Natural Language Processing

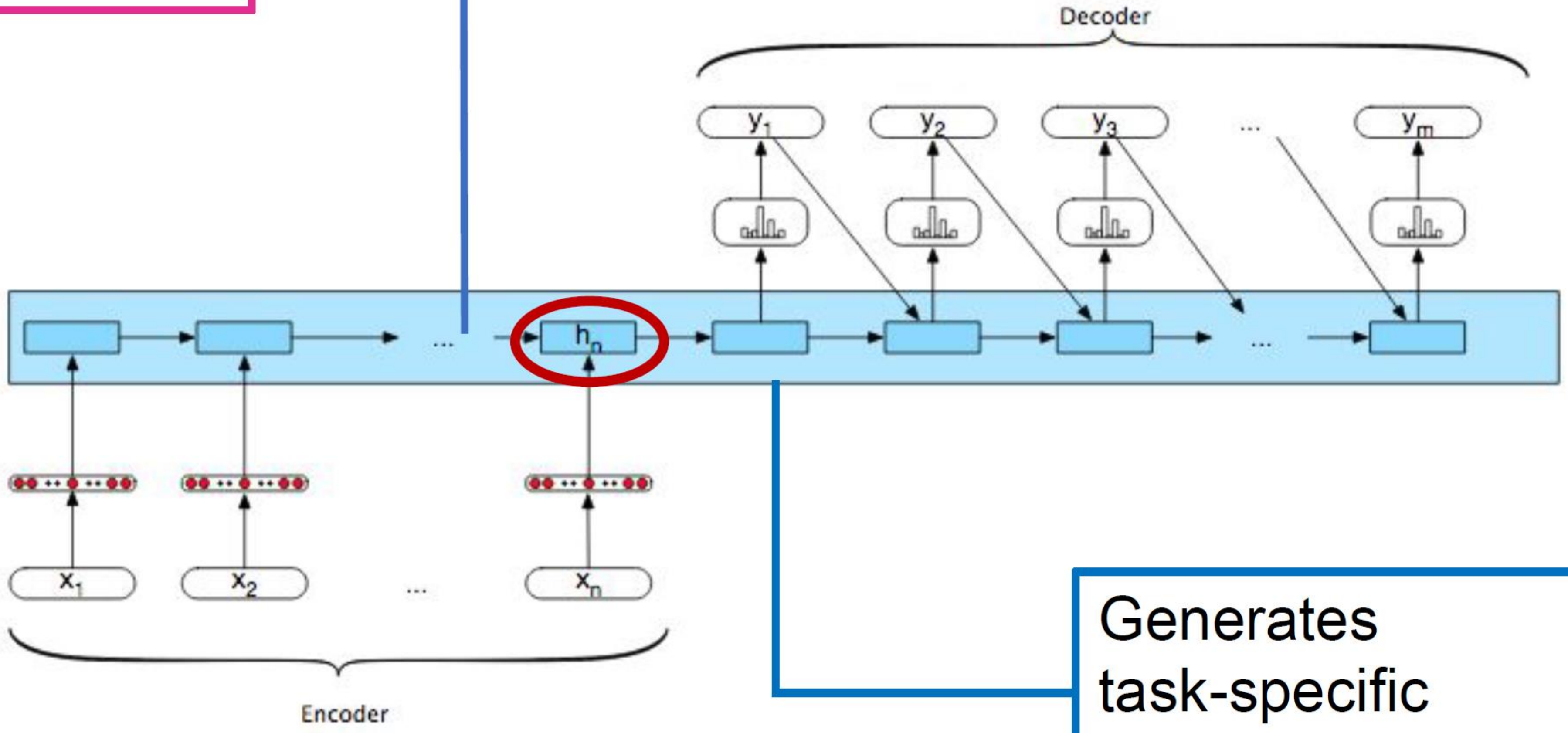
AIGC 5501

Transformers / BERT

Instructor: Ritwick Dutta
Email: ritwick.dutta@humber.ca

Encoder-decoder framework

Encodes a representation of the input

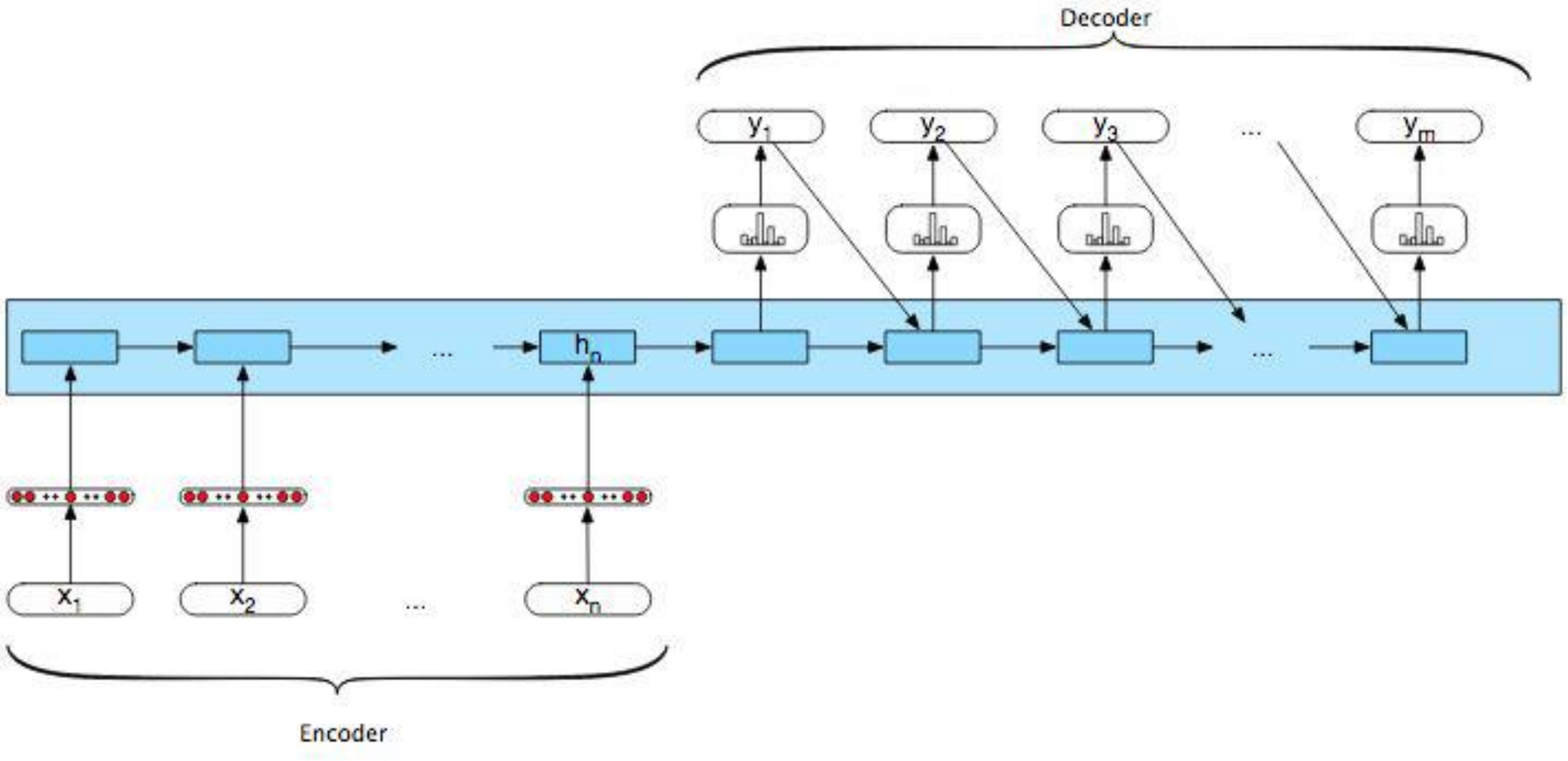


Generates task-specific output

Problems with encoder-decoder

- The **effect of the context** will decrease as decoder time steps increase.

