Modern NLP Models

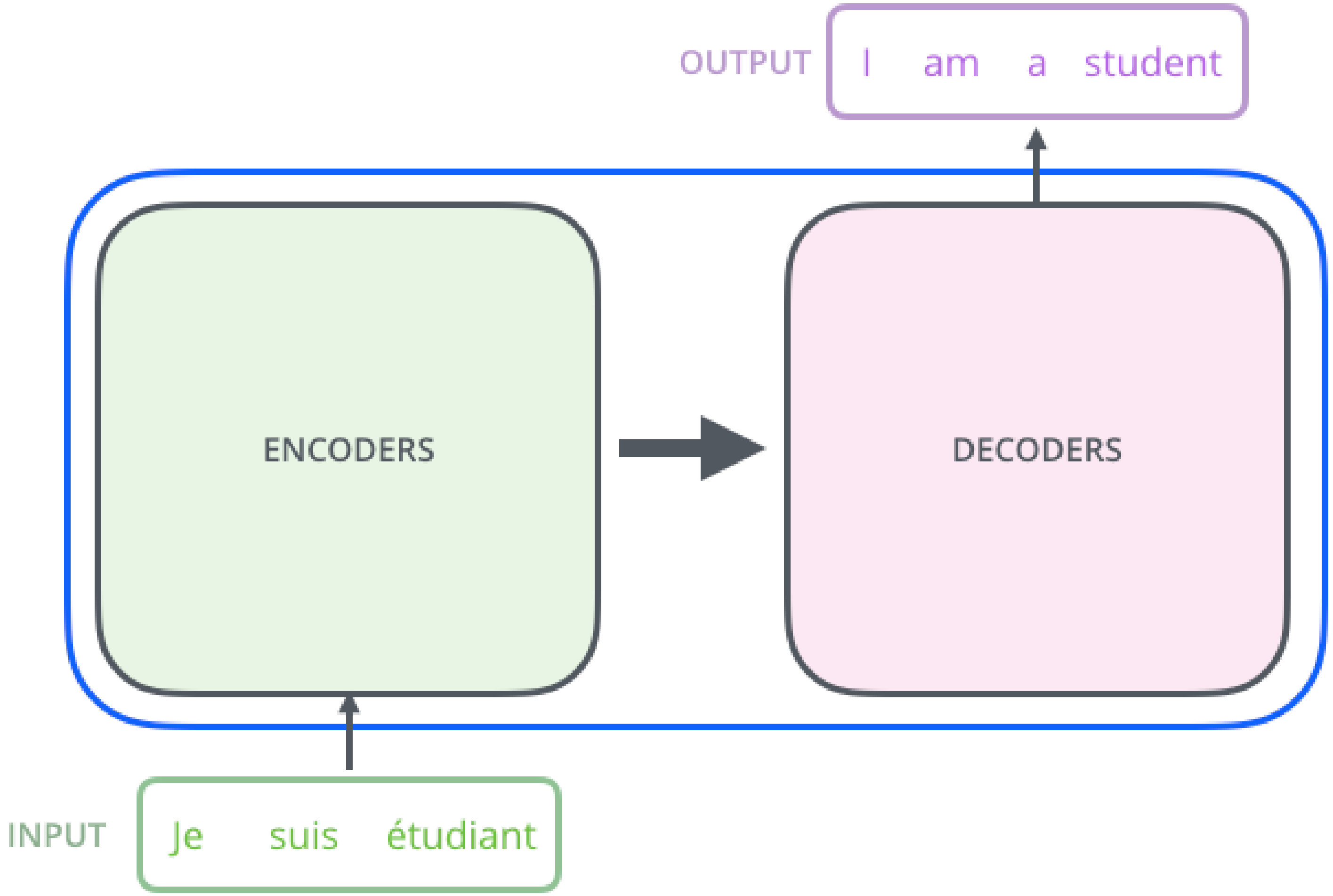
Prof. Kuan-Ting Lai

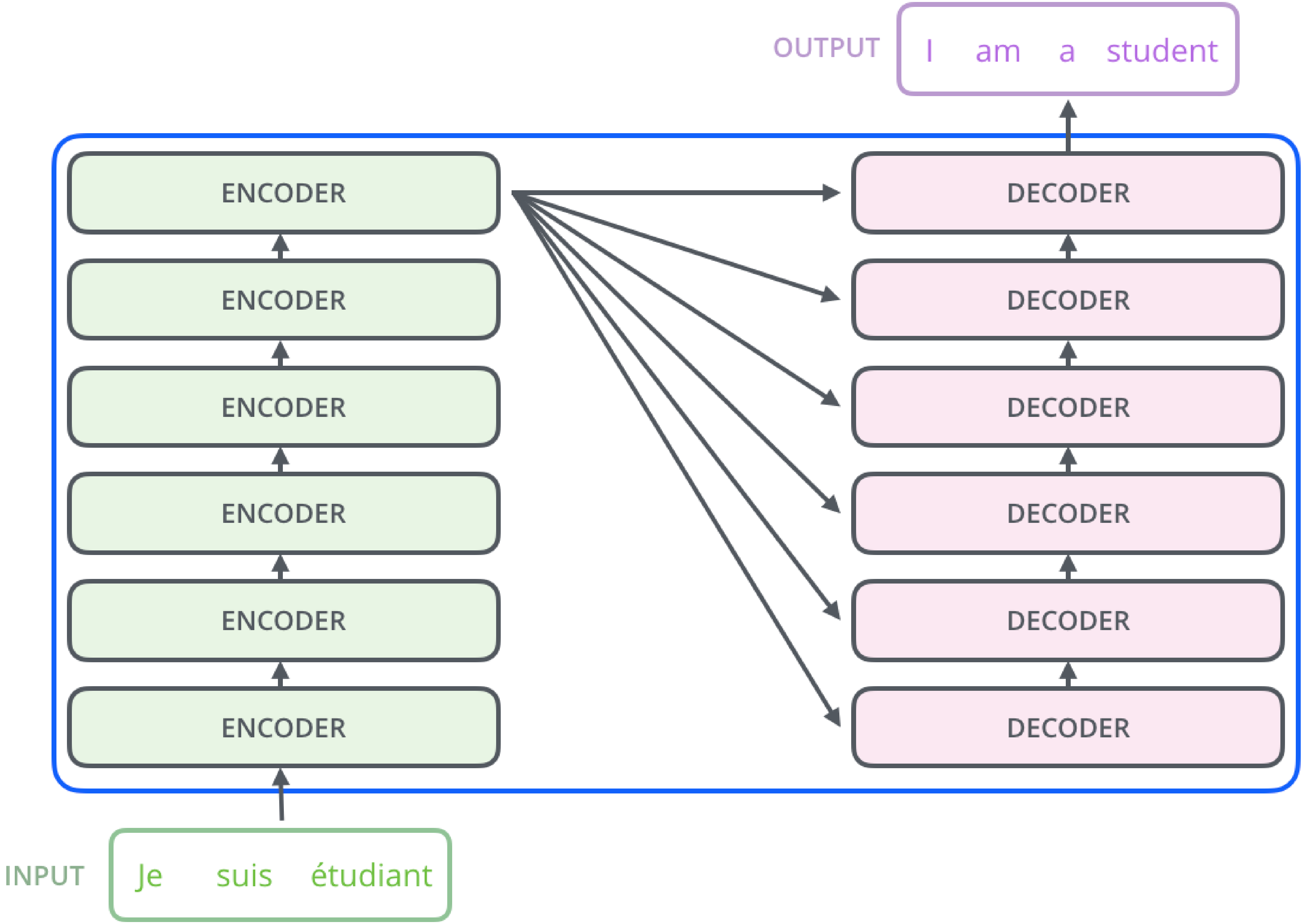
2022/5/16

Transformer Illustration

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

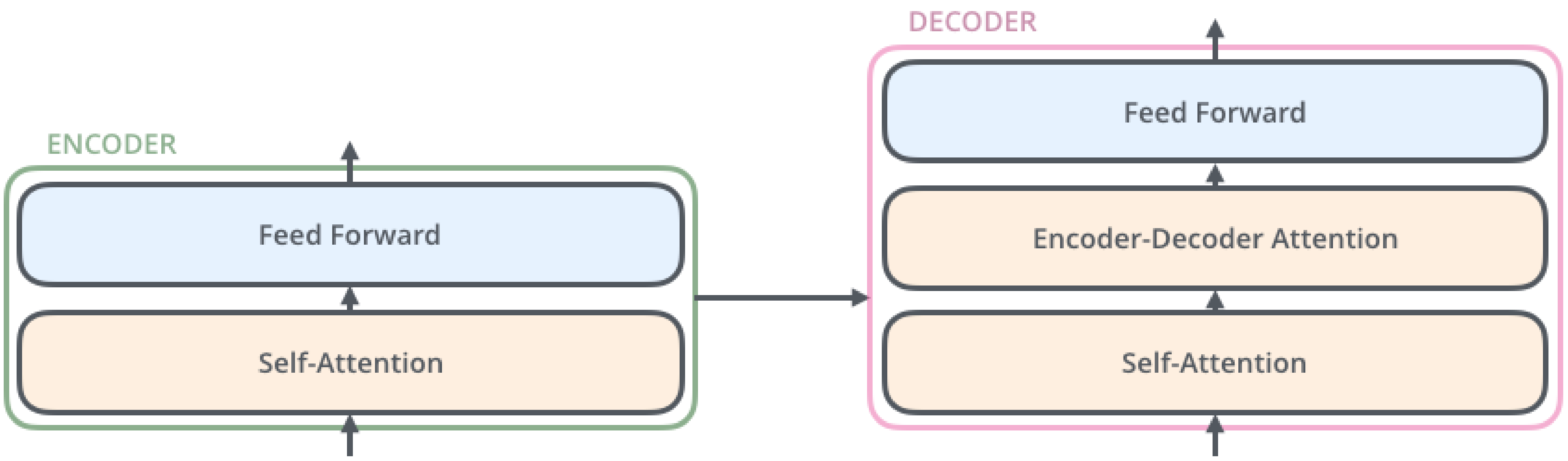
Encoder and Decoder



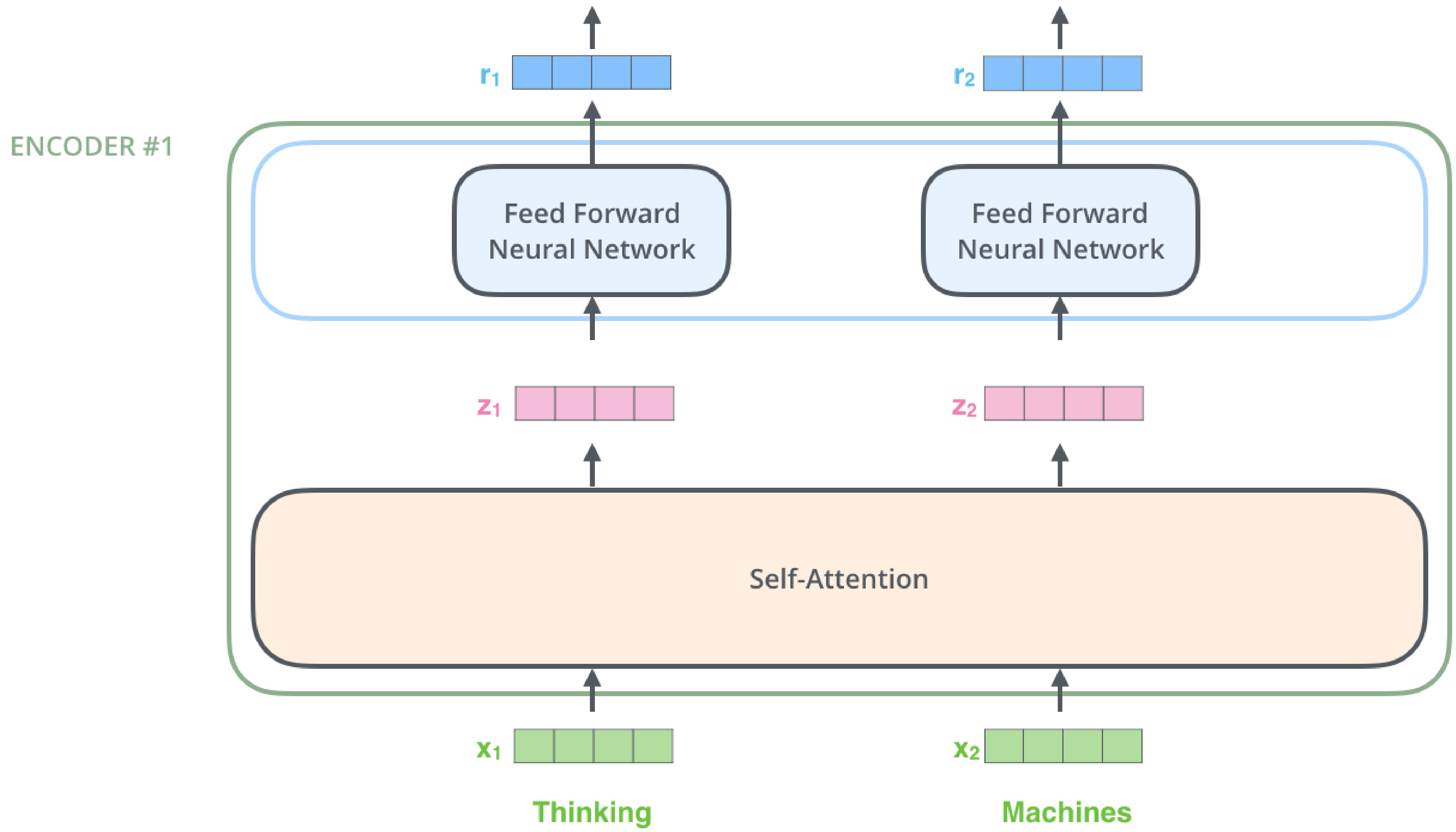

OUTPUT
I am a student
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant

