HERIOT WATT UNIVERSITY

Applied Text Analytics (F20/F21AA)

Introduction to Transformers

Heriot Watt University , Dubai Campus

Dr. Radu Mihailescu

Content

- Building blocks of a Tranformer
- Encoder-Decoder Architecture
- Attention
  - Self-attention
  - Multi-headed attention
  - Encoder-Decoder attention

Transformers

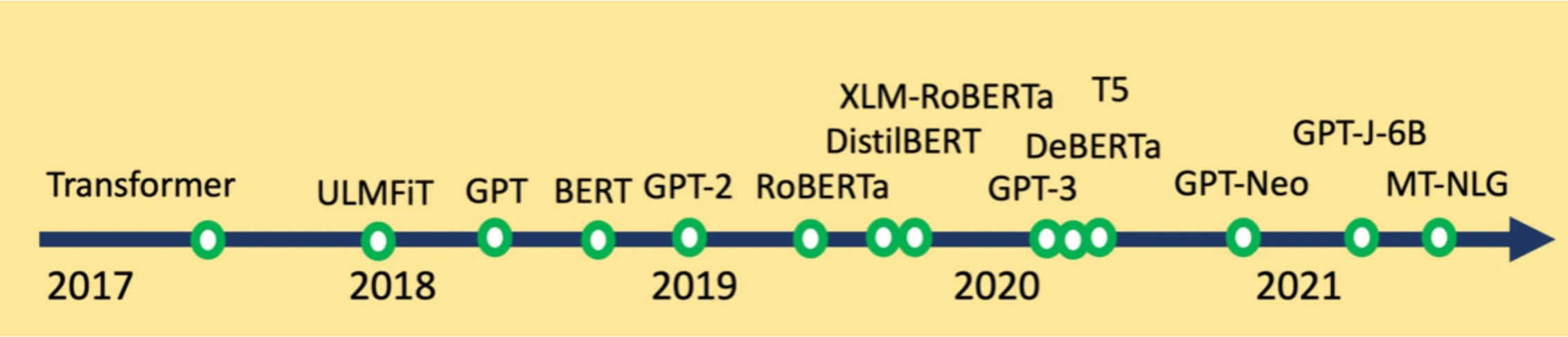
Neural network architecture for sequence modeling

- Based on work A. Vaswani et al., "Attention Is All You Need", (2017) https://arxiv.org/pdf/1706.03762.pdf
- Outperformed recurrent neural networks (RNNs) on machine translation tasks, both in terms of translation quality and training cost.

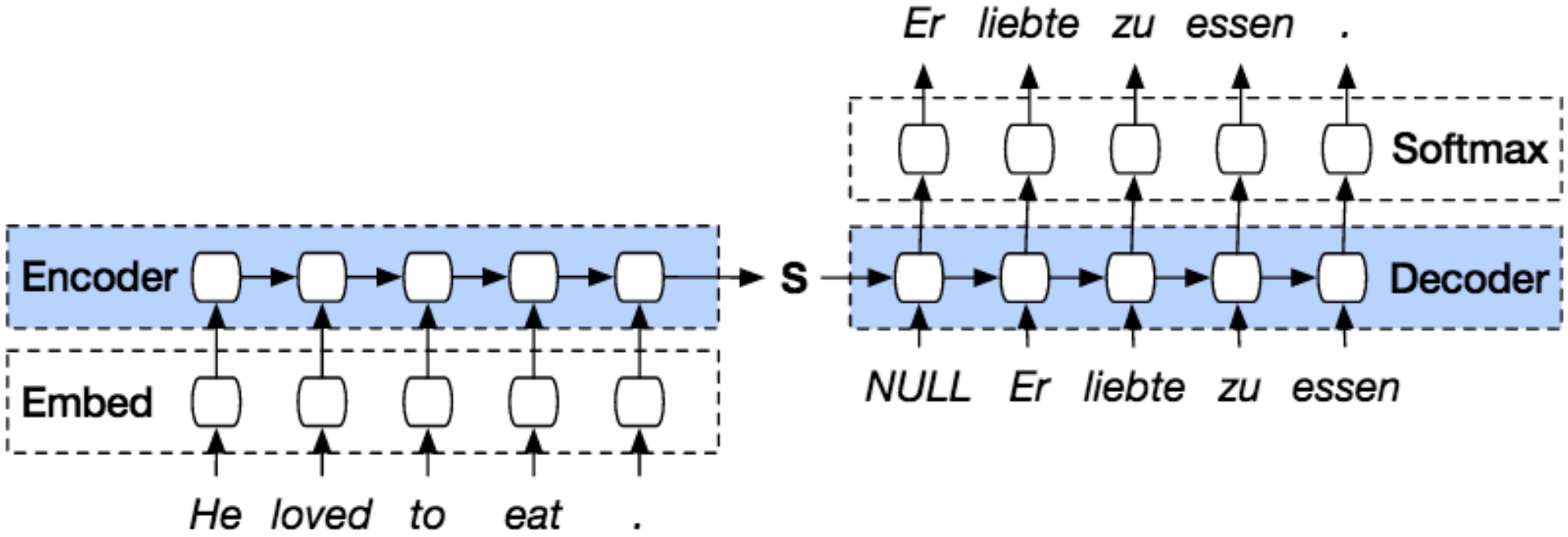
Today's most well-known transformers:

- Generative Pretrained Transformer (GPT)
- Bidirectional Encoder Representations from Transformers (BERT).

Transformers Timeline

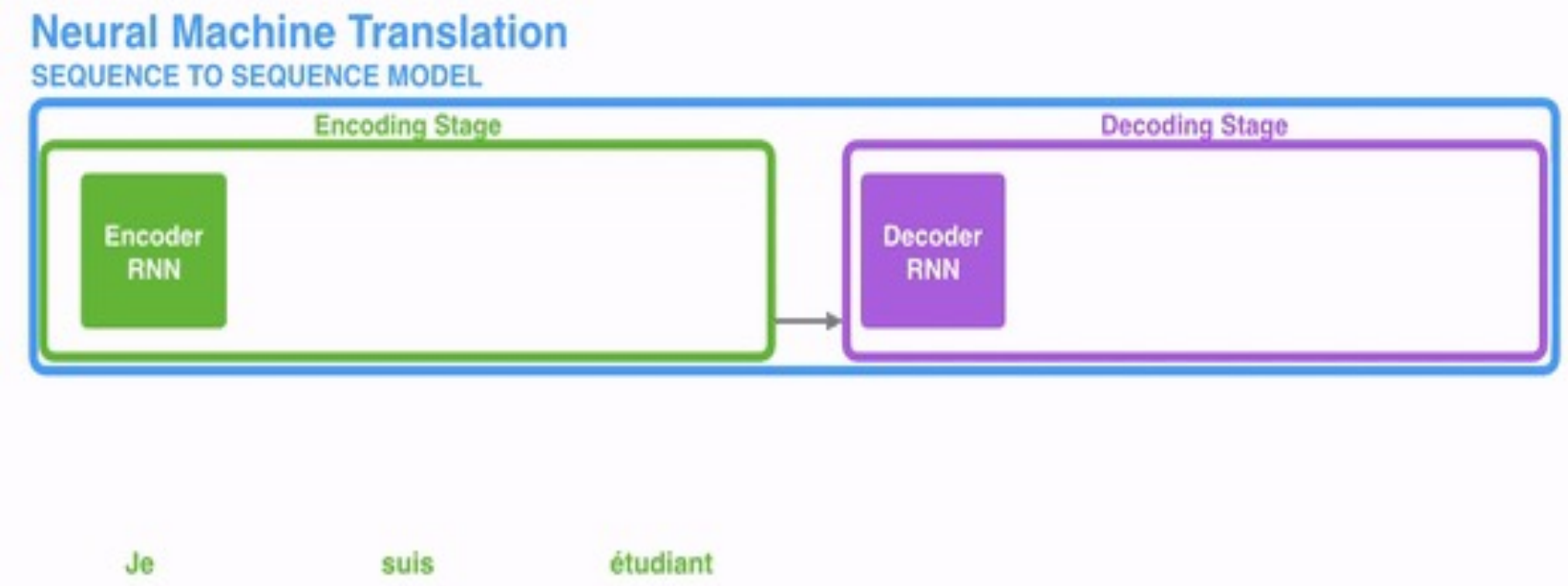


RNN Encoder-Decoder: Language Translation



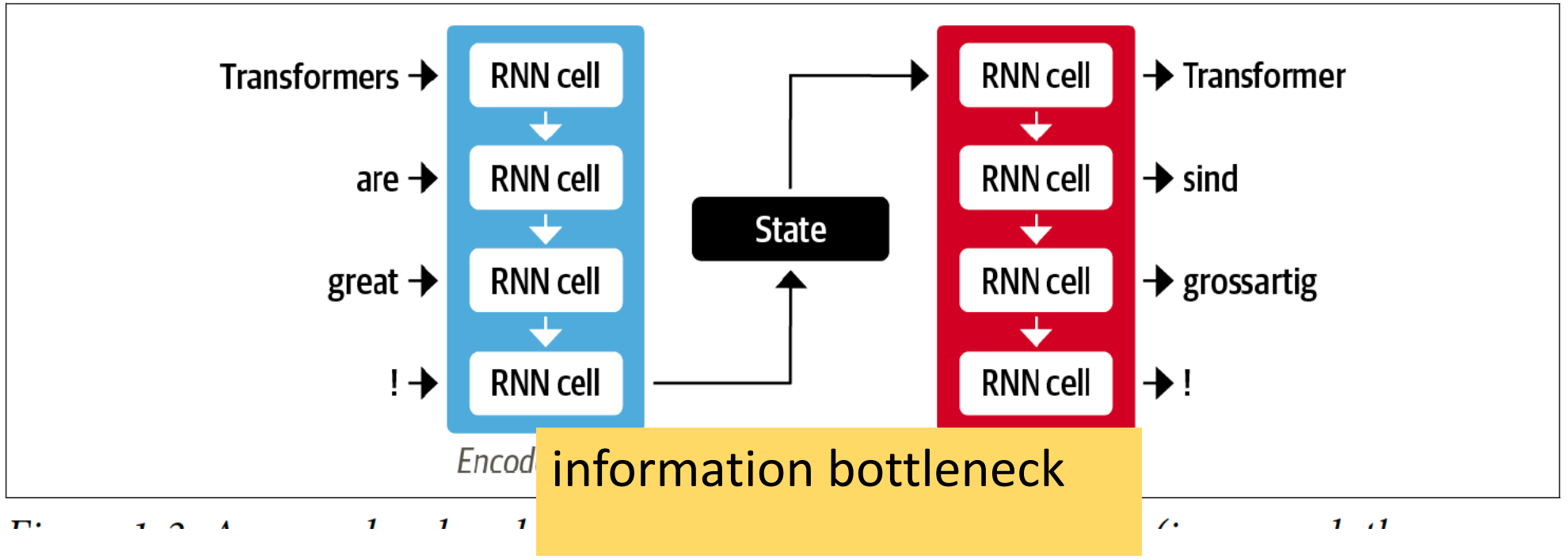
The idea is to use one LSTM to read the input sequence, *one timestep at a time*, to obtain large fixed-dimensional vector representation (S), and then to use another LSTM to extract the output sequence from that vector