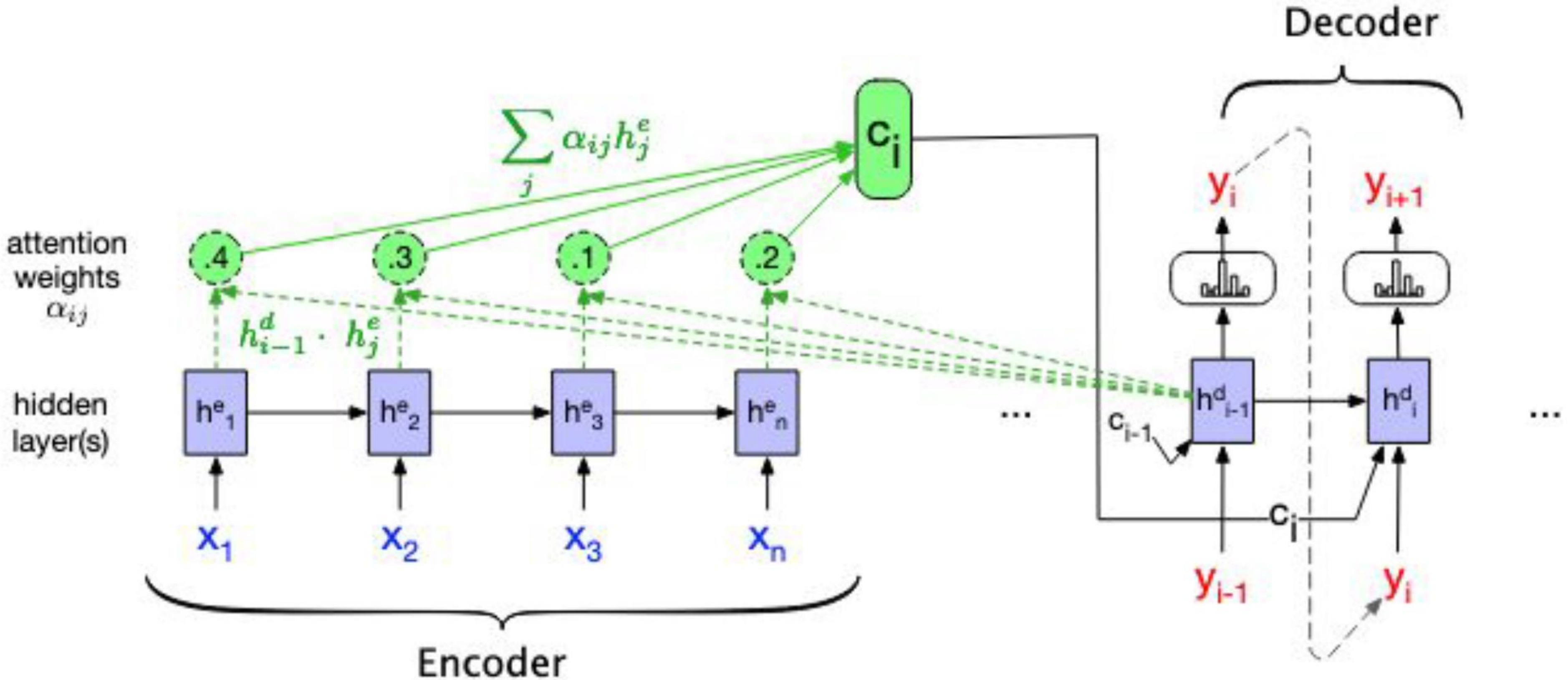
- **Loses useful information about the individual encoder states** that might prove useful in decoding (because only the last hidden state of the encoder is fed to the decoder).

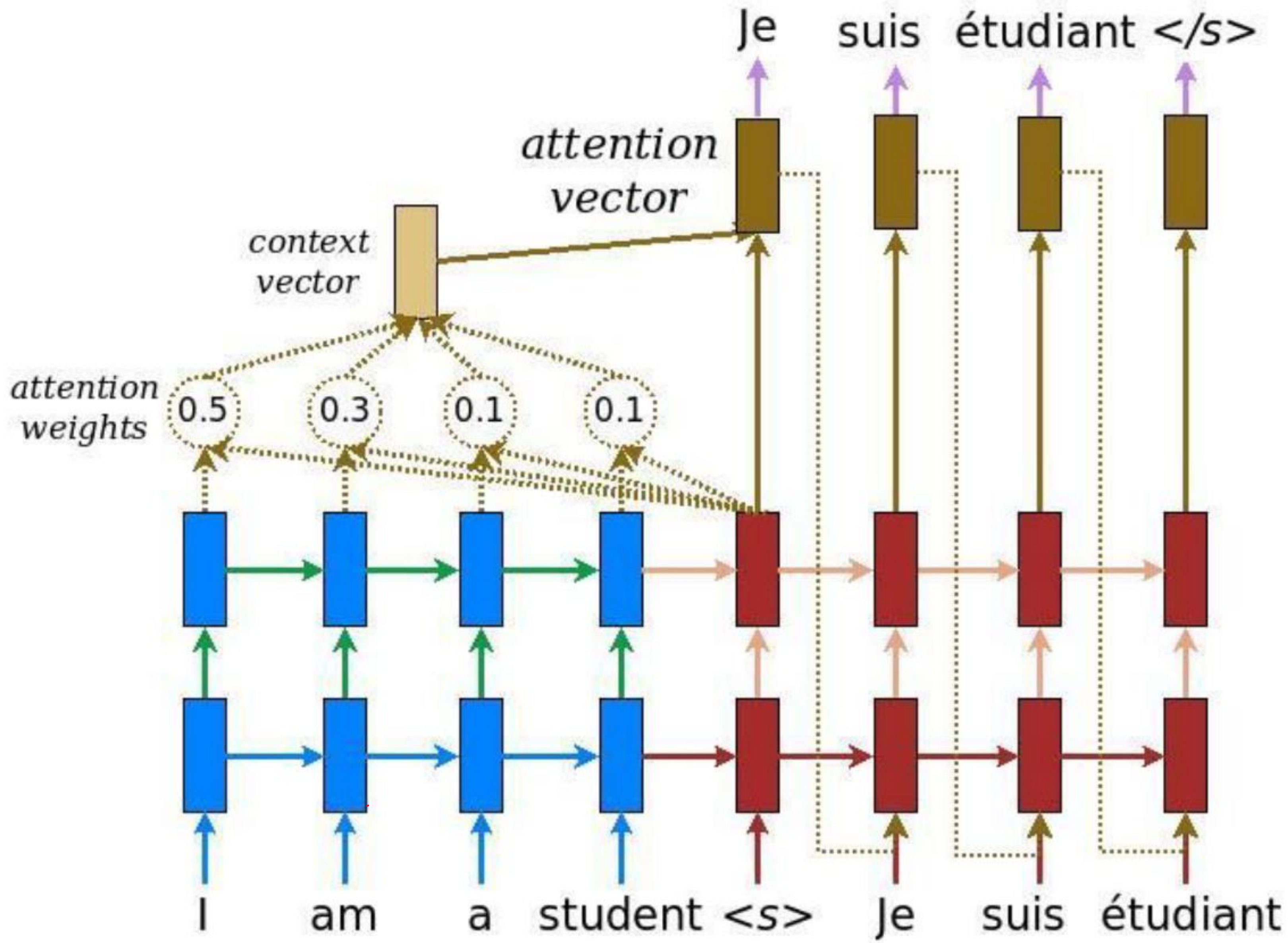


One approach to address this

- **Dynamically derive *c*** from the encoder hidden states at each point during decoding, refer to each as c_i
- Take **all** of the encoder hidden states into account
- Condition the computation of the current decoder state on c_i (and prior hidden state and previous output)

Attention





Je
suis
étudiant
</s>
attention vector
context vector
attention weights
0.5
0.3
0.1
0.1
I
am
a
student
<s>
Je
suis
étudiant

Attention and Importance

At its core, attention is taking a weighted average of encoder hidden states.

We are "paying attention" to states that receive large attention weights.

In general, we are weighting input words/corresponding encoder states proportional to their **"importance".**

Critical for getting good performance from RNNs on many NLP tasks.

RNNs + Attention: Can we do any better?

RNNs are inherently sequential. Sequential computation cannot be easily parallelized.

Attention on the other hand can be readily parallelized. And it is another mechanism for incorporating "history"/context, arguably a better one than recurrence.

Transformers

Introduced in **Attention Is All You Need** (Vaswani et al. *NeurIPS* 2017)

A purely attention-based architecture (highly parallelizable), i.e. **no recurrence**

Very deep model for NLP (12 layers)

Originally envisioned for seq2seq tasks (encoder is 6 layers, decoder is 6 layers)

The encoder and decoder are the same “architecture” applied differently

Transformer: Input Process

some number that makes
sense for the architecture
of the machine

Input: A sequence of tokens w_1 to w_N. **N must be 512.**

If sequence is innately smaller than 512, add “padding” tokens

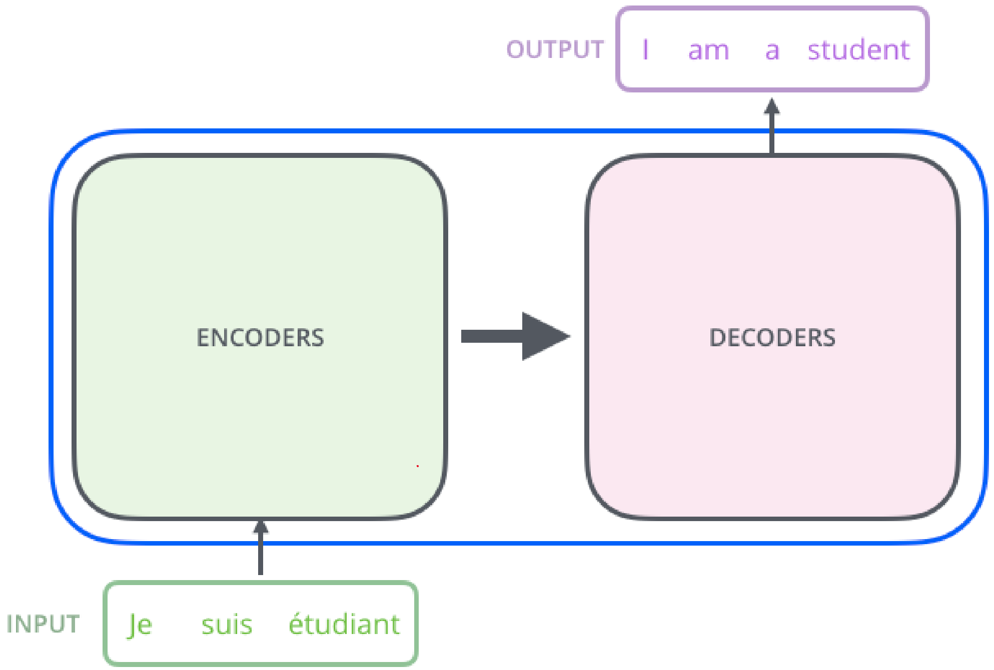
If sequence is innately longer than 512, break into 512 sized blocks.

