Transformers for NLP

Transformers and LLM

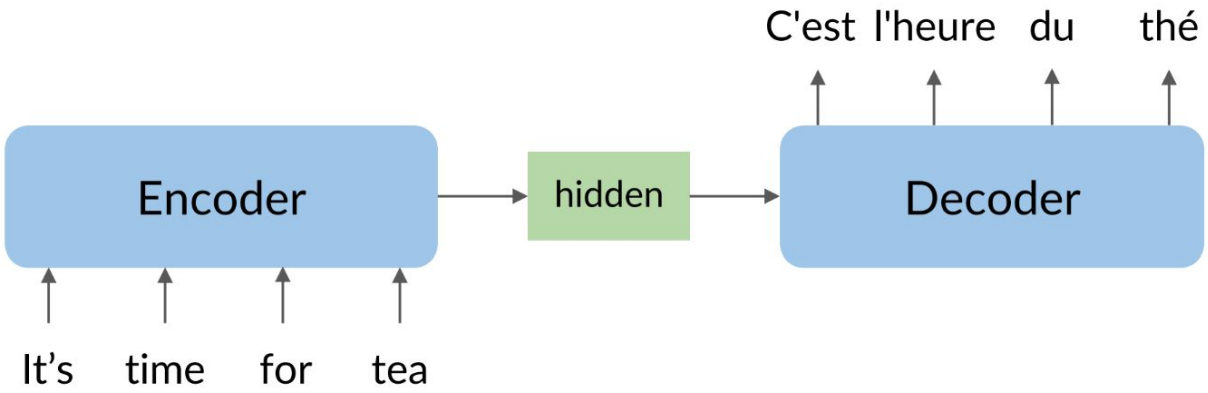
Outline

1. What is transformers
2. Explain how RNN’s seq2seq works
3. Why transformers. Explain with RNN/LSTM reference. Explain problems of memory issue and vanishing gradients. Advantages of Transformers. Parallel processing, Alleviating Vanishing Gradient Problem, Long-Range Dependency Capture, Transfer Learning Capabilities.
4. Components of transformers. Everything in single image
5. Explain basic components one by one. 1. Encoder 2. Self-Attention 3. Multi-head self attention 4. Decoder 5. Masked self attention 6. Cross Self Attention [Encode-decoder attention]

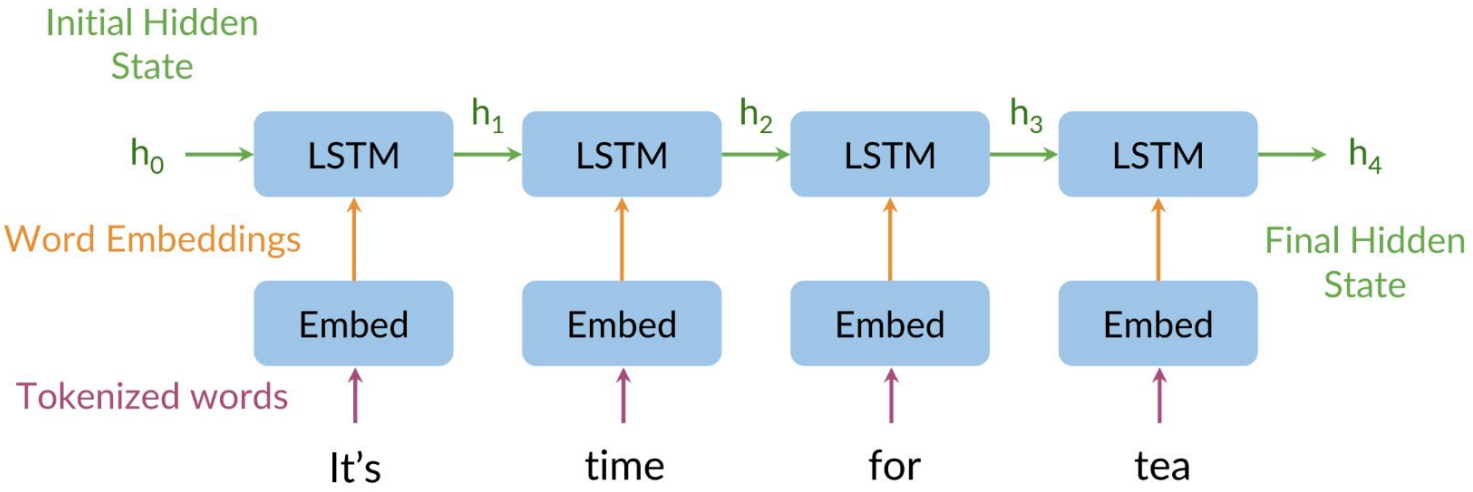
Seq2Seq model

- Introduced by Google in 2014
- Maps variable-length sequences to fixed-length memory
- Inputs and outputs can have different lengths
- LSTMs and GRUs to avoid vanishing and exploding gradient problems

Seq2Seq model

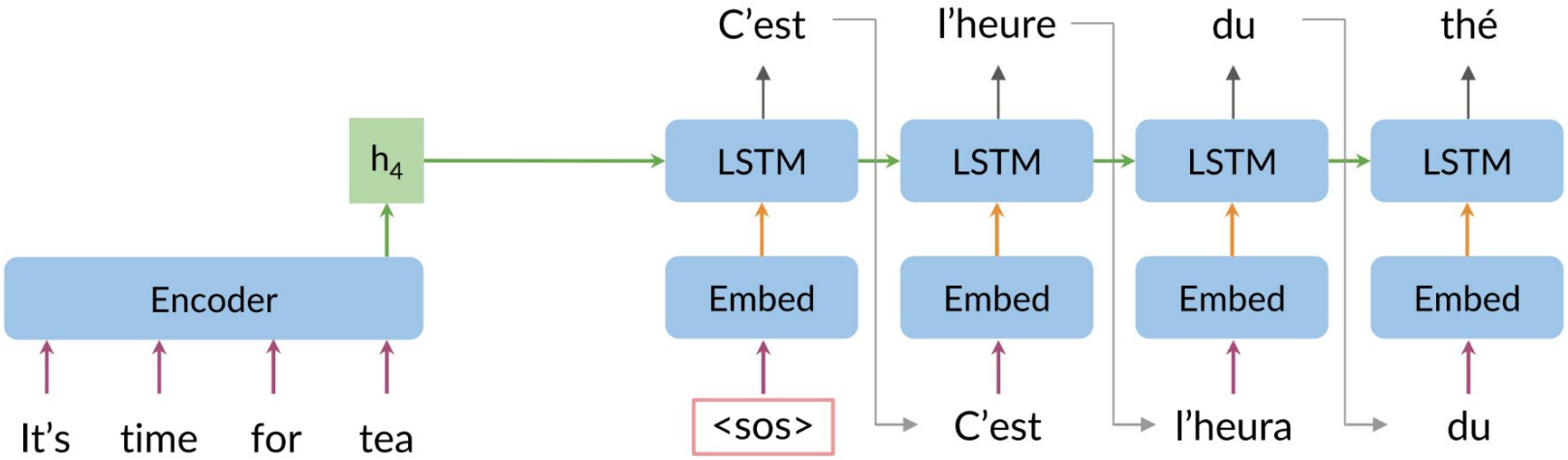


Seq2Seq encoder



Encodes the overall meaning of the sentence

Seq2Seq decoder

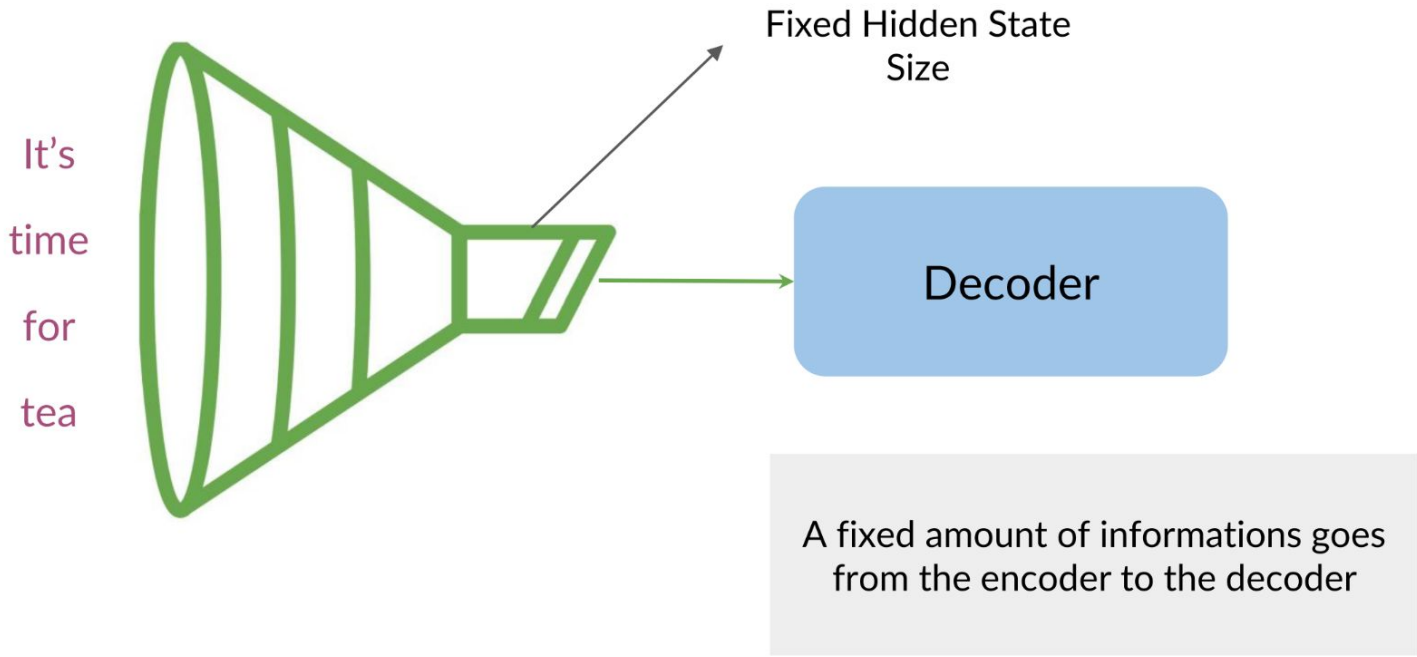


Seq2Seq shortcomings

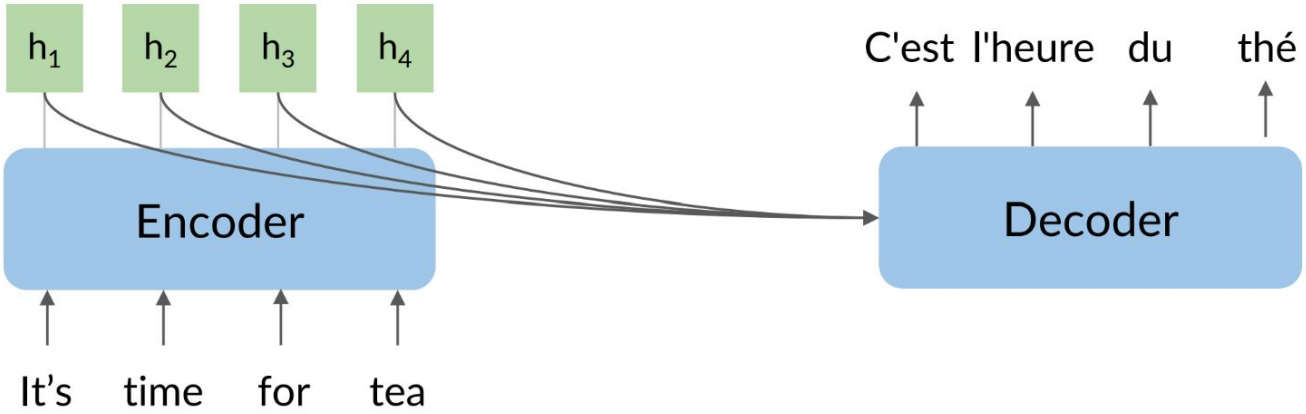
- Variable-length sentences + fixed-length memory =

