

Обработка естественного языка: Attention, Transformer

ЧТО ТАКОЕ ВНИМАНИЕ?

ВНИМАНИЕ

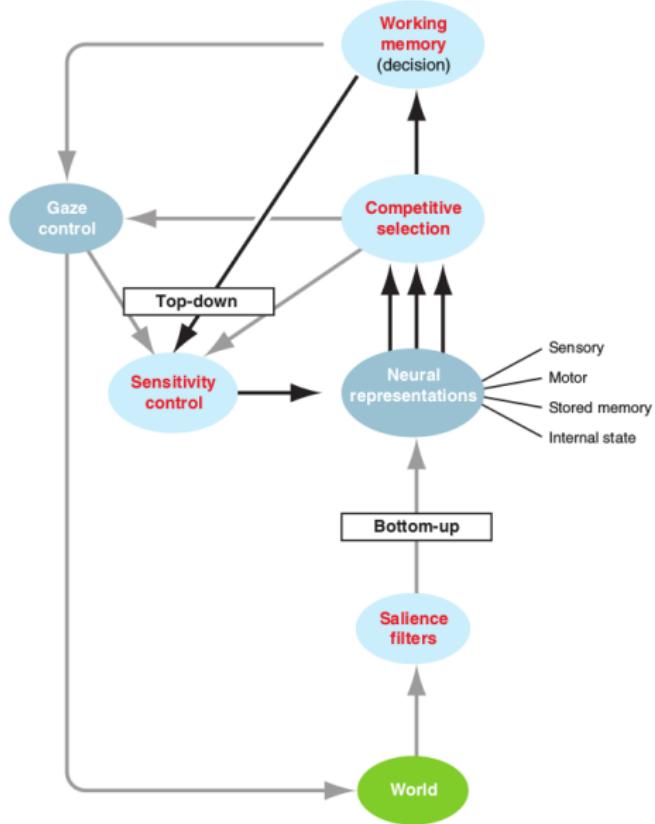
- Вы же сейчас внимательно меня слушаете, правильно?
- А что это значит?..
- Изображение с сетчатки тоже проходит через CNN*, но потом мы часть его замечаем, а часть не особенно. Что это значит?
- Оказывается, что это довольно сложный вопрос.
- А.Р. Лурия: внимание, память и активация коры.

*сравнение условно

ВНИМАНИЕ

- Дело во взаимодействии с рабочей памятью (Knudsen, 2007):

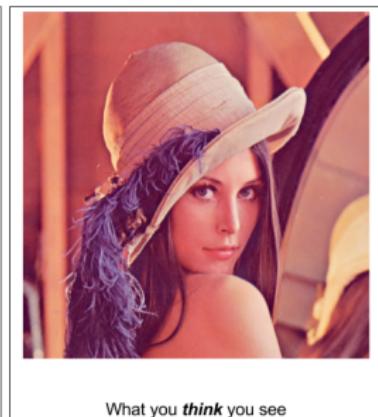
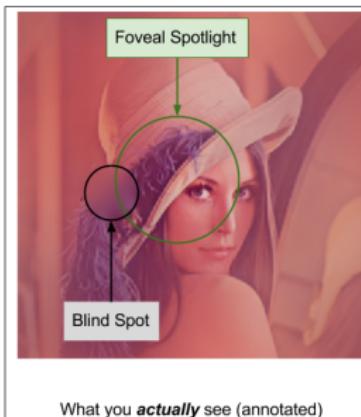
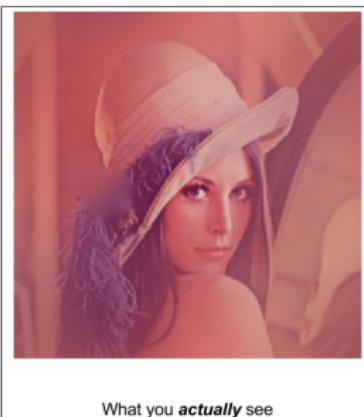
Работа по когнитивной нейробиологии. Рабочая память (working memory) — это когнитивная система с ограниченной емкостью, которая отвечает за временное удержание и обработку информации. Она важна для выполнения сложных когнитивных задач, таких как понимание языка, обучение и рассуждение. Кнудсен в своем исследовании рассматривает, как внимание управляет рабочей памятью, фокусируя ресурсы на информации, необходимой для текущей задачи, и подавляя менее значимую информацию.



- Как выбрать из очень большого объема поступающих данных именно то, что нужно делать прямо сейчас?
- Как научить сверточную сеть «концентрироваться» на нужной части изображения при распознавании объектов, а рекуррентную сеть — на релевантной части предложения во время машинного перевода?

ВНИМАНИЕ

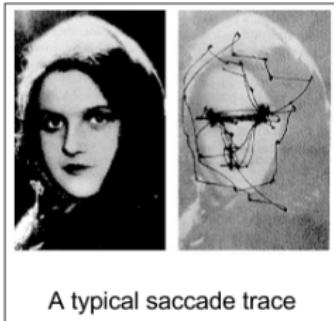
- Как это реализовать в нейронной сети? Особенno «сознательное» внимание.
- Но и «бессознательное» тоже; например, с картинками: мы же на самом деле мало чего видим в каждый момент времени.
- Центральная ямка сетчатки (fovea):



- Simons, Chabris, 1999: <https://www.youtube.com/watch?v=vJG698U2Mvo>

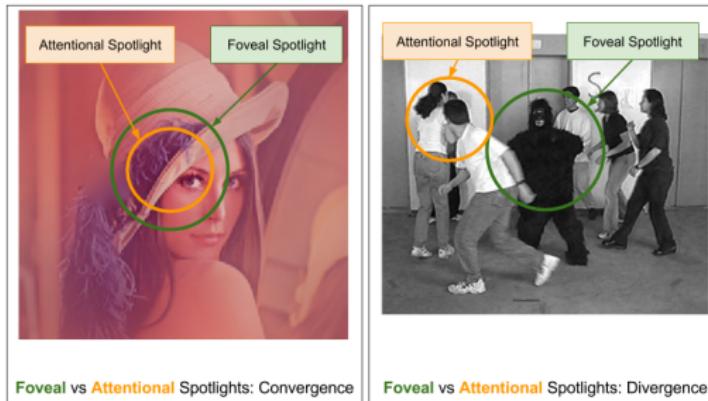
ВНИМАНИЕ

- Мы делаем саккады:



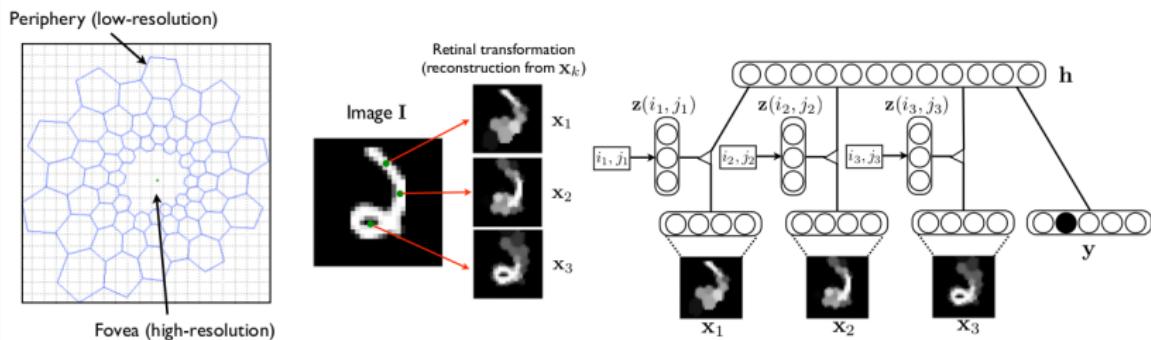
Саккады — это быстрые и резкие движения глаз, которые позволяют перемещать точку фокусировки с одного объекта на другой. В отличие от плавных движений, саккады происходят почти мгновенно и обеспечивают нашу способность быстро сканировать окружающее пространство, фиксируясь на наиболее значимых деталях

- Причём это тоже не всегда помогает:



FOVEAL GLIMPSES

- В нейросети мы тоже хотим осознанно понимать, «на что» смотреть.
- Одна из первых работ (Larochelle, Hinton, 2010):



- Пытаются моделировать положения фиксаций и строить последовательность при помощи RBM.
- Последовательность – значит...

Foveal glimpses — это концепция, вдохновлённая биологическим зрением, при которой фокус внимания сосредотачивается на небольших, но значимых областях изображения
RBM (Restricted Boltzmann Machine)

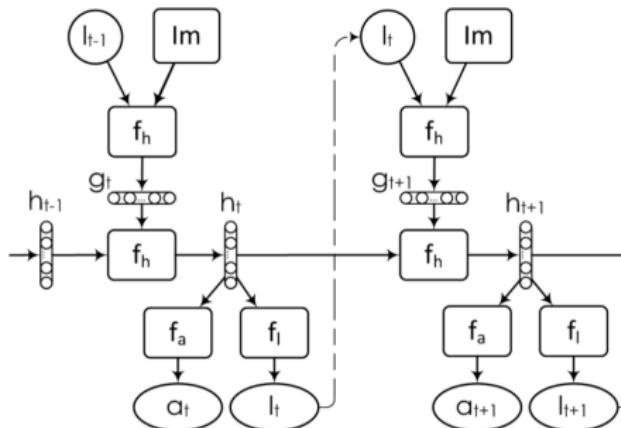
РЕКУРРЕНТНЫЕ МОДЕЛИ ЗРИТЕЛЬНОГО ВНИМАНИЯ

RECURRENT VISUAL ATTENTION

- По-настоящему всё появилось в (Mnih et al., 2014), «Recurrent Models of Visual Attention»:
 - из предыдущего \mathbf{h}_{t-1} и положения l_t для нового «взгляда» f_g делает \mathbf{g}_t , вход для шага t ;
 - из \mathbf{h}_{t-1} и \mathbf{g}_t функцией f_h получается \mathbf{h}_t ;
 - из него – «действие» $a_t = f_a(\mathbf{h}_t)$ и положение следующего «взгляда» $l_{t+1} = f_l(\mathbf{h}_t)$.

Модель обучается выбирать области, на которых следует сфокусироваться. Эти области называются "glimpses" — фрагменты, представляющие собой выборку ограниченных областей, которые анализируются для распознавания или классификации.

На каждом шаге модель выбирает следующую область для просмотра с помощью рекуррентной нейронной сети (RNN), настраивая свое внимание по мере поступления новой информации.



Модель обучает стратегию выбора следующей

области для внимания с помощью подкрепления, получая обратную связь о том, насколько полезным был выбранный “взгляд”.

Подход с управляемым вниманием улучшает точность распознавания, снижая при этом вычислительные затраты по сравнению с традиционными свёрточными нейронными сетями, которые анализируют изображения целиком.

RECURRENT VISUAL ATTENTION

- Давайте разберёмся в модели формально:

$$\mathbf{g}_t = f_g(\mathbf{x}_t, \mathbf{l}_{t-1}; \theta_g),$$

$$\mathbf{h}_t = f_h(\mathbf{h}_{t-1}, \mathbf{g}_t; \theta_h),$$

$$\mathbf{l}_t \sim p(\cdot \mid f_l(\mathbf{h}_t; \theta_l)),$$

$$a_t \sim p(\cdot \mid f_a(\mathbf{h}_t; \theta_a)).$$

- После очередного действия получается новое наблюдение \mathbf{x}_{t+1} и награда r_t , которая будет скорее всего в конце, после всех шагов, за правильную классификацию.
- Что это напоминает?..

RECURRENT VISUAL ATTENTION

- ...о да, это reinforcement learning!
- Выучить надо стохастическую стратегию $\pi((\mathbf{l}_t, a_t) \mid \mathbf{s}_{1:t}; \theta)$, которая по истории будет выдавать следующее действие.
- У нас π задаётся через RNN, а оптимизировать надо

$$J(\theta) = \mathbb{E}_{p(\mathbf{s}_{1:T}; \theta)} [R] = \mathbb{E}_{p(\mathbf{s}_{1:T}; \theta)} \left[\sum_{t=1}^T r_t \right].$$

- Выглядит очень сложно – ожидание по последовательностям действий, т.е. по пространству большой размерности.
- Но есть методы – давайте сделаем preview тому, что потом будет в reinforcement learning.

RECURRENT VISUAL ATTENTION

- (Williams, 1992): алгоритм REINFORCE, в котором доказывается и используется выборочная оценка этого ожидания. Давайте выведем, а потом применим к нашему случаю...
- Нам надо оптимизировать

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^T r_t(s_t, a_t) \right] \approx \frac{1}{M} \sum_{i=1}^M \sum_{t=1}^T r(s_t^{(i)}, a_t^{(i)}),$$

где мы взяли M примеров траекторий τ .

- Определим $r(\tau) = \sum_t r(s_t, a_t)$. Тогда

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\tau,$$

т.е. тот самый страшный интеграл по траекториям.

- Но оказывается, что можно продифференцировать по θ ...

- Продифференцируем по θ :

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \int \nabla_{\theta} \pi_{\theta}(\tau) r(\tau) d\tau \\&= \int \pi_{\theta}(\tau) \frac{\nabla_{\theta} \pi_{\theta}(\tau)}{\pi_{\theta}(\tau)} r(\tau) d\tau \\&= \int \pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau) d\tau \\&= \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)] \\&\approx \frac{1}{M} \sum_{i=1}^M \nabla_{\theta} \log \pi_{\theta}(\tau^{(i)}) r^{(i)}(\tau),\end{aligned}$$

если приблизить выборкой; но сначала давайте ещё посмотрим на $\pi_{\theta}(\tau)$...

RECURRENT VISUAL ATTENTION

- Вероятность определяется как

$$\pi_\theta(\tau) = p(s_1) \prod_{t=1}^T \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t).$$

- Берём логарифм, а потом заметим, что от θ зависят только действия:

$$\begin{aligned}\nabla_\theta \log \pi_\theta(\tau) &= \\ &= \nabla_\theta \left(\log p(s_1) + \sum_{t=1}^T \log \pi_\theta(a_t | s_t) + \sum_{t=1}^T \log p(s_{t+1} | s_t, a_t) \right) = \\ &= \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t | s_t).\end{aligned}$$

RECURRENT VISUAL ATTENTION

- Итого получается вполне tractable градиент:

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[r(\tau) \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right] \\ &\approx \frac{1}{M} \sum_{i=1}^M r(\tau^{(i)}) \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}).\end{aligned}$$

- У нас тоже можно считать, что награда R даётся целиком:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{M} \sum_{i=1}^M \sum_{t=1}^T \nabla_{\theta} \log \pi \left(l_t^{(i)}, a_t^{(i)} | s_{1:t}^{(i)}; \theta \right) R^{(i)}.$$

- Т.е. надо уметь считать $\log \pi \left(l_t^{(i)}, a_t^{(i)} | s_{1:t}^{(i)}; \theta \right)$, но в случае RNN это просто градиент сети, который можно посчитать через backpropagation.
- Ещё можно сделать частично supervised loss на последнем шаге, где мы знаем классификацию.

RECURRENT VISUAL ATTENTION

- Результаты:



(a) Translated MNIST inputs.



(b) Cluttered Translated MNIST inputs.

(a) 28x28 MNIST

Model	Error
FC, 2 layers (256 hiddens each)	1.69%
Convolutional, 2 layers	1.21%
RAM, 2 glimpses, 8×8 , 1 scale	3.79%
RAM, 3 glimpses, 8×8 , 1 scale	1.51%
RAM, 4 glimpses, 8×8 , 1 scale	1.54%
RAM, 5 glimpses, 8×8 , 1 scale	1.34%
RAM, 6 glimpses, 8×8 , 1 scale	1.12%
RAM, 7 glimpses, 8×8 , 1 scale	1.07%

(b) 60x60 Translated MNIST

Model	Error
FC, 2 layers (64 hiddens each)	6.42%
FC, 2 layers (256 hiddens each)	2.63%
Convolutional, 2 layers	1.62%
RAM, 4 glimpses, 12×12 , 3 scales	1.54%
RAM, 6 glimpses, 12×12 , 3 scales	1.22%
RAM, 8 glimpses, 12×12 , 3 scales	1.2%

RECURRENT VISUAL ATTENTION

- Результаты:

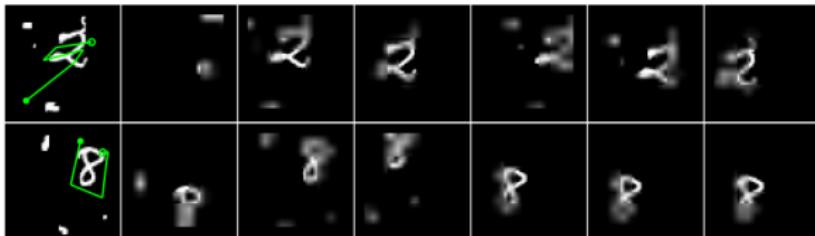
(a) 60x60 Cluttered Translated MNIST

Model	Error
FC, 2 layers (64 hiddens each)	28.58%
FC, 2 layers (256 hiddens each)	11.96%
Convolutional, 2 layers	8.09%
RAM, 4 glimpses, 12×12 , 3 scales	4.96%
RAM, 6 glimpses, 12×12 , 3 scales	4.08%
RAM, 8 glimpses, 12×12 , 3 scales	4.04%
RAM, 8 random glimpses	14.4%

(b) 100x100 Cluttered Translated MNIST

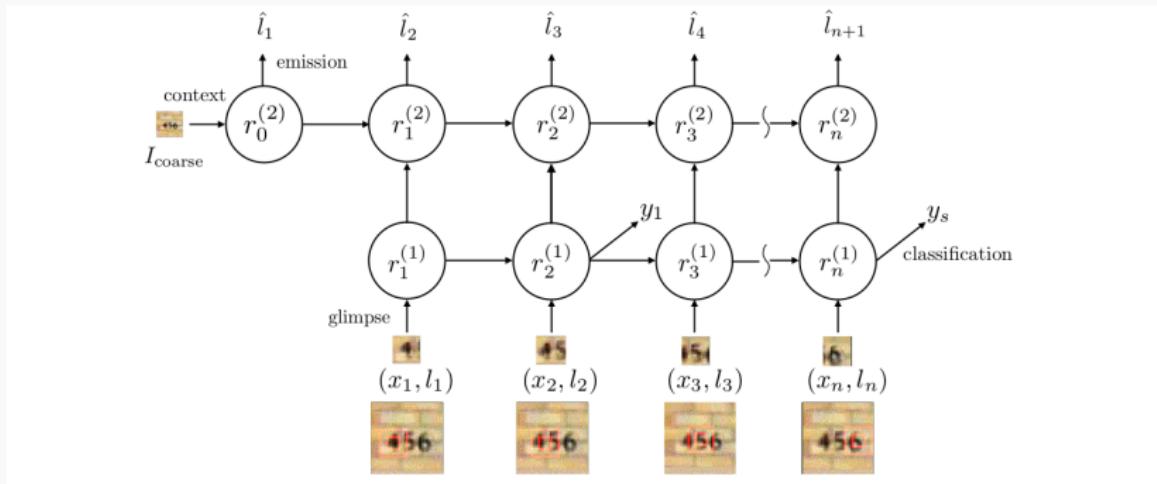
Model	Error
Convolutional, 2 layers	14.35%
RAM, 4 glimpses, 12×12 , 4 scales	9.41%
RAM, 6 glimpses, 12×12 , 4 scales	8.31%
RAM, 8 glimpses, 12×12 , 4 scales	8.11%
RAM, 8 random glimpses	28.4%

- А вот как внимание гуляет по картинке:



RECURRENT VISUAL ATTENTION

- В следующей работе (Ba et al., 2015) сделали глубокую модель:



- Кстати, обучали по-другому, вариационными методами. Как это?..

RECURRENT VISUAL ATTENTION

- Нам нужно классифицировать, т.е. $p(y | \mathbf{x}, \theta)$ максимизировать.
- Маргинализуем по положениям glimpses:

$$\log p(y | \mathbf{x}, \theta) = \log \sum_l p(l | \mathbf{x}, \theta) p(y | l, \mathbf{x}, \theta).$$

- Запишем вариационную нижнюю оценку свободной энергии (как её получить?):

$$\begin{aligned} \log \sum_l p(l | \mathbf{x}, \theta) p(y | l, \mathbf{x}, \theta) &\geq \sum_l p(l | \mathbf{x}, \theta) \log p(y, l | \mathbf{x}, \theta) + H[l] \\ &= \sum_l p(l | \mathbf{x}, \theta) \log p(y | l, \mathbf{x}, \theta). \end{aligned}$$

RECURRENT VISUAL ATTENTION

- И теперь можно брать производные:

$$\begin{aligned}\frac{\partial J}{\partial \theta} &= \sum_l p(l | \mathbf{x}, \theta) \frac{\partial \log p(y | l, \mathbf{x}, \theta)}{\partial \theta} + \sum_l \log p(y | l, \mathbf{x}, \theta) \frac{\partial p(l | \mathbf{x}, \theta)}{\partial \theta} \\ &= \sum_l p(l | \mathbf{x}, \theta) \left[\frac{\partial \log p(y | l, \mathbf{x}, \theta)}{\partial \theta} + \log p(y | l, \mathbf{x}, \theta) \frac{\partial \log p(l | \mathbf{x}, \theta)}{\partial \theta} \right].\end{aligned}$$

- А эту сумму уже будем приближать выборкой:

$$\frac{\partial J}{\partial \theta} \approx \frac{1}{M} \sum_{i=1}^M \left[\frac{\partial \log p(y | l^{(i)}, \mathbf{x}, \theta)}{\partial \theta} + \log p(y | l^{(i)}, \mathbf{x}, \theta) \frac{\partial \log p(l^{(i)} | \mathbf{x}, \theta)}{\partial \theta} \right],$$

где $l^{(i)} \sim p(l_n | \mathbf{x}, \theta) = \mathcal{N}(l_n | \hat{l}_n, \Sigma)$.

- И это уже алгоритм: сэмплируем glimpses, потом используем их в backpropagation.
- Как и в (Mnih et al., 2014), надо бы уменьшить дисперсию; для этого вычитают baseline, пока не будем углубляться.

RECURRENT VISUAL ATTENTION

- Получился интересный результат – мы увидели, что примерно один и тот же алгоритм может получиться с двух разных сторон:
 - из обучения с подкреплением через REINFORCE;
 - из вариационной оценки собственно целевой функции.
- Важный гиперпараметр – размер *glimpse*, т.е. как переводить единицы измерений в системе координат *glimpses* в пиксели.
- То же самое легко расширить на последовательную классификацию нескольких объектов – просто фиксированное число *glimpses* на объект, потом классифицируем, плюс терминальное действие в конце всего.

МАШИННЫЙ ПЕРЕВОД: ENCODER-DECODER И ВНИМАНИЕ

МАШИННЫЙ ПЕРЕВОД

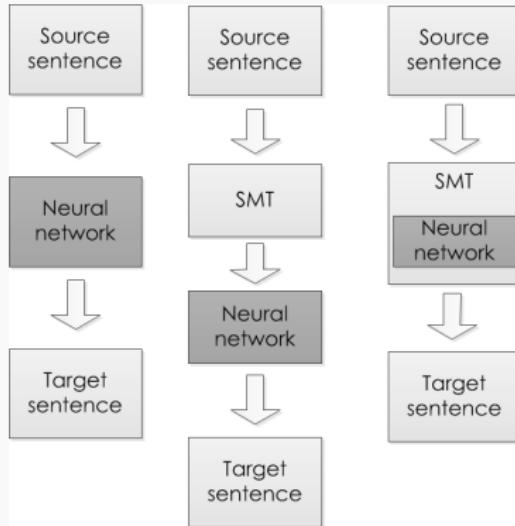
- Перевод – очень хорошая задача:
 - очевидно очень практическая;
 - очевидно очень высокоуровневая, требует понимания;
 - считается довольно неплохо квантифицируемой (BLEU, TER – хотя см. выше);
 - имеет большие доступные датасеты параллельных переводов.

Идея BLEU: Чем больше n-грамм совпадает с эталонным переводом, тем выше оценка BLEU

Идея TER: Чем меньше правок нужно, тем качественнее перевод

МАШИННЫЙ ПЕРЕВОД

- Статистический машинный перевод (statistical machine translation, SMT): моделируем условную вероятность $p(y | x)$ перевода y при условии исходного текста x .
- Классический SMT: моделируем $\log p(y | x)$ линейной комбинацией признаков, строим признаки.



МАШИННЫЙ ПЕРЕВОД

- Нам больше интересно моделирование sequence-to-sequence:
 - RNN естественным образом моделирует последовательность $X = (x_1, x_2, \dots, x_T)$ как $p(x_1), p(x_2 | x_1), \dots, p(x_T | x_{<T}) = p(x_T | x_{T-1}, \dots, x_1)$, и теперь $p(X)$ – это просто
$$p(X) = p(x_1)p(x_2 | x_1) \dots p(x_k | x_{<k}) \dots p(x_T | x_{<T});$$
 - так RNN и в языковых моделях используются;
 - предсказываем следующее слово на основе скрытого состояния и предыдущего слова;
- Как применить эту идею к переводу?

