Лекция 4. Seq2seq (Sequence-to-Sequence) и Attention

Seq2seq (Sequence-to-Sequence) и Attention

- Seq2seq
- Attention
- Transfomer
- Сегментация по подсловам: BPE
- Анализ и интерпретируемость

Задача перевода текста

- Человеческий перевод:

вероятность $y^* = \arg\max_y p(y|x)$ интуитивна и определяется опытом человека.

- Машинный перевод:

вероятность $y' = \arg\max_y p(y|x,\theta)$, где θ −параметр модели.

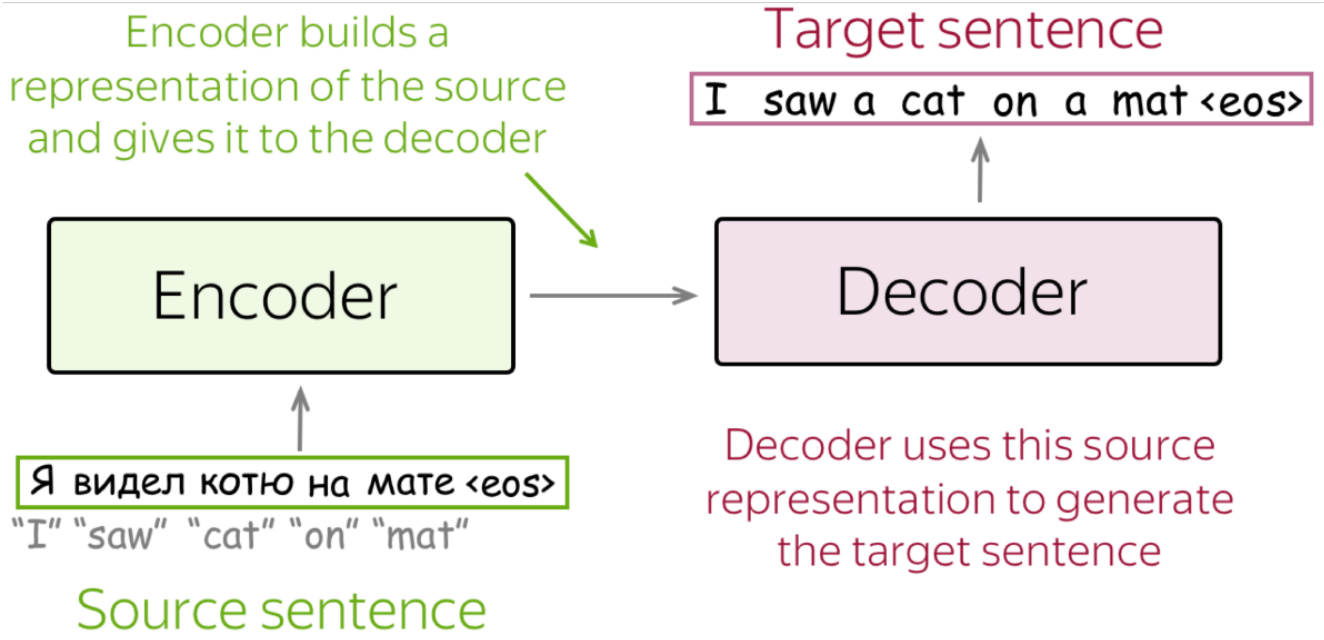
Возникают вопросы **моделирования** (как выглядит модель), **обучения** (как найти θ) и **поиска** (как найти argmax).

- modeling

How does the model for $p(y|x,\theta)$ look like?

Архитектура «кодировщик-декодировщик» (encoder-decoder)

- Кодировщик – считывает исходное предложение и создаёт его представление
- Декодировщик – использует исходное представление от кодировщика для генерации целевой последовательности.



Модели условного языка

Языковая модель: $P(y_1, y_2, \ldots, y_n) = \prod_{t=1}^{n} p(y_t | y_{<t})$

Условная языковая модель: $P(y_1, y_2, \ldots, y_n, |x) = \prod_{t=1}^{n} p(y_t | y_{<t}, x)$

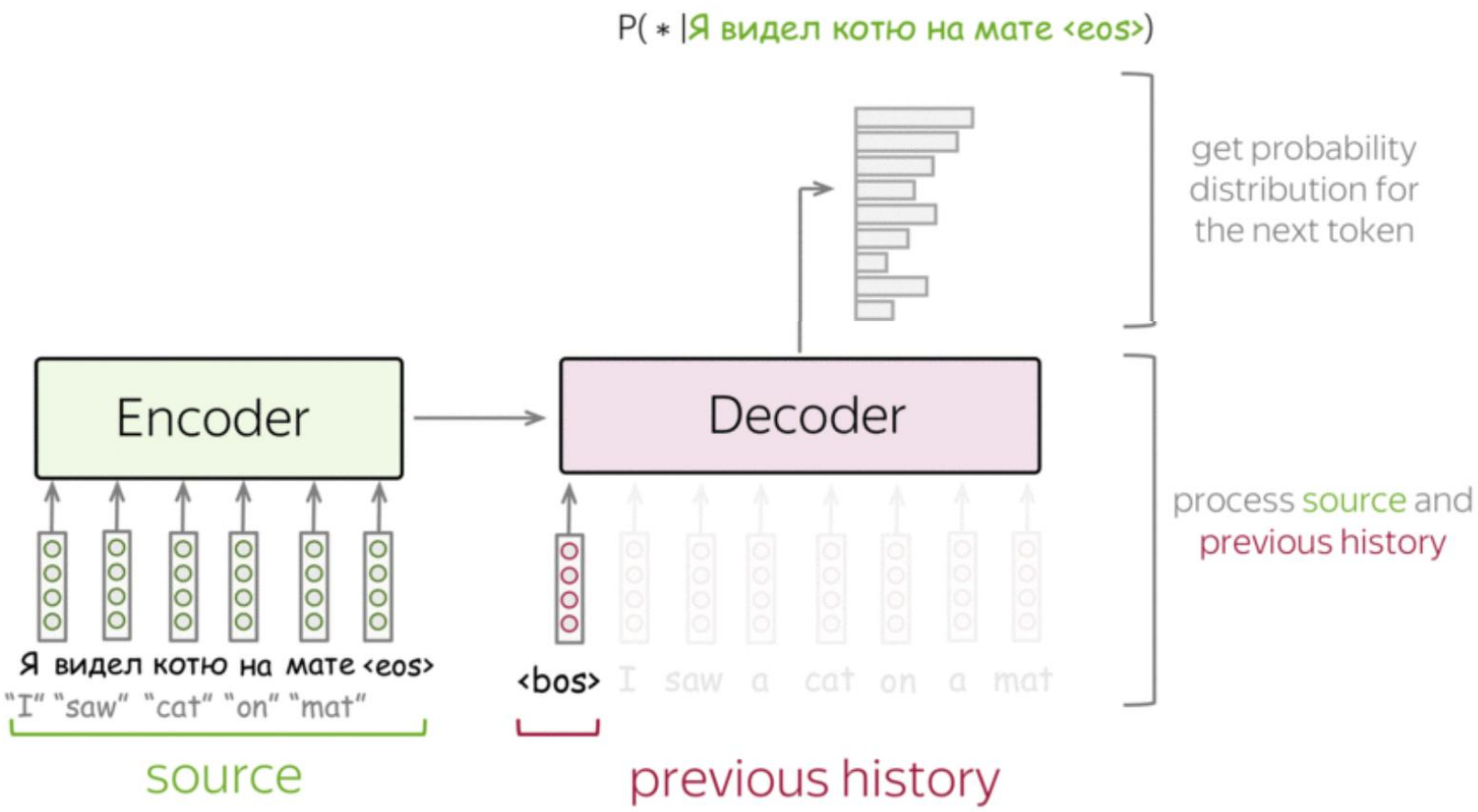
condition on source x

Общий вид

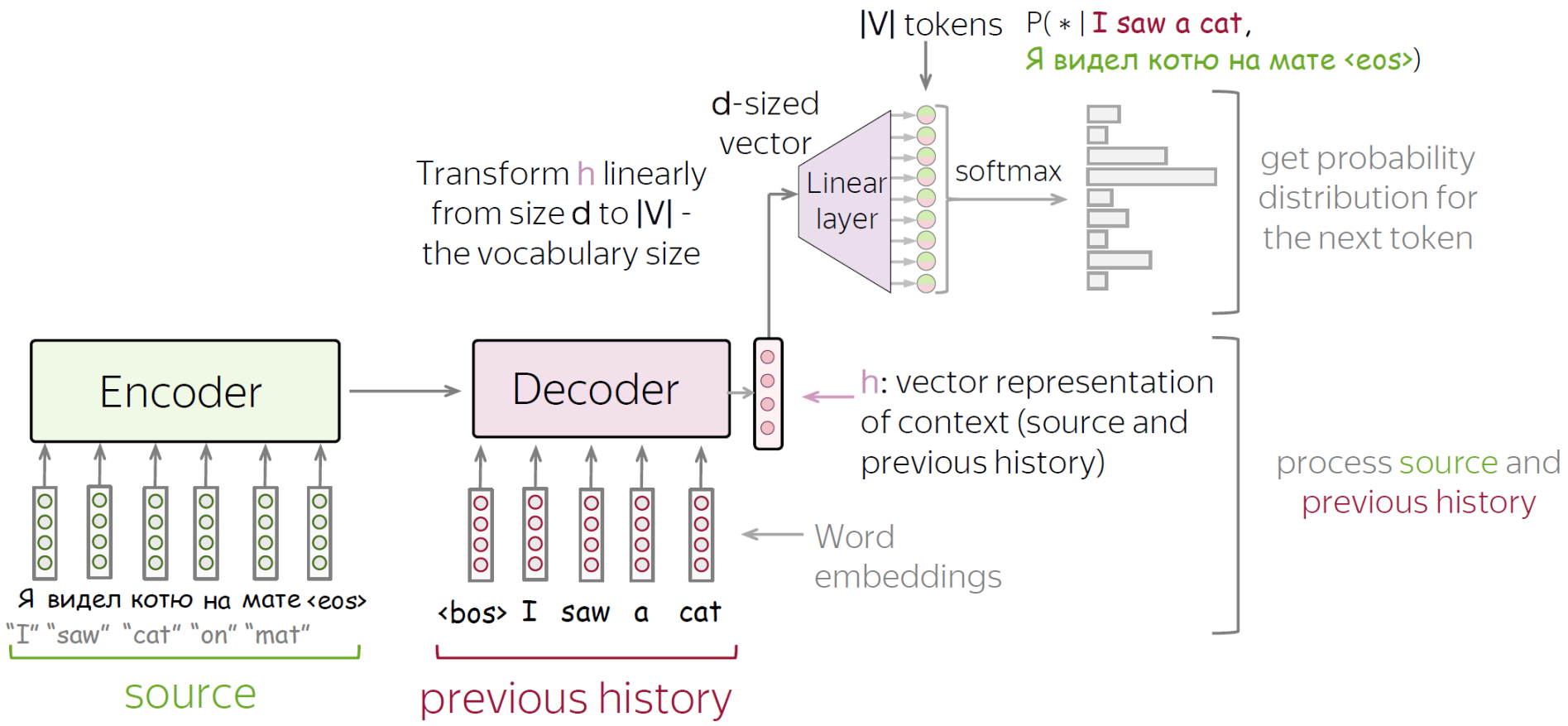
Общий процесс имеет вид:

- источник данных и ранее сгенерированные слова передаются в сеть;
- получаем векторное представление контекста (как исходного, так и предыдущего);
- на основе этого векторного представления прогнозируем распределение вероятностей для следующего токена.

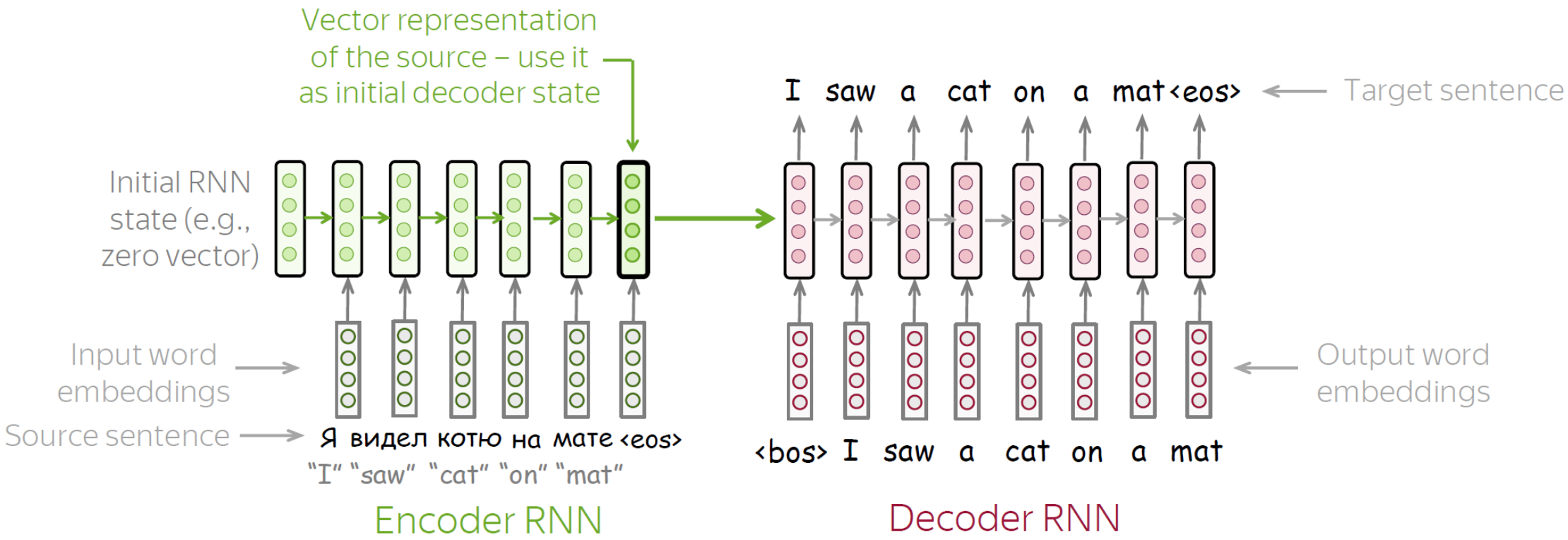
Общий вид


P(* |Я видел котю на мате <eos>)
get probability distribution for the next token
Encoder
Decoder
process source and previous history
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos>
I saw a cat on a mat
source
previous history

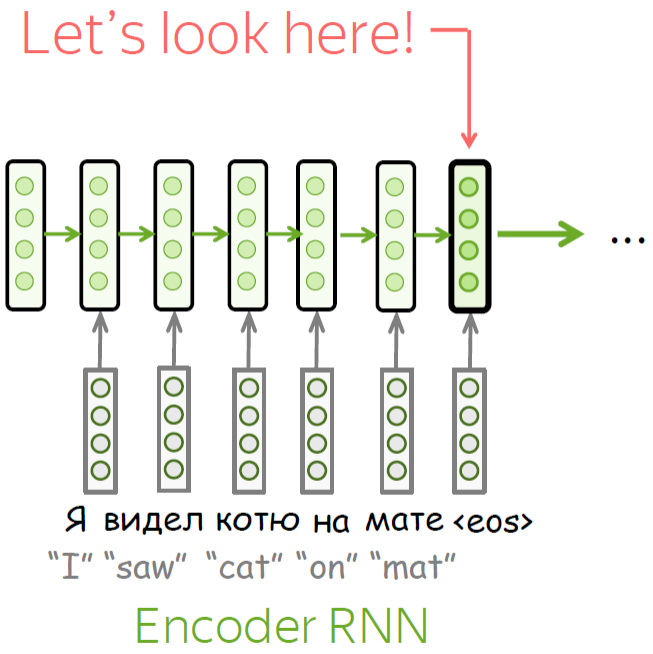
Высокоуровневая схема

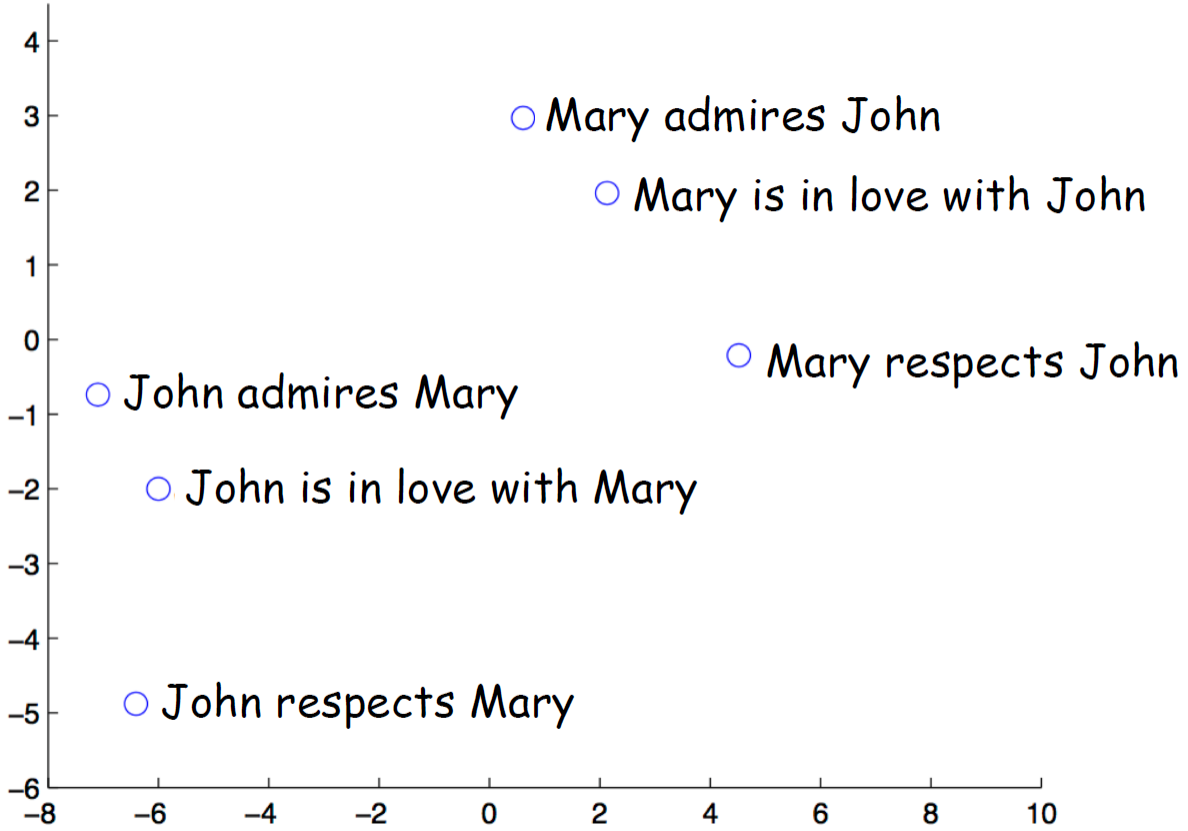


Простейшая модель: RNN-кодер и декодер

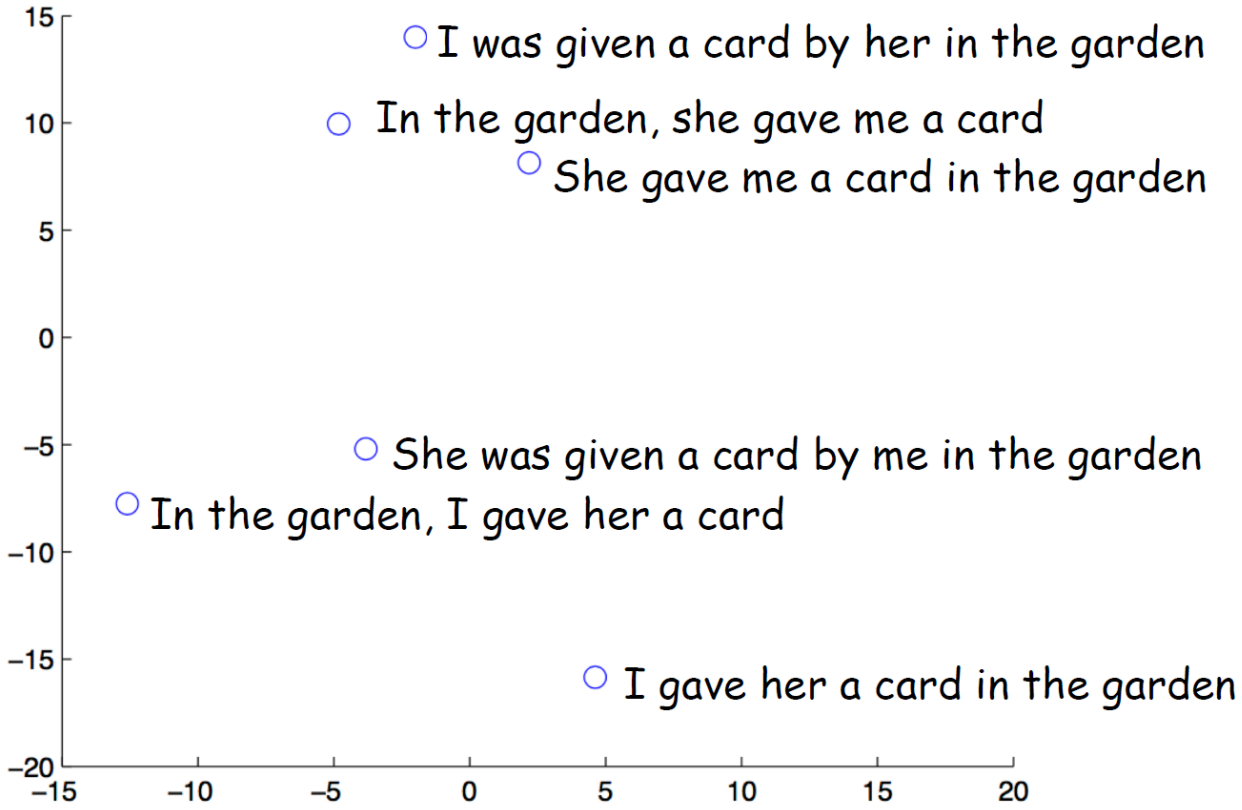


Конечное состояние кодировщика





Конечное состояние кодировщика



- learning

How to find θ?

