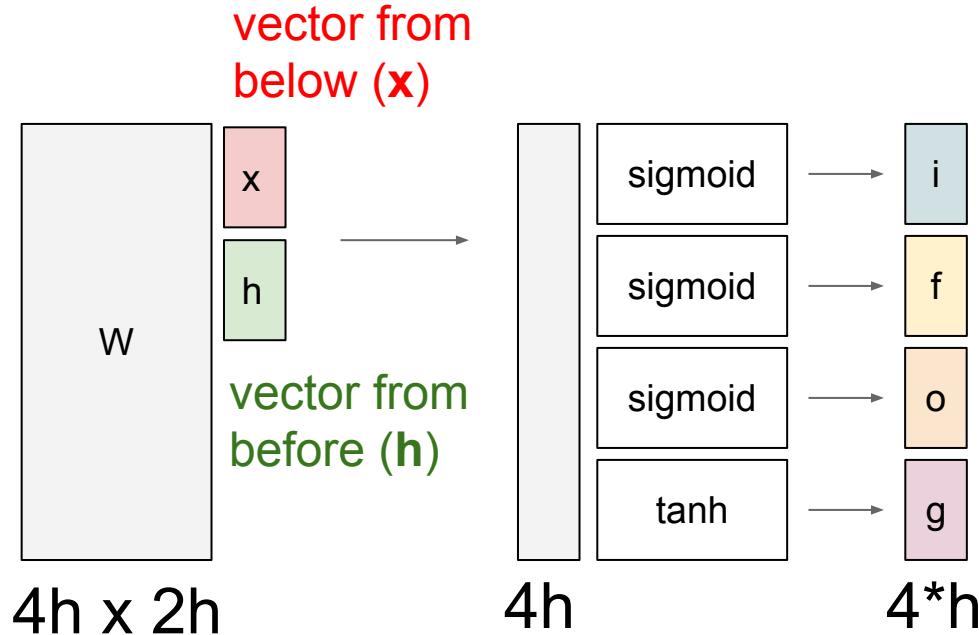
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

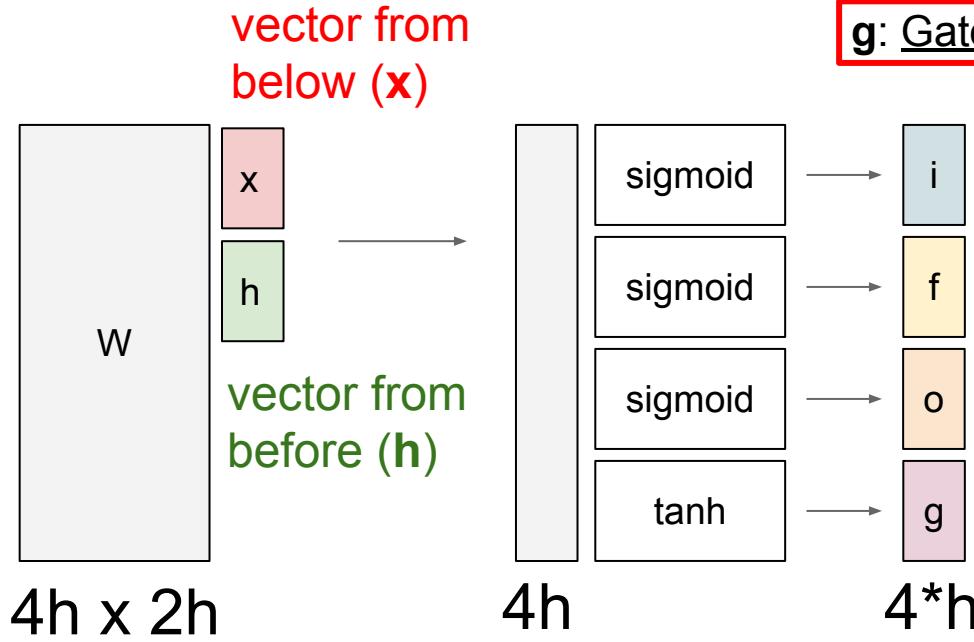
Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



g: Gate (?), How much to write to cell

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

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

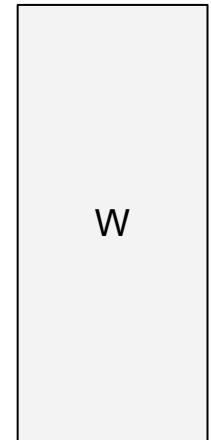
$$h_t = o \odot \tanh(c_t)$$

Long Short Term Memory (LSTM)

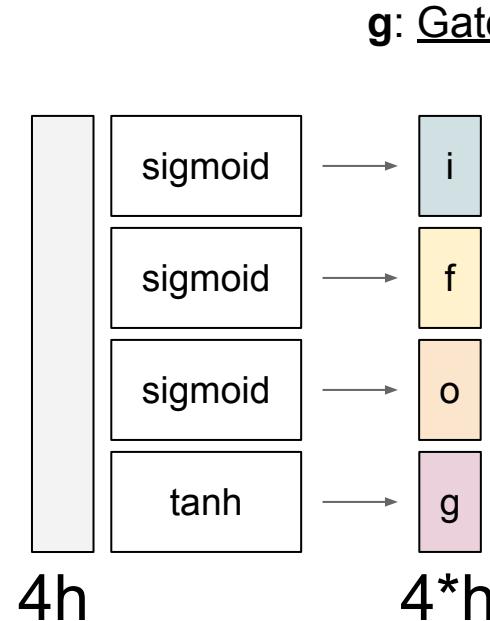
[Hochreiter et al., 1997]

i: Input gate, whether to write to cell

vector from
below (x)



4h x 2h



g: Gate gate (?), How much to write to cell

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

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

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

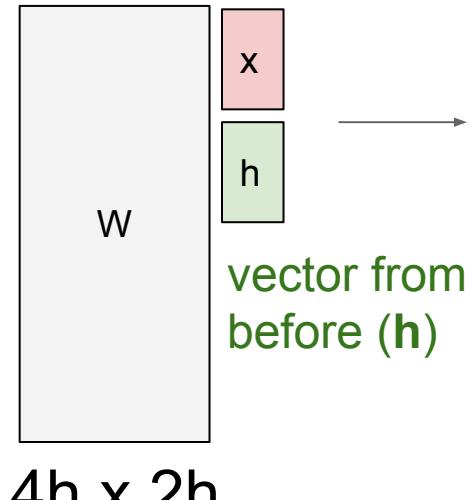
Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