Sequence-to-sequence model

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 two tabs: "Text" (selected) and "Documents". The main area shows a translation from English to Russian. 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 also displayed in its phonetic transcription: "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 character count of "138/5000". On the far right, there is a "Send feedback" link.

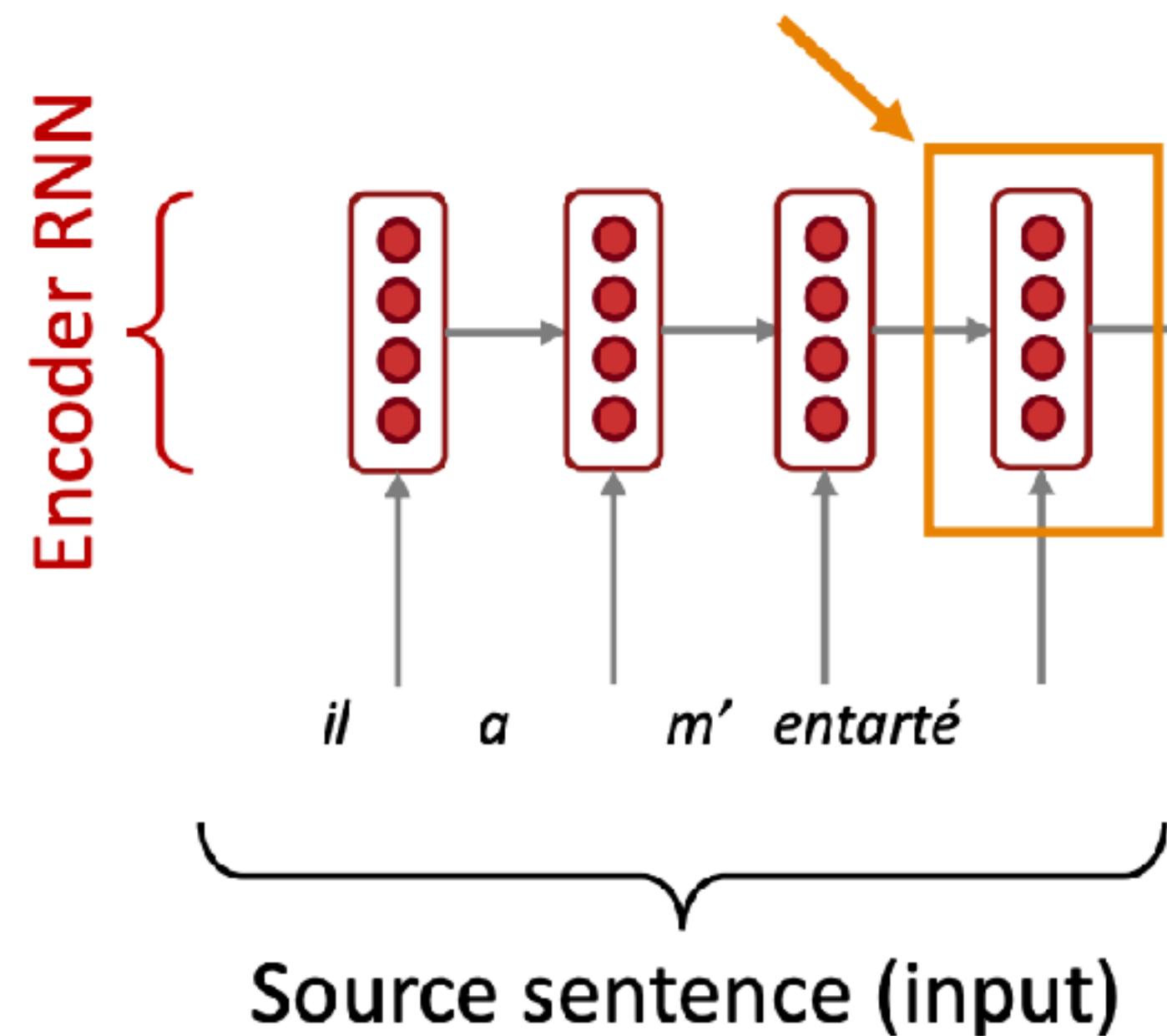
Авторегрессионная модель $p(y|x) = \prod_{i=1}^n p(y_i|y_1, \dots, y_{i-1}, \underline{x})$

x - текст на исходном языке (source)

y - перевод текста на другой язык (target)

Sequence-to-sequence model

Последний hidden state -
представление всего
исходного текста



Sequence-to-sequence (seq2seq) модель — это архитектура нейронных сетей, предназначенная для обработки последовательностей данных и создания другой последовательности на её основе. Она особенно полезна в задачах, где длина входных и выходных последовательностей может отличаться. Примеры таких задач: машинный перевод, текстовое резюмирование, генерация текста и обработка разговорной речи.

Seq2seq модель обычно состоит из двух главных компонентов:

- Энкодер (Encoder) — обрабатывает входную последовательность и сжимает её в фиксированное представление (вектор состояния), содержащее всю необходимую информацию о входных данных.
- В энкодере обычно используется рекуррентная нейронная сеть (RNN), такая как LSTM или GRU, которая накапливает состояние на каждом шаге последовательности и «запоминает» информацию о входных данных.
- Декодер (Decoder) — генерирует выходную последовательность, используя скрытое состояние энкодера. На каждом шаге декодер предсказывает следующее значение выходной последовательности, опираясь на своё текущее состояние и на выход предыдущего шага.
- Декодер также может быть рекуррентной сетью, которая предсказывает выходные данные последовательно.

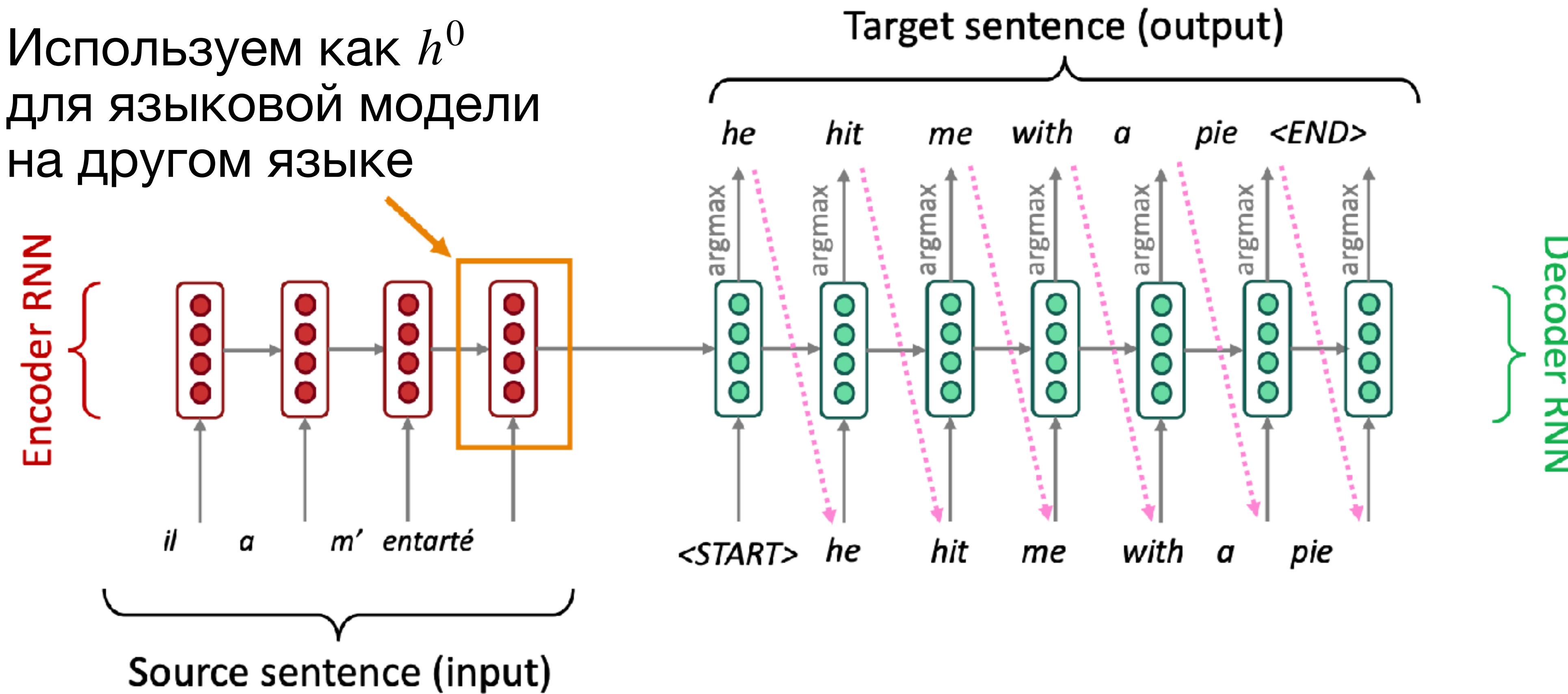
Принцип работы seq2seq модели

- Этап кодирования: энкодер обрабатывает последовательность и сохраняет накопленное состояние. Обычно используется последнее скрытое состояние энкодера в качестве входного состояния для декодера.
- Этап декодирования: декодер начинает с инициализированного состояния и генерирует выходный токен на каждом временном шаге, используя предсказанный токен на предыдущем шаге как вход. Этот процесс повторяется до тех пор, пока не будет достигнут конечный токен (например, <EOS>).

[Image credit](#)

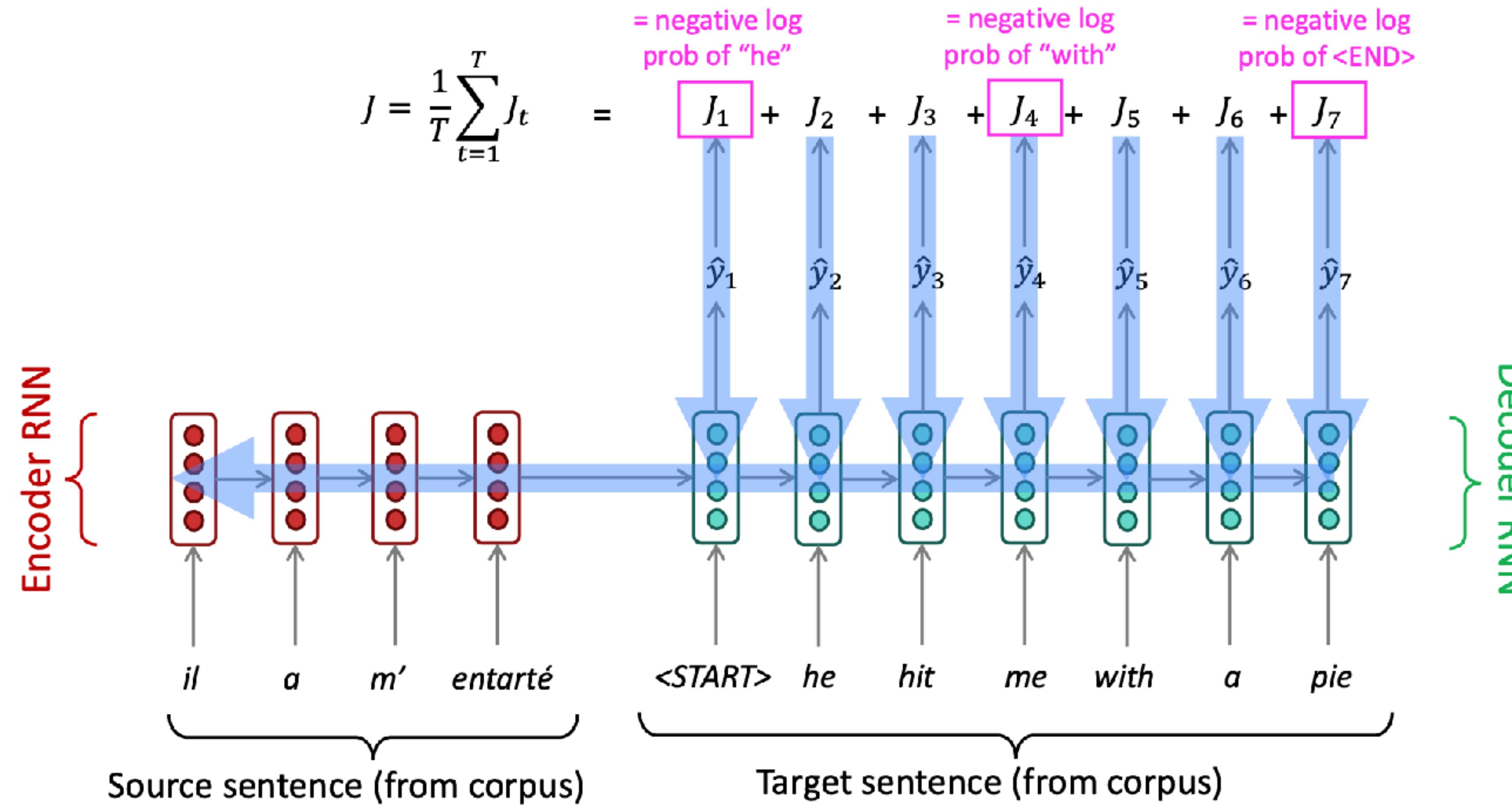
Sequence-to-sequence model

Используем как h^0
для языковой модели
на другом языке



Sequence-to-sequence model

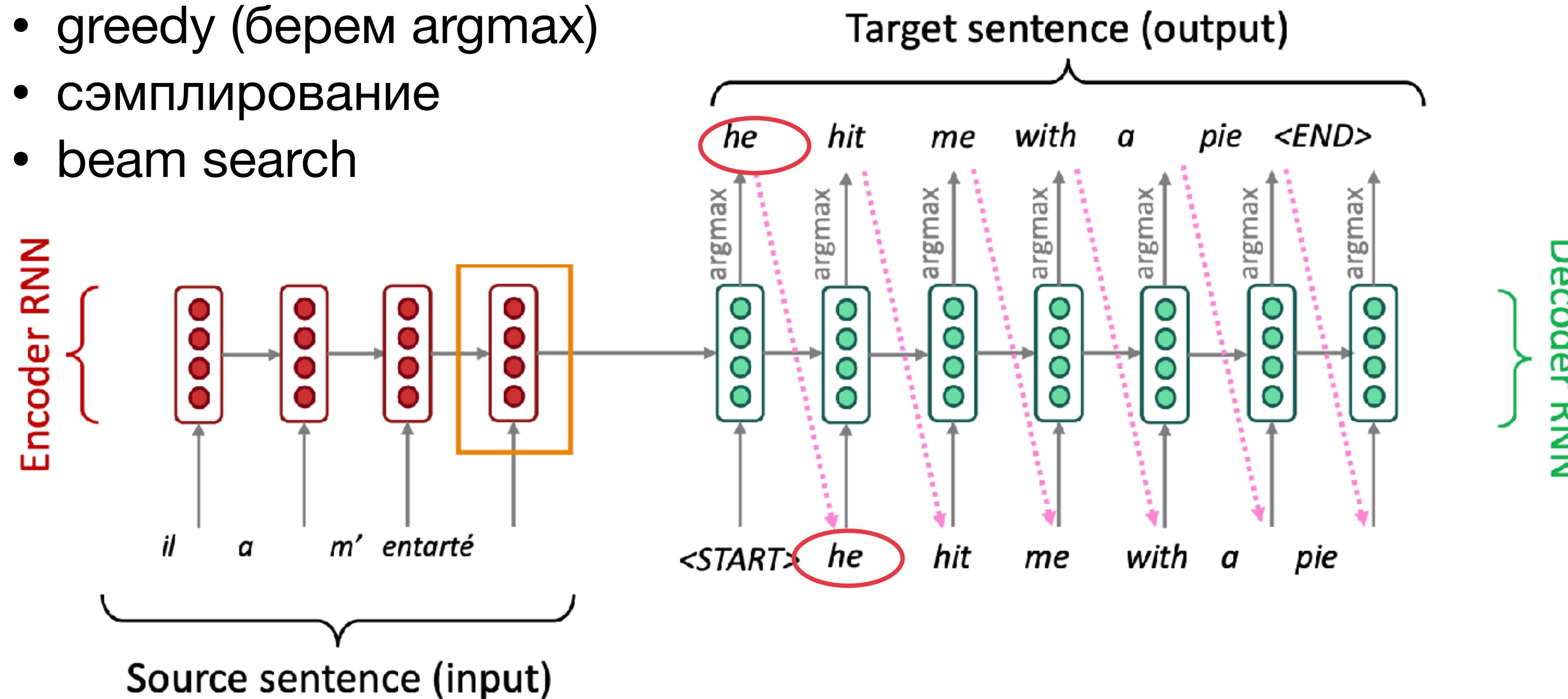
Обучение: нужен параллельный корпус



Sequence-to-sequence model

Генерация:

- greedy (берем argmax)
- сэмплирование
- beam search



Beam Search — это усовершенствованный вариант жадного поиска, который сохраняет не только одно, а несколько (beam width) лучших промежуточных решений на каждом шаге, что позволяет искать более широкое пространство решений и находить более оптимальные результаты. Инициализация: Начинается с начального состояния (например, стартового токена) и задается фиксированная ширина луча (beam width), которая определяет количество лучших решений, которые будут отслеживаться на каждом шаге. Генерация кандидатов: Оценка кандидатов: Каждый из сгенерированных кандидатов оценивается с использованием модели (например, нейронной сети), чтобы получить вероятности для каждого токена.

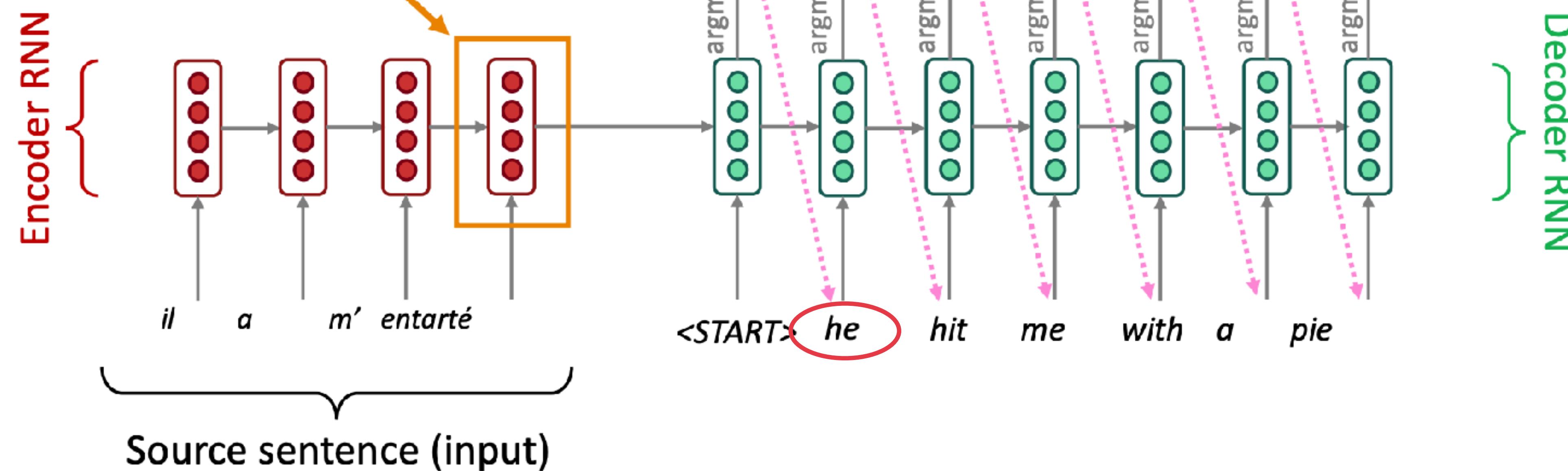
Сортировка и отбор: Из всех возможных токенов на текущем шаге выбираются топ- n (где n — ширина луча) кандидатов с наивысшими вероятностями. Эти кандидаты затем комбинируются с предыдущими токенами для формирования новых состояний для следующего временного шага. Повторение: Шаги 2–4 повторяются до тех пор, пока не будет достигнут токен окончания последовательности (например, <EOS>), или пока не будет достигнута максимальная длина последовательности. Выбор лучшего результата: По завершении поиска выбирается наиболее вероятная последовательность из сохраненных кандидатов.

[Image credit](#)

Sequence-to-sequence model

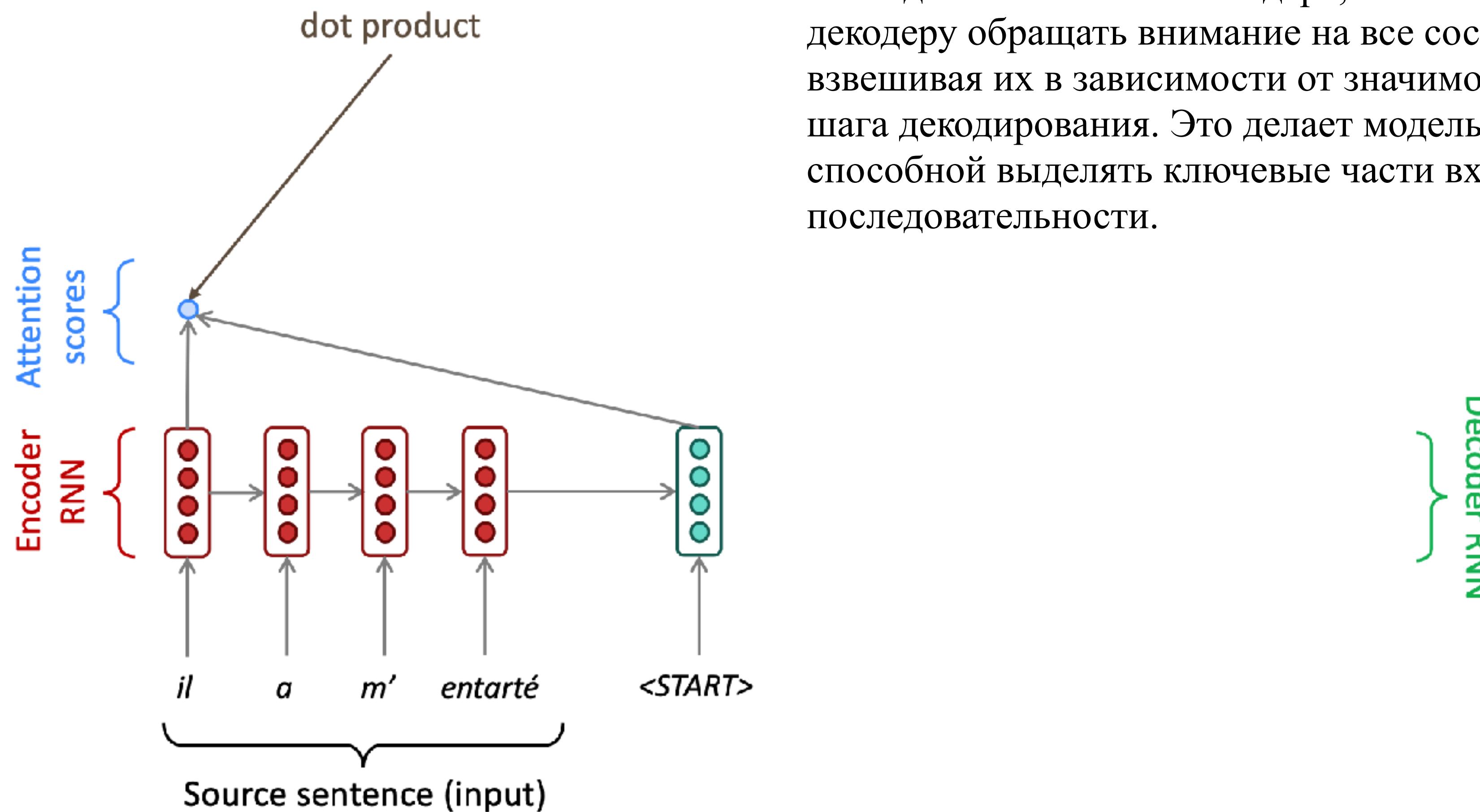
Informational bottleneck:

