Attention and Transformer Basics

Mohamed Afham

Mohamed Afham

- Graduate Student @ Technical University of Darmstadt
- Former AI Resident @ Meta AI
- Former Research Intern @ MBZUAI
- BSc @ University of Moratuwa, Sri Lanka

Acknowledgements

- Content borrowed from Stanford course CS25: https://web.stanford.edu/class/cs25/
- Content borrowed from Harvard course CS287: https://harvard-iacs.github.io/CS287/schedule
- Content borrowed from Jay Alammar's blog post: https://jalammar.github.io/illustrated-transformer/

Attention Timeline

1990s — **Prehistoric Era**

Rule-based methods, parsing, RNNs, LSTMs

2014 — **Simple attention mechanisms**

2017 — **Beginning of transformers**

Attention is all you need

2018 — **Explosion of transformers in NLP**

BERT, GPT-3

2018-2020 — **Explosion into other fields**

Explosion into other fields: ViTs, Alphafold-2.

2021-2022 — **Start of Generative Era**

Codex, Decision Transformers, GPT-X, DALL·E

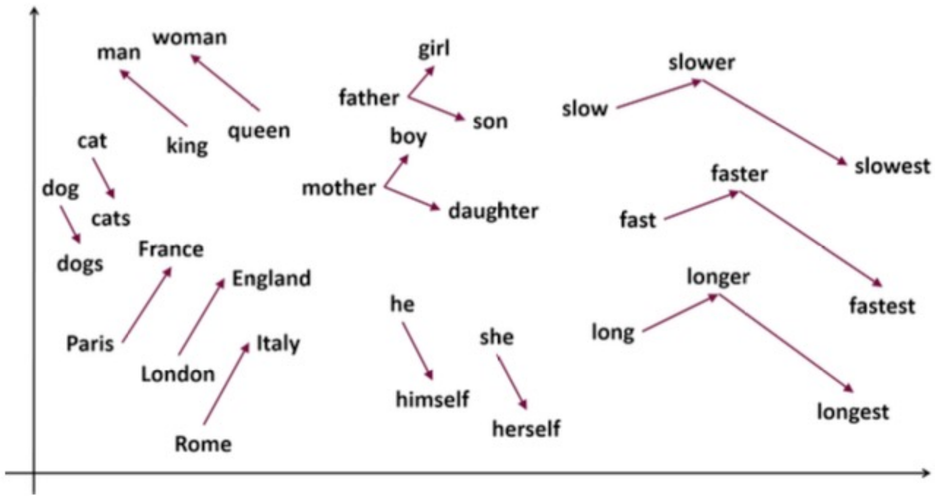
2024 — **Present Day**

Huge models, more applications: Chat-GPT, GPT-4, Gemini, Llama and open-source LLMs, Whisper, Robotics Transformer, Stable Diffusion, Sora, and so much more...!

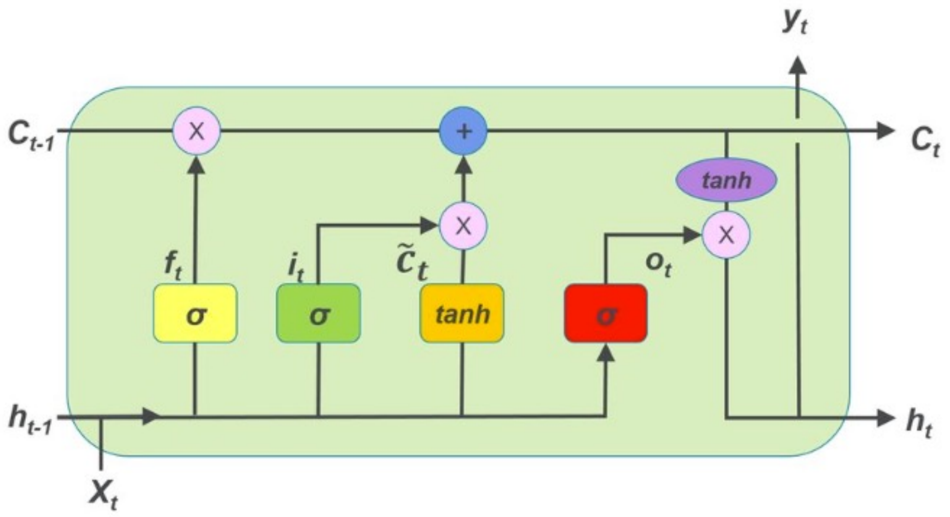
Future (?!)

Word Embeddings

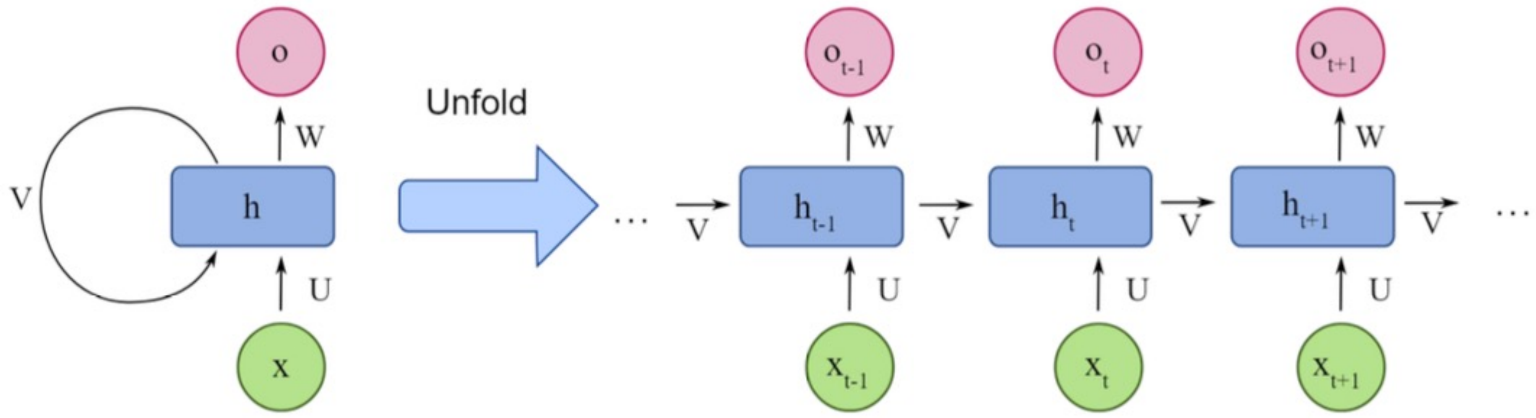
- Represent each word as a “vector” of numbers
- Converts a “discrete” representation to “continuous”, allowing for:
  - More “fine-grained” representations of words
  - Useful computations such as cosine/eucl distance
  - Visualization and mapping of words onto a semantic space
- Examples:
  - Word2Vec (2013), GloVe, BERT, ELMo



Seq2seq Models



- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory Networks (LSTMs)
- "Dependency" and info between tokens
- Gates to "control memory" and flow of information



Challenges in Seq2seq Models

- Unable to handle long-range dependencies
    - Very little contextual understanding
- Sequential Processing
    - Slower processing time
- Vanishing Gradient in long sequences

Transformers to the rescue

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

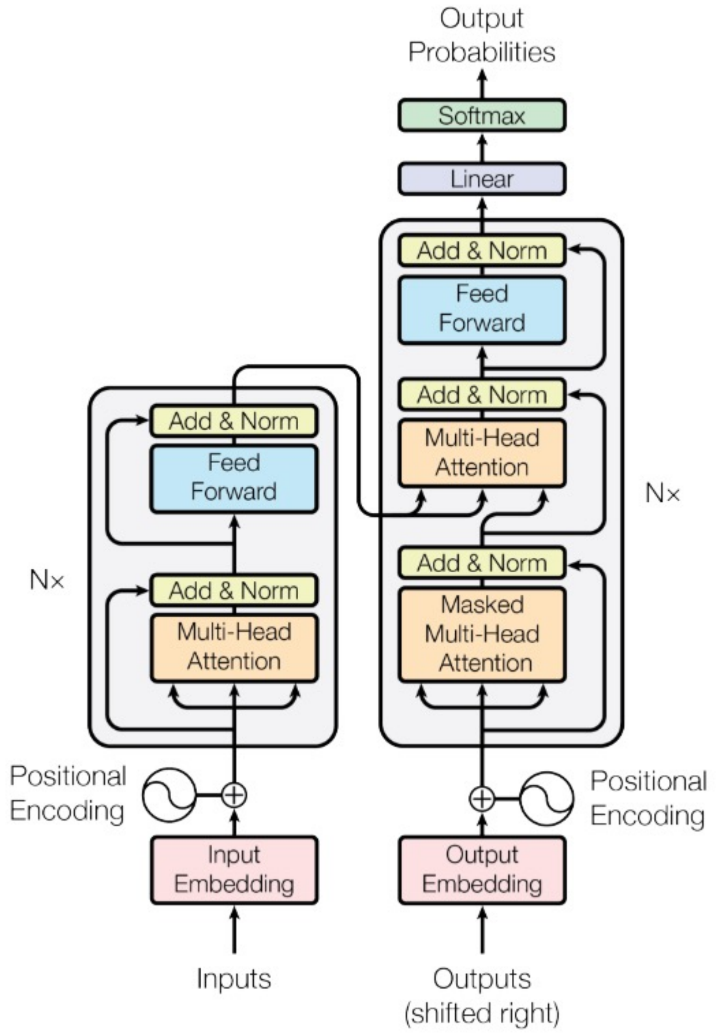
Jakob Uszkoreit[*]
Google Research
usz@google.com

