Introduction to Nature Language Processing

2nd lecture

Boston University

CS 640, AI

Slides by Margrit Betke, Yiwen Gu

Outline (Recap)

- What is NLP?
- Key NLP tasks
  - Text Classification, NER, MT, Sentiment Analysis, QA, Summarization

- NLP Techniques and Approaches
  - Traditional techs,
  - RNNs
  - **Attention Mechanism & Transformers**

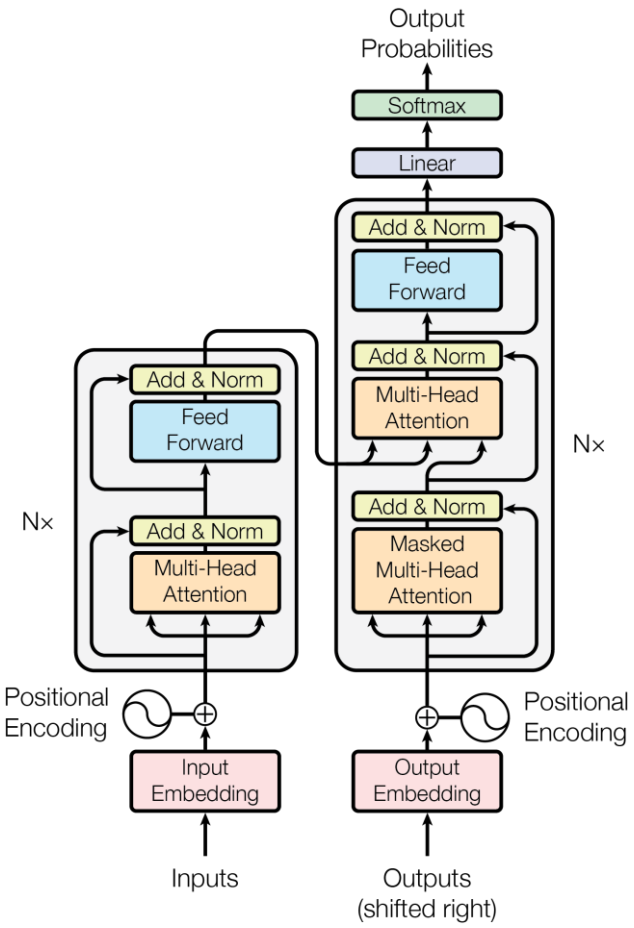
RNNs: Limitations

- **Difficulty in Capturing Long-Term Dependencies**: Even with improvements like LSTMs and GRUs, RNNs can struggle with very long sequences.
- **Sequential Computation:** RNNs process one time step at a time, which makes training slower, especially for long sequences.

→ **Transformers**: RNNs have been largely supplanted by **transformers**, which use **self-attention mechanisms** to process entire sequences **in parallel**, making them more efficient and better at capturing long-range dependencies.

Transformers

Transformers



Attention Is All You Need

Ashish et al., 2017

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

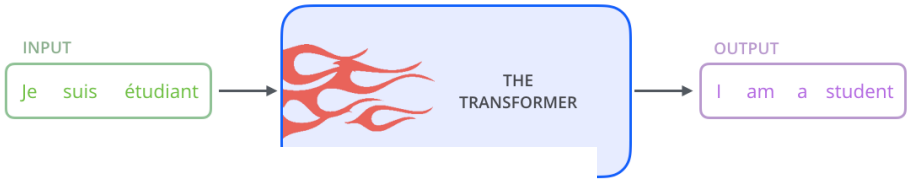
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

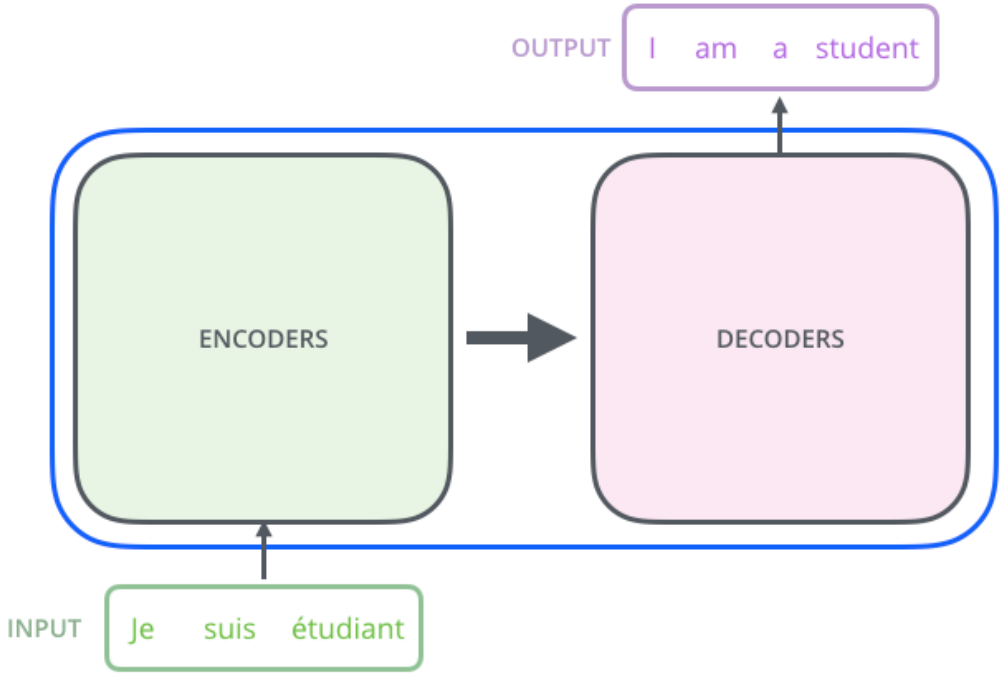
Example: Translate the French to English

Visualization Credit to Jay Alammar

Example: Translate the French to English

Some transformers consist of decoders only, such as GPTs.