RNN Sequence to Sequence model



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

RNN Encoder Decoder

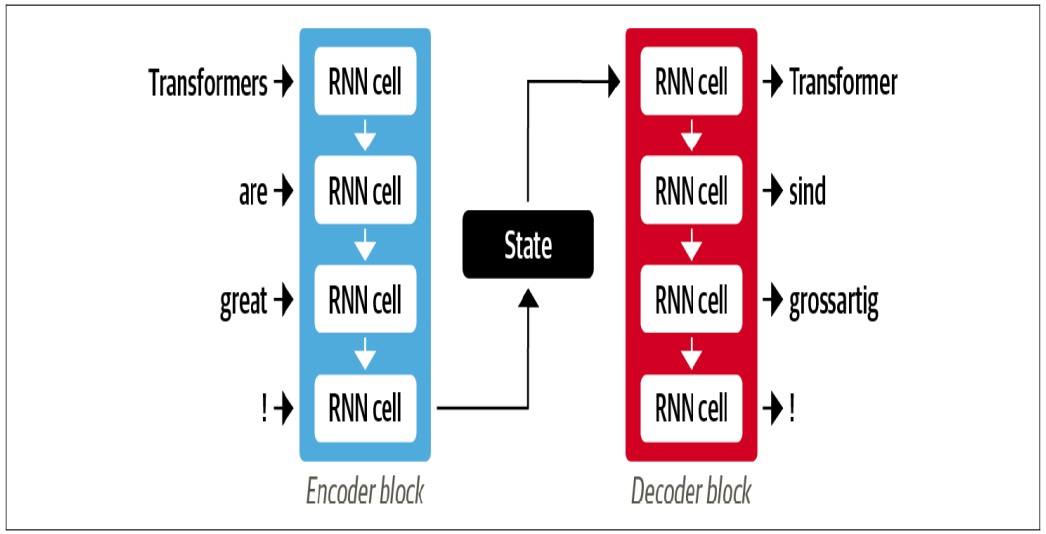


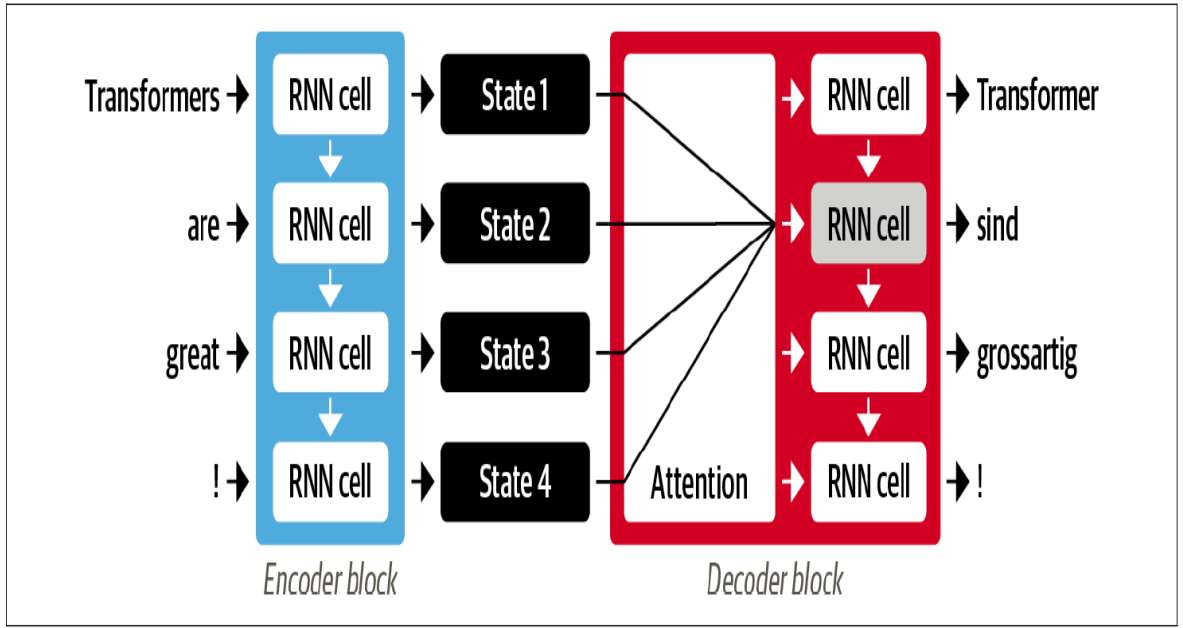
- "state" has to represent the meaning of the whole input sequence
- the decoder has access to only "state" while generating the output.
- challenging for long sequences, so information at the start of the sequence might be lost in the process of compressing everything to a single, fixed representation.

Attention

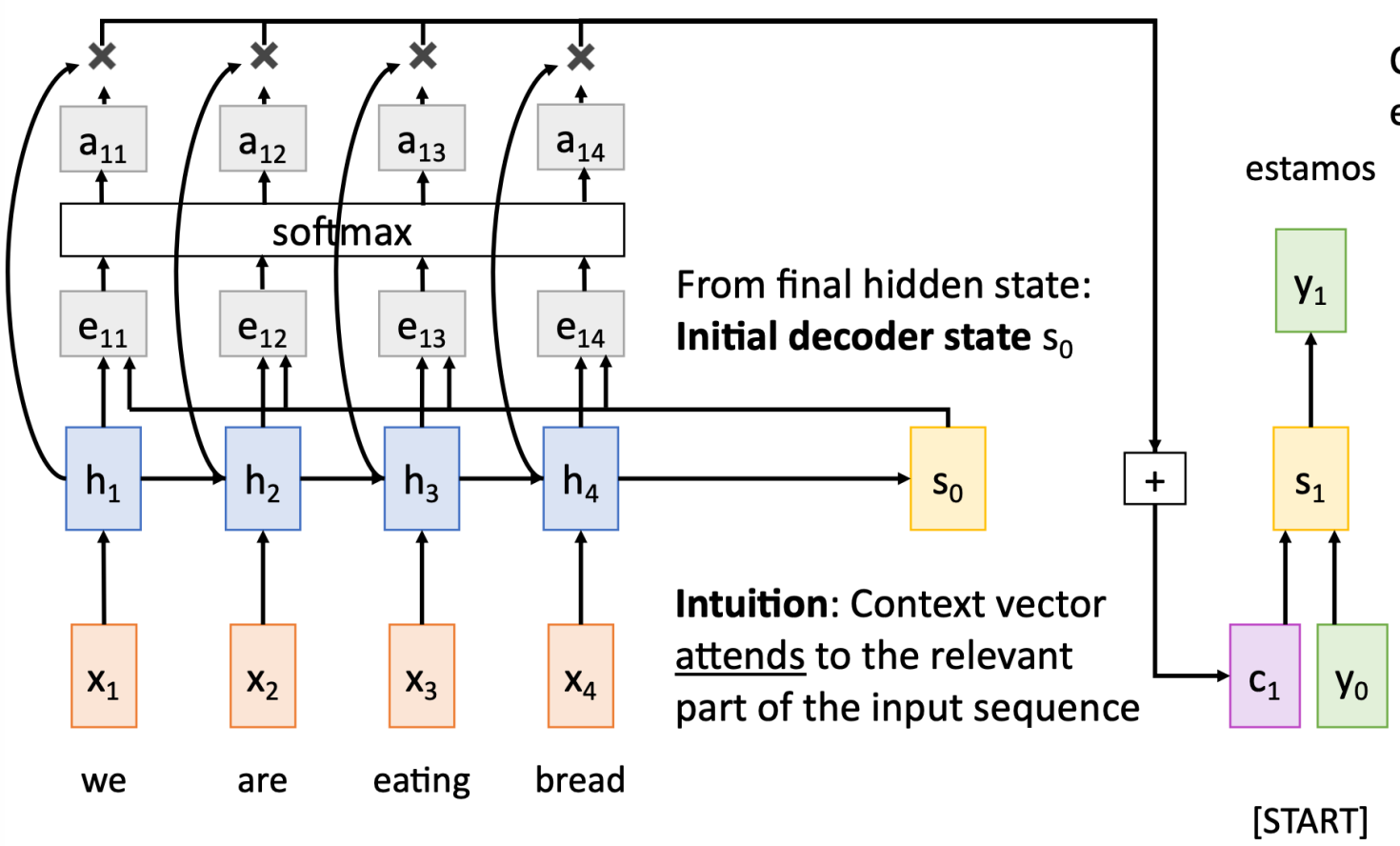
- Attention mechanisms revolutionized neural machine translation (and deep learning in general), allowing a significant improvement in the state of the art, especially for long sentences (e.g., over 30 words).

- instead of producing a single hidden state for the input sequence, the encoder outputs a hidden state at each step that the decoder can access.





RNN Encoder Decoder (with attention mechanism)



Compute (scalar) **alignment scores**
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ (f_{att} is an MLP)

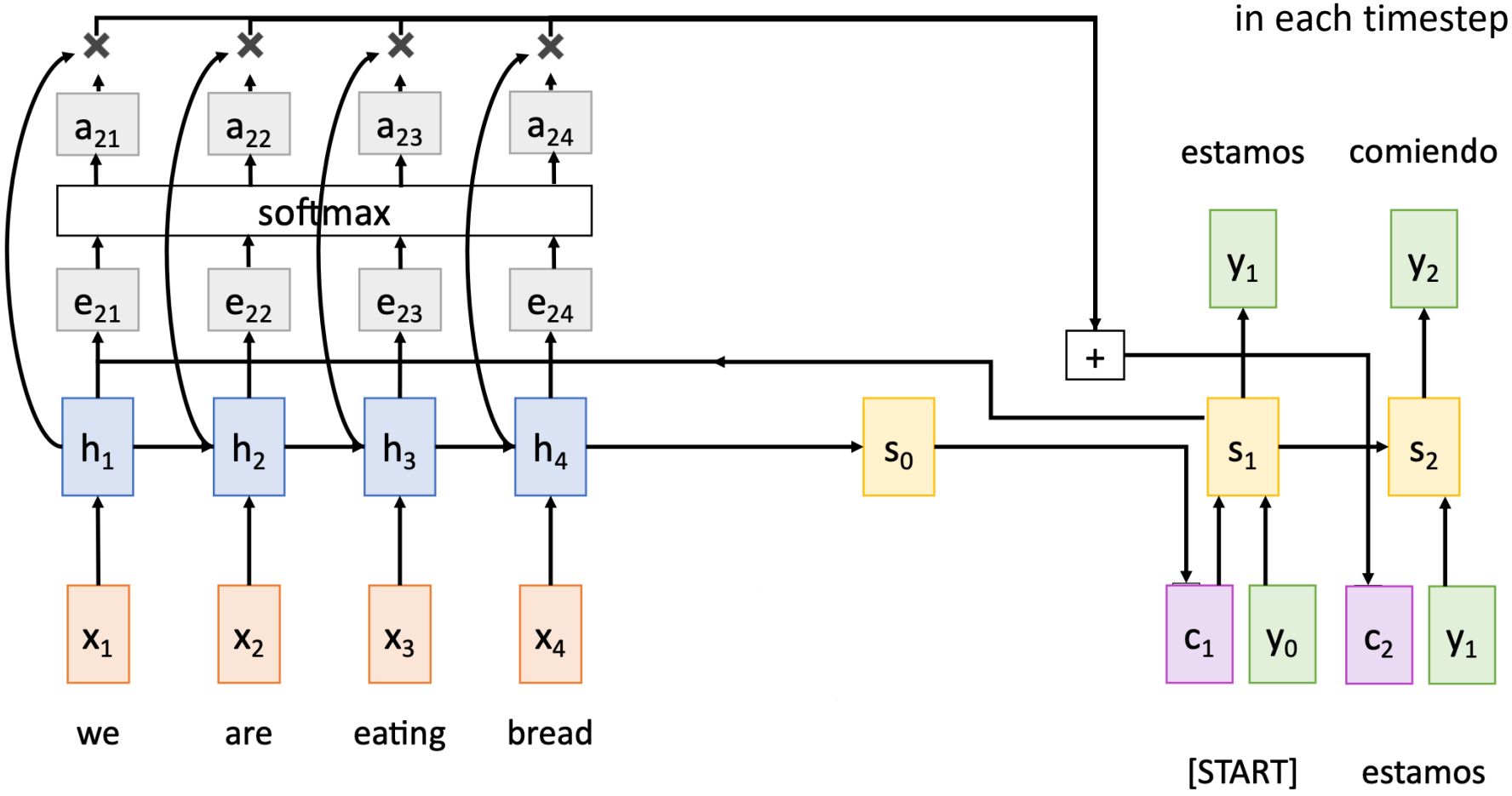
Normalize alignment scores to get **attention weights**
$0 < a_{t,i} < 1 \quad \sum_i a_{t,i} = 1$