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

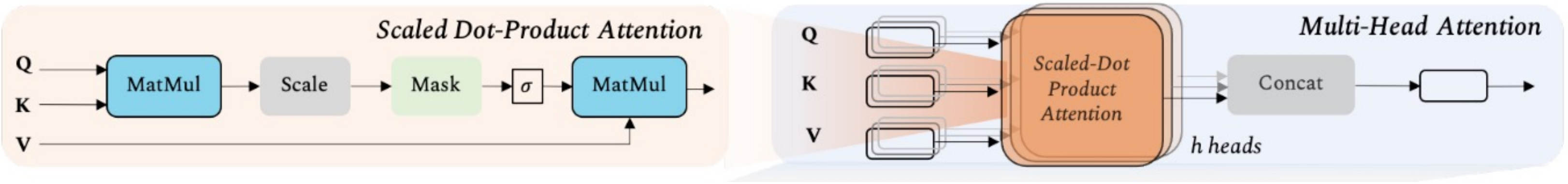


Over 120,000 citations and counting....

Key facts about transformers

- Introduced for machine translation task

- Self-attention mechanism



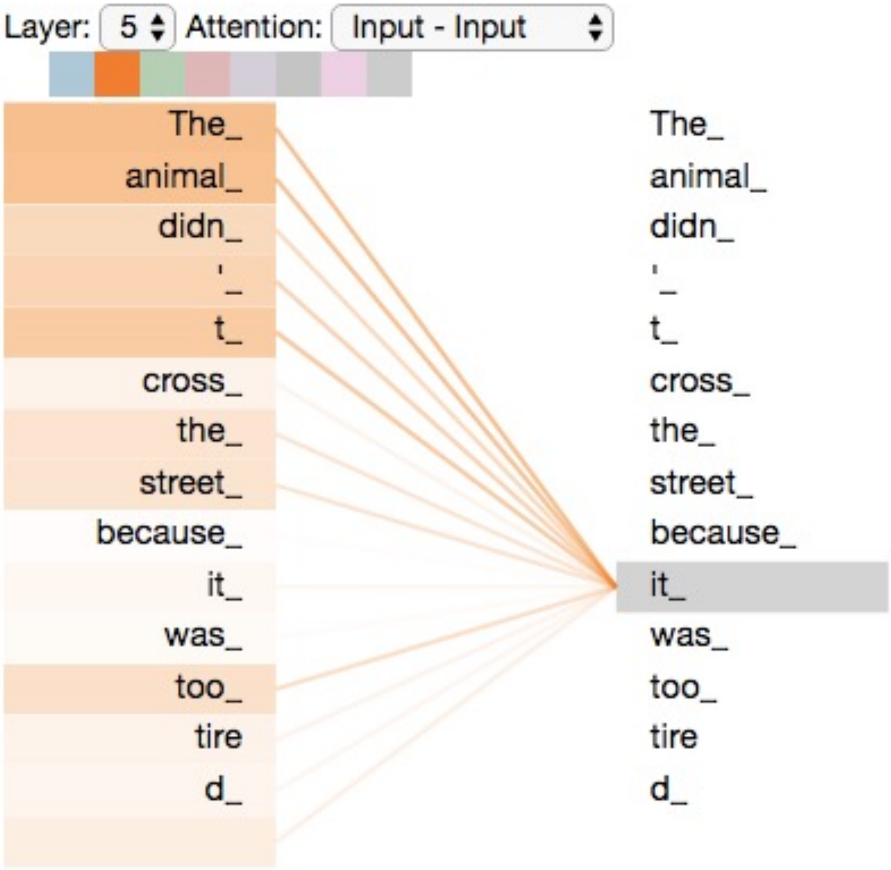
- Parallel processing
    - Self-attention mechanism allows for the parallel processing of entire sequences resulting in significant speed-ups in both training and inference as all elements of the sequence can be processed simultaneously

Attention in Machine Learning

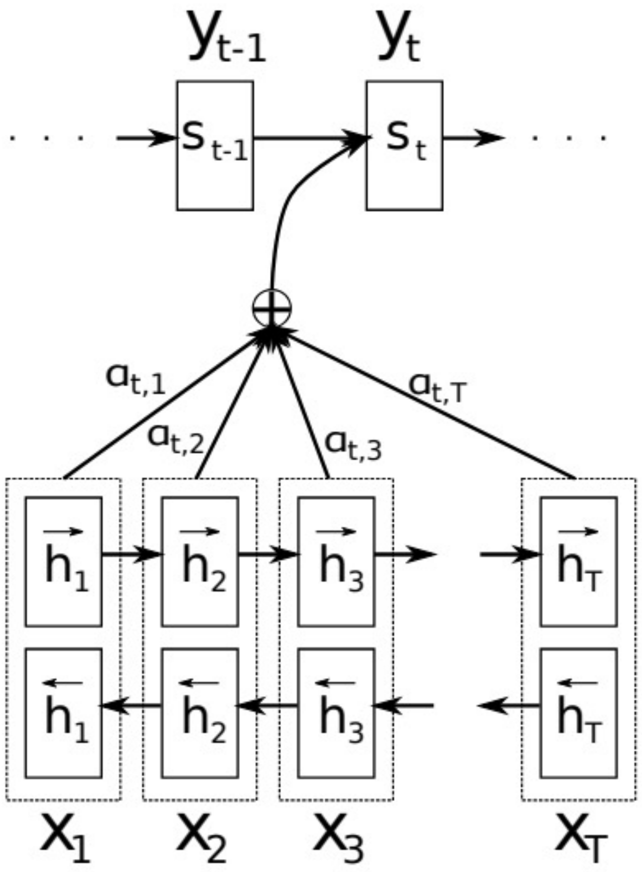
Consider the sentence:

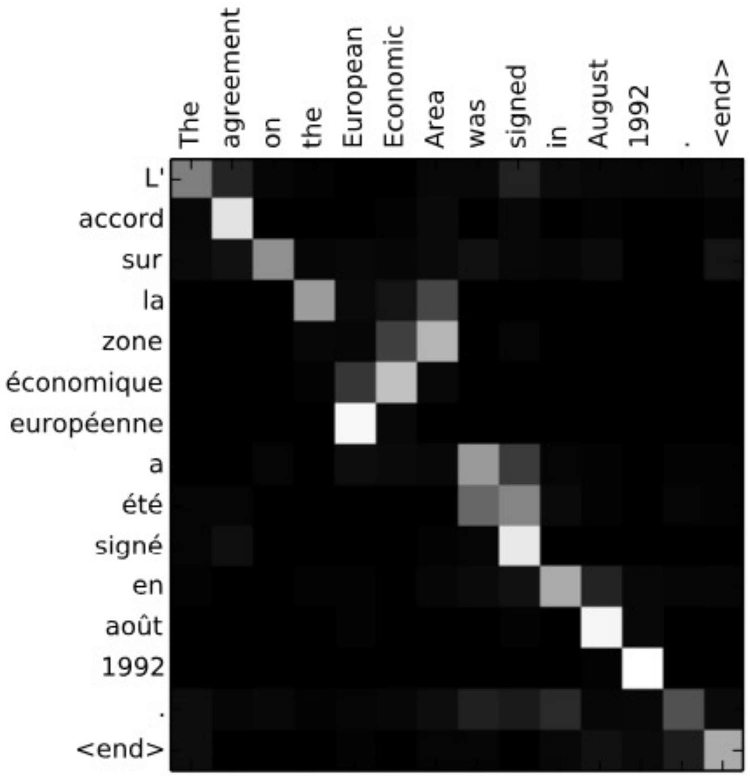
The animal didn't cross the street because it was too tired

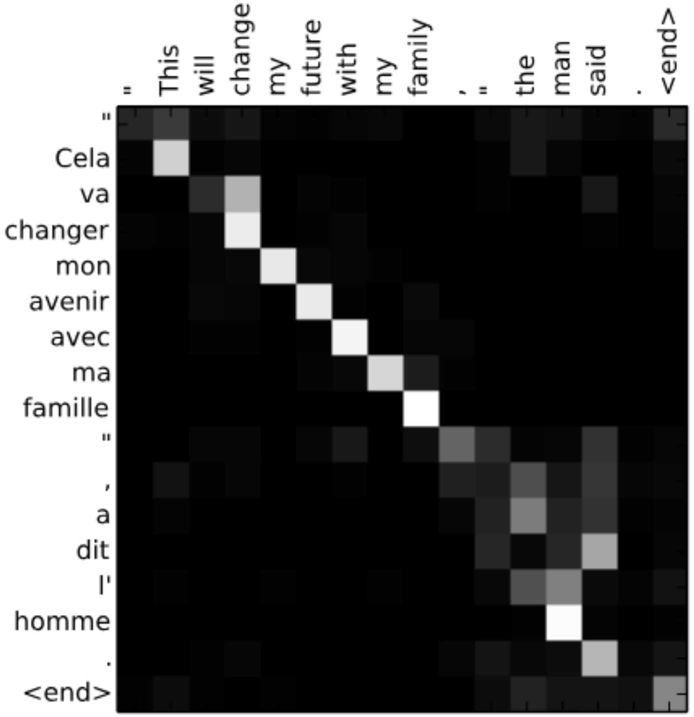
What does “it” in the sentence refer to ?