В одном векторе должна быть вся информация

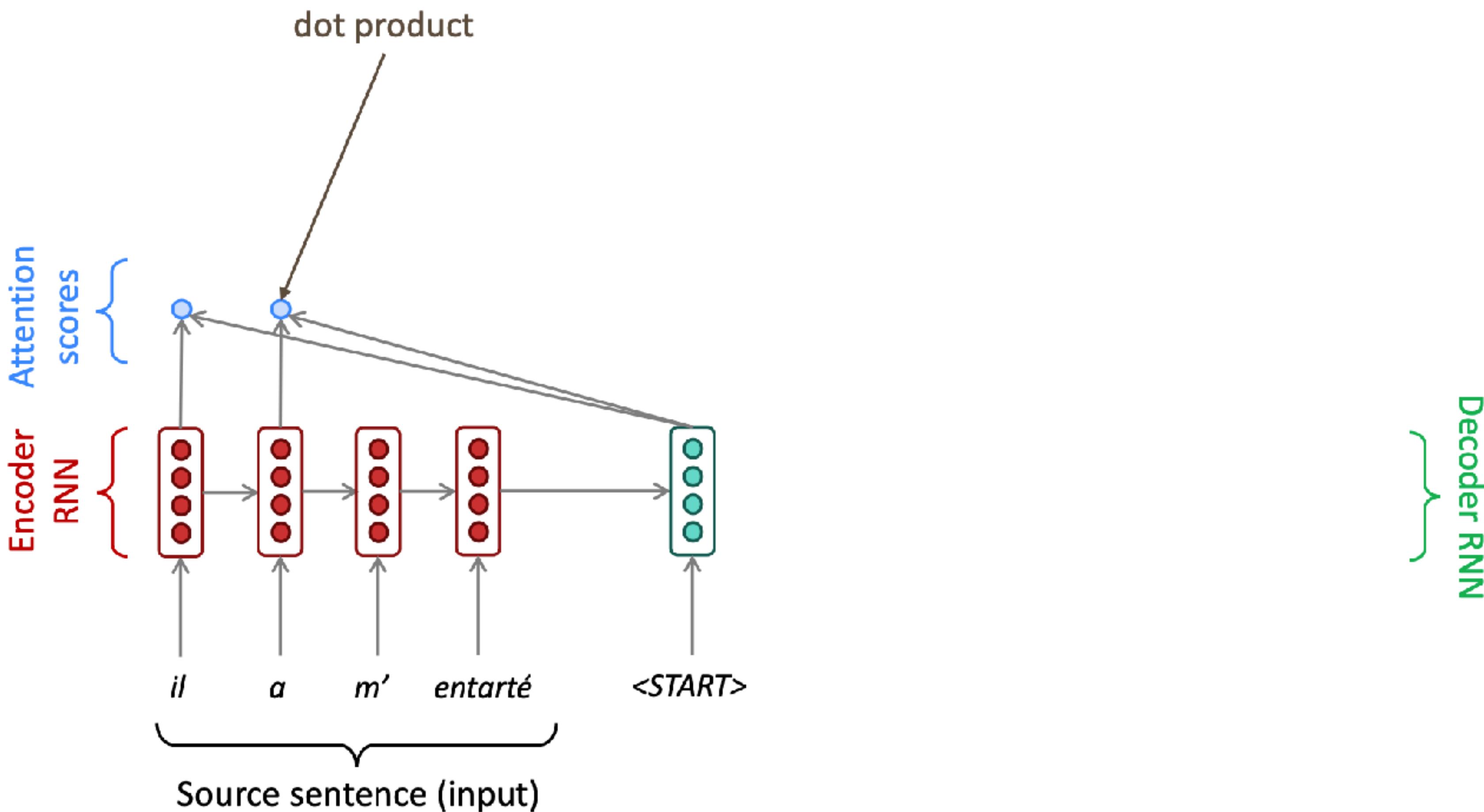


Attention

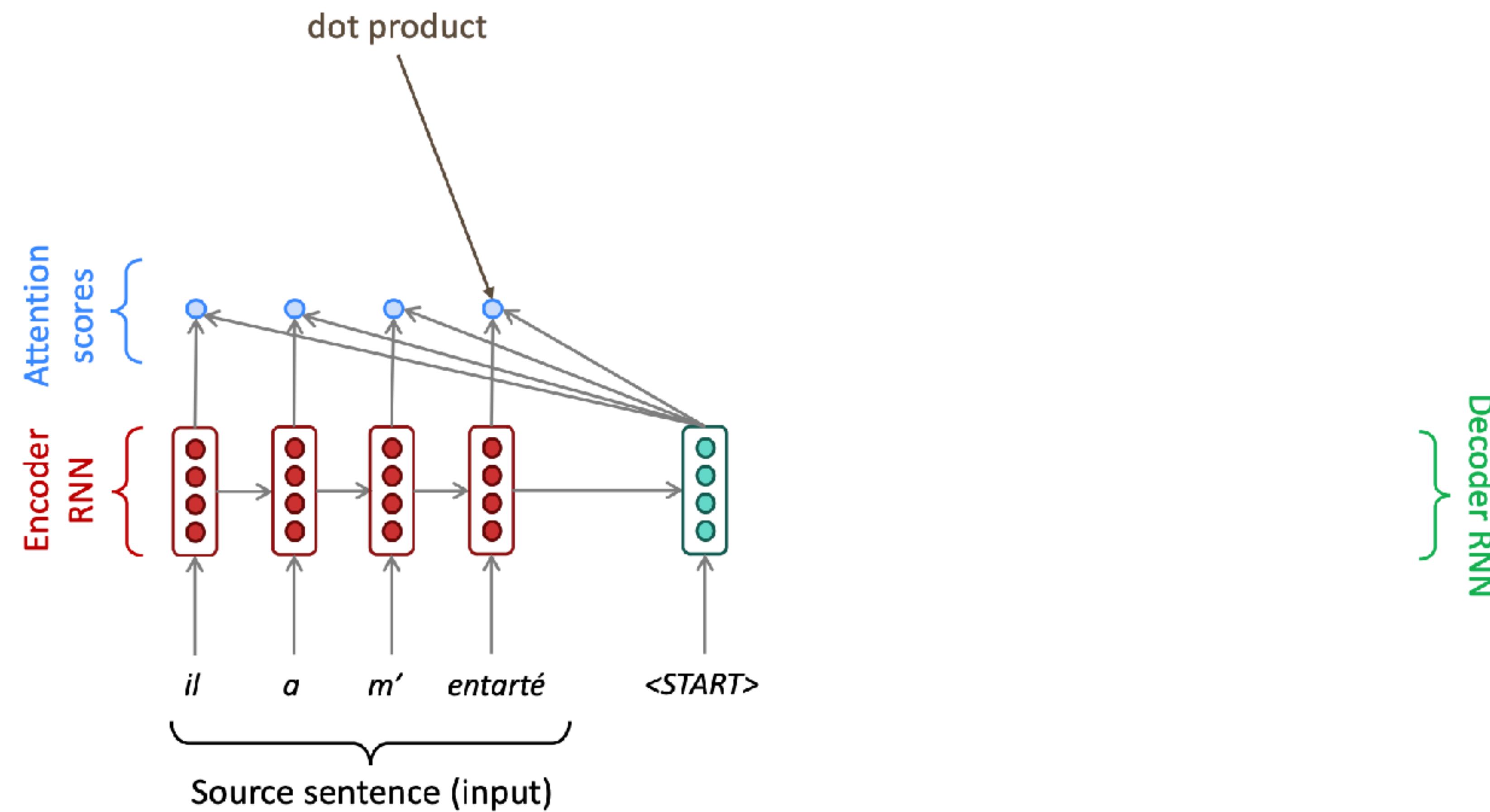
Механизм внимания значительно улучшает качество seq2seq моделей, особенно при работе с длинными последовательностями. Вместо использования только последнего состояния энкодера, attention позволяет декодеру обращать внимание на все состояния энкодера, взвешивая их в зависимости от значимости для текущего шага декодирования. Это делает модель более гибкой и способной выделять ключевые части входной последовательности.