Compute context vector as linear combination of hidden states
$c_t = \sum_i a_{t,i} h_i$

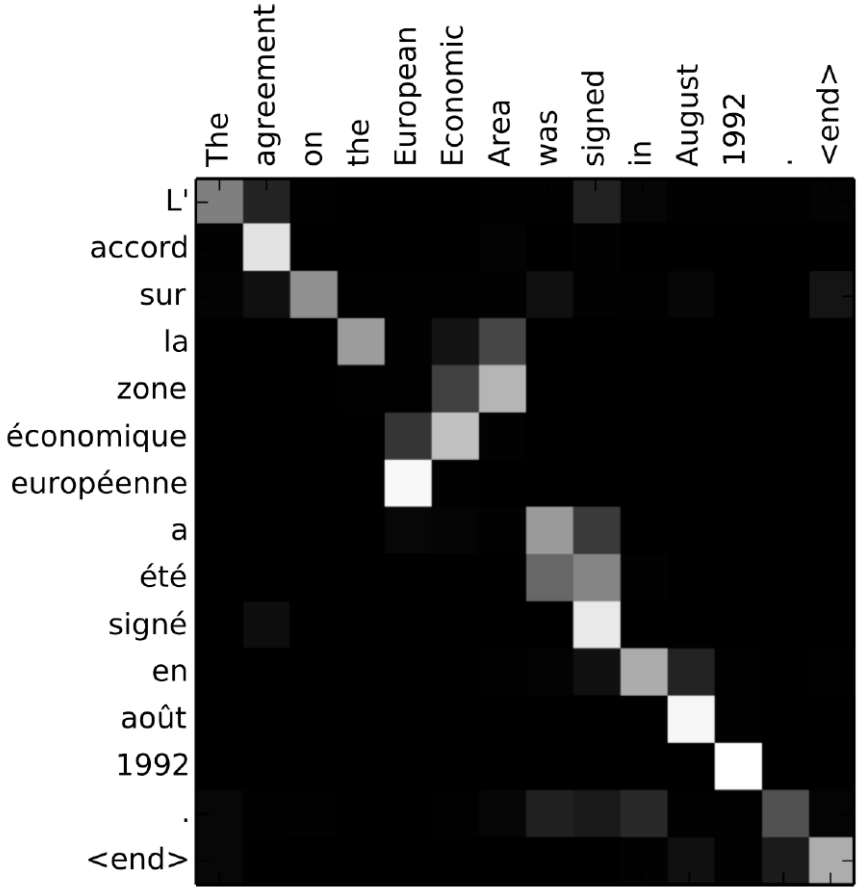
Use context vector in decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c_t)$

RNN Encoder Decoder(with attention mechanism)

Use a different context vector in each timestep of decoder



RNN Encoder Decoder (with attention mechanism)



decoder assign a different amount of weight, or "attention," to each of the encoder states

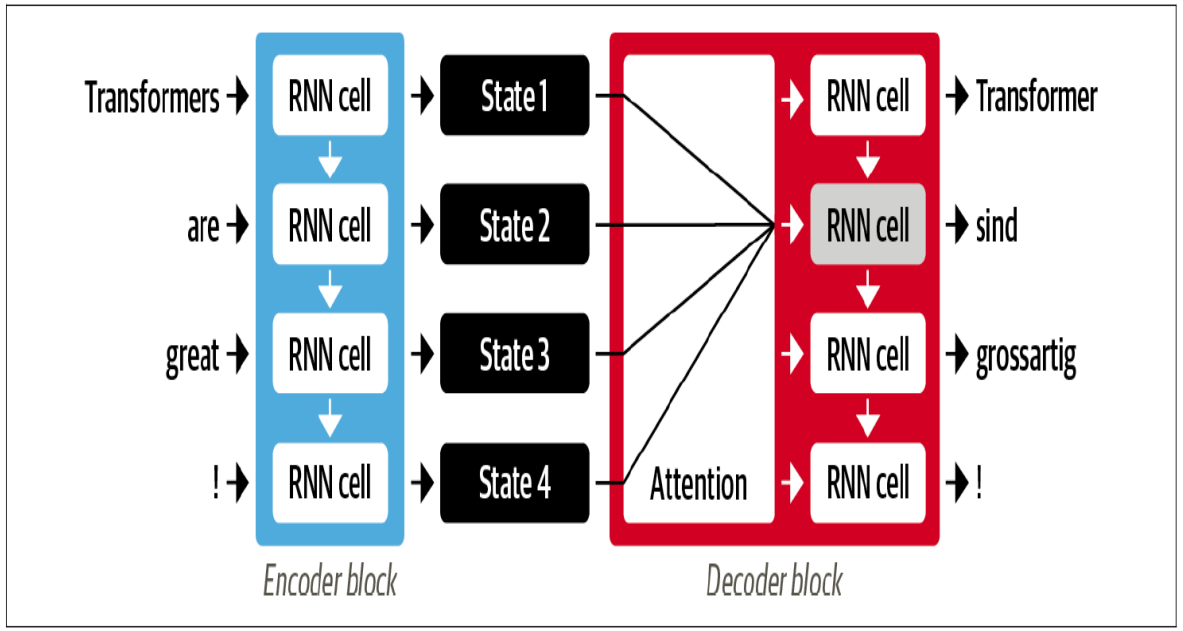
Each pixel denotes a weight.

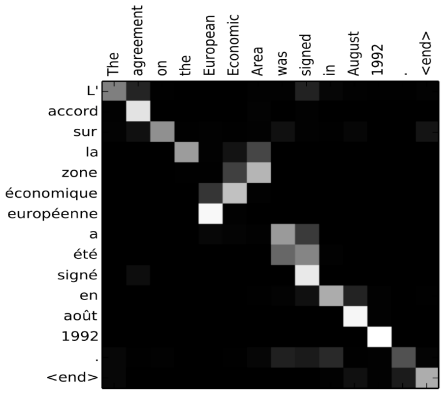
attention-based models are able to learn nontrivial alignments between the words in a generated translation and those in a source sentence

D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translat
(2014).

the source sentence (English) and the generated translation (French),

RNN Encoder Decoder (with attention mechanism)





- major shortcoming with using recurrent models for the encoder and decoder: the computations are inherently sequential and cannot be parallelized across the input sequence

Generalizing attention layers

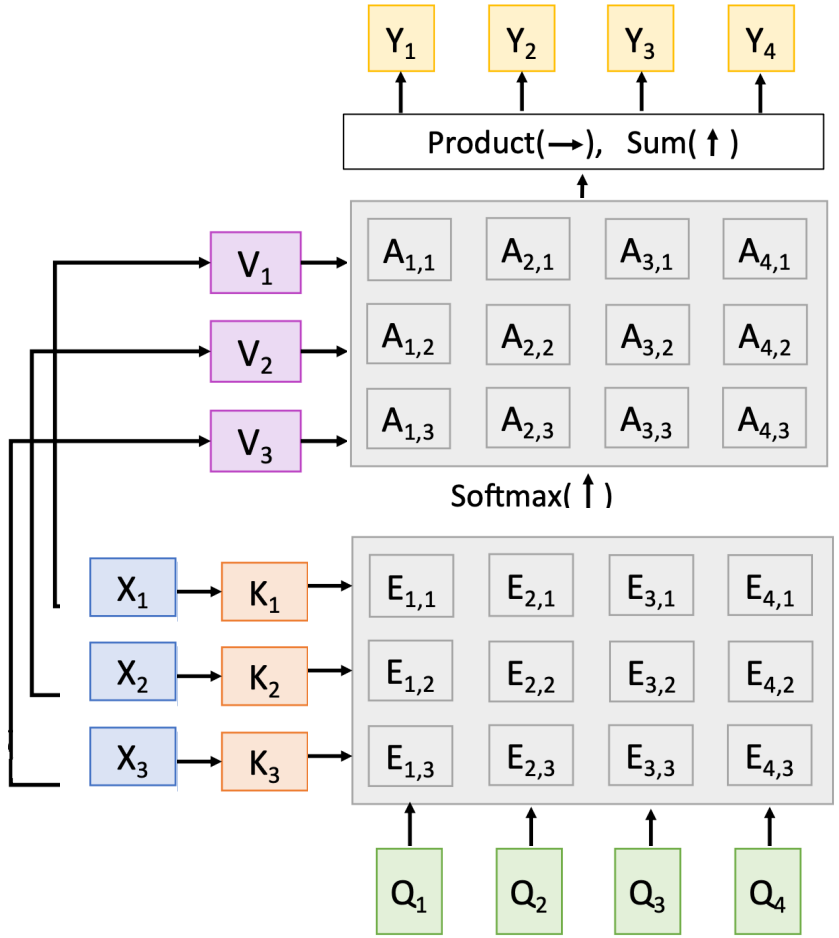
The query, key and value concepts come from information retrieval syst.

For example, when you type a query to search for some video on Youtube, the search engine will map your **query** against a set of **keys** (video title, description etc.) associated with candidate videos in the database, then present you the best matched videos (**values**).

Generelazing attention layers

Key matrix: W_K (Shape: D_X x D_Q)
Value matrix: W_V (Shape: D_X x D_V)

Key vectors: $K = XW_K$ (Shape: N_X x D_Q)
Value Vectors: $V = XW_V$ (Shape: N_X x D_V)



Self-attention layers

Every input vector $\mathbf{x}_i$ is used in three different ways in the self attention operation:

- Queries
- Key
- Values

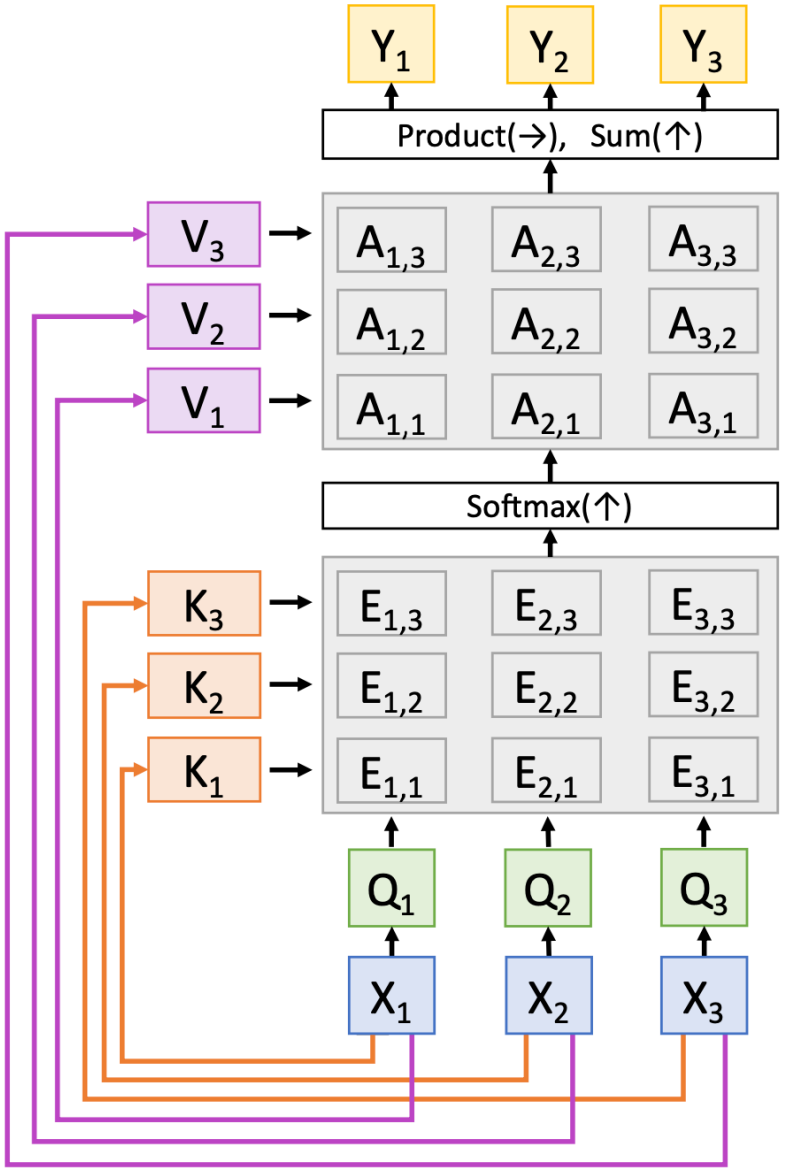
We add three **k×k** weight matrices $\mathbf{W}$q, $\mathbf{W}$k, $\mathbf{W}$v and compute three linear transformations of each x_i, for the three different parts of the self attention

