CS247: ADVANCED DATA MINING

07: Text Data: Transformers

Instructor: Yizhou Sun

yzsun@cs.ucla.edu

February 7, 2024

Text Data: Transformers

- Introduction
- Attention is all you need
- Pretraining: BERT and GPT
- Summary

The First Transformer Paper

- NIPS 2017: https://arxiv.org/pdf/1706.03762.pdf

Attention Is All You Need

107k+ citations by 2/4/2024!

Note: Word2Vec Paper (NIPS 2013): 42.3K citations

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

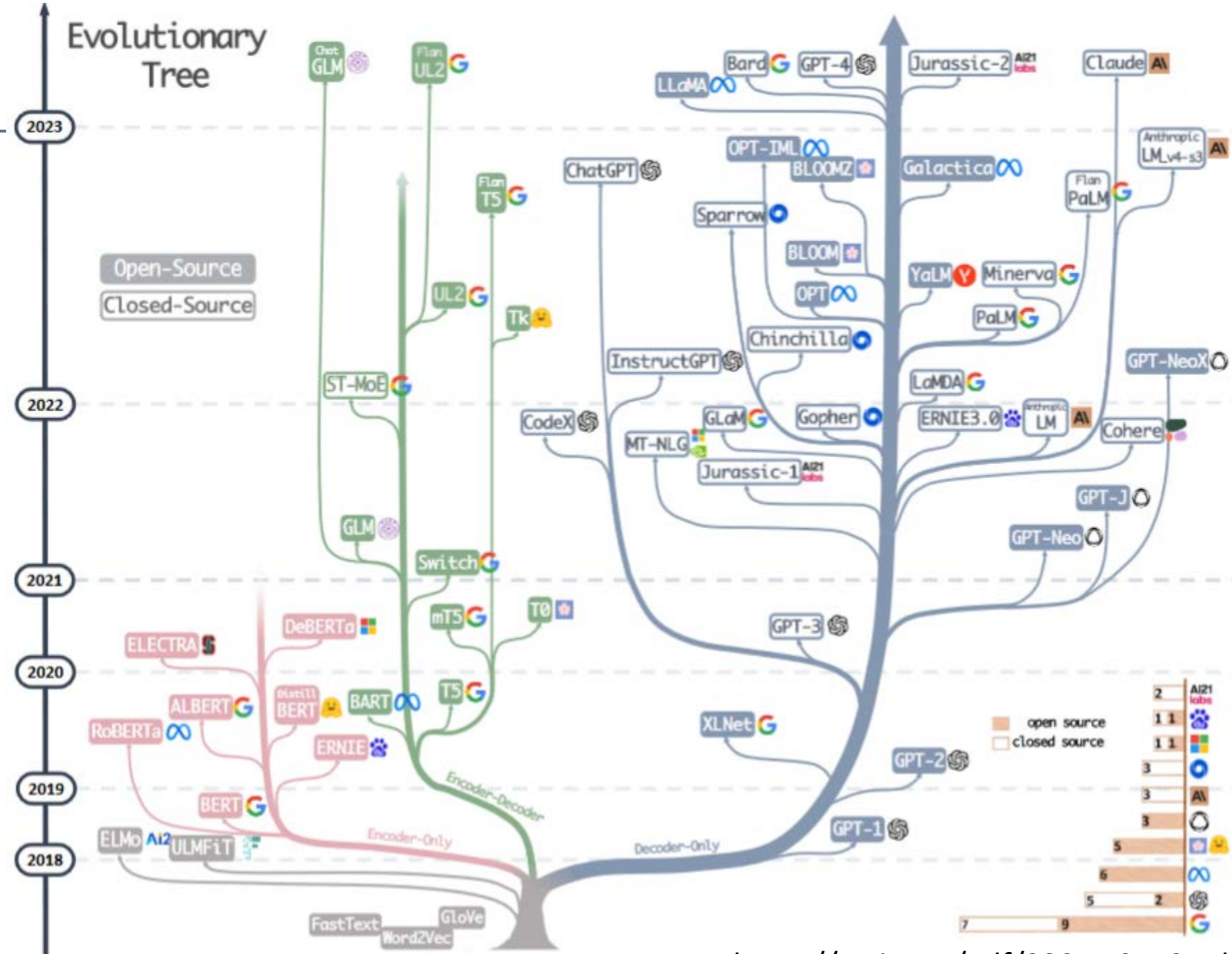
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

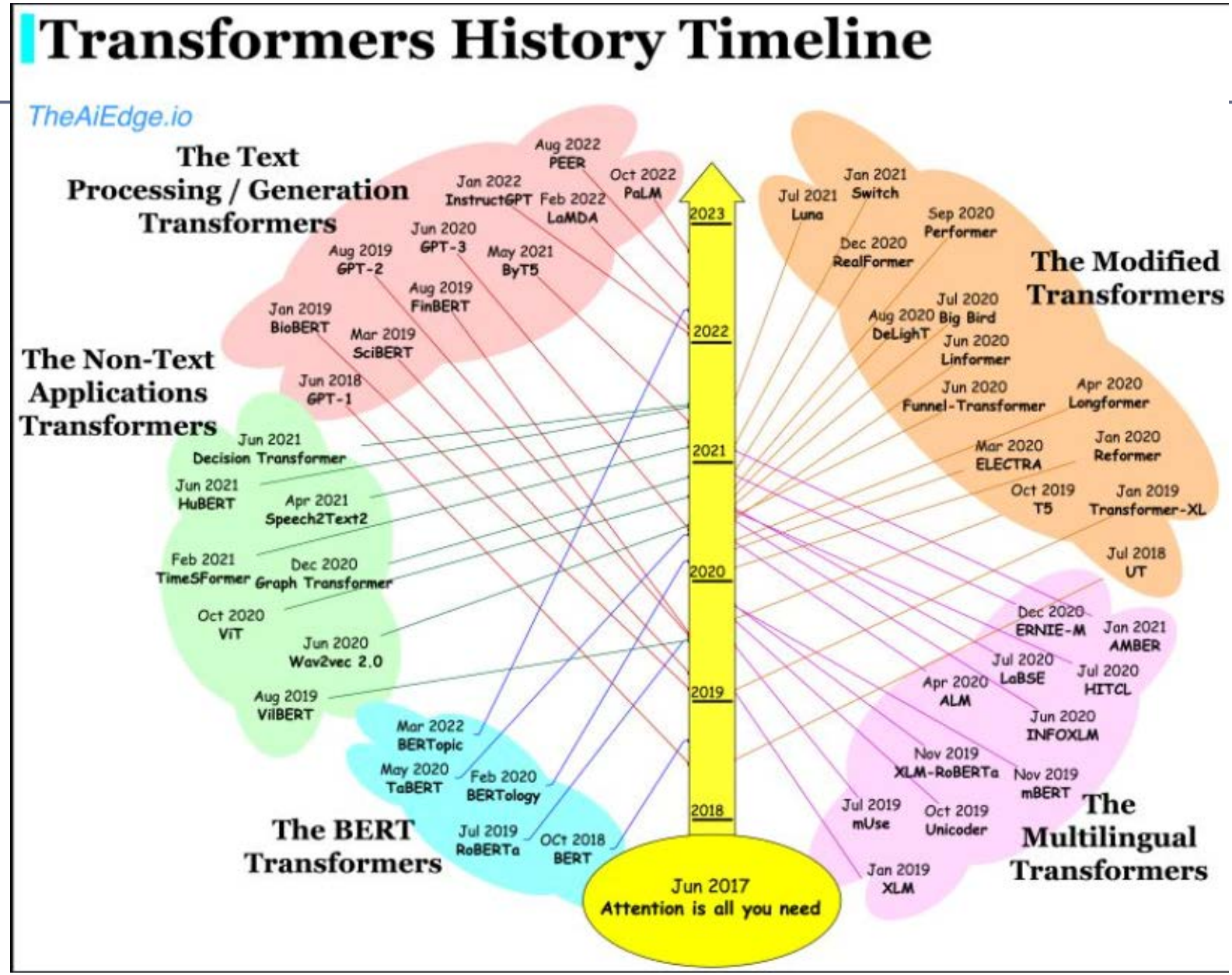
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