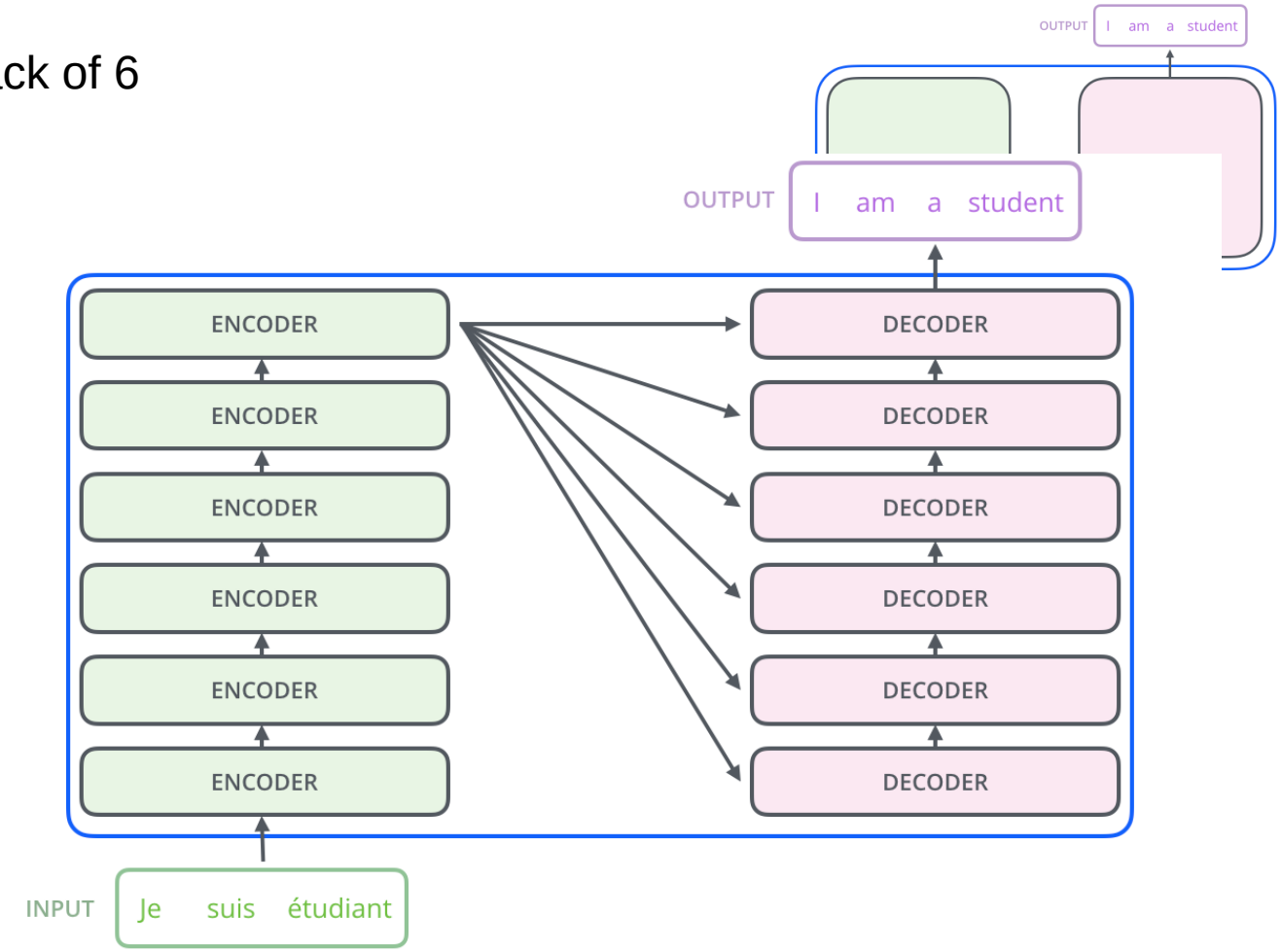




Visualization Credit to Jay Alammar

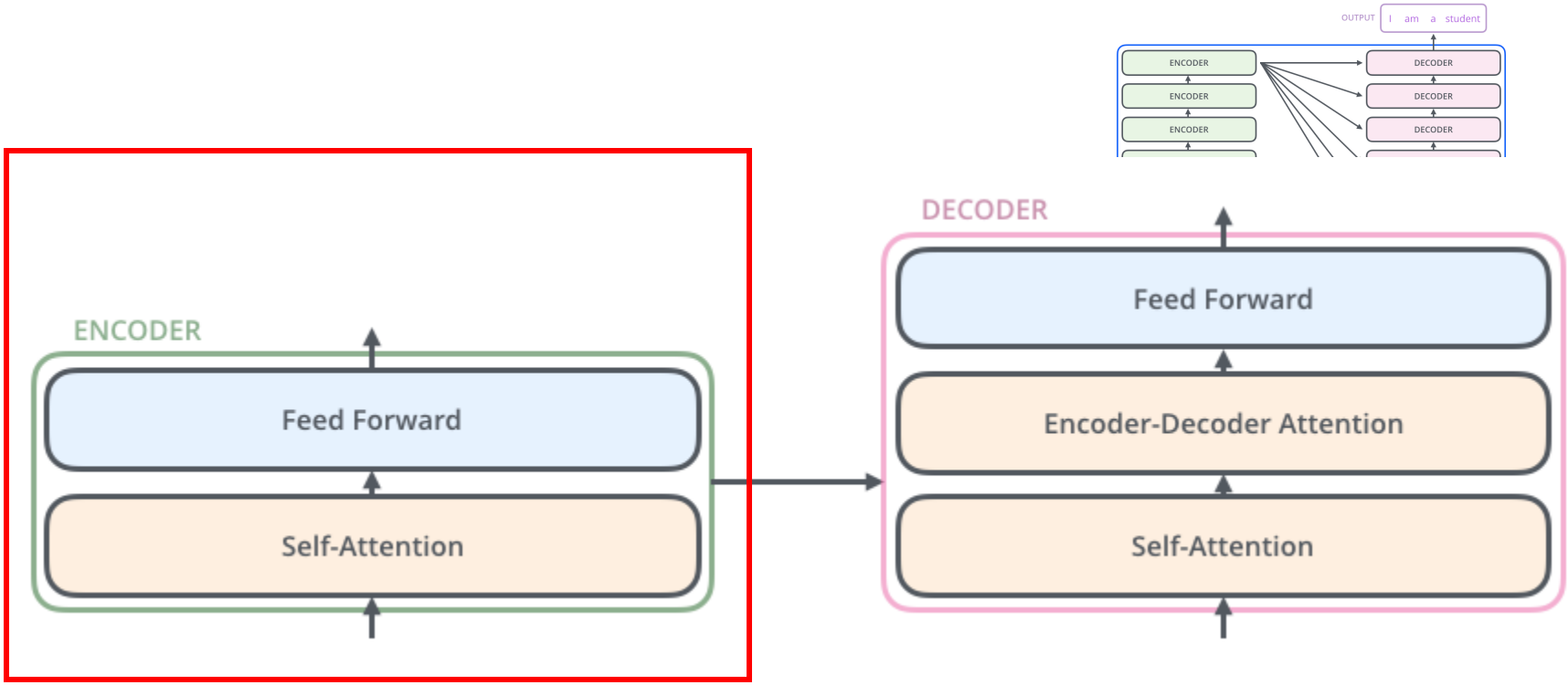
Example: Translate the French to English

Both: a stack of 6



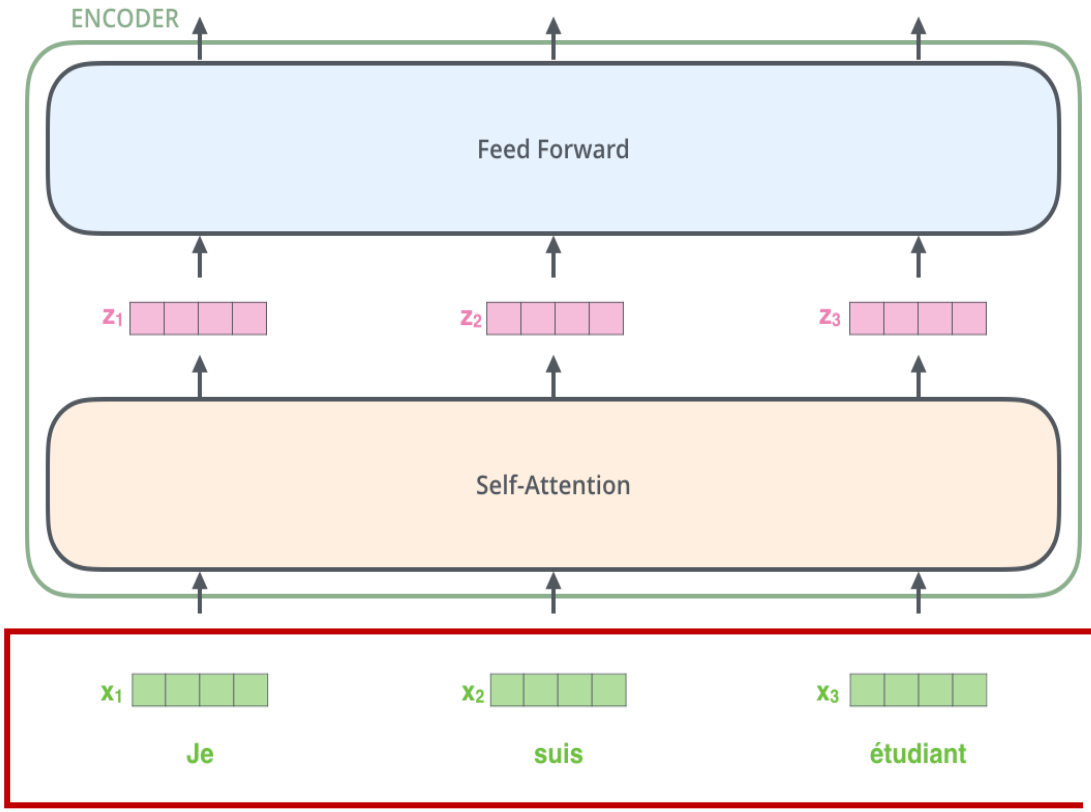
Visualization Credit to Jay Alammar

Example: Translate the French to English



Visualization Credit to Jay Alammar

Example: Translate the French to English



Encoder #1

Each word is embedded into a vector of size 512

Visualization Credit to Jay Alammar

Attention Mechanism

Attention(Q,K,V) = softmax(Q K^T) V

Q = query vector = current English (or French) word

K key and V value = memory of words seen before

Goal: Find key(s) most similar to query and retrieve value(s) that correspond to this/these key(s)

