

Глубинное обучение

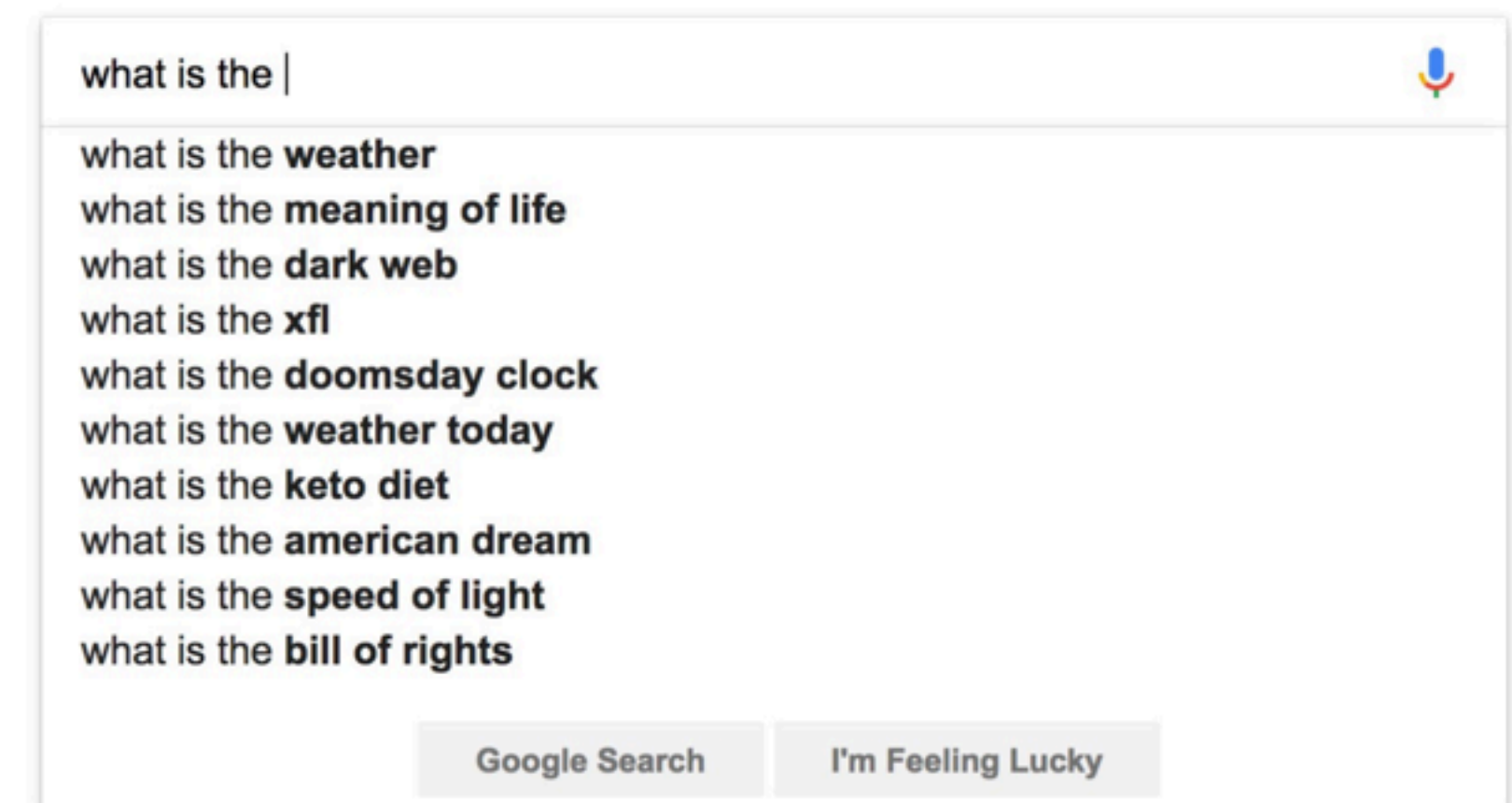
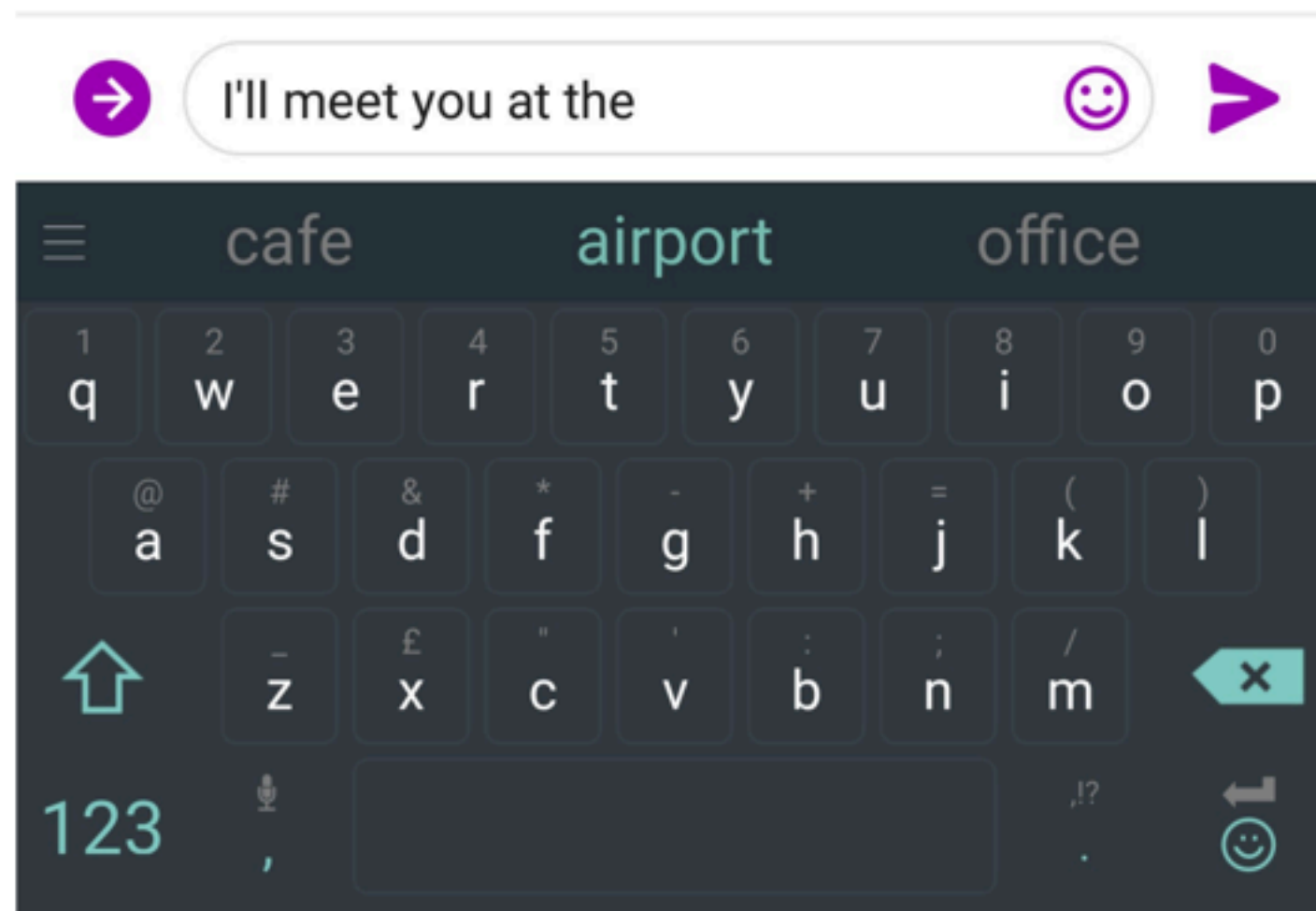
Обработка естественного языка: RNN, LSTM, Seq2seq

Лазарев Михаил

Языковые модели (LM)

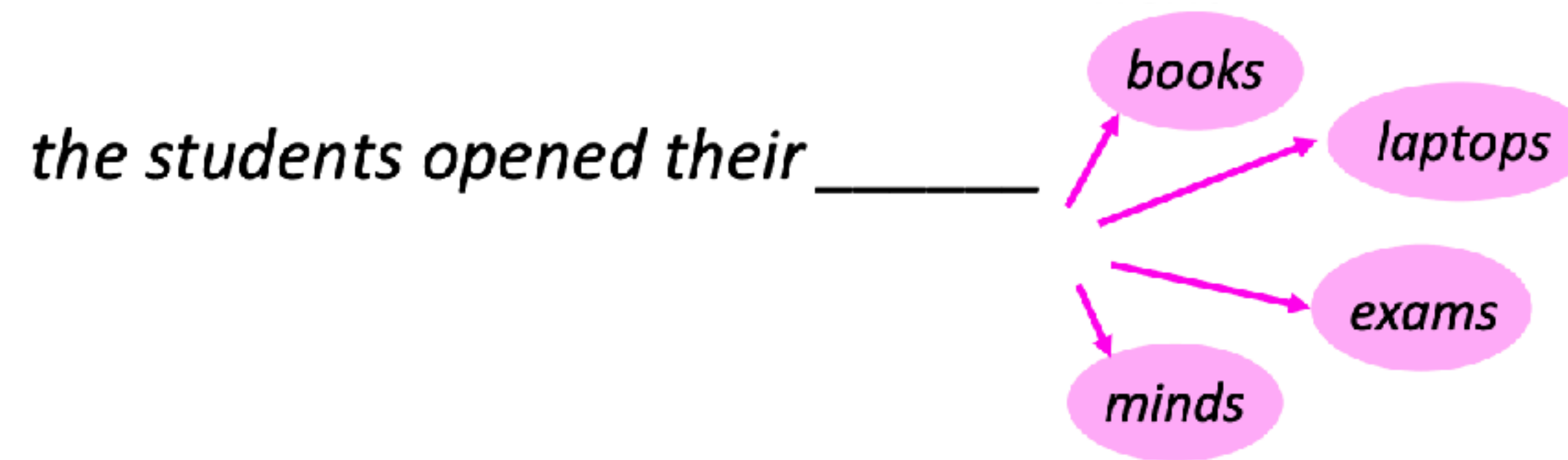
Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим



Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим

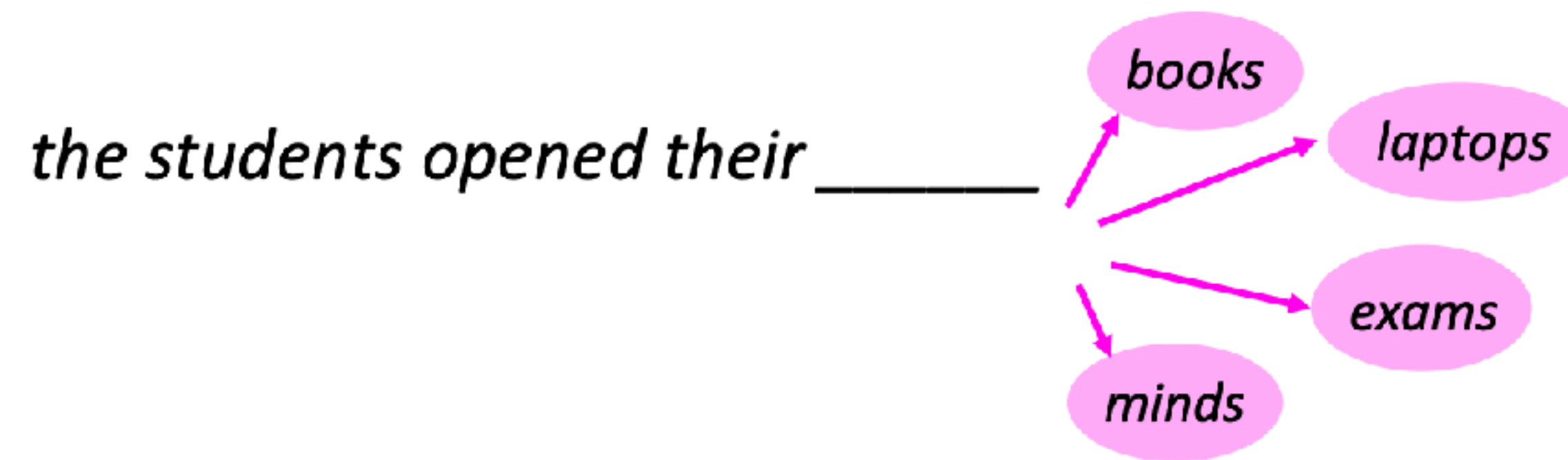


Авторегрессионная модель

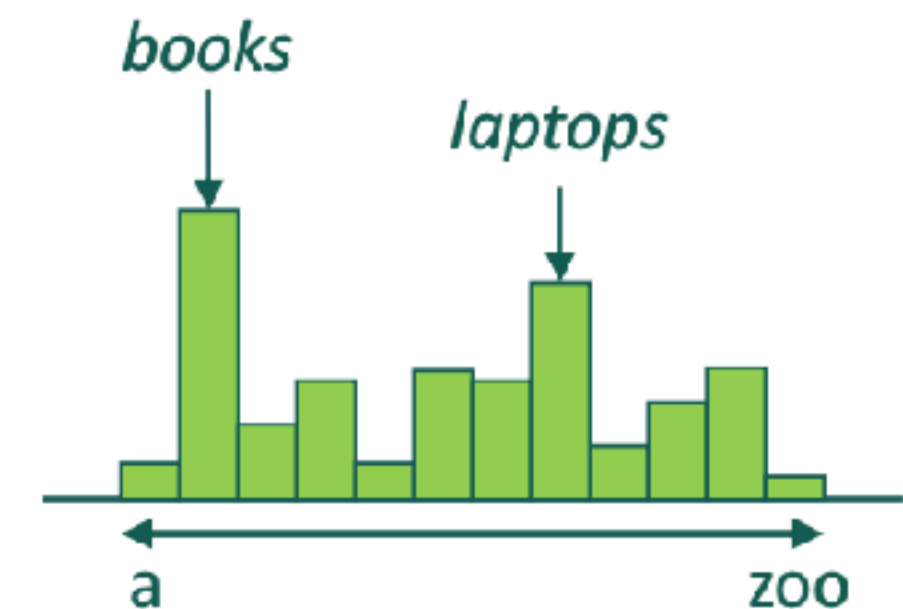
$$p(x) = p(x_n | x_1, \dots, x_{n-1}) \cdot p(x_2 | x_1) \cdot p(x_1) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$

Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим

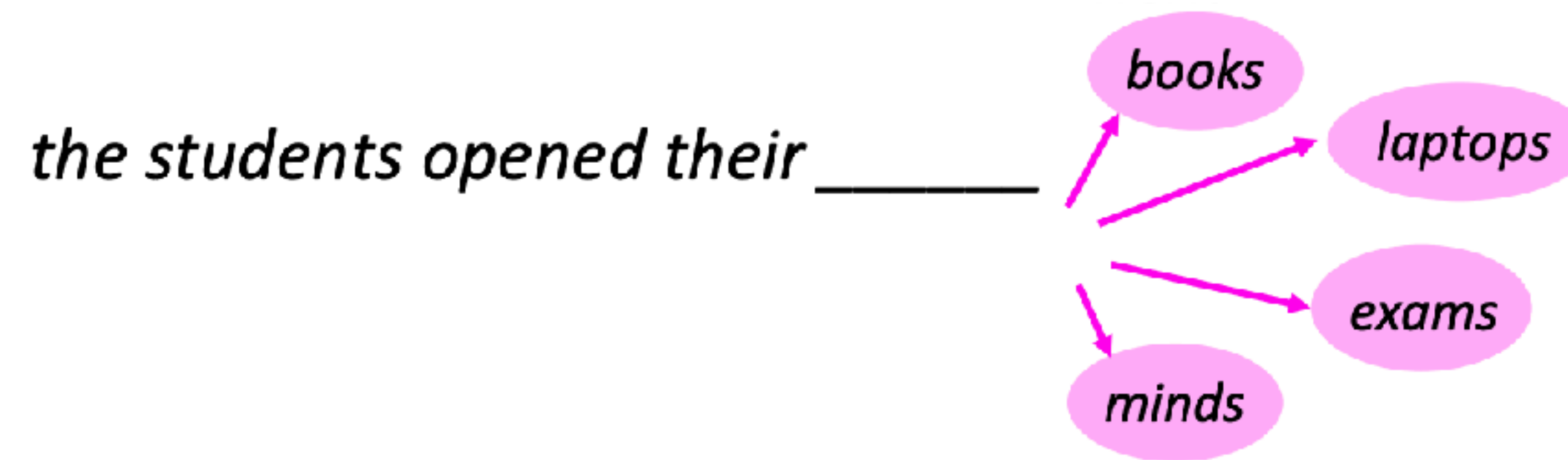


На выходе 5го шага: $p(x_5 | the, students, opened, their), x_5 \in Vocab$



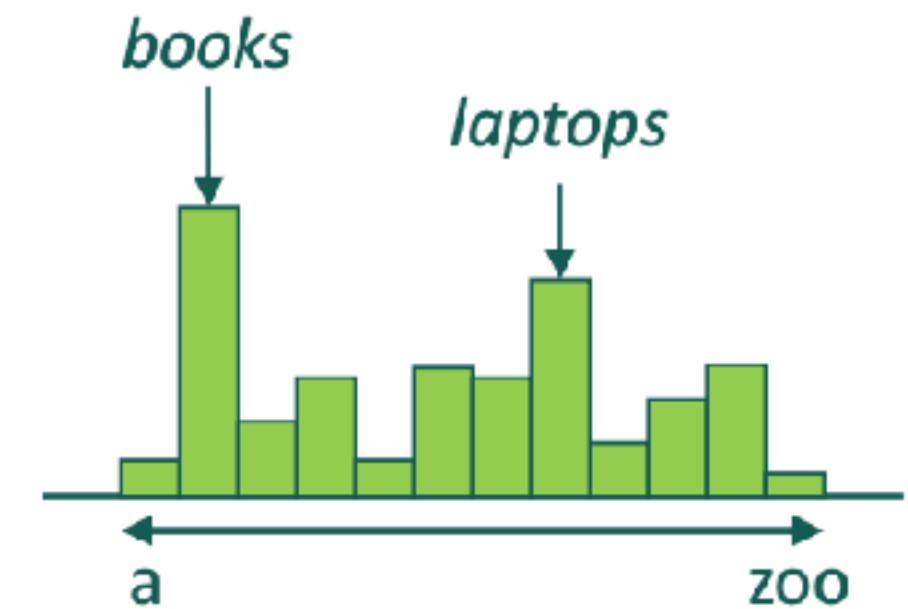
Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим



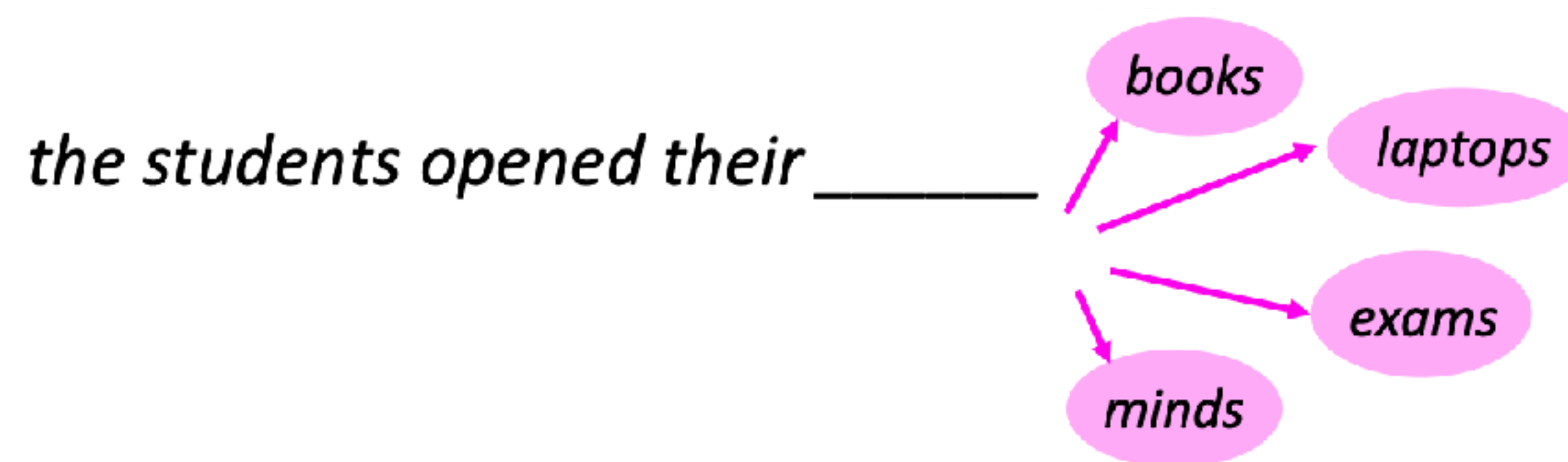
На выходе 5го шага: $p(x_5 | the, students, opened, their), x_5 \in Vocab$