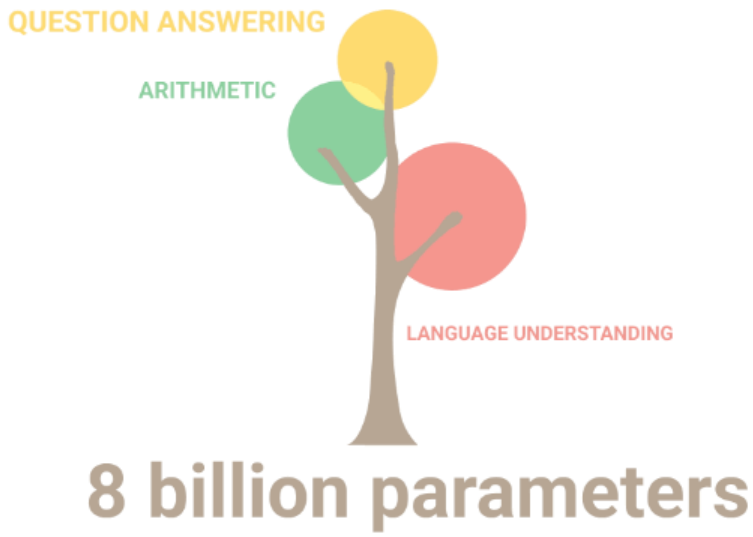


https://arxiv.org/pdf/2304.13712.pdf



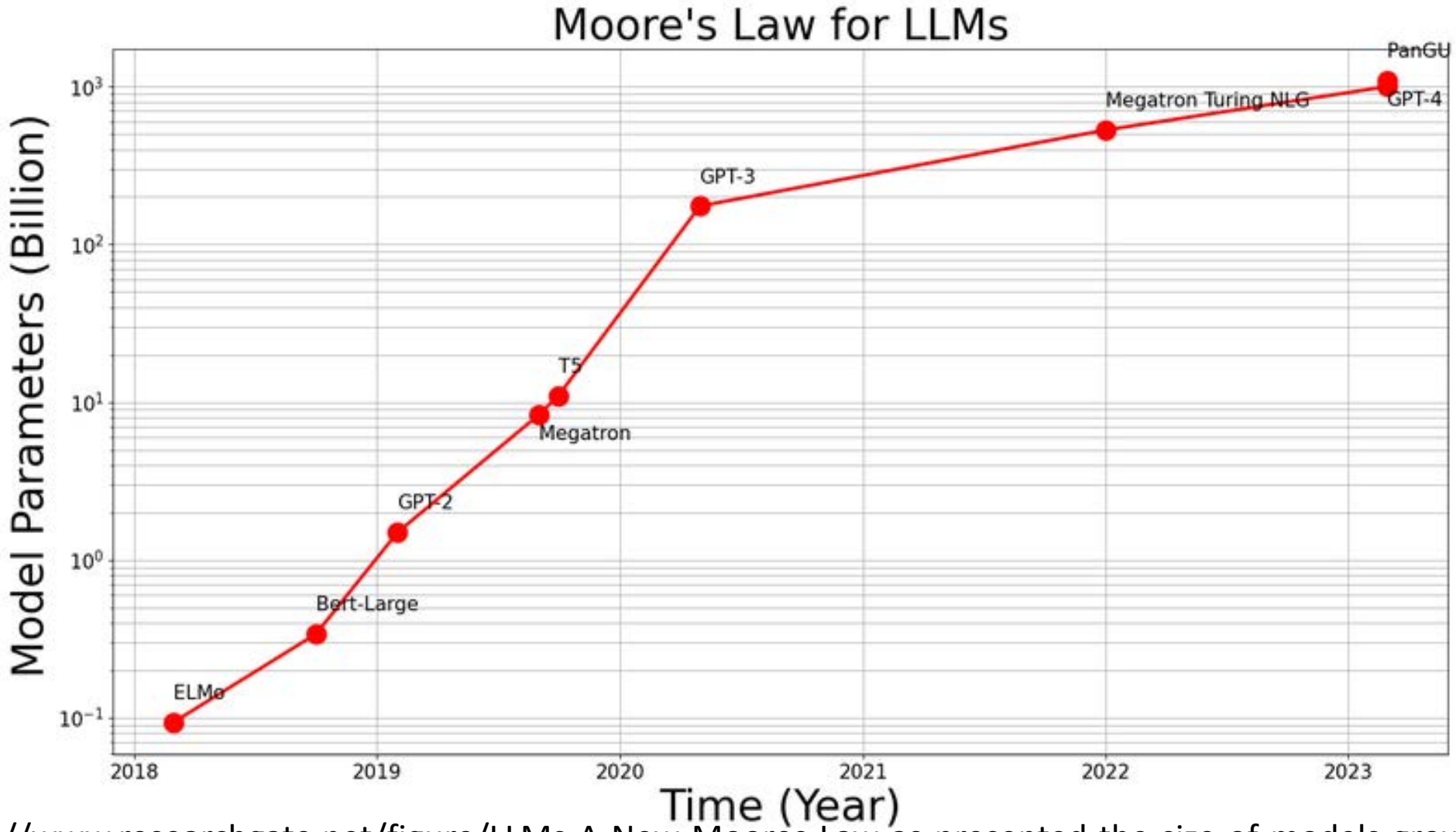
https://vinija.ai/models/Transformers/

Emergent capability of Large Models



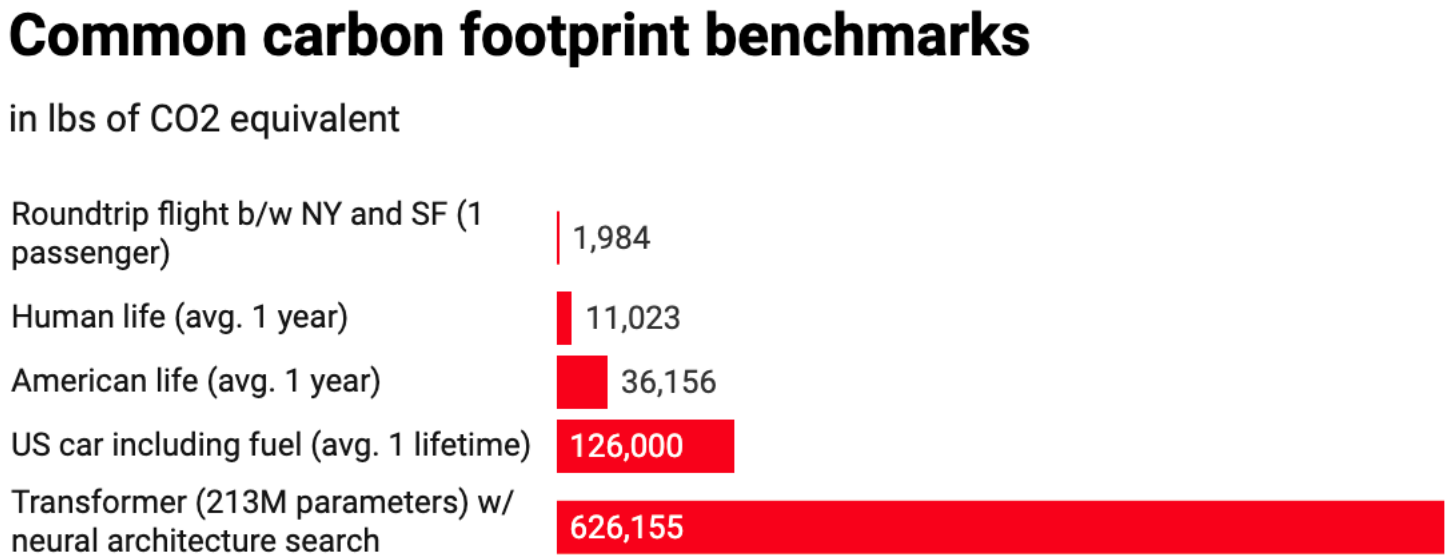
https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

The New Moore's Law?



https://www.researchgate.net/figure/LLMs-A-New-Moores-Law-as-presented-the-size-of-models-grows-exponentially-with-time_fig2_372248458

Large-scale deep learning is power hungry

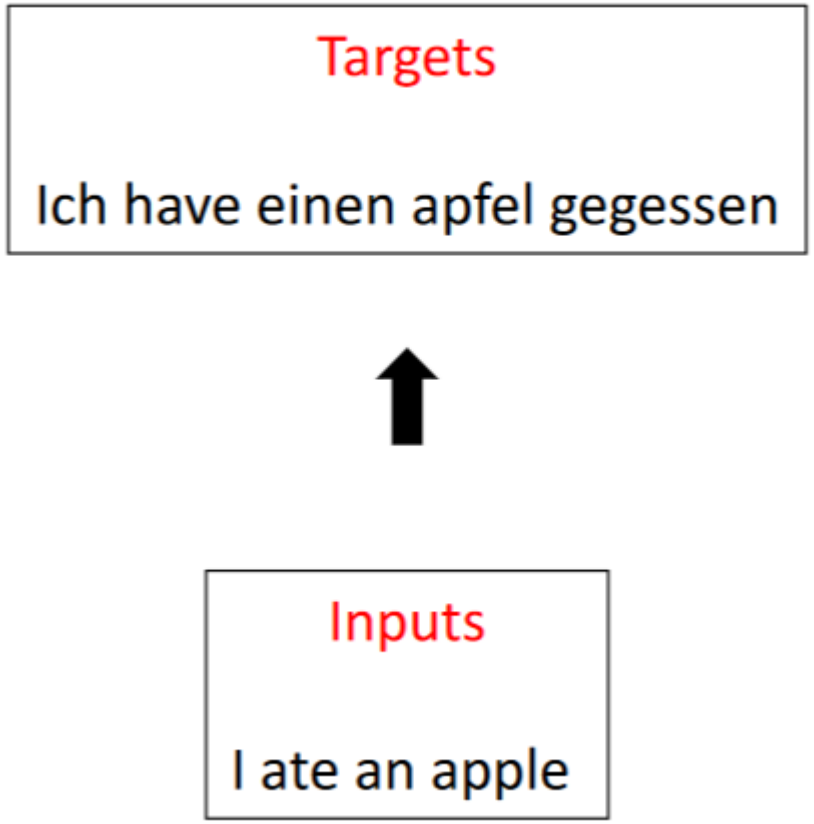


- These gigantic models require enormous computational resources and draw substantial energy and environmental cost
  - GPT-3 requires 190,000 Kwh of energy, 187,400 lbs CO_2
  - NAS with Transformer costs 626,000 lbs CO_2
- We need sustainable and environmentally friendly AI

Text Data: Transformers

- Introduction
- Attention is all you need
- Pretraining: BERT and GPT
- Summary

Start from Machine Translation Task



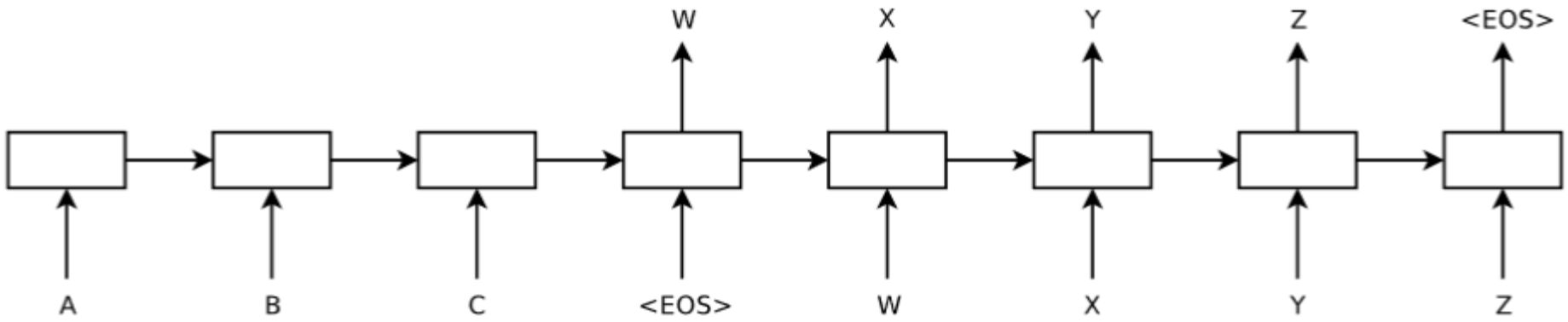
Problem Formalization

- Input: $(x_1, x_2, \ldots, x_T)$, a sequence of T tokens
- Output: $(y_1, y_2, \ldots, y_{T'})$, a sequence of T' tokens

- Goal: model the conditional probability for $y_1, y_2, \ldots, y_{T'}$
  - $p(y_1, y_2, \ldots, y_{T'} | x_1, x_2, \ldots, x_T) = p(y_1 | x_1, x_2, \ldots, x_T) \times p(y_2 | x_1, x_2, \ldots, x_T, y_1) \times p(y_3 | x_1, x_2, \ldots, x_T, y_1, y_2) \times \cdots \times p(y_{T'} | x_1, x_2, \ldots, x_T, y_1, y_2, \ldots, y_{T'-1}) = \prod_t p(y_t | x_1, x_2, \ldots, x_T, y_1, \ldots, y_{t-1})$

Recap: Seq2Seq

- A special case of RNN: sequence to sequence model
  - Encoder: Use LSTM to map input to a vector (latent representation)
    - $\boldsymbol{c} = f(x_1, x_2, \dots, x_T)$
  - Decoder: Use LSTM to decode the vector to the target sequence
    - $p(y_t|y_1, \dots, y_{t-1}, \boldsymbol{c}) = g(y_1, \dots, y_{t-1}, \boldsymbol{c}) = g(y_{t-1}, \boldsymbol{s}_t, \boldsymbol{c})$, where $\boldsymbol{s}_t$ is the hidden state



https://arxiv.org/pdf/1409.3215.pdf

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the

Some Details

- Special token
  - End of sentence token: <EOS>, enables the model to handle sequence with any length
- Technically
  - $p(y_t|x_1, x_2, \dots, x_T, y_1, \dots, y_{t-1}) \approx p(y_t|y_1, \dots, y_{t-1}, \boldsymbol{c})$

Can we do better? – Attention!

- ICLR 2015 https://arxiv.org/pdf/1409.0473.pdf

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

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

ABSTRACT

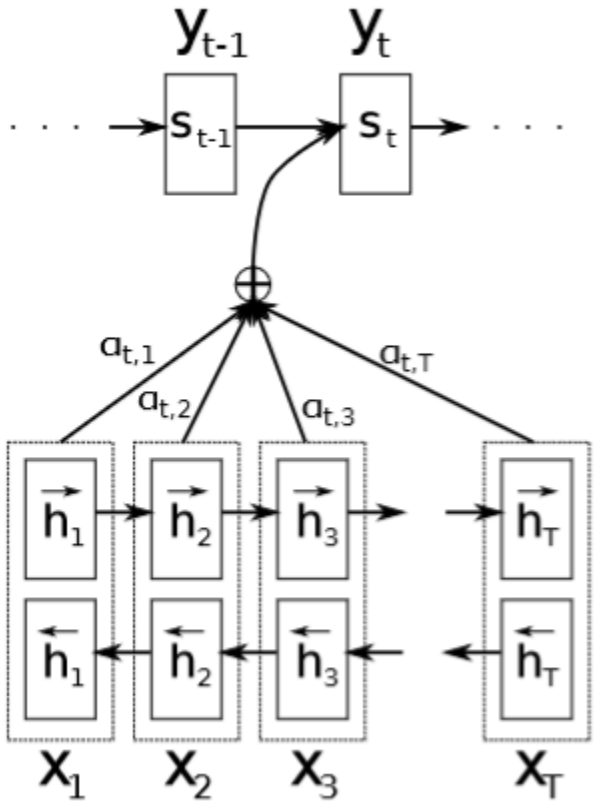
Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search

Intuition

- Attend different words in the input to predict the target
- No need to compress entire input sequence to one fixed vector
  - Performance deteriorate if input sequence length increases for Seq2seq model

The Attention Mechanism and Bahdanau Attention

- Each target word has a specific context
  - $p(y_t|x_1, x_2, \dots, x_T, y_1, \dots, y_{t-1}) =$
    $p(y_t|y_1, \dots, y_{t-1}, \boldsymbol{c}_t) = g(y_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}_t)$
- The context is a weighted average of input token's latent representation
  - $\boldsymbol{c}_t = \sum_j \alpha_{tj} h_j$
  - $\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_j \exp(e_{tj})}$
  - $e_{tj} = a(s_{t-1}, h_j) = v_a^T \tanh(W_a s_{t-1} + U_a h_j)$



