Attention, Transformer and BERT

Prof. Kuan-Ting Lai

2021/4/17

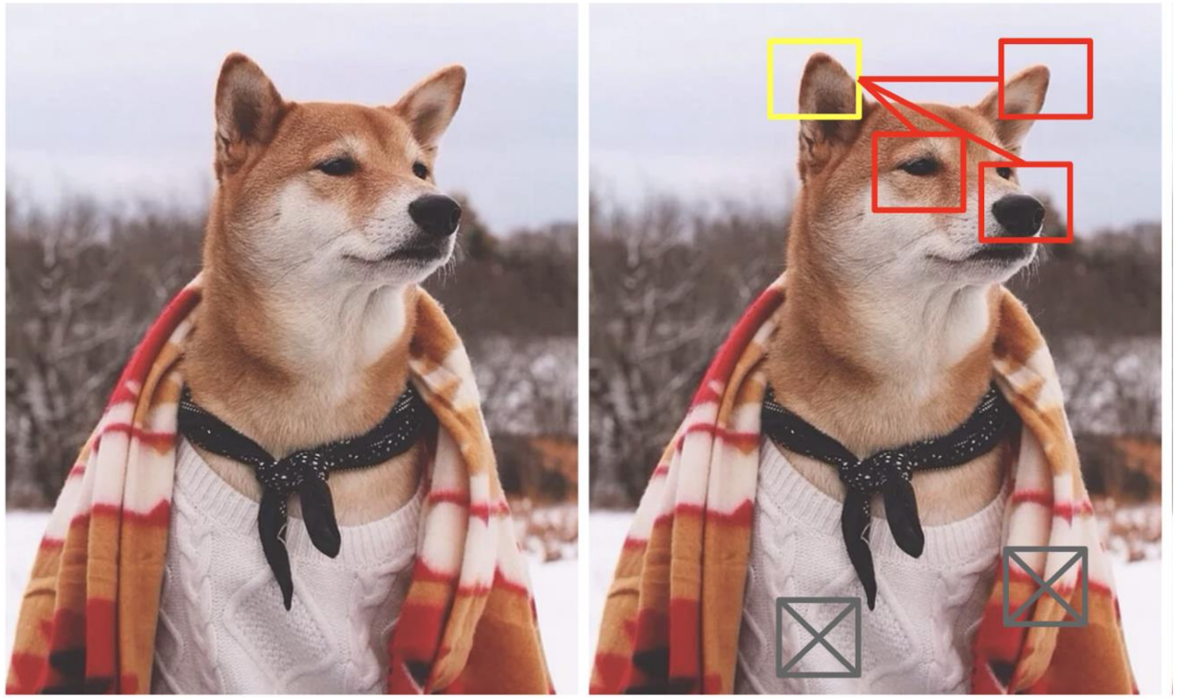
Attention is All You Need!

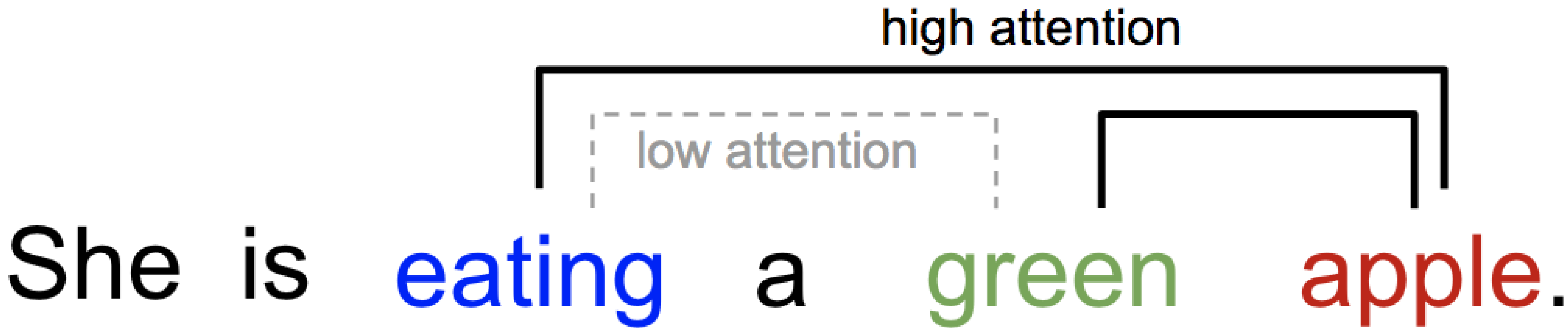
A. Waswani et al., *NIPS*, 2017

Google Brain & University of Toronto

Attention

- Visual attention and textual attention

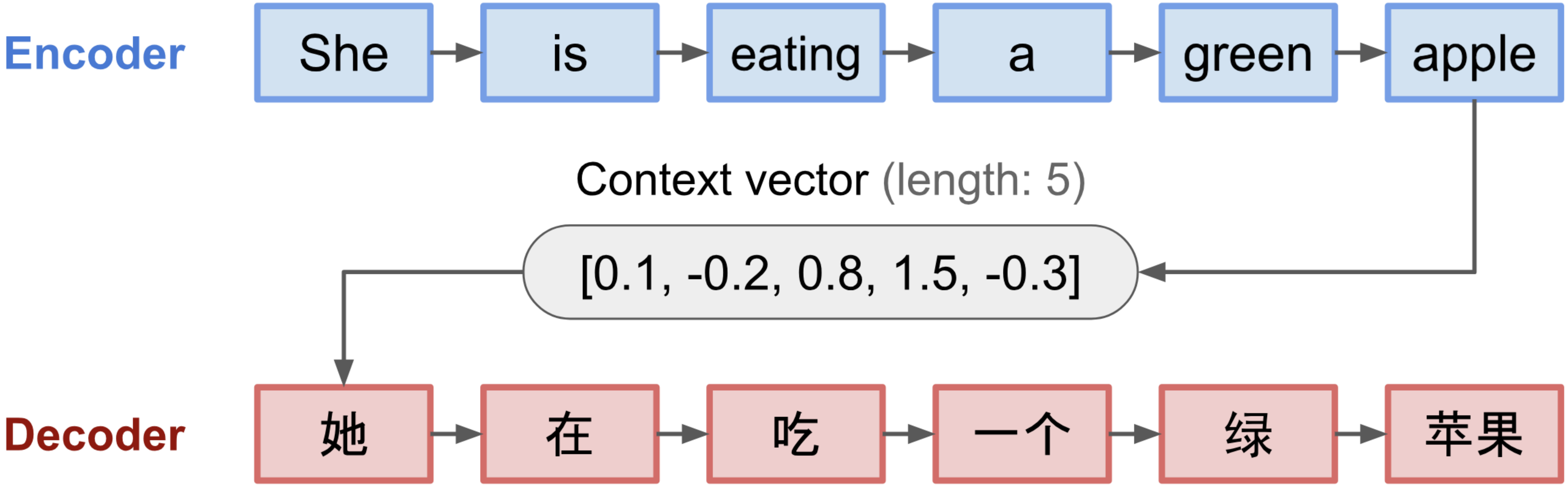




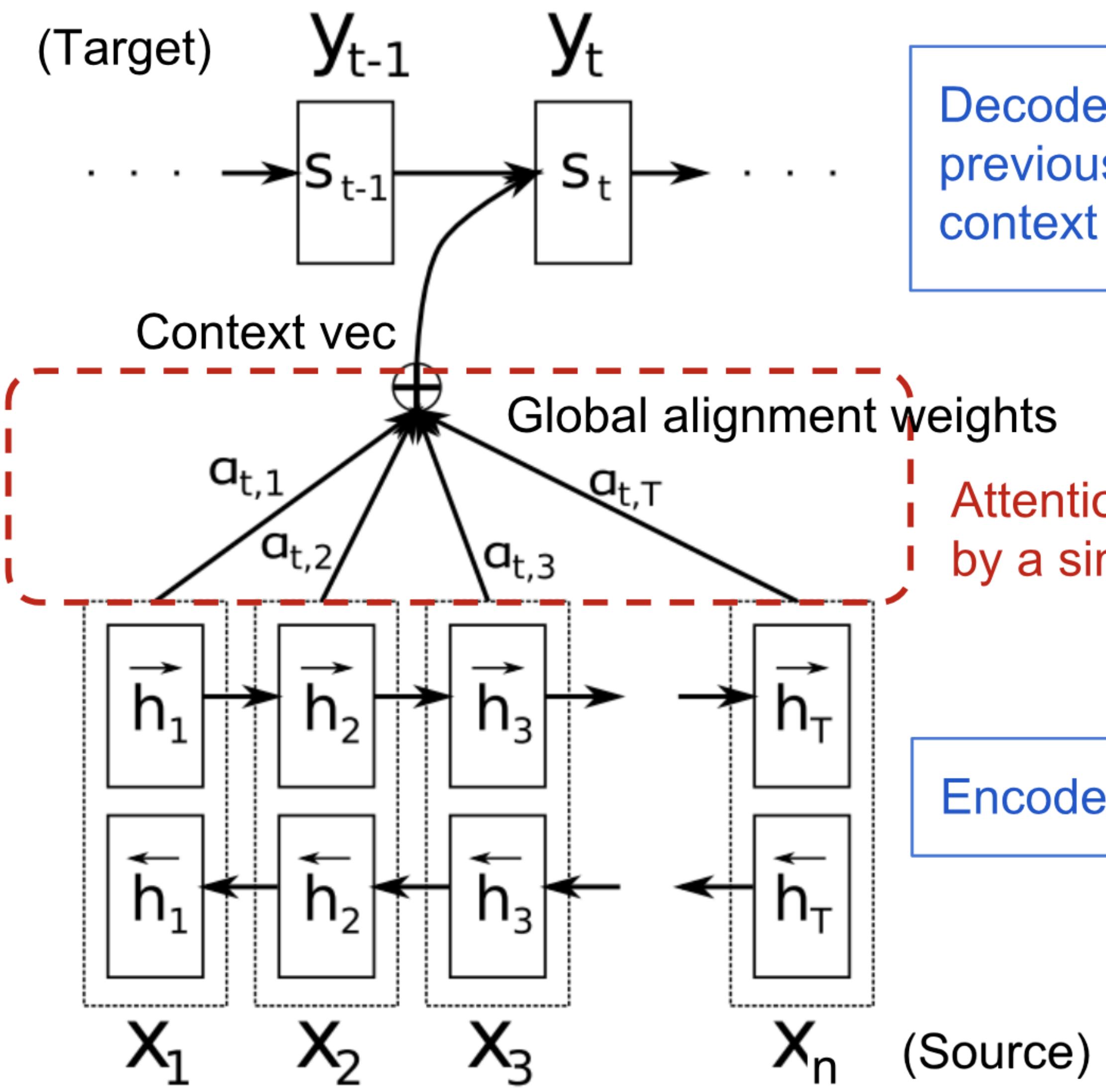
https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Seq2seq model

- Language translation



Attention = Vector of Importance Weights



Decoder: RNN with input from previous state + dynamic context vector.

