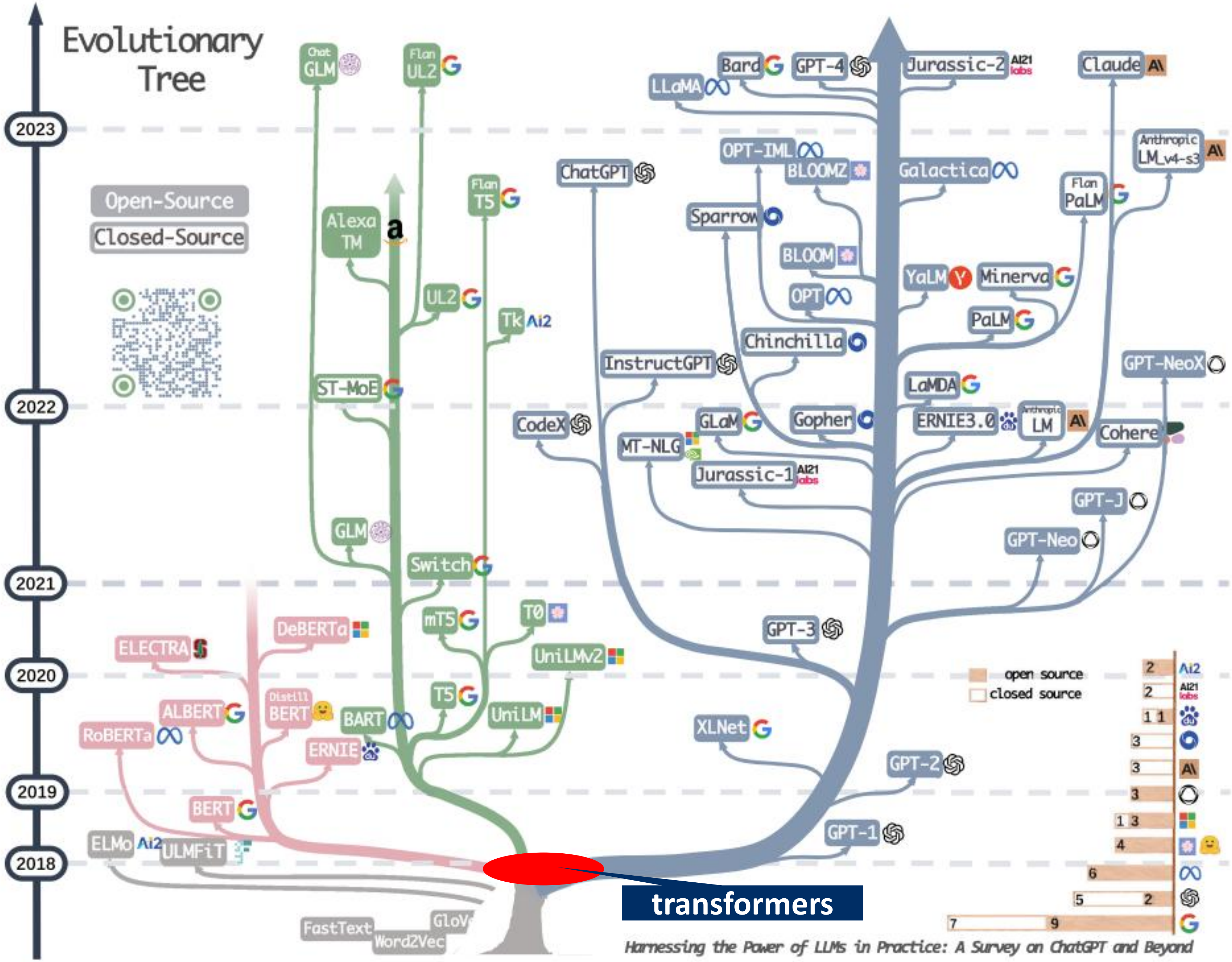
University of Antwerp
I Adrem I Adrem Data Lab

Attention Is All You Need

NeurIPS 2017

Authors: Ashish Vaswani (Google Brain) et al.

Presenter: Toon Calders



Evolutionary Tree
2023
2022
2021
2020
2019
2018
Open-Source
Closed-Source
Chat GLM
Flan UL2
Alexa TM
Flan T5
UL2
Tk
ST-MoE
GLM
Switch
mT5
T0
UniLMv2
T5
UniLM
BART
ERNIE
DeBERTa
ELECTRA
ALBERT
Distill BERT
RoBERTa
BERT
ELMo
ULMFiT
FastText
Word2Vec
GloVe
LLaMA
Bard
GPT-4
Jurassic-2
Claude
ChatGPT
OPT-IML
BLOOMZ
Galactica
Anthropic LM_v4-s3
Flan PaLM
Sparrow
BLOOM
YaLM
Minerva
OPT
PaLM
Chinchilla
InstructGPT
GPT-NeoX
LaMDA
CodeX
GLaM
Gopher
ERNIE3.0
Anthropic LM
Cohere
MT-NLG
Jurassic-1
GPT-J
GPT-Neo
GPT-3
XLNet
GPT-2
GPT-1
open source
closed source
transformers
Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond

Introduction

Problem domain:

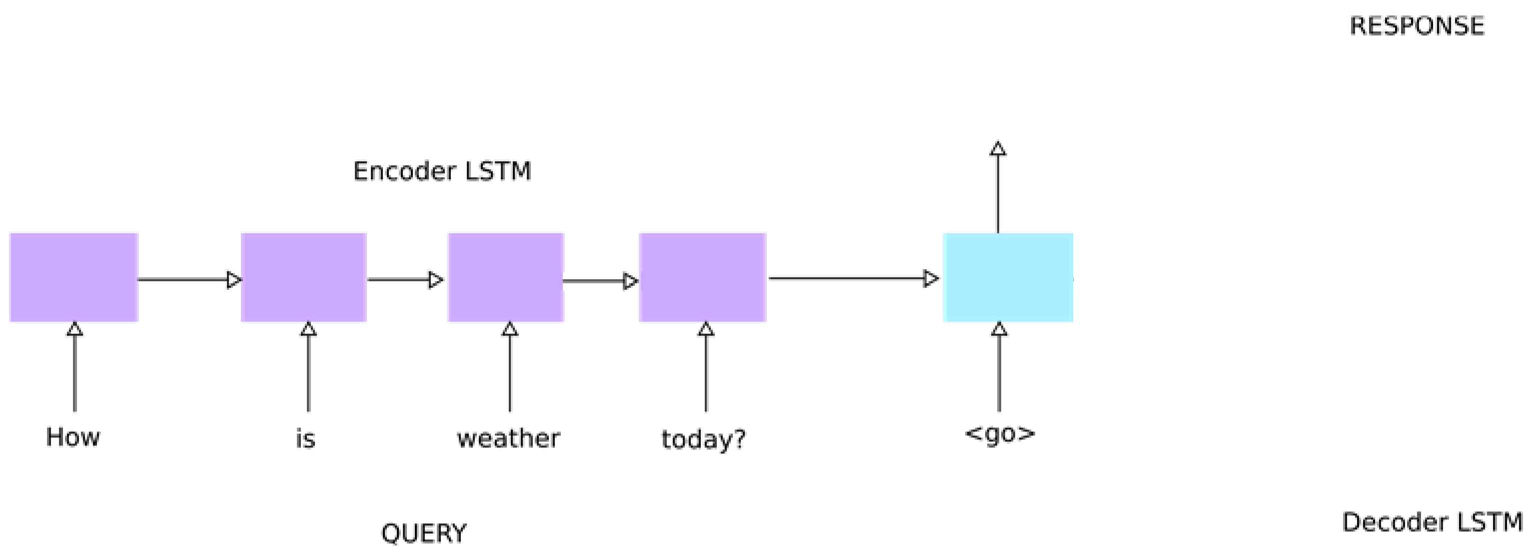
- **Learning *Sequence Transduction* Models**
  - e.g. translation, Q&A

- **New model architecture: the *transformer***
  - *Encoder-decoder* architecture
  - Systematic use of *attention mechanism*
  - No convolutions nor recurrence

- **Good performance on *Machine Translation* tasks**

“Old” Sequence-to-Sequence Models

- **Common approach before transformers: Recurrent Neural Network architectures based on LSTMs***



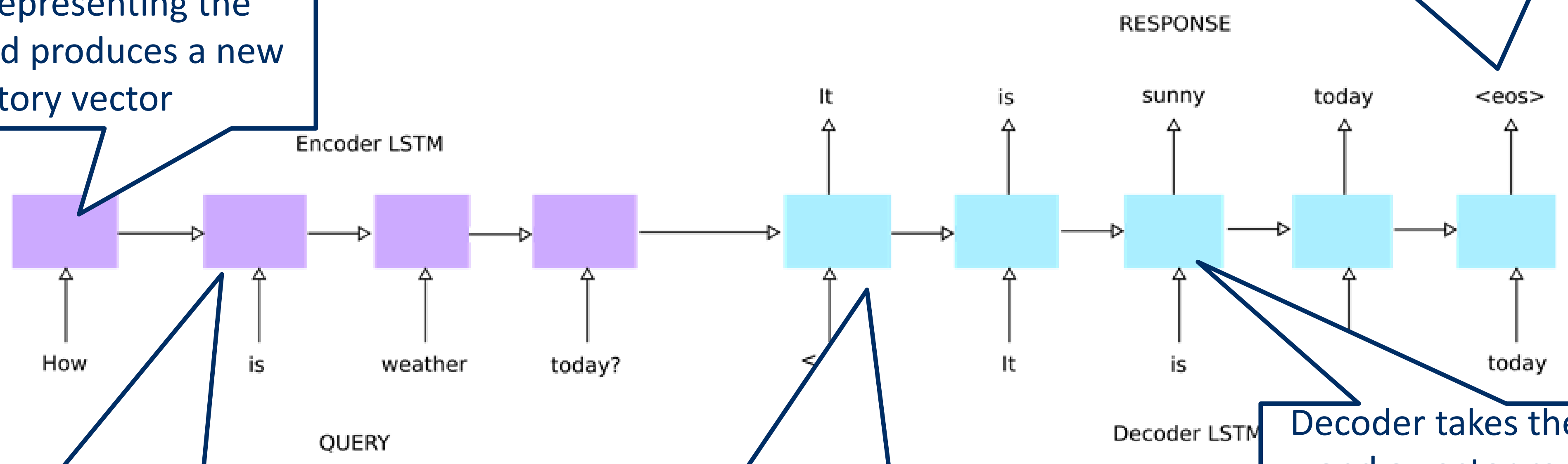
* LSTM: Long short term memory

"Old" Sequence-to-Sequence Models

- **Common approach before transformers: Recurrent Neural Network architectures based on LSTMs***

Encoder takes a token and a vector representing the history and produces a new history vector

Special token denoting the end of the sentence



All encoders are the same function (share weights)

All decoders are the same function (share weights)

Decoder takes the previous word and a vector representing the history and produces a vector and an output token

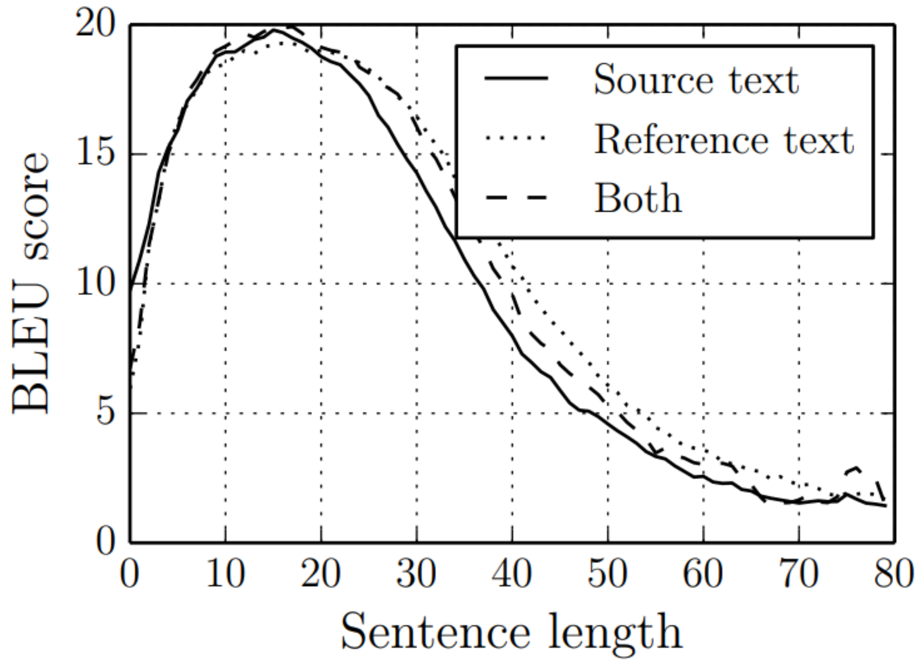
* LSTM: Long short term memory

“Old” Sequence – To – Sequence Models

- **Common techniques:**
  - Recurrent Neural Networks based on LSTMs
  - Convolutional techniques

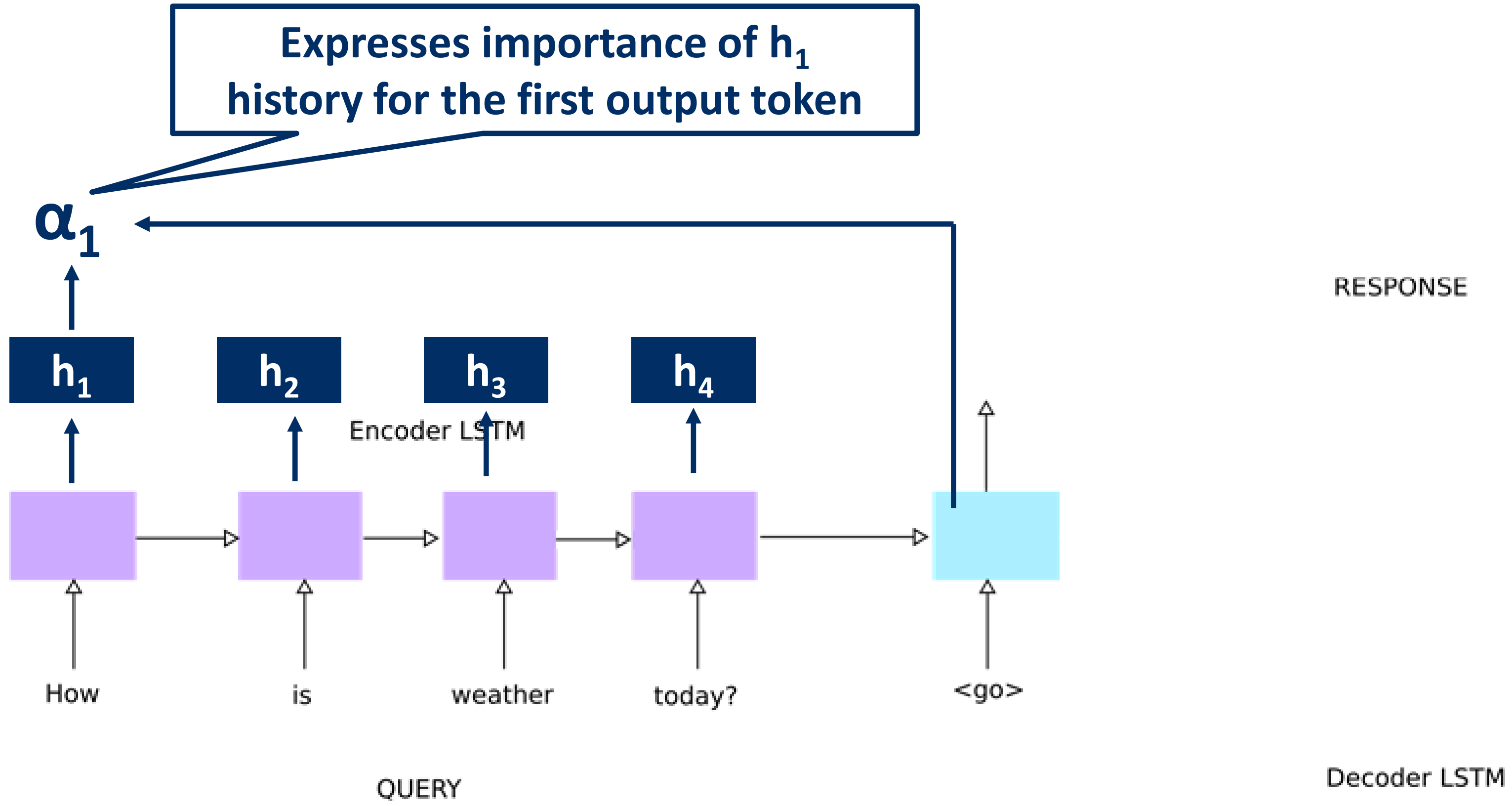
- **Disadvantages:**
  - Long dependencies hard to capture
  - Limited possibility for parallellism