A General Framework of Attention

- Given a dataset of m tuples of **keys** and **values**

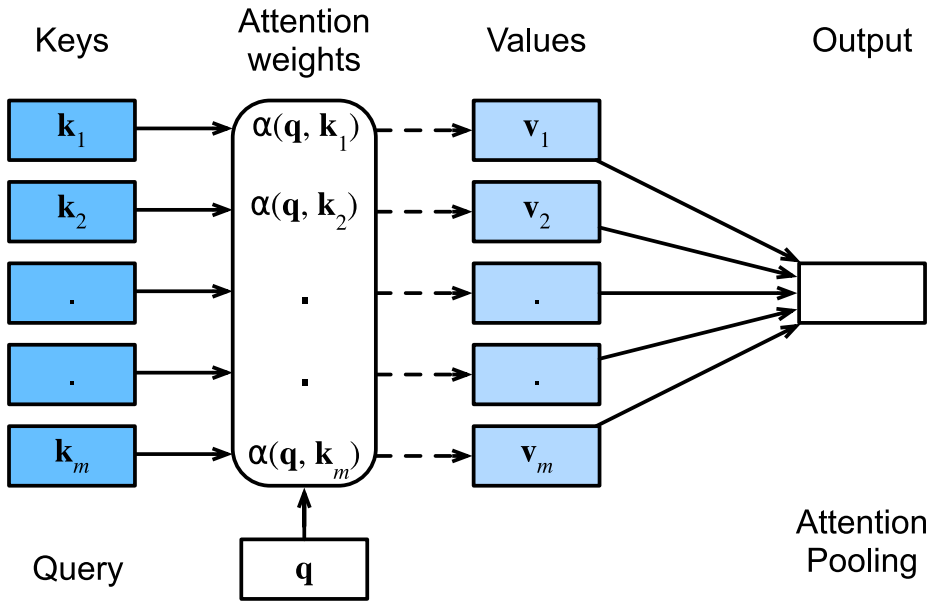
$$\mathcal{D} \stackrel{\text{def}}{=} \{(\mathbf{k}_1, \mathbf{v}_1), \ldots (\mathbf{k}_m, \mathbf{v}_m)\}$$

- Given **q** as a query vector, its attention over *D:*

$$\text{Attention}(\mathbf{q}, \mathcal{D}) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i,$$

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \frac{\exp(a(\mathbf{q}, \mathbf{k}_i))}{\sum_j \exp(a(\mathbf{q}, \mathbf{k}_j))}.$$

https://d2l.ai/chapter_attention-mechanisms-and-transformers/queries-keys-values.html



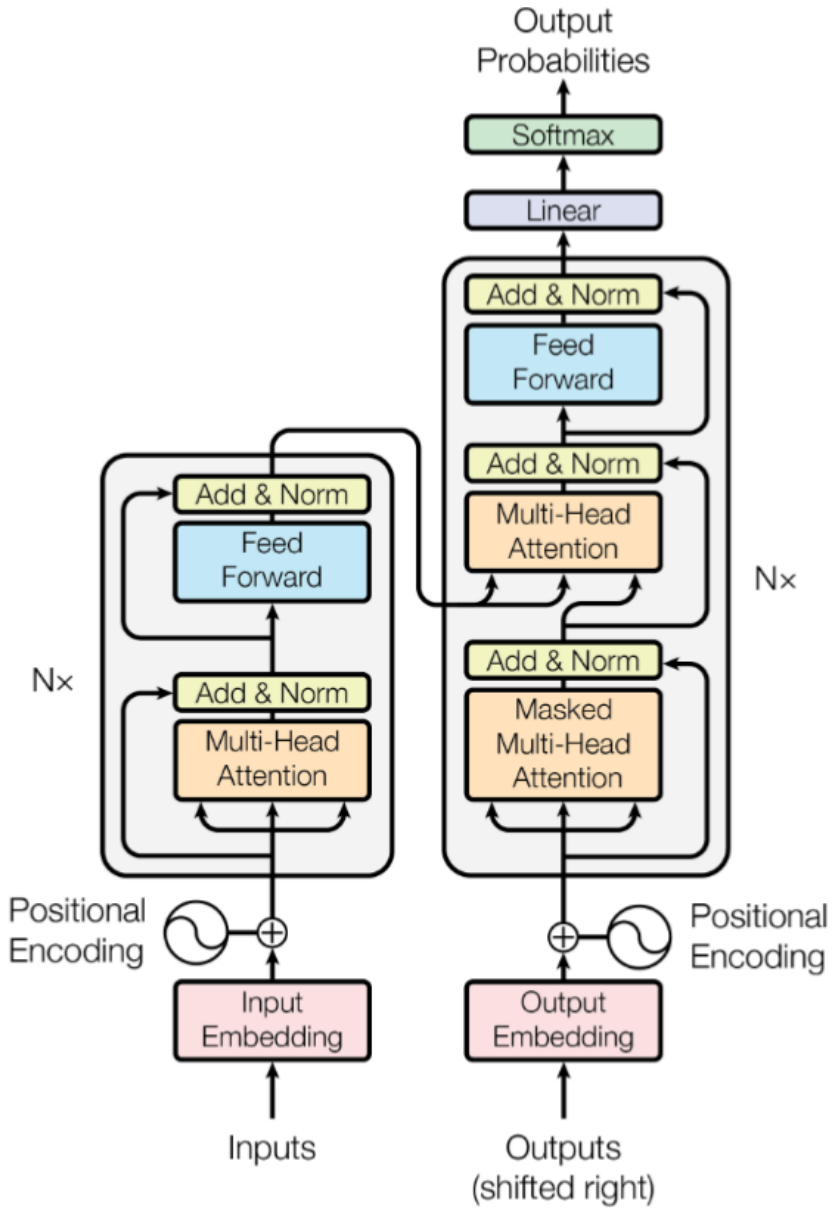
Recap: KNN and Kernel Methods

- Weighted KNN
  - $y_q = \frac{\sum w_i y_i}{\sum w_i}$, where x_i's are x_q's nearest neighbors
  - E.g., $w_i = \exp(-\left|\left|x_q - x_i\right|\right|^2 / 2\sigma^2)$, a gaussian kernel

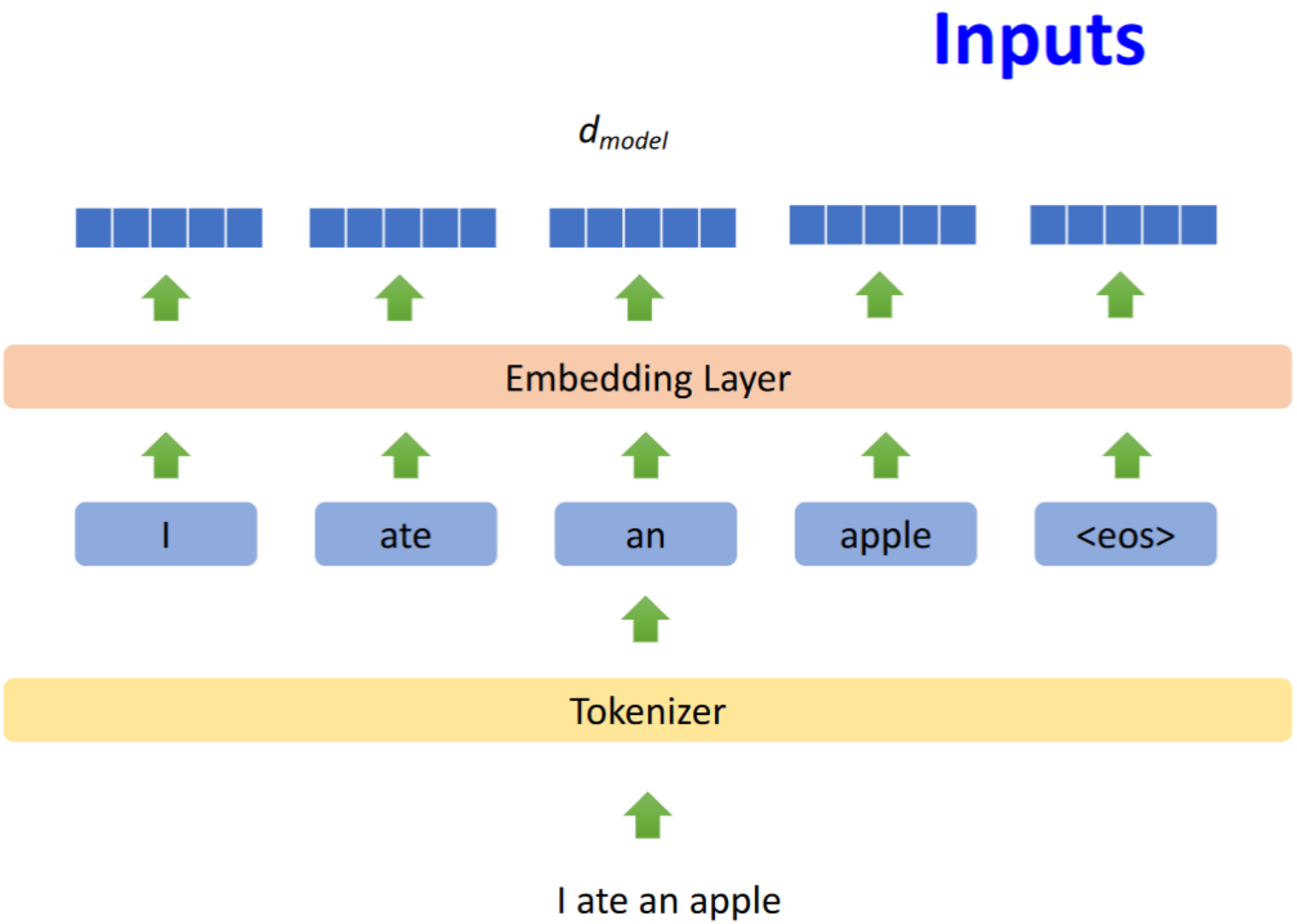
Now to the Transformer

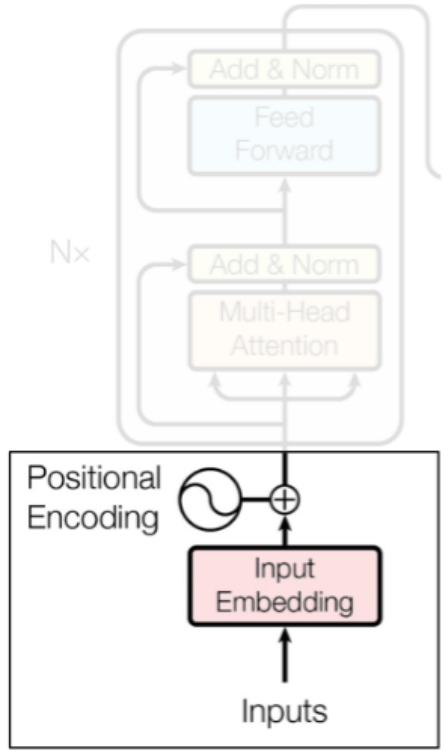
- Left: Encoder
  - Representation
- Right: Decoder
  - Generation

- The following slides are adapted from:
  - https://deeplearning.cs.cmu.edu/F23/document/slides/lec19.transformersLLMs.pdf