$$\mathbf{q}_i = \boldsymbol{W}_q \mathbf{x}_i \qquad \mathbf{k}_i = \boldsymbol{W}_k \mathbf{x}_i \qquad \boldsymbol{v}_i = \boldsymbol{W}_v \mathbf{x}_i$$

$$w'_{ij} = \mathbf{q}_i{}^{\mathsf{T}} \mathbf{k}_j$$

$$w_{ij} = \mathrm{softmax}(w'_{ij})$$

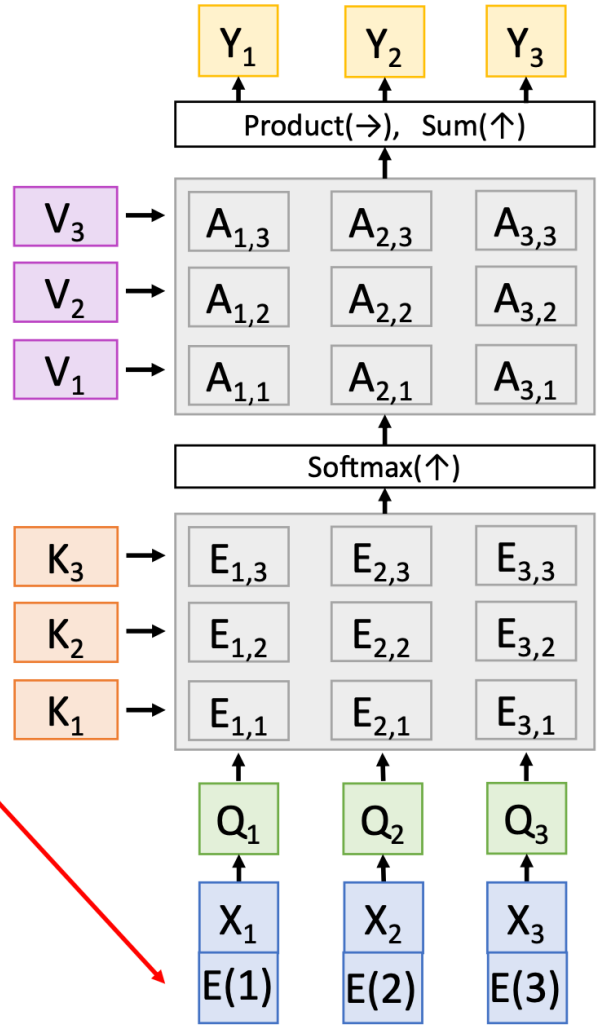
$$\mathbf{y}_i = \sum_j w_{ij} \boldsymbol{v}_j \ .$$



Self-attention layers

Self-attention doesn't "know" the order of the vectors it is processing!

In order to make processing position- aware, concatenate input with **positional encoding**



Positional Embedding

- Used to inject information about token positions into the input to model the sequential nature of text.

- The token embeddings are thus combined with positional embeddings that contain positional information for each token.

- Types of positional embeddings:
  - Absolute positional representations: use static patterns consisting of modulated sine and cosine signals to encode the positions of the tokens.
  - Relative positional representations: take into account the surrounding tokens are most important.

Positional Embedding Example

- https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

$$P(k, 2i+1) = \cos\left(\frac{k}{n^{2i/d}}\right)$$

n: User-defined scalar, originally set to 10,000 by the authors of Attention Is All You Need.

d= Dimension of the output embedding space

0 <= i <d/2 used for mapping column indices (single value for both sin and cos)

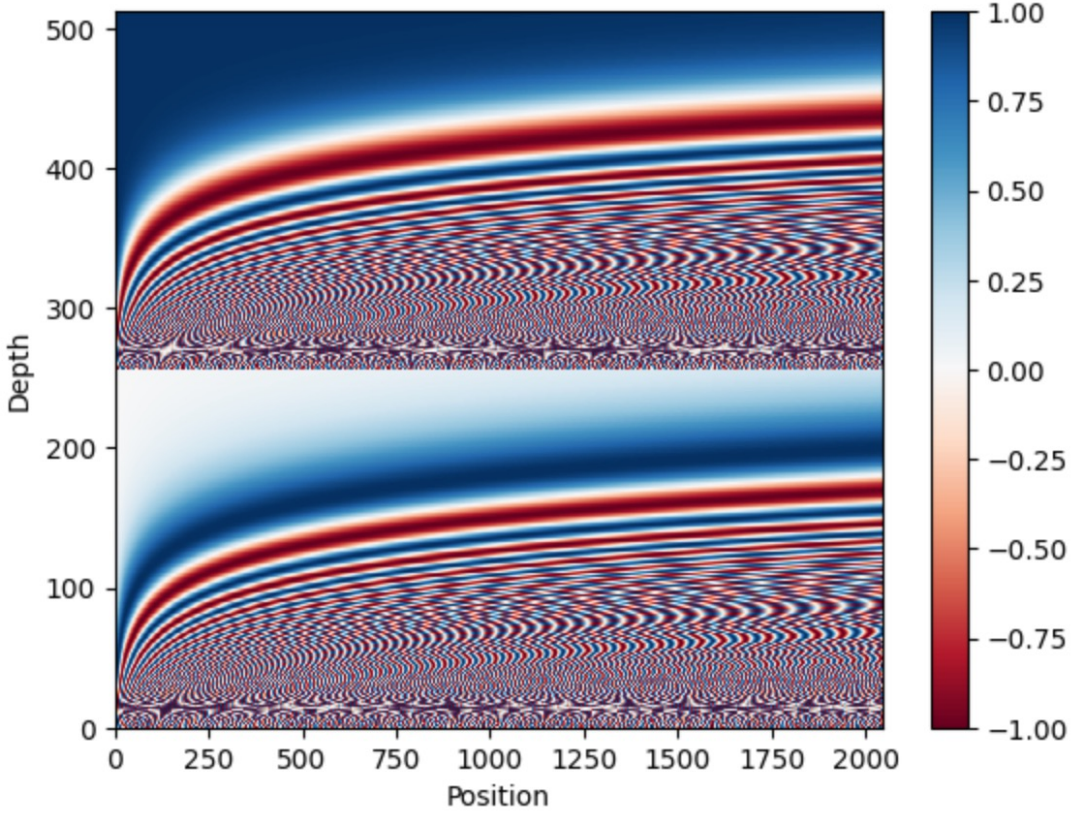
Recent

Positional Encoding Matrix with d=4, n=100

Sequence	Index of token, k	i=0	i=0	i=1	i=1
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

Positional Encoding Matrix for the sequence 'I am a robot'

Positional Embedding Exmaple

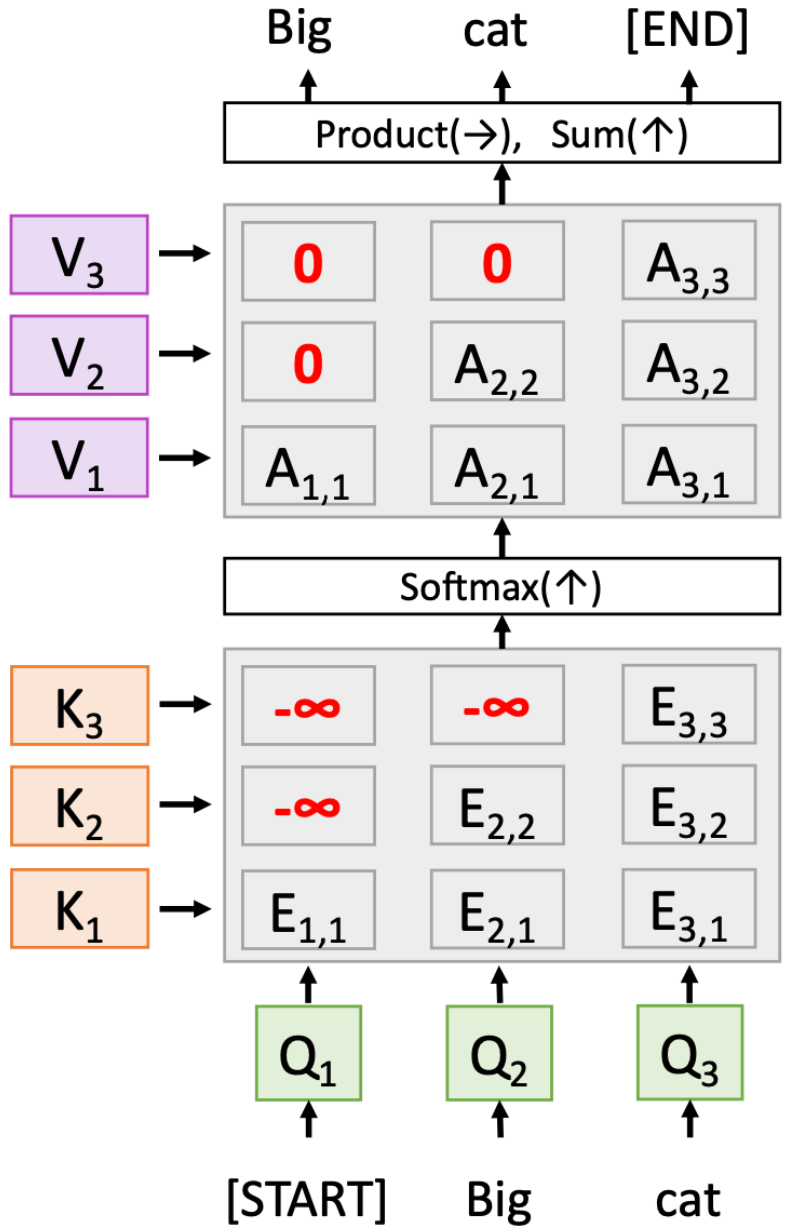


Input sequence →	I	am	a	Robot
	↓	↓	↓	↓
Word embedding →	v_0 = embedding vector(I)	v_1 = embedding vector(am)	v_2 = embedding vector(a)	v_3 = embedding vector(Robot)
	+			
Positional Encoding Matrix →	P_0 = Positional vector(I)	P_1 = Positional vector(am)	P_2 = Positional vector(a)	P_3 = Positional vector(Robot)
	=			
Output of positional encoding layer	y_0 = Positional encoding(I)	y_1 = Positional encoding(am)	y_2 = Positional encoding(a)	y_3 = Positional encoding(Robot)