- **Solution for long dependencies:**
  - Attention mechanisms



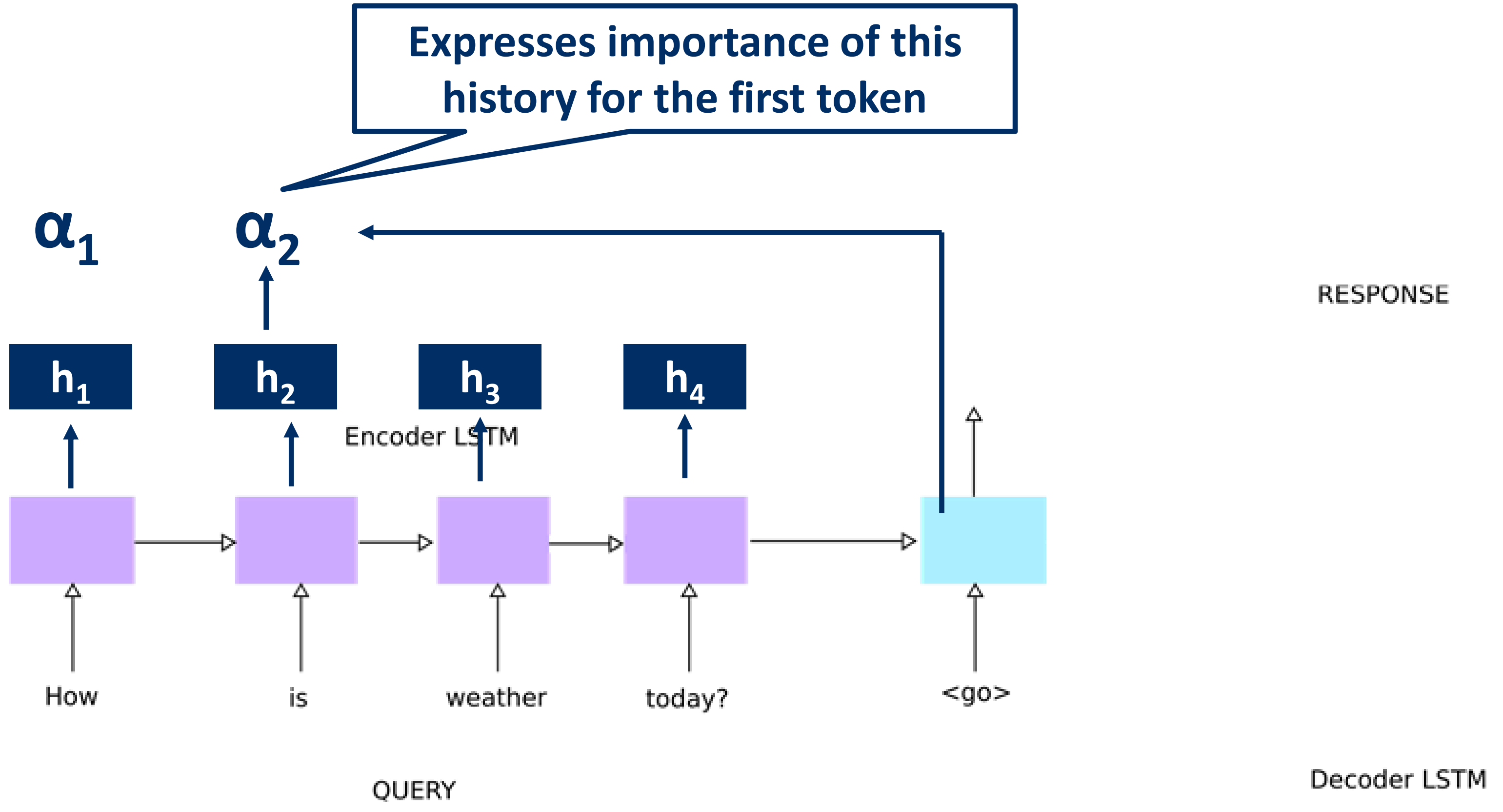
Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014b). On the properties of neural ¨ machine translation: Encoder-Decoder approaches. In Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation.

Attention Mechanism



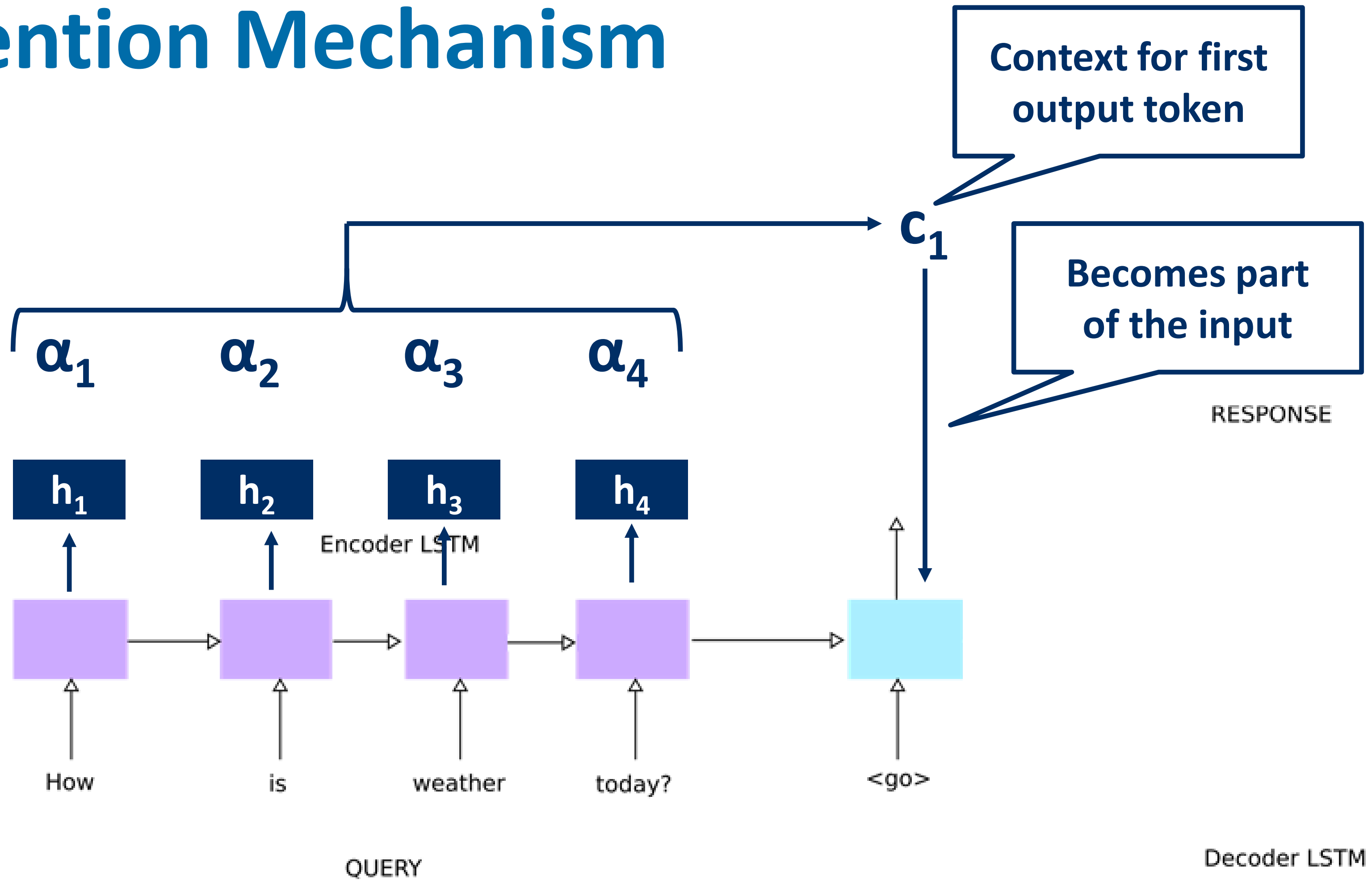
Bahdanau, Dzmitry, Kyung Hyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

Attention Mechanism



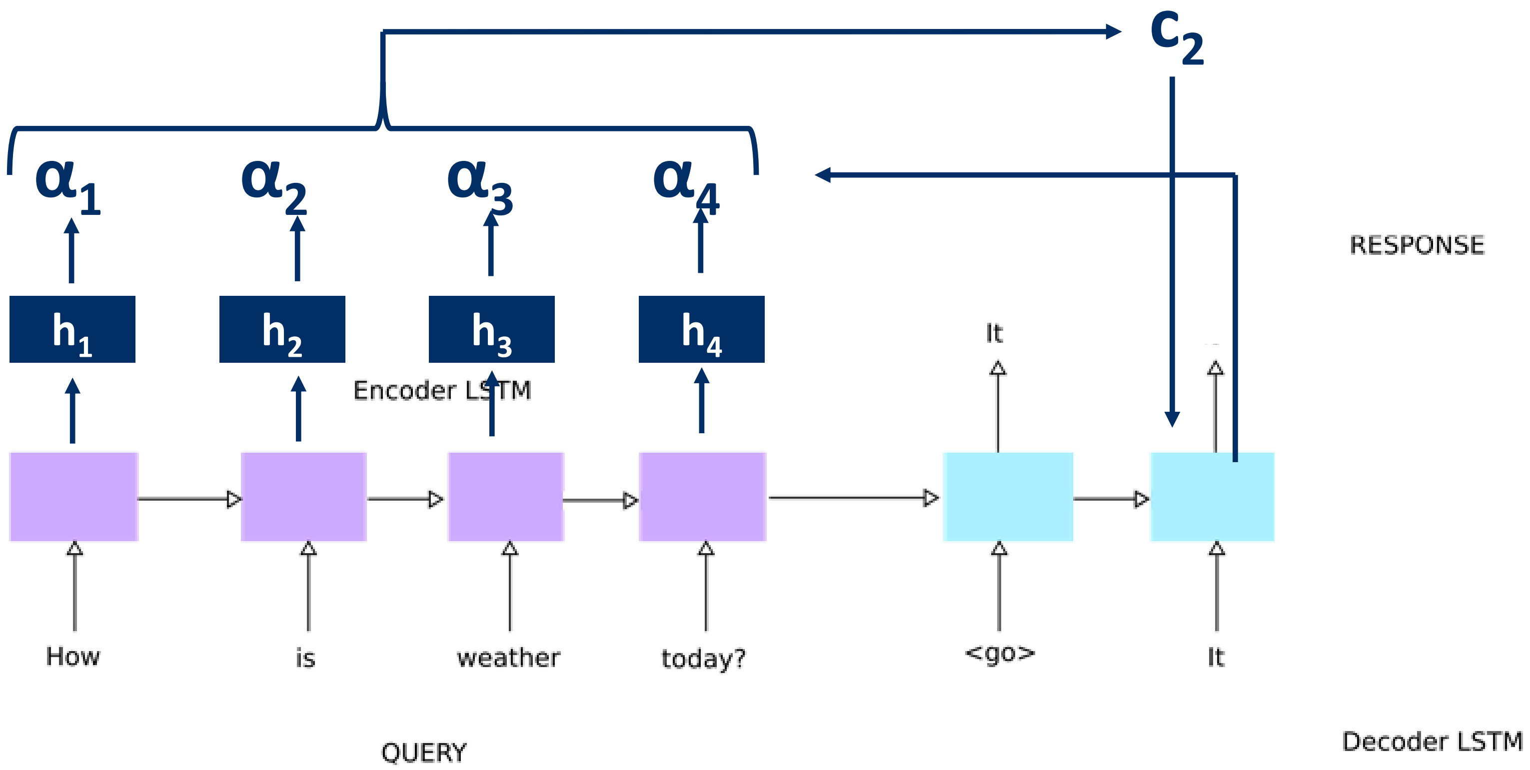
Bahdanau, Dzmitry, Kyung Hyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

Attention Mechanism



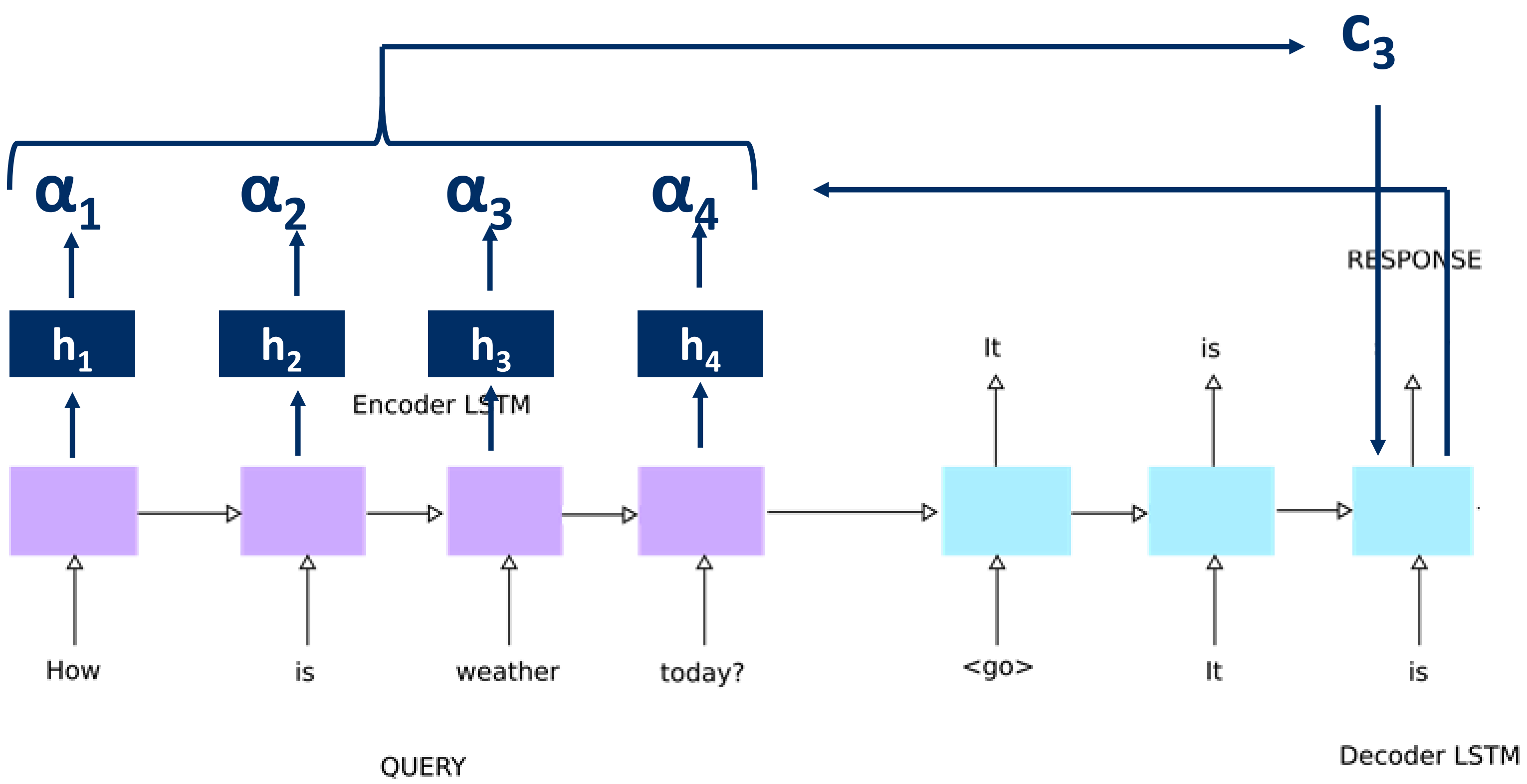
Bahdanau, Dzmitry, Kyung Hyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