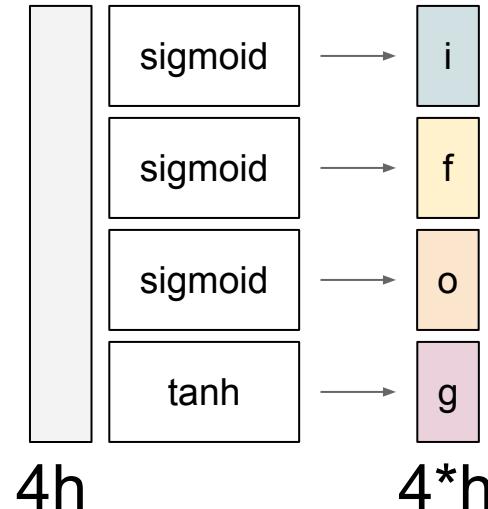
i: Input gate, whether to write to cell
f: Forget gate, Whether to erase cell

g: Gate gate (?), How much to write to cell

vector from
below (x)



vector from
before (h)



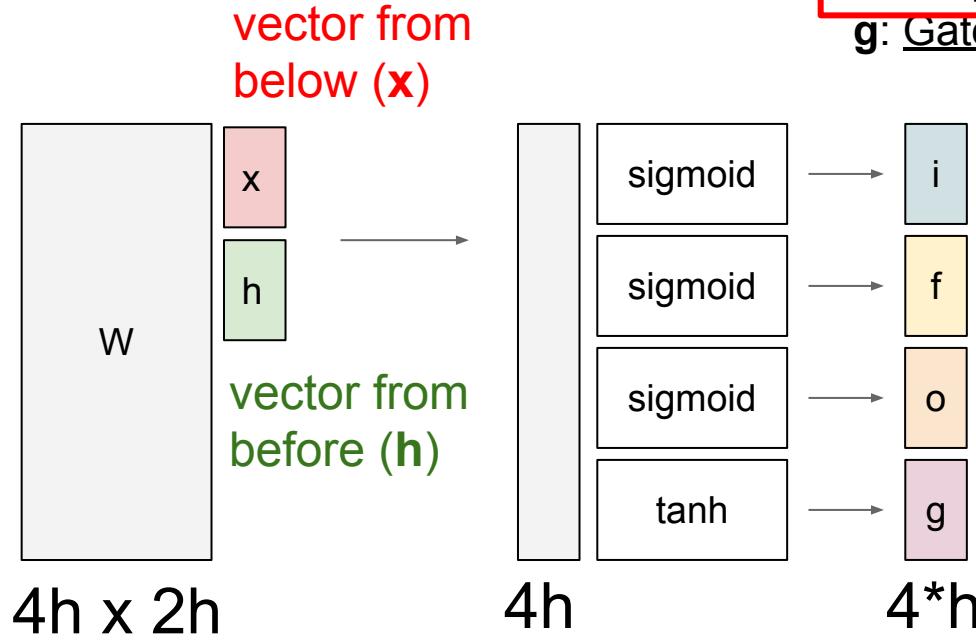
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



- i: Input gate, whether to write to cell
- f: Forget gate, Whether to erase cell
- o: Output gate, How much to reveal cell**
- g: Gate gate (?), How much to write to cell

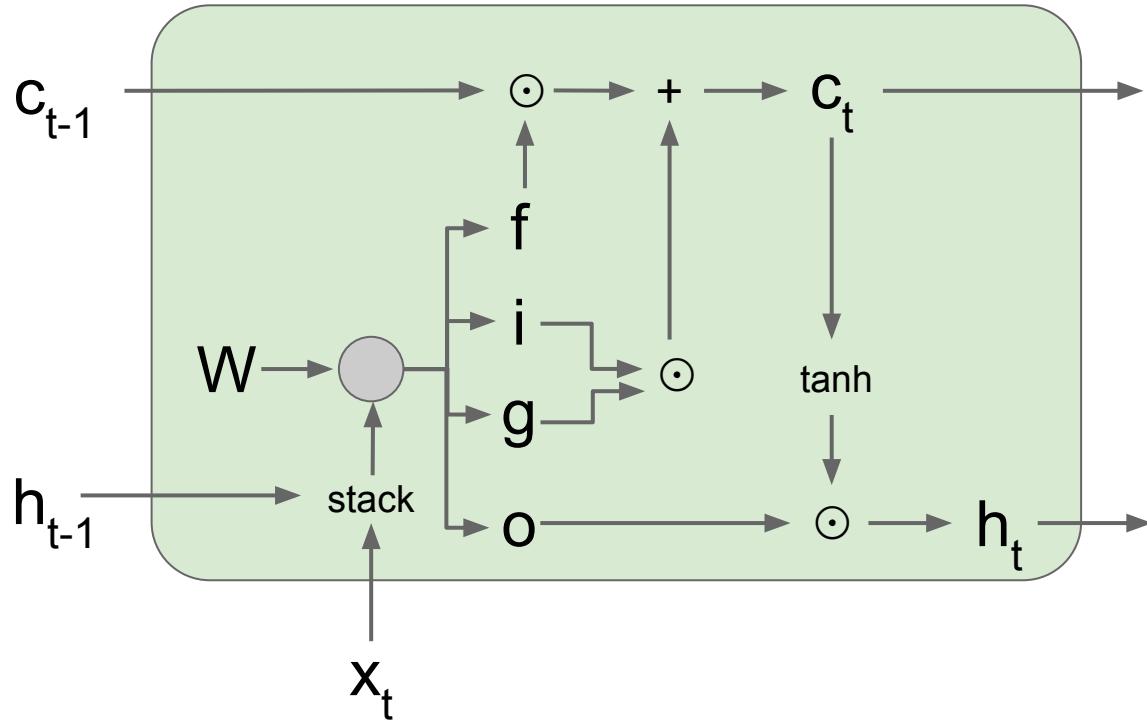
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma & & & \\ & \sigma & & \\ & & \sigma & \\ & & & \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



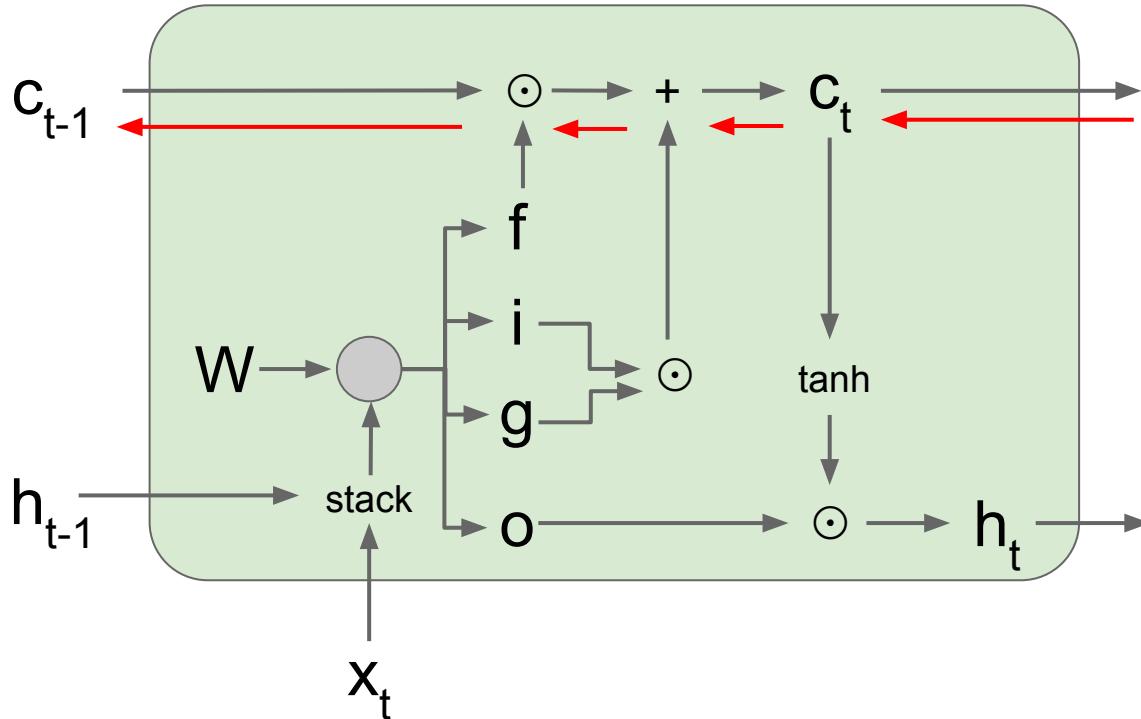
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]



