Архитектура трансформер

Трус Владлена

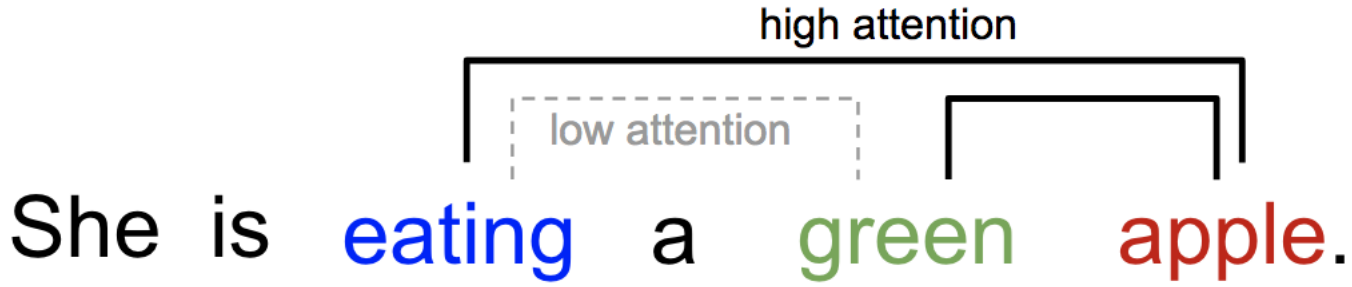
БПМИ172

Для чего нужна архитектура Трансформер

- Перевод с одного языка на другой
- Прогнозирование временных рядов
- Constituency parsing (грамматический разбор)

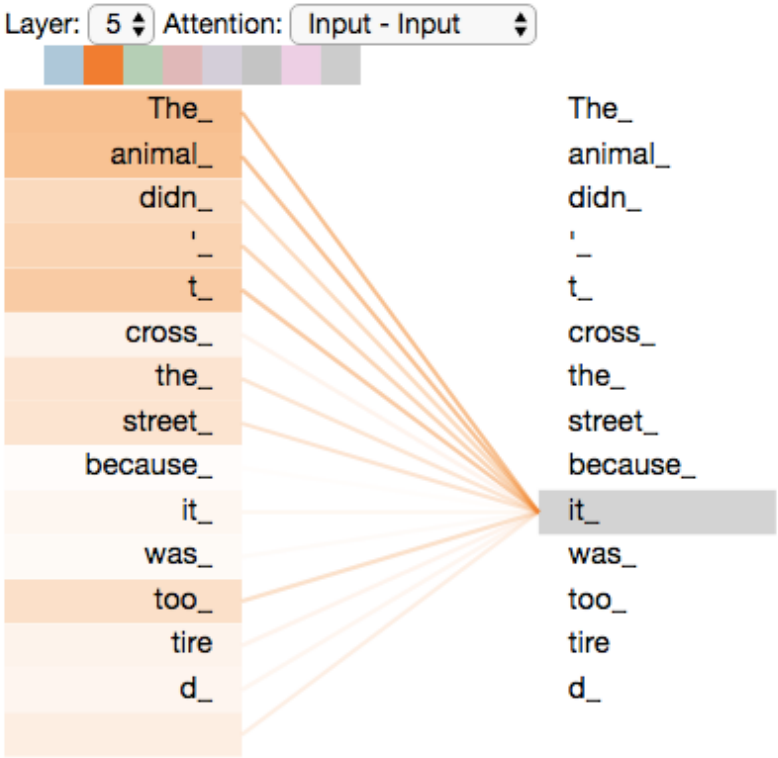
Основная идея Attention

Мы можем объяснить связь между словами в одном предложении или в близком контексте. Когда мы видим слово «едят», мы ожидаем, что очень скоро встретимся со словом обозначающем пищу. Термин цвета также описывает еду, но, вероятно, не так, как «еда» напрямую.

