SecretLLM

Attention is All You Need: The Transformer Architecture

Prof. Dr. Michael Färber

Focus

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

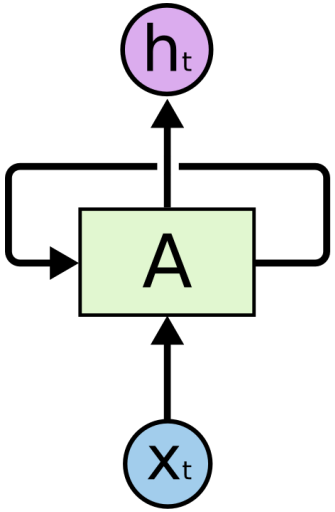
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

- Published 2017.
- Many many citations.
- Focus on attention.

Recap: Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN)



Recurrent Neural Networks have loops

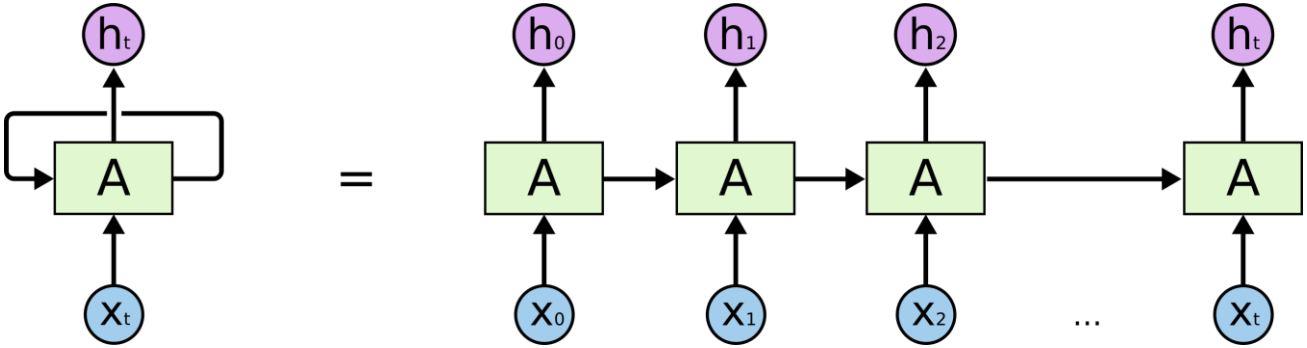
A:

- looks at some input x_t and
- outputs a value h_t

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Network (RNN)

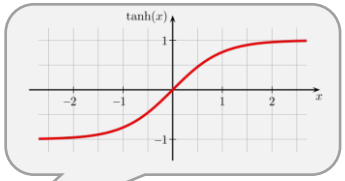
Unrolled recurrent neural network



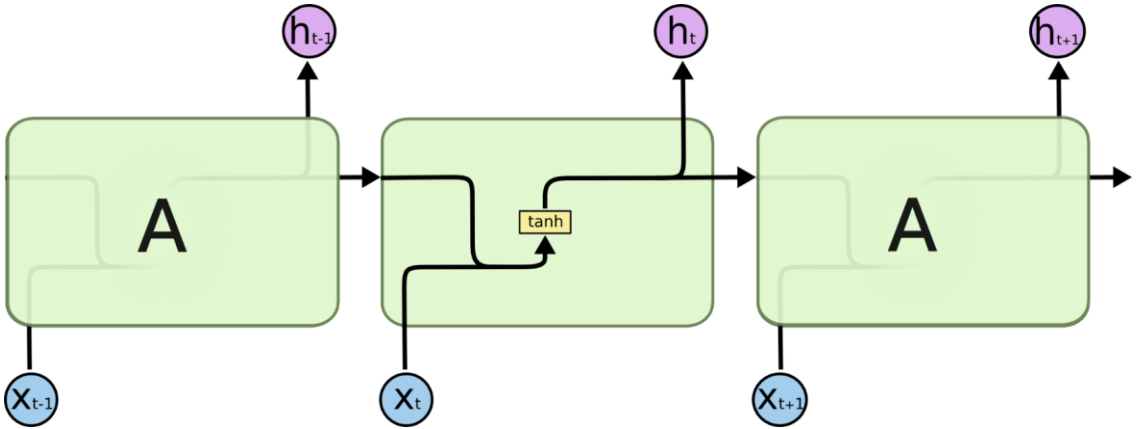
This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Network (RNN)



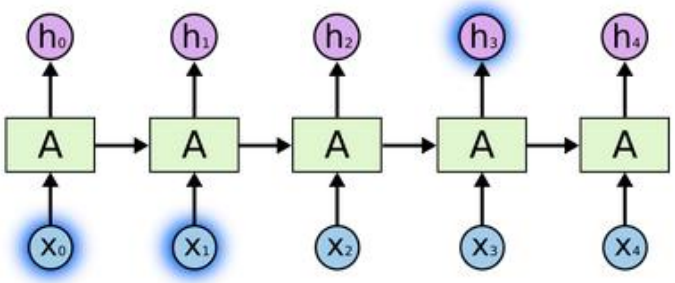
Standard RNN (e.g. with tanh layer)



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Network (RNN)

Problem of Long-Term Dependencies

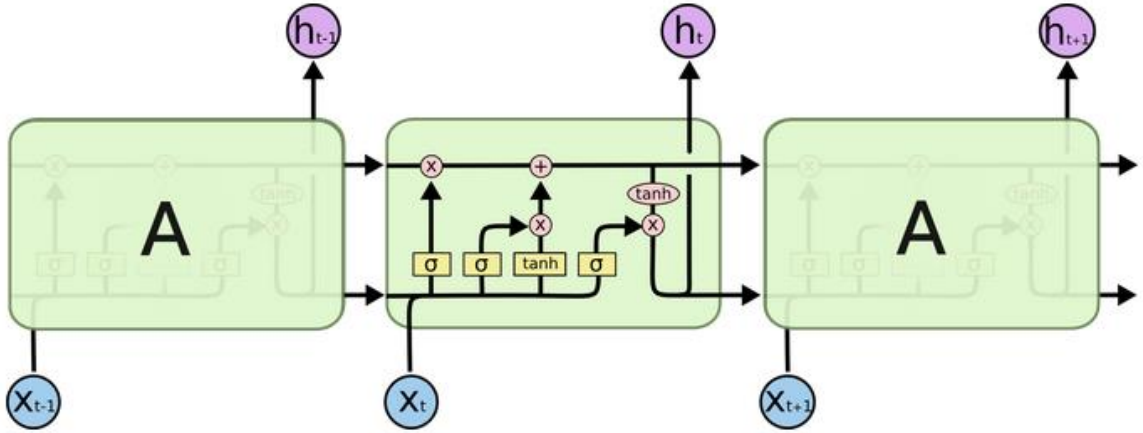


RNN do not capture long-term dependencies and fails to keep the context.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (LSTM)

Repeating module in LSTM

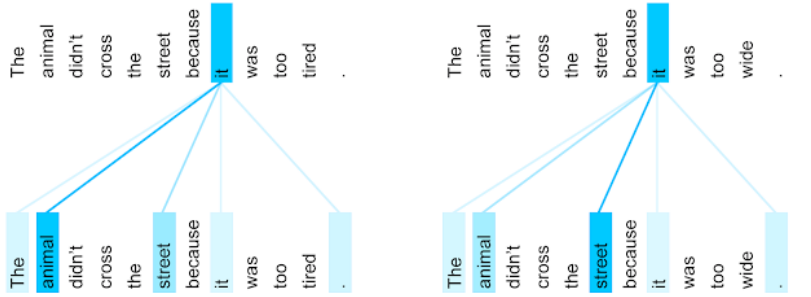


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Cons of LSTMs

- Sequential computation inhibits parallelization
- No explicit modeling of long and short range dependencies
- “Distance” between positions is linear (O(n)), i.e., too long LSTMs perform badly

Preview: Transformer: distance in O(1)