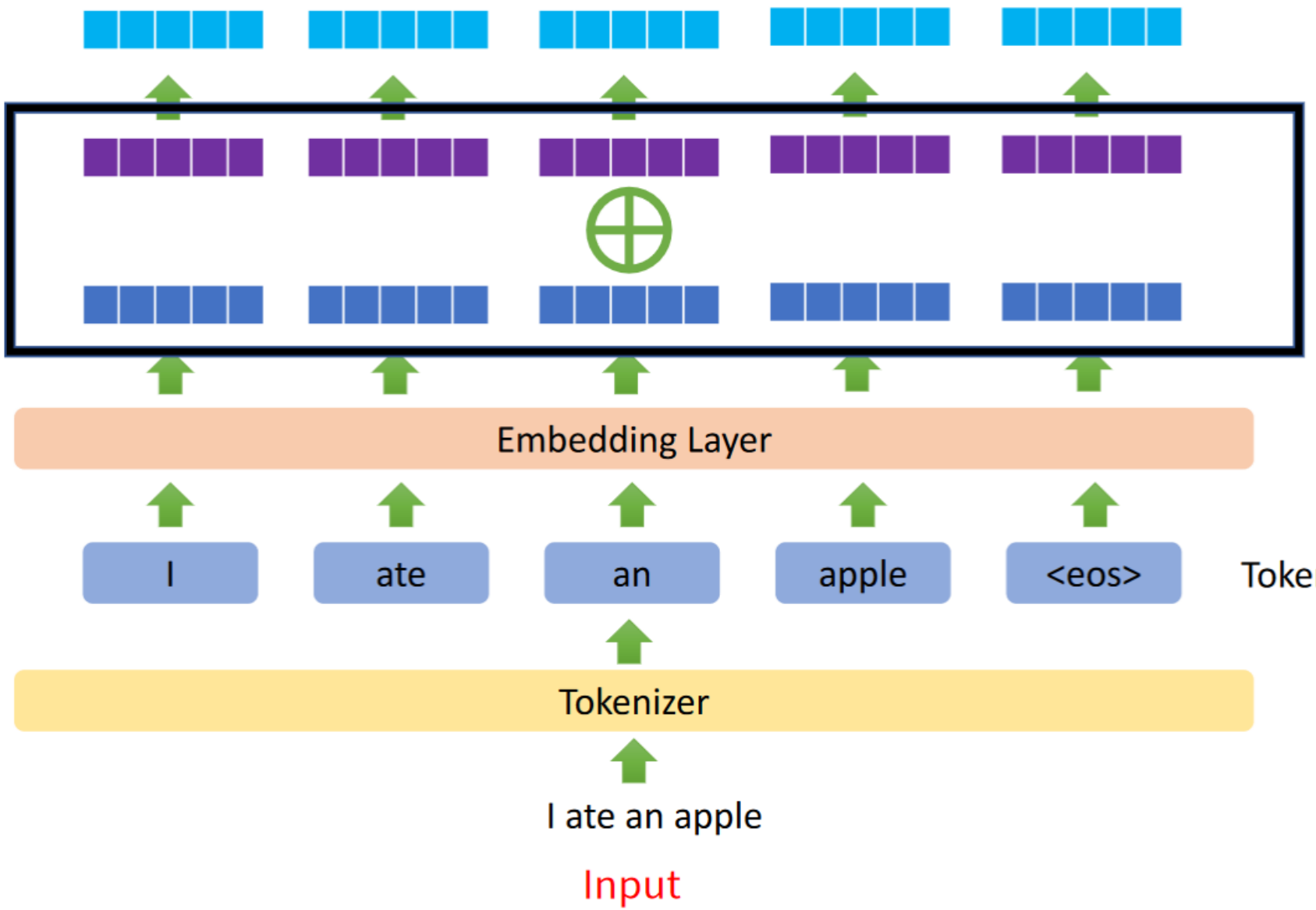


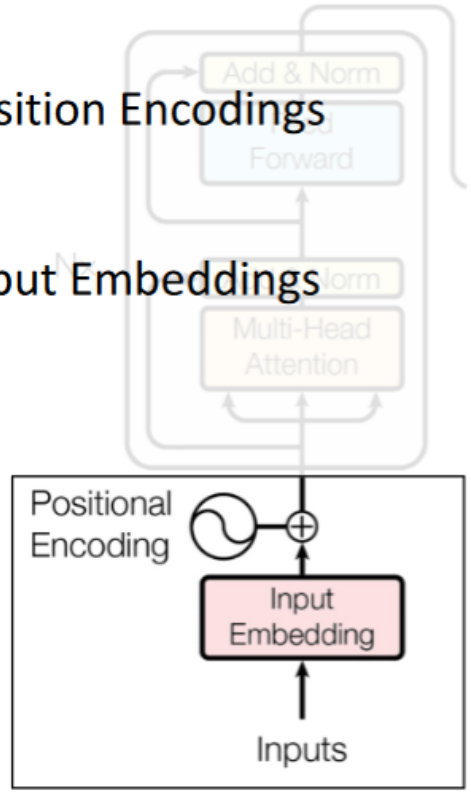
Generate learnable embedding for each token





Positional Encoding



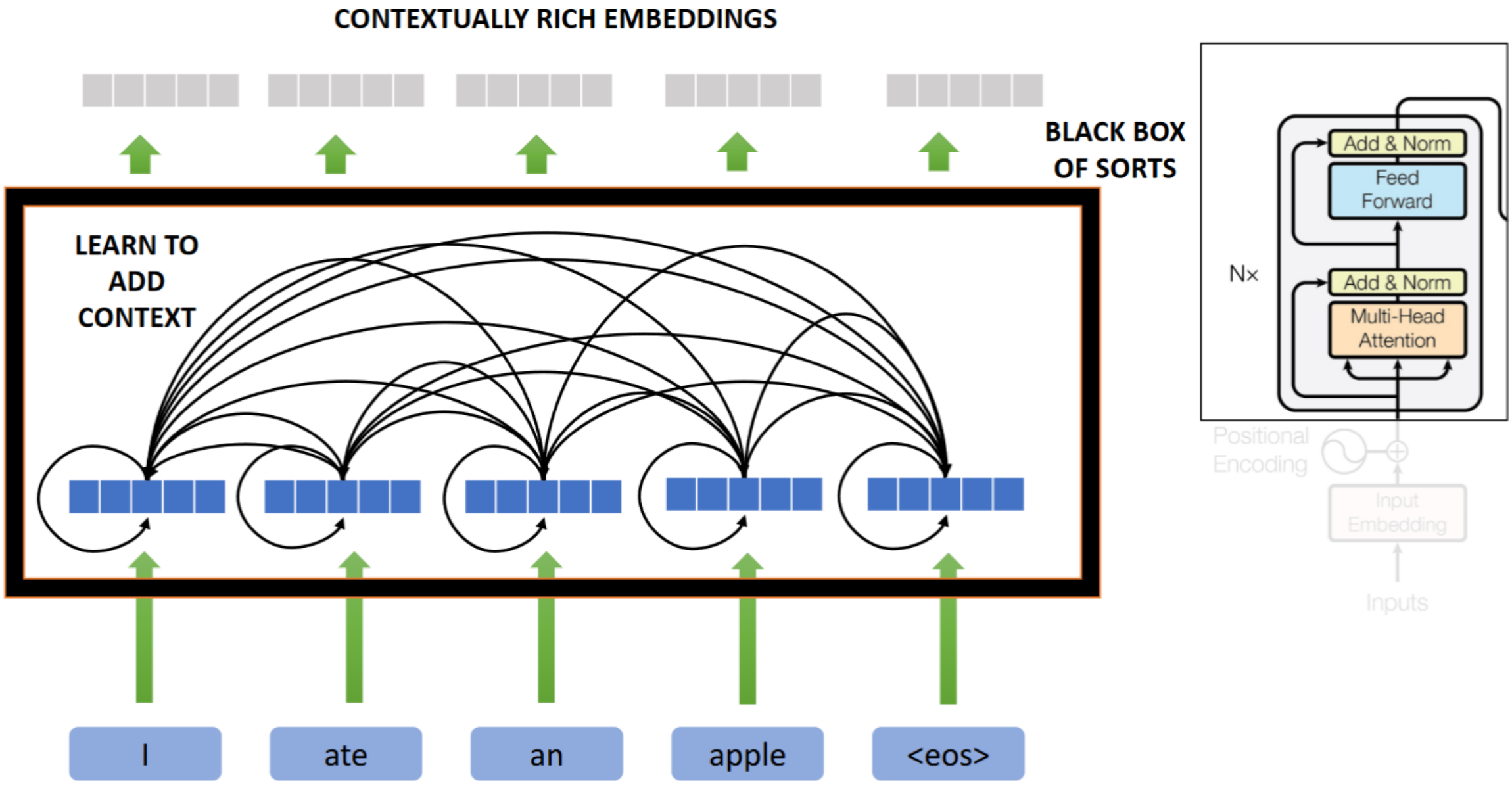


Positional Encoding

- For each dimension, read out values from sine and cosine with different frequencies

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

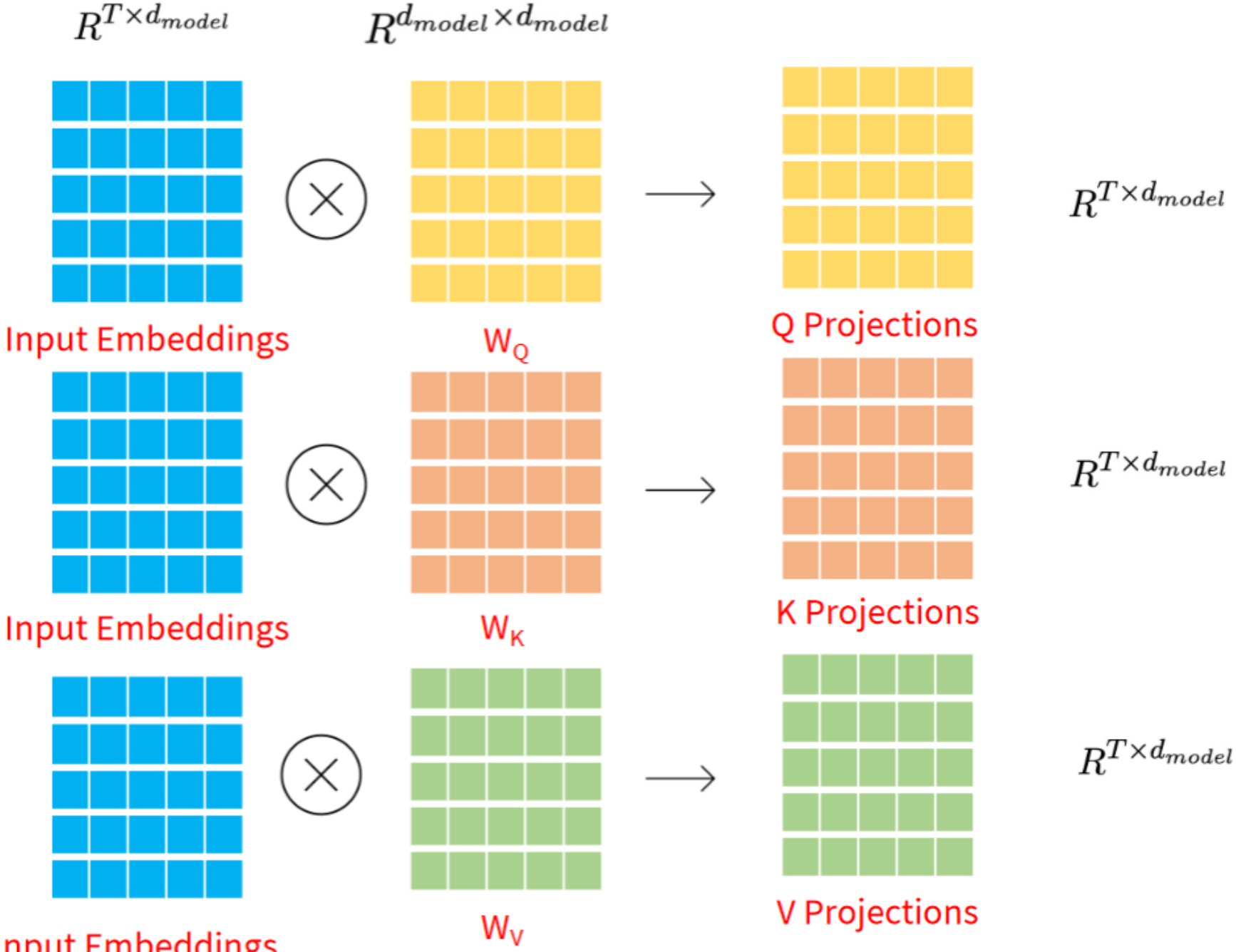
Encoder


CONTEXTUALLY RICH EMBEDDINGS
BLACK BOX OF SORTS
LEARN TO ADD CONTEXT
I
ate
an
apple
<eos>
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