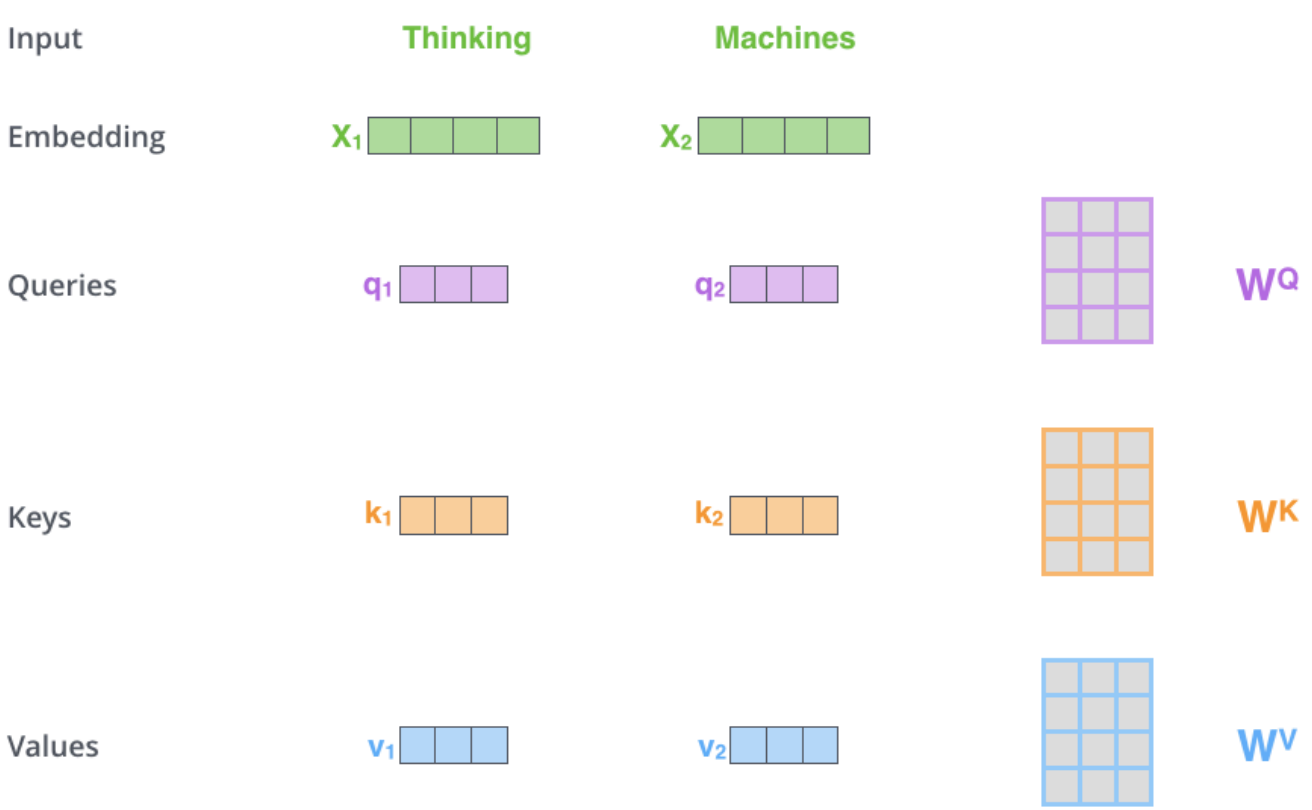


Self-Attention

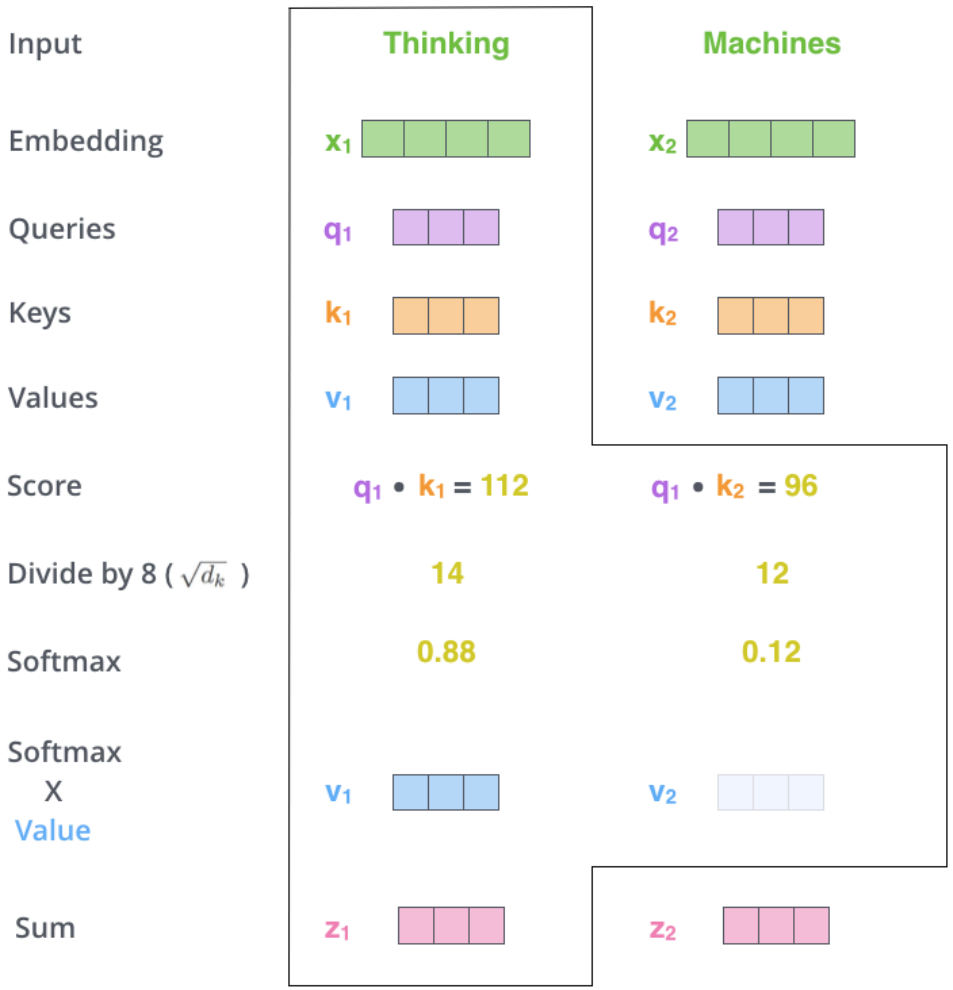
Ниже механизм self-attention позволяет нам узнать корреляцию между текущими словами и предыдущей частью предложения.

