Attention is All You Need

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I.
NIPS 2017

David Semedo

df.semedo@campus.fct.unl.pt

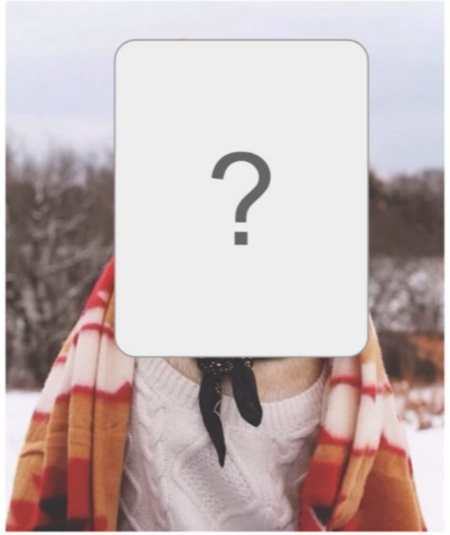
NOVA Search Reading group

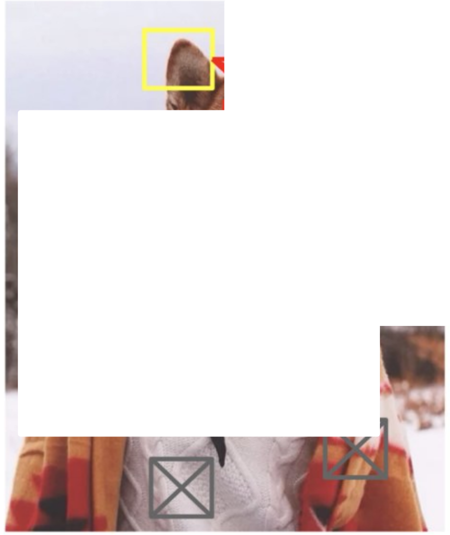
First of all, why Attention?

Let's consider a simple example:

She is eating a

Now with an image:



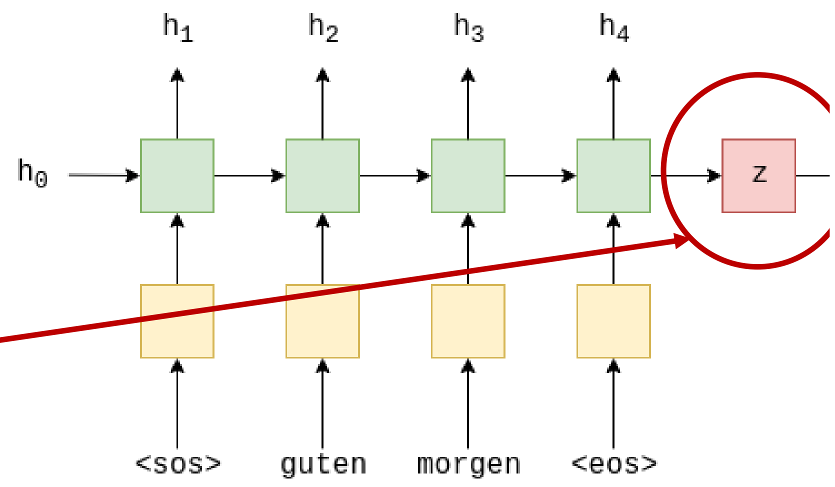


Seq2Seq Models

Encoder: Produce a context vector that compresses information regarding the whole sentence.

Decoder: Takes the context vector z and an input word, and performs a prediction.

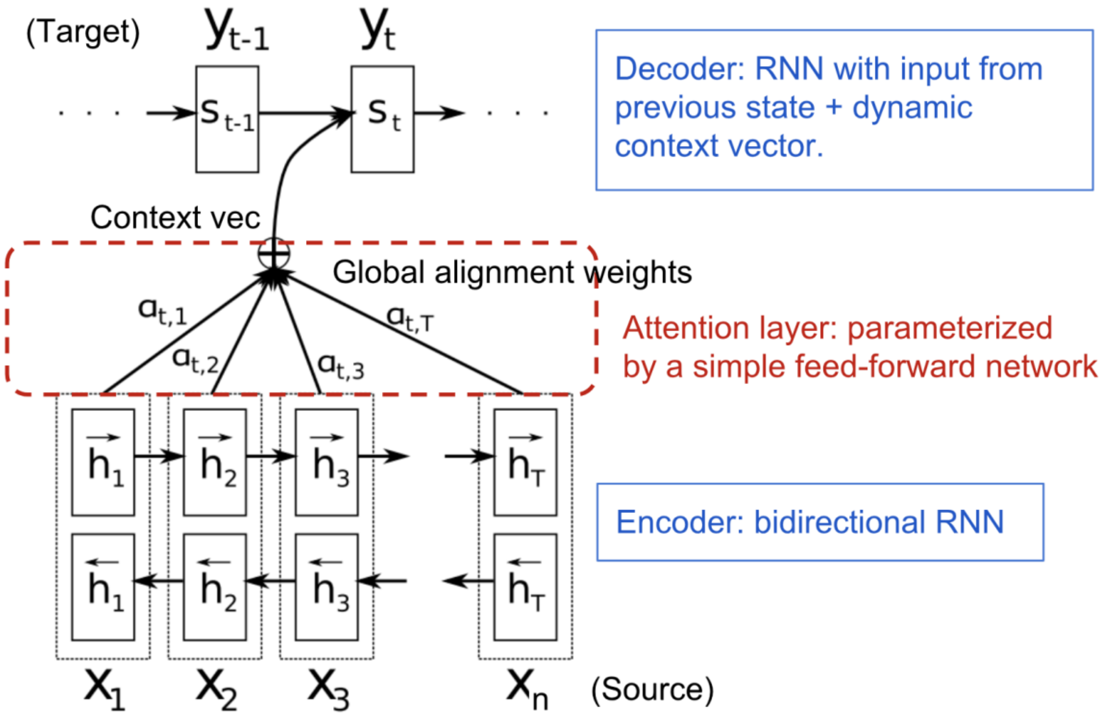
German to English



Bottleneck!