Attention layer: parameterized by a simple feed-forward network

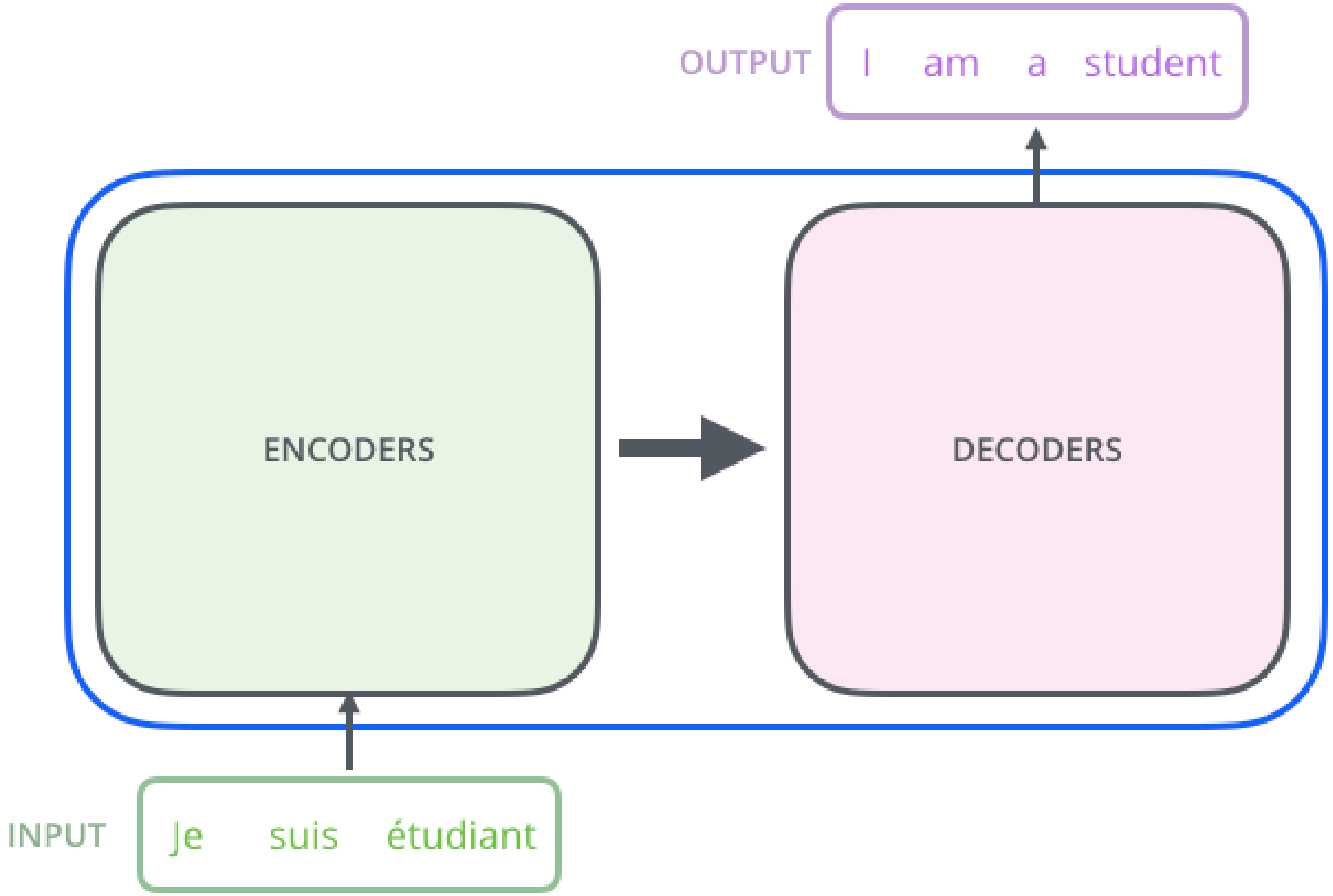
Additive Attention

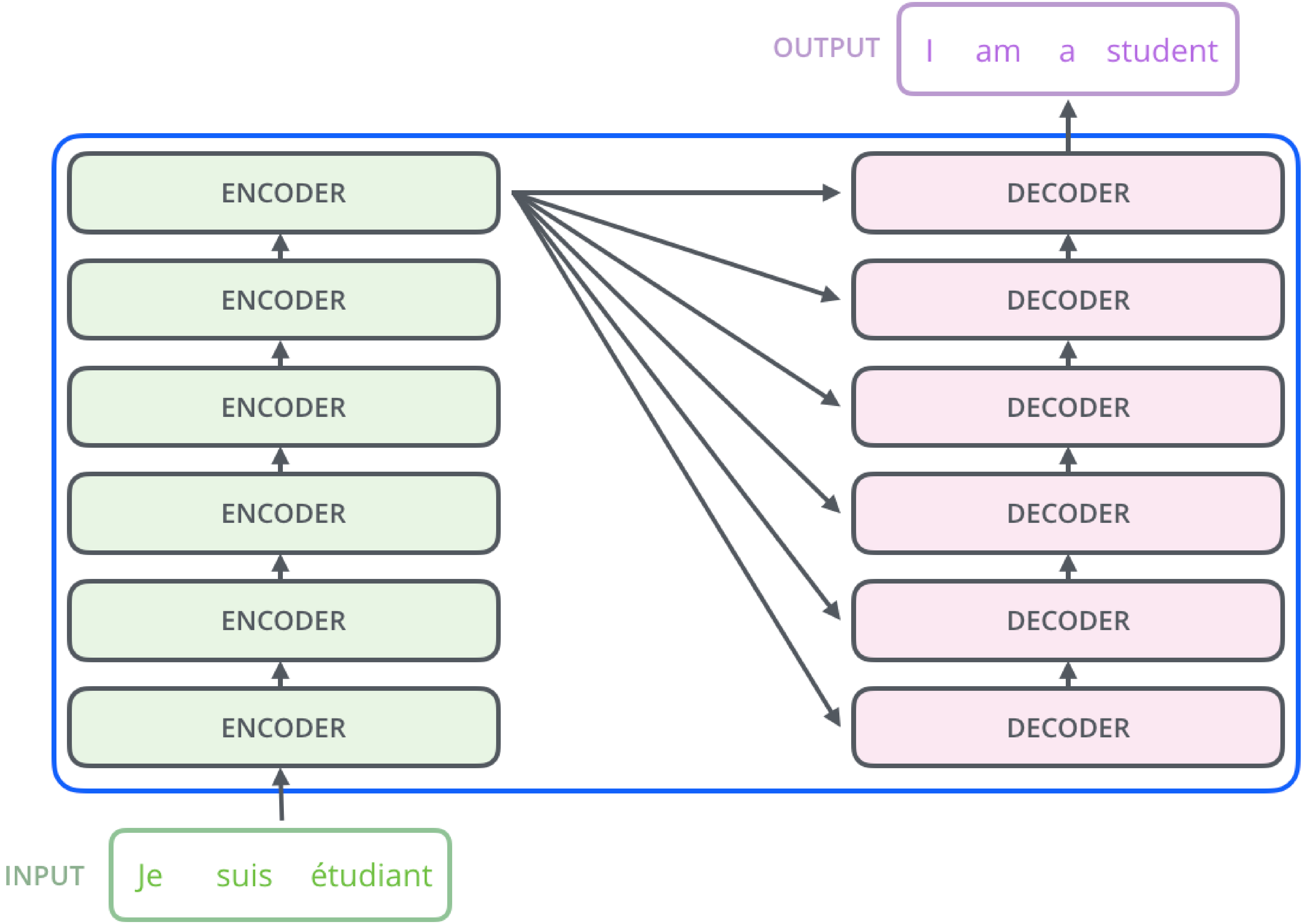
Encoder: bidirectional RNN

Transformer

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

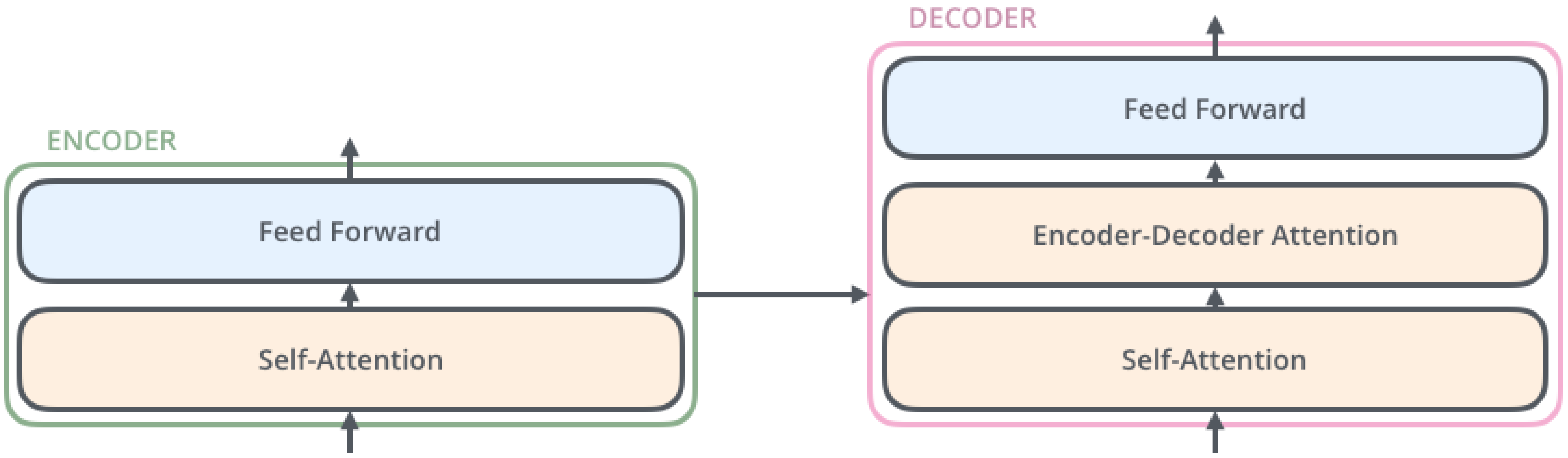
Encoder and Decoder

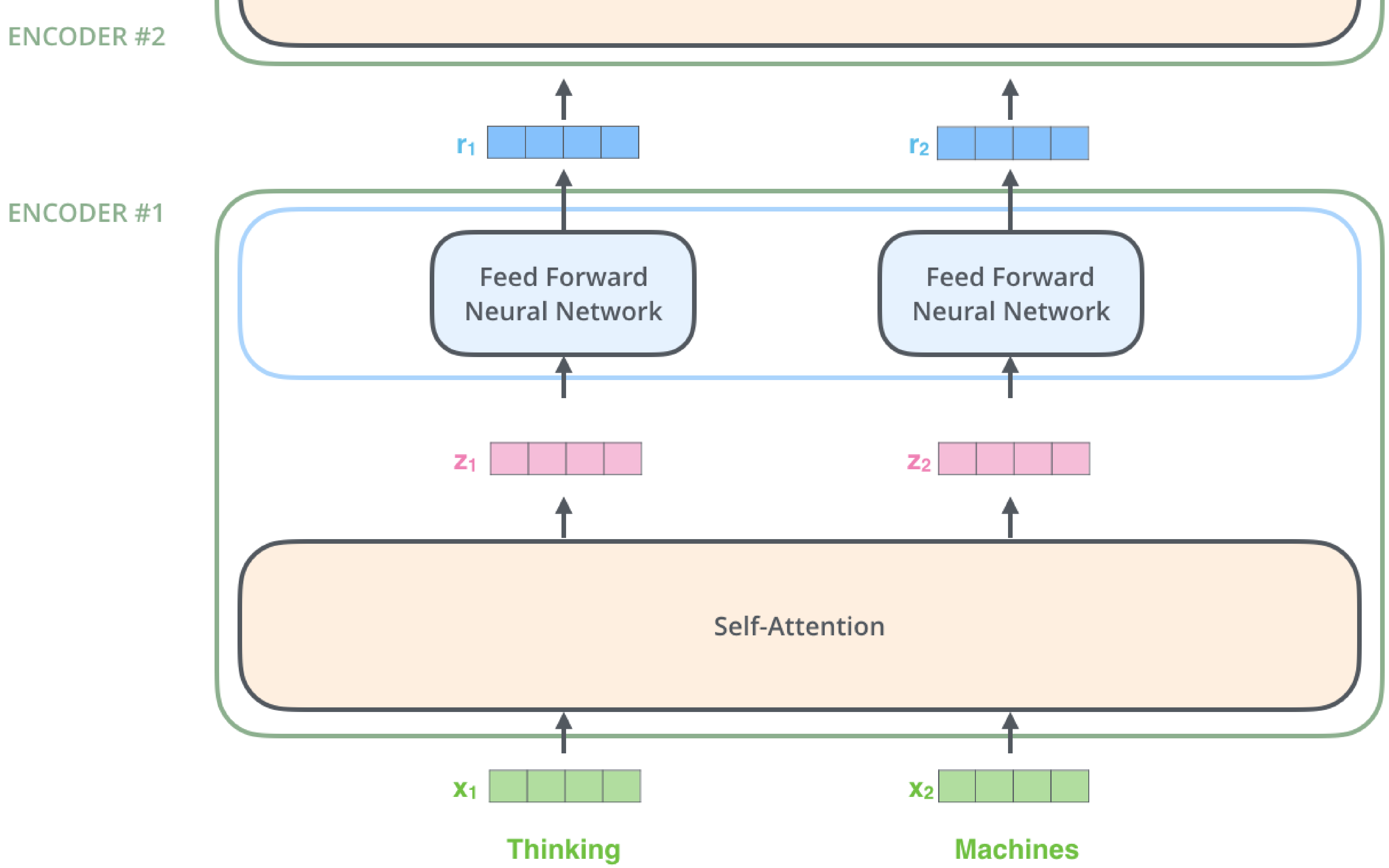



OUTPUT
I am a student
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant

Structure of the Encoder and Decoder

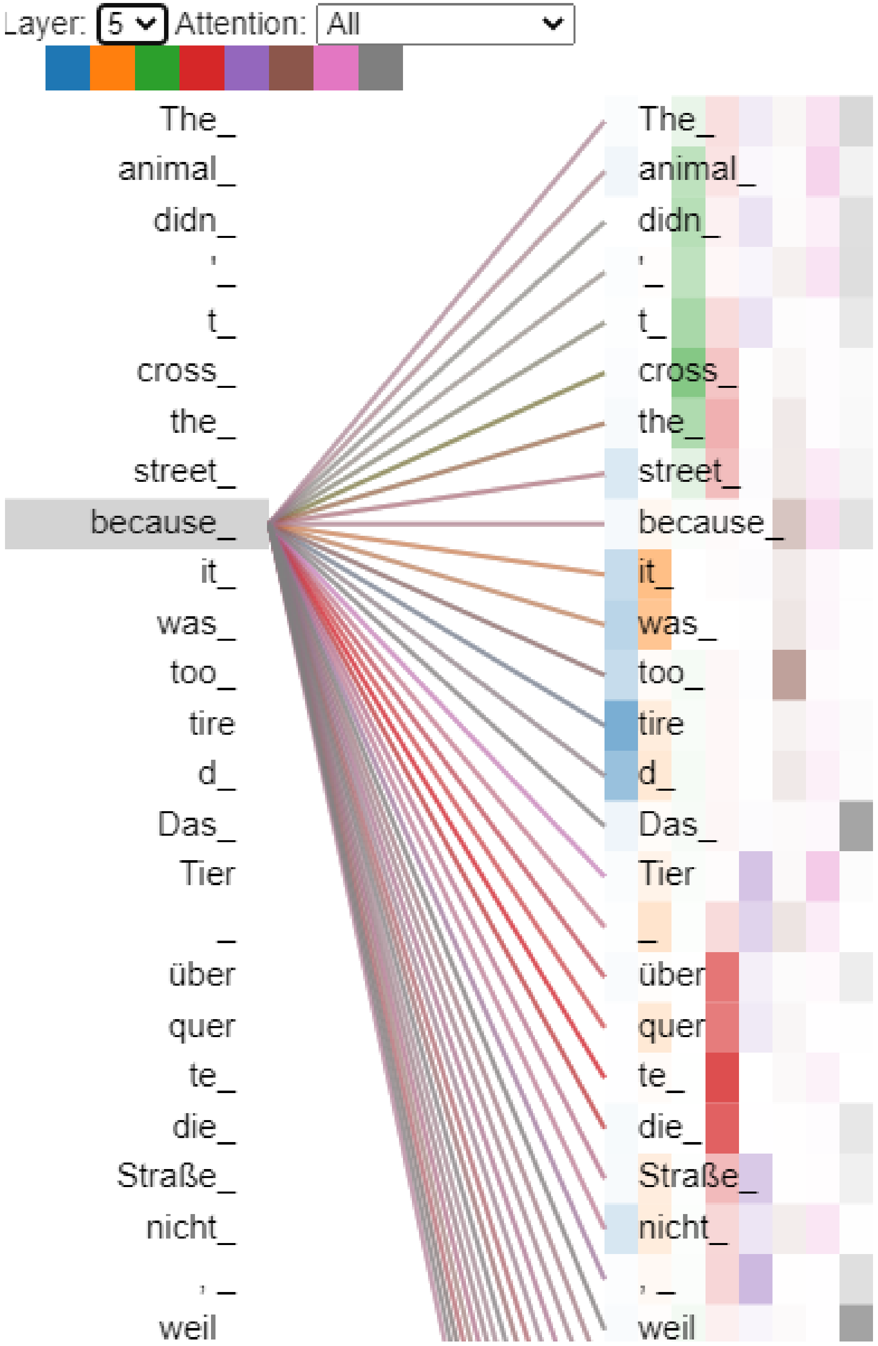
- Self-attention
- Encoder-decoder attention




ENCODER #2
r1
r2
ENCODER #1
Feed Forward Neural Network
Feed Forward Neural Network
z1
z2
Self-Attention
x1
x2
Thinking
Machines

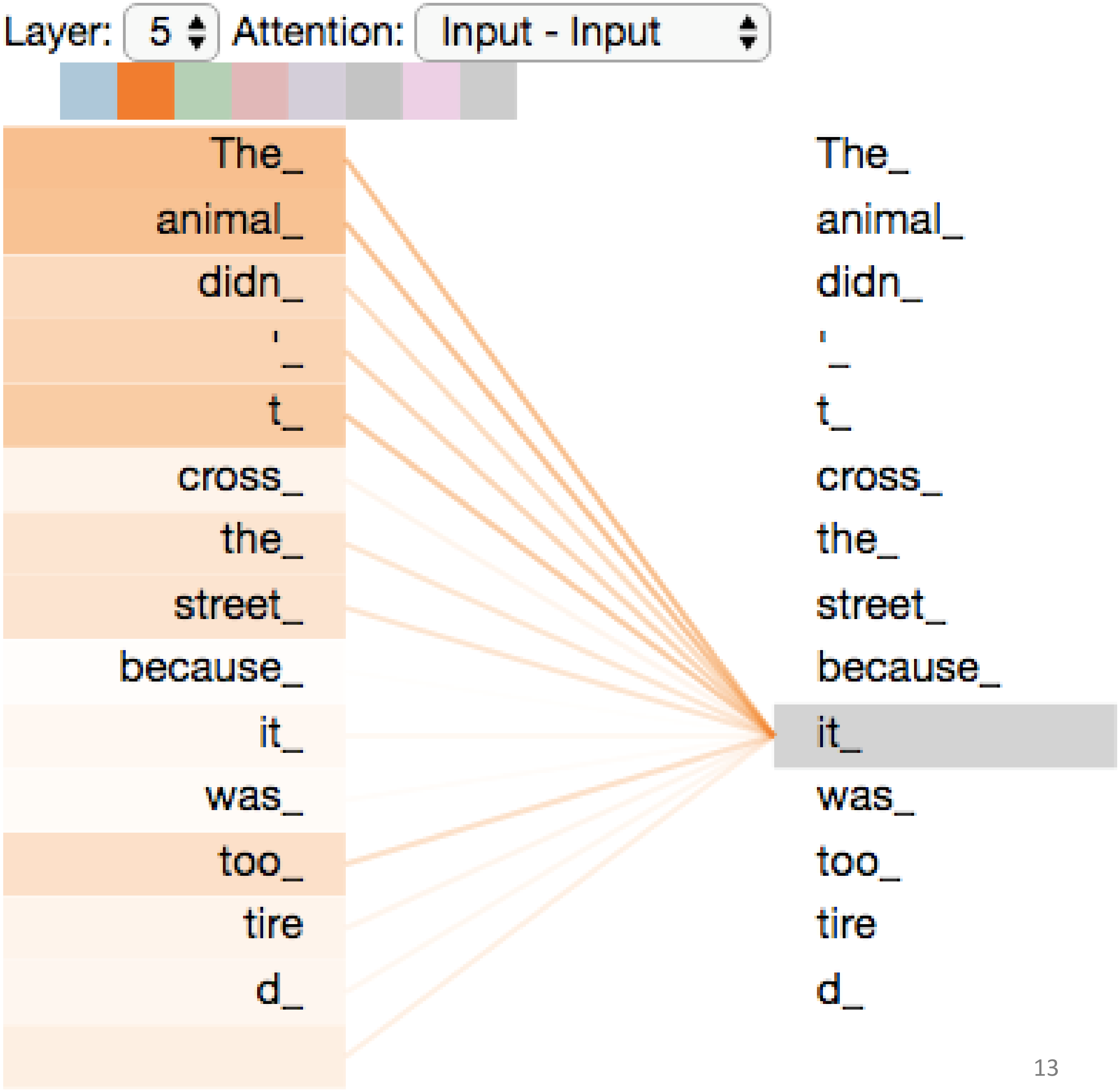
Tensor2Tensor Notebook

- https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb



Self Attention

Self-attention Example



Self-attention (query, key, value)

q: query (to match others)

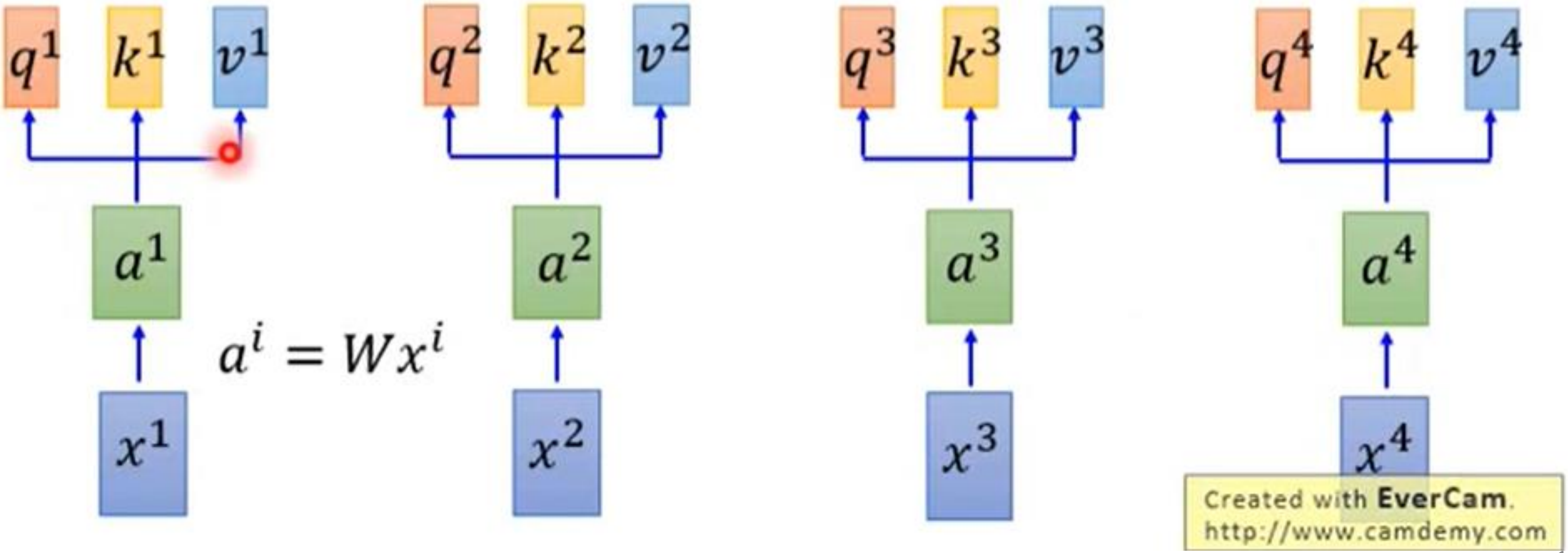
$$q^i = W^q a^i$$

k: key (to be matched)