Masked Self-attention

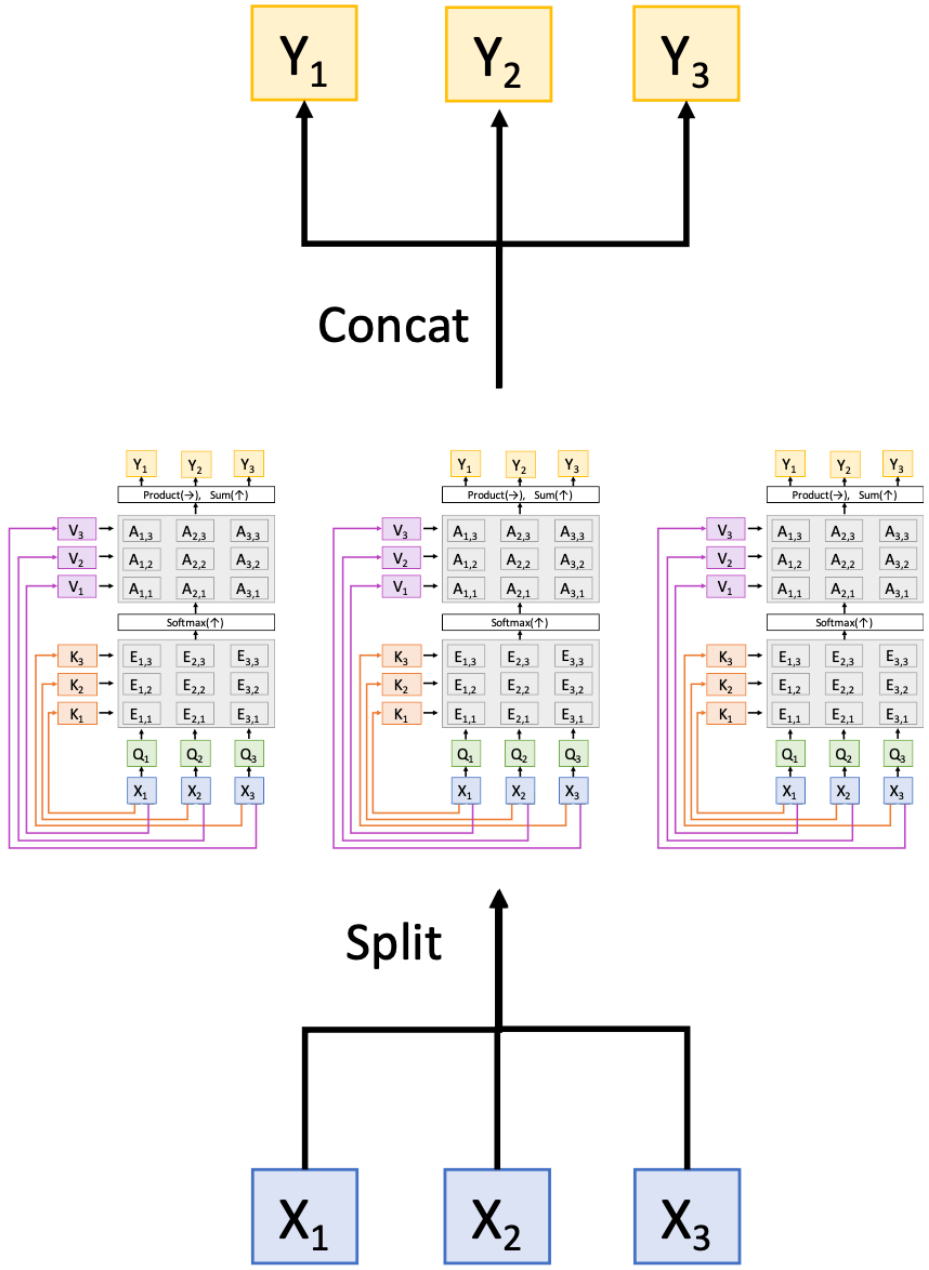
Don’t let vectors “look ahead” in the sequence

Used for language modelling (predict next word)



Multi-head Self-attention

Use H independent “Attention Heads” in parallel (input split into equal chunks)



Hyperparameters:

- Query dimension D_Q
- Number of heads H

Self-Attention is Cheap

	FLOPs	
Self-Attention	$O(\text{length}^2 \cdot \text{dim})$	$= 4 \cdot 10^9$
RNN (LSTM)	$O(\text{length} \cdot \text{dim}^2)$	$= 16 \cdot 10^9$
Convolution	$O(\text{length} \cdot \text{dim}^2 \cdot \text{kernel\_width})$	$= 6 \cdot 10^9$

length=1000 dim=1000 kernel_width=3

Transformer Neural Networks

Transformers (original architecture)

- No recurence (sequential processing)
- Rely on Multi-head self-attention
- Allow attention to operate on all the states in the same layer of the neural network

The Transformer Architecture

Encoder

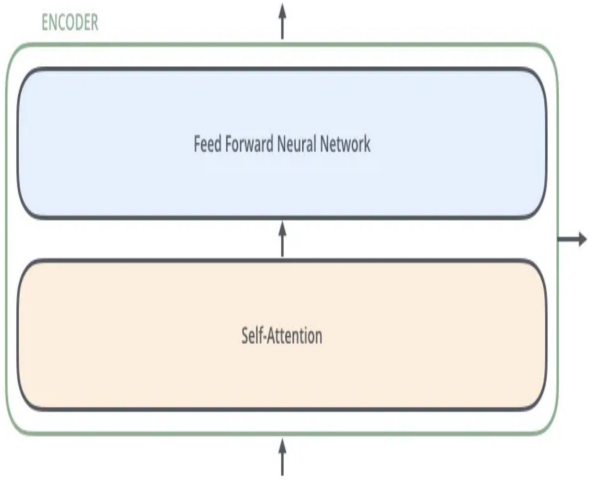
- Converts an input sequence of tokens into a sequence of embedding vectors, often called the hidden state or context

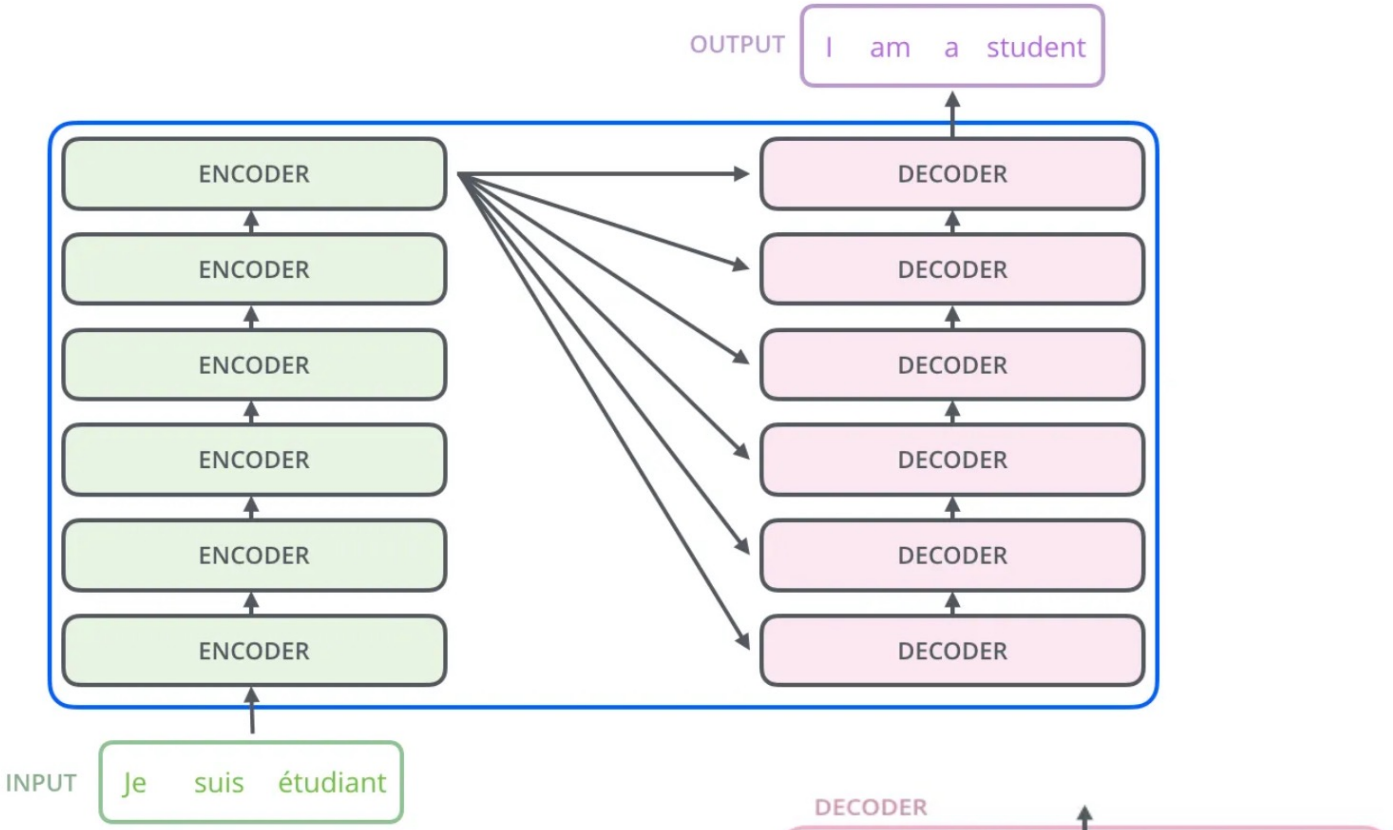
Decoder

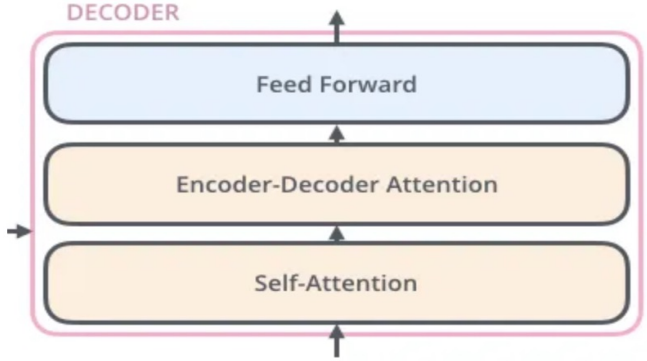
- Uses the encoder's hidden state to *iteratively* generate an output sequence of tokens, one token at a time

- What are Transformer Encoders & Decoders composed of?