*Sutskever, Ilya et al. "Sequence to Sequence Learning with Neural Networks." **NIPS 2014**.*

Seq2Seq Models and (Additive) Attention



Let's check the math:

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$

$$\alpha_{t,i} = \text{align}(y_t, x_i) = \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}$$

Scoring functions:

$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$ (Additive)

$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ (General)

$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$ (Dot)

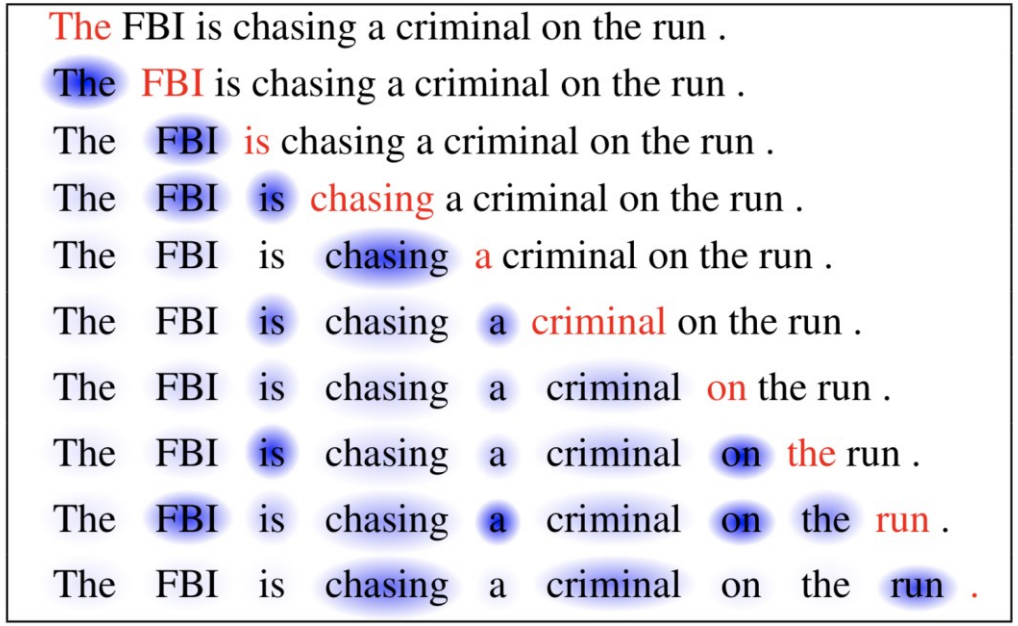
SOTA performance on Machine Translation! (in 2015)

*Bahdanau, Dzmitry et al. "Neural Machine Translation by Jointly Learning to Align and Translate." **ICLR 2015***

What if you only have one sentence? **Self-Attention**!

Language modeling, Sentiment Analysis, …

Forwarding a sentence - Red is the current word, blue intensity is the activation level:



Cheng, Jianpeng et al. "Long Short-Term Memory-Networks for Machine Reading." ***EMNLP 2016***
Lin, Zhouhan et al. "A Structured Self-attentive Sentence Embedding." ***ICLR 2017***

Attention checkpoint

What was accomplished:

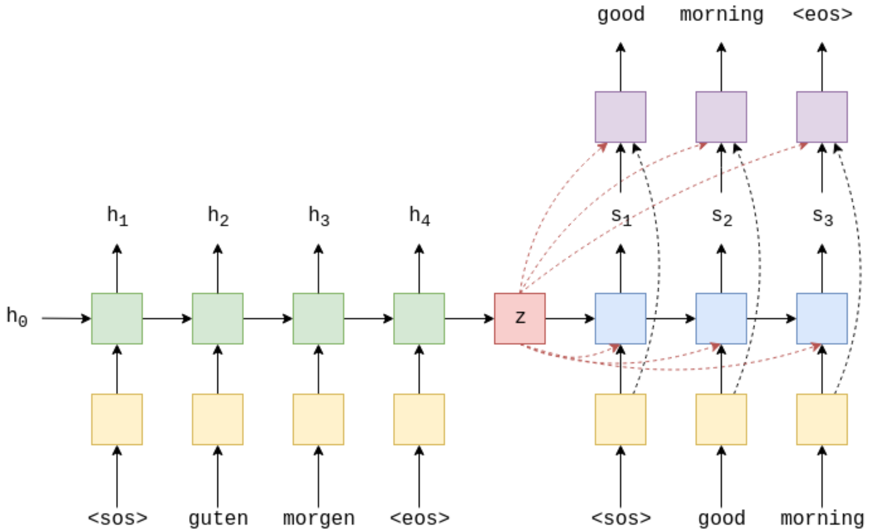
- Sequence models are able to attend to specific parts of a sentence when predicting.
- Attention helps dealing with long-term dependencies problems.
- Self-attention: attend to all words of an input sentence or all regions of an image.

Significant improvements in NLP and Vision tasks

What is still the main limitation?

<u>Recurrent architectures!</u>

- Gradient exploding/vanishing.
- Hard to parallelize.



Transformer

SOTA performance on Machine Translation (in 2017)

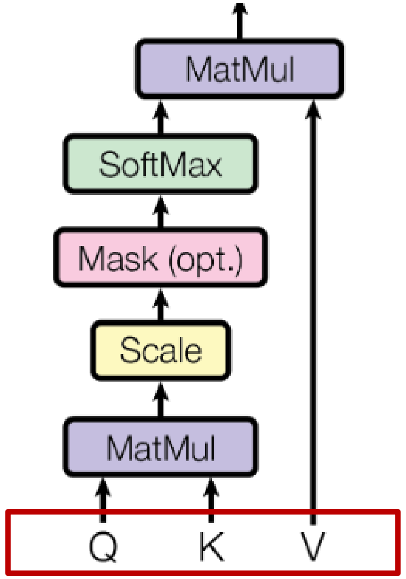
Its a neural architecture: the combination of several components resulted in a highly effective block for sequence modeling.

Major achievement: Recurrence is avoided by heavily relying on the self-attention mechanism.

Vaswani, Ashish et al. "Attention is All you Need." ***NIPS 2017***

Transformer – Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Query (Q), Key (K), Value (V)