A High-Level Look

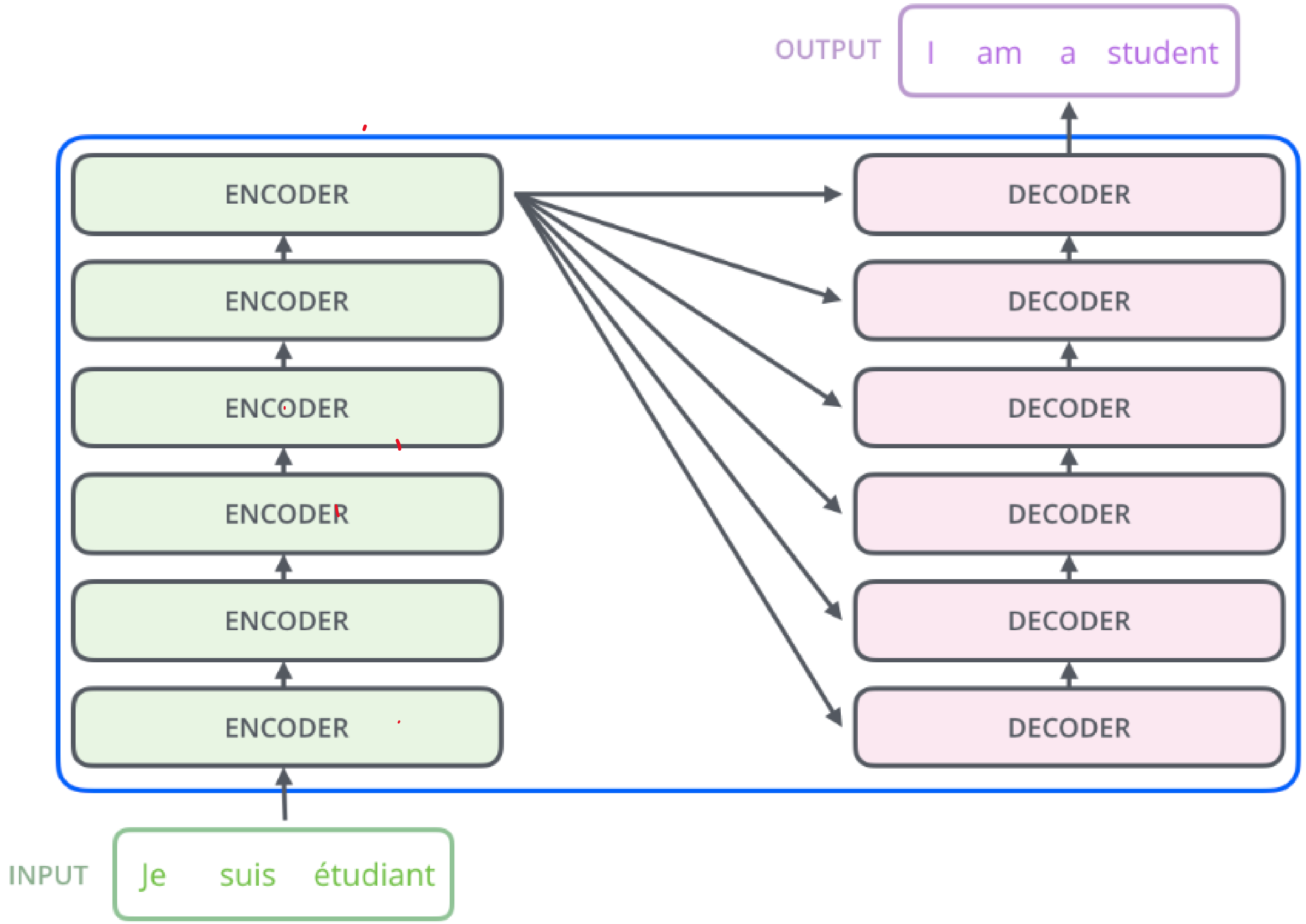
Let's begin by looking at the model as a single black box. In a machine translation application, it would take a sentence in one language, and output its translation in another.

Ref: https://jalammar.github.io/illustrated-transformer/

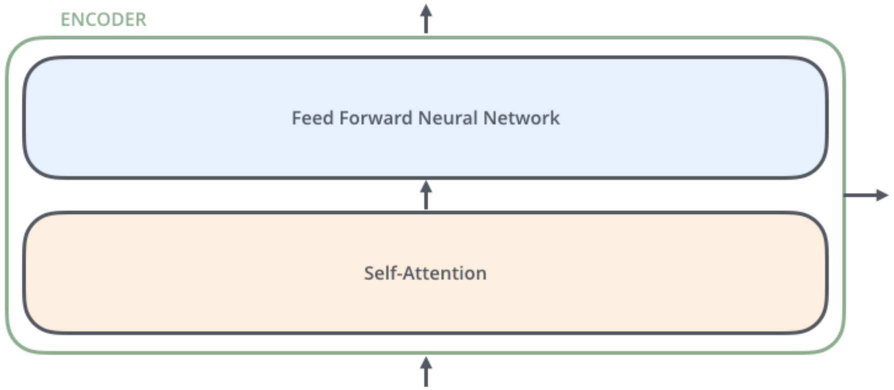
An Encoding component, a decoding component, and connections between them



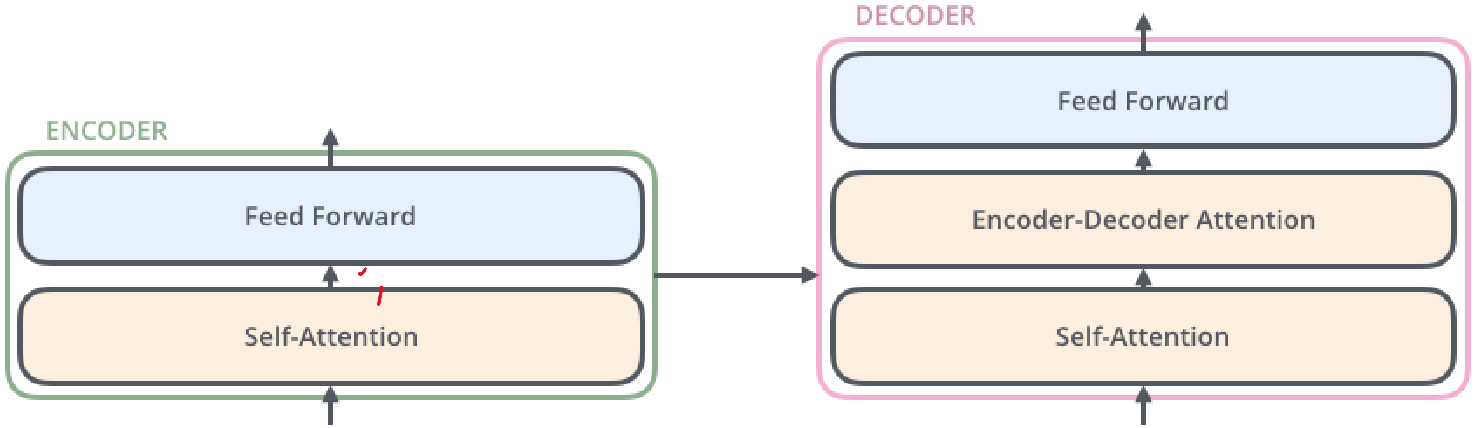
The **encoding component** is a stack of encoders (the paper stacks six of them on top of each other – there's nothing magical about the number six, one can definitely experiment with other arrangements). The **decoding component** is a stack of decoders of the same number.



The encoders are all identical in structure (yet they do not share weights). Each one is broken down into two sub-layers:

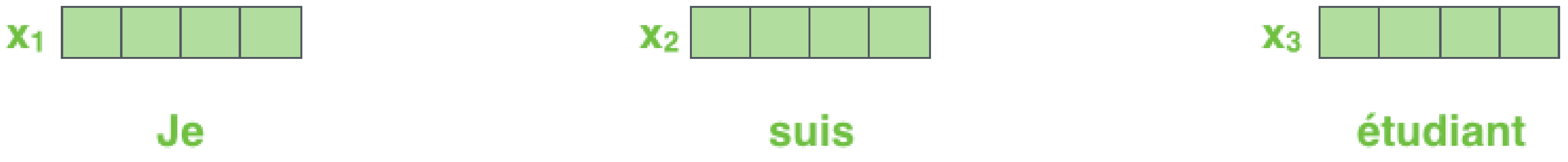


The encoder’s inputs first flow through a self-attention layer – a layer that helps the encoder look at other words in the input sentence as it encodes a specific word. We’ll look closer at self-attention later in the post.
The outputs of the self-attention layer are fed to a feed-forward neural network. The exact same feed-forward network is independently applied to each position.
The decoder has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence (similar what attention does in seq2seq models).