Structure of the Encoder and Decoder

- Self-attention
- Encoder-decoder attention



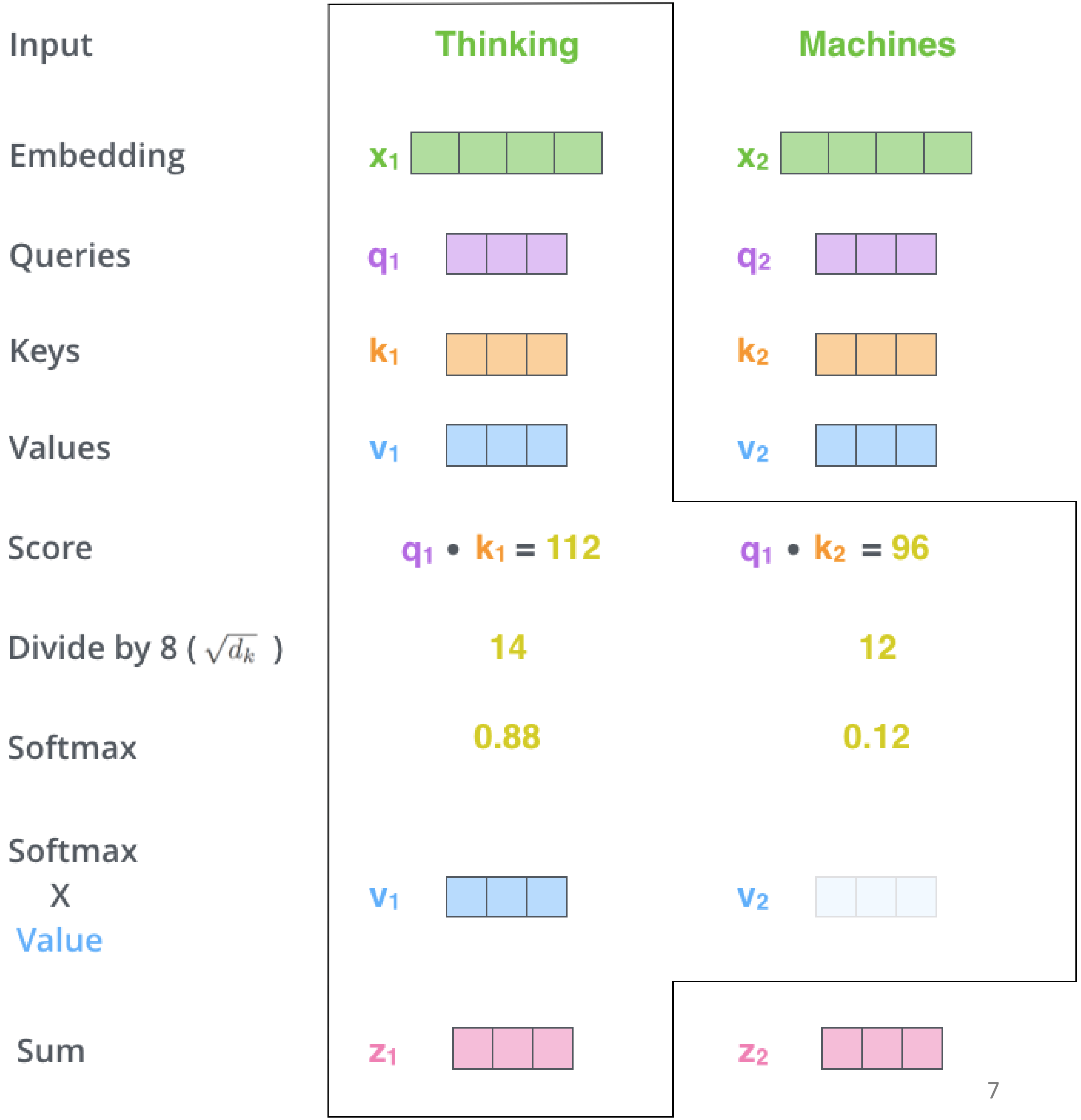
Start Encoding



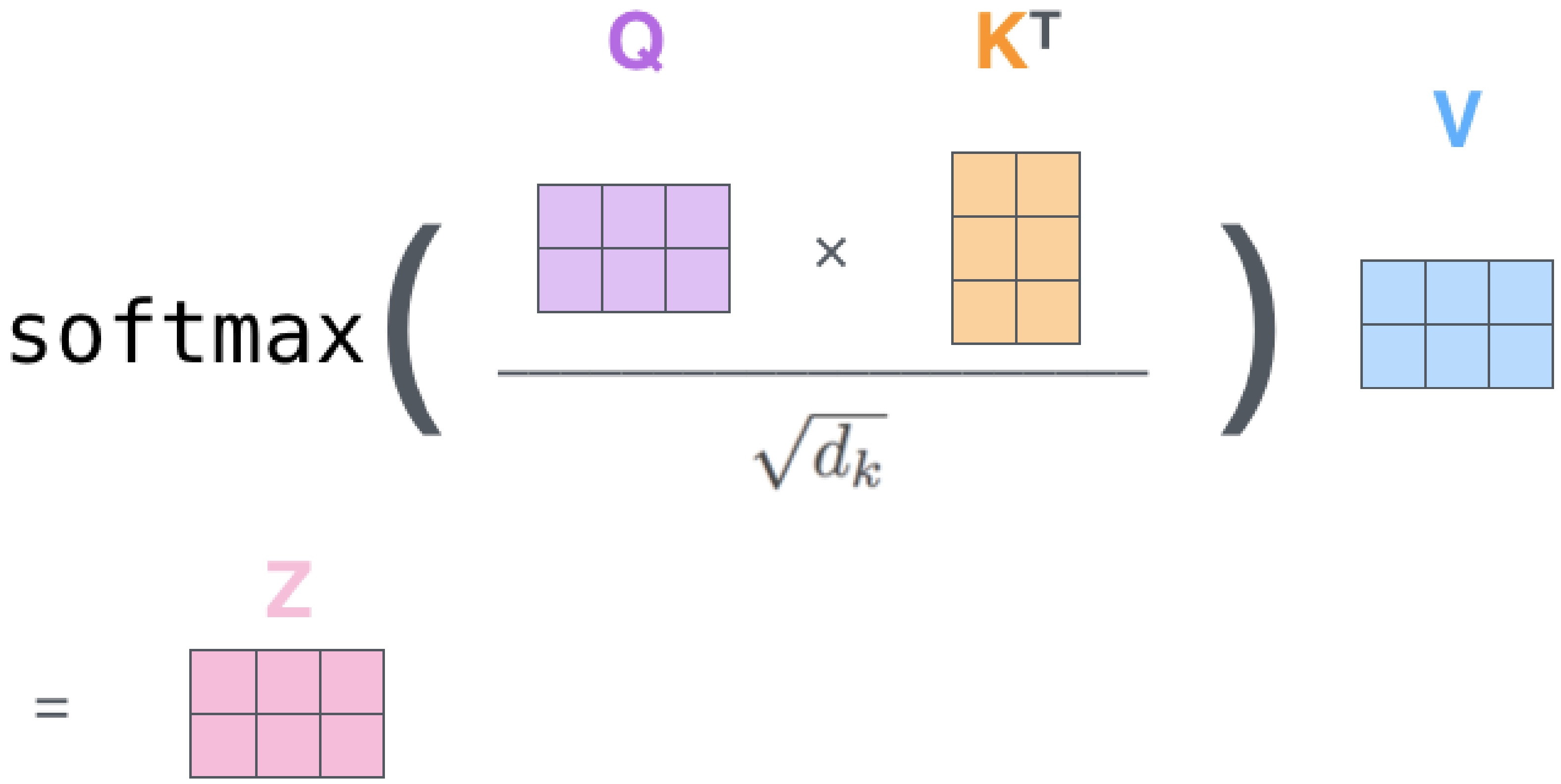
Calculate Values

- 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

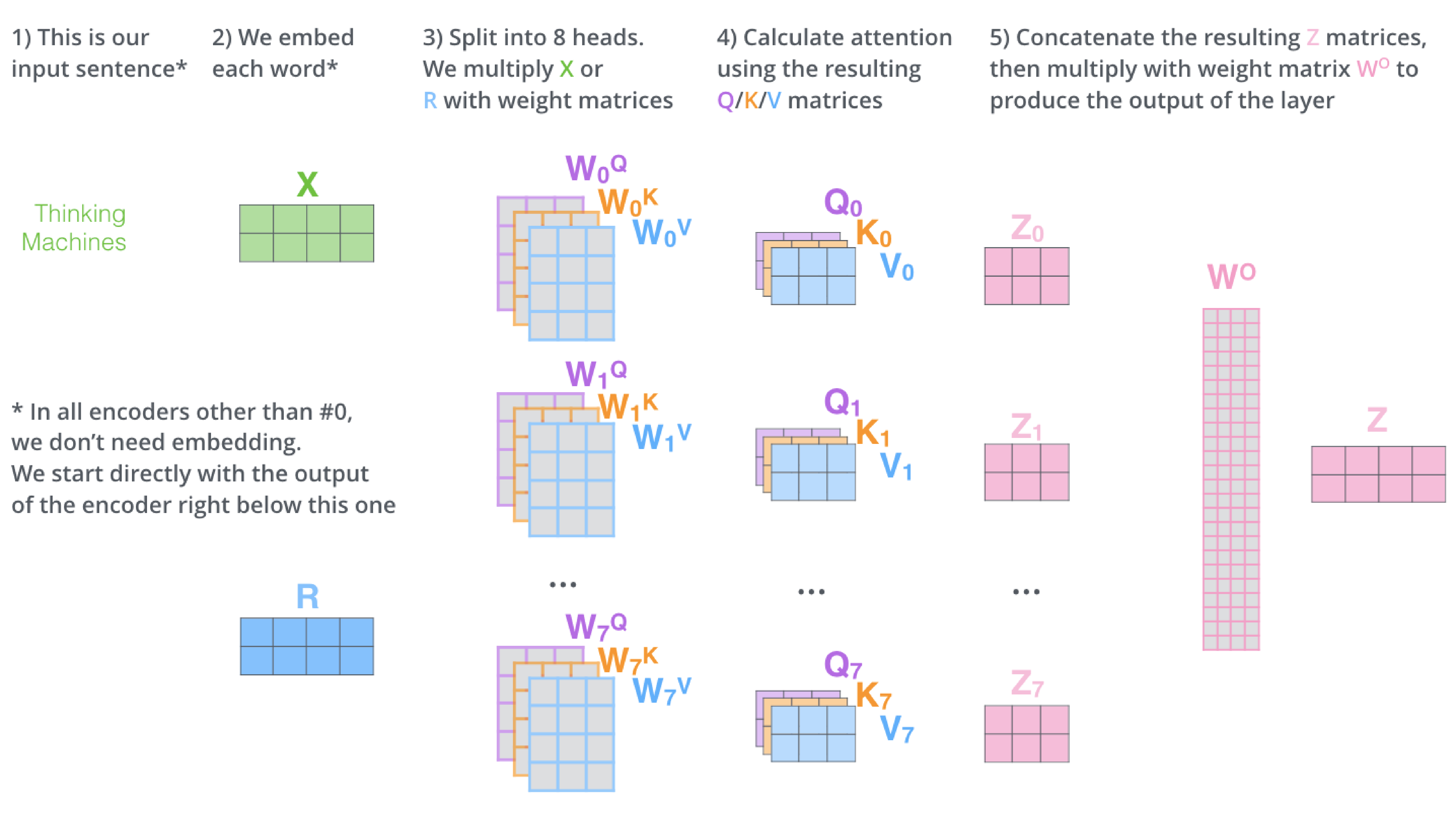
$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Final Output of the Self-attention Module



The Beast with Multiple Heads