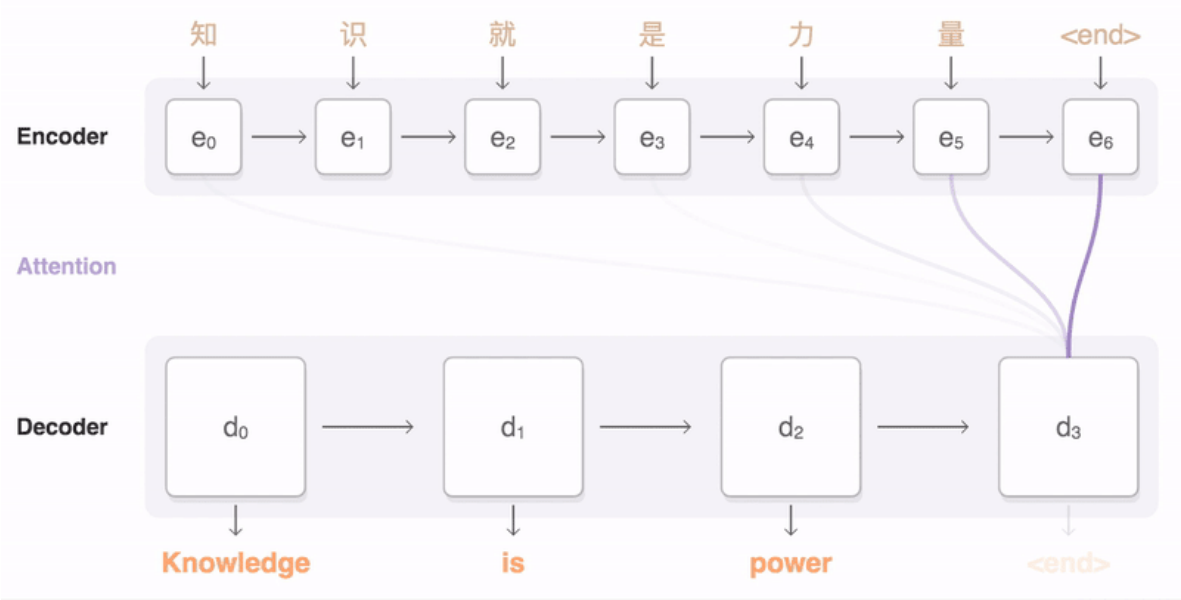


The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

See https://towardsdatascience.com/transformers-141e32e69591

Early Works on Attention

- Focus on specific information parts
  - E.g., specific words are decisive for translation



Source: https://google.github.io/seq2seq/

Transformer: Full Focus on Attention

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Source: “Attention is All You Need”

Self-Attention

- **Mechanism** to relate “different positions of a single sequence in order to compute a representation of the sequence” (see Transformer paper).
- Already existing: successfully applied to several **tasks** before Transformer: reading comprehension, abstractive summarization, textual entailment, task-independent sentence representations.

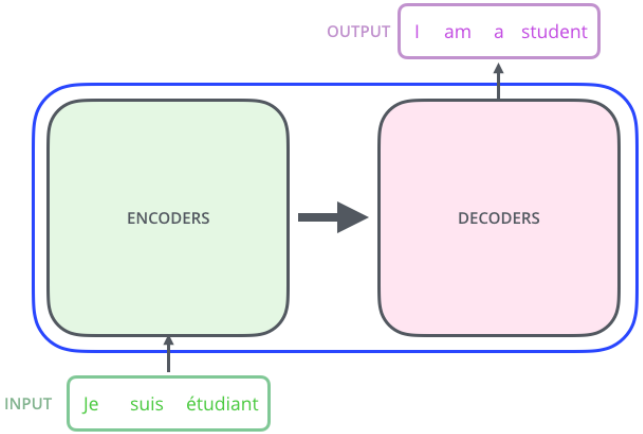
- Now: **Transformer** first “model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution”.

Transformer Architecture

Source: https://jalammar.github.io/illustrated-transformer/

Transformer Architecture

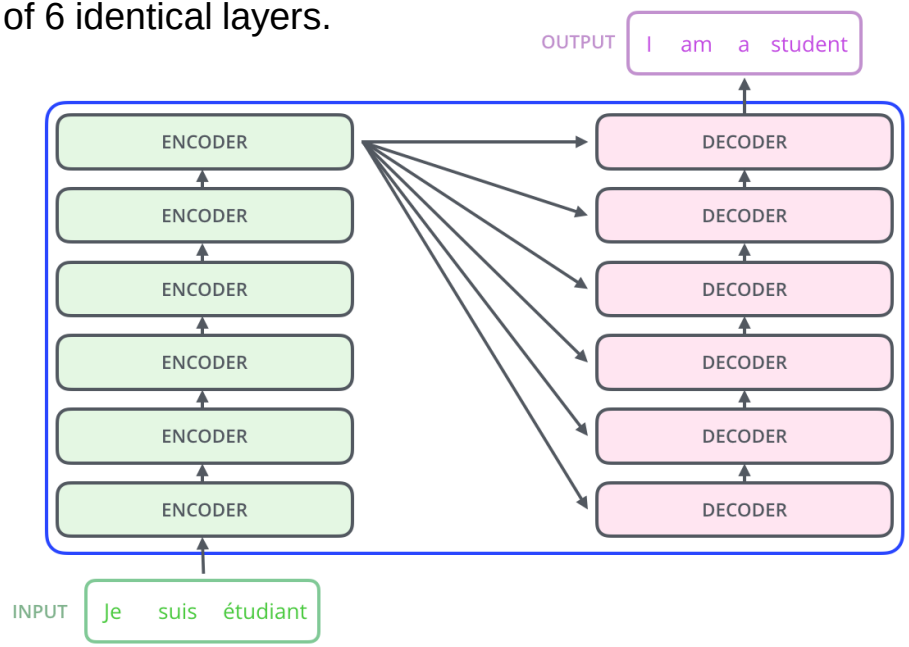
- “Still” encoder decoder architecture:
  - Encoder: Transform symbolic input into continuous representation (subsymbolic information).
  - Decoder: generate output sequence (symbols).



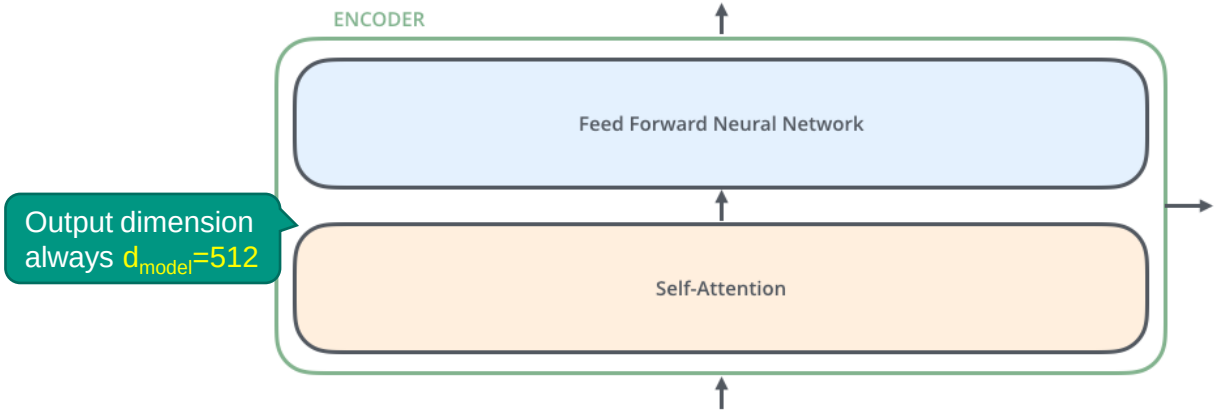
Source: https://jalammar.github.io/illustrated-transformer/

Encoder-Decoder Stack

“Let’s use several:”
Encoder is composed of 6 identical layers.
Decoder is composed of 6 identical layers.