- As sequence size increases, model performance decreases

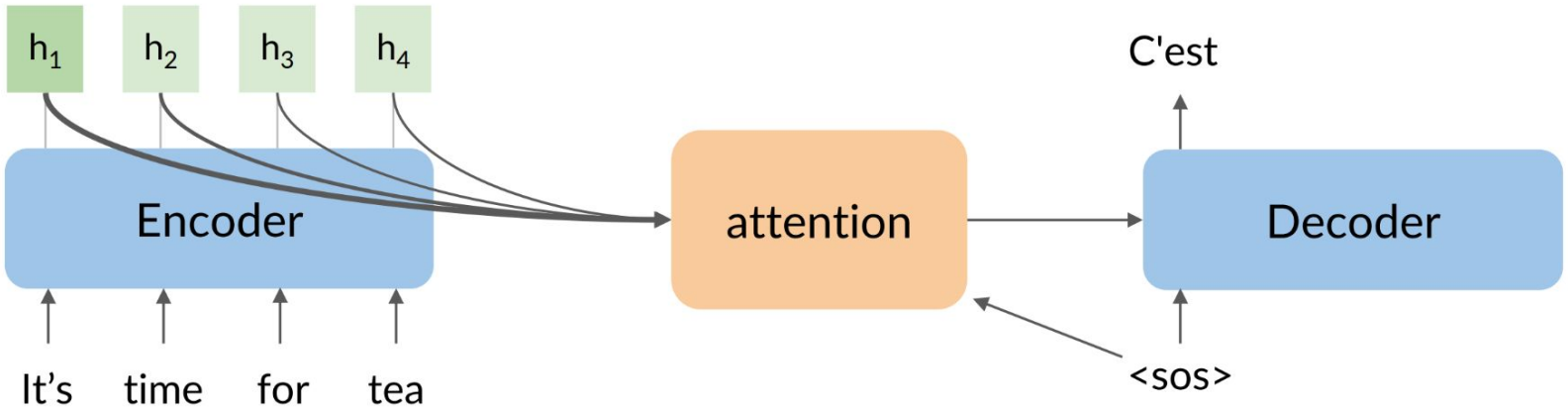
The information bottleneck



Use all the encoder hidden states?



Solution: focus attention in the right place



The model can focus on specific hidden states at every step

Seq2Seq model with attention

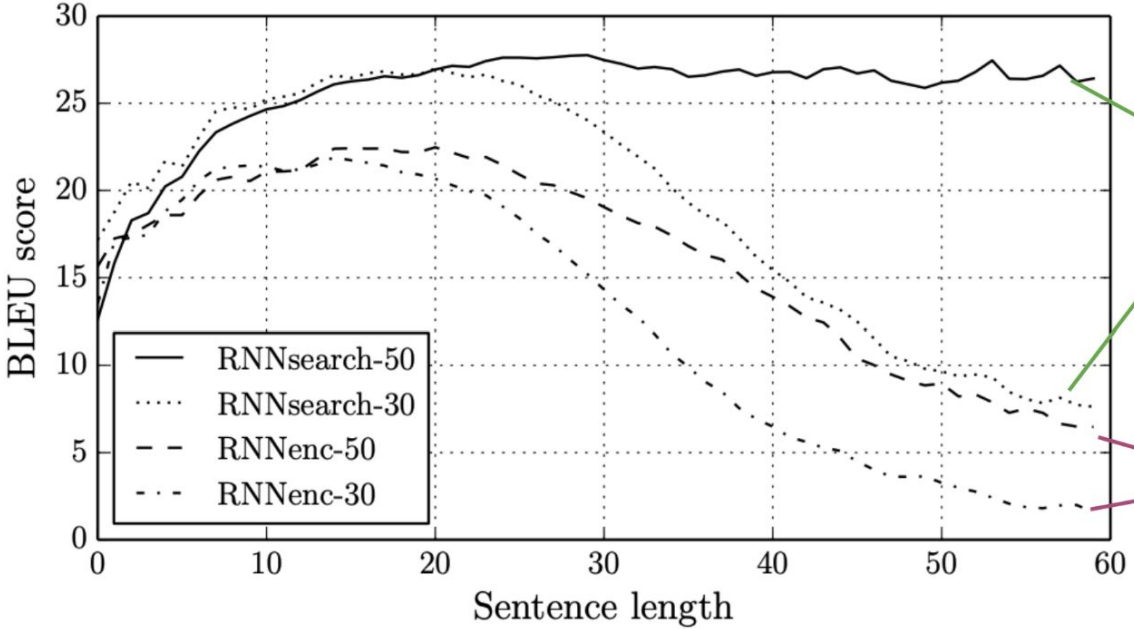
NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

1409.0473.pdf (arxiv.org)