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



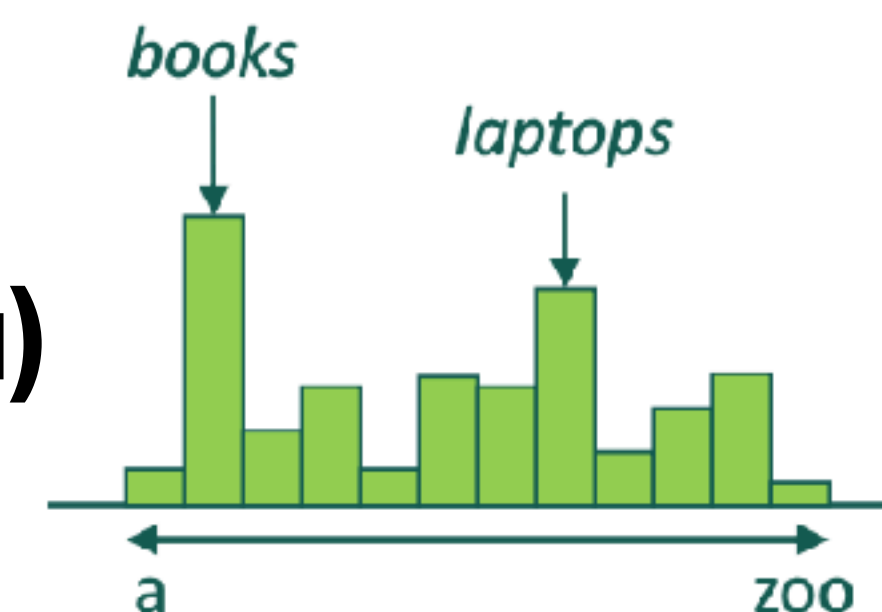
Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим



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

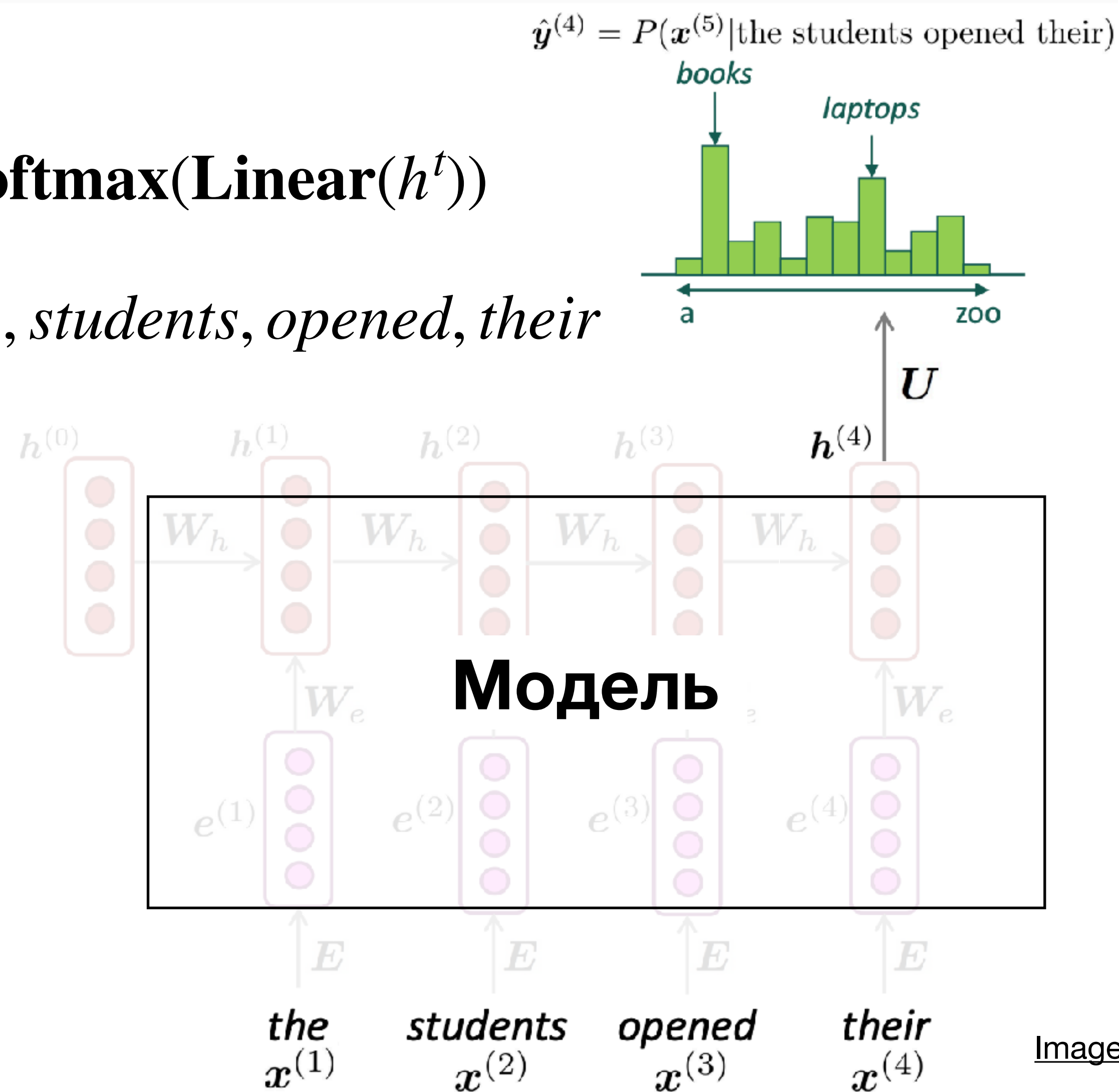
Каждый шаг - классификация (выбираем слово из словаря)



Рекуррентные нейросети (RNN)

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

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

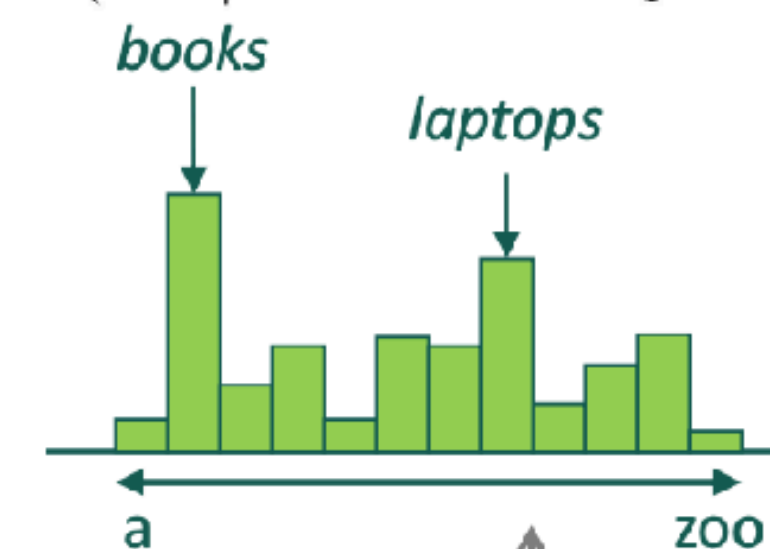


Рекуррентные нейросети (RNN)

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

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

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



U - веса Linear

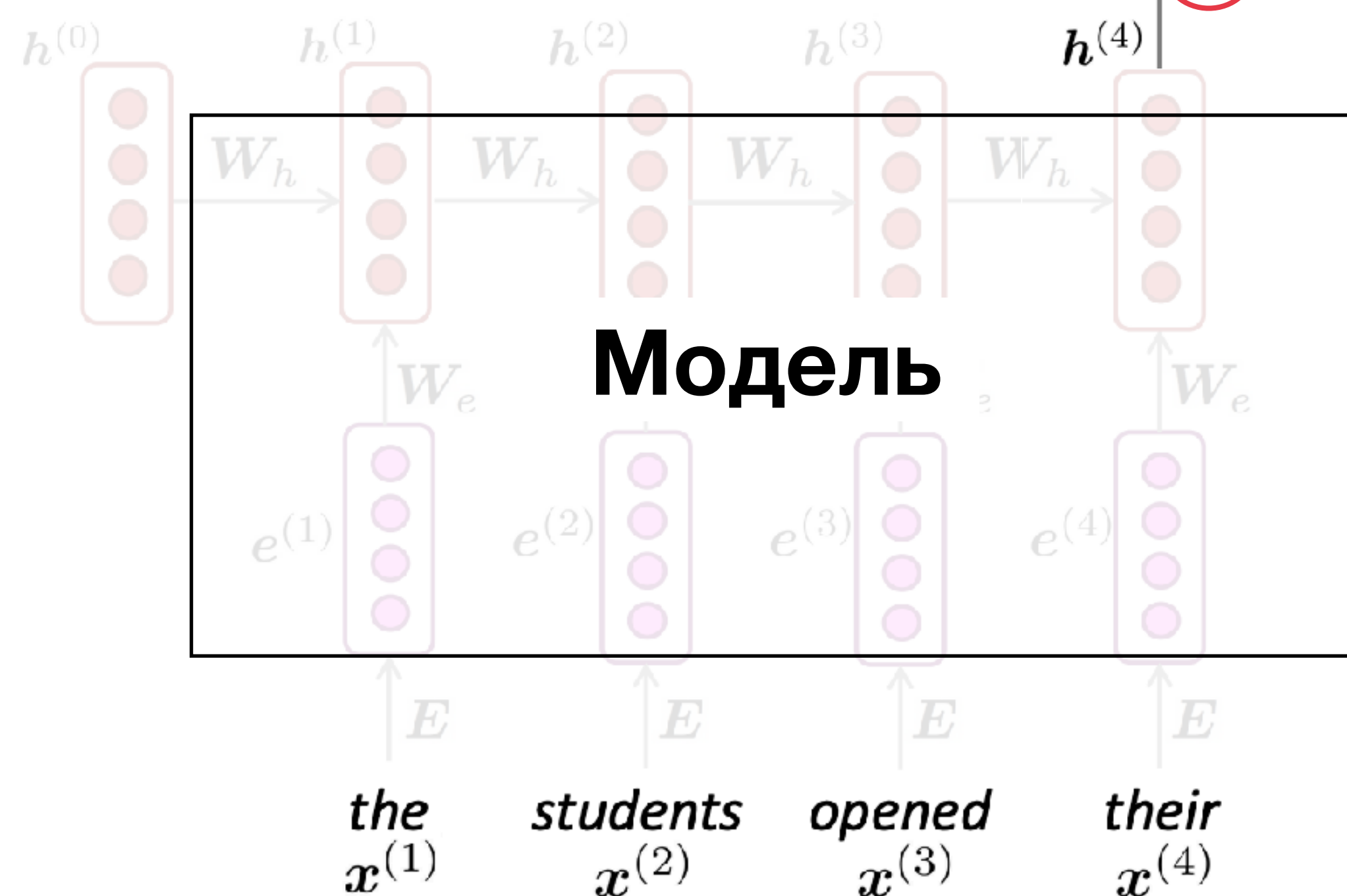


Image credit

Рекуррентные нейросети (RNN)

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

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

Как получить представления?

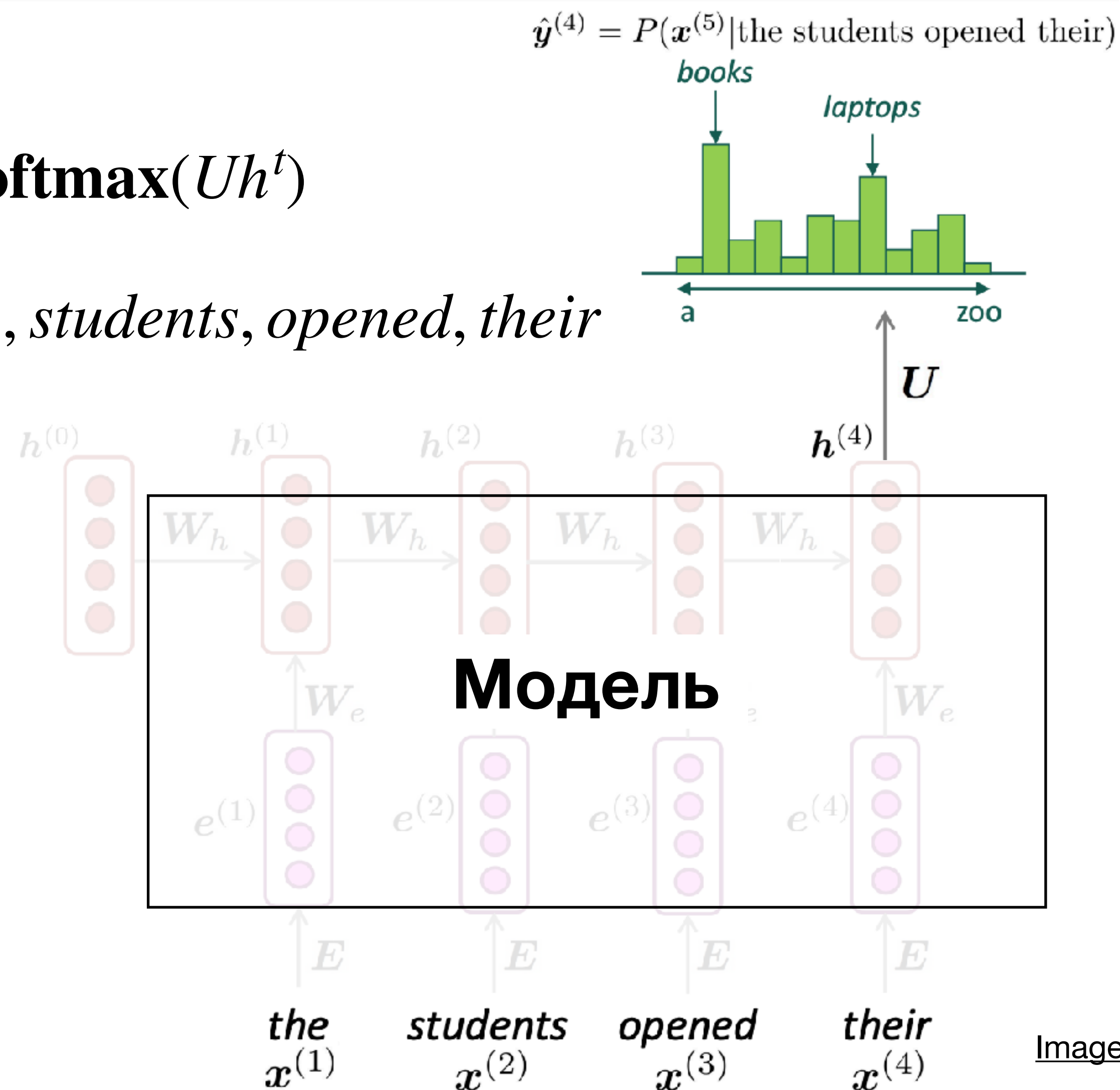


Image credit

Рекуррентные нейросети (RNN)

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

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)

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

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*

$$h^0 = 0 \quad \begin{matrix} h^{(0)} \\ \boxed{\begin{matrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{matrix}} \end{matrix}$$


the
 $x^{(1)}$

Image credit

Рекуррентные нейросети (RNN)

Шаг 1: пусть первое слово известно - *the*

Хотим предсказать второе - *students*

$$h^0 = 0$$


Возьмем эмбединг word2vec e^1 для *the*

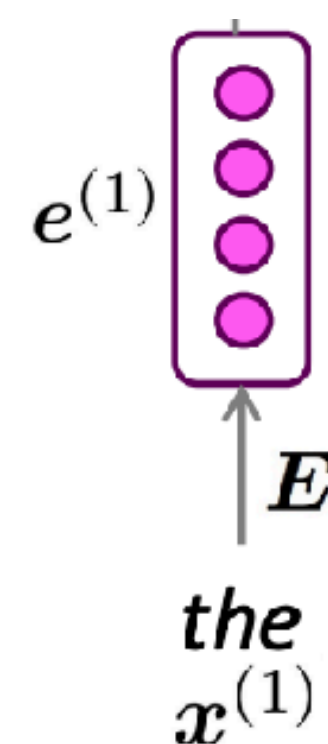


Image credit

Рекуррентные нейросети (RNN)

Шаг 1: пусть первое слово известно - *the*

Хотим предсказать второе - *students*

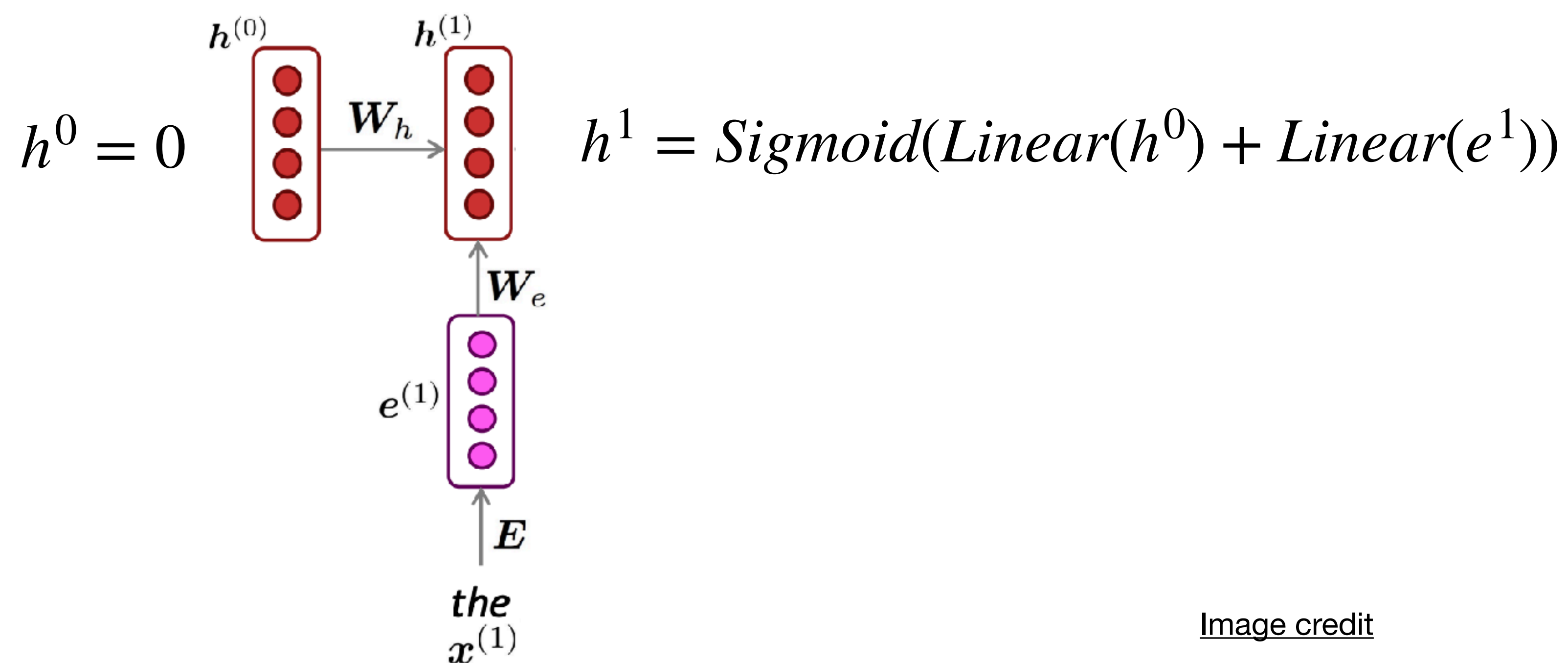


Image credit

Рекуррентные нейросети (RNN)

Шаг 1: пусть первое слово известно - *the*

Хотим предсказать второе - *students*

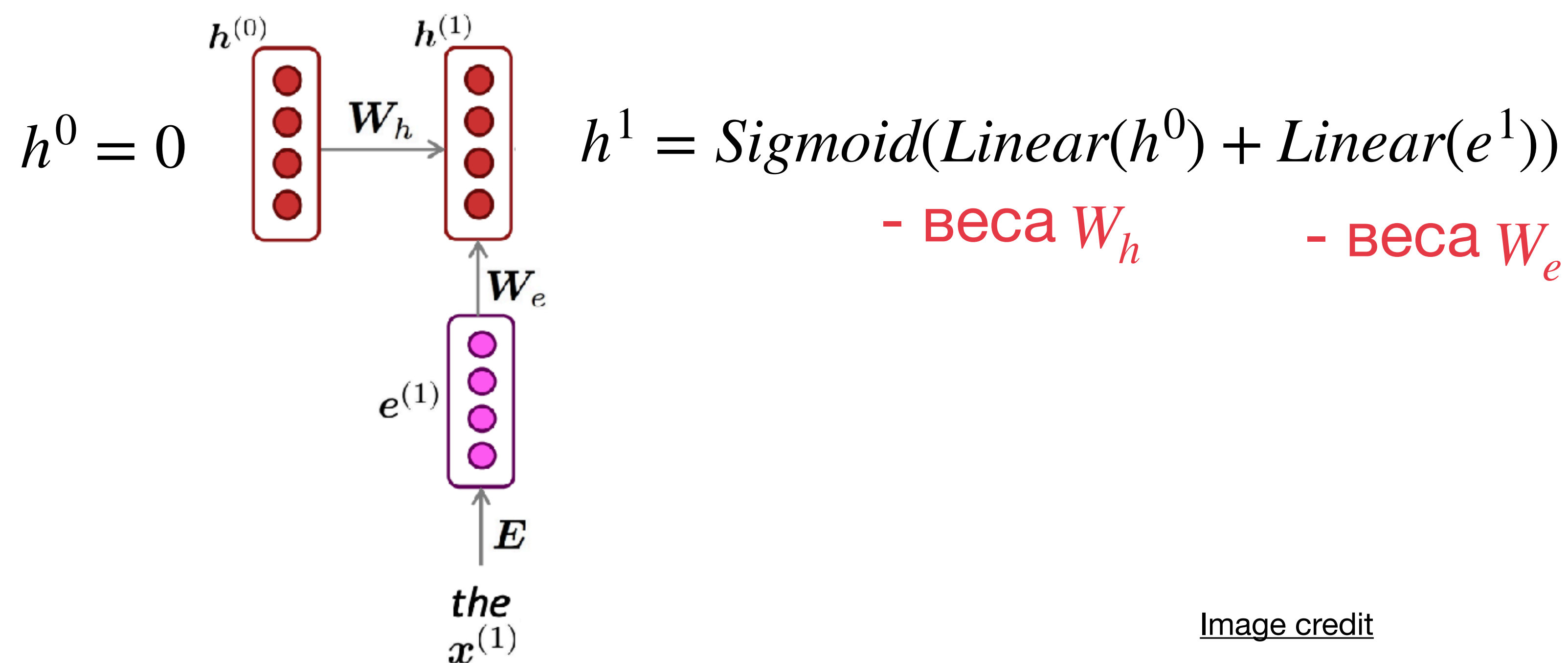


Image credit

Рекуррентные нейросети (RNN)