Attention in seq2seq models







Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention in image captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

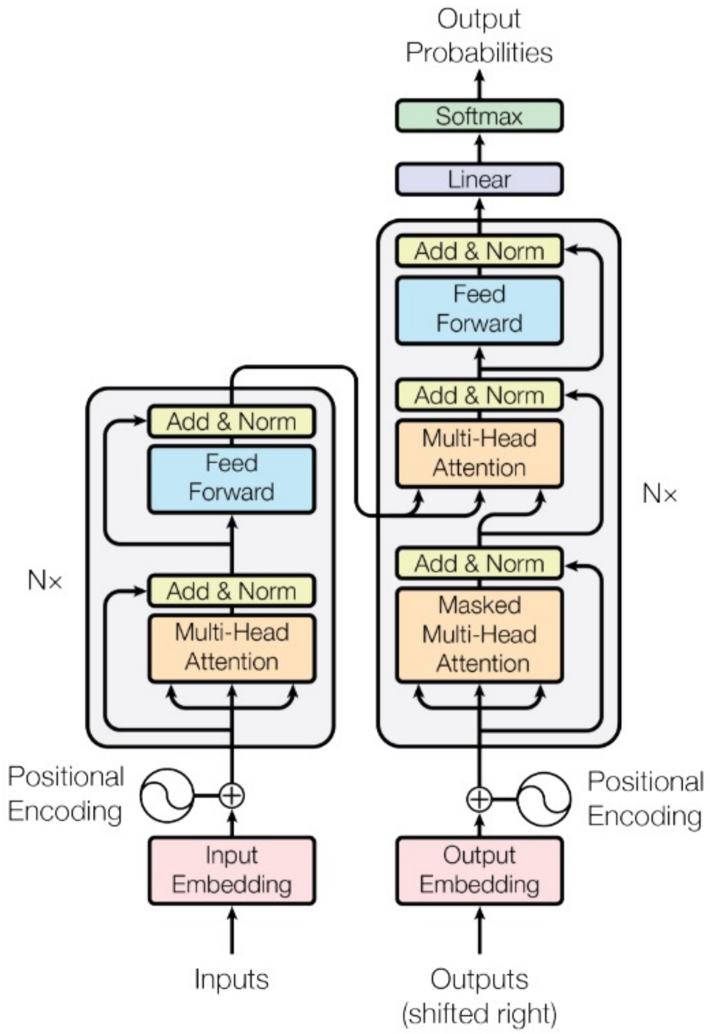
A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

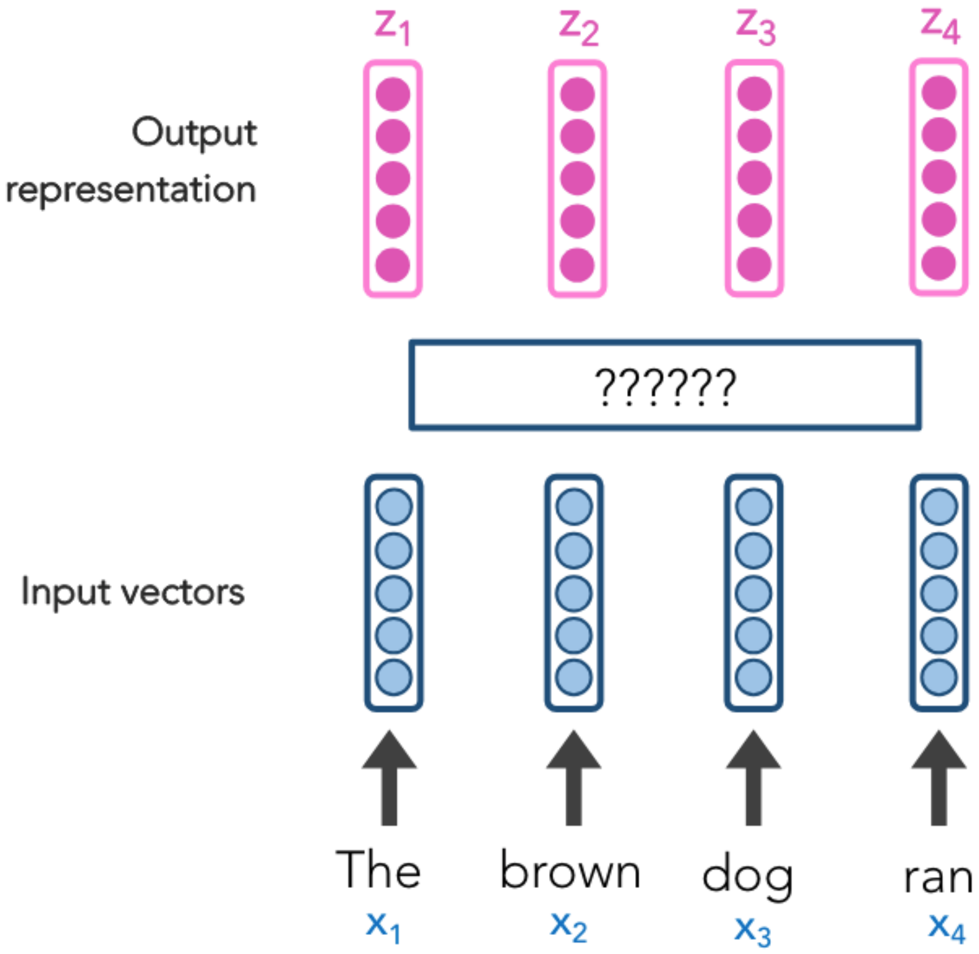
Self-Attention



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

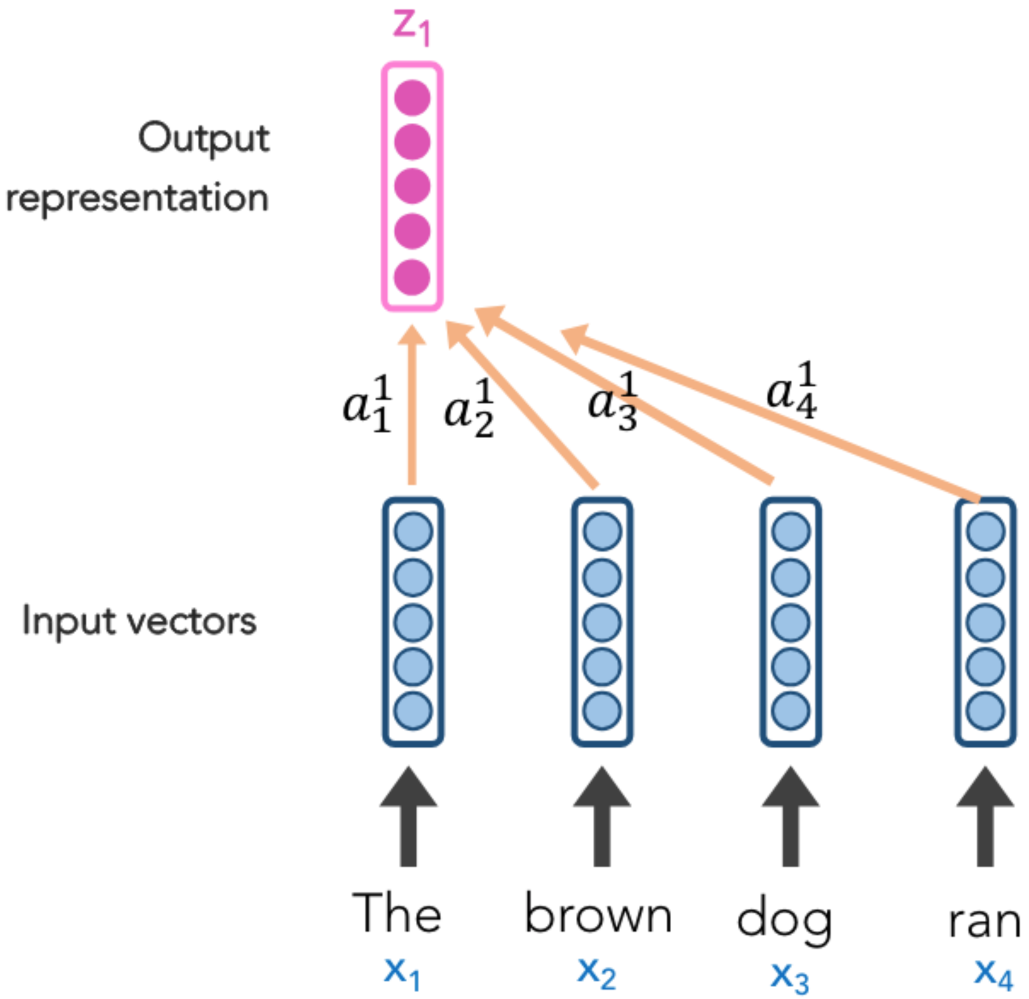
Recap: Softmax is a function that converts a vector of numbers into a probability distribution, where each value represents the probability of the corresponding class, and the sum of all probabilities is 1.

