

# Modern NLP Models

Prof. Kuan-Ting Lai

2022/5/16



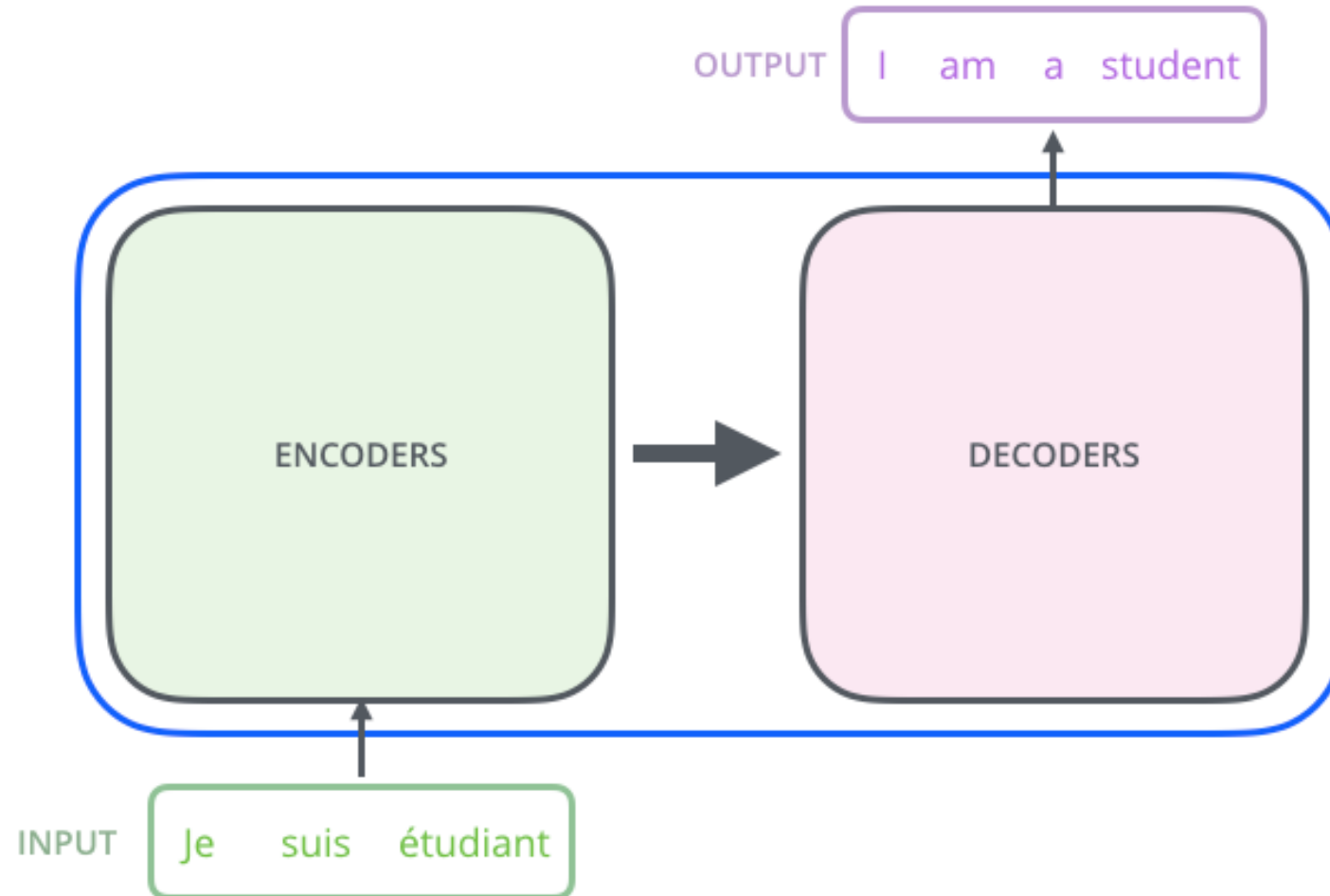
# Transformer Illustration

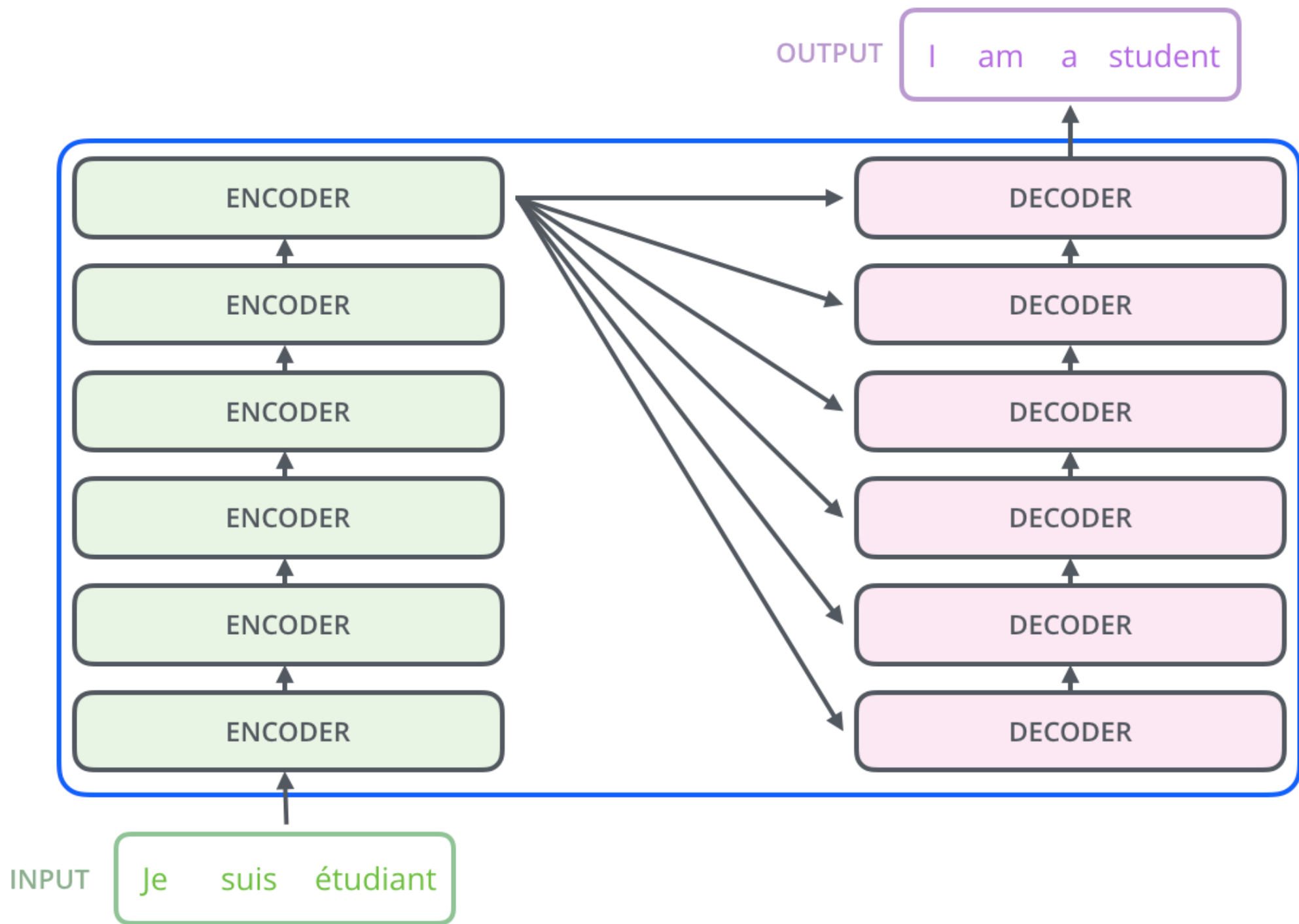


- <http://jalammar.github.io/illustrated-transformer/>



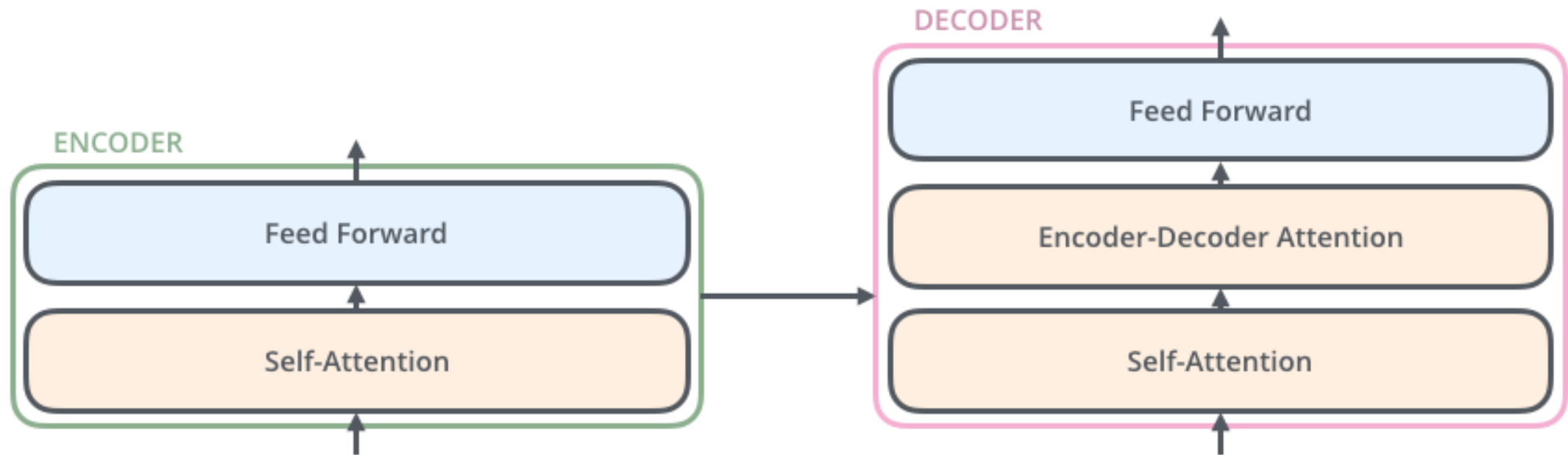
# Encoder and Decoder



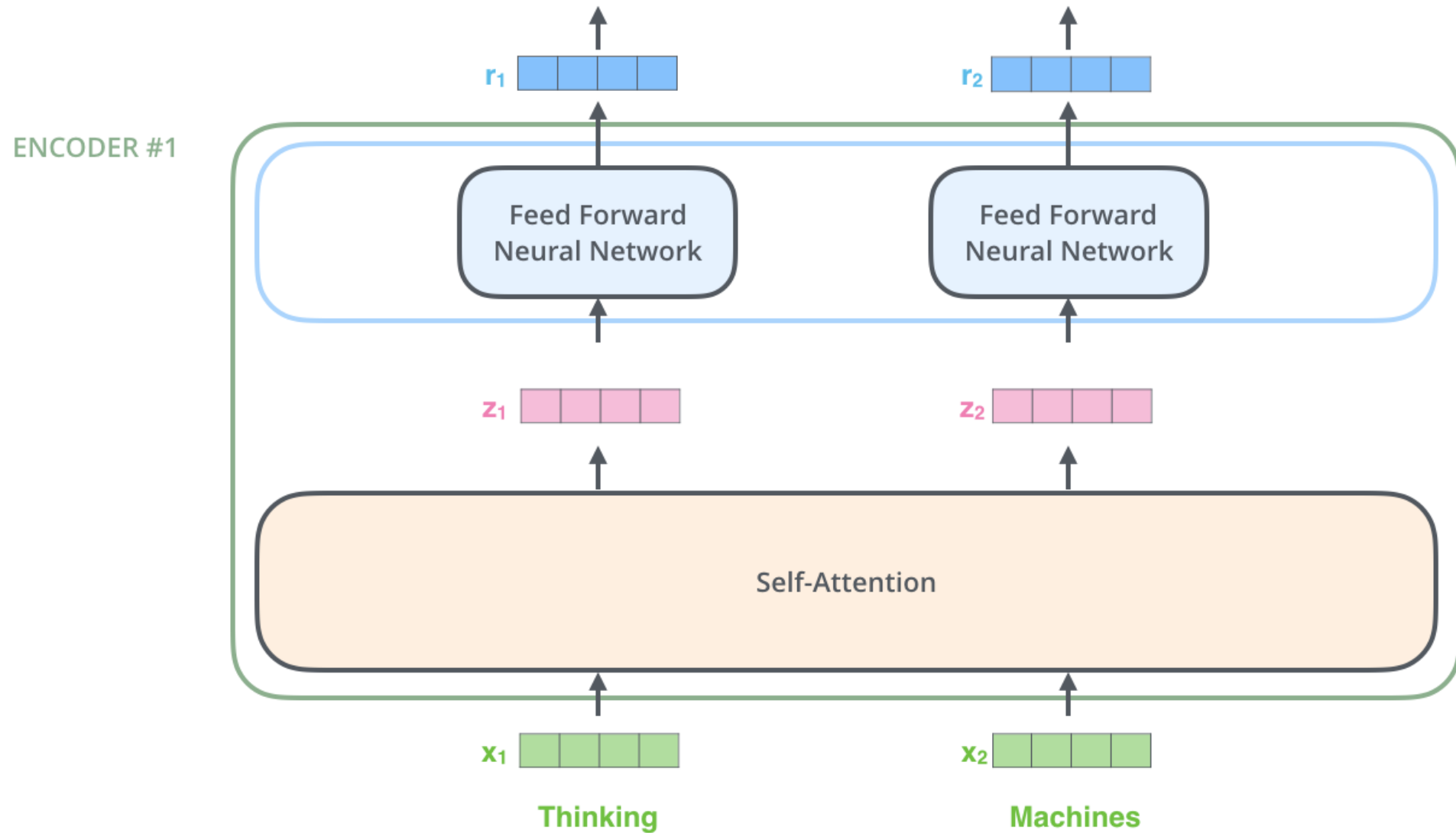


# Structure of the Encoder and Decoder

- Self-attention
- Encoder-decoder attention



# Start Encoding





# Calculate Values

- Calculate dot product of key and value vector
- Multiply each value vector by the Softmax score
- Sum up the weighted value vectors  $v_1$  and  $v_2$

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

Input

Embedding

Queries

Keys

Values

Score

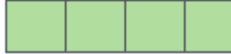
Divide by 8 (  $\sqrt{d_k}$  )