Шаг 1: пусть первое слово известно - *the*

Хотим предсказать второе - *students*

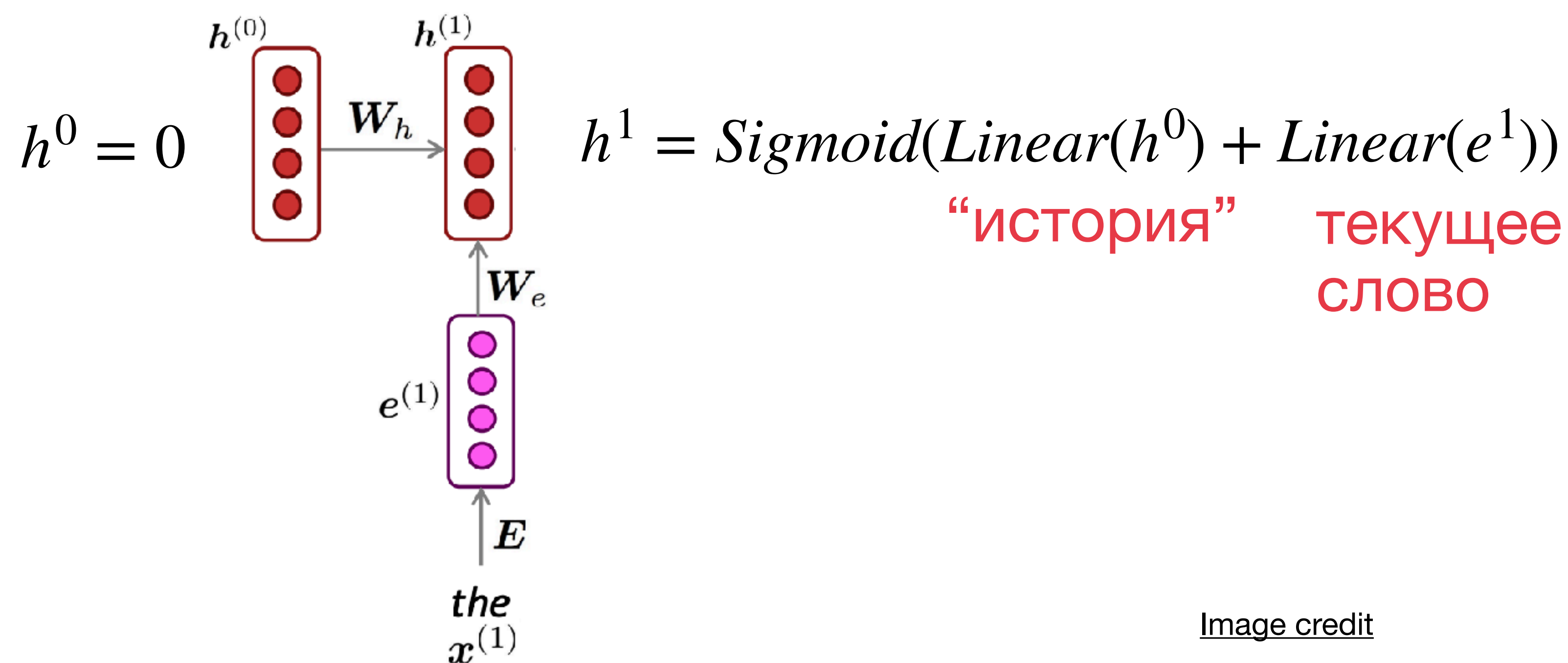


Image credit

Рекуррентные нейросети (RNN)

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

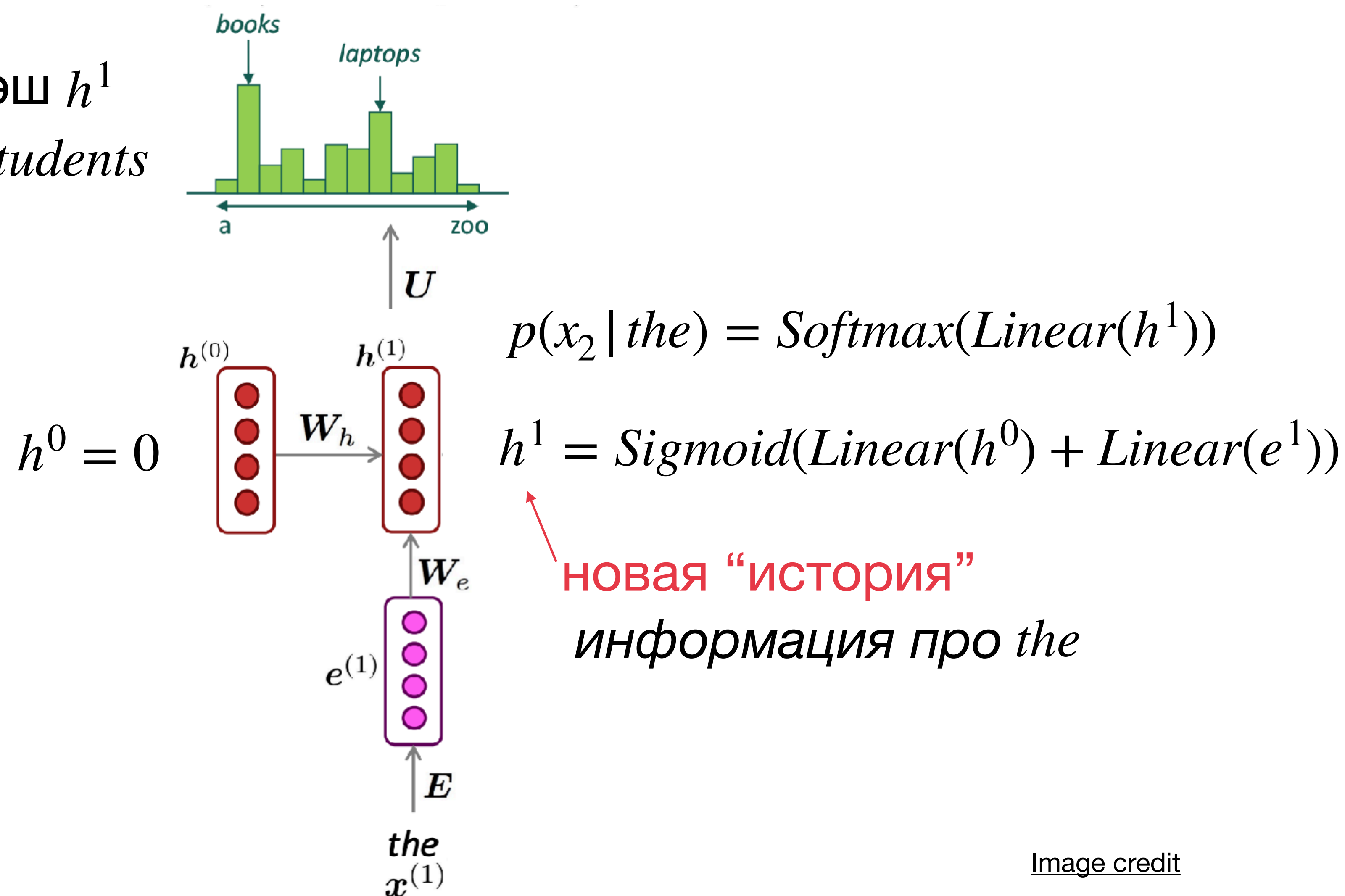


Image credit

Рекуррентные нейросети (RNN)

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

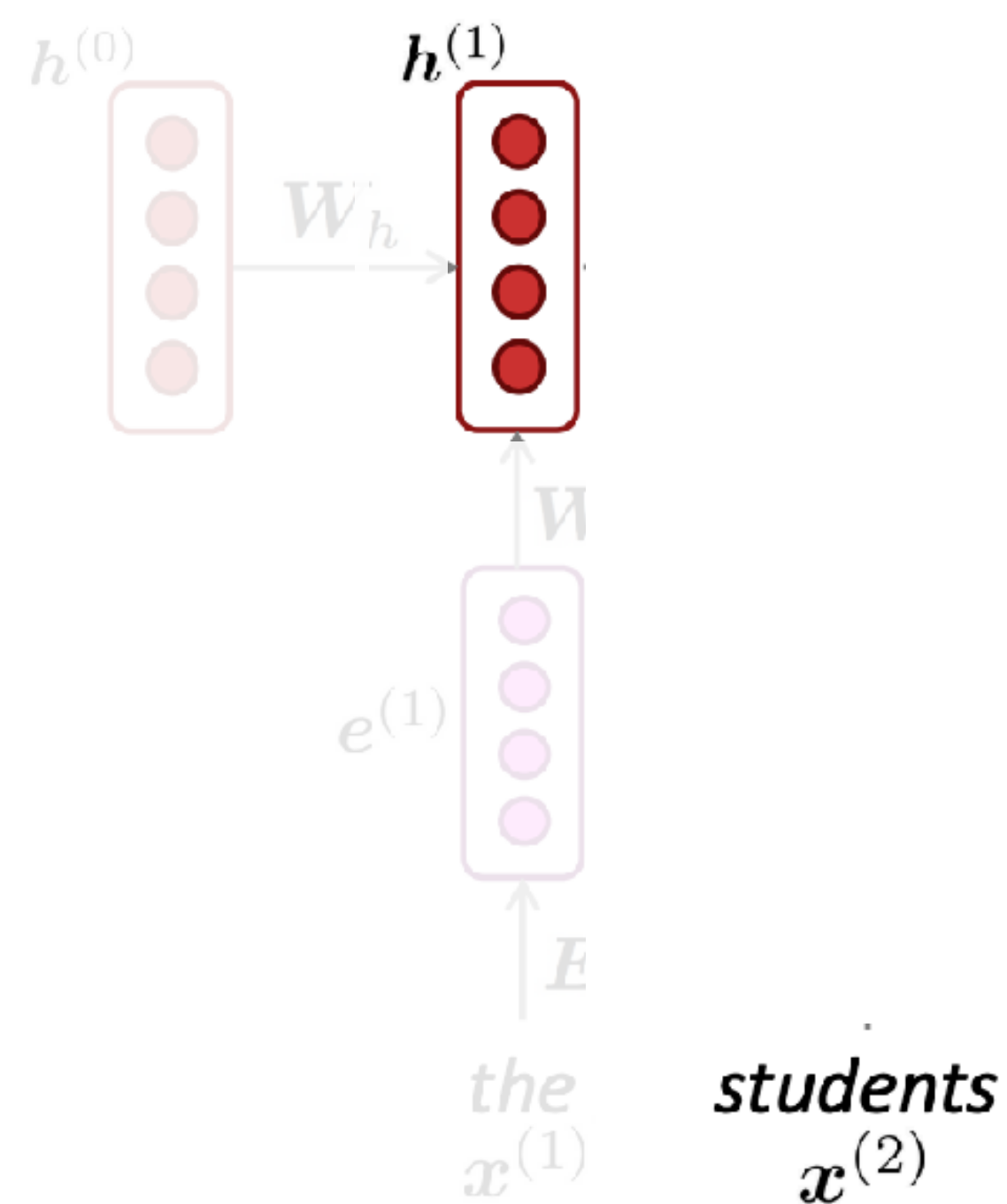


Image credit

Рекуррентные нейросети (RNN)

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

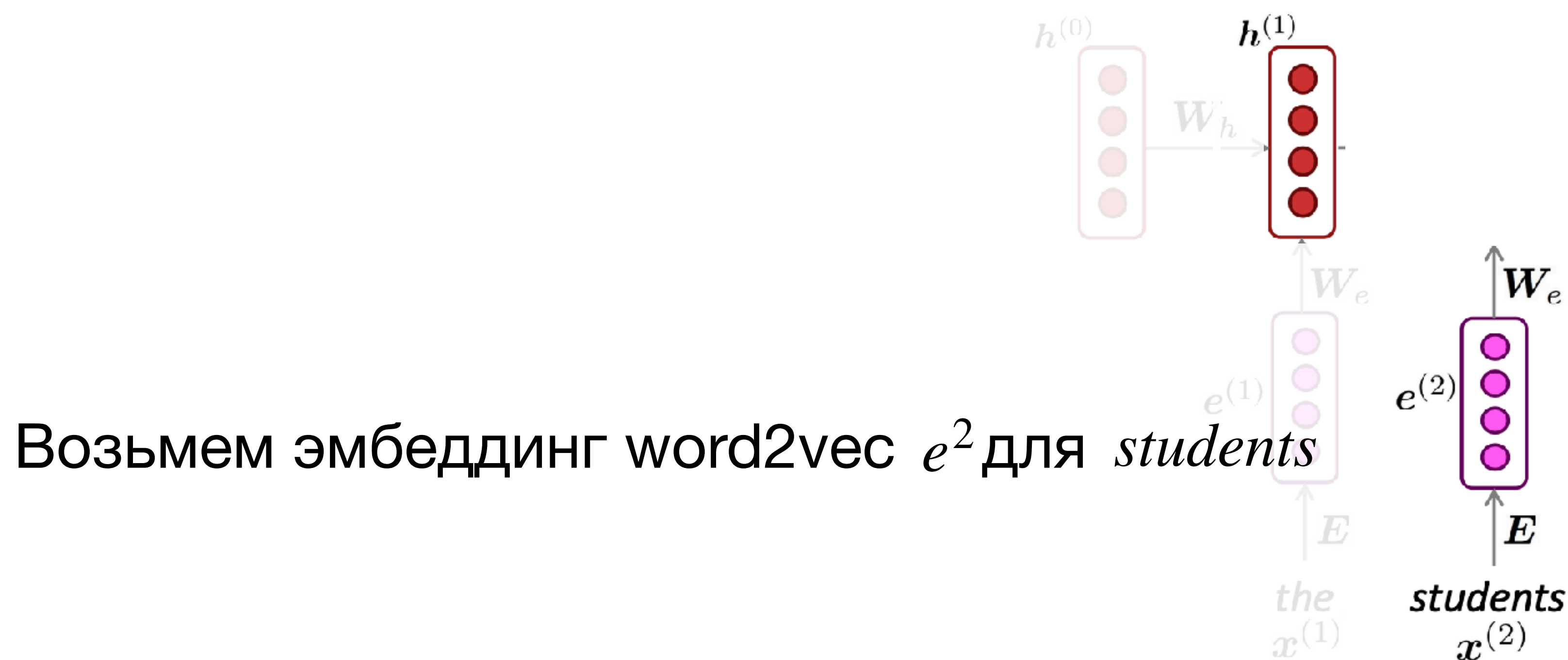


Image credit

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”
(*the*)

текущее
слово
(*students*)

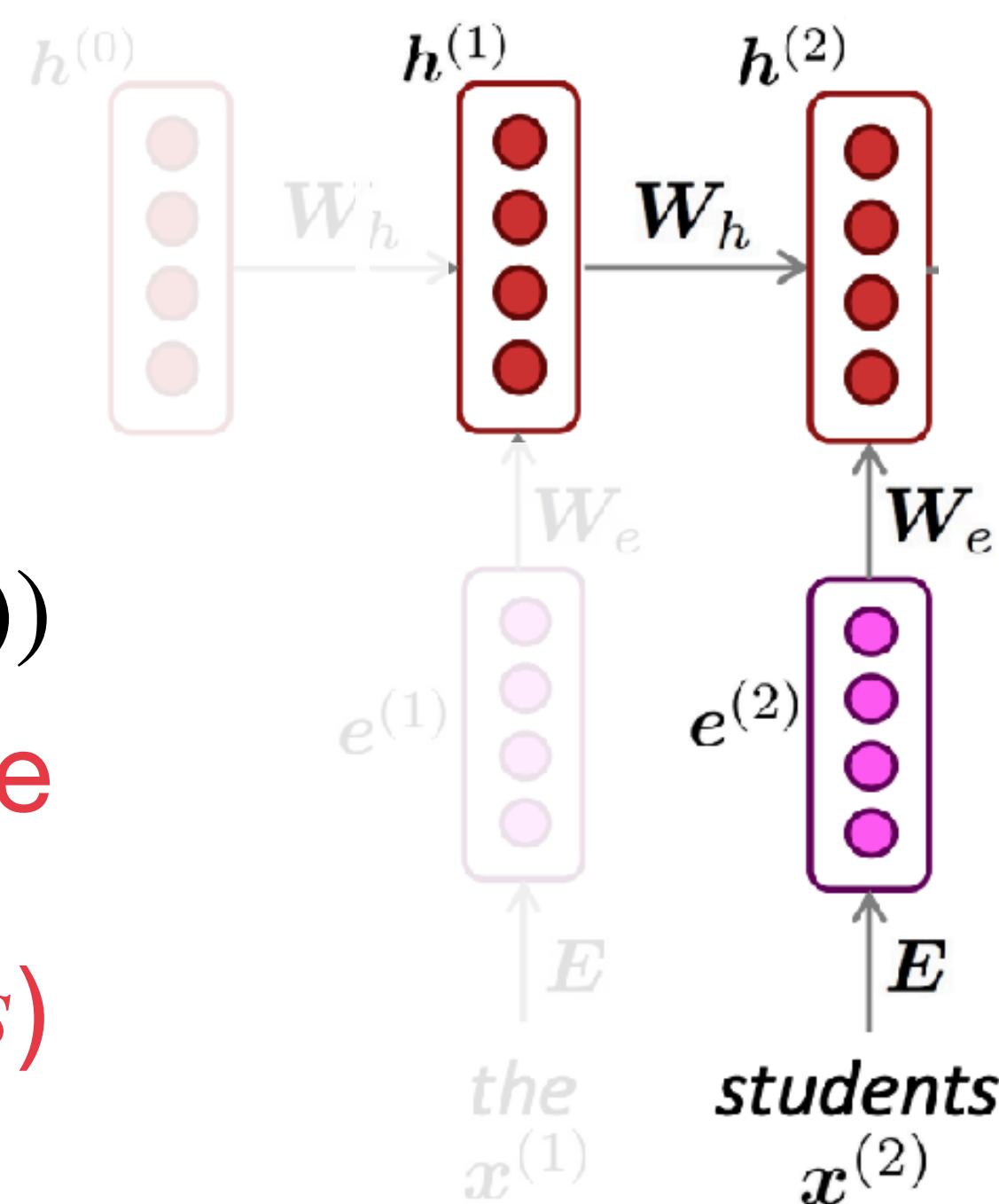


Image credit

Рекуррентные нейросети (RNN)

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

$$p(x_3 | the, students) = Softmax(Linear(h^2))$$

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

“история”
(*the*)

текущее
слово
(*students*)