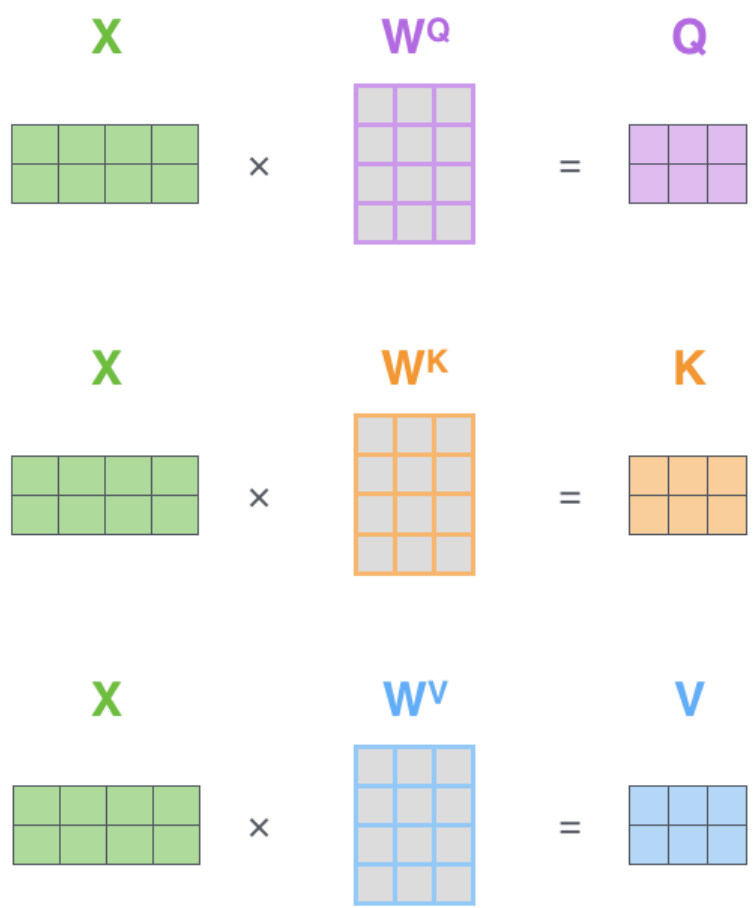


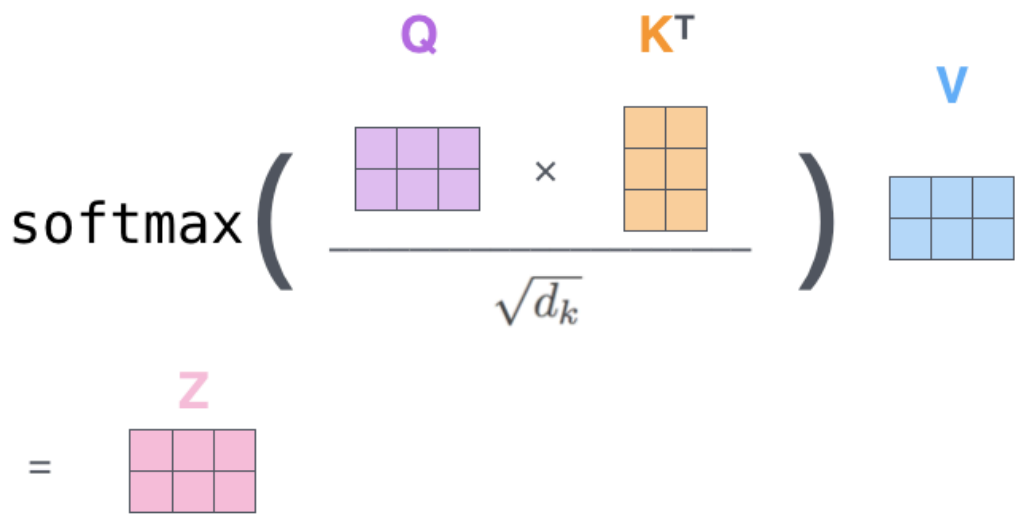
Self-attention



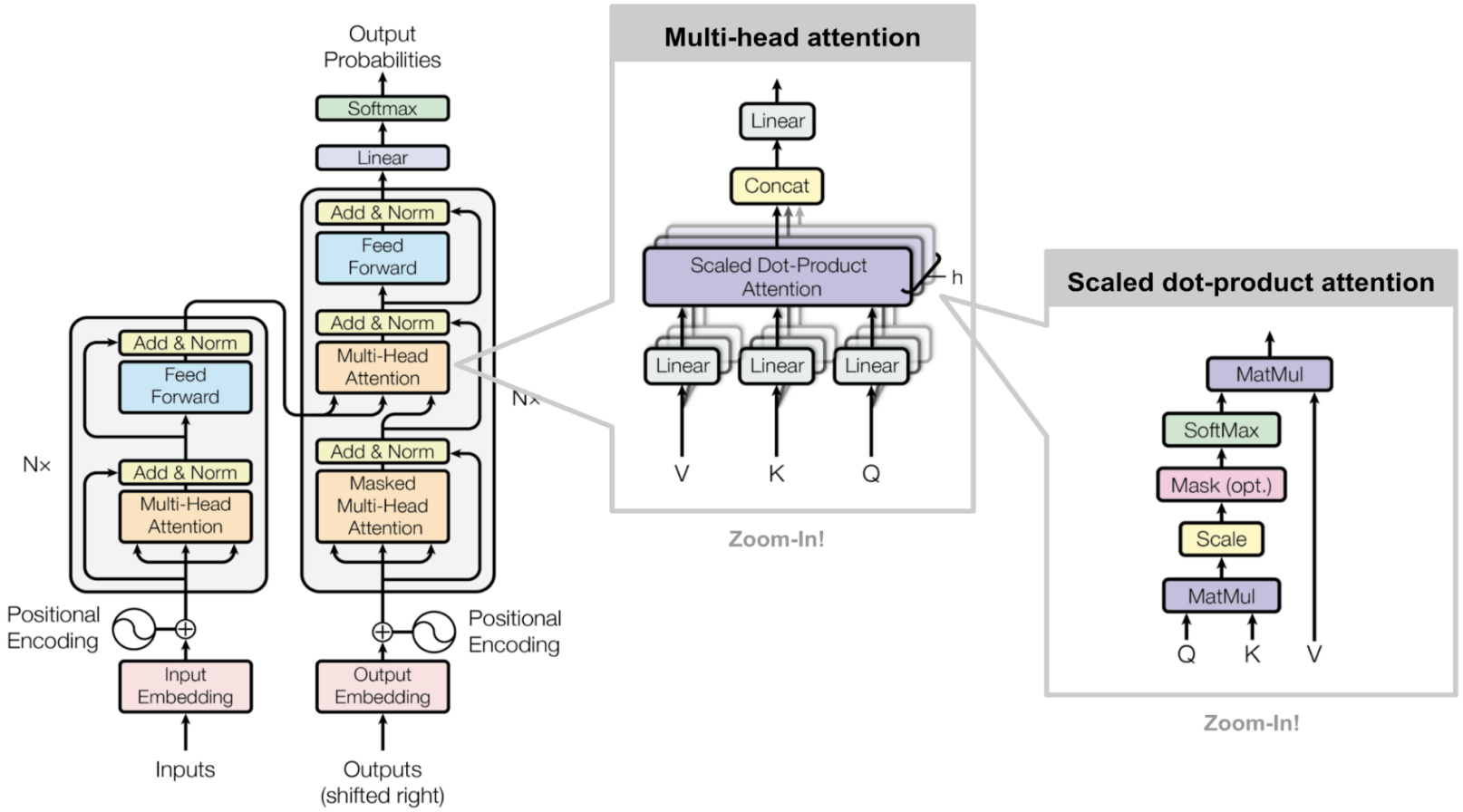
Self-attention



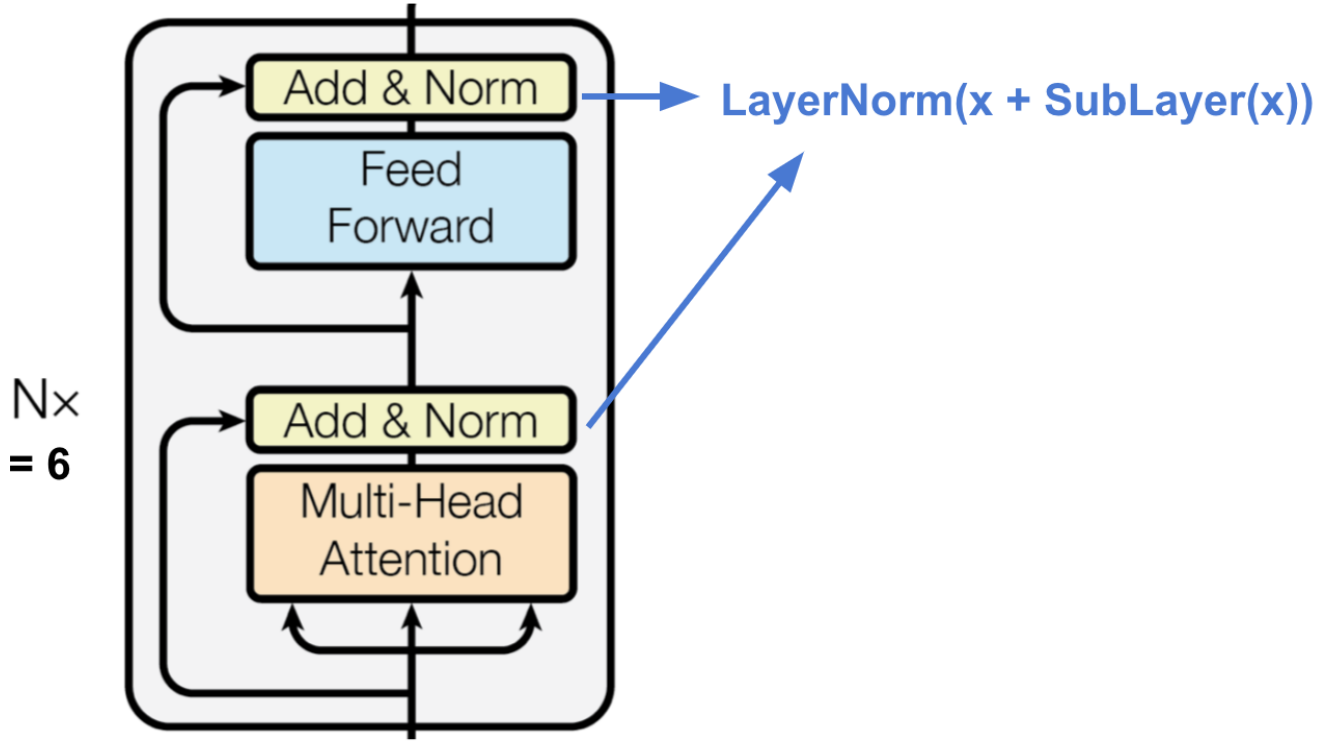

X
W^Q
Q
×
=
X
W^K
K
×
=
X
W^V
V
×
=


Q
K^T
V
softmax
×
$\sqrt{d_k}$
Z
=

Transformer