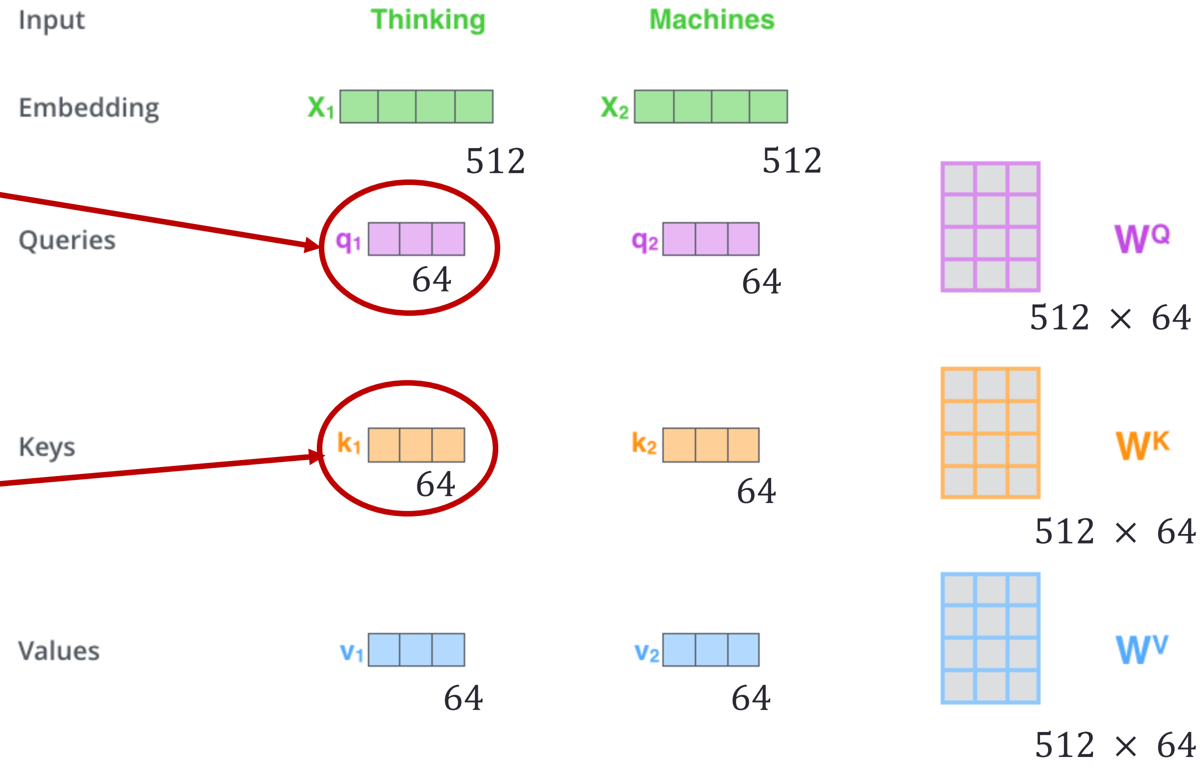
These are just abstractions, but .. let's think of them as a dictionary

$q_1 = x_1.W^q$

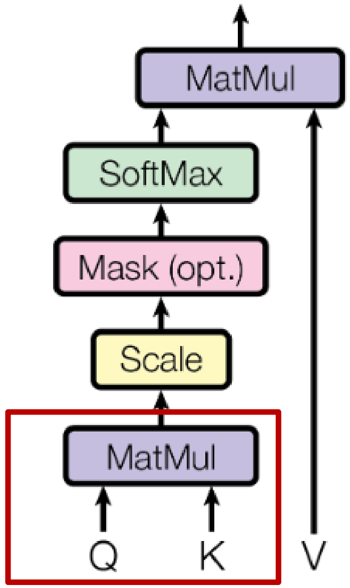
...

$k_1 = x_1.W^K$

...



Transformer – Scaled Dot-Product Attention



Query (Q), Key (K), Value (V)

These are just abstractions, but .. let's think of them as a dictionary

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{\boxed{QK^T}}{\sqrt{d_k}}\right)V$$

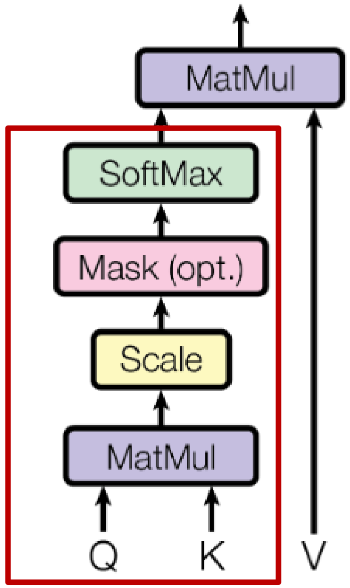
Compare (dot product) each word in the input sentence (query), with all the words we have in the sentence (keys)

Given an input word w, score it against all the other words in the input sentence.

Transformer – Scaled Dot-Product Attention

Query (Q), Key (K), Value (V)

These are just abstractions, but ..
let's think of them as a dictionary



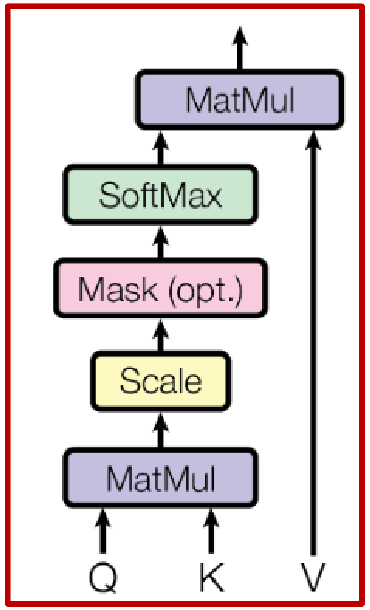
$$\text{Attention}(Q, K, V) = \boxed{\text{softmax}(\frac{QK^T}{\sqrt{d_k}})}V$$

softmax creates a probability distribution over the scores.

Normalizes all scores such that they are all positive and add up to 1.

Determines how much each word will be expressed at a given position.

Transformer – Scaled Dot-Product Attention



Query (Q), Key (K), Value (V)

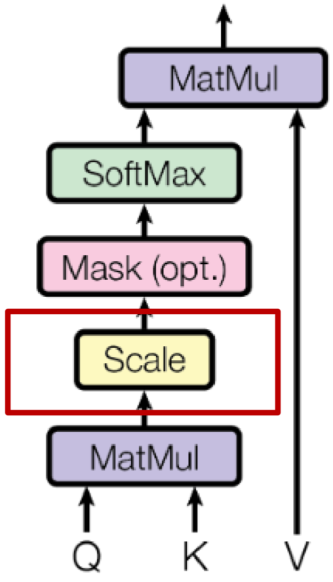
These are just abstractions, but .. let's think of them as a dictionary

$$\text{Attention}(Q, K, V) = \boxed{\text{softmax}(\frac{QK^T}{\sqrt{d_k}})V}$$

Compute a linear combination of the values V (all words in a sentence), using the outputs of the softmax as weights!

Why having the matrix V? It is important to keep intact the embeddings of values (V) -> We lost them on the lookup phase.