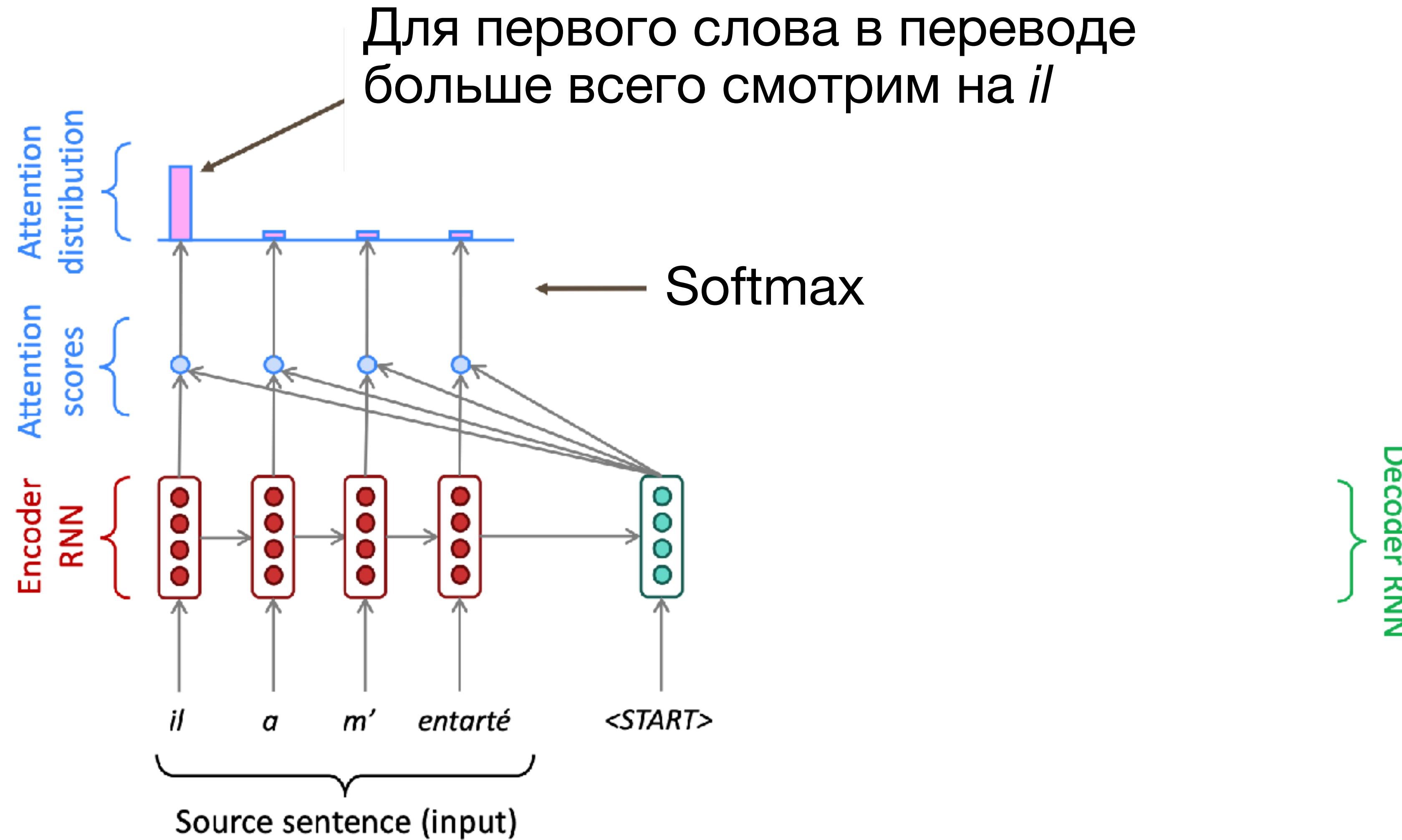
Attention



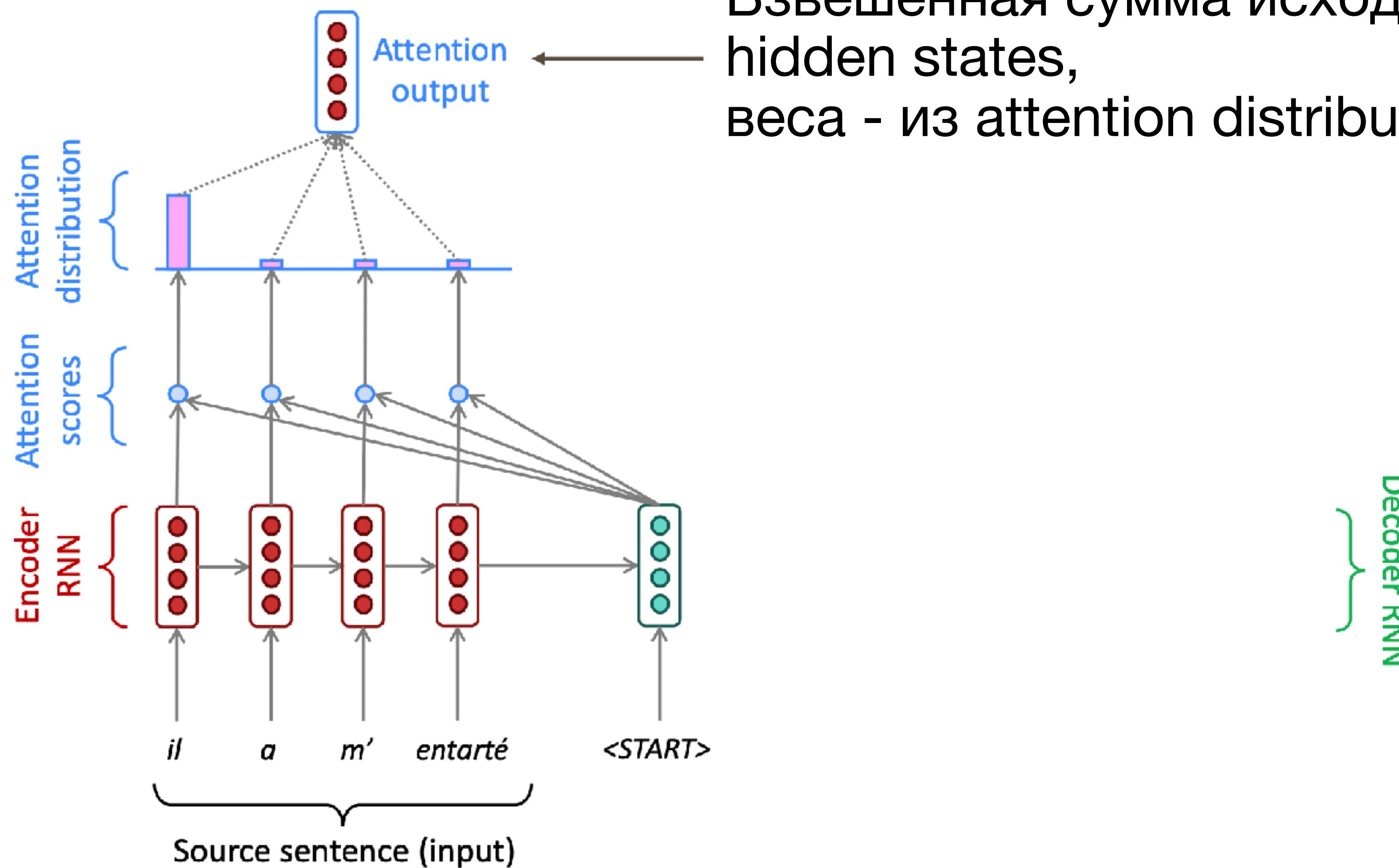
Attention



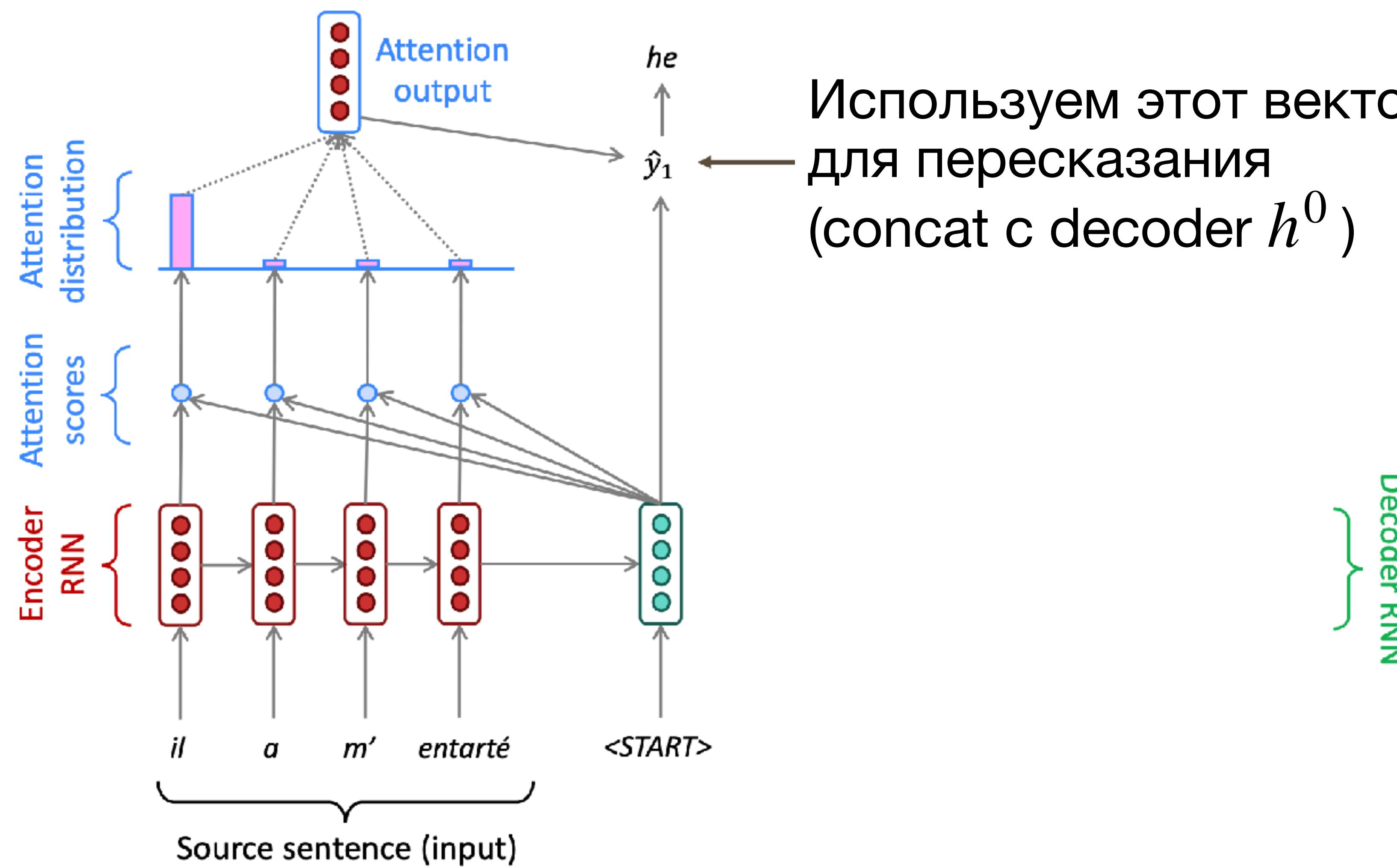
Attention



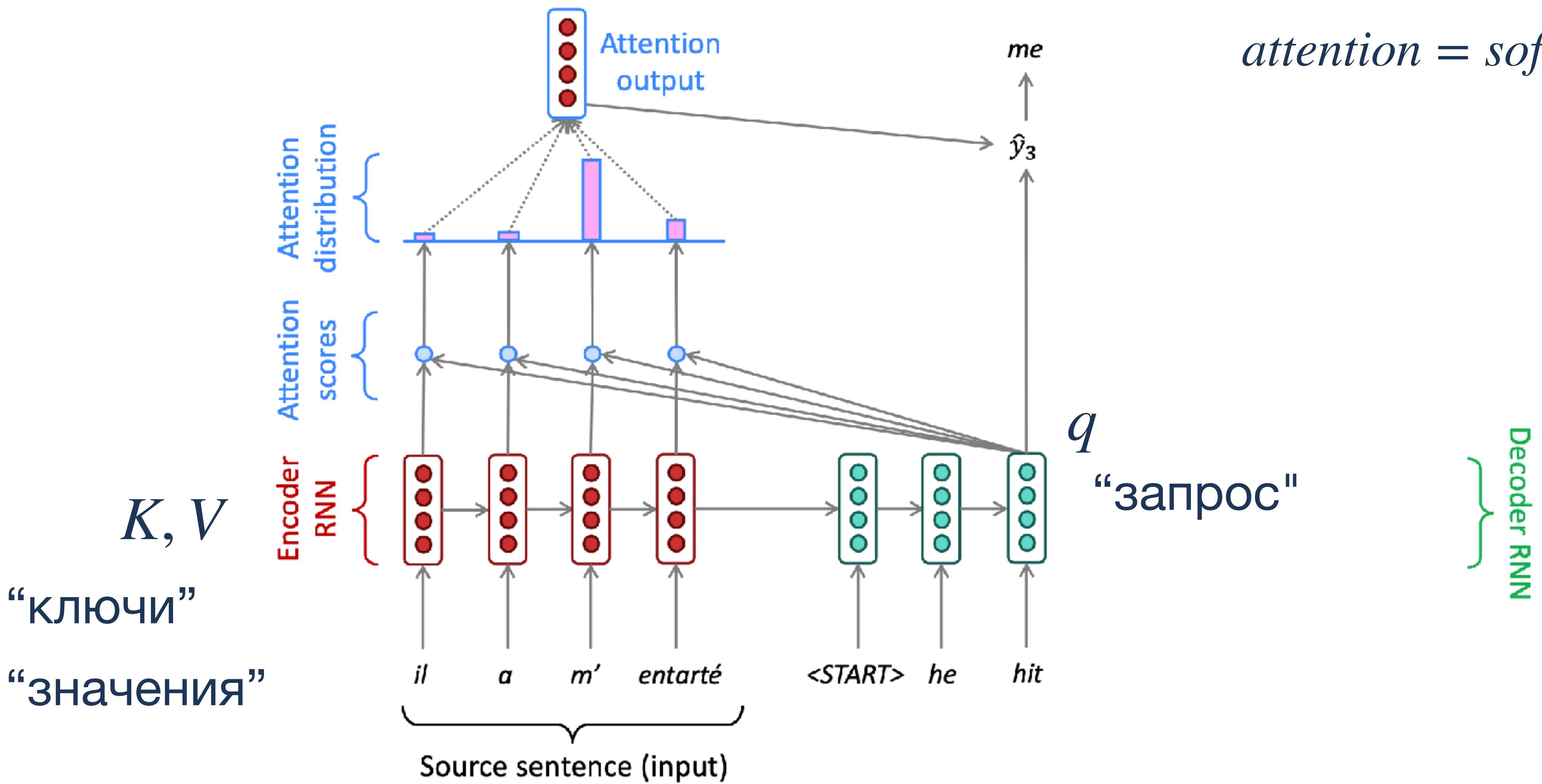
Attention



Attention

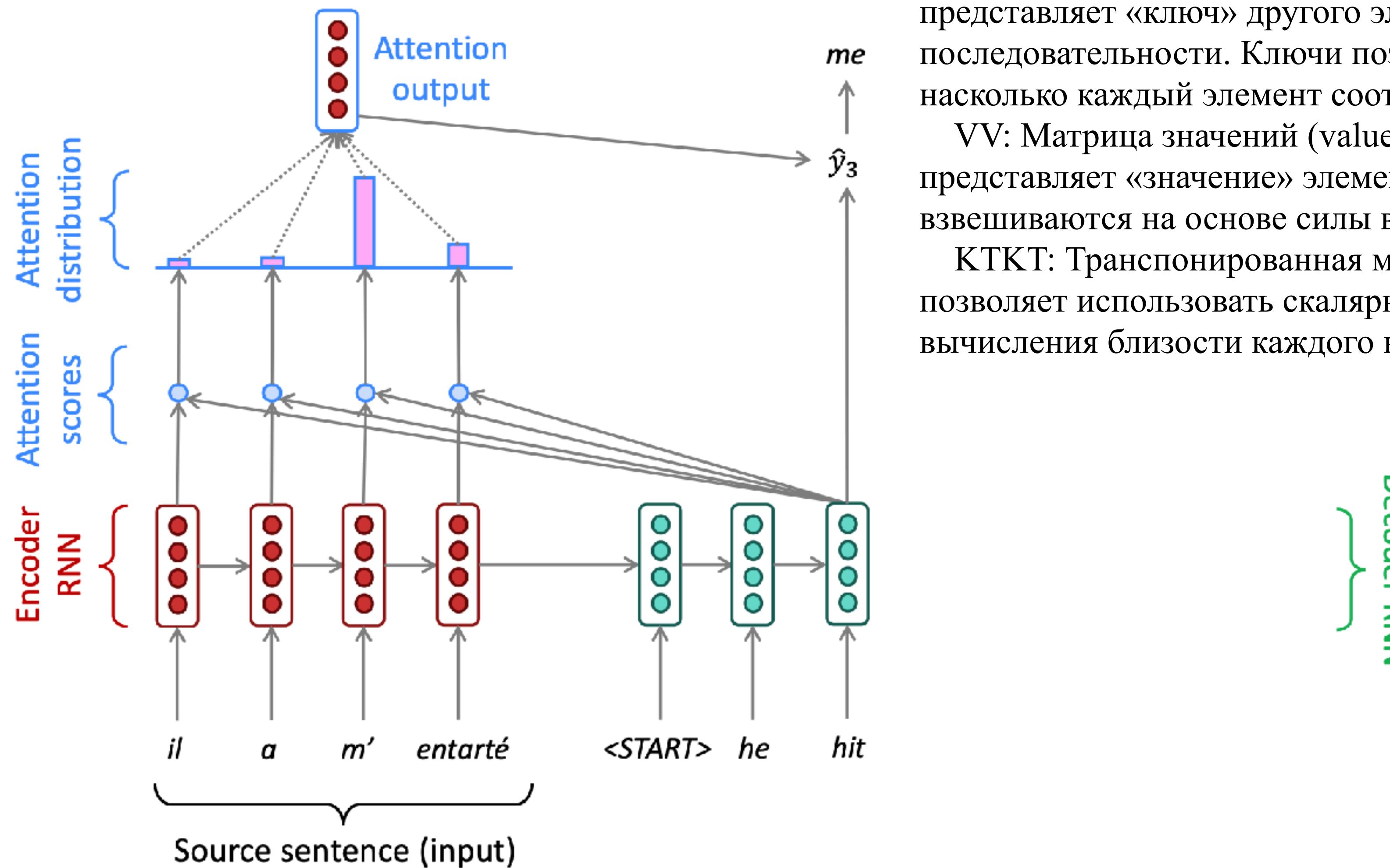


Attention



$$\text{attention} = \text{softmax}(qK^T)V$$

Attention



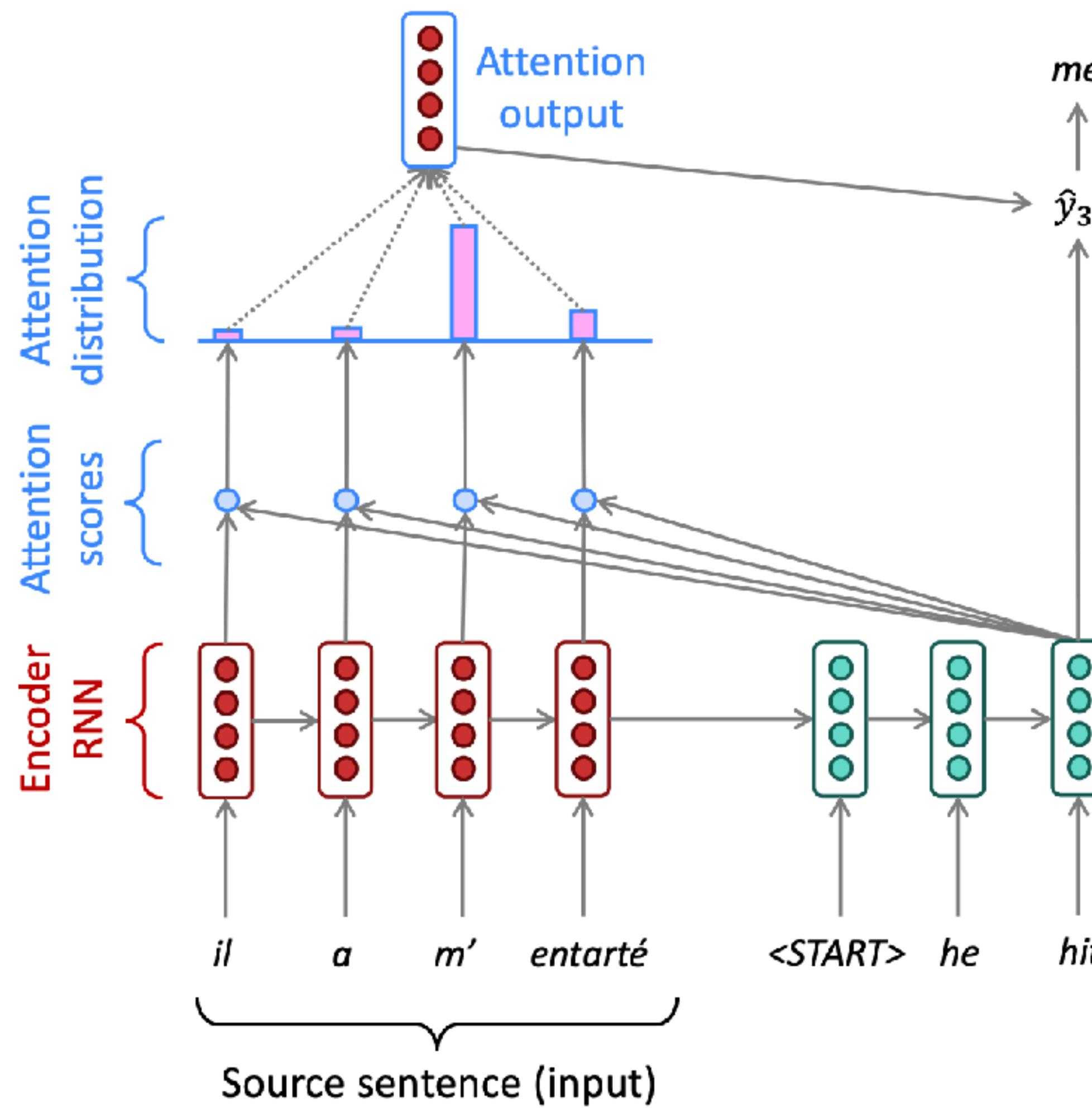
q: Вектор запроса (query), представляющий текущий элемент, к которому применяется внимание. Он указывает, какую информацию нужно «запросить» у остальных элементов.

KK: Матрица ключей (keys), где каждый столбец представляет «ключ» другого элемента в последовательности. Ключи позволяют определять, насколько каждый элемент соответствует запросу.

VV: Матрица значений (values), где каждый столбец представляет «значение» элемента. Эти значения взвешиваются на основе силы внимания к ним.

КТКТ: Транспонированная матрица ключей, что позволяет использовать скалярное произведение для вычисления близости каждого ключа к запросу qq.

Attention



Процесс вычисления внимания

Вычисление сходства (состыковка запроса и ключа):

qKT_{qKT}: Мы вычисляем скалярное произведение между запросом и каждым ключом, что создаёт оценки важности для каждого элемента относительно текущего запроса. Полученные оценки показывают, какие элементы важны для текущего запроса.

Применение Softmax:

softmax(qKT)softmax(qKT): Функция softmax применяется для нормализации оценок важности в диапазон от 0 до 1, чтобы получить вероятность для каждого элемента в последовательности. Эта вероятность показывает, насколько каждый элемент важен для данного запроса.

Взвешивание значений:

attention=softmax(qKT)Vattention=softmax(qKT)V: Нормализованные веса умножаются на значения, т.е. итоговое внимание представляет собой сумму значений, взвешенных по важности. В результате мы получаем представление информации, релевантной для текущего запроса.

Интуиция

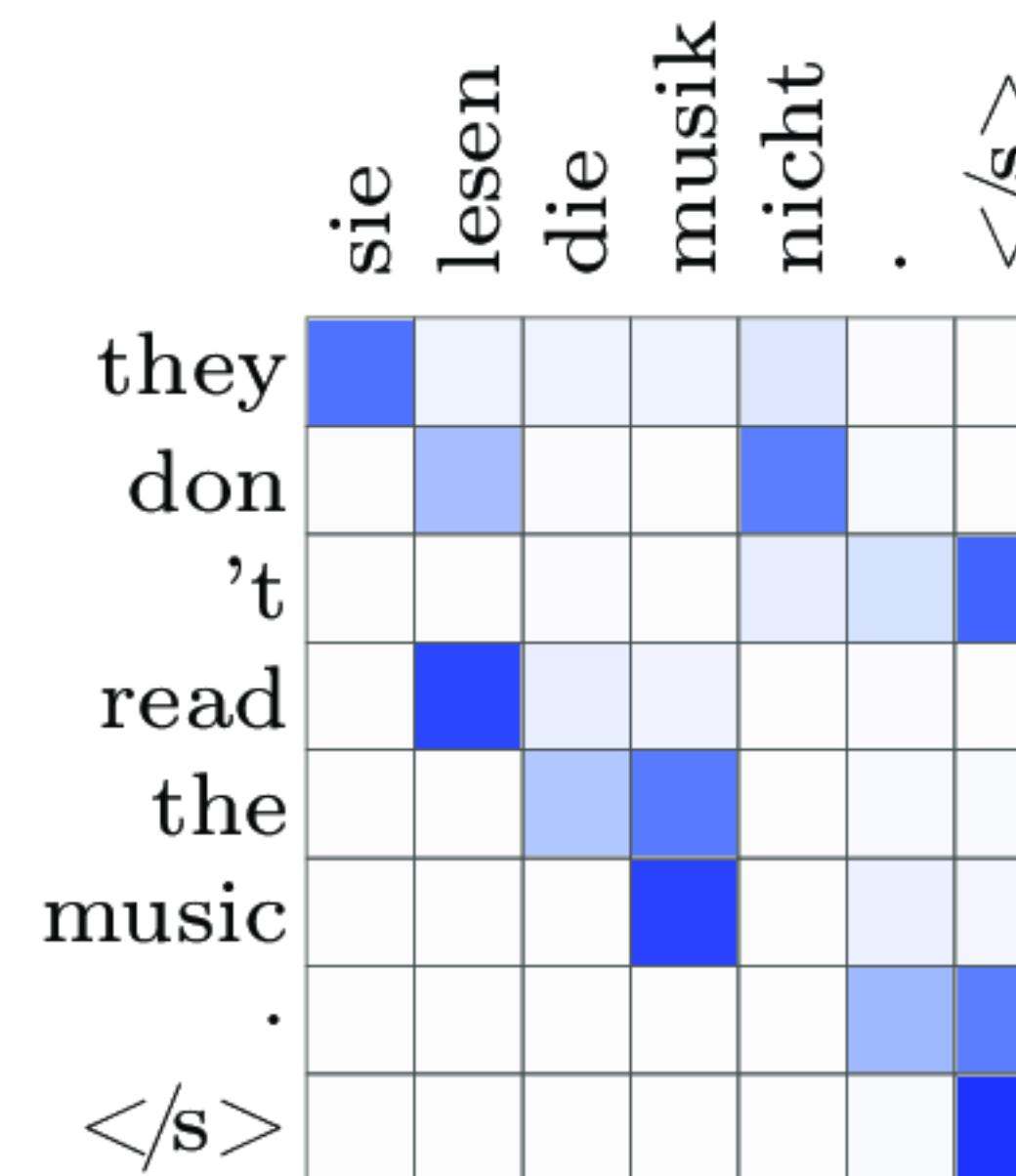
Итак, формула

attention=softmax(qKT)Vattention=softmax(qKT)V означает, что каждый элемент последовательности получает «вес» (насколько он важен), а затем его значение добавляется в итоговый ответ пропорционально этому весу.

[Image credit](#)

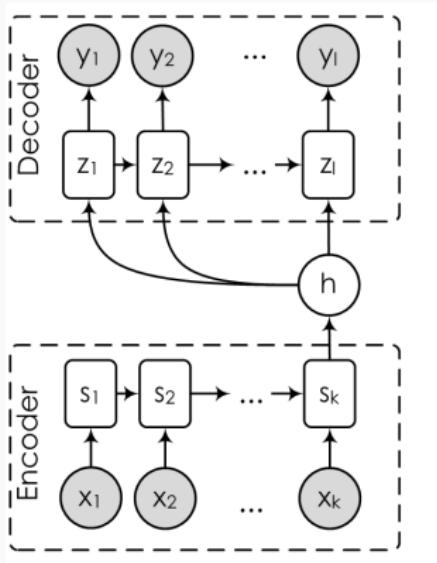
Attention

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



ENCODER-DECODER АРХИТЕКТУРЫ

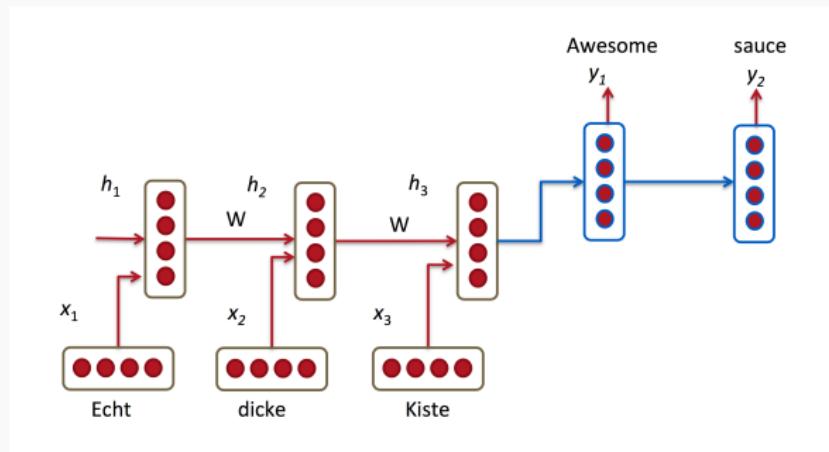
- Encoder-decoder архитектуры (Sutskever et al., 2014; Cho et al., 2014):



- Сначала кодируем, потом декодируем обратно.

ENCODER-DECODER АРХИТЕКТУРЫ

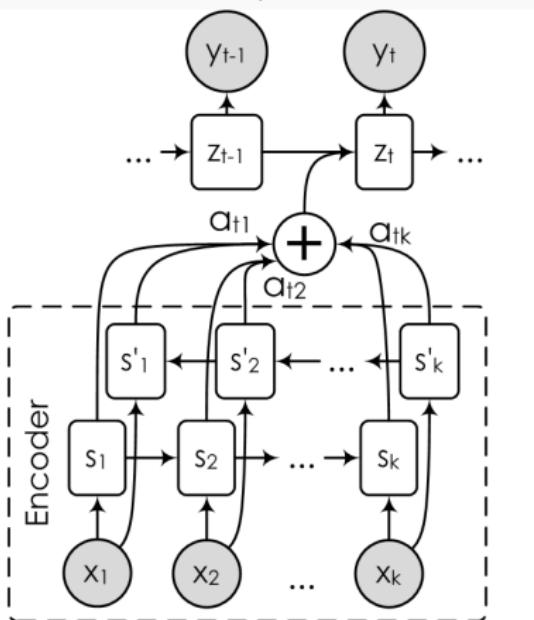
- Так же может работать и с переводом.



- Проблема: надо сжимать всё предложение в один вектор.
- С длинными участками текста это вообще перестаёт работать.

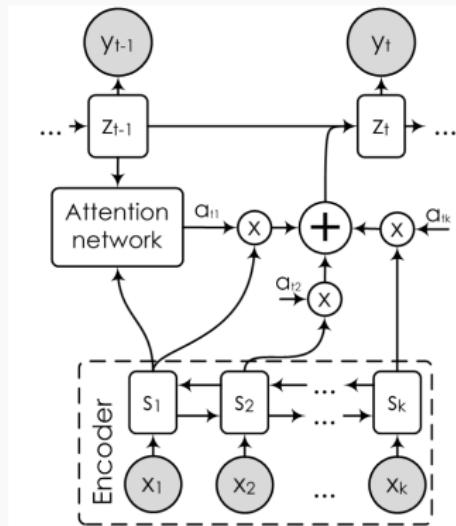
ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Решение: давайте обучим специальные веса, показывающие, насколько та или иная часть входа важна для текущего выхода.
- Прямое применение – двунаправленный LSTM плюс внимание (Bahdanau et al. 2014):



ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Мягкое внимание (soft attention) (Luong et al. 2015a; 2015b; Jean et al. 2015):
 - encoder – двунаправленная RNN, есть оба контекста;
 - сеть внимания выдаёт оценку релевантности – надо ли переводить это слово прямо сейчас?

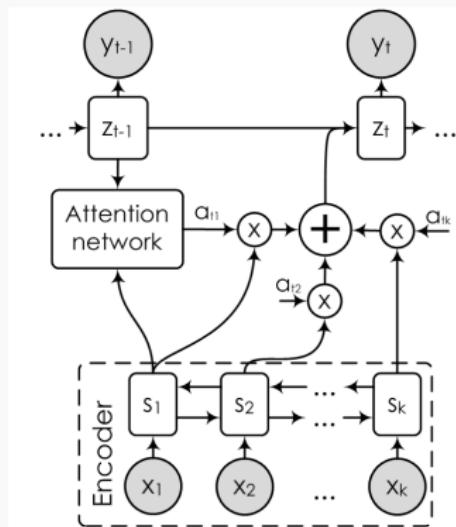


ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Формально очень просто: считаем веса внимания α_{tj} и перевзвешиваем векторы контекстов:

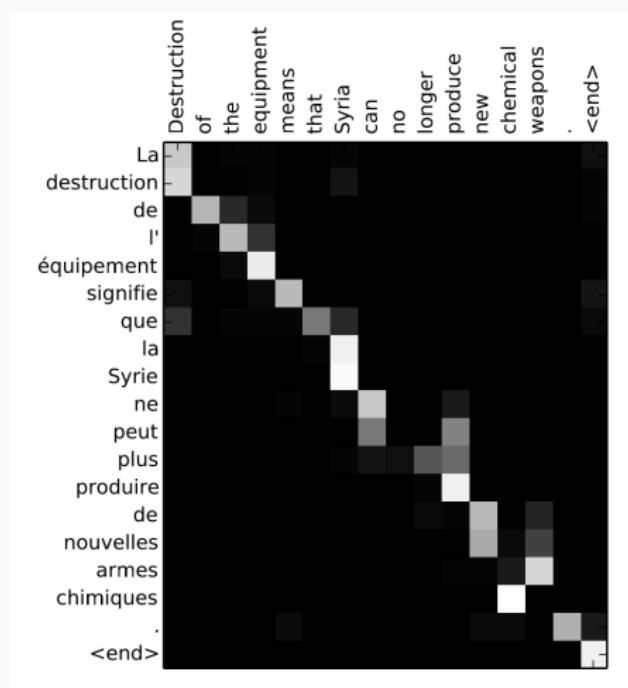
$$e_{tj} = a(z_{t-1}, j), \quad \alpha_{tj} = \text{softmax}(e_{tj}; e_{t*}),$$

$$c_t = \sum_j \alpha_{tj} h_j, \text{ и теперь } z_t = f(s_{t-1}, y_{t-1}, c_i).$$



ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- В результате можно визуализировать, на что смотрит сеть:



ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Получается гораздо лучше порядок слов:

Economic growth has slowed down in recent years .

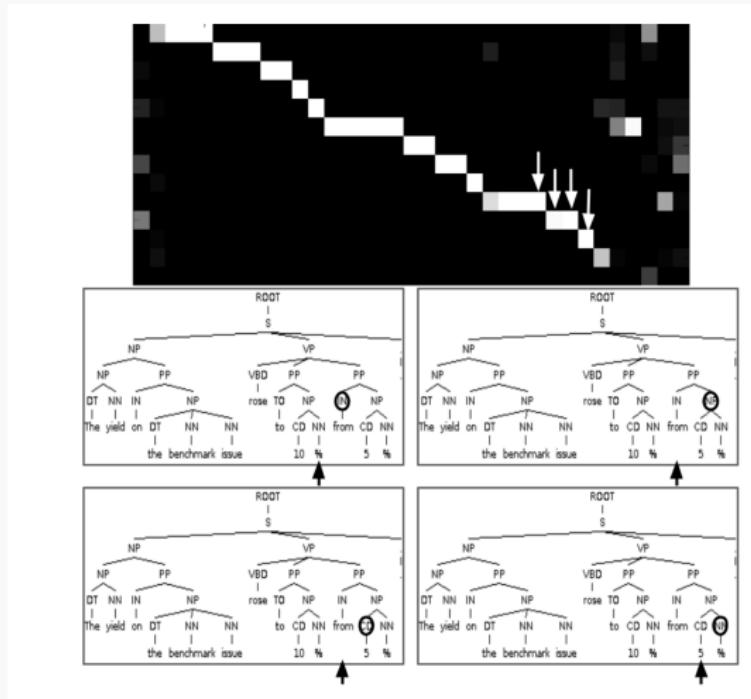
Das Wirtschaftswachstum hat sich in den letzten Jahren verlangsamt .

Economic growth has slowed down in recent years .

La croissance économique s' est ralentie ces dernières années .

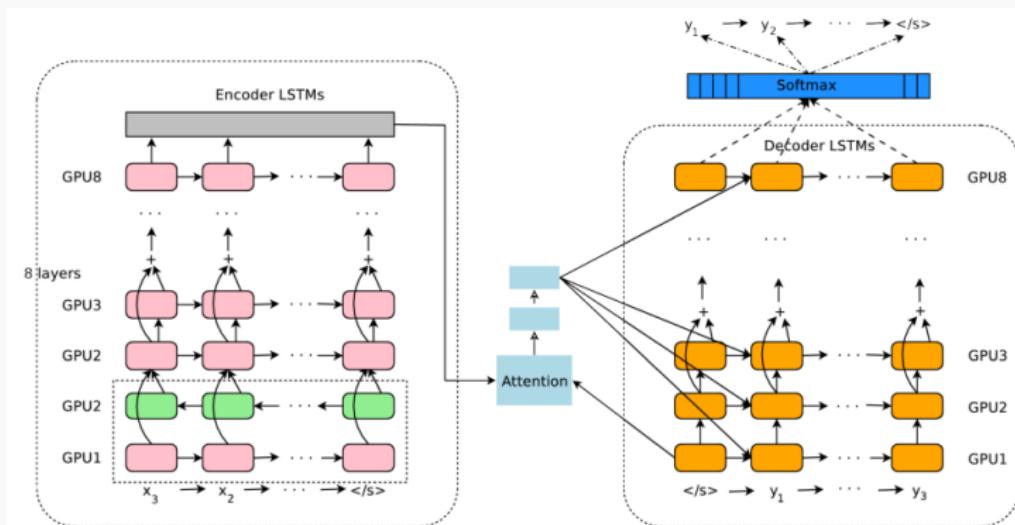
ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Другая необычная работа – «Grammar as a Foreign Language» (Vinyals et al., 2015)



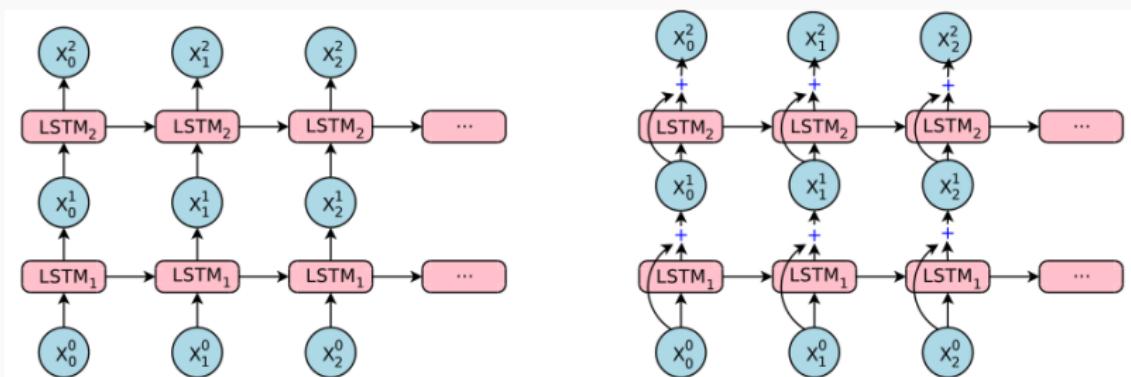
GOOGLE TRANSLATE

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
 - как на самом деле работает Google Translate;
 - базовая архитектура та же самая: encoder, decoder, attention;
 - RNN глубокие, по 8 уровней в encoder и decoder:



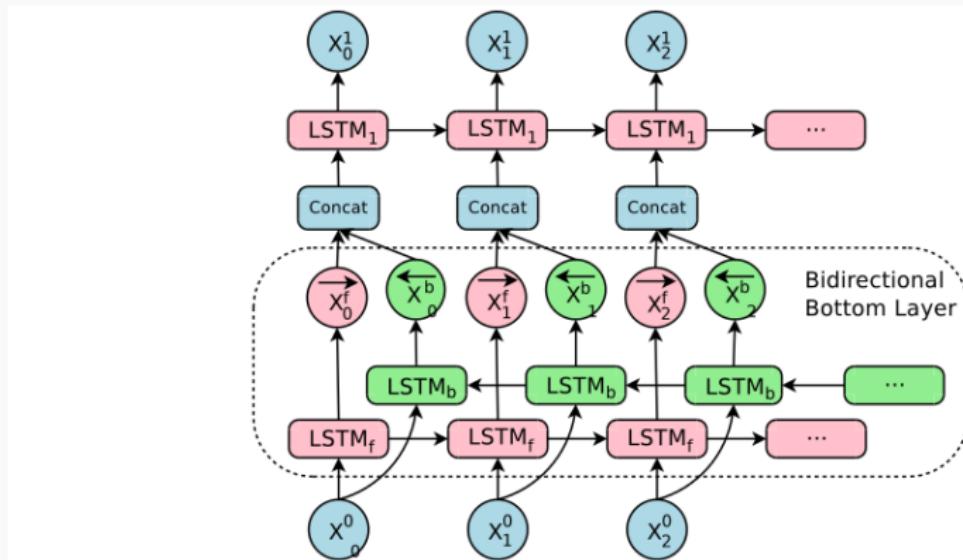
GOOGLE TRANSLATE

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation:*
 - просто stacked LSTM перестают работать далее 4-5 уровней;
 - поэтому добавляют остаточные связи, как в ResNet:



GOOGLE TRANSLATE

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
 - нижний уровень, естественно, двунаправленный:



- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
- в GNMT ещё две идеи о сегментации слов:
 - *wordpiece model*: разбить слова на кусочки (отдельной моделью); пример из статьи:

Jet makers feud over seat width with big orders at stake

превращается в

_J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

- *mixed word/character model*: конвертировать слова, не попадающие в словарь, в последовательность букв-токенов; пример из статьи:

Miki превращается в M <M>i <M>k <E>i

TEACHING MACHINES TO READ

- (Hermann et al., 2015): «Teaching machines to read and comprehend» (Google DeepMind)
- Предлагают новый способ построить датасет для понимания, автоматически создавая тройки (context, query, answer) из текстов новостей и т.п.