Кросс-энтропия

Source sequence:

Я видел котю на мате <eos>

"I" "saw" "cat" "on" "mat"

Target sequence: ← one training example

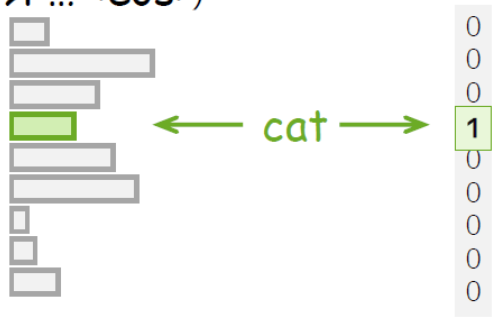
I saw a cat on a mat <eos> ← one step for this example

previous tokens

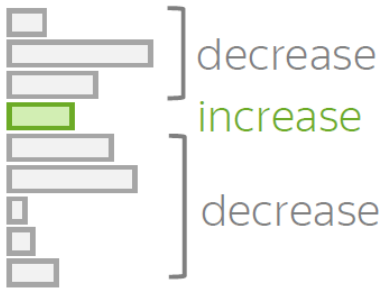
we want the model to predict this

Model prediction: p(* |I saw a, Я ... <eos>)

Target

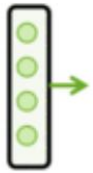


Loss = -log (p(cat)) → min



Кросс-энтропия

Encoder: read source

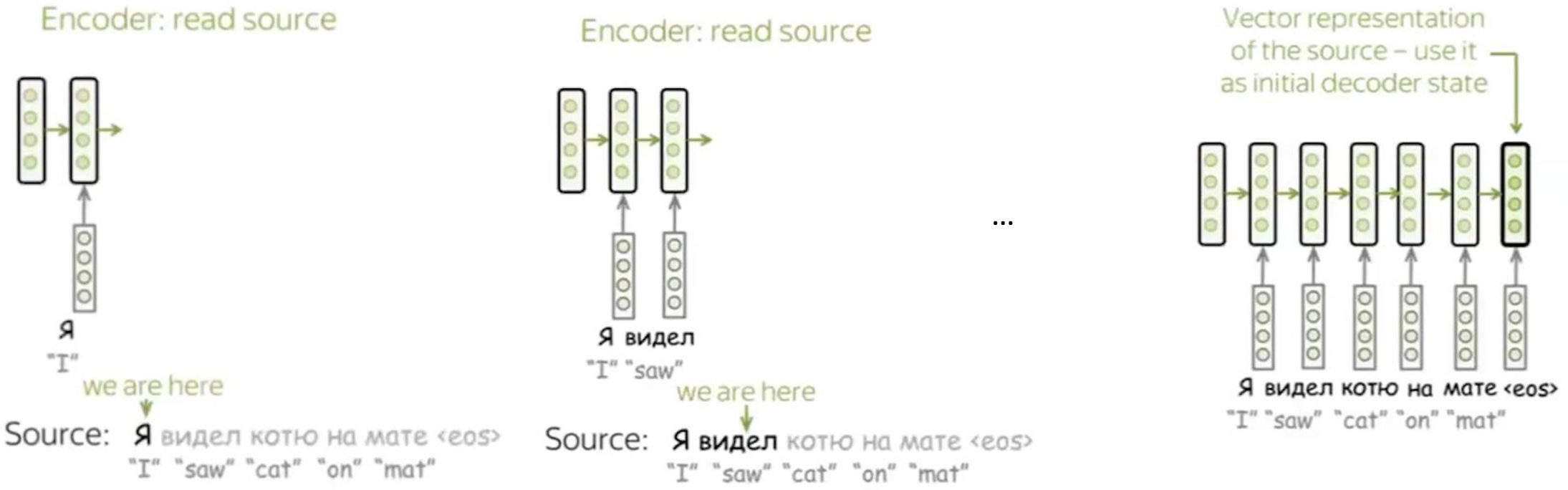


we are here

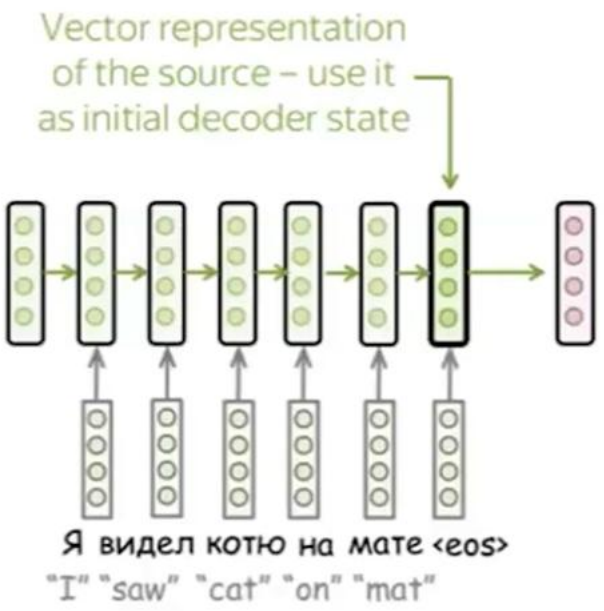
Source: Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"

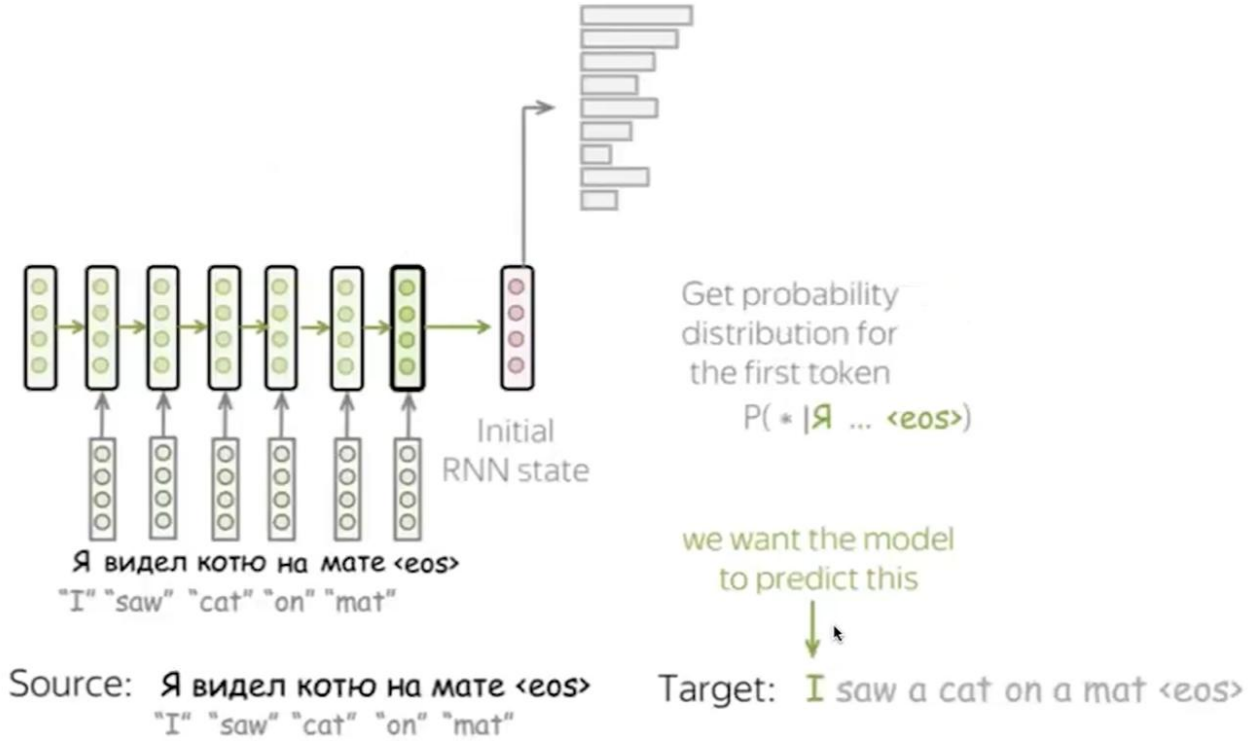
Target: I saw a cat on a mat <eos>

Кросс-энтропия

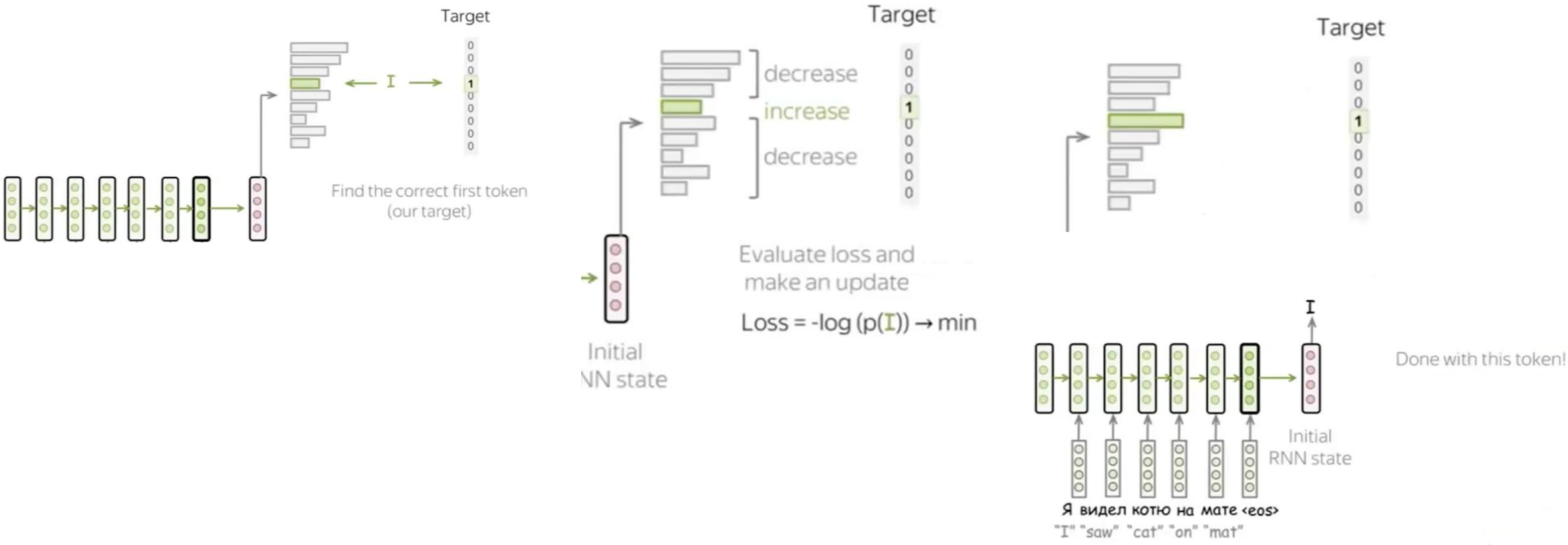


Кросс-энтропия

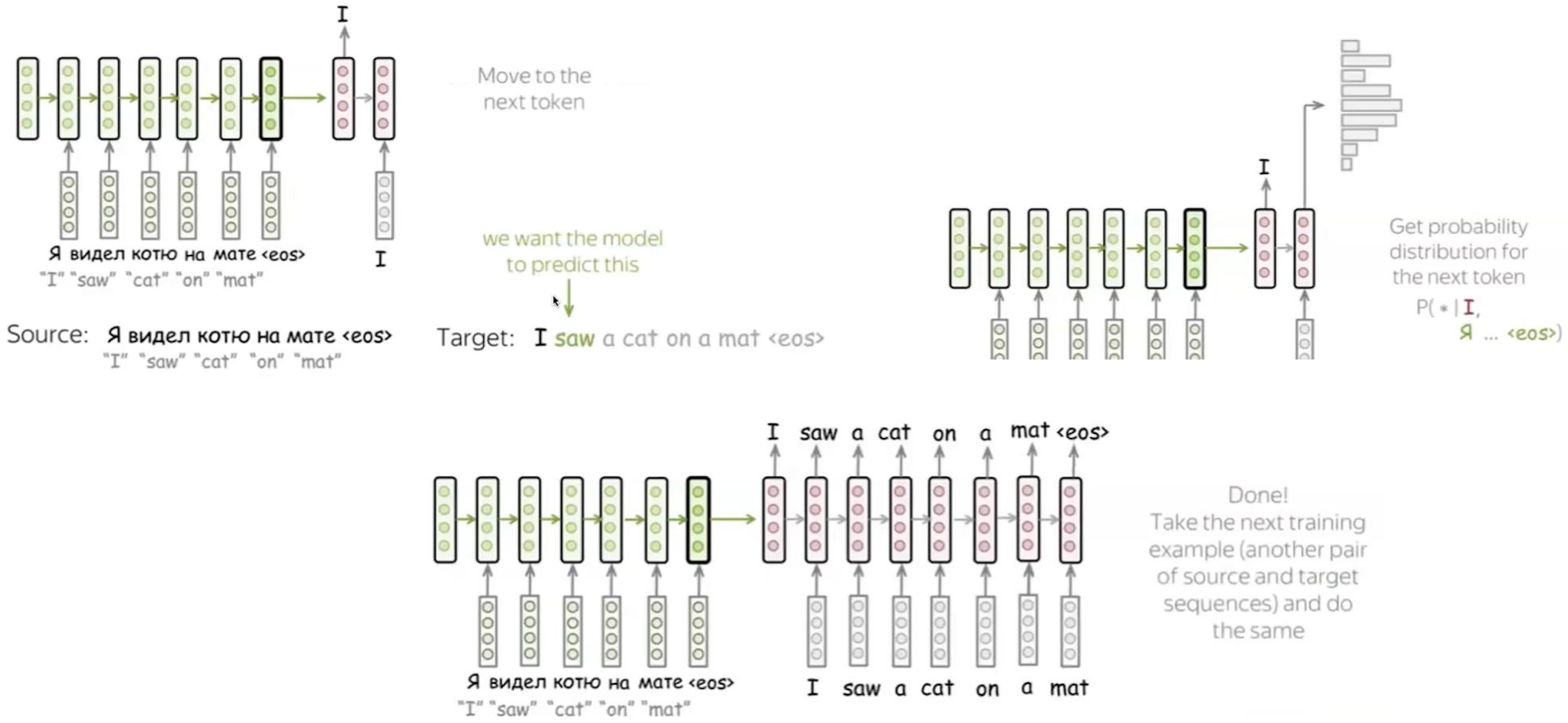




Кросс-энтропия



Кросс-энтропия



- search

How to find the argmax?

Как создать перевод?

$$y' = \arg\max_y p(y|x) = \arg\max_y \prod_{t=1}^{n} p(y_t|y_{<t}, x)$$

Простой вариант:

- жадное декодирование — на каждом шаге выбираем токен с наибольшей вероятностью

но $$\arg\max_y \prod_{t=1}^{n} p(y_t|y_{<t}, x) \neq \prod_{t=1}^{n} \arg\max_{y_t} p(y_t|y_{<t}, x)$$

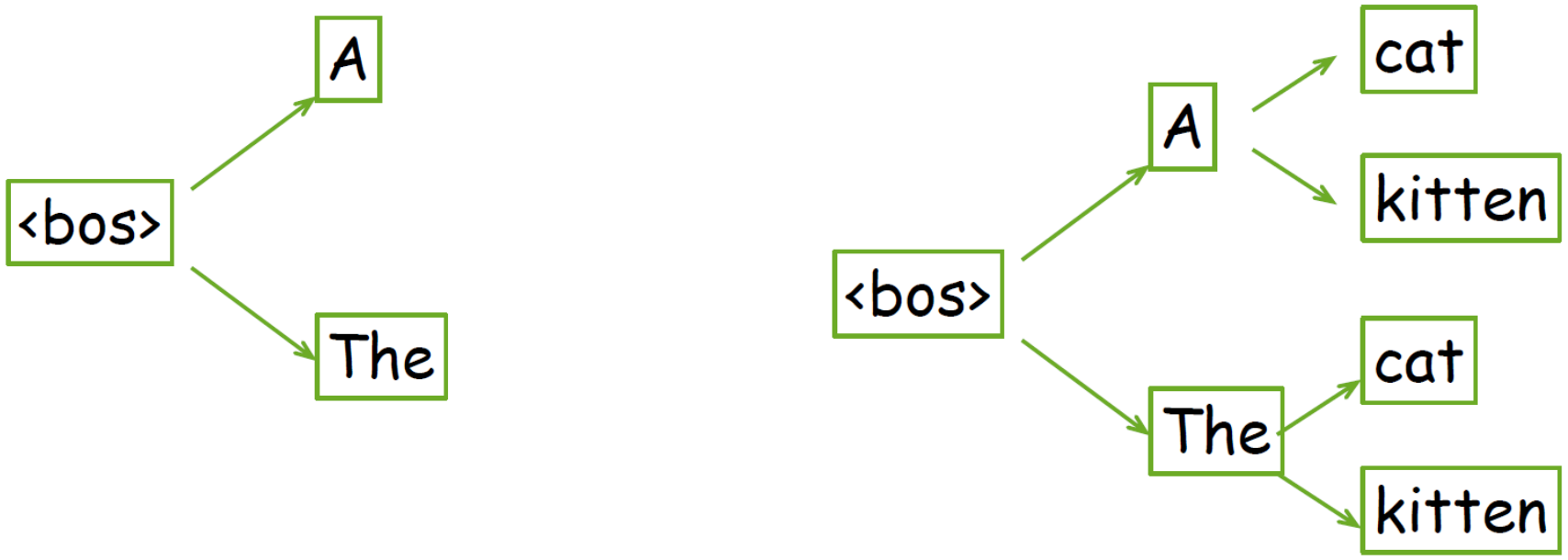
Beam Search (лучевой поиск)

На каждом этапе сохраняем несколько лучших гипотез.

Начинаем с маркера начала предложения или с пустой последовательности.

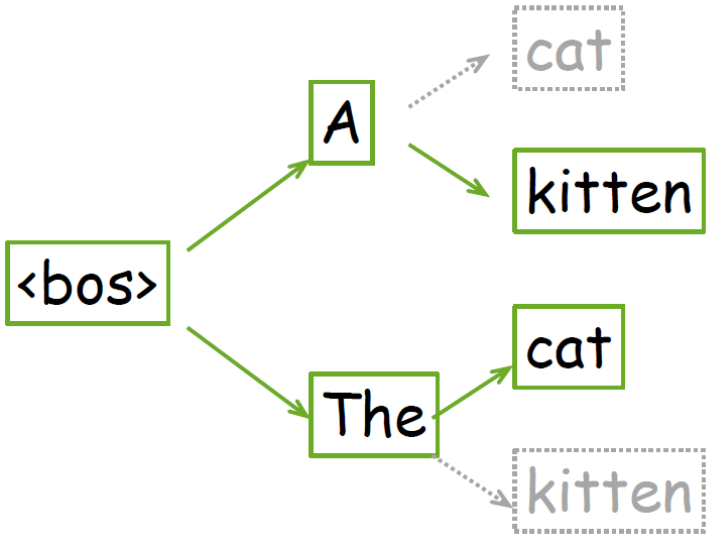
Beam Search (лучевой поиск)

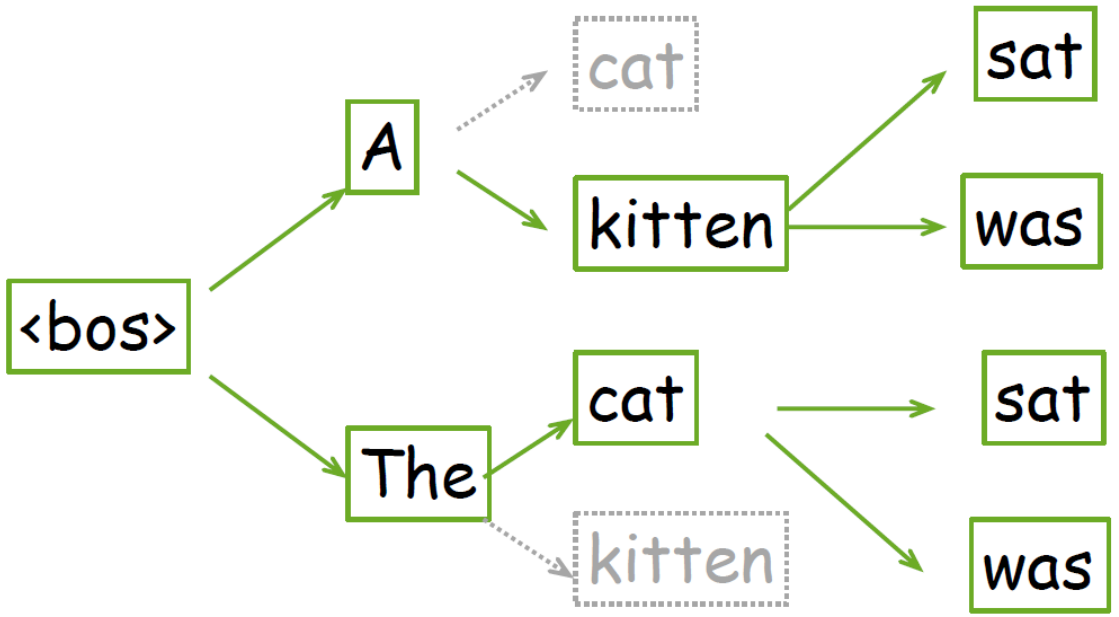
Сгенерировать: beam_size наиболее вероятных токенов



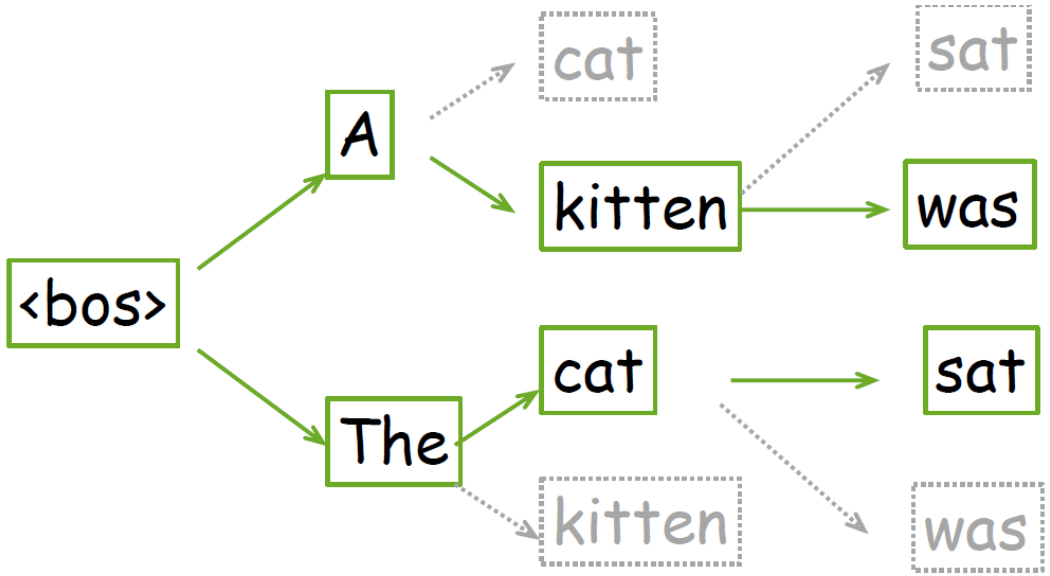
Beam Search (лучевой поиск)

Посмотрим на вероятности: P(The cat) > P(A kitten) > P(A cat) > P(The kitten)

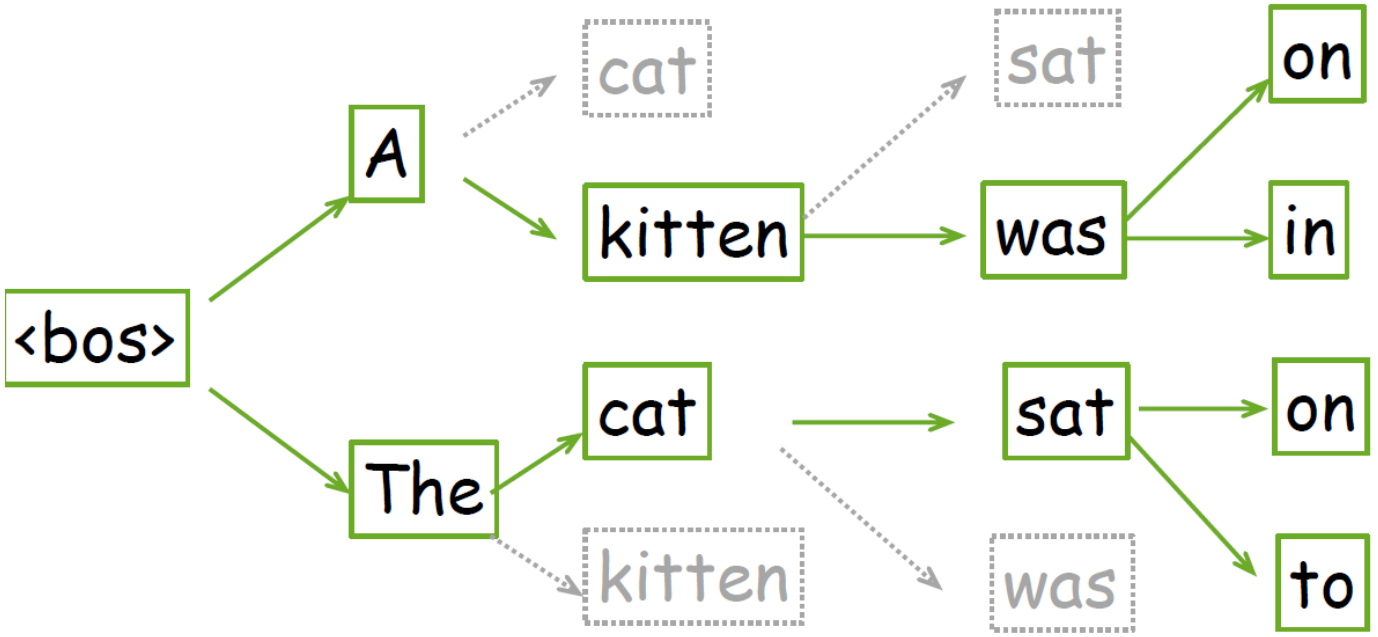




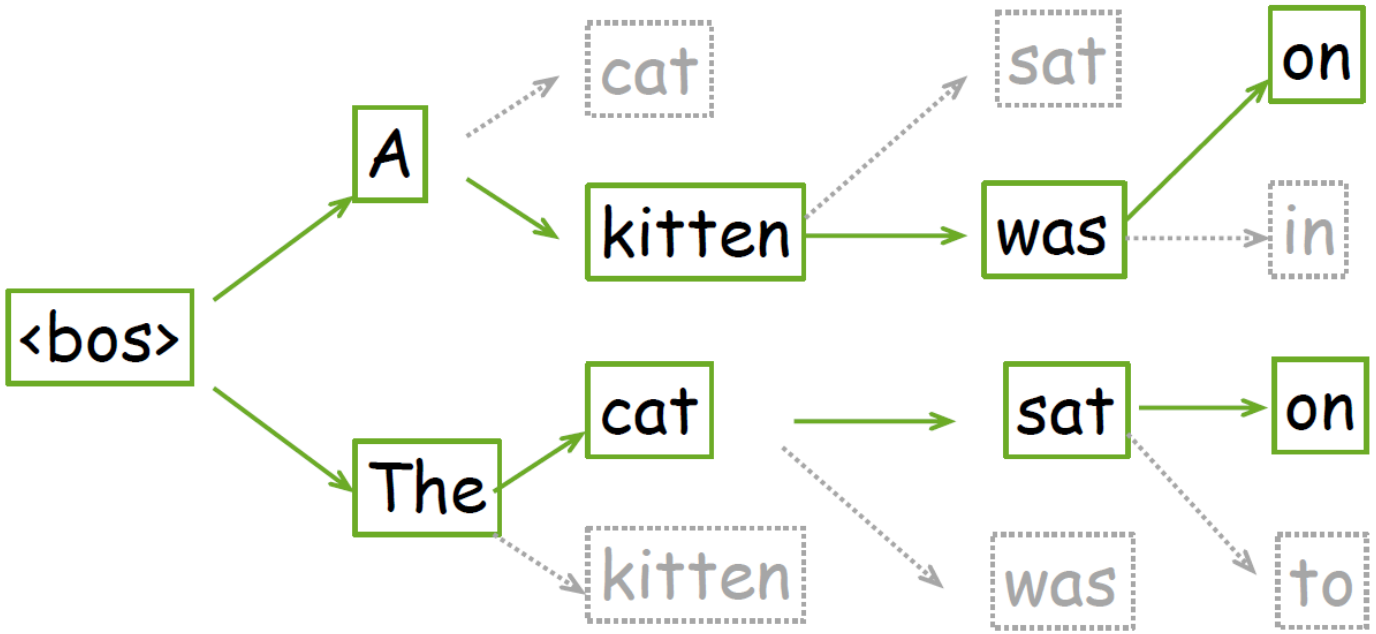
Beam Search (лучевой поиск)