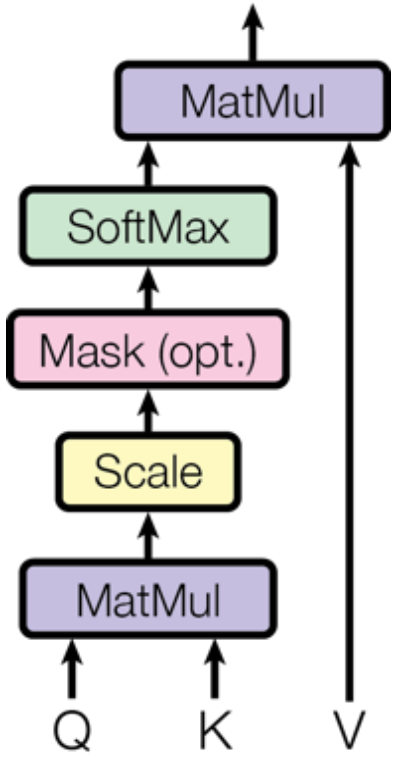


Encoder



- Стек из N = 6 одинаковых слоев.
- Каждый слой имеет **multi-head self-attention** слой и простую **feed-forward полносвязную сеть**.
- Каждый подслой принимает **residual** соединение и **нормализирующий** слой. Все субслои выводят данные одной и той же размерности d=512