Transformer consists of six encoders and six decoders

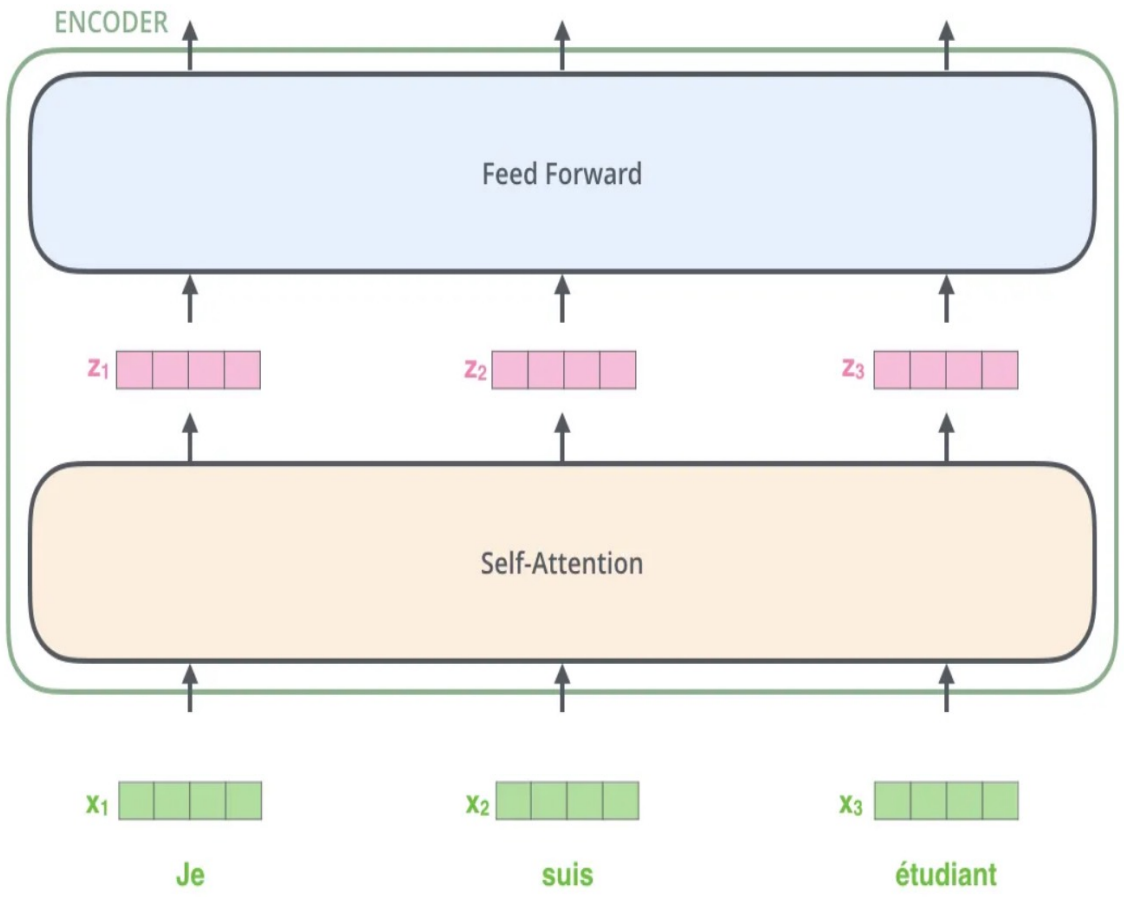


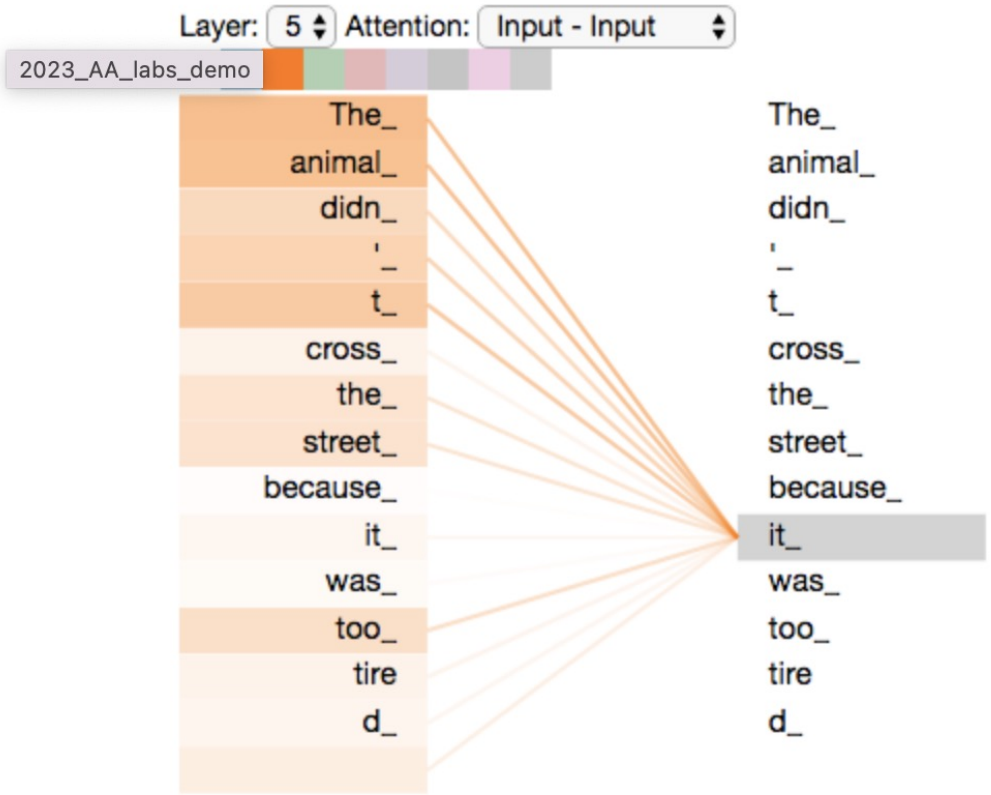




https://jalammar.github.io/illustrated-transformer/

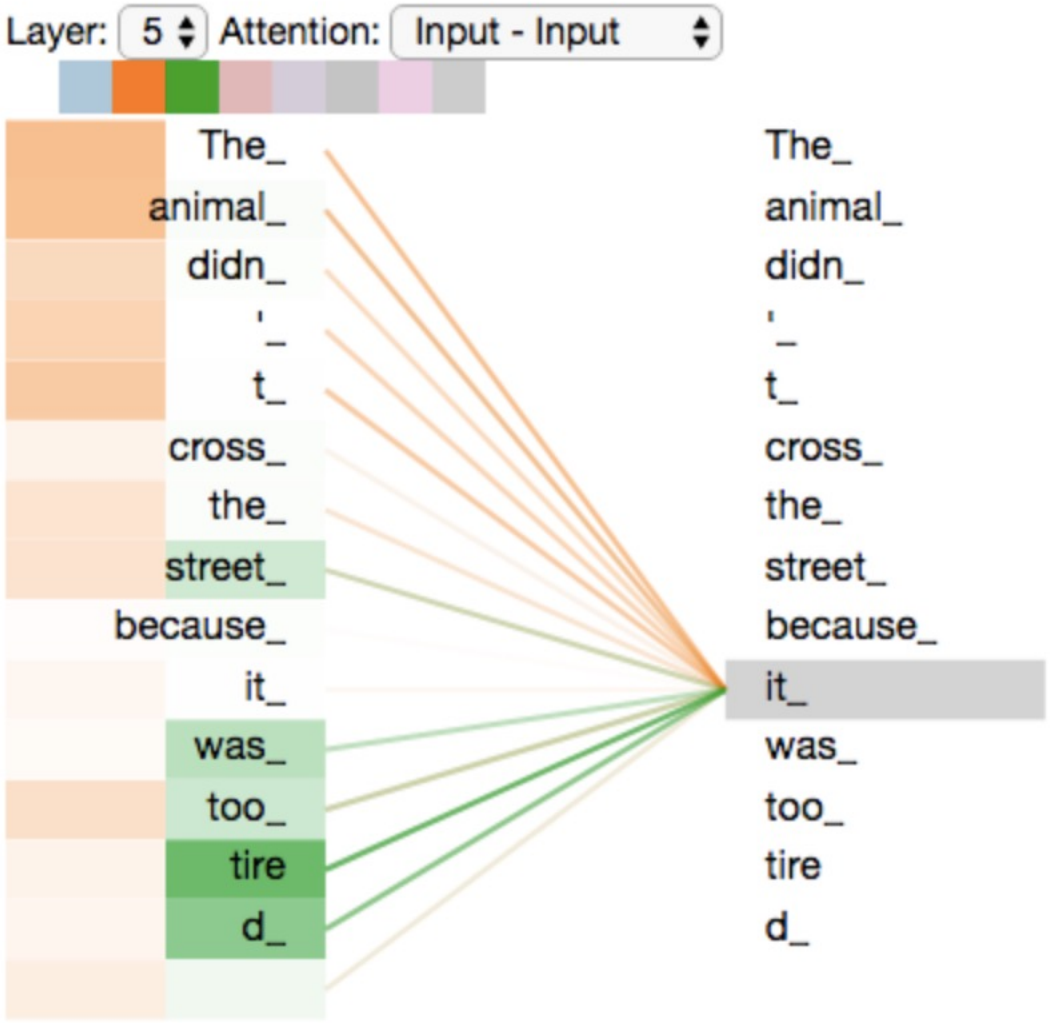
Encoder self-attention



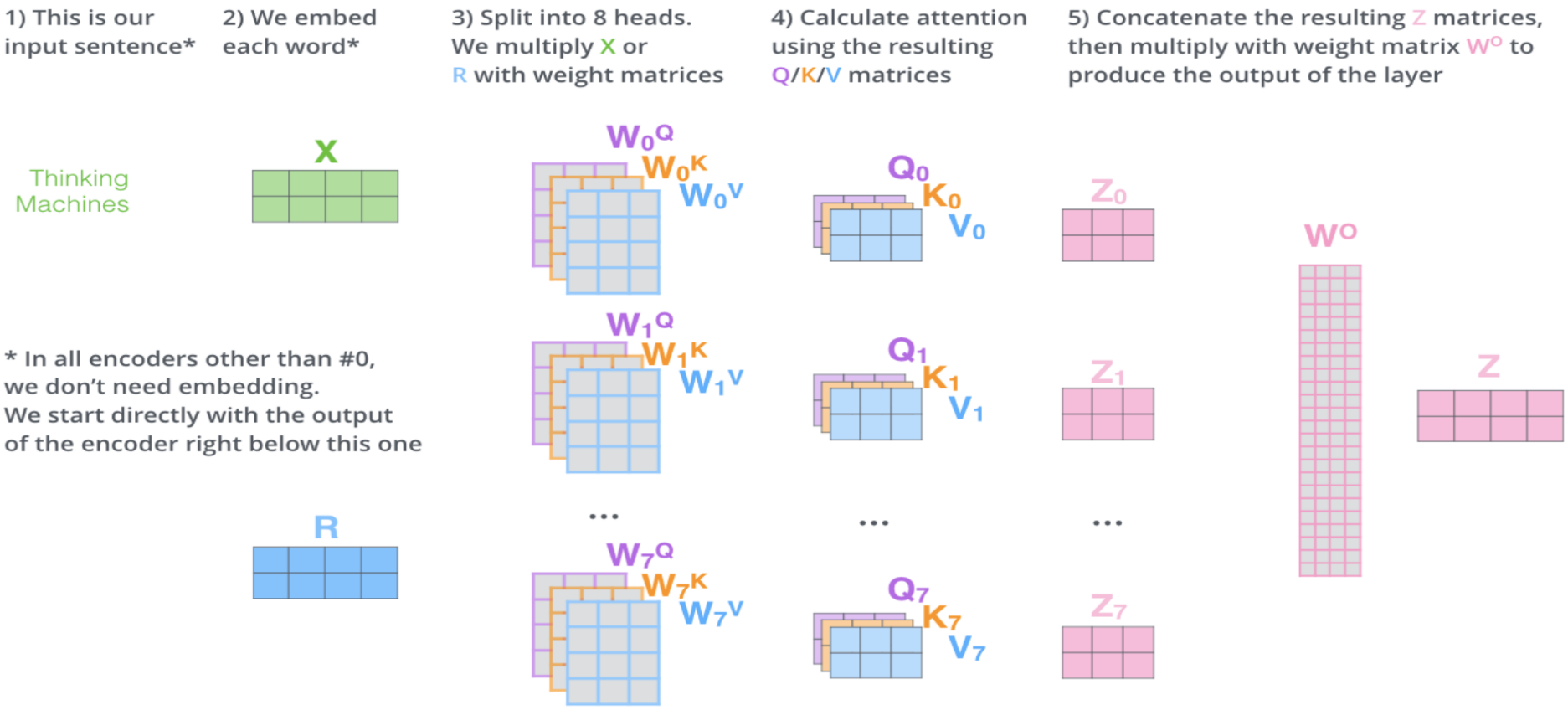


Multi-headed attention

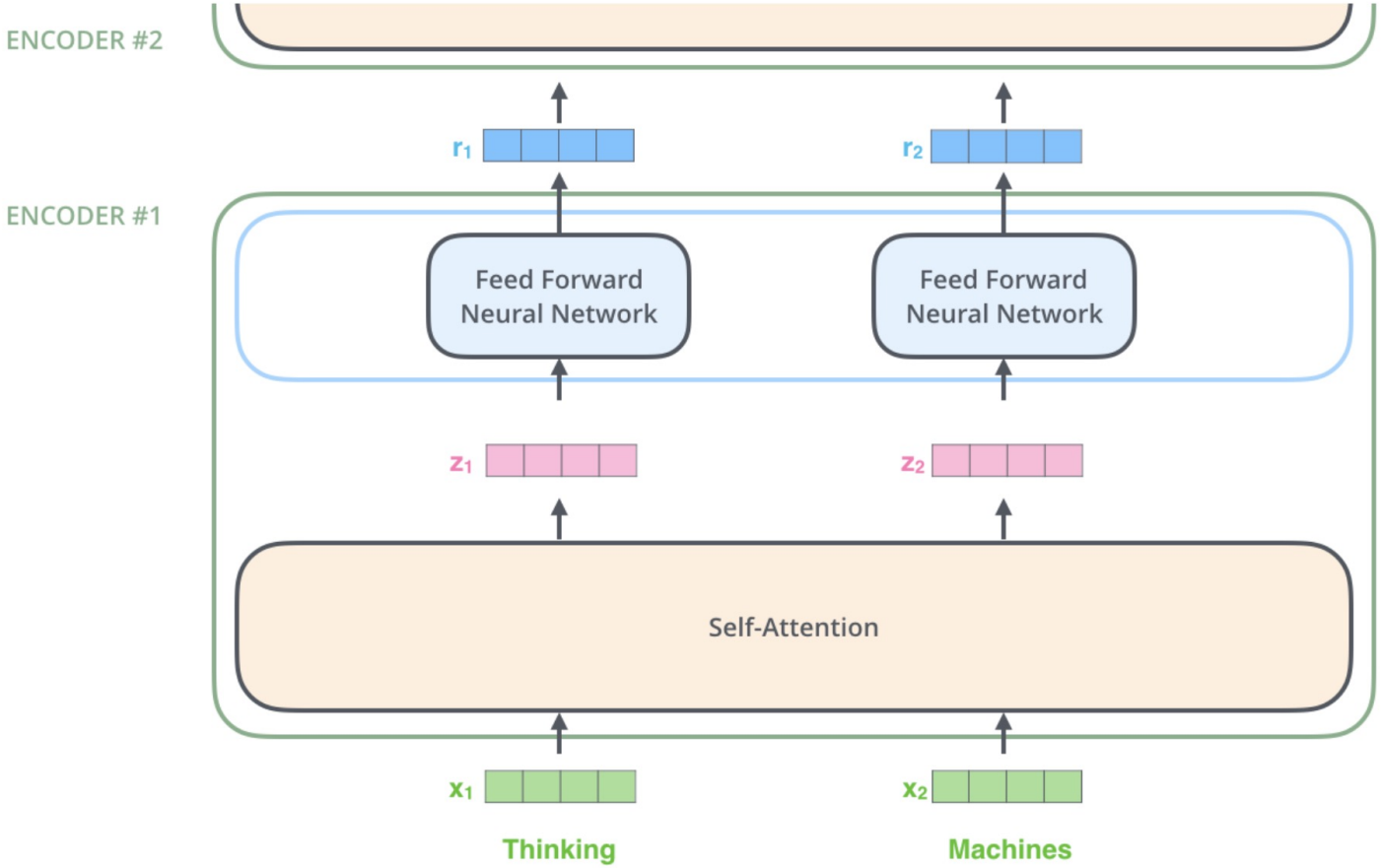