Performance

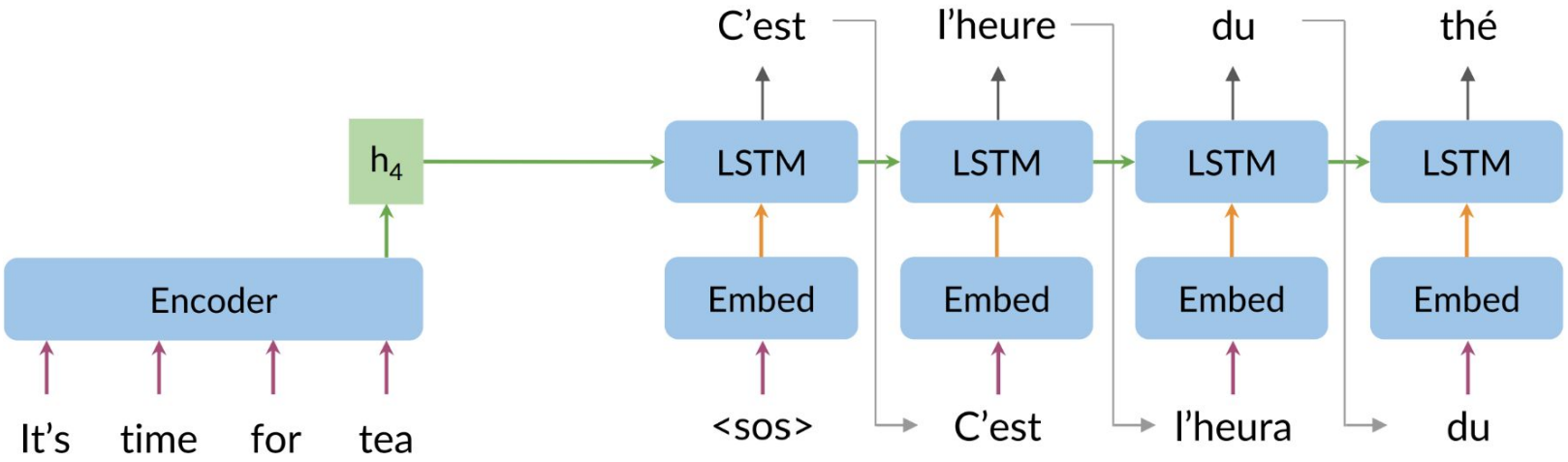


Seq2Seq with Attention

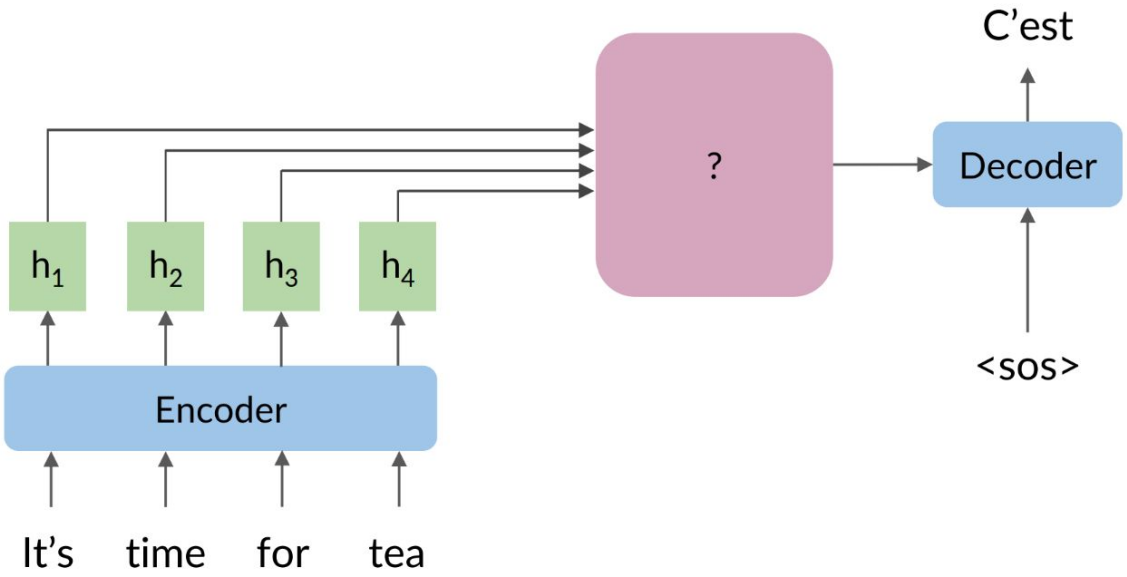
Traditional Seq2Seq Models

Greater BLEU is better

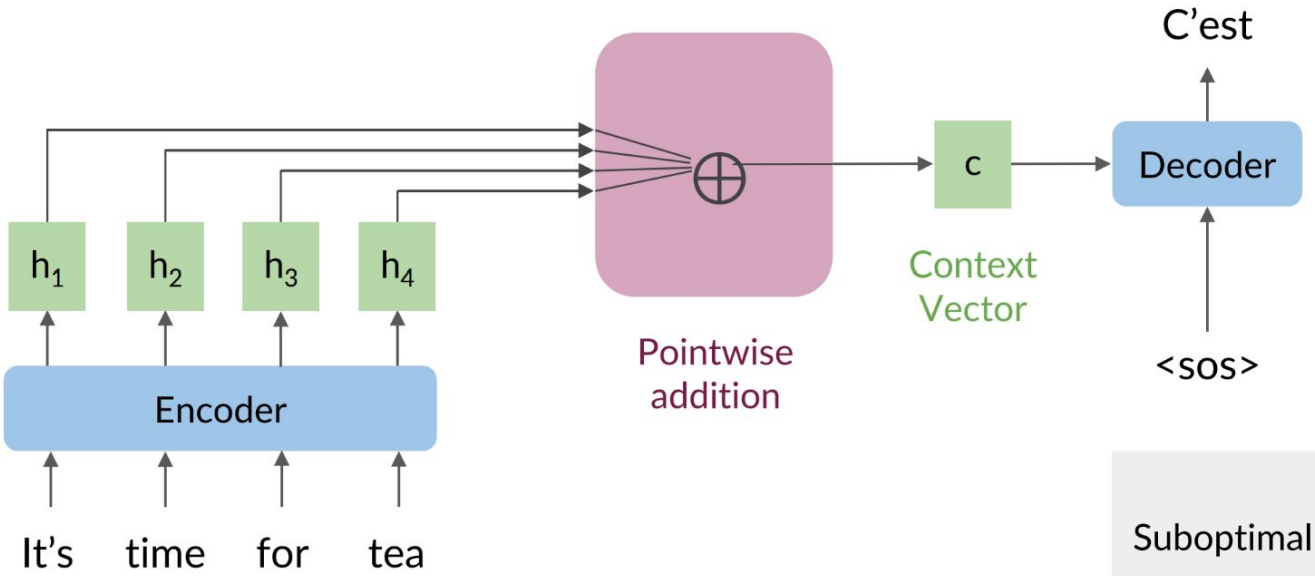
Traditional seq2seq models



How to use all the hidden states?

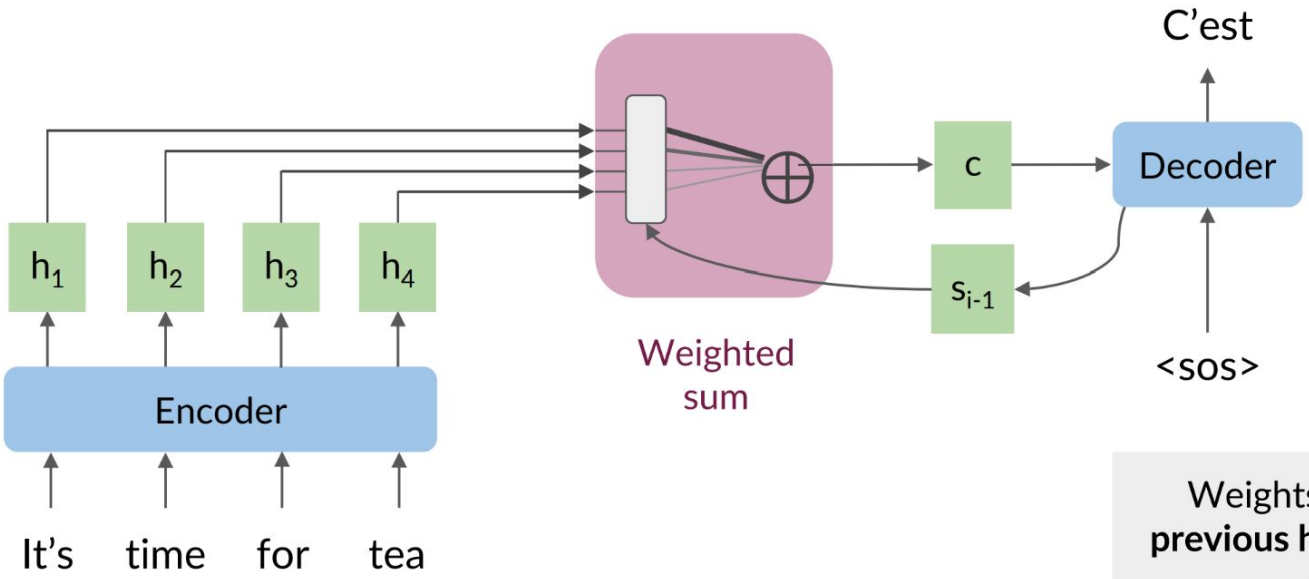


How to use all the hidden states?



Suboptimal use of information

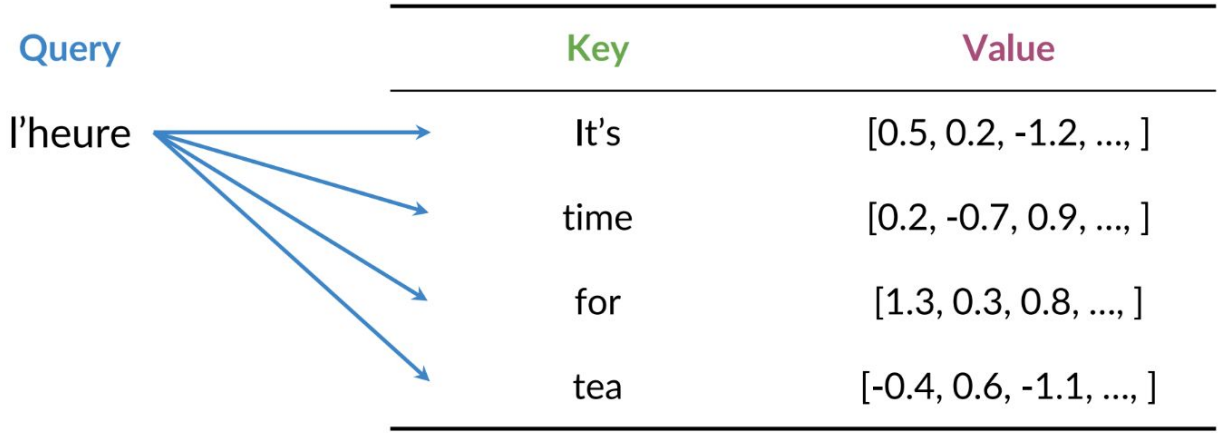
How to use all the hidden states?



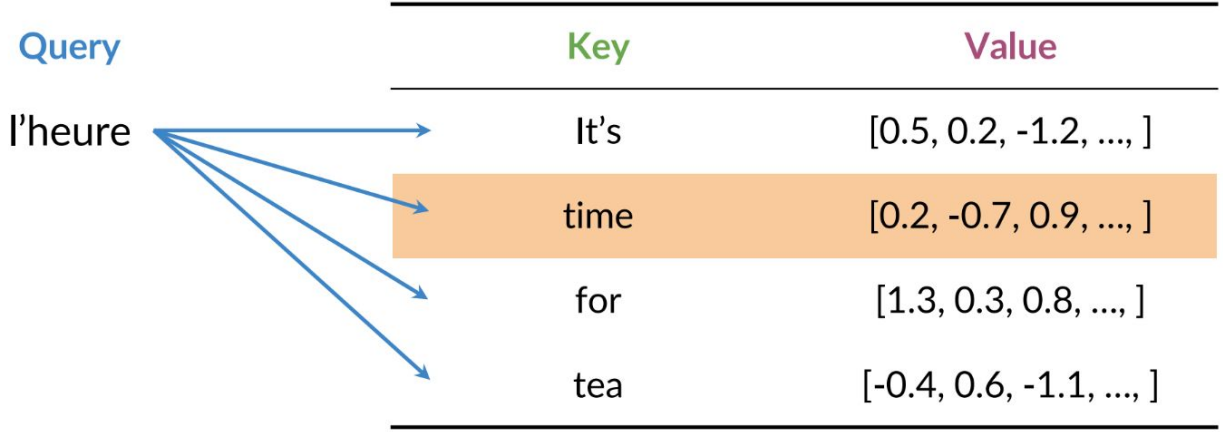
Weights depend on the **previous hidden state** in the decoder

Queries, Keys, Values and Attention

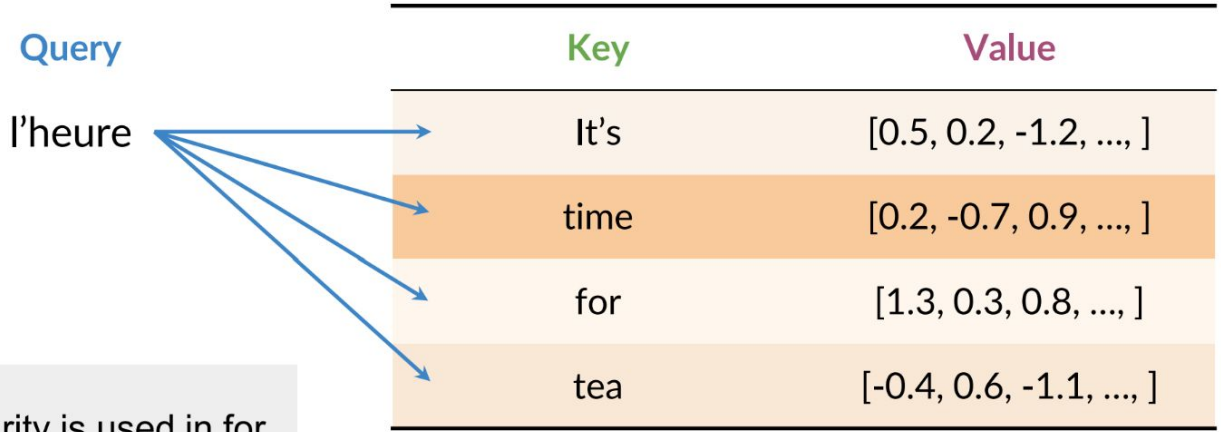
Queries, Keys, Values



Queries, Keys, Values

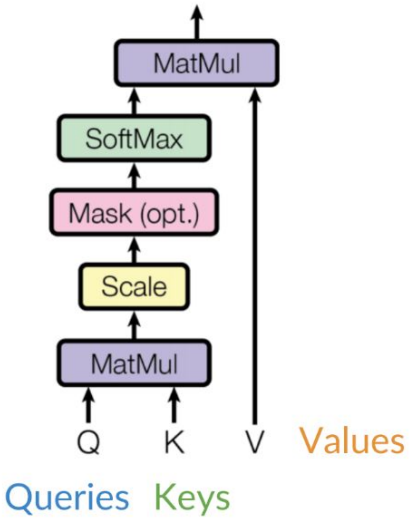


Queries, Keys, Values



Similarity is used in for weighted sum

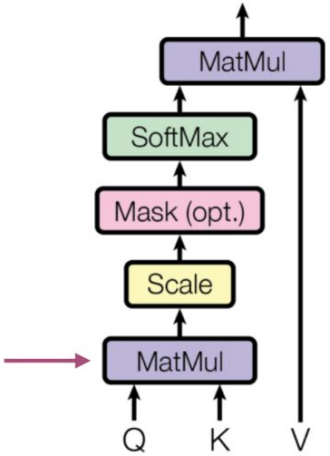
Scaled dot-product attention



$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

(Vaswani et al., 2017)