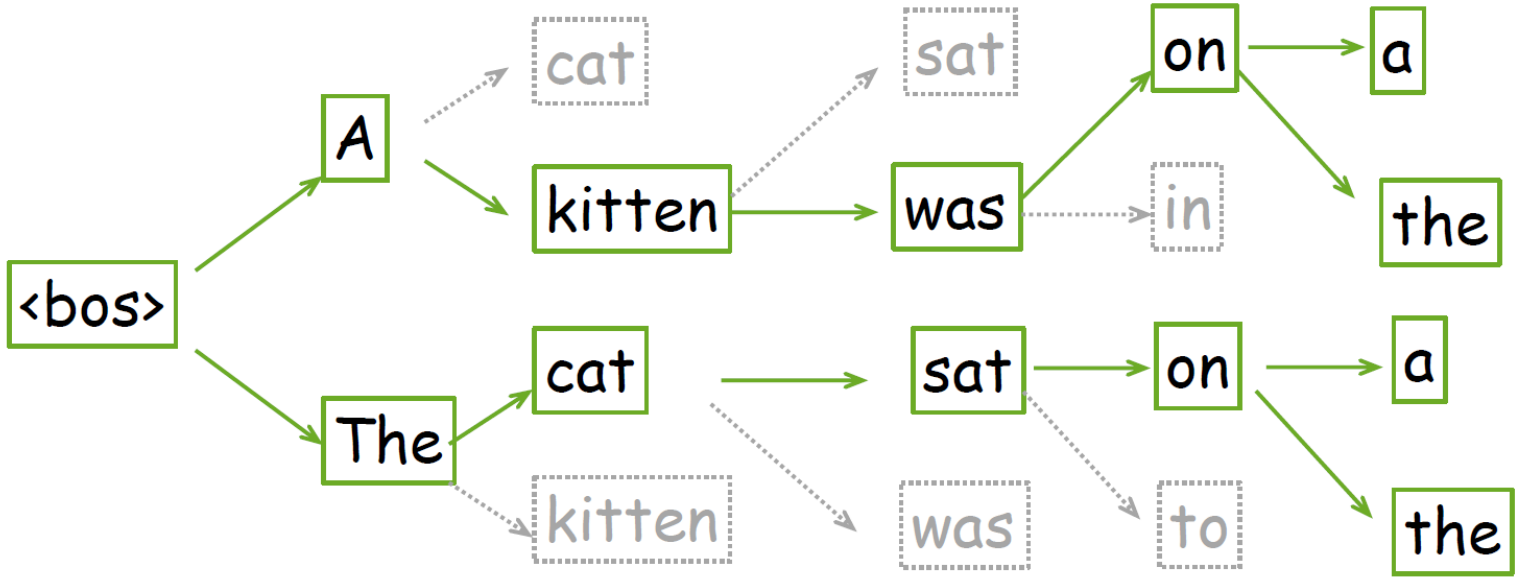
Beam Search (лучевой поиск)



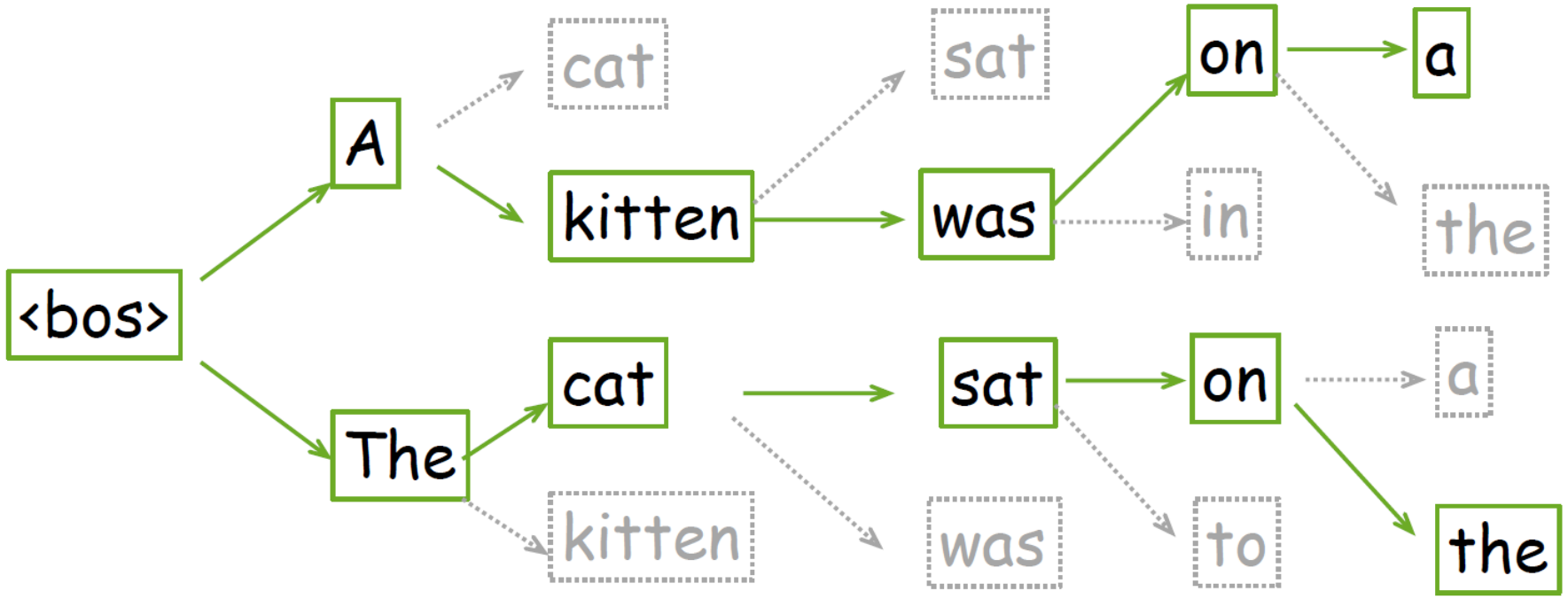
Beam Search (лучевой поиск)



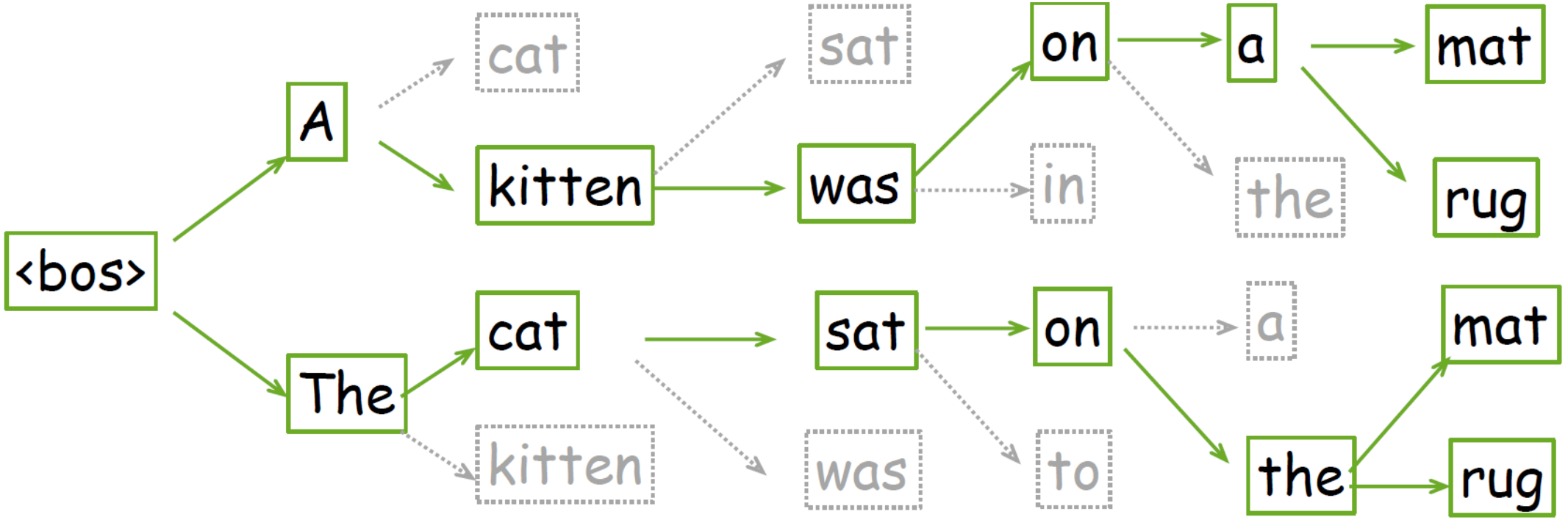
Beam Search (лучевой поиск)



Beam Search (лучевой поиск)

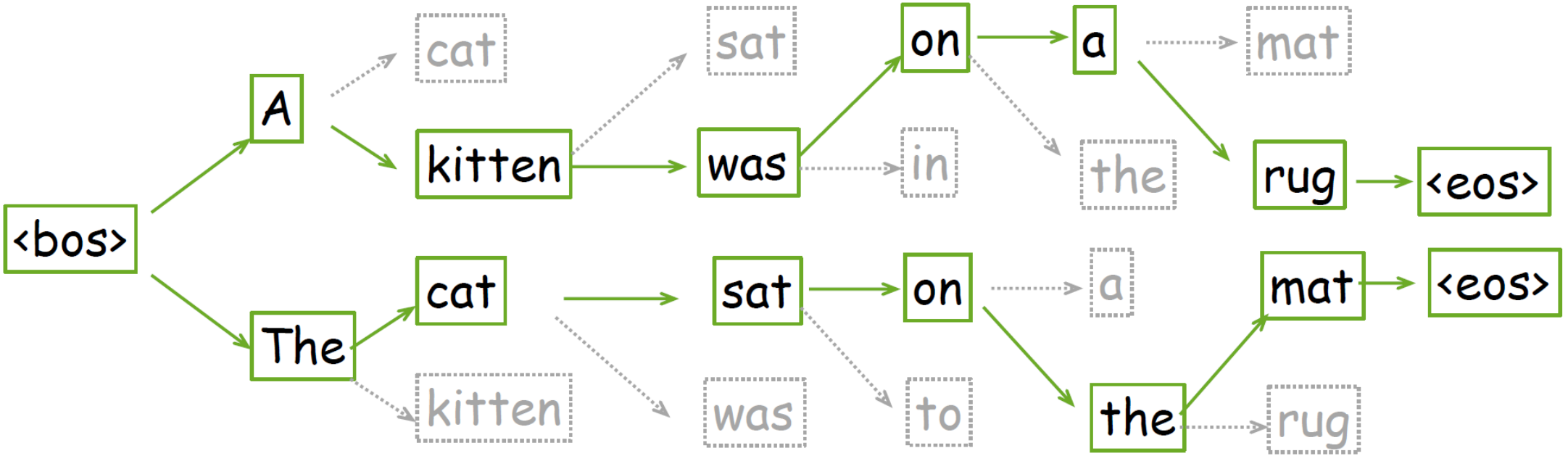


Beam Search (лучевой поиск)



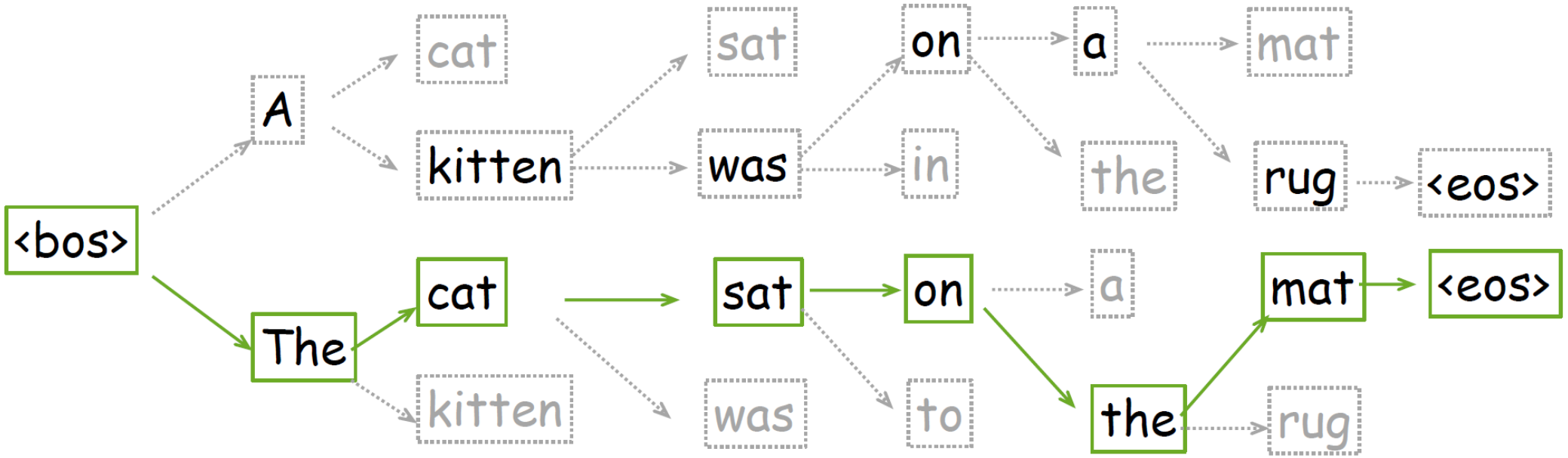
Beam Search (лучевой поиск)

Все гипотезы завершены, генерация окончена



Beam Search (лучевой поиск)

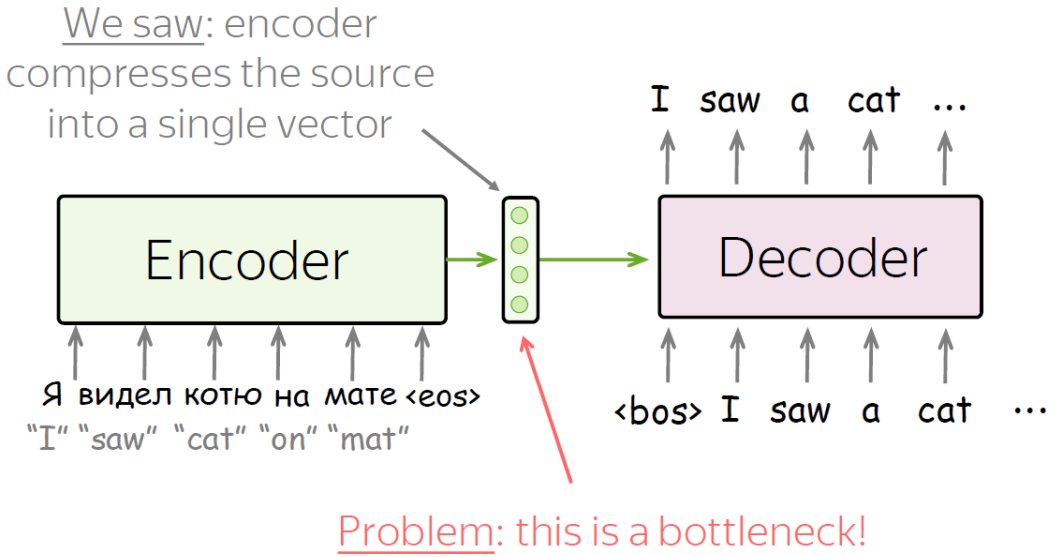
Выберем гипотезу с наибольшей вероятностью.



Attention. Проблема фиксированного представления кодера

Фиксированное представление исходного текста плохо:

- кодеру сложно сжать предложение
- декодеру может потребоваться разная информация на разных этапах



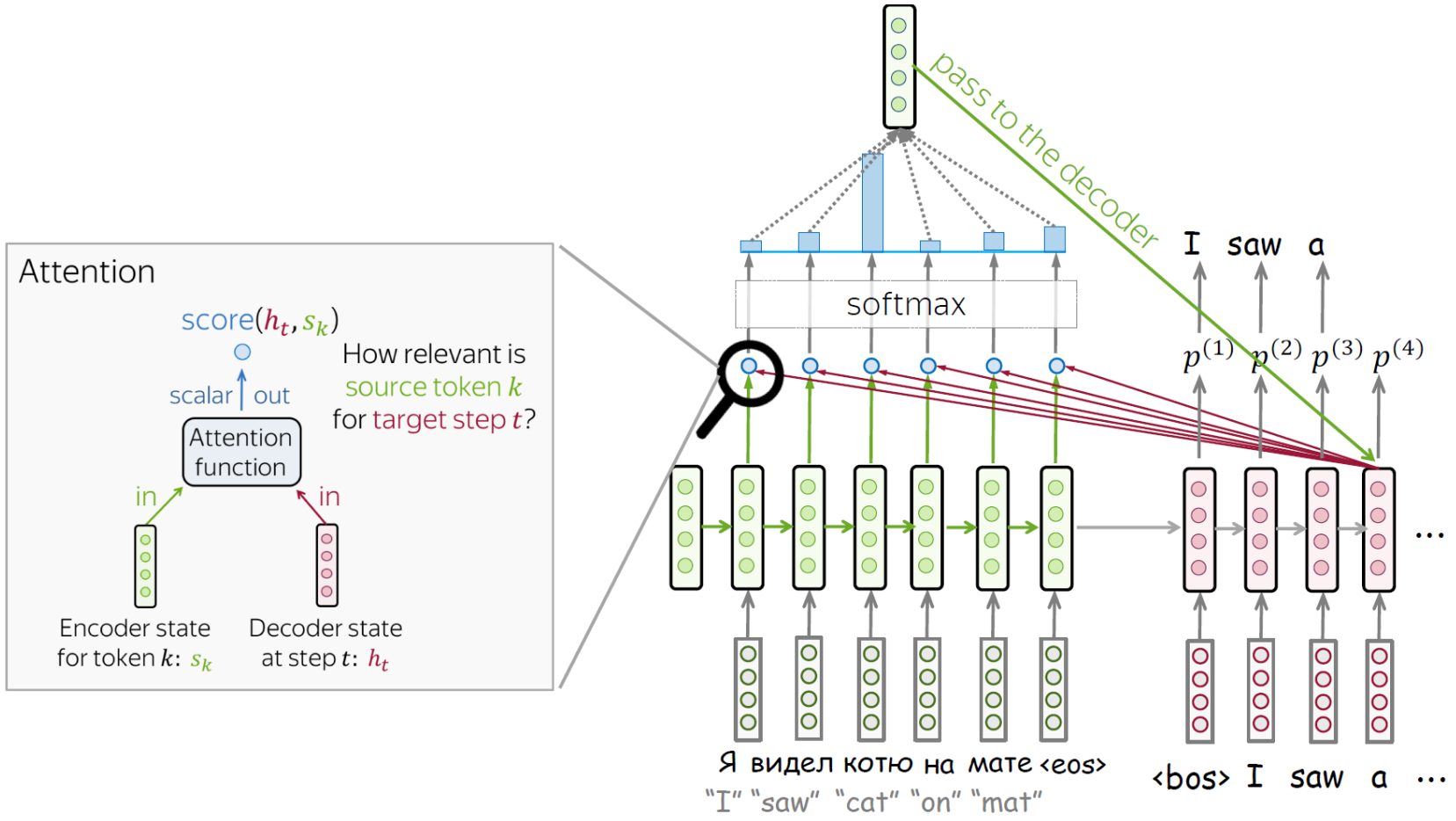
Attention

Attention: на разных этапах позволяем модели «фокусироваться» на разных частях входных данных.

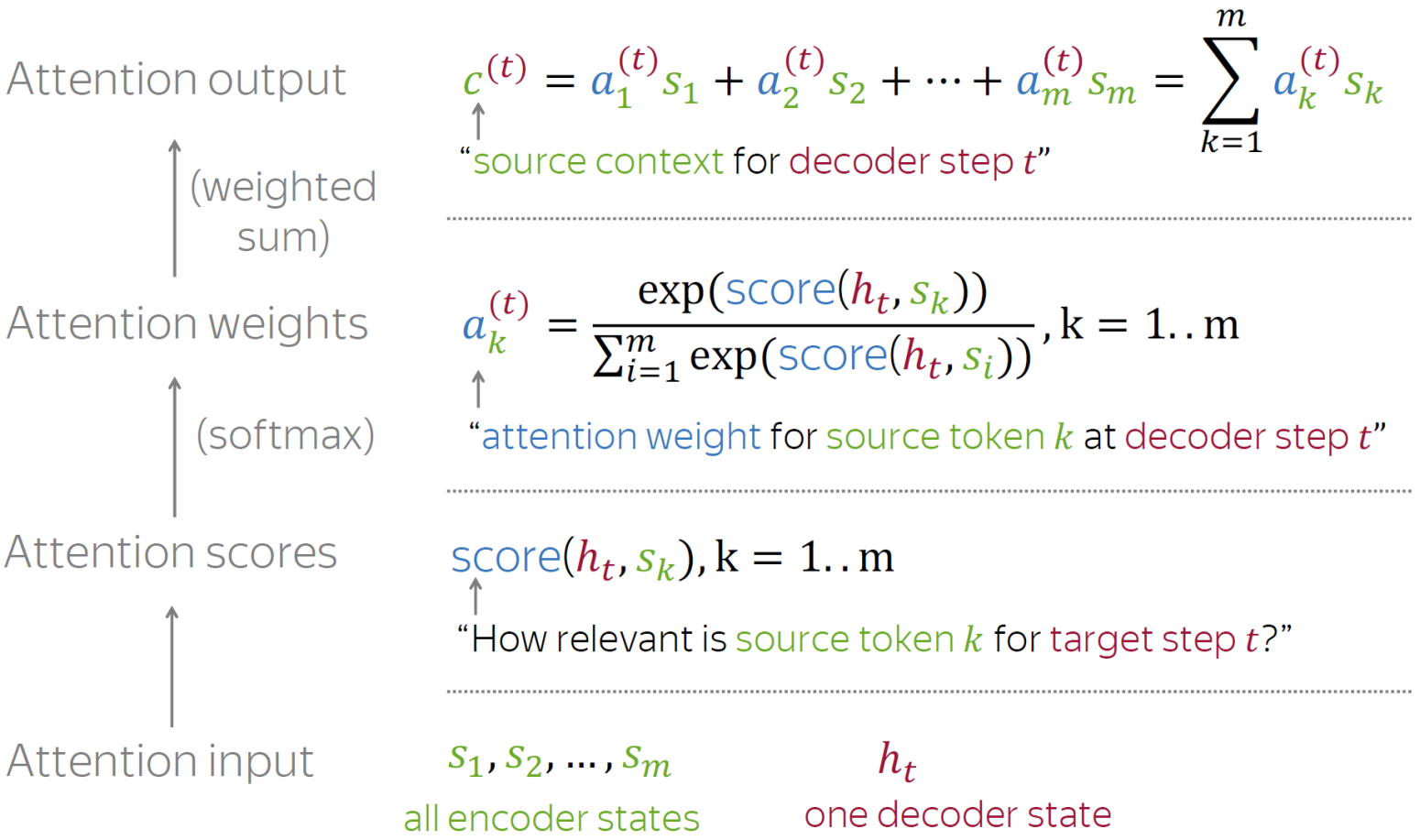
Выход attention: взвешенная сумма состояний кодера с весами attention.

Веса attention: распределение по исходным токенам.