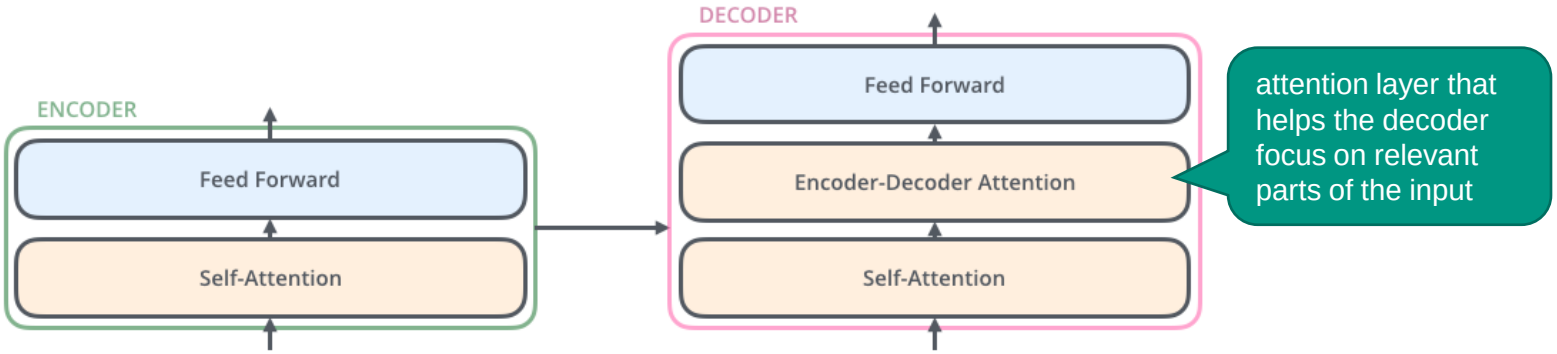


Details of Each Encoder



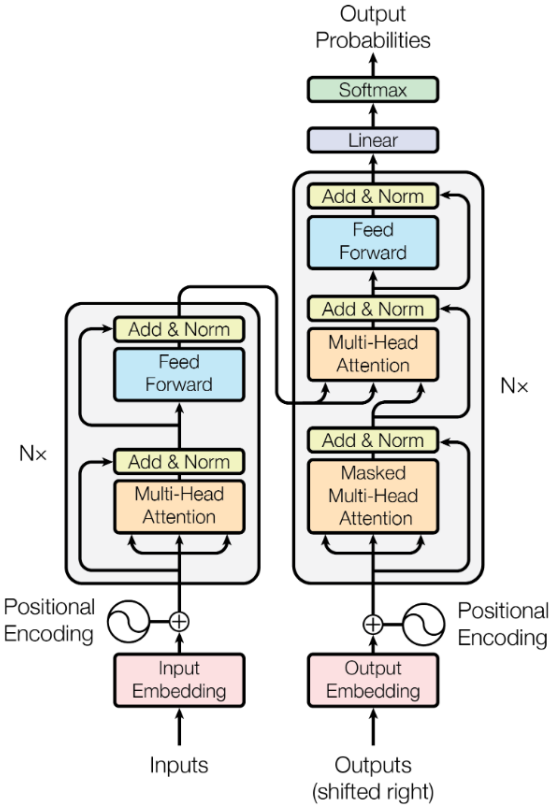
Source: https://jalammar.github.io/illustrated-transformer/

Details of Each Decoder



Source: https://jalammar.github.io/illustrated-transformer/

Full Transformer Architecture

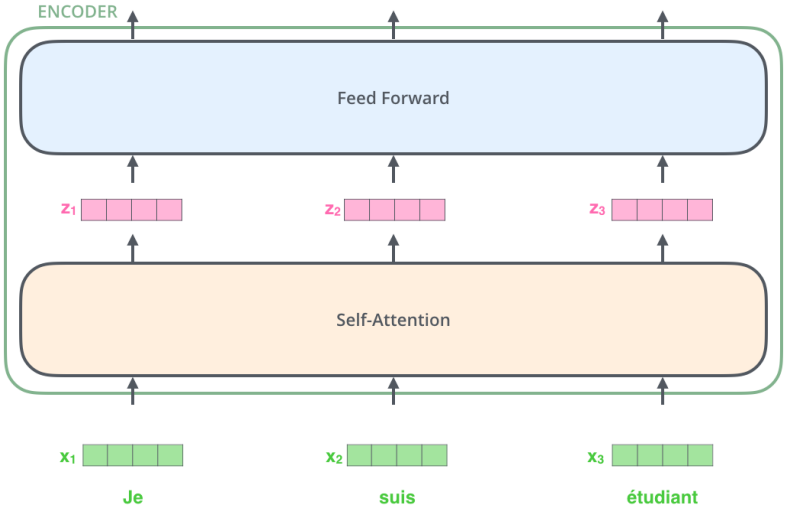


(Input) Embeddings

- Embedd each input unit (e.g., word) based on an embedding algorithm.
- Use typically available, pretrained embeddings.
- Size: e.g., 512 dim.

Embeddings

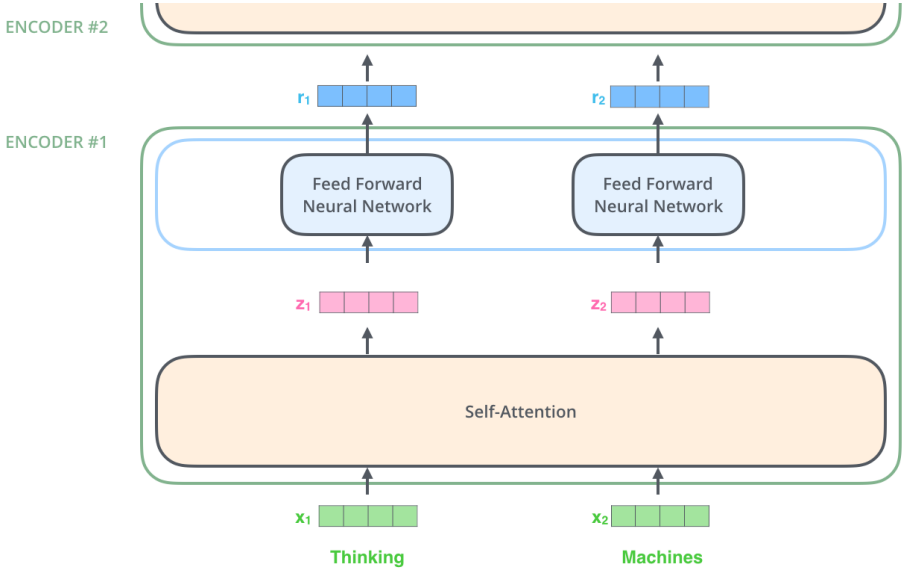
- Each position flows through its own path in the encoder, no dependencies as in case of LSTM.



Source: https://jalammar.github.io/illustrated-transformer/

Embeddings

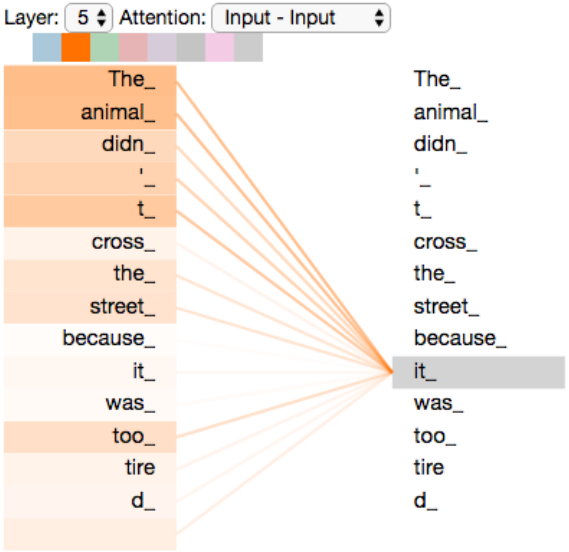
- Output of encoder also embeddings.
- Words embeddings for first encoder; in other encoders, use output of previous encoder.



