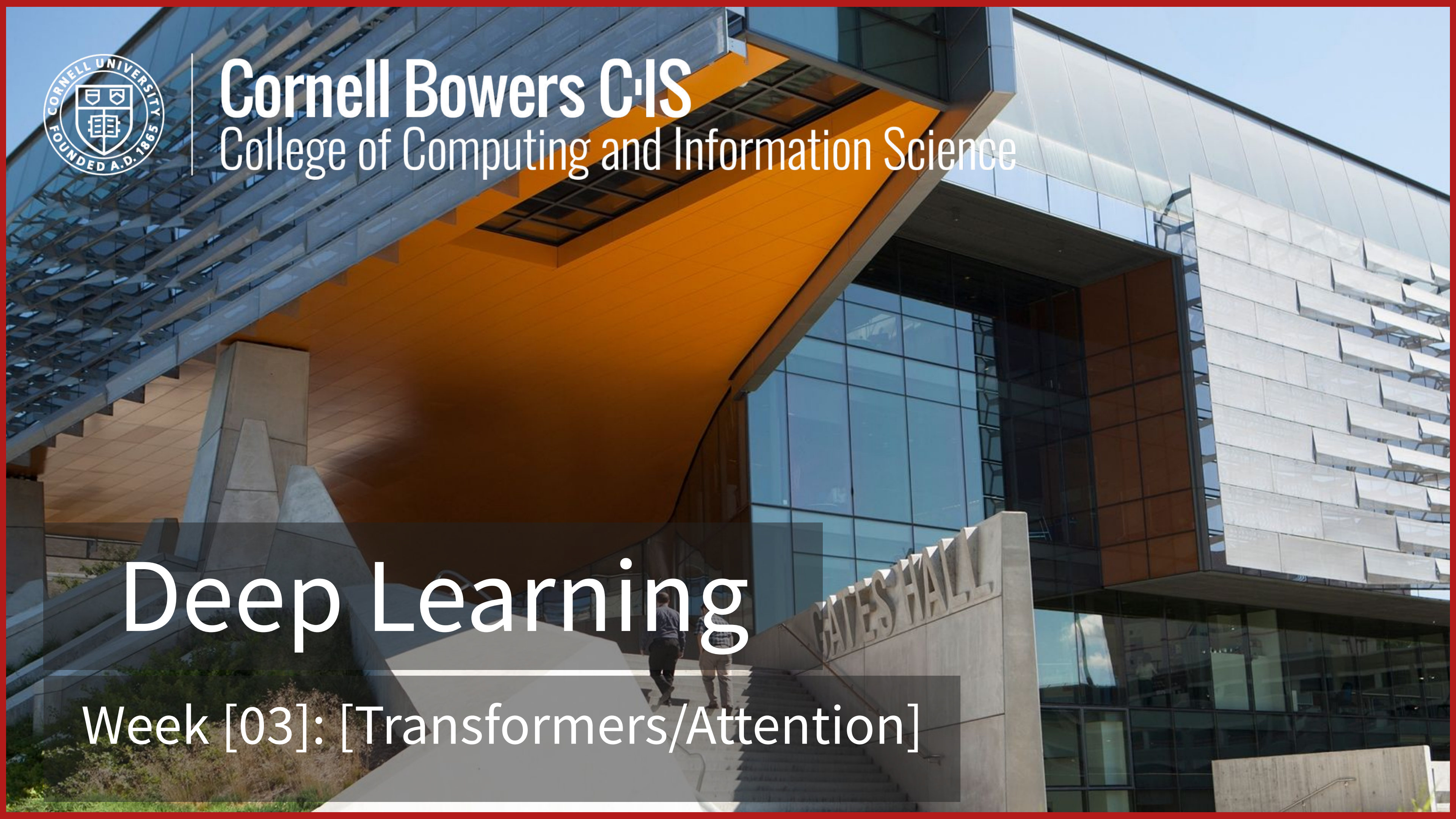


Cornell Bowers C·IS
College of Computing and Information Science

Deep Learning

Week [03]: [Transformers/Attention]

Cornell Bowers C·IS

Thanks to

Varsha Kishore
Justin Lovelace
Anissa Dallmann
Elizabeth Kelmenson

Cornell Bowers C·IS

LSTMs

- Add a cell state to store information
  - Gradient flows along the cell state
- Update cell state with parameterized gating functions
- Performs better with long sequences

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

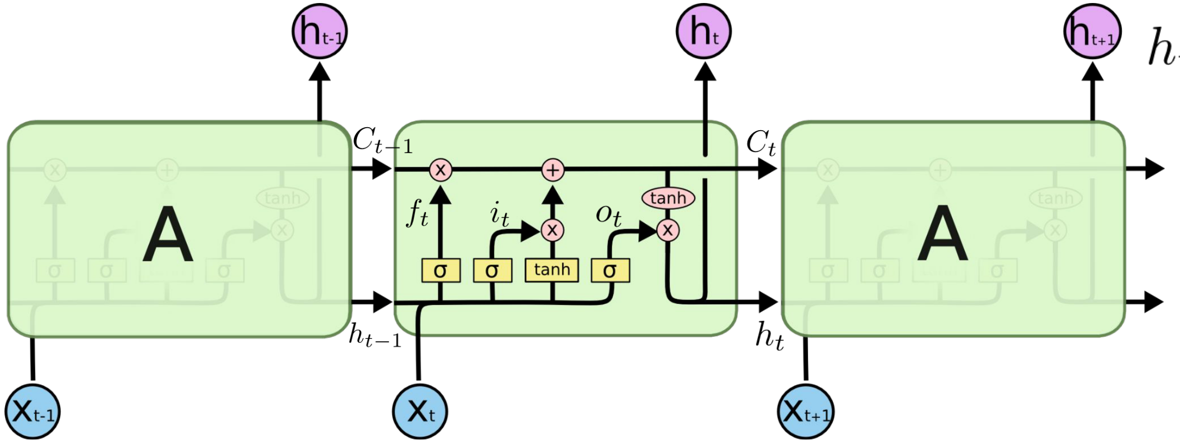
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

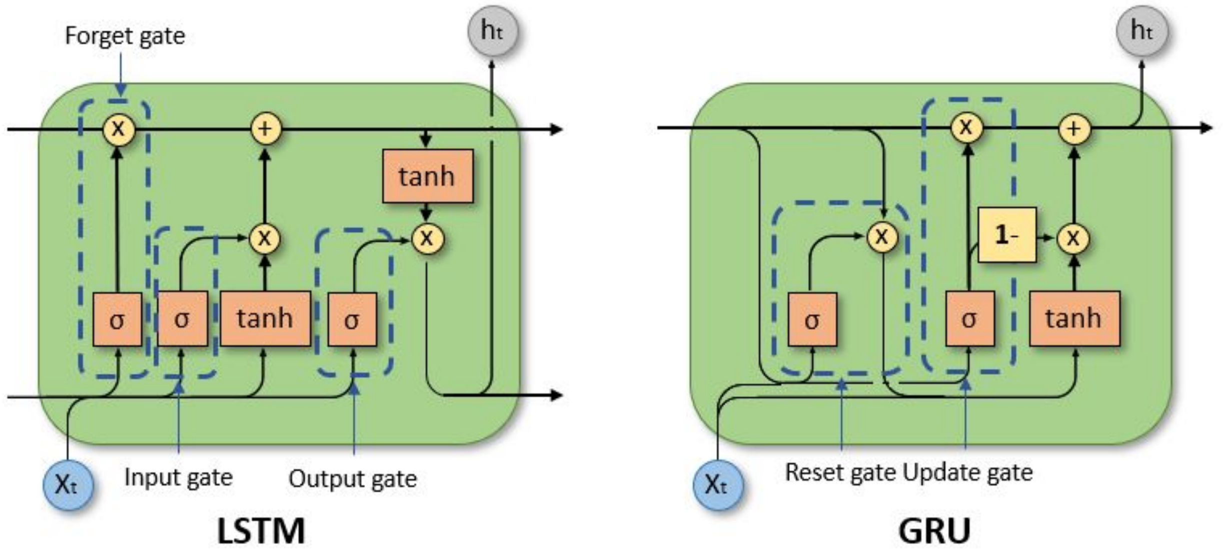
$$h_t = o_t * \tanh\left(C_t\right)$$



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

Cornell Bowers C·IS

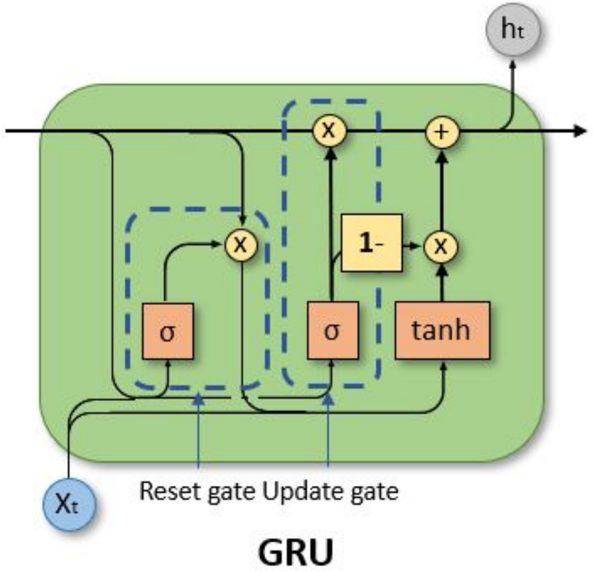
Gated recurrent units (GRUs)



https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Cornell Bowers C·IS

Gated recurrent units (GRUs)



$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$\hat{h}_t = \phi(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = (1 - z_t) \odot \hat{h}_t + z_t \odot h_{t-1}$$

Cornell Bowers C·IS

Logistics

- HW2 is out, due 03/06
- HW3 is out 03/06, due 03/13
- Project 2 out on Tuesday
- Tuesday 10am office hours might change
- We will talking about projects on Thursday
    - HW3 will be a shorter

Cornell Bowers C·IS

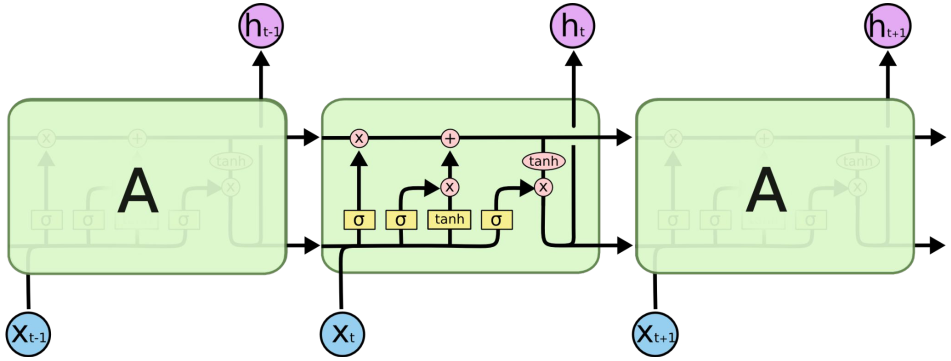
What you will learn today

- Issues with LSTMs
  - Meaning of words flows both ways (fixable)
  - Bottleneck layers (fixable)
  - LSTMs are slow (not fixable)
- Attention:
  - Scaled Inner-product attention
  - Self-Attention
  - Masked self-attention
  - Multi-head attention
  - Cross-attention
- New Concepts:
  - Positional Embedding
  - Layer Norm
- New Algorithms:
  - ELMO
  - Transformer
  - BERT

Cornell Bowers C·IS

Previously: Using LSTMs to solve sequence problems

- Process sequences one element at a time.
- Maintain a 'memory' (cell state) to capture information about previous steps.
- Mitigates the RNN vanishing gradient problem
- Suitable for time series, speech, text, and other sequential data.



Cornell Bowers C·IS

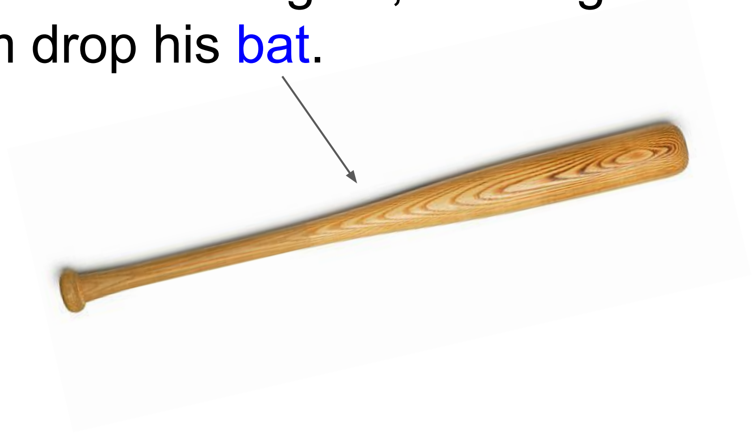
Issue #1

Context influences words in both directions

Cornell Bowers C·IS

Context influences word meaning.

A bat flew out of the dugout, startling the baseball player and making him drop his bat.