Self-Attention



The goal of self-attention is to create a contextual representation z_i of the input sequence

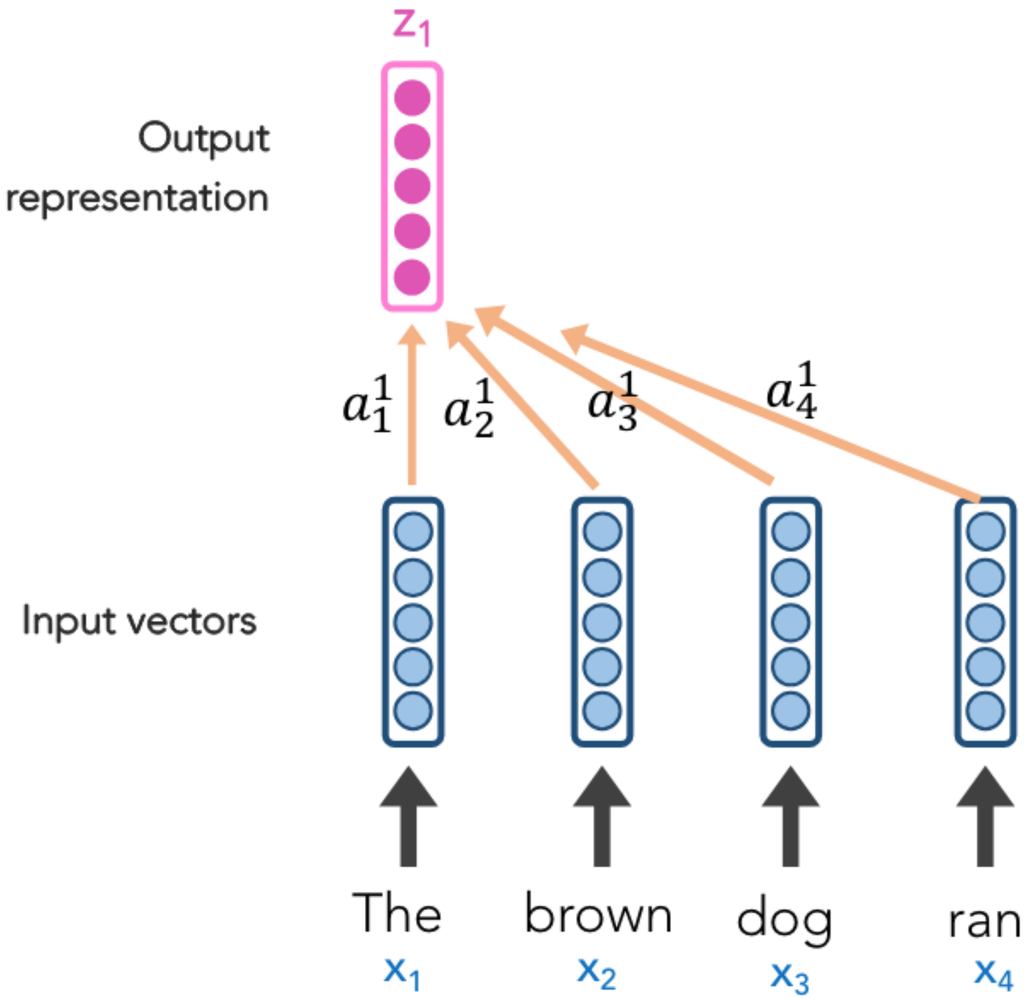
Self-Attention



The goal of self-attention is to create a contextual representation z_i of the input sequence

z_1 will be a weighted contribution of x_1, x_2, x_3, x_4

Self-Attention

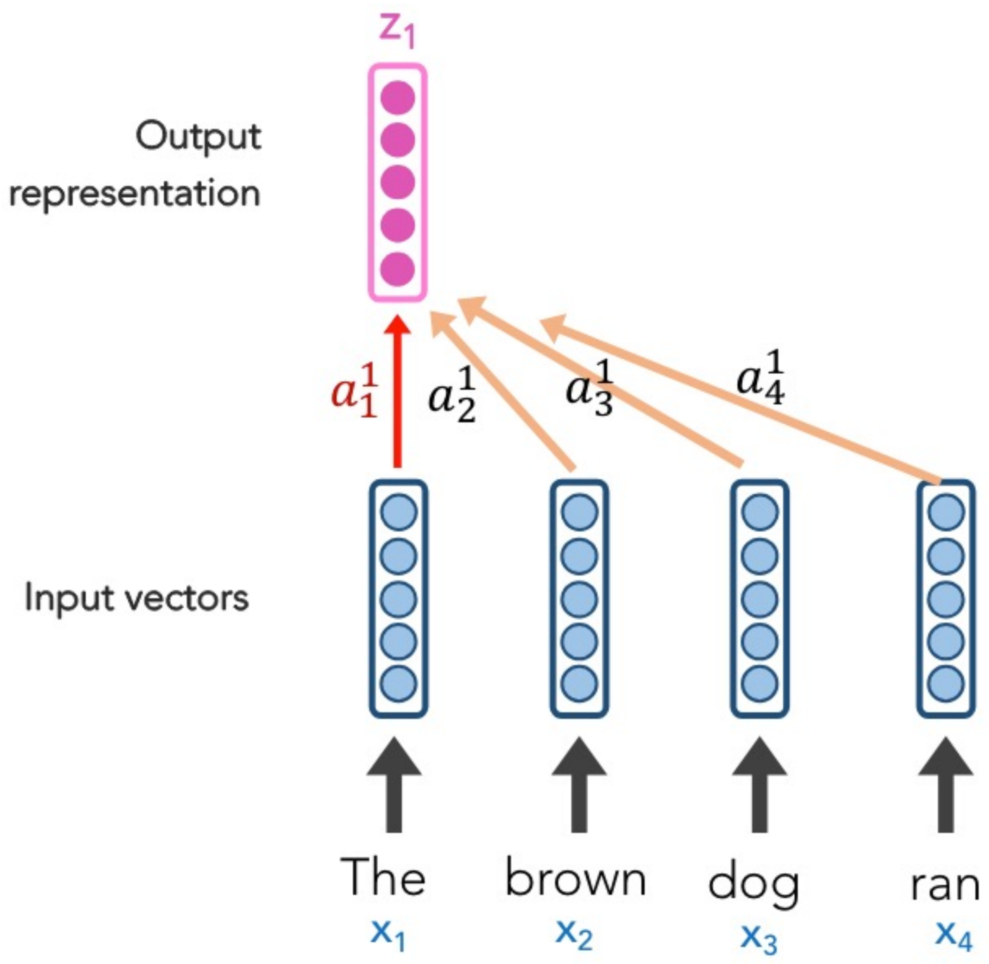


The goal of self-attention is to create a contextual representation z_i of the input sequence

z_1 will be a weighted contribution of x_1, x_2, x_3, x_4

a_i^1 could be considered as the weighted contribution of x_i towards the contextual representation z_1

Self-Attention



Each x_i has three associated vectors namely:

- Query q_i
- Key k_i
- Value v_i

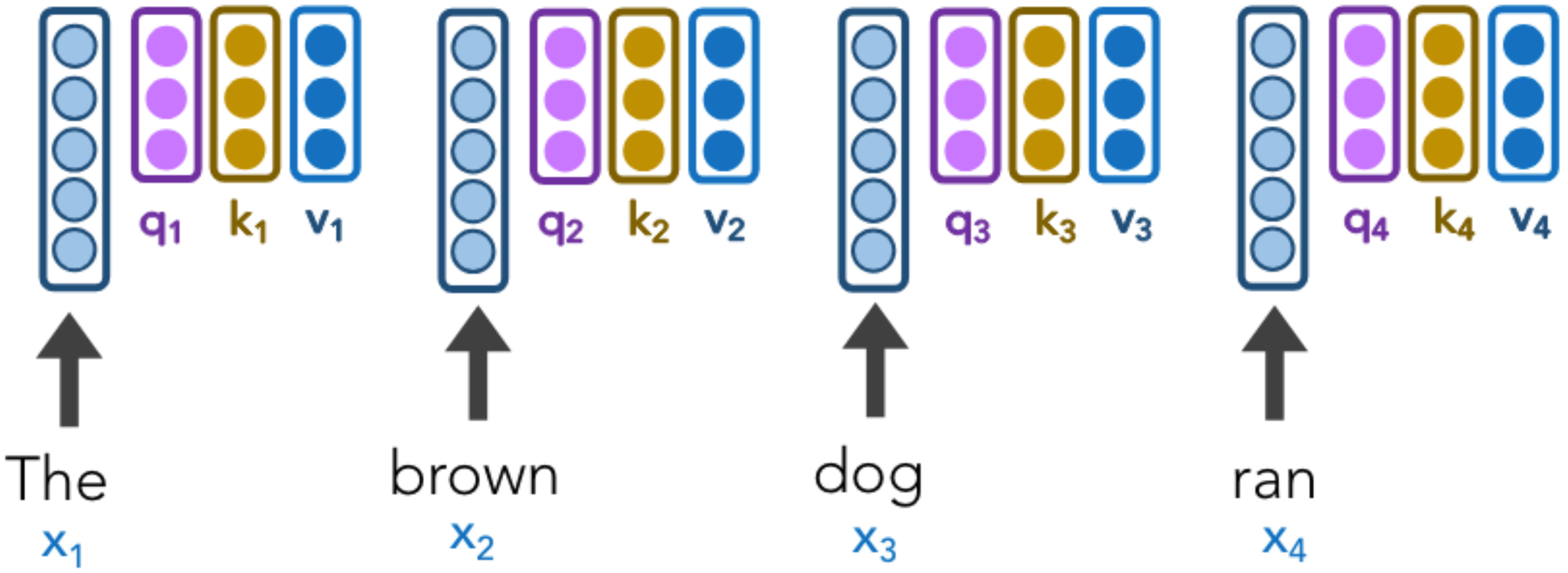
Self-Attention

One self-attention head has three weight matrices: W_q, W_k, W_v. The associated vectors are obtained by multiplying the input vector with the learnable weight matrices.