Source: https://jalammar.github.io/illustrated-transformer/

Self-Attention

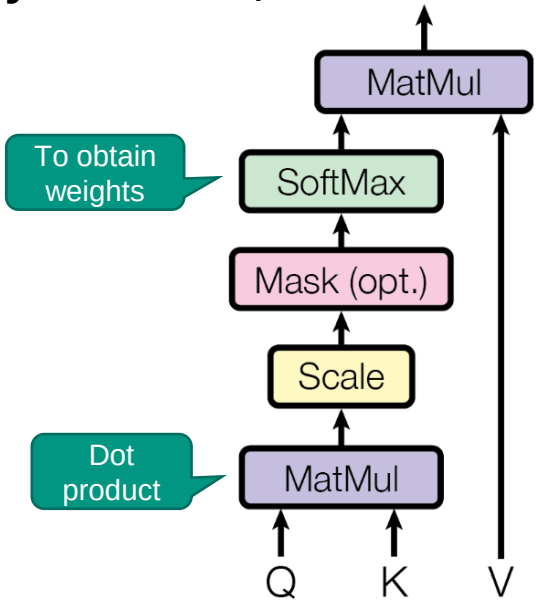
- Self attention allows to look at other positions in the input sequence.
- “Understanding” of other relevant words.



Source/Play field:
https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb

Scaled Dot-Product Attention

- Newly introduced in Transformer model.
- Using **queries**, **keys**, **values** (vectors of dim. d_k, d_k, d_v).



Dot product: similarity score to determine how much attention the model should give to the other token when processing the current token.

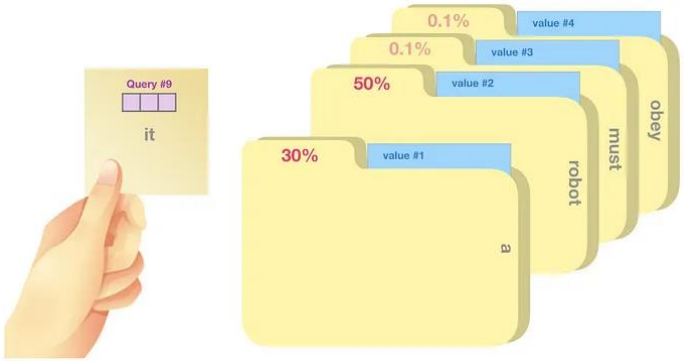
keys: "who to pay attention to“
values: "what information to pass on"

Scaled Dot-Product Attention

The key/value/query concept is analogous to retrieval systems.

Scenario: Searching videos on Youtube

- *Search engine will map the **query** (text in the search bar)*
- *against a set of **keys** (video title, description, etc.) associated with candidate videos in their database,*
- *then present you the best matched videos (**values**).*

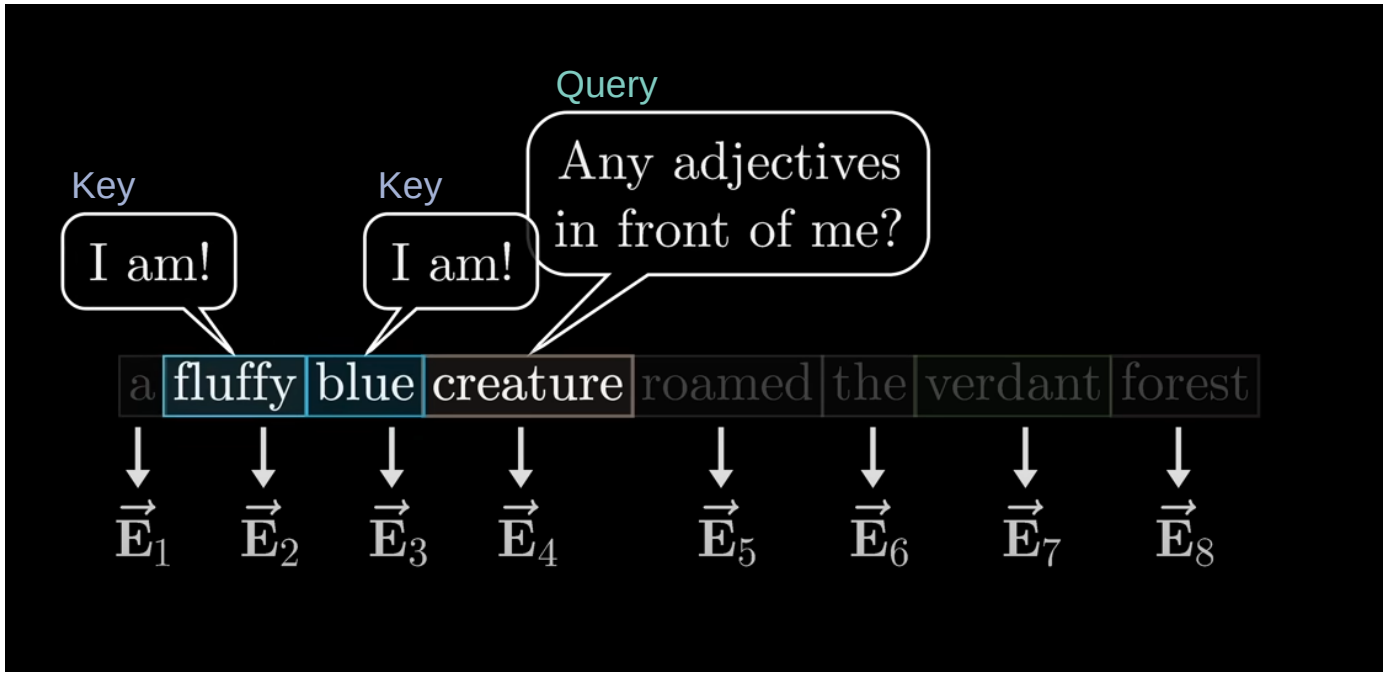


Scaled Dot-Product Attention

Attention is, in effect, the addition context to a **query** from relevant words

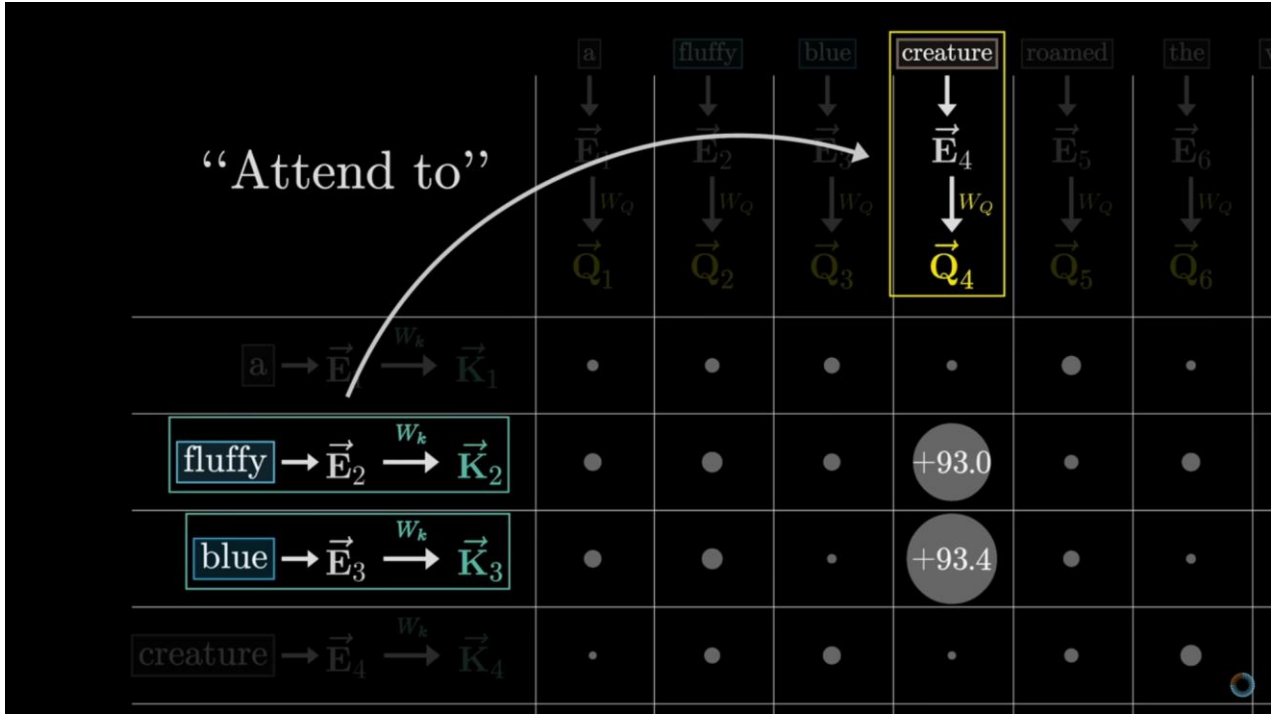
- Query — Word(s) that contextual information is added to
- Key — Words that are used to get contextual information from
- Value — Decide how much contextualized information to keep