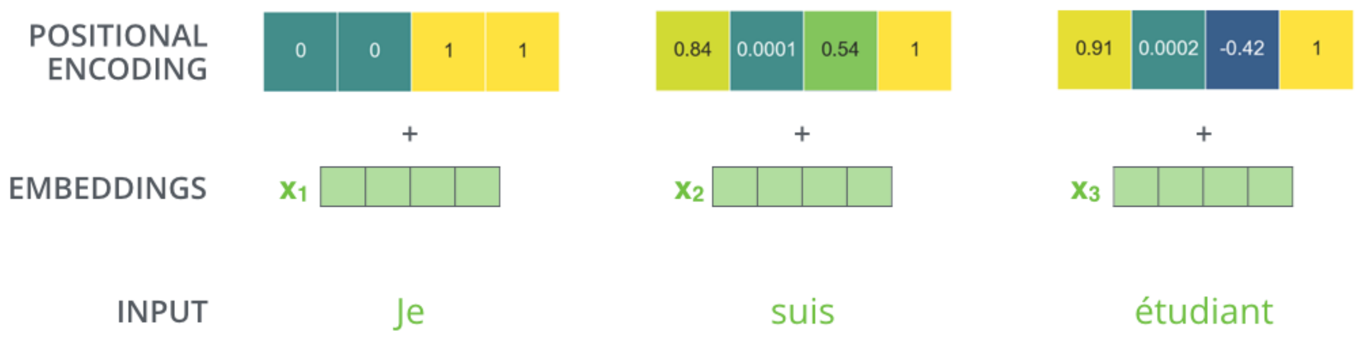
1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
R
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
W0Q W0K W0V
W1Q W1K W1V
W7Q W7K W7V
Q0 K0 V0
Q1 K1 V1
Q7 K7 V7
Z0
Z1
Z7
WO
Z
...

Positional Encoding

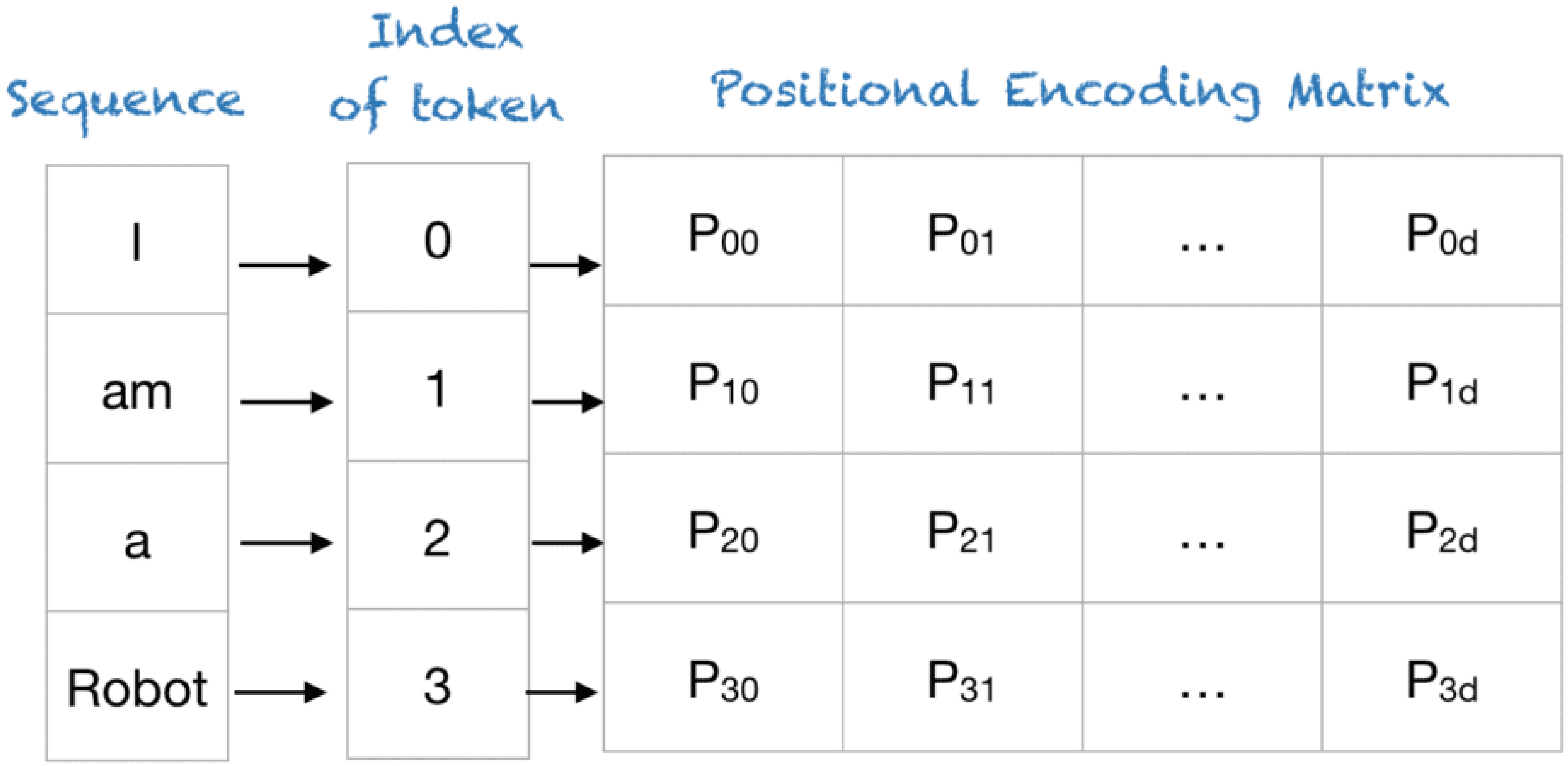
- Use sine and cosine functions of different frequencies
  - pos: word position
  - i: dimension index
  - $d_{model} = 512$