Softmax = $\sum_i e^{qki} / \sum_j e^{q\,kj}$ v_i produces probability distribution over keys with peaks for keys similar to query

Attention Mechanism

Attention(Q,K,V) = softmax(Q K^T) V

Acts as a weight mask over V

Q = query vector = current English (or French) word

K key and V value = memory of words seen before

Goal: Find key(s) most similar to query and retrieve value(s) that correspond to this/these key(s)

Softmax = $\sum_i e^{qki} / \sum_j e^{q\,kj}$ v_i produces probability distribution over keys with peaks for keys similar to query

Attention Mechanism

Very fast:
2 matrix multiplications & 1 softmax operation

Attention(Q,K,V) = softmax(Q K^T) V

Acts as a weight mask over V

Q = query vector = current English (or French) word

K key and V value = memory of words seen before

Goal: Find key(s) most similar to query and retrieve value(s) that correspond to this/these key(s)

Softmax = $\sum_i e^{qki} / \sum_j e^{q\,kj}$ v_i produces probability distribution over keys with peaks for keys similar to query

Attention Mechanism

Very fast:
2 matrix multiplications & 1 softmax operation

Attention(Q,K,V) = softmax(Q K^T/ sqrt(d_k)) V

Technical detail:
sqrt(d_k) normalization needed for training

Acts as a weight mask over V

Q = query vector = current English (or French) word

K key and V value = memory of words seen before

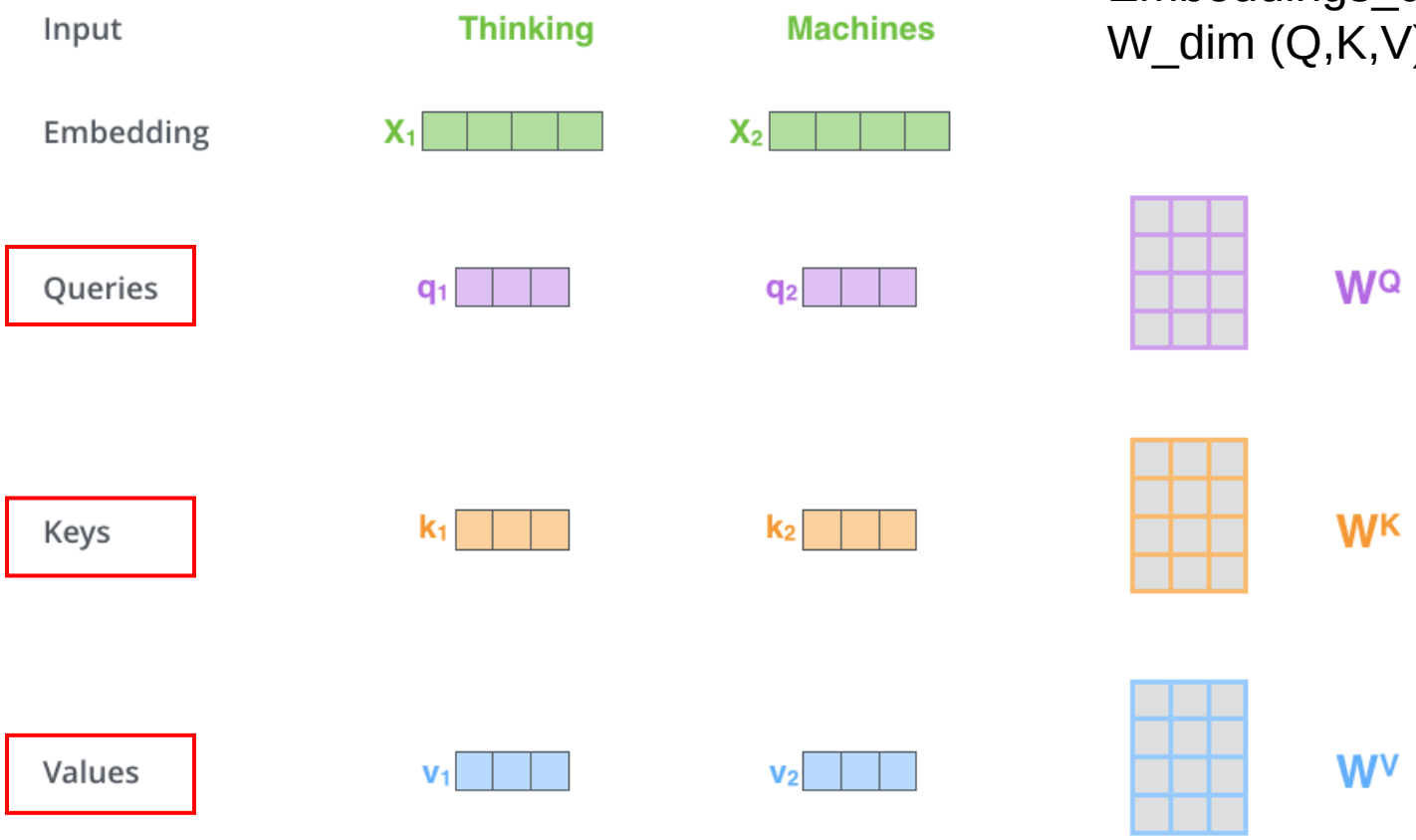
Goal: Find key(s) most similar to query and retrieve value(s) that correspond to this/these key(s)

Softmax = $\sum_i e^{qki} / \sum_j e^{q\,kj}$ v_i produces probability distribution over keys with peaks for keys similar to query

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

How to get Q,K,V?

Embeddings_dim: 512
W_dim (Q,K,V): 512 x 64

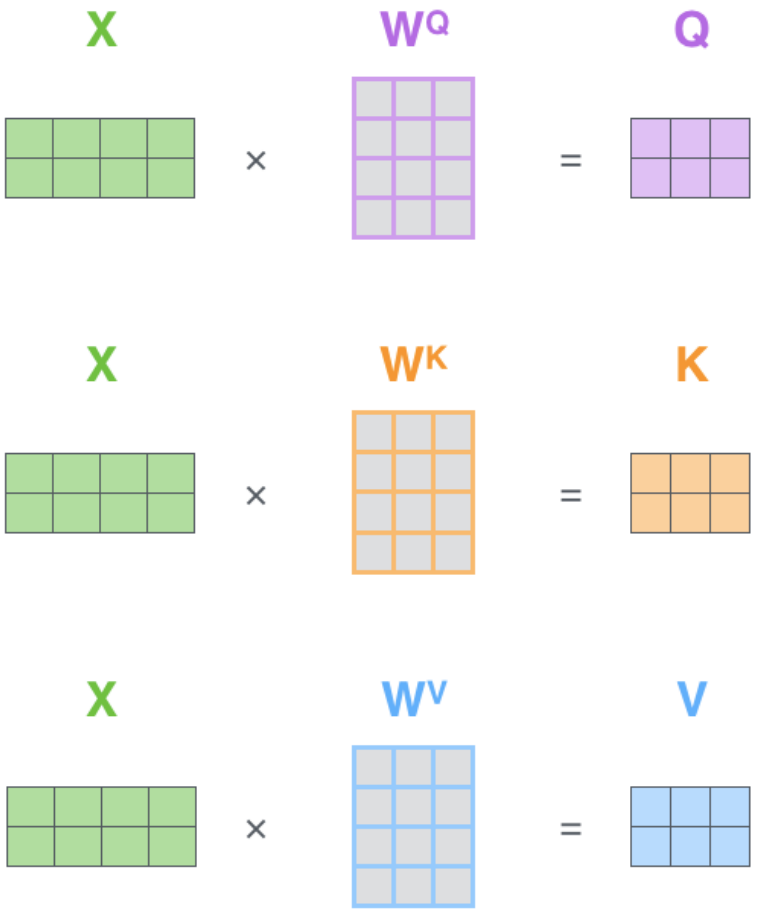


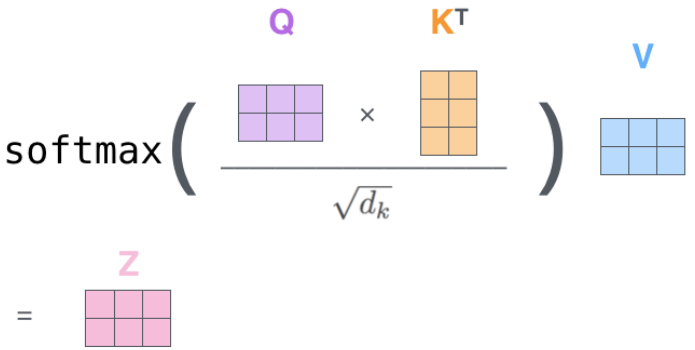
Visualization Credit to Jay Alammar

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

How to get Q,K,V?