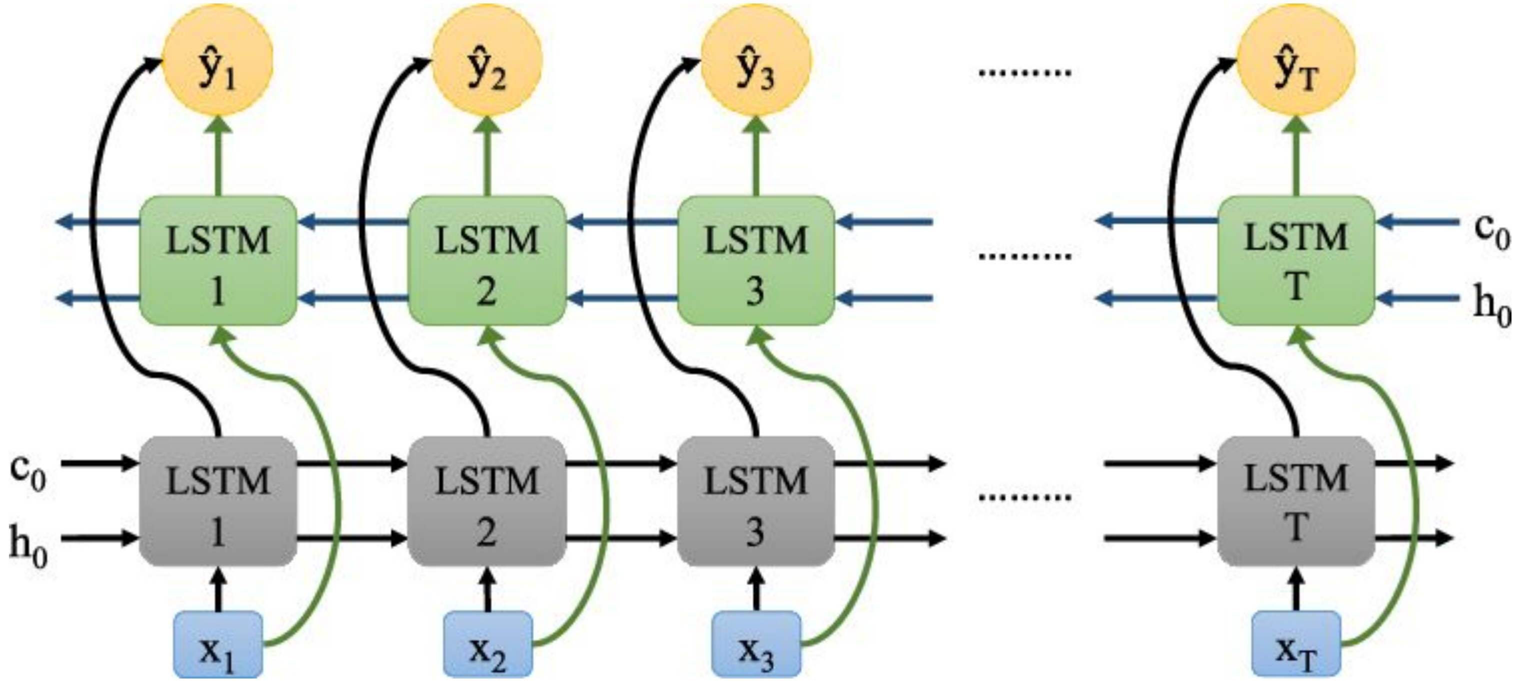
Context influences word meaning.

Each word pays attention to the words around it, focusing more on the words which provide important information about its meaning.

A bat flew out of the dugout, startling the baseball player and making him drop his bat.

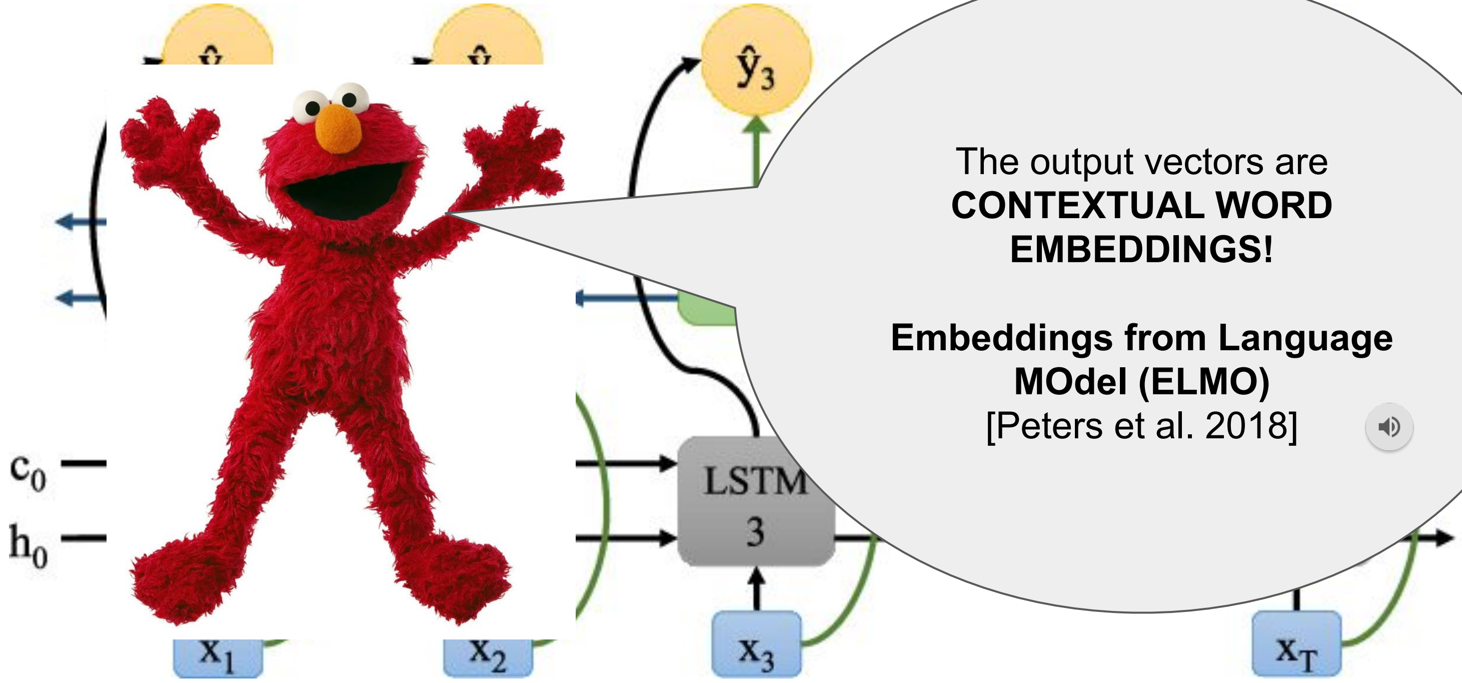
Cornell Bowers C·IS

Bidirectional LSTM



Cornell Bowers C·IS

Bidirectional LSTM



The output vectors are
CONTEXTUAL WORD EMBEDDINGS!

Embeddings from Language MOdel (ELMO)
[Peters et al. 2018]

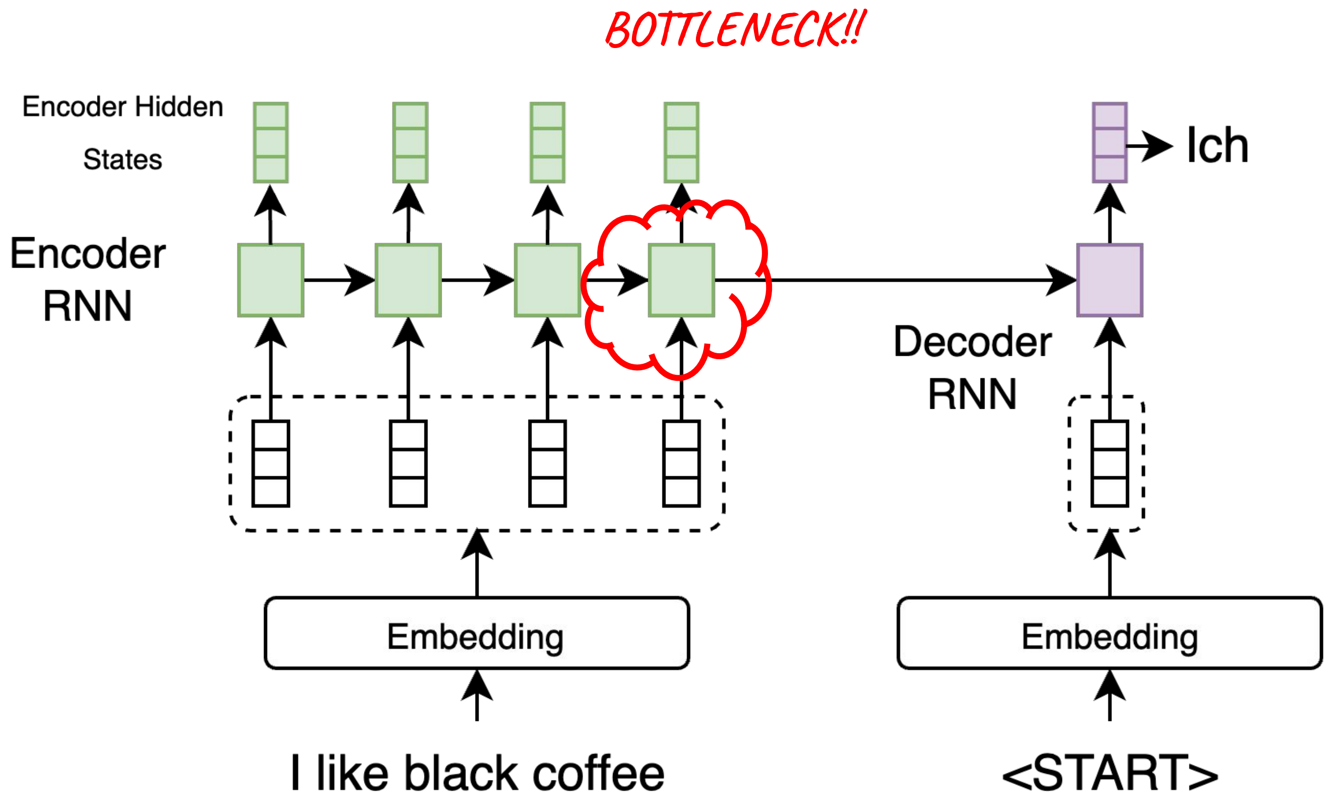
Cornell Bowers C·IS

Issue #2

The Bottleneck!

Cornell Bowers C·IS

Bottleneck Problem

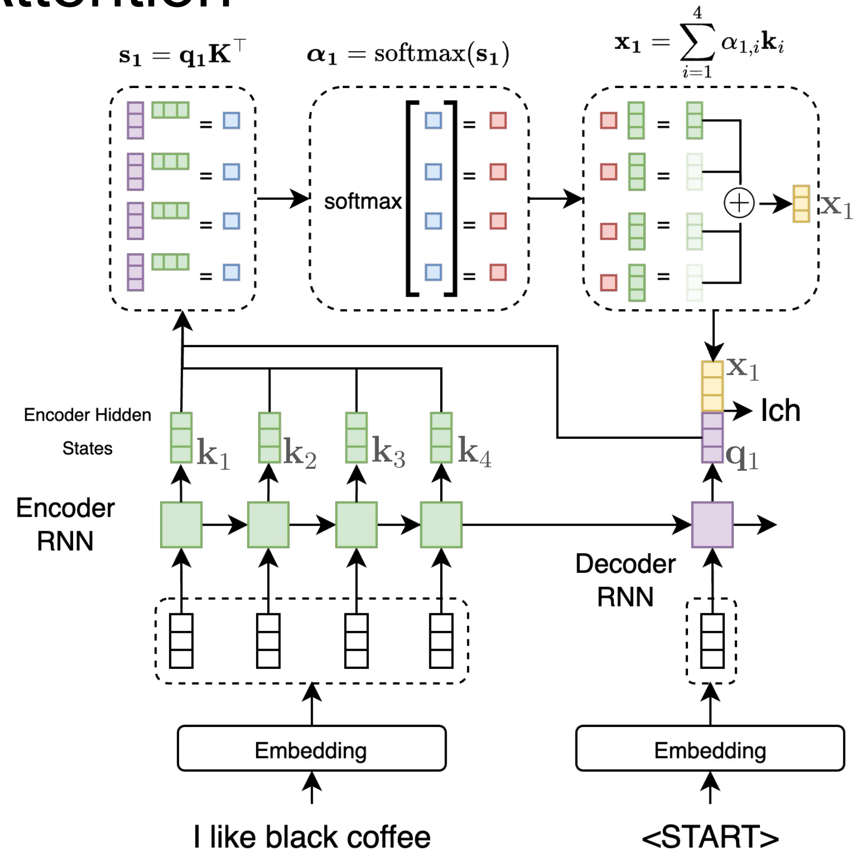


$\mathbf{k_1}$

Cornell Bowers C·IS

RNN with Attention

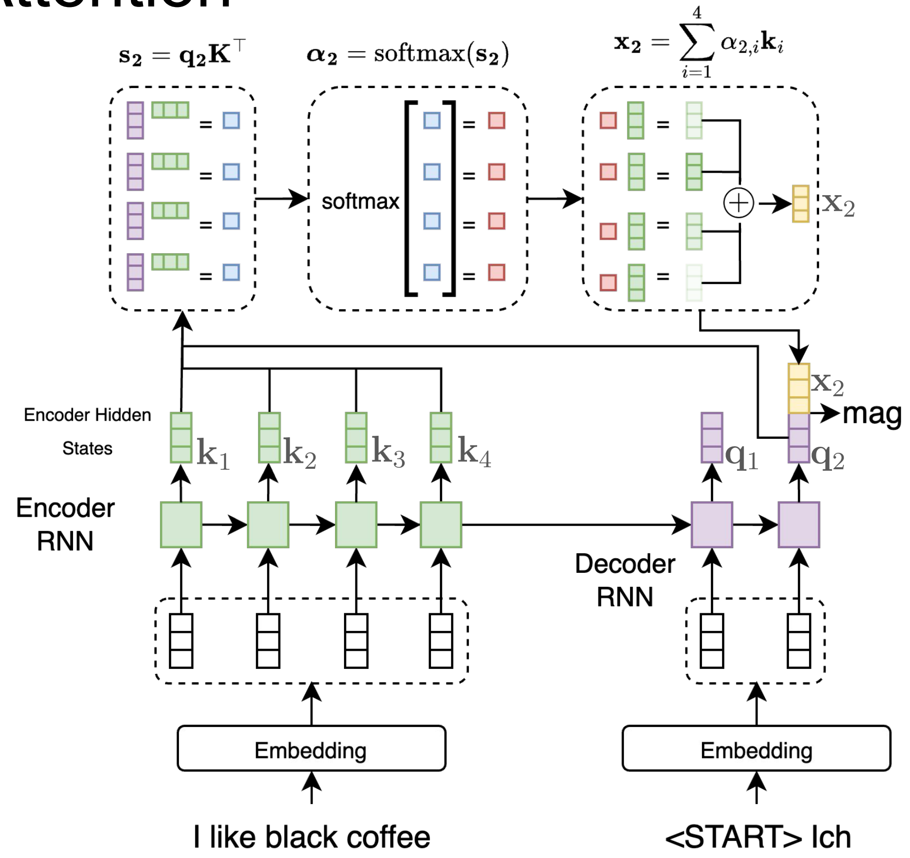
[Bahdanau et al. 2014]