- $q_i = W_q \, x_i$
- $k_i = W_k \, x_i$
- $v_i = W_v \, x_i$

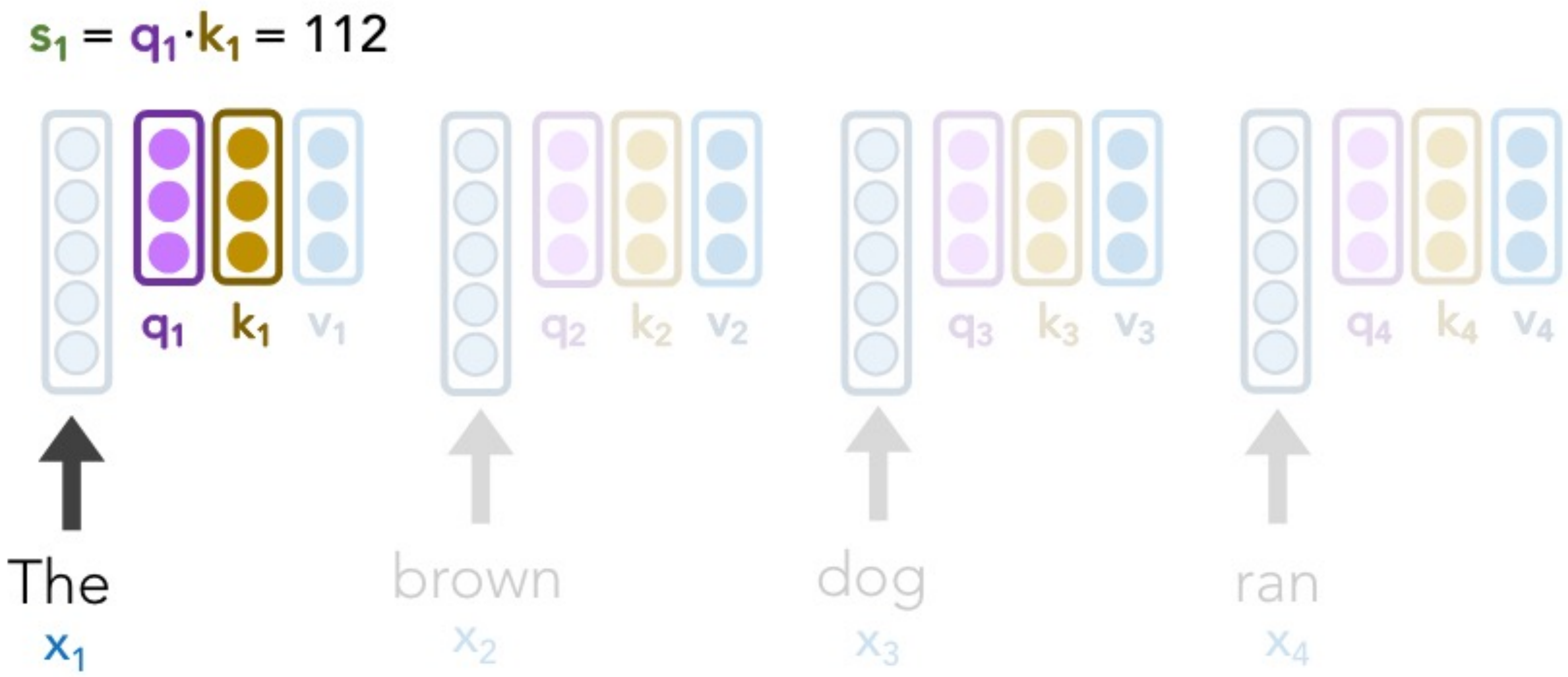
Each x_i has three associated vectors namely:

- Query q_i
- Key k_i
- Value v_i



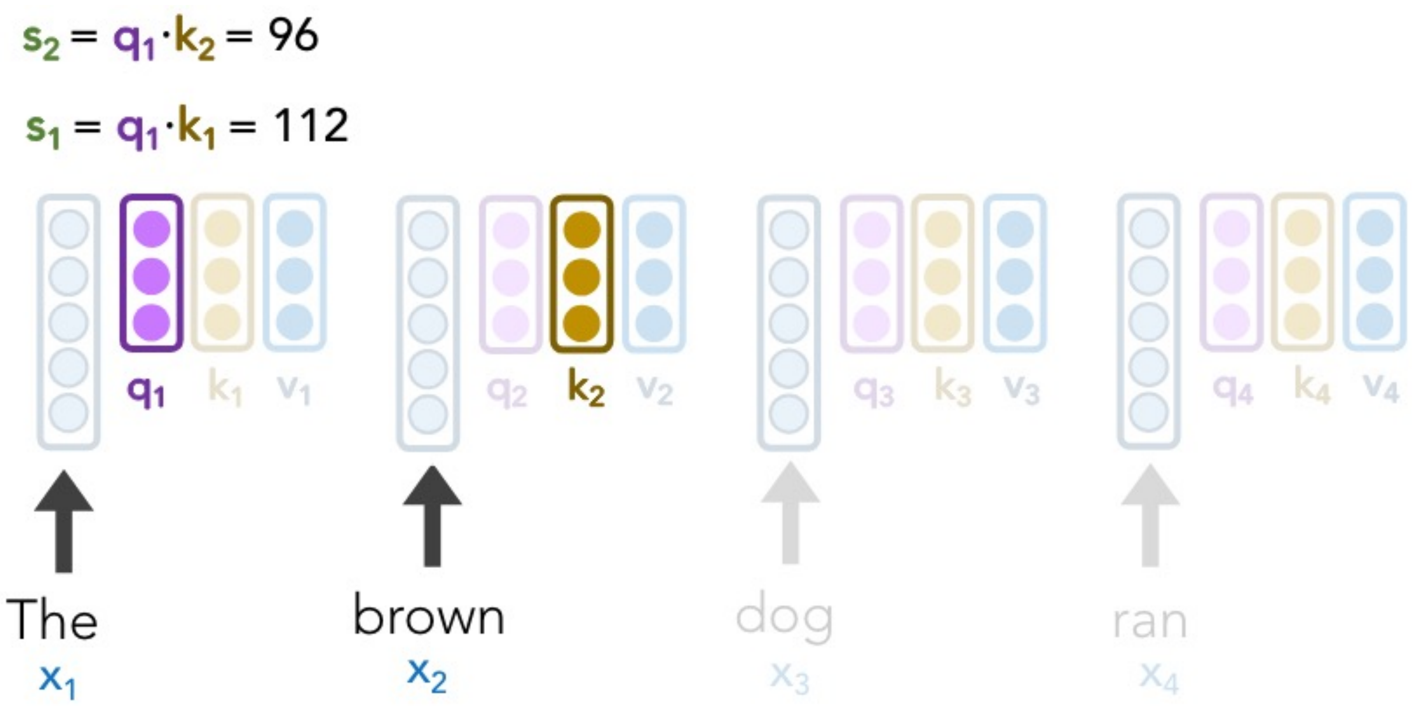
Self-Attention

For a given token (word), let's calculate the scores s_1, s_2, s_3, s_4 which represent how much attention to pay to each respective "token" v_i



Self-Attention

For a given token (word), let's calculate the scores s_1, s_2, s_3, s_4 which represent how much attention to pay to each respective "token" v_i