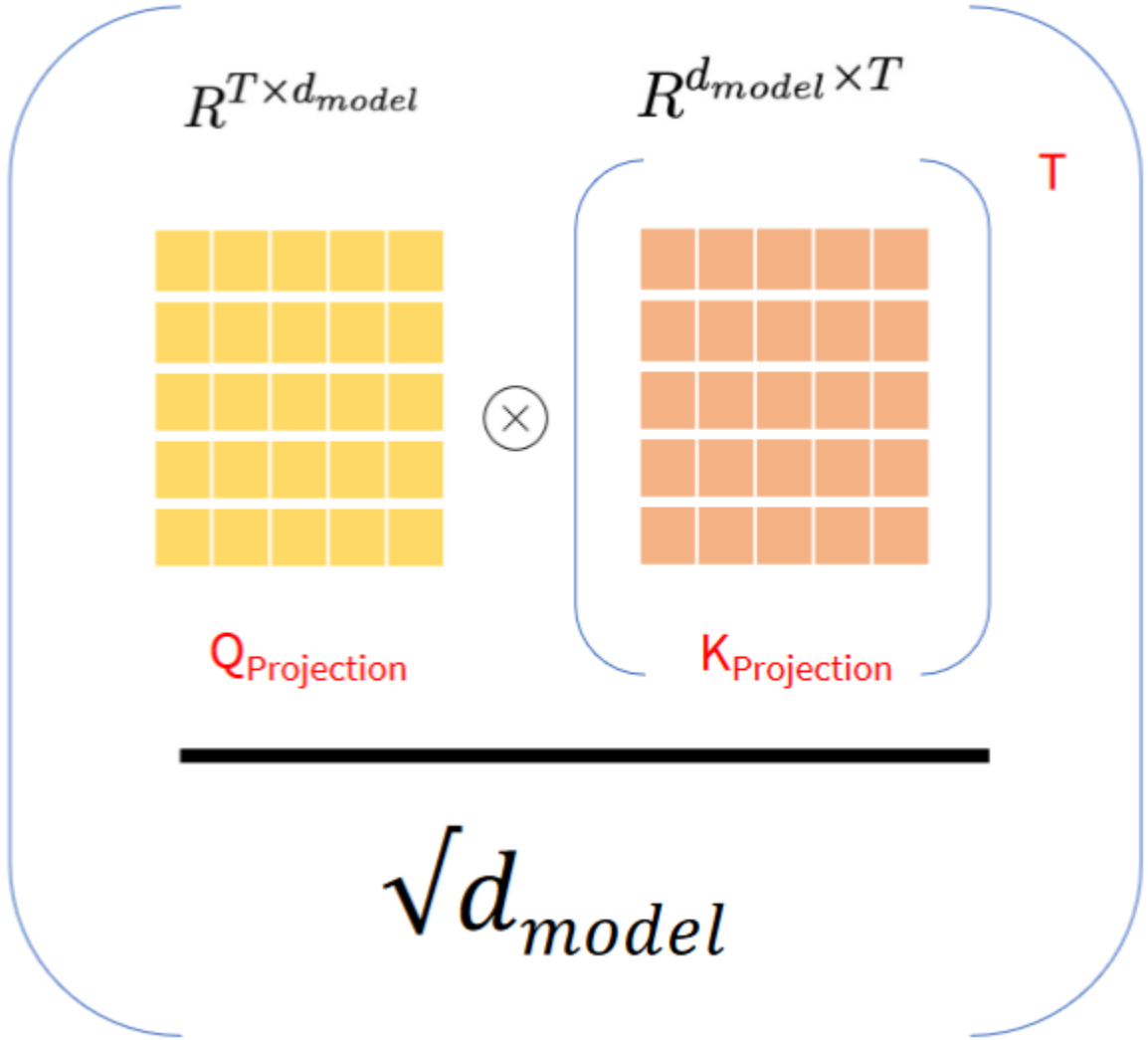
Scaled Dot Product Attention

- Each token's representation is a weighted average over all other token's representations
- For any token pair (i,j), let token i be the query
  - $Query: Q_i = W^Q h_i$; Key: $K_j = W^K h_j$; Value: $V_j = W^V h_j$
  - $\alpha_{ij} = softmax(\frac{W^Q h_i \cdot W^K h_j}{\sqrt{d_{model}}})$
    - Softmax to make sure $\sum_j \alpha_{ij} = 1$
  - $h_{i,new} = \sum_j \alpha_{ij} W^V h_j$
  - Matrix form:
    - $Q = (Q_1^T; Q_2^T; \ldots; Q_n^T)$
    - $K = (K_1^T; K_2^T; \ldots; K_n^T)$
    - $V = (V_1^T; V_2^T; \ldots; V_n^T)$

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


$R^{T \times d_{model}}$
$R^{d_{model} \times d_{model}}$
Input Embeddings
W_Q
Q Projections
$R^{T \times d_{model}}$
Input Embeddings
W_K
K Projections
$R^{T \times d_{model}}$
Input Embeddings
W_V
V Projections
$R^{T \times d_{model}}$

softmax

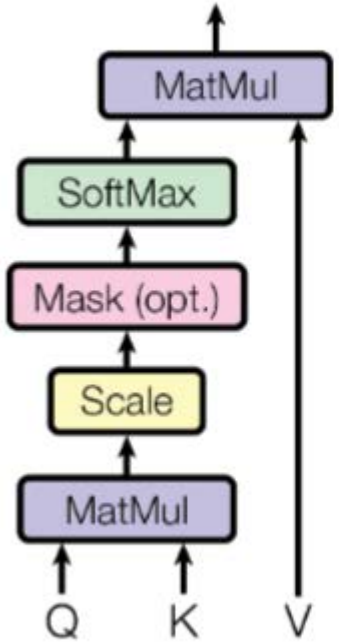

$R^{T \times d_{model}}$
$R^{d_{model} \times T}$
T
$Q_{Projection}$
$K_{Projection}$
$\sqrt{d_{model}}$