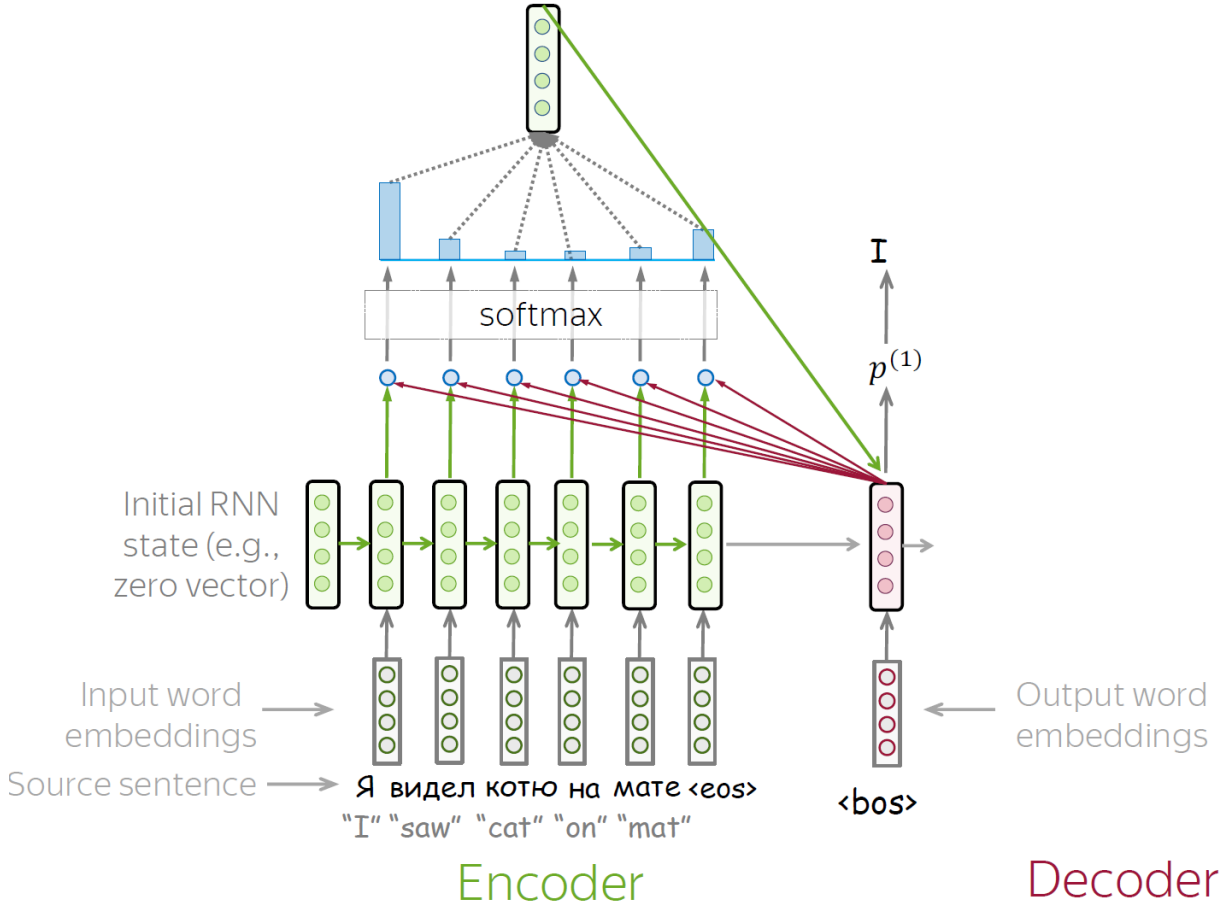
Attention


Attention
score(h_t, s_k)
scalar
out
Attention function
in
in
How relevant is source token k for target step t?
Encoder state for token k: s_k
Decoder state at step t: h_t
pass to the decoder
softmax
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
I saw a
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...

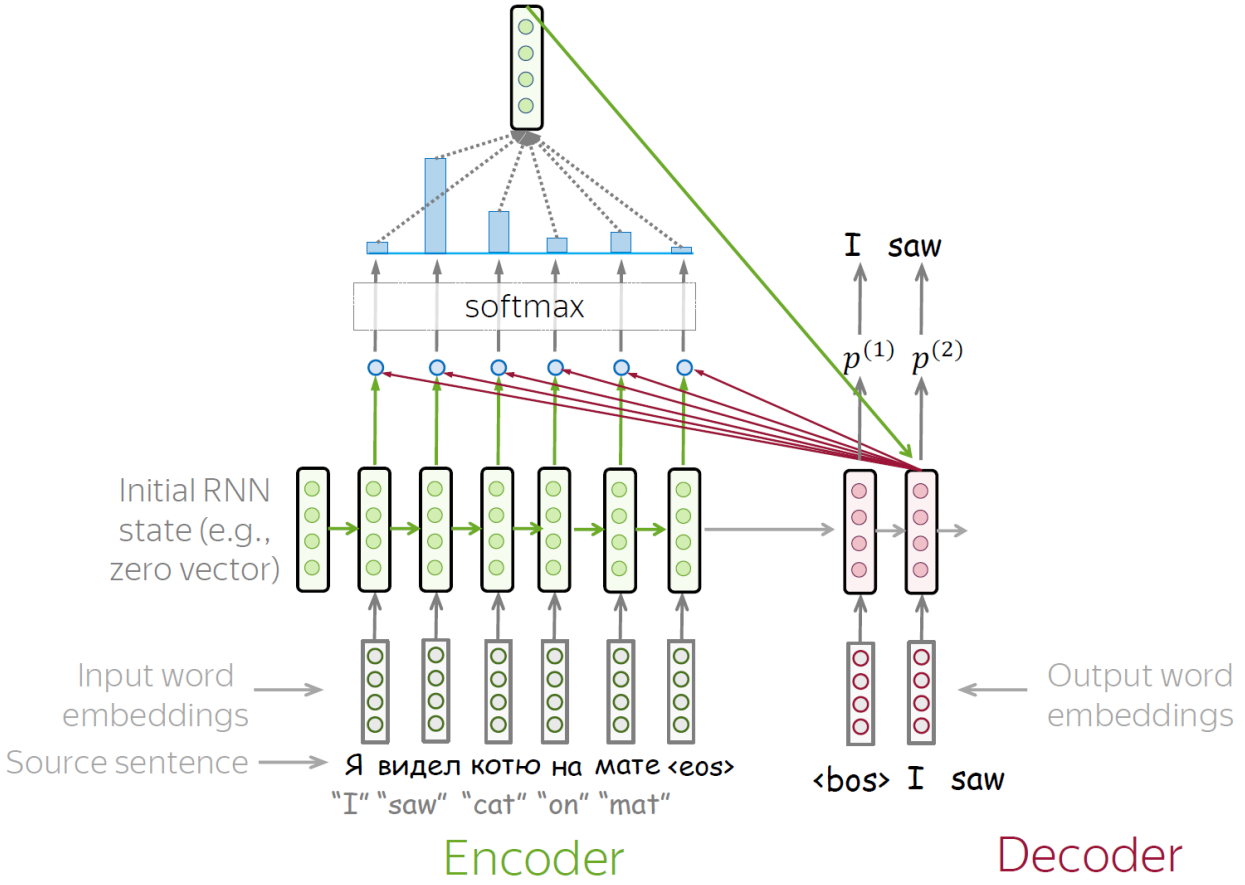
Вычислительная схема



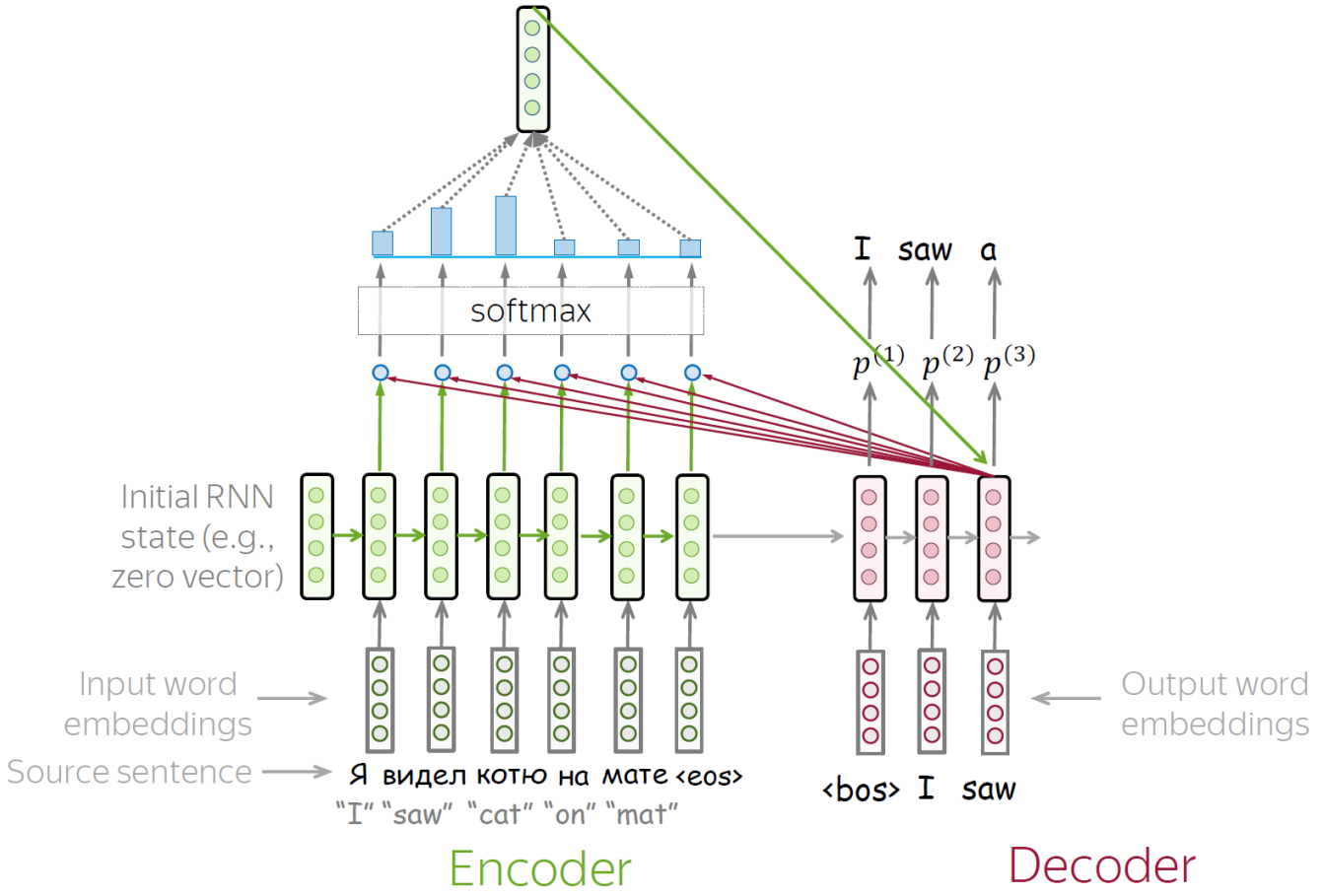
Модель учится выбирать соответствующие токены



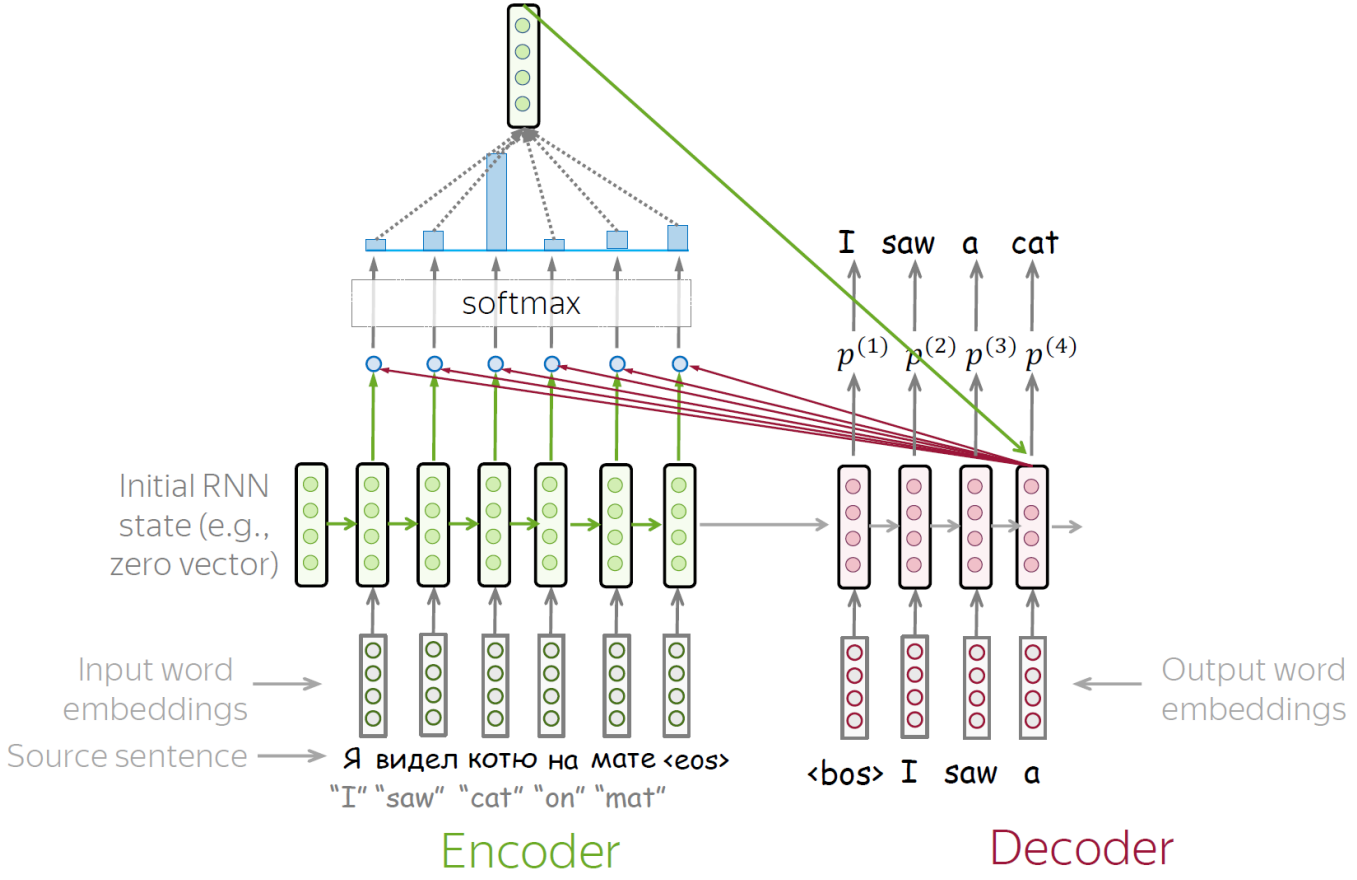
Модель учится выбирать соответствующие токены



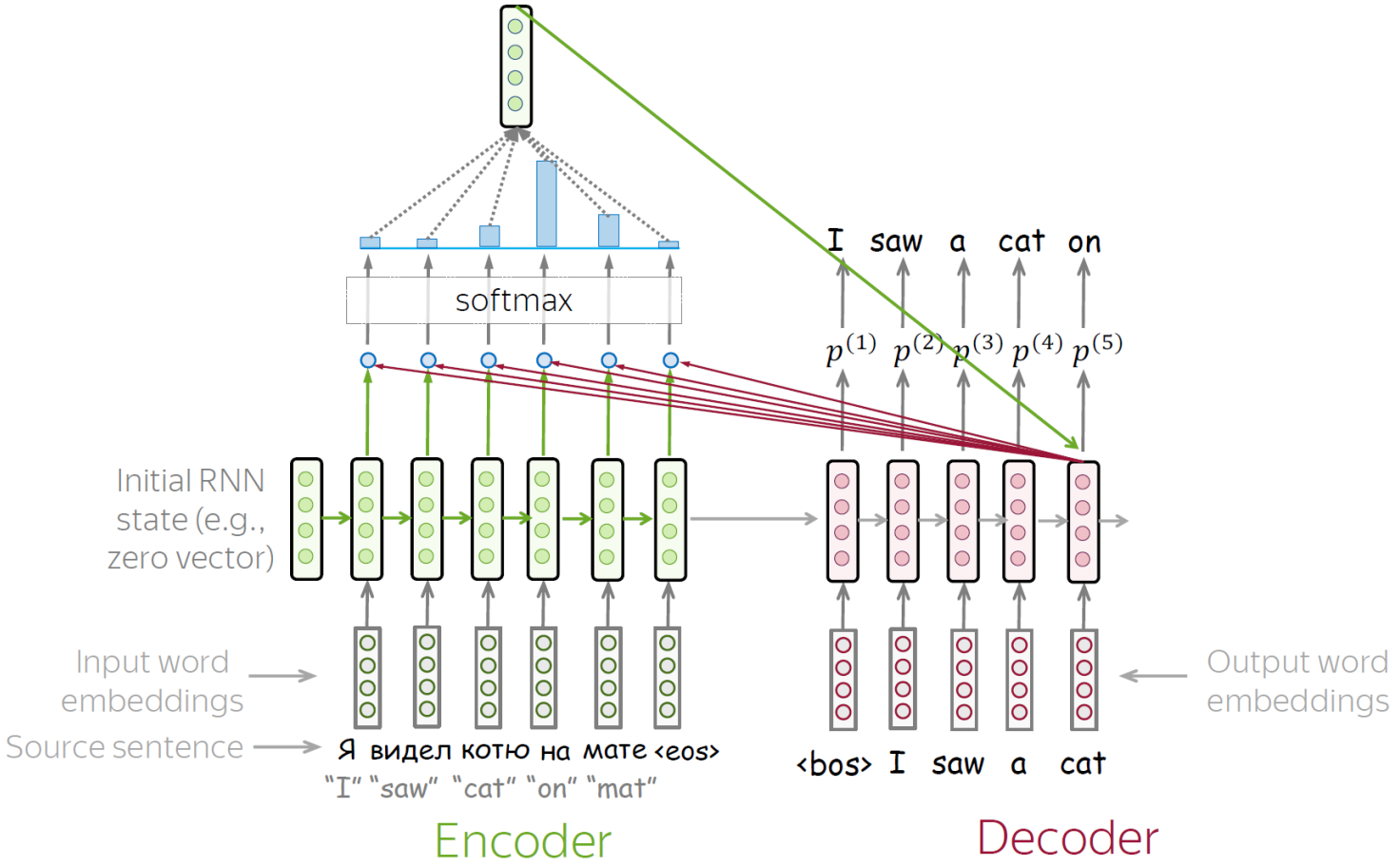
Модель учится выбирать соответствующие токены



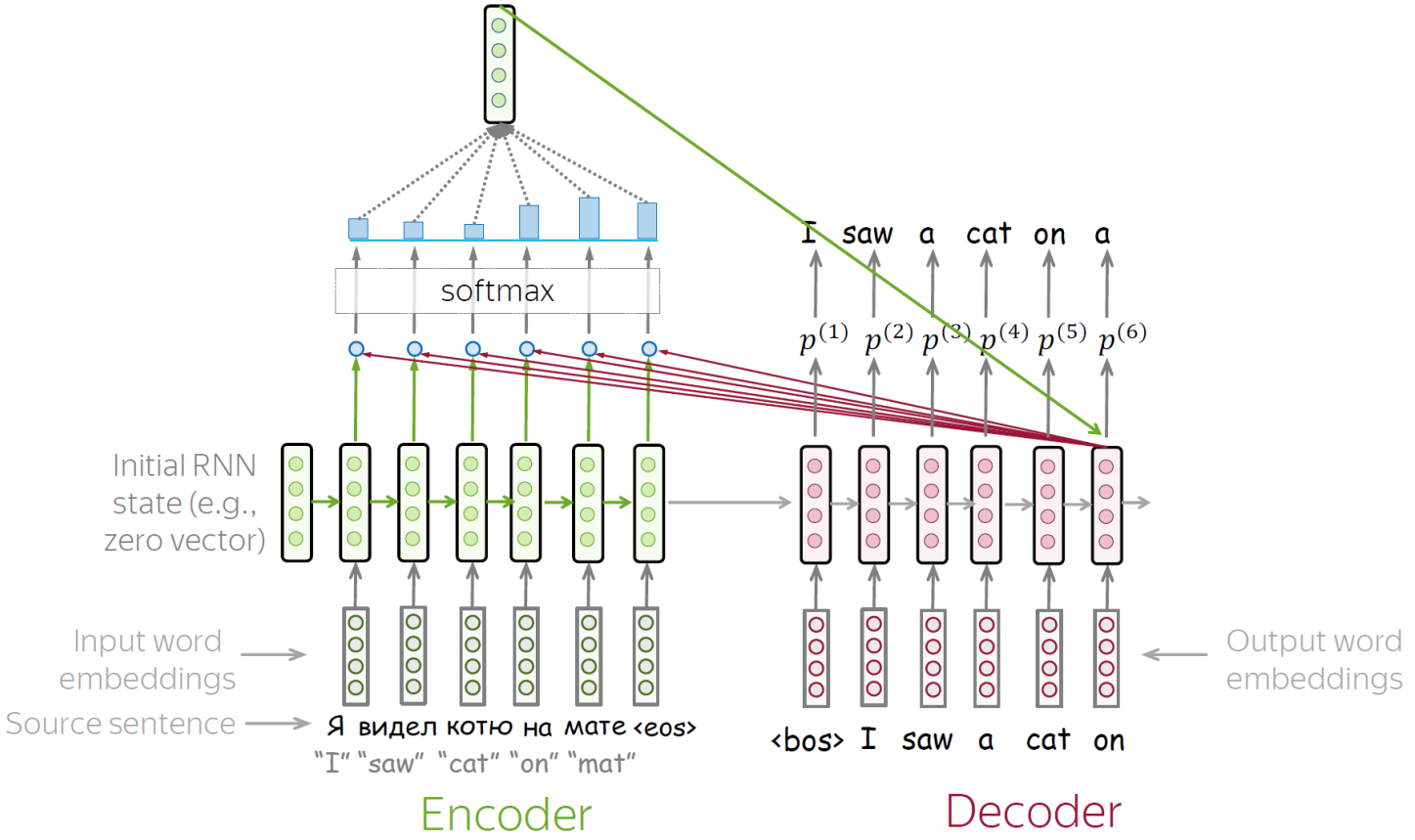
Модель учится выбирать соответствующие токены



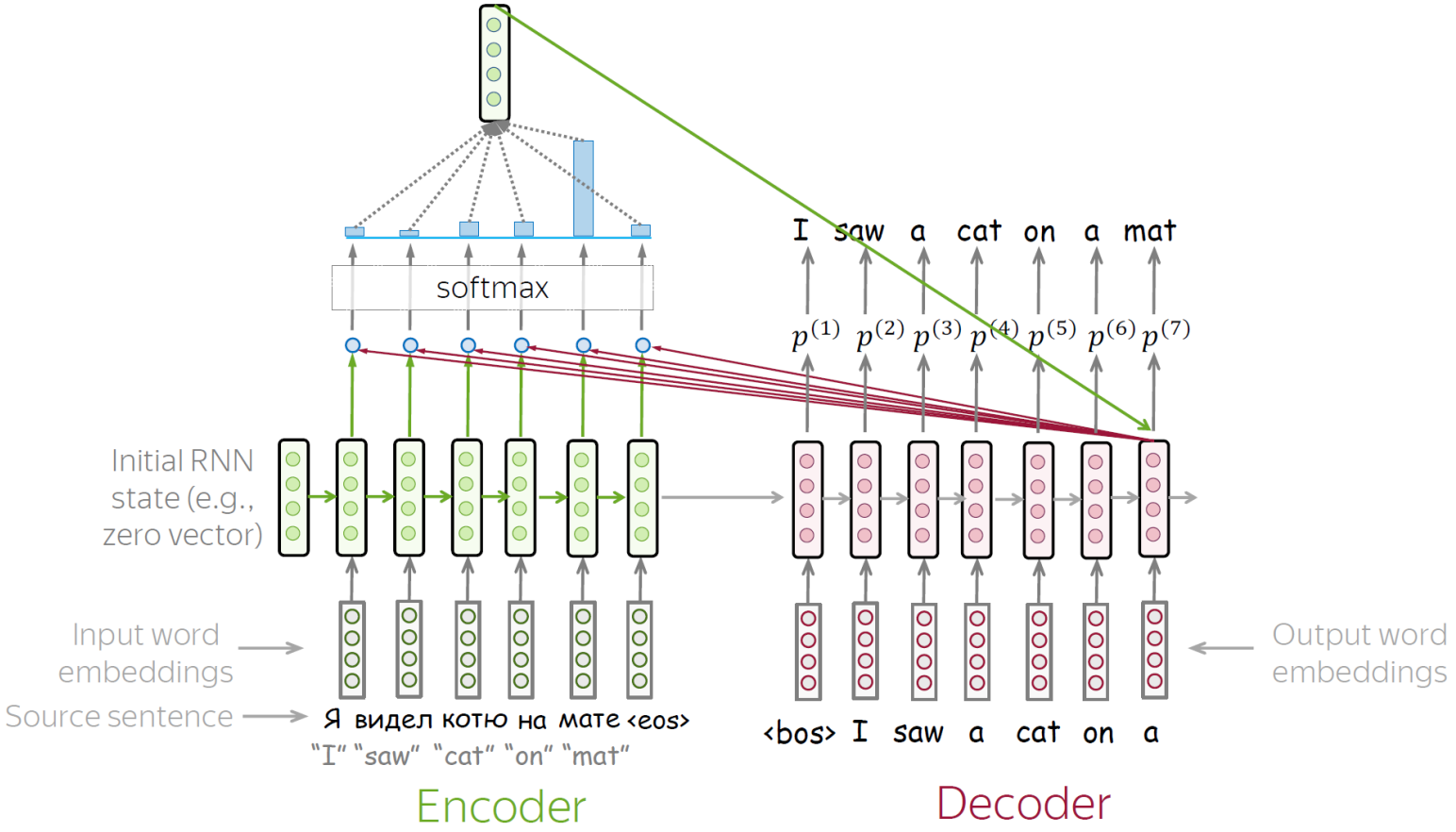
Модель учится выбирать соответствующие токены