$$k^i = W^k a^i$$

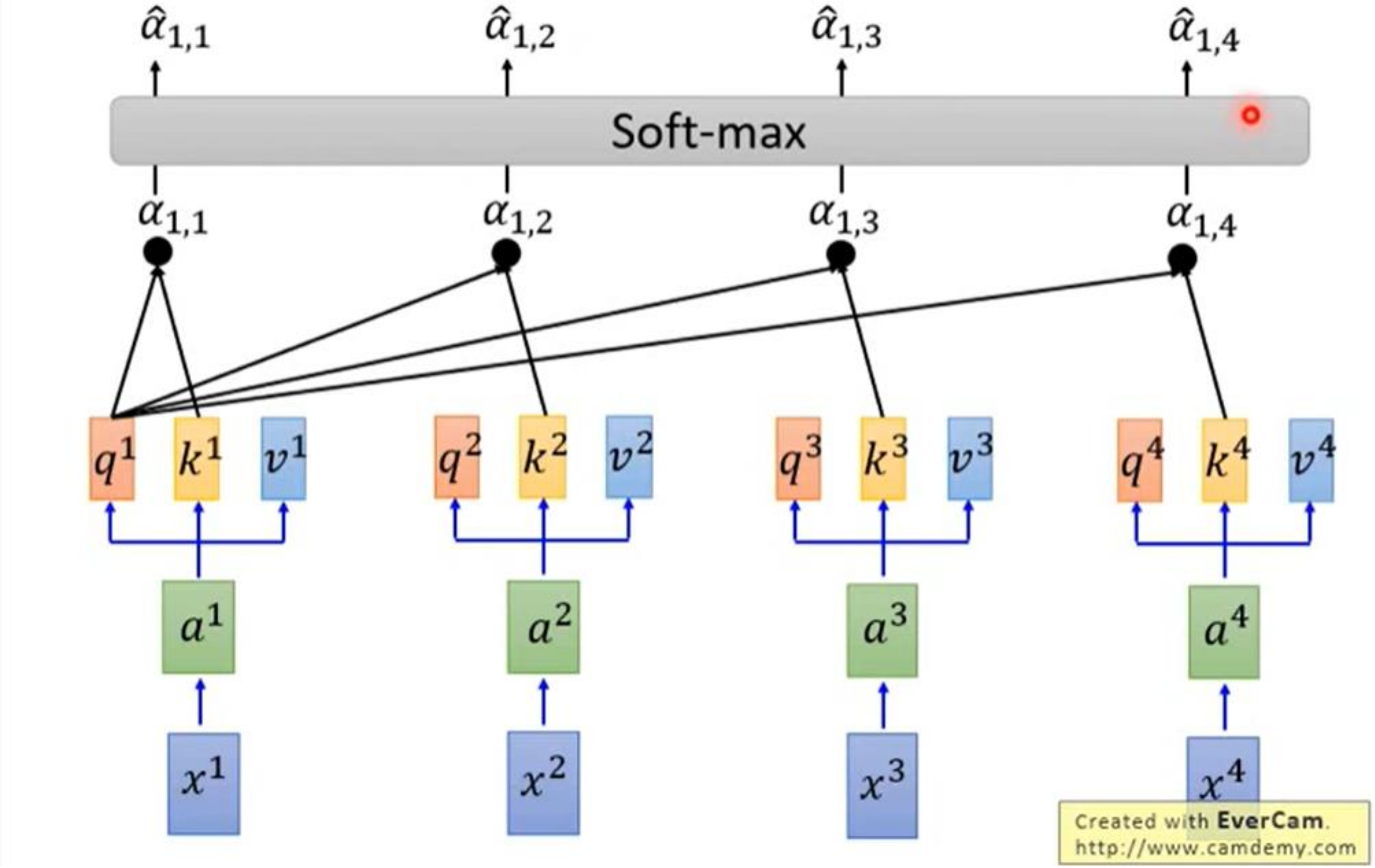
v: information to be extracted

$$v^i = W^v a^i$$



https://www.youtube.com/watch?v=ugWDIIOHtPA&t=1089s

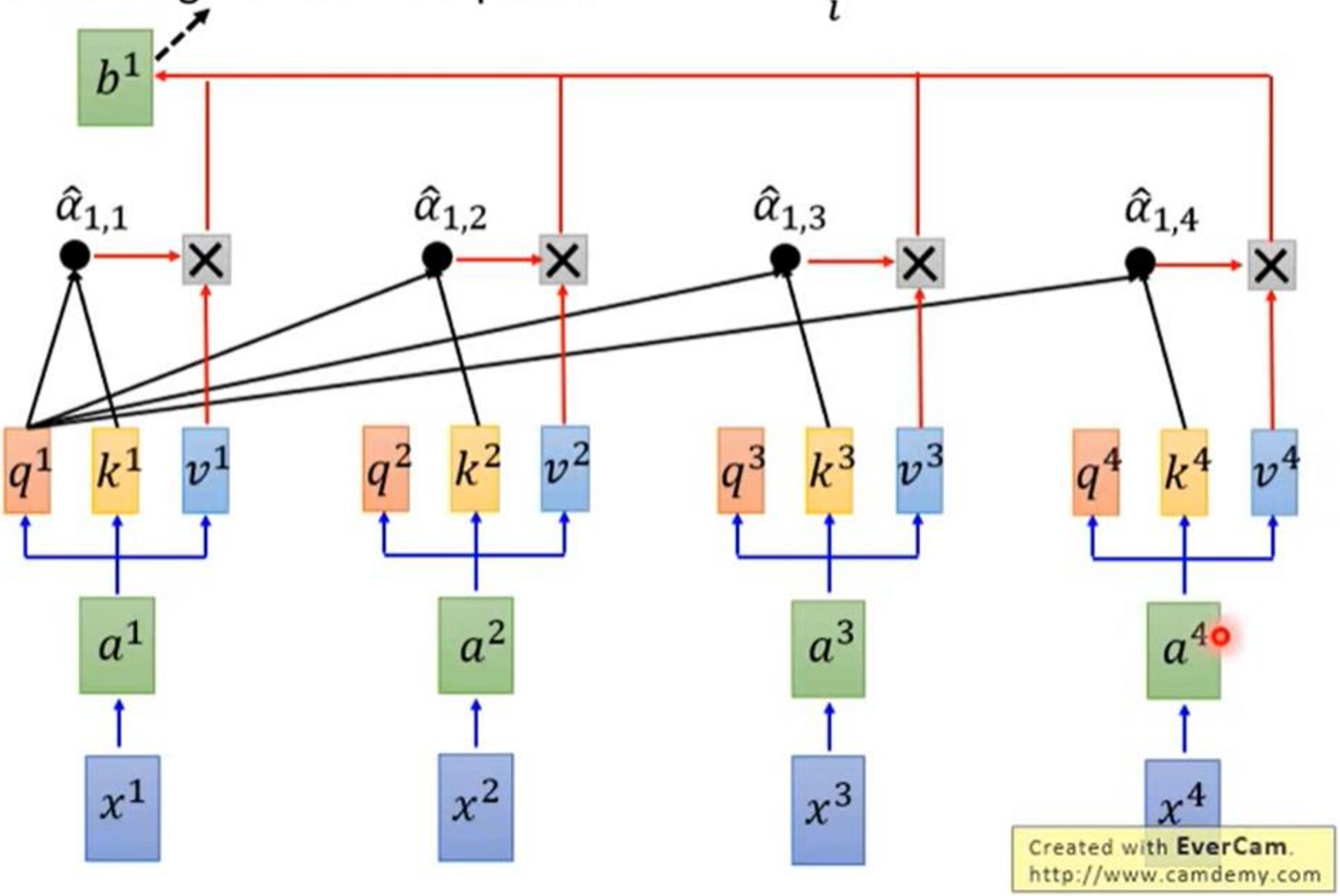
Self-attention



Self-attention

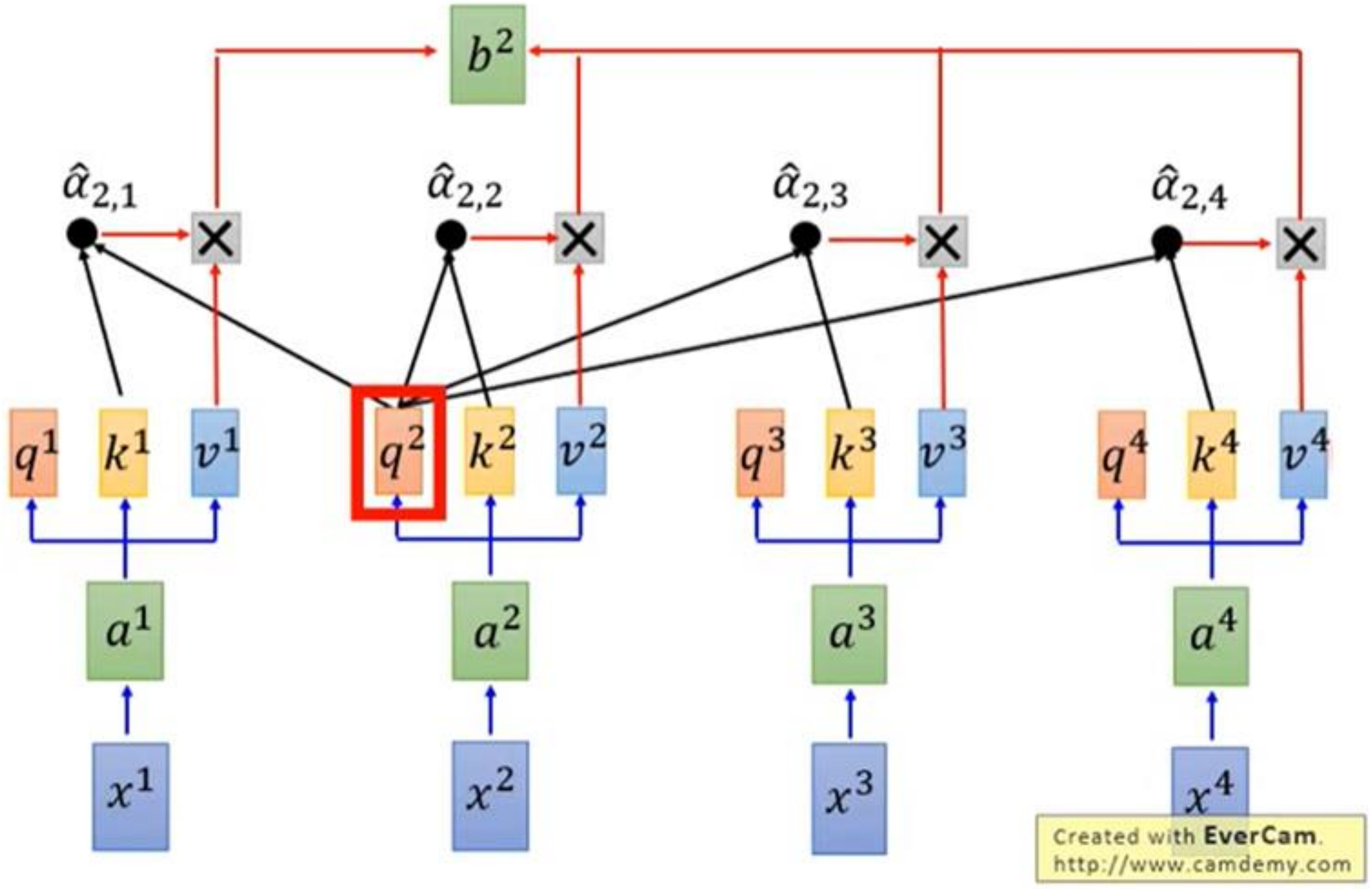
Considering the whole sequence

$$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$

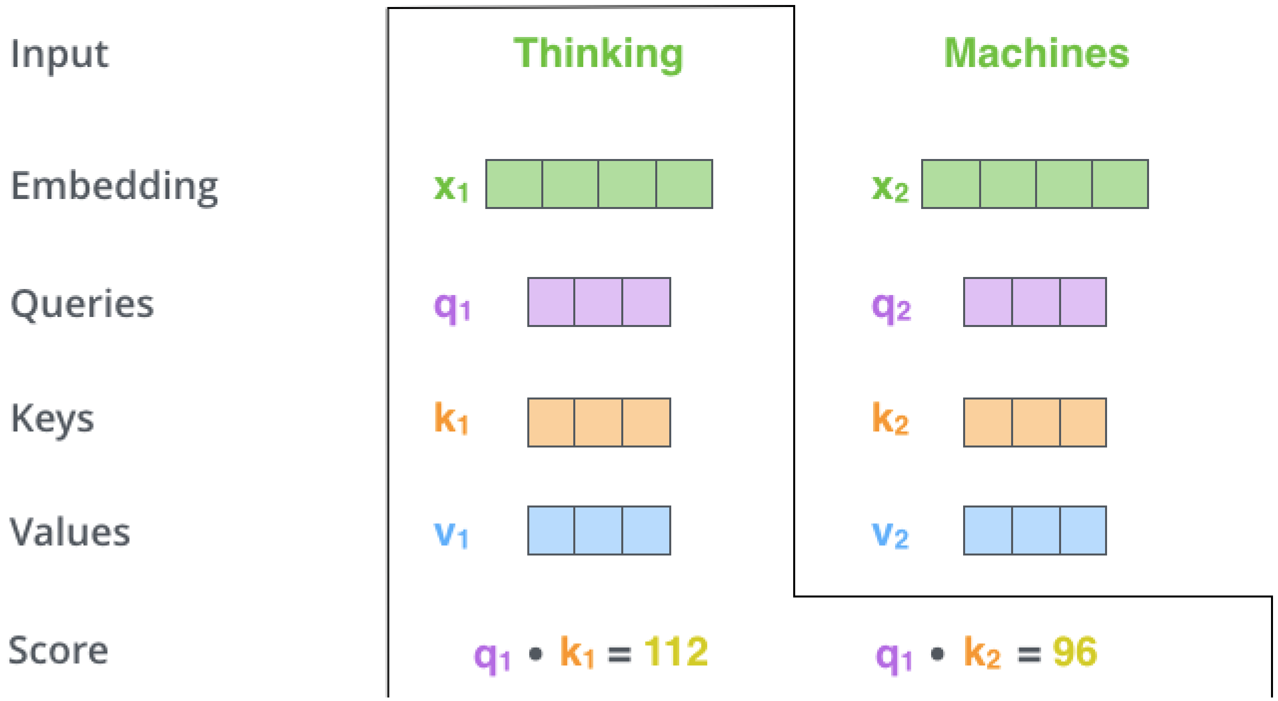


Calculating b^2

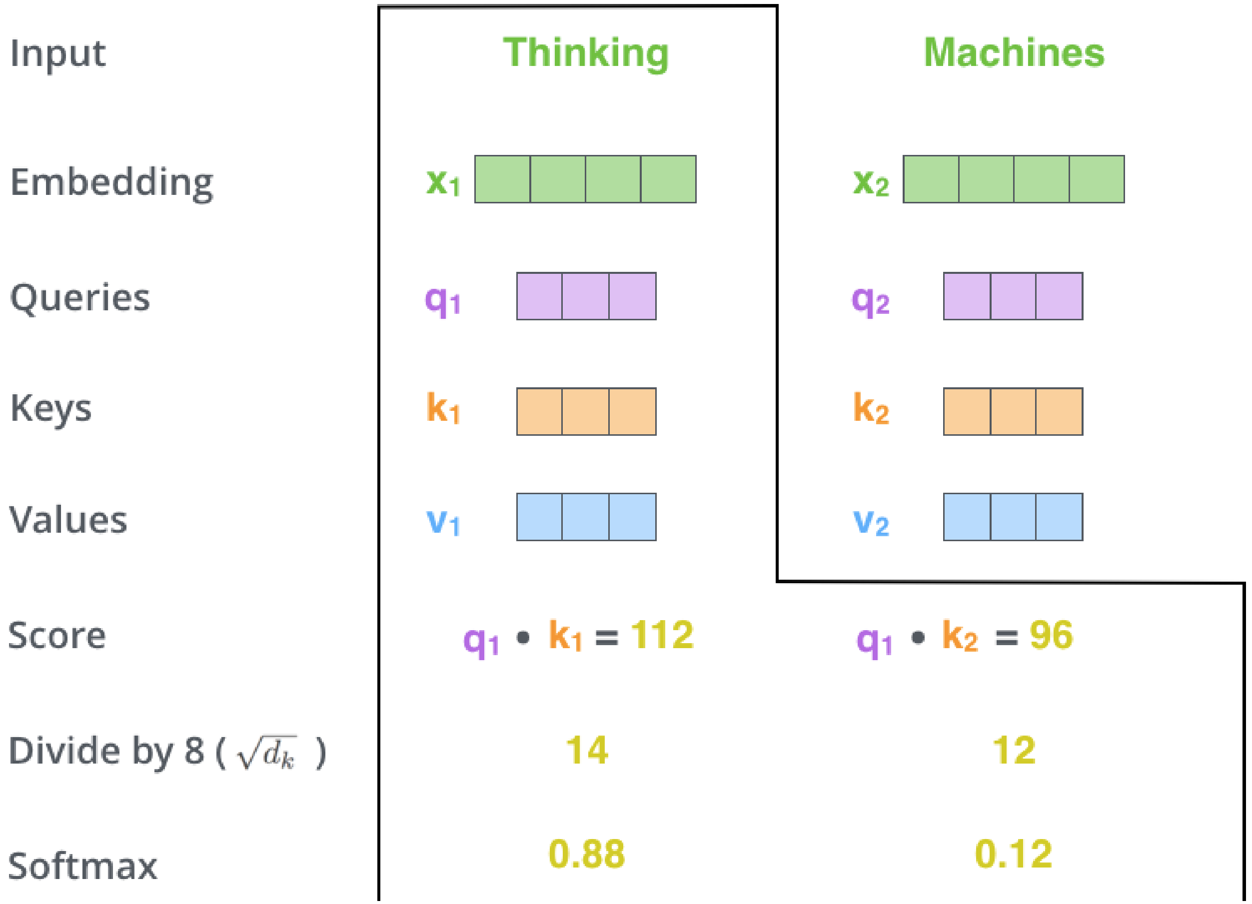
$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



Calculate Dot Product of Query & Value

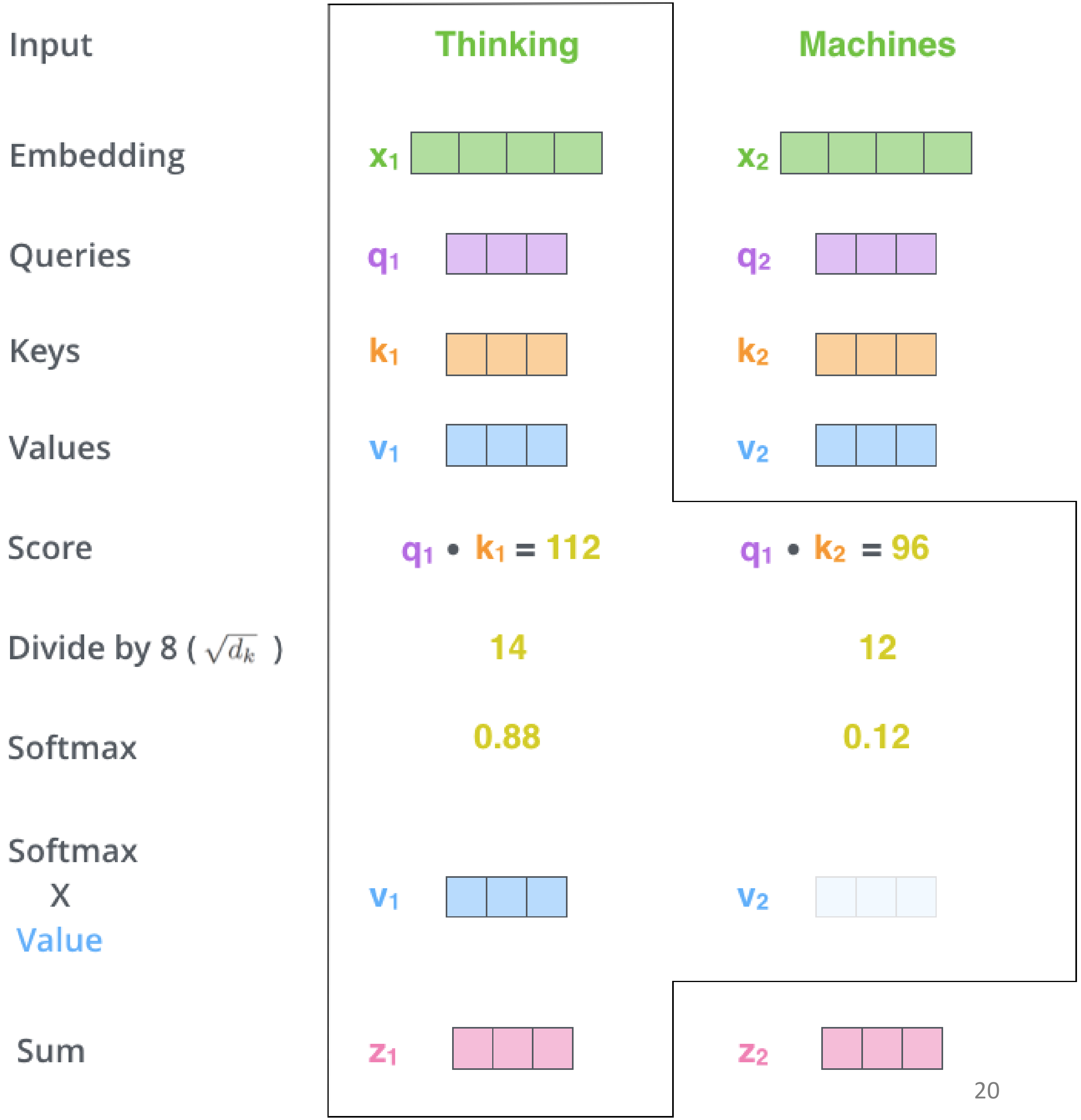


Softmax Normalization



Calculate Values

- Multiply each value vector by the Softmax score
- Sum up the weighted value vectors