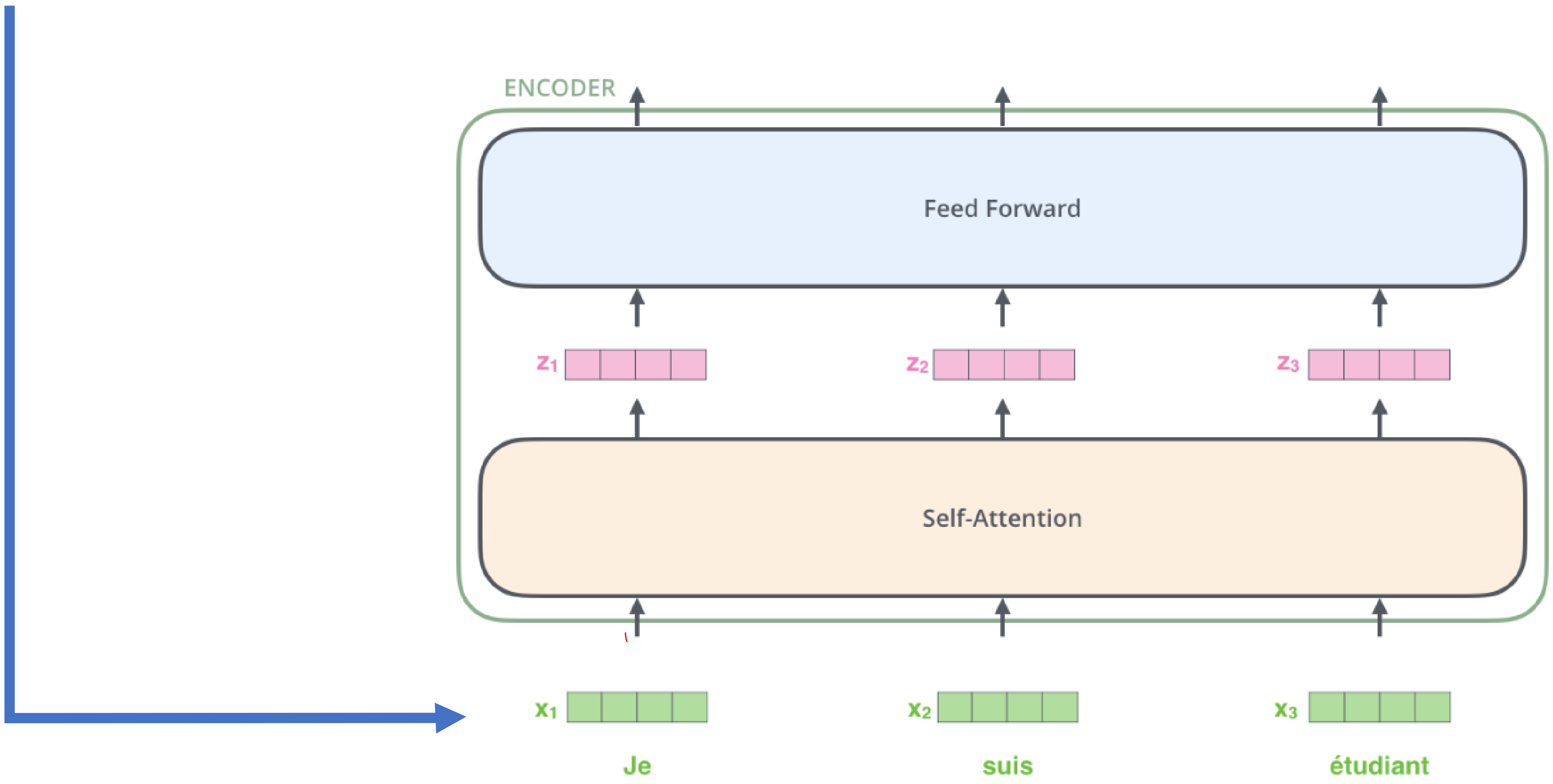


Bringing The Tensors Into The Picture

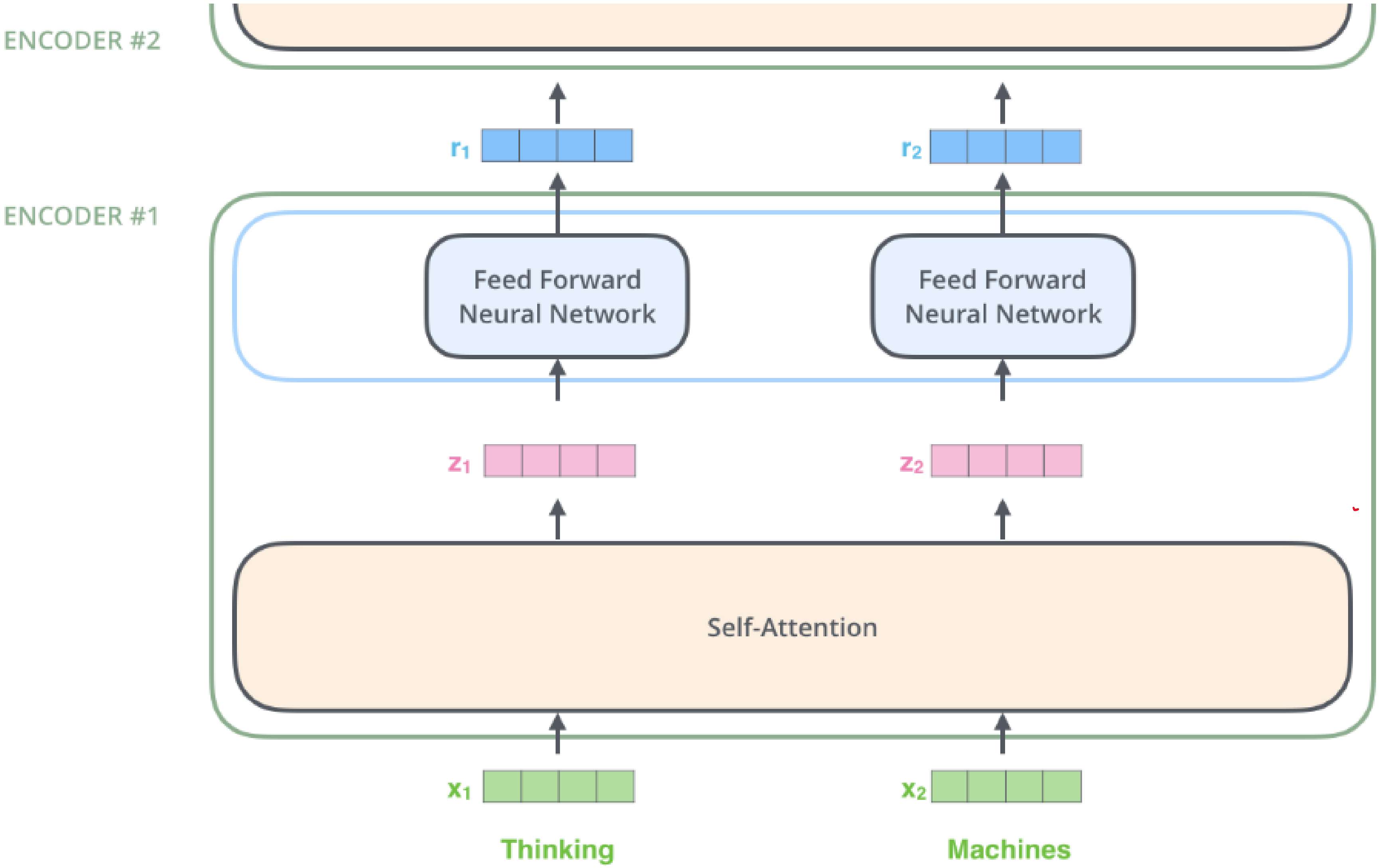
As is the case in NLP applications in general, we begin by turning each input word into a vector using an embedding algorithm.



Each word is embedded into a vector of size 512. We'll represent those vectors with these simple boxes.



Now Encoding!

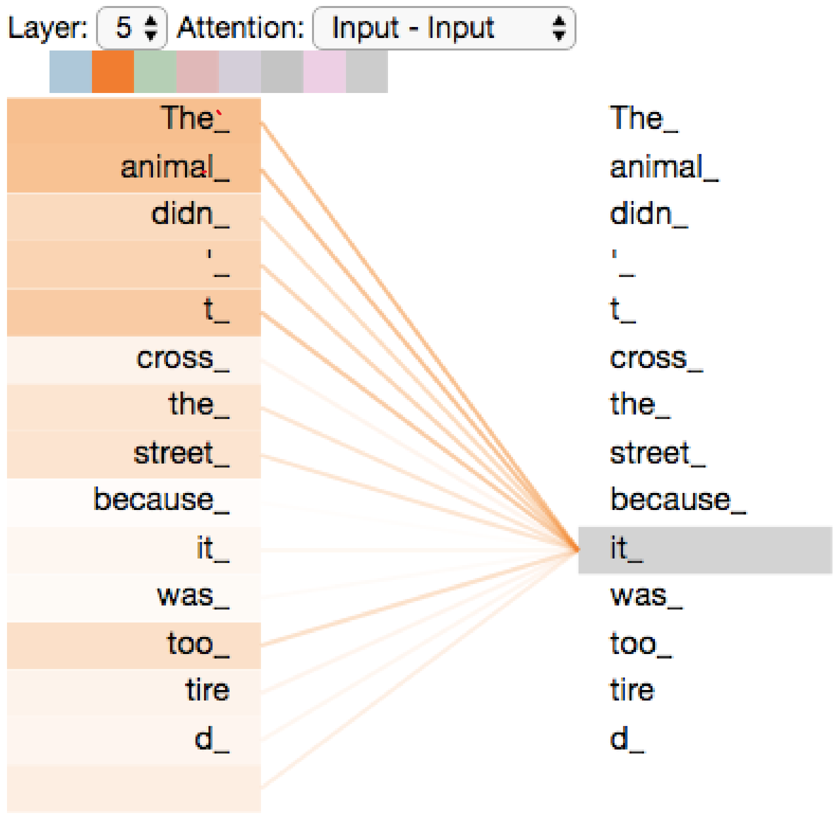


The word at each position passes through a self-attention process. Then, they each pass through a feed-forward neural network -- the exact same network with each vector flowing through it separately.