Multi-Head Attention

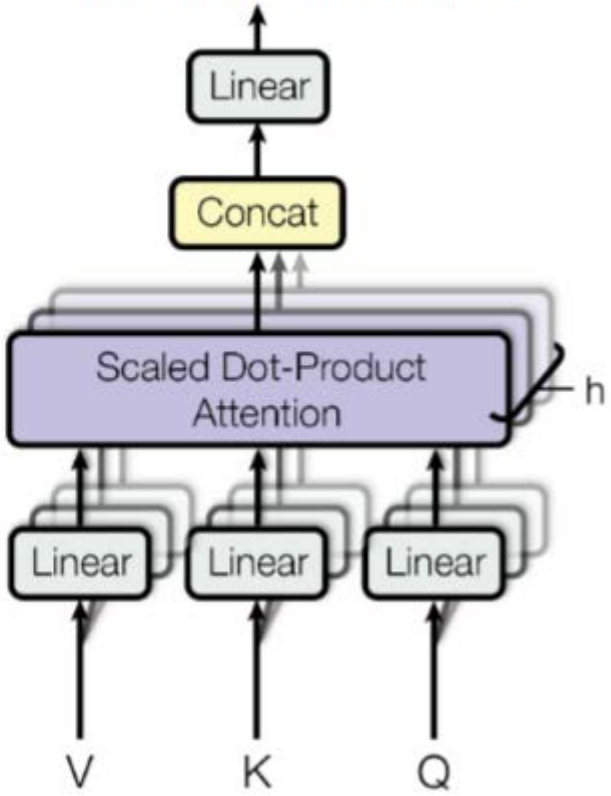
- Multiple W^Q, W^K, W^V

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

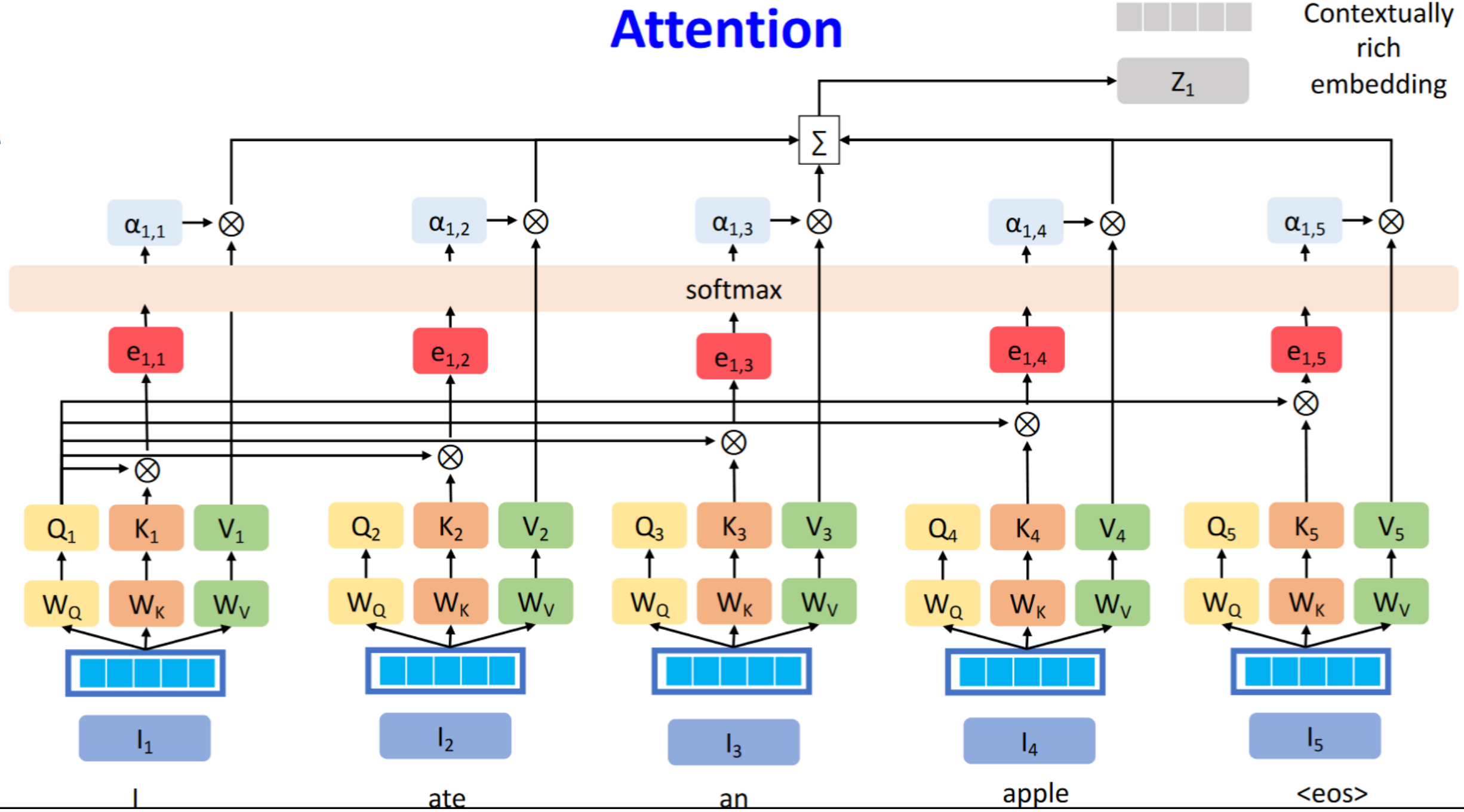
Scaled Dot-Product Attention



Multi-Head Attention

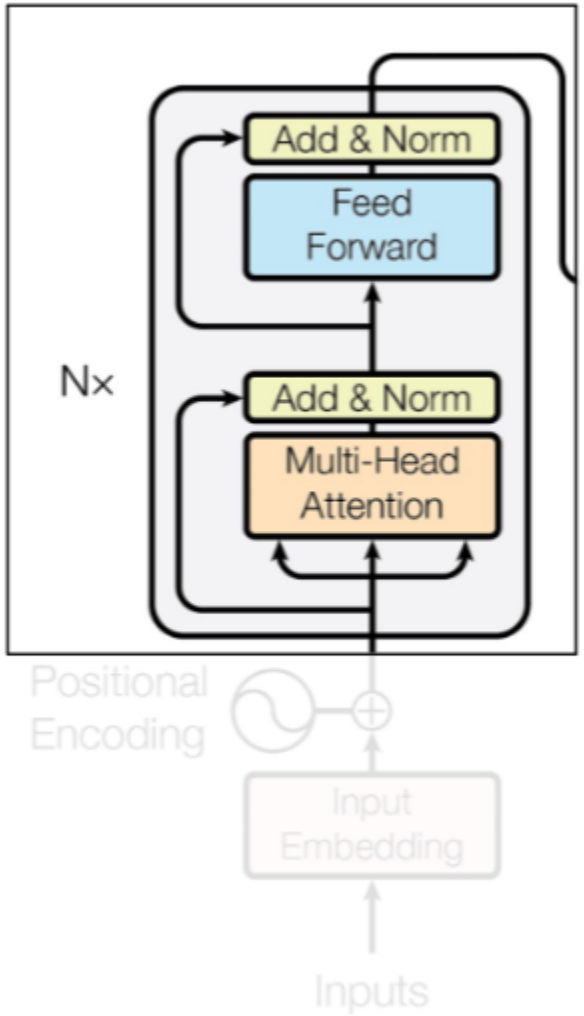


Attention

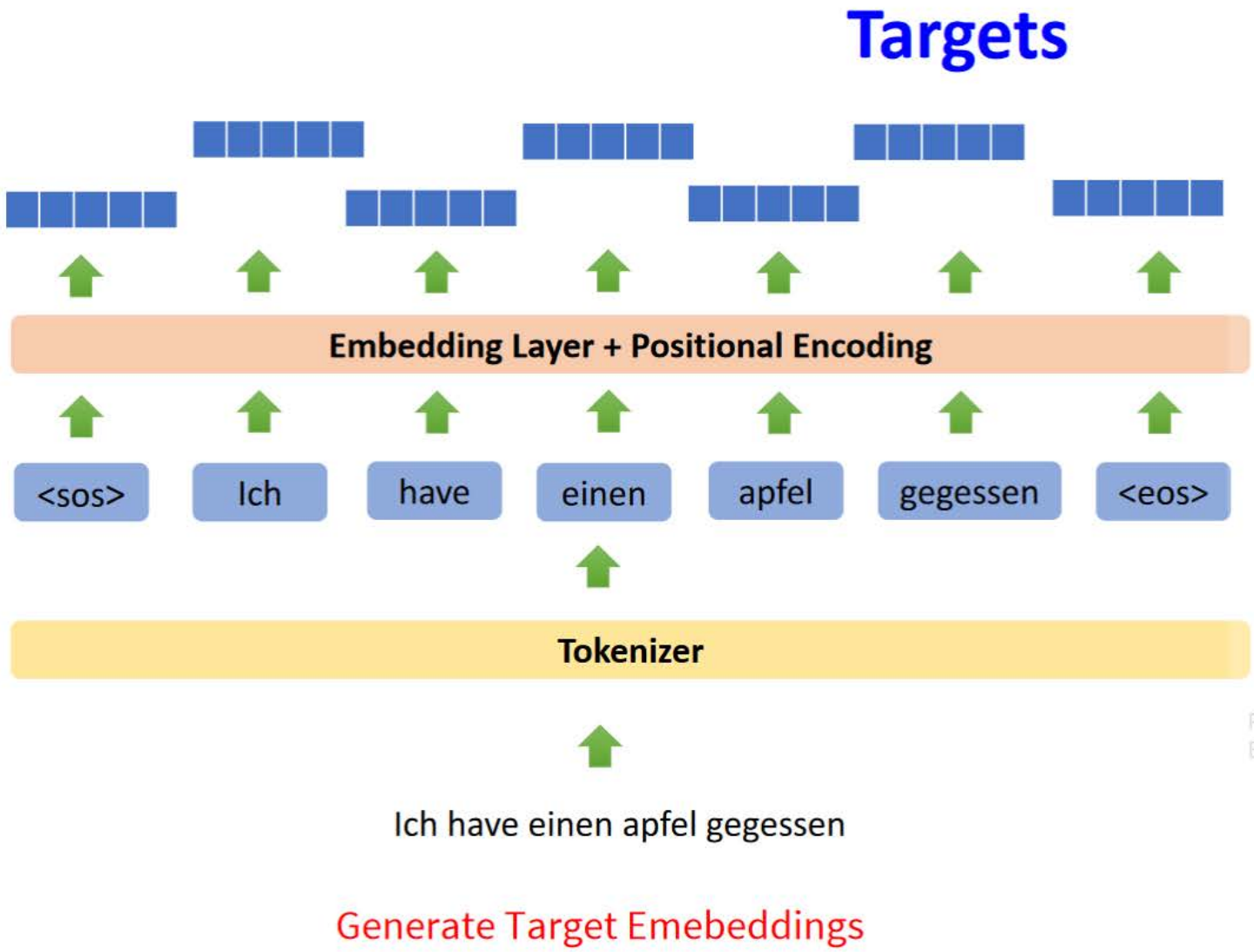

Contextually rich embedding
Z_1
Σ
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,3}$
$\alpha_{1,4}$
$\alpha_{1,5}$
softmax
$e_{1,1}$
$e_{1,2}$
$e_{1,3}$
$e_{1,4}$
$e_{1,5}$
Q_1 K_1 V_1
Q_2 K_2 V_2
Q_3 K_3 V_3
Q_4 K_4 V_4
Q_5 K_5 V_5
W_Q W_K W_V
I_1
I_2
I_3
I_4
I_5
I
ate
an
apple
<eos>

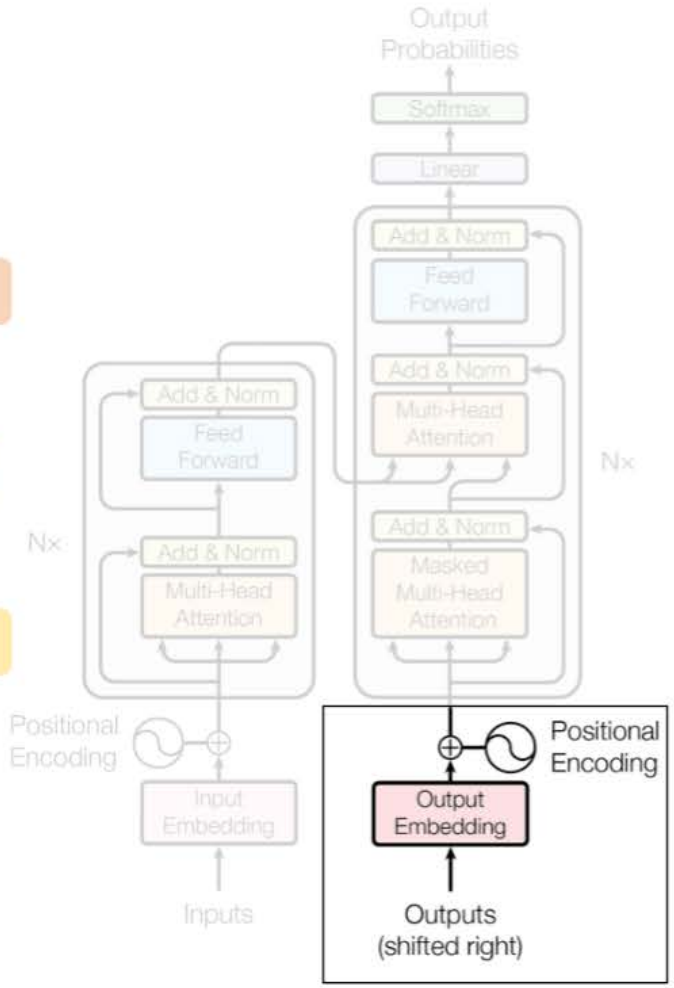
Add and Norm

- Add:
  - Residual connection
  - Input + new embedding
- Norm:
  - Layer normalization $y = \frac{x - \mu}{\sigma}$
    - Every output neuron is standardized
  - Critical
    - https://arxiv.org/pdf/2305.02582.pdf



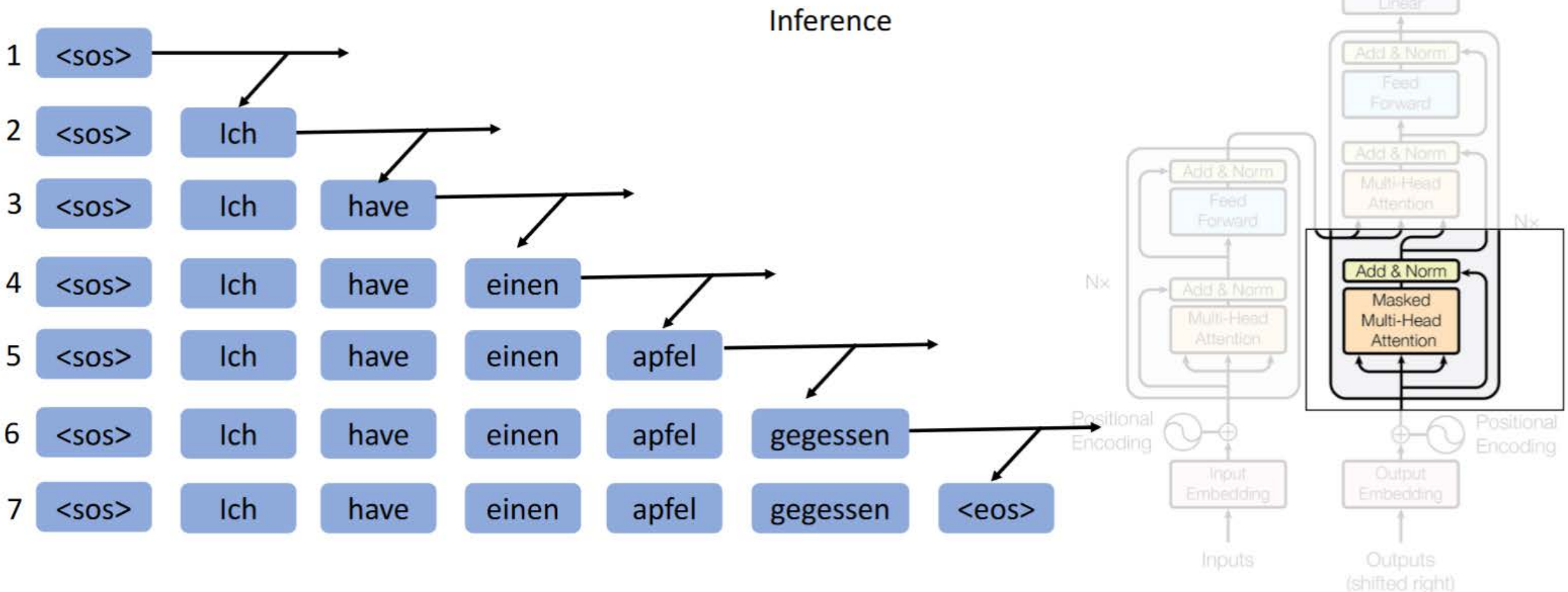
Now come to the decoding part





Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)



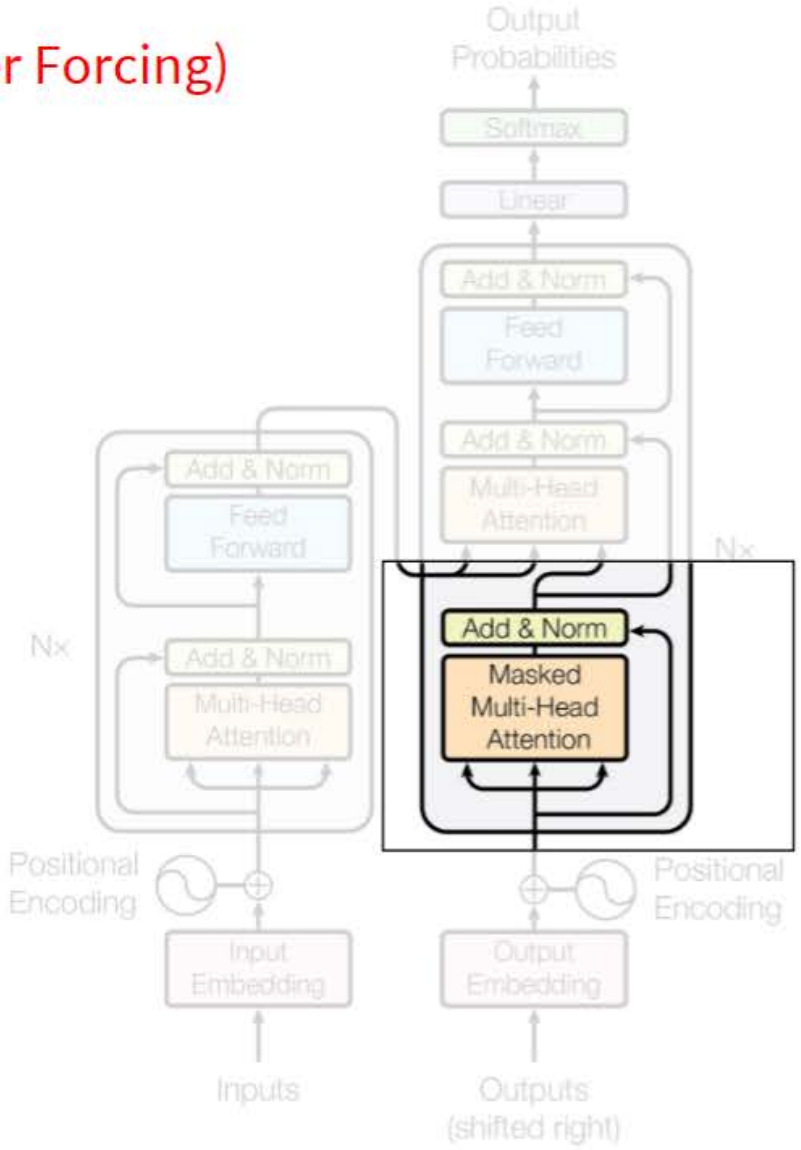
Training stage: soft mask to allow parallelization

Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

1	<sos>	- ∞	- ∞	- ∞	- ∞	- ∞	- ∞
2	<sos>	Ich	- ∞	- ∞	- ∞	- ∞	- ∞
3	<sos>	Ich	have	- ∞	- ∞	- ∞	- ∞
4	<sos>	Ich	have	einen	- ∞	- ∞	- ∞
5	<sos>	Ich	have	einen	apfel	- ∞	- ∞
6	<sos>	Ich	have	einen	apfel	gegessen	- ∞
7	<sos>	Ich	have	einen	apfel	gegessen	<eos>

Softmax -> - ∞ -> 0



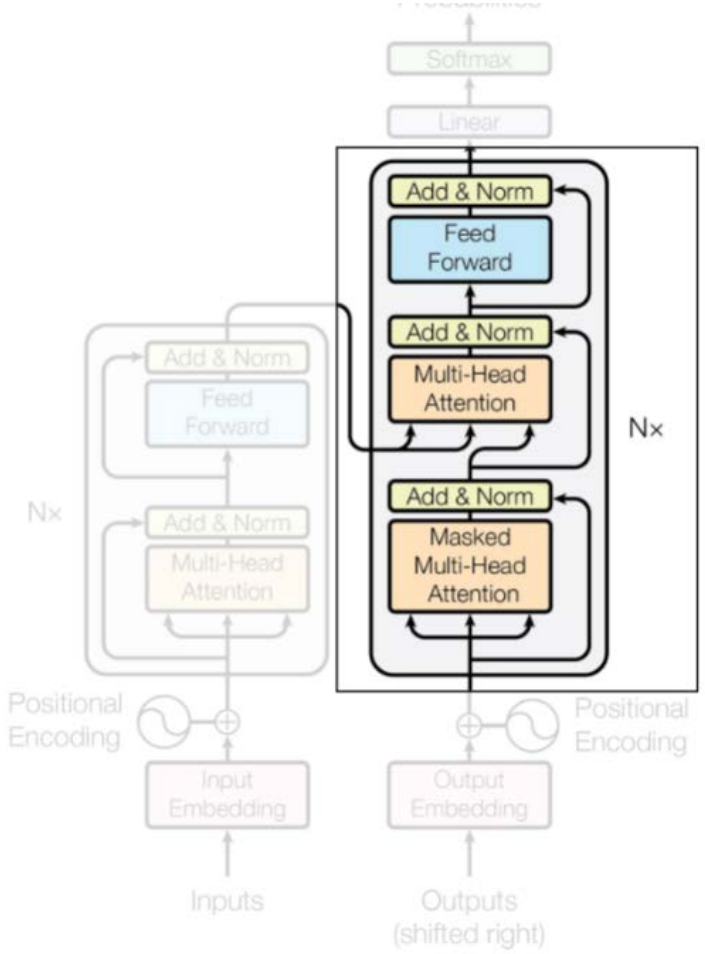
Encoder-Decoder Attention

Encoder

Keys from **Encoder Outputs**
Values from **Encoder Outputs**

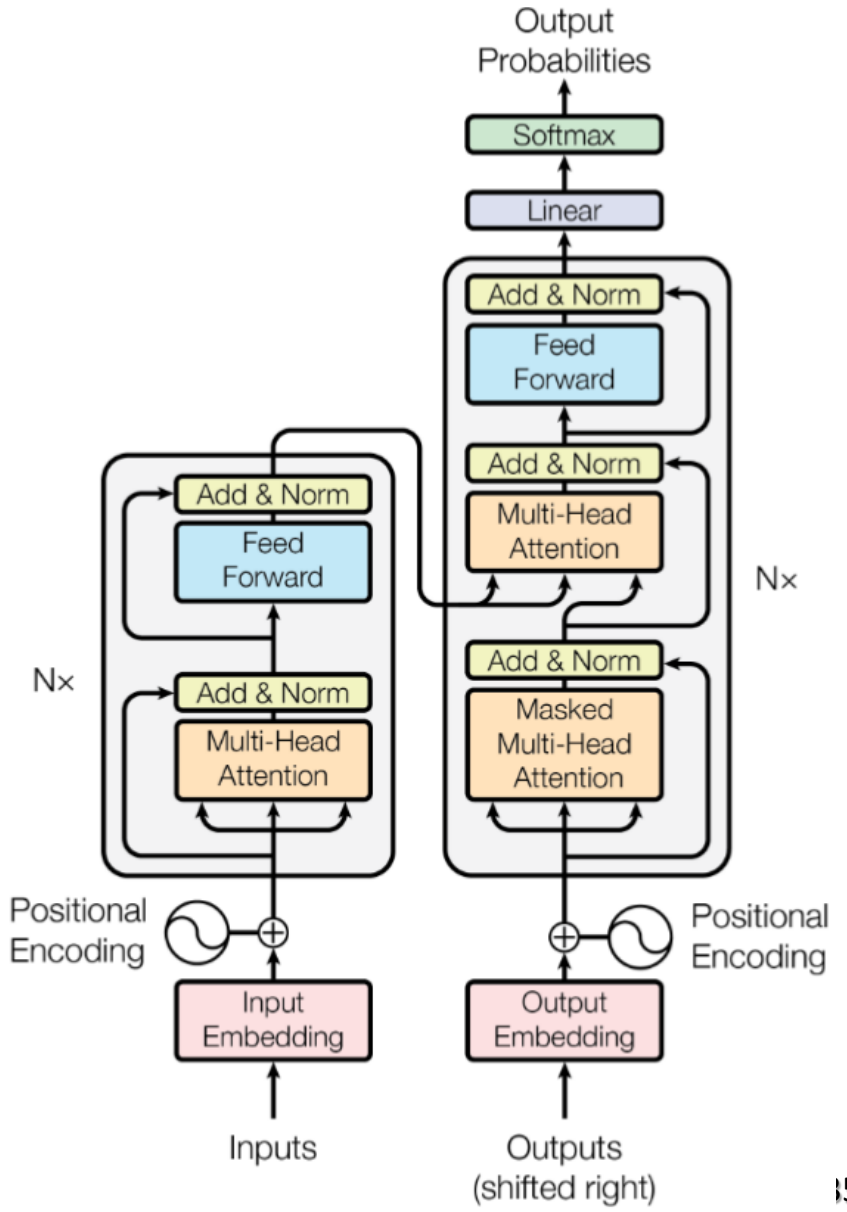
Decoder

Queries from **Decoder Inputs**



Putting together

- $p(y_t|x_1, x_2, \dots, x_T, y_1, \dots, y_{t-1}) =$
  $\mathrm{Dec}(\mathrm{Enc}(x_1, x_2, \dots, x_T), y_1, \dots, y_{t-1})$



Question

- What's the training objective?

Text Data: Transformers

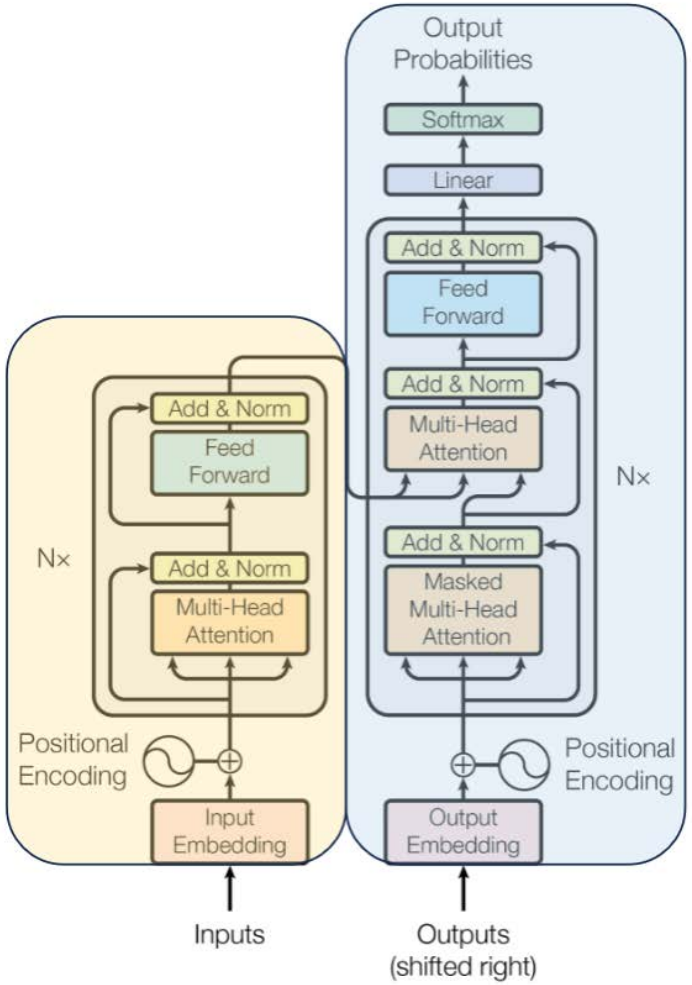
- Introduction
- Attention is all you need
- Pretraining: BERT and GPT
- Summary

The Era of LLMs

- Pre-training models with large corpus

BERT
Oct 2018

Representation



GPT
Jun 2018

Generation

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

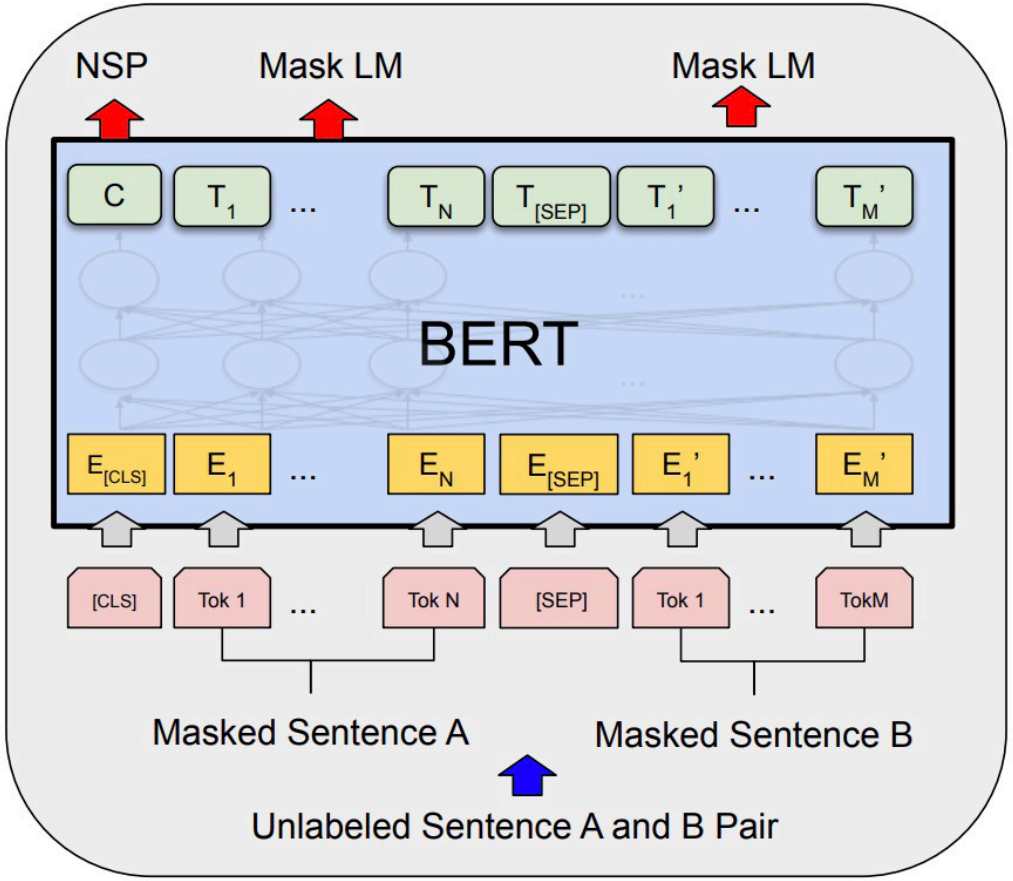
We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language repre-