Scaled Dot-Product Attention



Source: https://www.youtube.com/watch?v=eMlx5fFNoYc

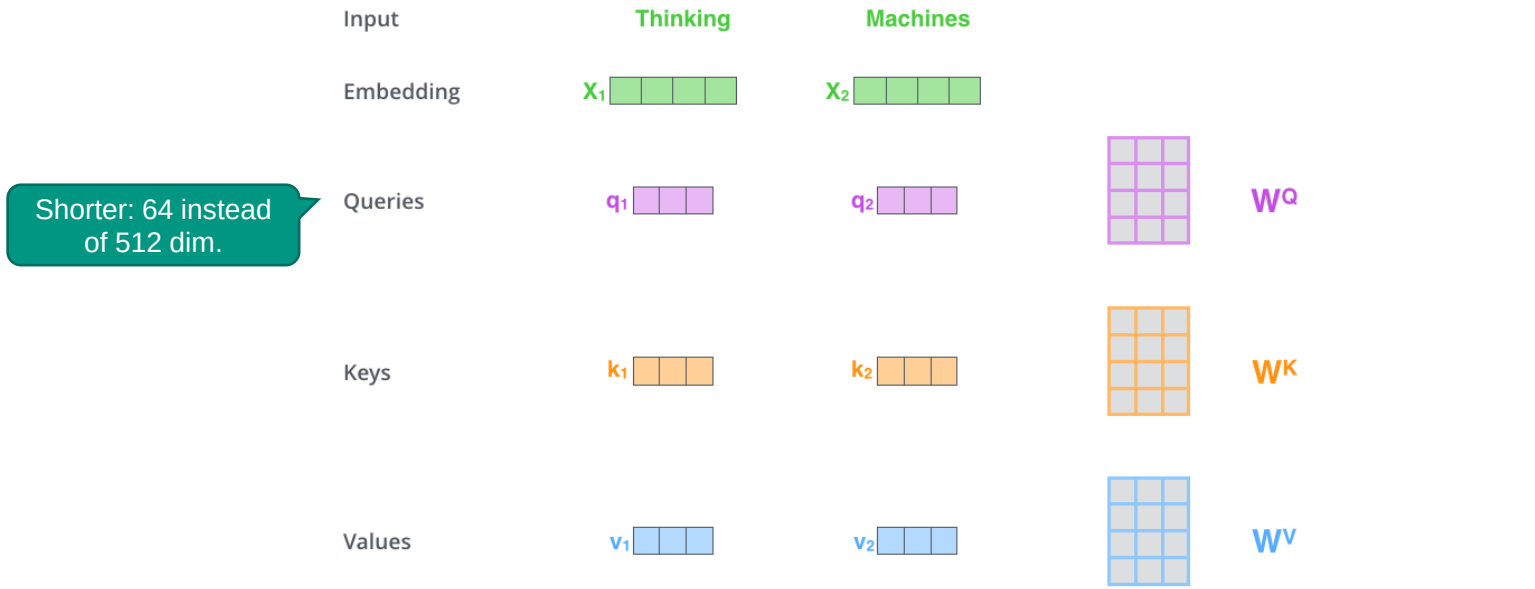
Scaled Dot-Product Attention



Source: https://www.youtube.com/watch?v=eMlx5fFNoYc

Scaled Dot-Product Attention: Step 1

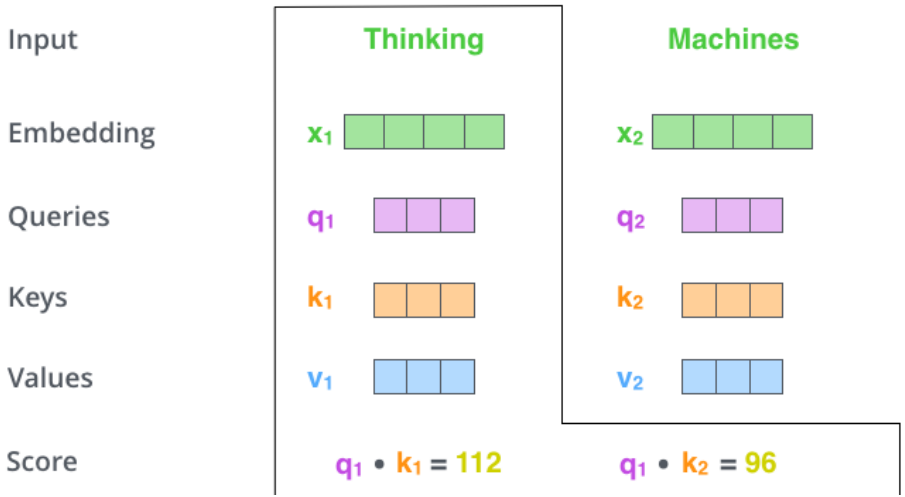
Create vectors queries, keys, values (e.g., for each word in input sentence; via matrix multiplication):



Source: https://jalammar.github.io/illustrated-transformer/

Scaled Dot-Product Attention: Step 2

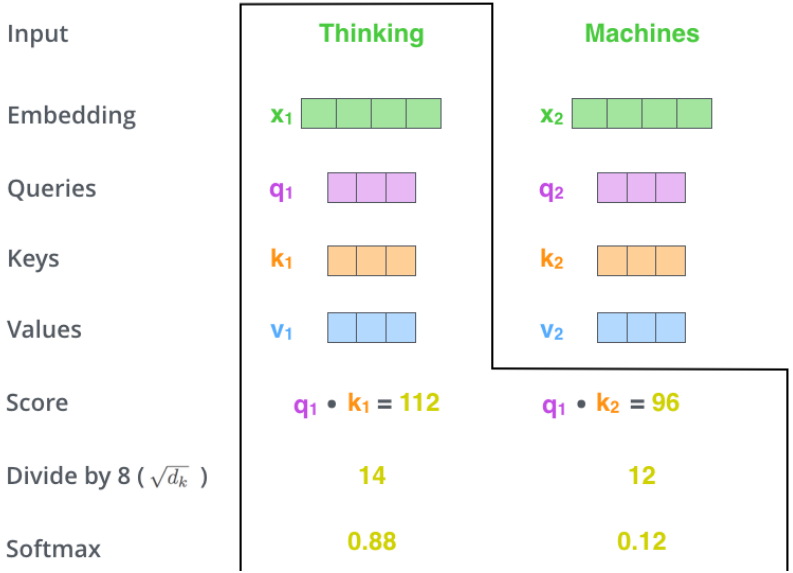
- Calculate (attention) score (e.g., each word of the input sentence against given word).
- Dot product of the query vector with the key vector of the respective word we are scoring.



Source: https://jalammar.github.io/illustrated-transformer/

Scaled Dot-Product Attention: Step 3 & 4

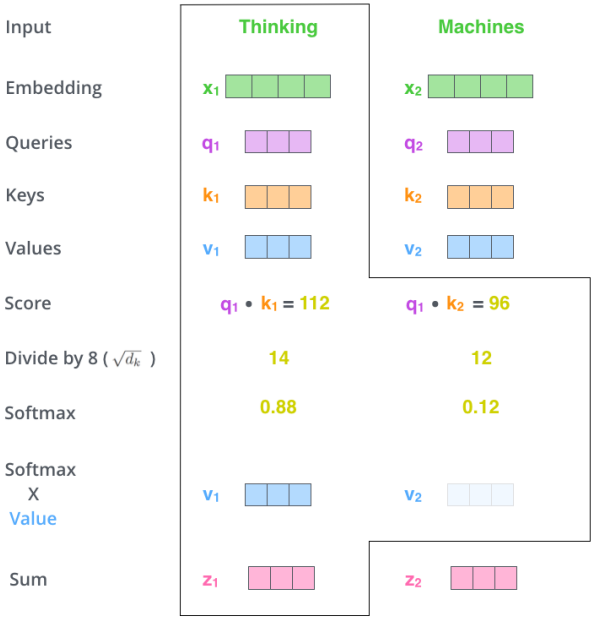
- Divide by 8 (more stable gradients).
- Softmax for normalization (sum up to 1).