Scaled dot-product attention

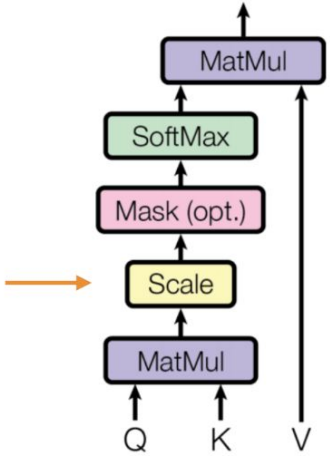


(Vaswani et al., 2017)

Similarity Between
Q and K

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Scaled dot-product attention

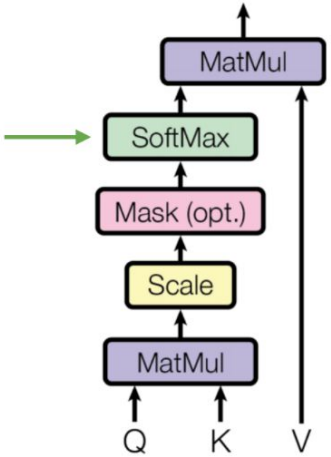


(Vaswani et al., 2017)

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Scale using the root of the key vector size

Scaled dot-product attention

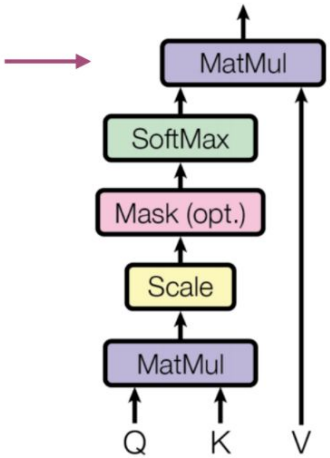


(Vaswani et al., 2017)

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Weights for the weighted sum

Scaled dot-product attention



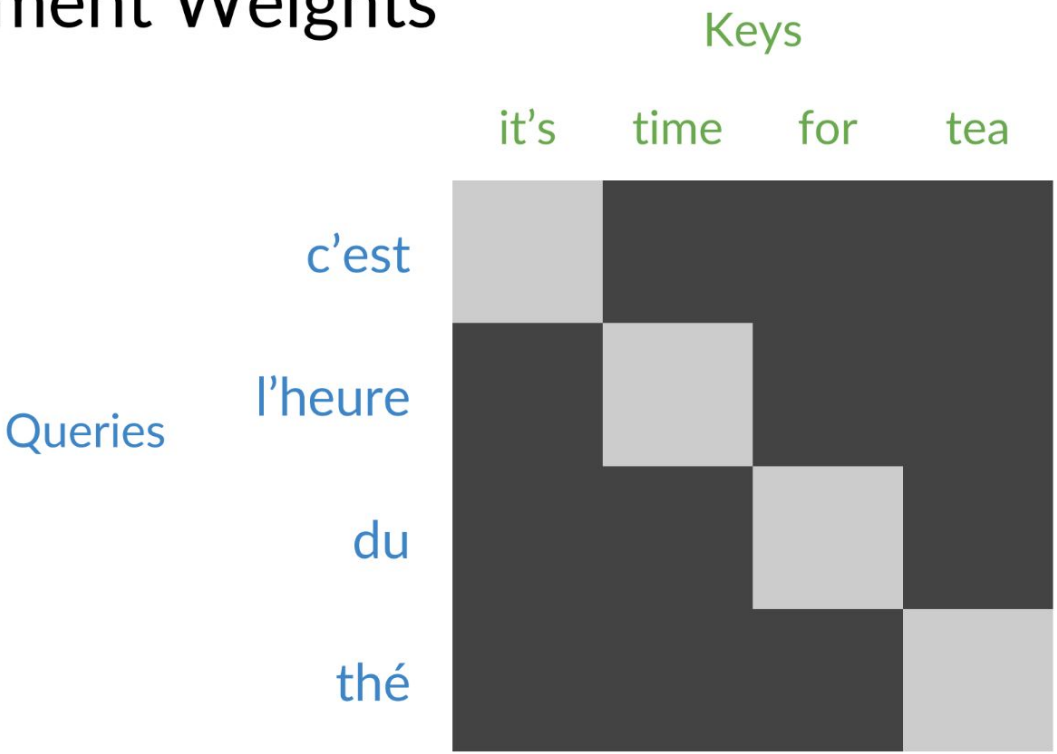
(Vaswani et al., 2017)

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Weighted sum of values V

Just two matrix multiplications and a Softmax!

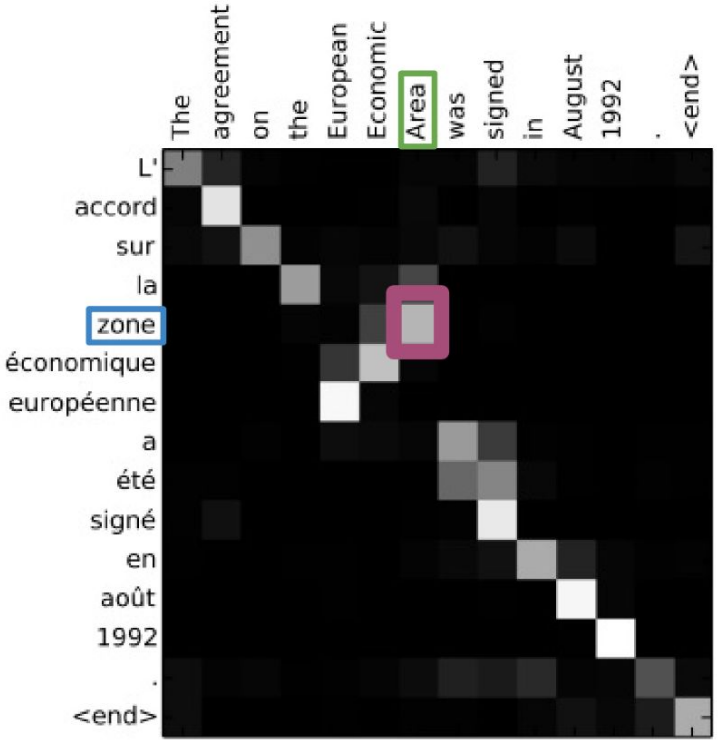
Alignment Weights



Similar words have large weights

Flexible attention

Works for languages with different grammar structures!



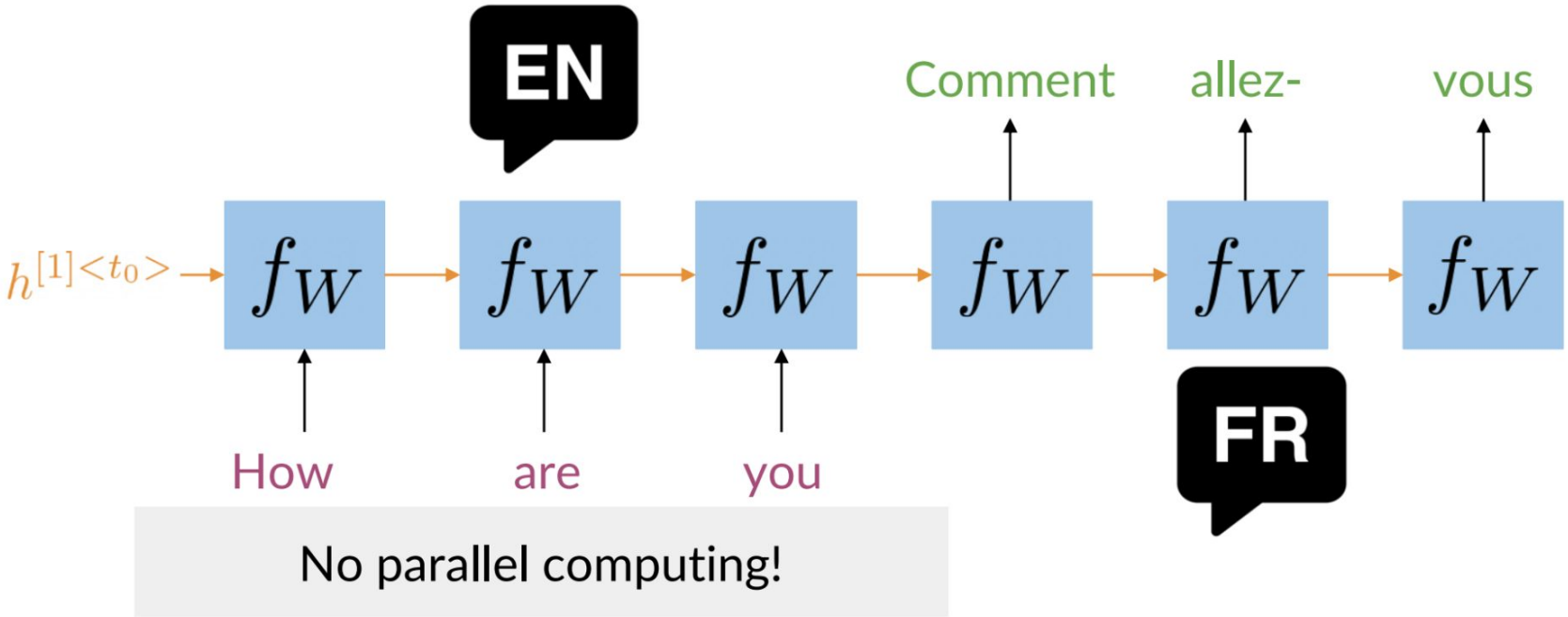
Bahdanau et al., 2015

Summary

- Attention is a layer that lets a model focus on what's important
- Queries, Values, and Keys are used for information retrieval inside the Attention layer
- Works for languages with very different grammatical structures