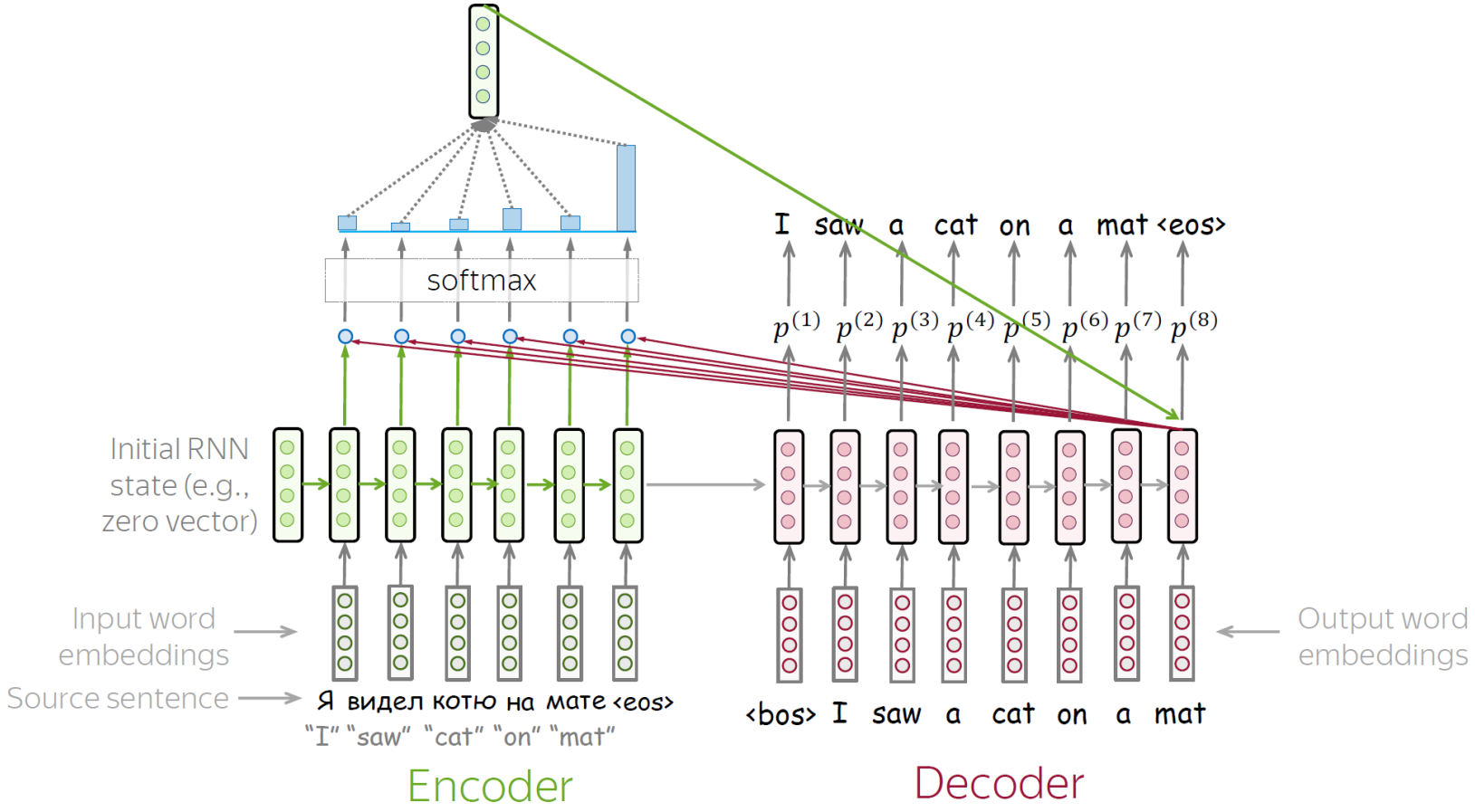
Модель учится выбирать соответствующие токены



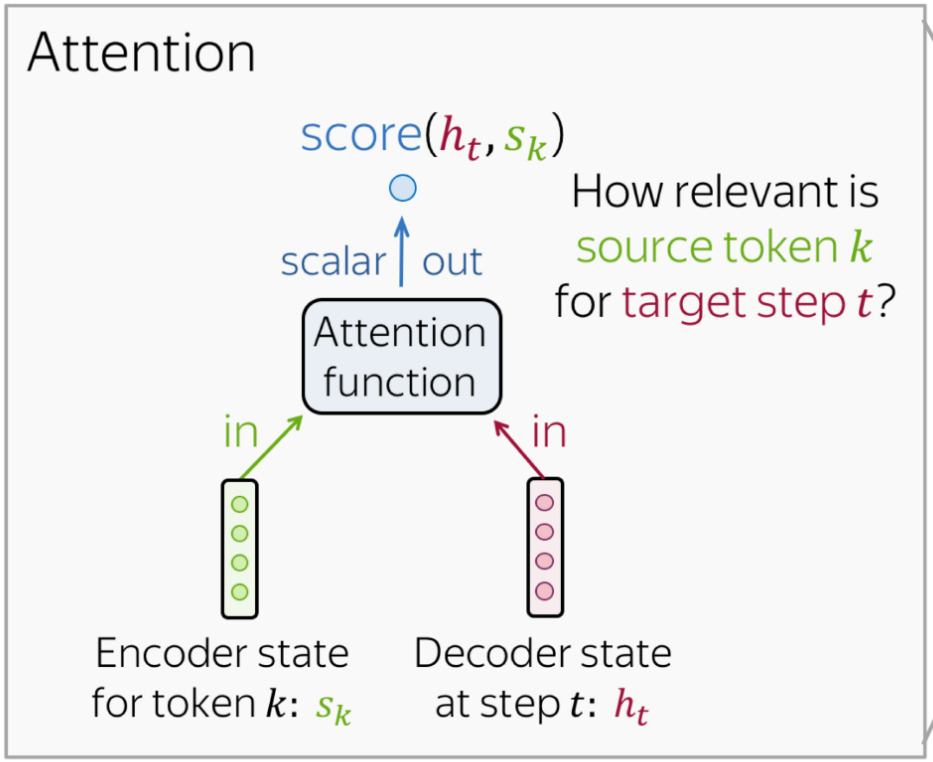
Модель учится выбирать соответствующие токены

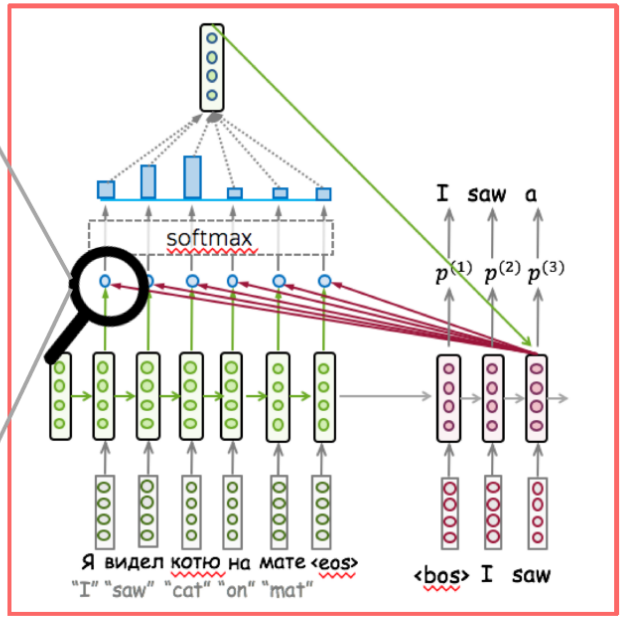


Модель учится выбирать соответствующие токены



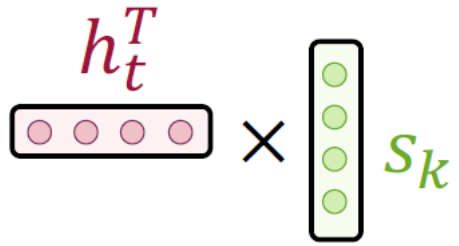
Функции оценки attention



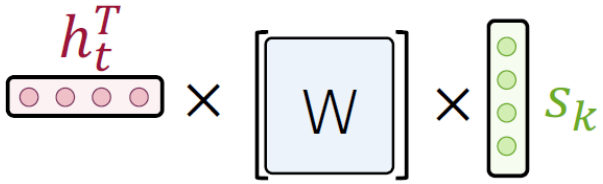


Функции оценки attention

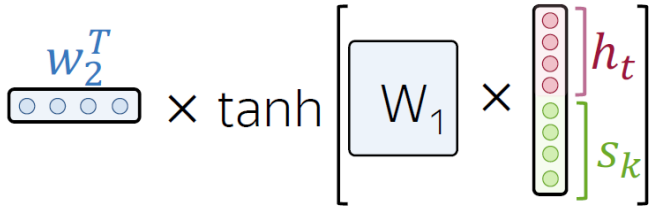
- Скалярное произведение: $\text{score}(h_t, s_k) = h_t^T \, s_k$



- Билинейная функция: $\text{score}(h_t, s_k) = h_t^T \, W \, s_k$

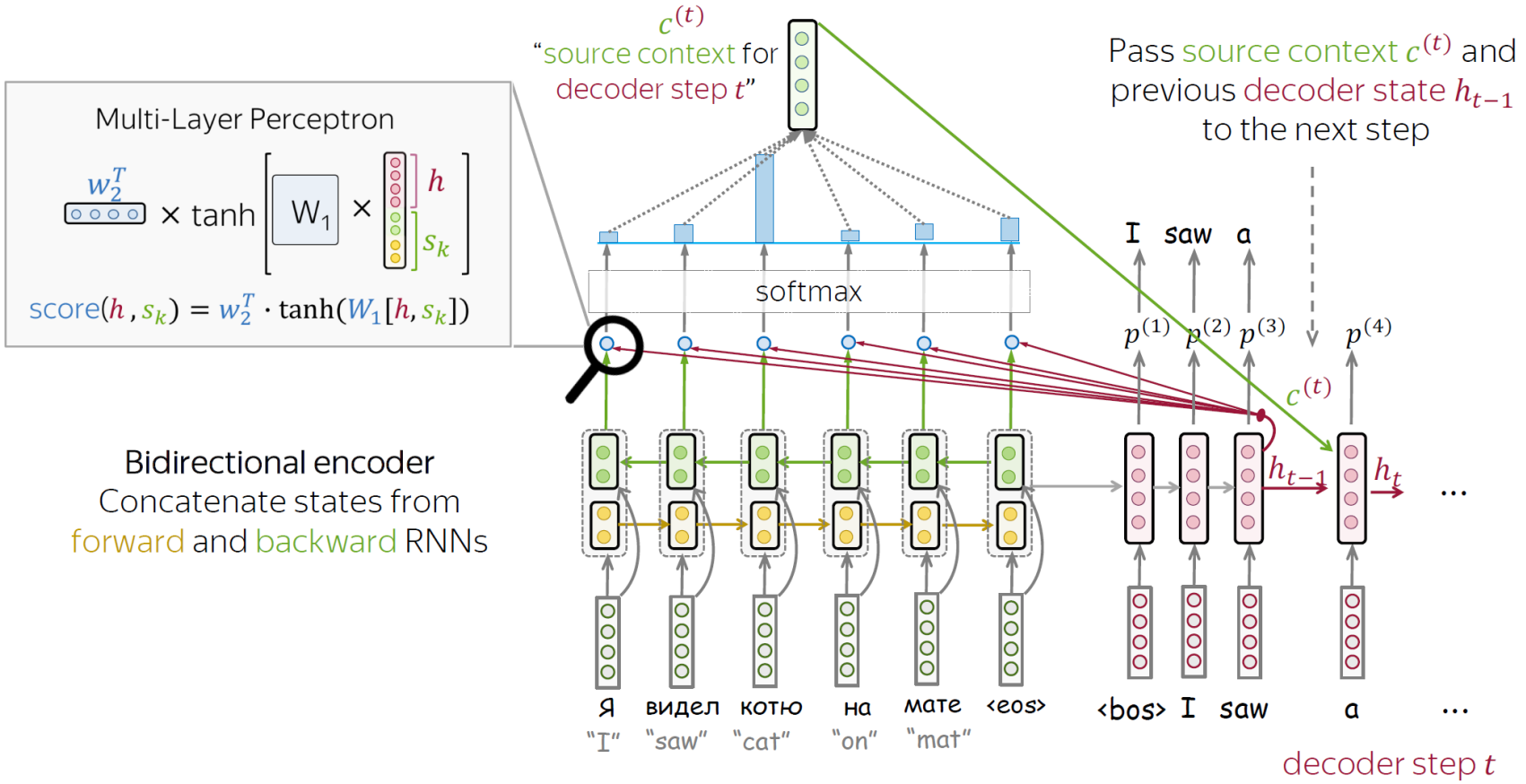


- Многослойный персептрон:

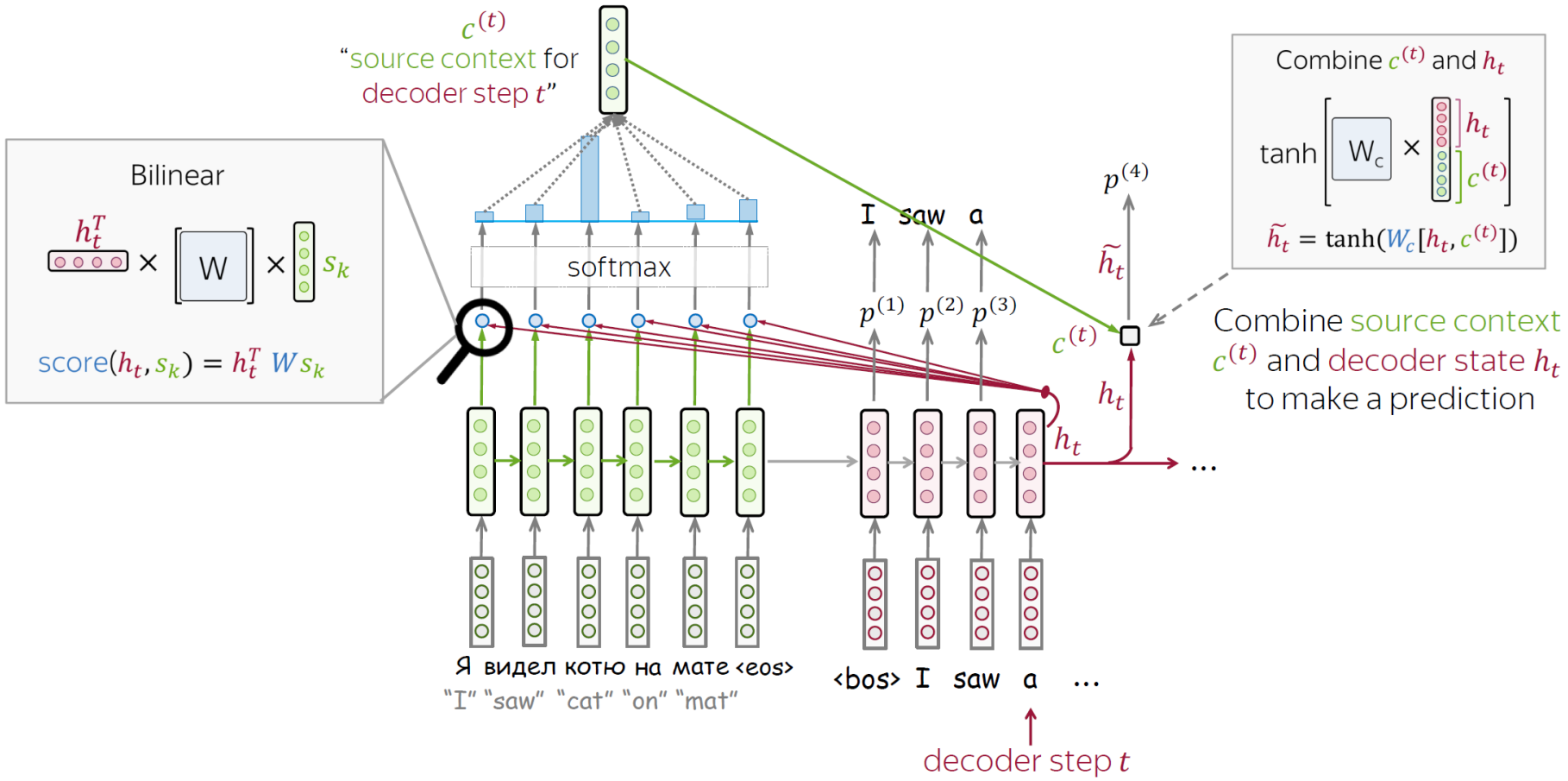


$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1 [h_t, s_k])$$

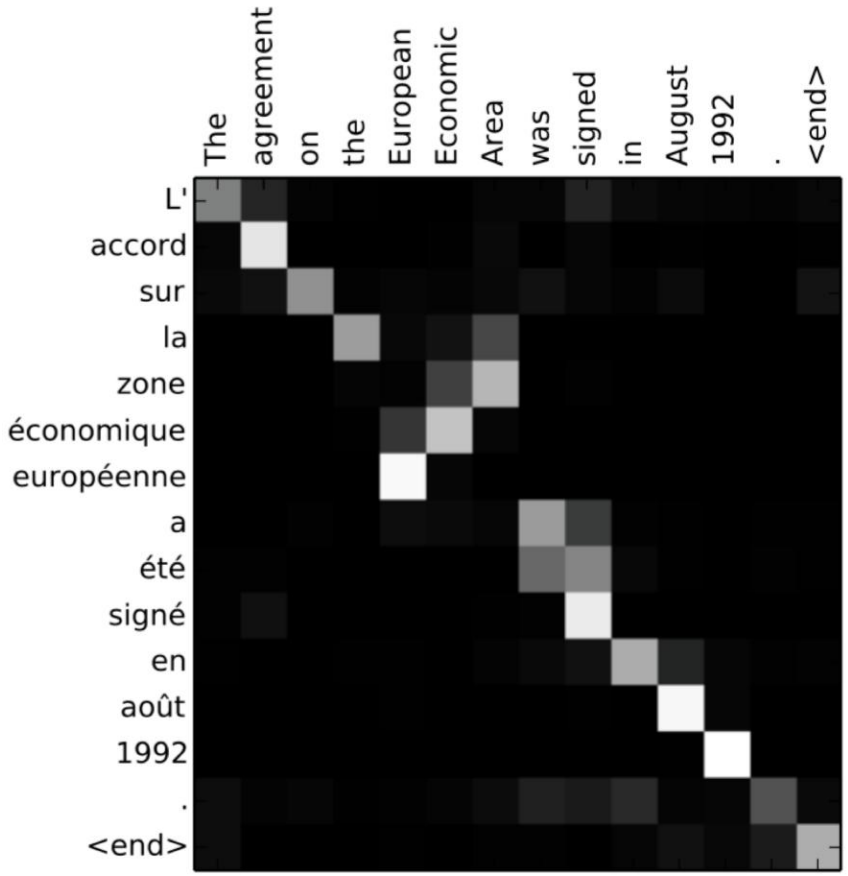
Модель Богданова (исходная модель attention)

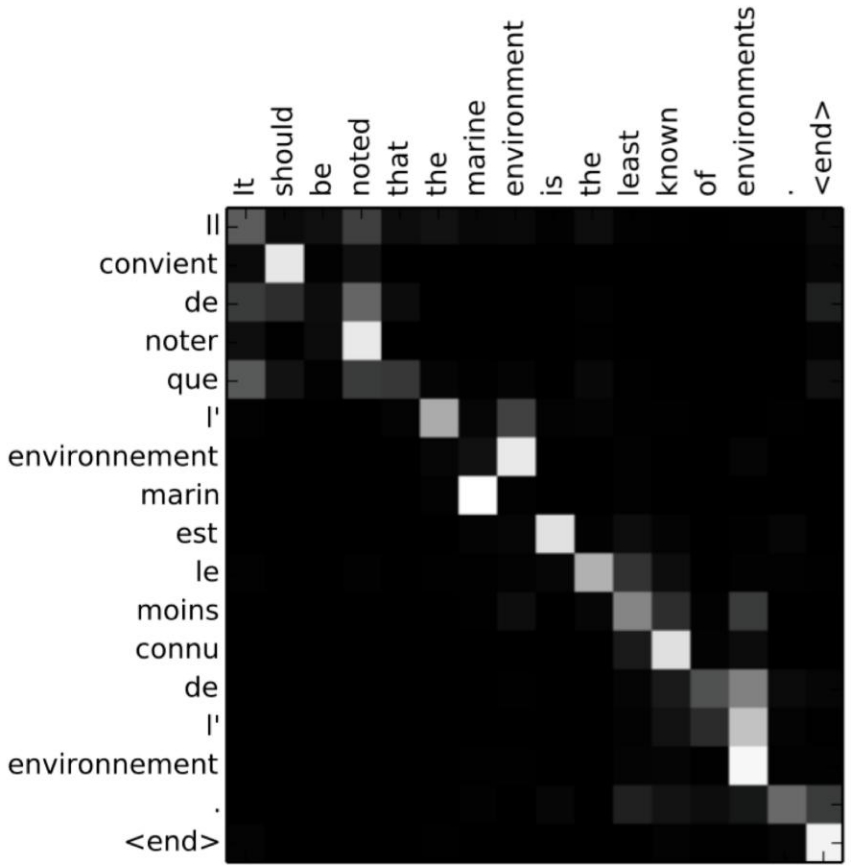


Модель Луонга



Выравнивание, выученное с attention





Transformer: основная идея

	Seq2seq without attention	Seq2seq with attention	Transformer
processing within encoder	RNN/CNN	RNN/CNN	attention
processing within decoder	RNN/CNN	RNN/CNN	attention
decoder-encoder interaction	static fixed-sized vector	attention	attention

Attention

Кодировщик

Все исходные токены смотрят друг на друга и обновляют представления (повторить N раз).

Декодировщик

Целевой токен на текущем шаге смотрит на предыдущие целевые токены и на исходные представления, обновляет представление (повторить N раз).

Почему это может быть лучше, чем RNN?

I arrived at the bank after crossing the ... ...street? ...river?

What does **bank** mean in this sentence?

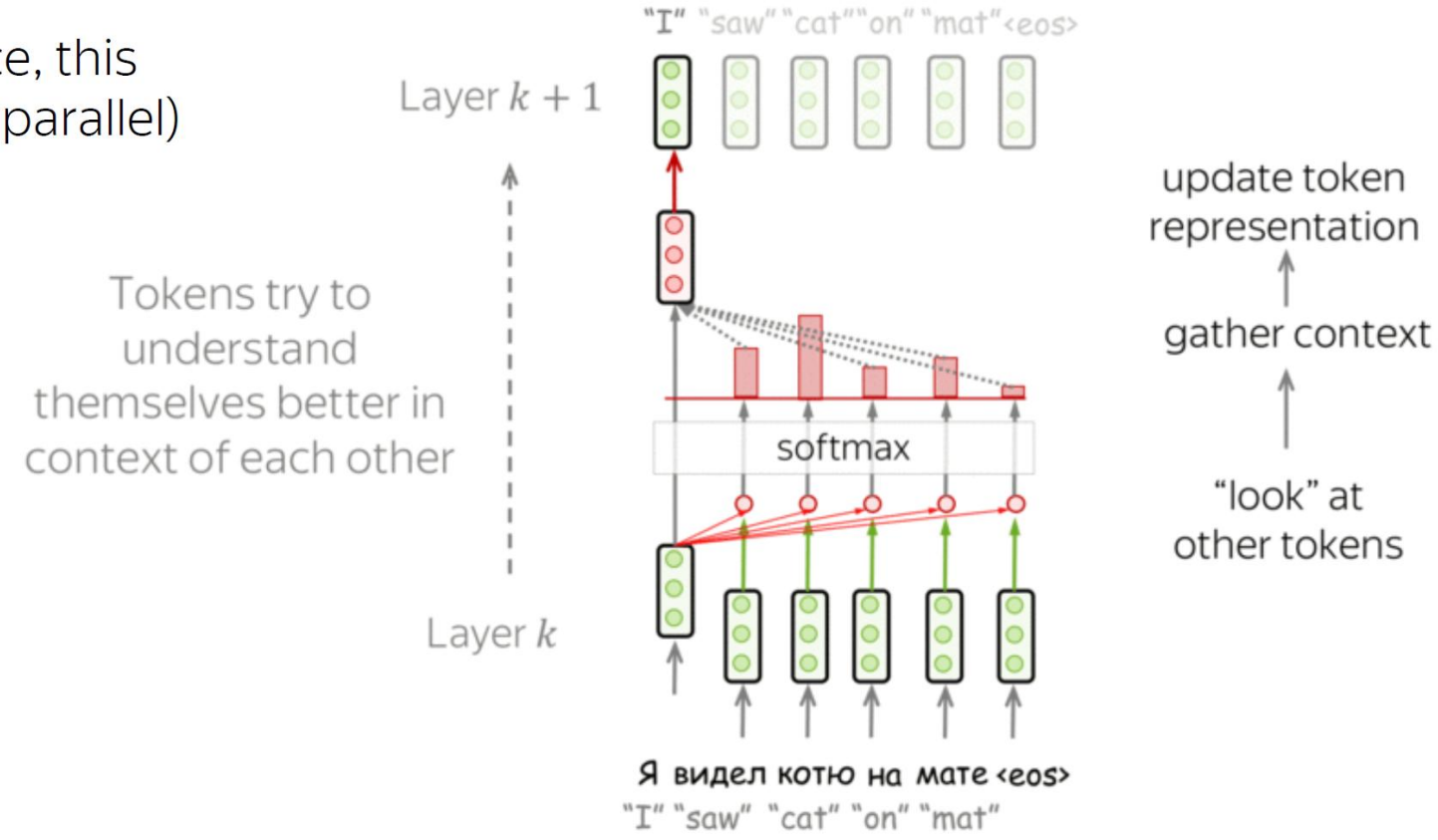
Self-Attention

Attention направлено из одного текущего состояния декодера во все состояния кодировщика.