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

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



Encoding Variable-length Sentences



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

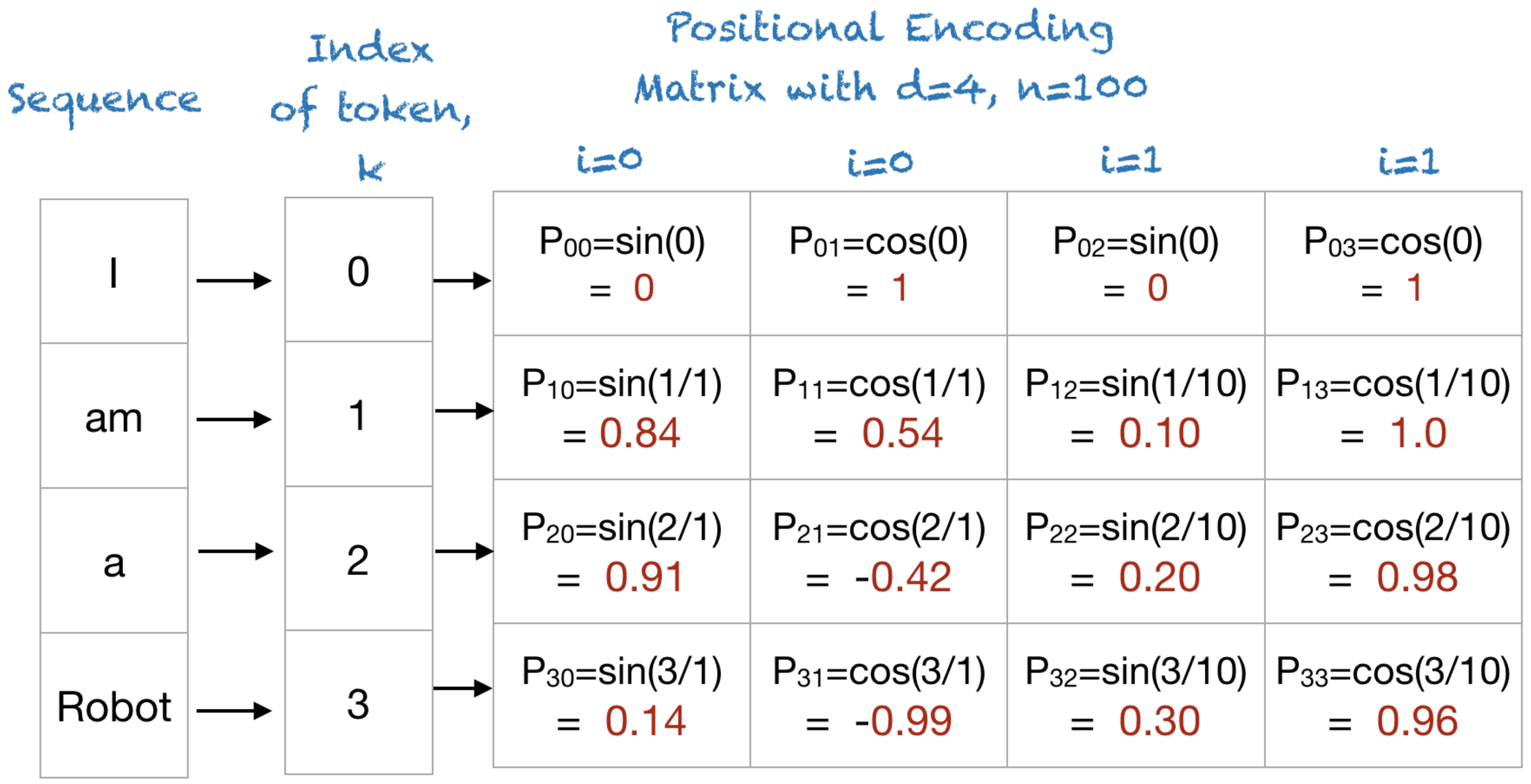
https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1

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

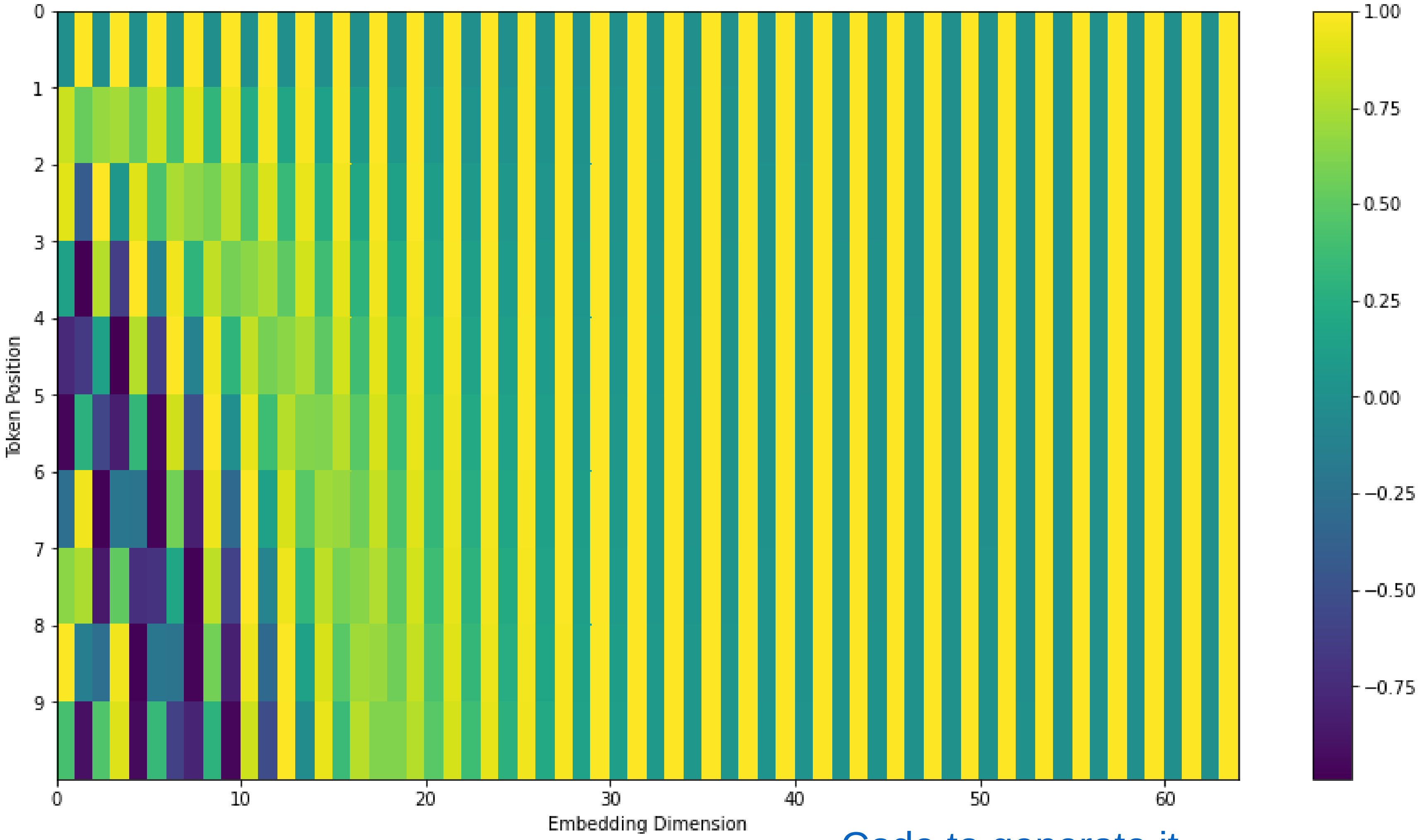
$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

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

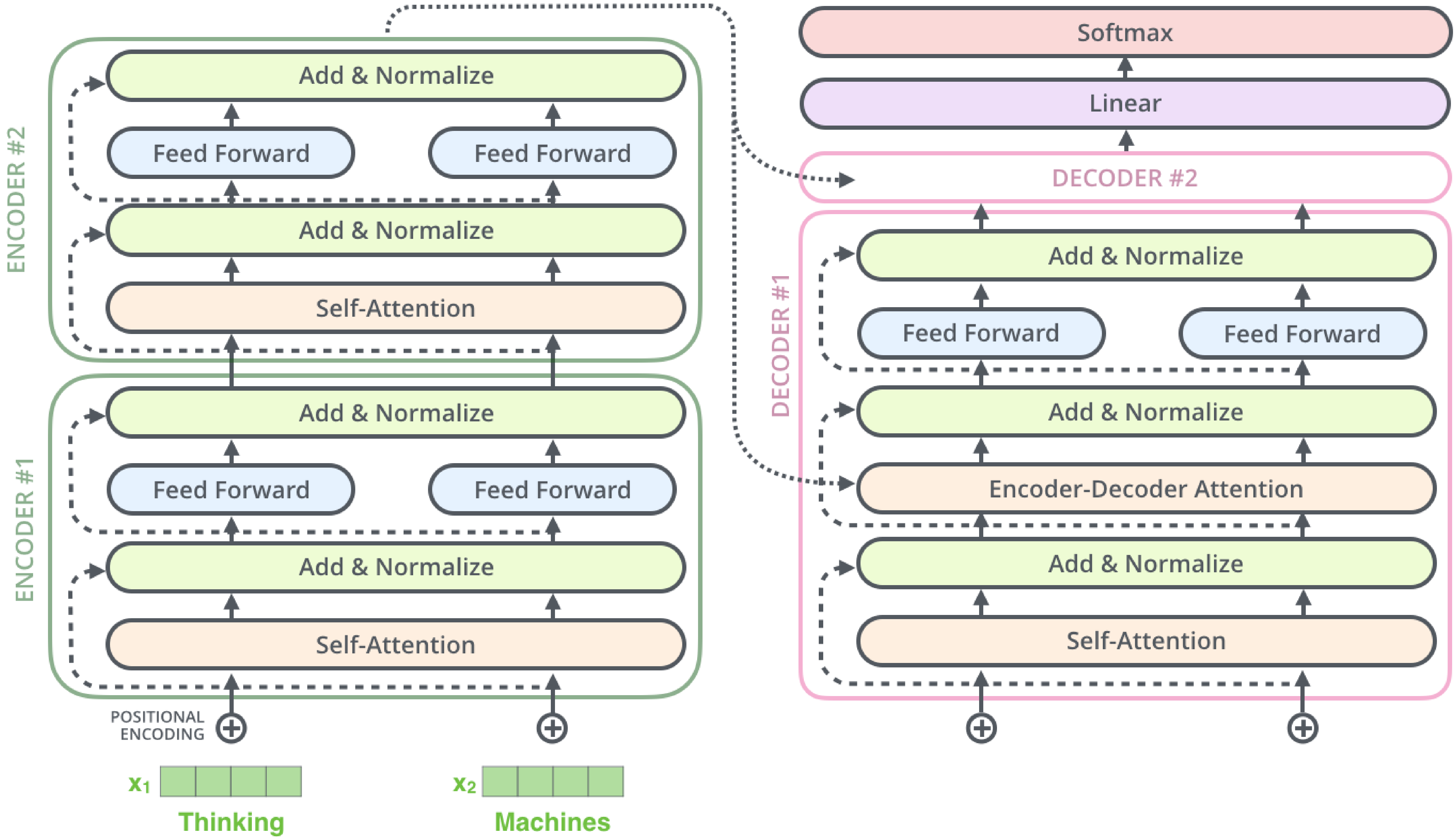


Positional Encoding Matrix for the sequence 'I am a robot'