Self-Attention head

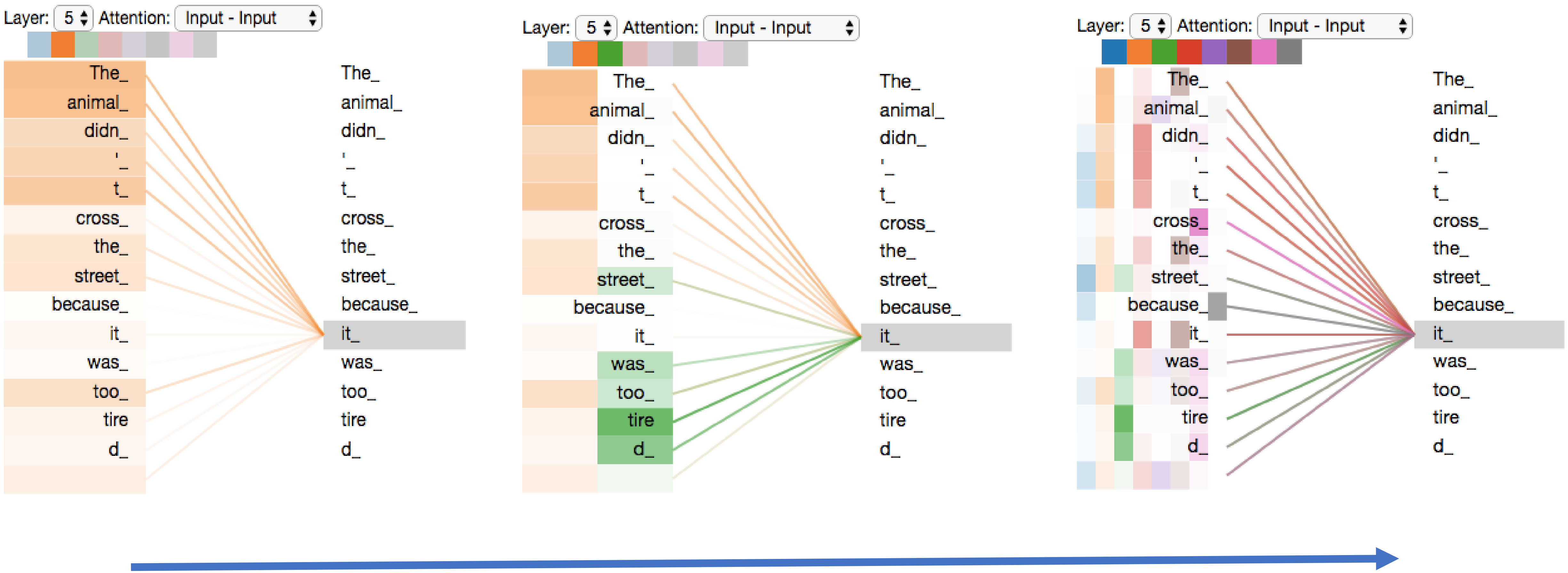
Say the following sentence is an input sentence we want to translate:

"`The animal didn't cross the street because it was too tired`"



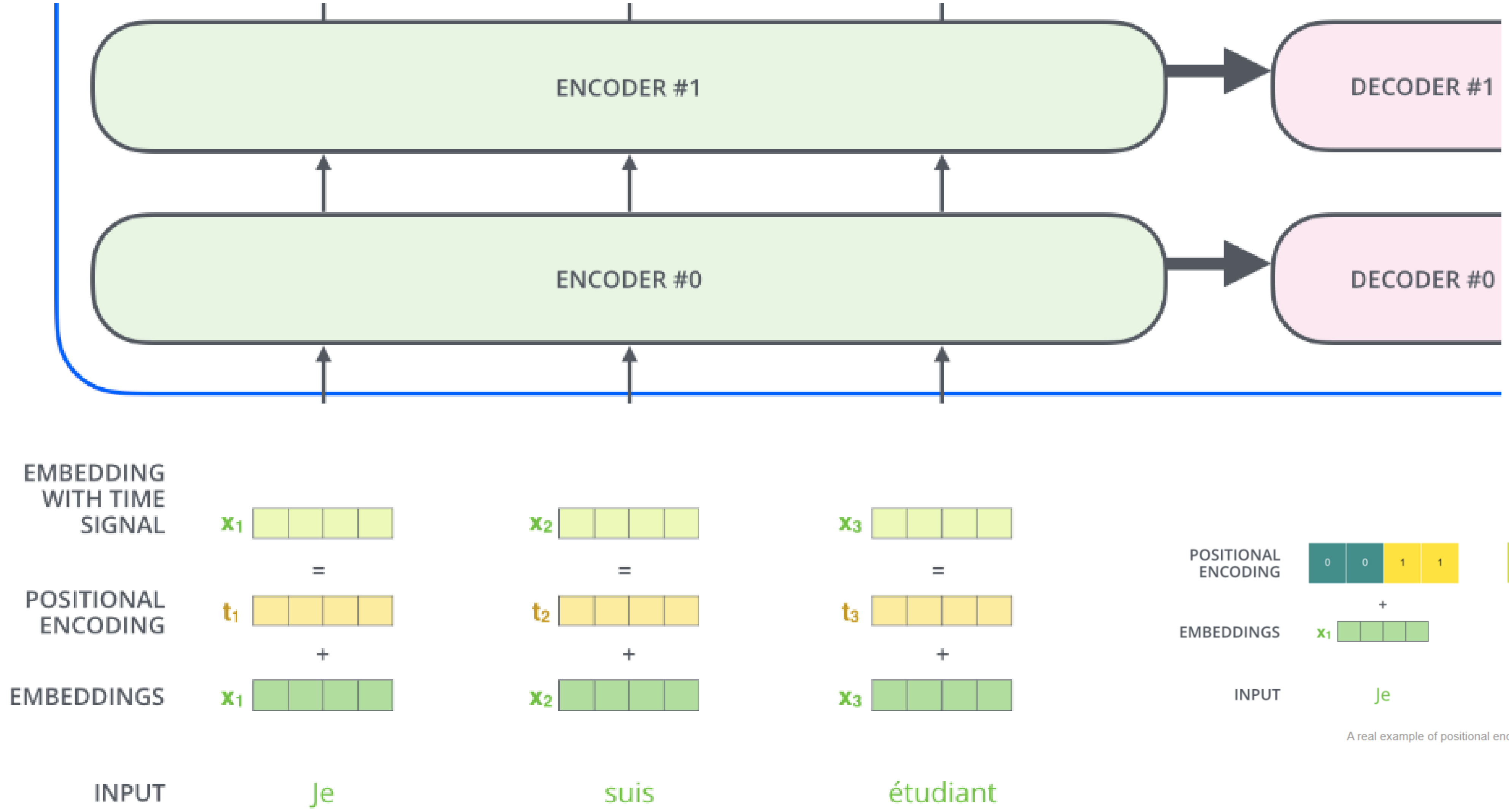
As we are encoding the word "it" in encoder #5 (the top encoder in the stack), part of the attention mechanism was focusing on "The Animal", and baked a part of its representation into the encoding of "it".