Transformer – Scaled Dot-Product Attention



Revisiting the scaling factor:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\boxed{\sqrt{d_k}}}\right)V$$

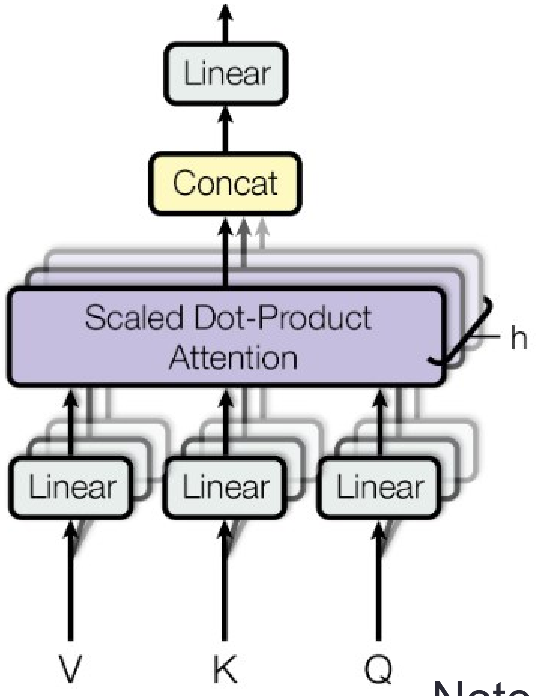
This scaling helps obtaining more stable gradients.
However, it does not affect the intuition of self-attention.

For high dimensional (d_k) vectors, dot products grow large magnitude, pushing the *softmax* into regions that have very small gradients.

The dot product $q.k = \sum_{i=1}^{d_k} q_i \,.\, k_i$, has mean 0 and variance d_k.
(Assuming that q and k are independent random variables with mean 0 and variance 1.)

Transformer – Multi-head Attention

Use $h = 8$ Scaled Dot-Product Attention blocks.



Allows the model to jointly attend to information from different representation subspaces at different positions.

With a single attention head, averaging inhibits this.

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \underline{\mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)}$$

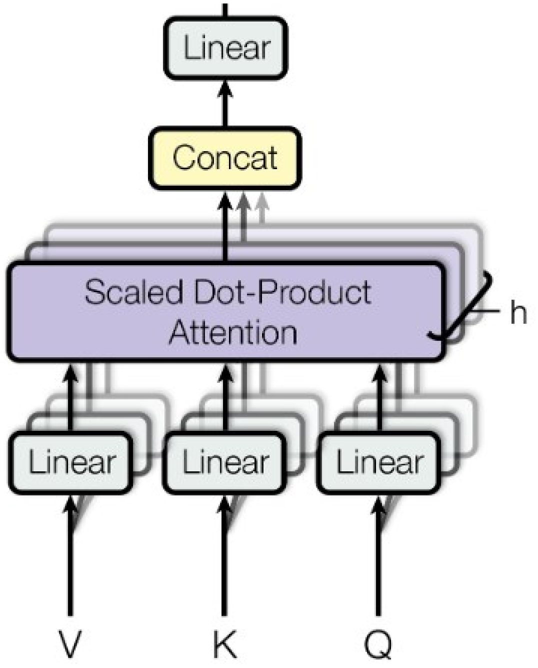
Scaled Dot-Product attention block.

$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$

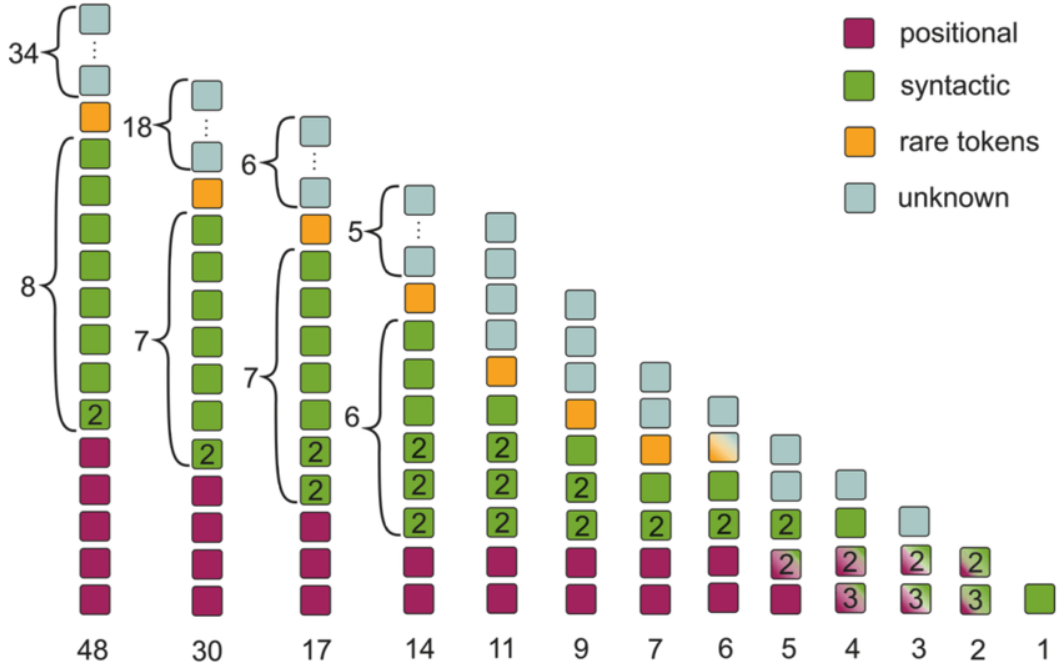
Note that each attention block is **independent**, meaning that this can be easily parallelized!

Transformer – Multi-head Attention

Use $h = 8$ Scaled Dot-Product Attention blocks.