Attention Mechanism



Bahdanau, Dzmitry, Kyung Hyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

Attention Mechanism



University of Antwerp
I Adrem I Adrem Data Lab

Bahdanau, Dzmitry, Kyung Hyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

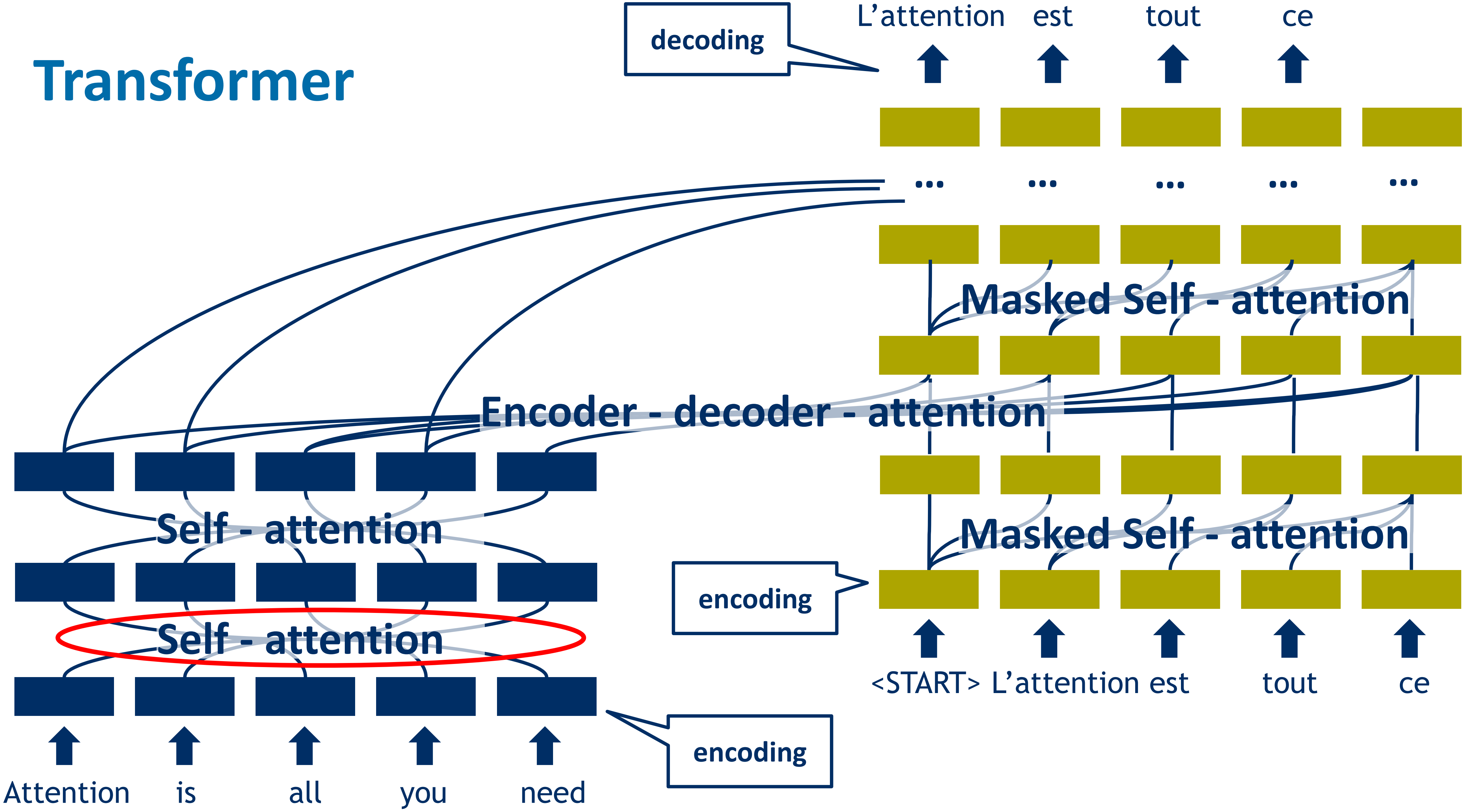
New Model Proposed in the Paper

- Encoder – Decoder structure

- Each layer has an encoding for each token
  - "Enriched" representation in next layer computed on previous layer

- Layers are connected using attention mechanism
  - No convolutional layers, recurrent neural nets, LSTM
  - Hence: Attention is all you need!

Transformer

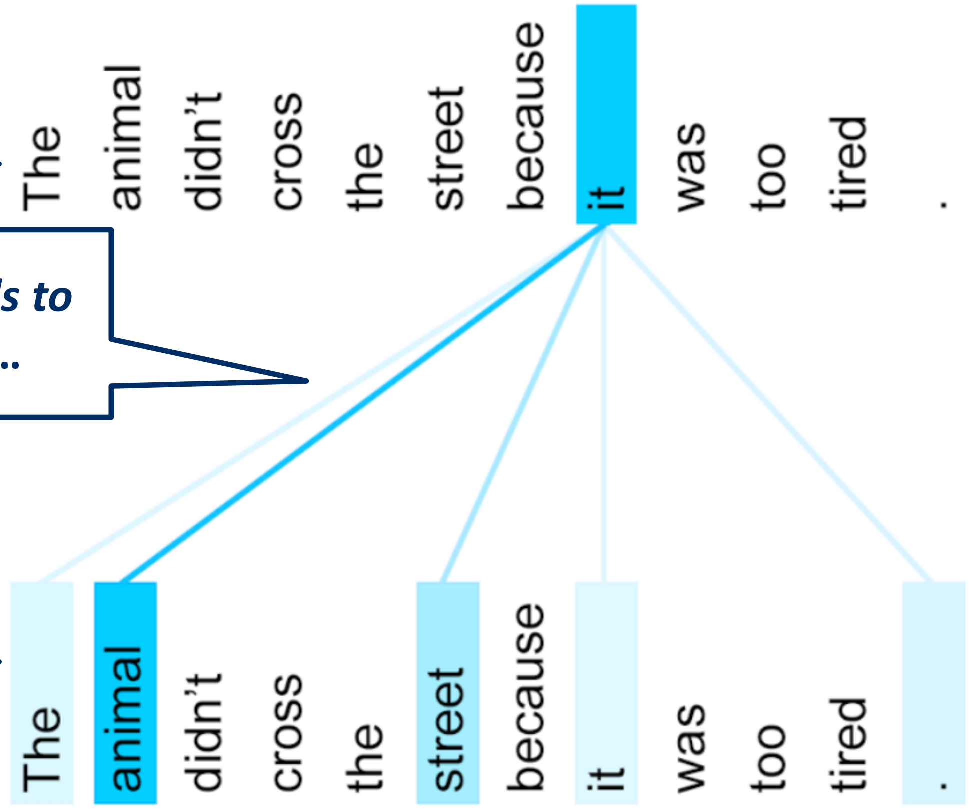


Attention Mechanism: Intuition

Representation in layer n+1

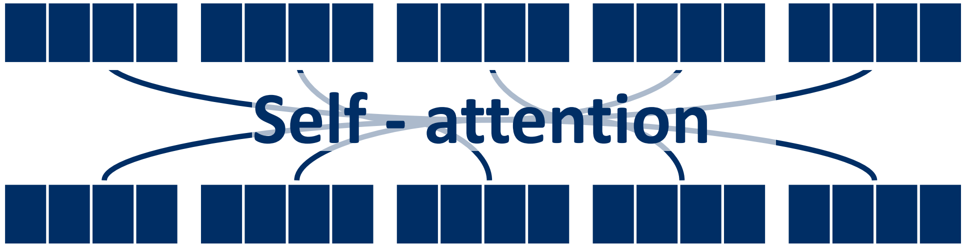
Representation of “it” *attends to* that of “animal”, “street”, …

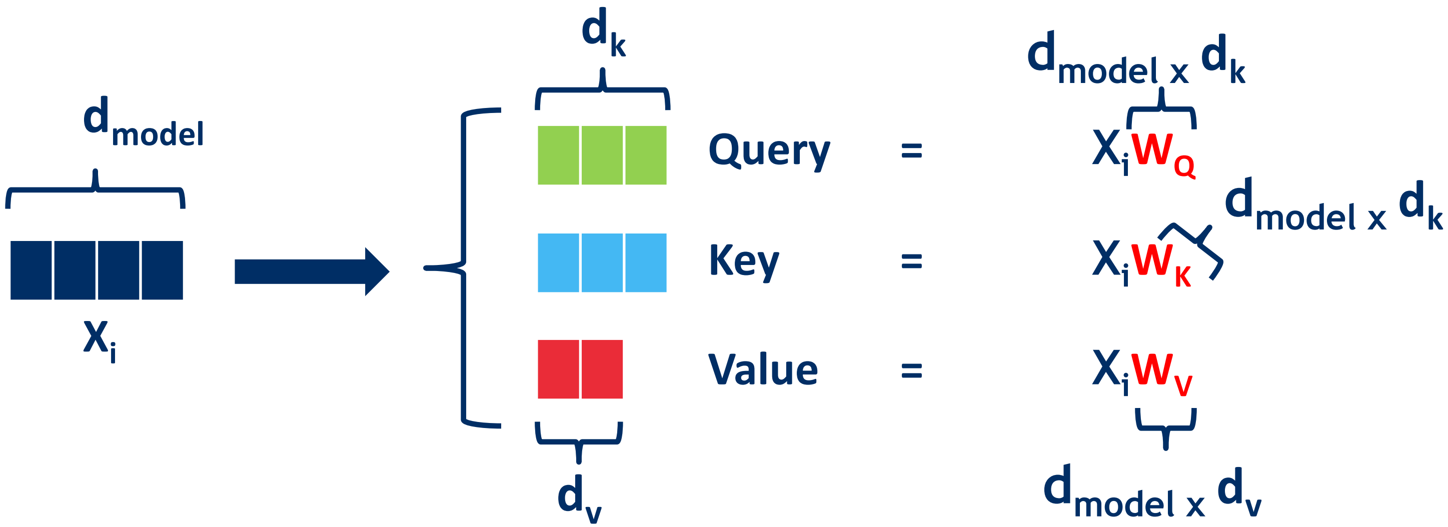
Representation in layer n



“Scaled Dot-Product” Attention

- : d_{model} – dimensional vector
  - Transformed into query, key, value
  - Using learned matrices W_Q, W_K, and W_V

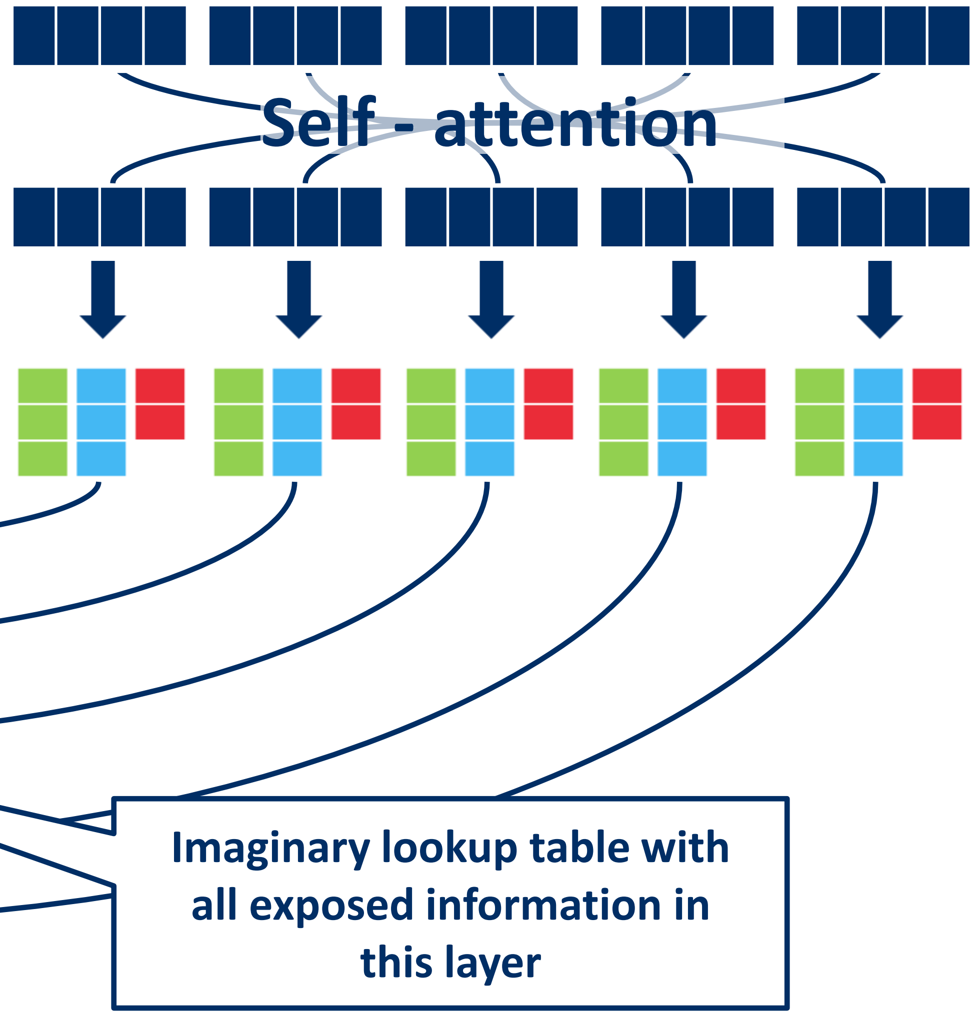




“Scaled Dot-Product” Attention

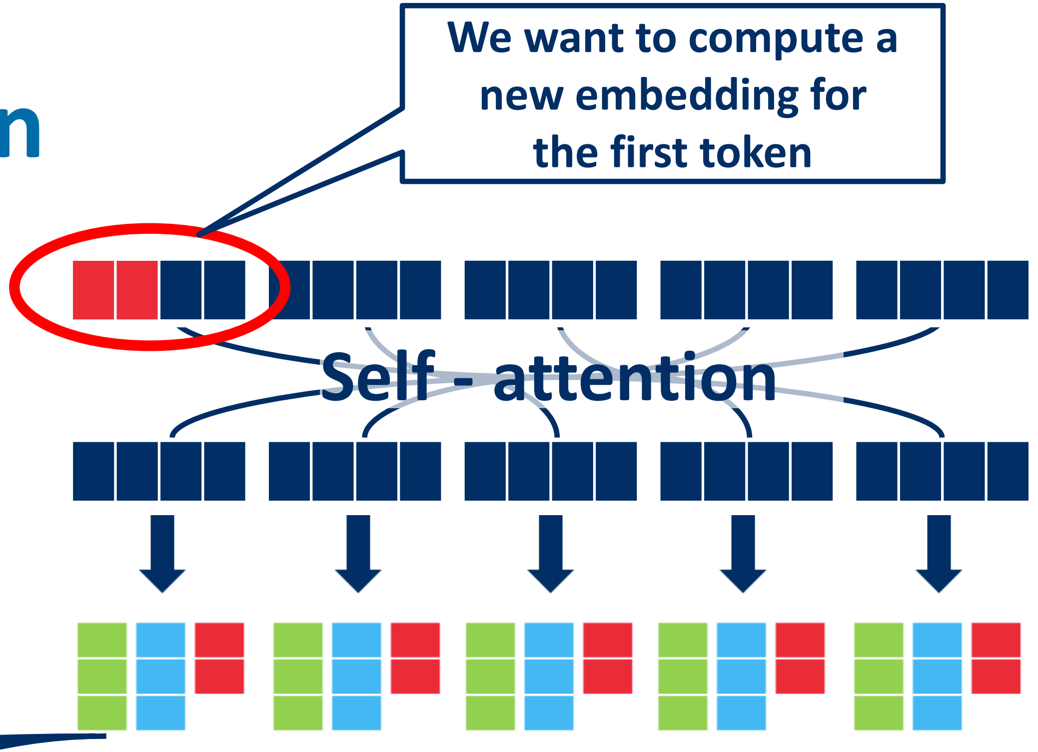
- : d_{model} – dimensional vector
  - Transformed into query, key, value
  - Using learned matrices W_Q, W_K, and W_V

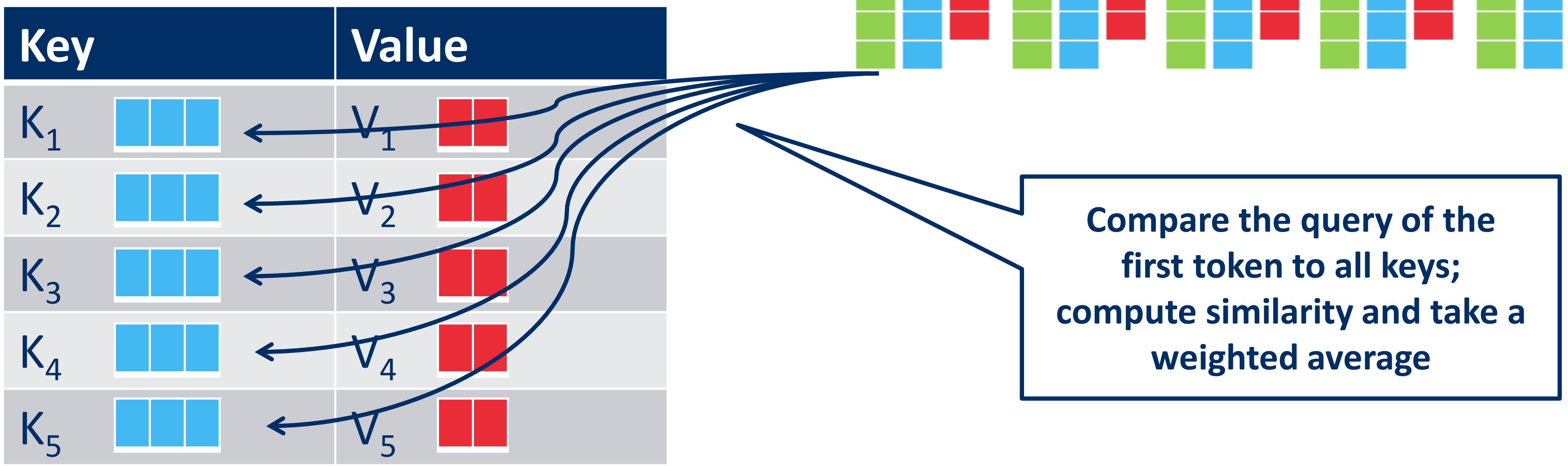
Key	Value
K_1	V_1
K_2	V_2
K_3	V_3
K_4	V_4
K_5	V_5