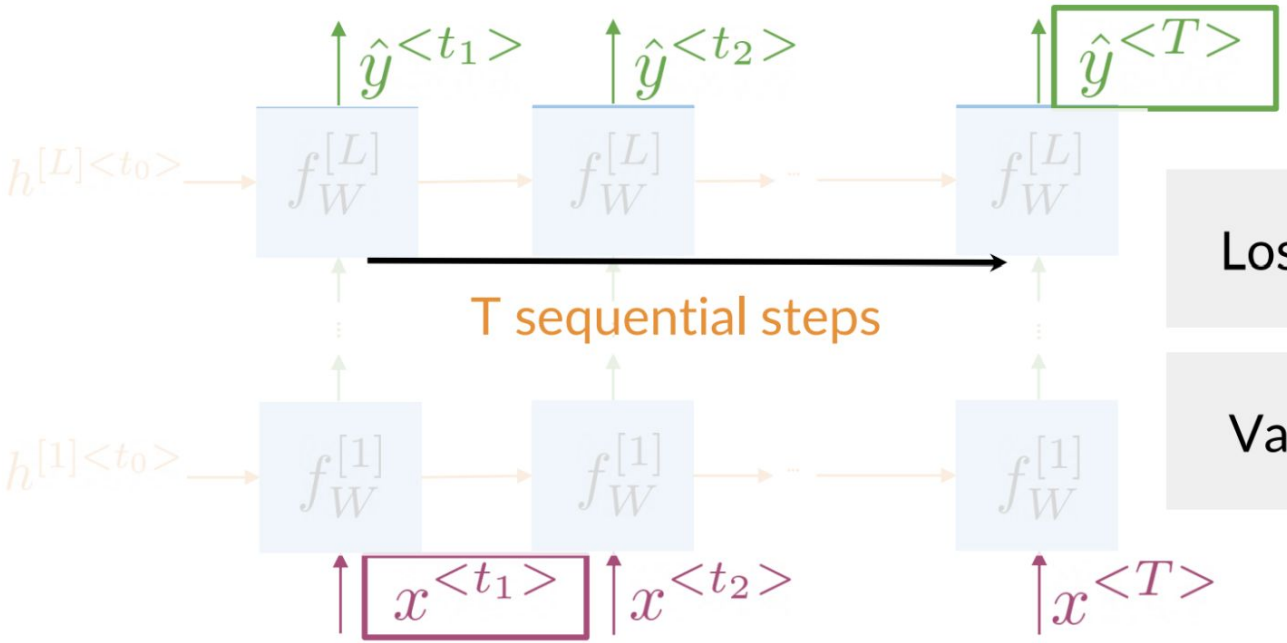
Transformers vs RNNs

Neural Machine Translation



No parallel computing!

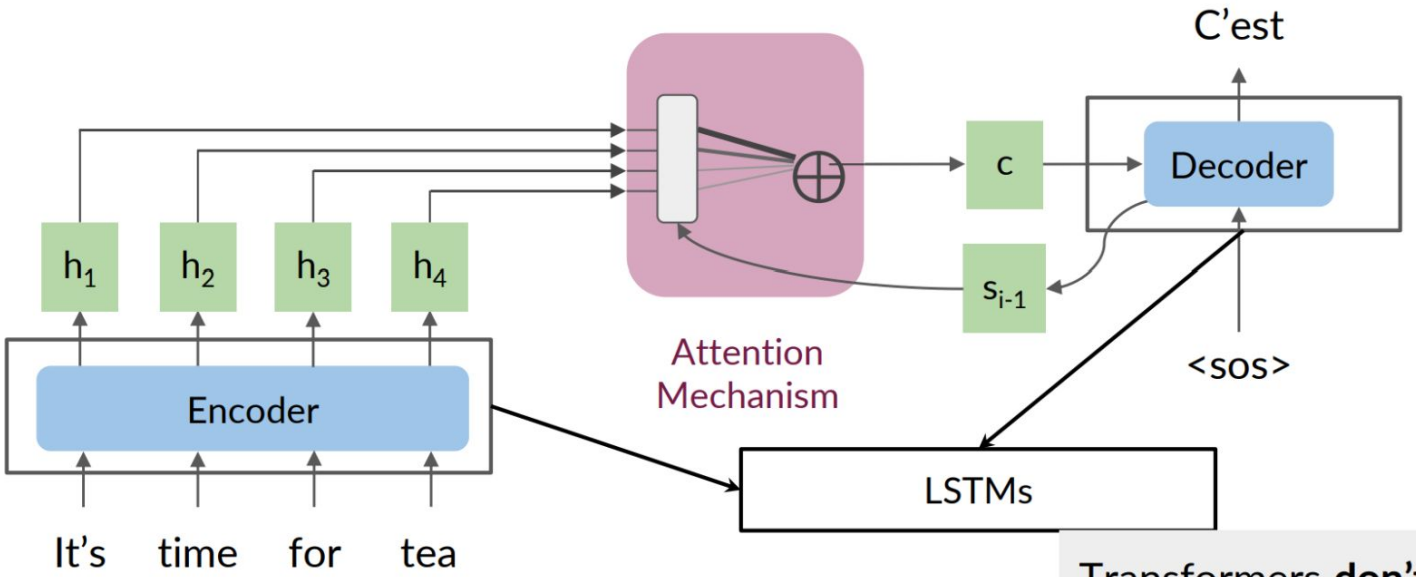
Seq2Seq Architectures



Loss of information

Vanishing gradient

RNNs vs Transformer: Encoder-Decoder



Transformers **don't** use RNNs, such as LSTMs or GRUs

Transformers Overview

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

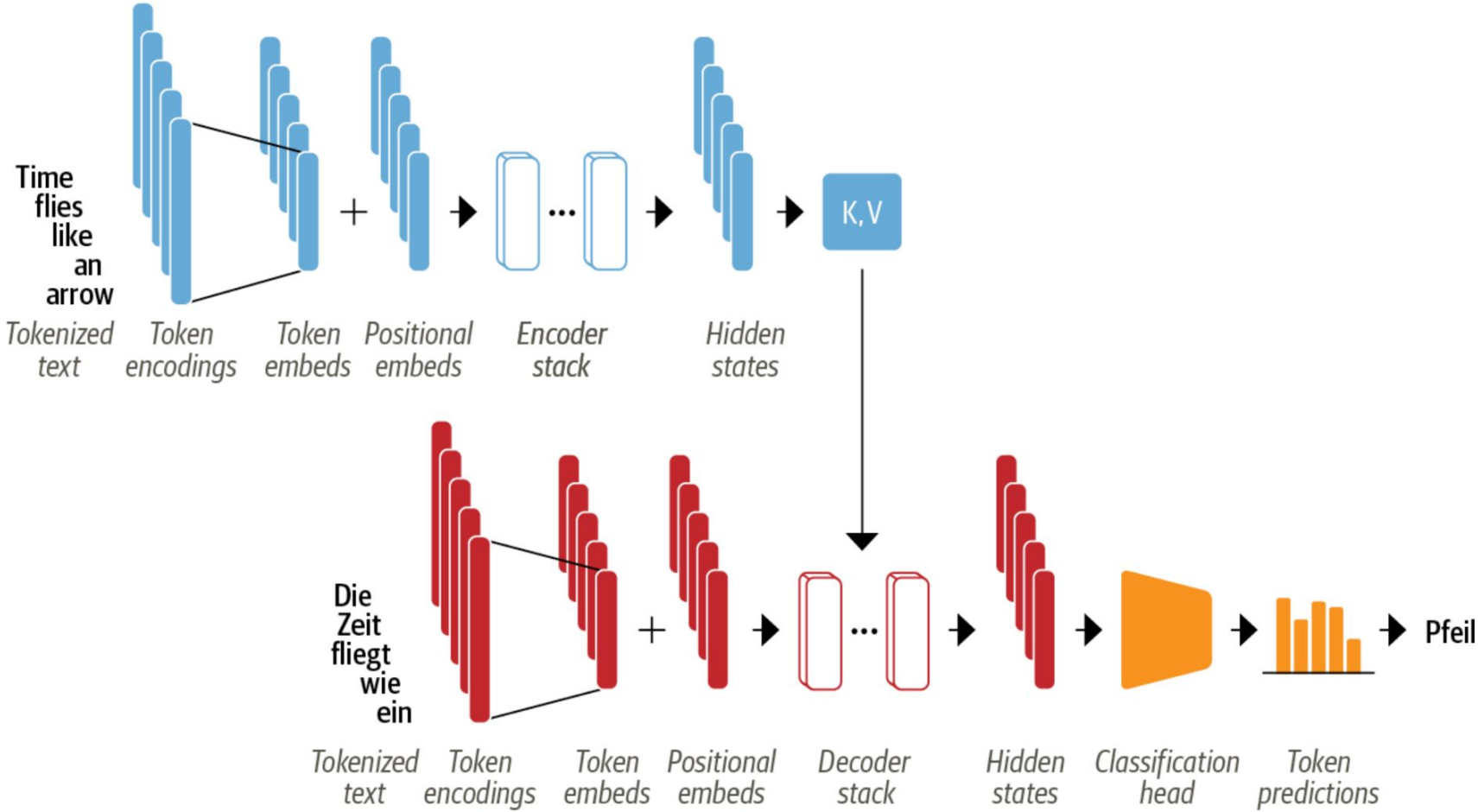
Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

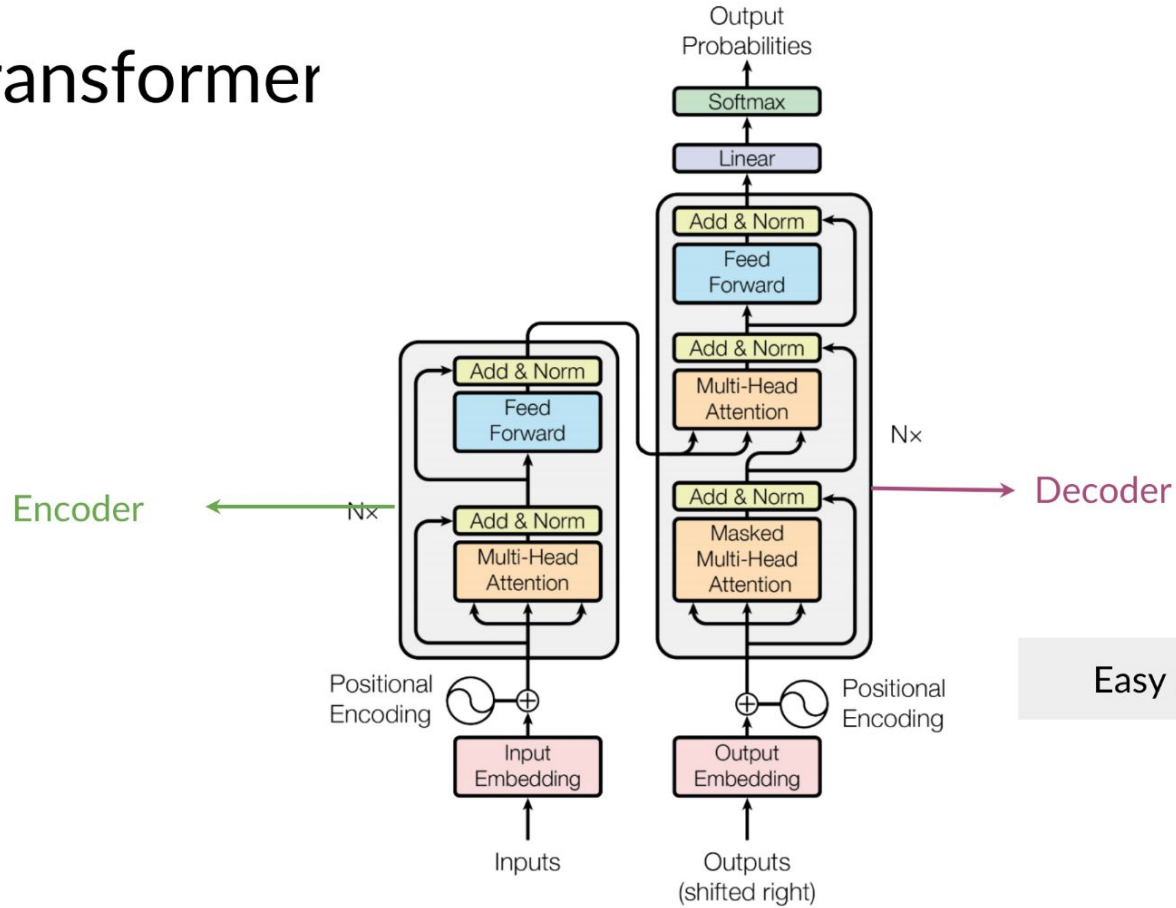
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