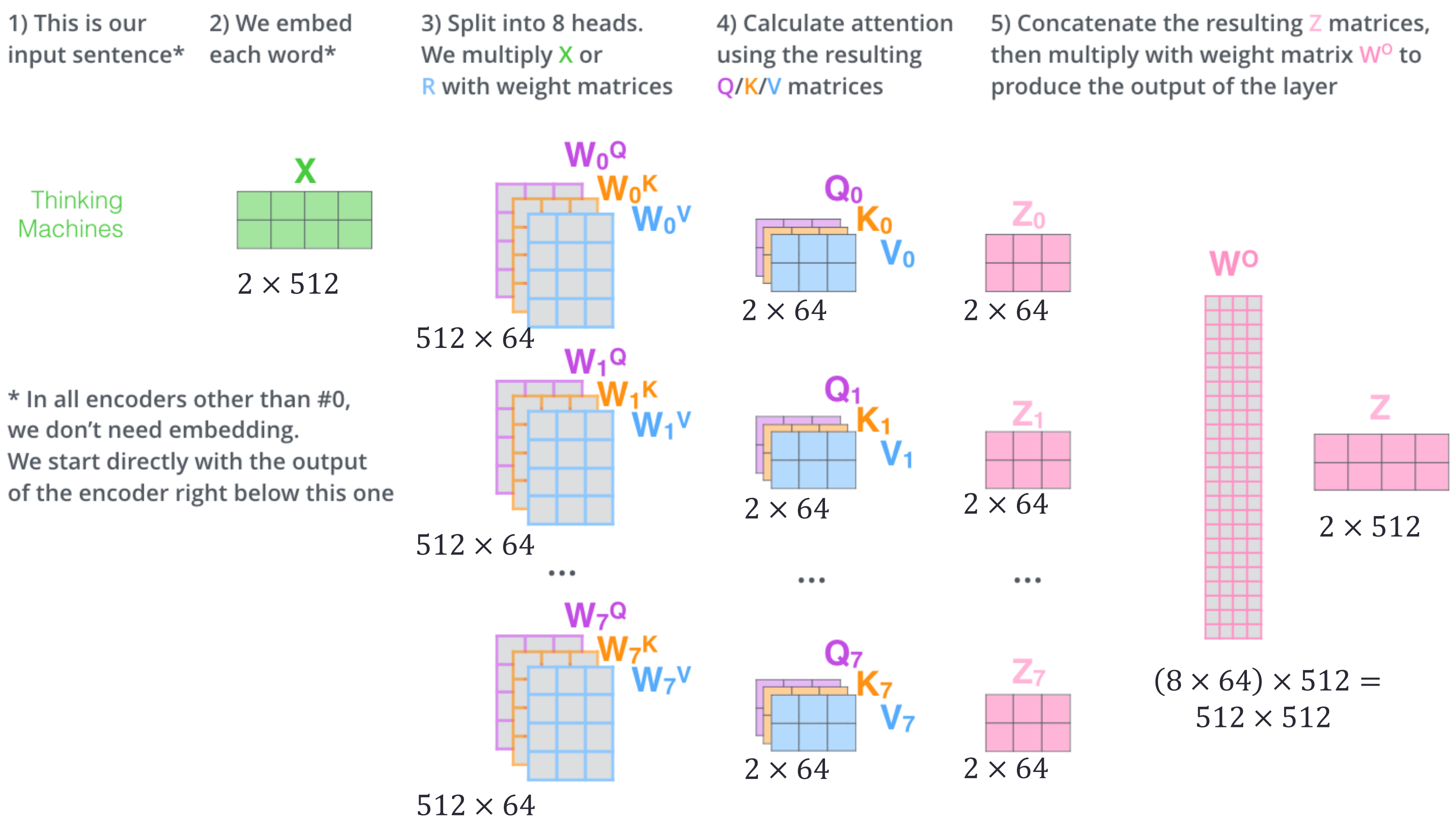


What are the different heads doing?



Voita, Elena et al. "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned." ***ACL 2019***

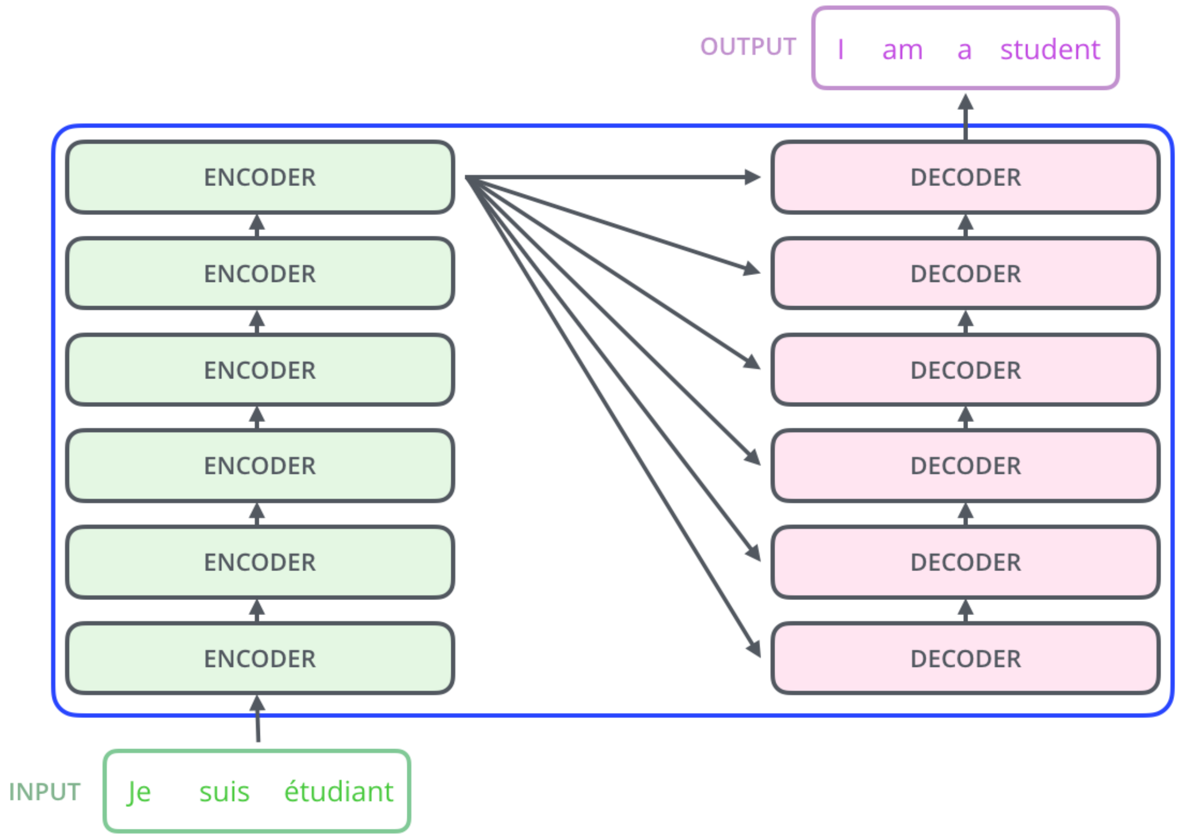
Transformer – Multi-head Attention



Transformer

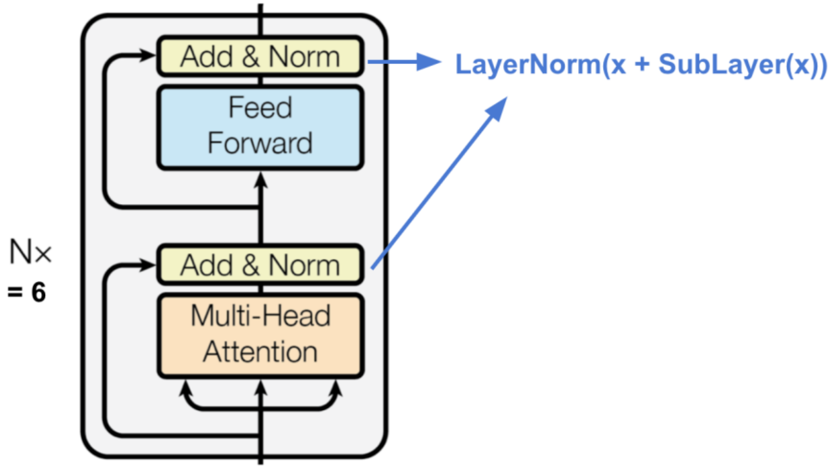
(Stacked) Encoder-Decoder Structure

- 6 stacked identical and independent encoders
- 6 stacked identical and independent decoders



Transformer - Encoder

Two blocks:
- Multi-Head attention + Add & Norm
- Position-wise Feed Forward + Add & Norm



Position-wise FF net: linear feedforward network applied to each position, independently.