Original Version	Anonymised Version
Context <p>The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the “Top Gear” host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon “to an unprovoked physical and verbal attack.” ...</p>	<p>the <i>ent381</i> producer allegedly struck by <i>ent212</i> will not press charges against the “ <i>ent153</i> ” host , his lawyer said friday . <i>ent212</i> , who hosted one of the most - watched television shows in the world , was dropped by the <i>ent381</i> wednesday after an internal investigation by the <i>ent180</i> broadcaster found he had subjected producer <i>ent193</i> “ to an unprovoked physical and verbal attack . ” ...</p>
Query <p>Producer X will not press charges against Jeremy Clarkson, his lawyer says.</p>	<p>producer X will not press charges against <i>ent212</i> , his lawyer says .</p>
Answer <p>Oisin Tymon</p>	<p><i>ent193</i></p>

Transformer

Transformer

The screenshot shows the Google Translate interface. At the top, there's a navigation bar with three horizontal dots on the right. Below it, two tabs are visible: 'Text' (selected) and 'Documents'. The main area has two language pairs: English to Russian and Russian to English. The English input field contains the sentence: "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 field contains the translation: "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 input sentence, there are microphone and speaker icons. In the center, it says "138/5000" and there's a pencil icon. To the right of the Russian translation, there are icons for copy, edit, and share, along with a "Send feedback" link.

Изначально предложен для перевода

Transformer

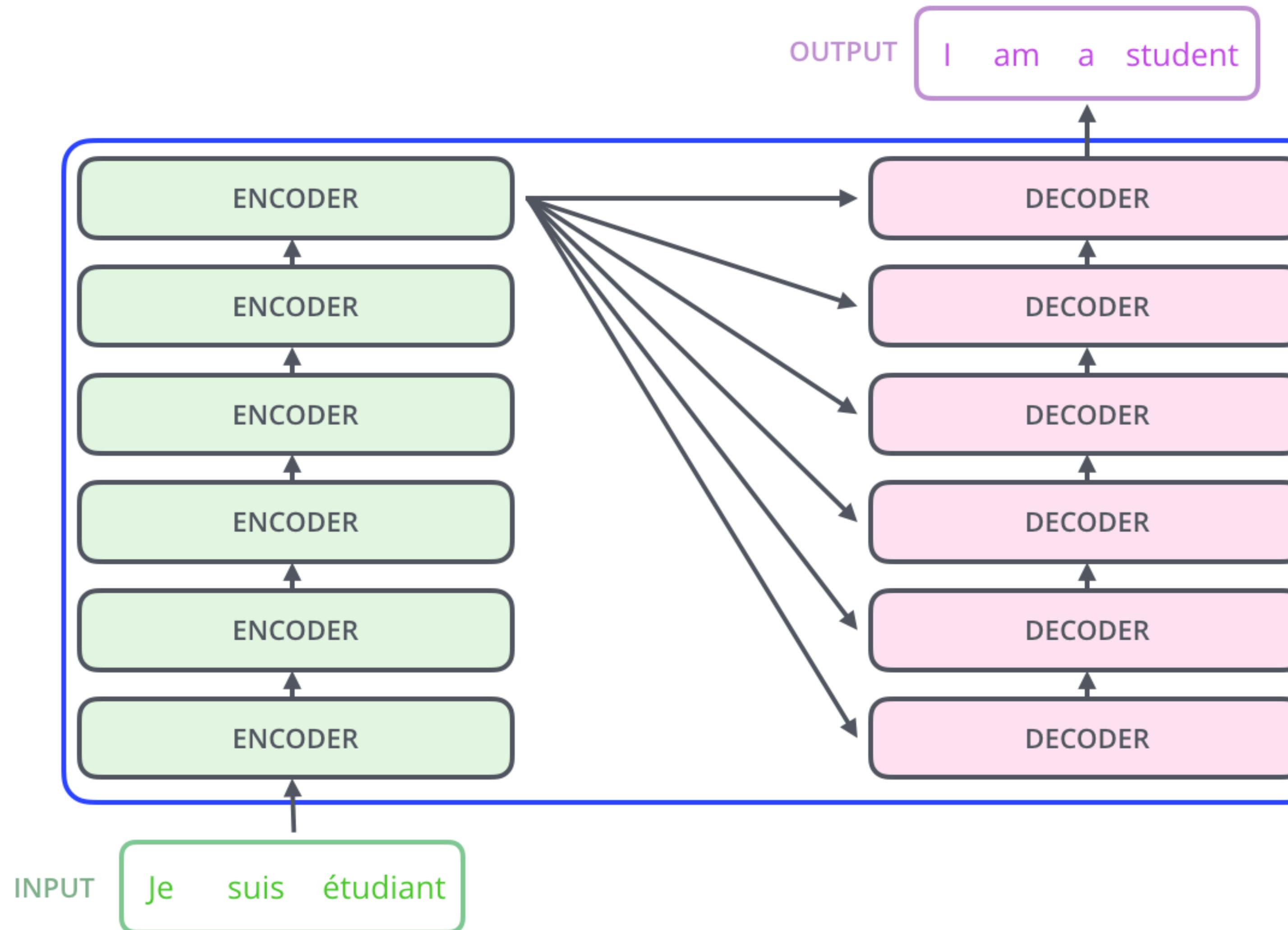
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1		$3.3 \cdot 10^{18}$
Transformer (big)	28.4	41.8		$2.3 \cdot 10^{19}$

Изначально предложен для перевода

- Лучше качество (в других задачах тоже!)
- Быстрее

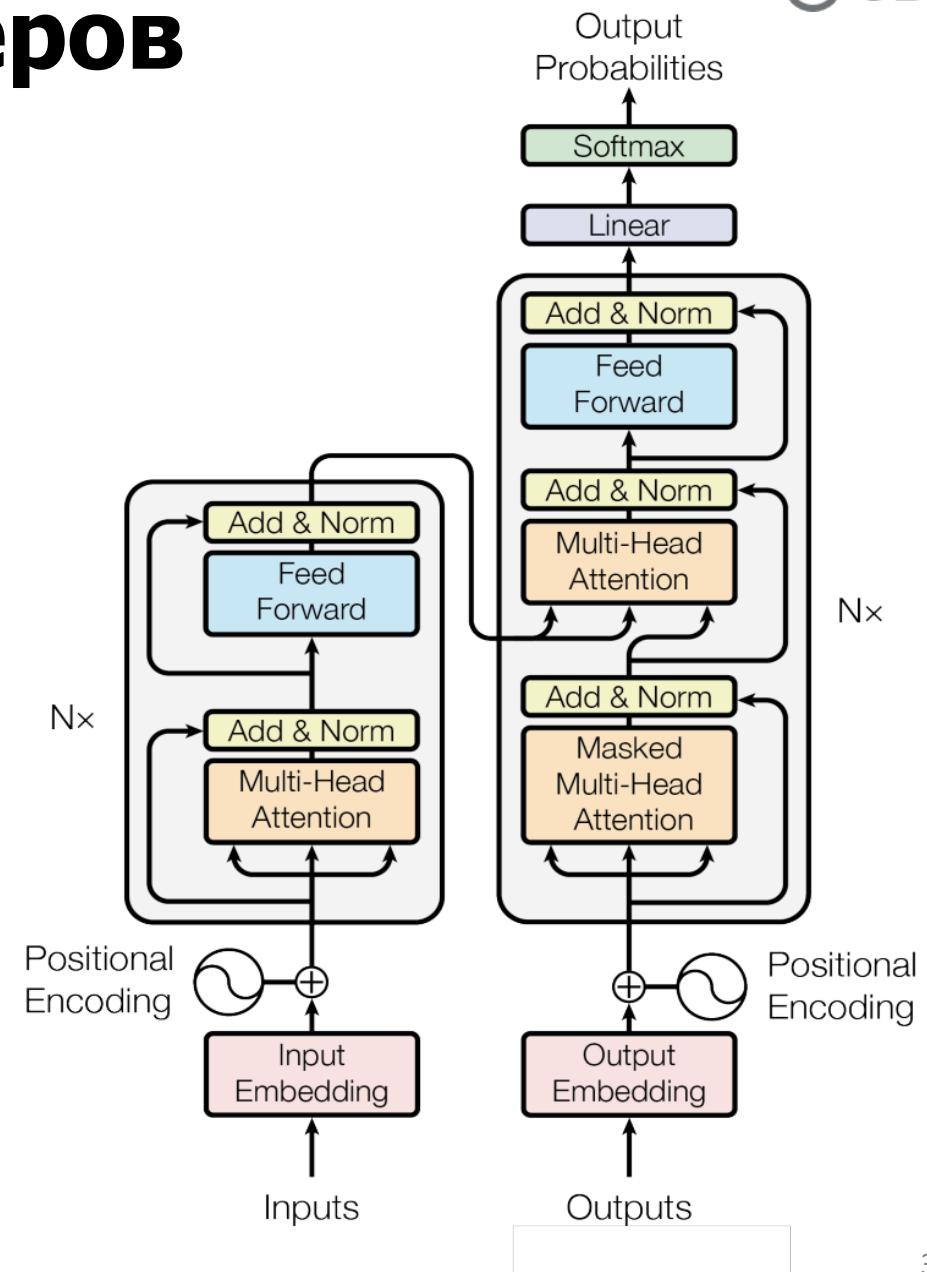
Transformer



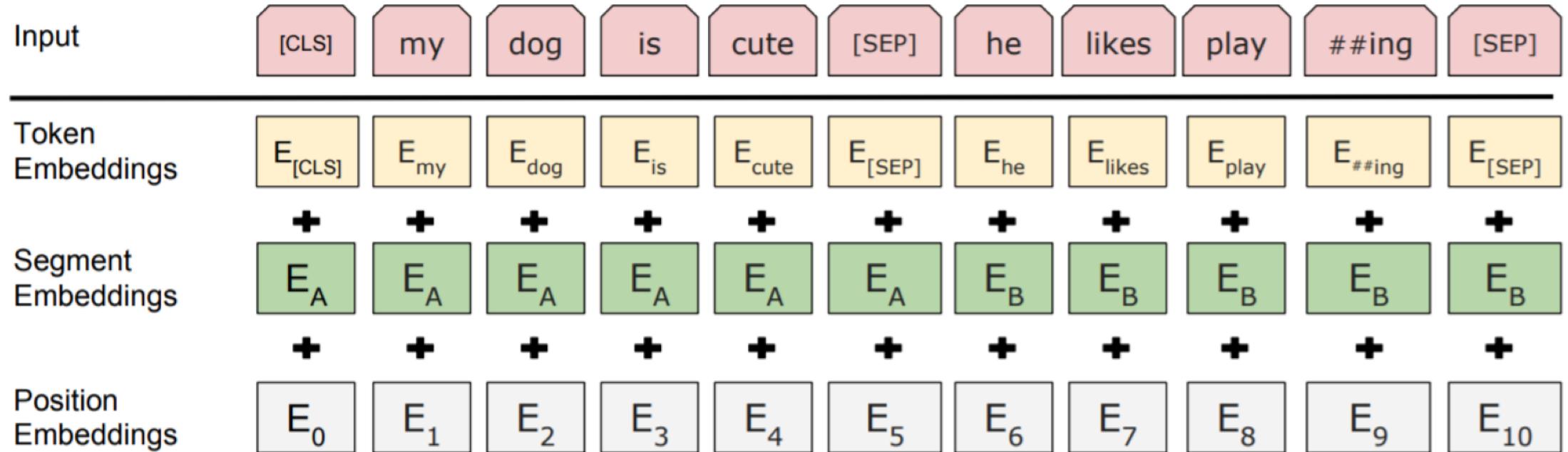
Архитектура трансформеров

Архитектура состоит из двух типов кодировщиков: энкодеры – левая часть трансформера и декодеры – правая половина трансформера, которые состоят из повторяющихся блоков, представляющих собой последовательность:

- Self-attention
- Суммирование выходов и нормализация слоя
- Residual connections
- Полносвязные слои нейронной сети



Embeddings + Positioning



Self-Attention

Вводятся весовые матрицы Q, K, V ($d \times d$ каждая):

$$\mathbf{q}_i = Q\mathbf{x}_i \text{ (queries)}$$

$$\mathbf{v}_i = V\mathbf{x}_i \text{ (values)}$$

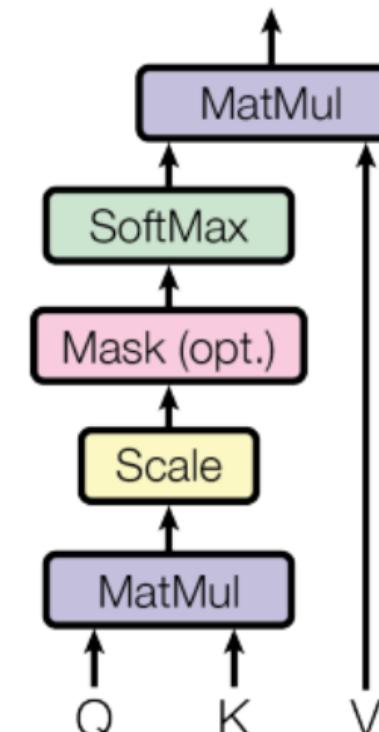
$$\mathbf{k}_i = K\mathbf{x}_i \text{ (keys)}$$

Далее вычисляется скалярное произведение Q и K и производится нормировка (softmax) и вычисляется output каждого слова:

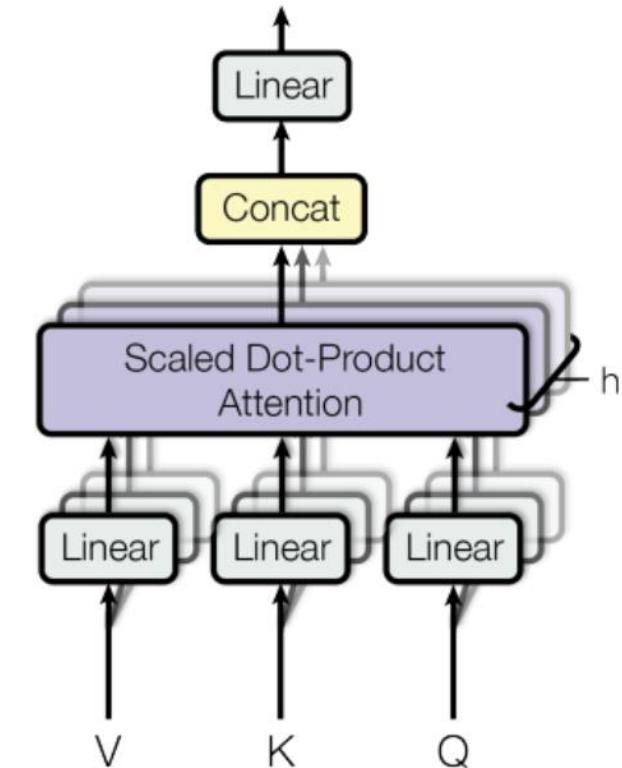
$$\mathbf{e}_{ij} = \mathbf{q}_i^\top \mathbf{k}_j \quad \alpha_{ij} = \frac{\exp(\mathbf{e}_{ij})}{\sum_j \exp(\mathbf{e}_{ij'})}$$

$$\mathbf{o}_i = \sum_j \alpha_{ij} \mathbf{v}_i$$

Scaled Dot-Product Attention



Multi-Head Attention



Multi-Headed Attention. Adapted from [Vaswani et al. \(2017\)](#)

Transformer: self-attention

Seq2seq attention: между decoder vector и encoder vectors

- какие слова в input “важны” для предсказания текущего output

$$\text{attention} = \text{softmax}(qK^T)V$$

Transformer: self-attention

Seq2seq attention: между decoder vector и encoder vectors

- какие слова в input “важны” для предсказания текущего output

$$\text{attention} = \text{softmax}(qK^T)V$$

Self-attention (encoder): между encoder векторами

- какие слова в input как соотносятся между собой

$$\text{attention} = \text{softmax}\left(\frac{QK^T}{d}\right)V$$

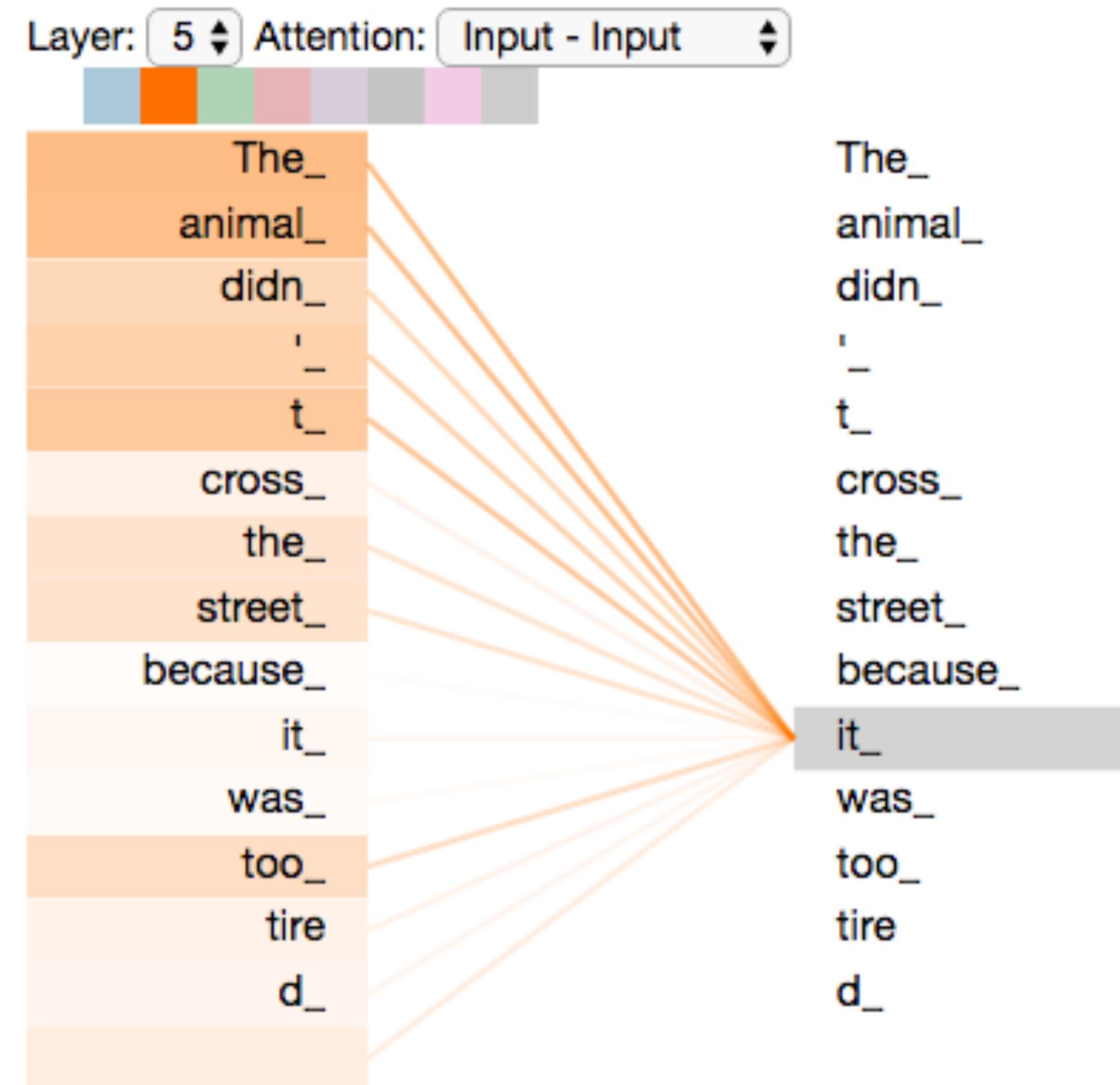
Transformer: self-attention

"The animal didn't cross the street because **it** was too tired"

На что указывает **it**?

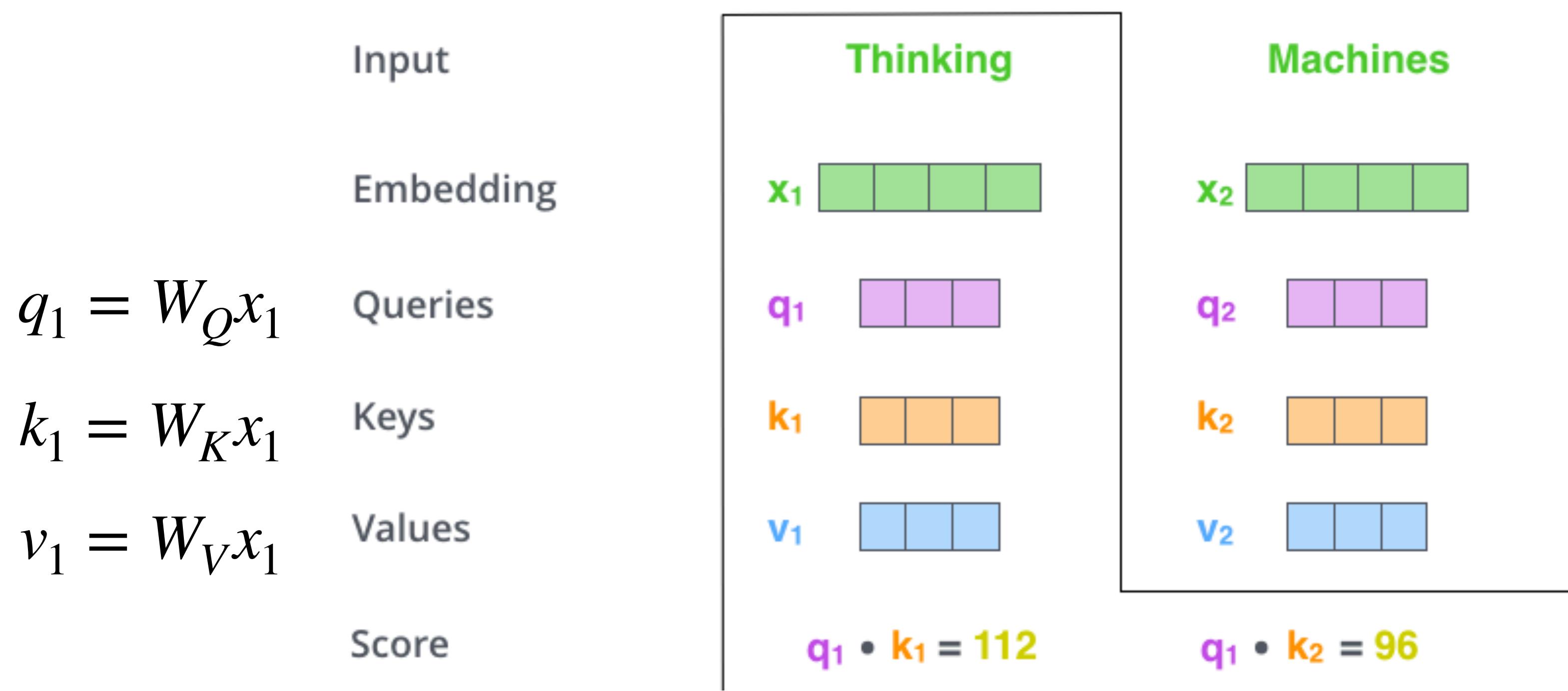
Transformer: self-attention

"The animal didn't cross the street because **it** was too tired"