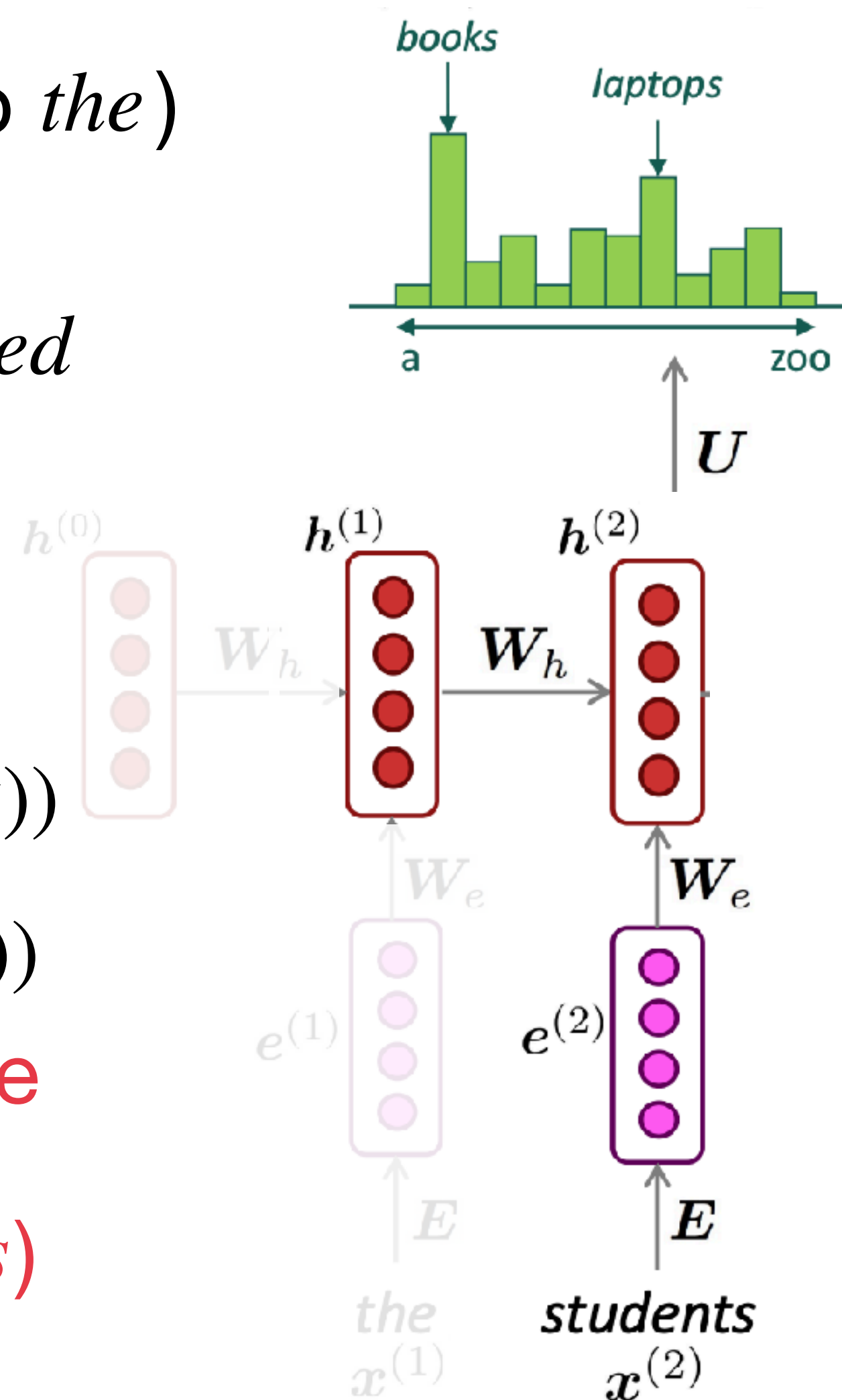


Image credit

Рекуррентные нейросети (RNN)

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

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

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

→ **НОВАЯ “ИСТОРИЯ”**

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

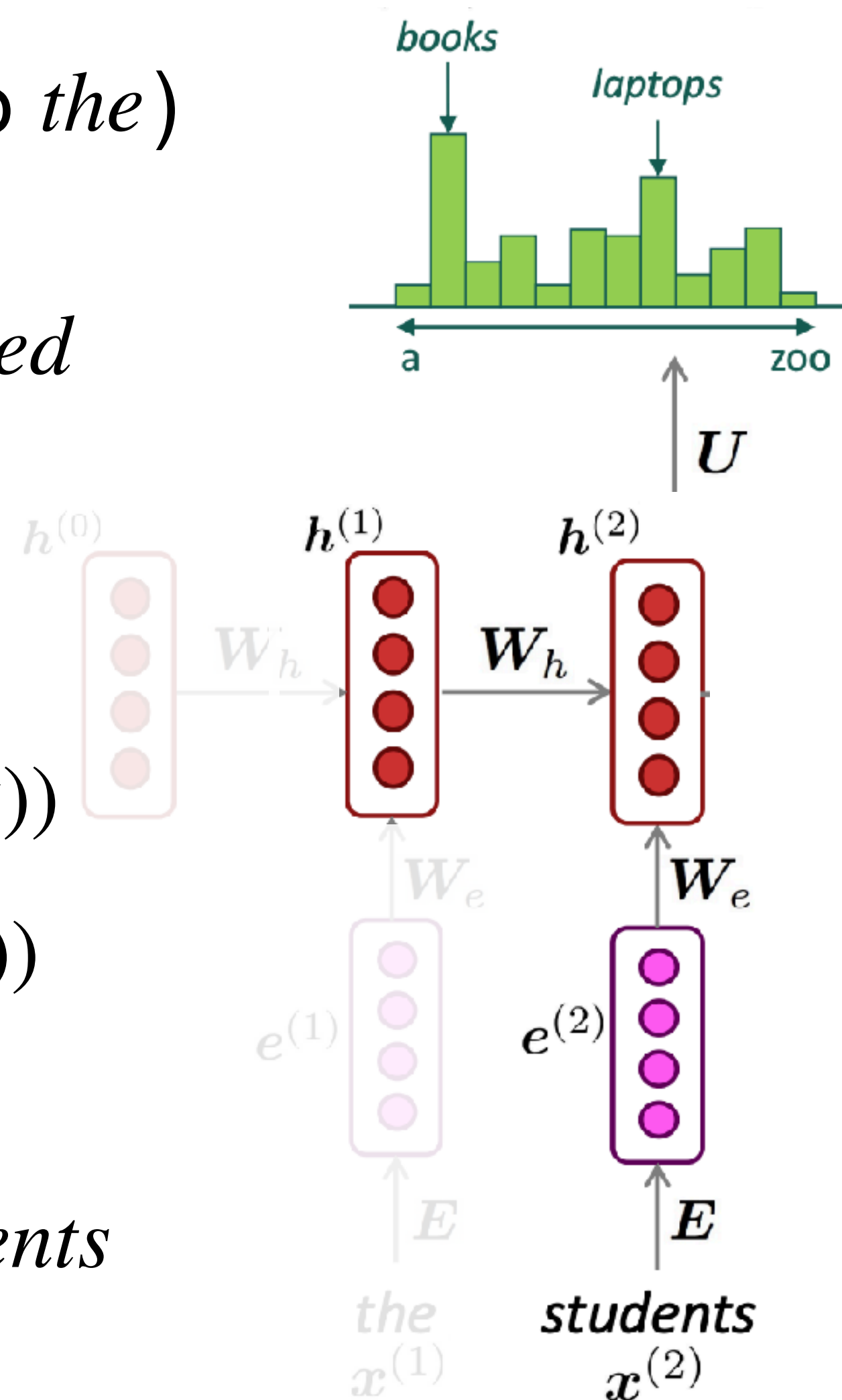


Image credit

Рекуррентные нейросети (RNN)

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

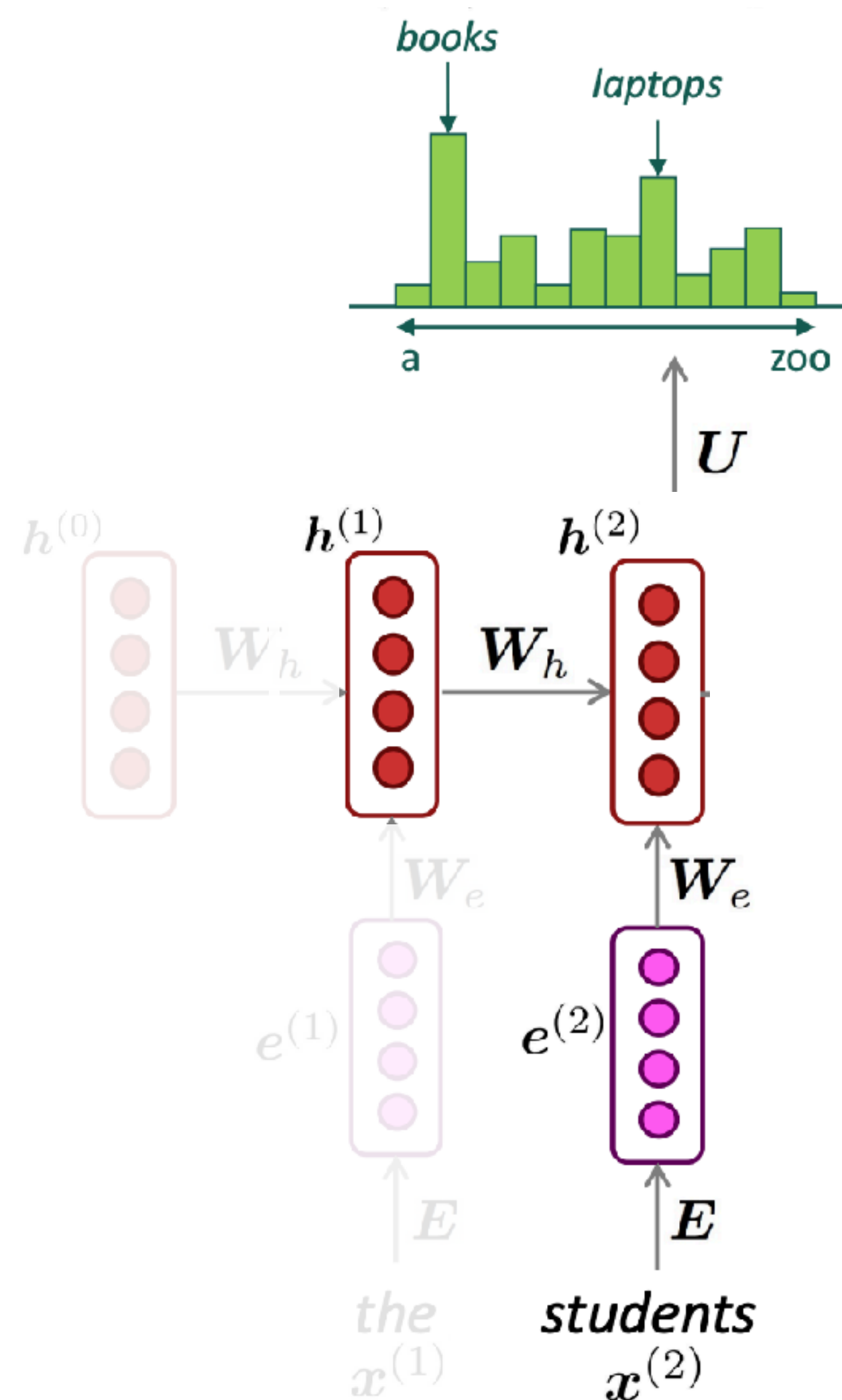


Image credit

Рекуррентные нейросети (RNN)

Два шага RNN

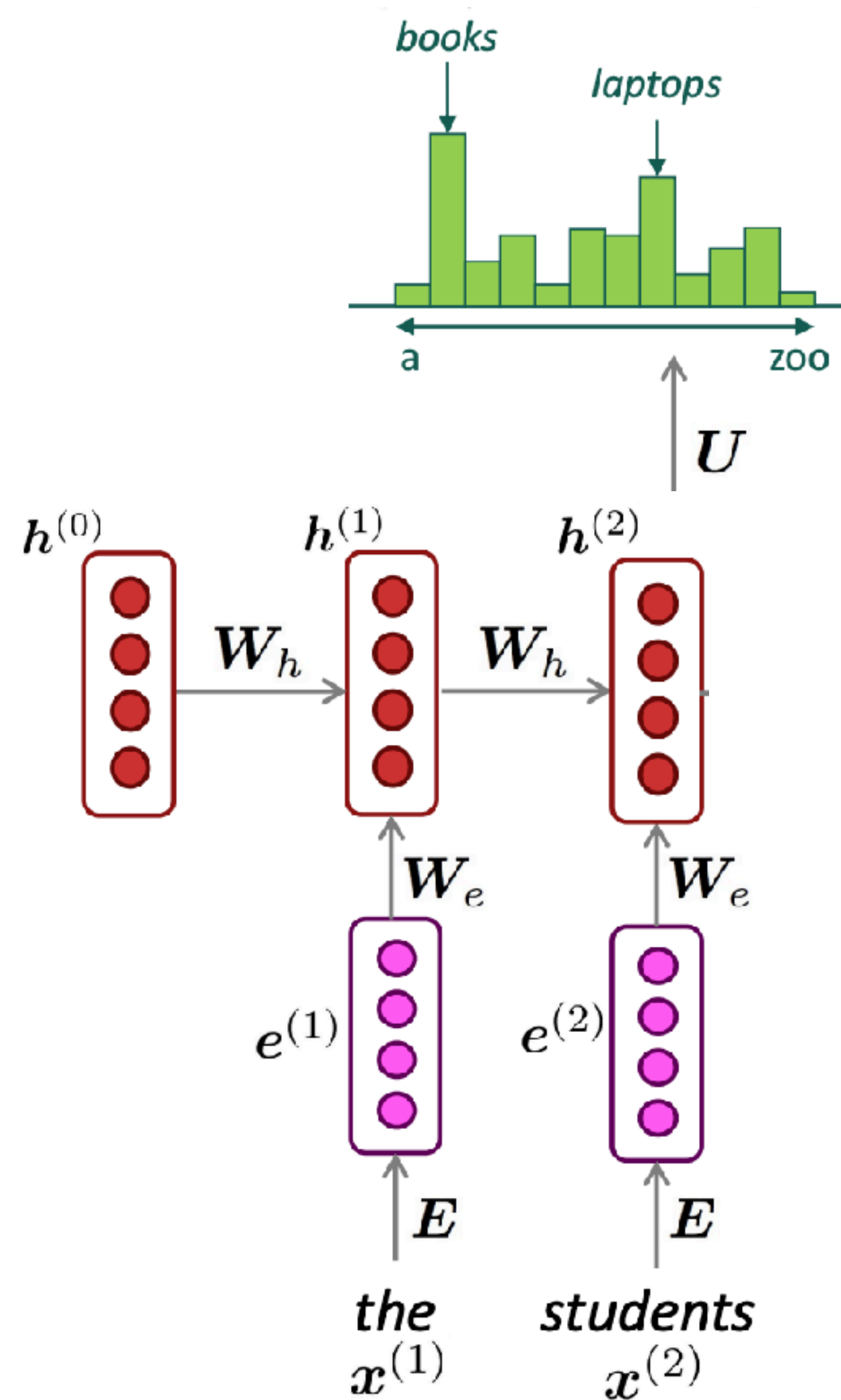


Image credit

Рекуррентные нейросети (RNN)

output distribution

$$\hat{y}^{(t)} = \text{softmax}(U h^{(t)} + b_2) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$$

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

word embeddings

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

words / one-hot vectors

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

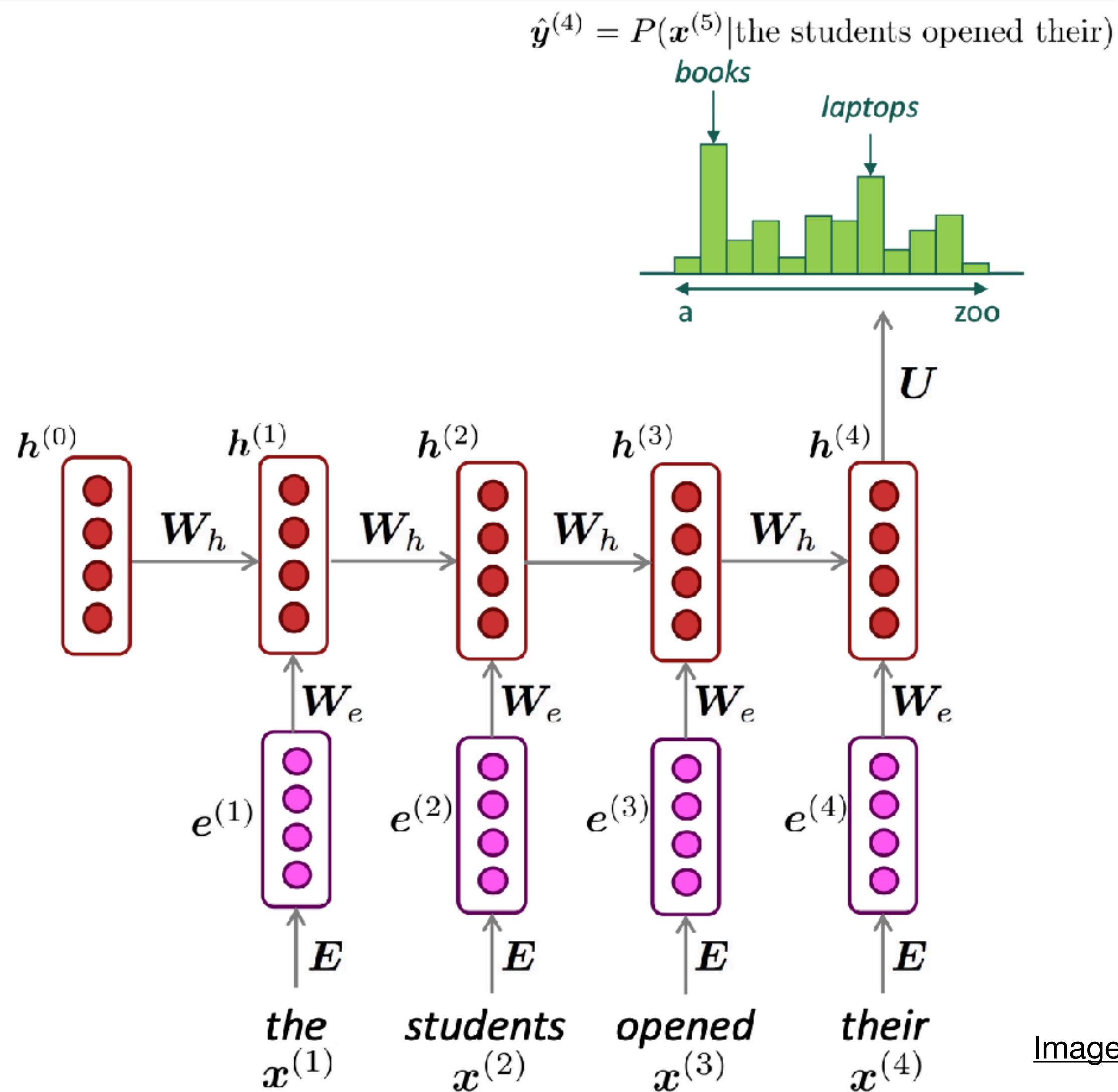


Image credit

Рекуррентные нейросети (RNN)

Обучение: лосс для классификации на каждом шаге

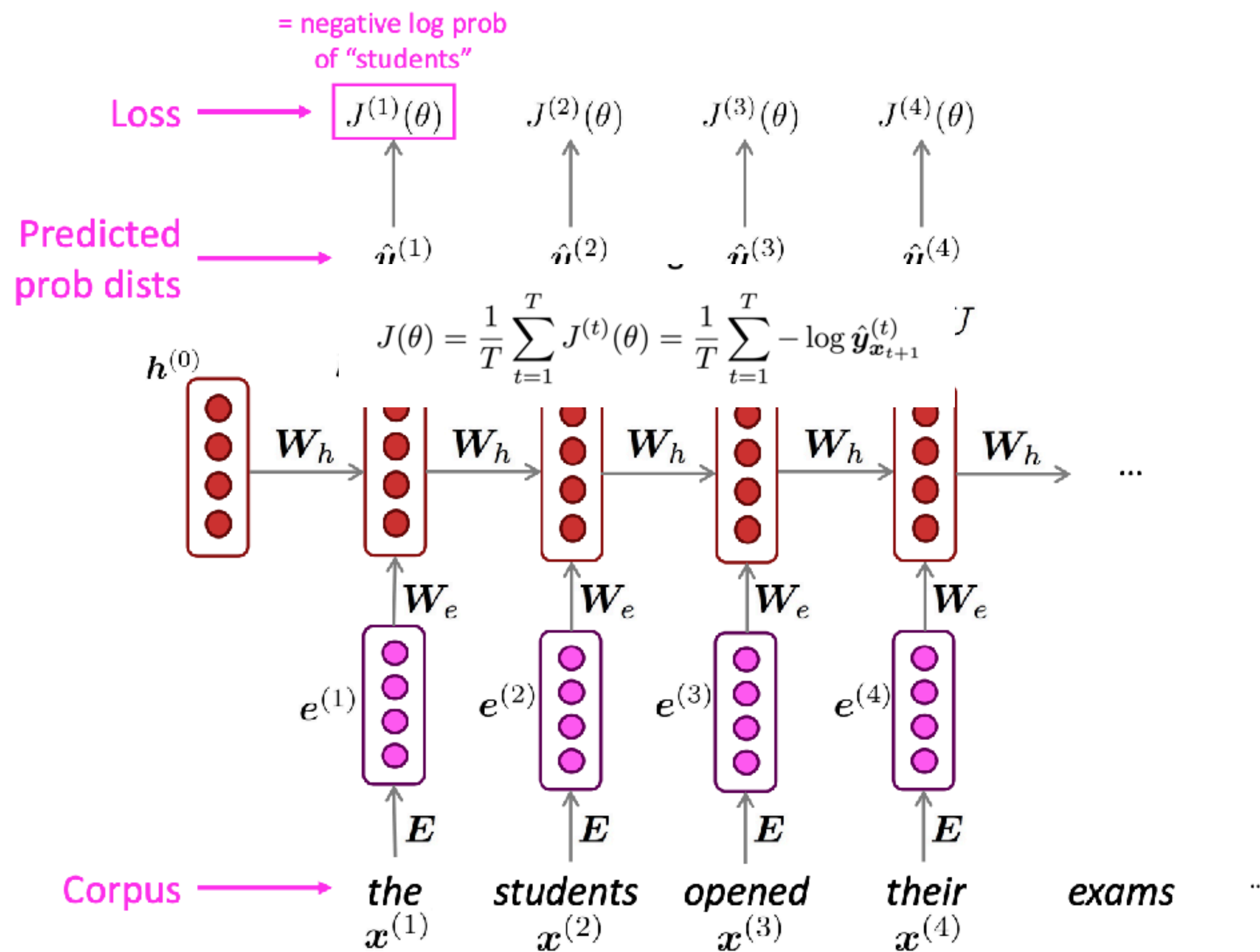


Image credit

Рекуррентные нейросети (RNN)