Softmax

Softmax  
X  
Value

Sum

Thinking

$x_1$  

$q_1$  

$k_1$  

$v_1$  

$q_1 \cdot k_1 = 112$

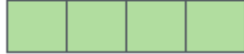
14

0.88

$v_1$  

$z_1$  

Machines

$x_2$  

$q_2$  

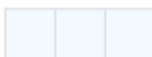
$k_2$  

$v_2$  

$q_1 \cdot k_2 = 96$

12

0.12

$v_2$  

$z_2$  

# Final Output of the Self-attention Module

$$\text{softmax} \left( \frac{\overset{\text{Q}}{\begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array}} \times \overset{\text{K}^T}{\begin{array}{|c|c|} \hline & \\ \hline & \\ \hline & \\ \hline \end{array}} \right) \overset{\text{V}}{\begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array}}$$

=

$\overset{\text{Z}}{\begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array}}$





# The Beast with Multiple Heads

1) This is our input sentence\*

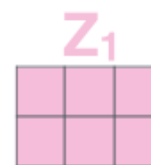
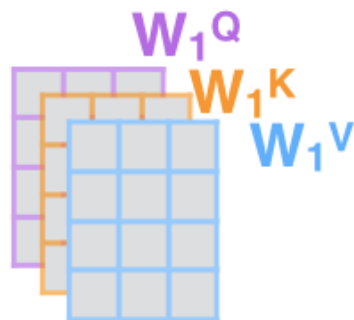
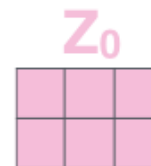
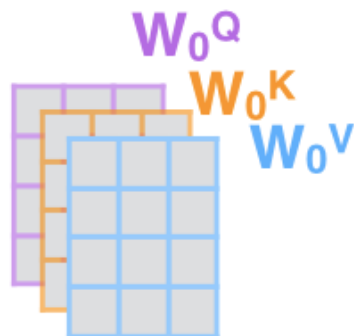
2) We embed each word\*

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

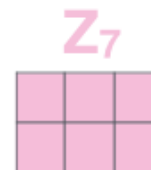
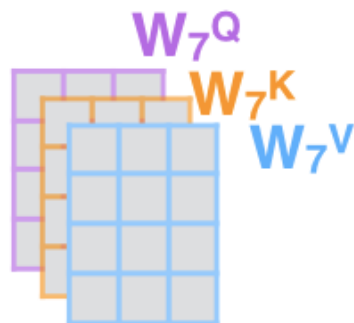
Thinking  
Machines



...

...

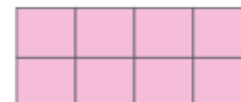
...



$W^O$



$Z$



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



# Positional Encoding

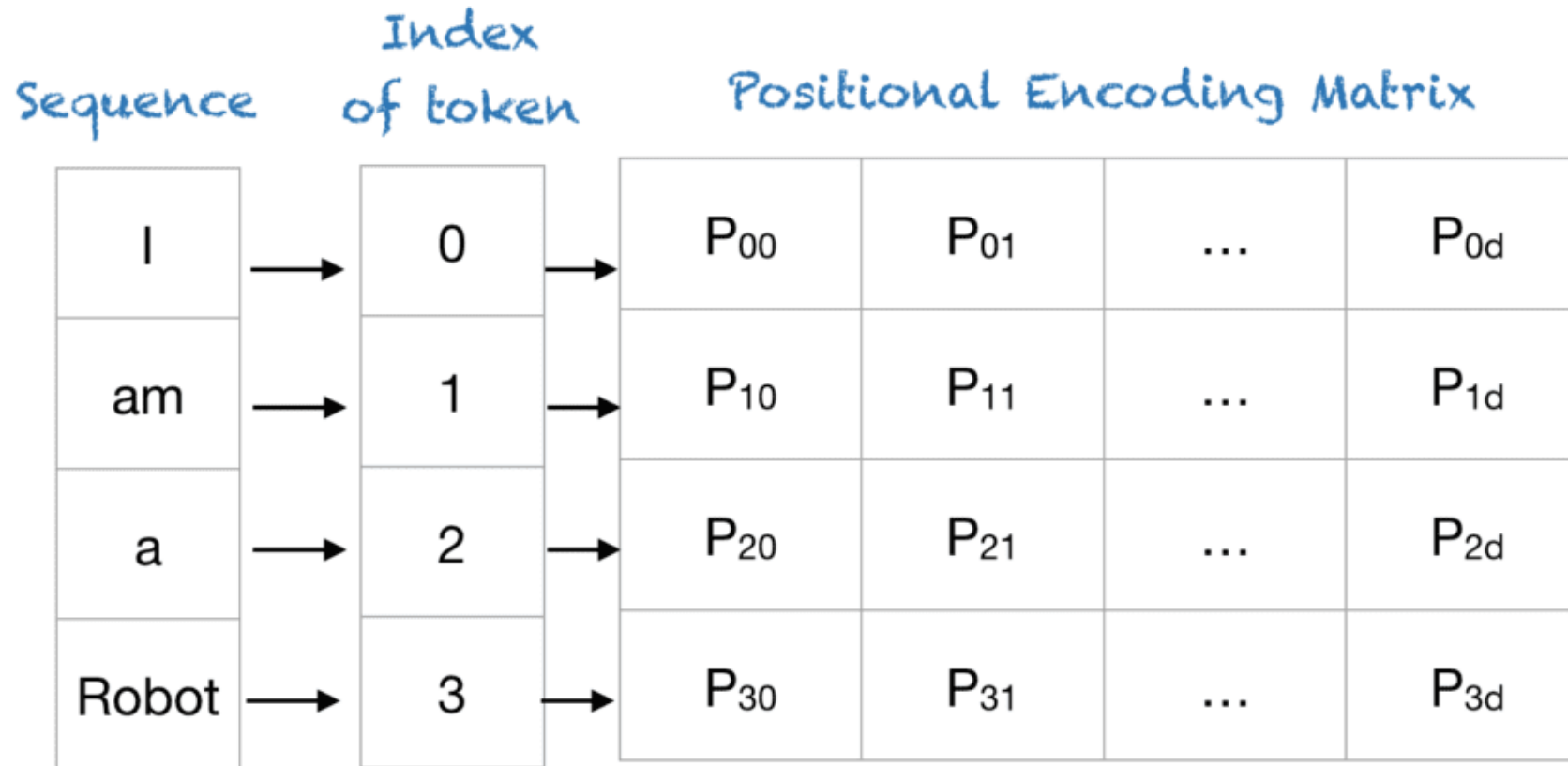
- Use sine and cosine functions of different frequencies
  - pos: word position
  - i: dimension index
  - $d_{\text{model}} = 512$

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

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



# Encoding Variable-length Sentences



Positional Encoding Matrix for the sequence 'I am a robot'

# Positional Encoding Layer in Transformers

- $k$ : Position of an object in input sequence,  $0 \leq k < L/2$
- $d$ : Dimension of the output embedding space
- $P(k, j)$ : Position function for mapping a position  $k$  in the input sequence to index  $(k, j)$  of the positional matrix
- $n$ : User defined scalar. Set to 10,000 by the authors of [Attention is all You Need](#).
- $i$ : Used for mapping to column indices  $0 \leq i < d/2$ . A single value of  $i$  maps to both sine and cosine functions