Norm corresponds to Layer Normalization [1]. Variant of batch normalization computed on a single example, thus being independent of the mini-batch size.

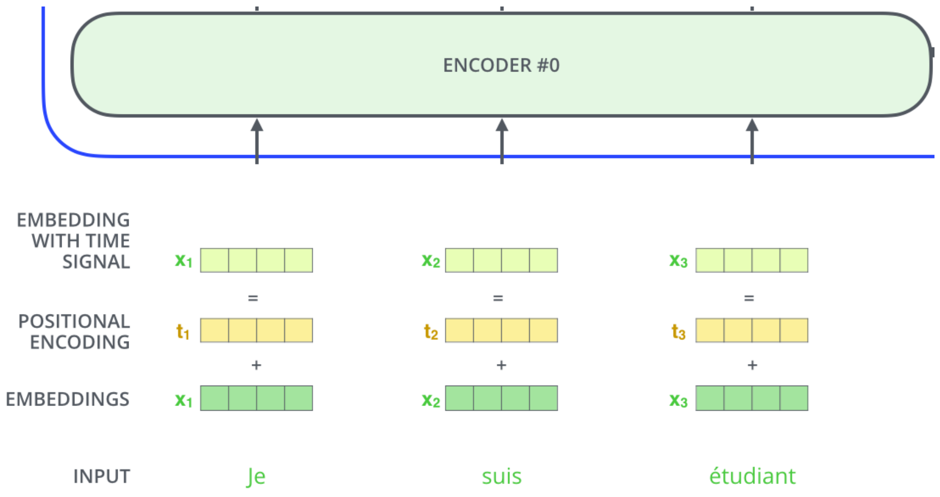
Residual connection: same reason as in ResNets. Learn the residual H(x)+x. Help gradients flow through a “simpler” path, during backpropagation.

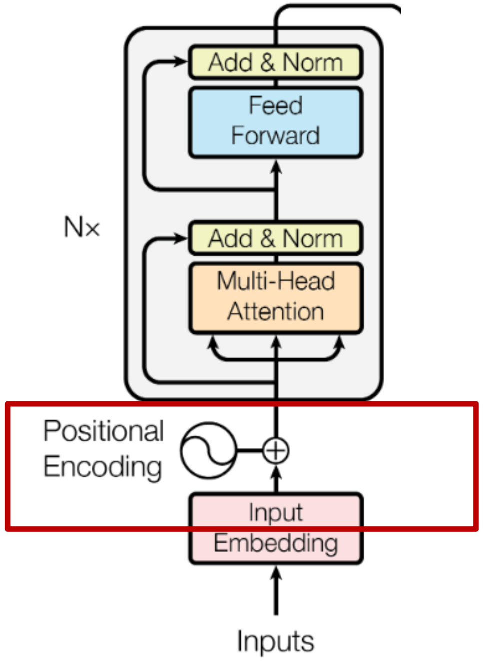
[1] Ba, Jimmy et al. “Layer Normalization.” ArXiv abs/1607.06450 2016

Transformer – Encoder – Positional Encoding

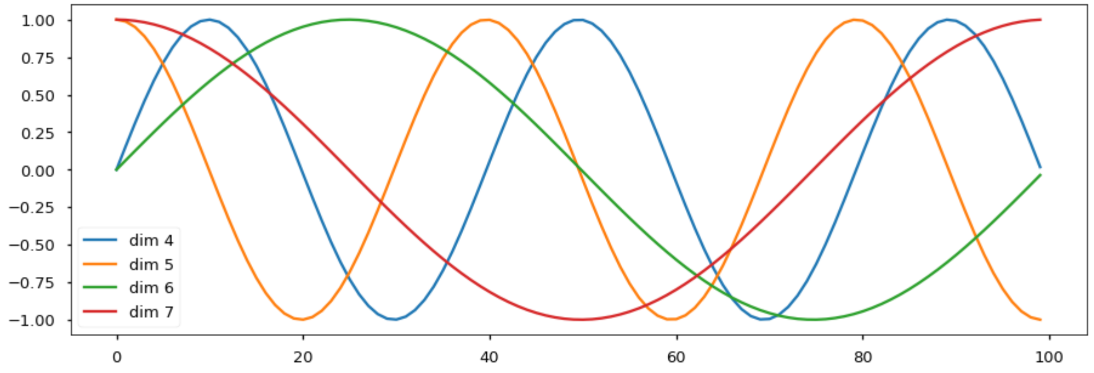
Why? N-grams!

Donald Trump is the president of the United States of America.





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



Each dimension i of the positional encoding corresponds to a sinusoid.

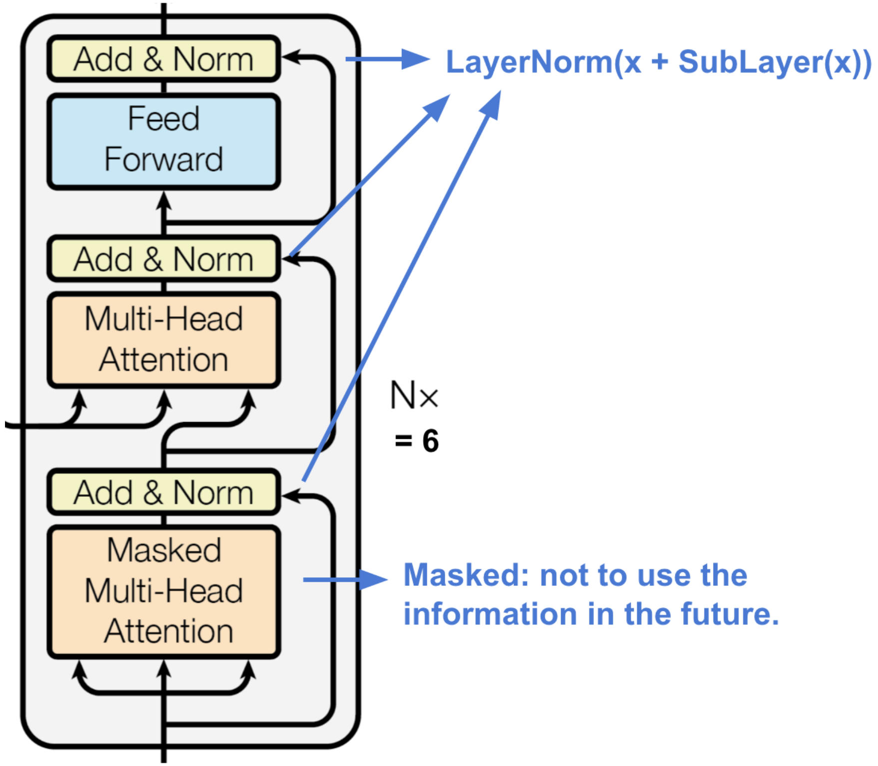
Allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