Backpropagation from c_t to c_{t-1} only elementwise multiplication by f , no matrix multiply by W

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

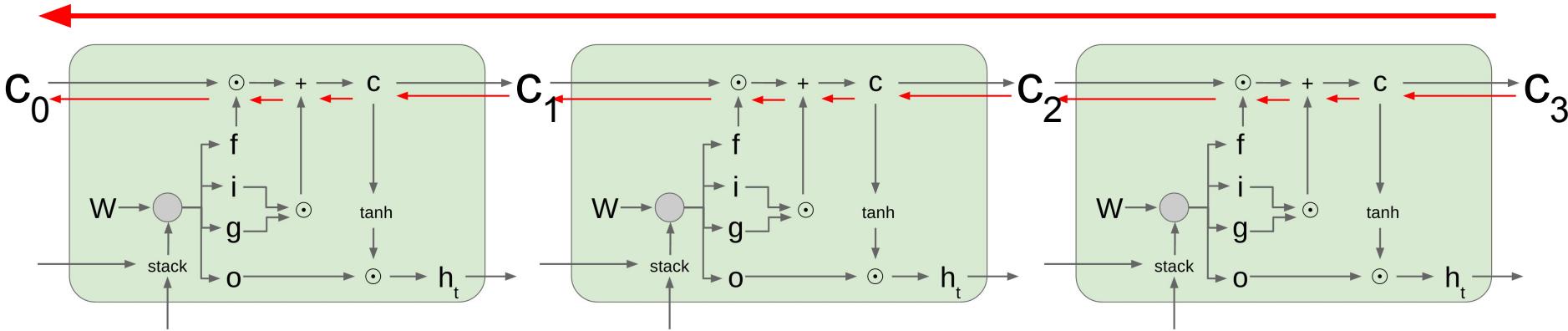
$$c_t = f \odot c_{t-1} + i \odot g$$

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

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

Uninterrupted gradient flow!



Notice that the gradient contains the **f** gate's vector of activations

- allows better control of gradients values, using suitable parameter updates of the forget gate.

Also notice that are added through the **f**, **i**, **g**, and **o** gates

- better balancing of gradient values

GRU

Упрощенная версия LSTM

GRU (Gated Recurrent Unit) — это еще один тип рекуррентной нейронной сети (RNN), который был предложен в 2014 году для улучшения обработки последовательных данных. GRU схожа с LSTM, но имеет более простую архитектуру, что делает её быстрее и проще в обучении, при этом сохраняя способность моделировать долгосрочные зависимости.

Основные характеристики GRU

Гейты:

В GRU используются два типа гейтов:

Гейт обновления (Update Gate): определяет, какую информацию из предыдущего состояния следует сохранить.

Гейт сброса (Reset Gate): решает, какую информацию из предыдущего состояния следует забыть.

Единое состояние:

В отличие от LSTM, где есть отдельные состояния ячейки и скрытого состояния, в GRU используется только одно состояние, что упрощает архитектуру.

Простота:

GRU имеет меньше параметров, чем LSTM, что может ускорить процесс обучения и снизить вероятность переобучения.

Update gate
Reset gate

$$\begin{aligned}\mathbf{u}^{(t)} &= \sigma \left(\mathbf{W}_u \mathbf{h}^{(t-1)} + \mathbf{U}_u \mathbf{x}^{(t)} + \mathbf{b}_u \right) \\ \mathbf{r}^{(t)} &= \sigma \left(\mathbf{W}_r \mathbf{h}^{(t-1)} + \mathbf{U}_r \mathbf{x}^{(t)} + \mathbf{b}_r \right)\end{aligned}$$

$$\begin{aligned}\tilde{\mathbf{h}}^{(t)} &= \tanh \left(\mathbf{W}_h (\mathbf{r}^{(t)} \circ \mathbf{h}^{(t-1)}) + \mathbf{U}_h \mathbf{x}^{(t)} + \mathbf{b}_h \right) \\ \mathbf{h}^{(t)} &= (1 - \mathbf{u}^{(t)}) \circ \mathbf{h}^{(t-1)} + \mathbf{u}^{(t)} \circ \tilde{\mathbf{h}}^{(t)}\end{aligned}$$

Рекуррентные нейросети (RNN)

GRU и LSTM - уменьшают проблему vanishing gradients

Преодоление проблемы исчезающих градиентов: LSTM сохраняют информацию на более длительные периоды, что помогает обучать модели на длинных последовательностях.

Recurrence for Vision

- LSTM were a good default choice until this year
- Use variants like GRU if you want faster compute and less parameters
- Use transformers (next lecture) as they are dominating NLP models
 - almost everyday there is a new vision transformer model