1706.03762.pdf (arxiv.org)

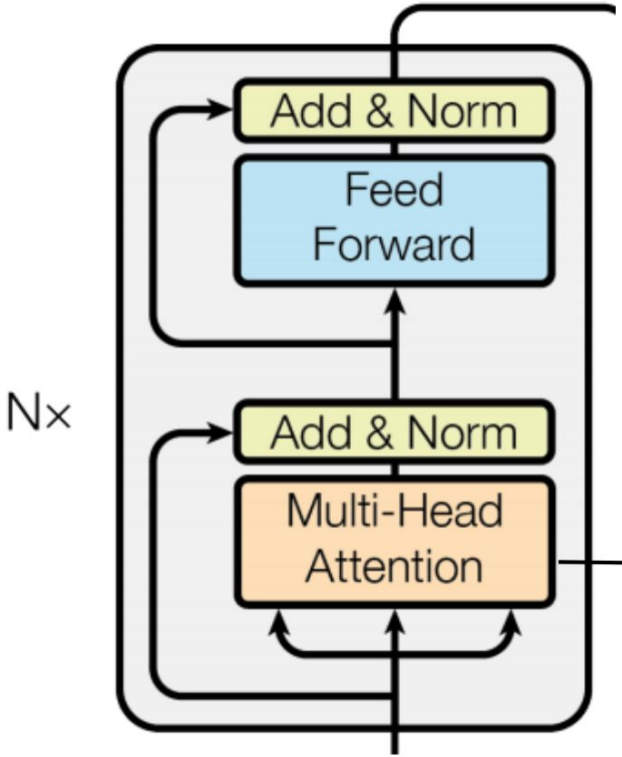

Time
flies
like
an
arrow
K, V
Die
Zeit
fliegt
wie
ein
Pfeil
Tokenized text
Token encodings
Token embeds
Positional embeds
Encoder stack
Hidden states
Tokenized text
Token encodings
Token embeds
Positional embeds
Decoder stack
Hidden states
Classification head
Token predictions

The Transformer



Easy to parallelize!

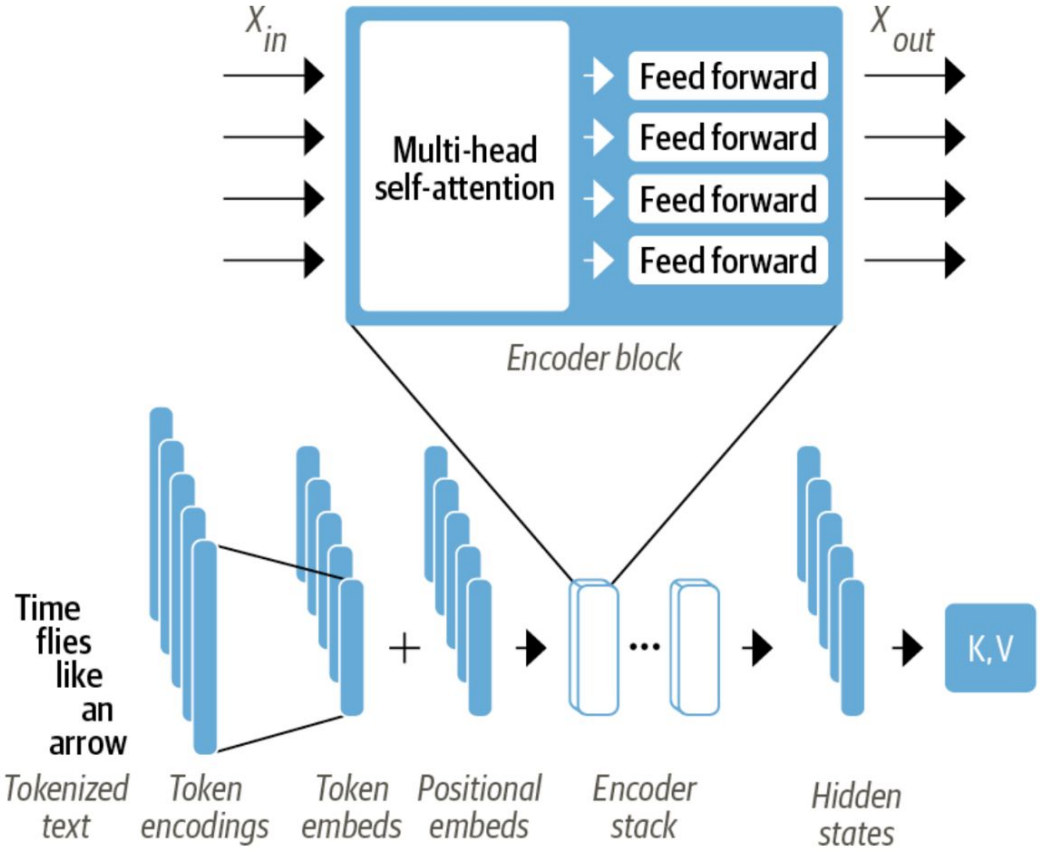
The Encoder



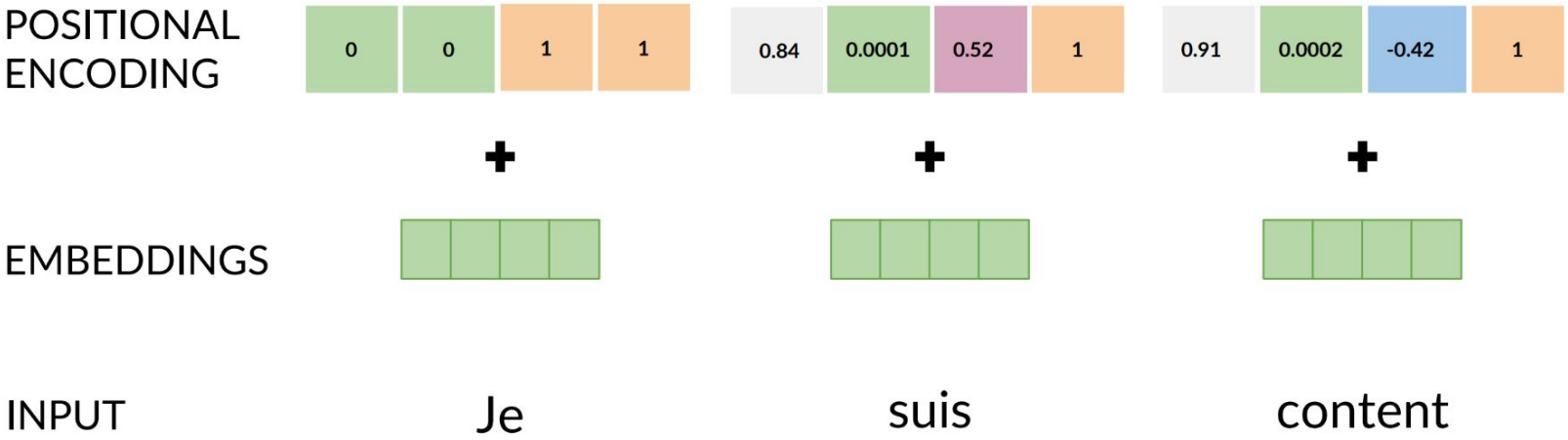
Provides contextual representation of each item in the input sequence

Self-Attention

Every item in the input attends to every other item in the sequence

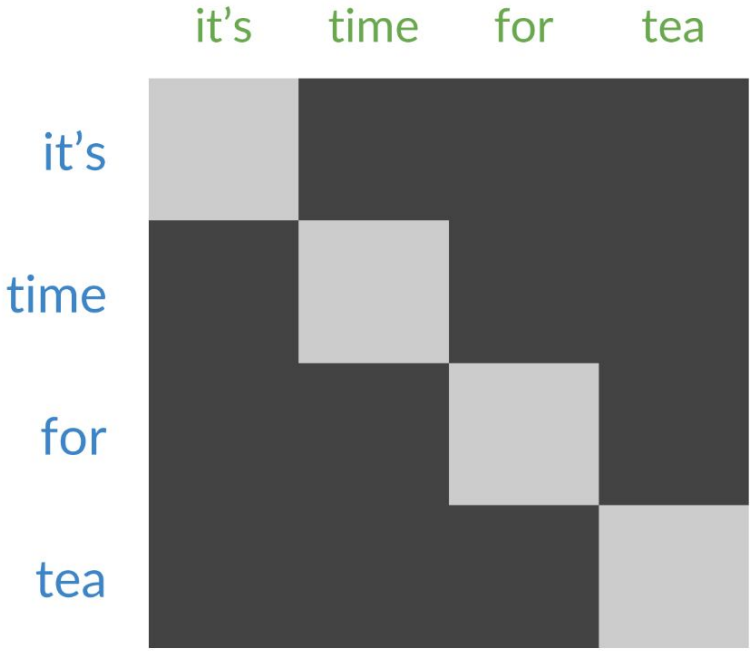

X_{in}
Multi-head self-attention
Feed forward
Feed forward
Feed forward
Feed forward
X_{out}
Encoder block
Time flies like an arrow
K, V
Tokenized text
Token encodings
Token embeds
Positional embeds
Encoder stack
Hidden states