Transformer - Decoder

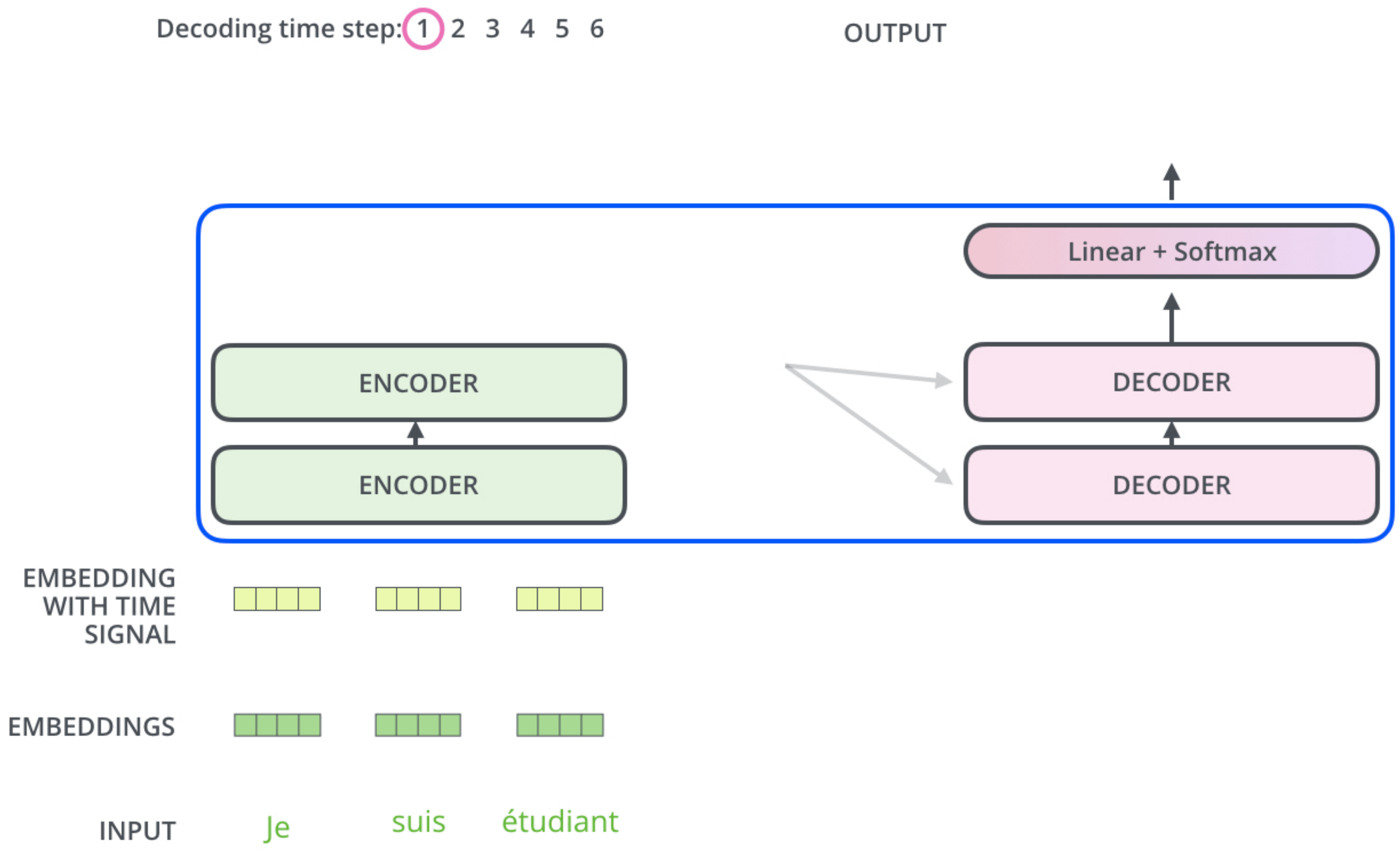
Three blocks:

- Multi-Head attention + Add & Norm -> Takes the output of the encoder
- Position-wise Feed Forward + Add & Norm
- Masked Multi-head attention + Add & Norm -> Takes as input a word from the target sequence.