Final Output of Self-attention

$$\mathrm{softmax}\left(\frac{Q \times K^{T}}{\sqrt{d_k}}\right) V = Z$$

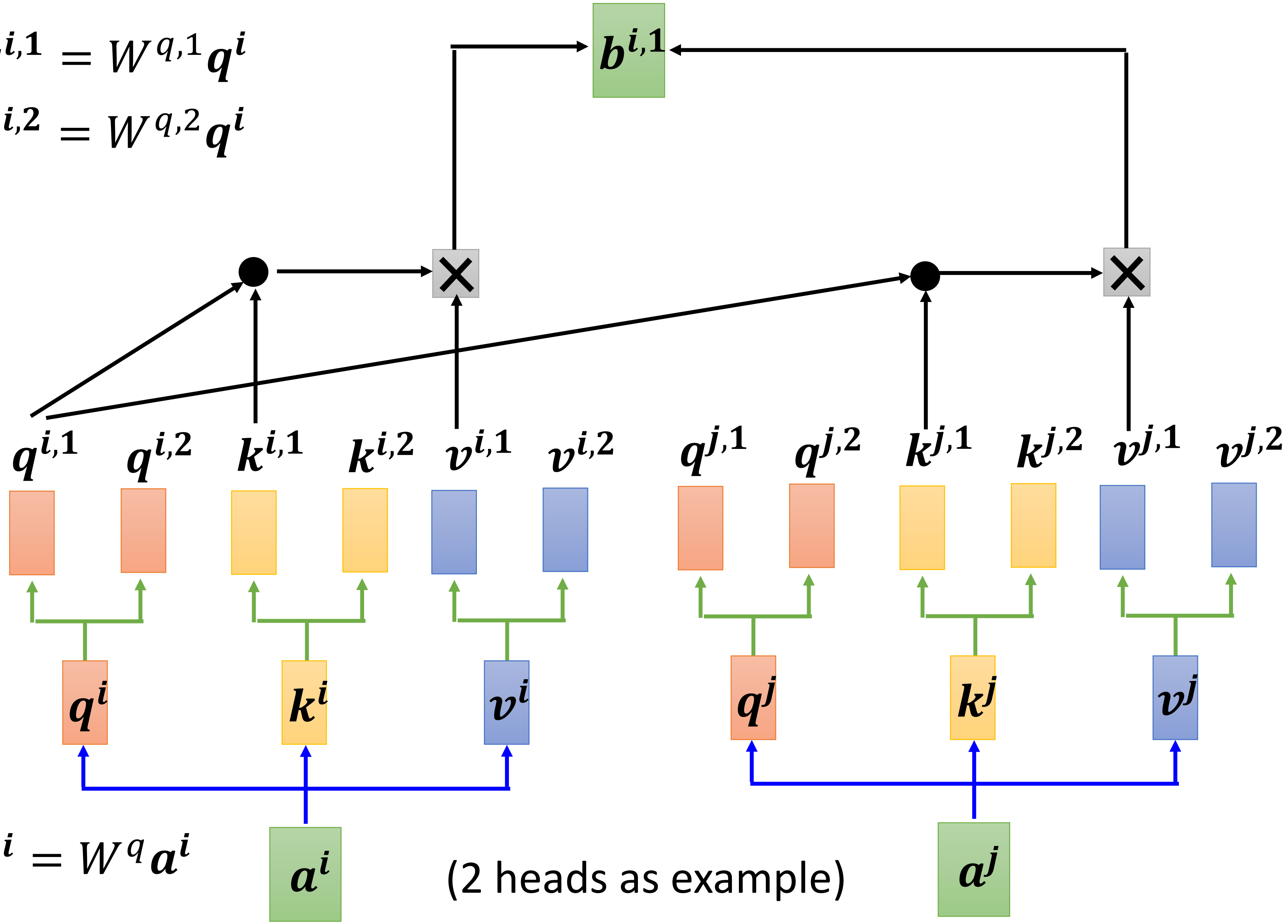
The Beast With Multiple Heads

Multi-head Self-attention

https://speech.ee.ntu.edu.tw/~hylee/ml/ml2021-course-data/self_v7.pptx

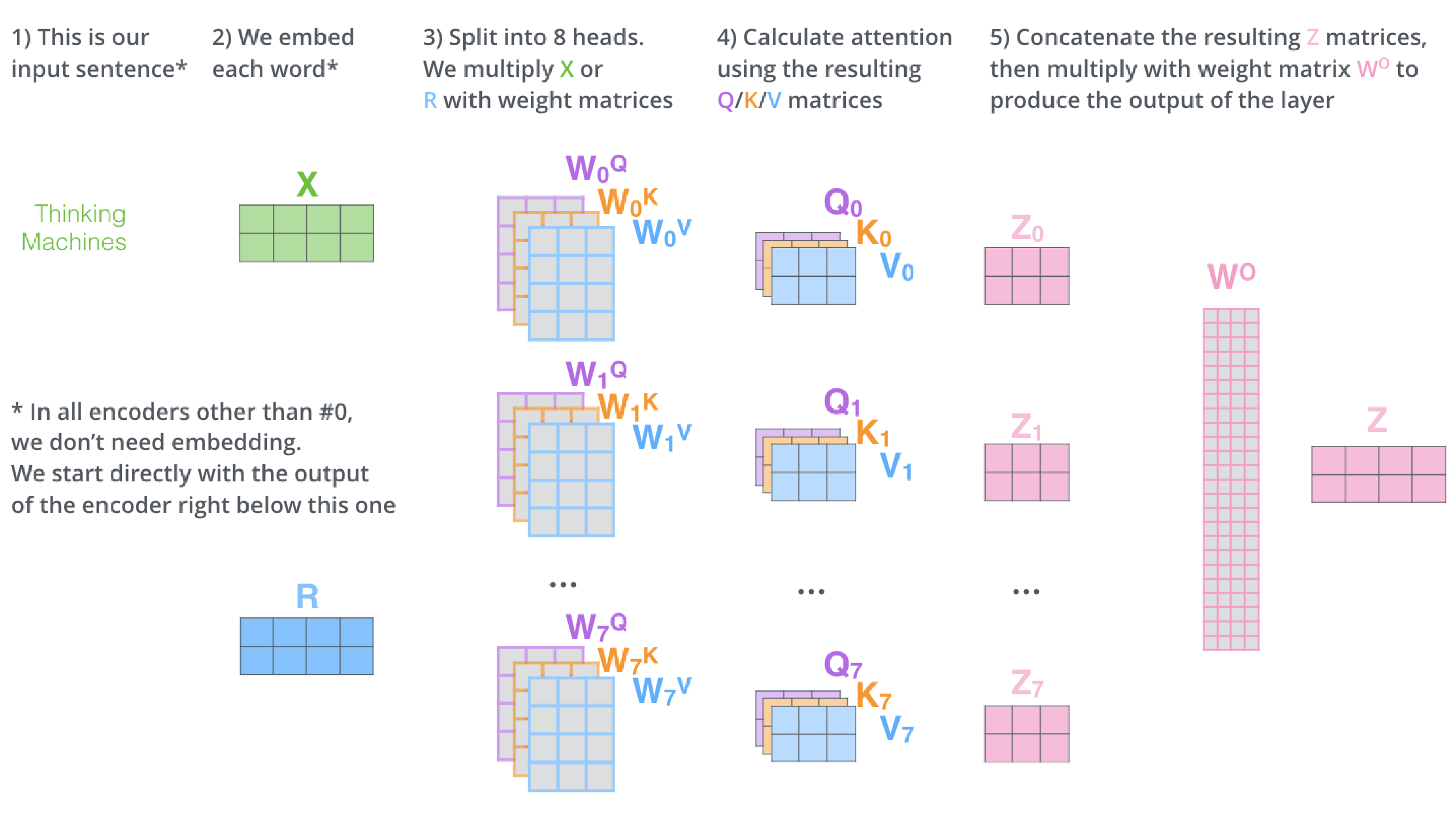
$$q^{i,1} = W^{q,1} q^{i}$$

$$q^{i,2} = W^{q,2} q^{i}$$

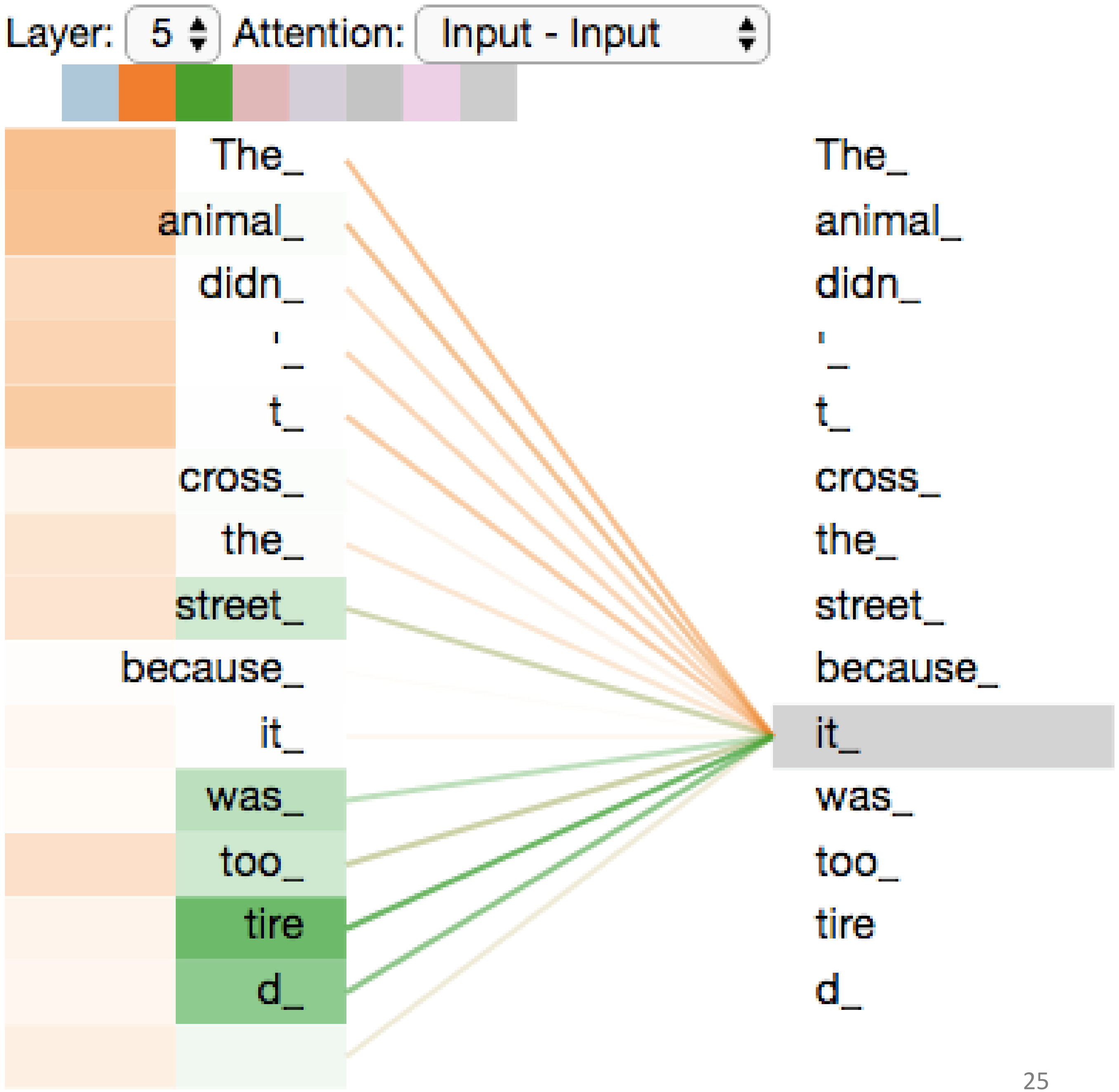


$$q^{i} = W^{q} a^{i}$$

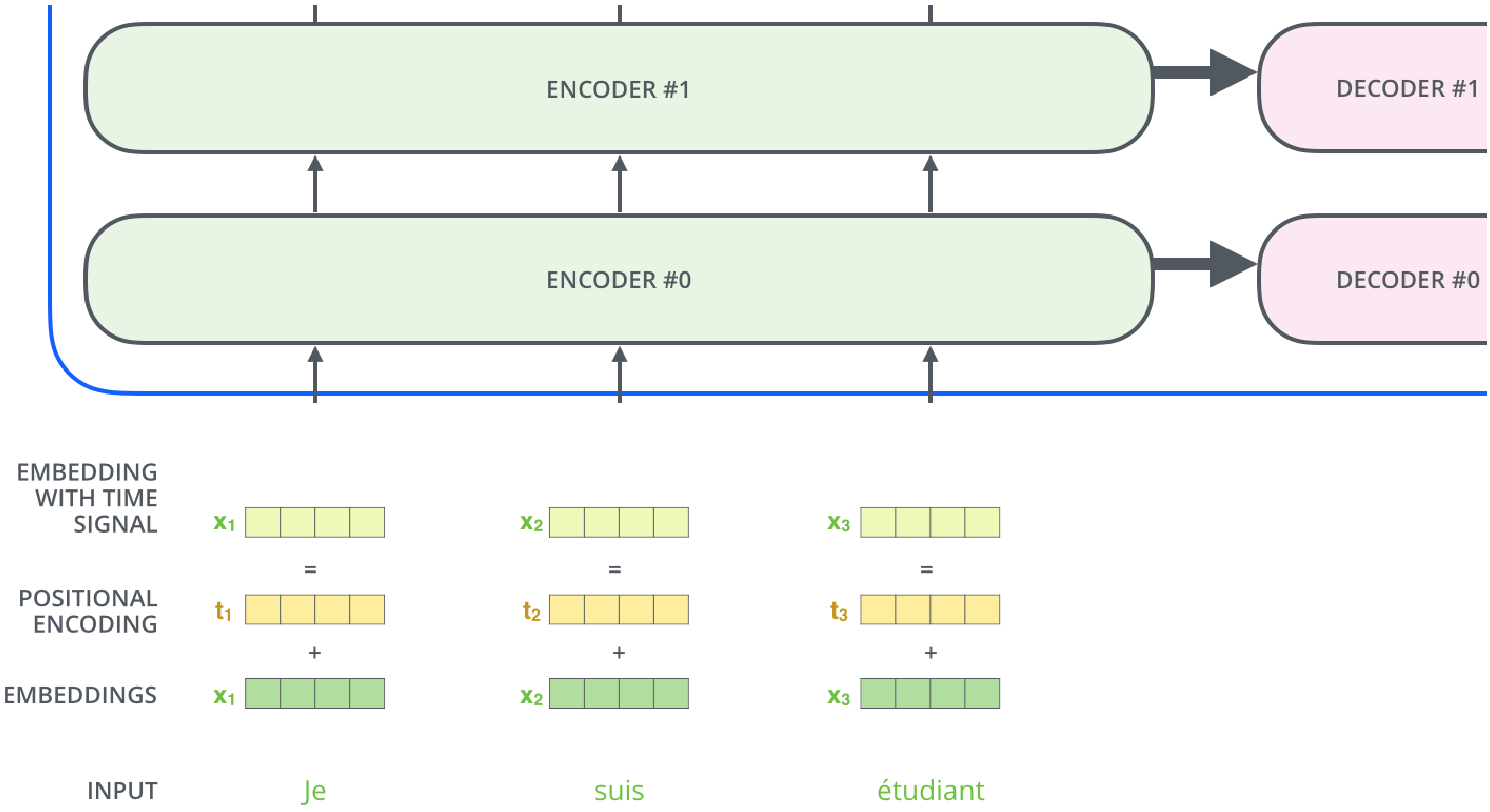
(2 heads as example)


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
W_0^Q
W_0^K
W_0^V
W_1^Q
W_1^K
W_1^V
W_7^Q
W_7^K
W_7^V
Q_0
K_0
V_0
Q_1
K_1
V_1
Q_7
K_7
V_7
Z_0
Z_1
Z_7
W^O
Z