$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

$$P(k, 2i + 1) = \cos\left(\frac{k}{n^{2i/d}}\right)$$

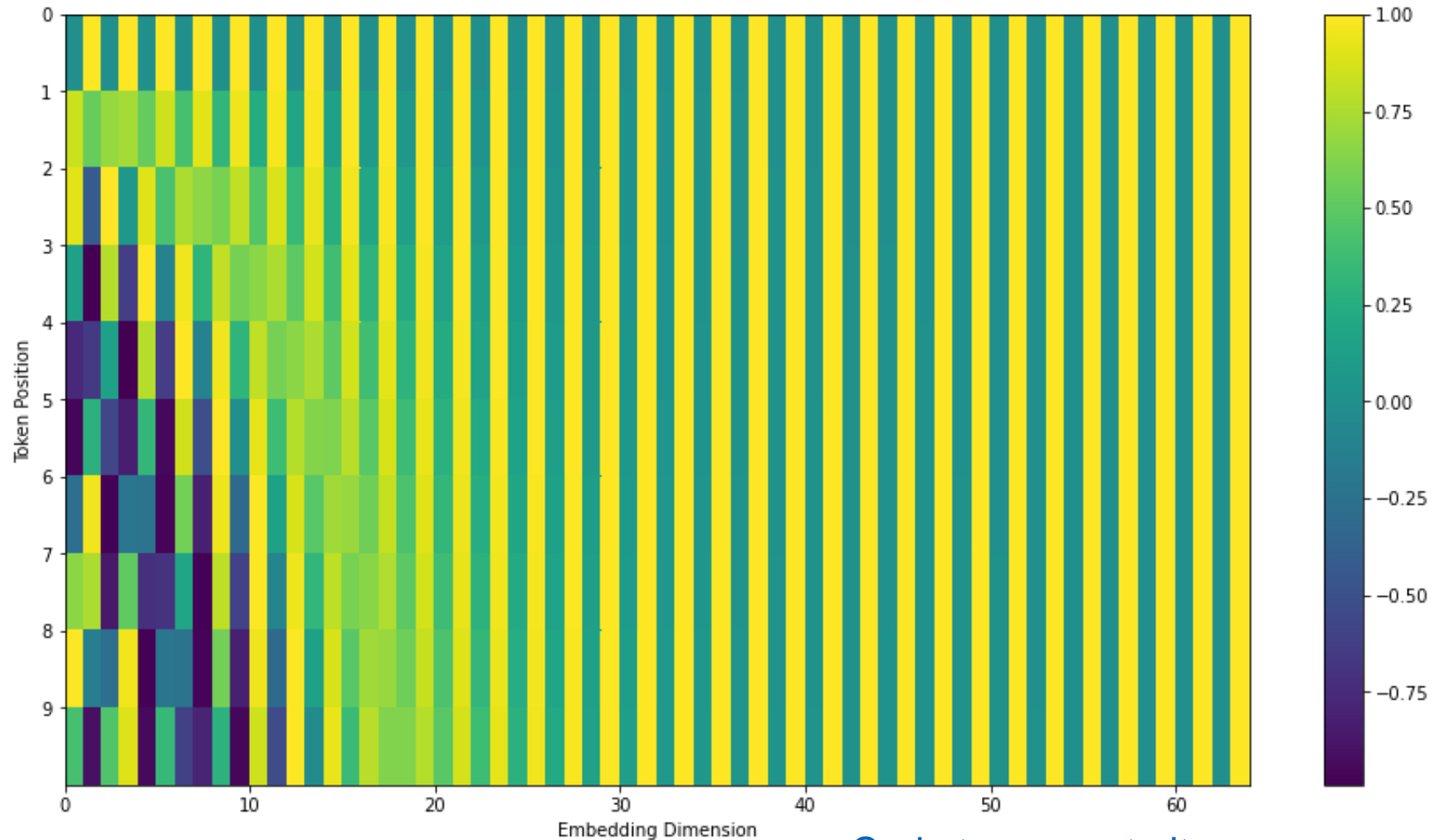
# Positional Encoding

Matrix with  $d=4$ ,  $n=100$

Sequence	Index of token, $k$	$i=0$	$i=0$	$i=1$	$i=1$
I	0	$P_{00}=\sin(0)$ = 0	$P_{01}=\cos(0)$ = 1	$P_{02}=\sin(0)$ = 0	$P_{03}=\cos(0)$ = 1
am	1	$P_{10}=\sin(1/1)$ = 0.84	$P_{11}=\cos(1/1)$ = 0.54	$P_{12}=\sin(1/10)$ = 0.10	$P_{13}=\cos(1/10)$ = 1.0
a	2	$P_{20}=\sin(2/1)$ = 0.91	$P_{21}=\cos(2/1)$ = -0.42	$P_{22}=\sin(2/10)$ = 0.20	$P_{23}=\cos(2/10)$ = 0.98
Robot	3	$P_{30}=\sin(3/1)$ = 0.14	$P_{31}=\cos(3/1)$ = -0.99	$P_{32}=\sin(3/10)$ = 0.30	$P_{33}=\cos(3/10)$ = 0.96

Positional Encoding Matrix for the sequence 'I am a robot'

# Visualizing Positional Encoding



[Code to generate it](#)