- It gives the attention layer multiple “representation subspaces”
- Therefore multiple sets of Query/Key/Value weight matrices
- The Transformer uses eight attention heads=eight sets for each encoder/decoder



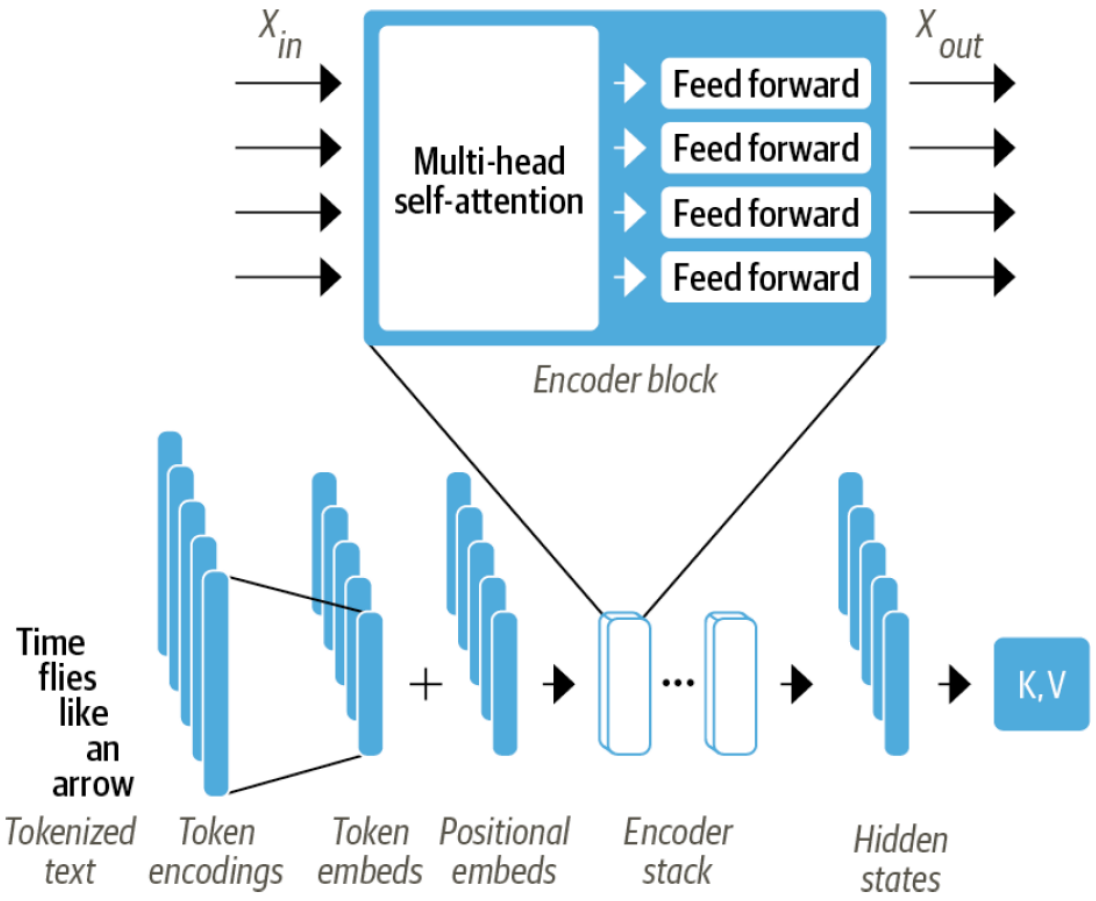
Encoder self-attention (putting it all together)



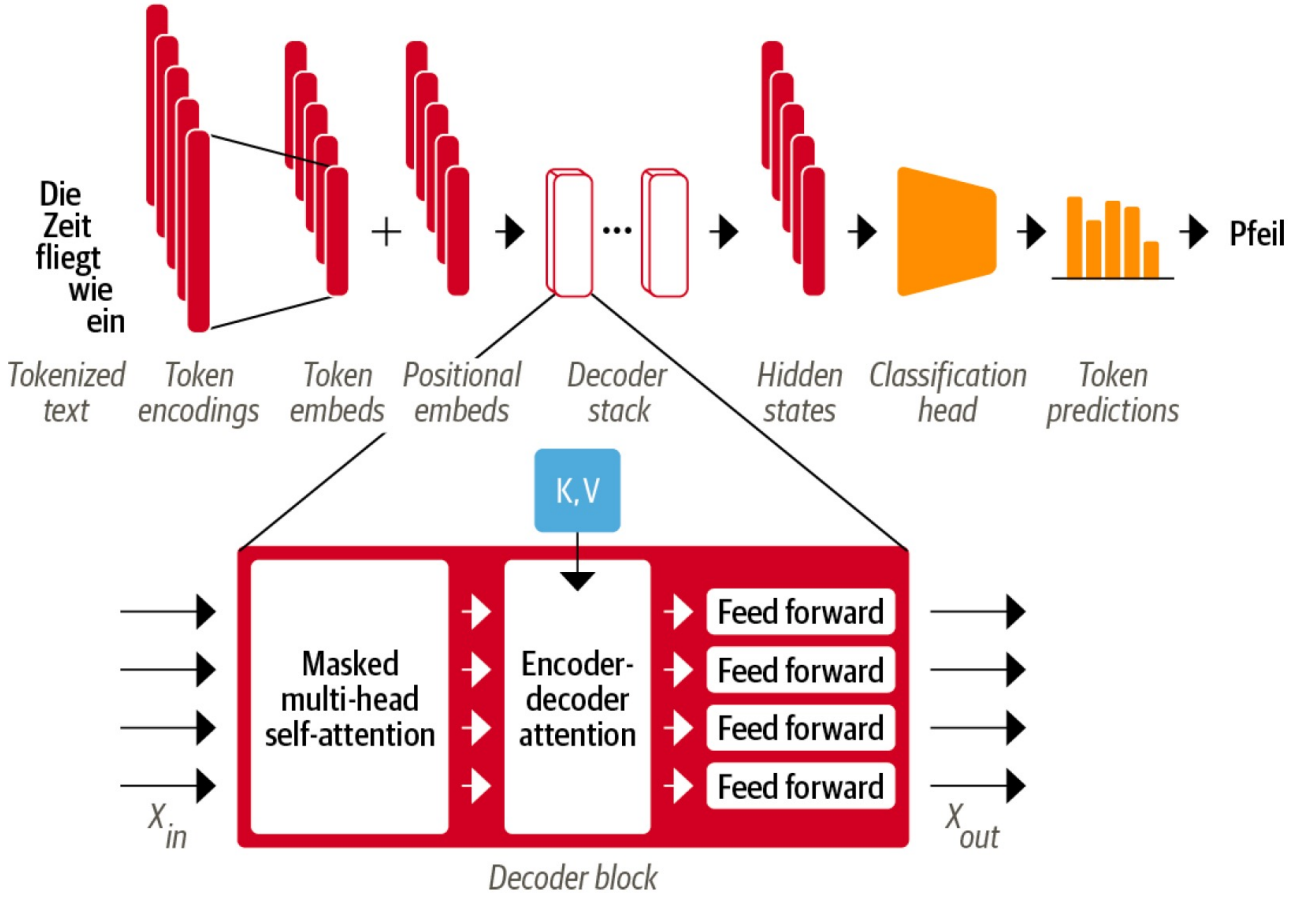
Encoder self-attention (putting it all together)