Source: https://jalammar.github.io/illustrated-transformer/

Scaled Dot-Product Attention: Step 5 & 6

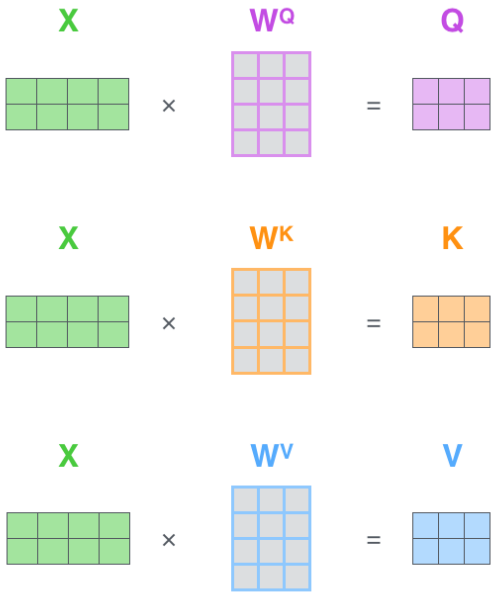
- Multiply each value vector by the softmax score.
- Sum up the weighted value vectors (then used for the feed-forward network).



Source: https://jalammar.github.io/illustrated-transformer/

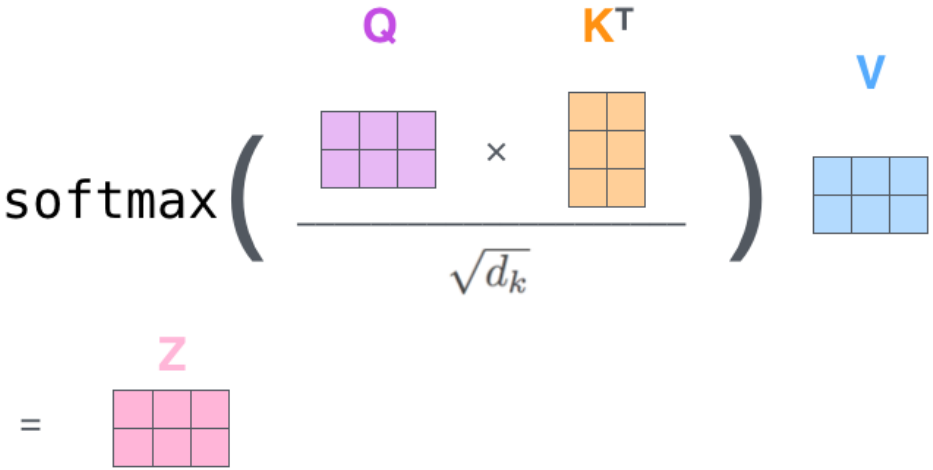
Scaled Dot-Product Attention *with Matrices*

- Step 1:



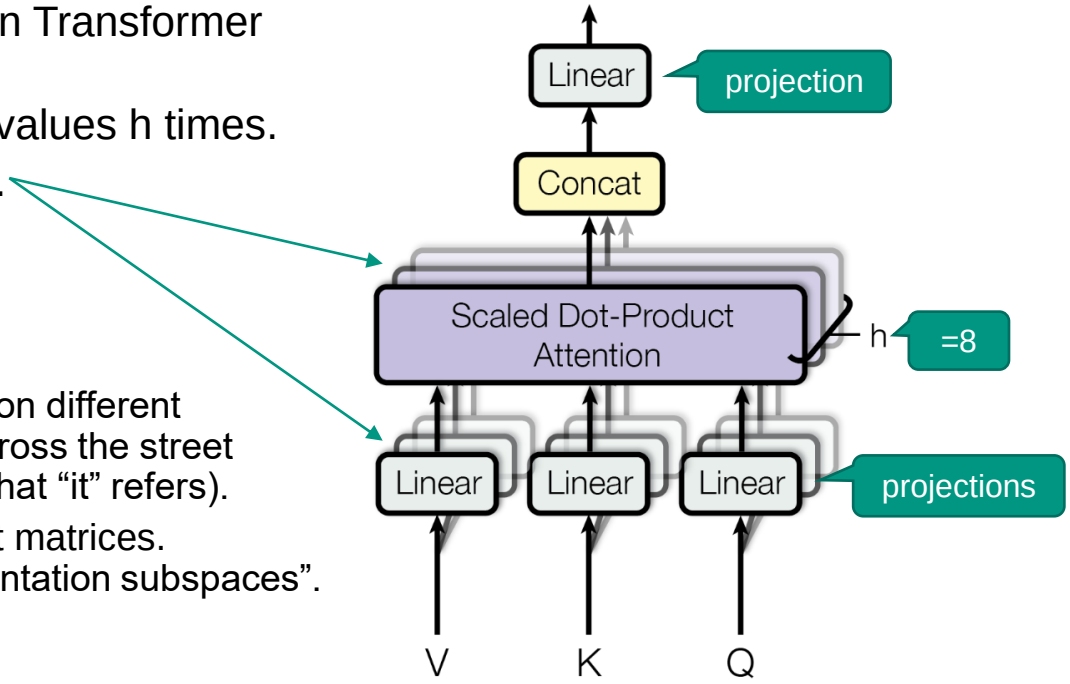
Scaled Dot-Product Attention with Matrices

- Remaining steps:



Multi-Head Attention

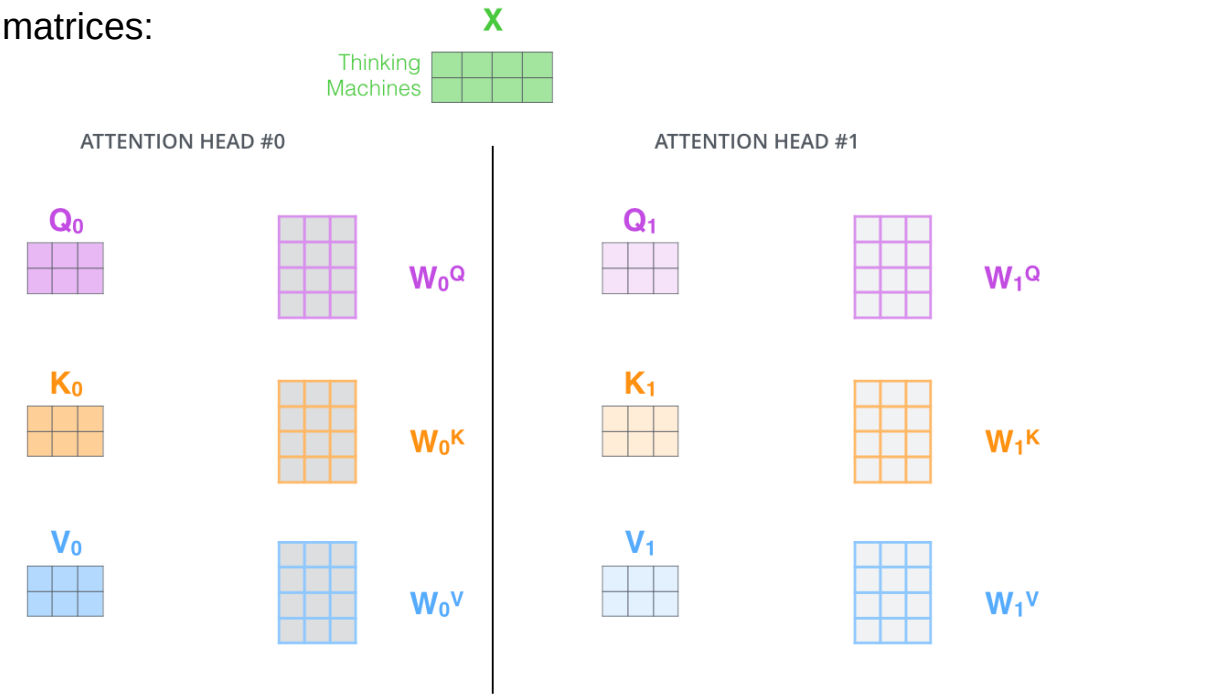
- Additional novel method introduced in Transformer paper; “more complex attention.”
- Idea: Linearly project queries, keys, values h times.
- Perform attention function in parallel.