In the Matrix Form





Visualization Credit to Jay Alammar

Multi-Head Self-Attention

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

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Question:

Known:
Number of heads: h=8,
Model dim = Embedding dim: d=512,
Weight matrices for K, Q, V are in the shape of 512 x 64,

What is the shape of W^o?

d_k

64*8 x 512
See explanation on the next slide

Multi-Head Self-Attention

1) This is our input sentence*

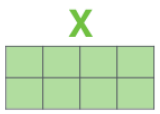
2) We embed each word*

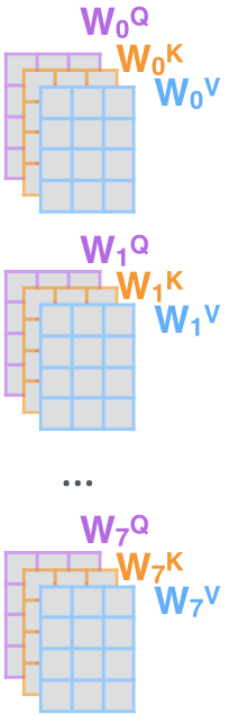
3) Split into 8 heads. We multiply X or R with weight matrices

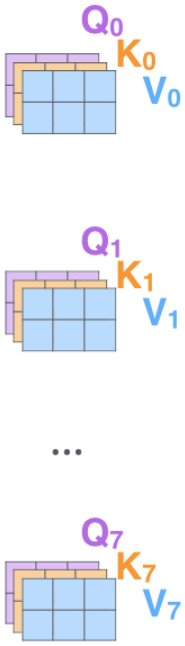
4) Calculate attention using the resulting Q/K/V matrices

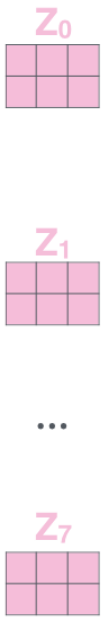
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

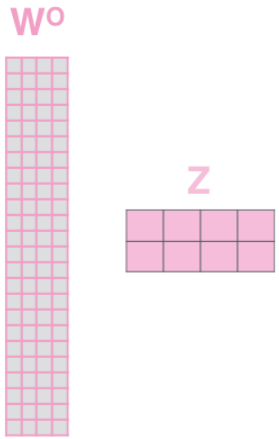
Thinking Machines











* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

In this example,
n = 2 (not relevant to W^o),
h = 8
d = 4
d_k =3
The shape of the W^o is drawn accordingly, i.e. (3x8, 4)

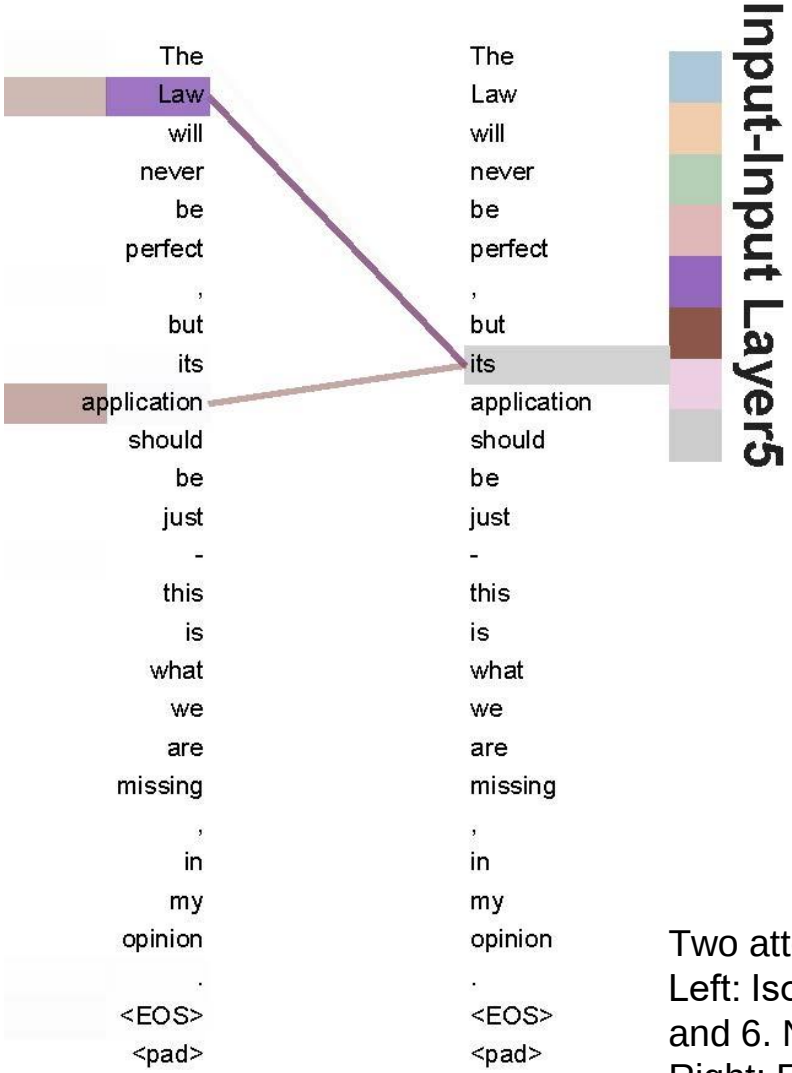
Visualization Credit to Jay Alammar

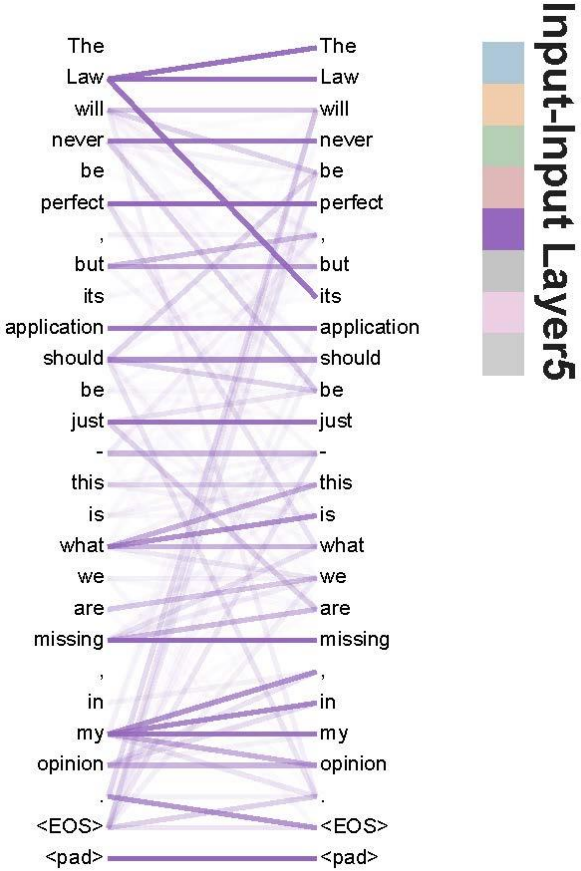
Why Multi-Head Attention?

- Multiple attention layers (heads) in paraellel
- Each head uses different linear transformation
- Different heads can learn different relationships

Source: Attention is all you need

Visualization of Attentions





Two attention heads in layer 5 of 6, involved in **anaphora resolution**.
Left: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word..
Right: Full attentions for head 5.