Обучение: лосс для классификации на каждом шаге

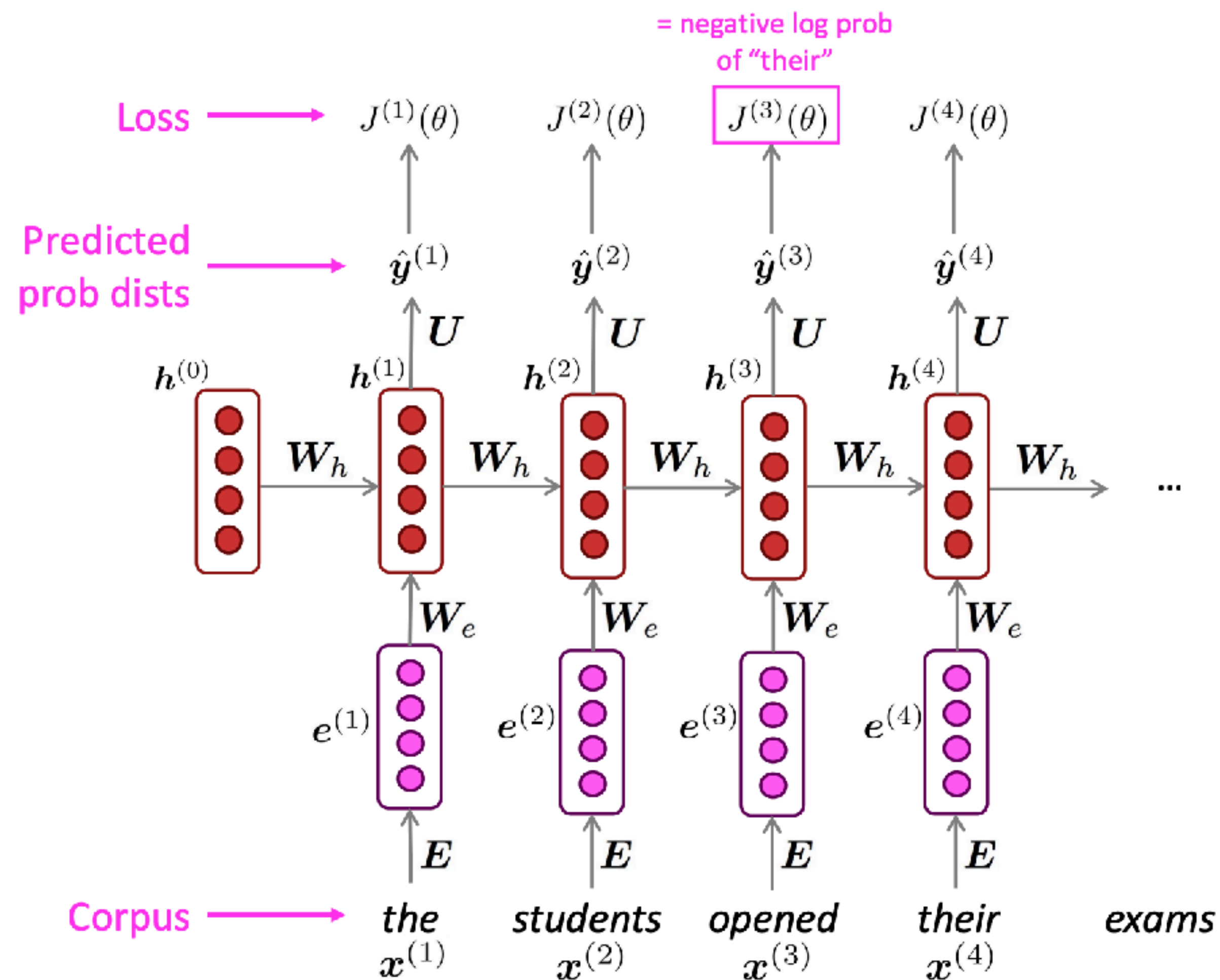


Image credit

Рекуррентные нейросети (RNN)

Обучение: лосс для классификации на каждом шаге

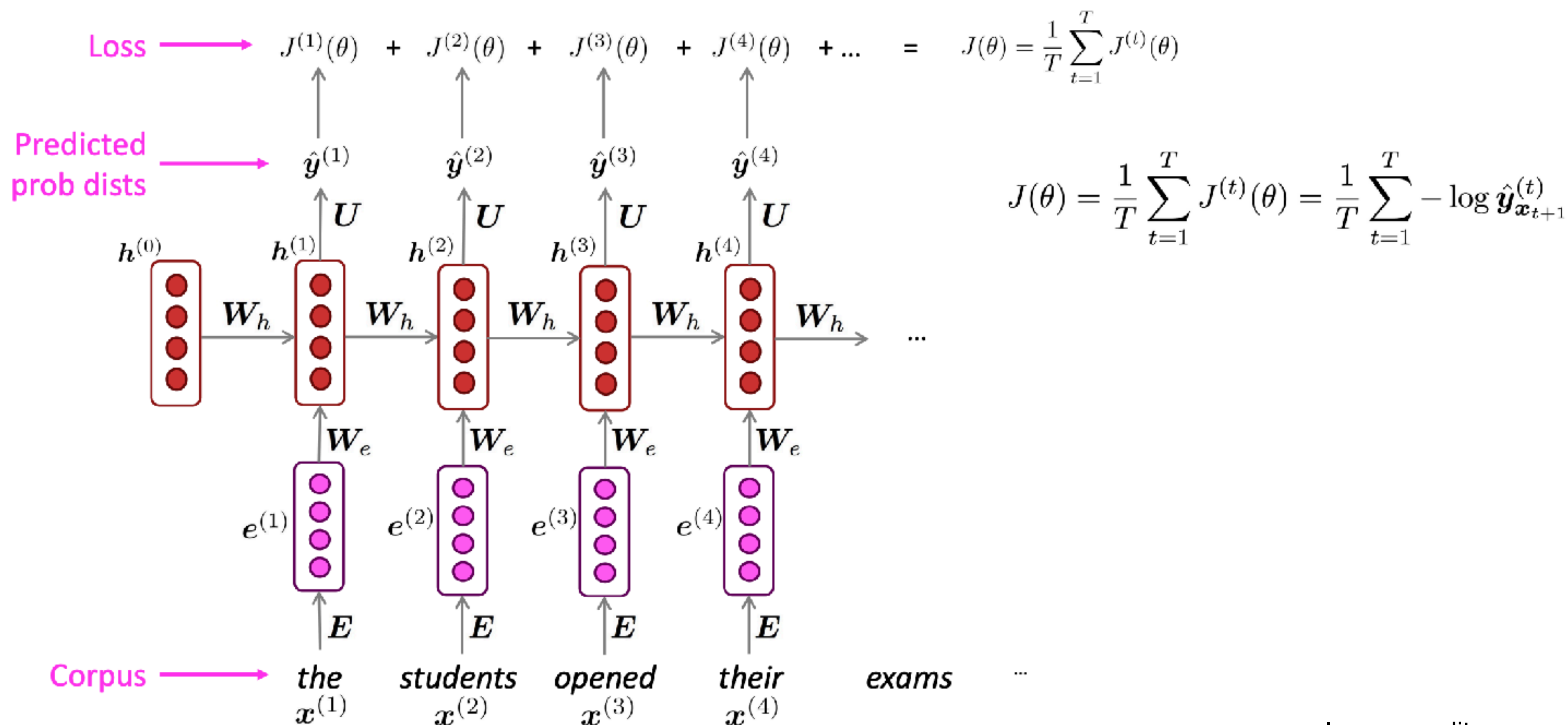
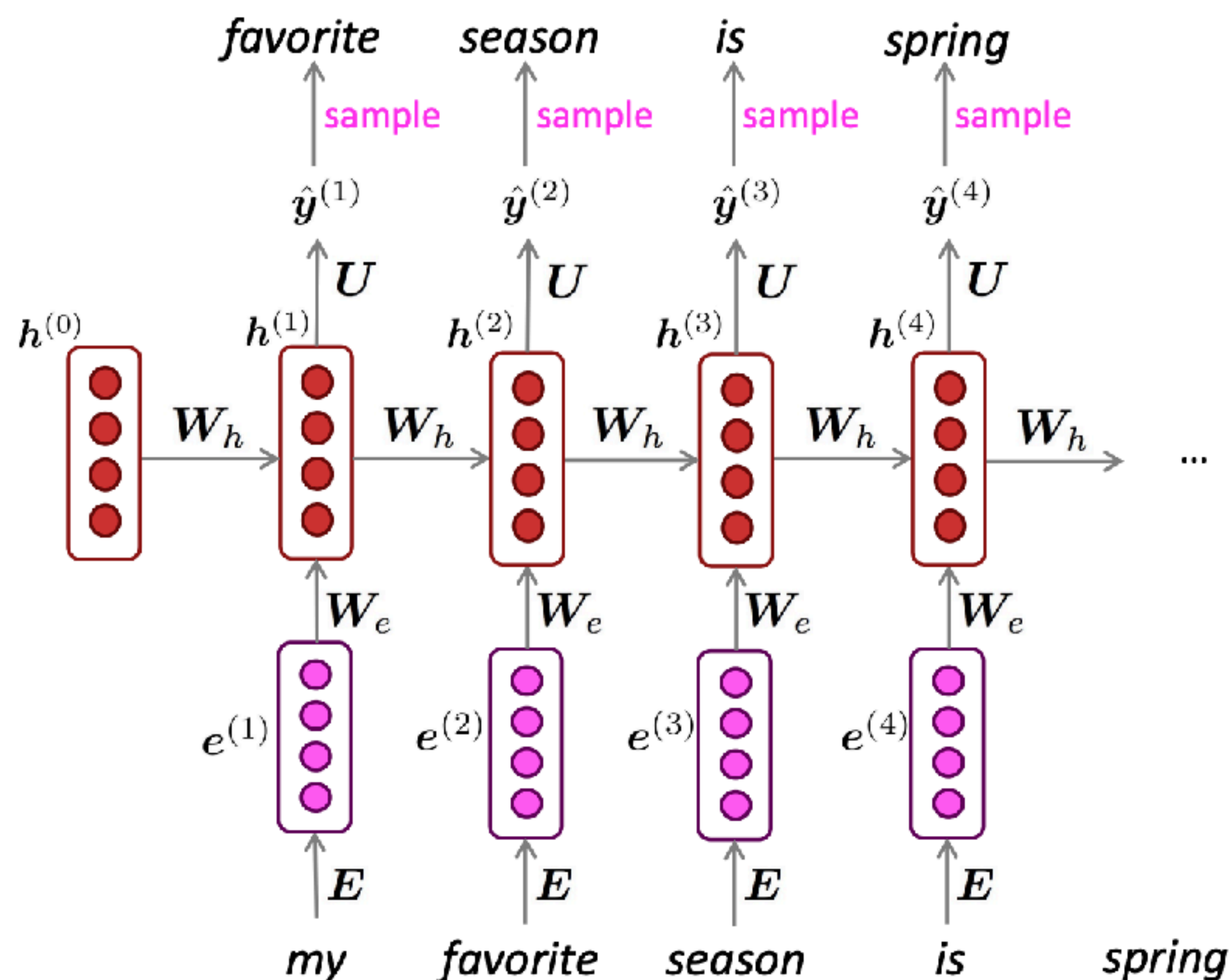


Image credit

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

$$\hat{y}^{(t)} = \text{softmax}(U h^{(t)} + b_2) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$$

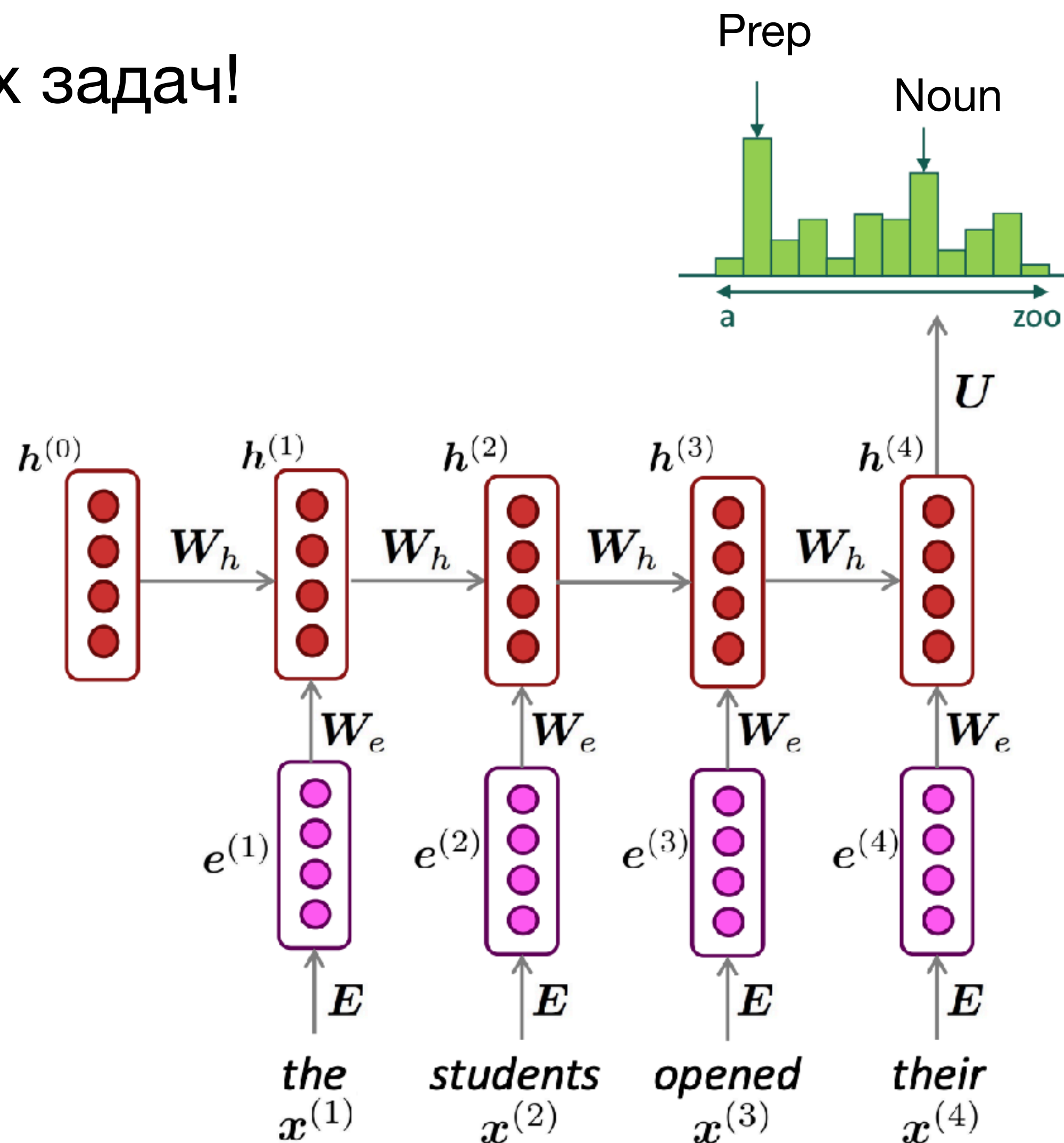
$h^{(0)}$ is the initial hidden state

word embeddings

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

words / one-hot vectors

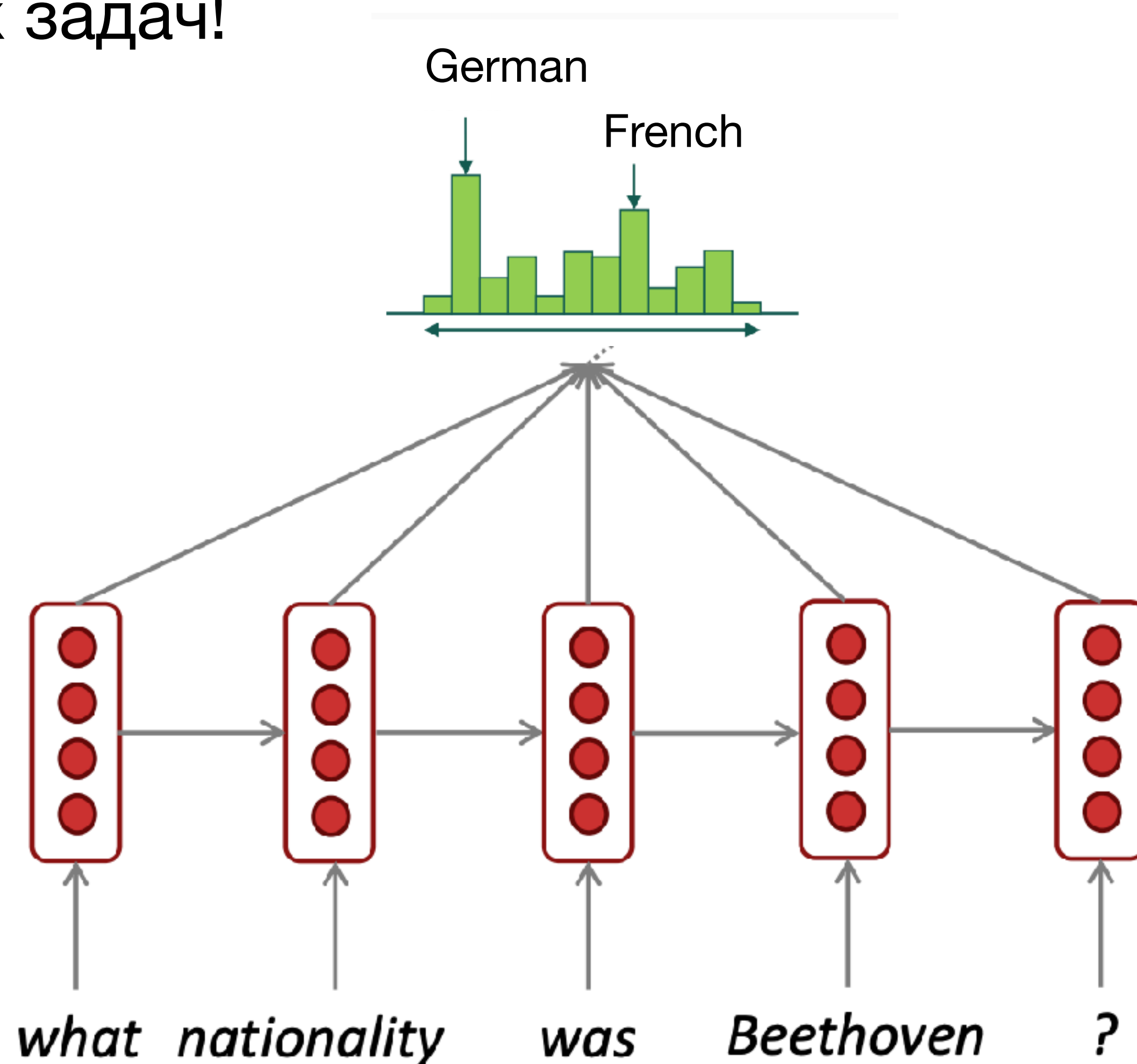
$$x^{(t)} \in \mathbb{R}^{|V|}$$



Рекуррентные нейросети (RNN)

Можно использовать для других задач!