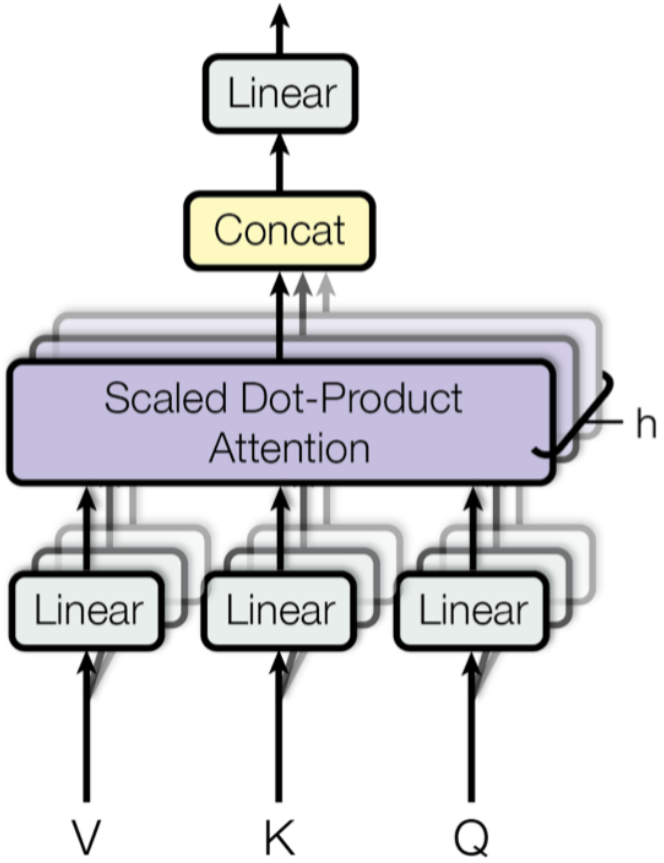
Scaled Dot-Product Attention



Выход - взвешенная сумма значений, где вес, назначенный каждому значению, определяется почленным произведением запроса со всеми ключами:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