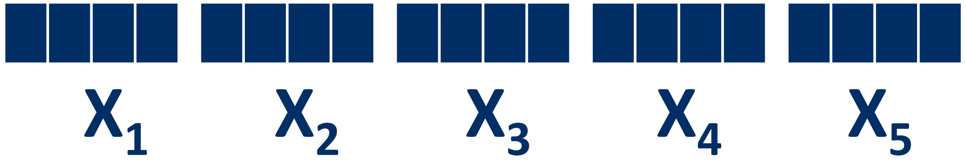
"Scaled Dot-Product" Attention

- : d_{model} – dimensional vector
  - Transformed into query, key, value
  - Using learned matrices W_Q, W_K, and W_V

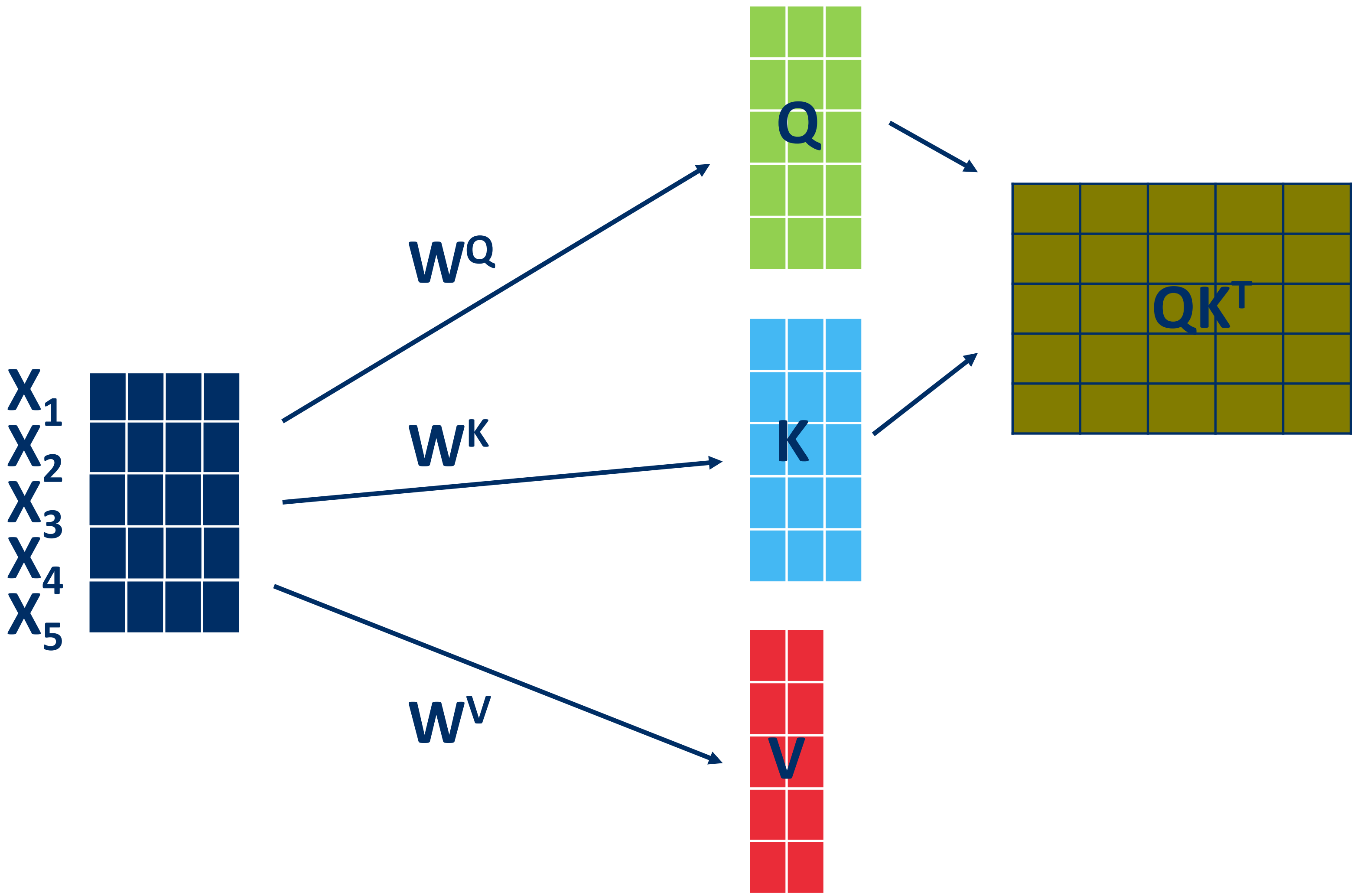




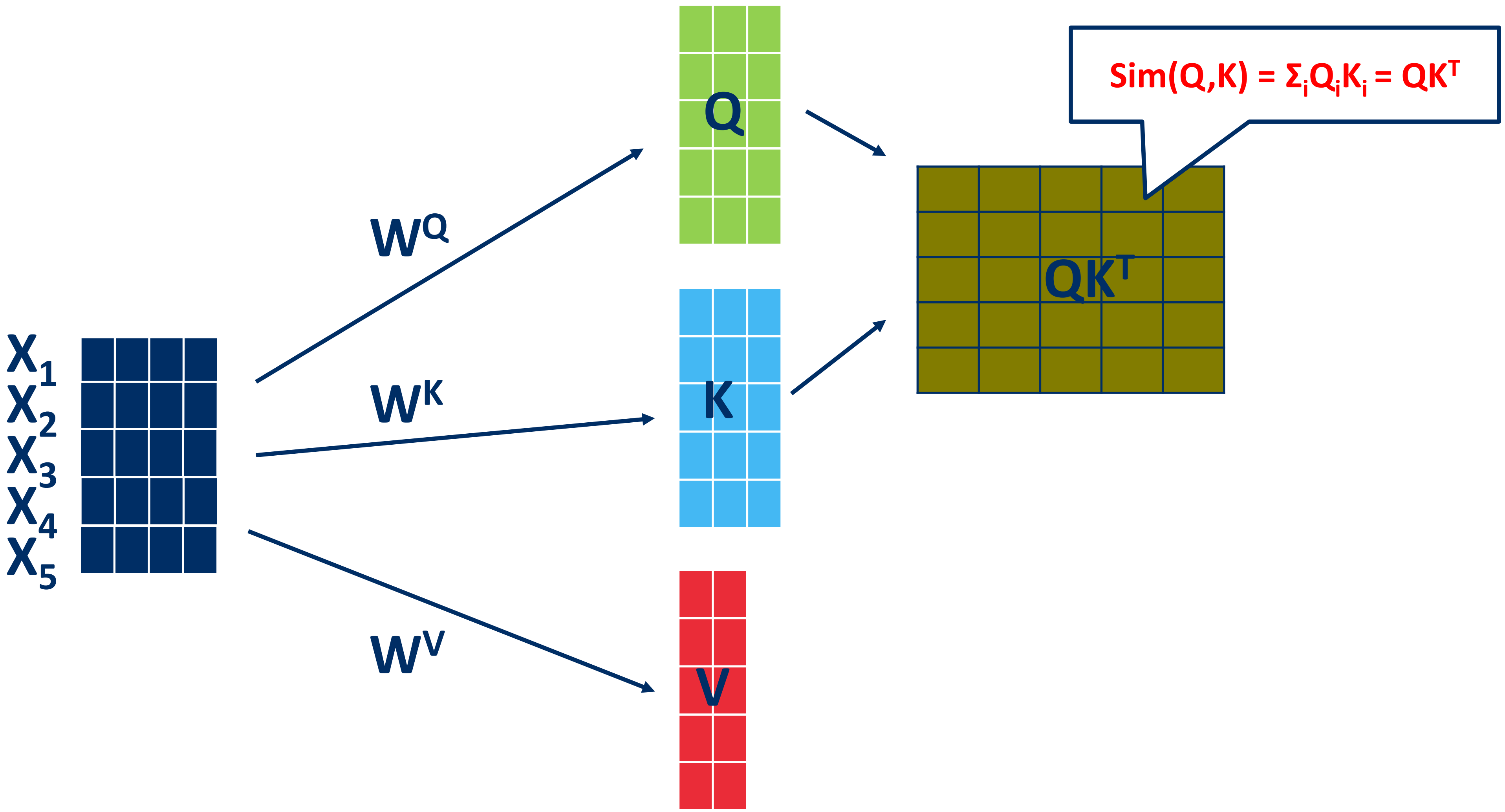
“Scaled Dot-Product” : Implementation



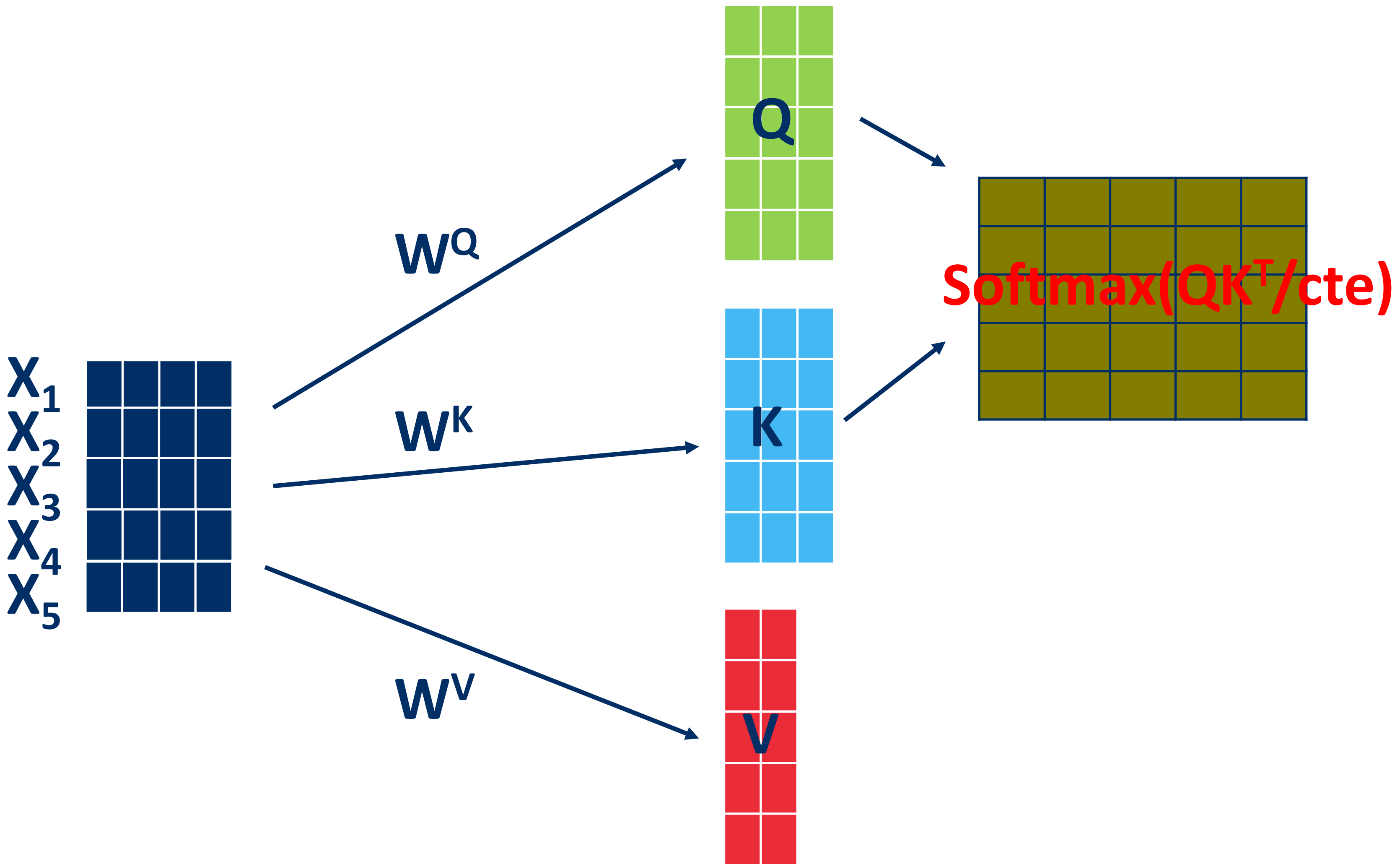
“Scaled Dot-Product” : Implementation



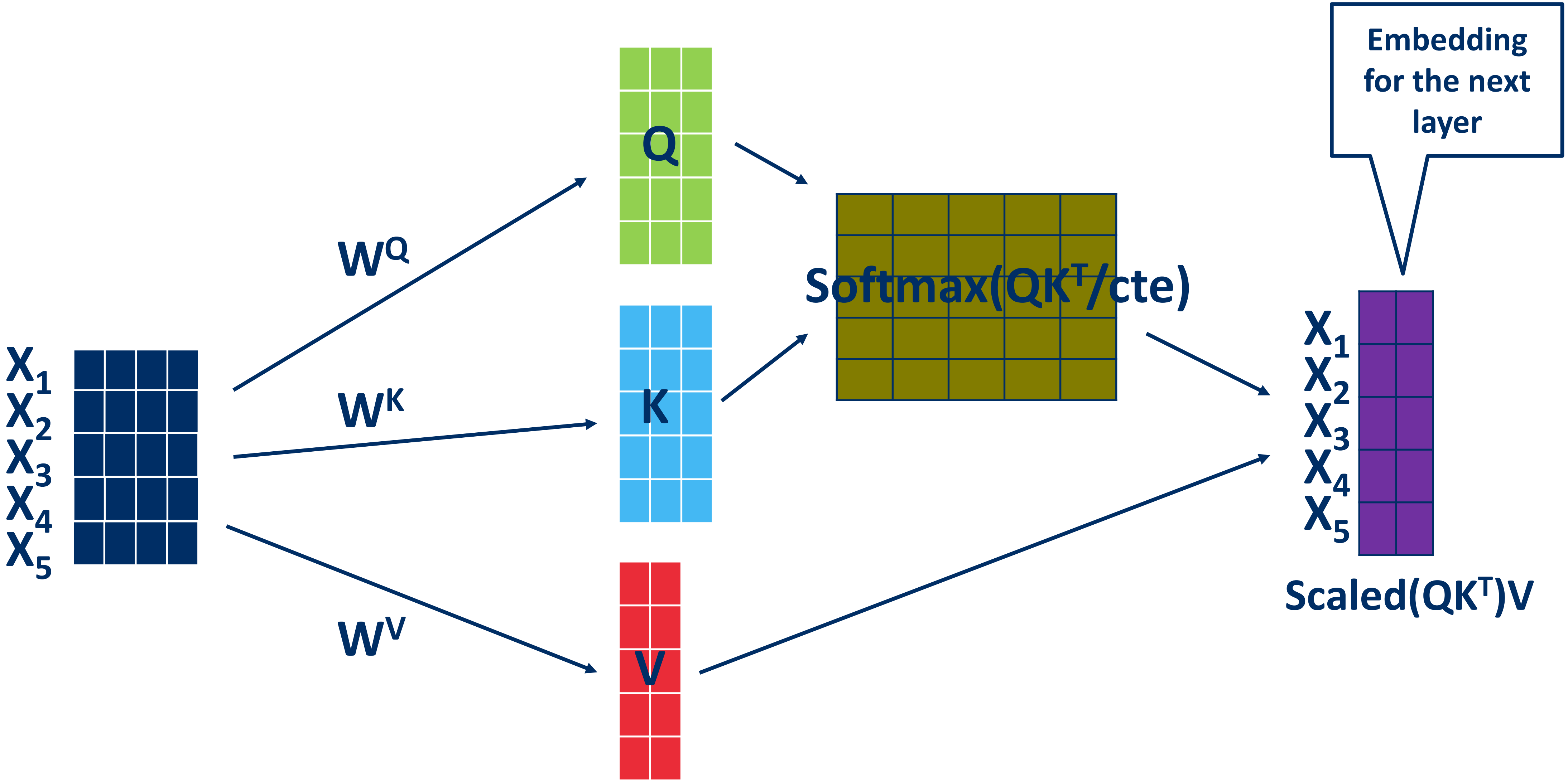
"Scaled Dot-Product" : Implementation



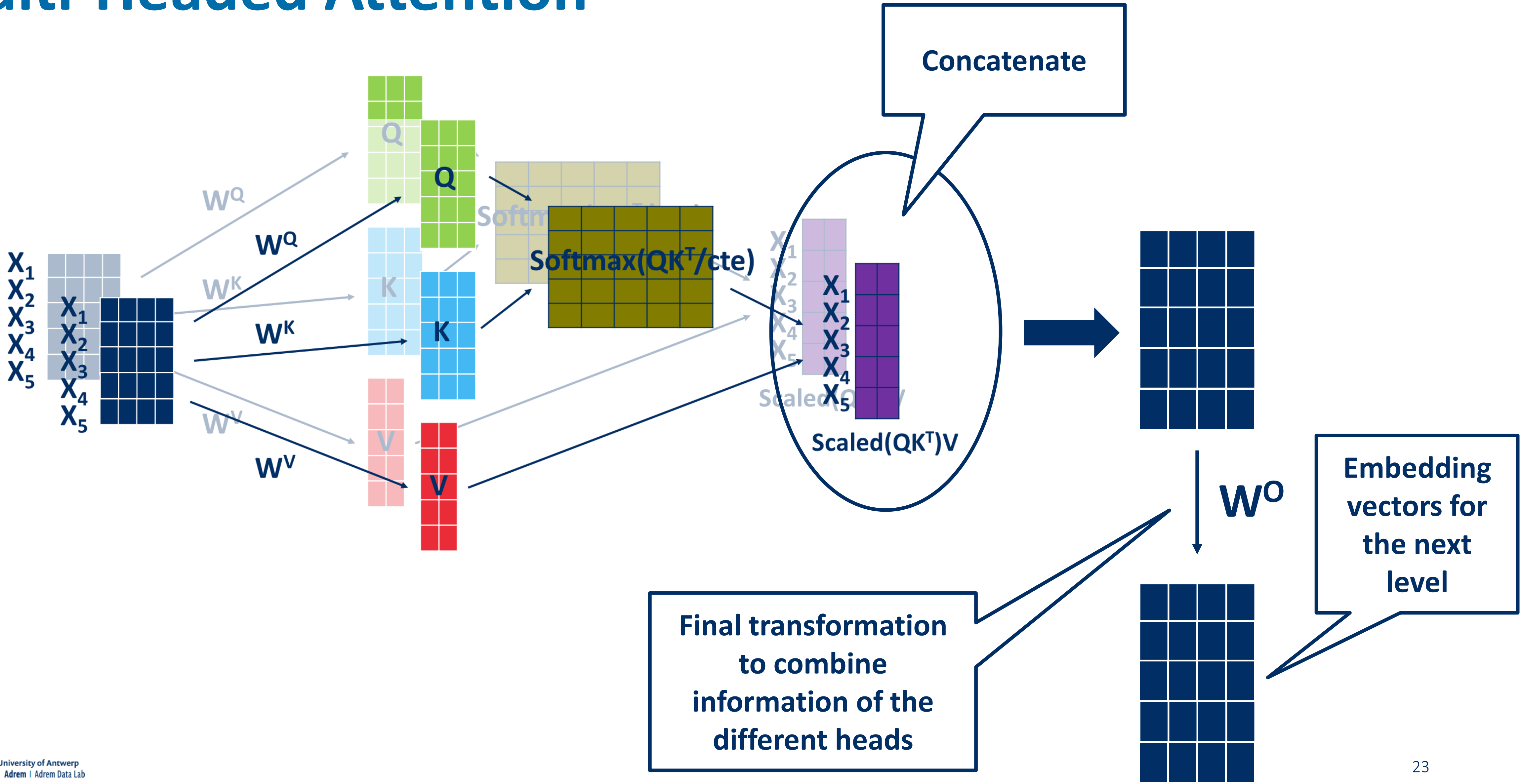
“Scaled Dot-Product” : Implementation



"Scaled Dot-Product" : Implementation