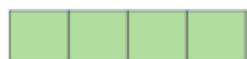


ENCODER #2

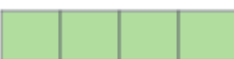
ENCODER #1

DECODER #1

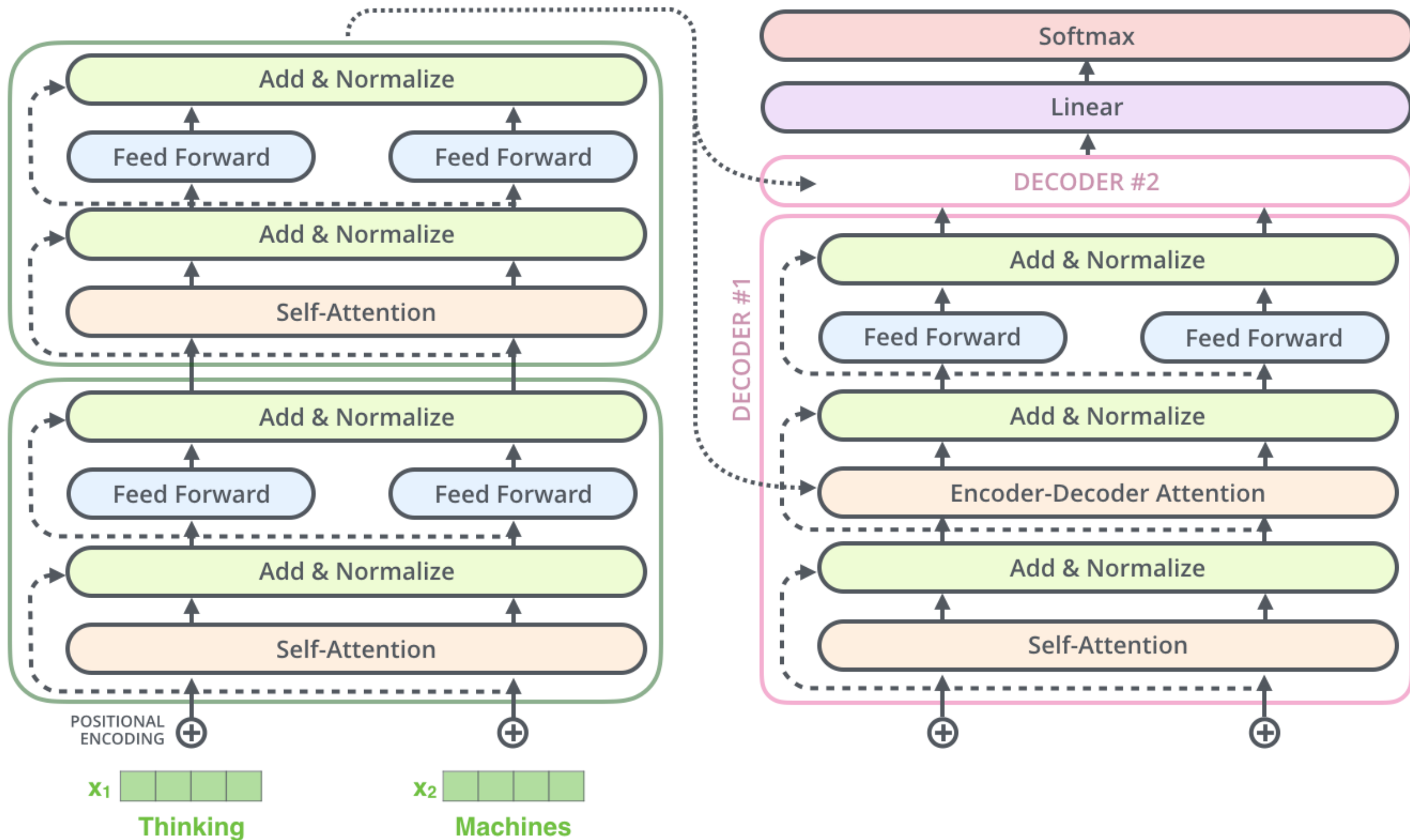
DECODER #2

POSITIONAL  
ENCODING $x_1$ 

Thinking

 $x_2$ 

Machines







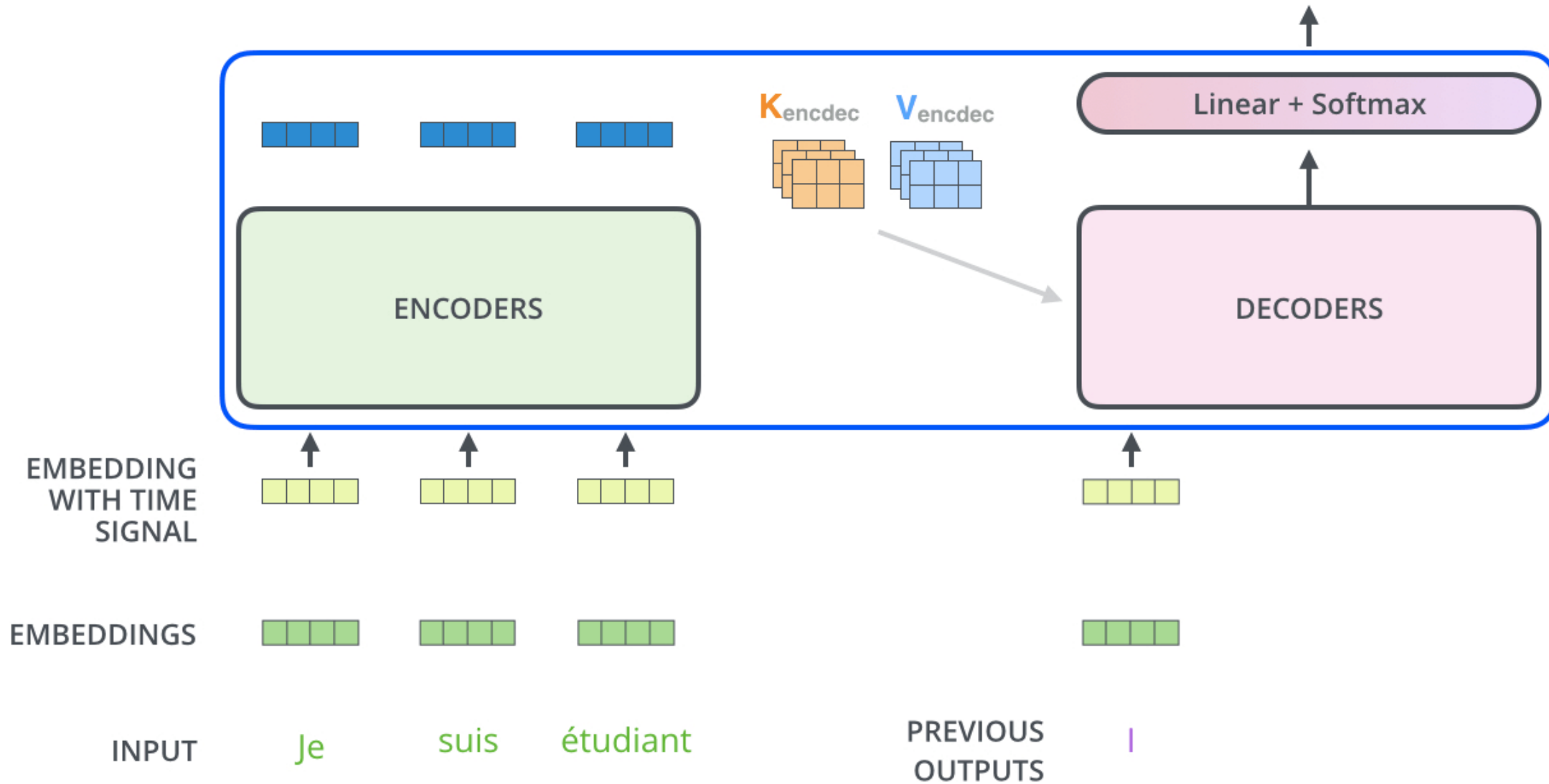
# Transformer Decoder



Decoding time step: 1 2 3 4 5 6

OUTPUT |

# Decoder



Which word in our vocabulary  
is associated with this index?

Get the index of the cell  
with the highest value  
(**argmax**)

**log\_probs**



am

5

Softmax

**logits**



Linear

Decoder stack output



# Output of Decoder

Output Vocabulary

WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

One-hot encoding of the word "am"



# Trained Model Outputs

Output Vocabulary:    a        am        I        thanks    student    <eos>

position #1

0.01	0.02	0.93	0.01	0.03	0.01
------	------	------	------	------	------

position #2

0.01	0.8	0.1	0.05	0.01	0.03
------	-----	-----	------	------	------

position #3

0.99	0.001	0.001	0.001	0.002	0.001
------	-------	-------	-------	-------	-------

position #4

0.001	0.002	0.001	0.02	0.94	0.01
-------	-------	-------	------	------	------

position #5

0.01	0.01	0.001	0.001	0.001	0.98
------	------	-------	-------	-------	------

a        am        I        thanks    student    <eos>



# Latest NLP Models (2018 - )

GPT, ELMo, BERT

Generative Pre-trained  
Transformer (GPT)



Embeddings  
from Language  
Models (ELMo)



Bidirectional Encoder  
Representations from  
Transformers (BERT)





# BERT: Bidirectional Encoder Representations from Transformers (2019)

- Use “Masked Language Model” to train the bidirectional transformer encoder
  - Randomly masked out some tokens and train models to predict them
- Fine-tuning on different tasks
- Achieved state-of-the-art results on multiple NLP tasks

**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

<https://arxiv.org/abs/1810.04805>

1 - **Semi-supervised** training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

### Semi-supervised Learning Step

**Model:**



**Dataset:**



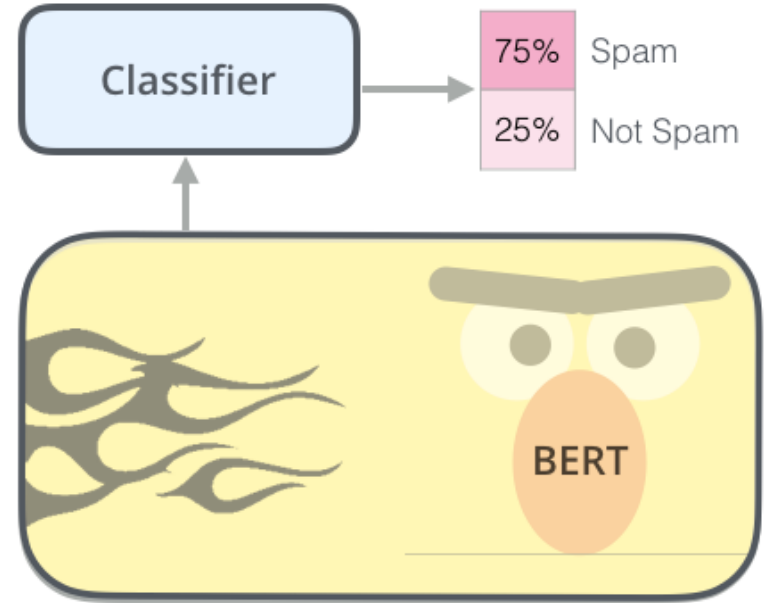
**Objective:**

Predict the masked word  
(language modeling)

2 - **Supervised** training on a specific task with a labeled dataset.