Multi-head attention

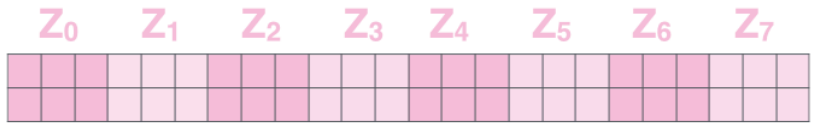


Вектор attention вычисляется параллельно через scaled dot-product attention h раз. Результаты независимого внимания просто объединяются и линейно преобразуются в ожидаемую размерность.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

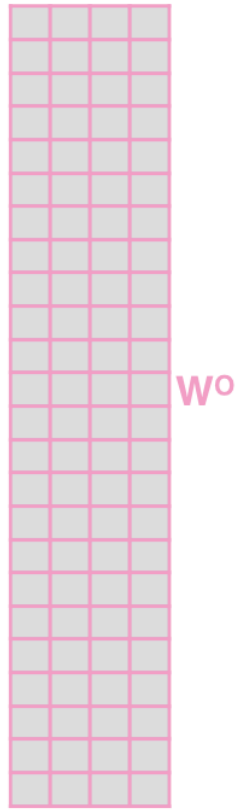
Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

1) Concatenate all the attention heads



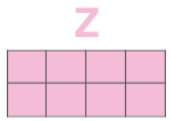
2) Multiply with a weight matrix W^O that was trained jointly with the model