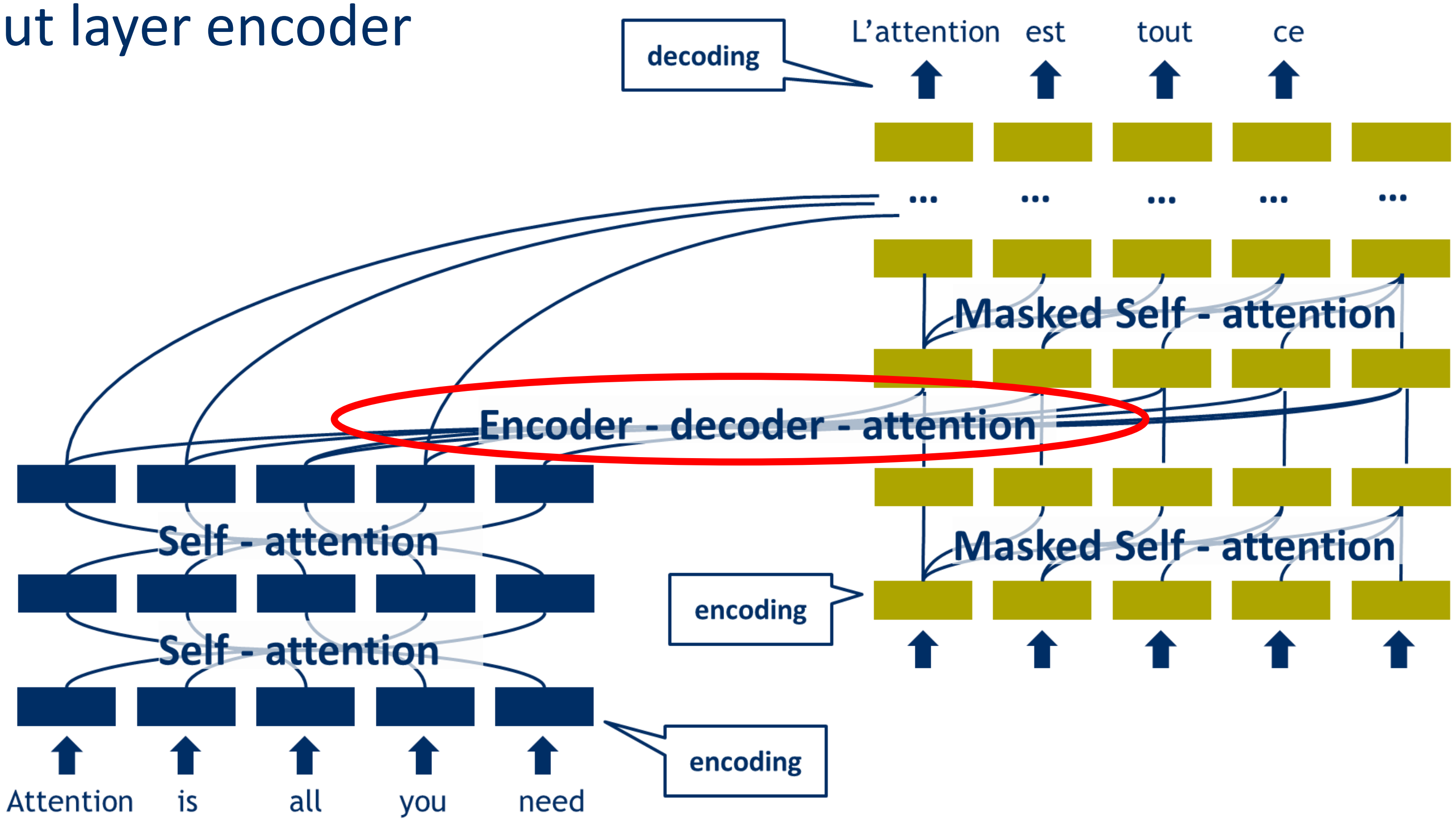


Multi-Headed Attention



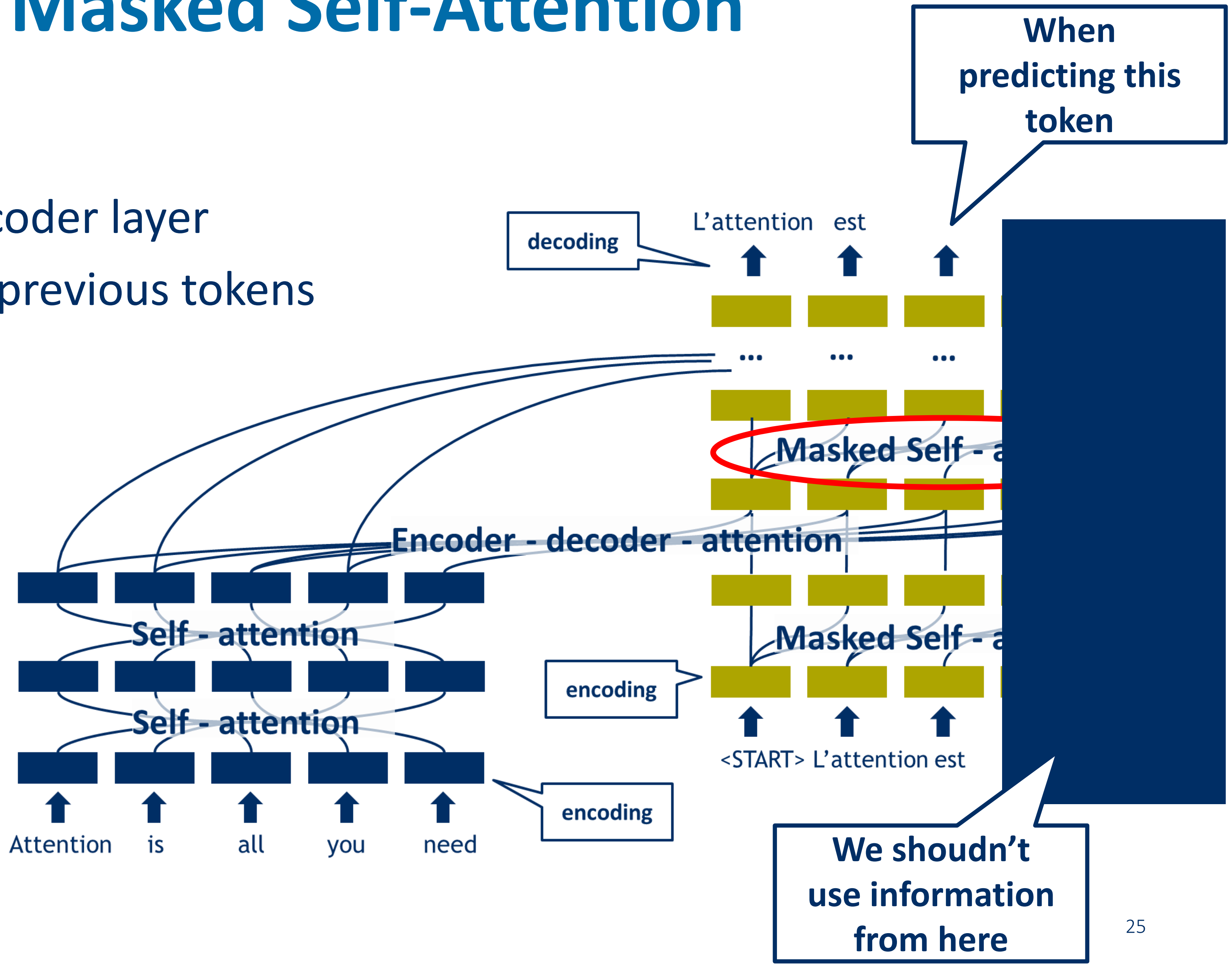
Encoder – Decoder attention

- Very similar construction
  - key, value come from output layer encoder
  - query from decoder layer

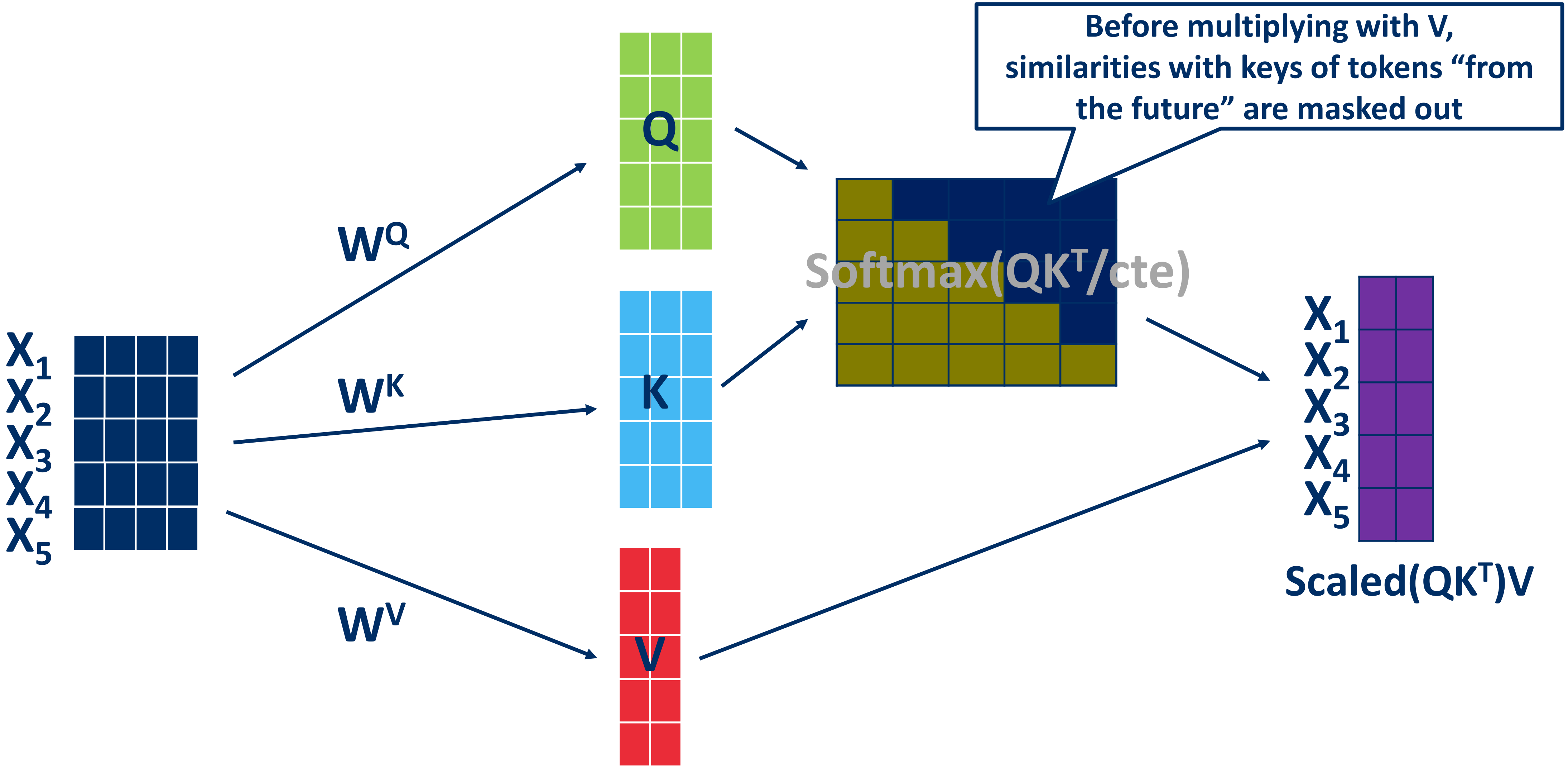


Decoder Attention: Masked Self-Attention

- Again, very similar
  - Key, value, query from decoder layer
  - Tokens can only attend to previous tokens



Masked Self-Attention



Transformer - Complete Architecture

Decoding tokens to text

Linear transformation & softmax turning into token probabilities

idem

Fully-connected network (one hidden layer)