Two-head Self-attention

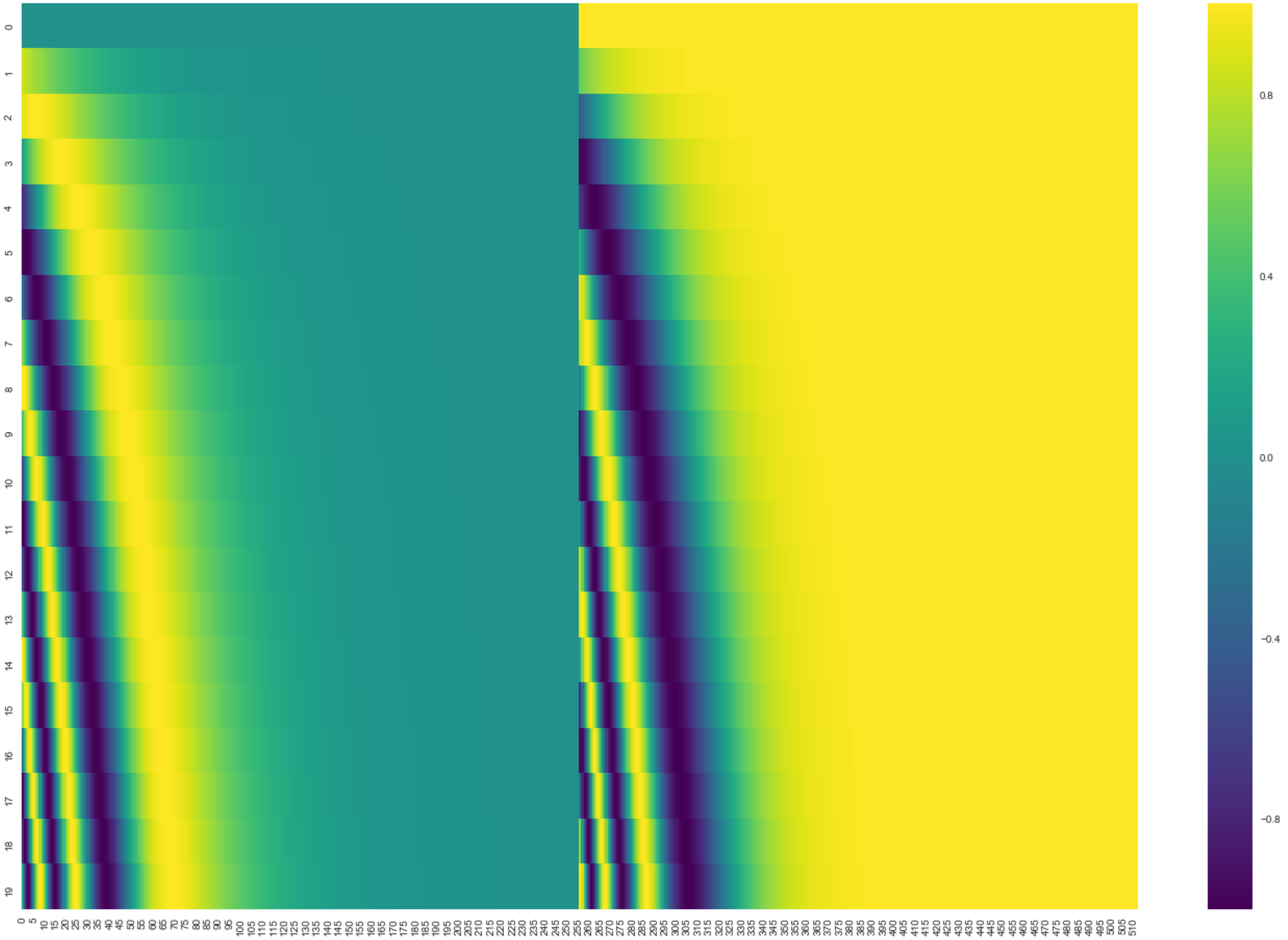


Positional Encoding

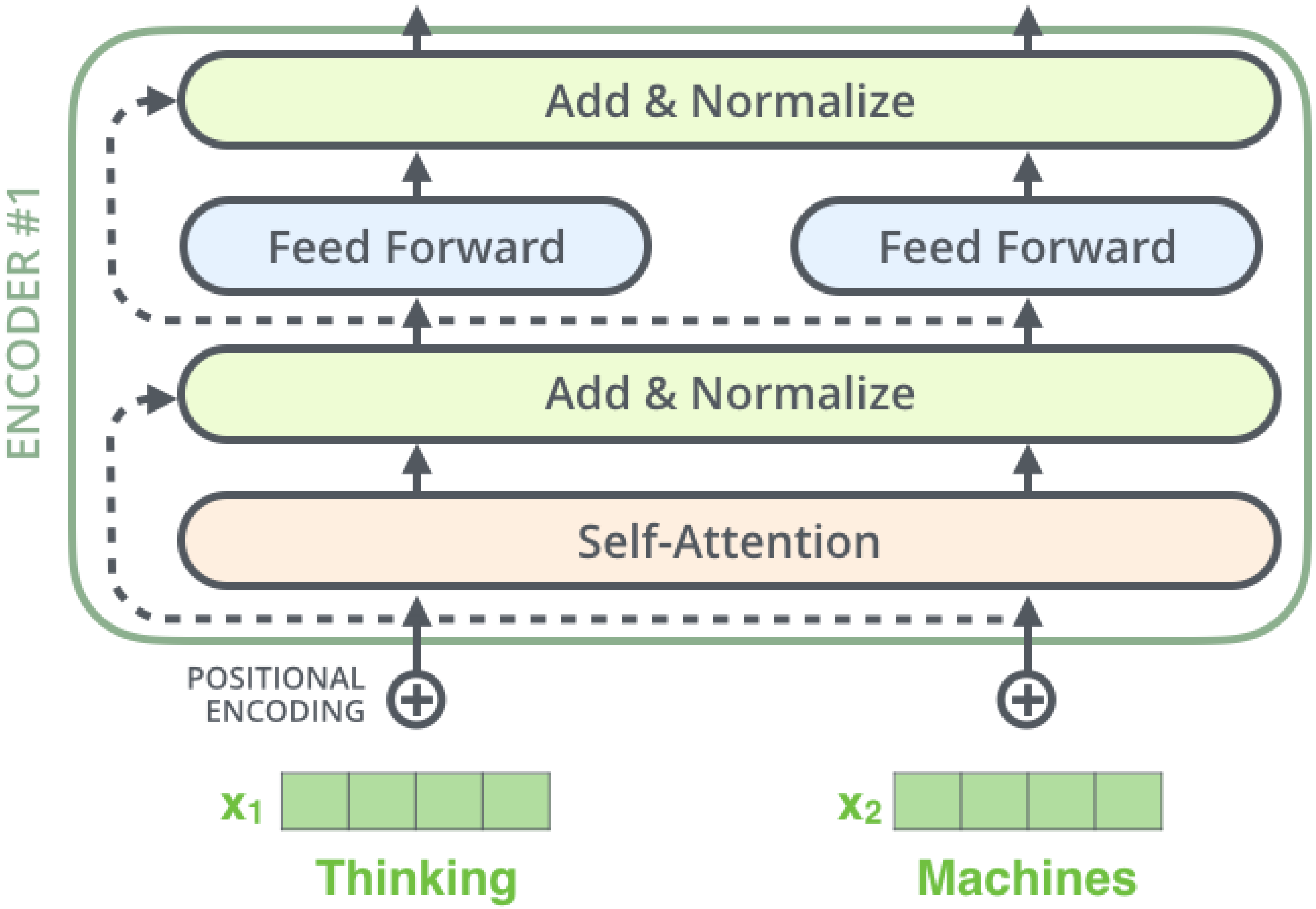


Positional Encoding Visualization

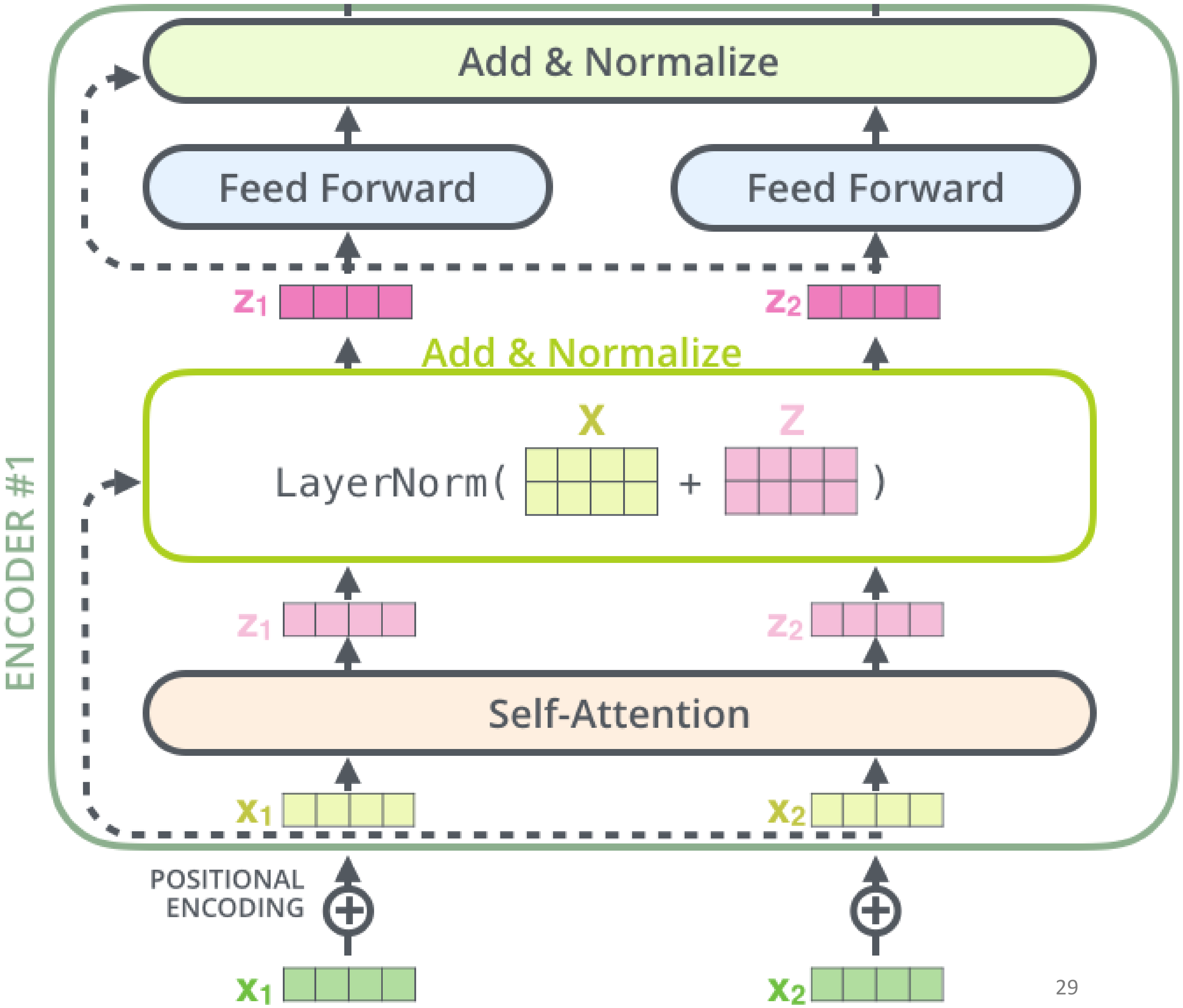
- Visualize positional encoding for 20 words (rows) with an embedding size of 512 (columns)

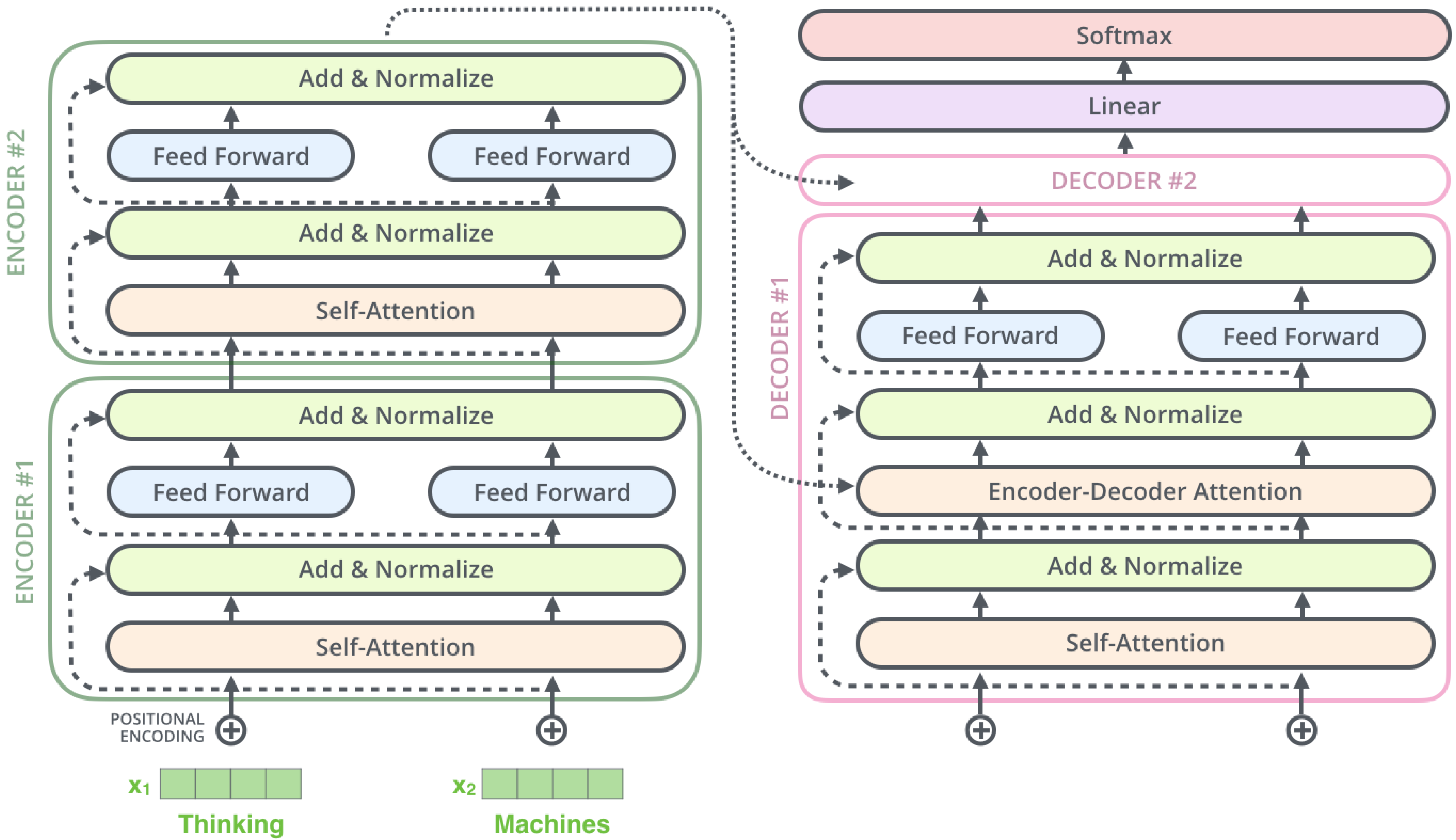


Adding Residual Connections



Layer Normalization



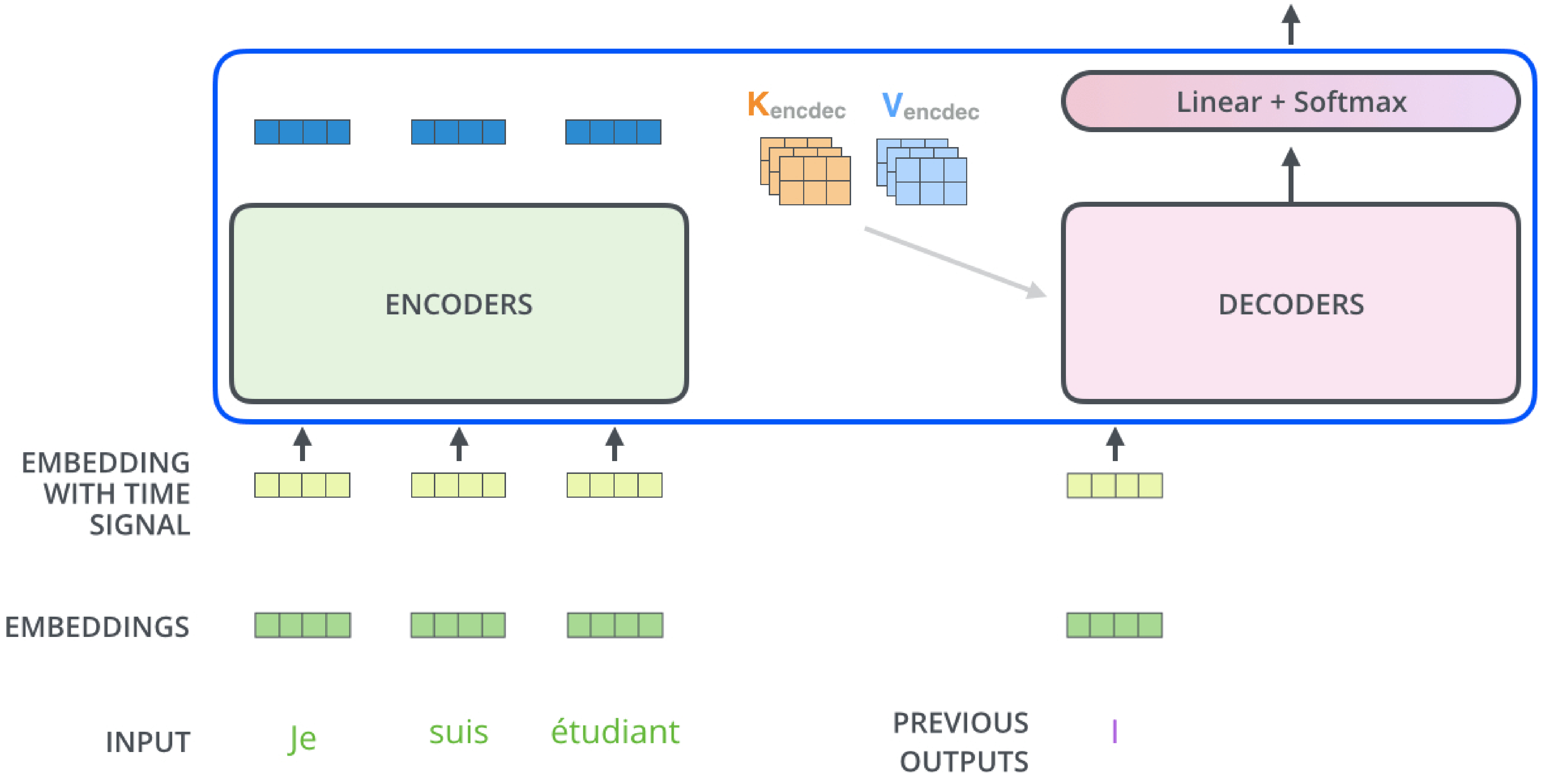


Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention
ENCODER #2
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize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

Decoder

Decoding time step: 1 2 3 4 5 6 OUTPUT I

Decoder



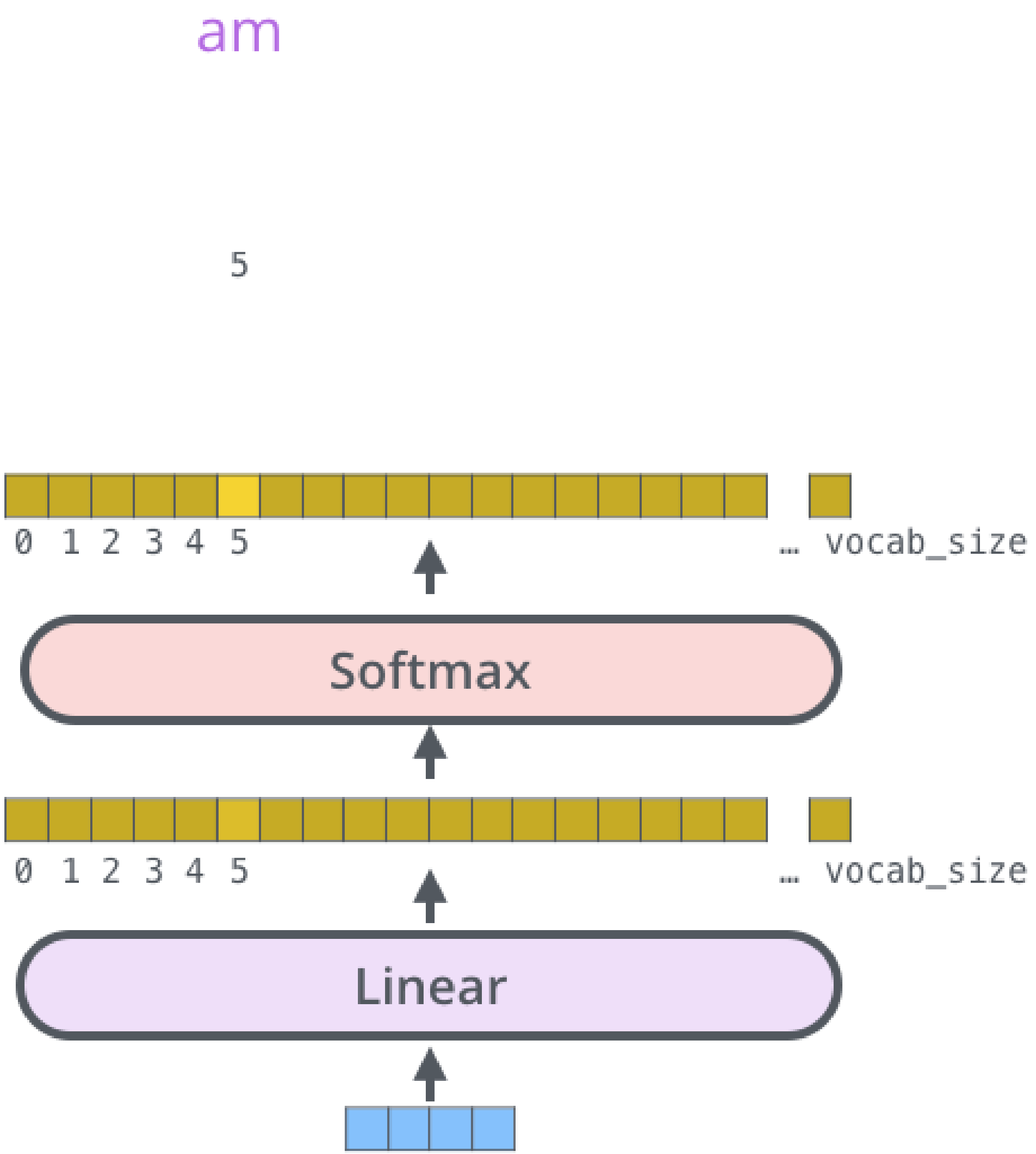
Which word in our vocabulary is associated with this index?

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

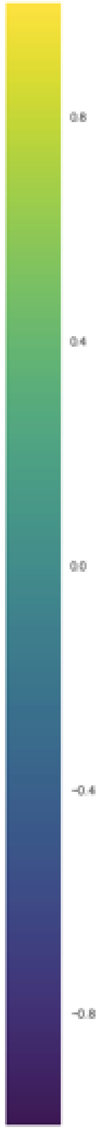
One-hot encoding of the word "am"

0.0	1.0	0.0	0.0	0.0	0.0

Target Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.0	0.0	1.0	0.0	0.0	0.0
position #2	0.0	1.0	0.0	0.0	0.0	0.0
position #3	1.0	0.0	0.0	0.0	0.0	0.0
position #4	0.0	0.0	0.0	0.0	1.0	0.0
position #5	0.0	0.0	0.0	0.0	0.0	1.0