X

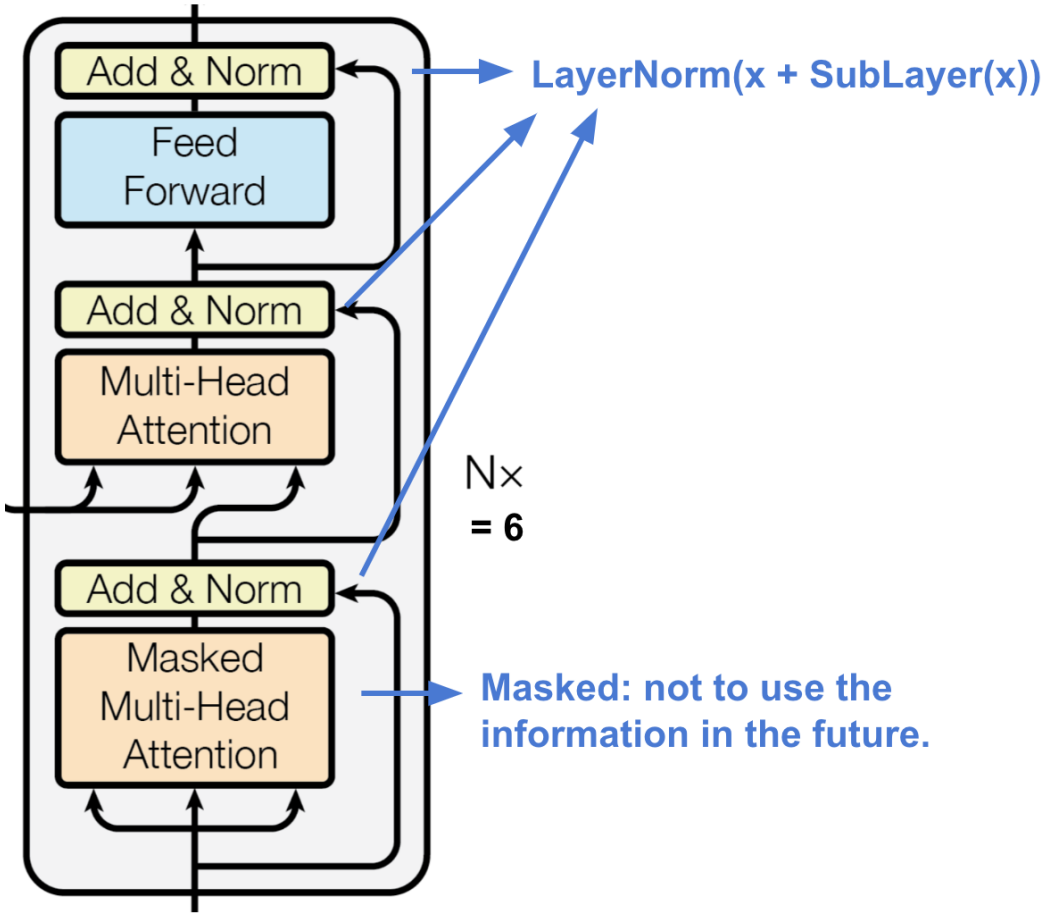


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



Decoder



- Стек из N = 6 одинаковых слоев
- Каждый слой имеет два подслоя реализующего механизм multi-head attention и один подслой полносвязной feed-forward сети.
- Каждый подслой содержит residual соединение и слой нормализации.
- Первый multi-head attention слой **модифицирован** таким образом, чтобы предотвратить посещение позиций следующих за текущей позицией, так как мы не хотим заглядывать в будущее целевой последовательности при прогнозировании текущей позиции.

Позиционное кодирование

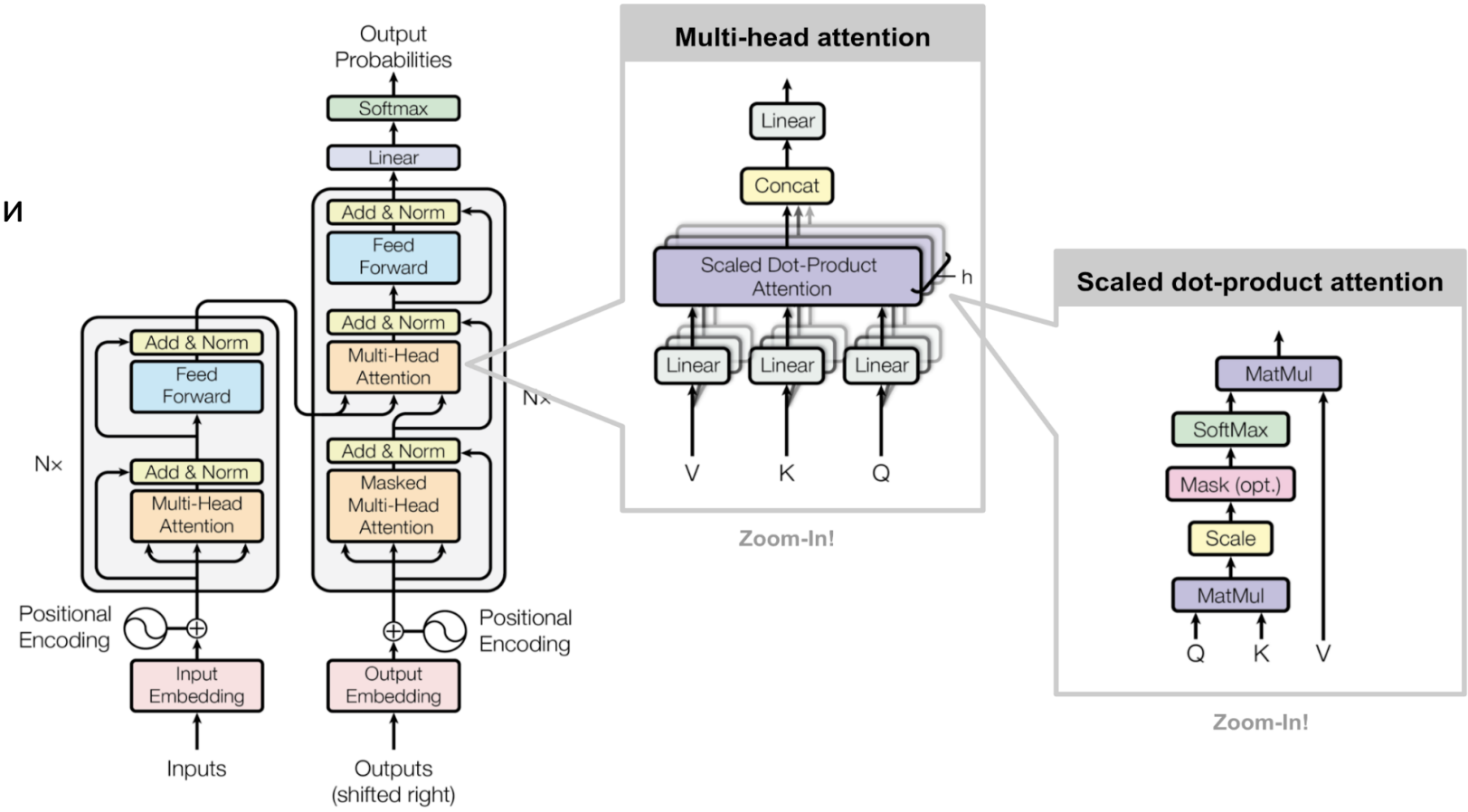
Используем, чтобы запомнить относительную или абсолютную позицию токена в последовательности.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