Cornell Bowers C·IS

RNN with Attention

[Bahdanau et al. 2014]



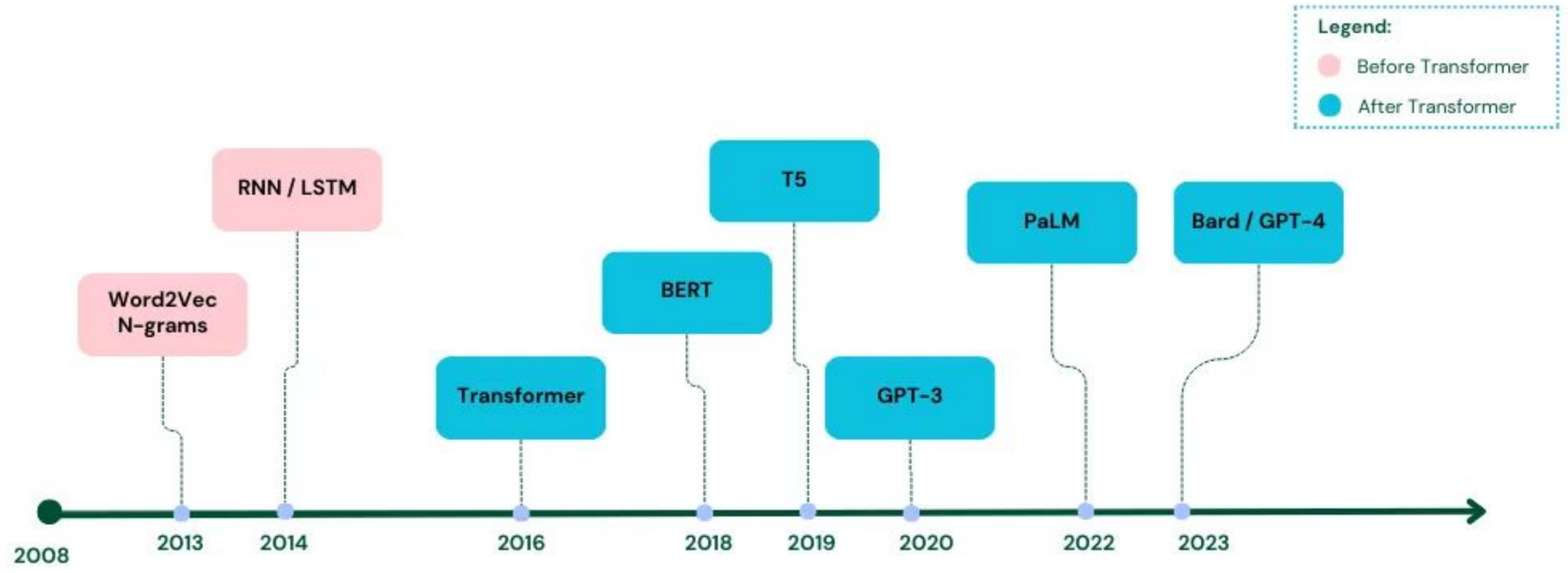
Cornell Bowers C·IS

Issue #3

Recurrent Neural Networks (including LSTMs) are slow on modern computers (with GPUs). Why? - Discuss

Cornell Bowers C·IS

Language Modelling History

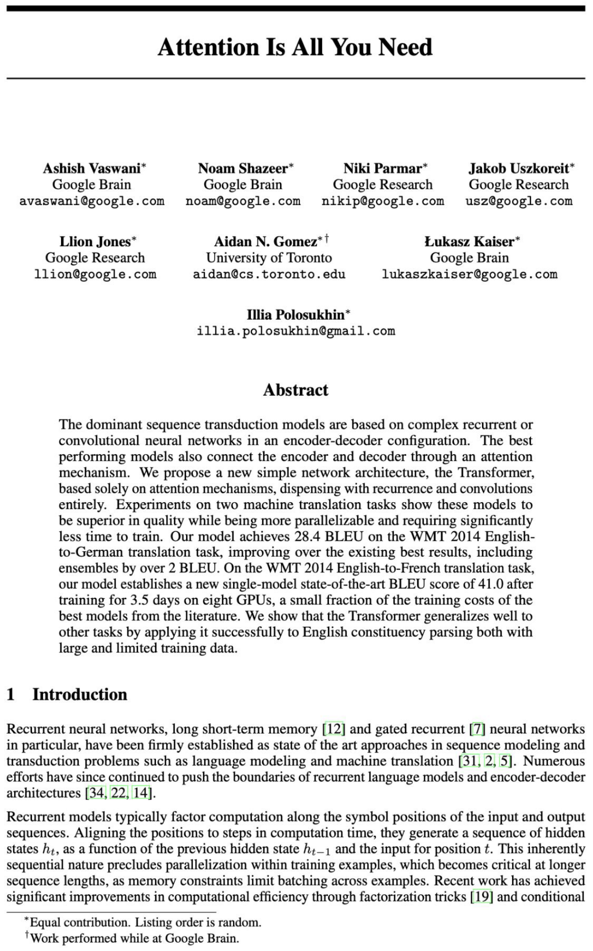


https://medium.com/@kirudang/language-model-history-before-and-after-transformer-the-ai-revolution-bedc7948a130

Cornell Bowers C·IS

TRANSFORMERS [Vaswani et al. 2017]

- **Parallel** alternative to LSTMs
- Far more **efficient** on GPUs
- **Removes recurrence**
- **Keeps attention**
- Currently dominant approach for most sequence tasks
  - E.g. Machine translation, summarizatior question answering, etc.



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 in an encoder-decoder configuration. 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 41.0 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.

1 Introduction

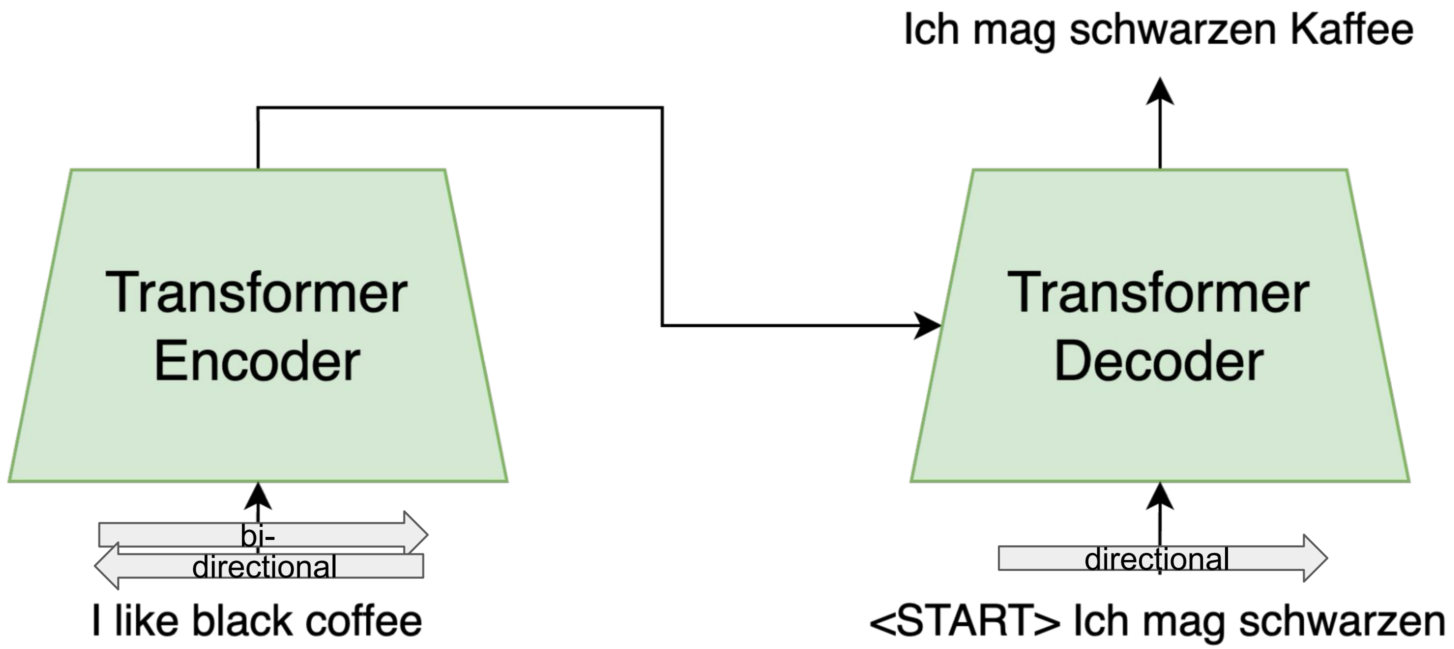
Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [31, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [34, 22, 14].

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [19] and conditional

[*]Equal contribution. Listing order is random.
[†]Work performed while at Google Brain.