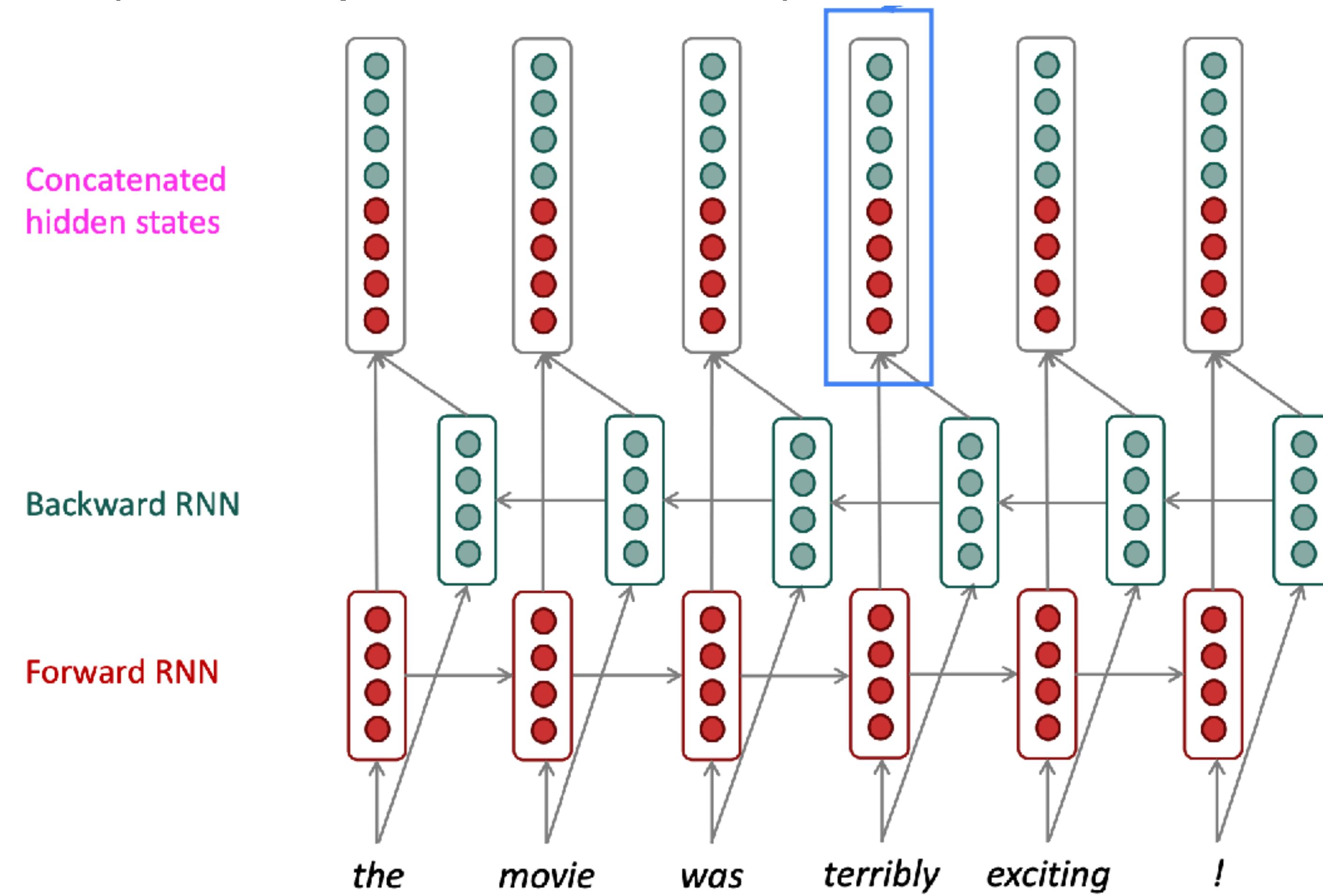
Su et al. "VI-bert: Pre-training of generic visual-linguistic representations." ICLR 2020

Lu et al. "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks." NeurIPS 2019

Li et al. "Visualbert: A simple and performant baseline for vision and language." *arXiv* 2019

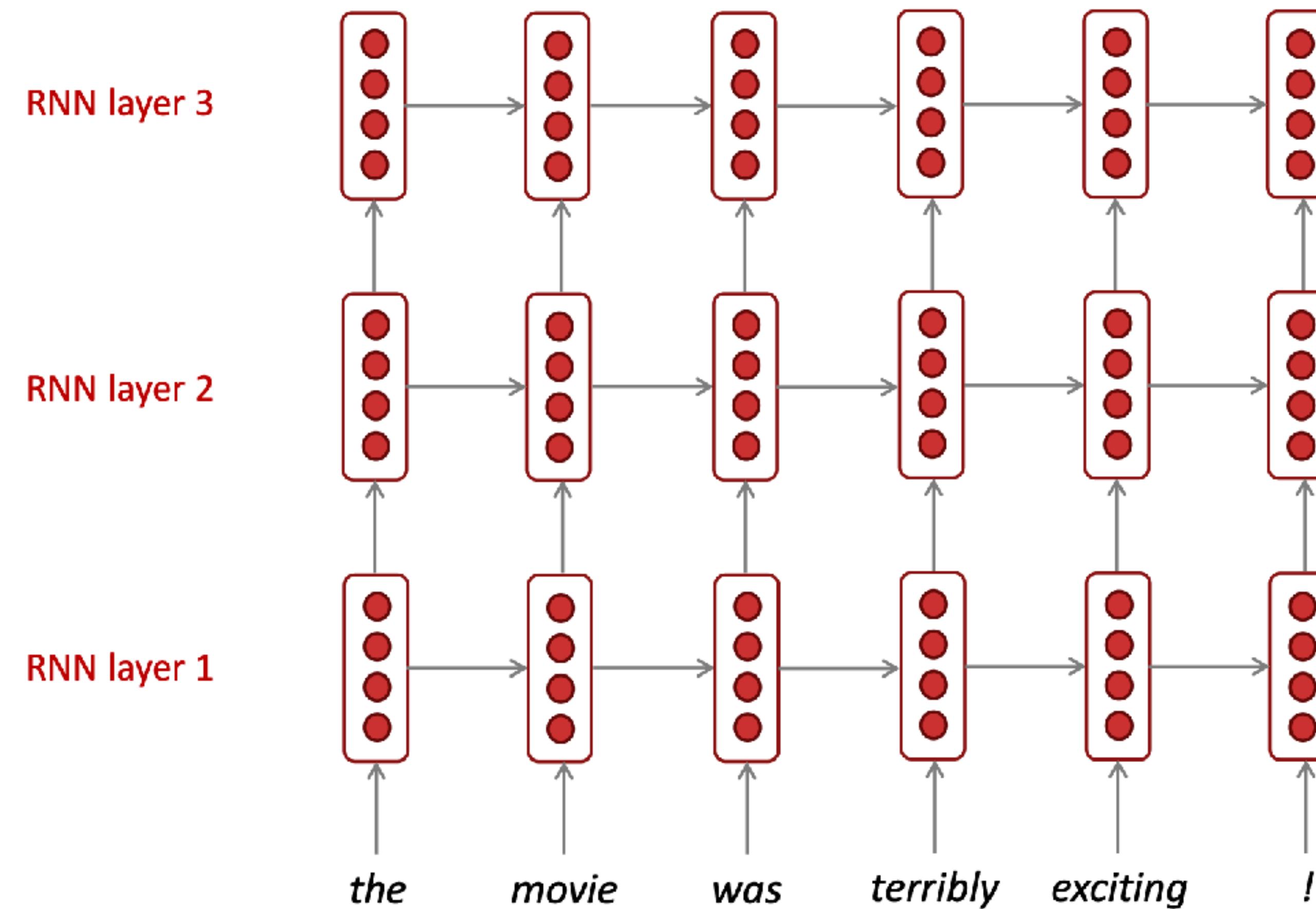
Рекуррентные нейросети (RNN)

Улучшения: bidirectional RNN (если порядок не важен)



Рекуррентные нейросети (RNN)

Улучшения: stacked RNN



Summary

- RNNs allow a lot of flexibility in architecture design
- Vanilla RNNs are simple but don't work very well
- Common to use LSTM or GRU: their additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish. Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research, as well as new paradigms for reasoning over sequences
- Better understanding (both theoretical and empirical) is needed.