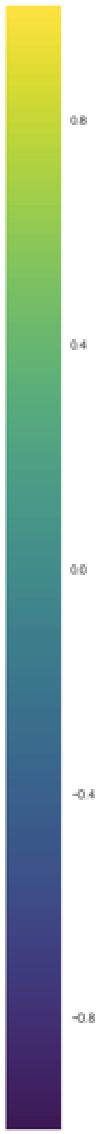
a am I thanks student <eos>



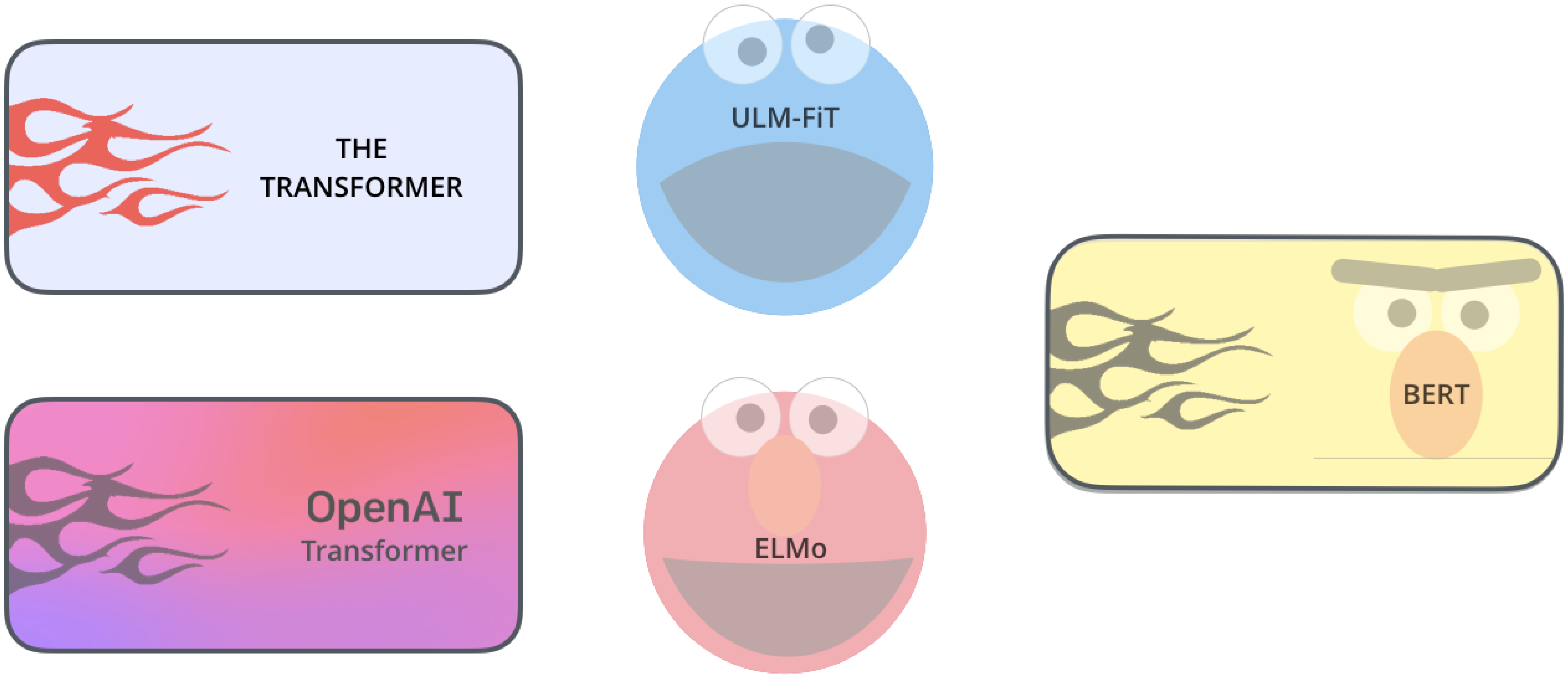
Trained Model Outputs

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>



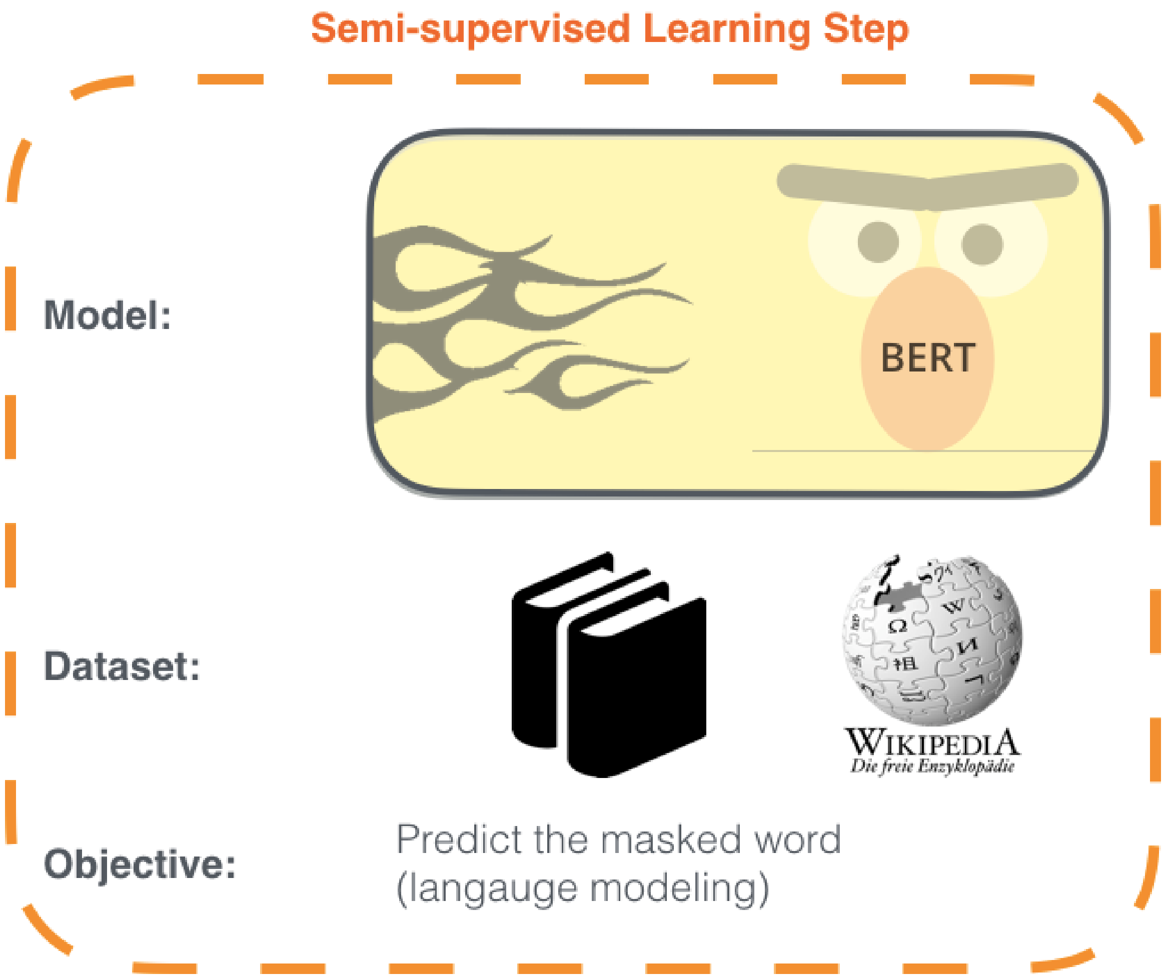
BERT, ELMo, GPT



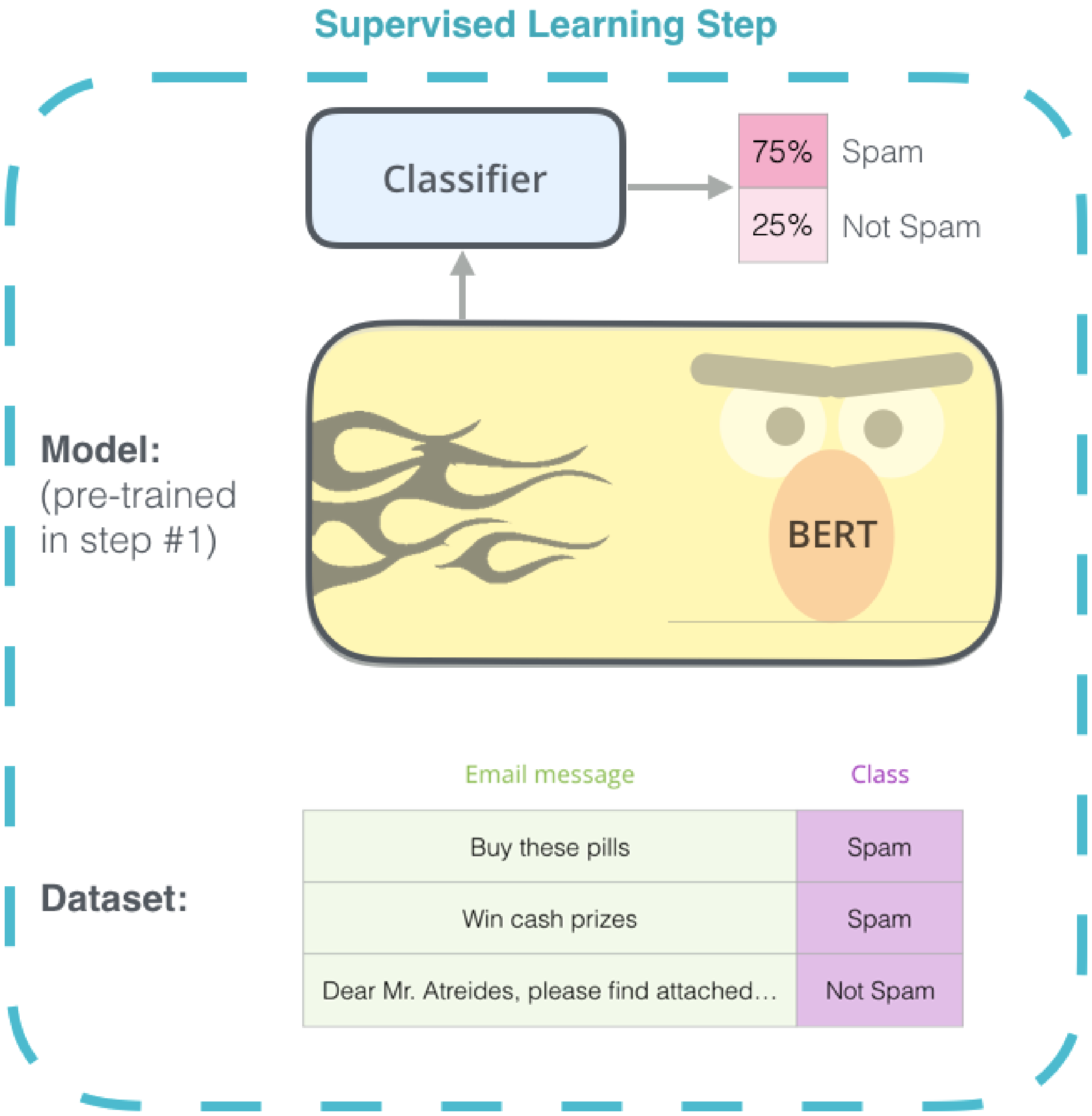
http://jalammar.github.io/illustrated-bert/

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

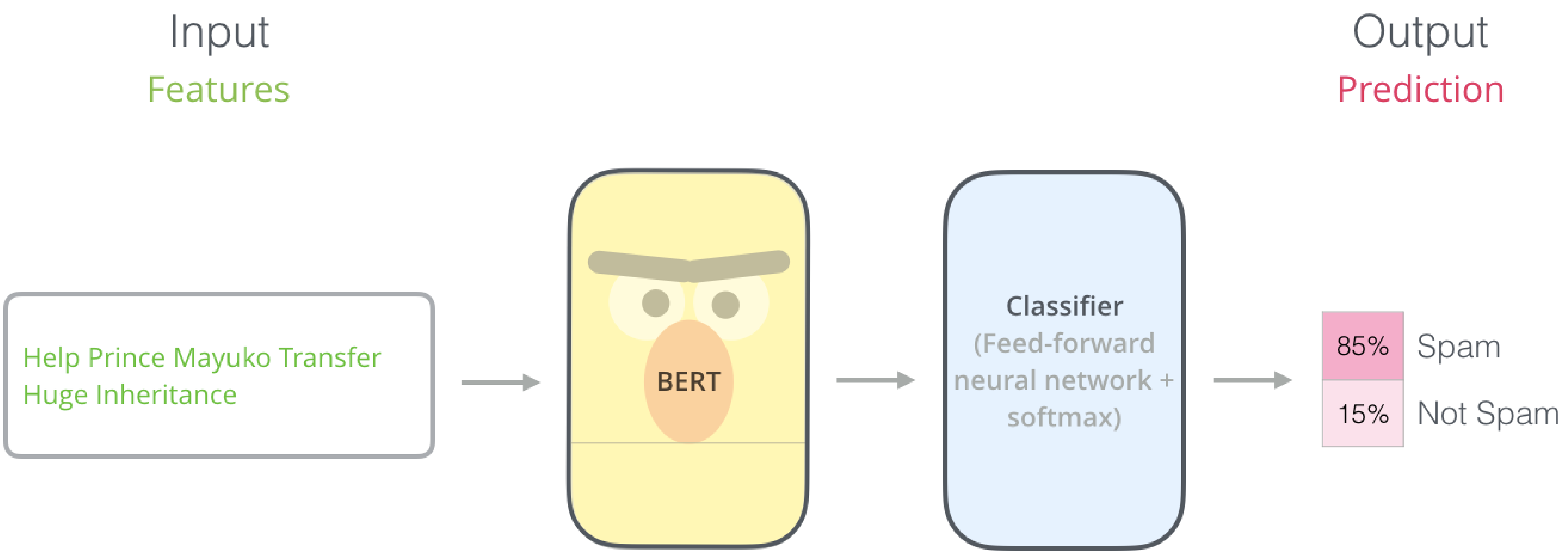
The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.



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

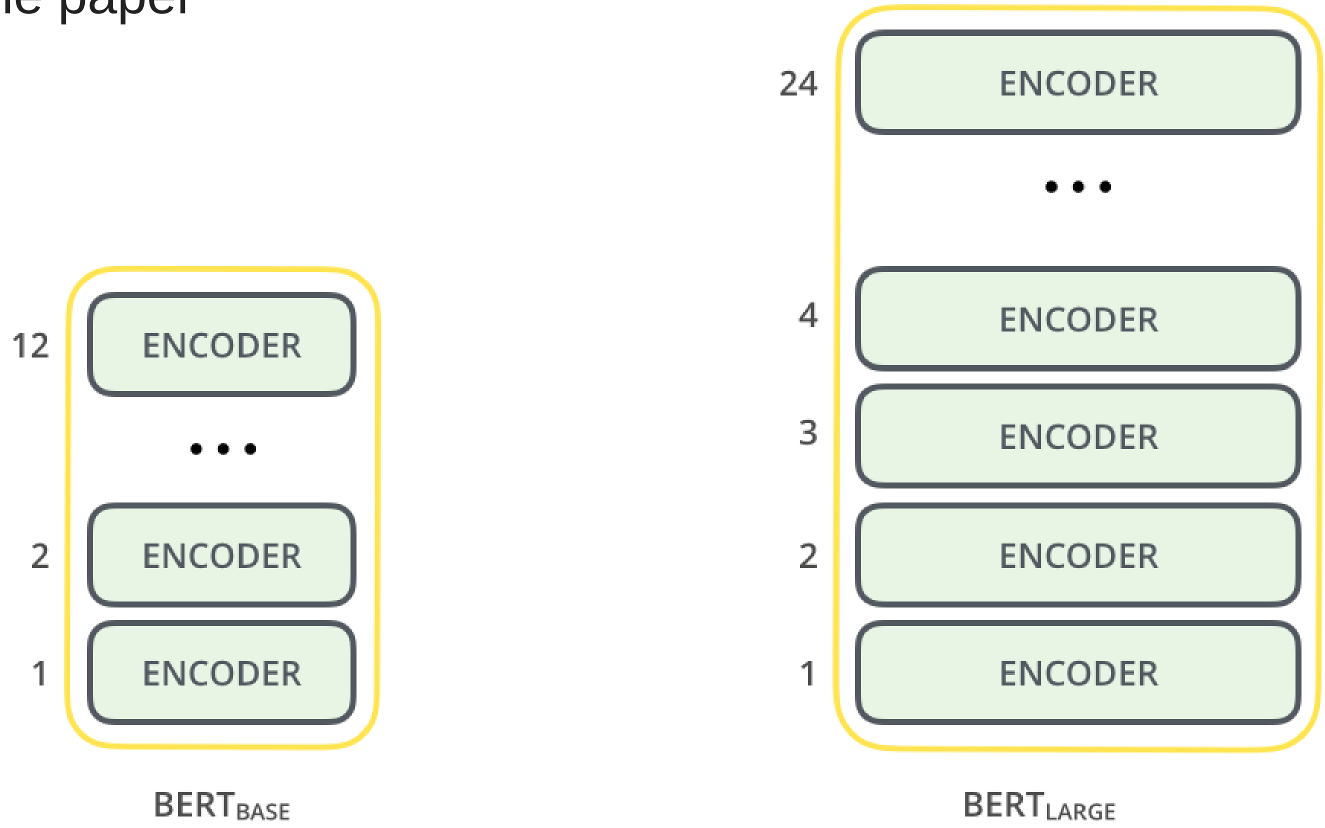


Sentence Classification

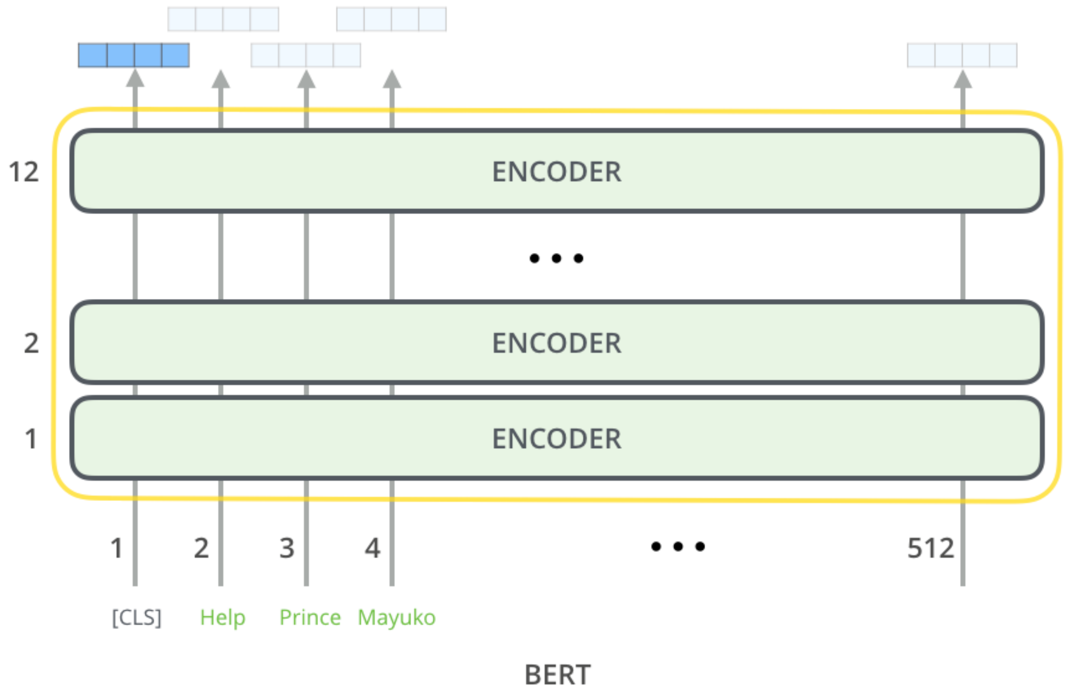


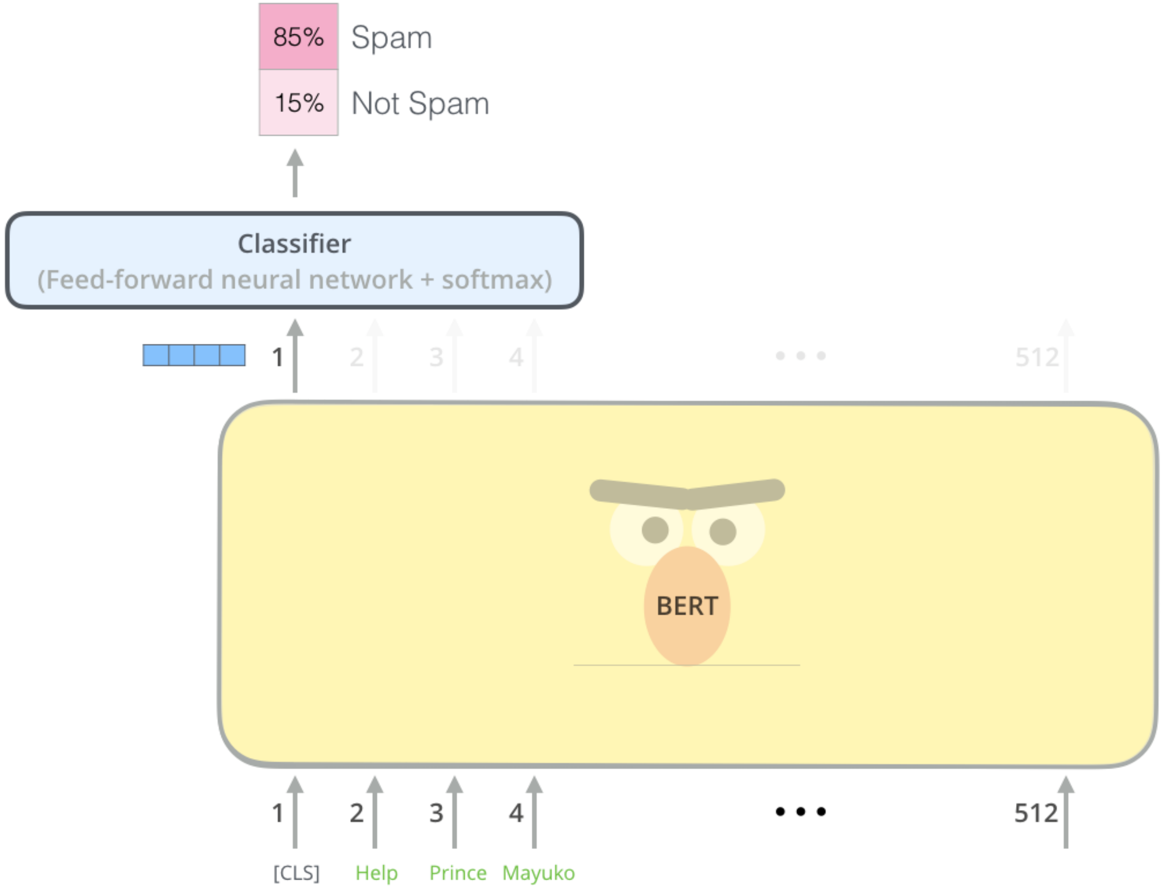
Model Architecture

- BERT BASE – Comparable in size to the OpenAI Transformer in order to compare performance
- BERT LARGE – A ridiculously huge model which achieved the state of the art results reported in the paper