Self-Attention смотрит из каждого состояния из набора состояний во все остальные состояния в том же наборе.

Self-Attention

(in practice, this happens in parallel)



Запрос, Ключ, Значение (Query, Key, Value)

Каждый вектор получает три представления («роли»):

Запрос (query). Вектор, на который направлено attention

«Привет, у тебя есть эта информация?»

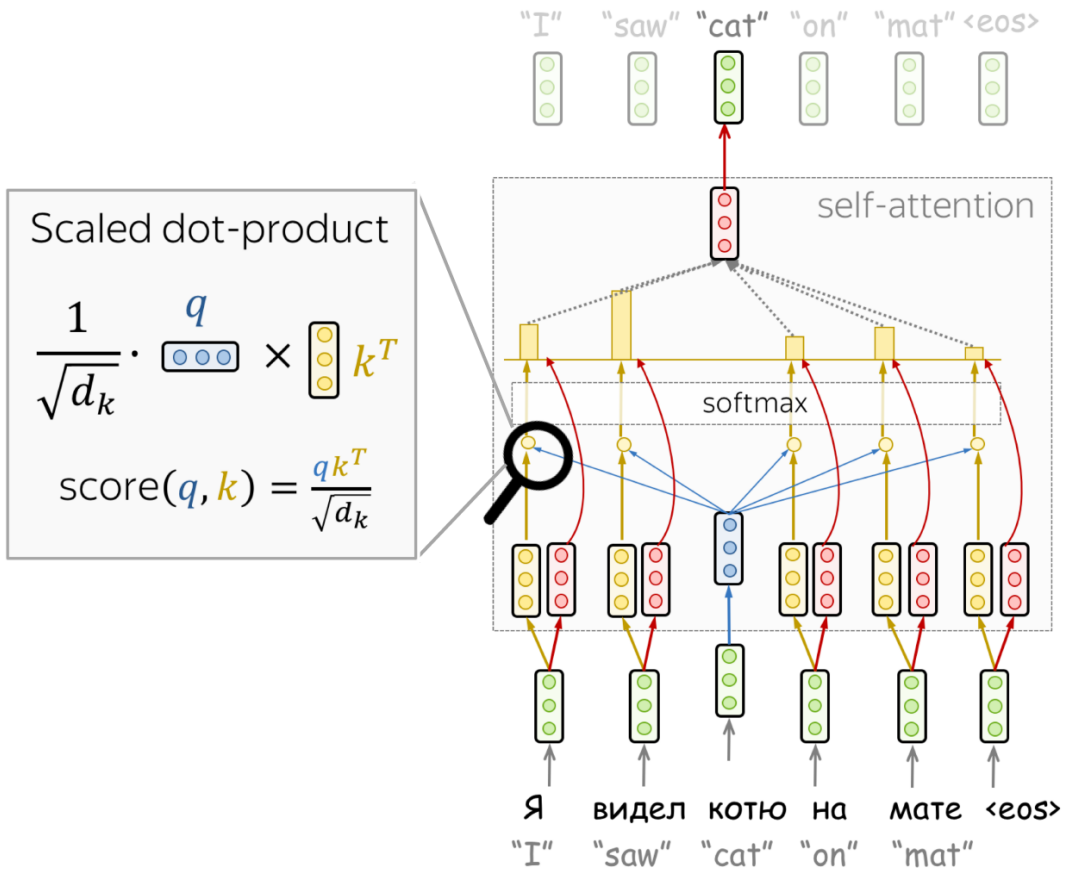
Ключ (key). Вектор, к которому обращается запрос для вычисления весов

«Привет, у меня есть информация — дайте мне большой вес!»

Значение (value). Их взвешенная сумма – это attention output

«Вот информация, которая у меня есть!»

Запрос, Ключ, Значение (Query, Key, Value)

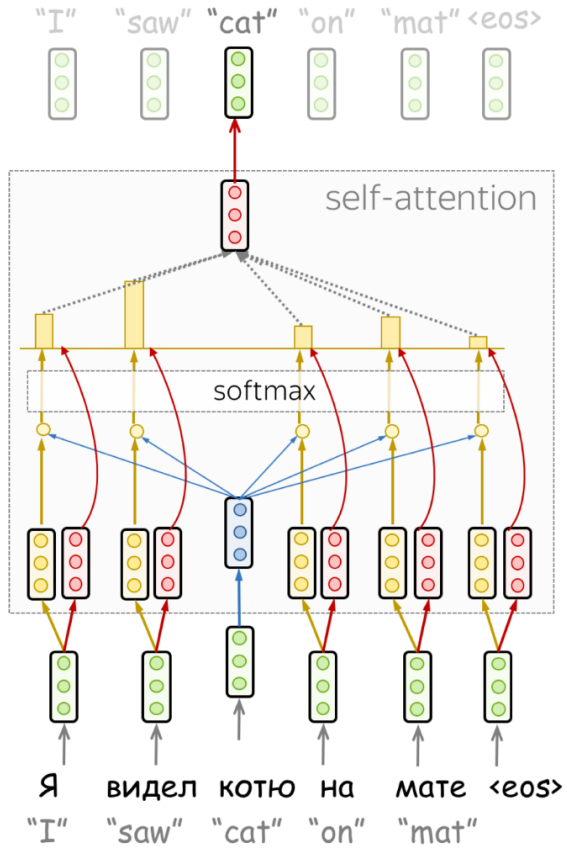


Запрос, Ключ, Значение (Query, Key, Value)

$$Attention(q, k, v) = \overbrace{softmax\left(\frac{qk^T}{\sqrt{d_k}}\right)}^{\text{Attention weights}} v$$

from (q) to (k, v)

vector dimensionality of K, V

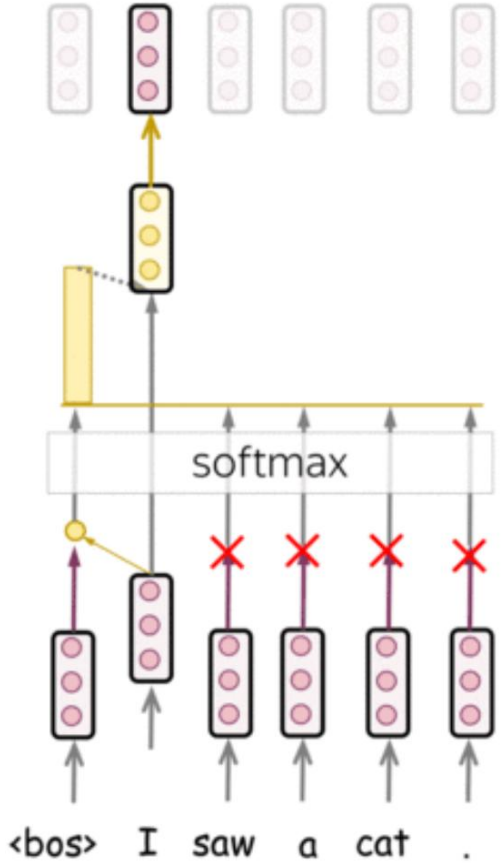


Скрытое Self-Attention

В декодере мы запрещаем просмотр будущих токенов – мы их не знаем.

Примечание: при обучении декодер обрабатывает все целевые токены одновременно – без масок он увидел бы будущее.

Скрытое Self-Attention

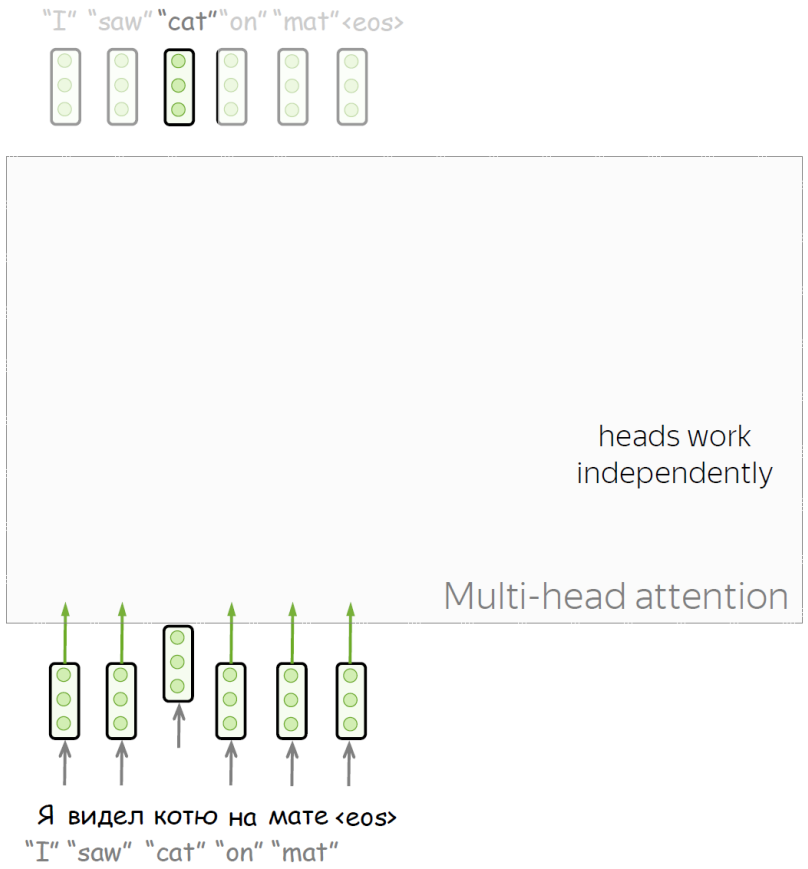


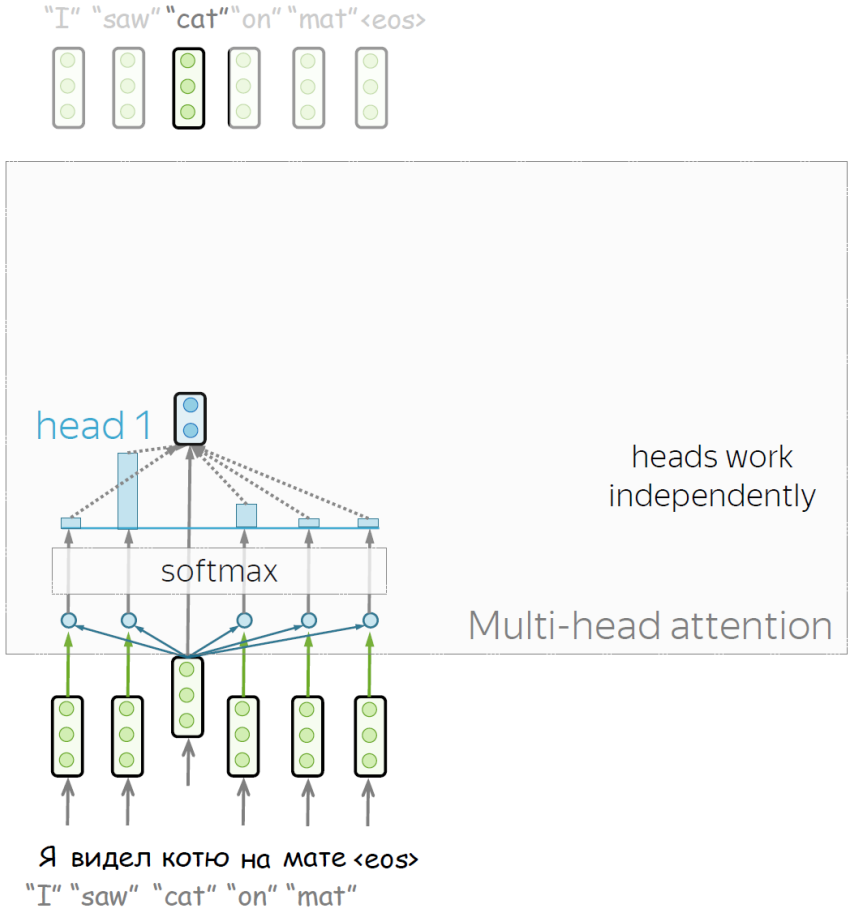
update token representation

gather context

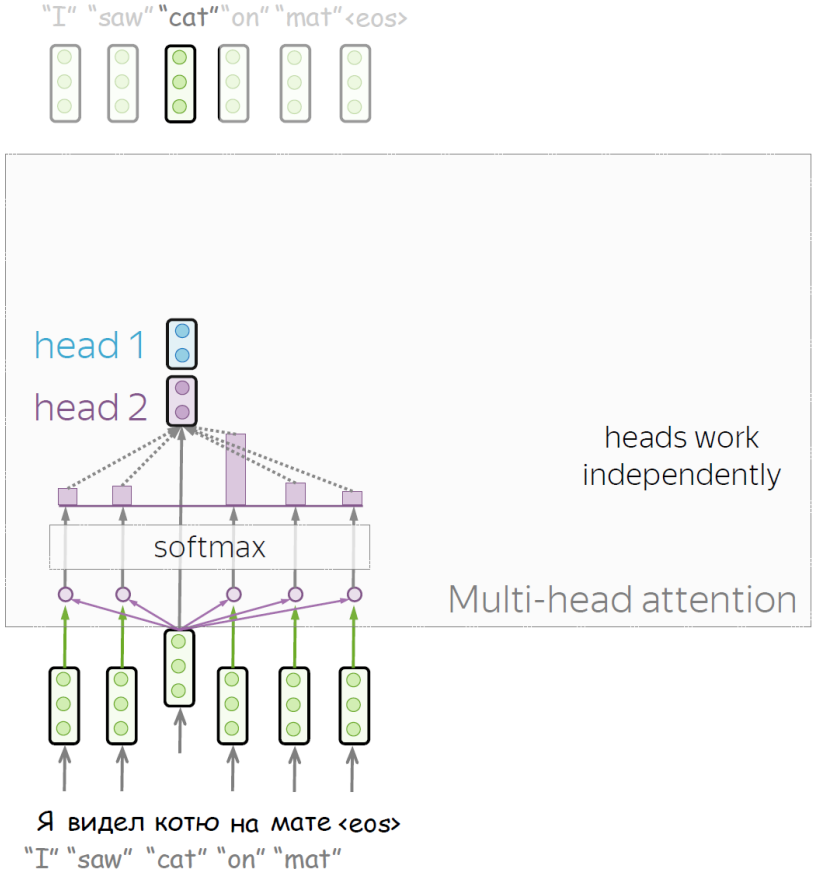
"look" at the previous tokens (future tokens are masked out)