Sequence-to-sequence model

Приложения: перевод

≡ Google Translate ⋮

Text Documents

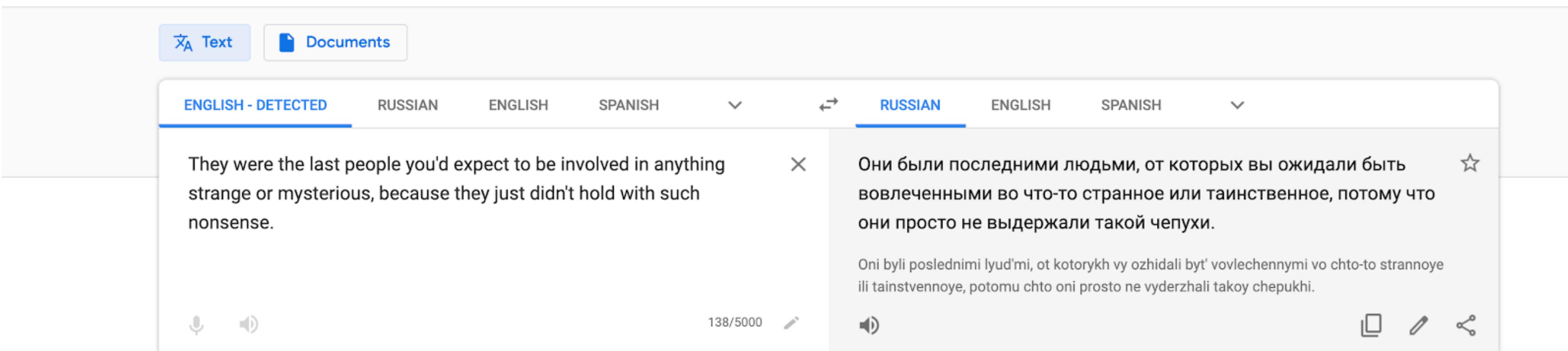
ENGLISH - DETECTED RUSSIAN ENGLISH SPANISH RUSSIAN ENGLISH SPANISH

They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense.

Они были последними людьми, от которых вы ожидали быть вовлеченными во что-то странное или таинственное, потому что они просто не выдержали такой чепухи.

Oni byli poslednimi lyud'mi, ot kotorikh vy ozhidali byt' vovlechennymi vo chto-to strannoye ili tainstvennoye, potomu chto oni prosto ne vyderzhali takoy chepukhi.

Send feedback



Приложения: перевод

The screenshot shows the Google Translate interface. At the top, it says "≡ Google Translate" and has a three-dot menu icon. Below that, there are tabs for "Text" (selected) and "Documents". The main area shows two language pairs: English to Russian and Russian to English. The English input is: "They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense." The Russian output is: "Они были последними людьми, от которых вы ожидали быть вовлеченными во что-то странное или таинственное, потому что они просто не выдержали такой чепухи." Below the translation, the Russian text is shown again with some additional characters: "Oni byli poslednimi lyud'mi, ot kotorikh vy ozhidali byt' vovlechennymi vo chto-to strannoye ili tainstvennoye, potomu chto oni prosto ne vyderzhali takoy chepukhi.". At the bottom of the translation box, there are icons for microphone, speaker, edit, and share, along with a word count "138/5000". On the far right, there is a "Send feedback" link.

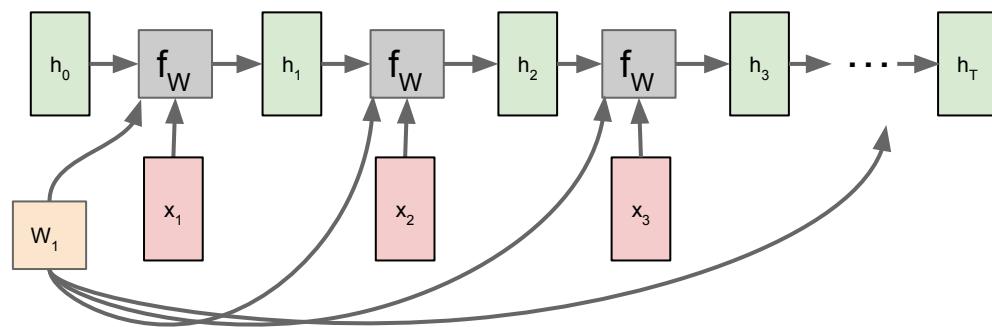
Авторегрессионная модель $p(y|x) = \prod_{i=1}^n p(y_i|y_1, \dots, y_{i-1}, \underline{x})$

x - текст на исходном языке (source)

y - перевод текста на другой язык (target)

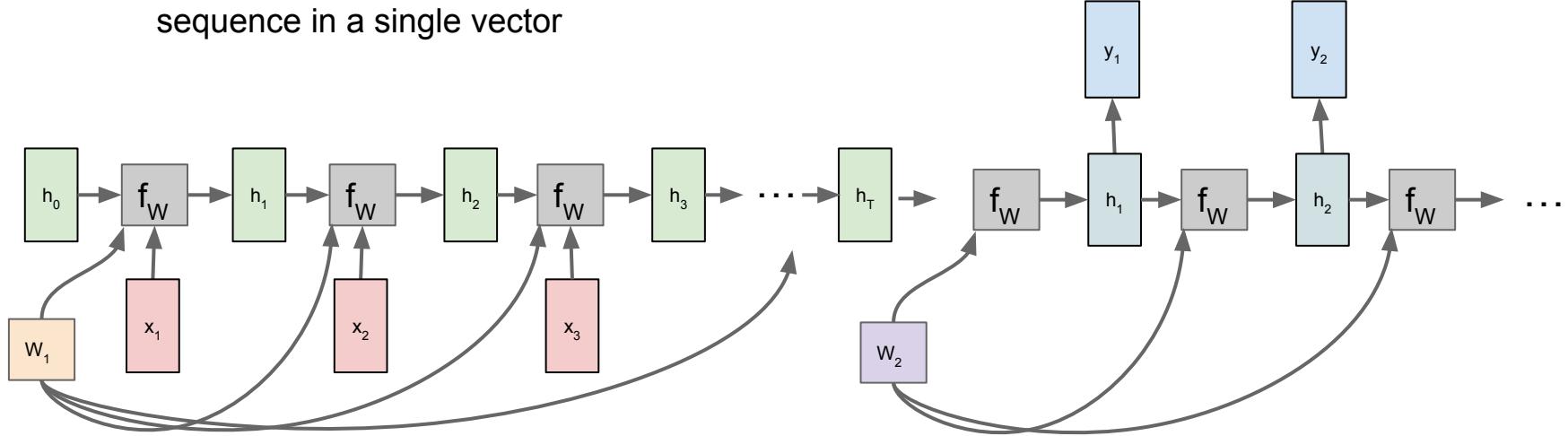
Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector



Sequence to Sequence: Many-to-one + one-to-many

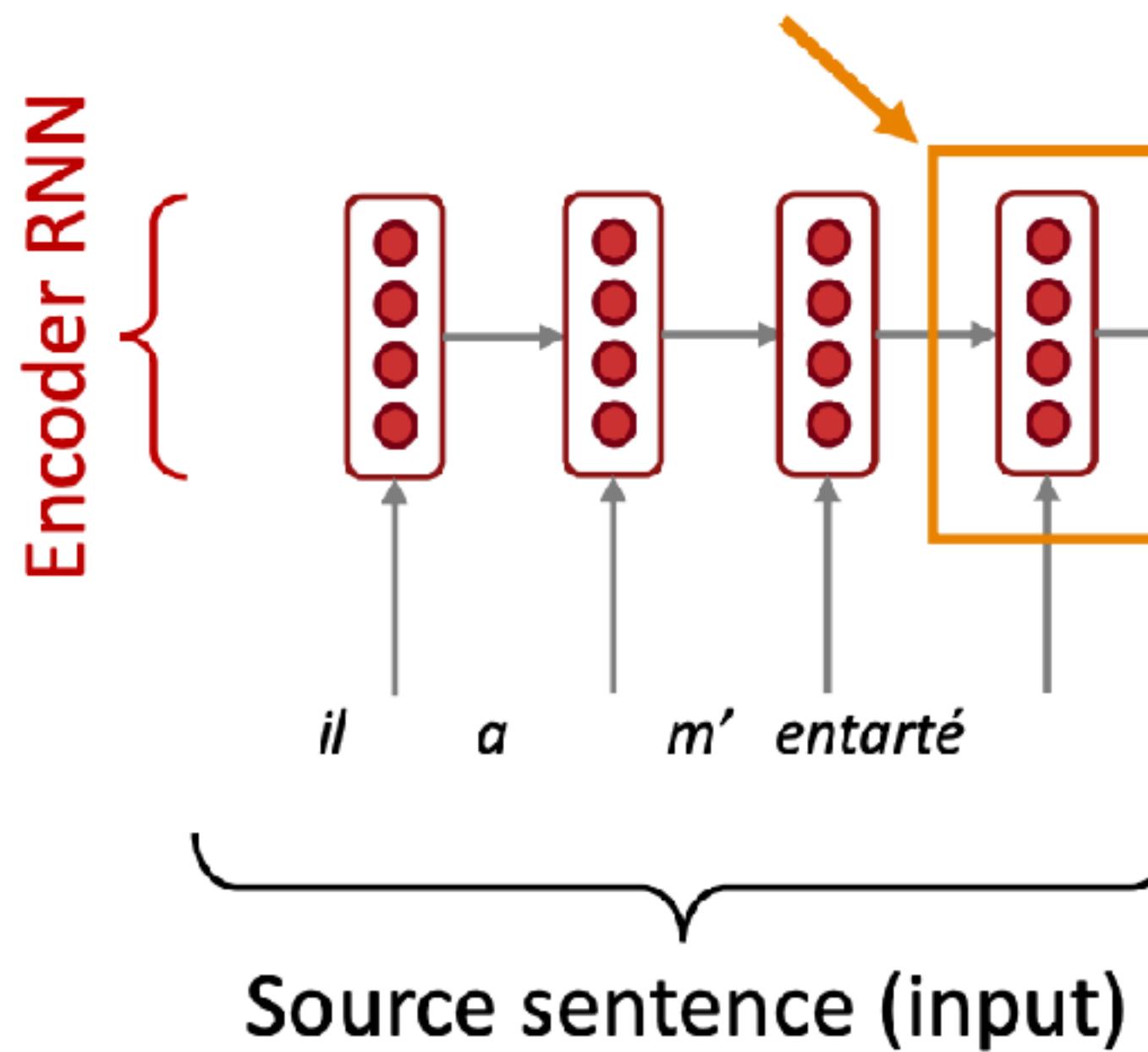
Many to one: Encode input sequence in a single vector



One to many: Produce output sequence from single input vector

Sequence-to-sequence model

Последний hidden state -
представление всего
исходного текста



Sequence-to-Sequence (Seq2Seq) Model — это тип нейронной сети, который предназначен для обработки и преобразования последовательностей данных одной длины в последовательности другой длины. Эта модель широко используется в задачах, связанных с обработкой естественного языка, таких как машинный перевод, генерация текста, подведение итогов и многие другие.
Основные компоненты Seq2Seq

Энкодер (Encoder):

Энкодер принимает входную последовательность и преобразует её в фиксированное представление (контекстный вектор). Обычно это RNN, LSTM или GRU.

Каждый элемент входной последовательности обрабатывается, и в конце энкодер выдает вектор, который представляет всю последовательность.

Декодер (Decoder):

Декодер принимает контекстный вектор от энкодера и генерирует выходную последовательность.