Again add instead of replace

Add attention to previous embedding; normalize weights in layer

***Encoder – Decoder* attention**

Add attention to previous embedding; normalize weights in layer

Self-attention

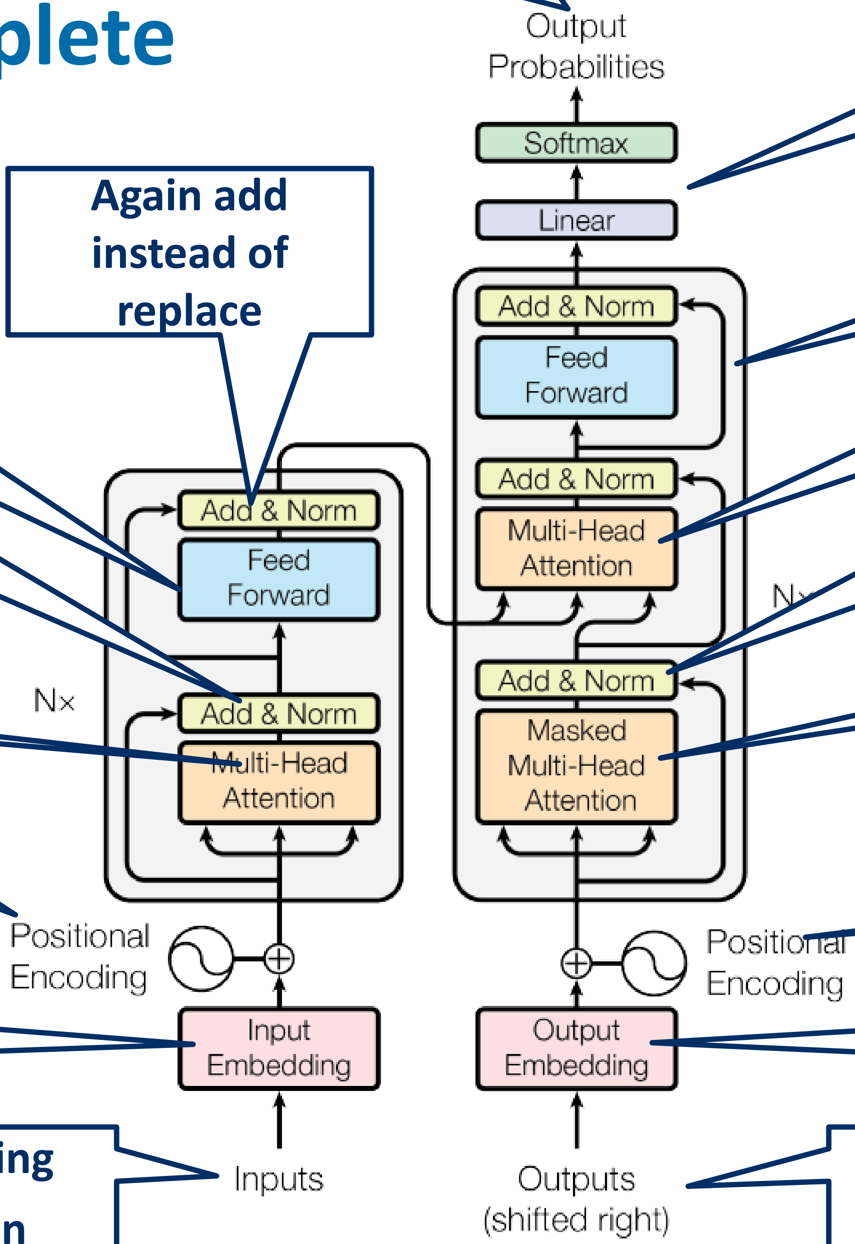
***Masked* Self-attention**

Encode position in embedding (see later)

Encode position in embedding (see later)

Linear transformation

Linear transformation



Byte Pair Encoding for tokenization

Byte Pair Encoding for tokenization. *Target language*

Byte Pair Encoding

- Turn text into numbers
  - fixed (subword-)tokenization scheme
  - Every sequence of characters can be tokenized

Example:

My name is Toon. → ['My', ' name', ' is', ' To', 'on', '.']

[3666, 1438, 318, 1675, 261, 13]

My name is Toon! → ['My', ' name', ' ', ' is', ' ', ' To', 'on', '!’]

[3666, 1438, 220, 318, 220, 1675, 261, 0]

Input Embedding

- **BPE has 37K tokens**
- **37K dim. 1-hot encoding → lower-dim. representation (trained)**

1	-1.5	3.3	-2.1	8.5	2.1	-2.6	8.1
2	0.3	-0.2	2.4	-2.1	6.6	2.1	7.2
…				…			
3,665	2.1	-2.6	8.1	3.3	-2.1	8.6	2.1
3,666	2.4	-2.4	2.1	-2.6	8.1	-9.3	7.9
3,667	3.4	-2.1	6.6	2.1	-4.2	1.3	9.2
…				…			
36,999	-3.4	2.1	-2.6	8.1	-2.3	-2.2	3.2
37,000	3.4	-2.1	8.6	2.1	2.1	-2.6	8.1

Token embedding

2.4	-2.4	2.1	-2.6	8.1	-9.3	7.9

'My' → 3666

Input token

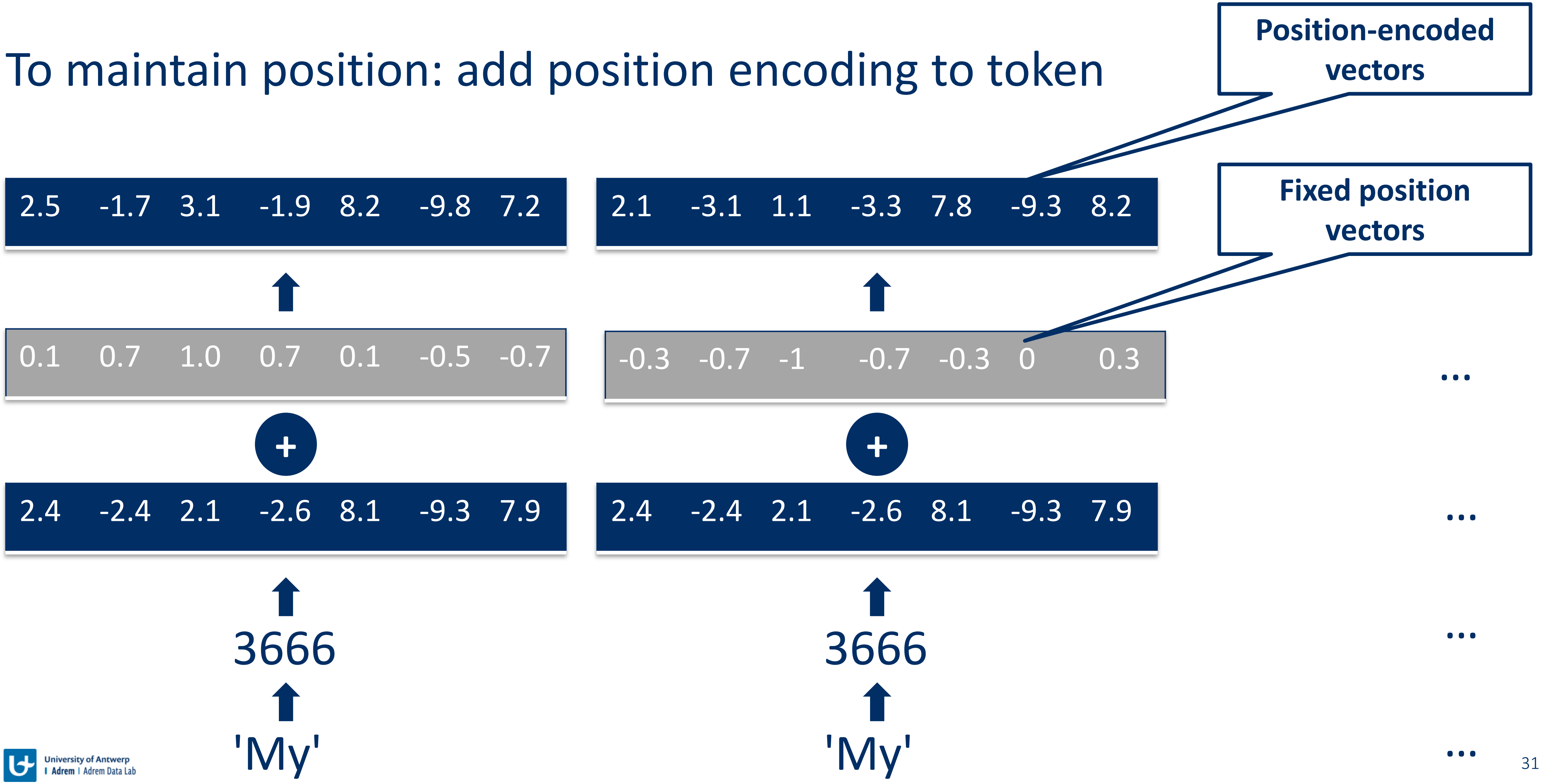
BPE

Positional Encoding

- Order of tokens is important:
  - Woman, without her **man,** is nothing.
  - Woman, without her**, man** is nothing.
- BPE:
  - [48081, 11, 1231, 607, **582**, **11**, 318, 2147, 13]
  - [48081, 11, 1231, 607, **11**, **582**, 318, 2147, 13]
- Gives *the same* key-value pairs
  - Is token “Woman” subject of the sentence or not?
  - Position of comma is essential!

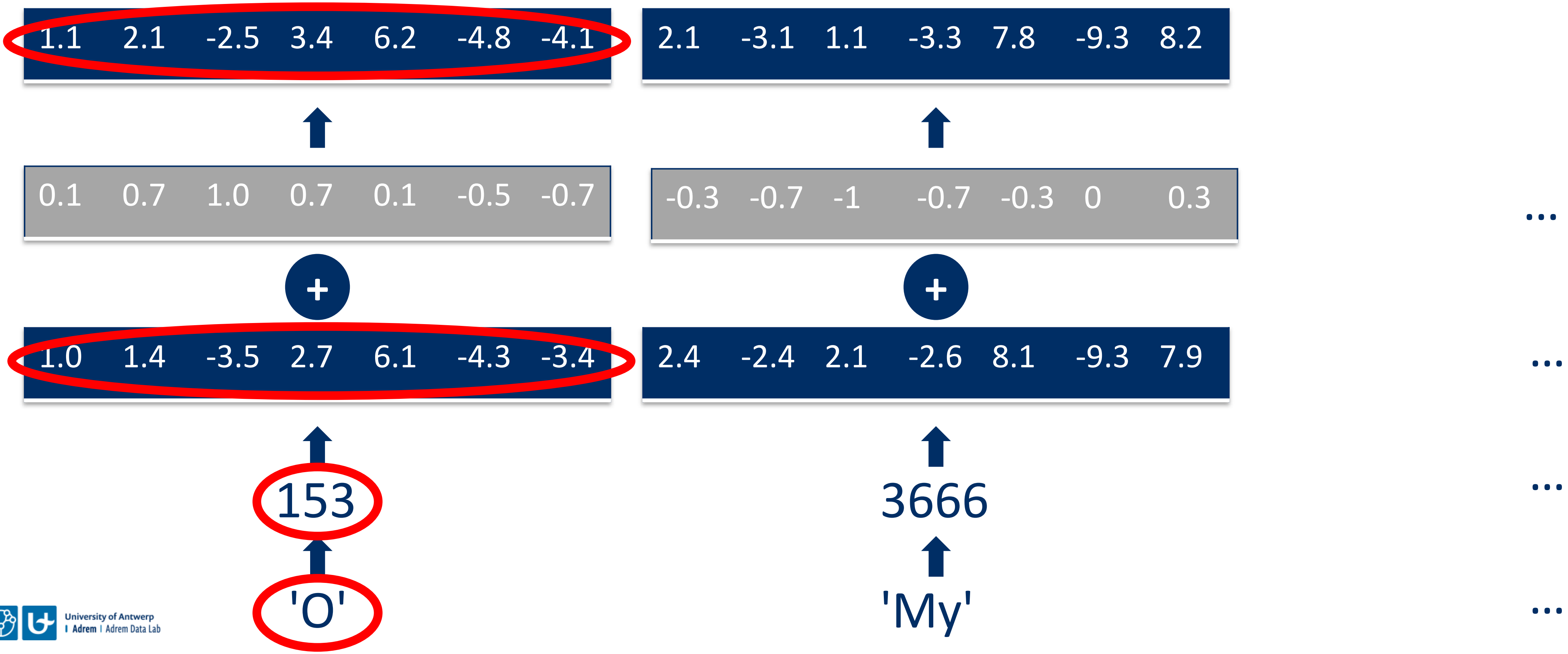
Positional Encoding

- To maintain position: add position encoding to token



Positional Encoding

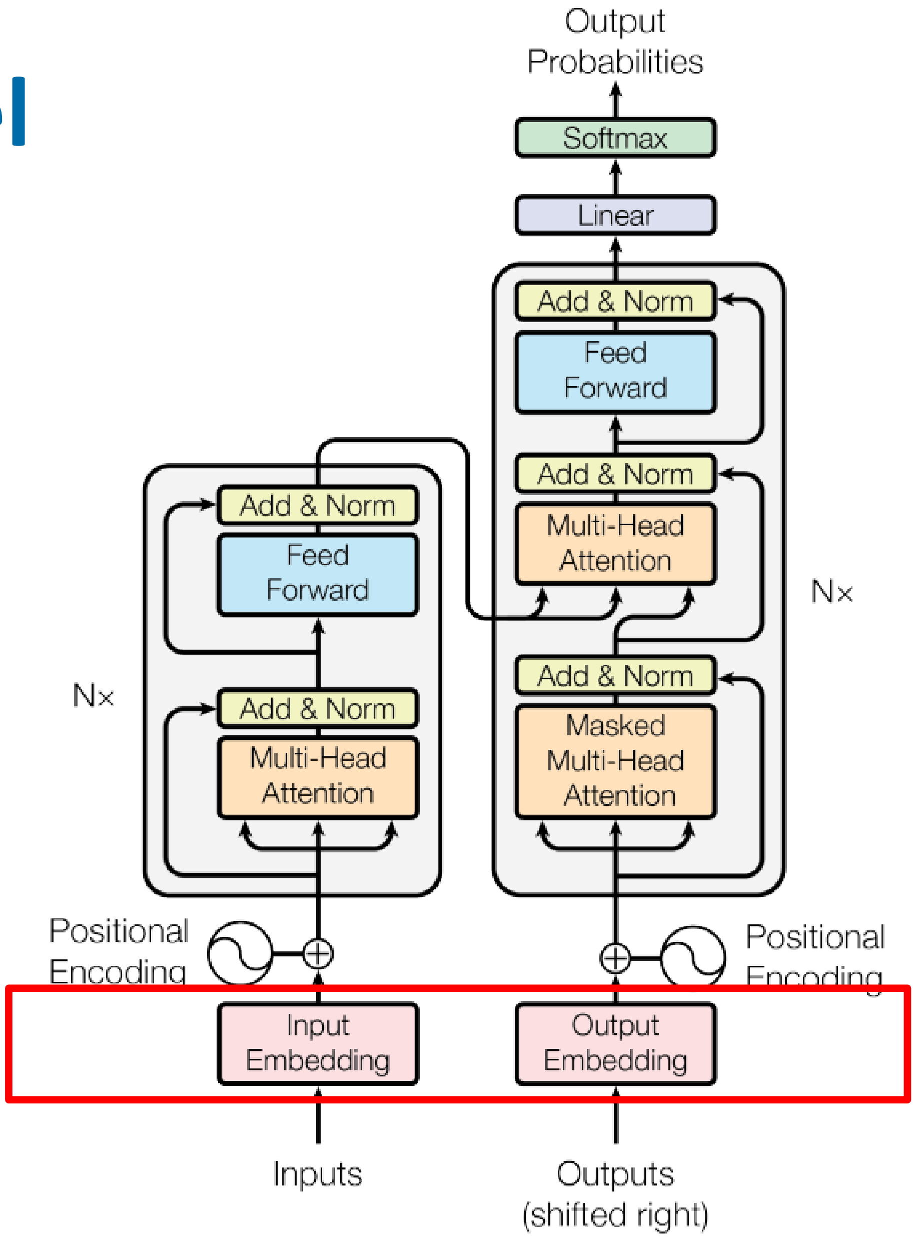
- To maintain position: add position encoding to token