Transformer Encoder



Transformer Decoder

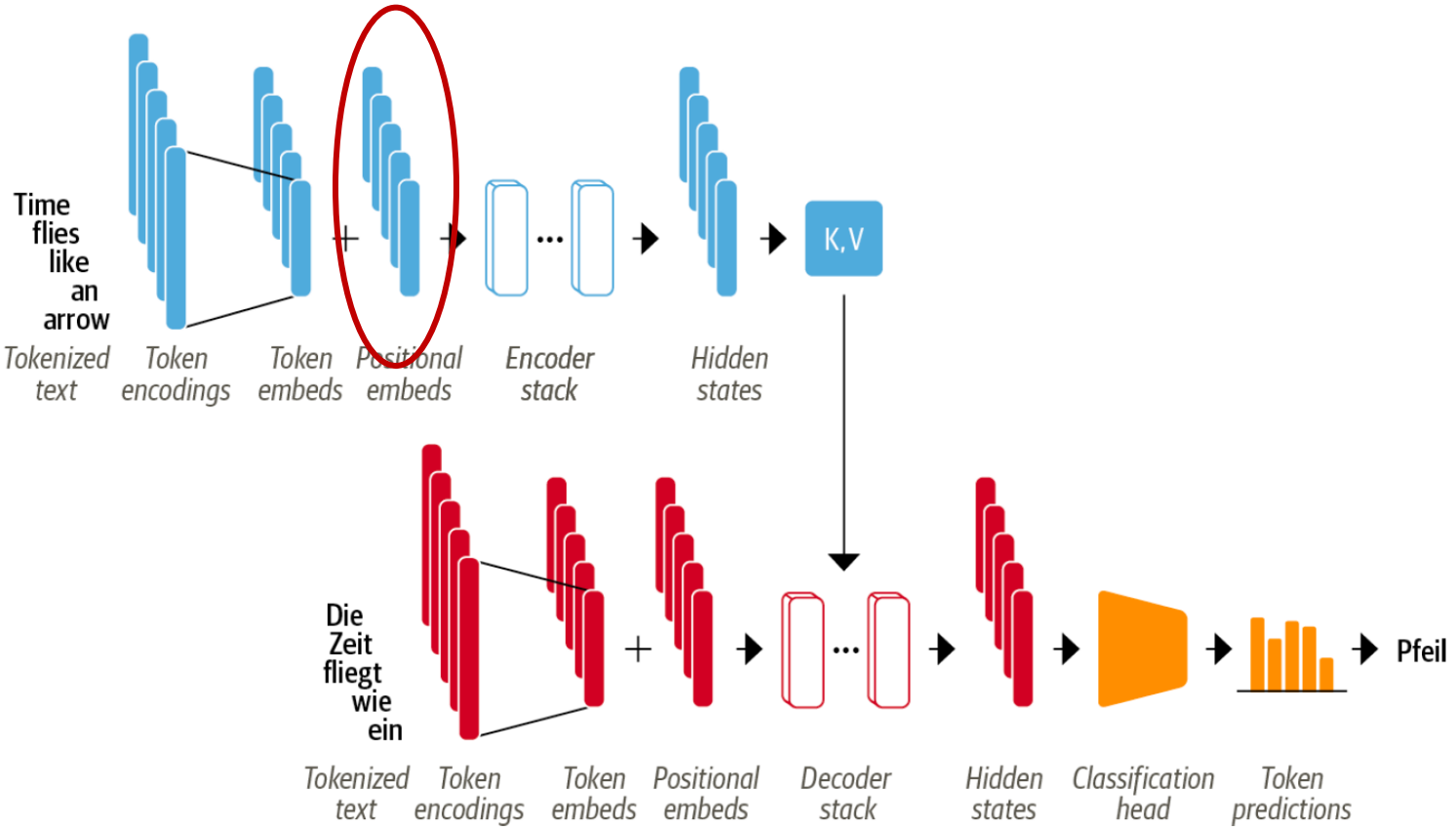


Encoder-decoder attention layer

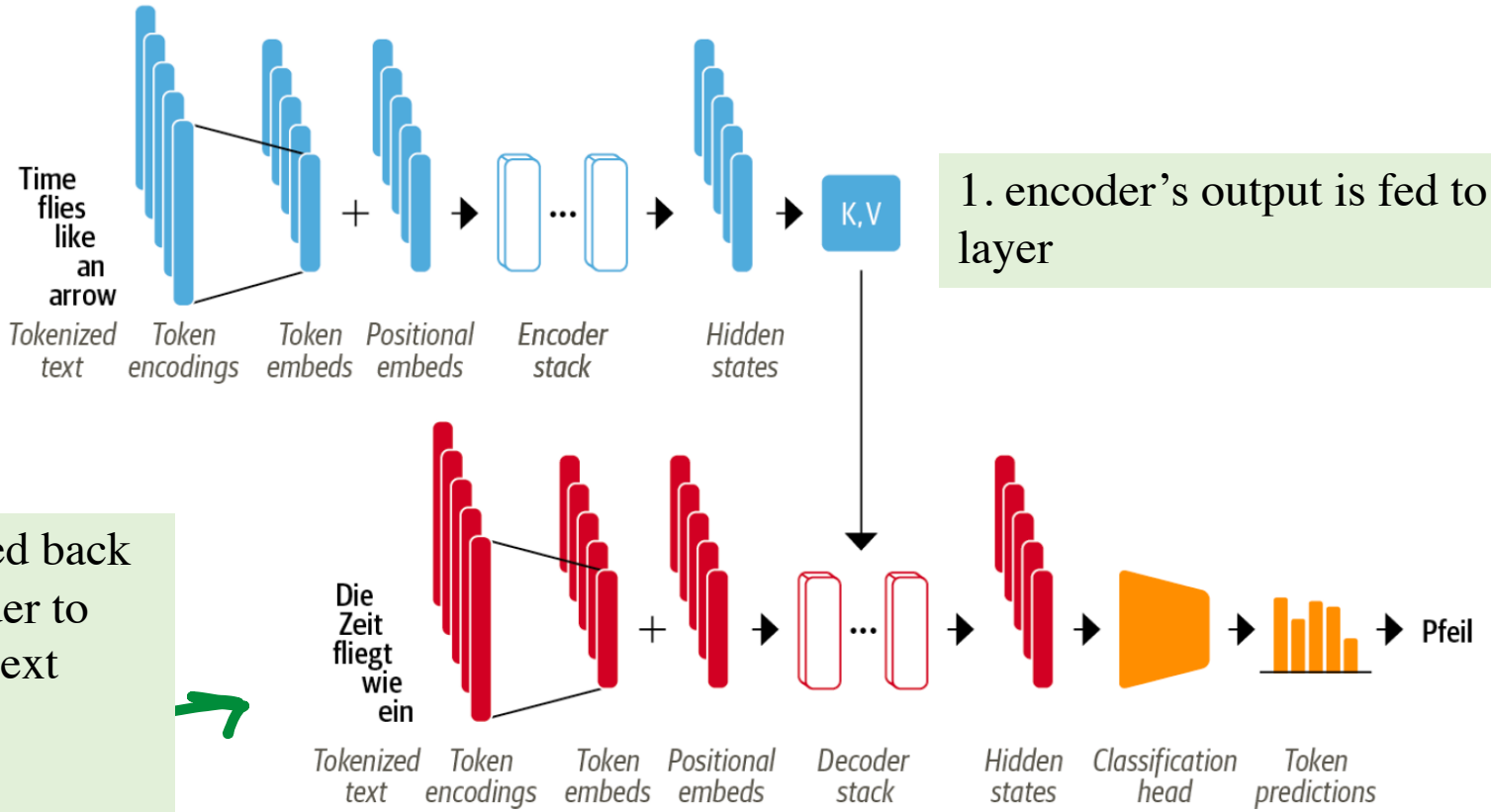
Performs multi-head attention over the output *key* and *value* vectors of the encoder stack, with the intermediate representations of the decoder acting as the queries.

This way the encoder-decoder attention layer learns how to relate tokens from two different sequences, such as two different languages. The decoder has access to the encoder keys and values in each block.

The Transformer Architecture



The Transformer Architecture

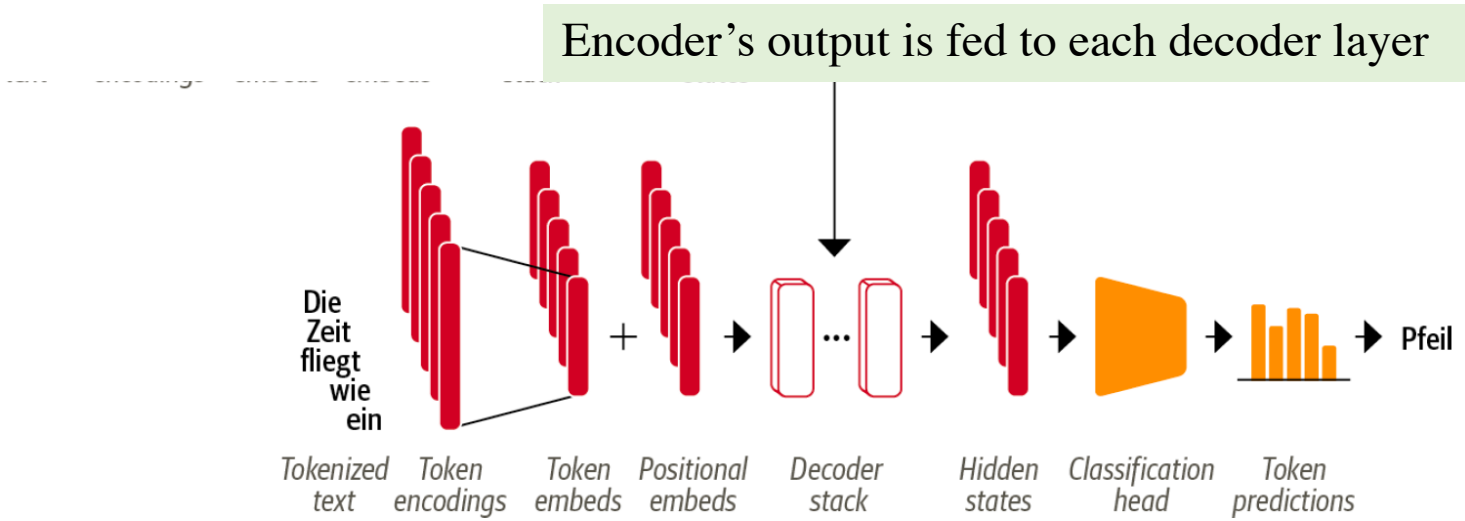


1. encoder's output is fed to each decoder layer

2. decoder then generates a prediction for the most probable next token in the sequence.

3. Output is fed back into the decoder to generate the next token.
Repeat until
a (EOS) token is reached.

The Transformer Architecture- Decoder side



<SOS>	Die
Die	Zeit
Die Zeit	fliegt
Die Zeit fliegt	wie
Die Zeit fliegt wie	ein
Die Zeit fliegt wie ein	Pfeil
Die Zeit fliegt wie ein Pfeil	<EOS>

The Transformer Categories

Originally designed for sequence-to-sequence tasks like machine translation, but both the encoder and decoder blocks were soon adapted as standalone models

Encoder-only

- convert an input sequence of text into a rich numerical representation
- Used for text classification or named entity recognition.
- BERT and its variants, like RoBERTa and DistilBERT

Decoder-only

- The family of GPT models belong to this class.
- models will autocomplete the sequence by iteratively predicting the most probable next word

- **Further Encoders in addition to Transformers:** BART and T5 models
  - suitable for machine translation and summarization tasks

Your best way for a gentle start !

Gentle introduction to Transformers Jay Allamar (channel)

https://jalammar.github.io/illustrated-transformer/ **(Great READ)**

- 8 video series of Transformers & NLP https://www.youtube.com/watch?v=jo1NZ3vCS90&list=PLaJCKi8Nk1hwaMUYxJMiM3jTB2o58A6WY

- ***Beautifully Illustrated: NLP Models from RNN to Transformer, oct 2022***
- *https://towardsdatascience.com/beautifully-illustrated-nlp-models-from-rnn-to-transformer-80d69faf2109*
- ***All you need to know about 'Attention' and 'Transformers' — In-depth Understanding — Part 1 (Feb 2022)****https://towardsdatascience.com/all-you-need-to-know-about-attention-and-transformers-in-depth-understanding-part-1-552f0b41d021*

Practical Part

- Recommended Tensorflow tutorial & case studies

- Neural machine translation with attention (translate text with seq2seq models)
  https://www.tensorflow.org/text/tutorials/nmt_with_attention

- Neural machine translation with a Transformer and Keras (translate text with transformer models

https://www.tensorflow.org/text/tutorials/transformer

Thank you!