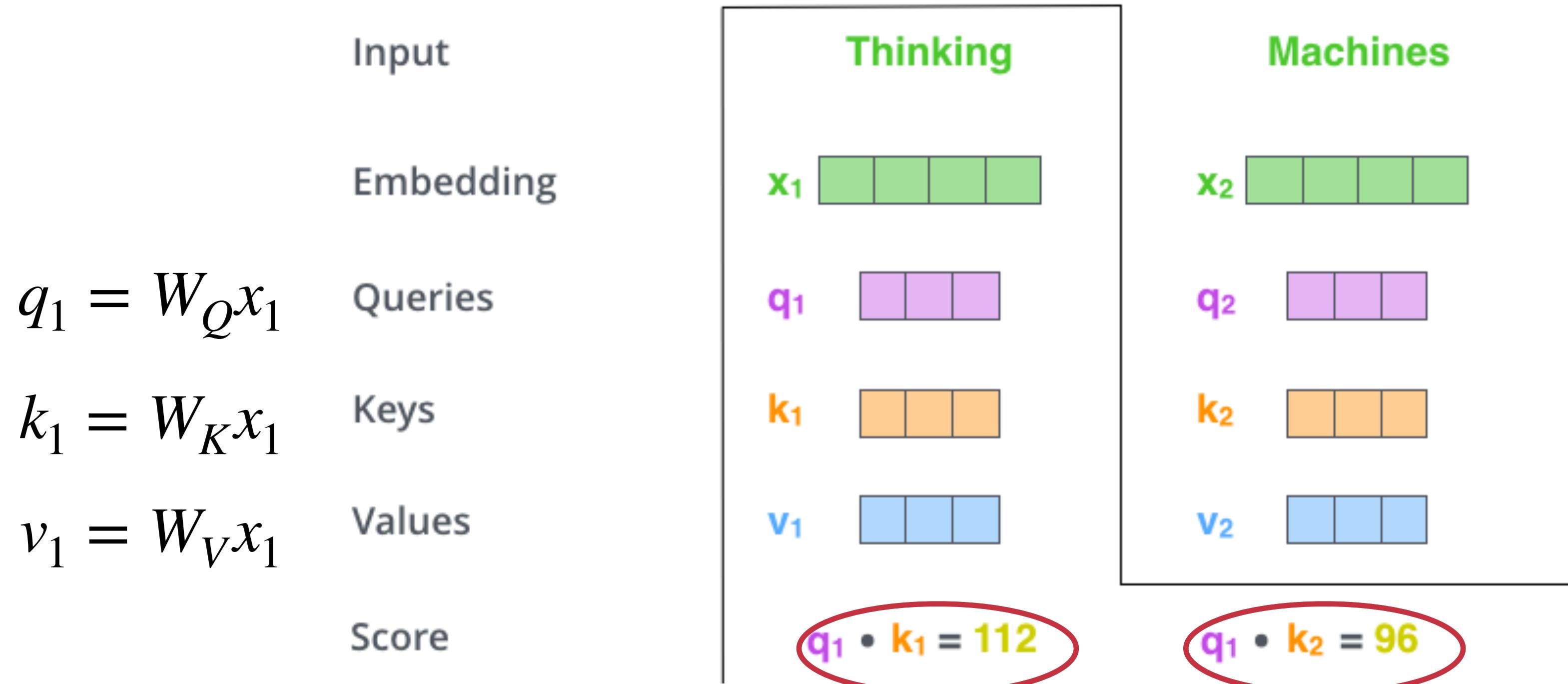
Transformer: self-attention

$$\text{attention} = \text{softmax}\left(\frac{QK^T}{d}\right)V$$



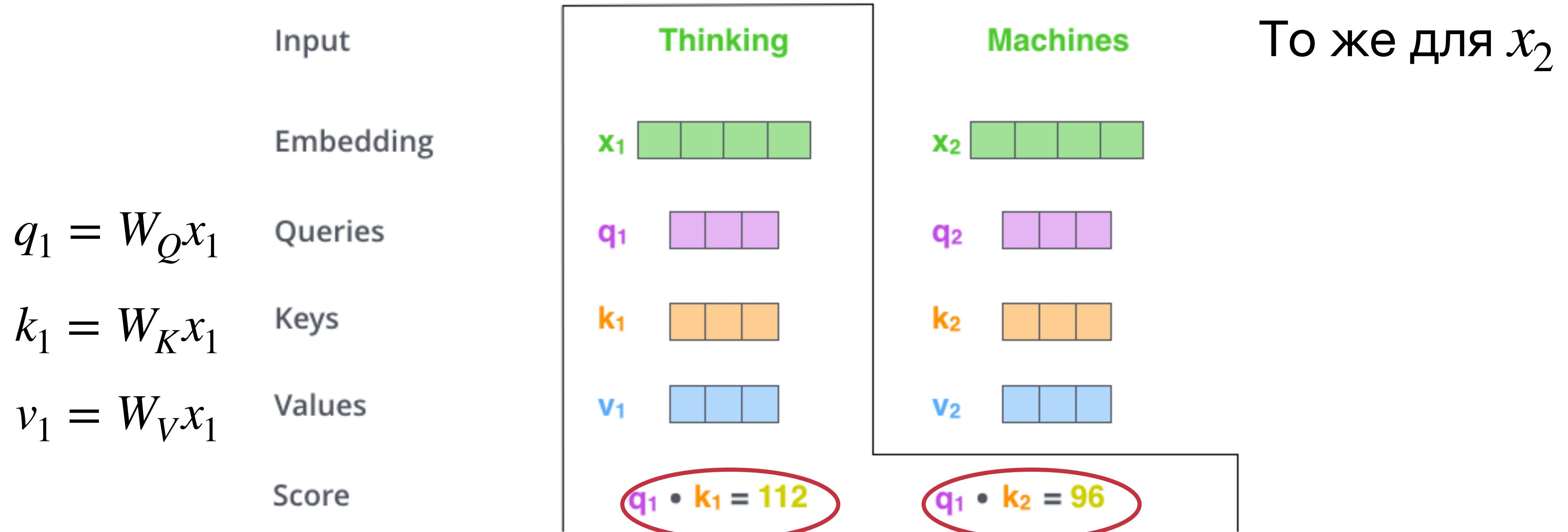
Transformer: self-attention

$$\text{attention} = \text{softmax}\left(\frac{QK^T}{d}\right)V$$



Transformer: self-attention

$$\text{attention} = \text{softmax}\left(\frac{QK^T}{d}\right)V$$



Transformer: self-attention

$$\text{attention} = \text{softmax}\left(\frac{QK^T}{d}\right)V$$

Input

Embedding

$$q_1 = W_Q x_1$$

Queries

$$k_1 = W_K x_1$$

Keys

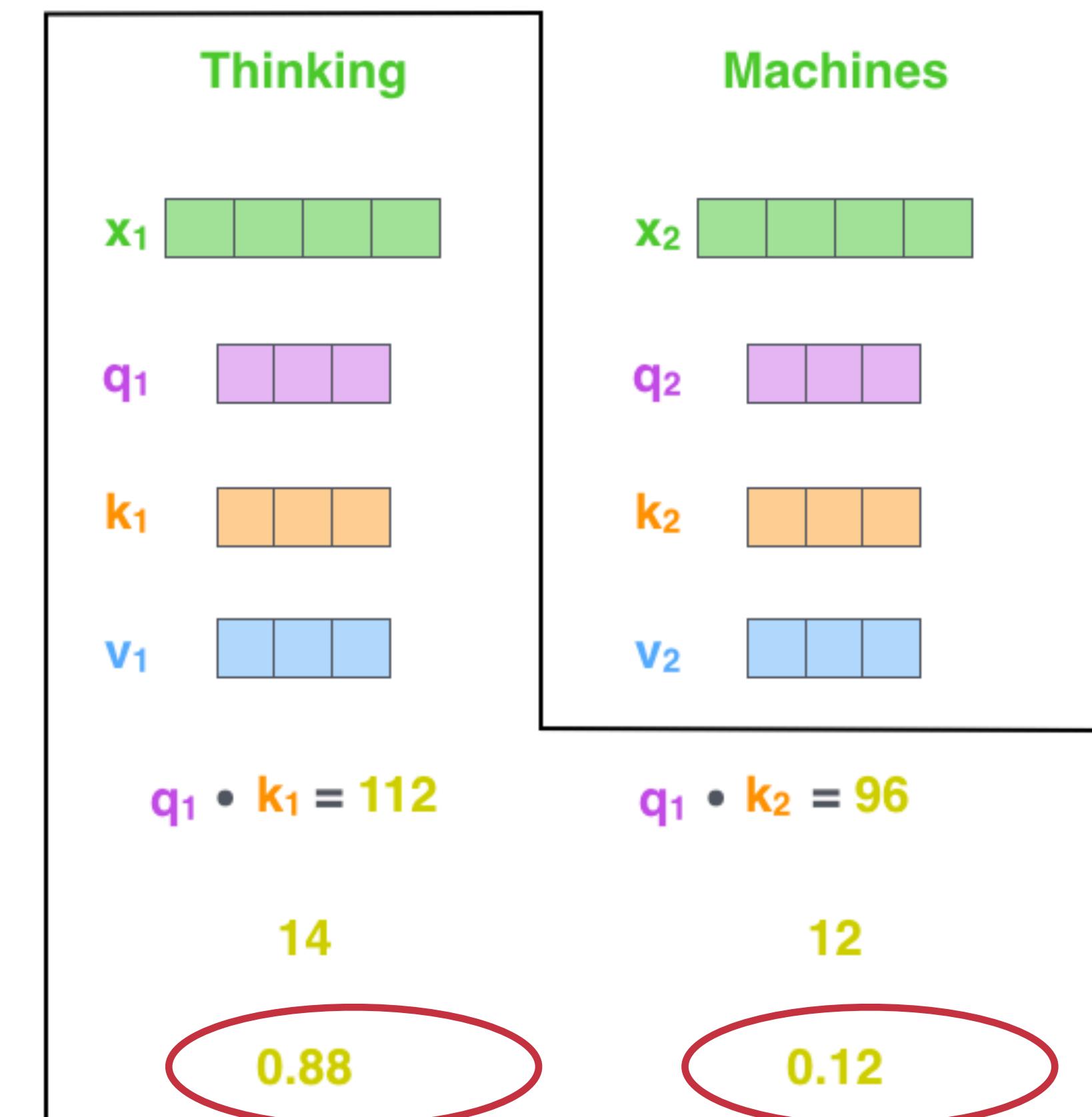
$$v_1 = W_V x_1$$

Values

Score

Divide by 8 ($\sqrt{d_k}$)

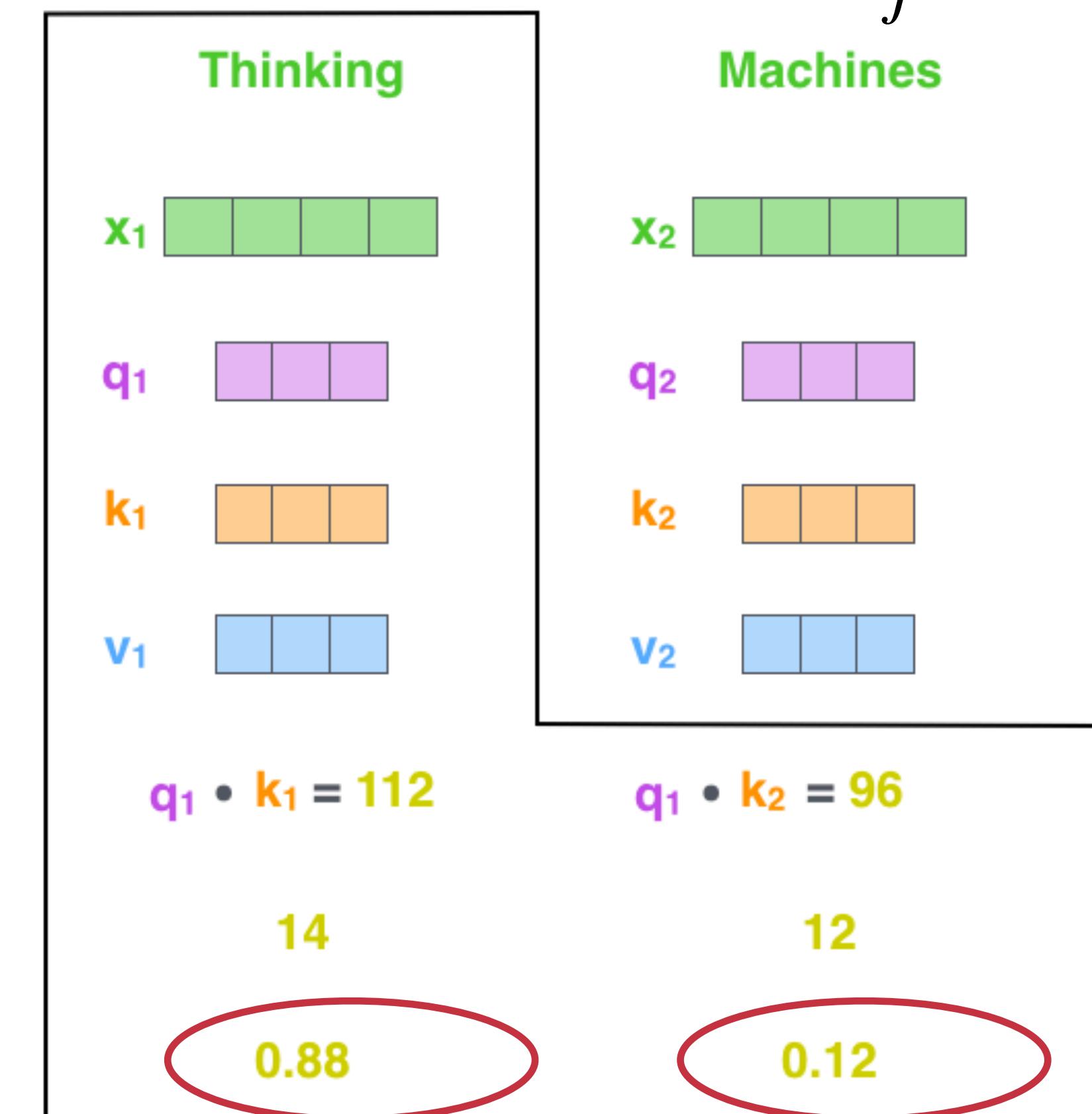
Softmax



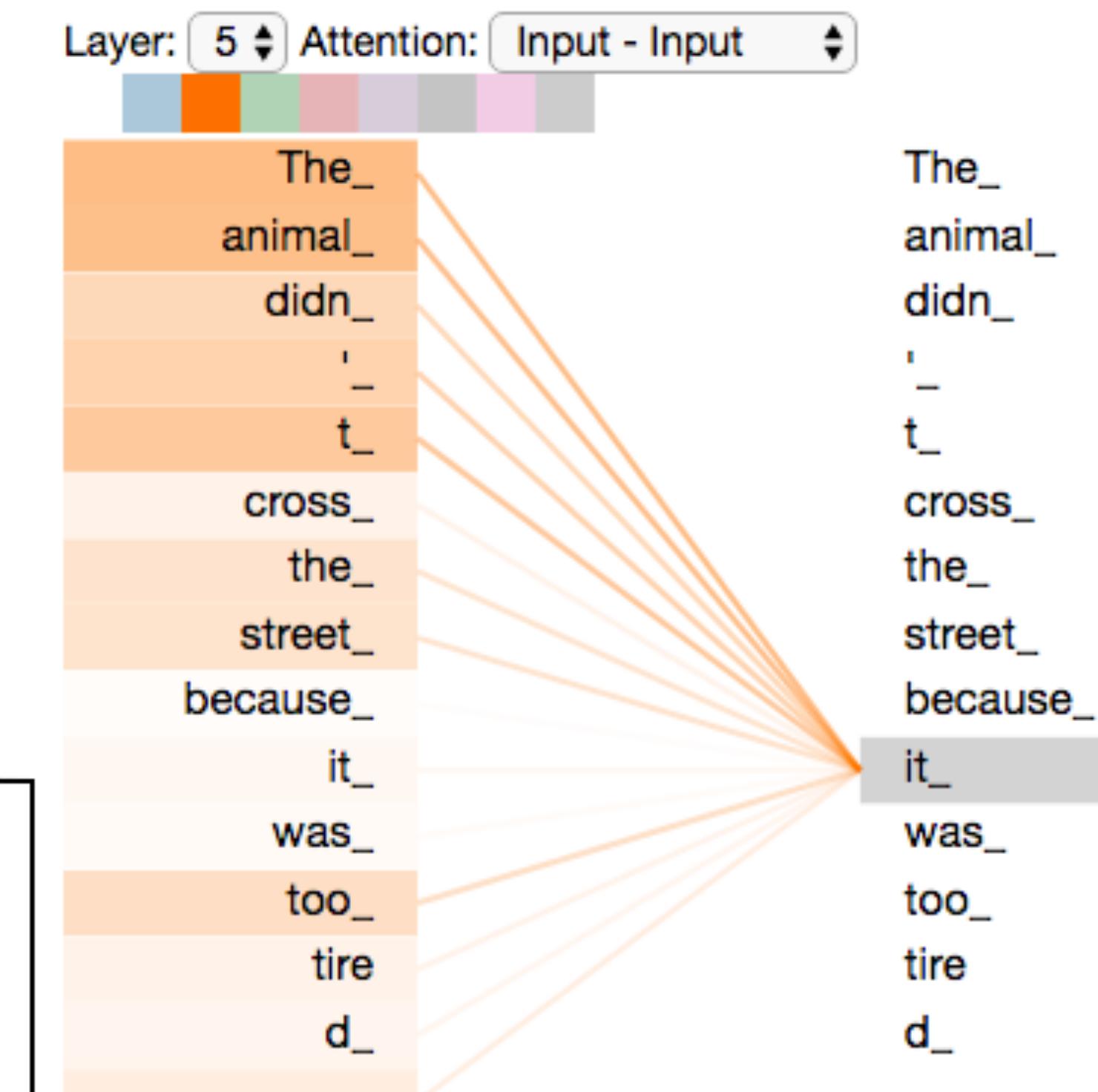
Transformer: self-attention

$$\text{attention} = \text{softmax}\left(\frac{QK^T}{d}\right)V$$

Input
 Embedding
 $q_1 = W_Q x_1$ Queries
 $k_1 = W_K x_1$ Keys
 $v_1 = W_V x_1$ Values
 Score
 Divide by 8 ($\sqrt{d_k}$)
 Softmax



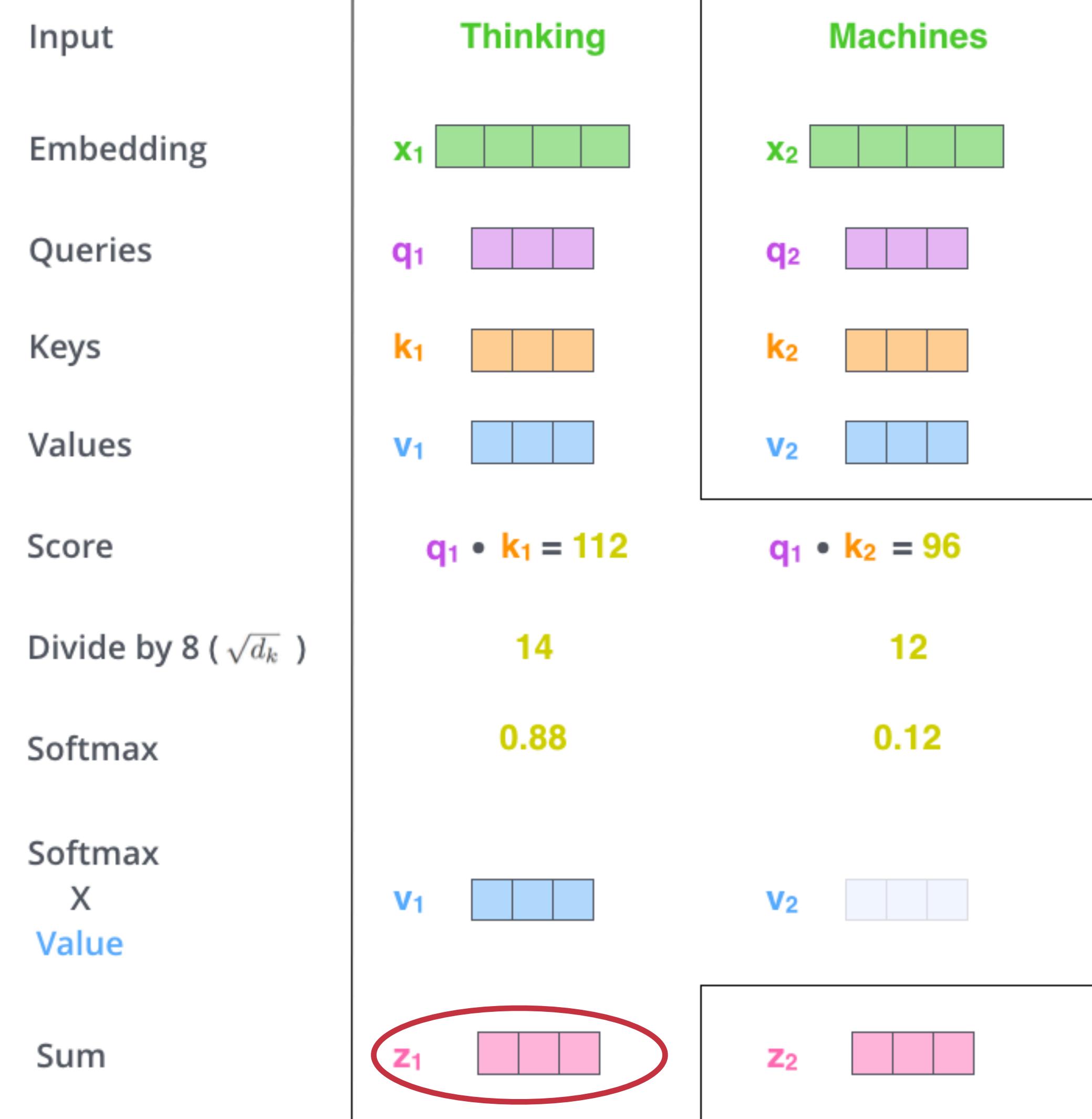
Для каждого x_i получаем веса, насколько он соотносится с x_j - всеми элементами входа



Transformer: self-attention

$$\text{attention} = \text{softmax}\left(\frac{QK^T}{d}\right)V$$

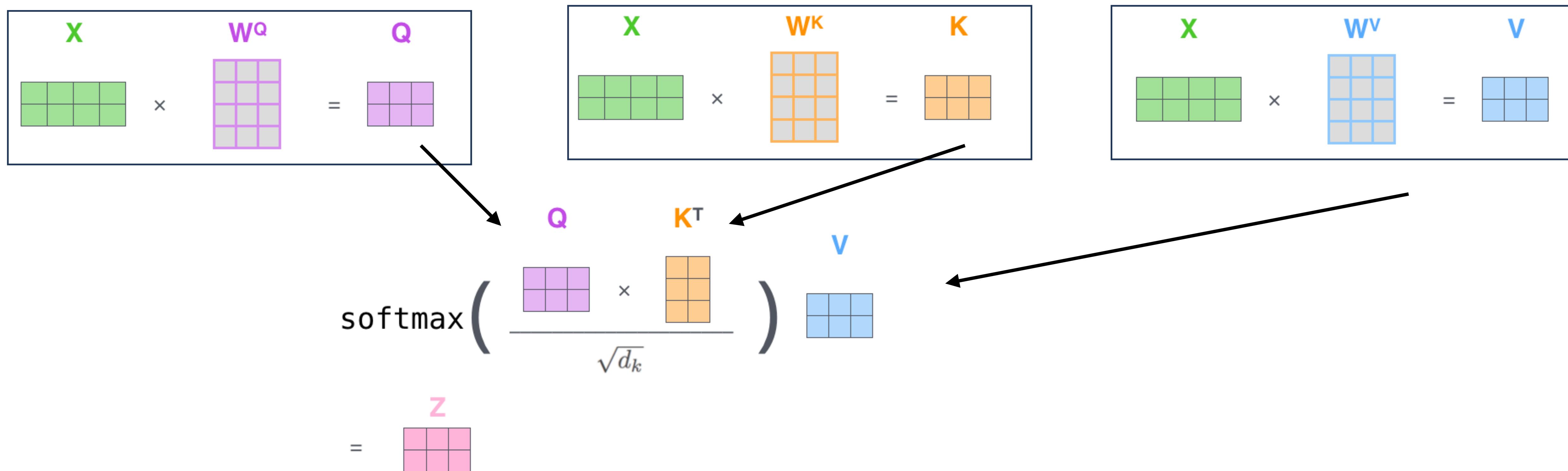
Взвешенная сумма всех v_j ,
веса - из softmax



Transformer: self-attention

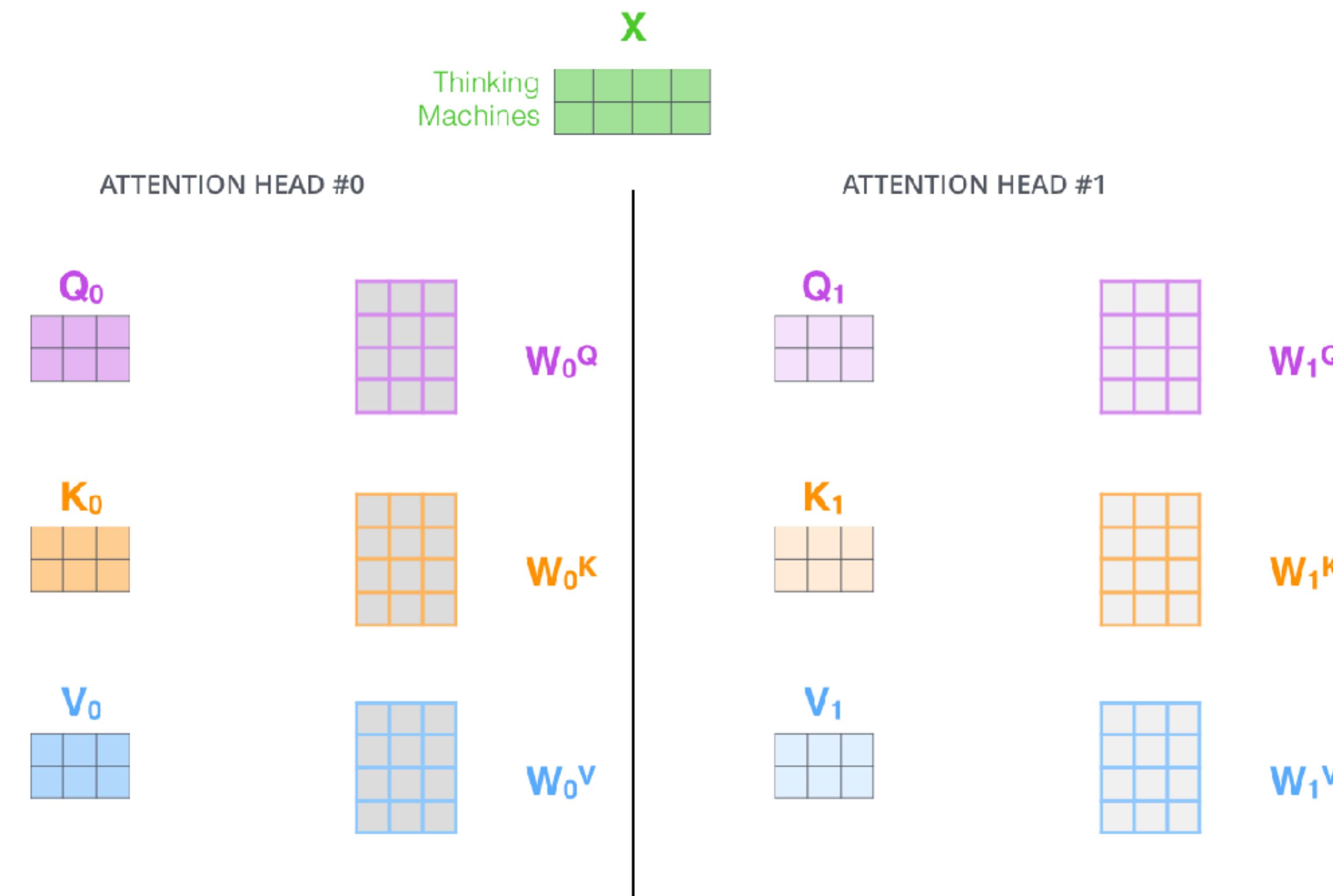
$$\text{attention} = \text{softmax}\left(\frac{QK^T}{d}\right)V$$