Visualizing Positional Encoding

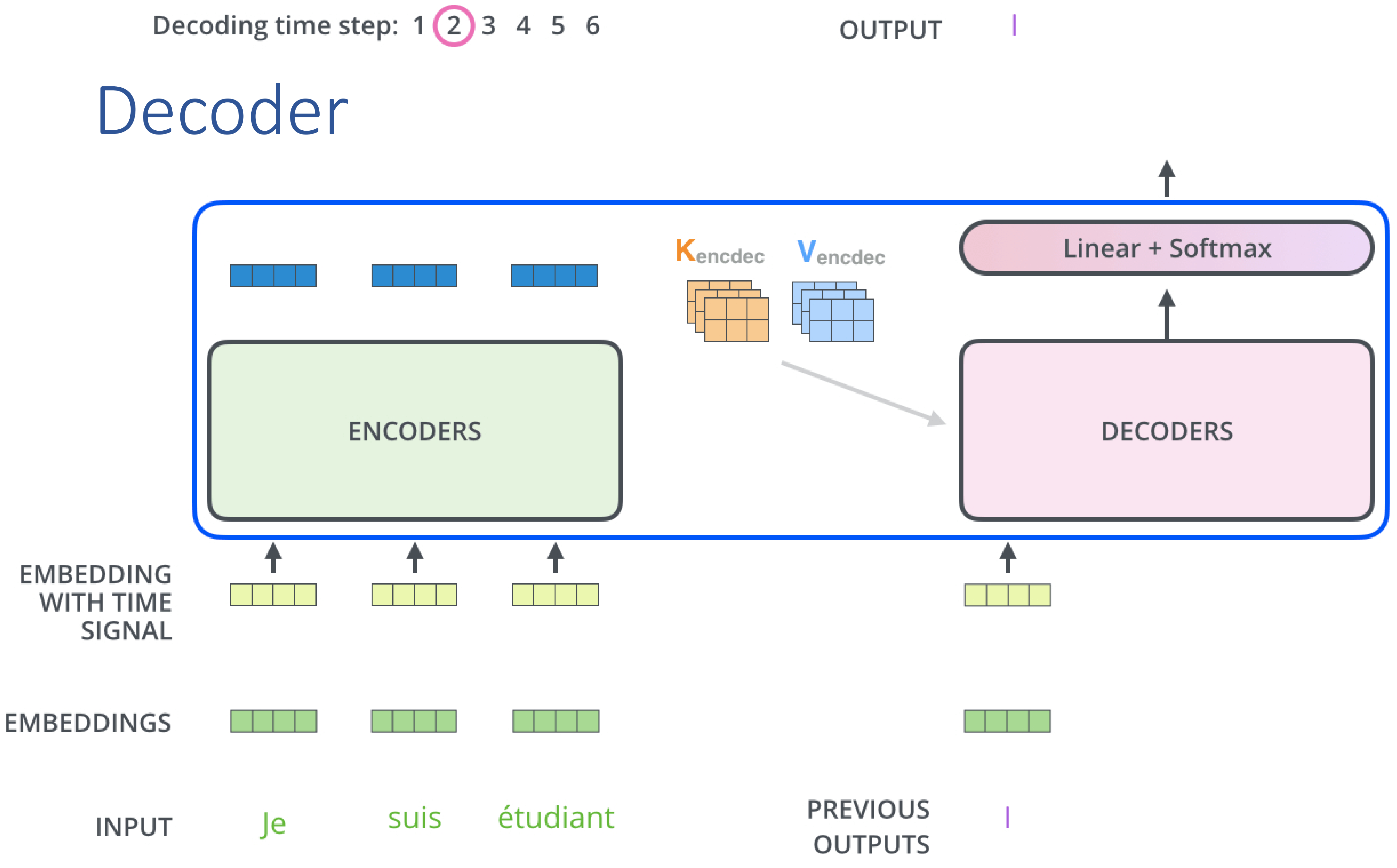


[Code to generate it]


Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention
ENCODER #2
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL ENCODING
x1
Thinking
x2
Machines

Transformer Decoder

Decoder



Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Kencdec
Vencdec
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

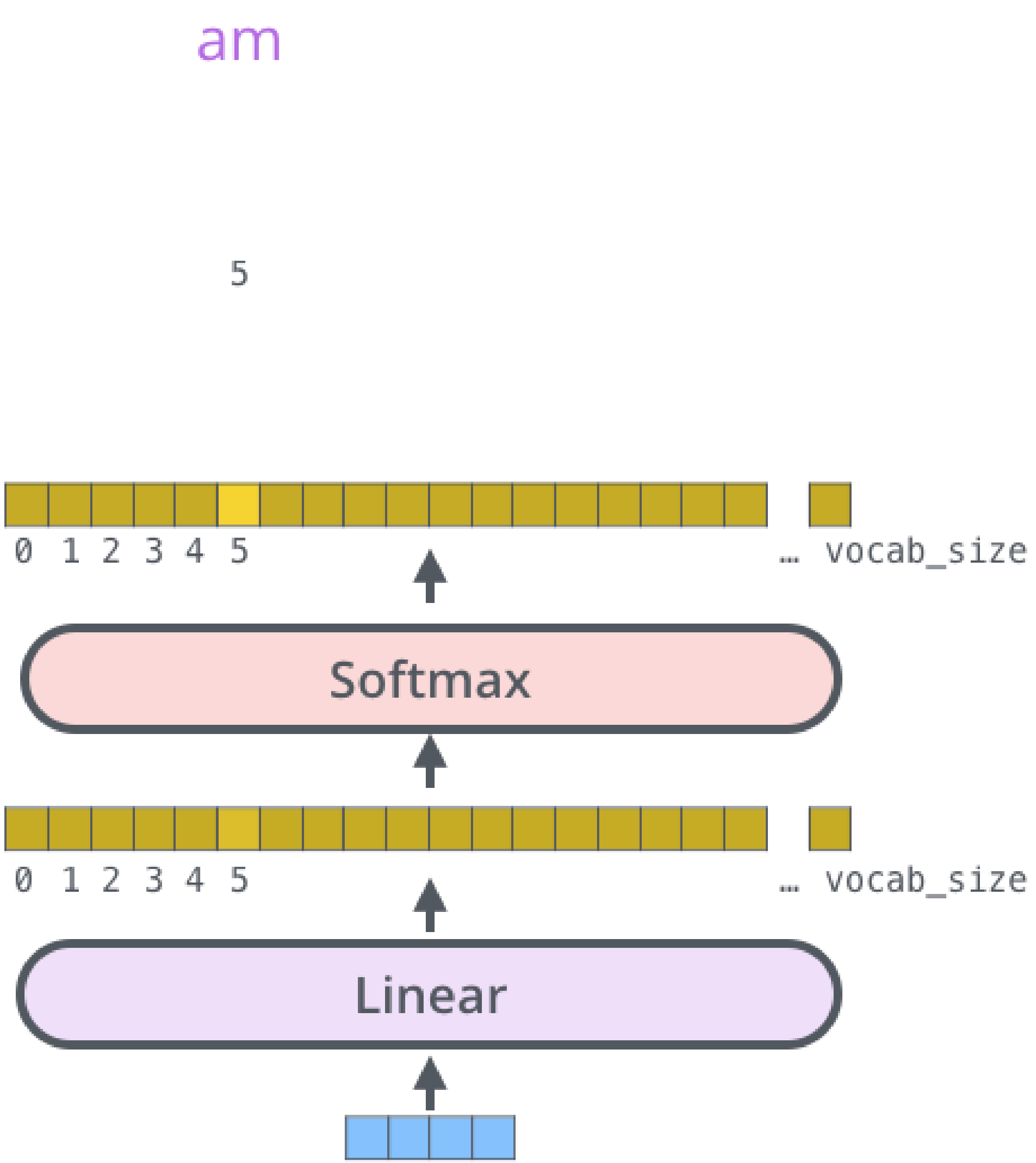
Which word in our vocabulary is associated with this index?

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

log_probs

logits

Decoder stack output



Output of Decoder

Output Vocabulary

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

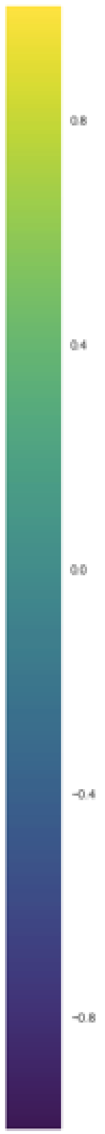
One-hot encoding of the word “am”

0.0	1.0	0.0	0.0	0.0	0.0

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

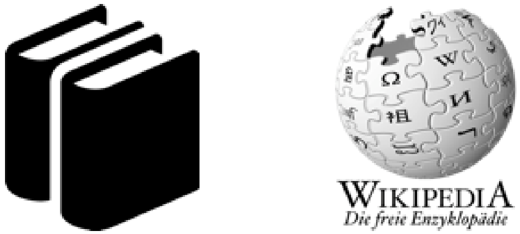
1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

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:

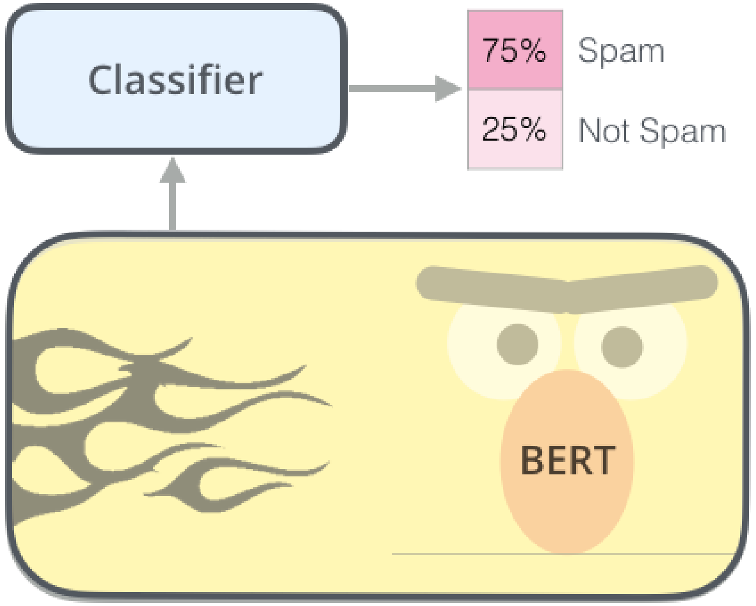


Objective: Predict the masked word (langauge modeling)

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

Supervised Learning Step

Model: (pre-trained in step #1)