- Why?
  - It expands the model’s ability to focus on different positions (e.g., for “The animal didn’t cross the street because it was too tired” knowing to what “it” refers).
  - Multiple sets of query/key/value weight matrices.
    → Attention layer has several “representation subspaces”.

Multi-Head Attention

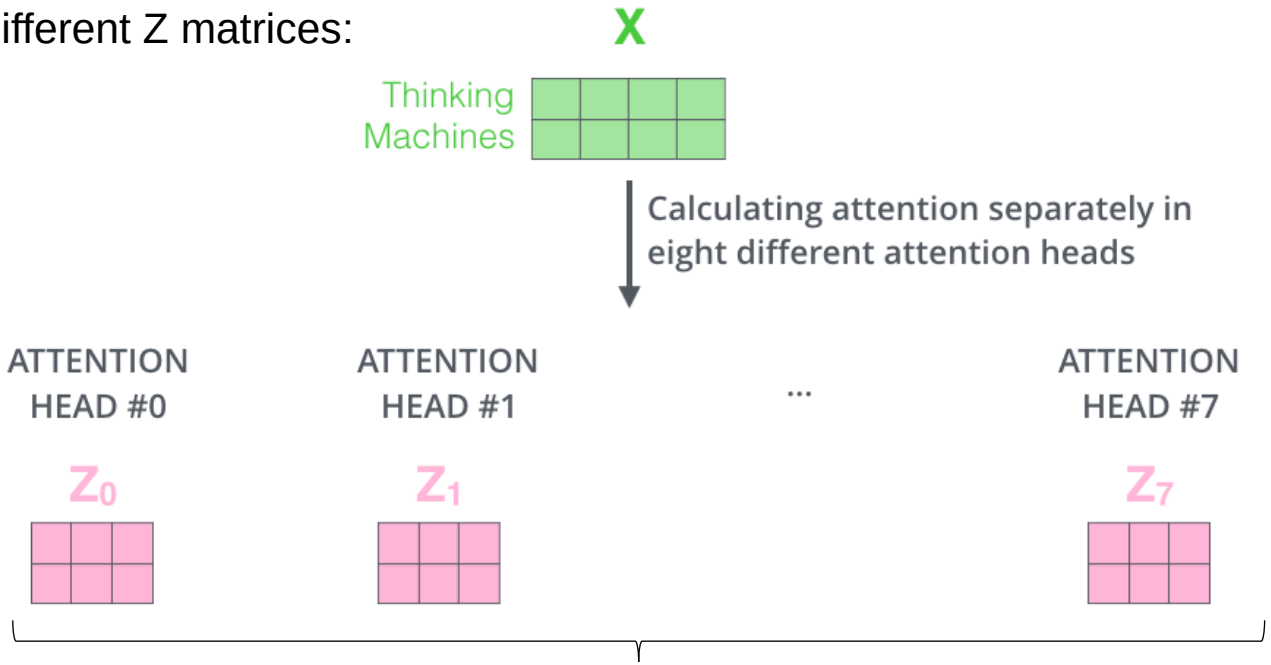
Several Q/K/V weight matrices:



Source: https://jalammar.github.io/illustrated-transformer/

Multi-Head Attention

Output: 8 different Z matrices:

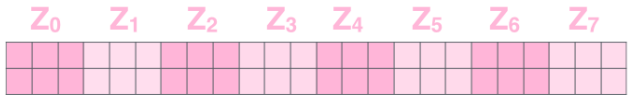


Too large for FNN in next step, we only need 1 matrix…

Multi-Head Attention

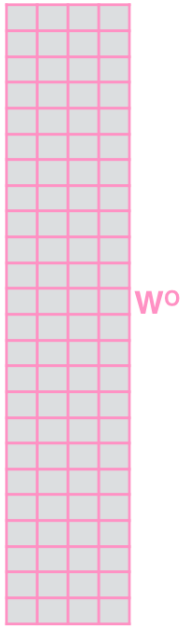
Condense to 1 matrix:

1) Concatenate all the attention heads



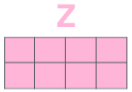
2) Multiply with a weight matrix W^O that was trained jointly with the model