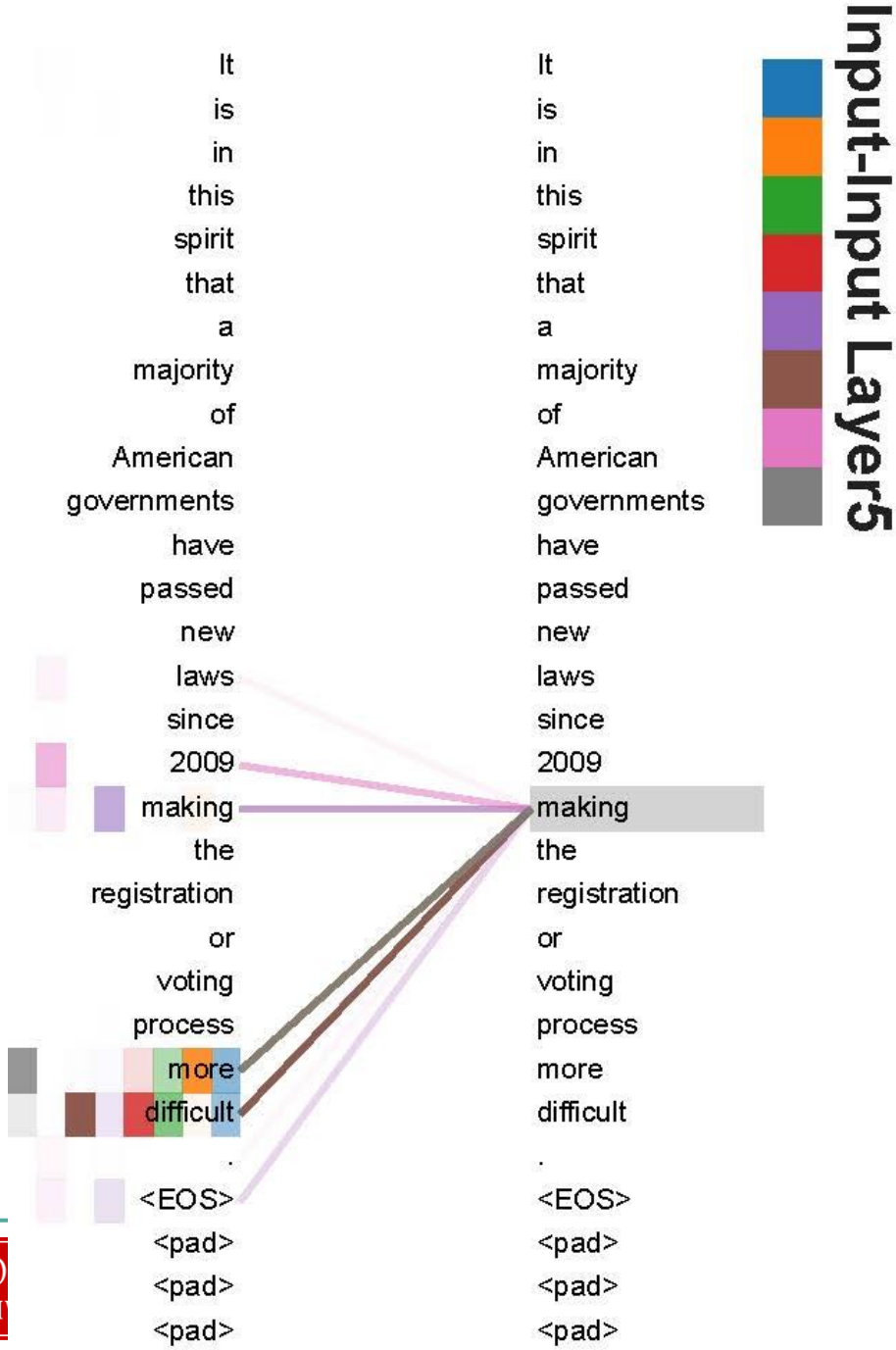
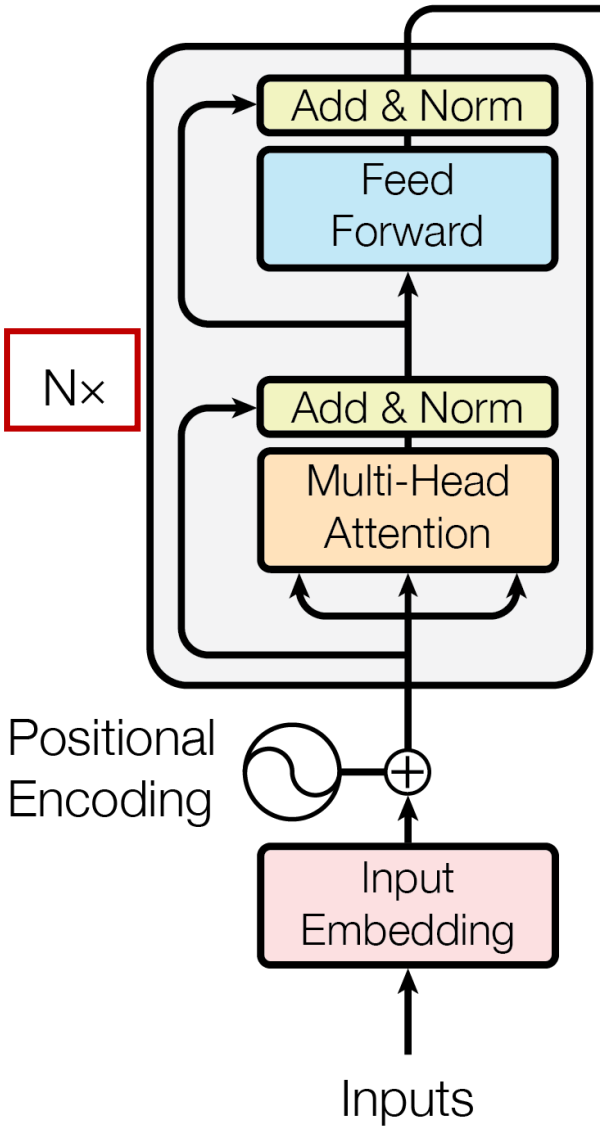


Figure 3: An example of the attention mechanism following **long-distance dependencies** in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’. Different colors represent different heads. Best viewed in color.