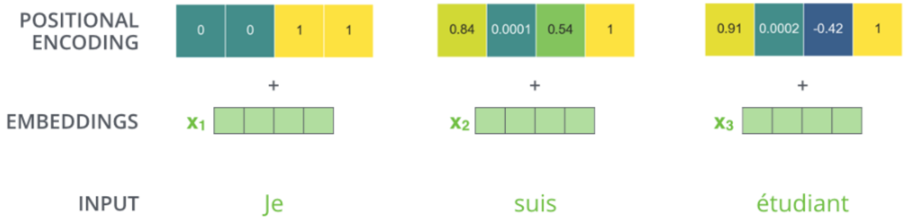
Self-Attention



One attention head

All attention heads

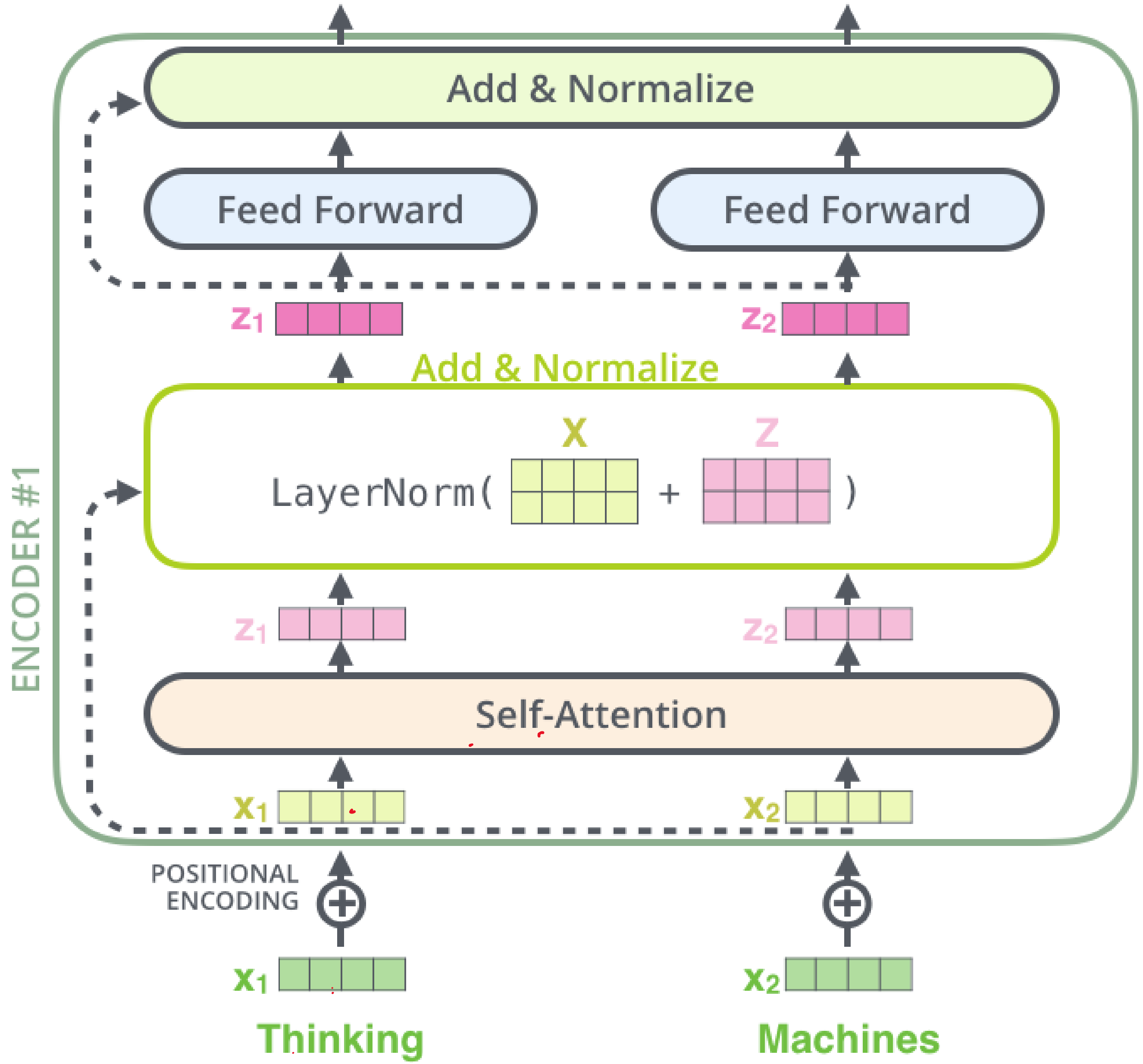




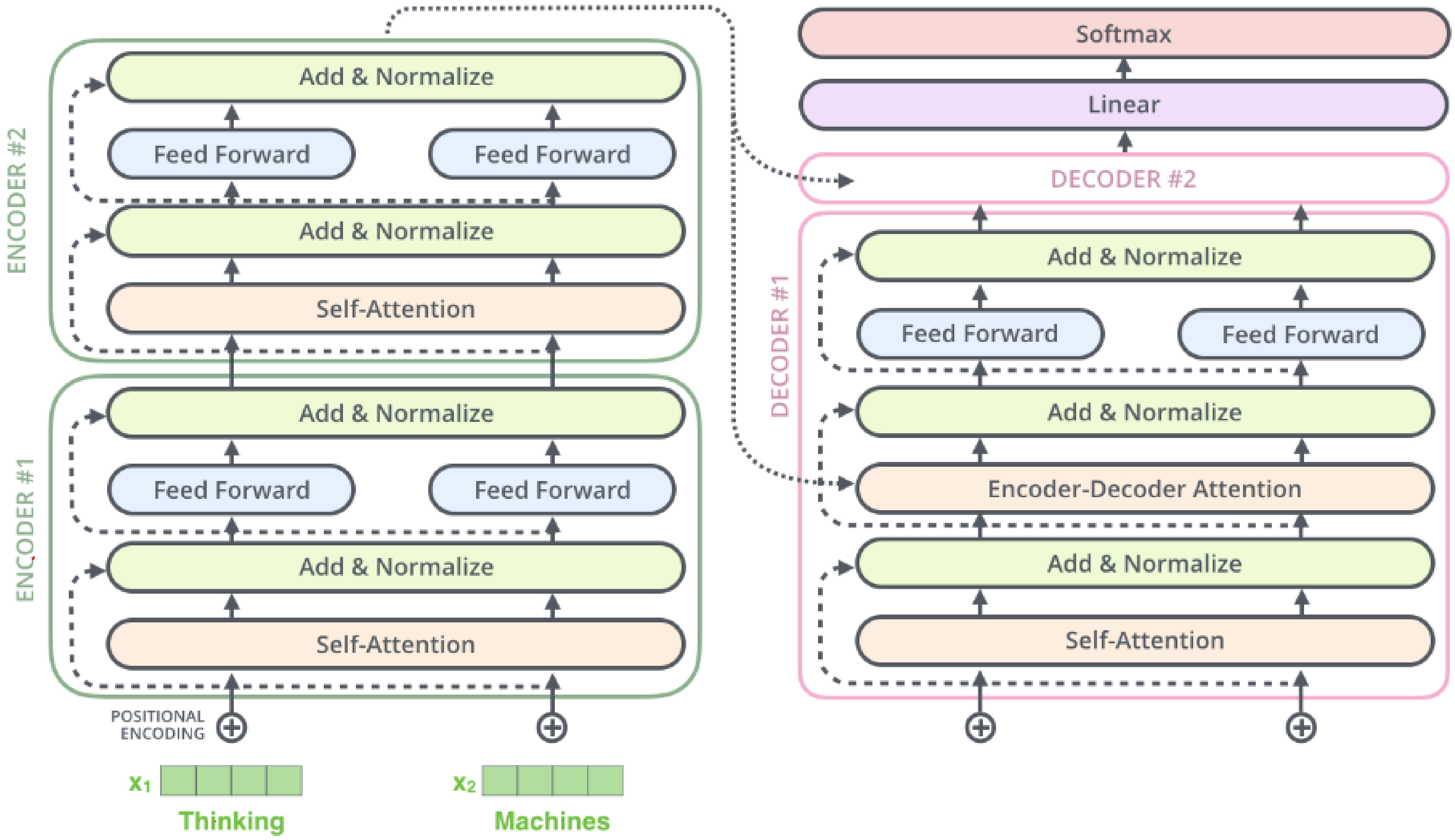
A real example of positional encoding with a toy embedding size of 4

To give the model a sense of the order of the words, we add positional encoding vectors -- the values of which follow a specific pattern.