Матричная запись



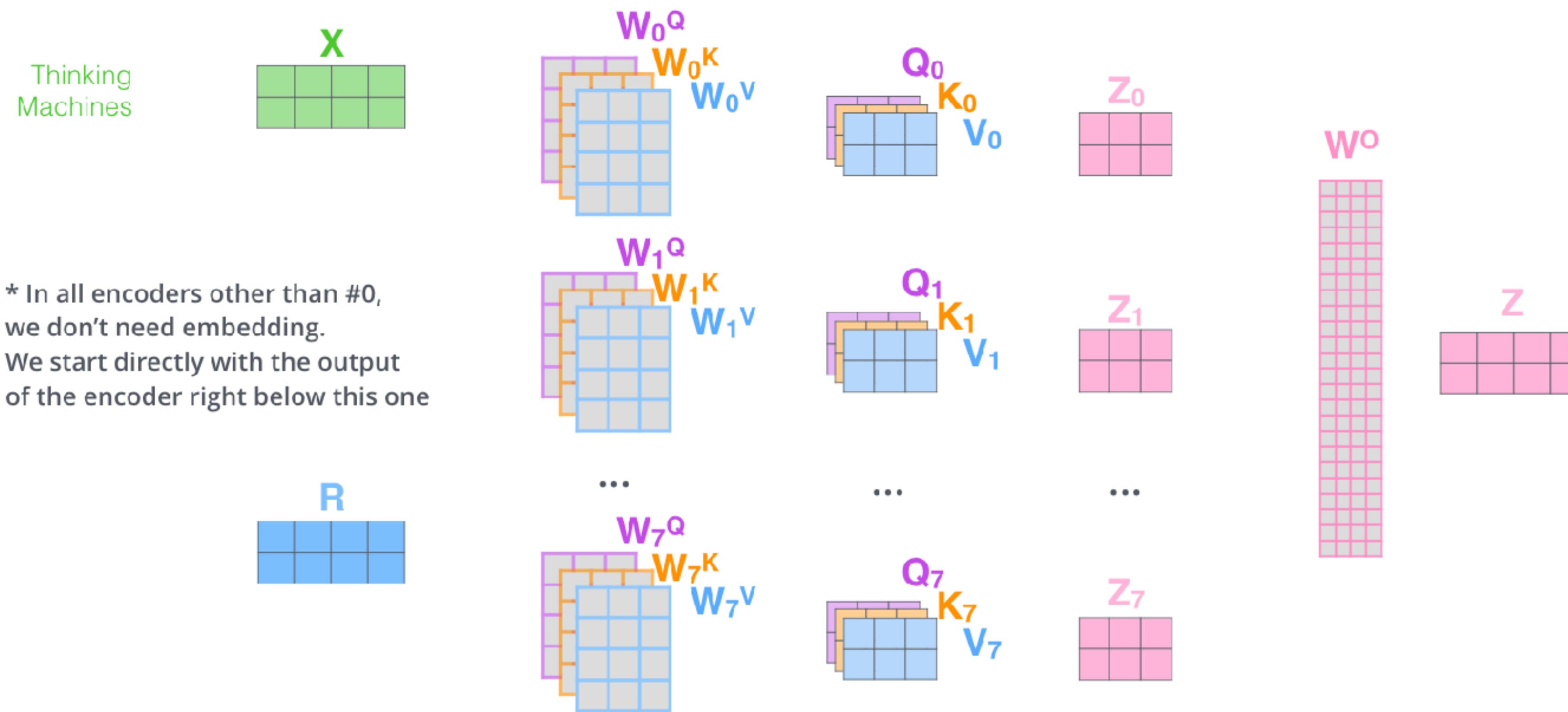
Transformer: multihead self-attention

Используем несколько self-attention слоев сразу - с разными весами



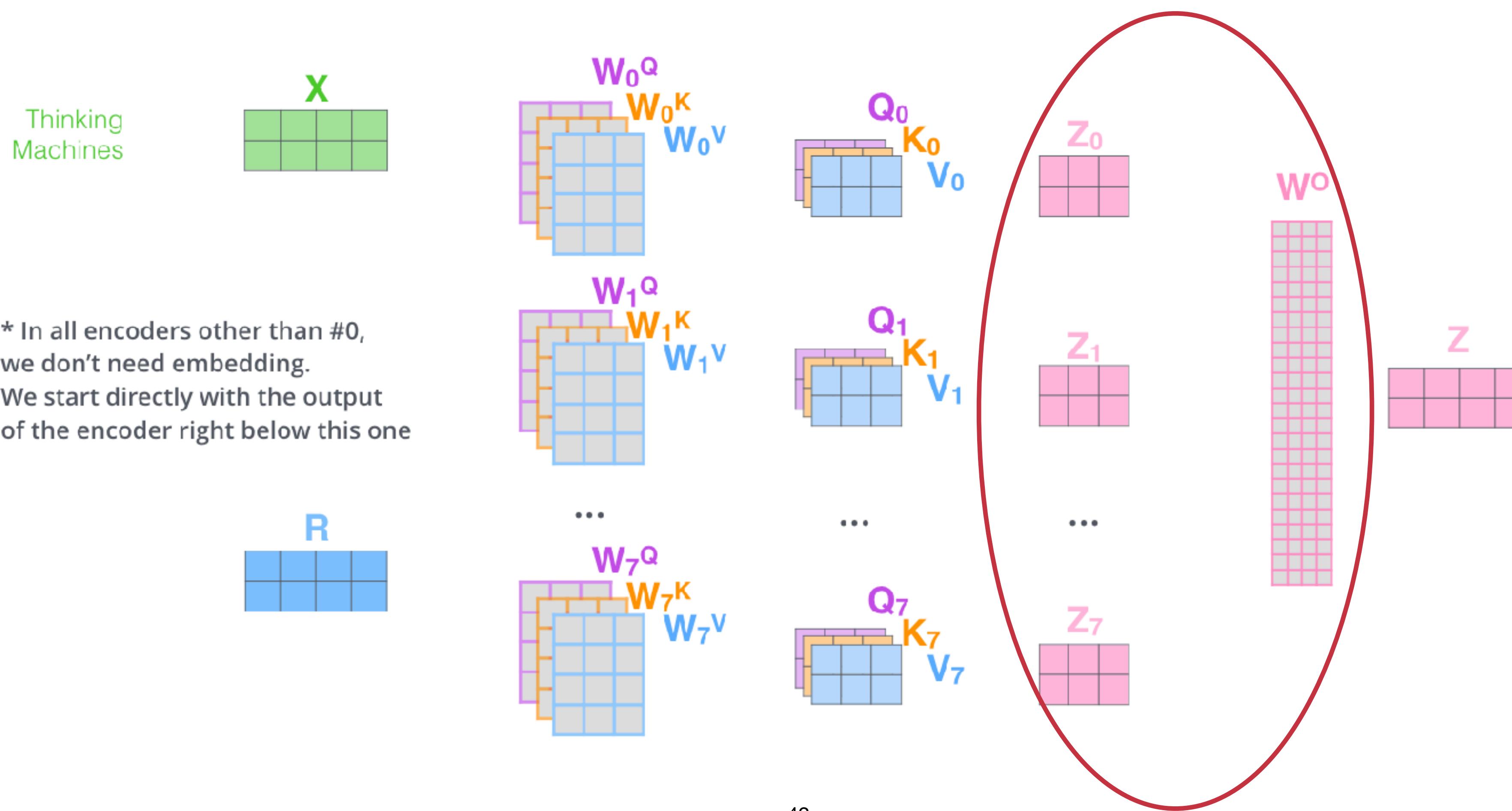
Transformer: multihead self-attention

Используем несколько self-attention слоев сразу - с разными весами



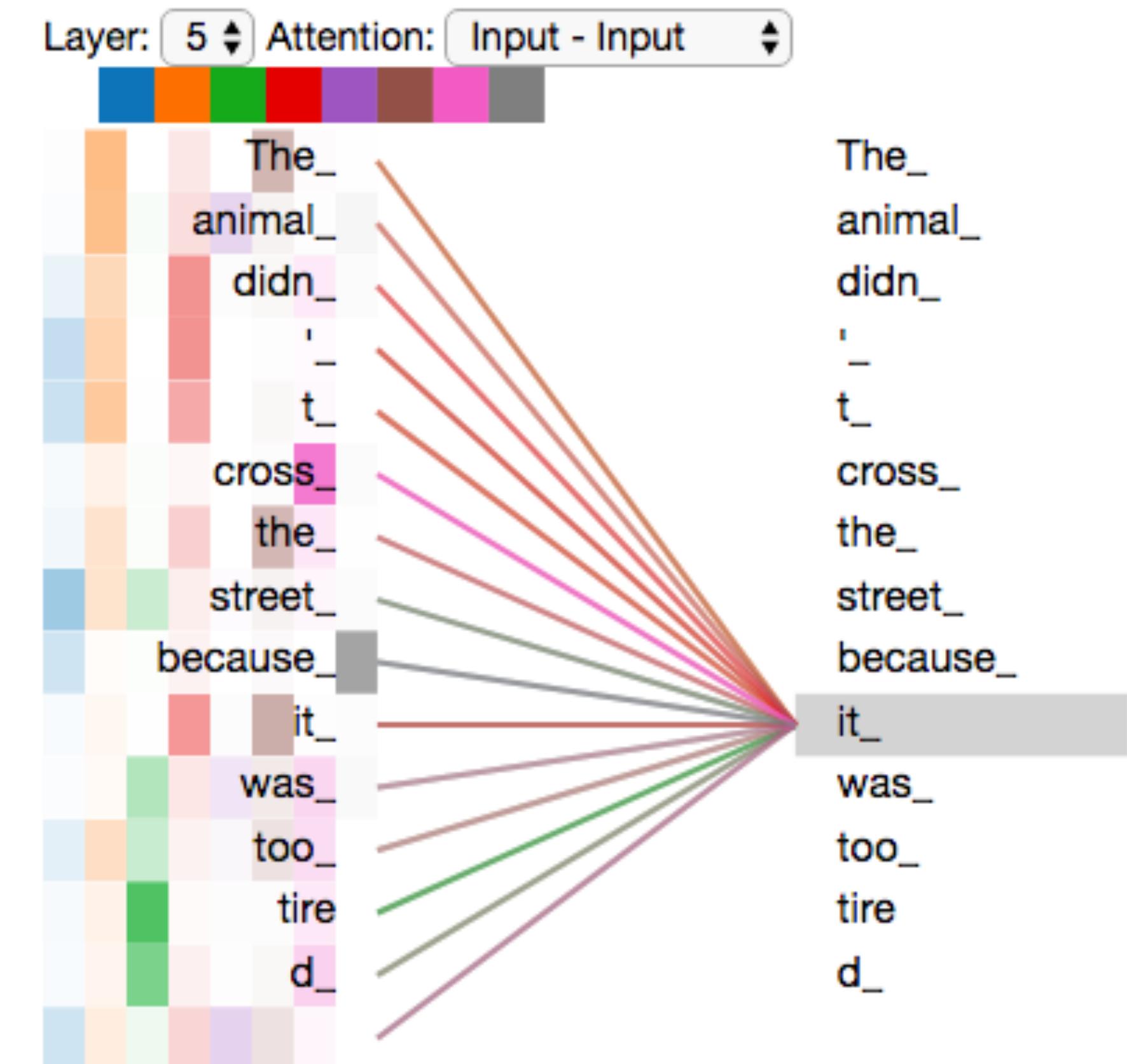
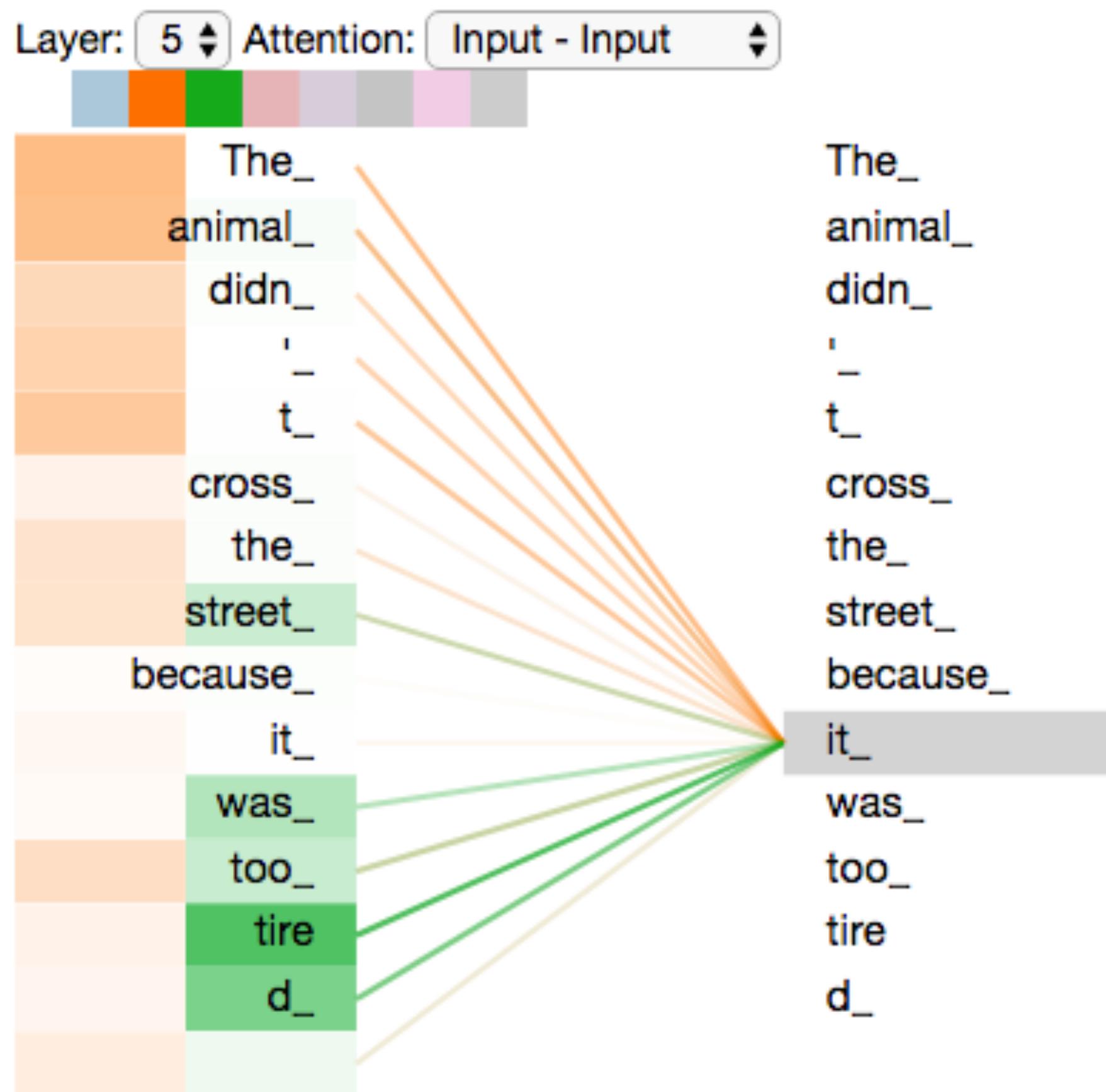
Transformer: multihead self-attention