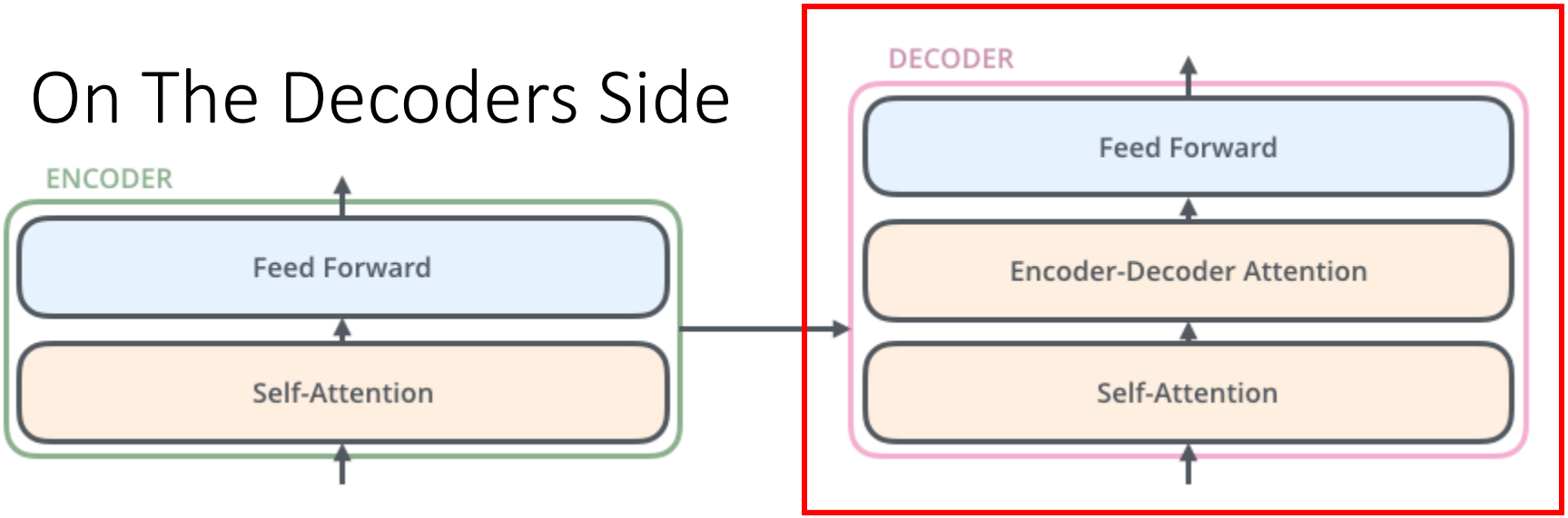


- Feed Forward:
  - Position-wise Feed-Forward Networks, process every word

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

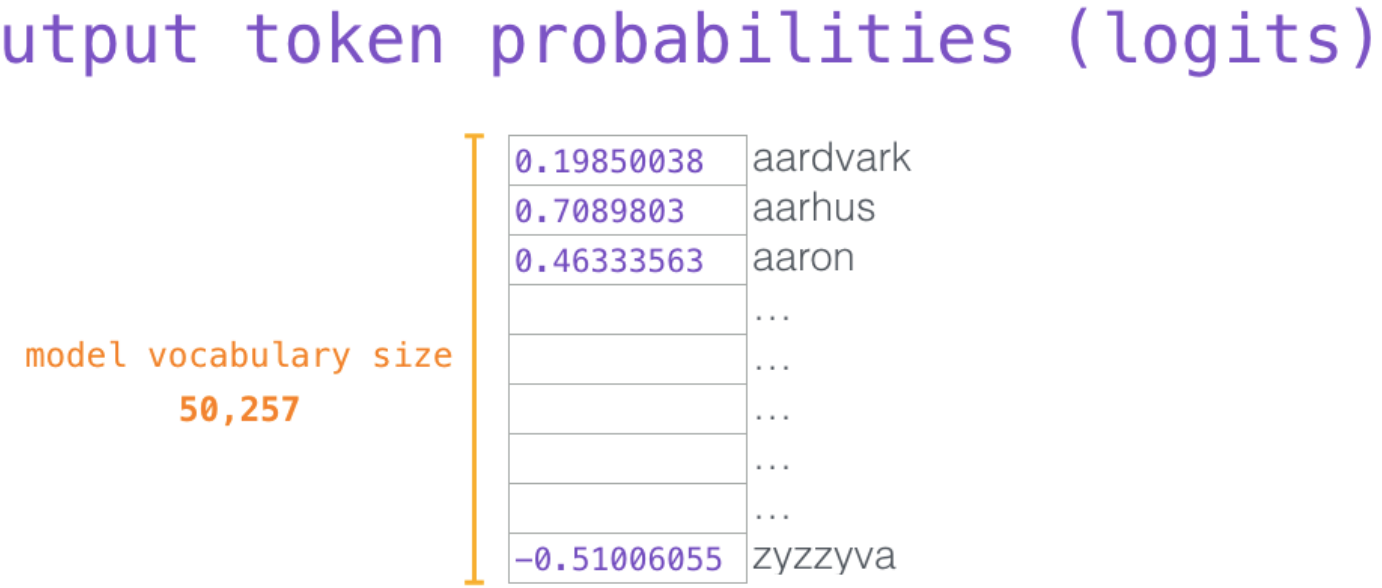
  - Where facts in LLM live (3Blue1Brown)
- Add & Norm:
  - Residual Connection and LayerNorm
- N x: Stacking
  - Stacking these attention models also allows for higher level reasoning
  - Lower layers focus more on word relationships and **syntax**, whereas higher layers more on contextual relationships and **semantics**

On The Decoders Side



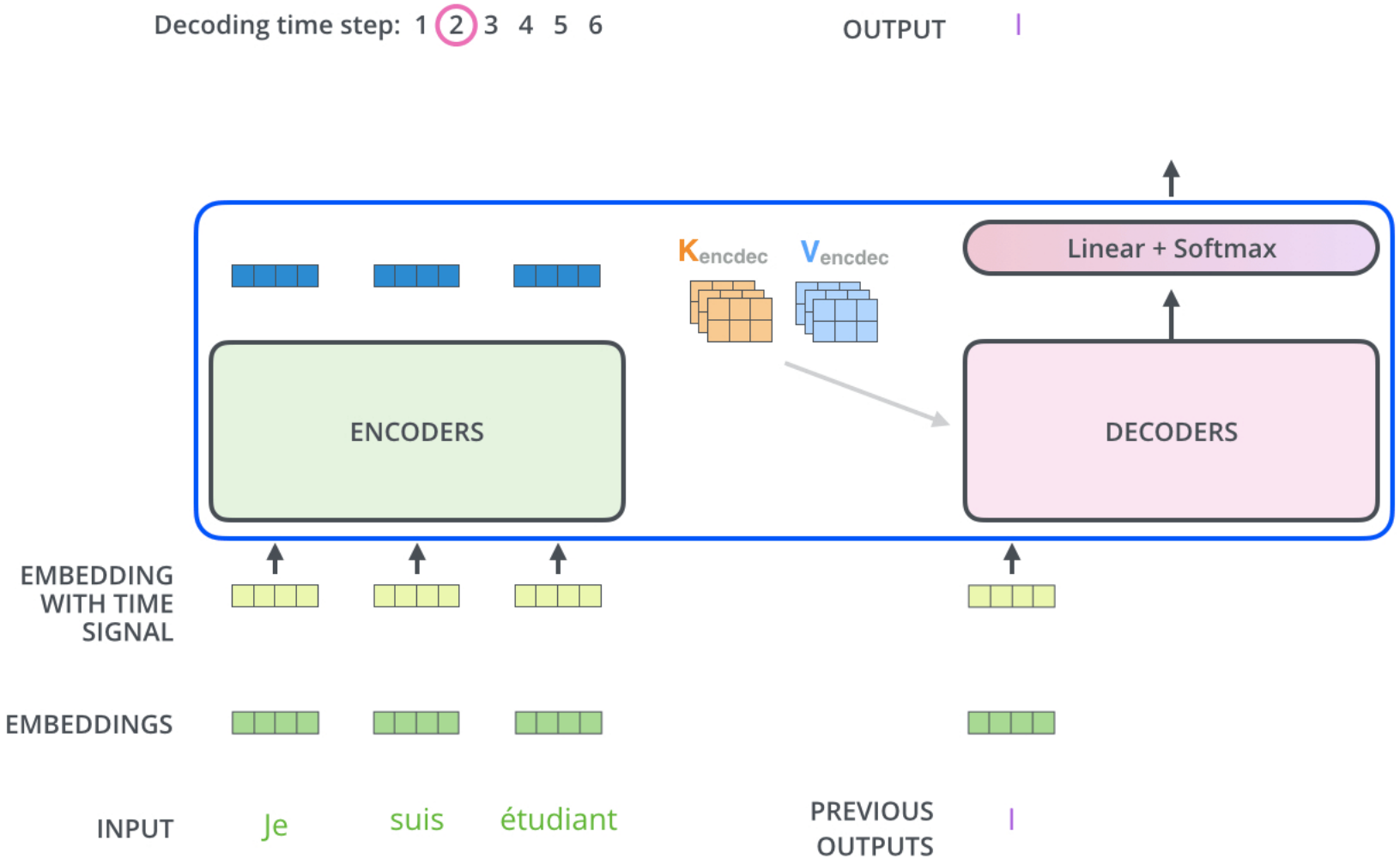
- Encoder-Decoder Attention (**Cross-Attention)**
  - This cross-attention allows the decoder to attend to the output of the encoder, combining information from both the input sequence (processed by the encoder) and the current decoder states.
  - Used in Visual Transformers (ViTs) → Later
- **Masked** Self-Attention:
  - Each token can **only** attend to previous tokens in the sequence.
  - This prevents the decoder from "cheating" by looking at tokens it hasn't generated yet, ensuring autoregressive generation.