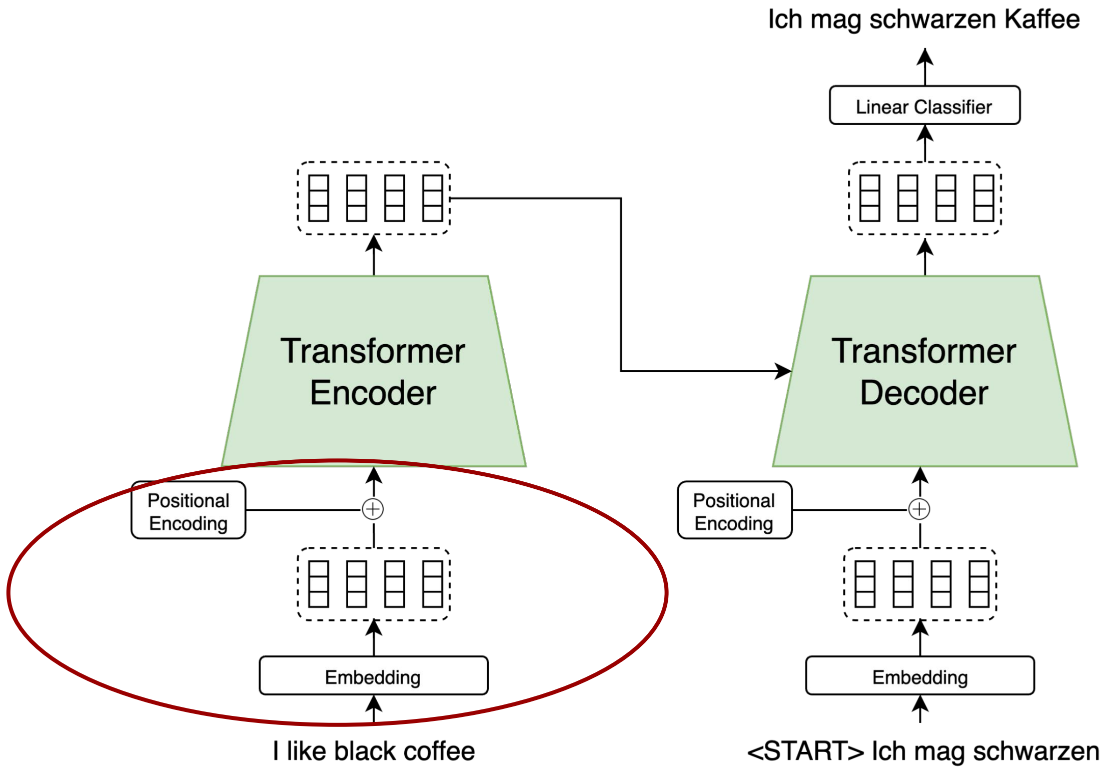
Cornell Bowers C·IS

Transformer Architecture



Cornell Bowers C·IS

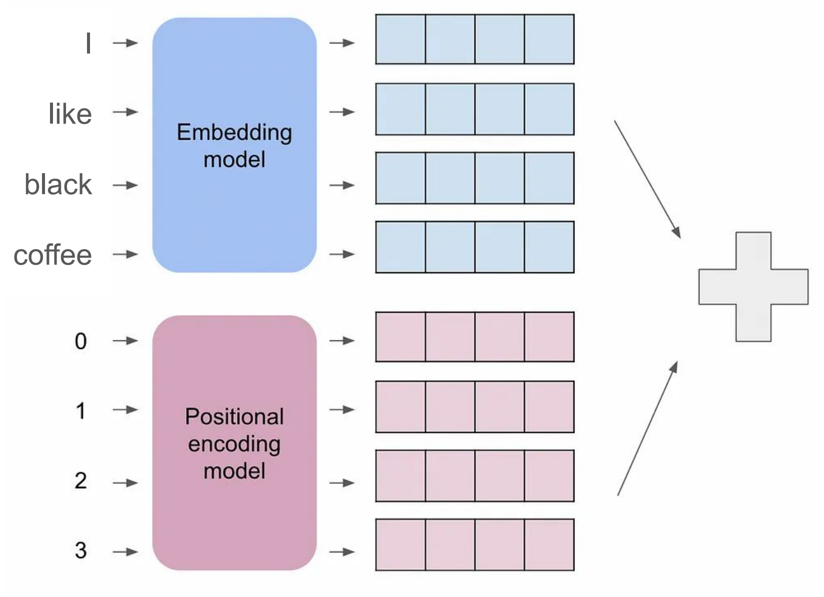
Transformer Architecture



Cornell Bowers C·IS

Positional Encoding

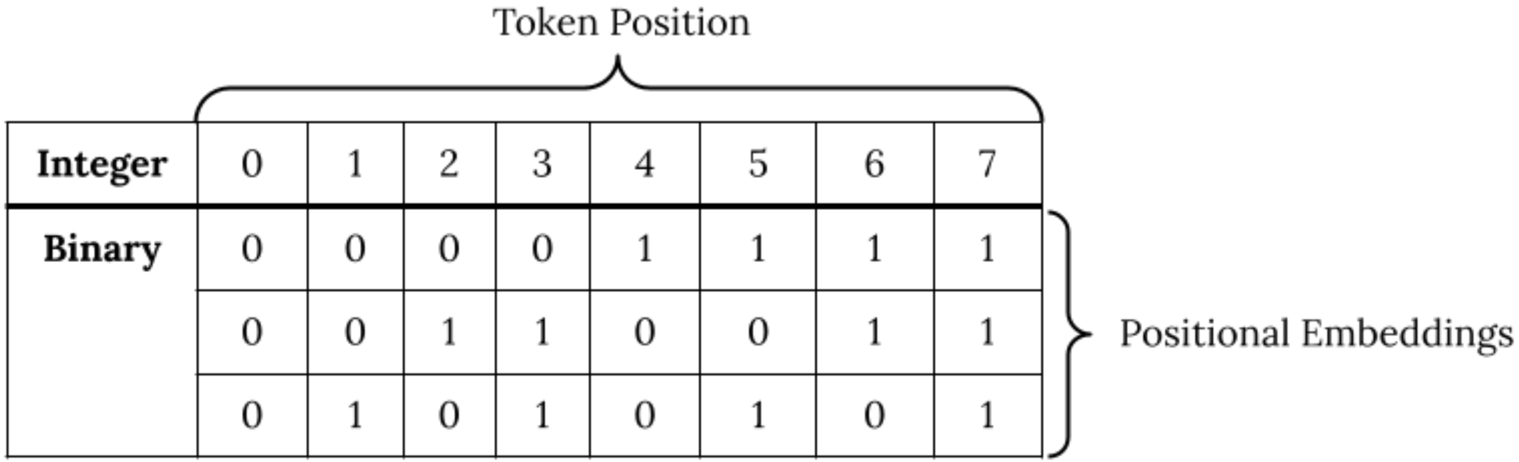
- Embedding model:
  - Lookup table
  - Non-contextual word embeddings
- Problem:
  - Word order matters!
  - In attention word order is lost
- Solution:
  - Encode the **token position** as a vector
  - Add this vector to the **word embedding**



[https://medium.com/@xuer.chen.human/llm-study-notes-positional-encoding-0639a1002ec0]

Positional Encoding

Neural networks are not good with single dimensional values.
Higher dimensions are better suited for hyper-plane based classifiers (internal to the neural network).
Naive Idea: Use binary encoding



Token Position

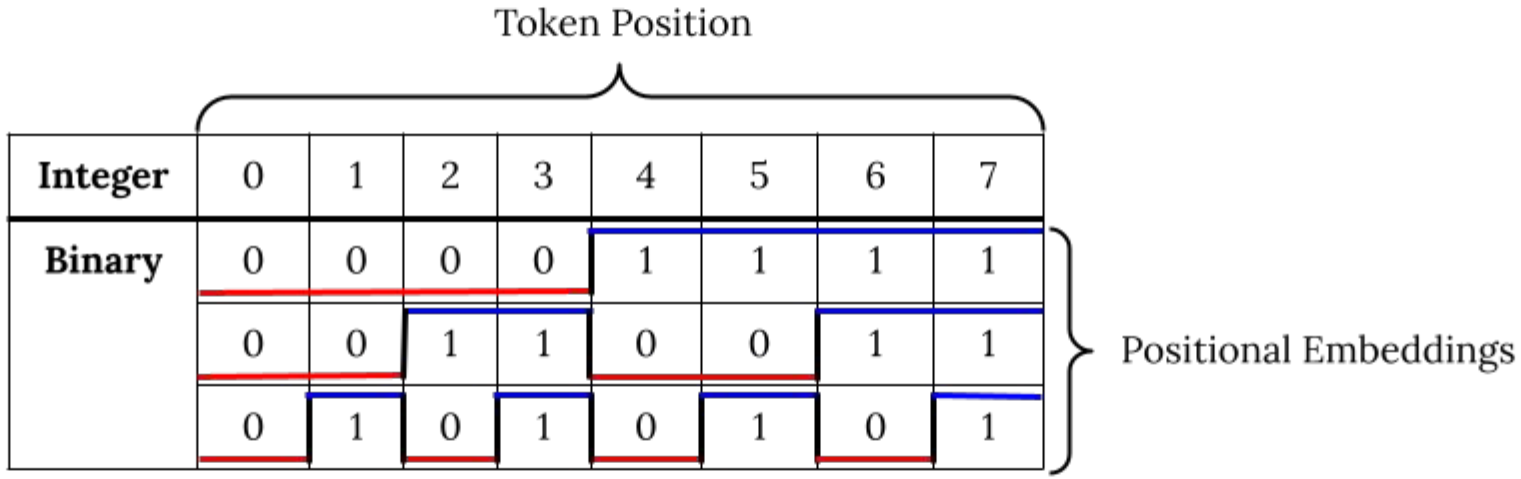
Integer	0	1	2	3	4	5	6	7
Binary	0	0	0	0	1	1	1	1
	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1

Positional Embeddings

Cornell Bowers C·IS

Positional Encoding

Neural networks are not good with single dimensional values.
Higher dimensions are better suited for hyper-plane based classifiers (internal to the neural network).
Naive Idea: Use binary encoding



Integer	0	1	2	3	4	5	6	7
Binary	0	0	0	0	1	1	1	1
	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1

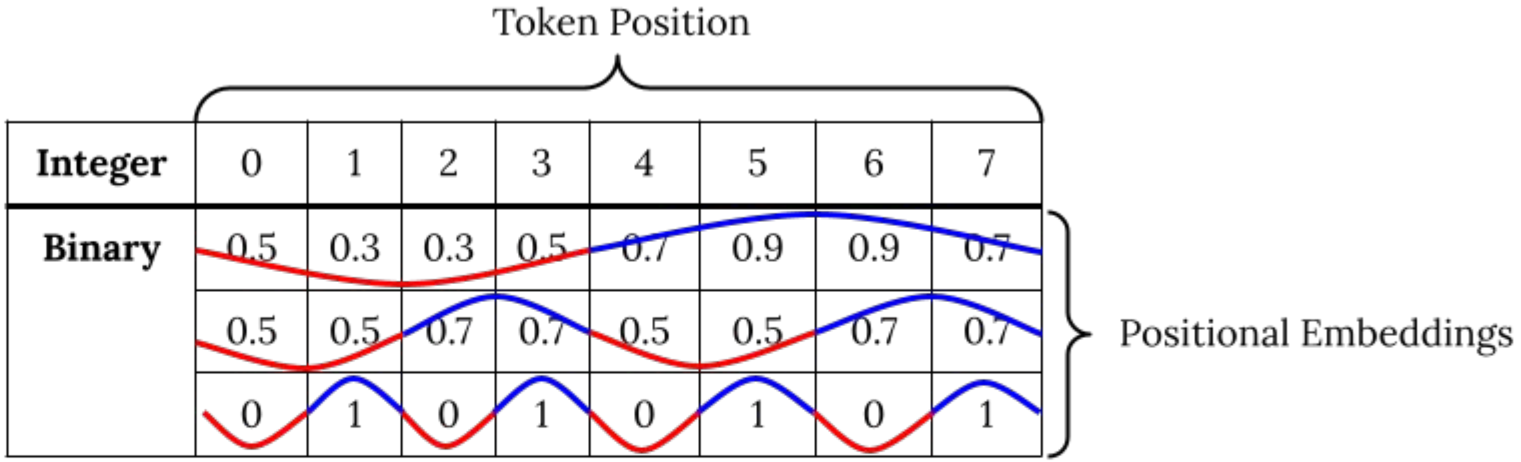
Cornell Bowers C·IS

Positional Encoding

Better idea: Use cosine / sine embeddings (Fourier features.)

For a position pos and embedding dimension i:

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



Token Position

Integer	0	1	2	3	4	5	6	7
Binary	0.5	0.3	0.3	0.5	0.7	0.9	0.9	0.7
	0.5	0.5	0.7	0.7	0.5	0.5	0.7	0.7
	0	1	0	1	0	1	0	1