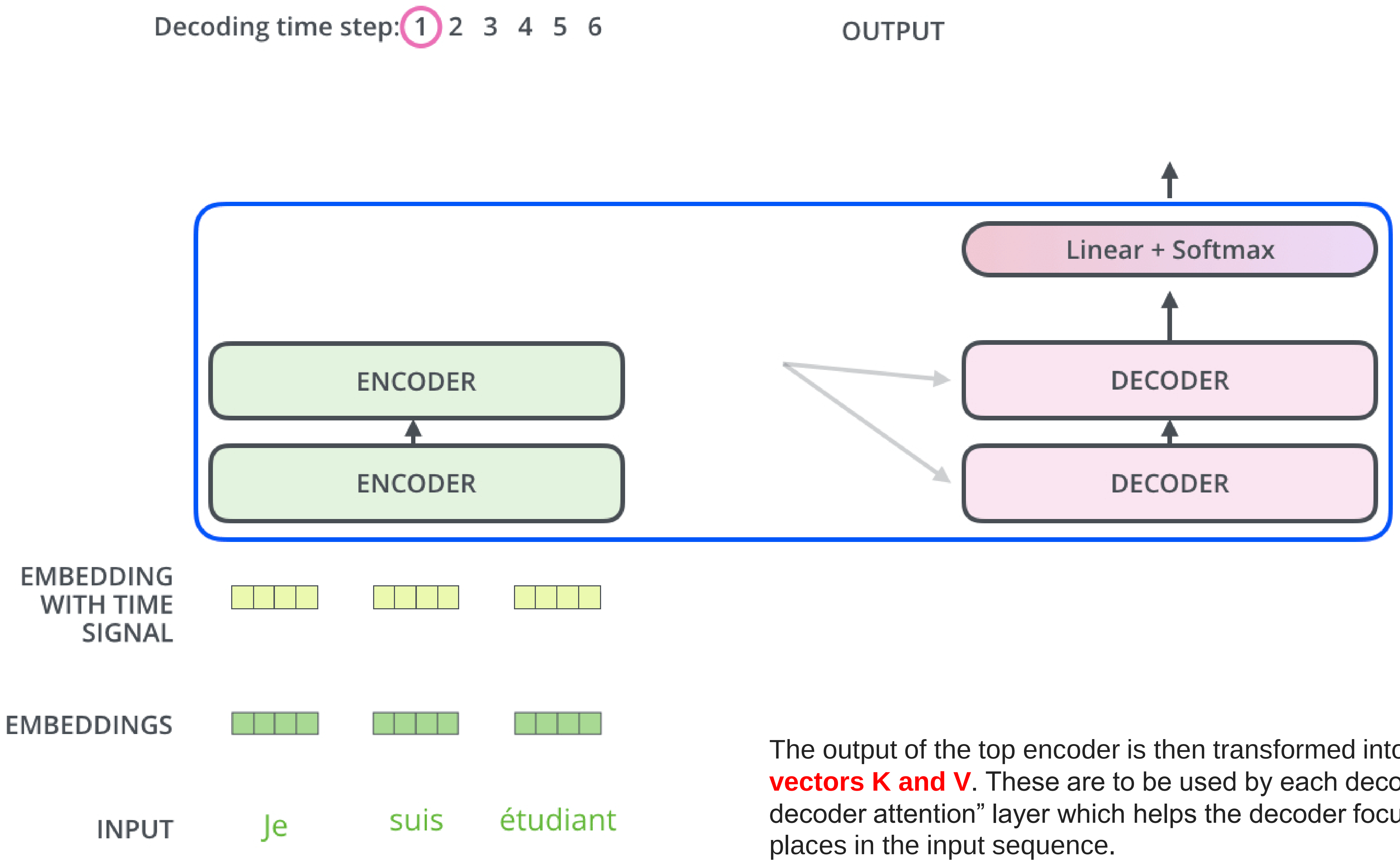
Complete Encoder



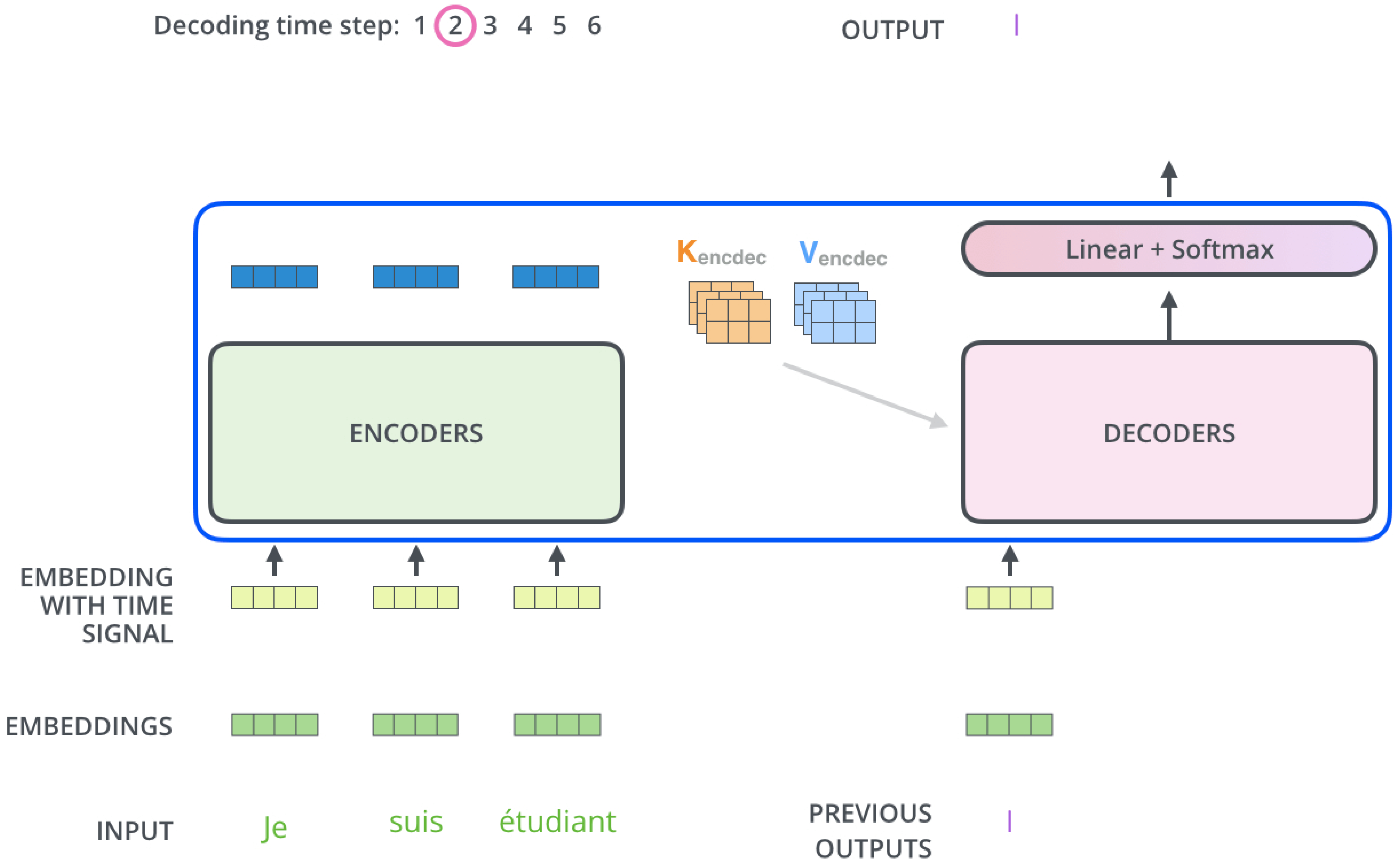
Transformer Architecture



The Decoder

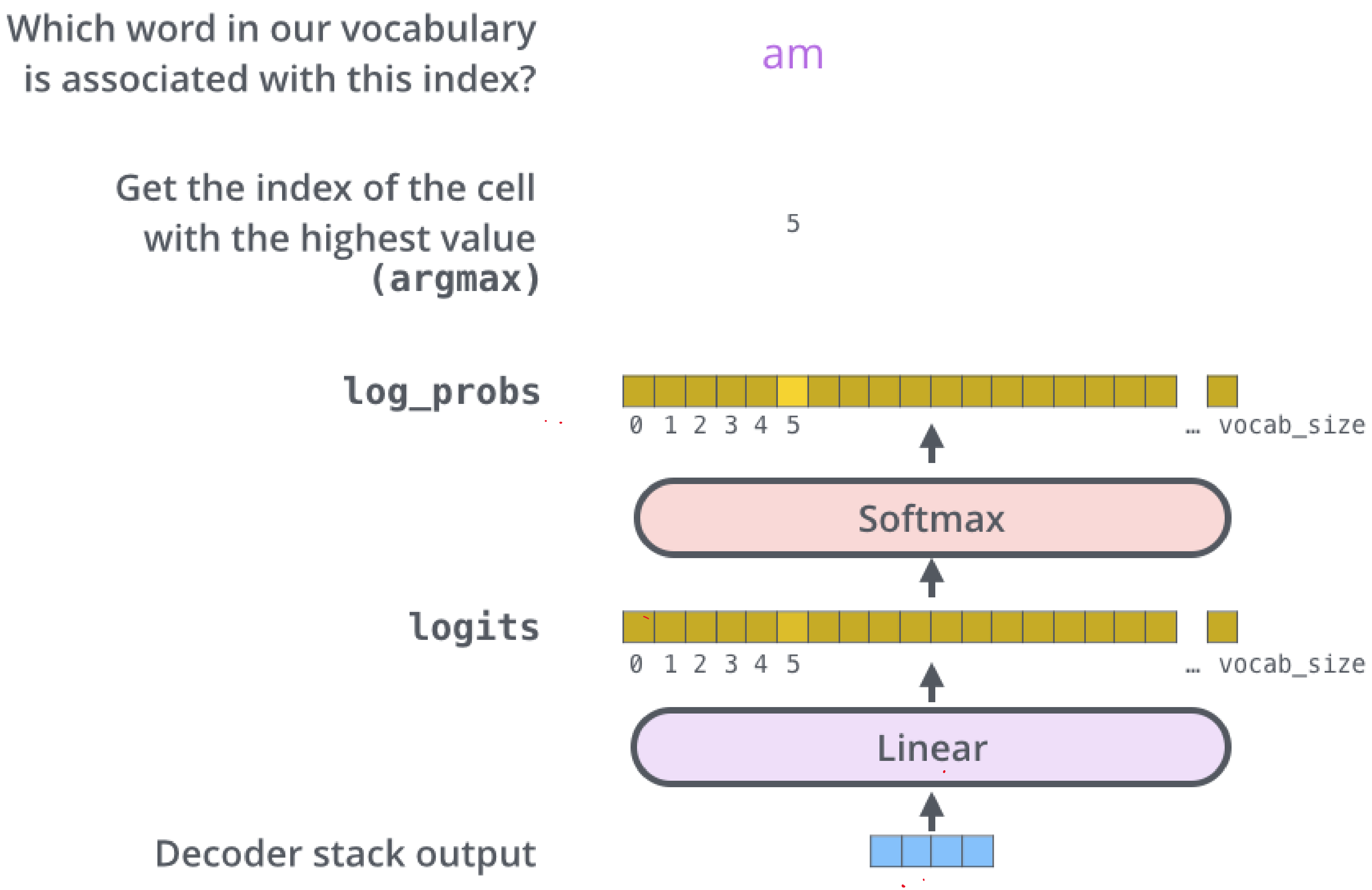


The output of the top encoder is then transformed into a set of **attention vectors K and V**. These are to be used by each decoder in its “encoder-decoder attention” layer which helps the decoder focus on appropriate places in the input sequence.