RNNs vs Transformer: Positional Encoding



Self-Attention

Queries, keys and values come from the **same sentence**



Weight matrix

Meaning of each word **within** the sentence

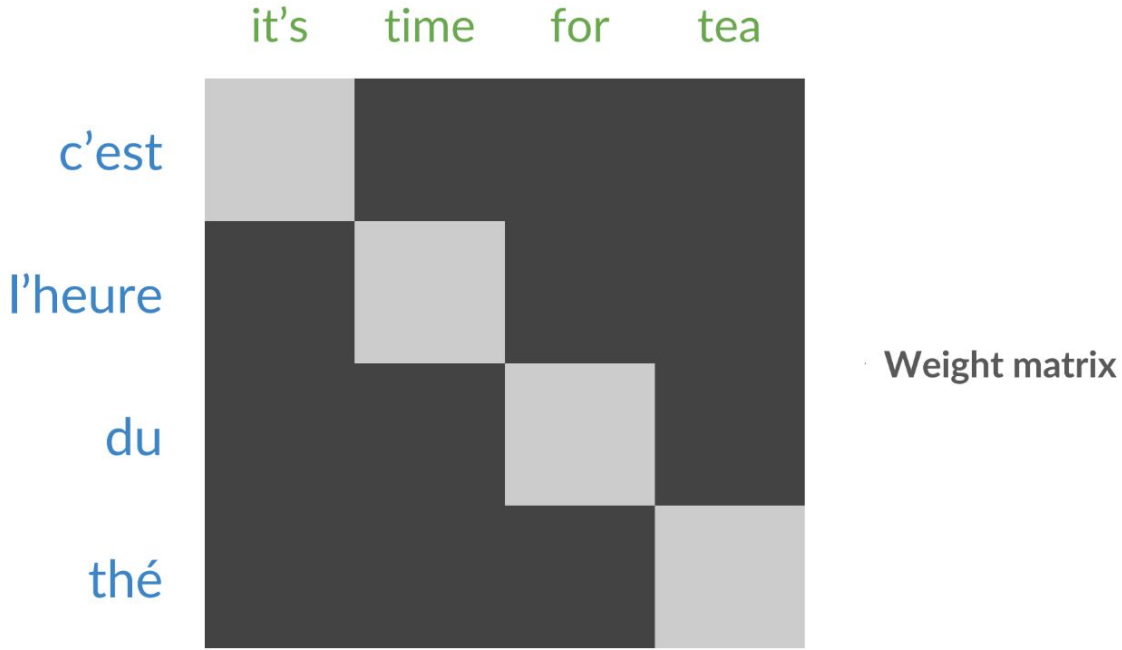
Masked Self-Attention

Queries, keys and values come from the **same sentence**. Queries don't attend to future positions.

Weight matrix

Encoder-Decoder Attention

Queries from one sentence, keys and values from another



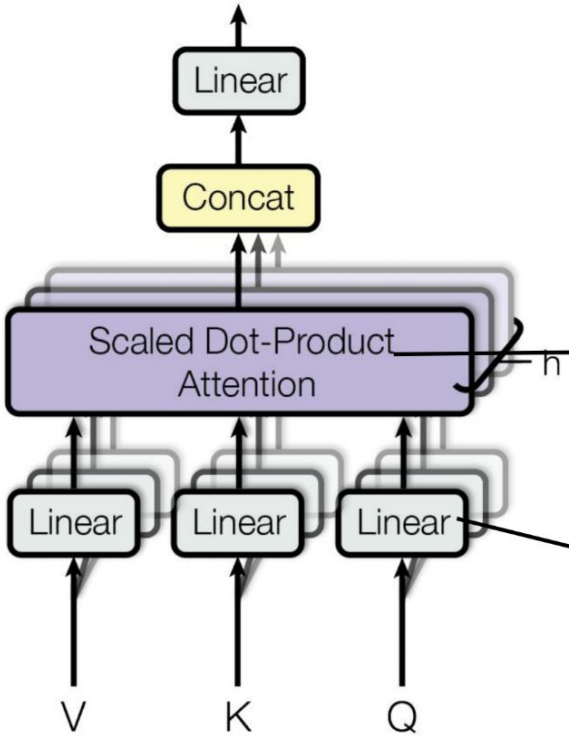
Summary

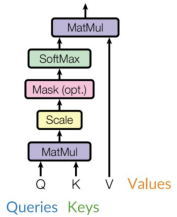
- There are three main ways of Attention: Encoder/Decoder, self-attention and masked self-attention.
- In self-attention, queries and keys come from the same sentence
- In masked self-attention queries cannot attend to the future

Multi-head Attention

Scaled Dot-Product Attention

Multi-Head Attention





(Vaswani et al., 2017)

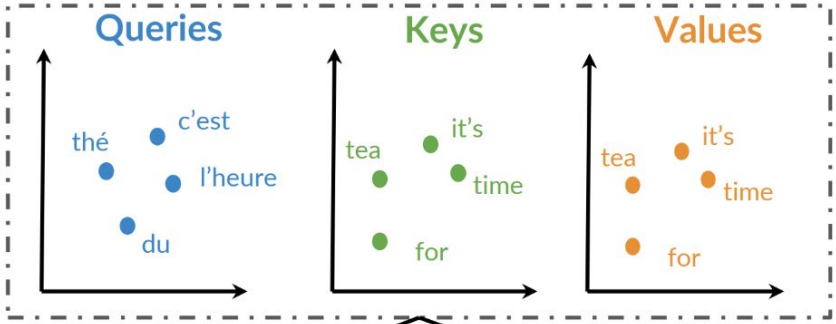
$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Scaled **dot-product attention multiple times** in parallel