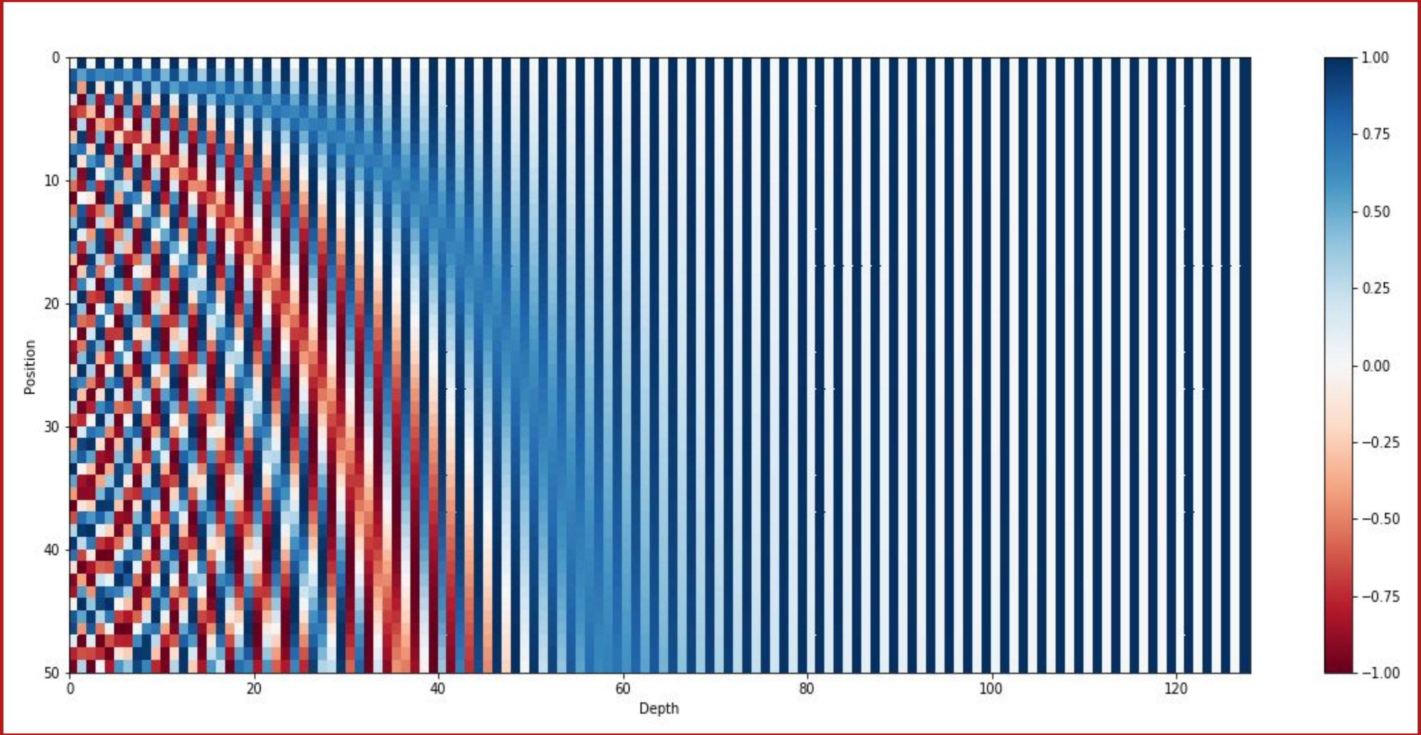
Positional Embeddings

Cornell Bowers C·IS

Example Positional Encoding

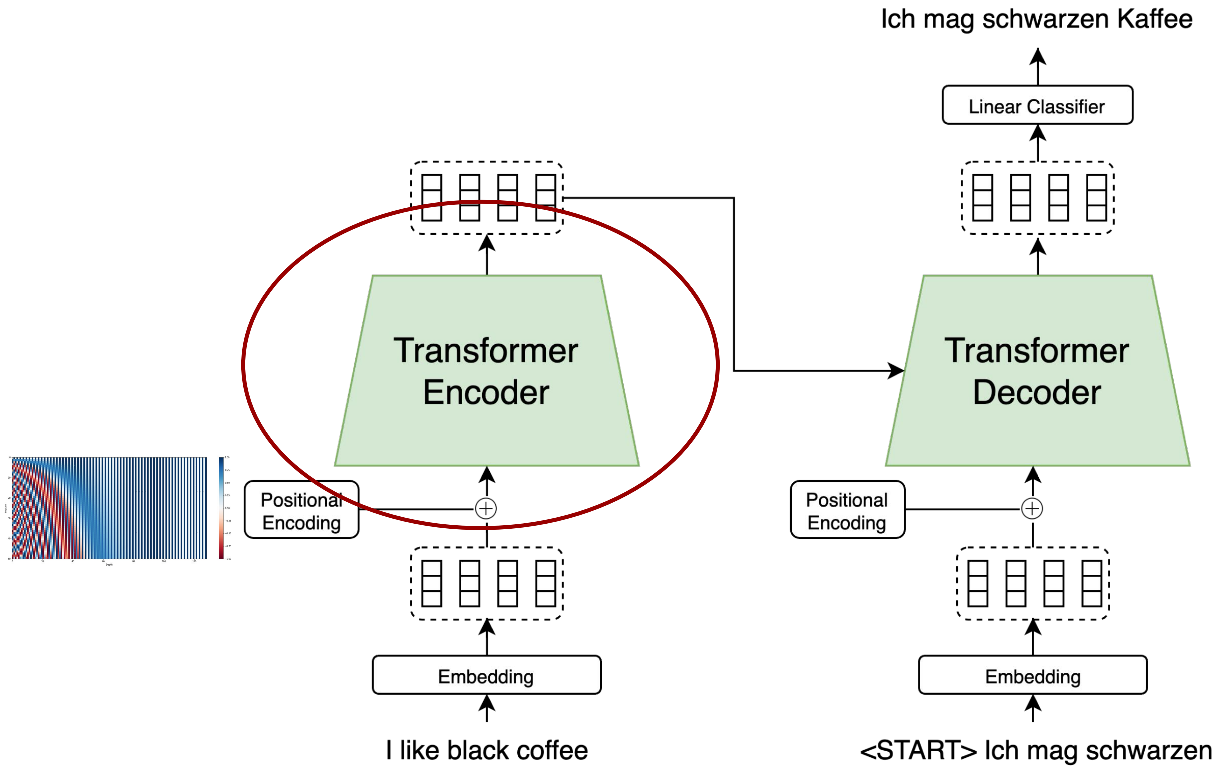
For a position pos and embedding dimension i:

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



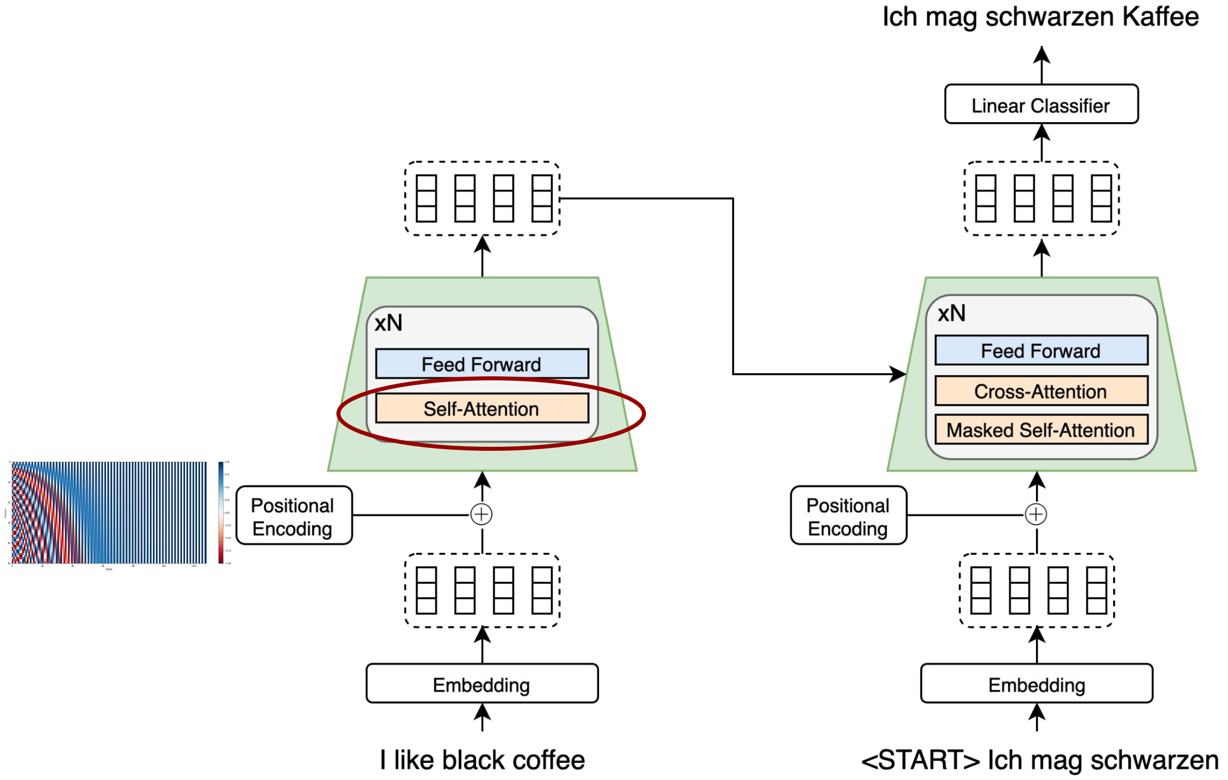
Cornell Bowers C·IS

Transformer Architecture



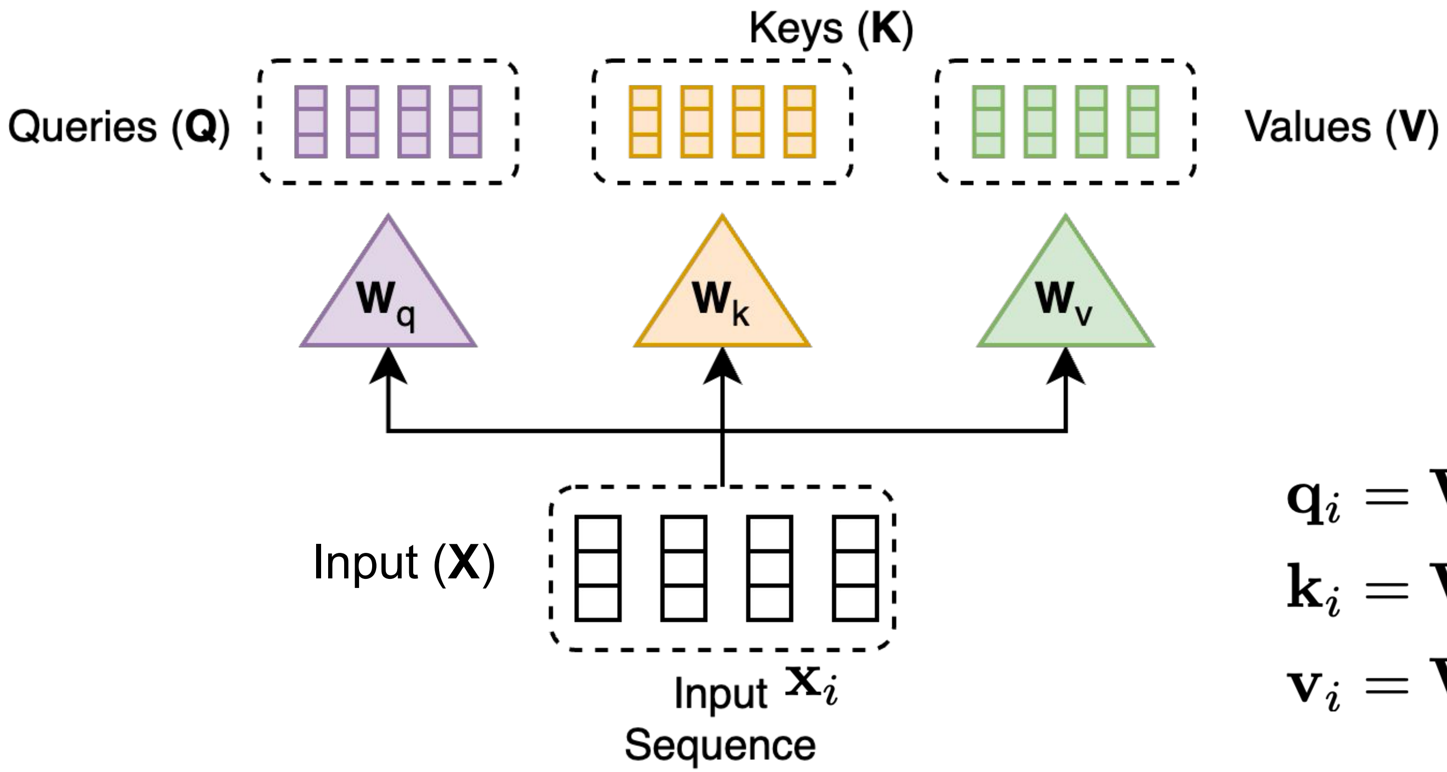
Cornell Bowers C·IS

Transformer Architecture



Cornell Bowers C·IS

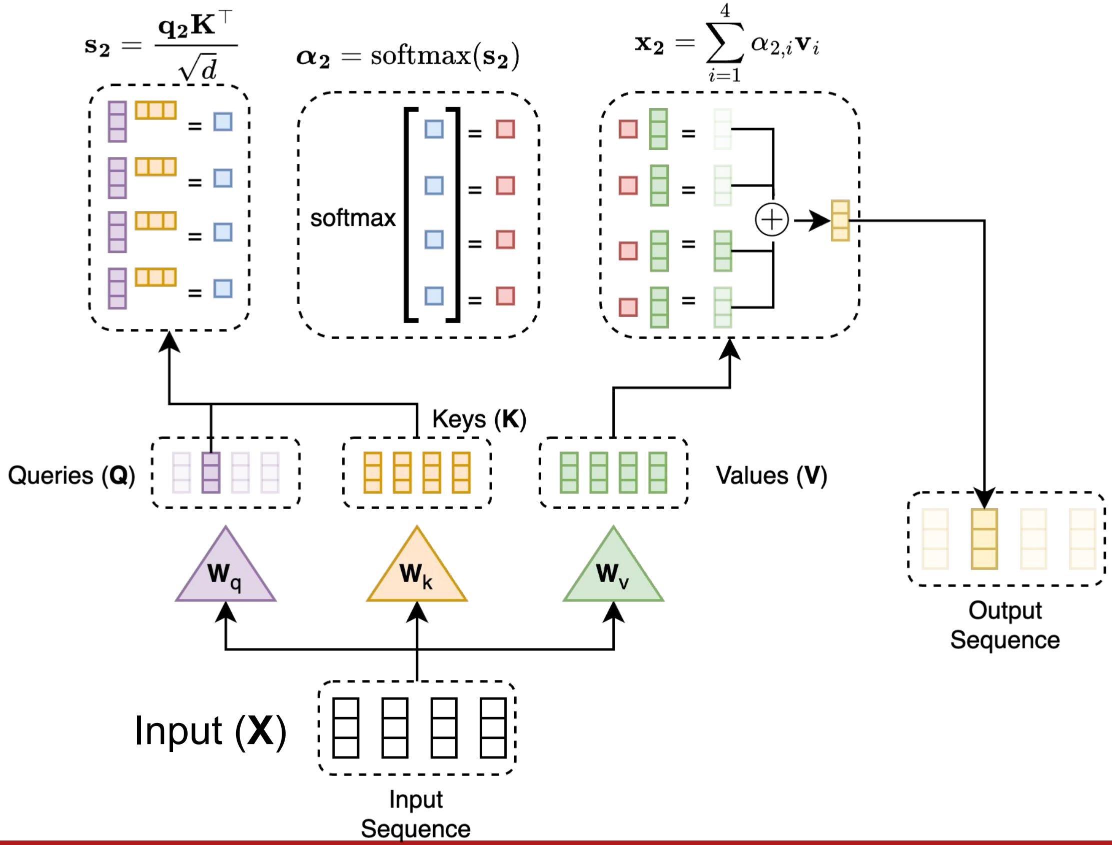
Self-Attention



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

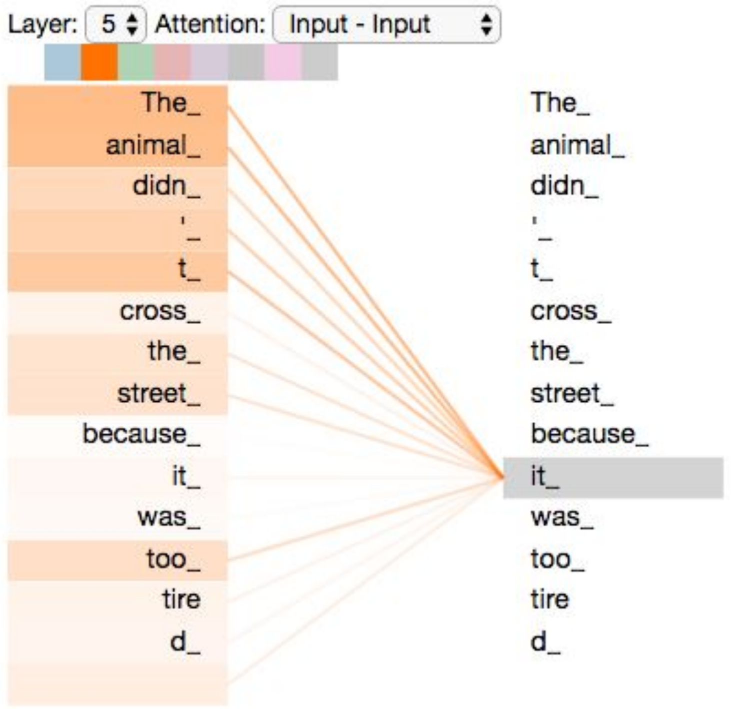
Cornell Bowers C·IS

Self-Attention


$\mathbf{s_2} = \frac{\mathbf{q_2}\mathbf{K}^\top}{\sqrt{d}}$
$\boldsymbol{\alpha_2} = \mathrm{softmax}(\mathbf{s_2})$
$\mathbf{x_2} = \sum_{i=1}^{4} \alpha_{2,i}\mathbf{v}_i$
softmax
=
Keys (**K**)
Queries (**Q**)
Values (**V**)
W_q
W_k
W_v
Output Sequence
Input (**X**)
Input Sequence

Cornell Bowers C·IS

Self-attention demo



Cornell Bowers C·IS

Discuss:

- Q, K, V are all (n x d) matrices. Consider have an input of shape b x n x d.