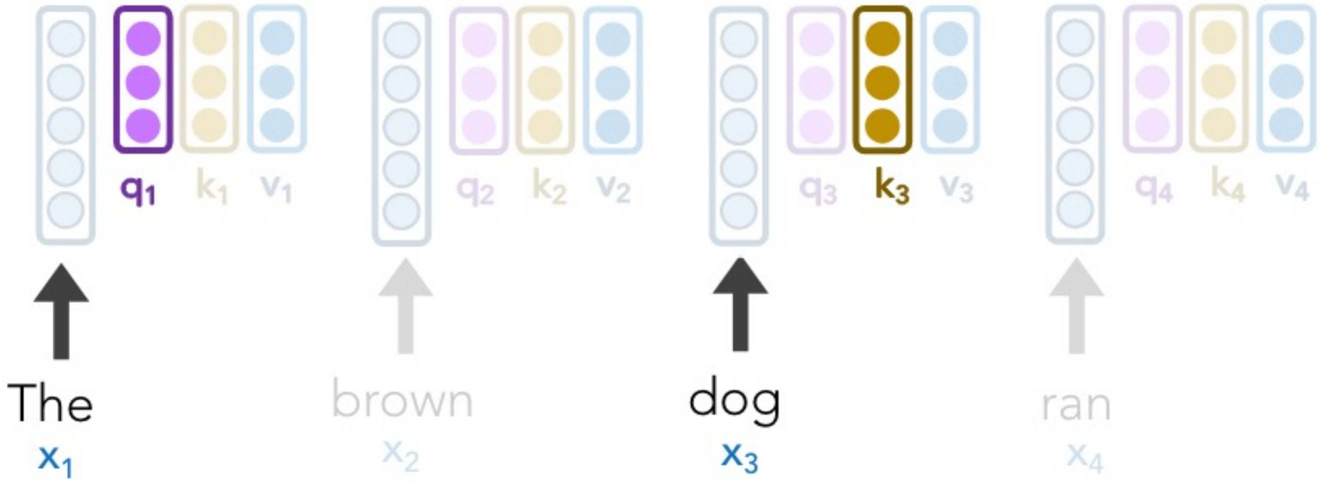
Self-Attention

For a given token (word), let's calculate the scores s_1, s_2, s_3, s_4 which represent how much attention to pay to each respective "token" v_i

$s_3 = q_1 \cdot k_3 = 16$

$s_2 = q_1 \cdot k_2 = 96$

$s_1 = q_1 \cdot k_1 = 112$



Self-Attention

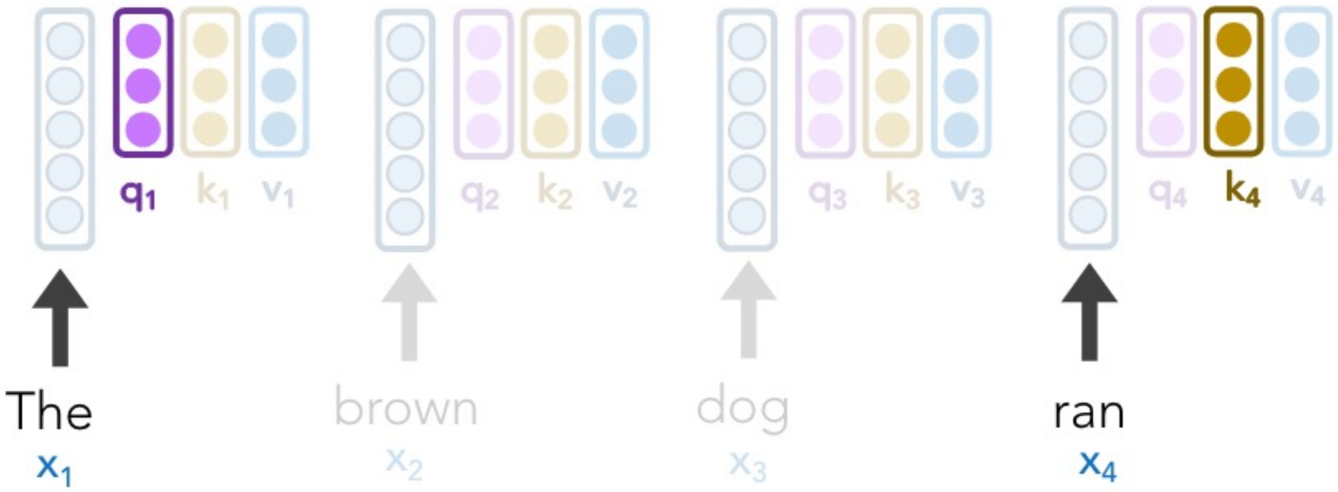
For a given token (word), let's calculate the scores s_1, s_2, s_3, s_4 which represent how much attention to pay to each respective "token" v_i

$s_4 = q_1 \cdot k_4 = 8$

$s_3 = q_1 \cdot k_3 = 16$

$s_2 = q_1 \cdot k_2 = 96$

$s_1 = q_1 \cdot k_1 = 112$



Self-Attention

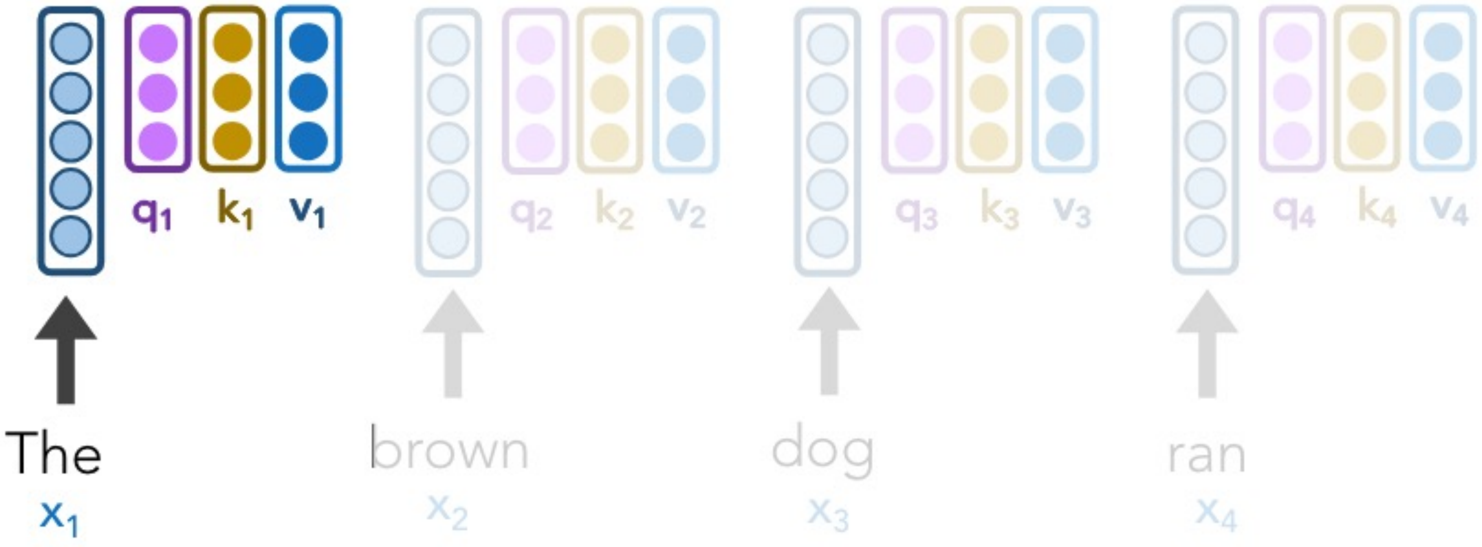
The scores s_1, s_2, s_3, s_4 don't sum upto 1. Let's divide it by $\sqrt{len(k_i)}$ and softmax it.

$s_4 = q_1 \cdot k_4 = 8$ $\quad a_4 = \sigma(s_4/8) = 0$

$s_3 = q_1 \cdot k_3 = 16$ $\quad a_3 = \sigma(s_3/8) = .01$

$s_2 = q_1 \cdot k_2 = 96$ $\quad a_2 = \sigma(s_2/8) = .12$

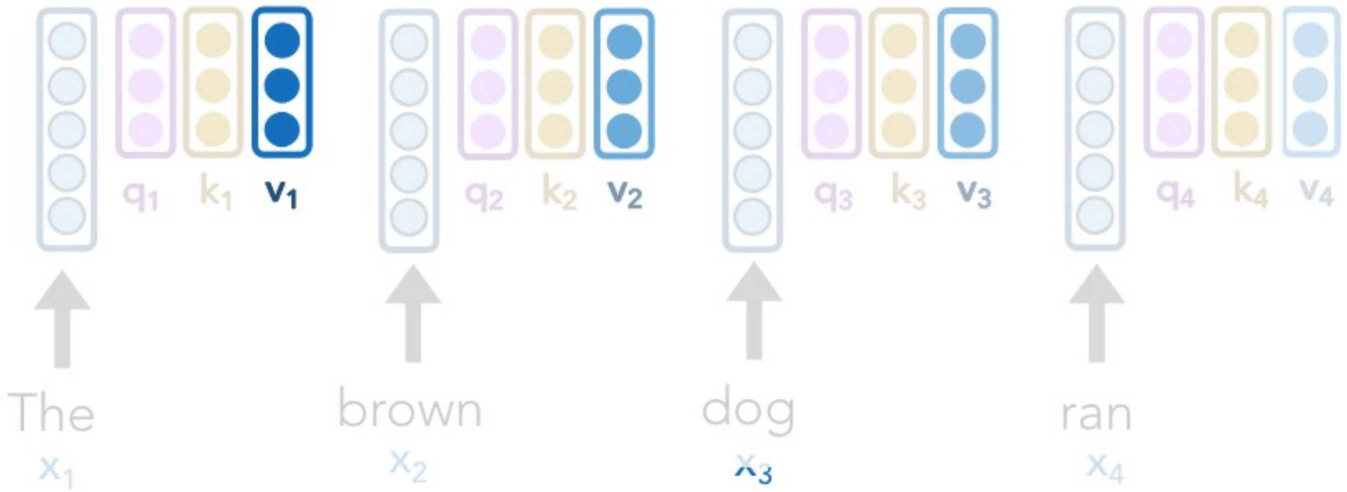
$s_1 = q_1 \cdot k_1 = 112$ $\quad a_1 = \sigma(s_1/8) = .87$