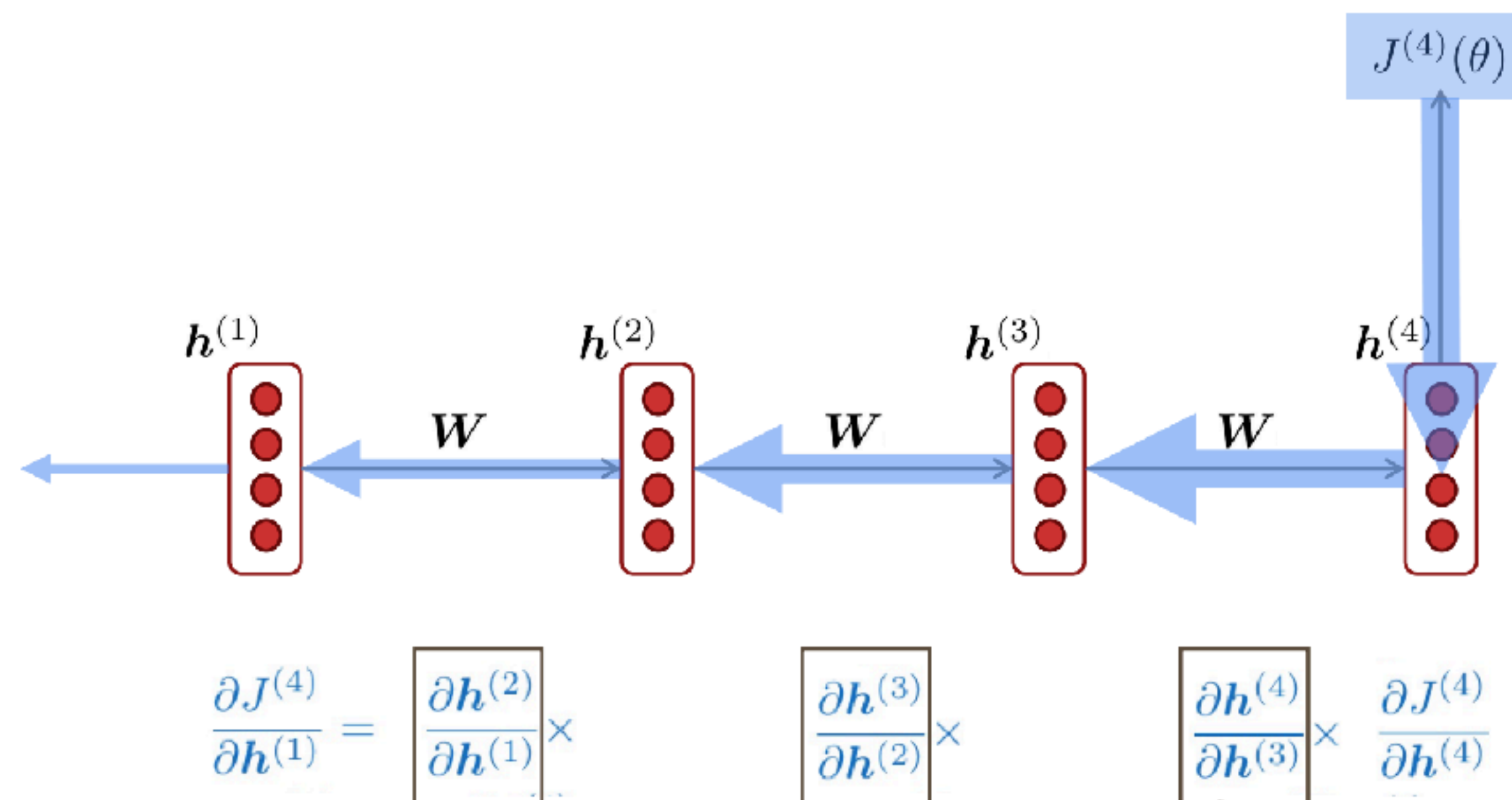
Sentence classification



Рекуррентные нейросети (RNN)

Проблемы?

- Медленно
- Vanishing gradients



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 _____*

Рекуррентные нейросети (RNN)

Проблемы?

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

Algorithm 1 Pseudo-code for norm clipping

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

LSTM

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

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

LSTM

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

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

Новая информация

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

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

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

All these are vectors of same length n

LSTM

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

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

Новая информация

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

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

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

All these are vectors of same length n

LSTM

Vanilla RNN постоянно переписывает $h^{(t)} = \sigma(W_h h^{(t-1)} + W_x x^{(t)} + b)$

Идея: добавить “память” $c^{(t)}$

Новая информация

$$\tilde{c}^{(t)} = \tanh(W_c h^{(t-1)} + U_c x^{(t)} + b_c)$$

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

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

All these are vectors of same length n

LSTM

Vanilla RNN постоянно переписывает $h^{(t)} = \sigma \left(W_h h^{(t-1)} + W_x x^{(t)} + b \right)$

Идея: добавить “память” $c^{(t)}$

Forget gate: контролируем сколько забыть

Input gate: контролируем сколько записать

$$f^{(t)} = \sigma \left(W_f h^{(t-1)} + U_f x^{(t)} + b_f \right)$$

$$i^{(t)} = \sigma \left(W_i h^{(t-1)} + U_i x^{(t)} + b_i \right)$$

Sigmoid function: all gate values are between 0 and 1

Новая информация

Можно записать новую и стереть старую

$$\tilde{c}^{(t)} = \tanh \left(W_c h^{(t-1)} + U_c x^{(t)} + b_c \right)$$

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

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

All these are vectors of same length n

LSTM

Vanilla RNN постоянно переписывает $h^{(t)} = \sigma \left(W_h h^{(t-1)} + W_x x^{(t)} + b \right)$

Идея: добавить “память” $c^{(t)}$

Sigmoid function: all gate values are between 0 and 1

Forget gate: контролируем сколько забыть $f^{(t)} = \sigma \left(W_f h^{(t-1)} + U_f x^{(t)} + b_f \right)$

Input gate: контролируем сколько записать $i^{(t)} = \sigma \left(W_i h^{(t-1)} + U_i x^{(t)} + b_i \right)$

Output gate: контролируем сколько считать $o^{(t)} = \sigma \left(W_o h^{(t-1)} + U_o x^{(t)} + b_o \right)$

Новая информация

$$\tilde{c}^{(t)} = \tanh \left(W_c h^{(t-1)} + U_c x^{(t)} + b_c \right)$$

Можно записать новую и стереть старую

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

Считывание

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

All these are vectors of same length n

GRU

Упрощенная версия LSTM

Update gate

$$\mathbf{u}^{(t)} = \sigma \left(\mathbf{W}_u \mathbf{h}^{(t-1)} + \mathbf{U}_u \mathbf{x}^{(t)} + \mathbf{b}_u \right)$$

Reset gate

$$\mathbf{r}^{(t)} = \sigma \left(\mathbf{W}_r \mathbf{h}^{(t-1)} + \mathbf{U}_r \mathbf{x}^{(t)} + \mathbf{b}_r \right)$$

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

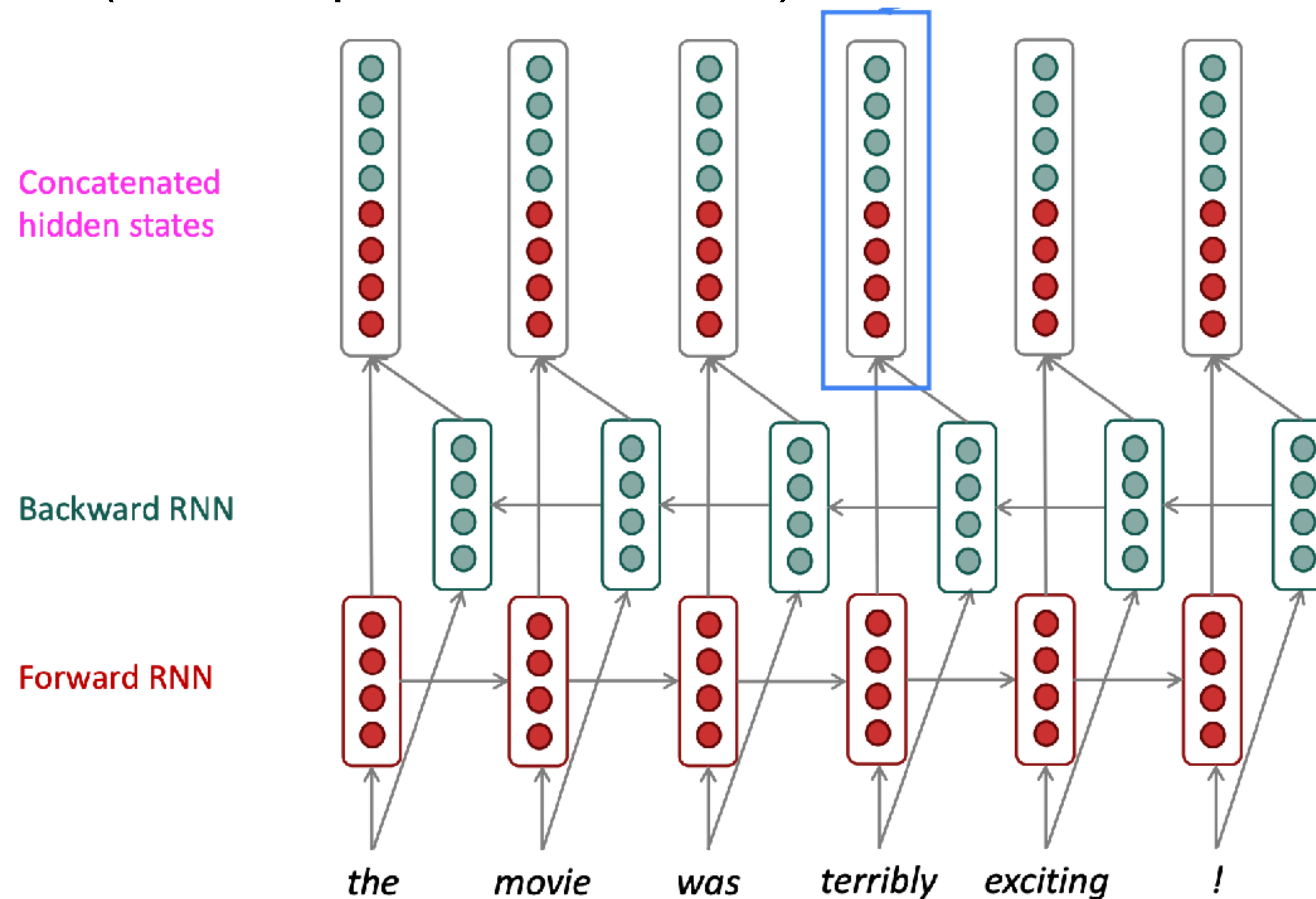
$$\mathbf{h}^{(t)} = (1 - \mathbf{u}^{(t)}) \circ \mathbf{h}^{(t-1)} + \mathbf{u}^{(t)} \circ \tilde{\mathbf{h}}^{(t)}$$

Рекуррентные нейросети (RNN)

GRU и **LSTM** - уменьшают проблему vanishing gradients (необязательно обновлять каждый раз информацию) - использовать их!

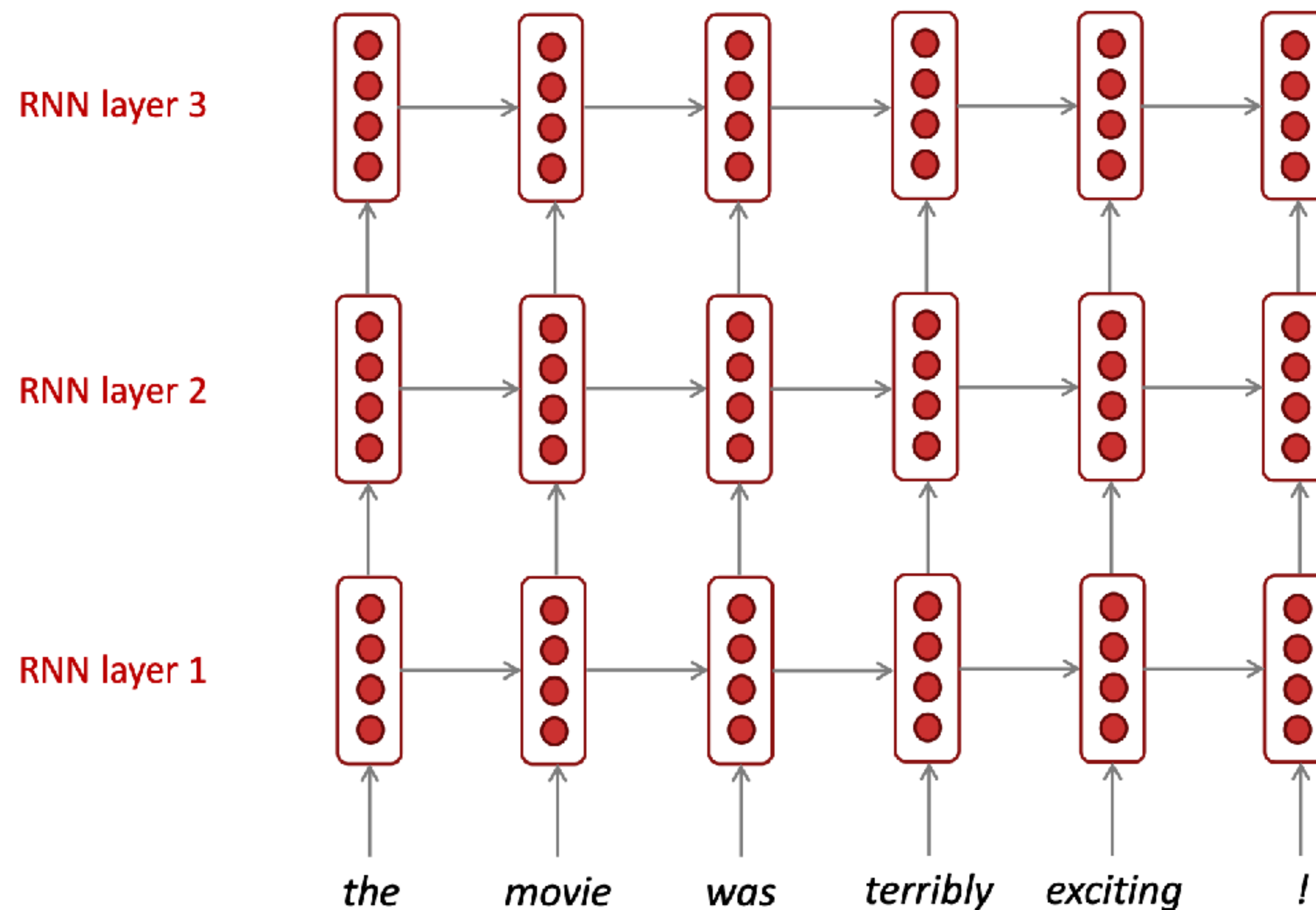
Рекуррентные нейросети (RNN)

Улучшения: bidirectional RNN (если порядок не важен)



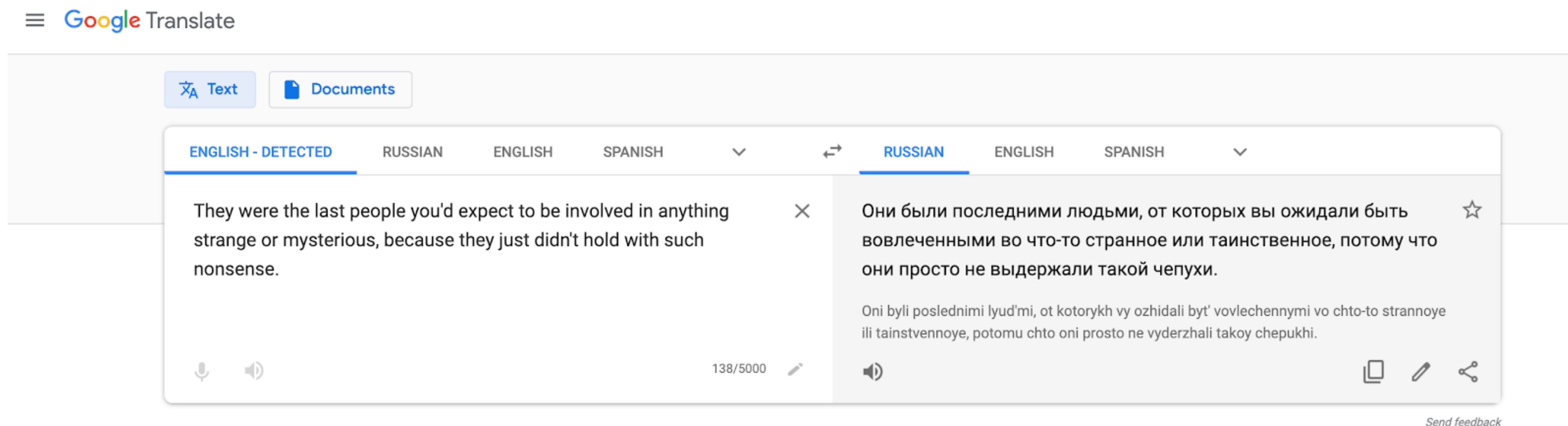
Рекуррентные нейросети (RNN)

Улучшения: stacked RNN

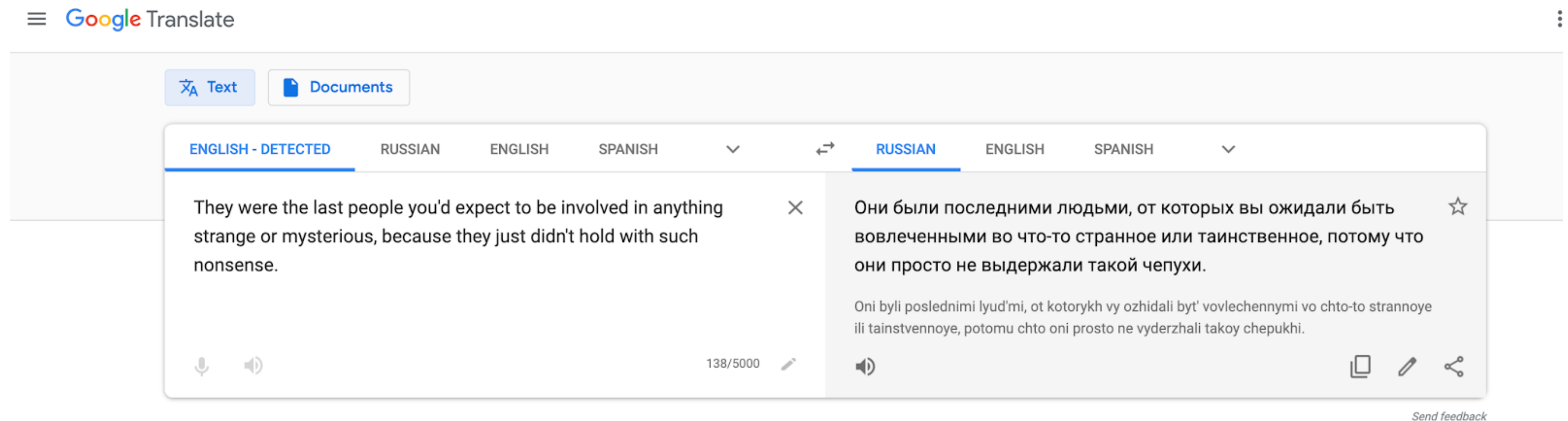


Sequence-to-sequence model

Приложения: перевод



Приложения: перевод



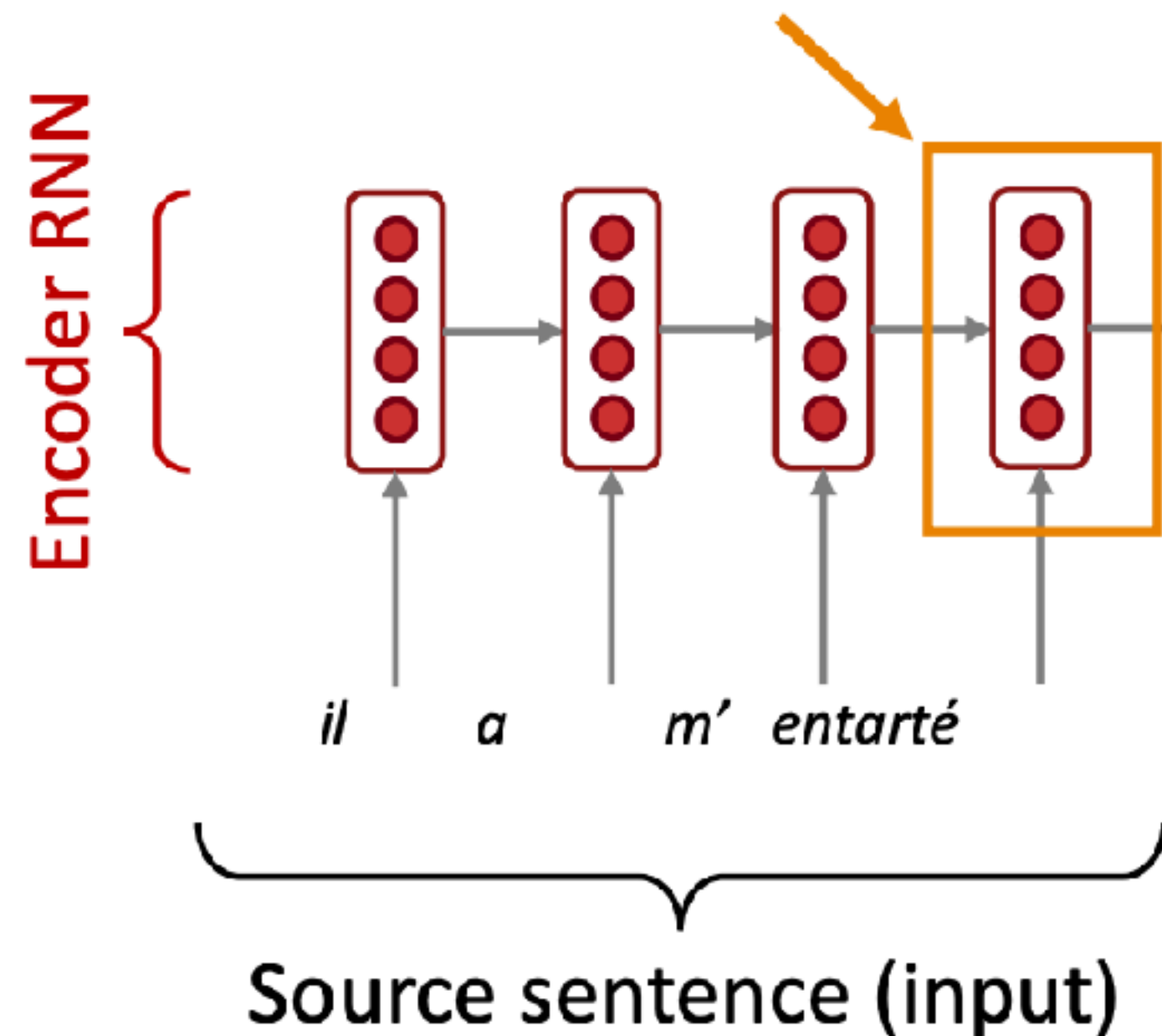
Аutoreгрессионная модель $p(y | \underline{x}) = \prod_{i=1}^n p(y_i | y_1, \dots, y_{i-1}, \underline{x})$

x - текст на исходном языке (source)