### Supervised Learning Step

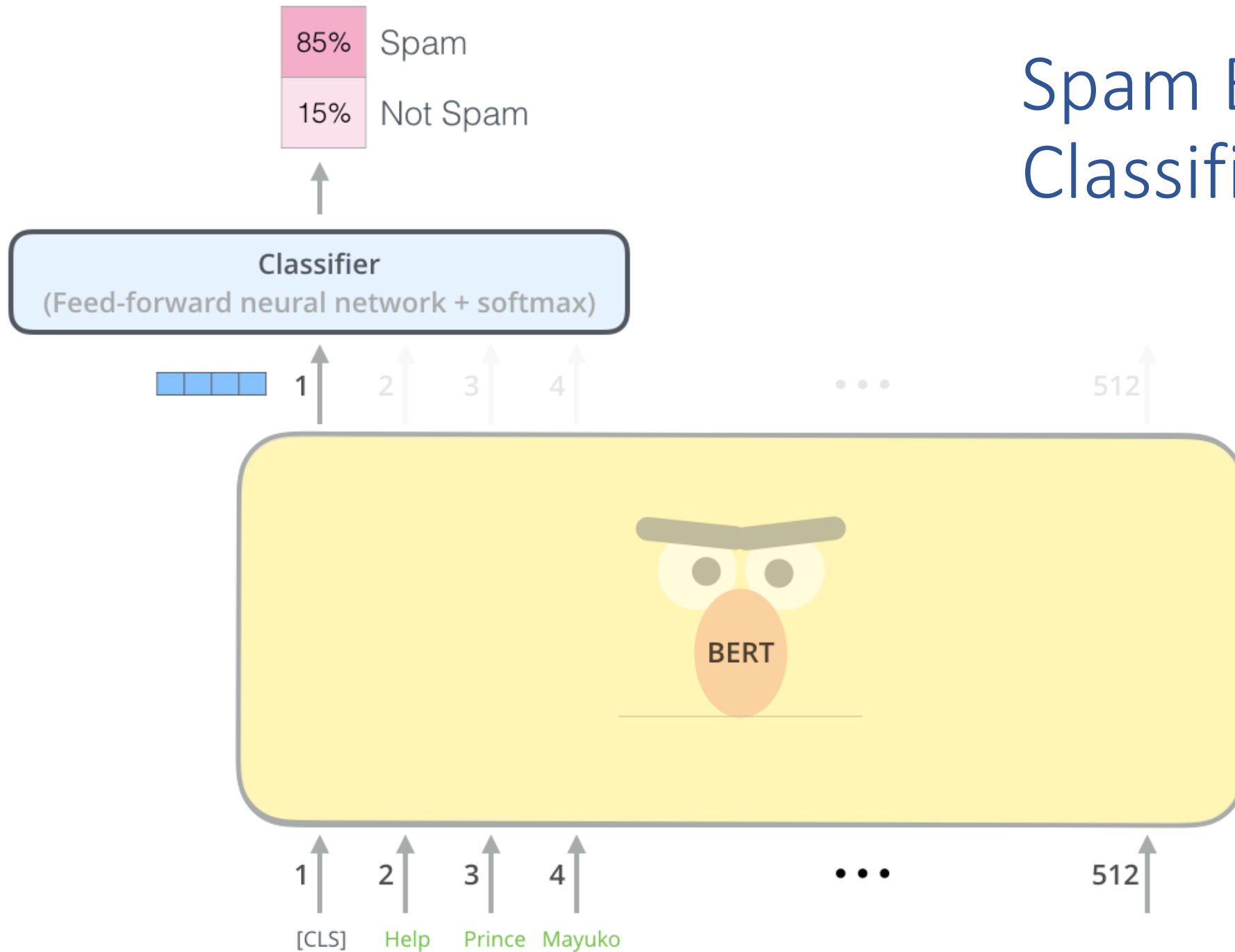
**Model:**  
(pre-trained  
in step #1)



**Dataset:**

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached...	Not Spam

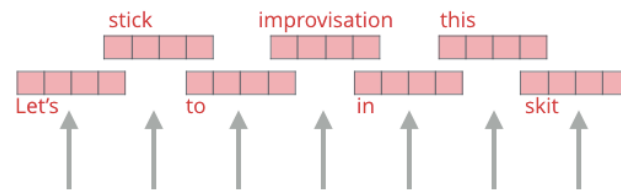
# Spam Email Classification



# Embeddings from Language Models (ELMo)

- Consider how words vary across contexts
- Use sentence as input and encoded it by bi-directional LSTM

ELMo  
Embeddings



Words to embed



## Deep contextualized word representations

**Matthew E. Peters<sup>†</sup>, Mark Neumann<sup>†</sup>, Mohit Iyyer<sup>†</sup>, Matt Gardner<sup>†</sup>,**  
{matthewp, markn, mohiti, mattg}@allenai.org

**Christopher Clark<sup>\*</sup>, Kenton Lee<sup>\*</sup>, Luke Zettlemoyer<sup>†\*</sup>**  
{csquared, kentonl, lsz}@cs.washington.edu

<sup>†</sup>Allen Institute for Artificial Intelligence

<sup>\*</sup>Paul G. Allen School of Computer Science & Engineering, University of Washington