Self-Attention

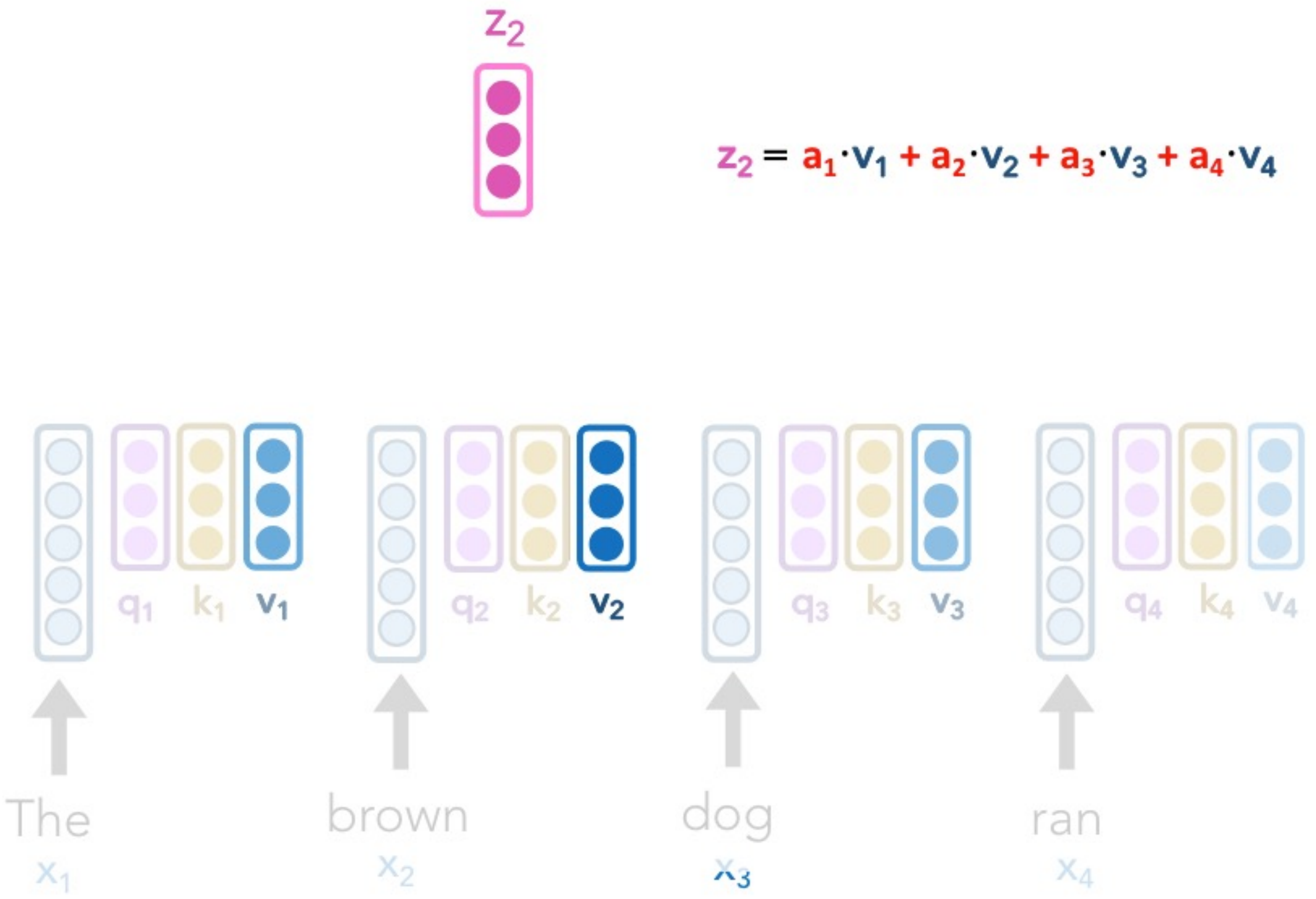
Weighing with v_i vectors and summing up will yield individual contextualized vector.

$$z_1 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$
$$= 0.87 \cdot v_1 + 0.12 \cdot v_2 + 0.01 \cdot v_3 + 0 \cdot v_4$$



Self-Attention

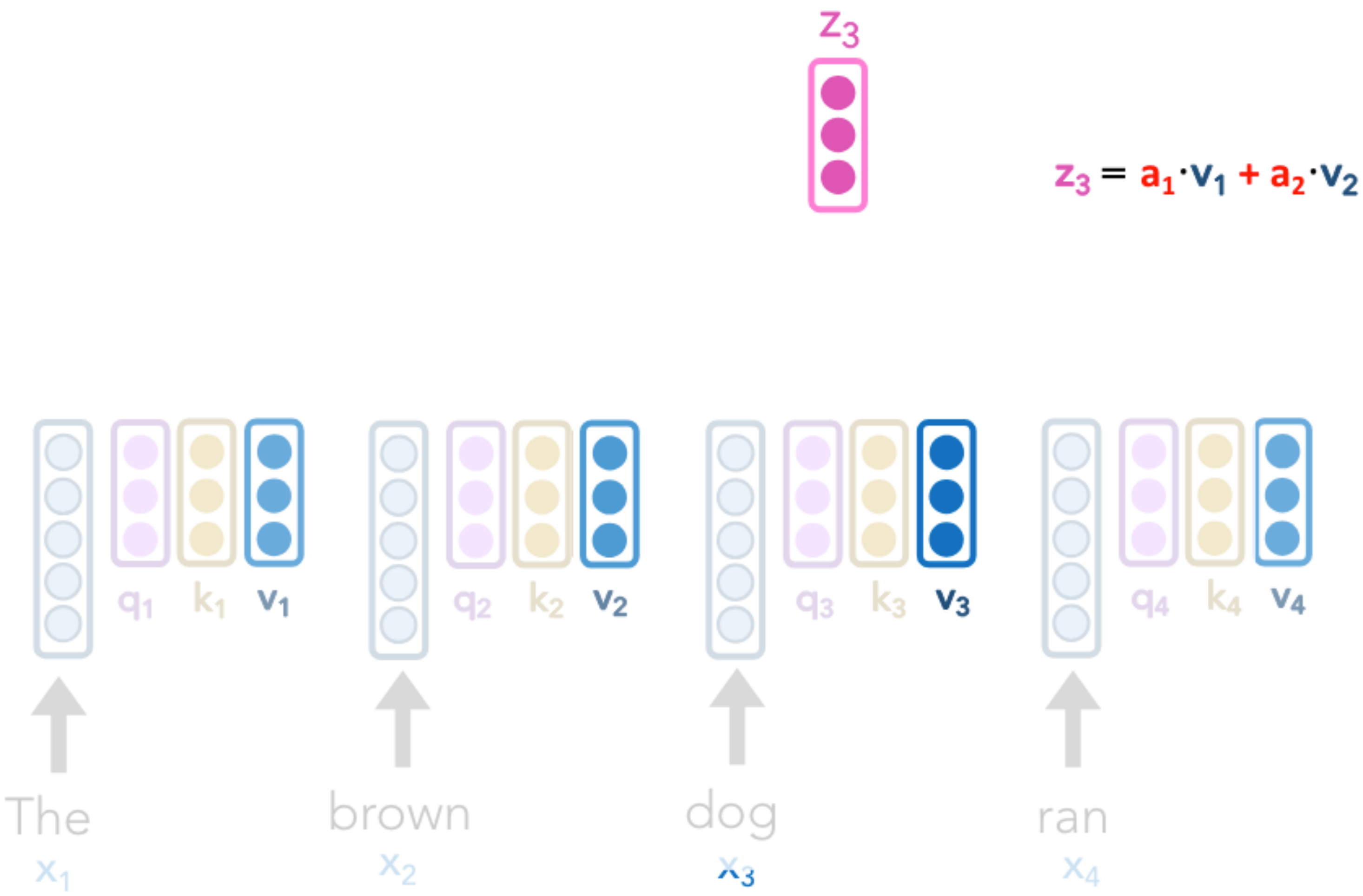
Repeating the procedure for all x_i will yield all the contextualized vectors