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that

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

BERT: Bidirectional Encoder Representations from Transformer

- BERT Pre-Training Corpus:
  - English Wikipedia - 2,500 million words
  - Book Corpus - 800 million words
- BERT Pre-Training Tasks:
  - MLM (Masked Language Modeling)
  - NSP (Next Sentence Prediction)
- BERT Pre-Training Results:
  - BERT-Base – 110M Params
  - BERT-Large – 340M Params



Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large

GPT: Generative Pre-Training

- GPT Pre-Training Corpus:
  - Books Corpus and English Wikipedia
- GPT Pre-Training Tasks:
  - Predict the next token, given the previous tokens
  - More learning signals than MLM
- GPT Pre-Training Results:
  - GPT – 117M Params

Architecture

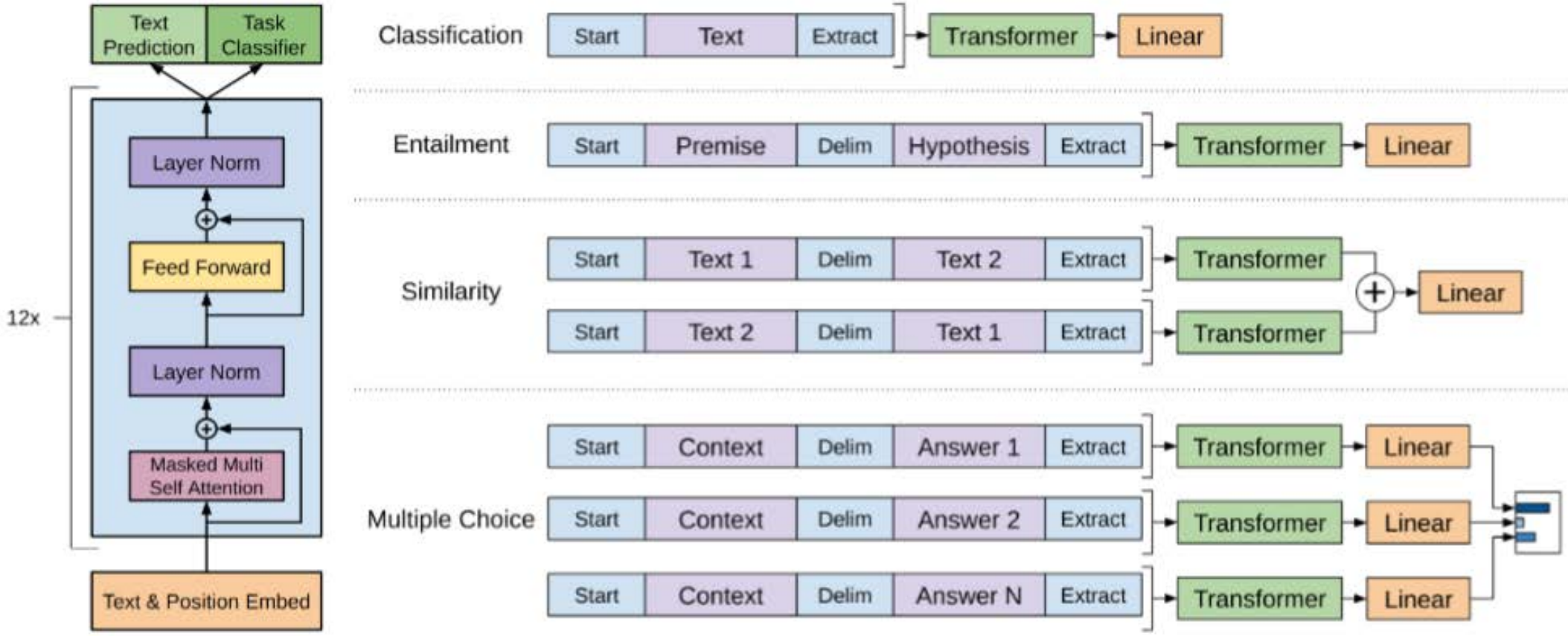
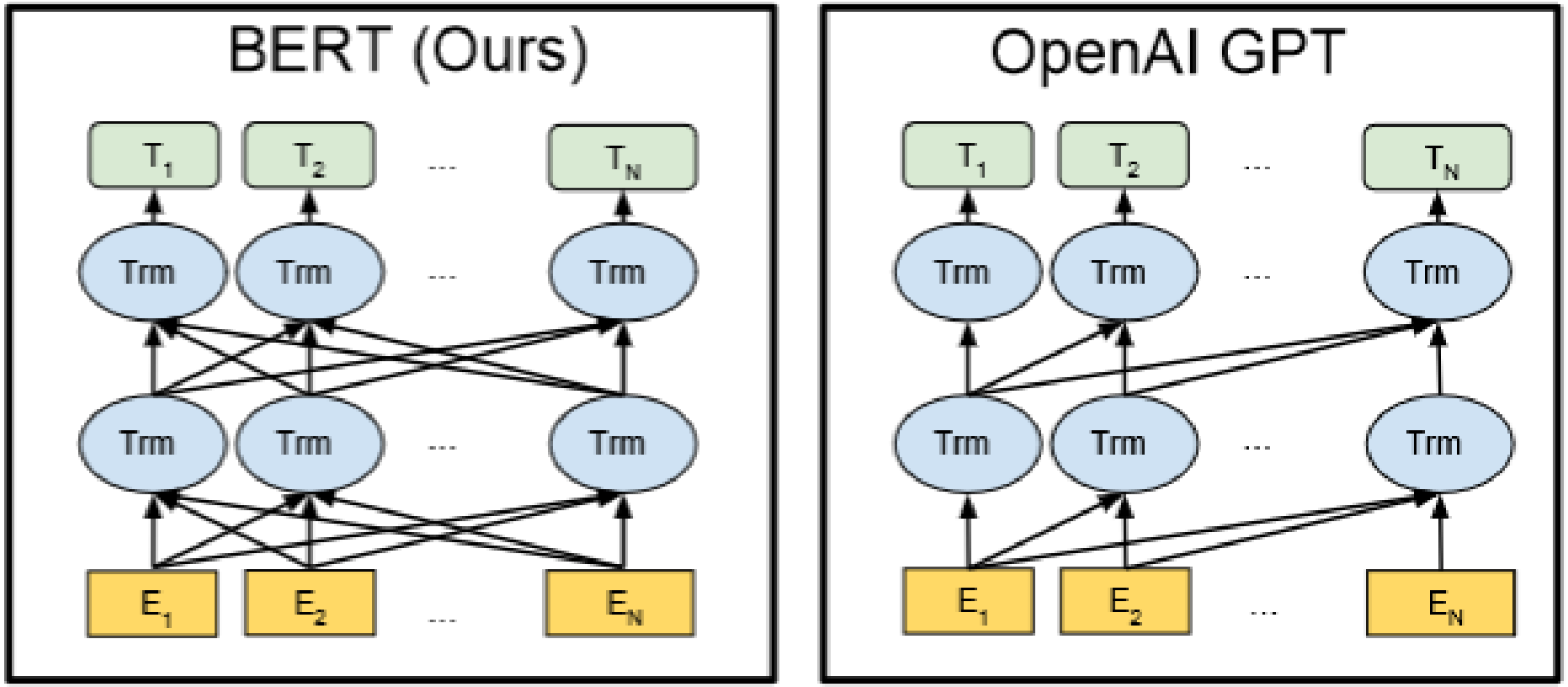


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

BERT Vs. GPT

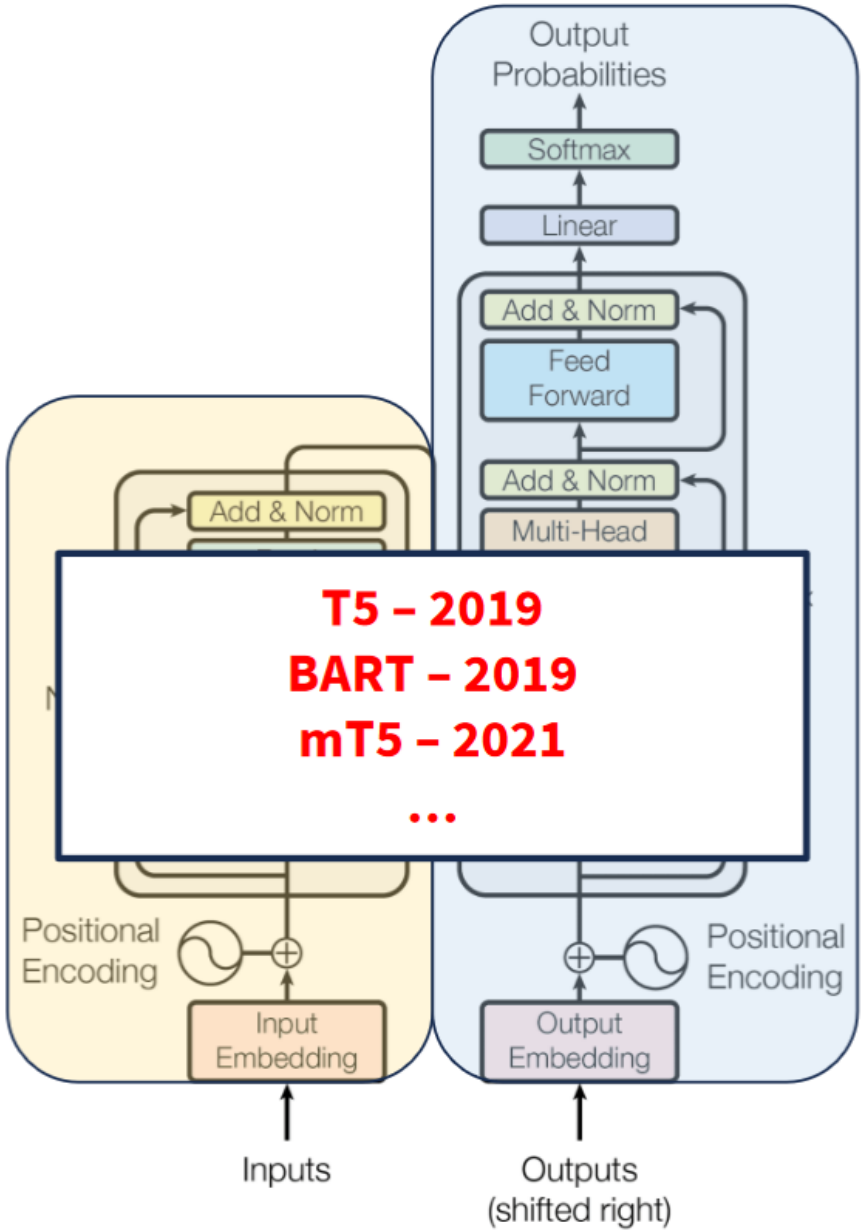
- From BERT paper



The LLM Era – Paradigm Shift in Machine Learning

BERT – 2018
DistilBERT – 2019
RoBERTa – 2019
ALBERT – 2019
ELECTRA – 2020
DeBERTa – 2020
…

Representation



GPT – 2018
GPT-2 – 2019
GPT-3 – 2020
GPT-Neo – 2021
GPT-3.5 (ChatGPT) – 2022
LLaMA – 2023
GPT-4 – 2023
…

Generation

Text Data: Transformers

- Introduction
- Attention is all you need
- Pretraining: BERT and GPT
- Summary

Summary

- Transformer
  - Encoder-Decoder Framework
  - Attention
- LLMs
  - Encoder-only pretraining: BERT
  - Decoder-only pretraining: GPT

References

- Course: https://deeplearning.cs.cmu.edu/F23/document/slides/lec19.transformersLLMs.pdf
- A detailed introduction of Attention and Transformers: https://d2l.ai/chapter_attention-mechanisms-and-transformers/index.html
- Interactive visualization of encoder-only models (BERT): https://colab.research.google.com/drive/1hXIQ77A4TYS4y3UthWF-Ci7V7vVUoxmQ?usp=sharing