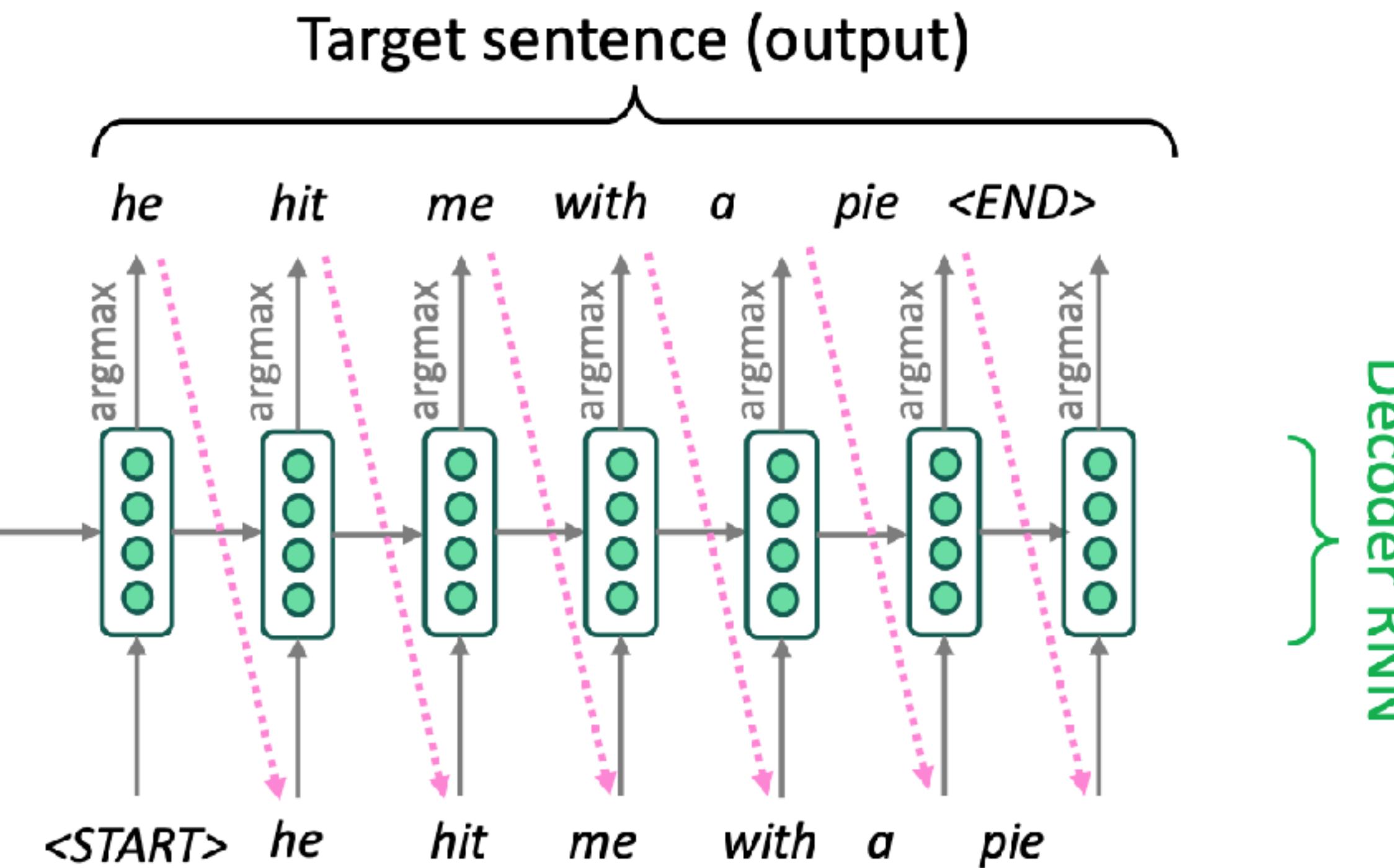
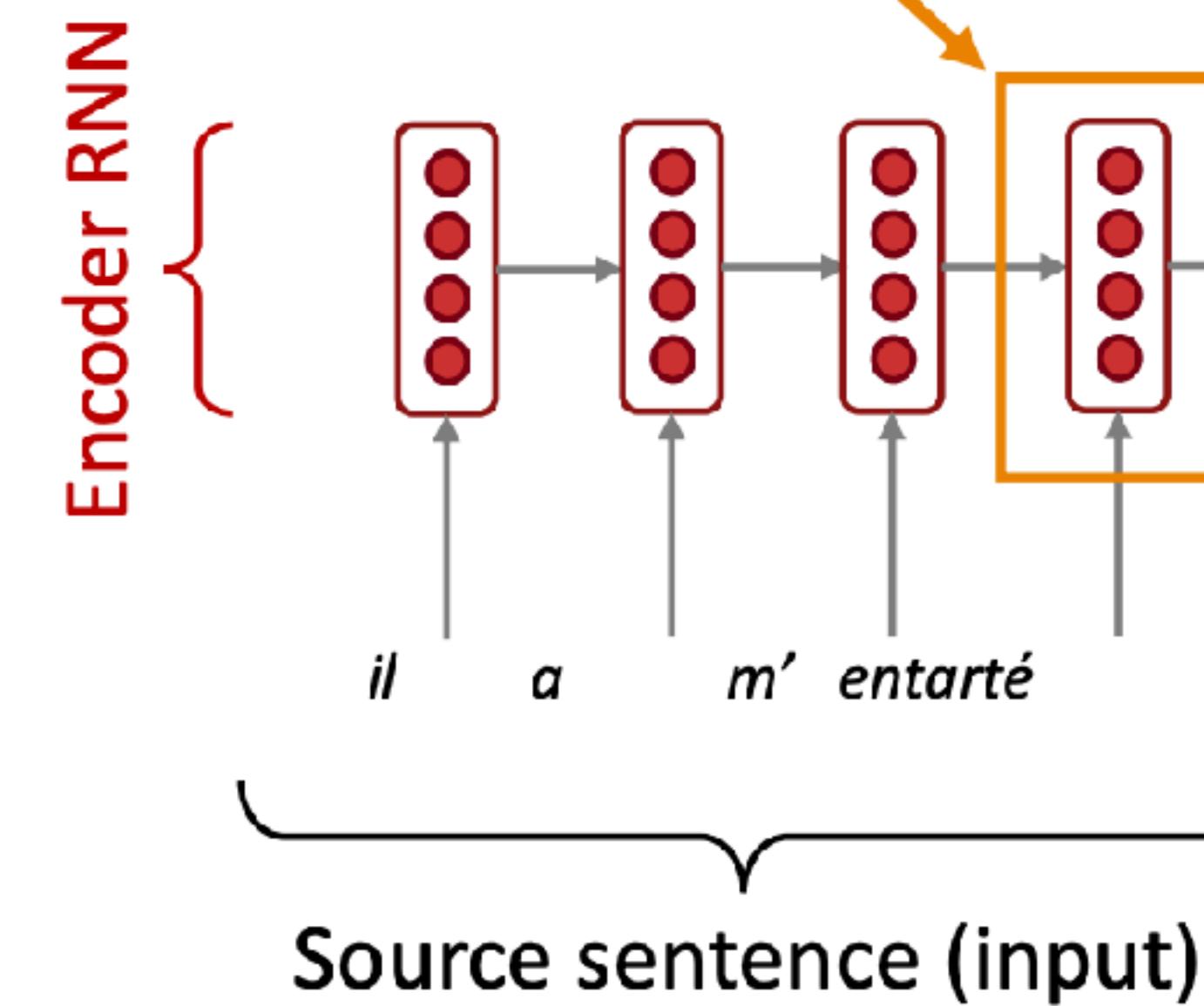
Он также может быть реализован с помощью RNN, LSTM или GRU.

Важно, чтобы декодер имел возможность получать предсказанные элементы в качестве входа для генерации следующего элемента (это называется teacher forcing).