- What is the shape of QK^T?
  - What does this matrix represent?
- What is the shape of the final output?
  - What does this matrix represent?

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

Cornell Bowers C·IS

Multi-Head Attention

What if I want to pay attention to different things at the same time!?

Content-based	This is my big red dog, Clifford.
Description-based	This is my big red dog, Clifford.
Reference-based	This is my big red dog, Clifford.

What’s useful depends on the task. How do I pick what to do?

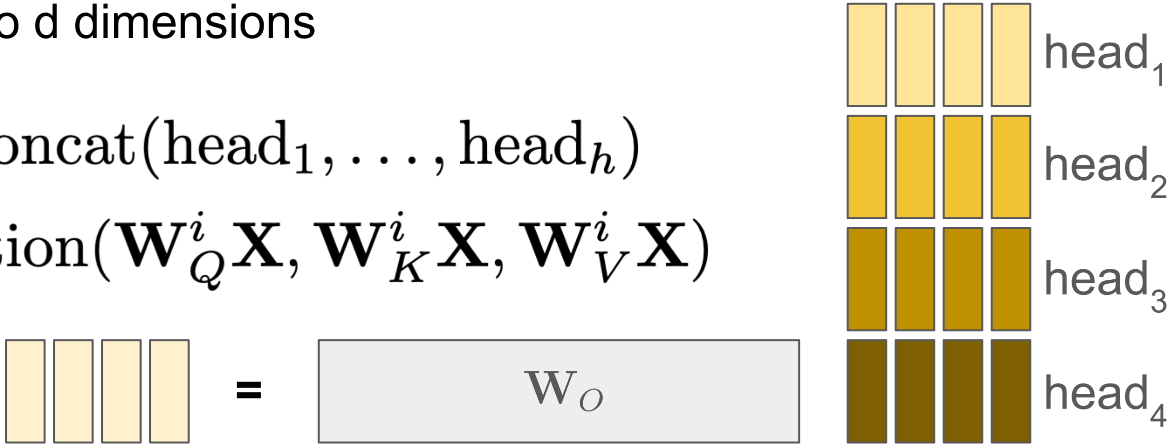
Cornell Bowers C·IS

Multi-Head Attention

- The Scaled Dot-Product Attention attends to one or few entries in the input key-value pairs.
- Idea: apply Scaled Dot-Product Attention multiple times on the linearly transformed inputs.
- Concatenate outputs
- Project them down into d dimensions

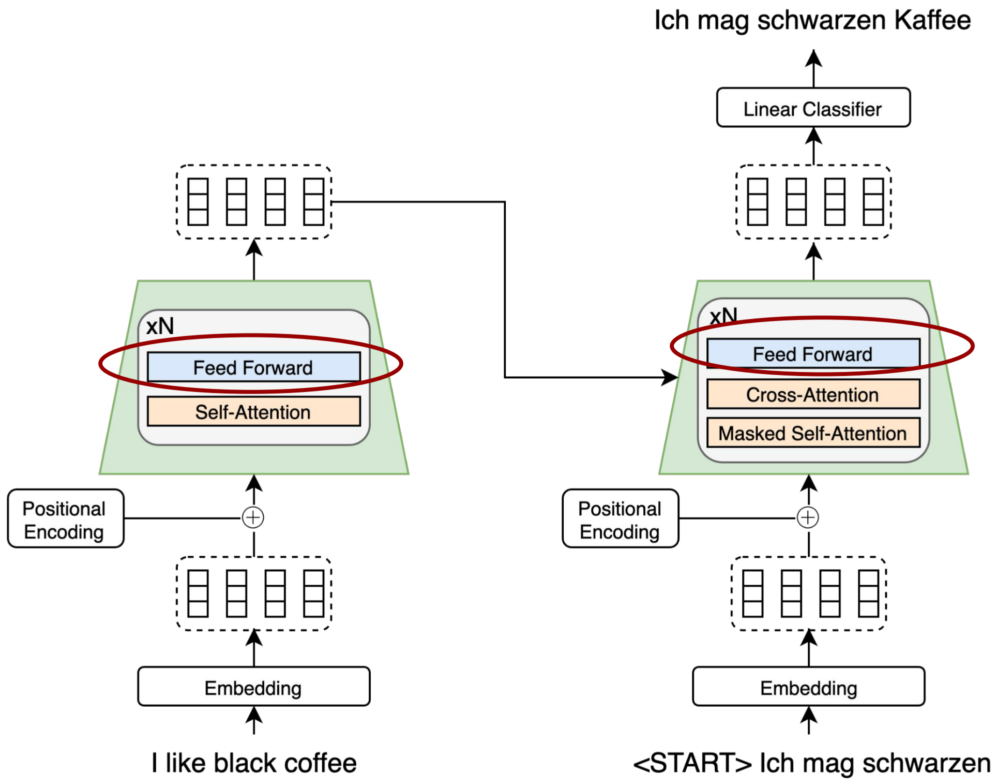
$$\mathrm{MultiHead}(\mathbf{X}) = \mathbf{W}_O \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)$$

$$\text{where } \mathrm{head}_i = \mathrm{Attention}(\mathbf{W}_Q^i \mathbf{X}, \mathbf{W}_K^i \mathbf{X}, \mathbf{W}_V^i \mathbf{X})$$



Cornell Bowers C·IS

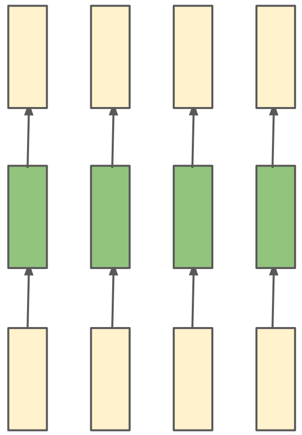
Transformer Architecture



Cornell Bowers C·IS

Point-wise Feed-forward Networks

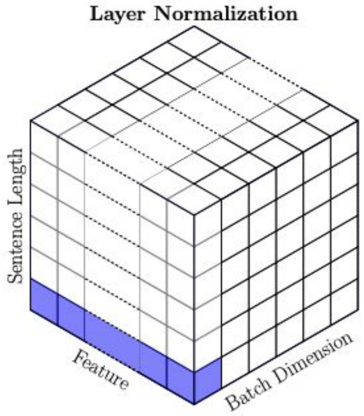
- Apply MLP (aka FFN) to each output vector
    - The **same** MLP to each vector
    - Completely **parallel**
    - $\mathrm{FFN}(\mathbf{x}) = \max(0, \mathbf{W}\mathbf{x} + b)$
    - where **W** and **b** are learned weights and biases



Cornell Bowers C·IS

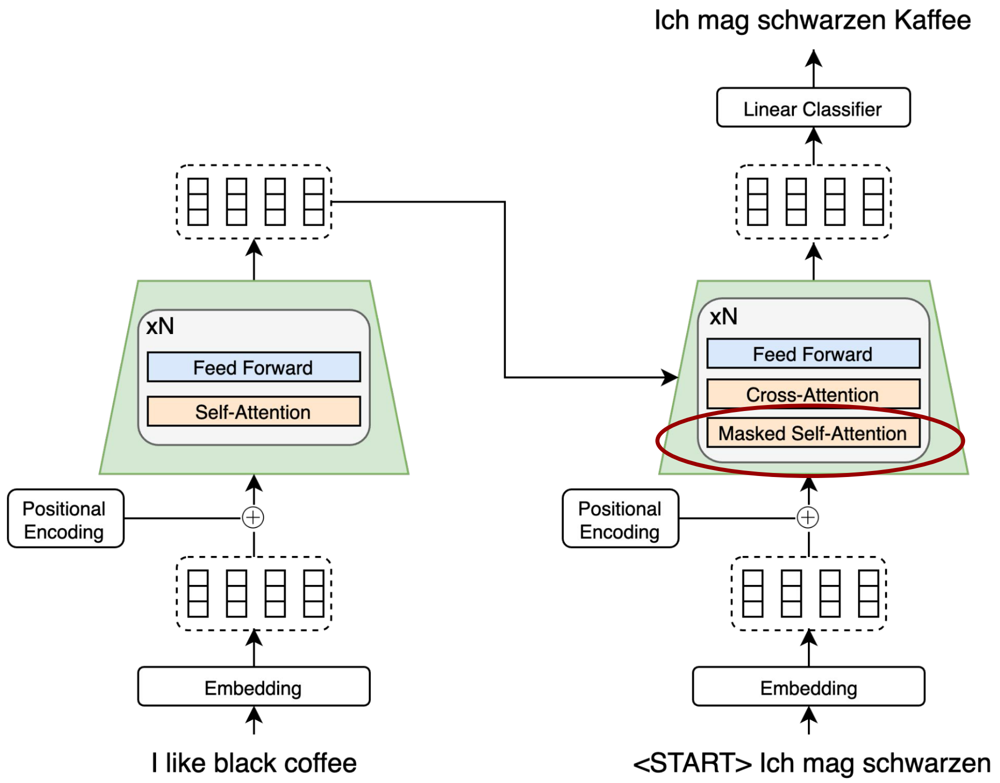
Layernorm

- The FFN and Self-Attention layers are followed by **Layer Normalization**.
- Same as Batch Norm, except:
    - Each word vector is normalized independently



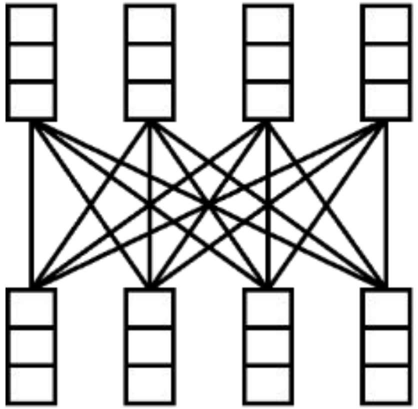
Cornell Bowers C·IS

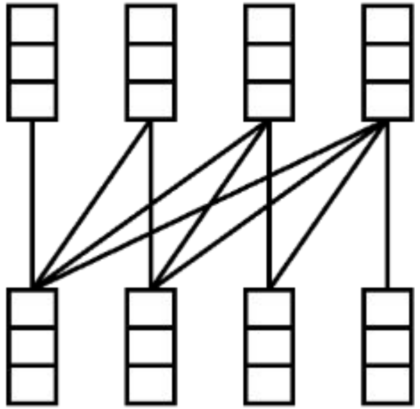
Transformer Architecture



Cornell Bowers C·IS

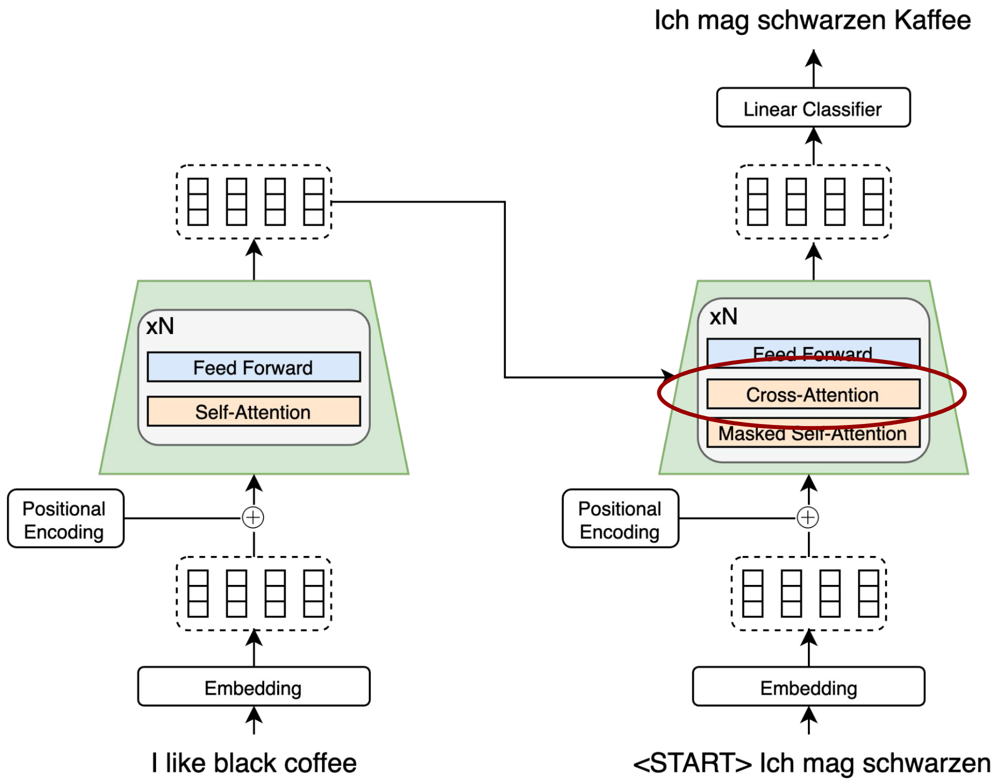
Self-Attention vs. Masked Self-Attention