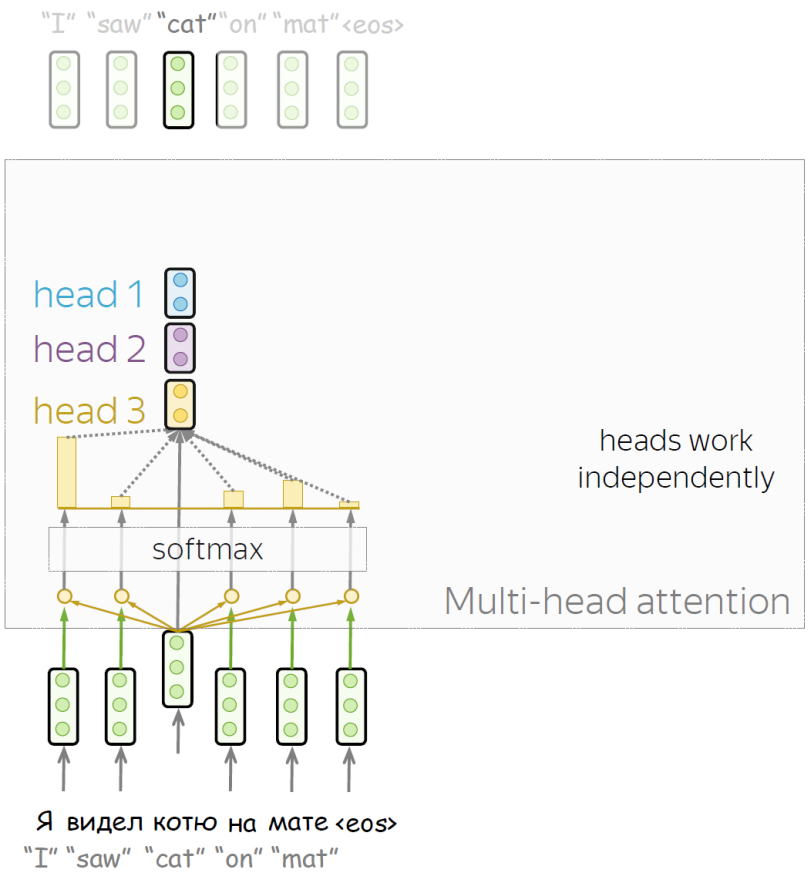
Multi-Head Attention



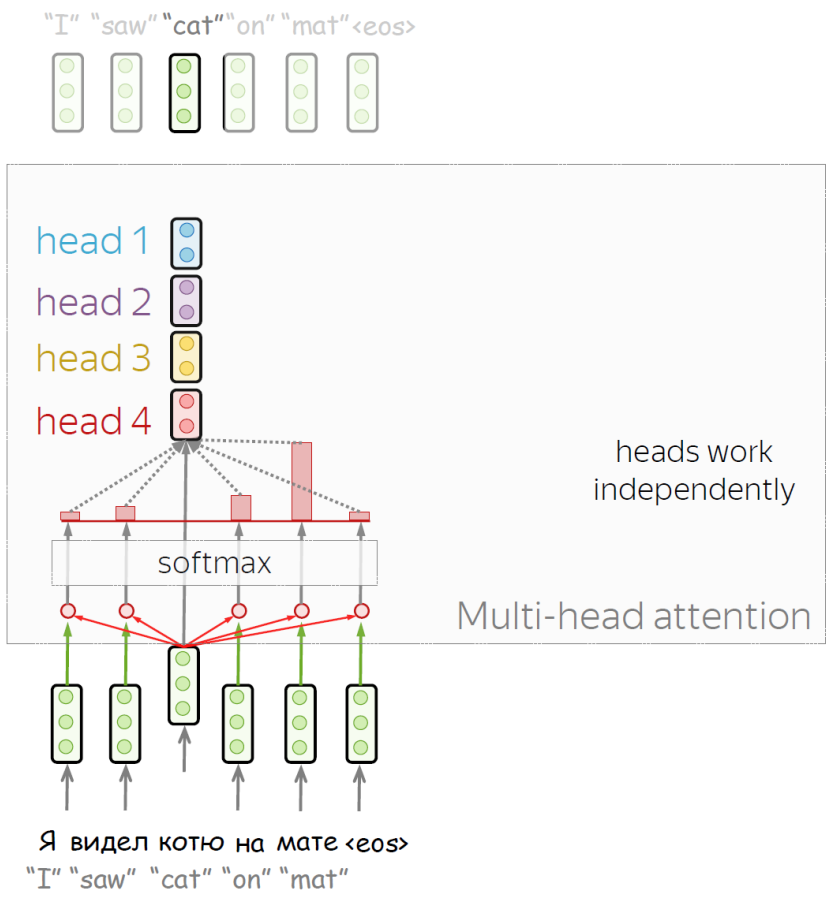


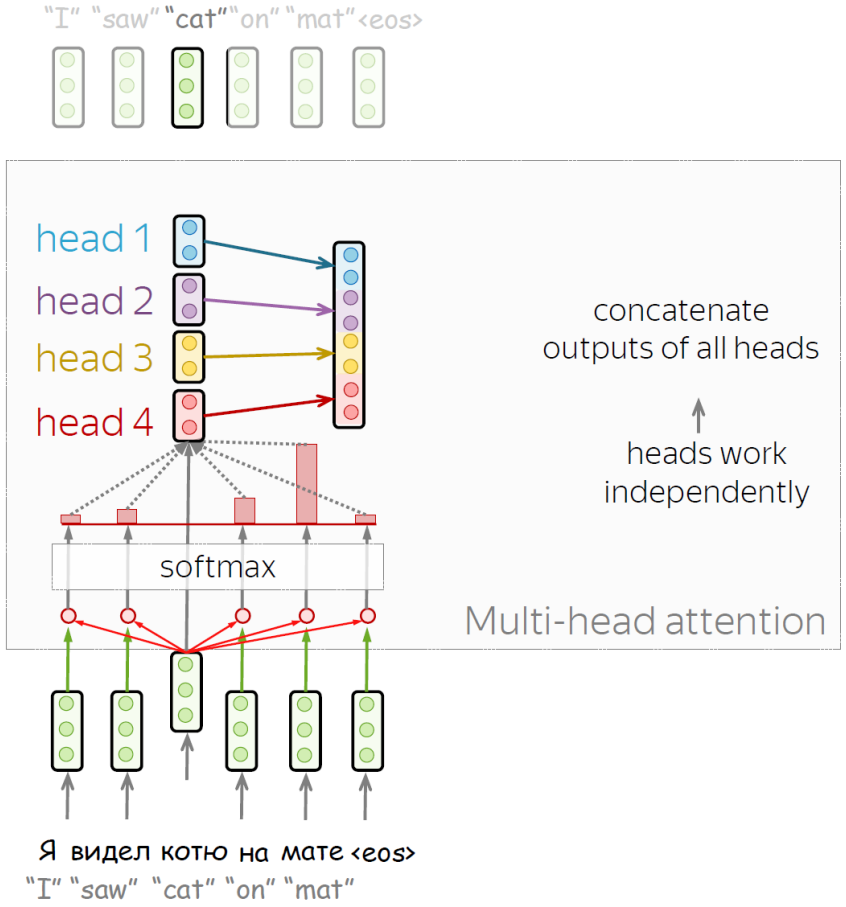
Multi-Head Attention



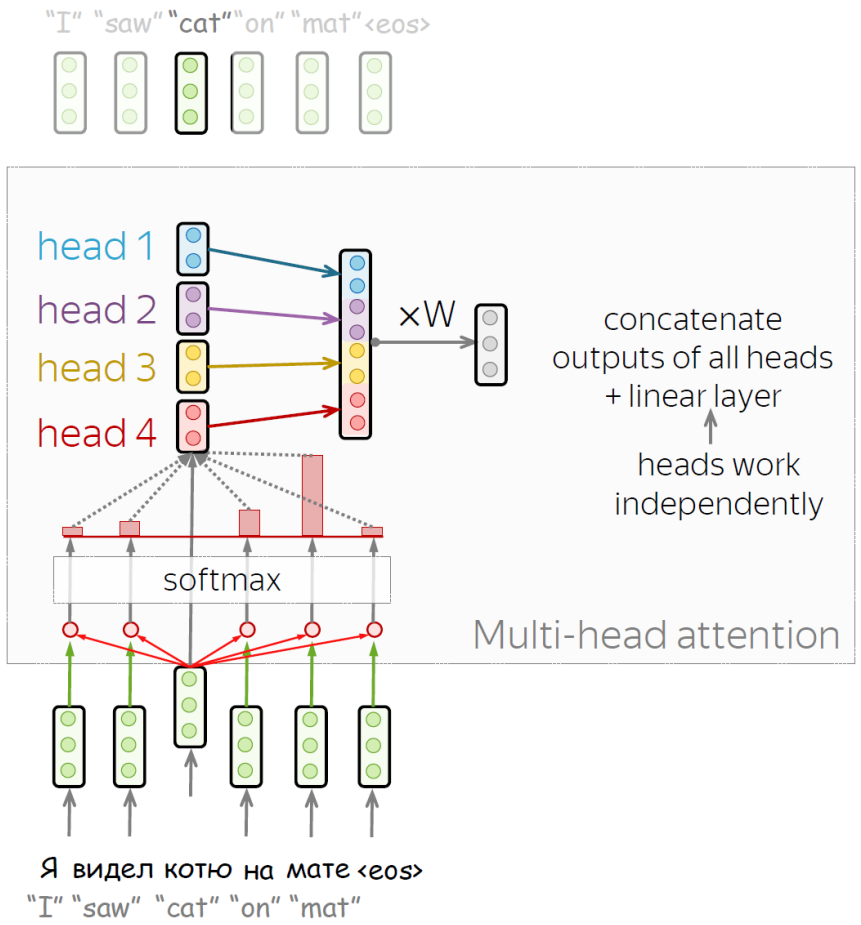


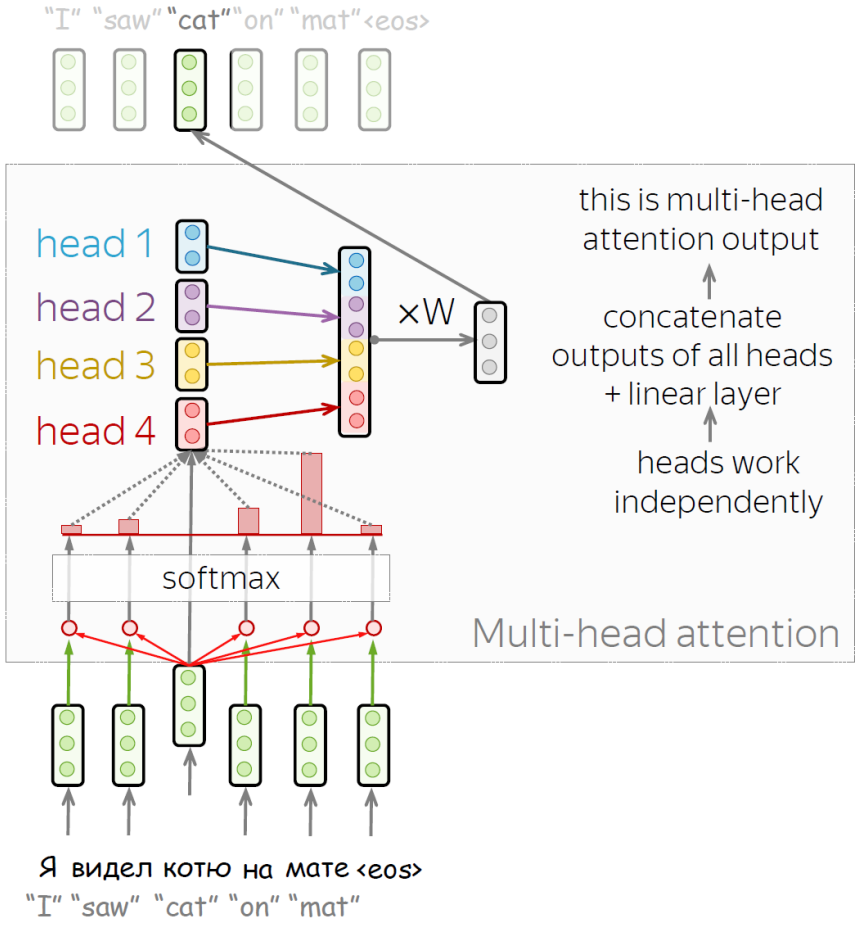
Multi-Head Attention





Multi-Head Attention

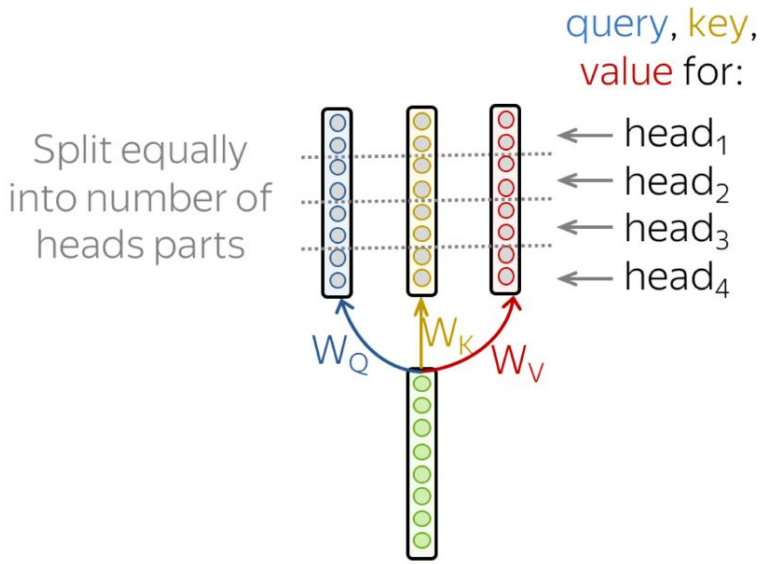




Multi-Head Attention

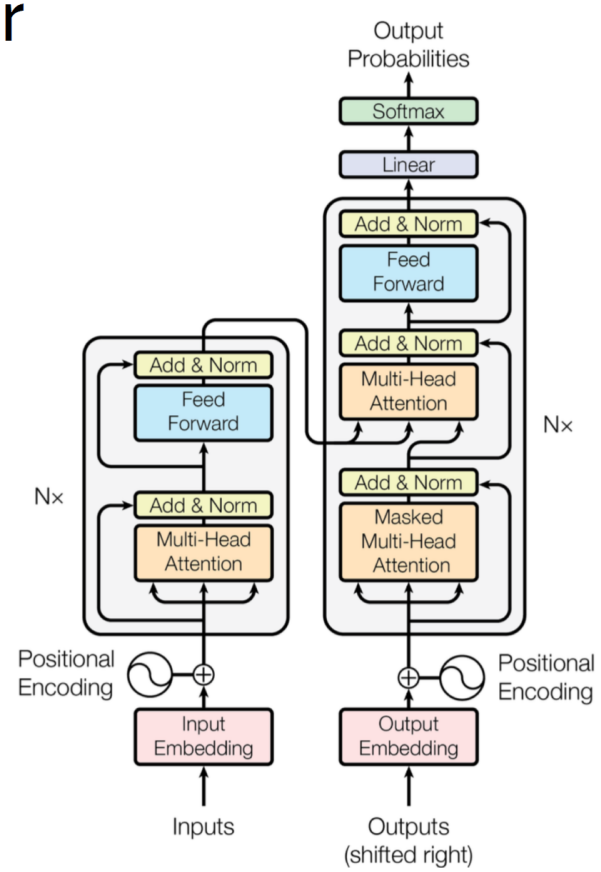
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_n)W_o,$$

$$\text{head}_i = \text{Attention}(QW_Q^i, KW_K^i, VW_V^i)$$



Transformer

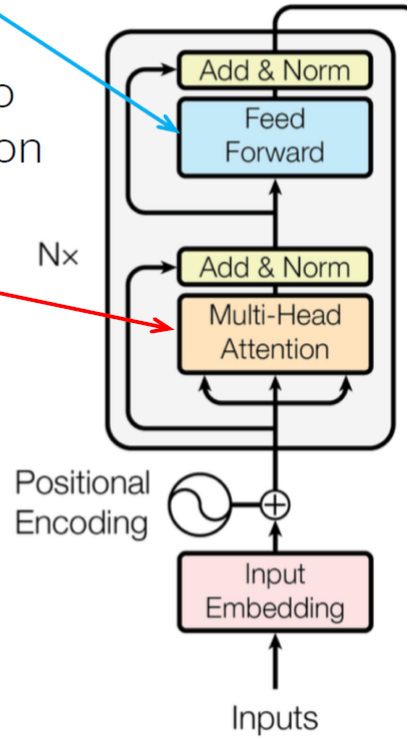
Transformer



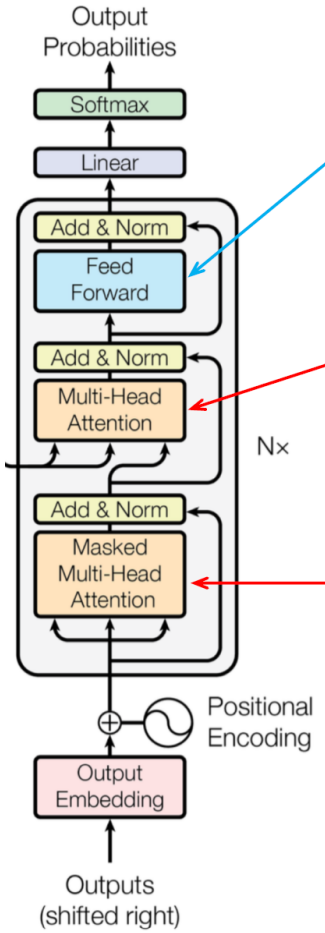
Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Encoder self-attention: tokens look at each other

queries, keys, values are computed from encoder states



Transformer



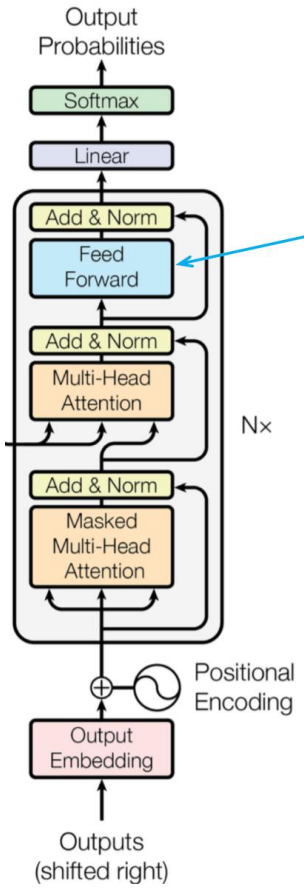
Feed-forward network: after taking information from other tokens, take a moment to think and process this information

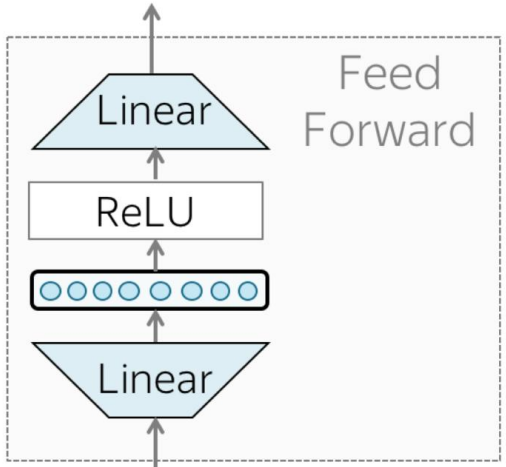
Decoder-encoder attention: target token looks at the source

queries – from decoder states; keys and values from encoder states

Decoder self-attention (masked): tokens look at the previous tokens

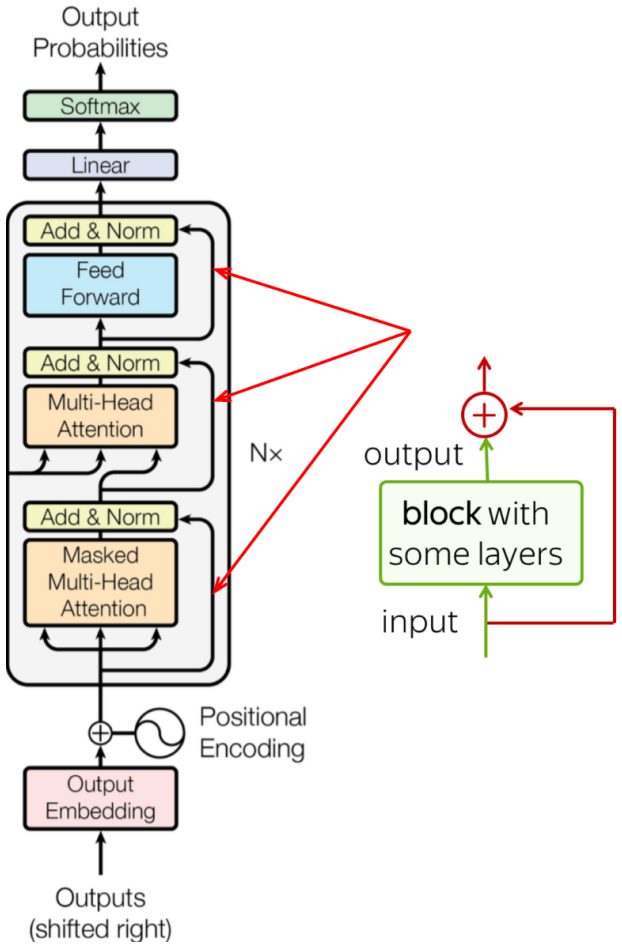
queries, keys, values are computed from decoder states



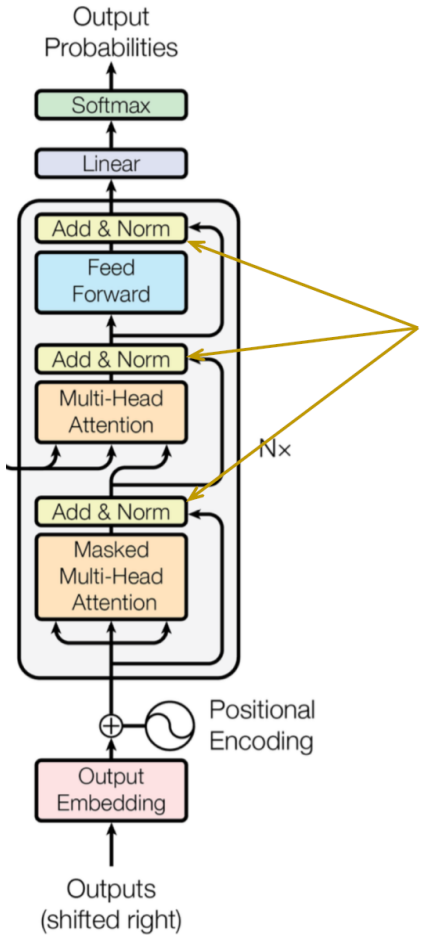


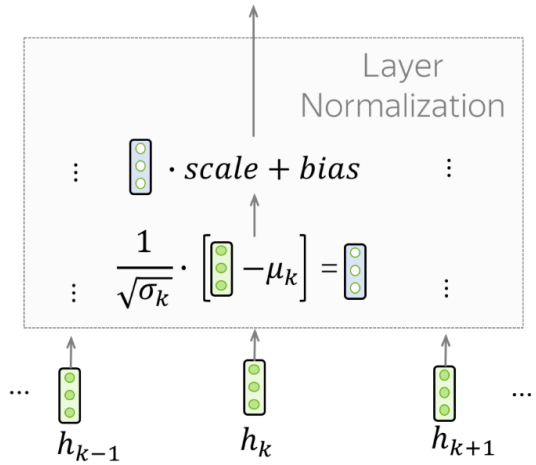
$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Transformer



Residual connection: add a block's input to its output





Позиционное кодирование

Проблема: без повторения или свертки модель ничего не знает о положении

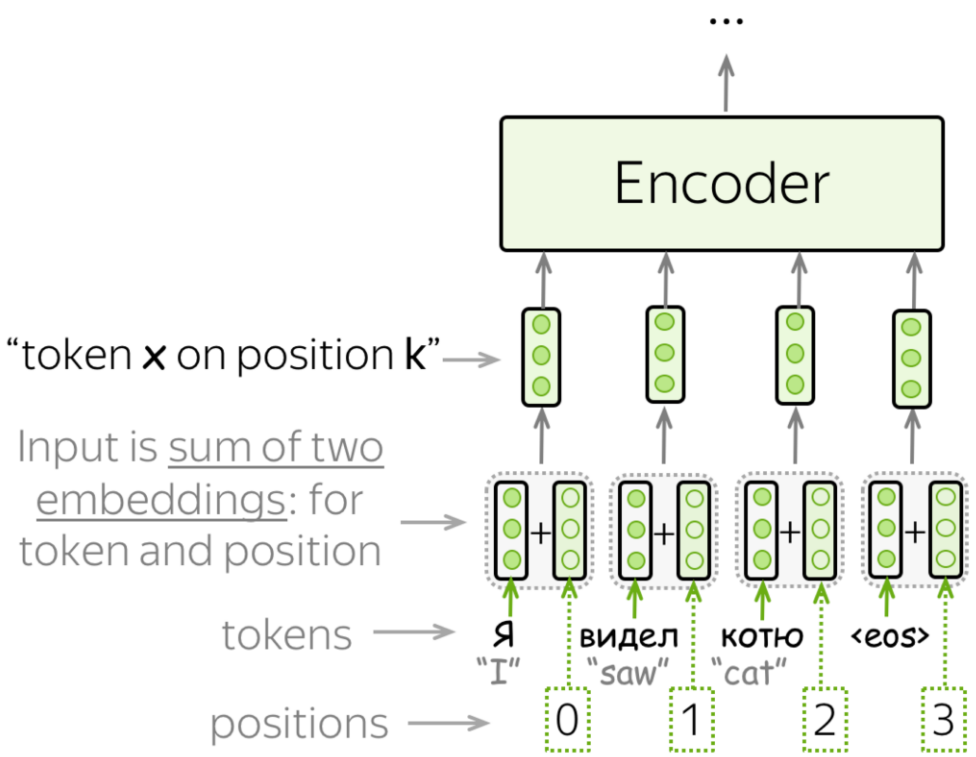
Решение: явно закодировать позицию

Фиксированные кодировки:

$$\mathrm{PE}_{pos,2i} = \sin(pos/10000^{2i/d_{model}}),$$

$$\mathrm{PE}_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}})$$

pos – позиция, i – размер

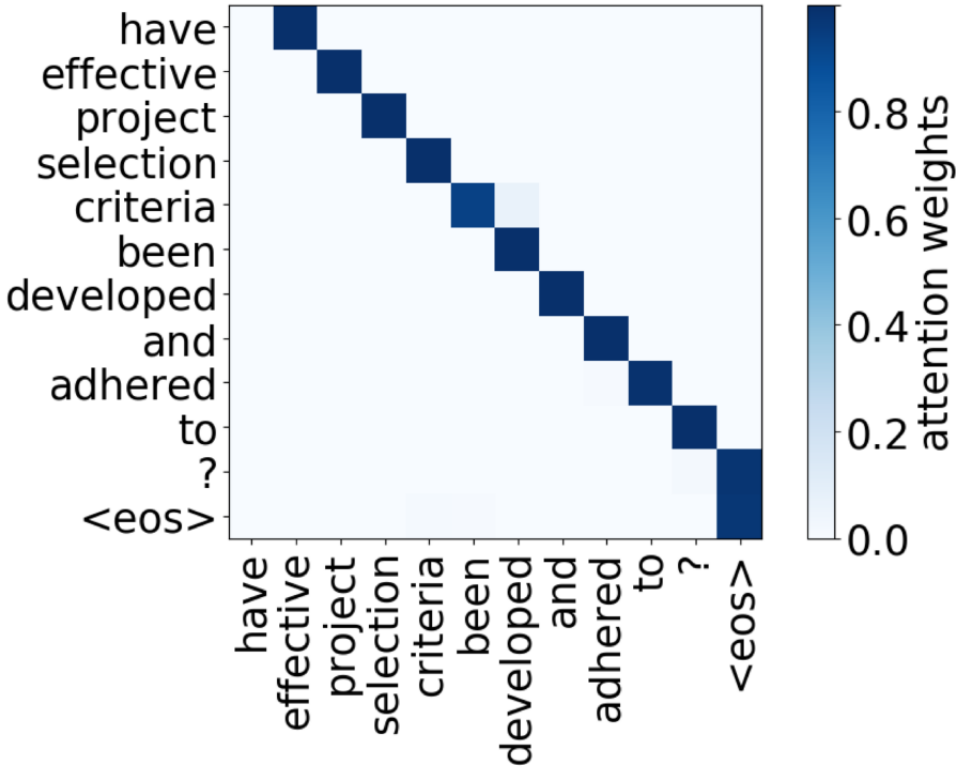


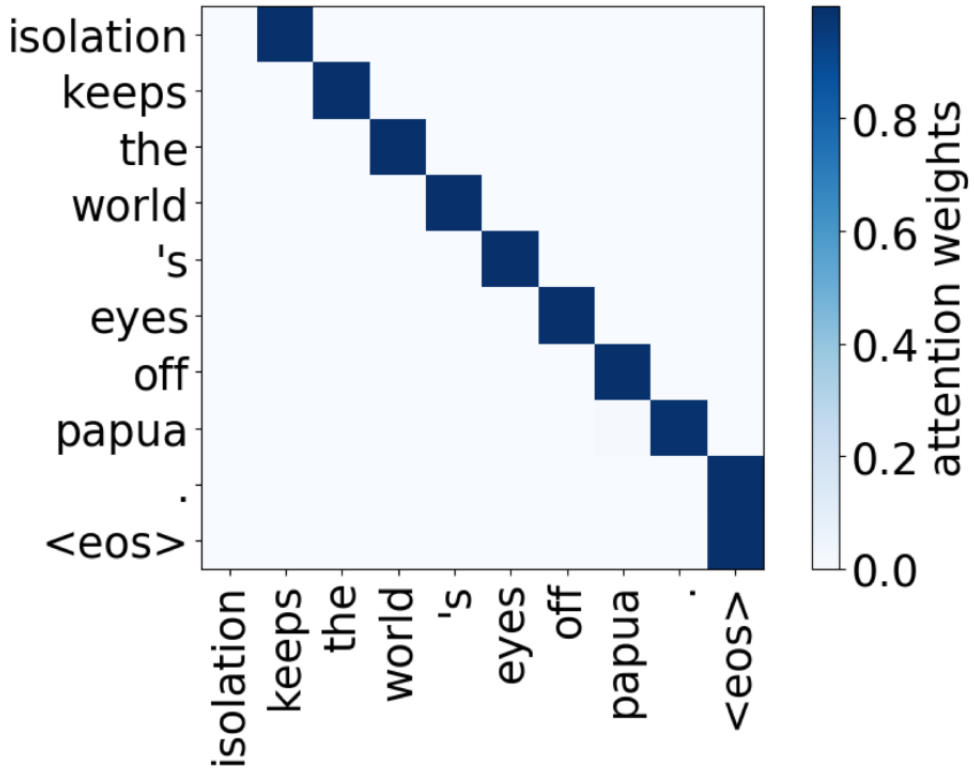
Анализ. Heads в Multi-Head Attention

Интерпретируемые роли «голов»:

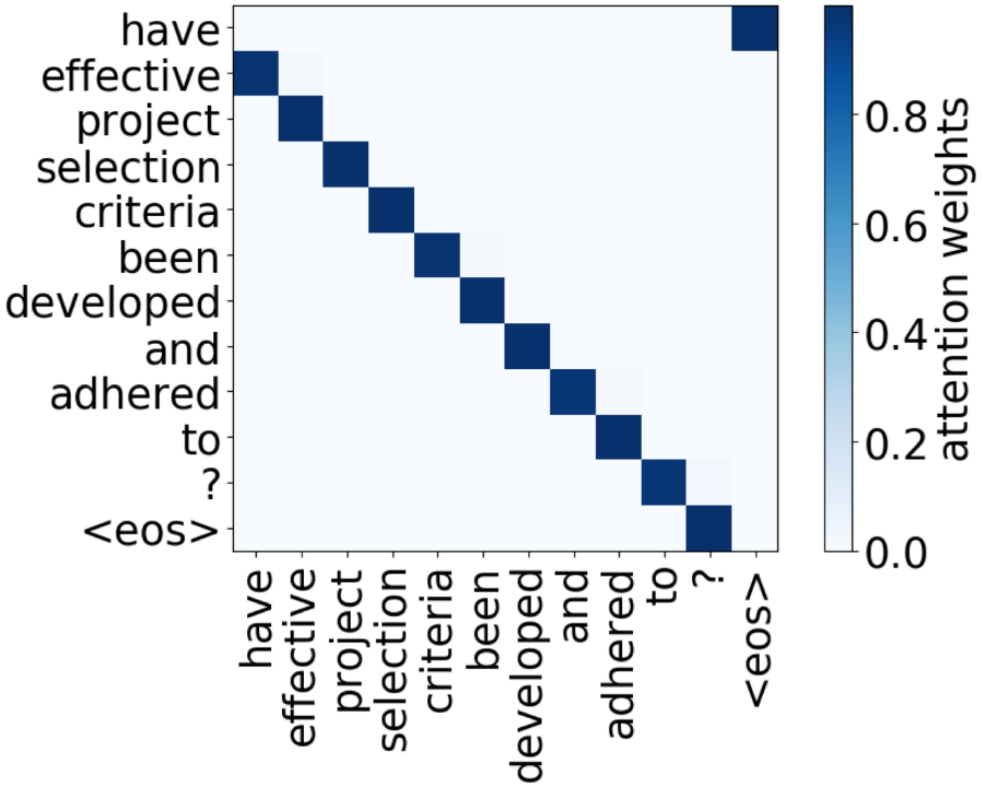
- Позиционные
- Синтаксические
- Внимание к редким токенам