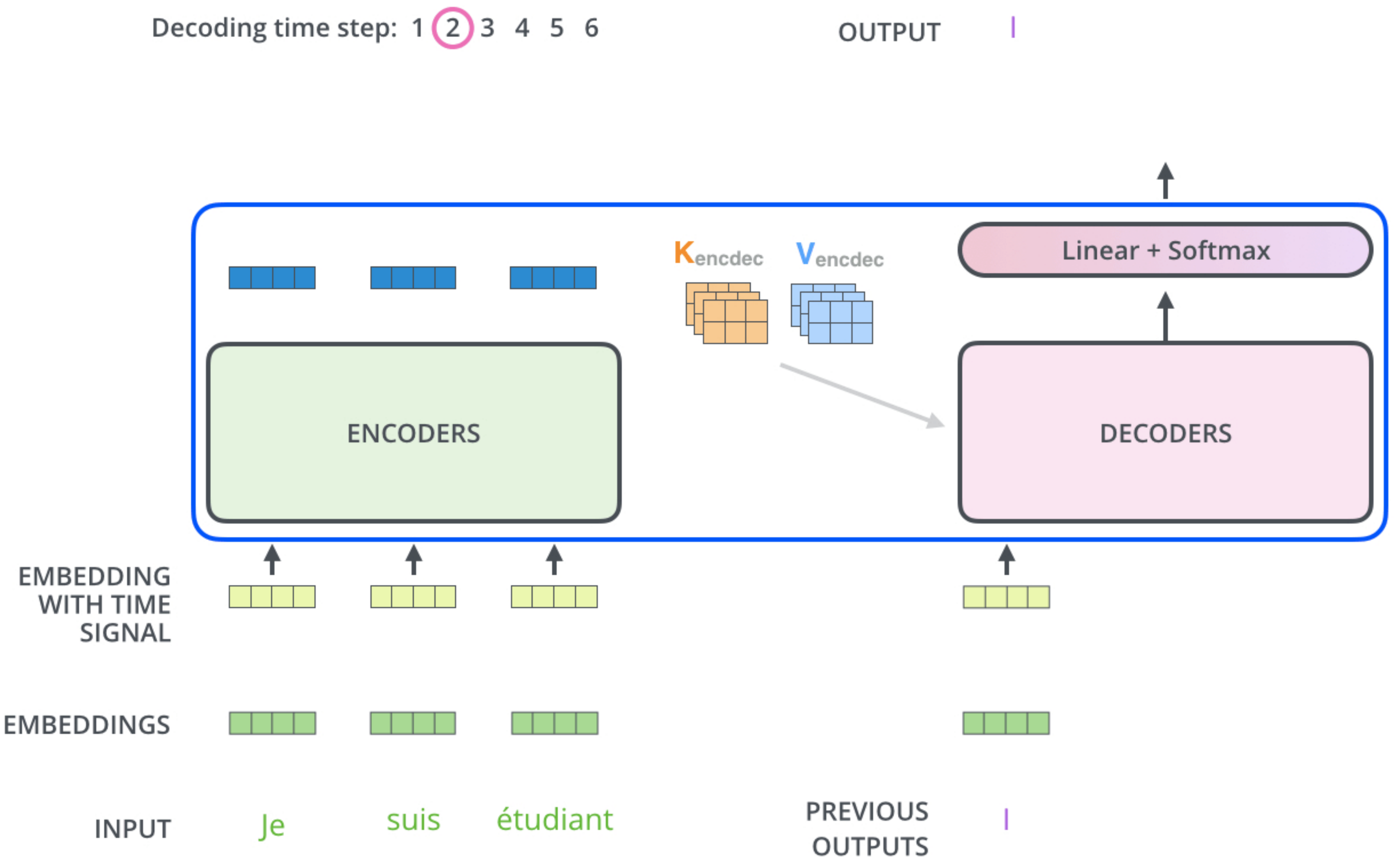
Queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder.



Transformer – Full Architecture flow



Transformer – Full Architecture flow



Transformer – (Personal) Architecture Intuition

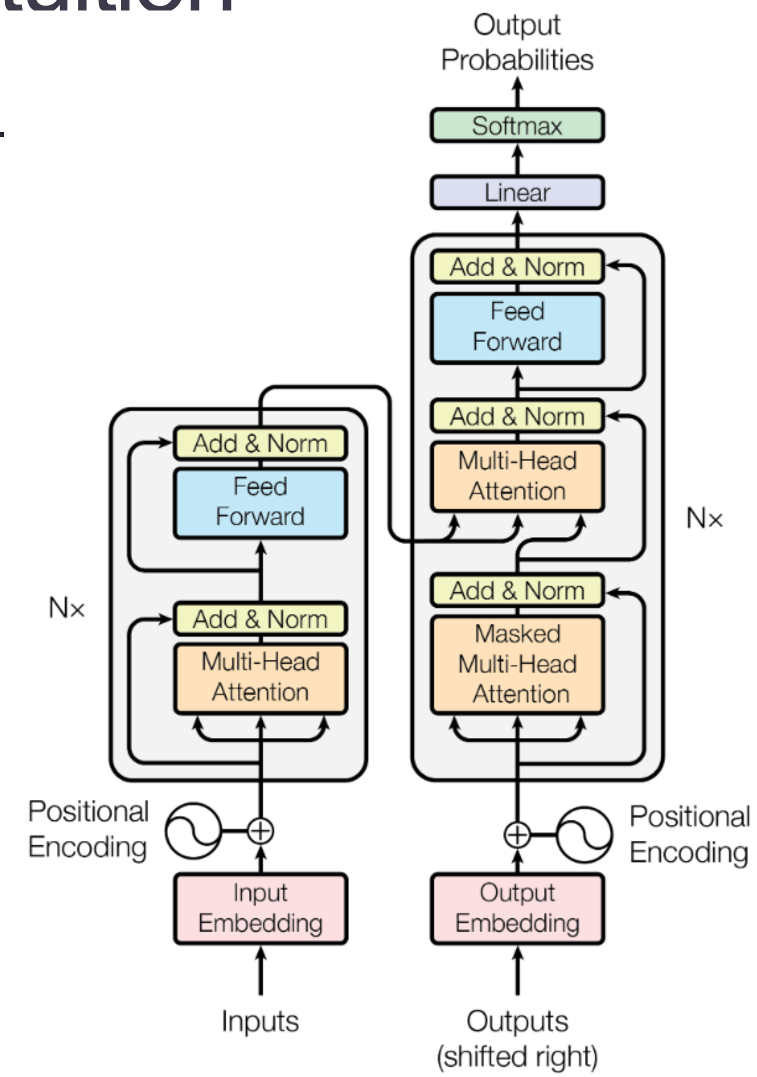
Stacked blocks: the deeper the model is, the higher the non-linearity aspect.

FFN: Perform non-linear transformation after self-attention, to obtain better representation.

Residual connections: Allow disabling Multi-Head Attention and/or FFN, throughout the whole stack. Help gradients propagation during training.

Core block of A - (Attention+Add & Norm) and B - (FFN+Add & Norm):

- For each input word w, get an intermediate embedding e that summarizes interaction between w and the whole sentence.
- Transform e by applying it non-linear transformation(s).



Results

Evaluated on the task of Machine Translation.

Datasets: WMT 2014 English-German (4.5M pairs) and WMT 2014 English-French (36M pairs).

Some training details:

- Adam w/ a custom scheduler: increases lr in first steps (warmup), then starts decreasing it.
- Regularization: dropout for all sub-layers.
- Batches created based on sequence length. ~25000 tokens/batch

Results – Machine Translation

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 [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

What happen next? Limitations and Extensions

Improved performance at reduced computational cost. Became widely adopted not only in NLP but also other fields (e.g. vision).

The Transformer has been used as a building block on many successful architectures (e.g. BERT, and its variants).

Limitations and Extensions

Can only deal with fixed-length sequences -> Transformer-XL [1] addresses this issue.

With the positional encoding scheme used, order is weakly incorporated. BERT learns the positional encodings.

*[1] Dai, Zihang et al. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context." **ACL 2019**.*

Resources

- Popel, Martin and Ondrej Bojar. “**Training Tips for the Transformer Model**.” The Prague Bulletin of Mathematical Linguistics 2018

- Step by step implementation of the paper: http://nlp.seas.harvard.edu/2018/04/03/attention.html
- https://github.com/tensorflow/tensor2tensor (official code)

- Several figures animations come from https://jalammar.github.io/. Contains a great visually illustrated explanation of the Transformer (link).

Thank you!