Number of Parameters? Base Model

- $N=6$, $d_{model}=512$, $d_{ff}=2048$, $h=8$, $d_k=64$, $d_v=64$

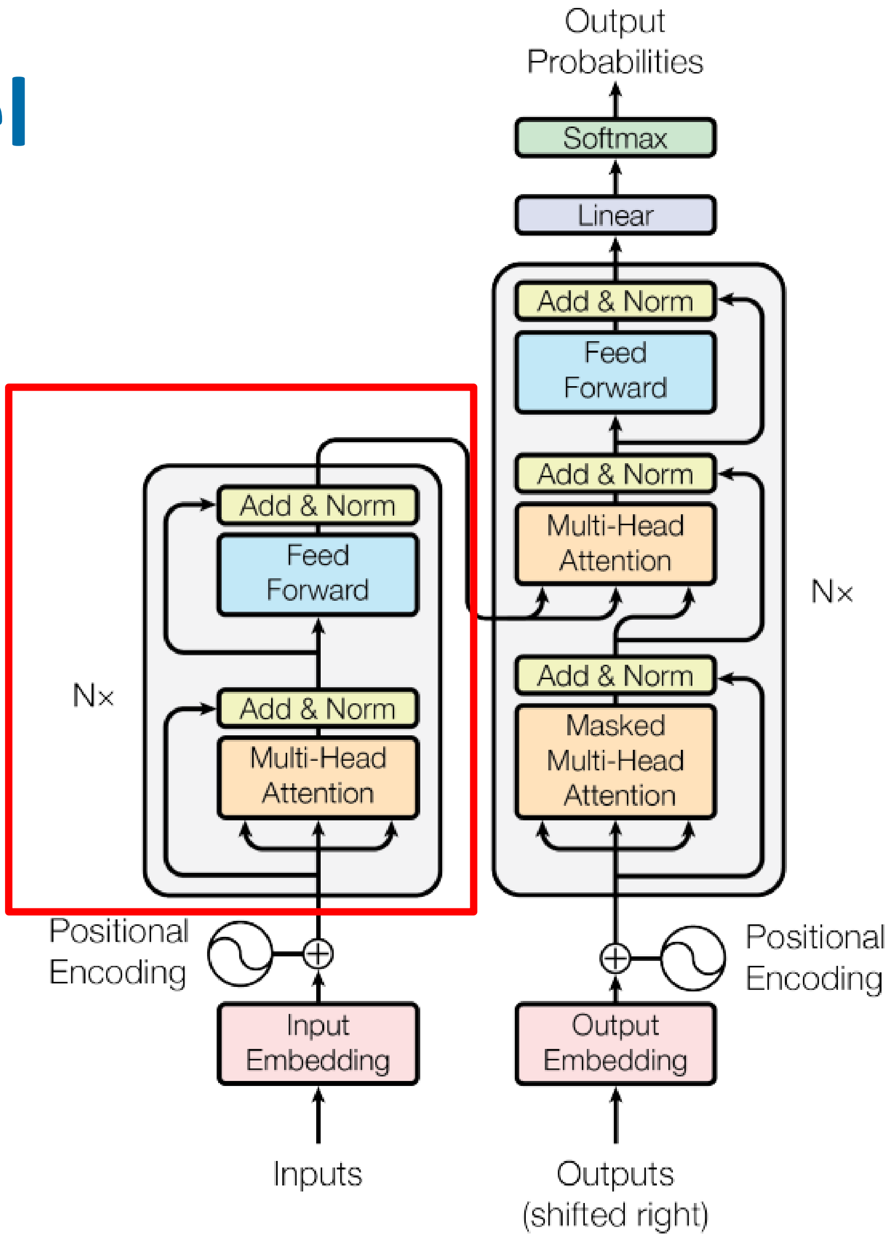
- **Input & output embedding: ≈ 19M**
  - They are shared
  - Shared vocabulary of 37,000 tokens
  - 512-dimensional representation



Number of Parameters? Base Model

- **$N=6$, $d_{model}=512$, $d_{ff}=2048$, $h=8$, $d_k=64$, $d_v=64$**

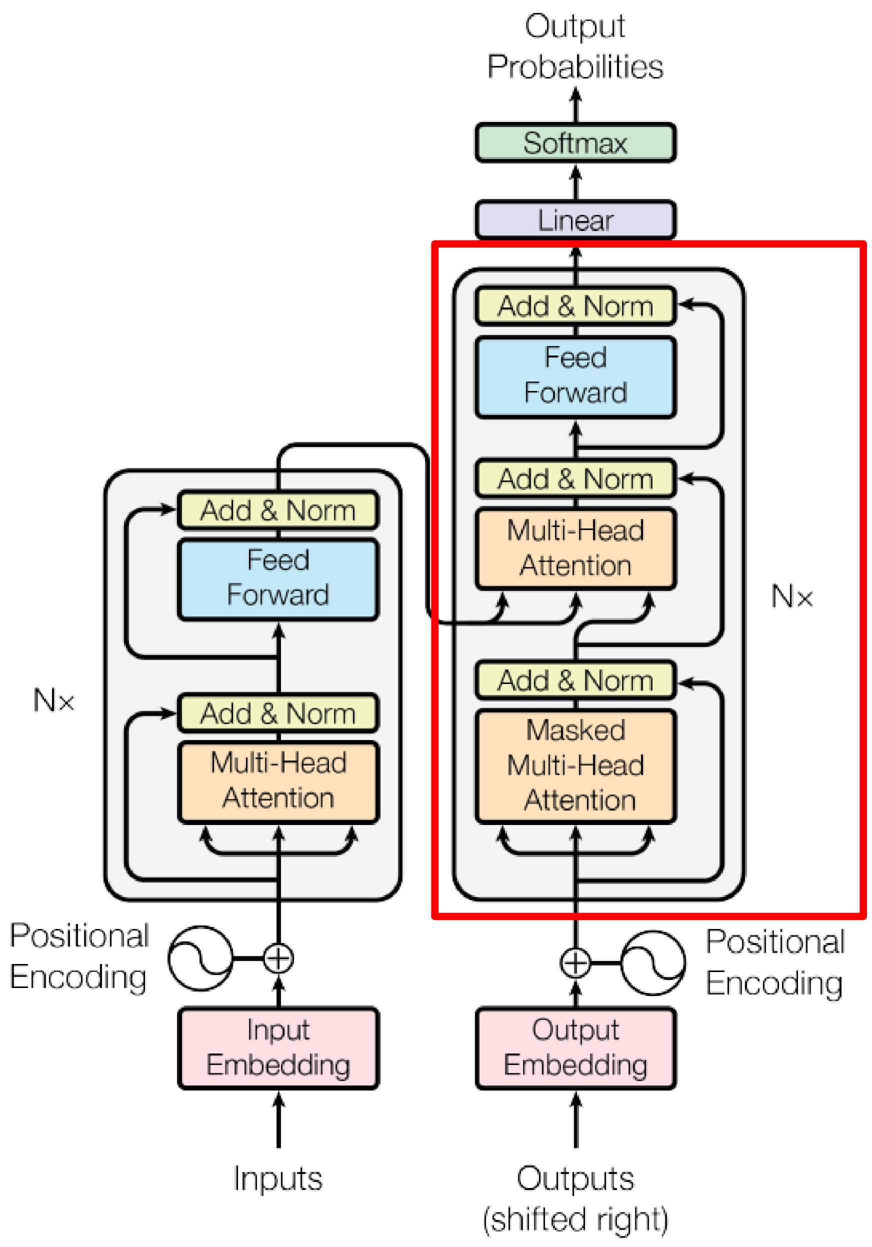
- **Attention mechanism: ≈ 6 x 1M**
  - 8 heads
    - Each matrix W^Q, W^K, W^V : 512 x 64
  - Matrix W^O : 512 x 512 = 262,144
- **Feed Forward: ≈ 6 x 2.1M**
  - (512+1) x 2048 + (2048+1) x 512
- **Total: ≈ 18.3M**



Number of Parameters? Base Model

- **$N=6$, $d_{model}=512$, $d_{ff}=2048$, $h=8$, $d_k=64$, $d_v=64$**

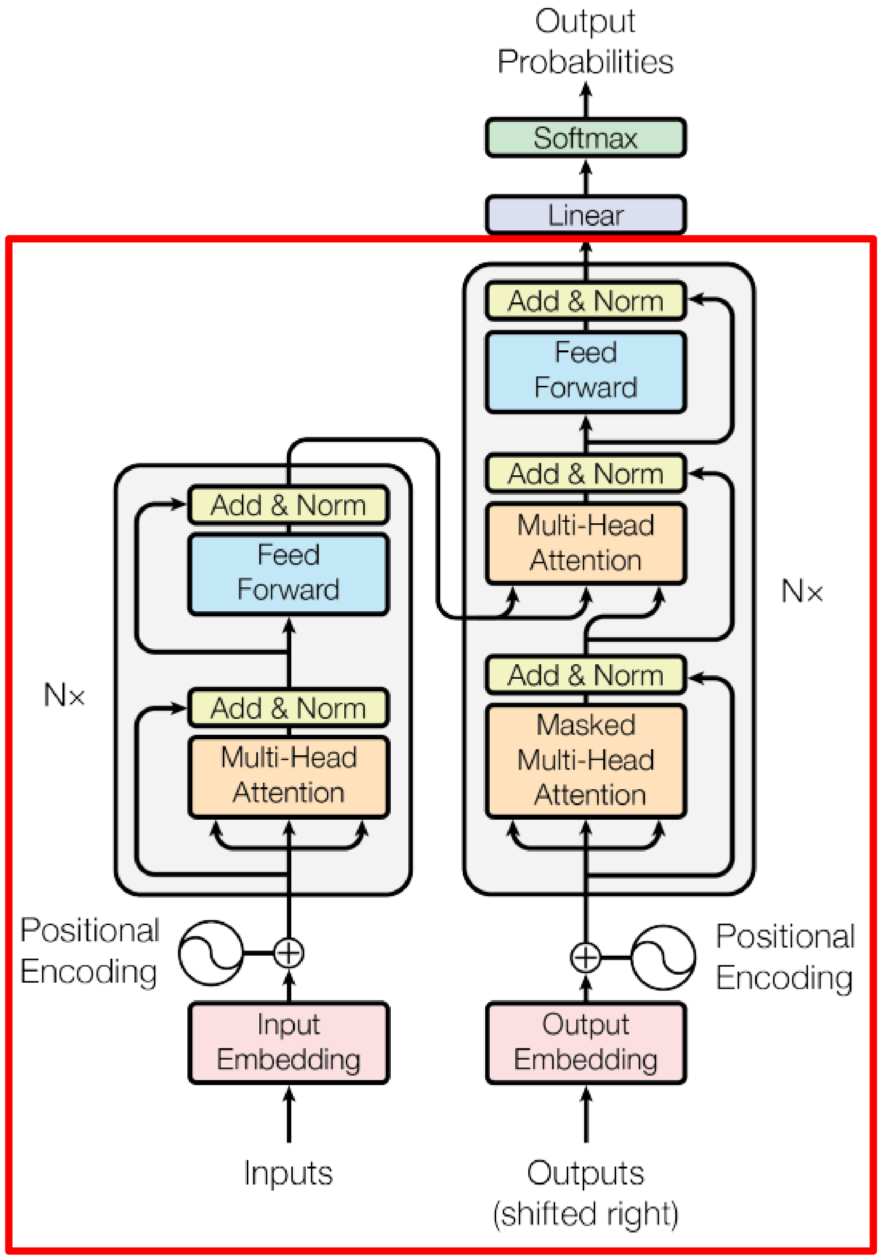
- **Attention mechanism: ≈ 2 x 6 x 1M**
  - Encoder – decoder attention
  - Masked self-attention
- **Feed Forward: ≈ 6 x 2.1M**
  - (512+1) x 2048 + (2048+1) x 512
- **Total: ≈ 24.6M**



Number of Parameters? Base Model

- **N=6, d_{model}=512, d_{ff}=2048, h=8, d_k=64, d_v=64**

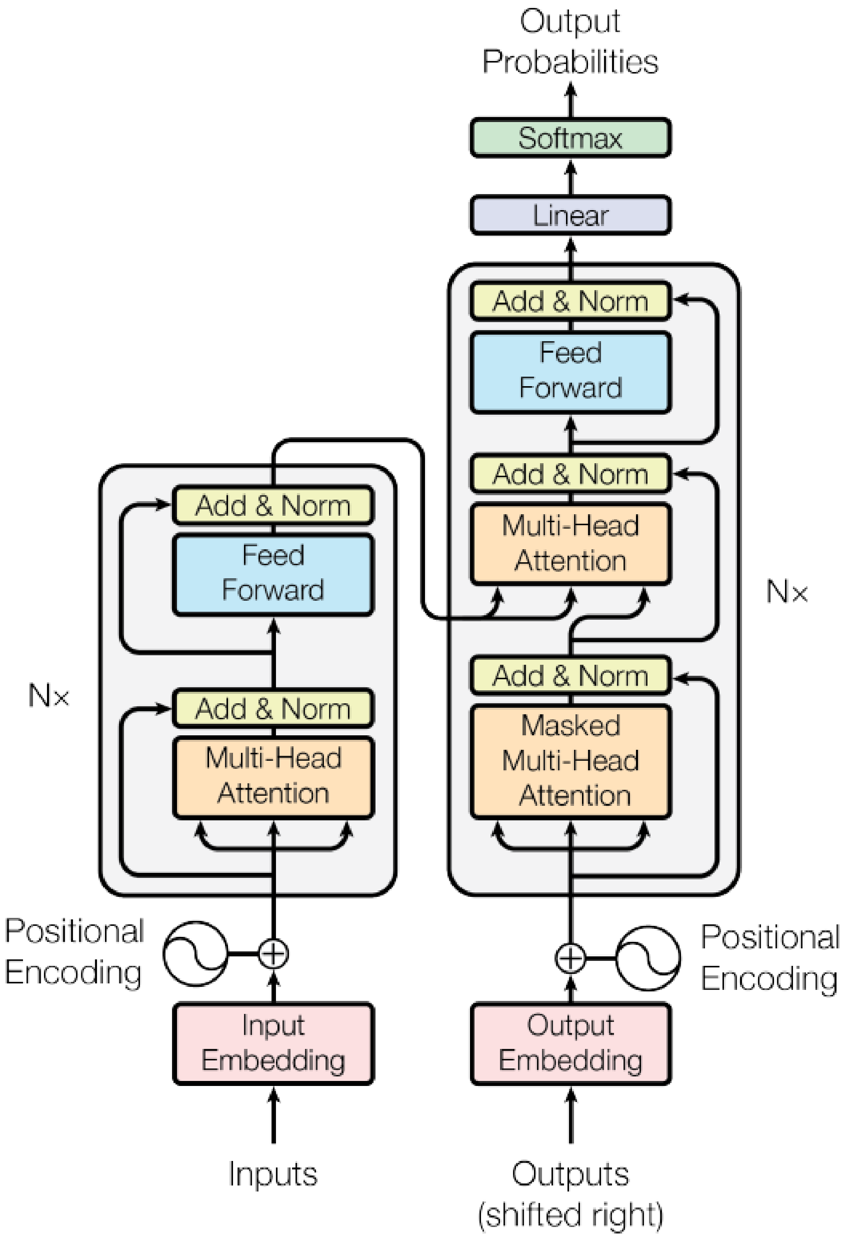
- **Total:**
  - Input/output embedding: ≈ 19M
  - Encoder: ≈ 18.3M
  - Decoder: ≈ 24.6M
- **≈ 61.9M parameters ***



* Paper states 65M Parameters ; this is due to estimations and some details we omitted (Norm)

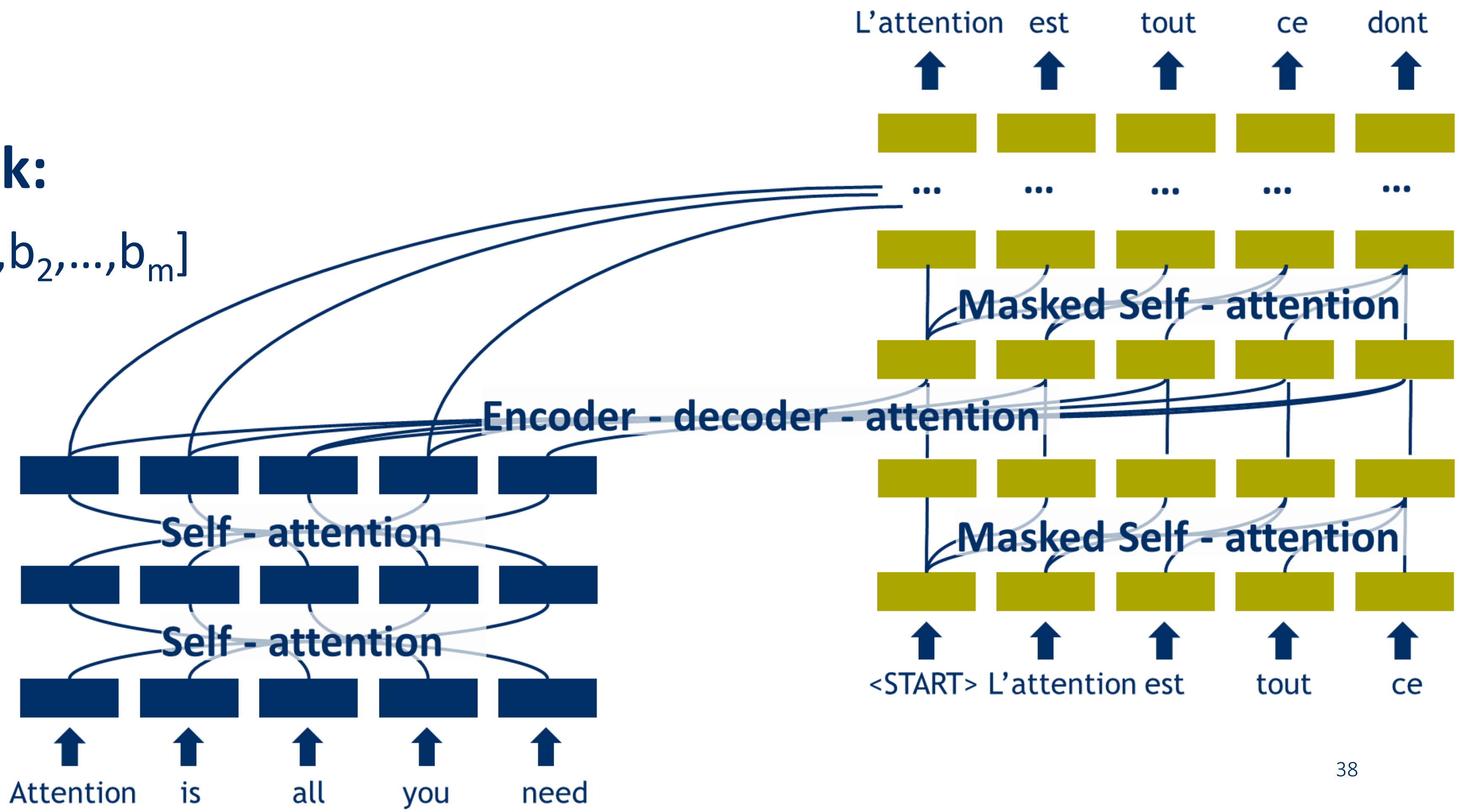
Number of Parameters?