Linear transformations of the input queries, keys and values

Multi-Head Attention - Overview

Original Embeddings

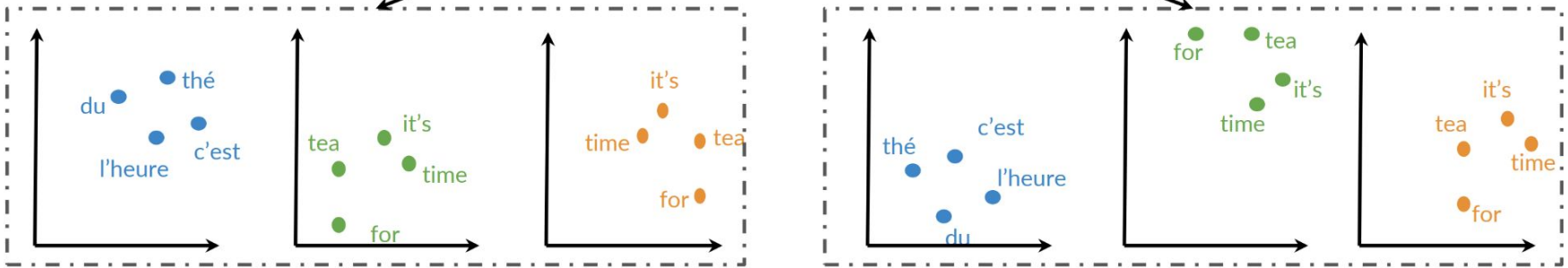


$W_1^Q\ W_1^K\ W_1^V$

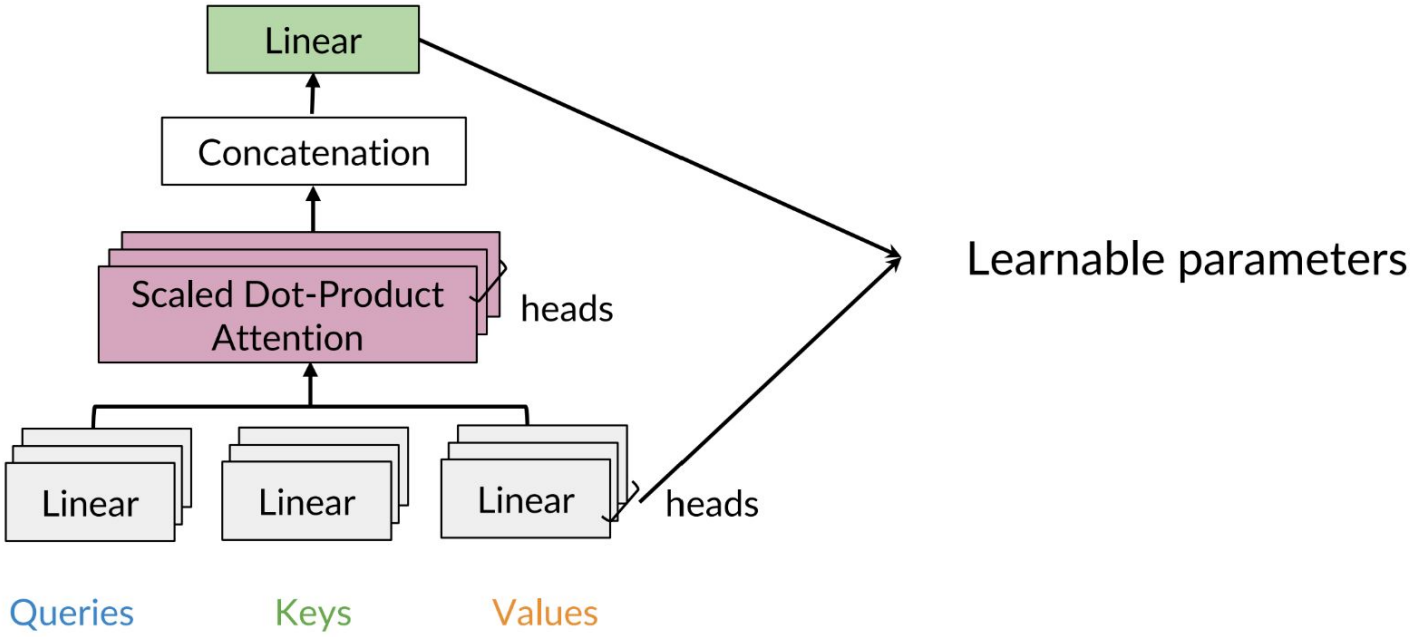
$W_2^Q\ W_2^K\ W_2^V$

Head 1

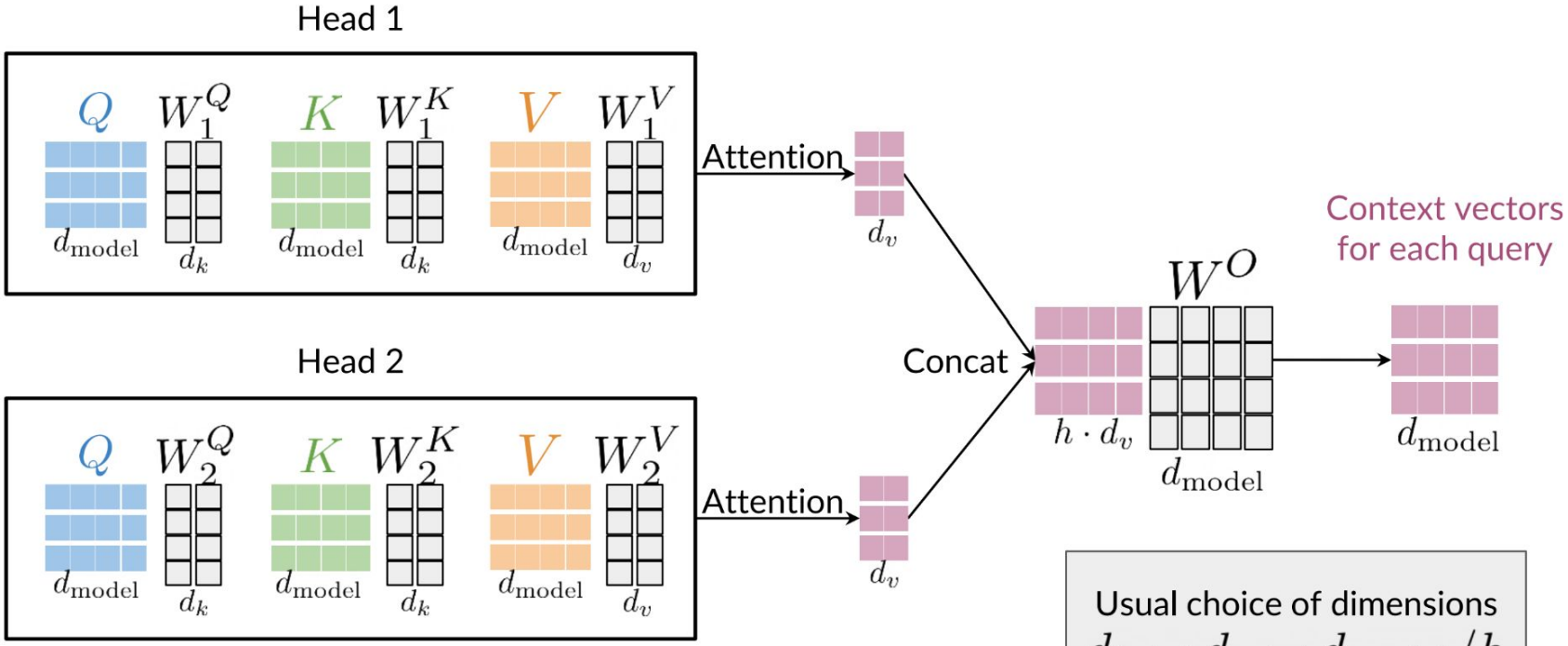
Head 2



Multi-Head Attention - Overview



Multi-Head Attention



d_{model} : Embedding size

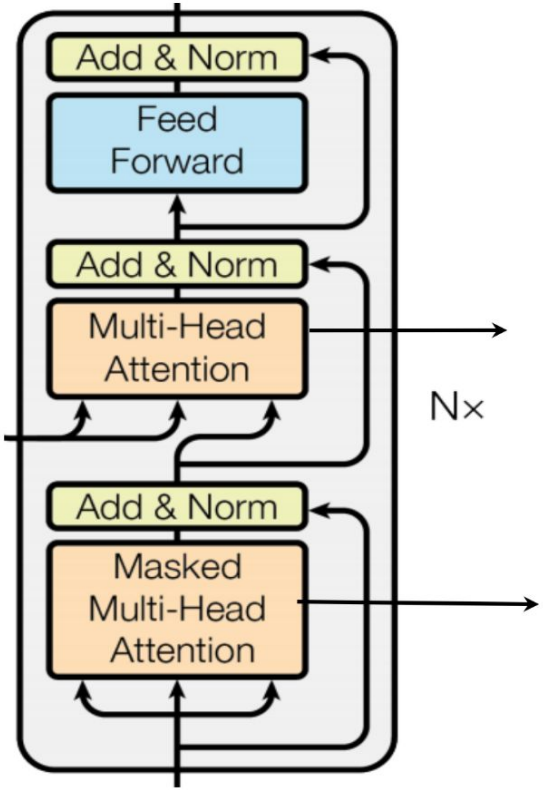
Usual choice of dimensions

$$d_k = d_v = d_{\text{model}}/h$$

Summary

- Multi-Headed models attend to information from different representations
- Parallel computations
- Similar computational cost to single-head attention

The Decoder

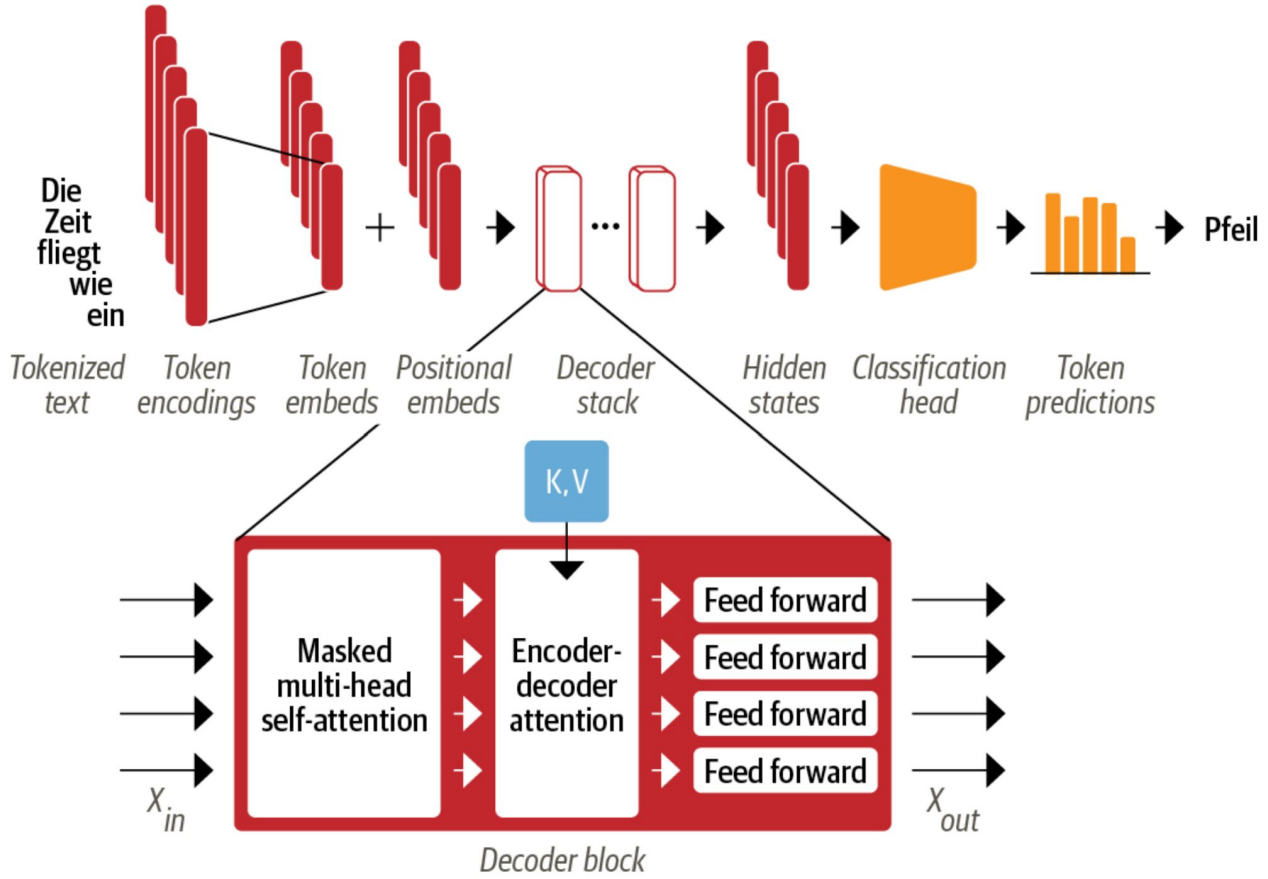


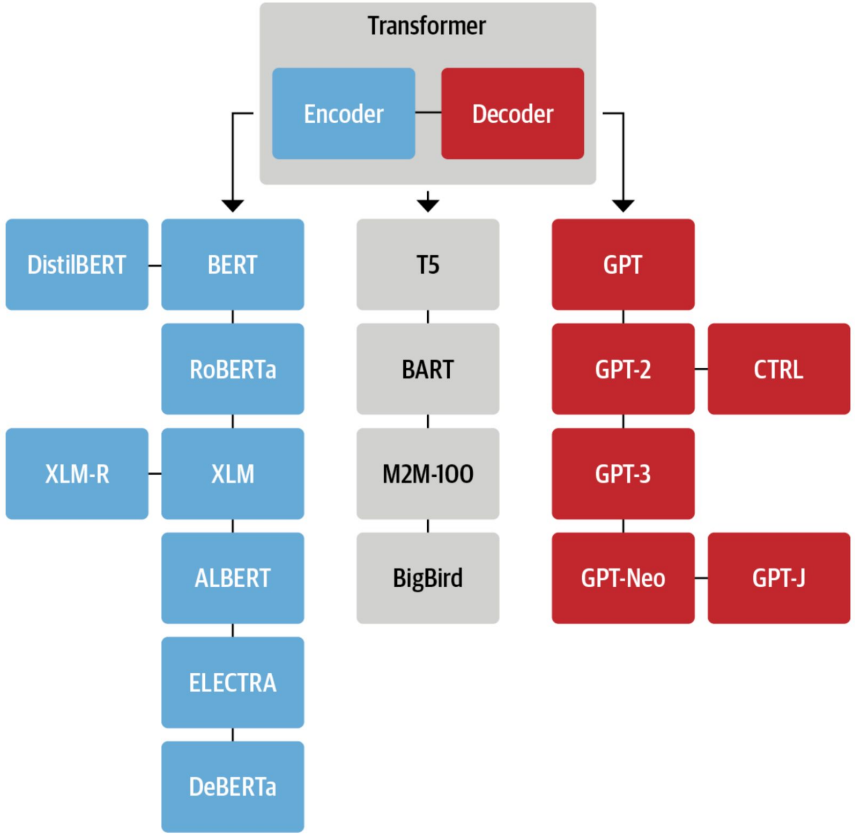
Encoder-Decoder Attention

Every position from the decoder attents to the outputs from the encoder

Masked Self-Attention

Every position attends to **previous** positions

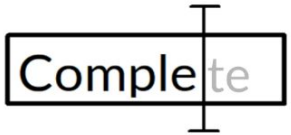

Die
Zeit
fliegt
wie
ein
Pfeil
Tokenized text
Token encodings
Token embeds
Positional embeds
Decoder stack
Hidden states
Classification head
Token predictions
K, V
Masked multi-head self-attention
Encoder-decoder attention
Feed forward
Feed forward
Feed forward
Feed forward
X in
X out
Decoder block


Transformer
Encoder
Decoder
DistilBERT
BERT
RoBERTa
XLM-R
XLM
ALBERT
ELECTRA
DeBERTa
T5
BART
M2M-100
BigBird
GPT
GPT-2
CTRL
GPT-3
GPT-Neo
GPT-J

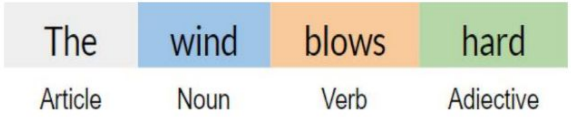
Transformer Applications

Transformer NLP applications

Text summarization



Auto-Complete



Named entity recognition (NER)

Question answering (Q&A)

Translation

Chat-bots

Other NLP tasks

- Sentiment Analysis
- Market Intelligence
- Text Classification
- Character Recognition
- Spell Checking

State of the Art Transformers

Radford, A., et al. (2018) Open AI	GPT-2: Generative Pre-training for Transformer
Devlin, J., et al. (2018) Google AI Language	BERT:Bidirectional Encoder Representations from Transformers
Colin, R., et al. (2019) Google	T5: Text-to-text transfer transformer