The output of each step is fed to the bottom decoder in the next time step, and the decoders bubble up their decoding results just like the encoders did.

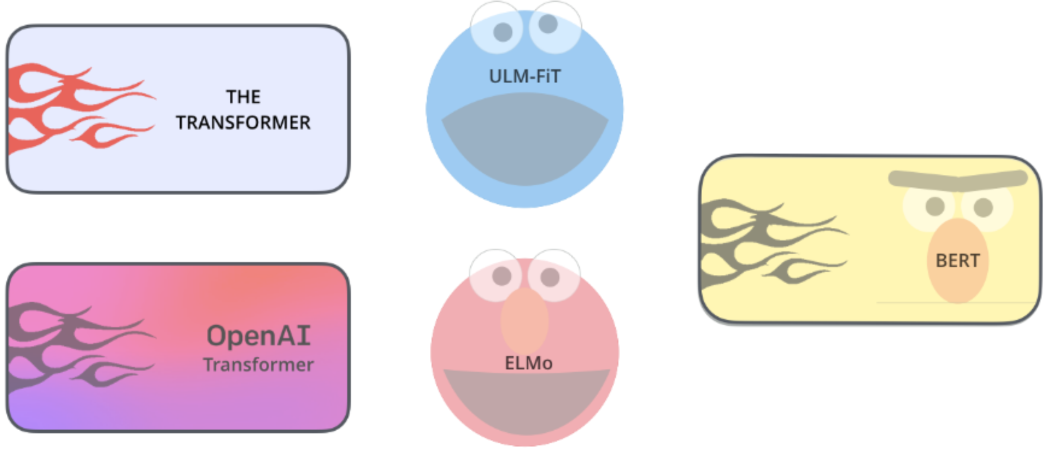
The Final Linear and Softmax Layer

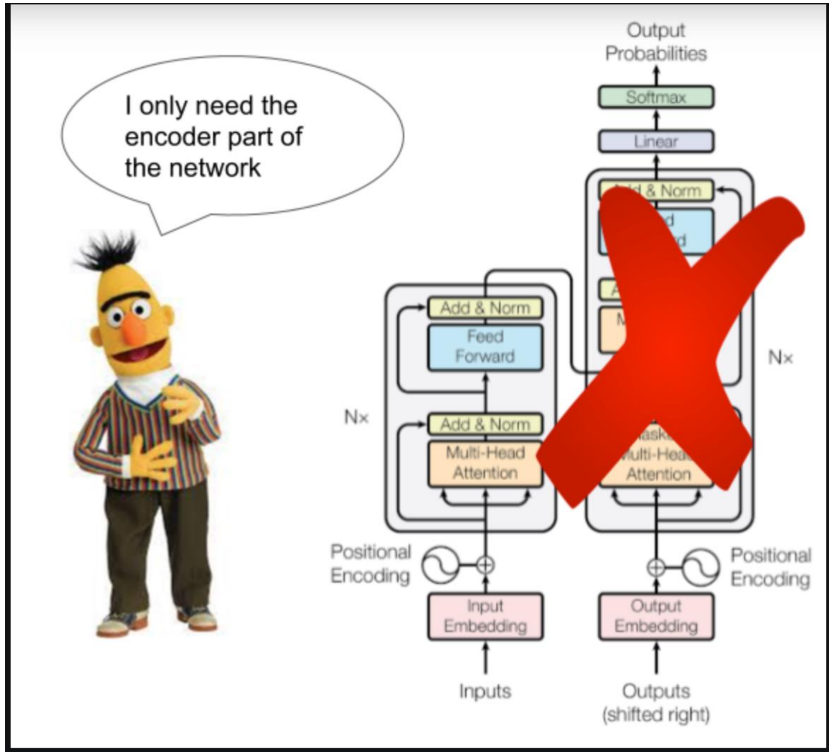


This figure starts from the bottom with the vector produced as the output of the decoder stack. It is then turned into an output word.

BERT - Bidirectional Encoder Representations from Transformers

BERT is a specific transformer-based language model
Produces *contextual* word embeddings (like ELMO)

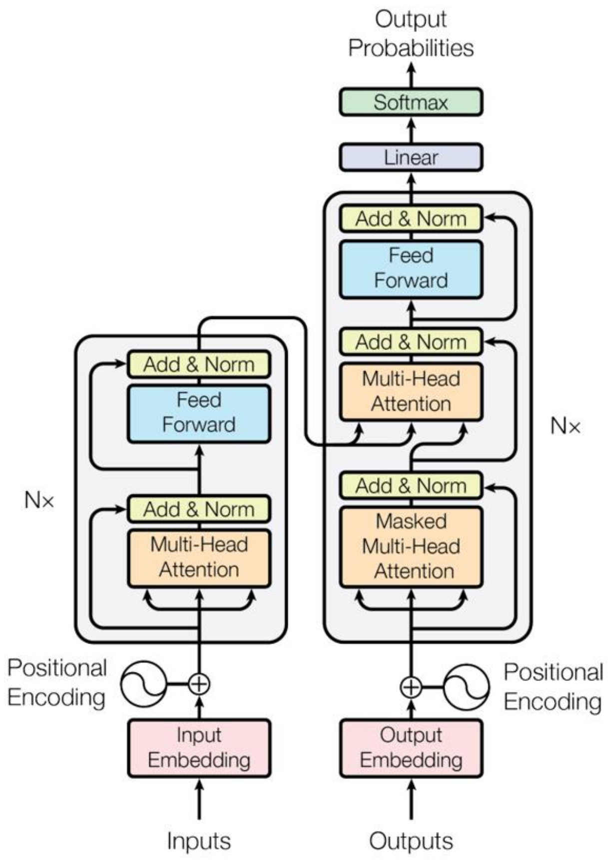




BERT only uses the encoder part of the original Transformer network

BERT

Encoder



GPT

Decoder

Aspect	BERT	GPT
Model Type	Transformer-based Encoder	Transformer-based Decoder
Directionality	Bidirectional	Unidirectional (Left-to-Right)
Layers	6 or 12 encoder layers	12 decoder Layers
Context Window	Fixed-length context	can vary depending on the length of the input text
Parameters	Over 100 million to 340 million parameters	117 million
Use Cases	NLP tasks like classification, NER, QA	Text generation, completion, conversation
Pre-training Objective	Masked Language Model (MLM)	Autoregressive Language Modeling
Fine-tuning	Task-specific layers added on top of BERT	Few-shot or one-shot task adaptation
Input Format	WordPiece or subword tokenization	Byte Pair Encoding (BPE) or similar
Contextual Embeddings	Used for both left and right context	Used for left context only

Hugging Face

Natural Language Processing

- Text Classification
- Token Classification
- Table Question Answering
- Question Answering
- Zero-Shot Classification
- Translation
- Summarization
- Feature Extraction
- Text Generation
- Text2Text Generation
- Fill-Mask
- Sentence Similarity

Audio

- Text-to-Speech
- Text-to-Audio
- Automatic Speech Recognition
- Audio-to-Audio
- Audio Classification
- Voice Activity Detection

Tabular

- Tabular Classification
- Tabular Regression

Reinforcement Learning

- Reinforcement Learning
- Robotics

Other

- Graph Machine Learning

Qwen/Qwen1.5-MoE-A2.7B
Text Generation • Updated 4 days ago • 3.57k • 120

pyp1/VoiceCraft
Updated 5 days ago • 124

ByteDance/AnimateDiff-Lightning
Text-to-Video • Updated 13 days ago • 94.1k • 414

google/gemma-7b
Text Generation • Updated Feb 28 • 138k • 2.7k

stabilityai/stable-code-instruct-3b
Text Generation • Updated 6 days ago • 3.32k • 112

mistralai/Mixtral-8x7B-Instruct-v0.1
Text Generation • Updated Feb 29 • 882k • 3.54k

IDKiro/sdxs-512-0.9
Text-to-Image • Updated 6 days ago • 3.88k • 84

NousResearch/OLMo-Bitnet-1B
Text Generation • Updated 4 days ago • 363 • 70

ByteDance/SDXL-Lightning
Text-to-Image • Updated 20 days ago • 617k • 1.51k

https://huggingface.co/

Final Projects

Thanks for submitting the group information

Choose one from the below list

a) Chatbot for Humber - students association assistant (Eng + French support)

b) Creative assistant for Humber marketing team (Eng + French support)

c) Document summarizer - PDF / word / reports (Eng + French support)