Use Pre-trained Encoders

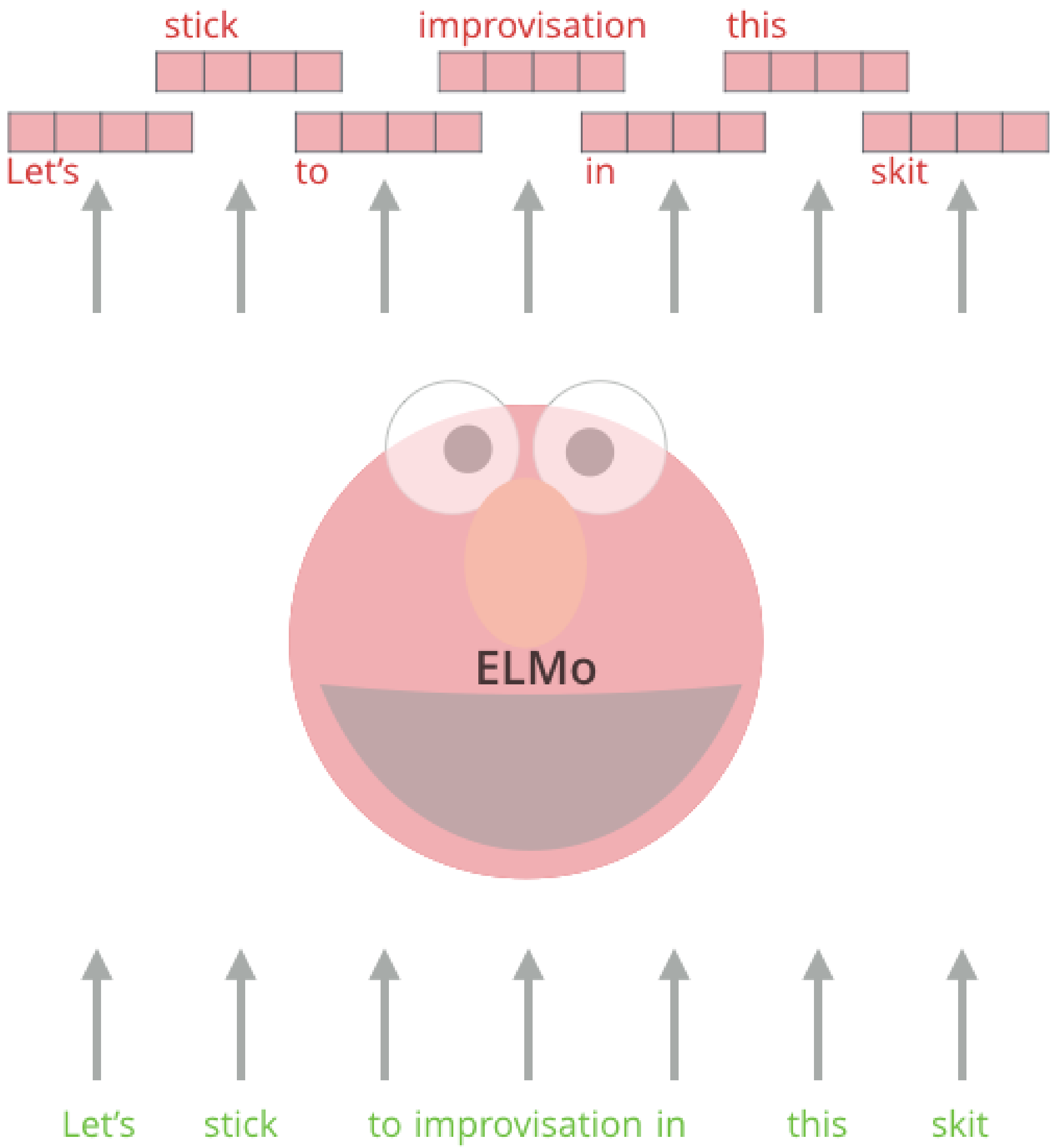


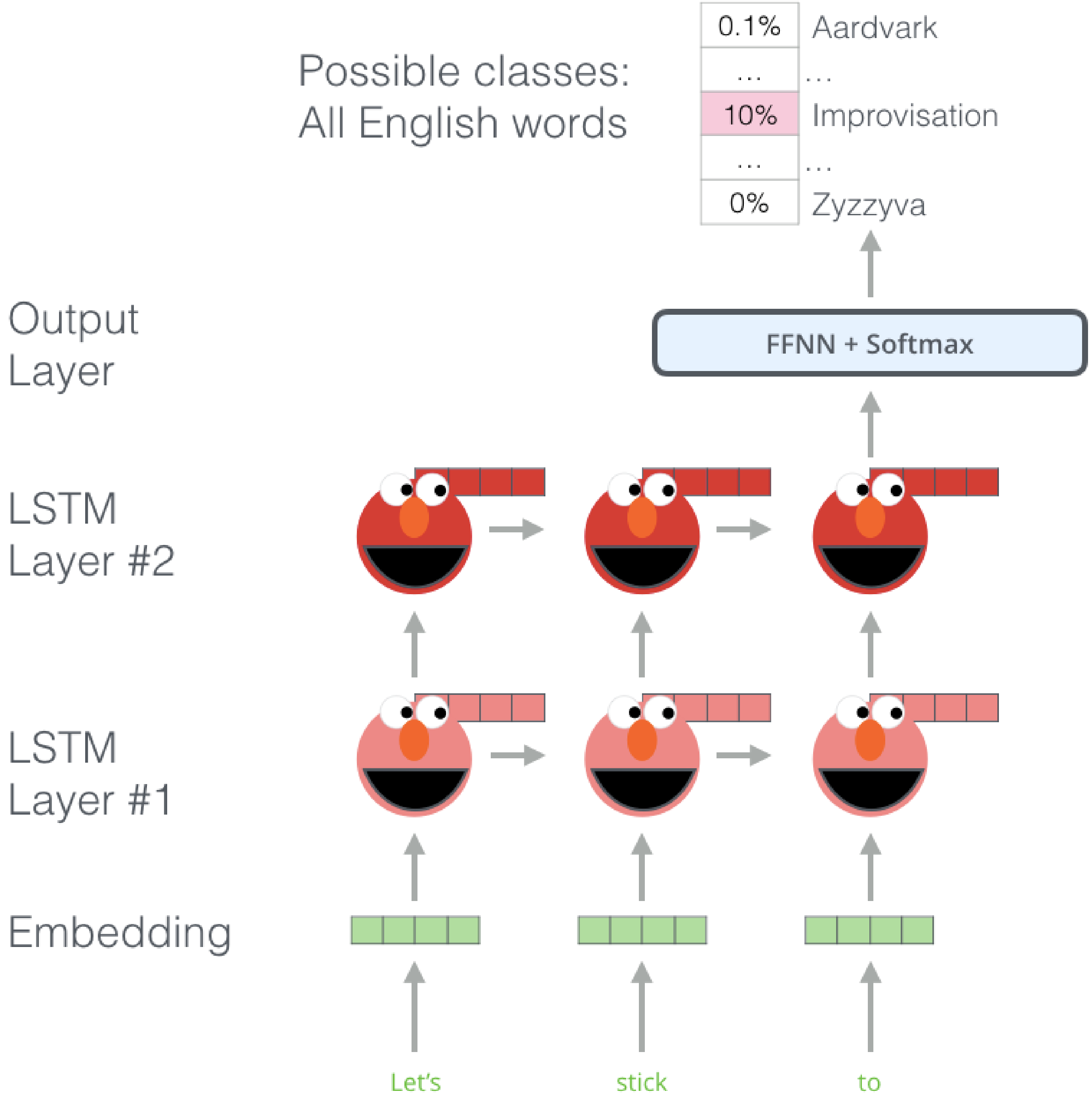


ELMo: Context Matters

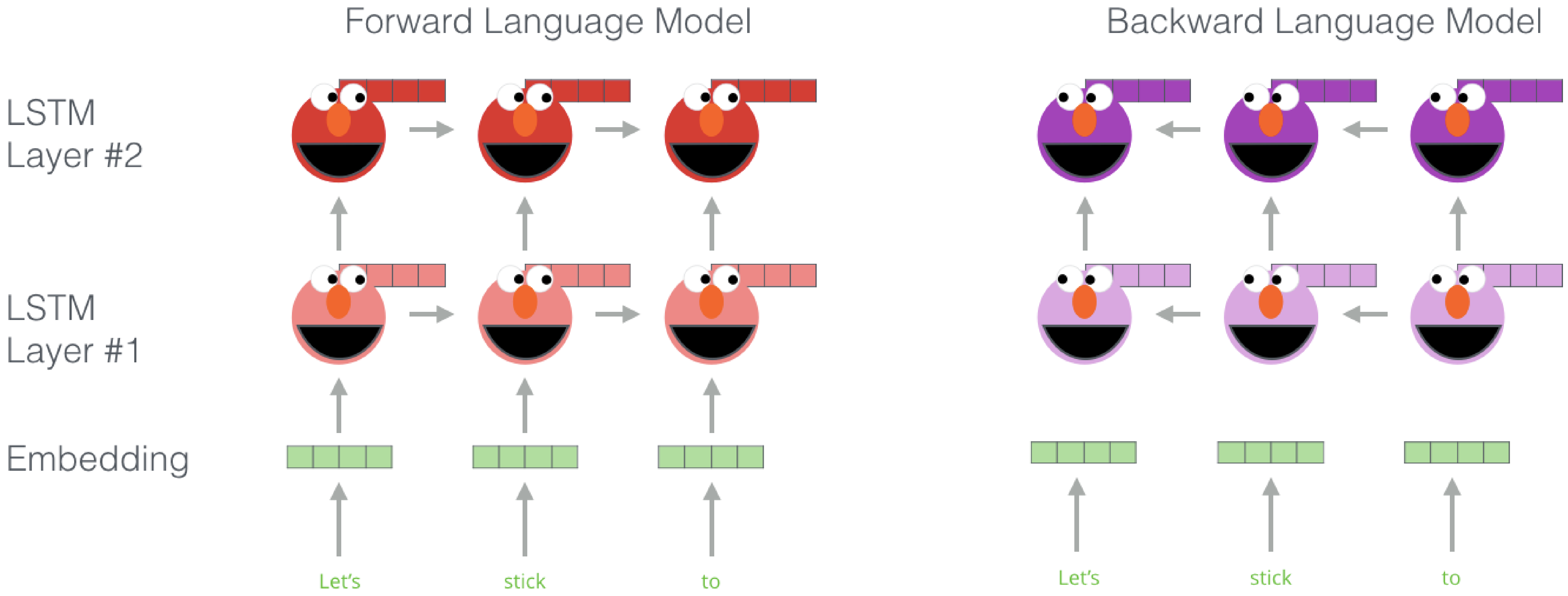
ELMo Embeddings

Words to embed



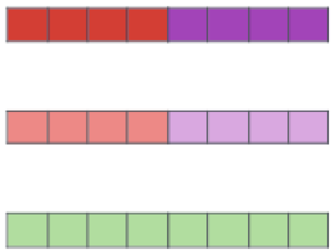

Possible classes:
All English words
0.1% Aardvark
... ...
10% Improvisation
... ...
0% Zyzzyva
Output
Layer
FFNN + Softmax
LSTM
Layer #2
LSTM
Layer #1
Embedding
Let's
stick
to

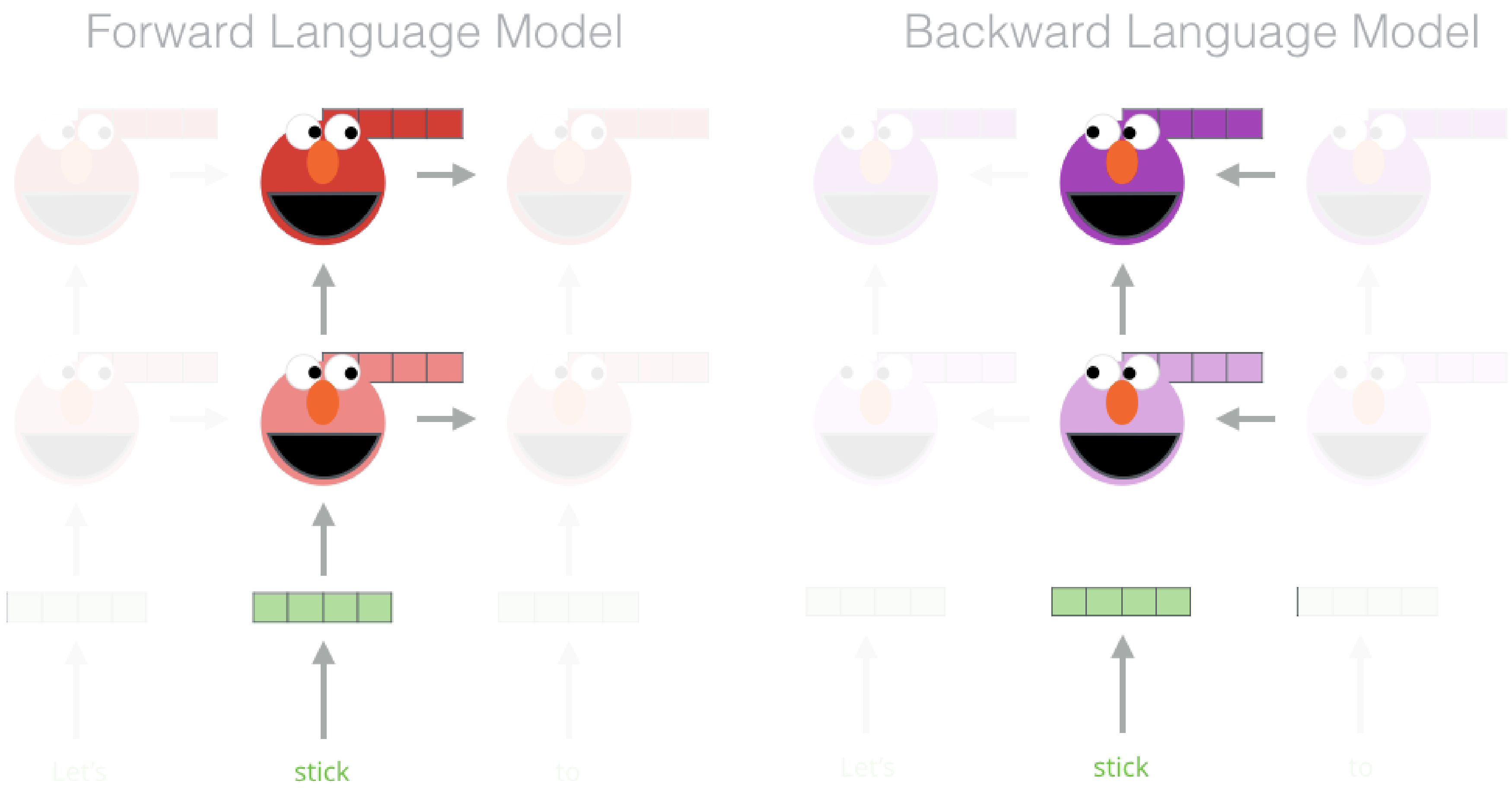
Embedding of “stick” in “Let’s stick to” - Step #1



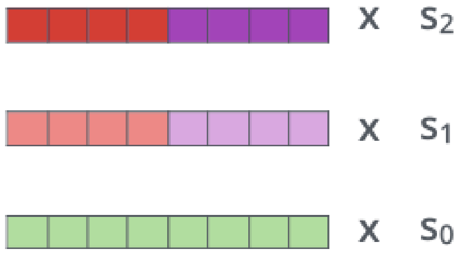
Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers





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

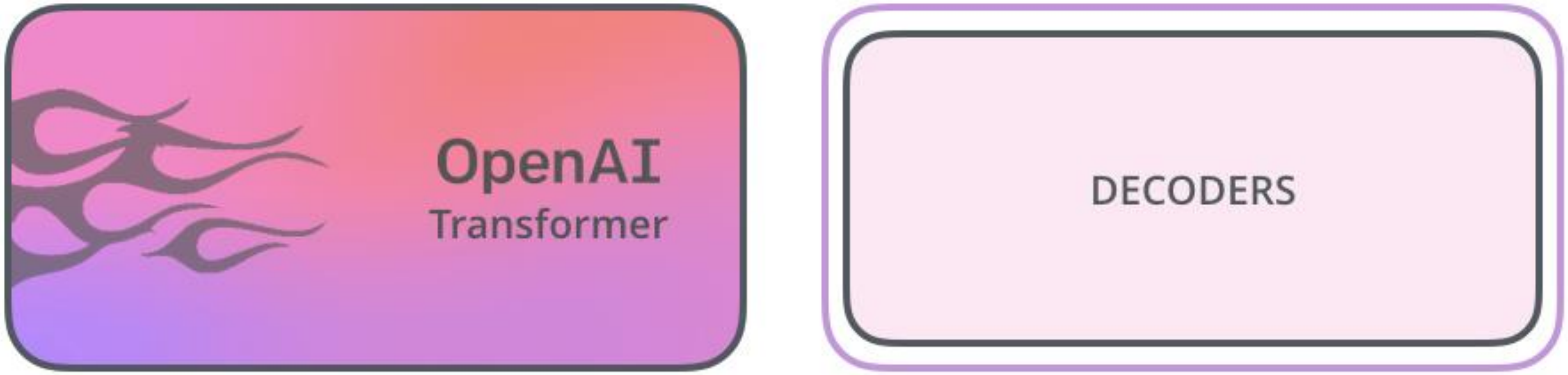


3- Sum the (now weighted) vectors

ELMo embedding of “stick” for this task in this context

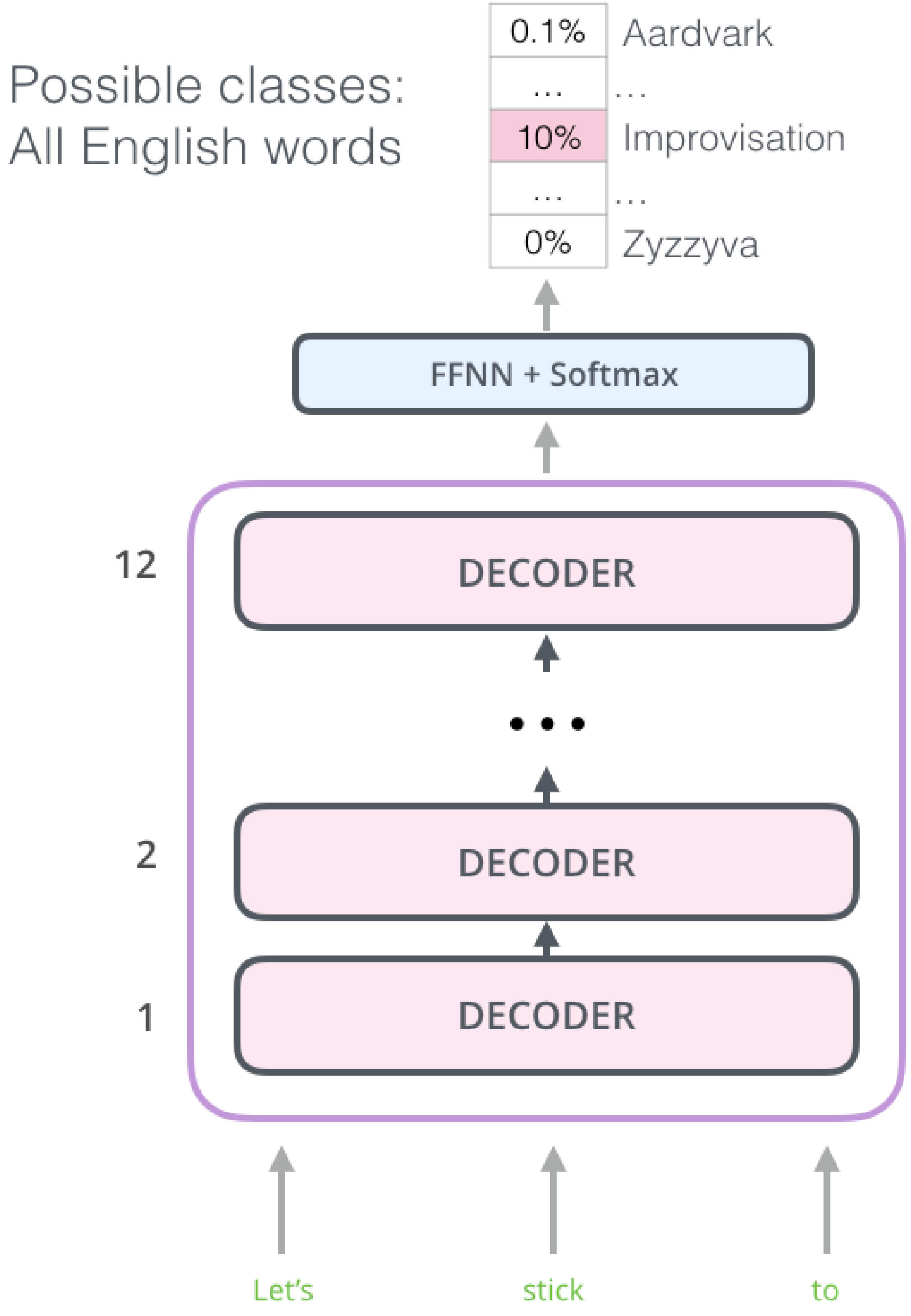
OpenAI Transformer

- Pre-training a Transformer Decoder for Language Modeling

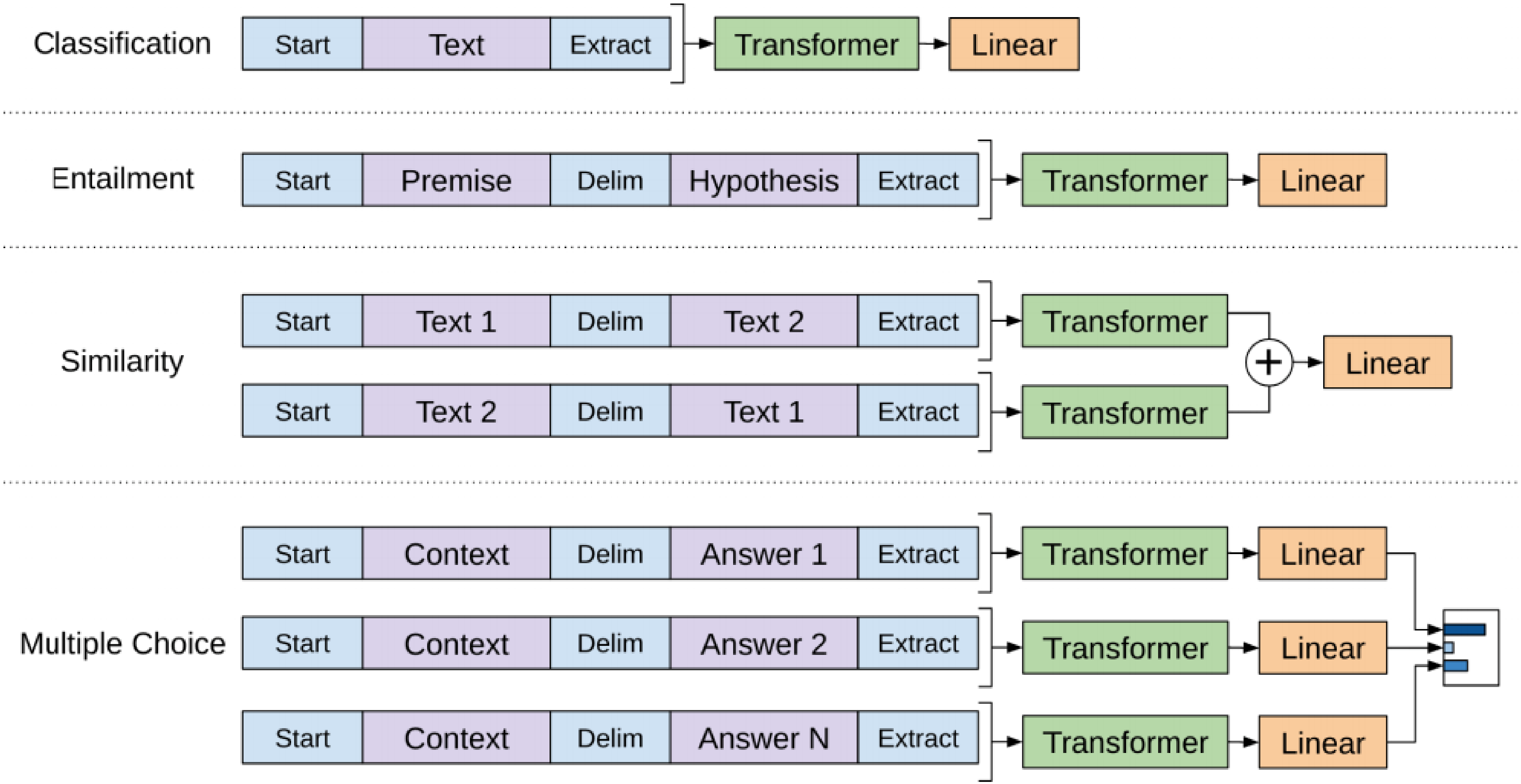


Re-train Decoder

- No encoder-decoder attention sublayer

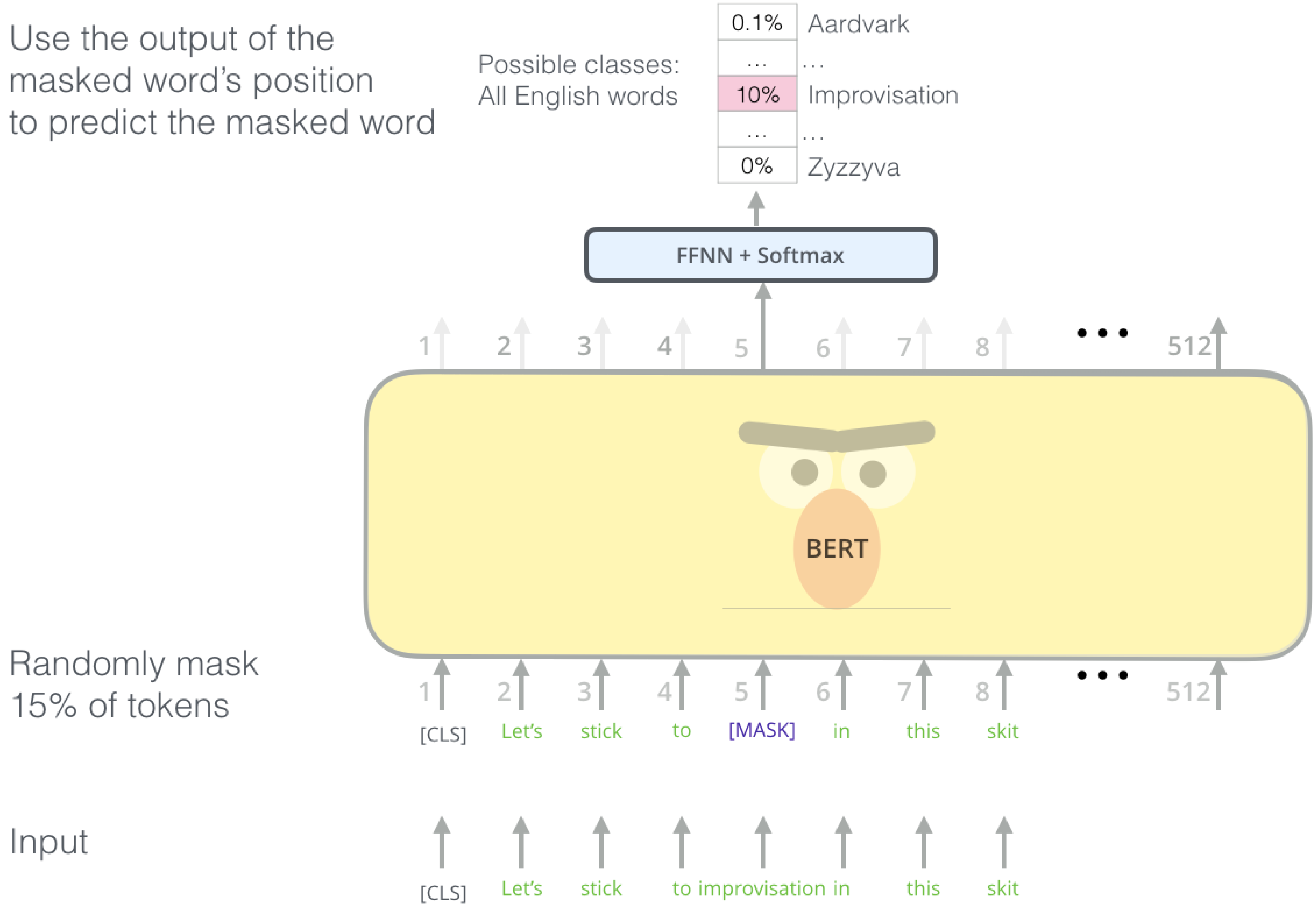


OpenAI Transformer for Different Tasks

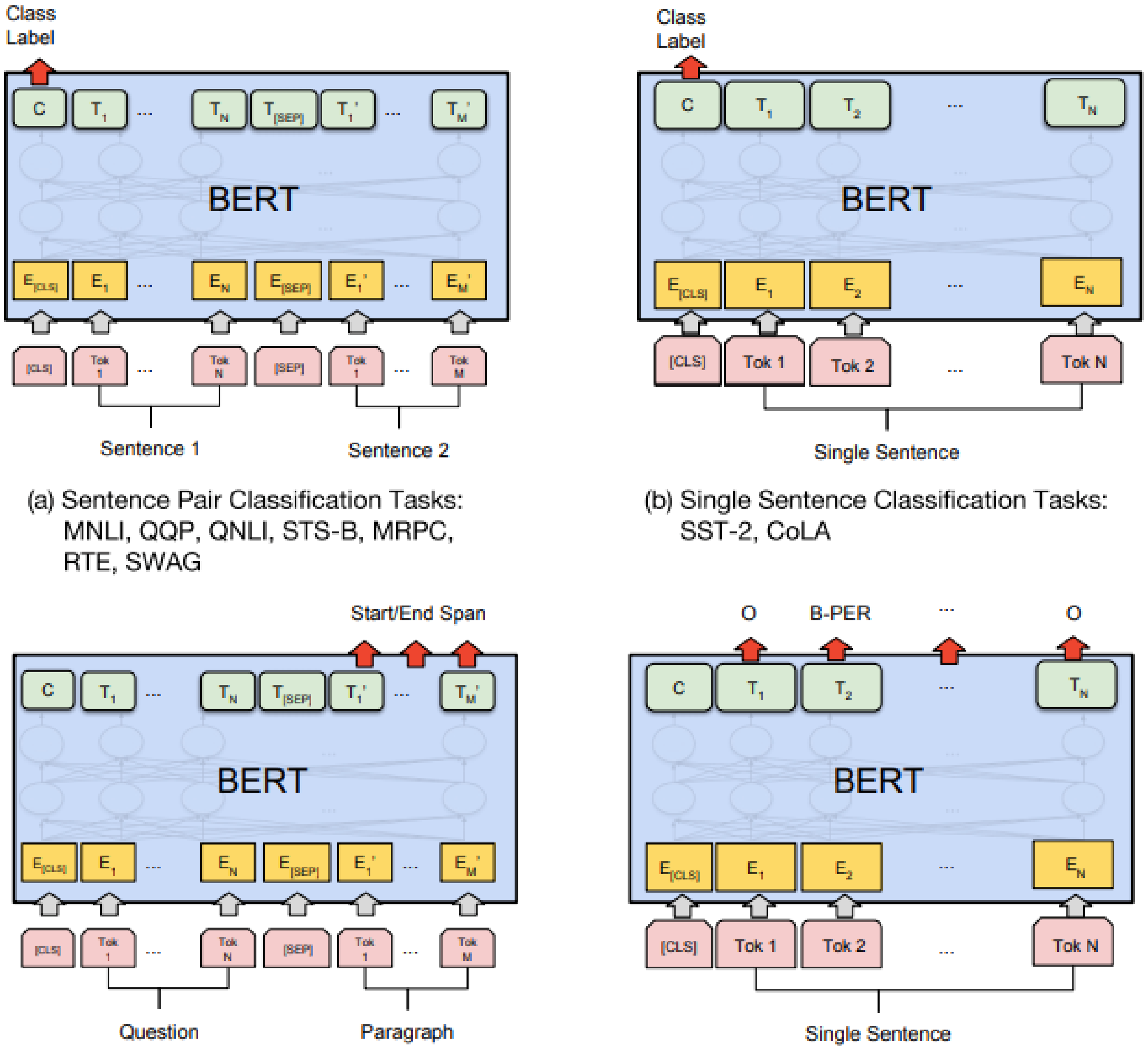


BERT: From Decoders to Encoders

- ELMo's language model was bi-directional, but the openAI transformer only trains a forward language model.
- Could we build a transformer-based model whose language model looks both forward and backwards, i.e. "is conditioned on both left and right context"?
  - **Masked Language Model**


Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
... ...
10% Improvisation
... ...
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
Randomly mask 15% of tokens
1 2 3 4 5 6 7 8 ... 512
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

BERT for Different tasks



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

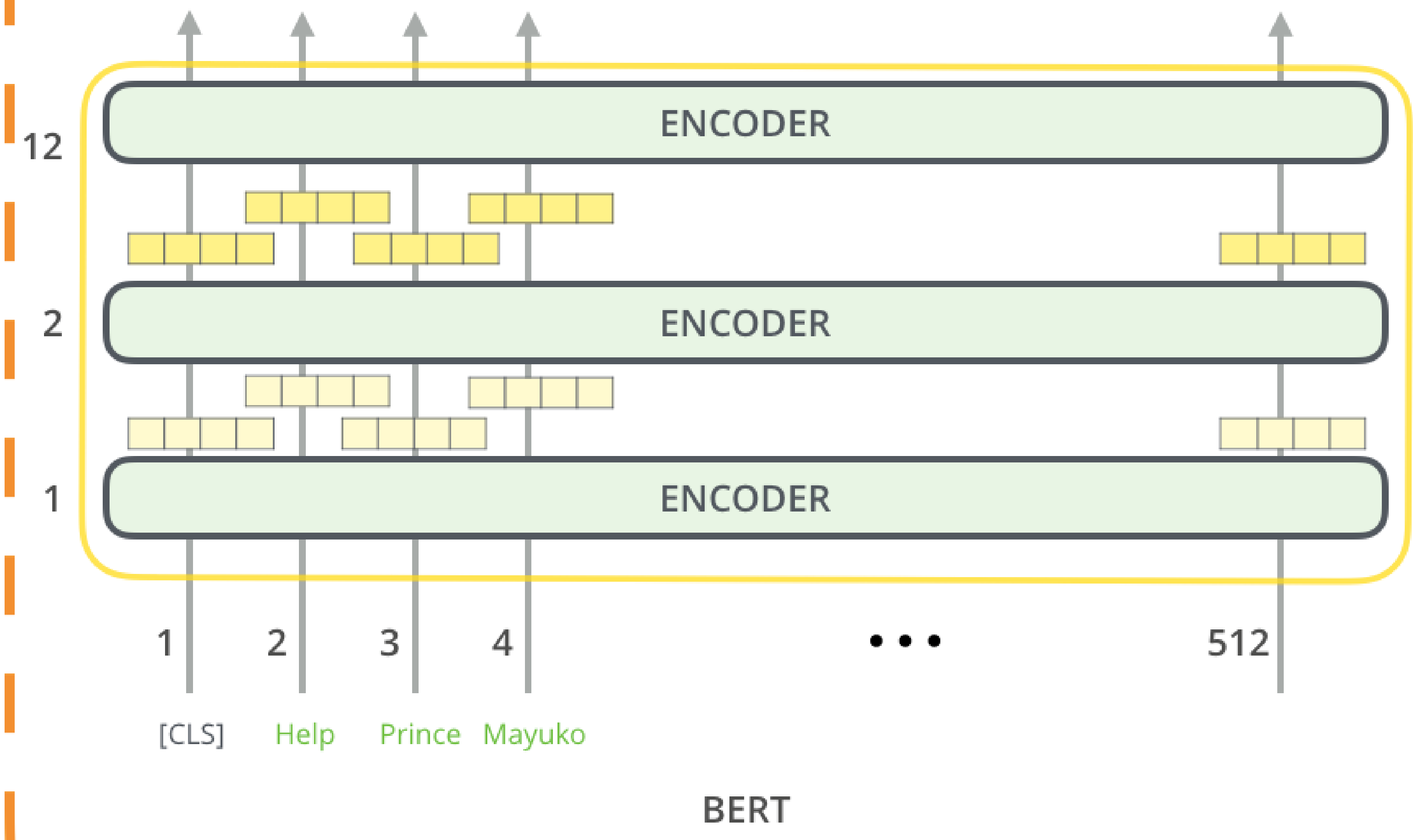
(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

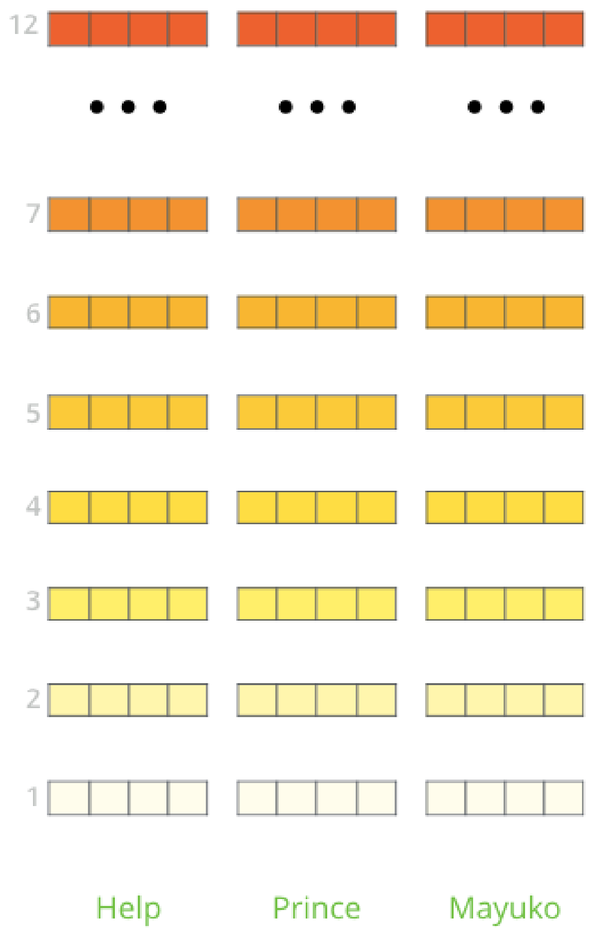
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Generate Contexualized Embeddings

BERT for Feature Extraction



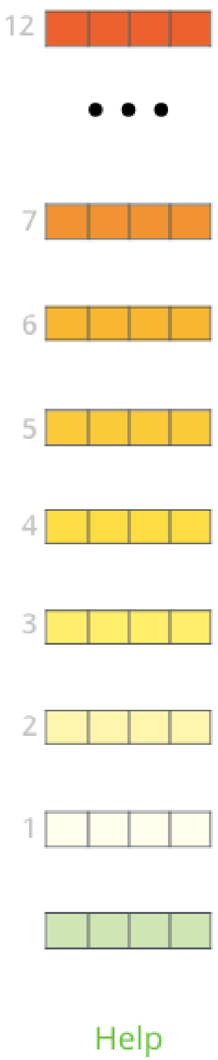
The output of each encoder layer along each token's path can be used as a feature representing that token.



But which one should we use?

What is the best contextualized embedding for “Help” in that context?

For named-entity recognition task CoNLL-2003 NER



	Dev F1 Score
First Layer (Embedding)	91.0
Last Hidden Layer (12)	94.9
Sum All 12 Layers (12 + … + 2 + 1)	95.5
Second-to-Last Hidden Layer (11)	95.6
Sum Last Four Hidden (12 + 11 + 10 + 9)	95.9
Concat Last Four Hidden (9, 10, 11, 12)	96.1

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