Используем несколько self-attention слоев сразу - с разными весами



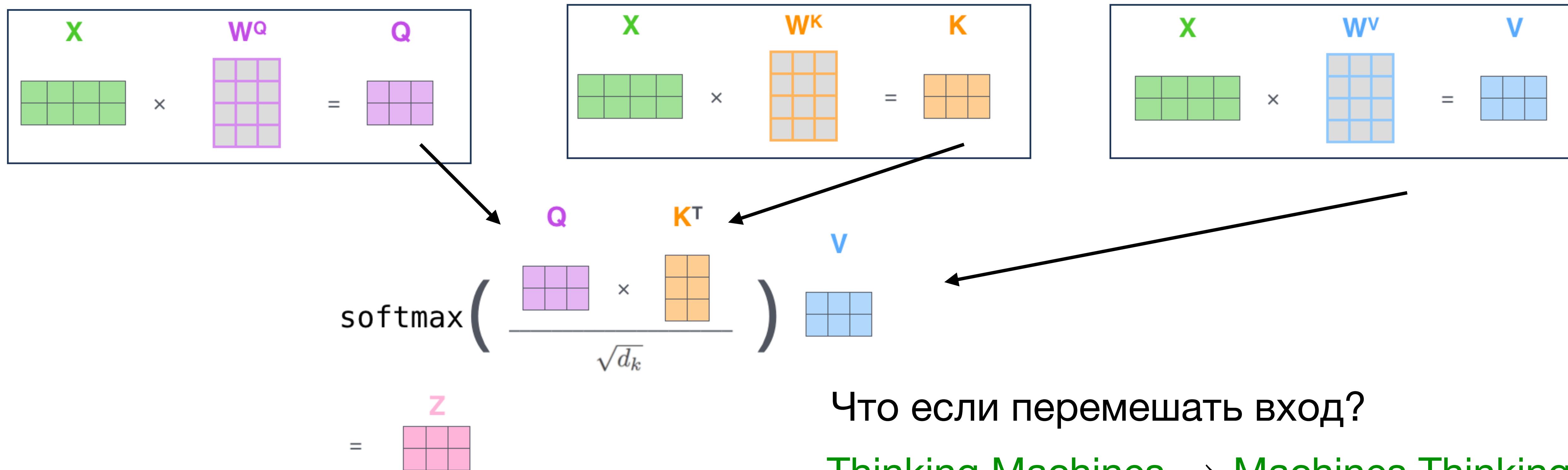
Transformer: multihead self-attention

Мотивация: разные self-attention “обращают внимание” на разные признаки

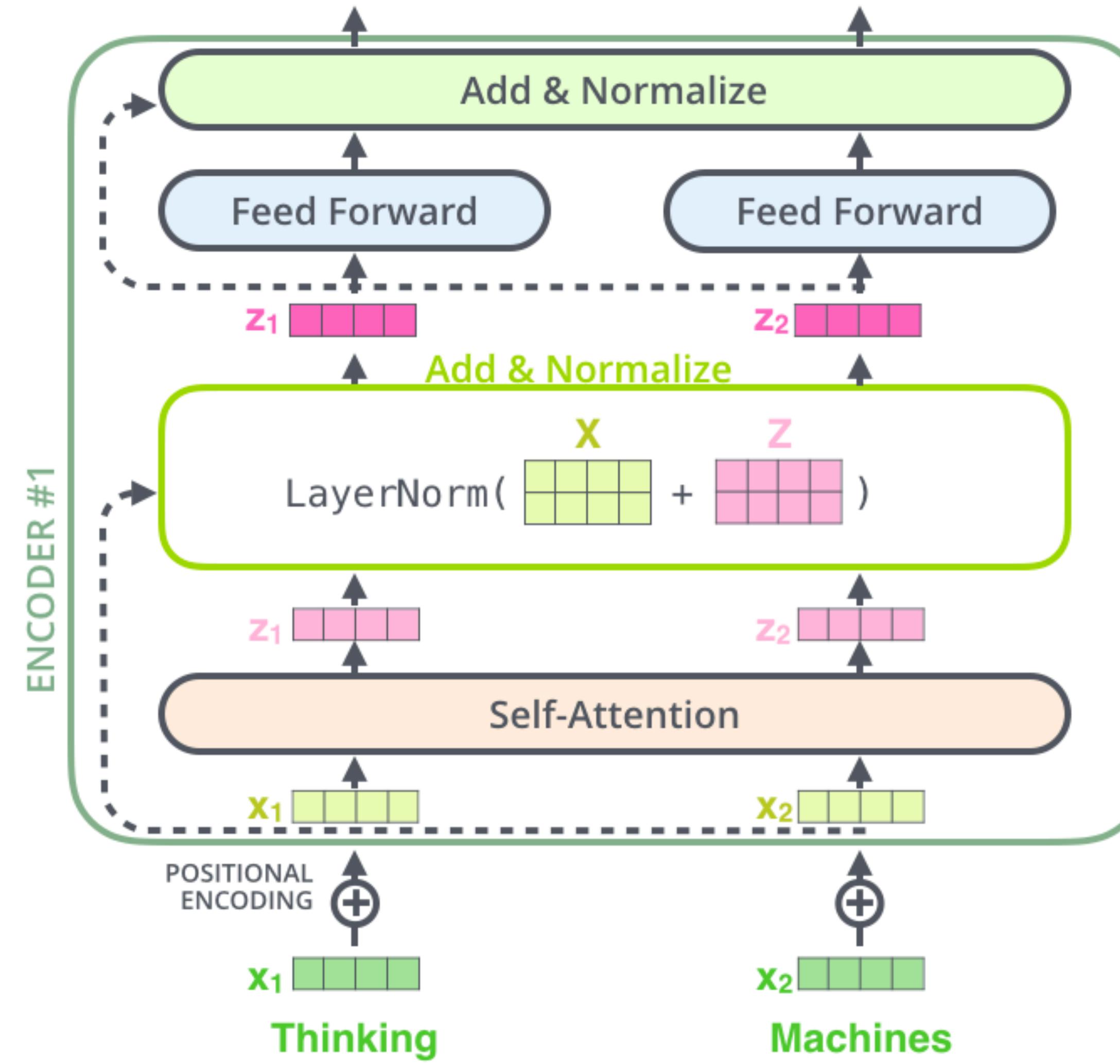


Transformer: multihead self-attention

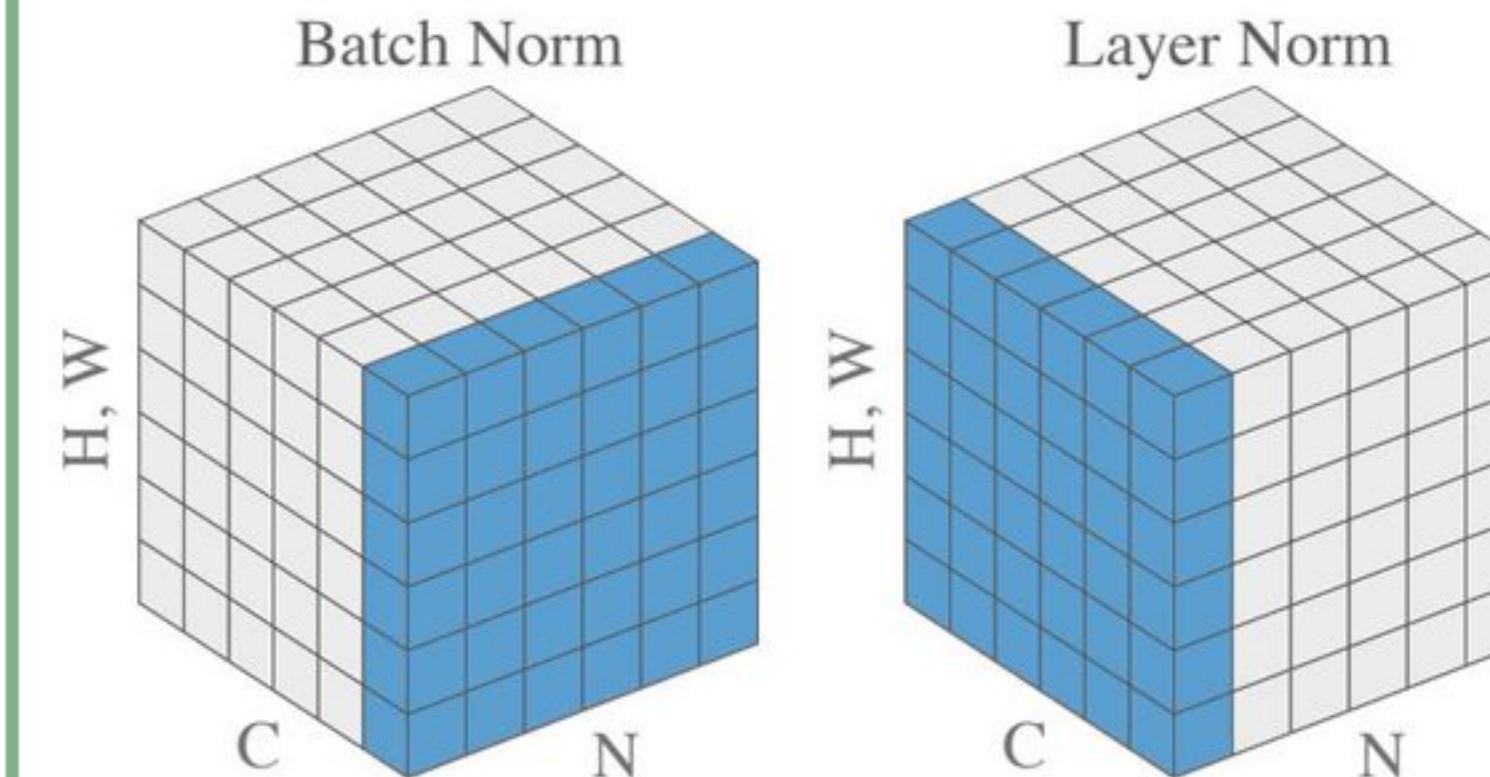
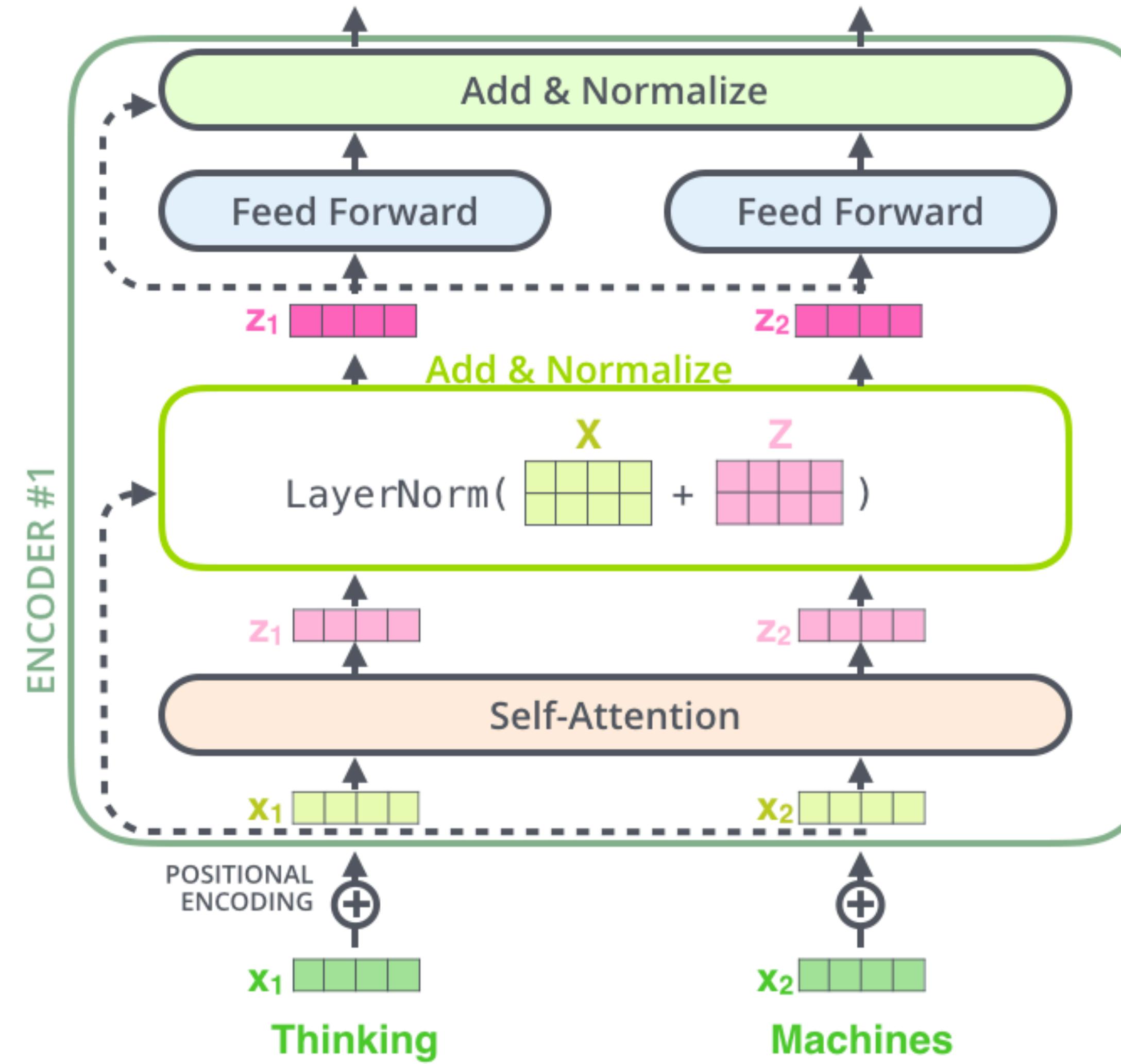
$$\text{attention} = \text{softmax}\left(\frac{QK^T}{d}\right)V$$



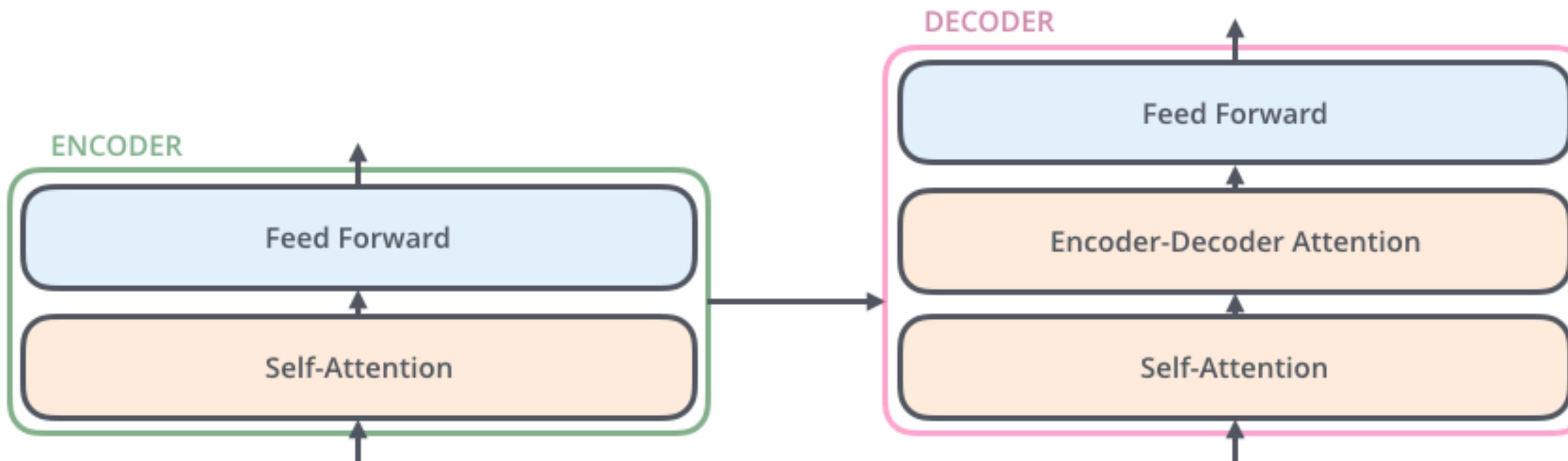
Transformer: Residuals & LayerNorm



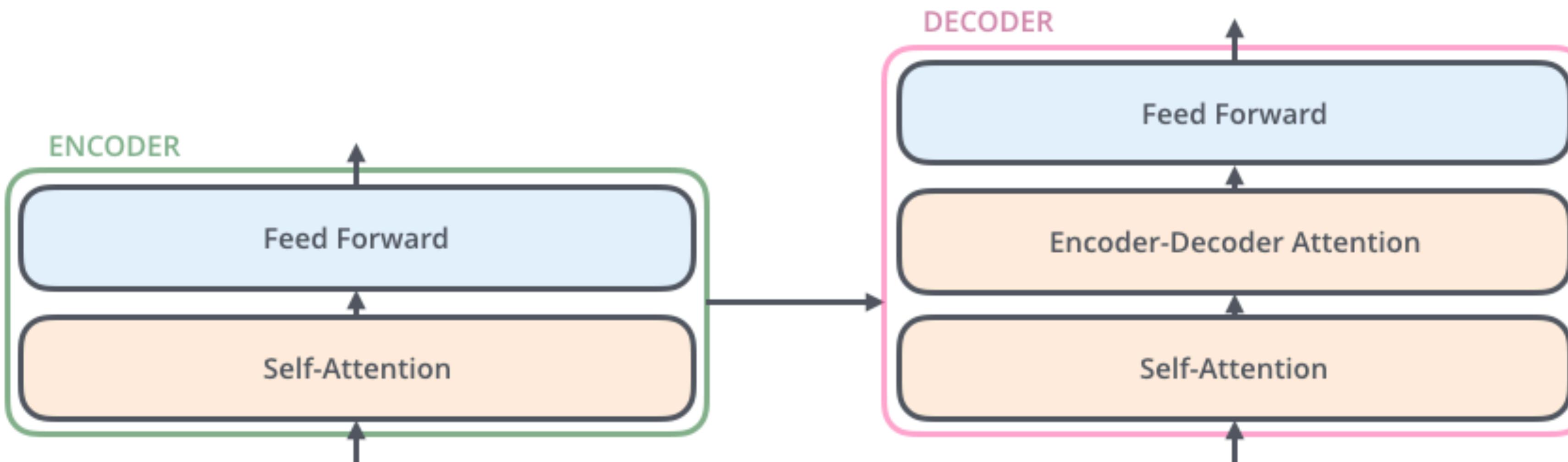
Transformer: Residuals & LayerNorm



Transformer: encoder-decoder attention

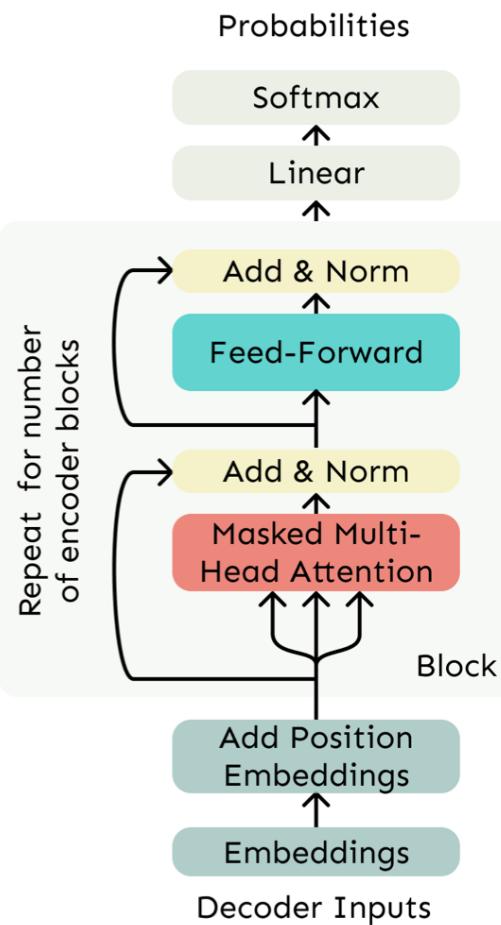


Transformer: encoder-decoder attention



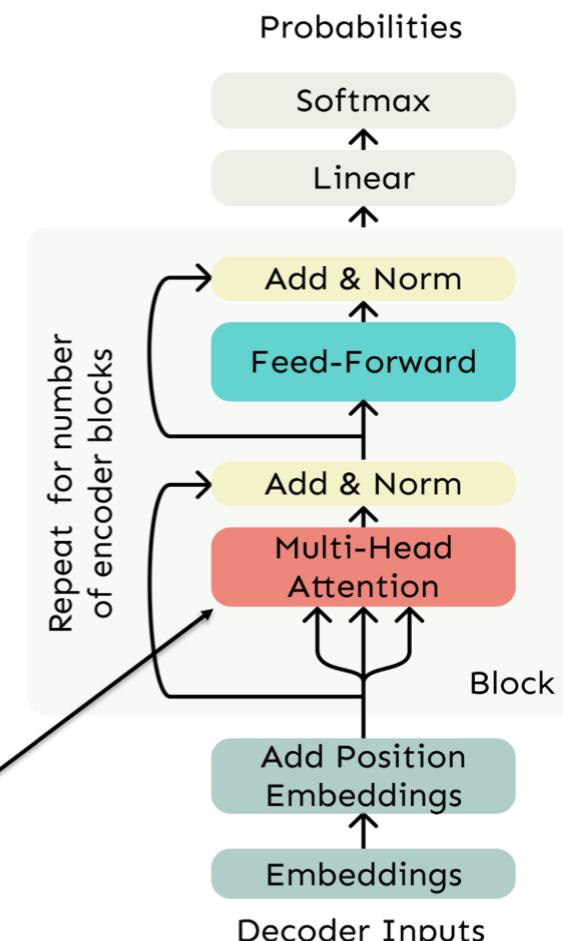
$$\text{Enc-Dec attention} = \text{softmax}\left(\frac{Q_{decoder}K_{encoder}^T}{d}\right)V_{encoder}$$

Decoder



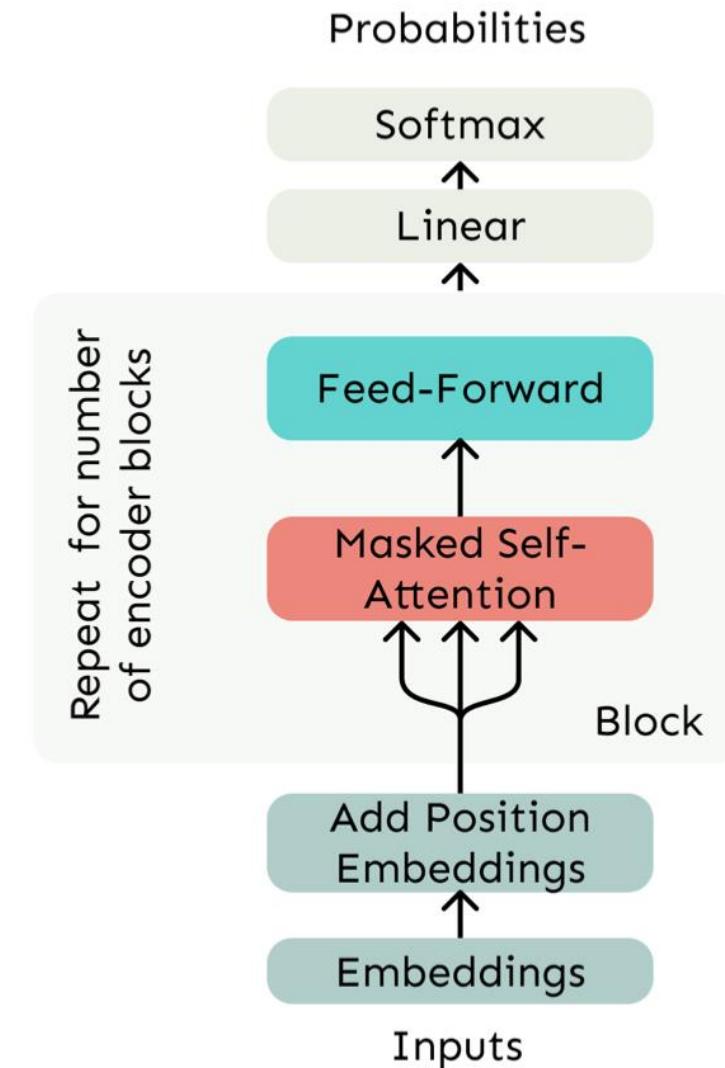
- Блоки:
 - Self-attention
 - Add & Norm
 - Feed-Forward
 - Add & Norm
- Decoder ограничивается односторонним контекстом
- Encoder позволяет учитывать контекст по обе стороны от слова, для этого в self-attention убирается маскирование

Encoder



Обоснование архитектуры трансформера

- Position representations:
Специфицируют порядок последовательности, так как Self-attention не структурирует входы
- Self-attention:
Основа метода: разделение на три матрицы позволяет обратить внимание модели на различные особенности структуры эмбедингов
- Nonlinearities:
Часто implementируют в виде полносвязной нейронной сети + softmax, позволяет улавливать нелинейные взаимосвязи
- Masking:
Позволяет параллелизовать операции
Ограничивает возможность смотреть «в будущее», то есть препятствует data leakage



Вычислительная сложность

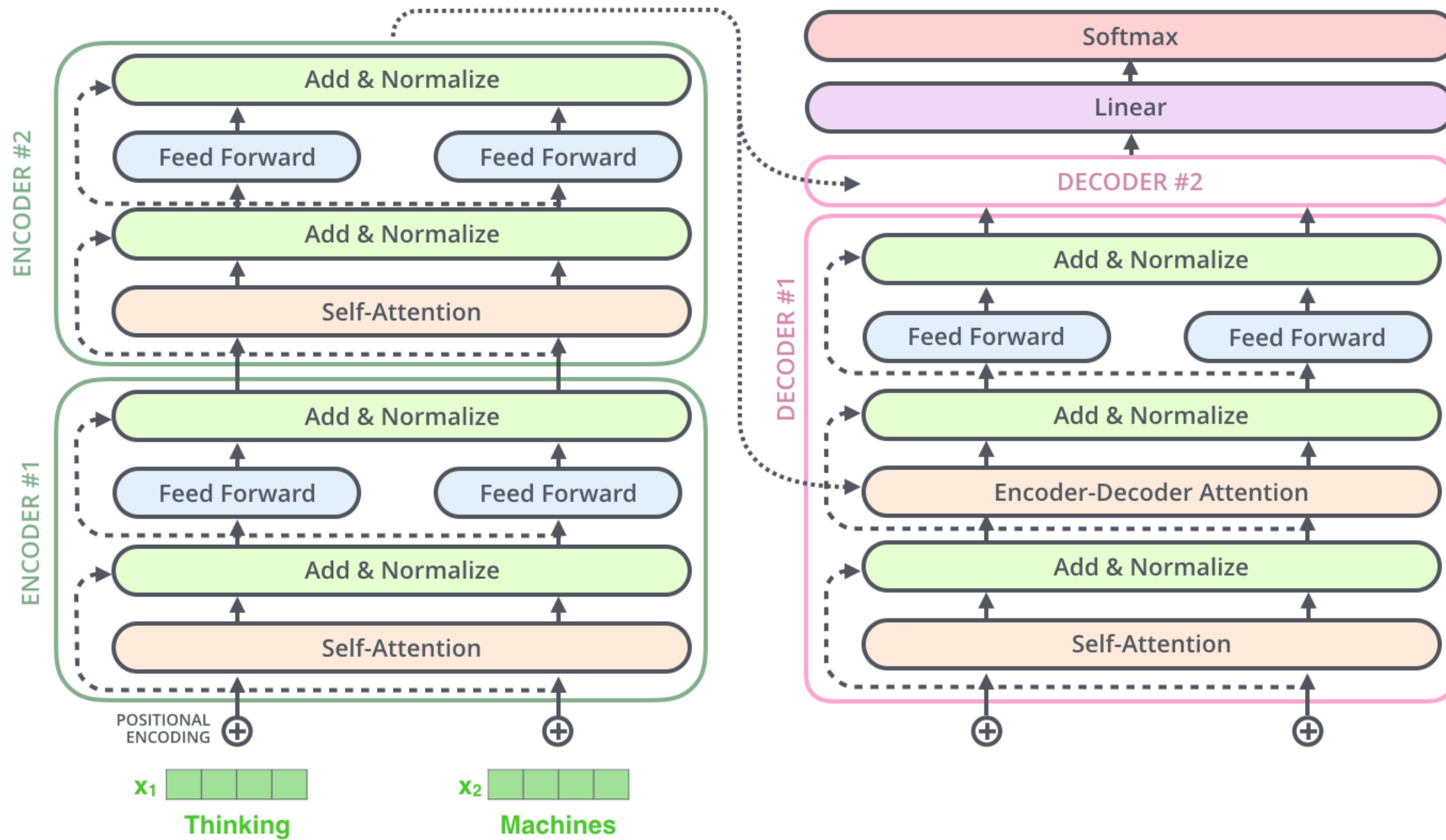
$$\begin{matrix} XQ \\ K^\top X^\top \end{matrix} = \boxed{XQK^\top X^\top} \in \mathbb{R}^{n \times n}$$

Need to compute all pairs of interactions!
 $O(n^2d)$

Таким образом сложность квадратично зависит от длины предложения и линейно от размера векторов эмбеддингов

Поэтому модели на основе трансформеров обучаются намного медленнее обучаются, чем те же LSTM

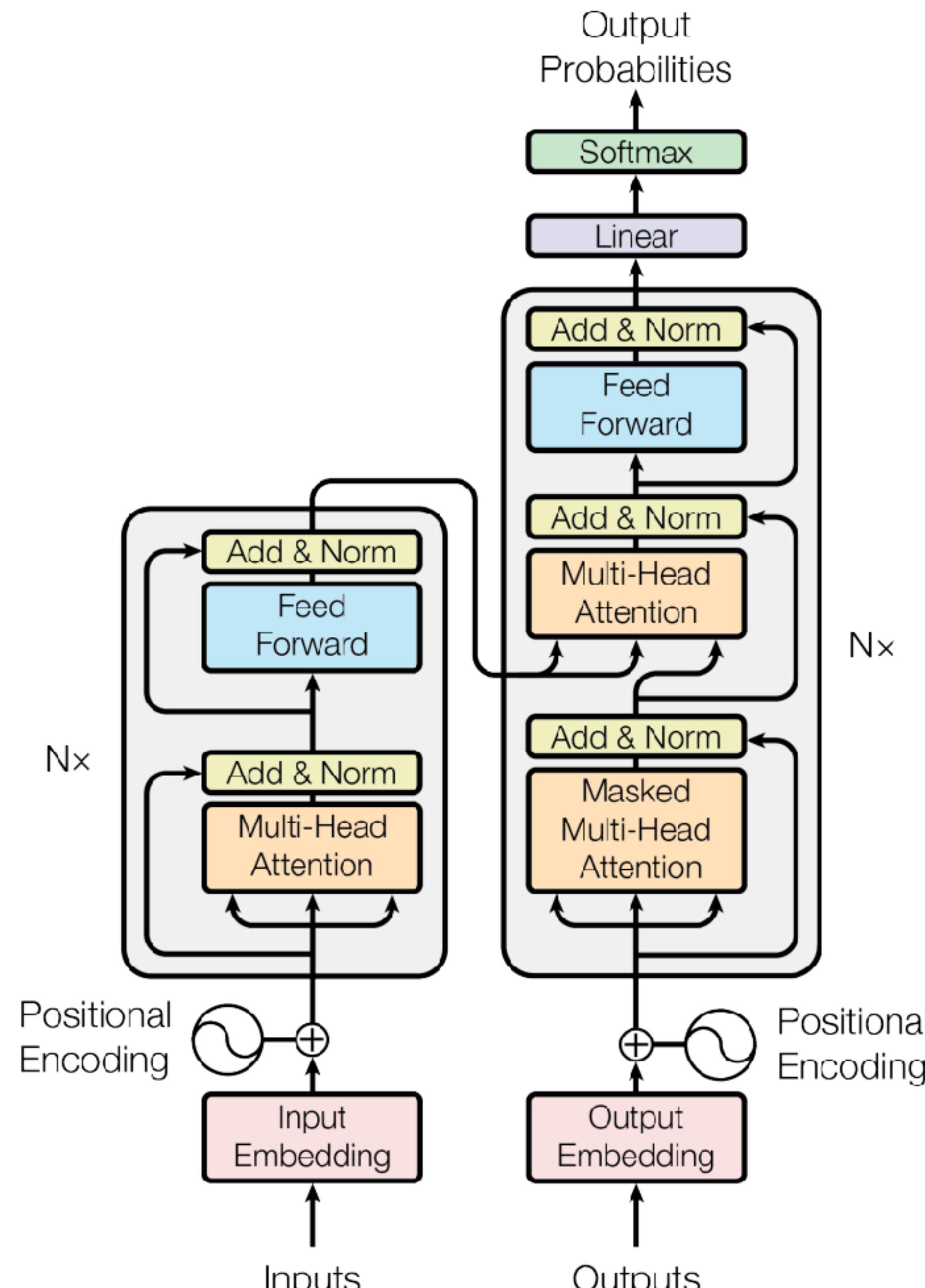
Transformer



Transformer

BERT

Encoder



GPT

Decoder