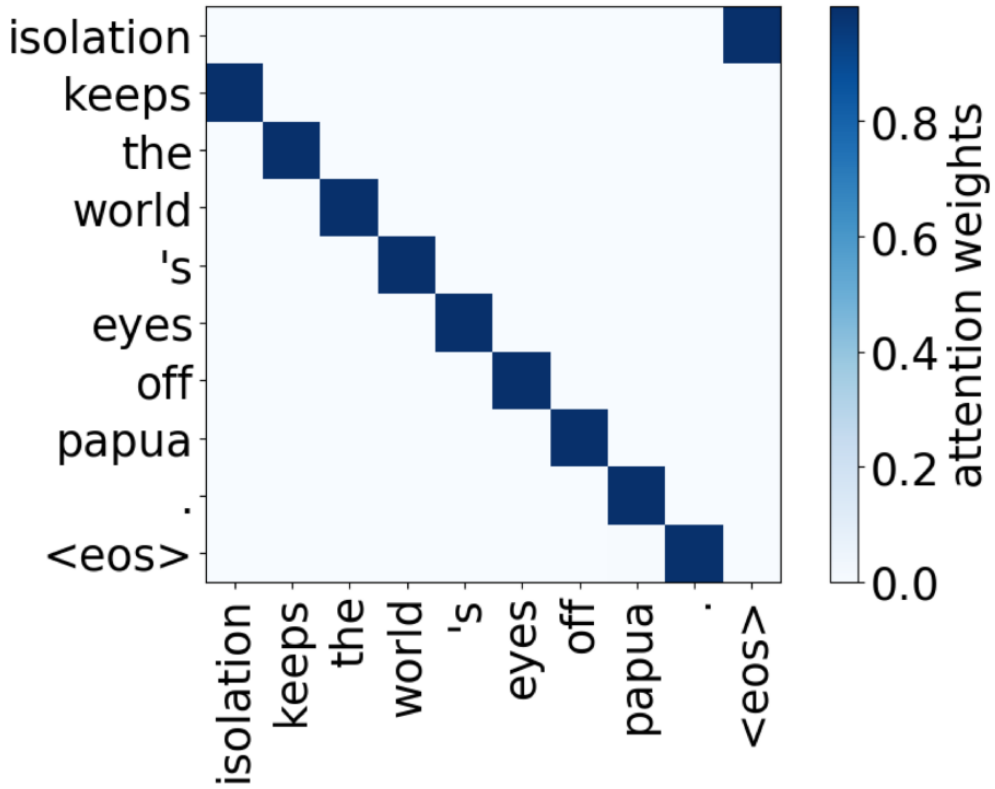
Позиционные головы: внимание к соседям



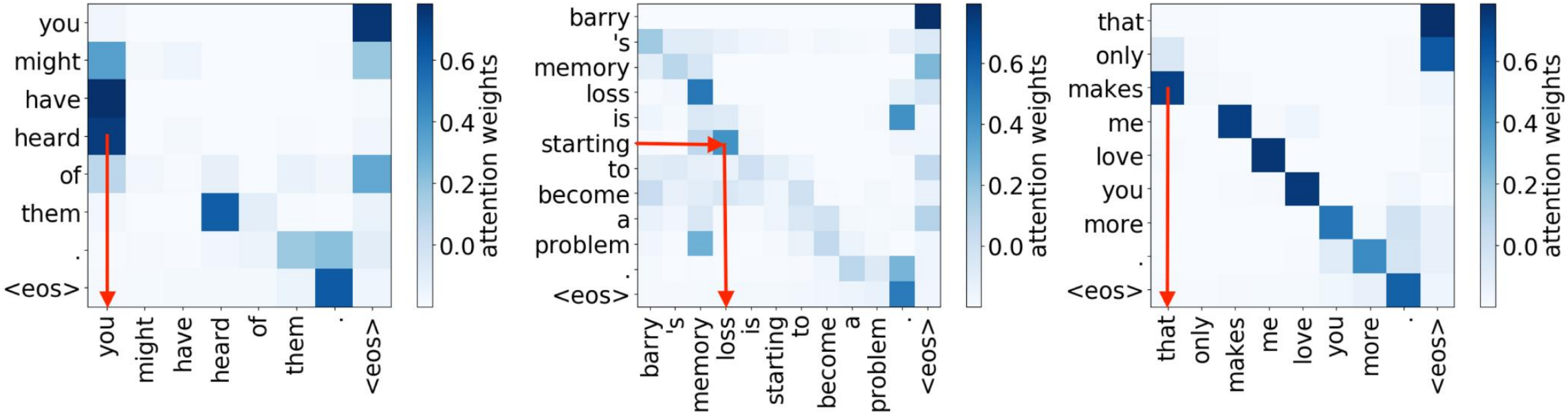


Позиционные головы: внимание к соседям



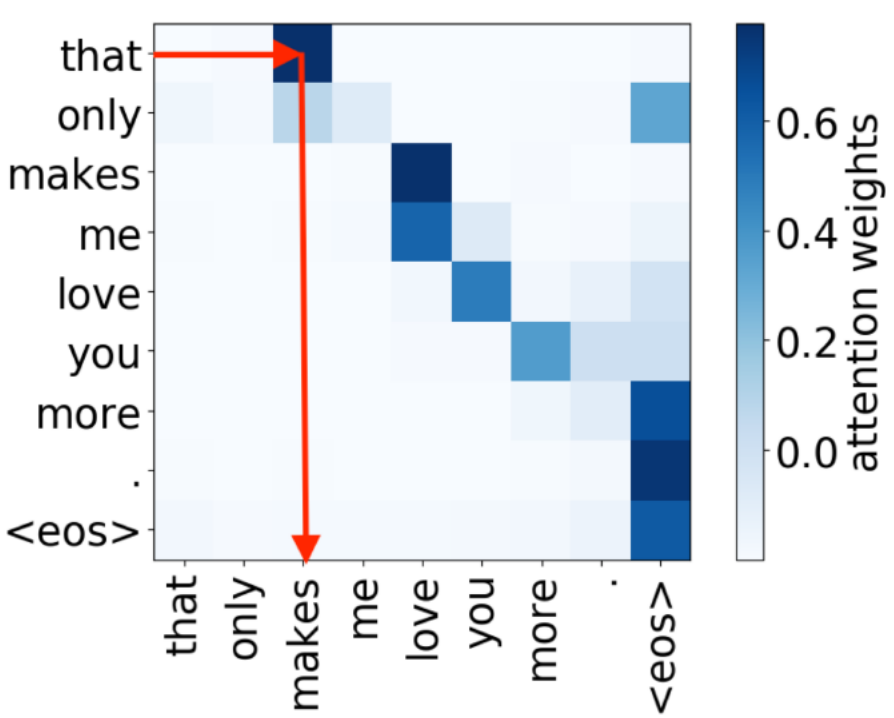


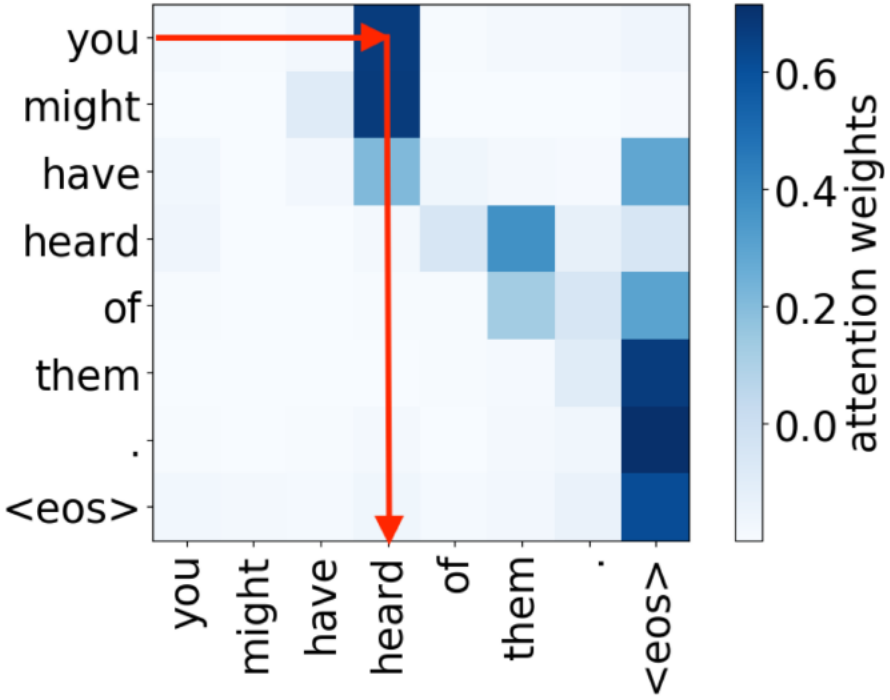
Синтаксические заголовки: отслеживание зависимостей



глагол->существительное

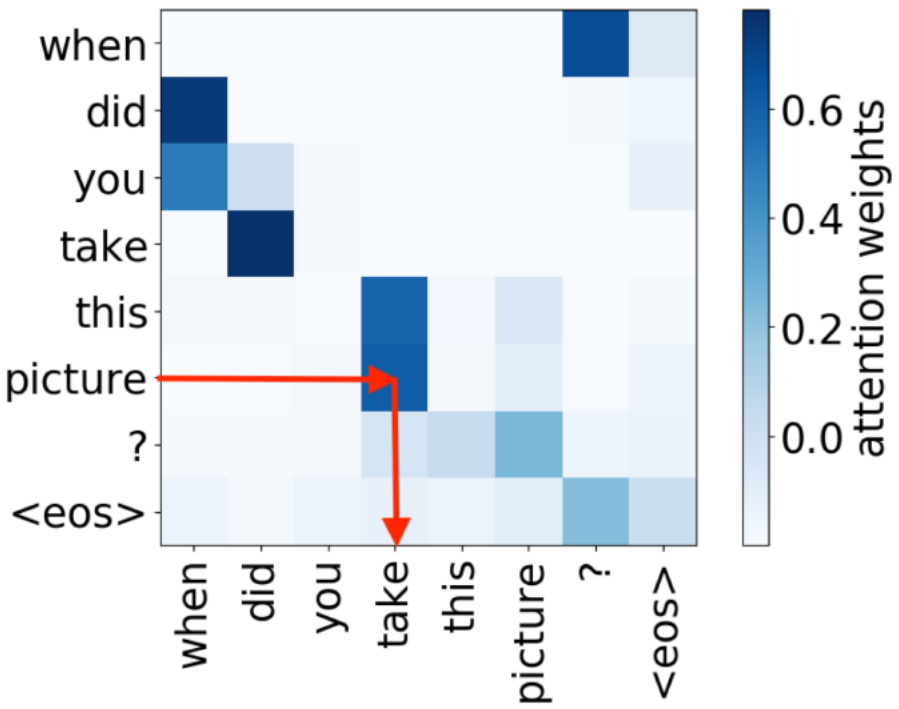
Синтаксические заголовки: отслеживание зависимостей

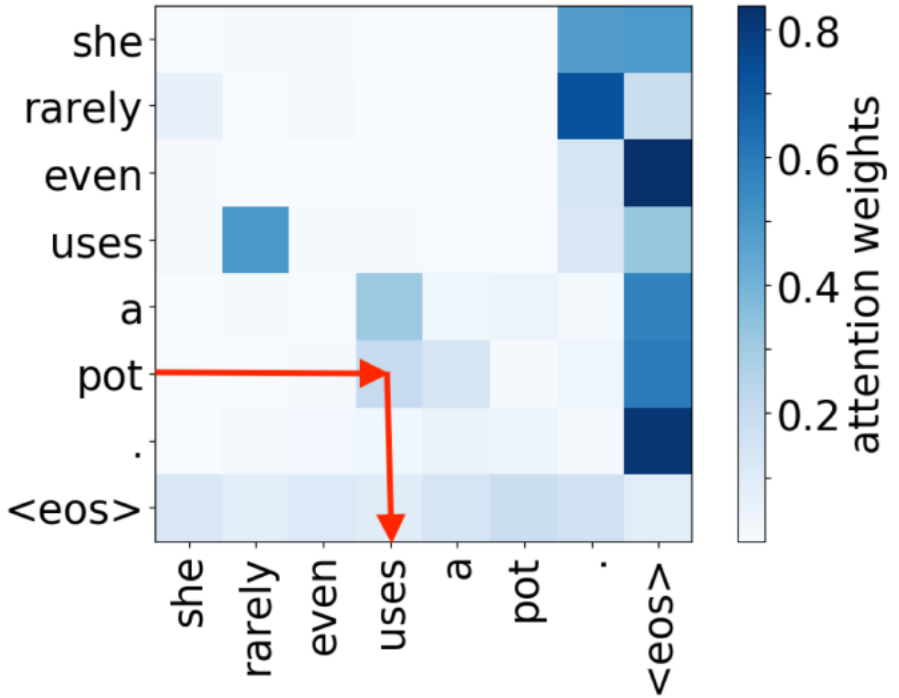




Существительное->глагол

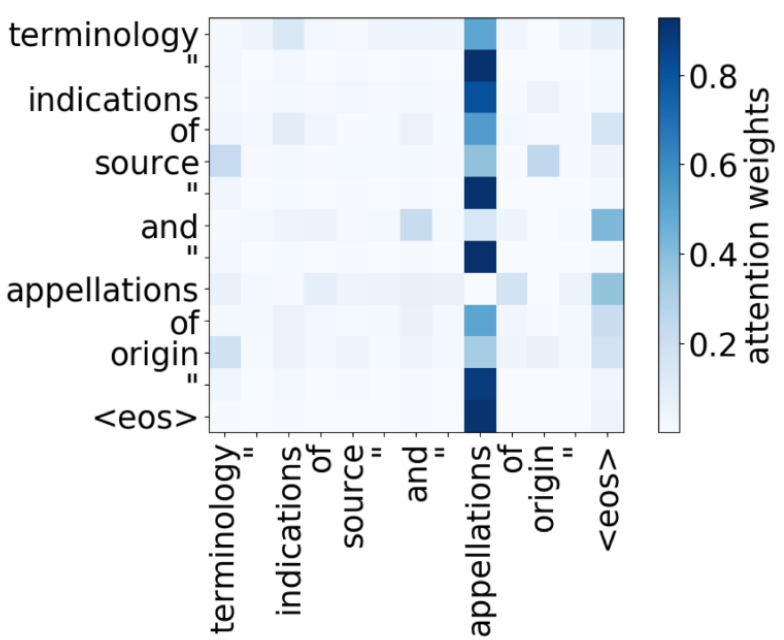
Синтаксические заголовки: отслеживание зависимостей

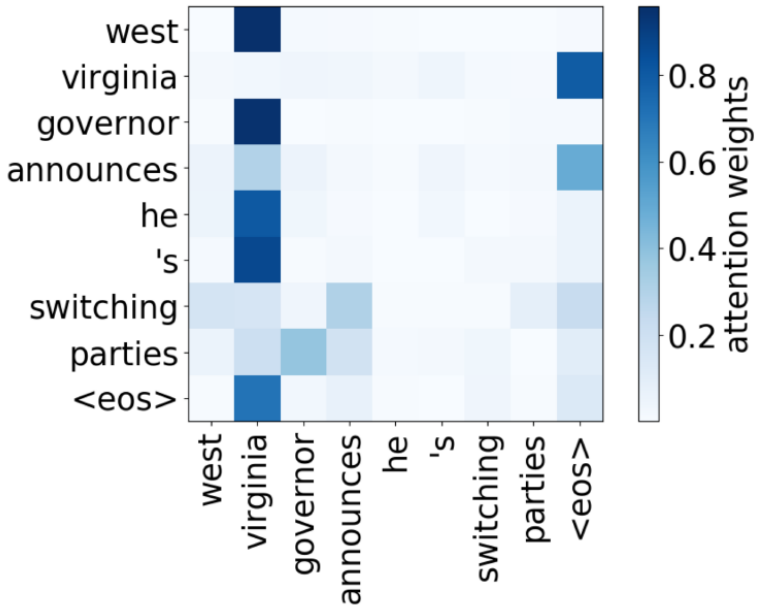


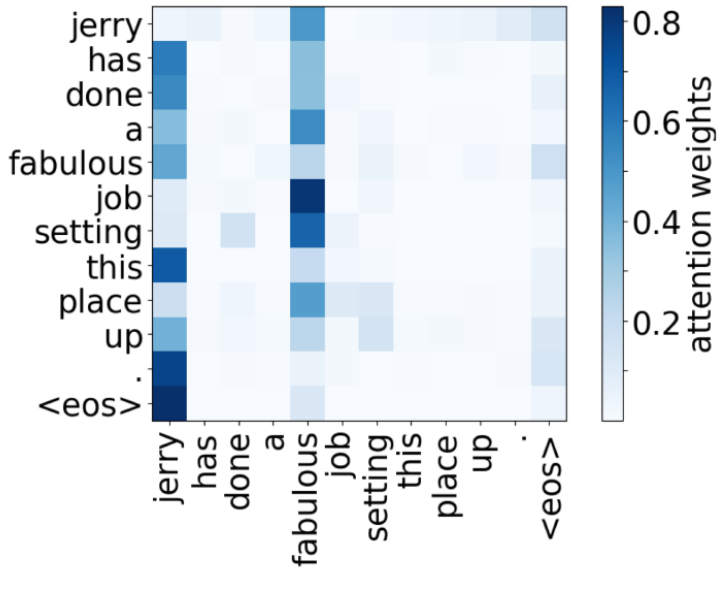


объект->глагол

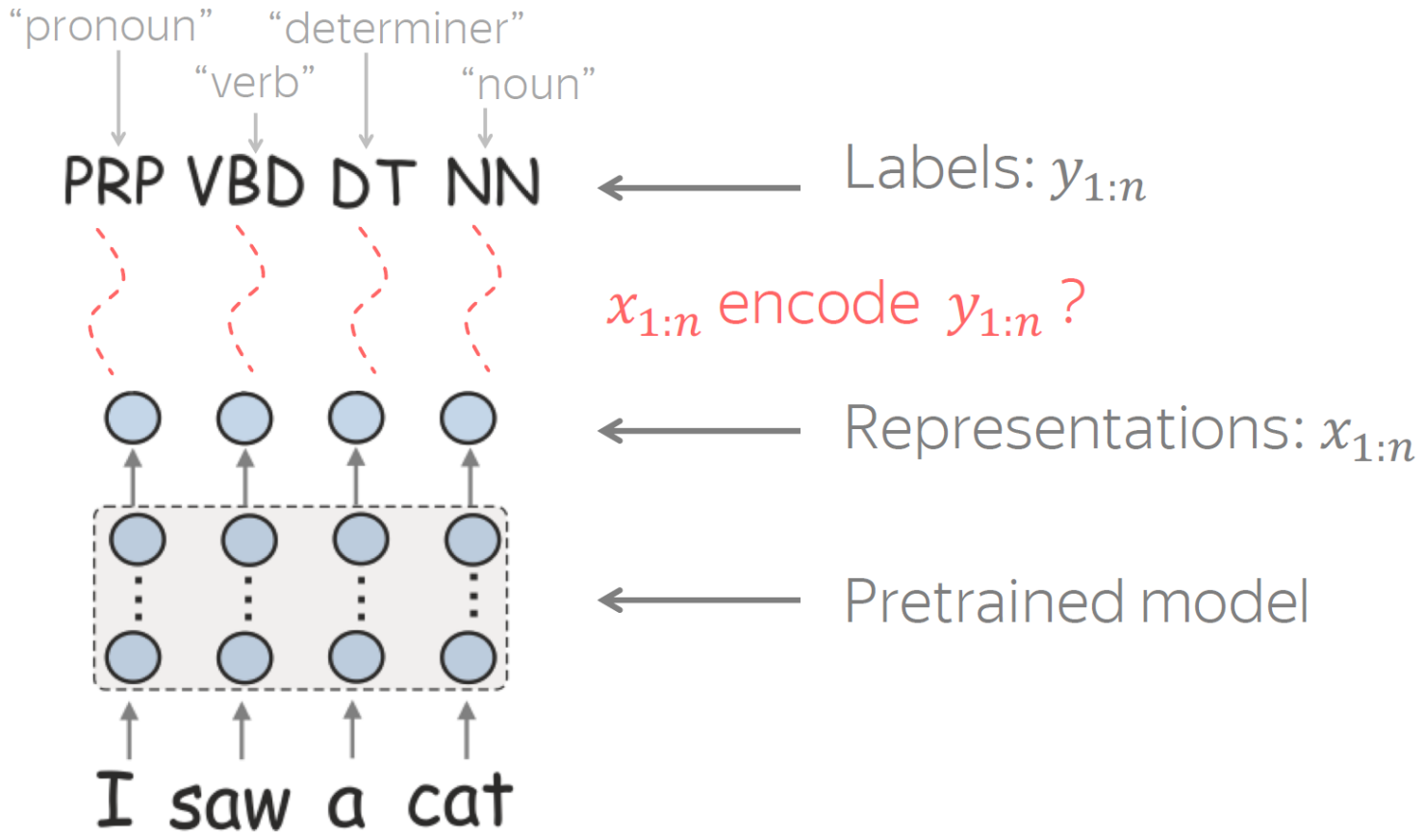
Внимание редким токенам



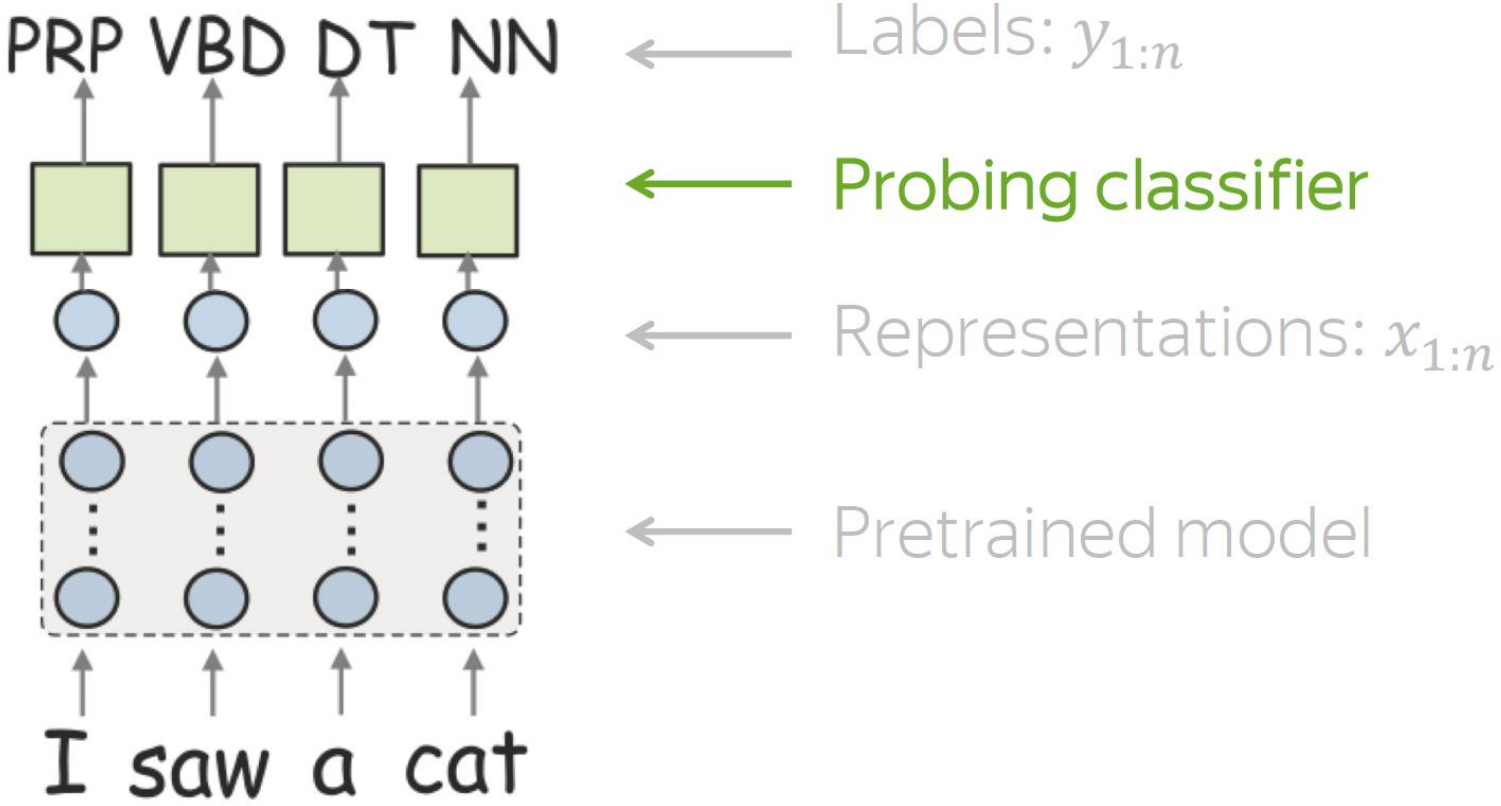




Как понять, отражает ли модель лингвистическое свойство?



Standard Probing: обучение классификатора, использование его точности



Что модели узнают о морфологии?