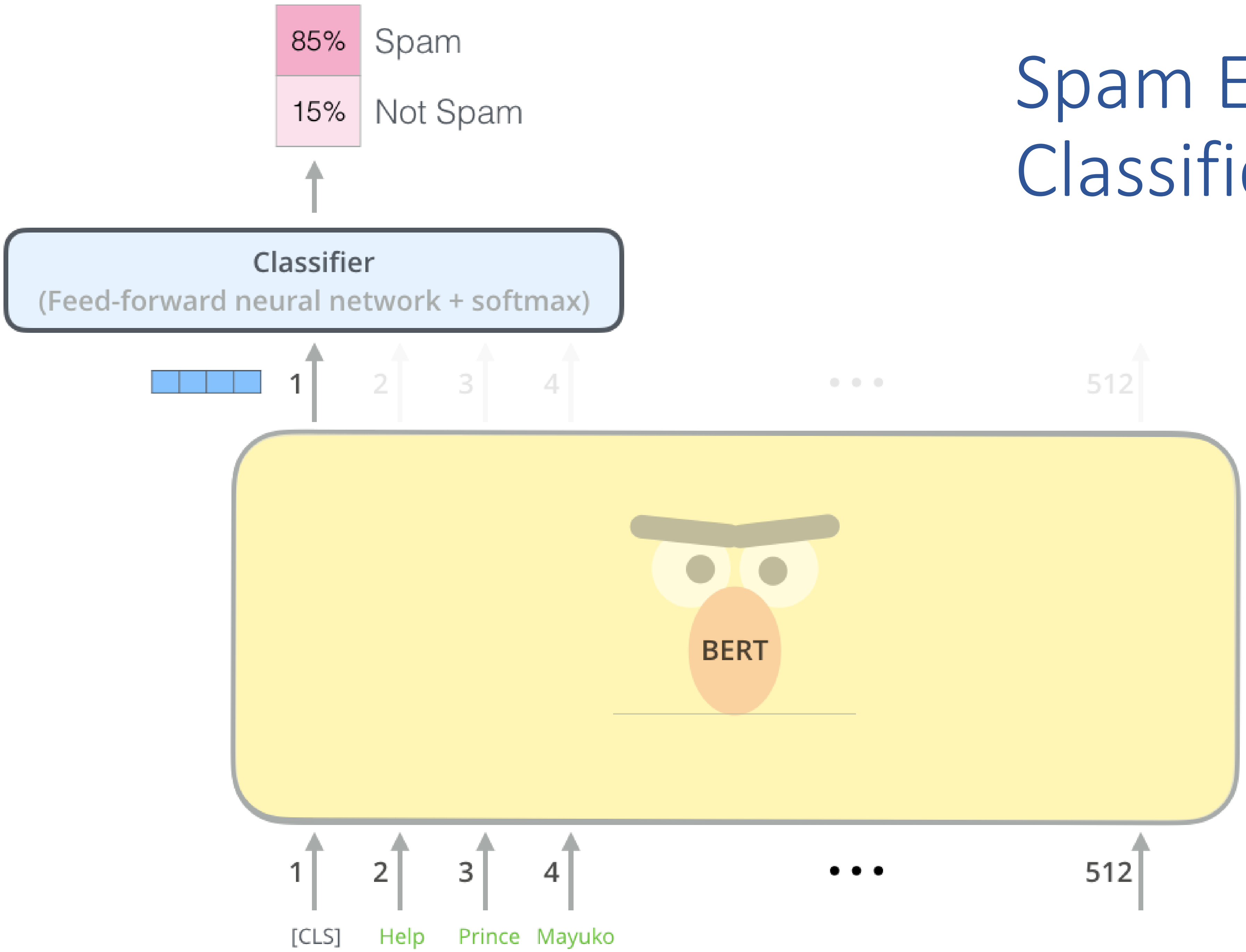


Dataset:

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

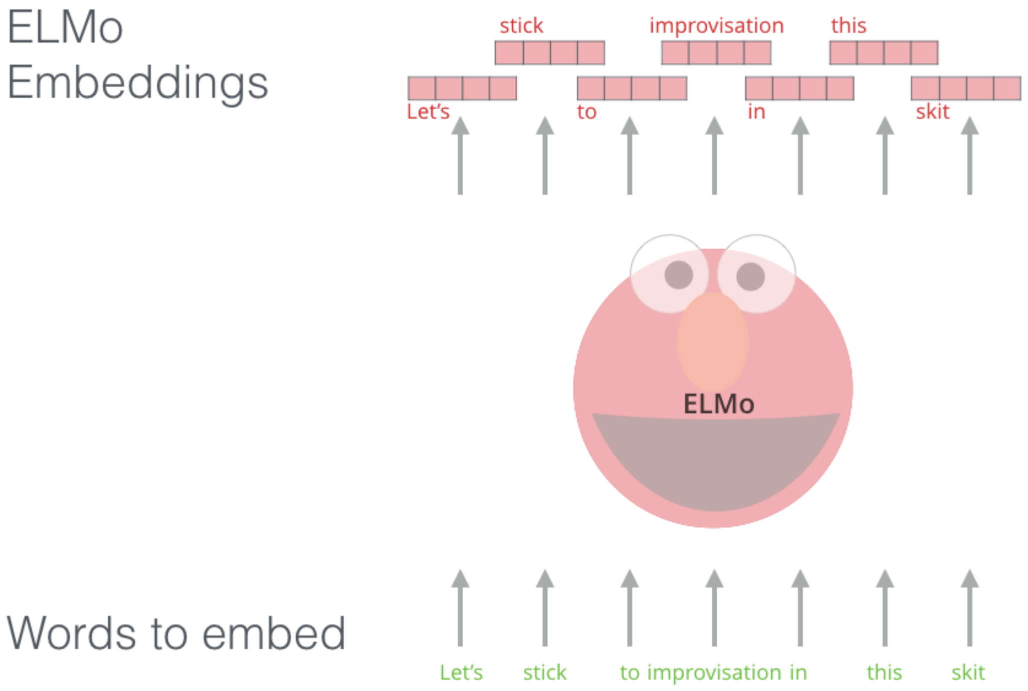
https://jalammar.github.io/illustrated-bert/

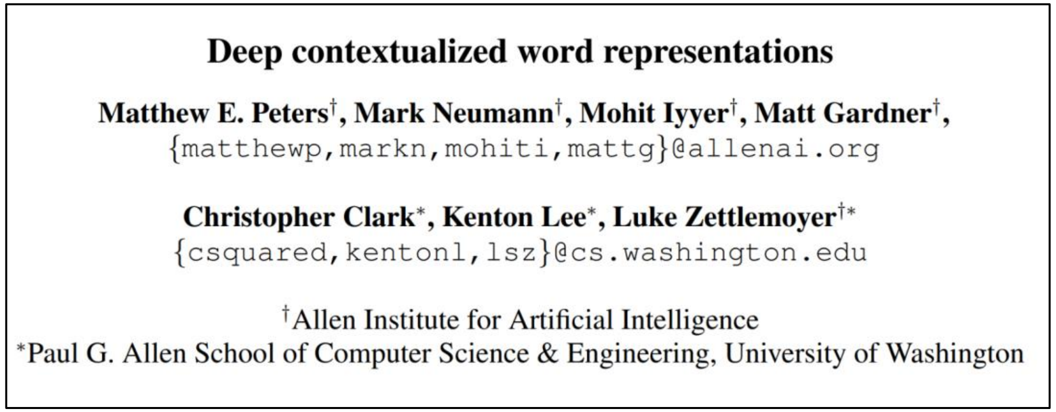
Spam Email Classification



Embeddings from Language Models (ELMo)

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




Deep contextualized word representations

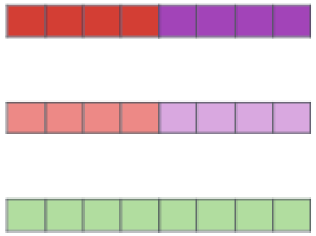
Matthew E. Peters†, **Mark Neumann**†, **Mohit Iyyer**†, **Matt Gardner**†,
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark*, **Kenton Lee***, **Luke Zettlemoyer**†*
{csquared,kentonl,lsz}@cs.washington.edu

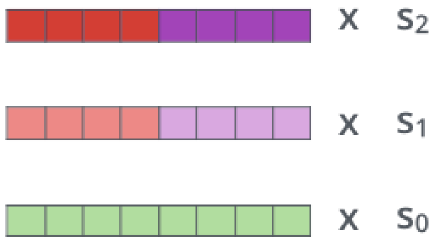
†Allen Institute for Artificial Intelligence
*Paul G. Allen School of Computer Science & Engineering, University of Washington

Use Bi-LSTM to create Word Embedding

1- Concatenate hidden layers

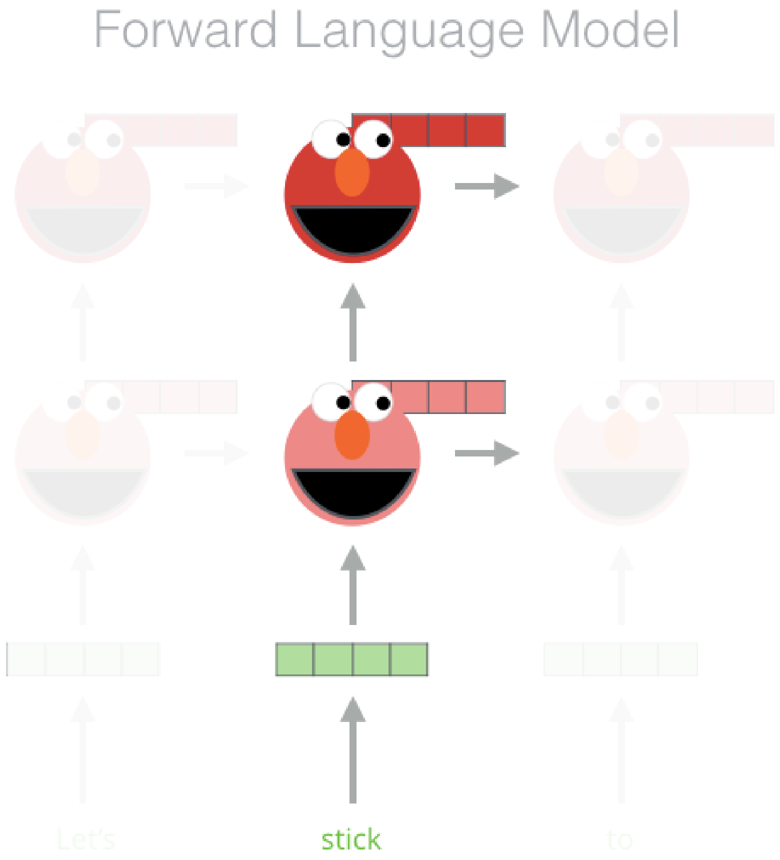


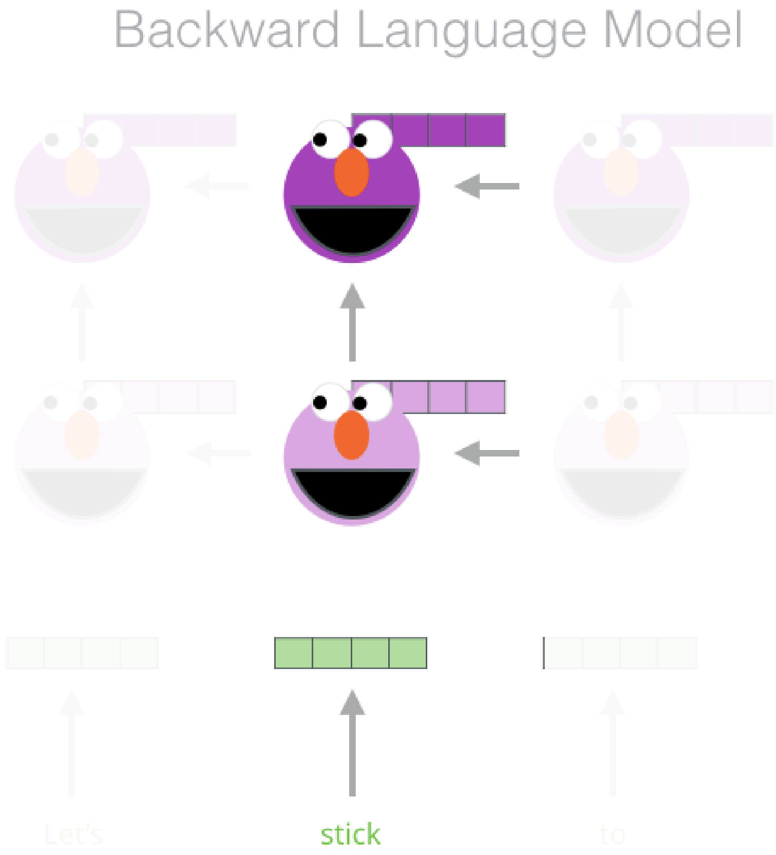
2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors

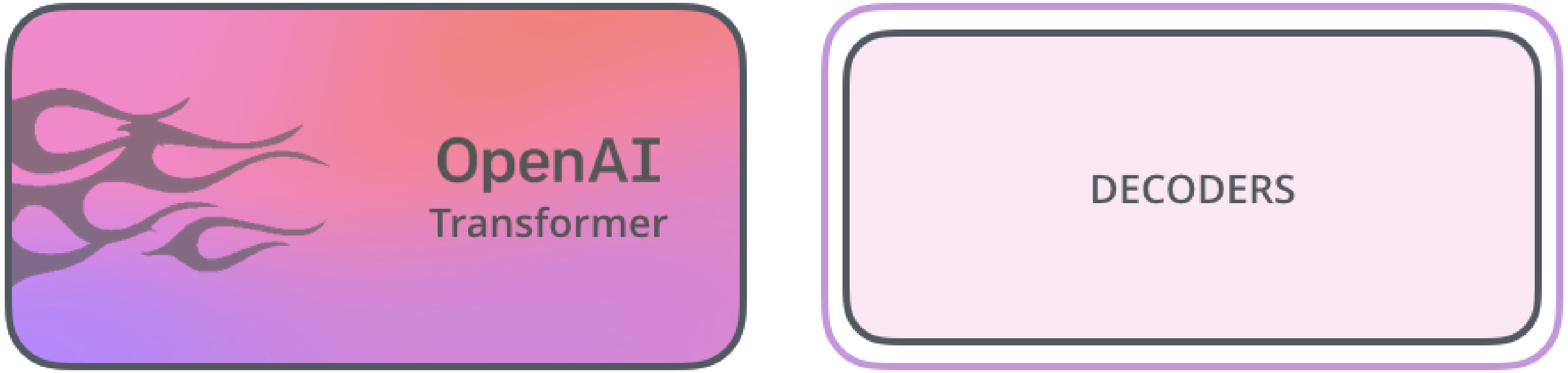
ELMo embedding of "stick" for this task in this context



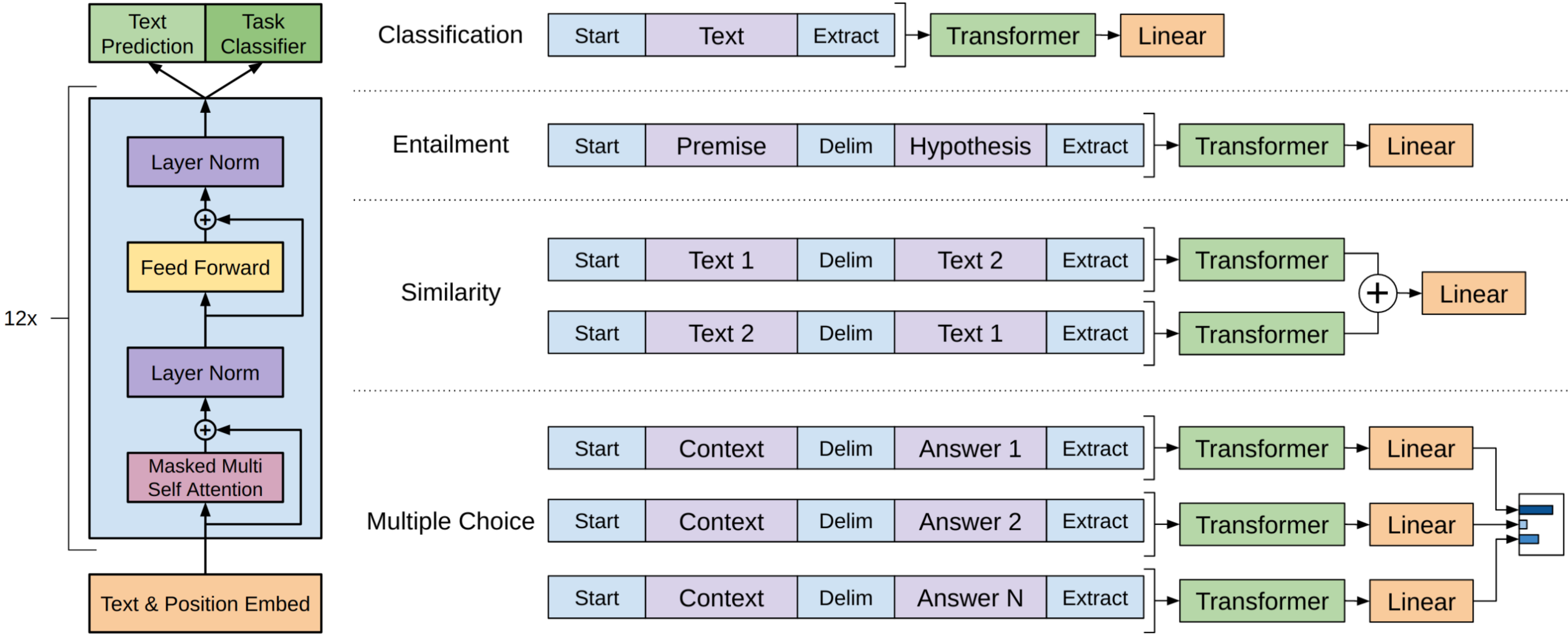


OpenAI GPT: Pre-training Transformer Decoders

- 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



https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

OpenAI GPT-2

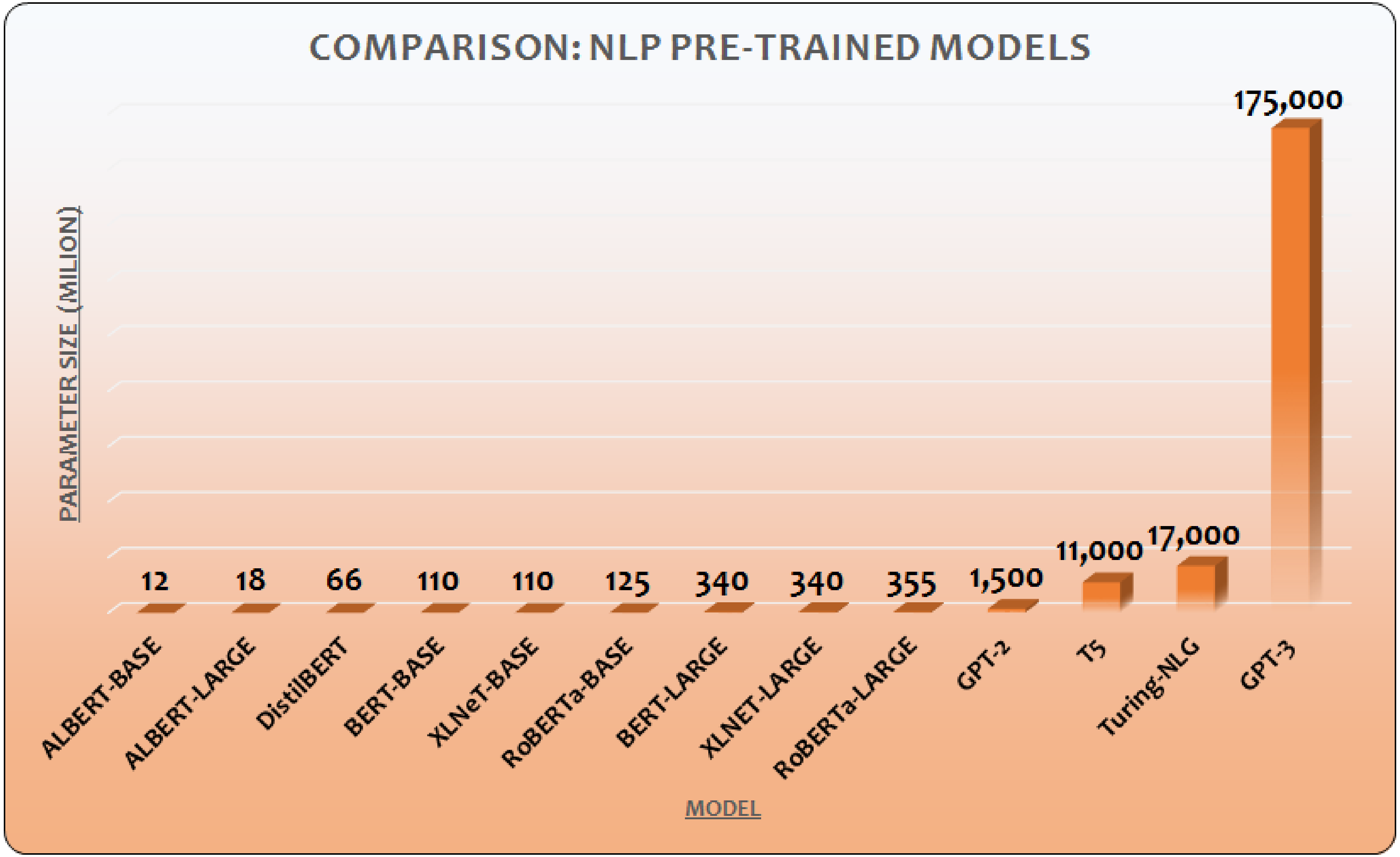
- Pre-trained using 40GB of Internet text
- Scale-up of GPT with 10X parameters trained with 10X data
- Other tricks
  - Layer normalization was moved to the input of each sub-block
  - An additional layer normalization was added after the final self-attention block