Self-Attention


Masked Self-Attention

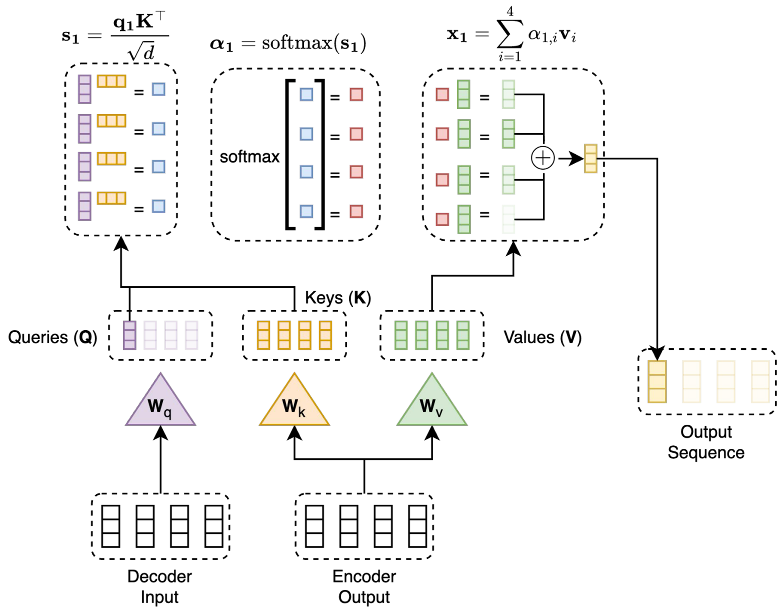
Cornell Bowers C·IS

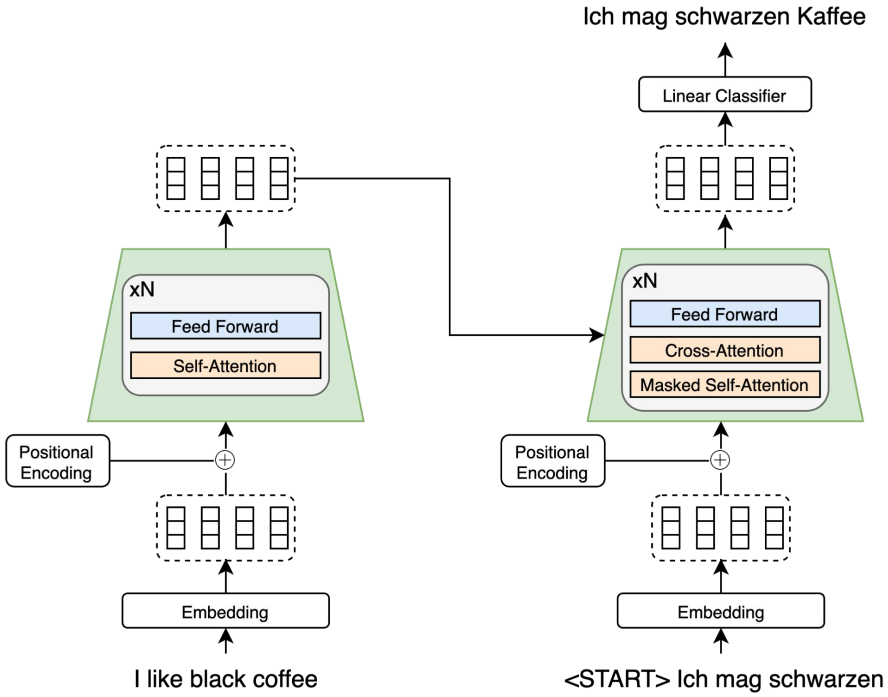
Transformer Architecture



Cornell Bowers C·IS

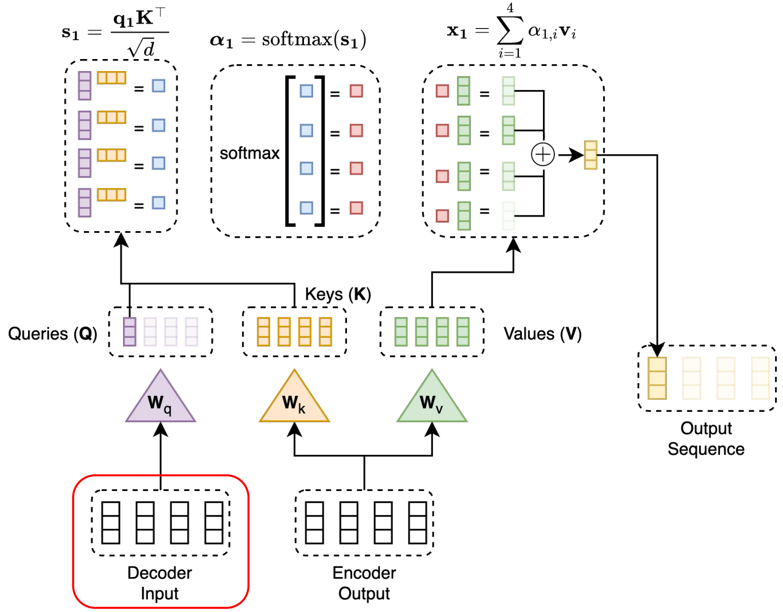
Cross-Attention


$\mathbf{s}_1 = \frac{\mathbf{q}_1 \mathbf{K}^\top}{\sqrt{d}}$
$\boldsymbol{\alpha}_1 = \text{softmax}(\mathbf{s}_1)$
$\mathbf{x}_1 = \sum_{i=1}^{4} \alpha_{1,i} \mathbf{v}_i$
softmax
Keys (K)
Queries (Q)
Values (V)
W_q
W_k
W_v
Output Sequence
Decoder Input
Encoder Output

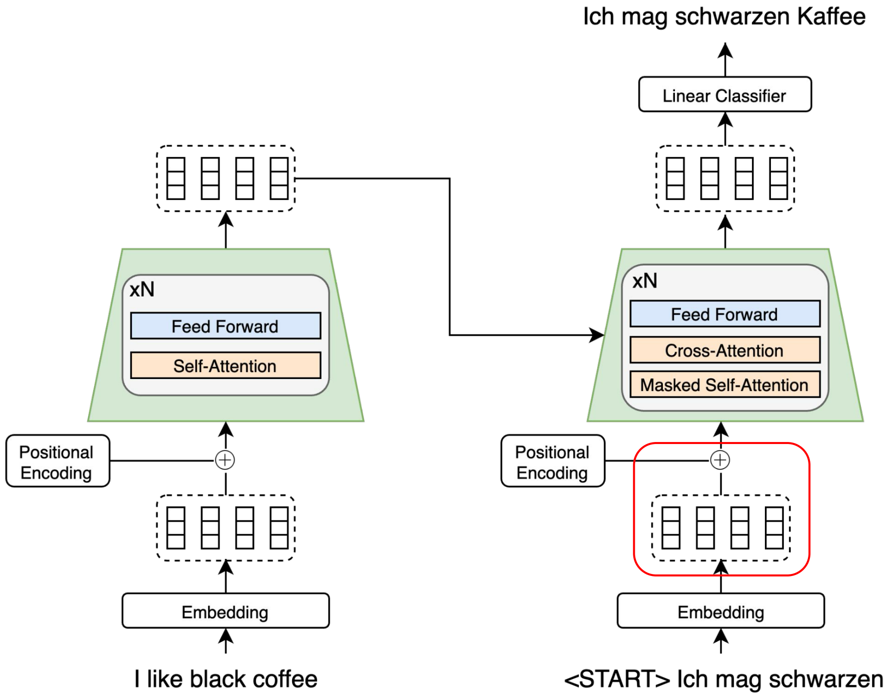


Cornell Bowers C·IS

Cross-Attention

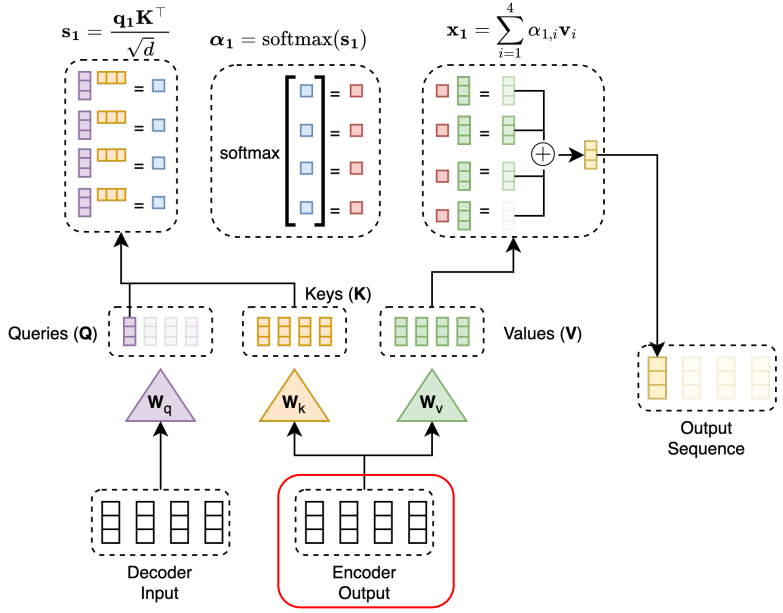


$\mathbf{s}_1 = \frac{\mathbf{q}_1 \mathbf{K}^\top}{\sqrt{d}}$
$\boldsymbol{\alpha}_1 = \mathrm{softmax}(\mathbf{s}_1)$
$\mathbf{x}_1 = \sum_{i=1}^{4} \alpha_{1,i} \mathbf{v}_i$
softmax
Keys (**K**)
Queries (**Q**)
Values (**V**)
W_q
W_k
W_v
Output Sequence
Decoder Input
Encoder Output