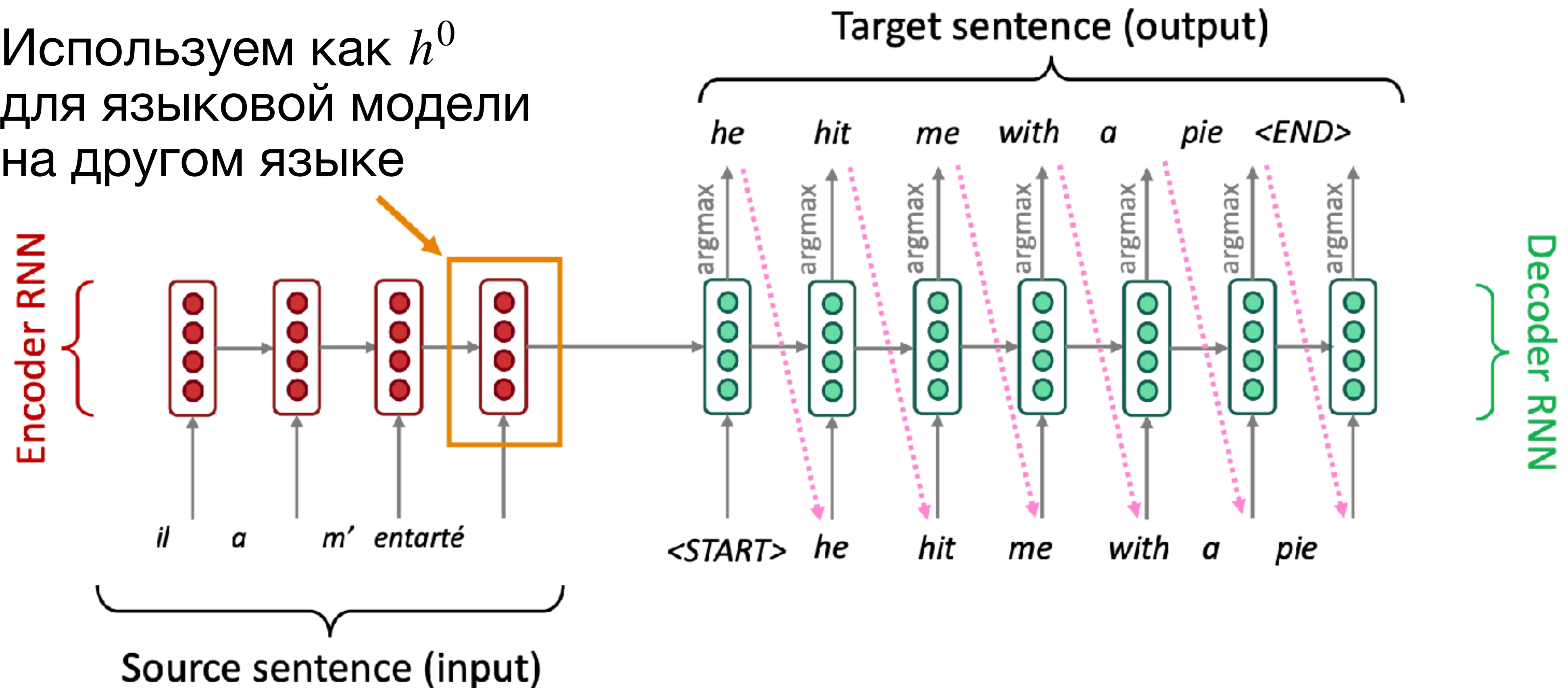
y - перевод текста на другой язык (target)

Sequence-to-sequence model

Последний hidden state -
представление всего
исходного текста

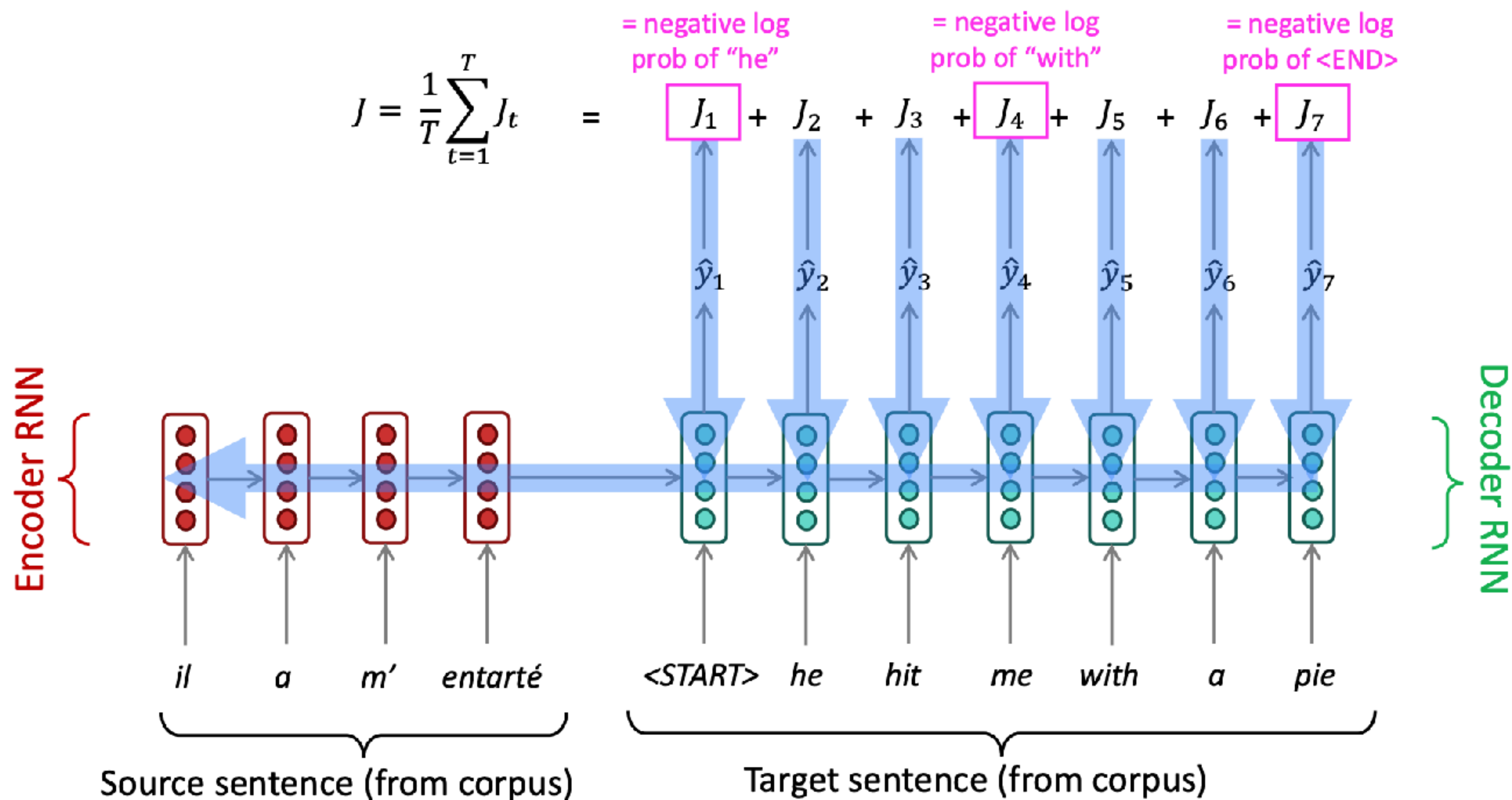


Используем как h^0
для языковой модели
на другом языке



Sequence-to-sequence model

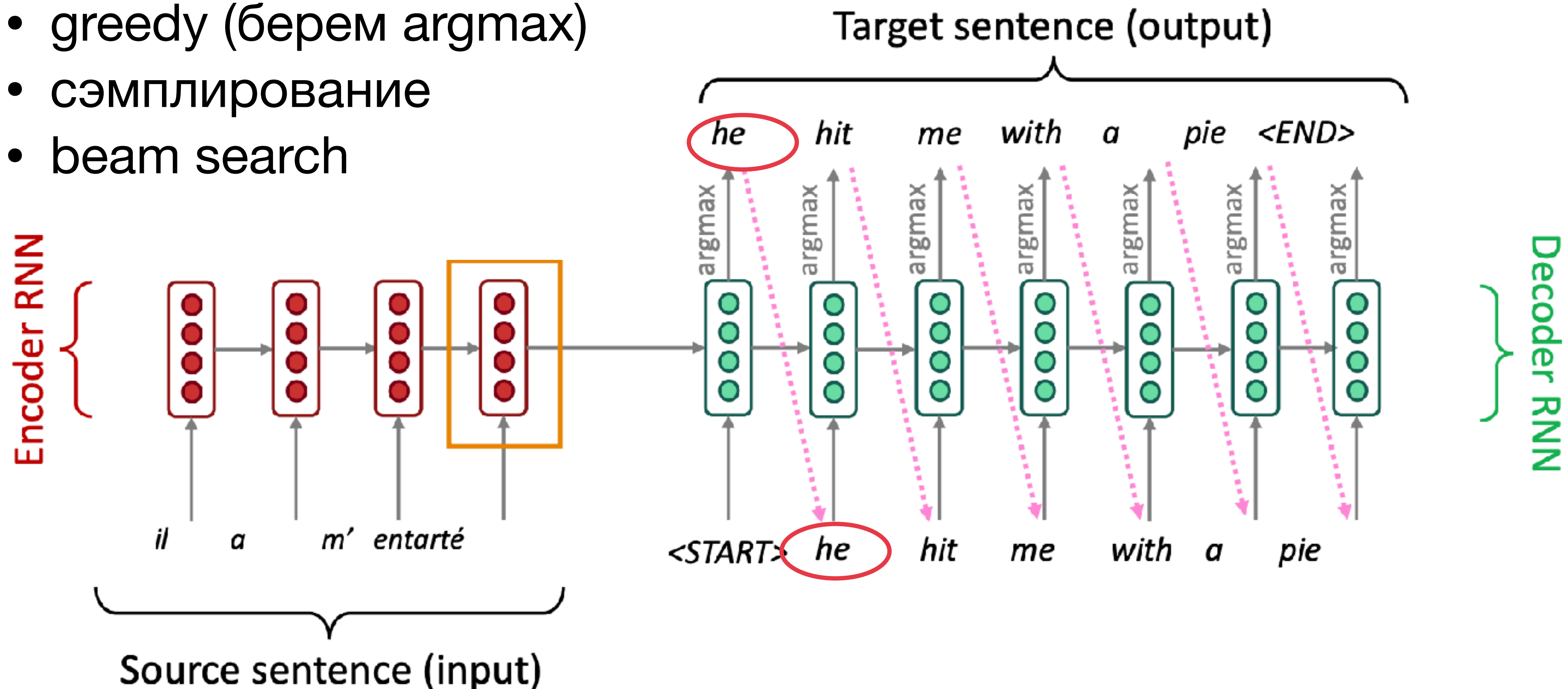
Обучение: нужен параллельный корпус



Sequence-to-sequence model

Генерация:

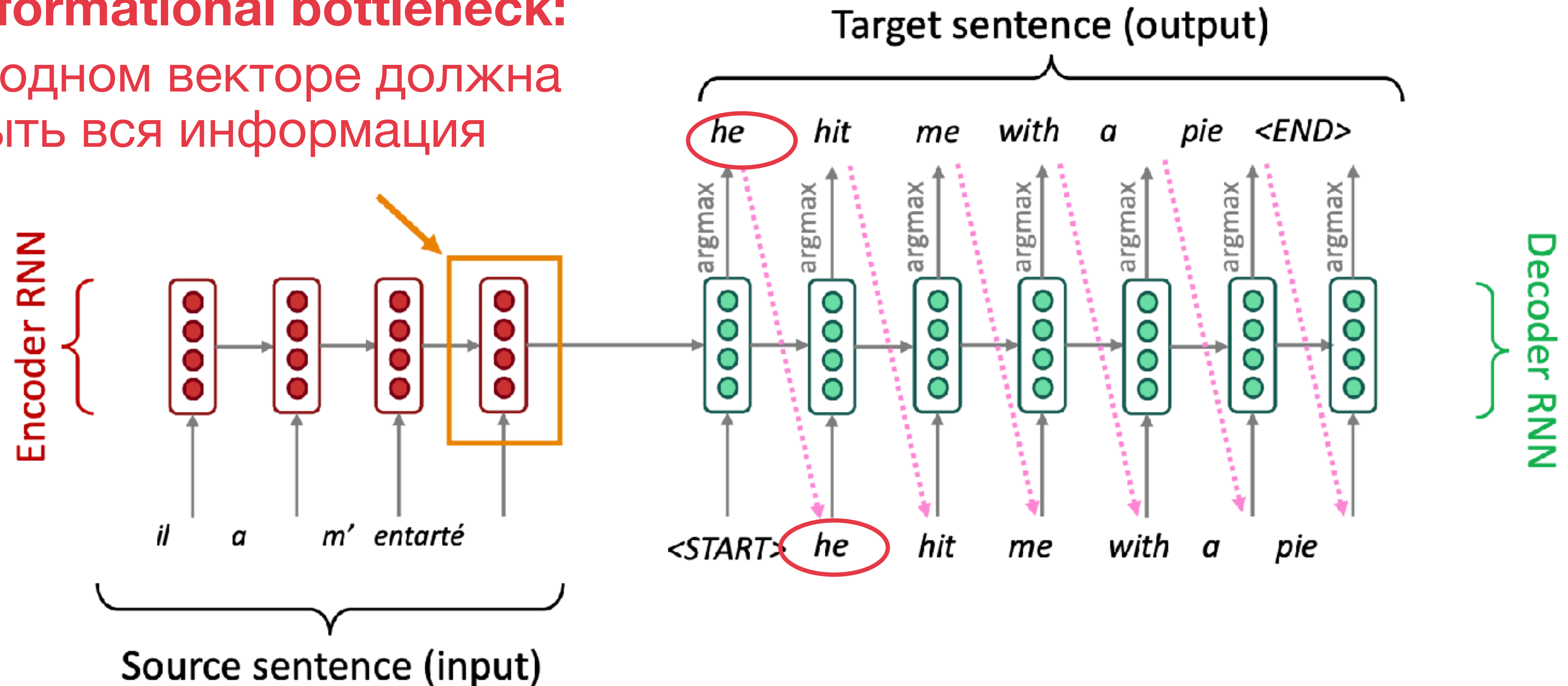
- greedy (берем argmax)
- сэмплирование
- beam search



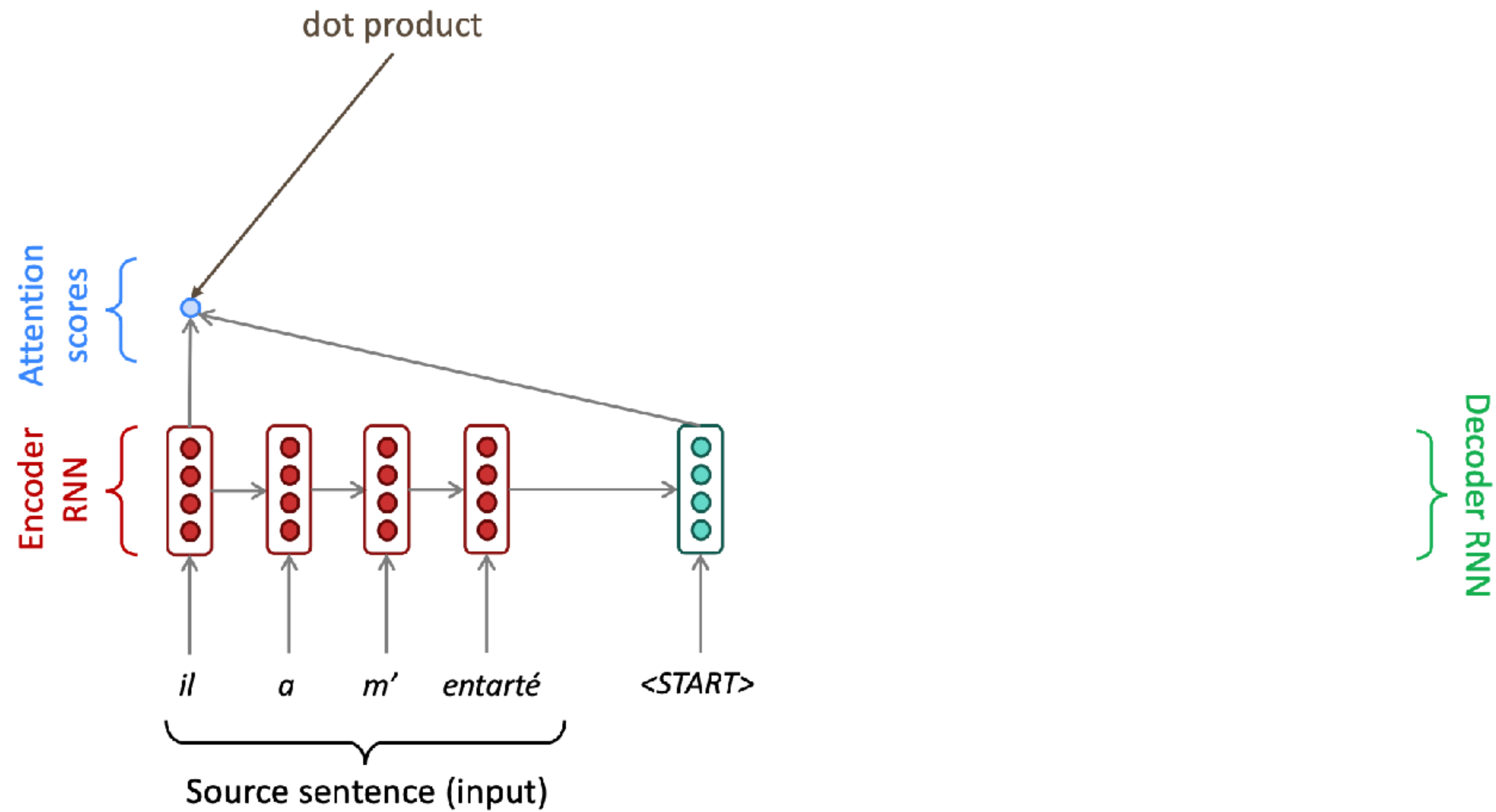
Sequence-to-sequence model

Informational bottleneck:

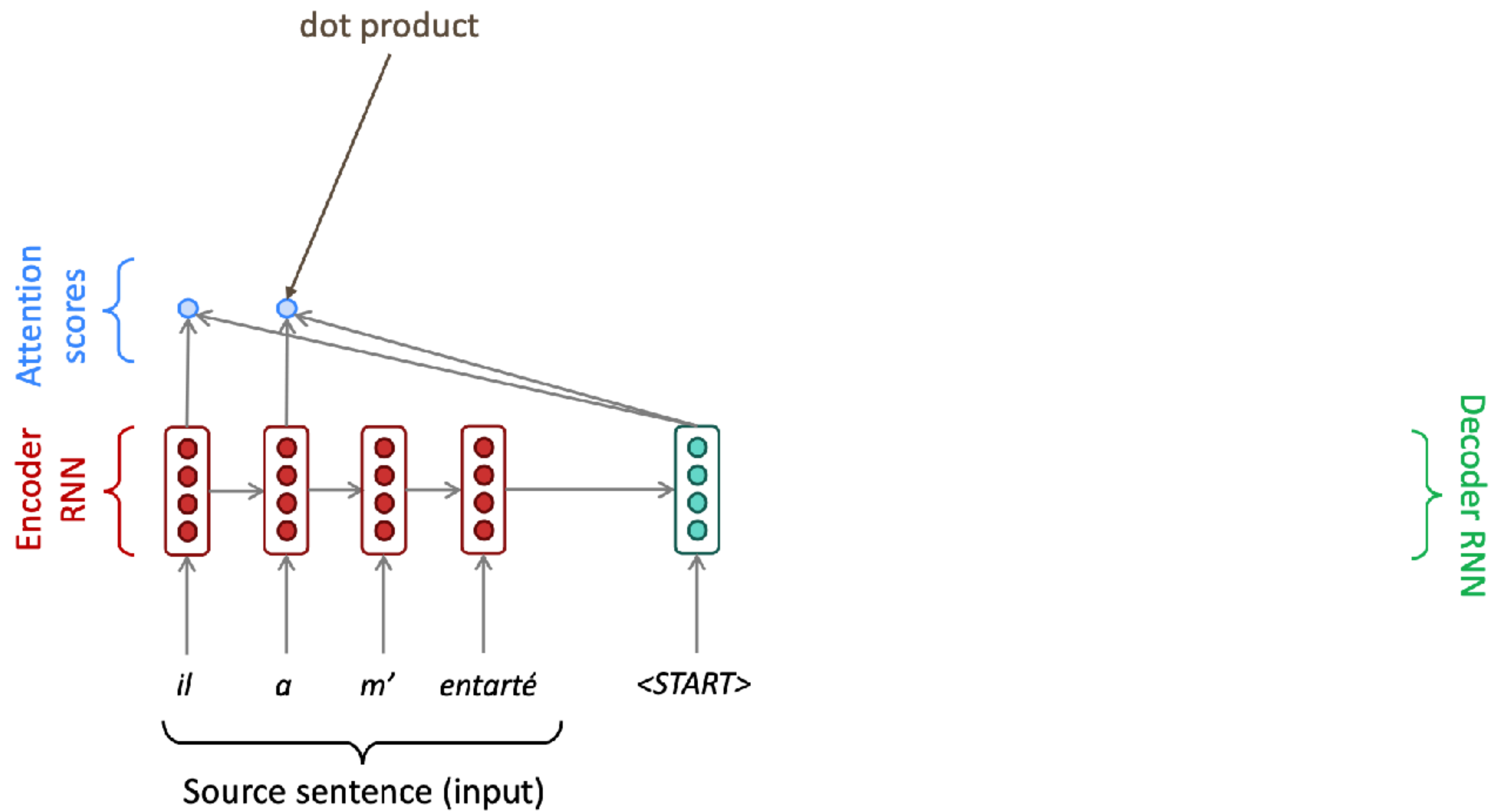
В одном векторе должна быть вся информация



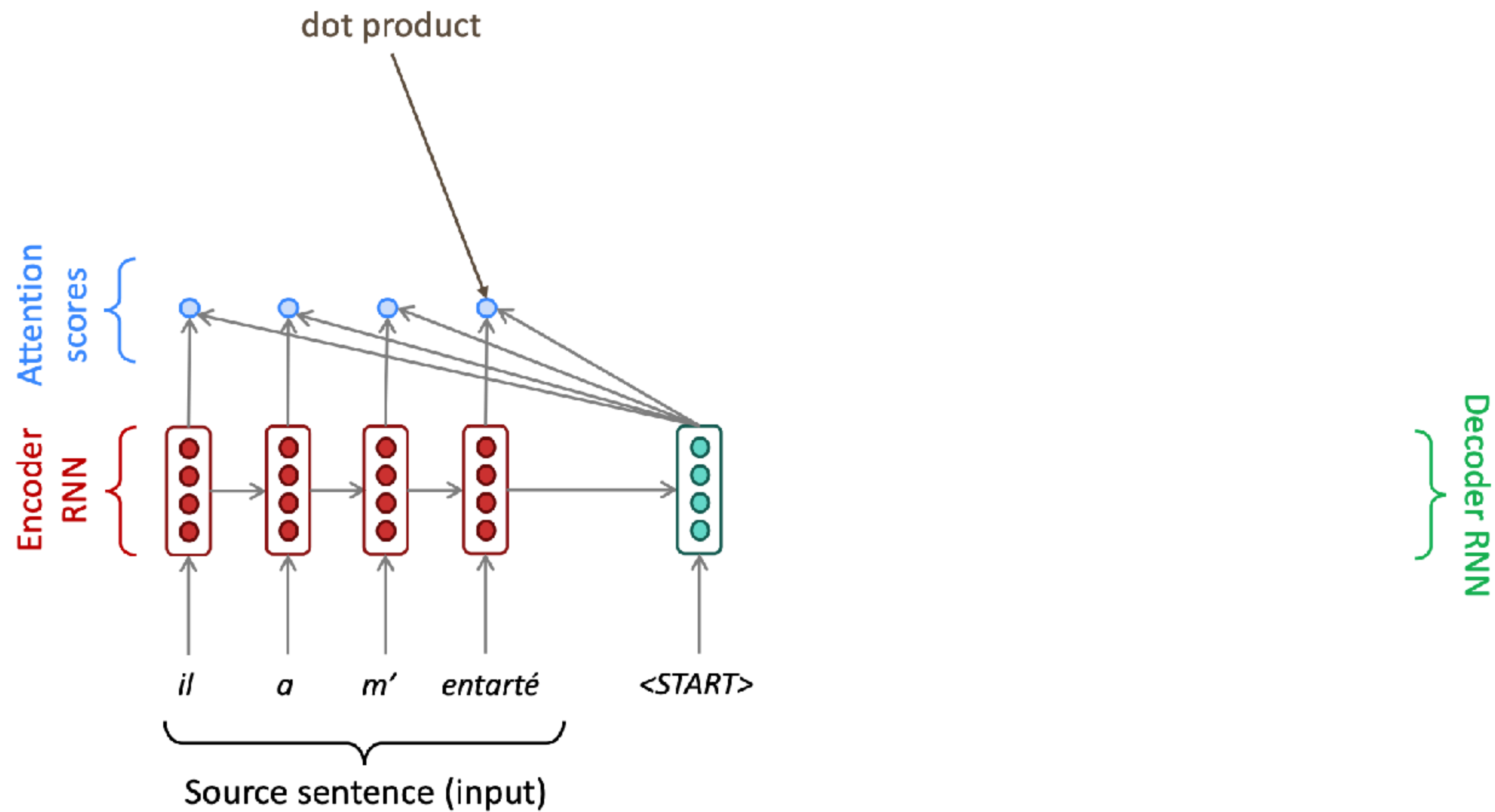
Attention



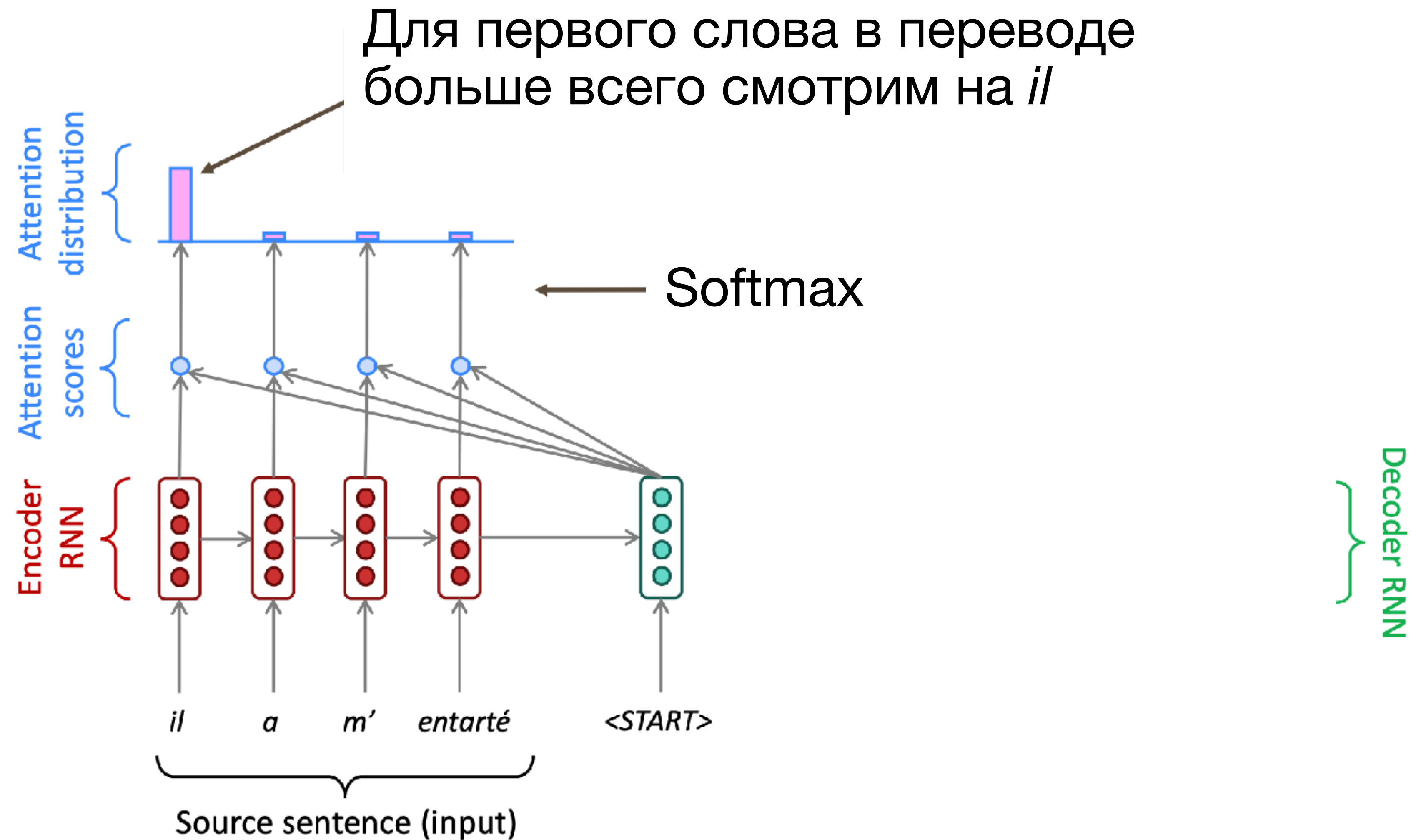
Attention



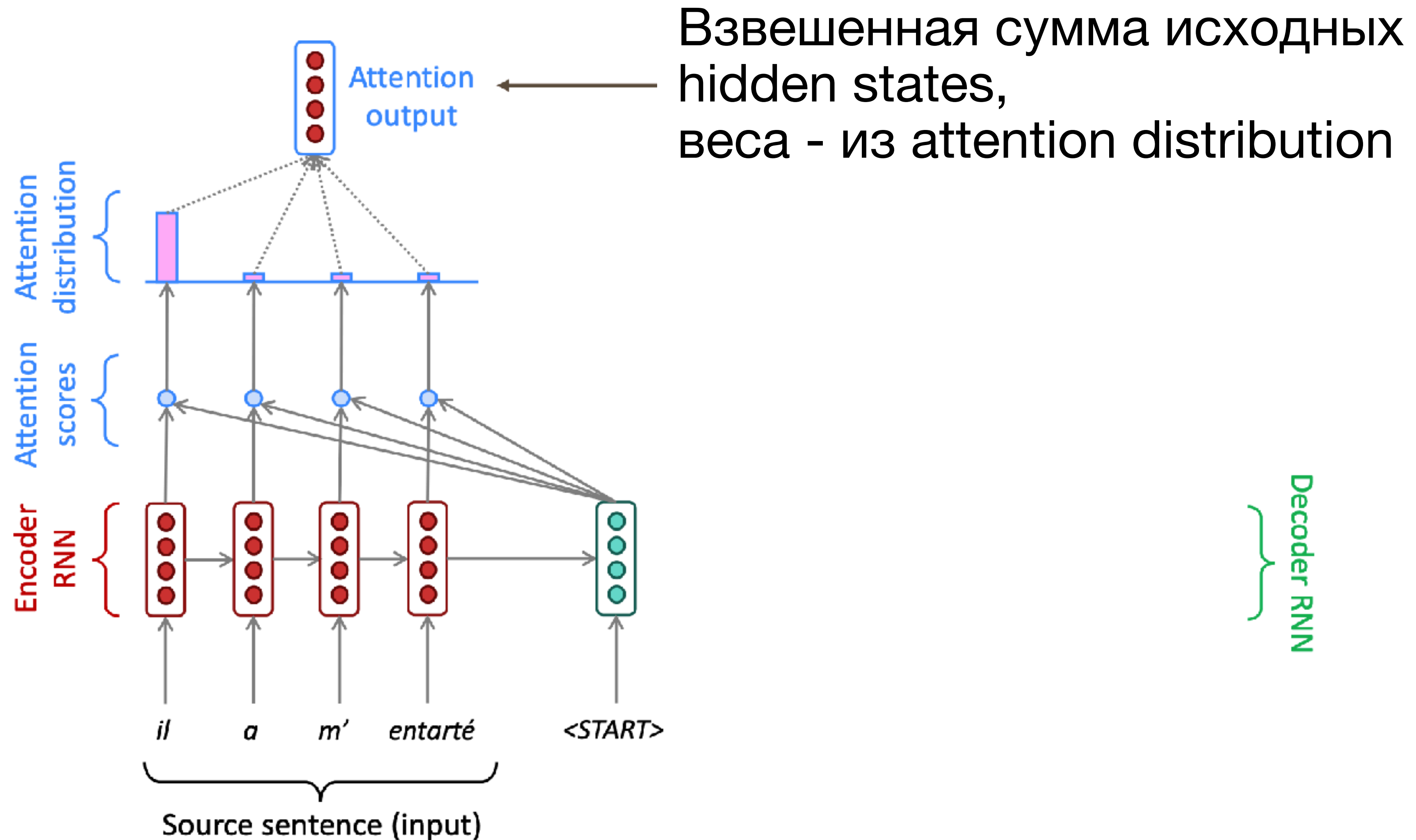
Attention



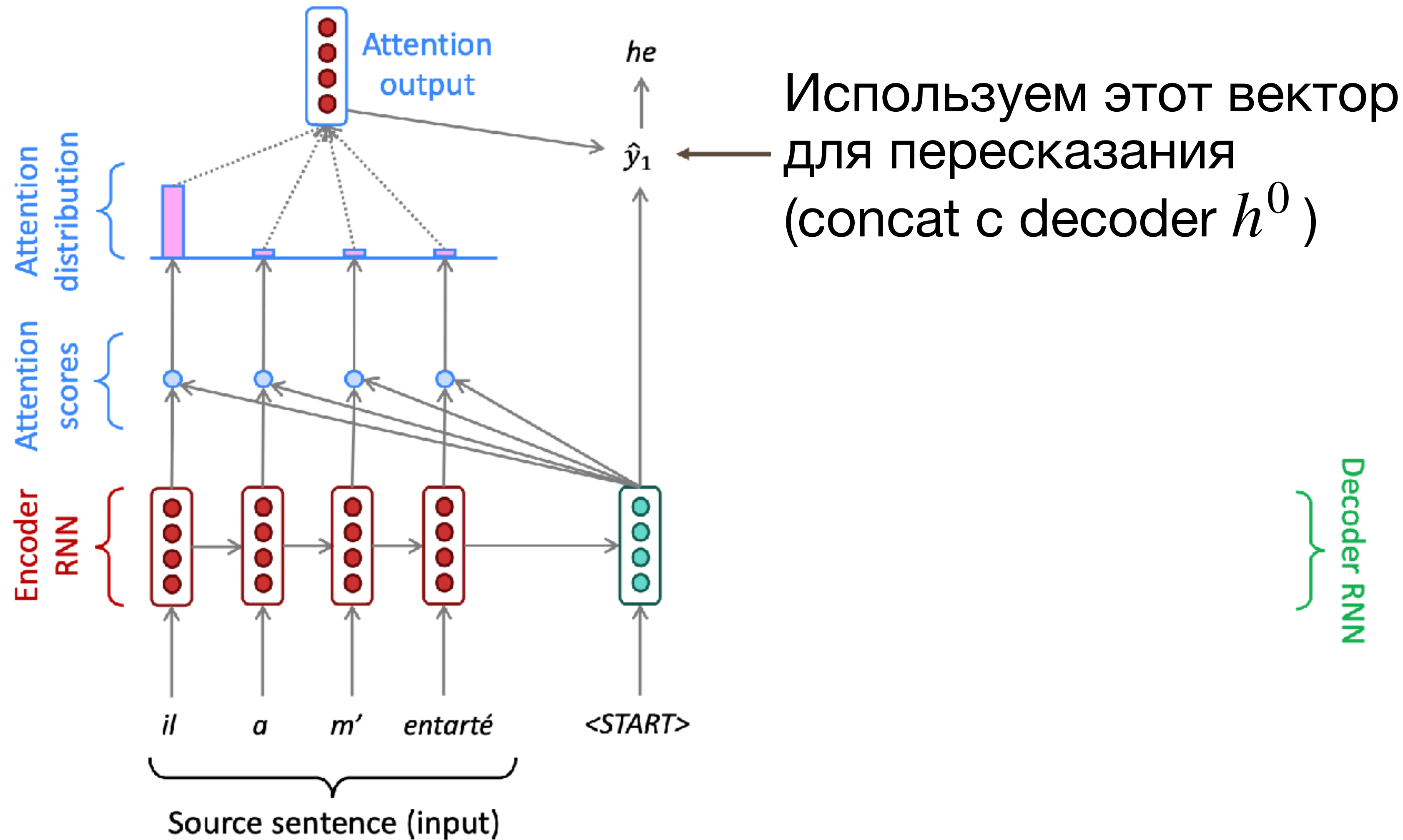
Attention



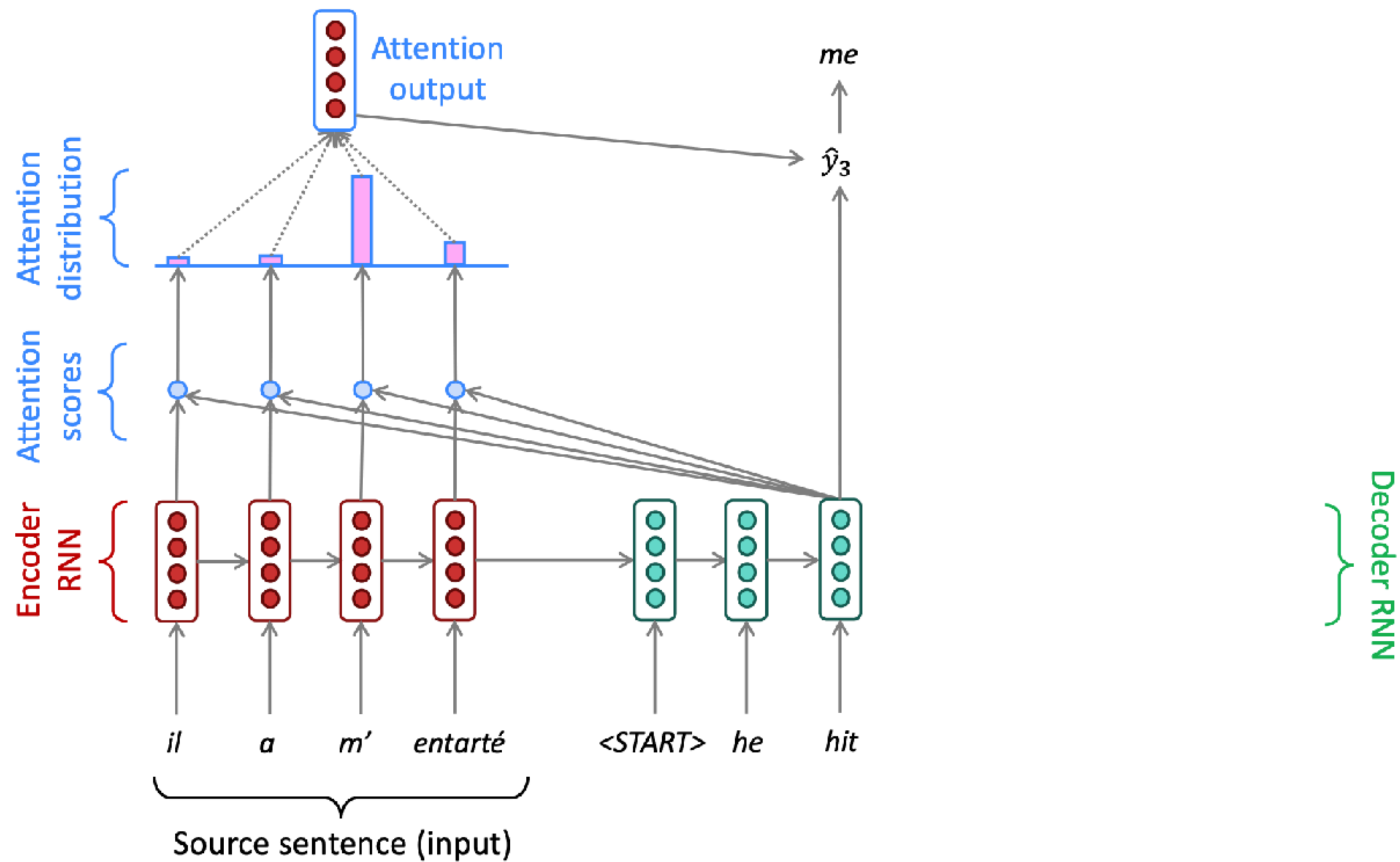
Attention



Attention



Attention



Attention

- Убирает bottleneck, всегда лучше качество
- Дает интерпретируемость (можно визуализировать распределения)