Last Step
linear + softmax, (sampling)



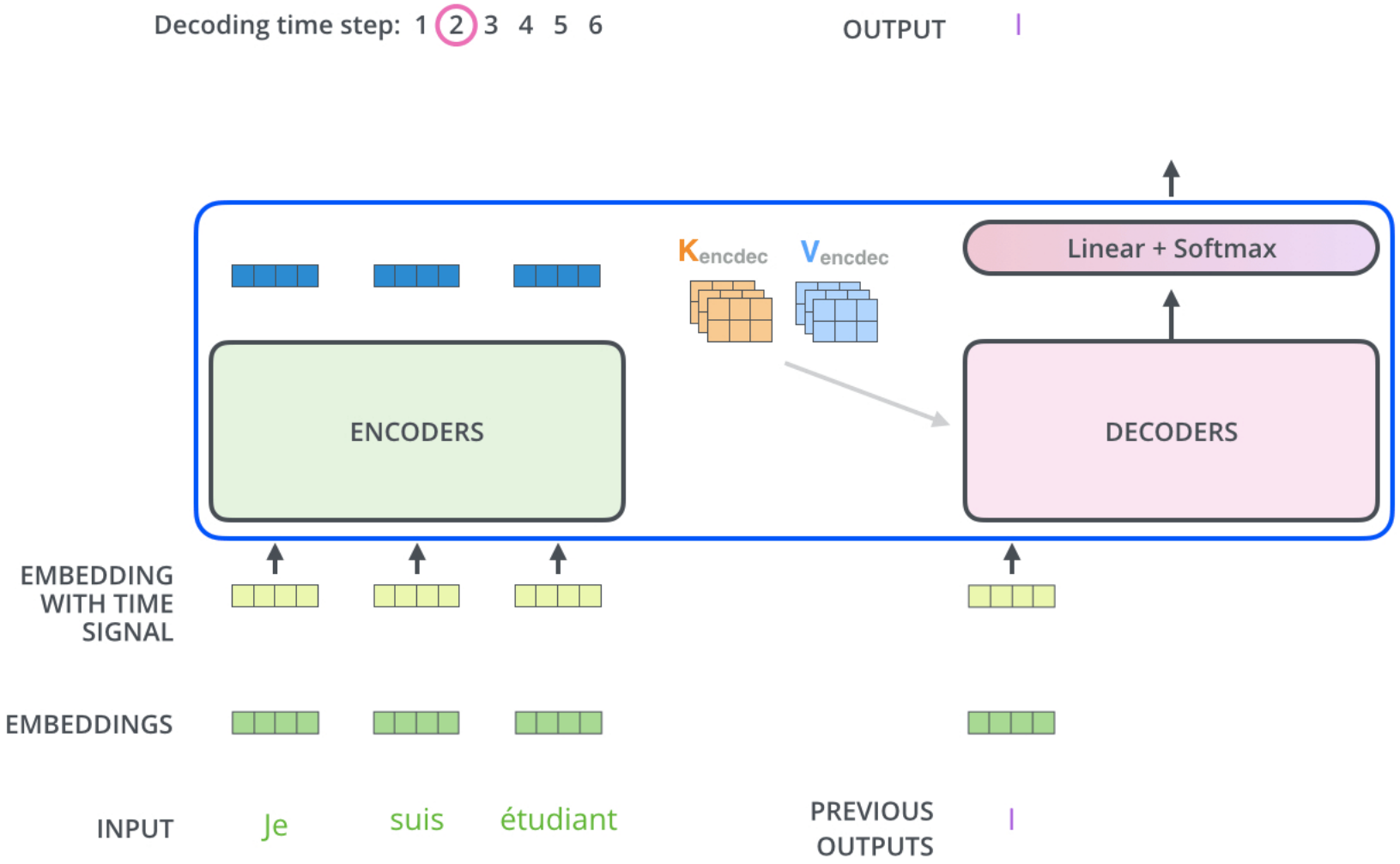
Visualization Credit to Jay Alammar

Putting Together



Visualization Credit to Jay Alammar

Putting Together



Visualization Credit to Jay Alammar

Putting Together

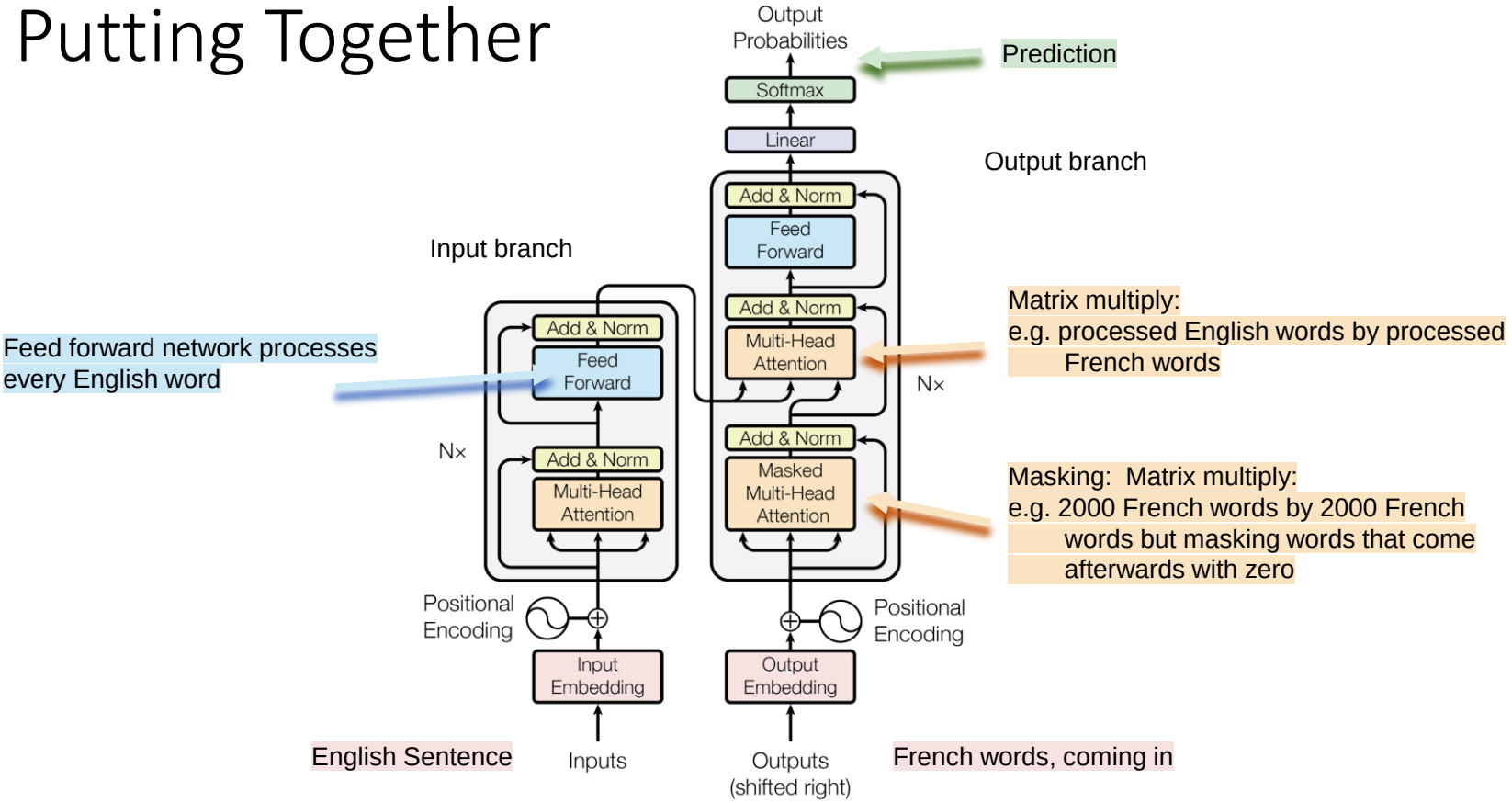


Figure 1: The Transformer - model architecture.

Note:
The training dataset (WMT) consists of parallel corpora from multiple languages. This means **that the source and the target are paired in the sentence level during the training**. In this way, the source sentence is passed through the encoder to get some representation; the target sentence is used during training to guide the decoder in generating the correct translation, with the model learning to minimize the difference between its output and the actual target sentence. During inference (when the model is actually translating sth.), only the source sentence is provided. And the model generates the target sentence word by word, based on its learned patterns from the paired training data.