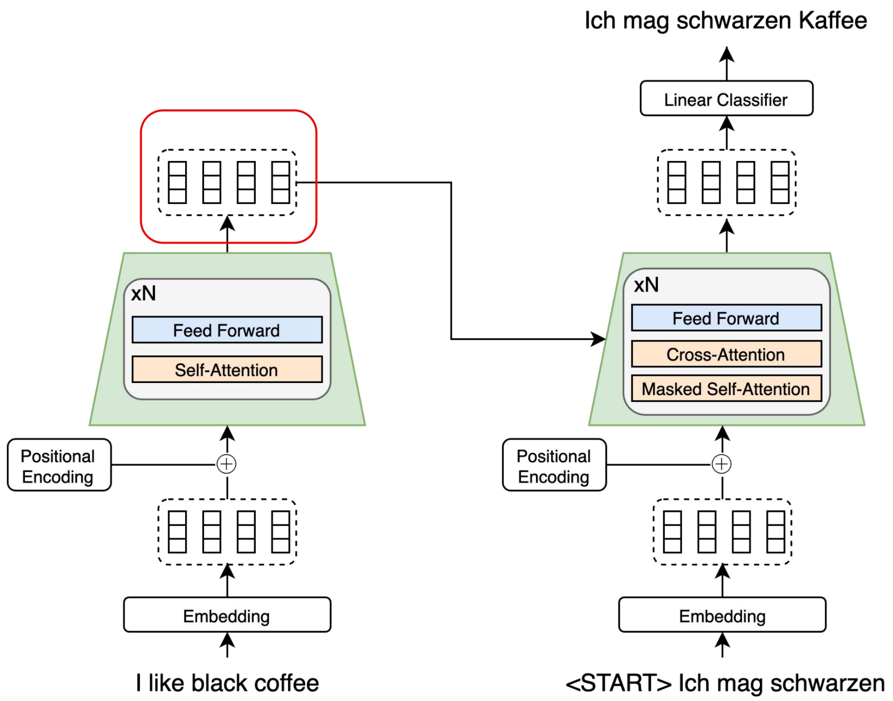


Cornell Bowers C·IS

Cross-Attention

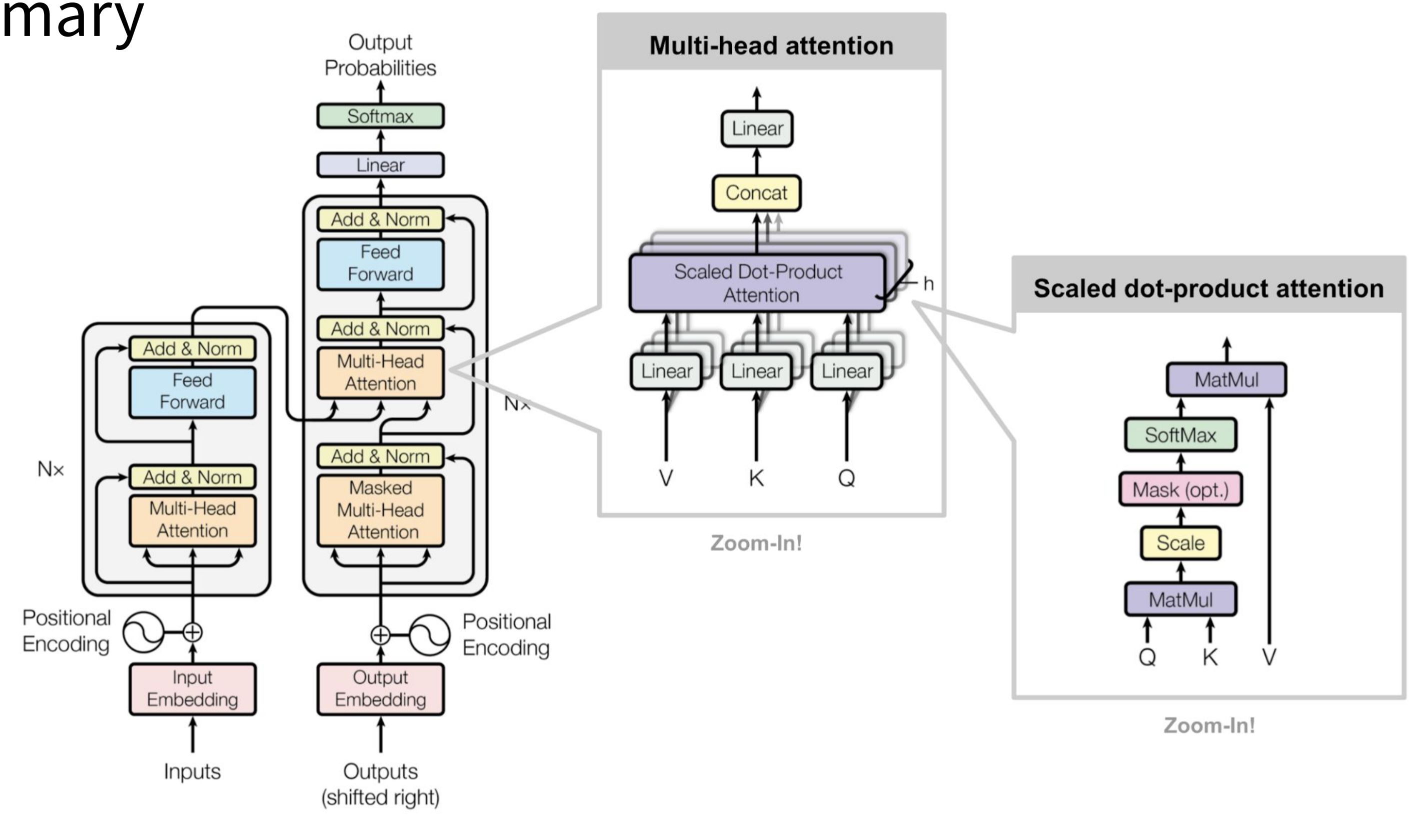


$\mathbf{s_1} = \frac{\mathbf{q_1}\mathbf{K}^\top}{\sqrt{d}}$
$\boldsymbol{\alpha_1} = \mathrm{softmax}(\mathbf{s_1})$
$\mathbf{x_1} = \sum_{i=1}^{4} \alpha_{1,i}\mathbf{v}_i$
softmax
Keys (**K**)
Queries (**Q**)
Values (**V**)
W_q
W_k
W_v
Output Sequence
Decoder Input
Encoder Output



Cornell Bowers C·IS

Summary



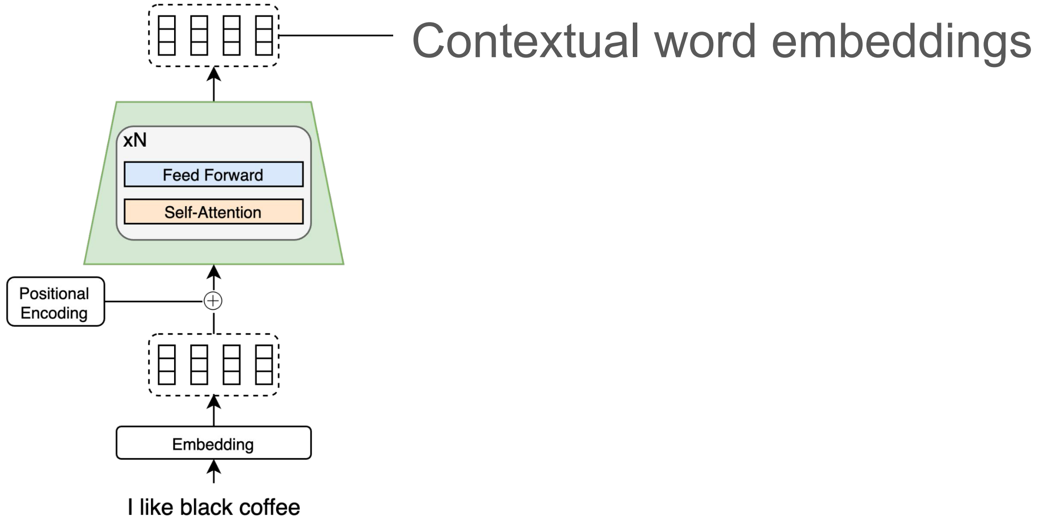
Cornell Bowers C·IS

Interactive Transformer Demo

Cornell Bowers C·IS

BERT (Bidirectional Encoder Representations from Transformers)

- Output of **Transformer Encoder** as contextual word embeddings
- Bidirectional Context
- Pre-trained on the language, and then fine-tuned

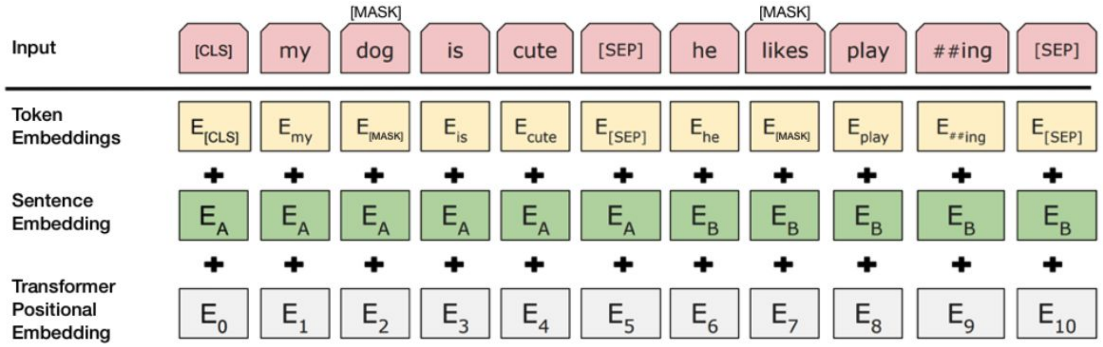


Cornell Bowers C·IS

BERT - Input Representation

Input:

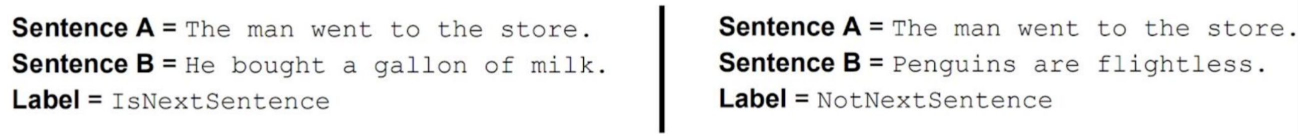
- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings



Training

- Masked Language Modelling
  - Mask out k% of the input words, and then predict the masked words
  - the man went to the store to [MASK] a [MASK] of milk
  - What can you use as a loss function?

- Next sentence prediction
  - To learn relationships between sentences, predict whether Sentence B is actual sentence that
  - proceeds Sentence A, or a random sentence



Sentence A = `The man went to the store.`
Sentence B = `He bought a gallon of milk.`
Label = `IsNextSentence`

Sentence A = `The man went to the store.`
Sentence B = `Penguins are flightless.`
Label = `NotNextSentence`

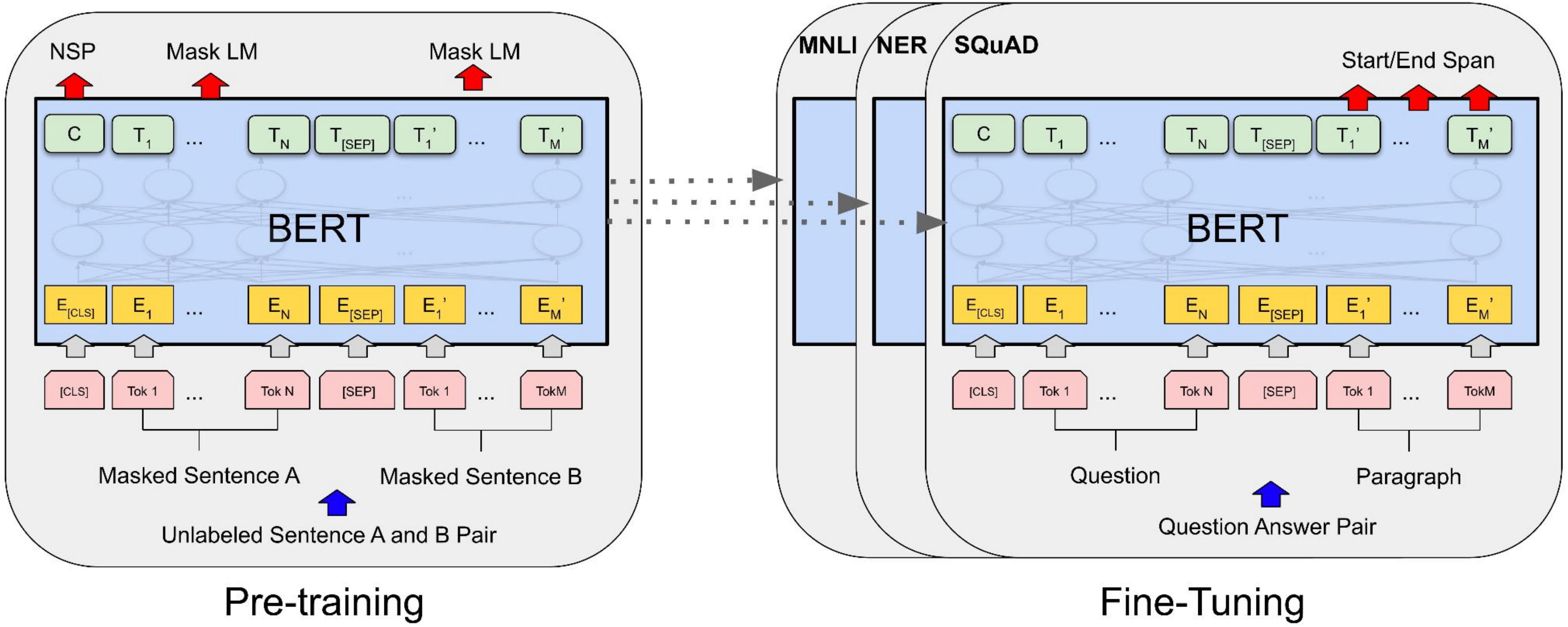
Cornell Bowers C·IS

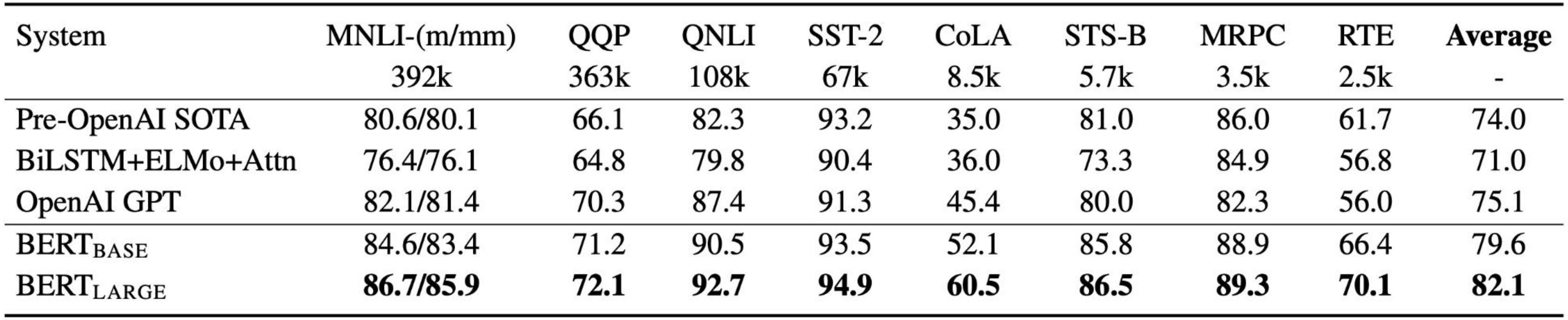
Model Details

- Data: Wikipedia (2.5B words) + BookCorpus (800M words)
- Batch Size: 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)
- Training Time: 1M steps (~40 epochs)
- Optimizer: AdamW, 1e-4 learning rate, linear decay
- BERT-Base: 12-layer, 768-hidden, 12-head, 110M params
- BERT-Large: 24-layer, 1024-hidden, 16-head, 340M params
- Trained on 4x4 or 8x8 TPU slice for 4 days

Cornell Bowers C·IS

Pre-training to Fine-tuning Pipeline





System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT$_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT$_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

https://arxiv.org/pdf/1810.04805.pdf

Cornell Bowers C·IS

Self-supervised Learning

- Labels are generated automatically, no human labeling process
- Benefits
  - Scales well
  - Cost-Efficient
  - Flexible
- Challenges
  - Larger datasets are required
  - More compute is necessary

Review

- LSTMs/GRUs are recurrent
- Self-attention can effectively replace recurrence in sequence-to-sequence models
- Transformers use self-attention and are parallelizable
- Pre-training using self-supervised learning help train large models that learn very good representations