# Use Bi-LSTM to create Word Embedding

1- Concatenate hidden layers



2- Multiply each vector by a weight based on the task

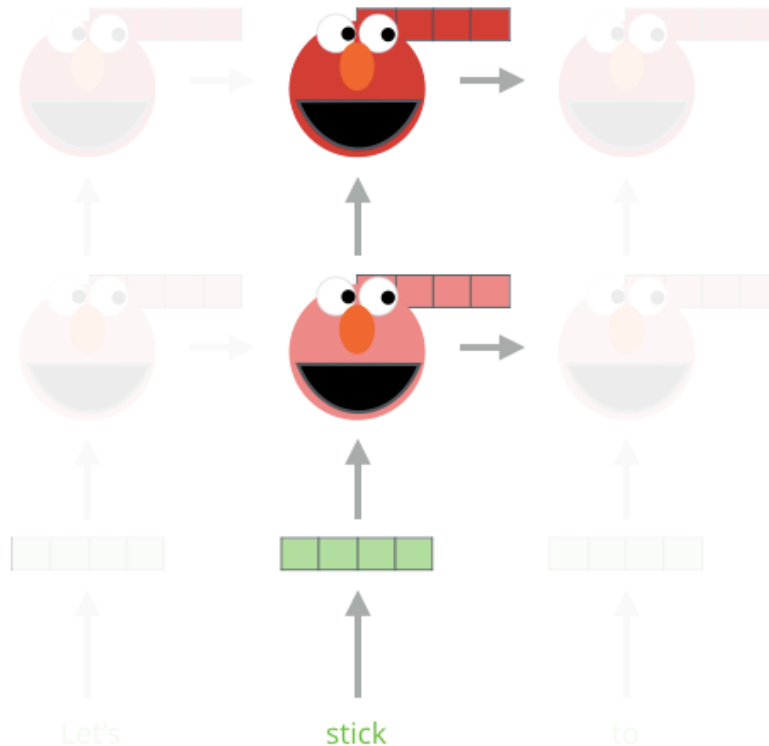


3- Sum the (now weighted) vectors

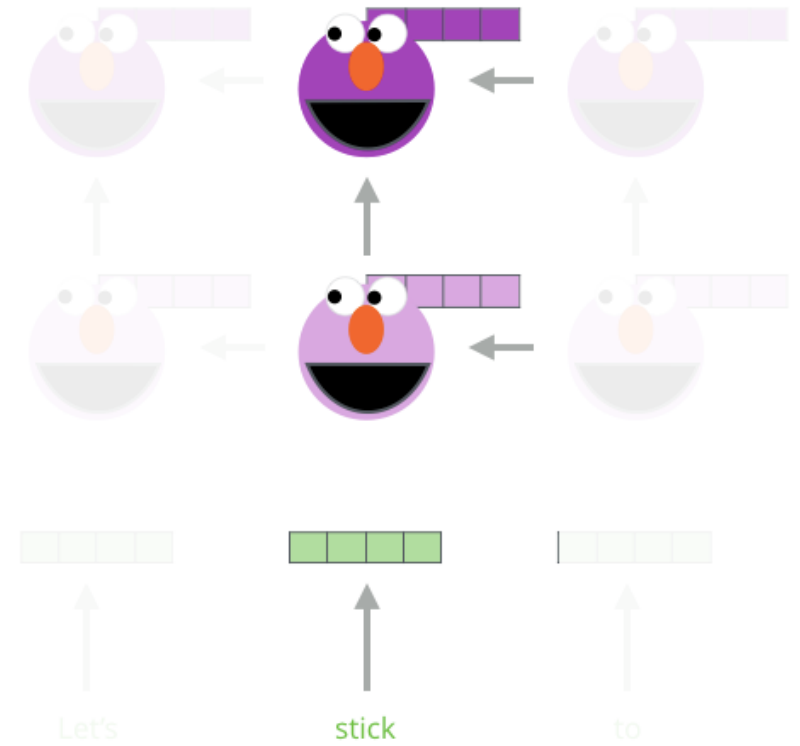


ELMo embedding of "stick" for this task in this context

Forward Language Model

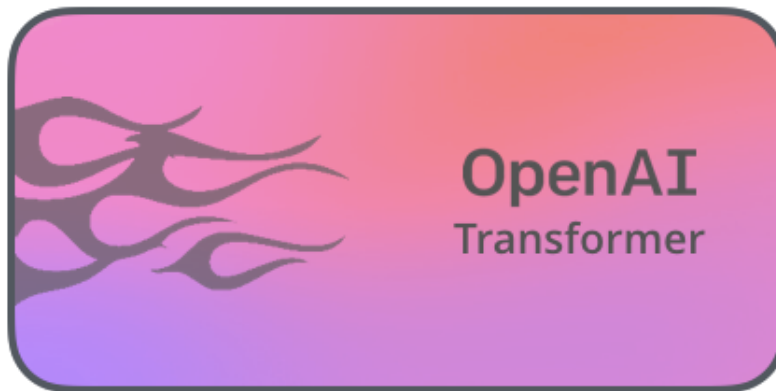


Backward Language Model

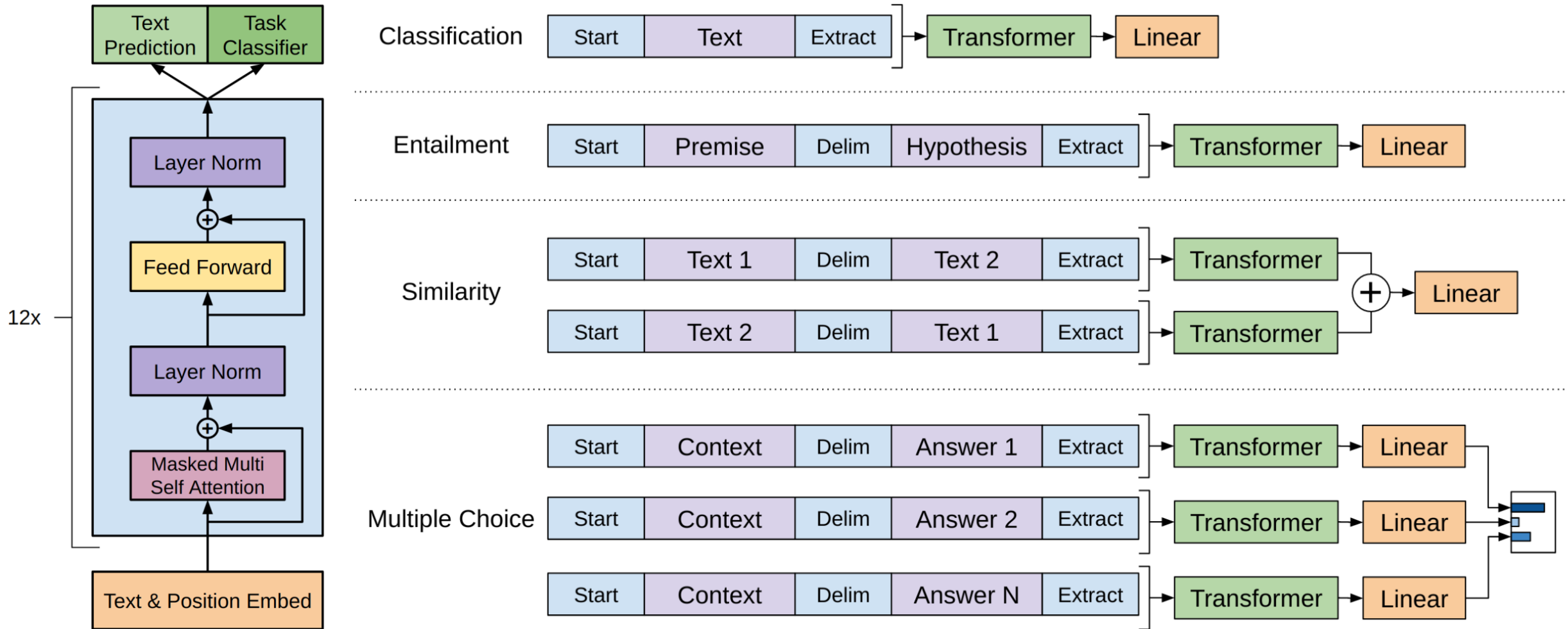


# OpenAI GPT: Pre-training Transformer Decoders

- Unsupervised pre-train transform decoders for predicting the next word (GPT: Generative Pre-Training)
- Use 12 Transformer decoders in GPT-1
  - GPT-1: [Improving Language Understanding with Unsupervised Learning \(2018\)](#)
  - GPT-2: [Better Language Models and Their Implications \(2019\)](#)
  - GPT-3: [Language Models are Few-Shot Learners \(2020\)](#)