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

https://openai.com/blog/better-language-models/

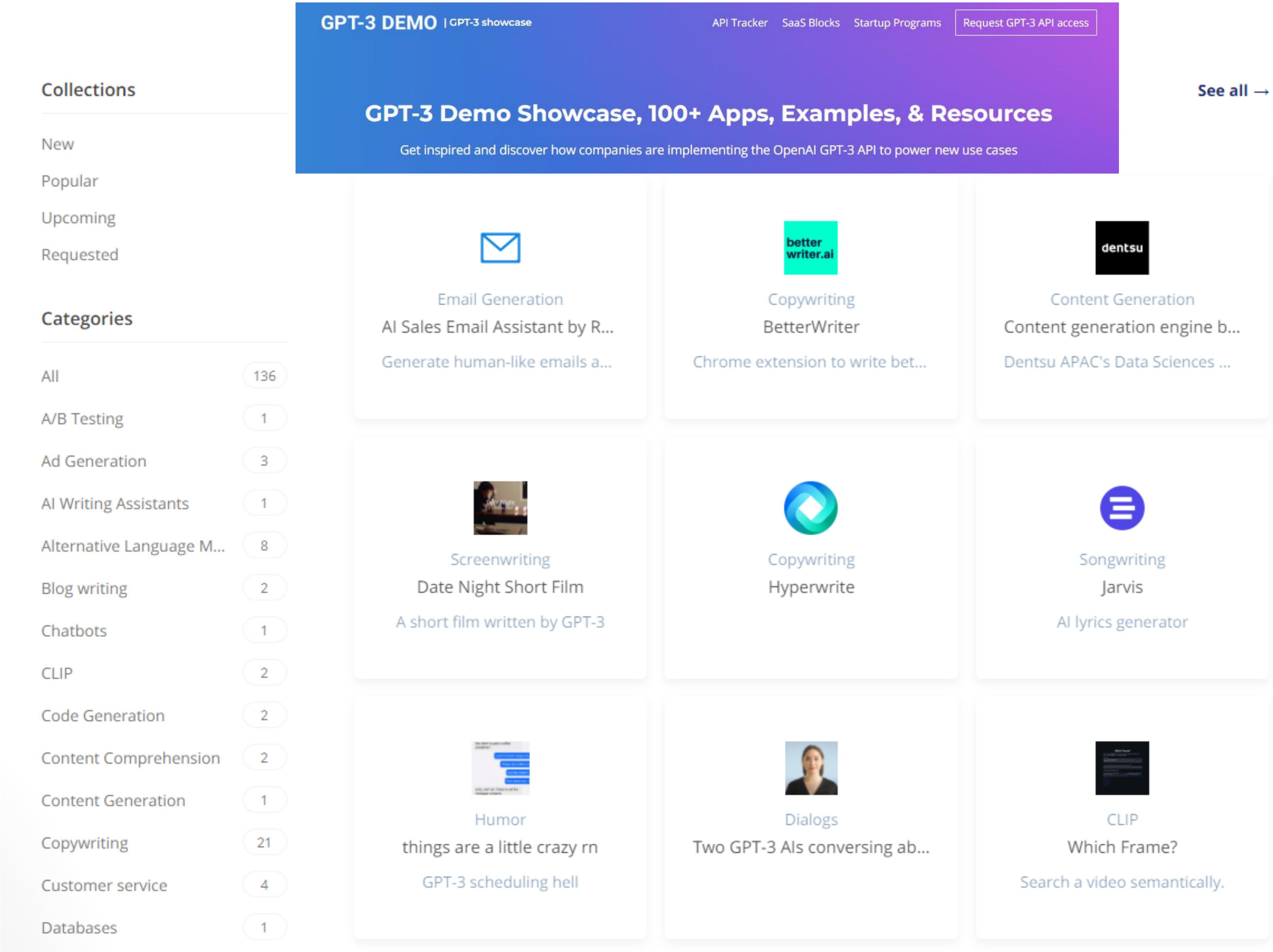
Size does Matter! GPT-3

- 175 Billion Parameters!
- 175×4=700GB
- 55 years and $4,600,000 to train - even with the [lowest priced GPU cloud on the market](https://medium.com/analytics-vidhya/openai-gpt-3-language-models-are-few-shot-learners-82531b3d3122).

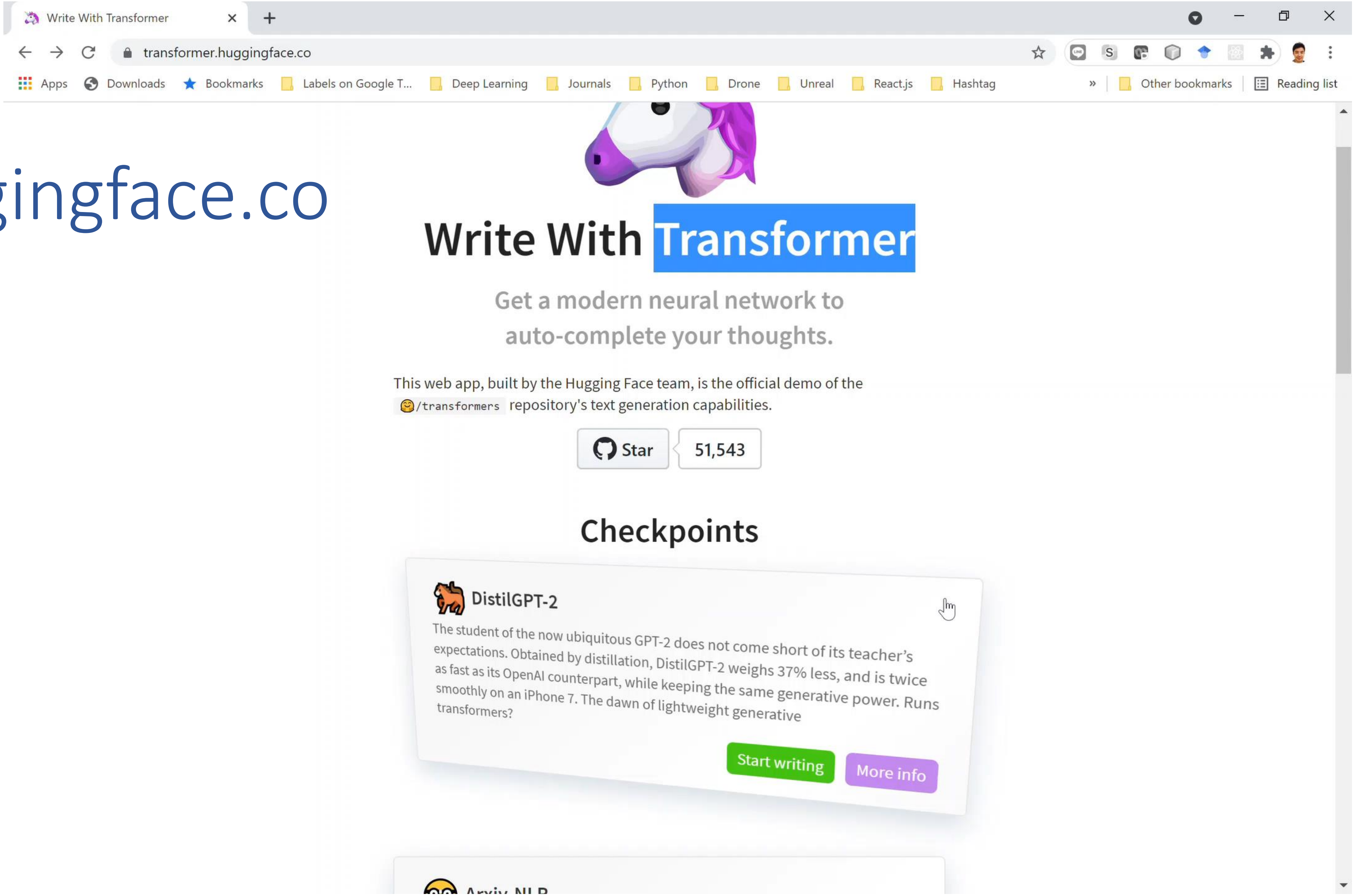


https://medium.com/analytics-vidhya/openai-gpt-3-language-models-are-few-shot-learners-82531b3d3122

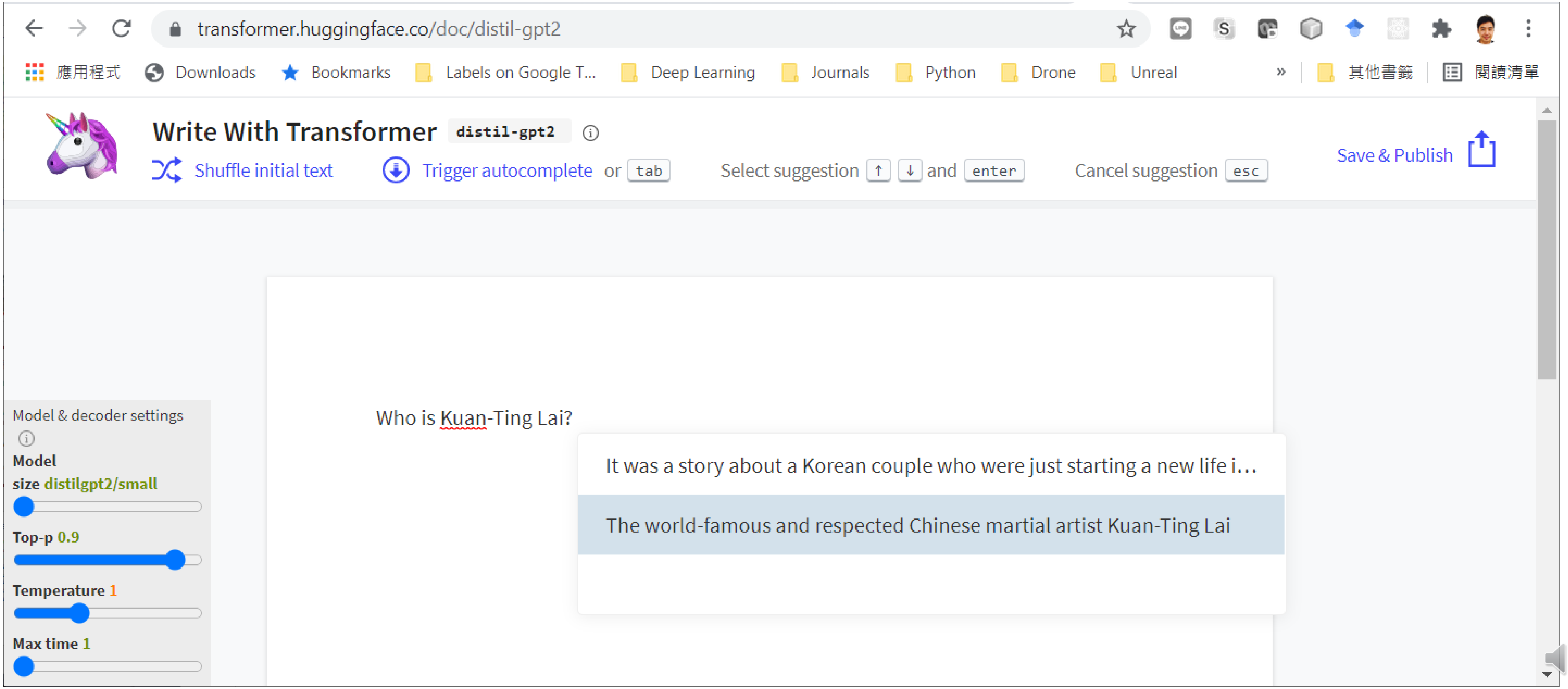
GPT3 Demo (gpt3demo.com)



huggingface.co



huggingface.co



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