- **Base Model**
  - N=6, d_{model}=512, d_{ff}=2048, h=8, d_k=64, d_v=64
  - ≈ 65M parameters
- **Big Model**
  - N=6, d_{model}=1024, d_{ff}=4096, h=16, d_k=64, d_v=64
  - ≈ 213M parameters



Training Regime

- **Dataset consists of pairs:**
  - Text A = $[a_1,a_2,\dots,a_n]$
  - Text B = $[b_1,b_2,\dots,b_m]$
- **Input to the network:**
  - $[a_1,a_2,\dots,a_n,\text{<BOS>},b_1,b_2,\dots,b_m]$
- **Expected output:**
  - $[b_1,b_2,\dots,b_m]$

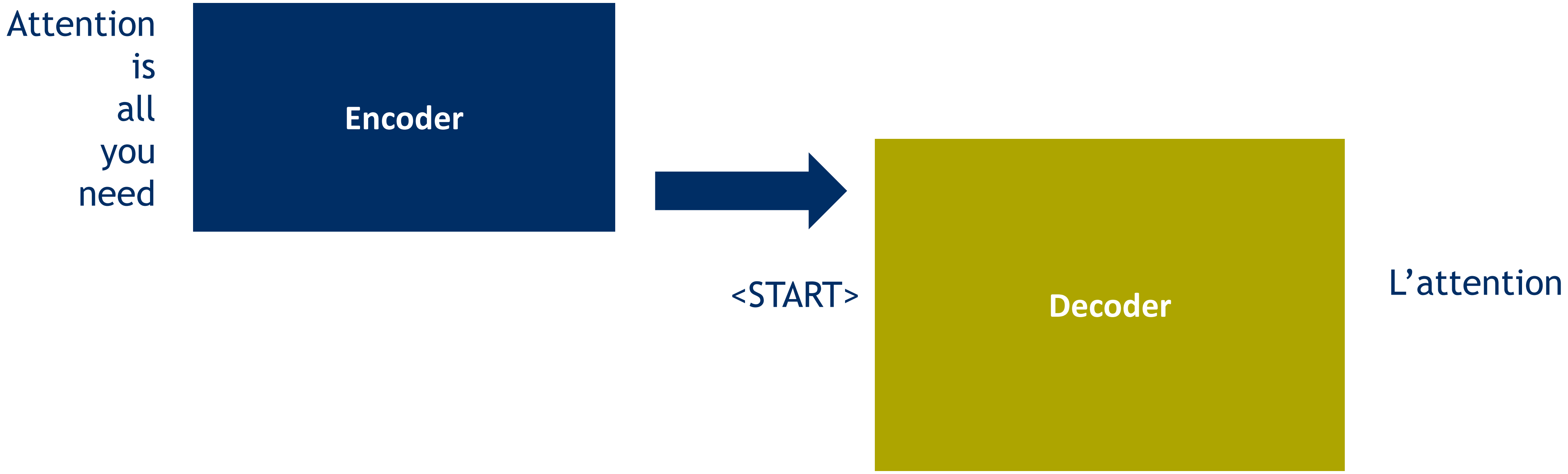


Training Regime

- **Objective function: logloss**
  - Model output = 1 distribution over tokens per output slot
  - Reward high probabilities for the correct token
- **Adam optimizer**
- **Different regularization techniques were used**
  - Dropout, Label smoothing
- **Learning rate varied during training**
  - First increase, then decrease

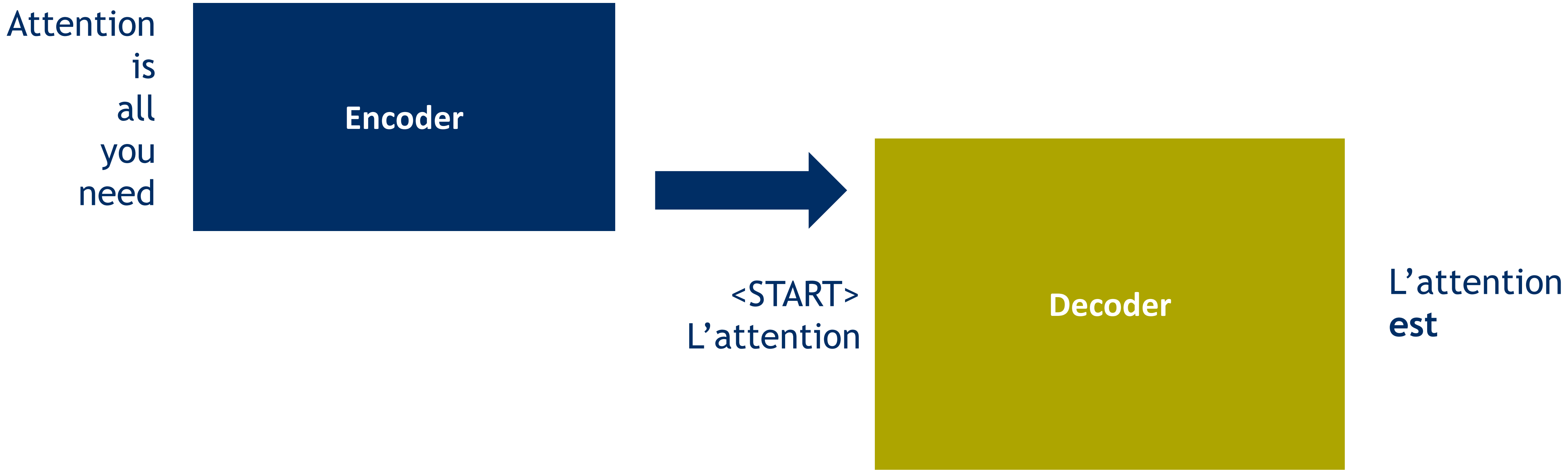
Testing: Generating All Answers

- Generate response tokens one by one
  - “autoregression”



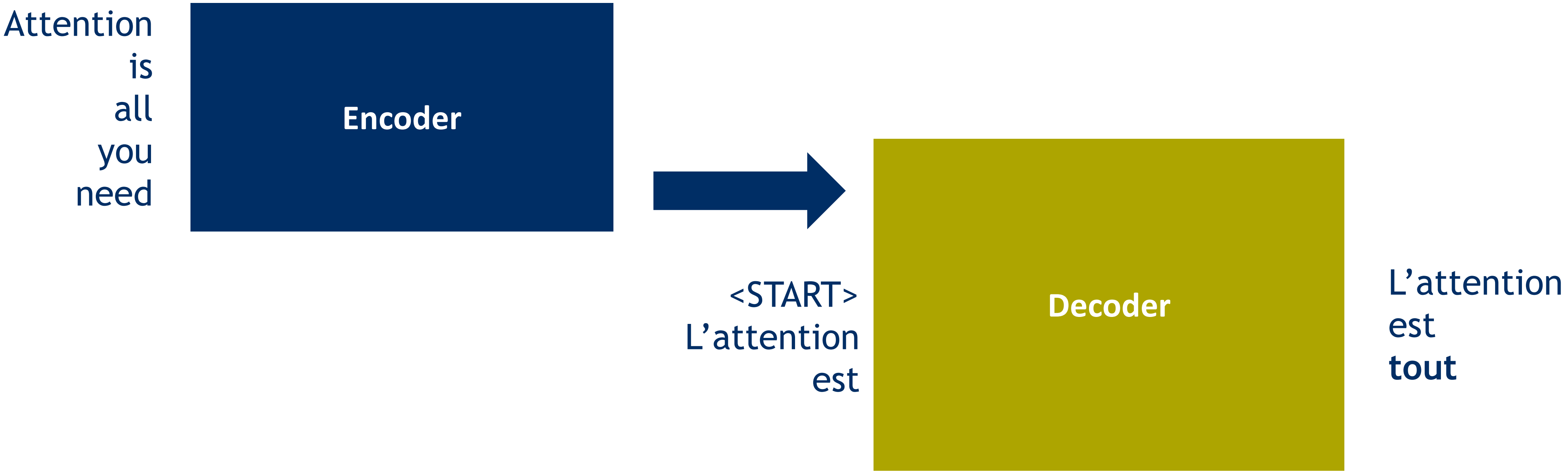
Testing: Generating All Answers

- Generate response tokens one by one
  - “autoregression”



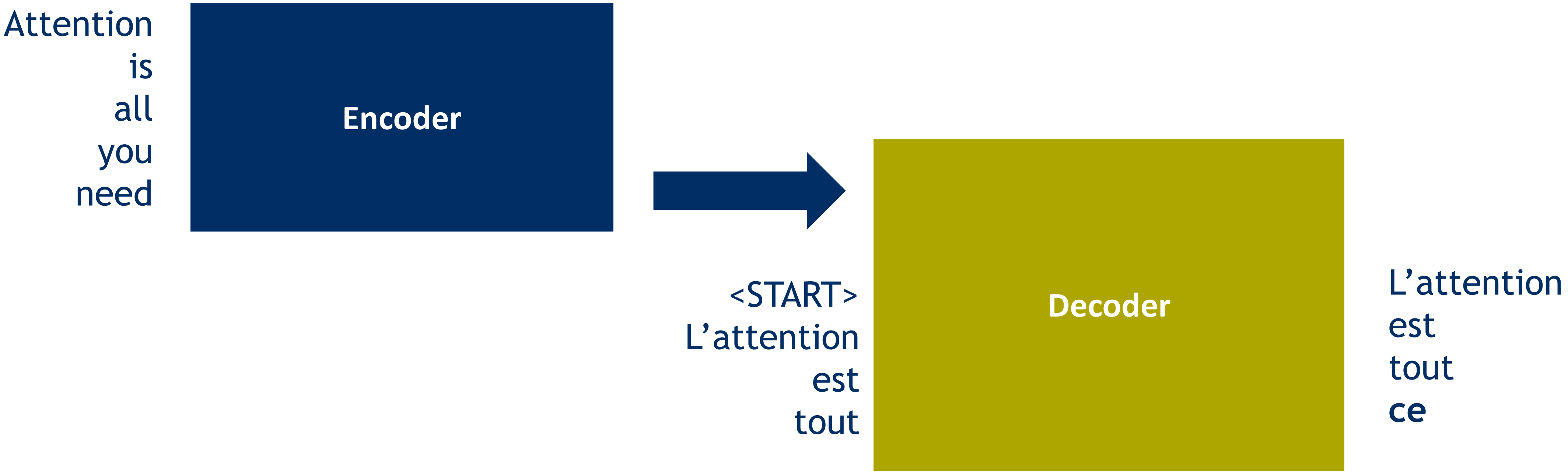
Testing: Generating All Answers

- Generate response tokens one by one
  - “autoregression”



Testing: Generating All Answers

- Generate response tokens one by one
  - “autoregression”



Experimental Results

- **2 translation tasks: EN-DE and EN-FR**
- **BLEU score used to assess quality of result (higher is better)**

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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

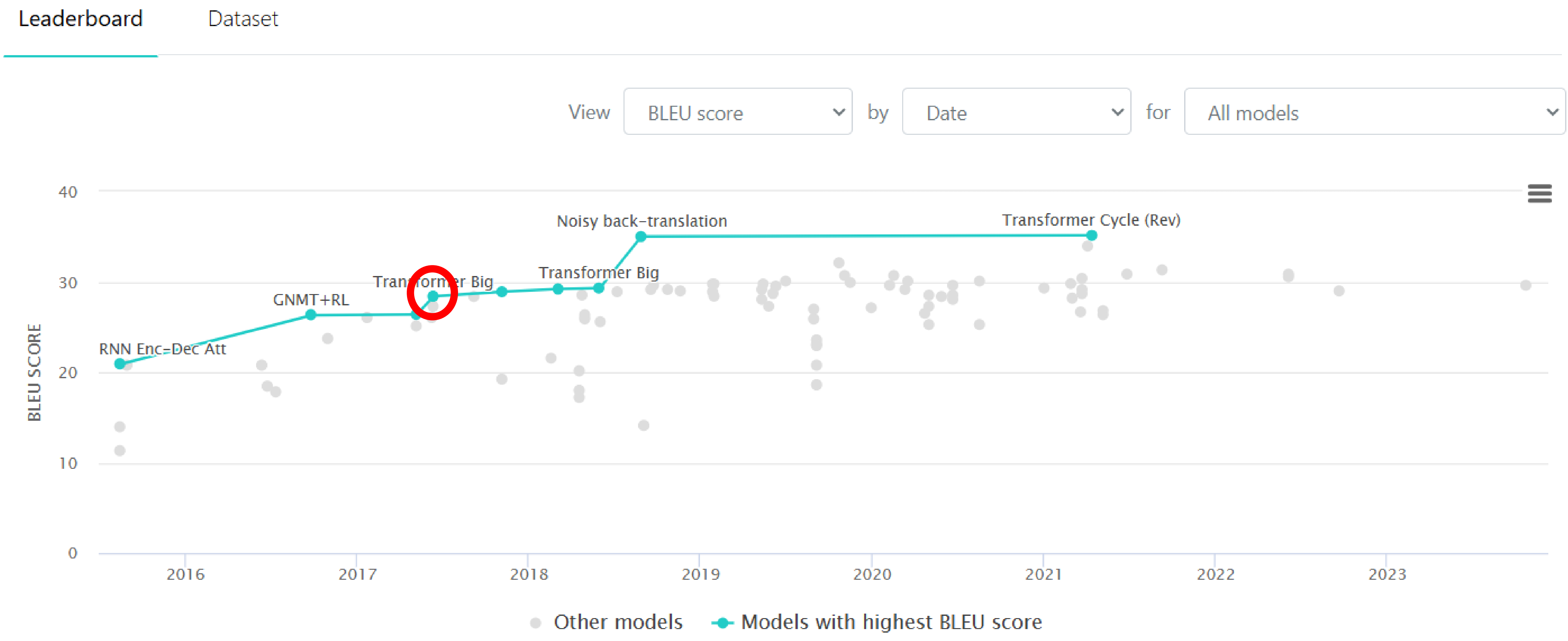
Conclusion

- **New architecture Transformer proposed**
  - Based on attention mechanism
  - No recurrent neural nets, convolutions needed

- **Experiments show promising behavior**
- **Translation task:**
  - Outperforms state-of-the art in accuracy
  - For lower or comparable training costs

- **… and the rest is history …**

Machine Translation on WMT2014 English-German



University of Antwerp | Adrem | Adrem Data Lab

https://paperswithcode.com/task/machine-translation

Sources

- Vaswani, A. et al. (2017). Attention is all you need. Advances in Neural Information Processing Systems.
- Cho, K. et al. (2014). On the properties of neural machine translation: Encoder–Decoder approaches. In Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation.
- Bahdanau, D. et al. (2015). "Neural machine translation by jointly learning to align and translate." 3rd International Conference on Learning Representations
- Papineni, K. et al. (2002). Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics (pp. 311-318).
- Yang, J. et al. (2024) "Harnessing the power of llms in practice: A survey on chatgpt and beyond." ACM Transactions on Knowledge Discovery from Data 18.6: 1-32.
- Raschka, S. (2024). Build a Large Language Model (From Scratch). Simon and Schuster.

- https://www.kaggle.com/datasets/mohamedlotfy50/wmt-2014-english-german
- https://paperswithcode.com/task/machine-translation
- https://jalammar.github.io/illustrated-transformer/