# OpenAI GPT for Different Tasks





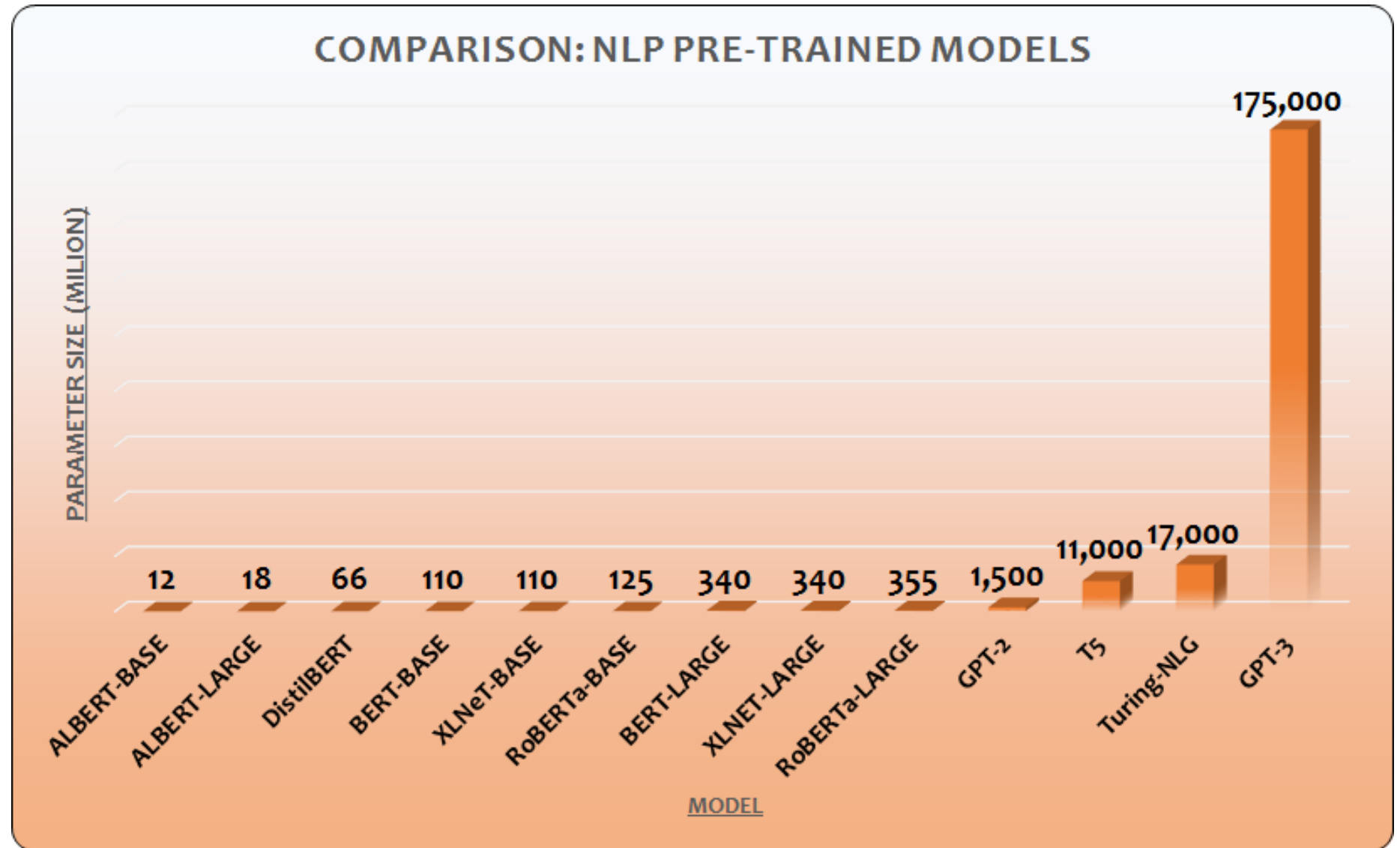
# OpenAI GPT-2

- Pre-trained using 40GB of Internet text
- Scale-up of GPT with 10X parameters trained with 10X data
- Other tricks
  - Layer normalization was moved to the input of each sub-block
  - An additional layer normalization was added after the final self-attention block

Parameters	Layers	$d_{model}$
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

# Size does Matter! GPT-3

- 175 Billion Parameters!
- $175 \times 4 = 700\text{GB}$
- 55 years and \$4,600,000 to train - even with the lowest priced GPU cloud on the market.



# GPT3 Demo ([gpt3demo.com](https://gpt3demo.com))

Collections

New

Popular

Upcoming

Requested

Categories

All136

A/B Testing1

Ad Generation3

AI Writing Assistants1

Alternative Language M...8

Blog writing2

Chatbots1

CLIP2

Code Generation2

Content Comprehension2

Content Generation1

Copywriting21

Customer service4


Databases1

GPT-3 DEMO | GPT-3 showcase

API TrackerSaaS BlocksStartup ProgramsRequest GPT-3 API access

GPT-3 Demo Showcase, 100+ Apps, Examples, & Resources


Get inspired and discover how companies are implementing the OpenAI GPT-3 API to power new use cases



Email Generation

AI Sales Email Assistant by R...


Generate human-like emails a...



Copywriting

BetterWriter


Chrome extension to write bet...



Content Generation

Content generation engine b...


Dentsu APAC's Data Sciences ...



Screenwriting


Date Night Short Film

A short film written by GPT-3



Copywriting


Hyperwrite



Songwriting

Jarvis


AI lyrics generator



Humor


things are a little crazy rn

GPT-3 scheduling hell



Dialogs

Two GPT-3 AIs conversing ab...



CLIP

Which Frame?

Search a video semantically.

See all →

32

huggingface.co



# Write With Transformer

Get a modern neural network to  
auto-complete your thoughts.

This web app, built by the Hugging Face team, is the official demo of the  
🤖/transformers repository's text generation capabilities.



Star

51,543

## Checkpoints



### DistilGPT-2

The student of the now ubiquitous GPT-2 does not come short of its teacher's expectations. Obtained by distillation, DistilGPT-2 weighs 37% less, and is twice as fast as its OpenAI counterpart, while keeping the same generative power. Runs smoothly on an iPhone 7. The dawn of lightweight generative transformers?

Start writing

More info







OpenAI GPT-2

# huggingface.co

← → ↻ 🔒 transformer.huggingface.co/doc/distil-gpt2 ☆ 🗨️ S 📁 ⬆️ ⚙️ 🧩 👤 ⋮

🌐 應用程式 📁 Downloads ★ Bookmarks 📁 Labels on Google T... 📁 Deep Learning 📁 Journals 📁 Python 📁 Drone 📁 Unreal » 📁 其他書籤 📄 閱讀清單

 **Write With Transformer** distil-gpt2 ⓘ

 Shuffle initial text  Trigger autocomplete or tab Select suggestion ↑ ↓ and enter Cancel suggestion esc Save & Publish 

Model & decoder settings ⓘ

Model size **distilgpt2/small**

Top-p **0.9**

Temperature **1**

Max time **1**

Who is Kuan-Ting Lai?

It was a story about a Korean couple who were just starting a new life i...

The world-famous and respected Chinese martial artist Kuan-Ting Lai

🔊

# References

1. <https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html>
2. <http://jalammar.github.io/illustrated-transformer/>
3. <http://jalammar.github.io/illustrated-bert/>
4. Hong-Yi Lee, Transformer, 2019, <https://www.youtube.com/watch?v=ugWDIIOHtPA>