X

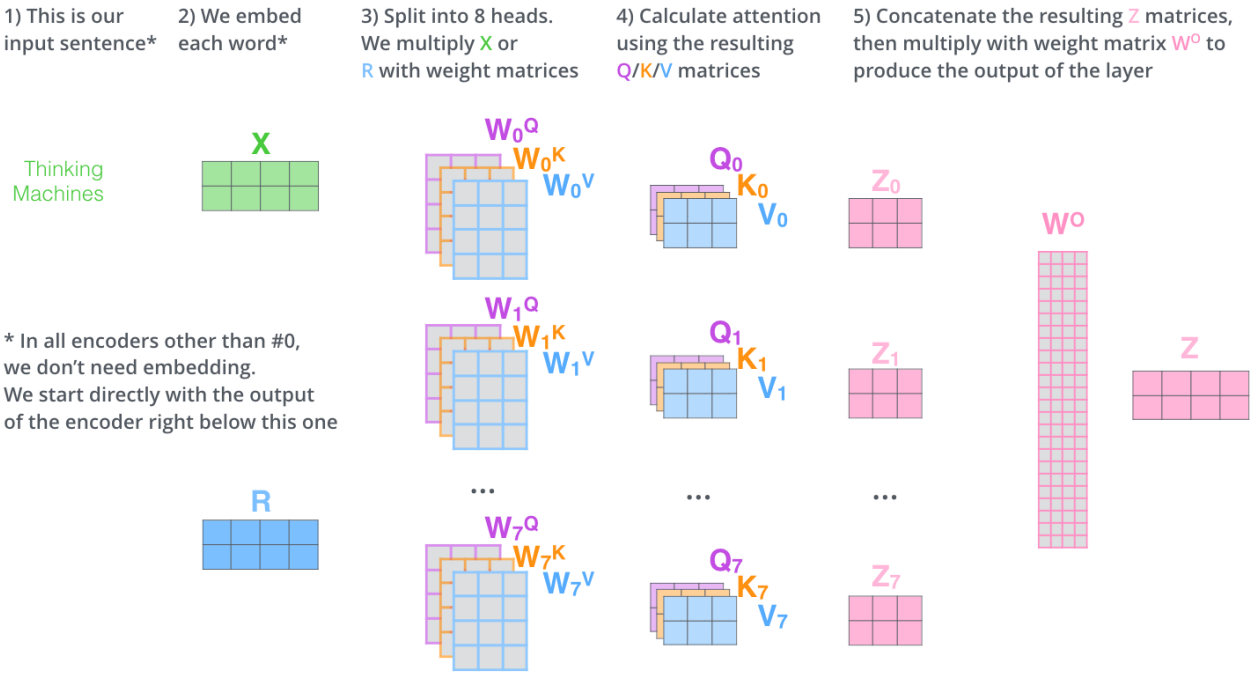


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=

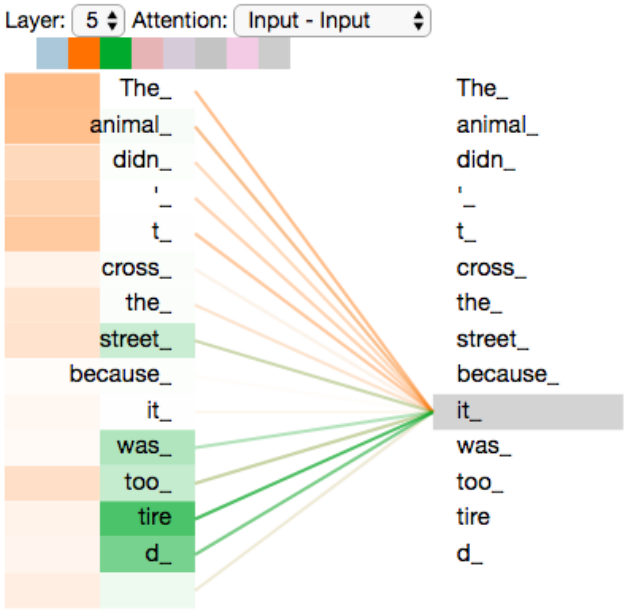


Multi-Head Attention – Overview

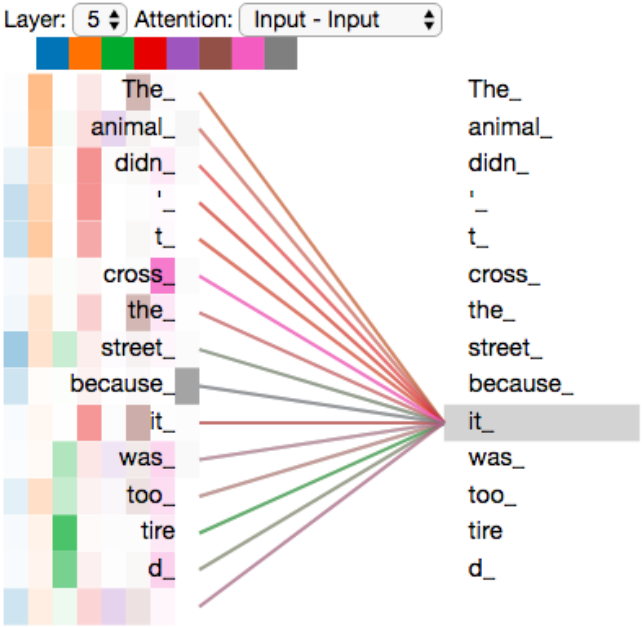


Source: https://jalammar.github.io/illustrated-transformer/

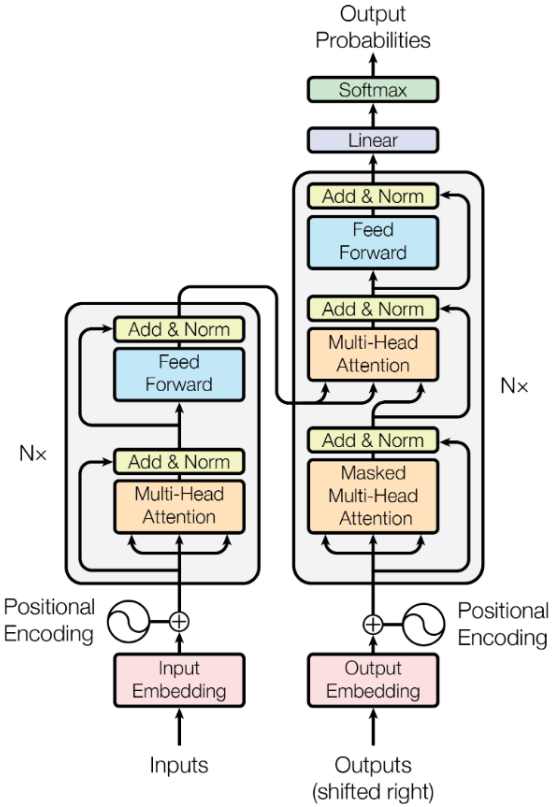
Multi-Head Attention Example



With all (8) attention heads:



Full Transformer Architecture



Position-wise Feed-Forward Networks

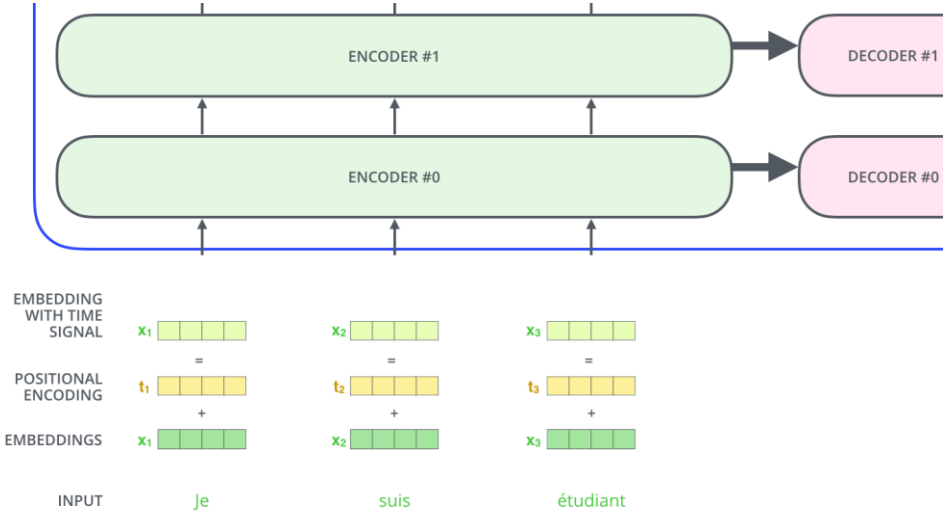
- After the attention sub-layers in encoders and decoders.
- 2 linear transformations with a ReLU activation in between.

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

- Dimensionality of input/output: 512.
- Dimensionality of inner-layer: 2048.
- W_1/W_2: different parameters from layer to layer (but same across positions).

Positional Encoding

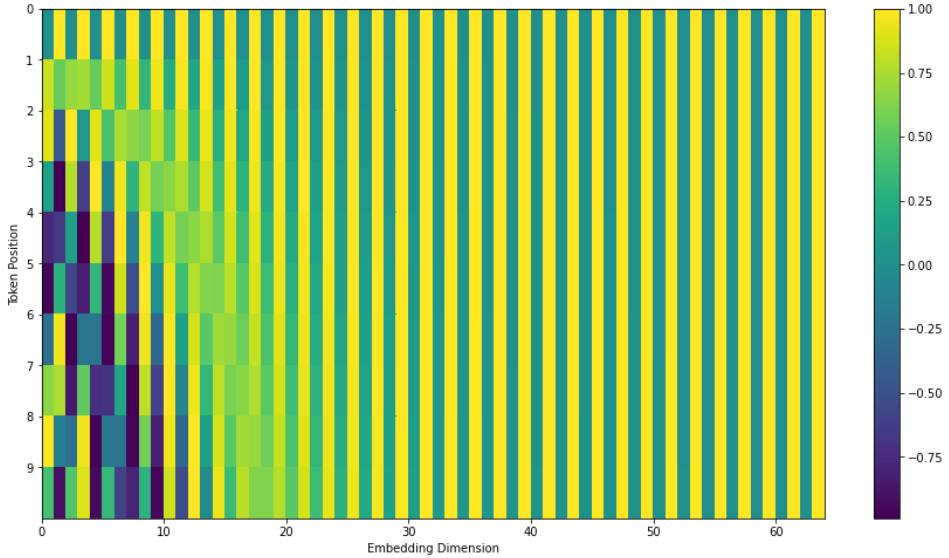
- Position information needed, otherwise bag-of-words.
- Transformer adds a vector to each input embedding to have meaningful distances between embedding vectors.



Source: https://jalammar.github.io/illustrated-transformer/

Positional Encoding

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



Source: https://jalammar.github.io/illustrated-transformer/
See also https://towardsdatascience.com/master-positional-encoding-part-i-63c05d90a0c3

Why Self-Attention

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Often n < d

Source: “Attention is All You Need”, https://arxiv.org/pdf/1706.03762.pdf

➔ “more efficient” (faster), “more effective” (state-of-the-art results), “more interpretable”

Training

Using

- 8 GPUs “only”, 12h–3.5d training.
- Adam optimizer.
- Residual dropout.
- Label smoothing (preventing overfitting and/or overconfidence).

Applications of Transformer

- Machine translation
- Document summarization
- Document generation
- Named entity recognition
- Biological sequence analysis
- Video understanding
- Image processing
- IoT
- …

Further Material

- “LSTM is dead. Long Live Transformers!”: https://www.youtube.com/watch?v=S27pHKBEp30
- Jupyter Notebook provided as part of the official Transformer repository (Tensor2Tensor): https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb
- Attention visualized: https://colab.research.google.com/drive/1rPk3ohrmVclqhH7uQ7qys4oznDdAhpzF
- Visualized transformer: https://jalammar.github.io/illustrated-transformer/