Transformer Architecture Complexity (per layer)

- n= number of words in sequence
- d= network dimension

Number of operations: $n^2 d$

Number of activations: $n^2 + n d$

Much better than RNNs with number of operations $n d^2$

Transformer Architecture Complexity (per layer)

- n= number of words in sequence (<70 words per sentence)
- d= network dimension (easily go beyond 1000 in a transformer)

Every word attends to every word

To get the dot product of Q, K (both of size **d/h**), then multiply by h for multi-head

Number of operations: $n^2 d$ e.g., 70x70x1000=4.9 mill

Number of activations: $n^2 + n d$ From FFN, ReLU

From self-attention, softmax

Much better than RNNs with number of operations $n d^2$

e.g., 70x1000x1000=70 mill

Training a Transformer

- ADAM optimizer
- Dropout during training at every layer
- Label smoothing
- Auto-regressive decoding with beam-search
- Checkpoint-averaging

- Library available: https://github.com/tensorflow/tensor2tensor

How much data to train?

All of it…

All text on the internet?

Is that legal?

AI & Ethics!

All text on the internet?

Is that legal?

AI & Ethics!

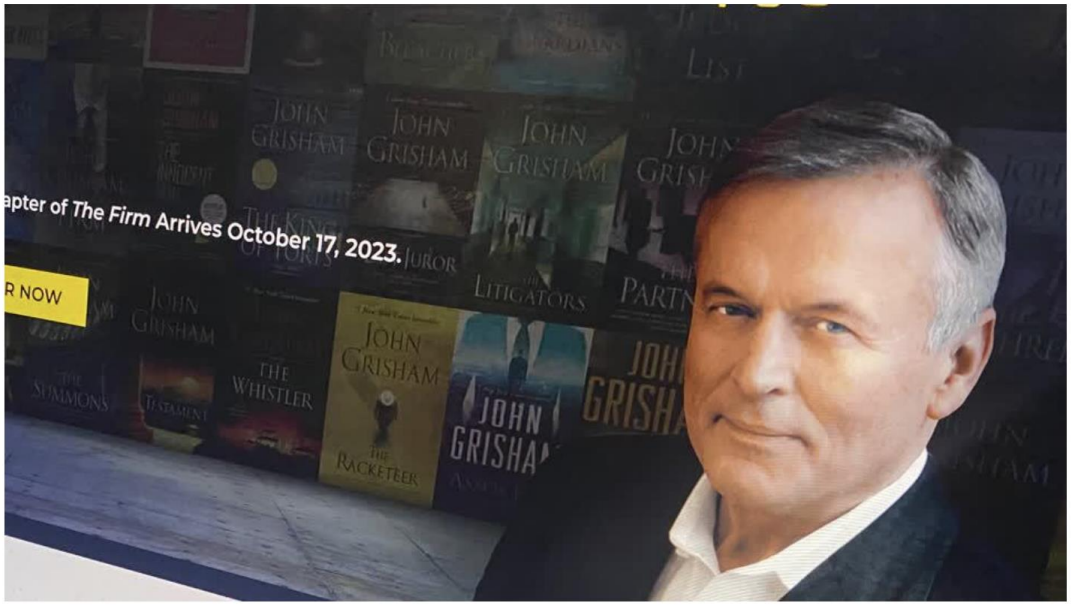
REUTERS® World Business Markets Sustainability Legal Breakingviews Technology Investig

Litigation | Copyright | Litigation | Technology | Intellectual Property

John Grisham, other top US authors sue OpenAI over copyrights

By **Blake Brittain**

September 21, 2023 6:34 AM EDT · Updated 7 months ago



All text on the internet?

Is that legal?

AI & Ethics!

December 27, 2023

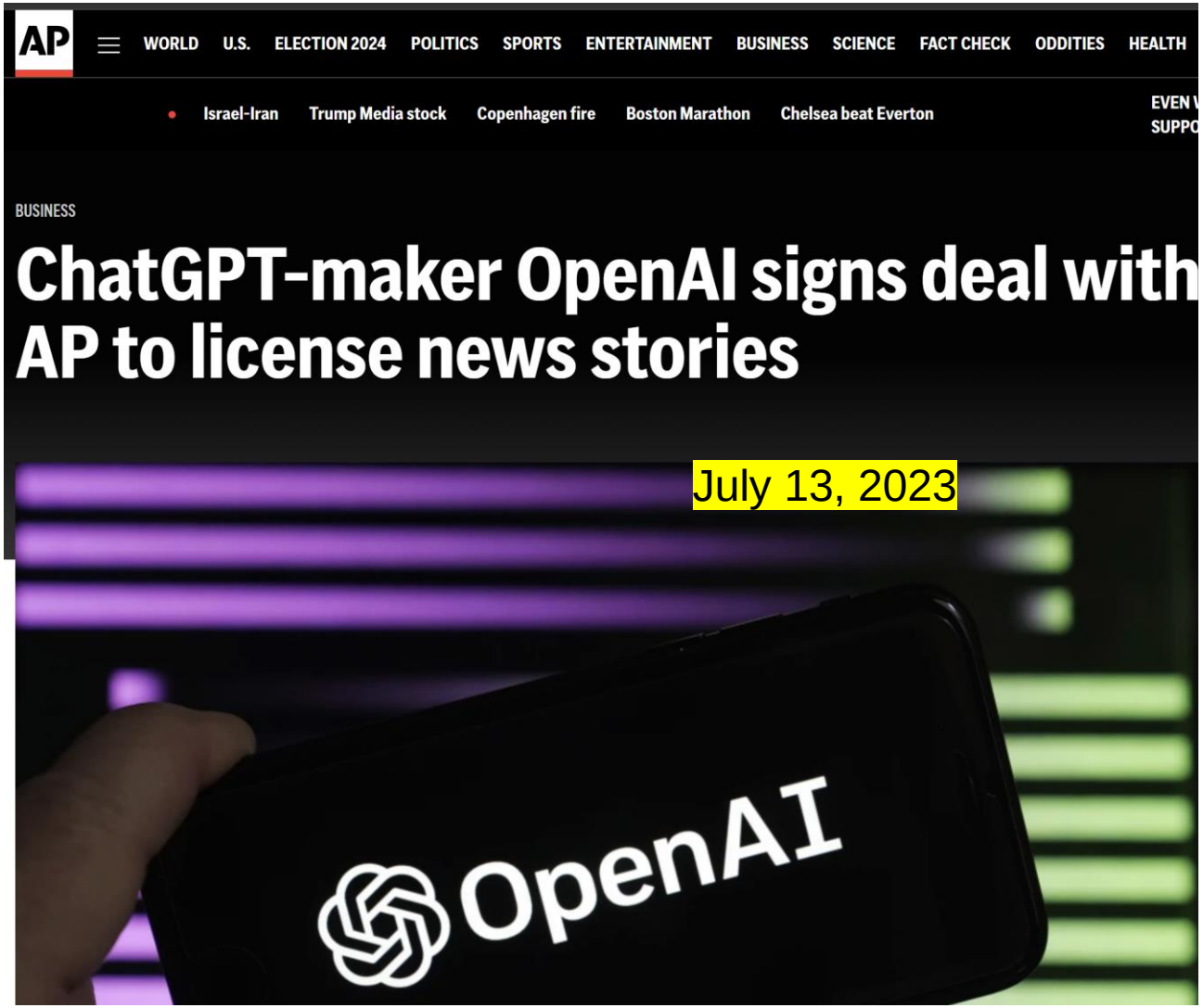
The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work

Millions of articles from The New York Times were used to train chatbots that now compete with it, the lawsuit said.

All text on the internet?

Is that legal?

AI & Ethics!



96 layers (decoder blocks)
2048 tokens

Training the 175 billion parameters of GPT-3 on "all text on the internet" on a single GPU or computer would take 355 years and $4,600,000

Lambdalabs.com

How long did it take OpenAI to train GPT-3?

a month

What did OpenAI train on?

GPT-3 training data[1]:9

Dataset	# tokens	Proportion within training
Common Crawl	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

Source: Wikipedia

What about GPT-4?

- 1 trillion parameters
- Sam Altman stated that the cost of training GPT-4 was more than $100 million.

Source: Wikipedia

What about GPT-4?

- 1 trillion parameters
- Sam Altman stated that the cost of training GPT-4 was more than $100 million.

Why are the lawsuits so costly?

- We don't know how to "untrain" neural networks.
- "Unlearning" is an exciting research area!

Source: Wikipedia

Learning Outcomes

- Understand how transformers work
  - Encoder-Decoder (e.g. THE Transformer, BERT)
  - Decoder Only (e.g. GPTs)
  - Attention Mechanism
    - self-attention,
    - Masked self-attention,
    - cross-attention