Контекстный вектор:

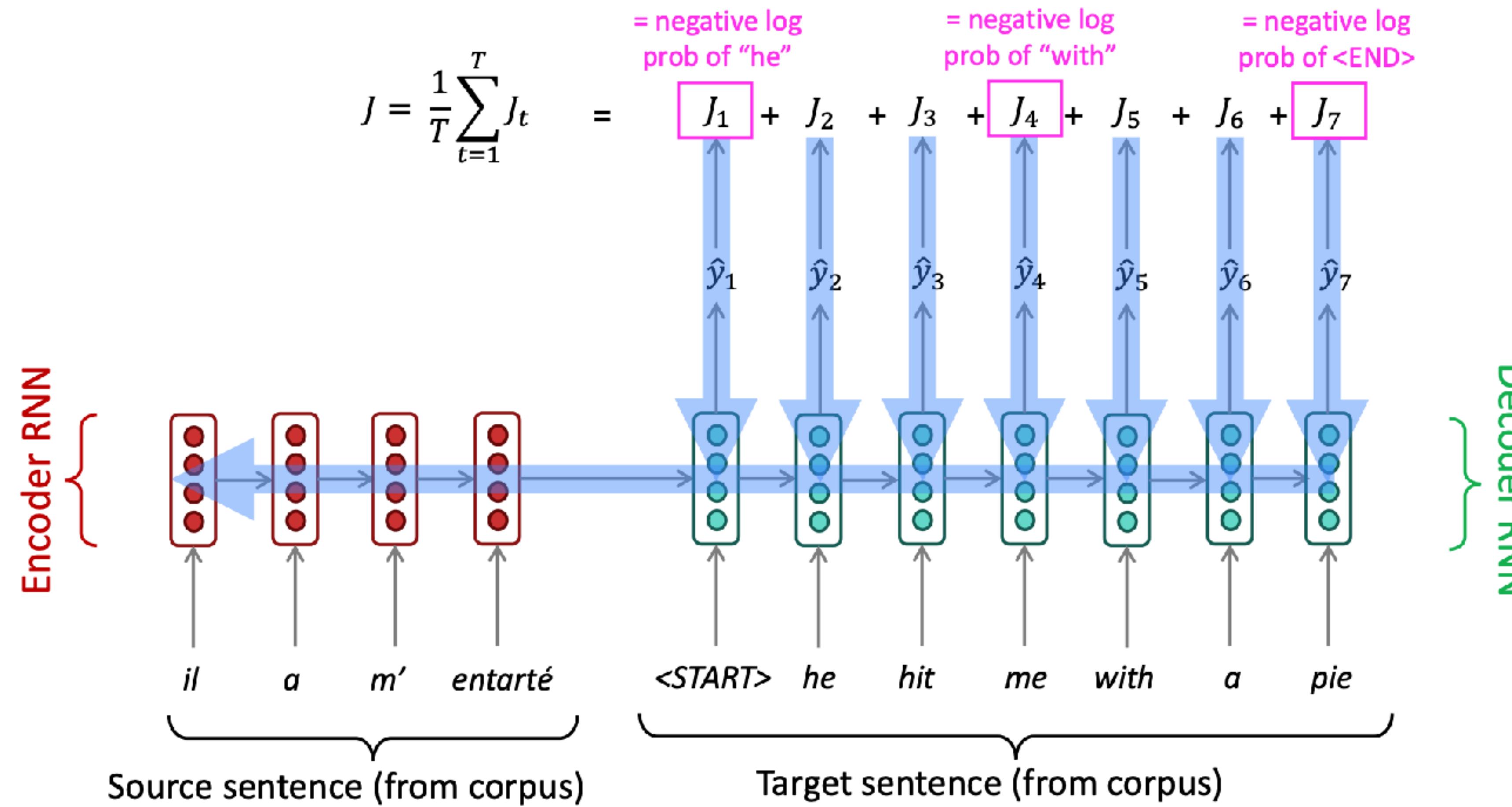
Этот вектор, созданный энкодером, содержит всю информацию о входной последовательности, и декодер использует его для генерации выходной последовательности.

Используем как h^0
для языковой модели
на другом языке



Sequence-to-sequence model

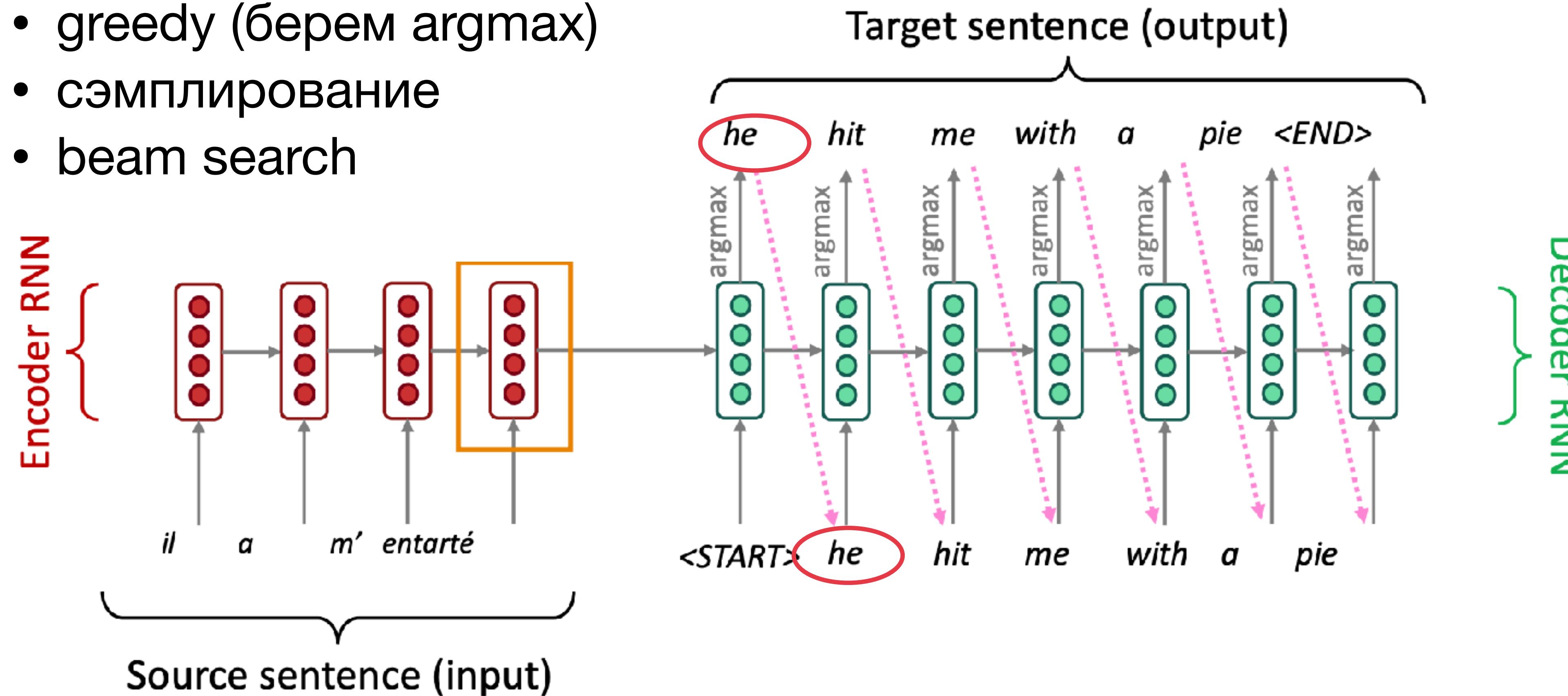
Обучение: нужен параллельный корпус



Sequence-to-sequence model

Генерация:

- greedy (берем argmax)
- сэмплирование
- beam search



Sequence-to-sequence model

Informational bottleneck:

В одном векторе должна быть вся информация