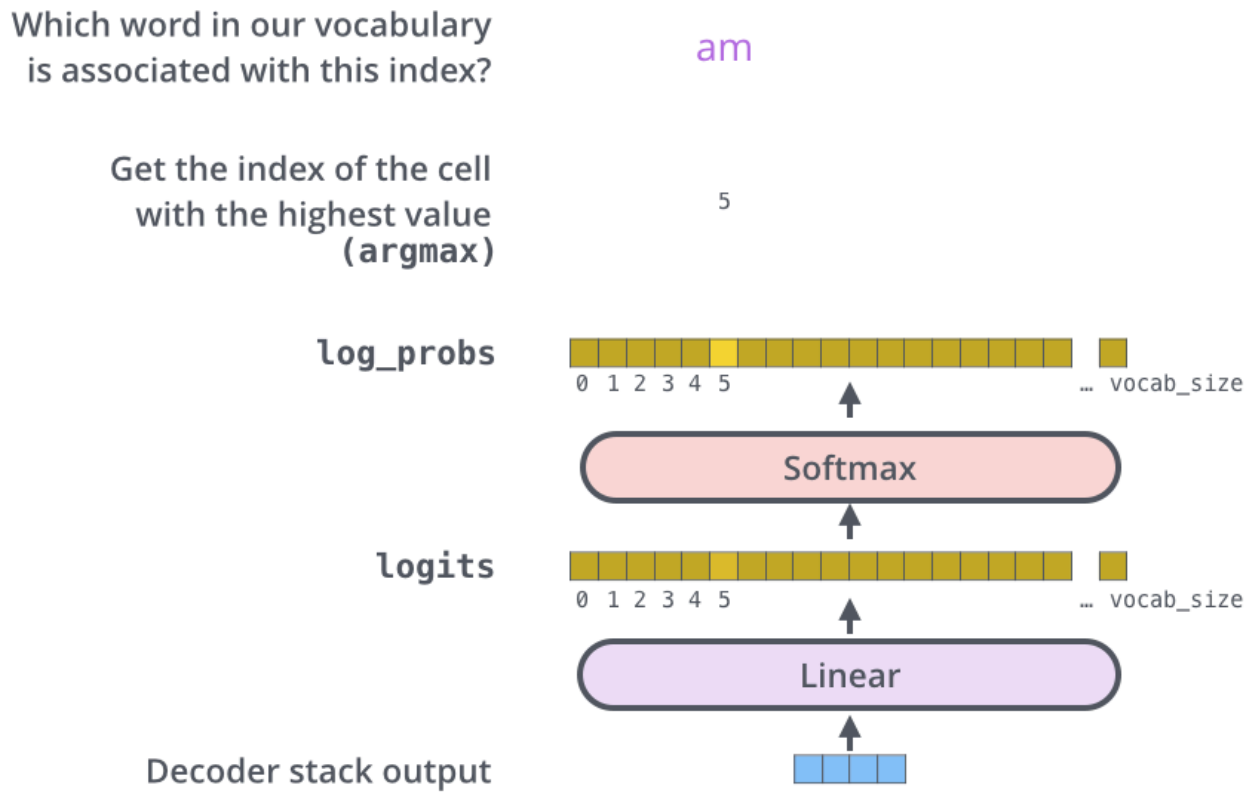
где pos - позиция, а i – размерность.

Transformer

- Обе последовательности исходные и целевые сначала проходят через embedding слои для получения данных одной и той же размерности d_model=512.
- Применяется позиционное кодирование, которое суммируется с embedding выходом
- Softmax и линейный слой добавляются к финальному выходу декодера.



Линейный слой



Регуляризация

Применяется dropout к выходу каждого подслоя, прежде чем он будет добавлен к входу подслоя и нормализован. Так же применяется dropout к суммам embeddings и позиционных кодировок в encoder и decoder.

Используется сглаживание меток, которое делает модель более «неуверенной», но повышает точность и BLEU.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Плюсы:

- Увеличился BLEU по сравнению с предыдущими методами.
- Обращаться к предыдущим словам просто
- Вычисления можно проводить параллельно

Минусы:

- Сложность реализации
- Attention добавляет больше весов в модель, что может увеличить время обучения

Вопросы

1) Расскажите кратко об архитектуре модели.

2) Scaled Dot-Product Attention: записать формулу для Attention, объяснить смысл.

3) Как в данной модели применяются эмбеддинги и softmax

4) Записать формулу Multi-Head и объяснить смысл.

Список источников

- https://arxiv.org/pdf/1706.03762.pdf
- http://jalammar.github.io/illustrated-transformer/
- https://habr.com/ru/post/458992/
- https://medium.com/@joealato/attention-in-nlp-734c6fa9d983
- https://pathmind.com/wiki/attention-mechanism-memory-network
- https://mchromiak.github.io/articles/2017/Sep/12/Transformer-Attention-is-all-you-need/#.XfLNly1ePjC
- http://artintelligence.ru/attention-attention/
- https://towardsdatascience.com/transformers-141e32e69591