Self-Attention

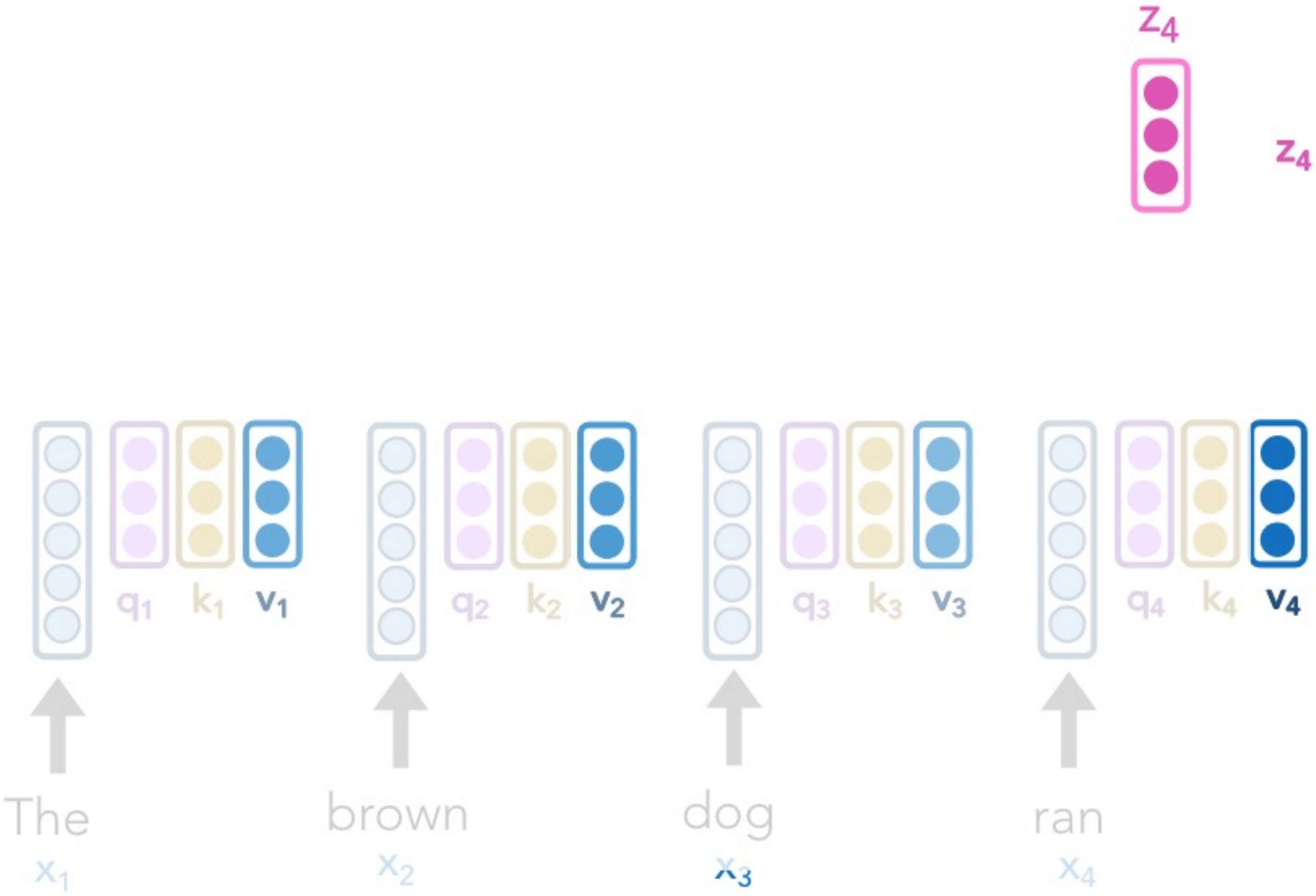
Repeating the procedure for all x_i will yield all the contextualized vectors

$$z_3 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$



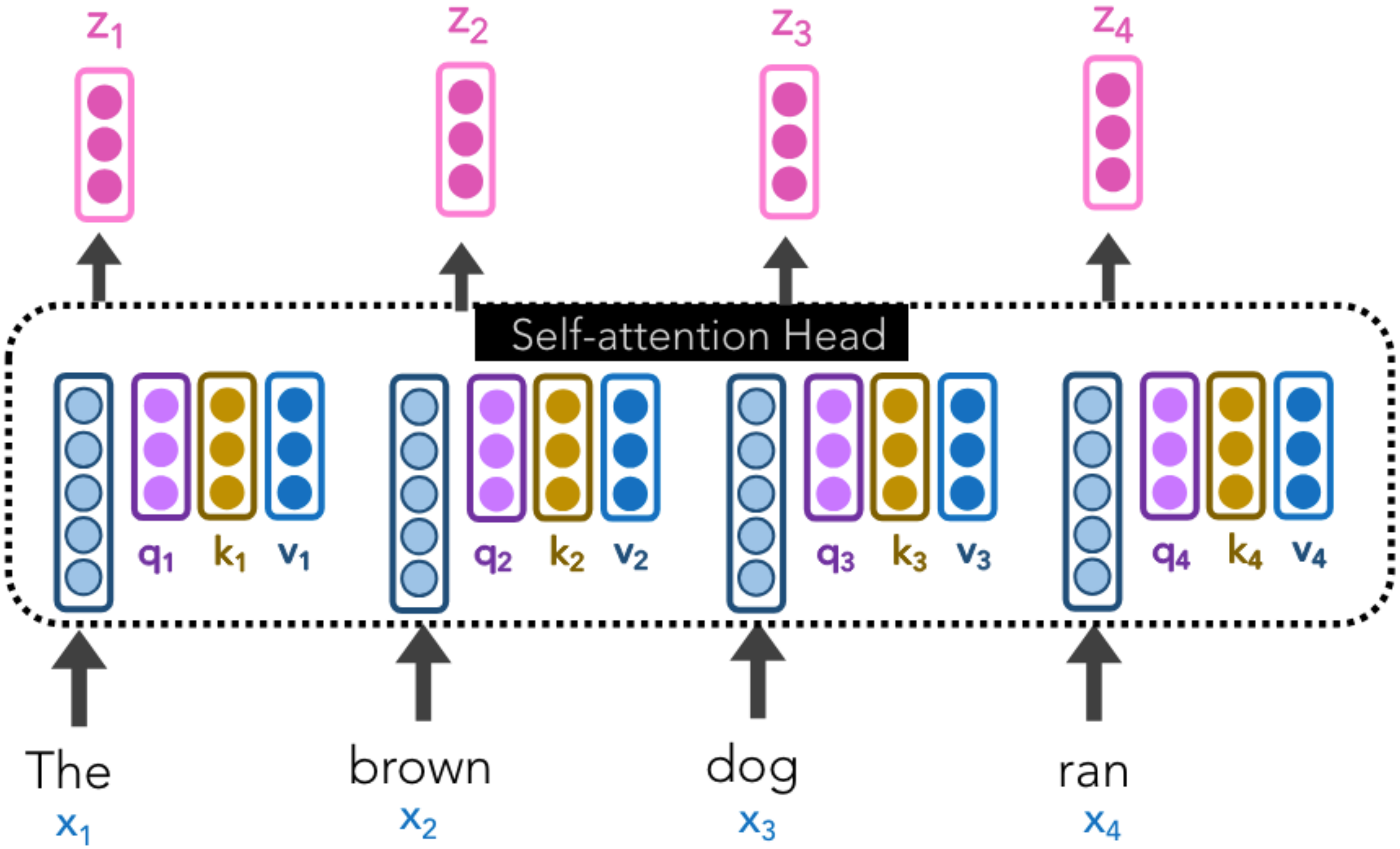
Self-Attention

Repeating the procedure for all x_i will yield all the contextualized vectors



$$z_4 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$

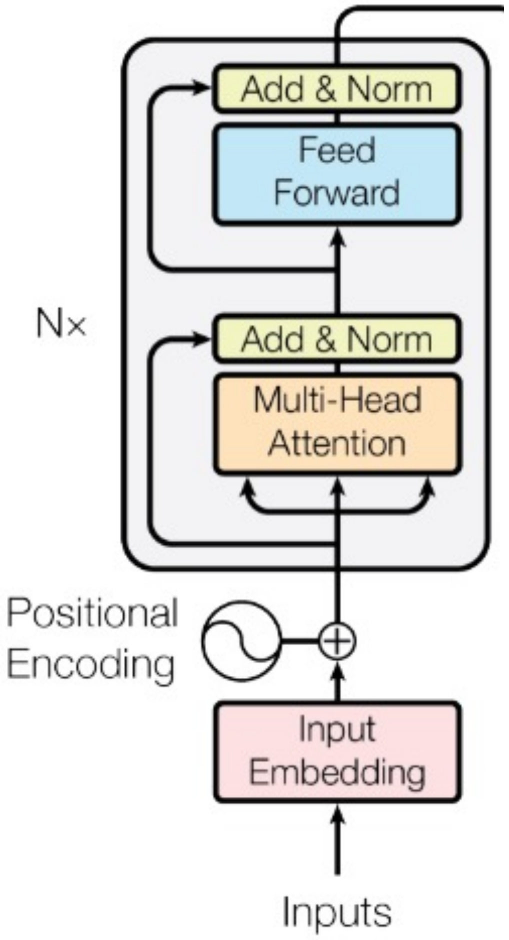
Self-Attention



Takeaway:

Self-Attention is powerful; allows us to create great, context-aware representations

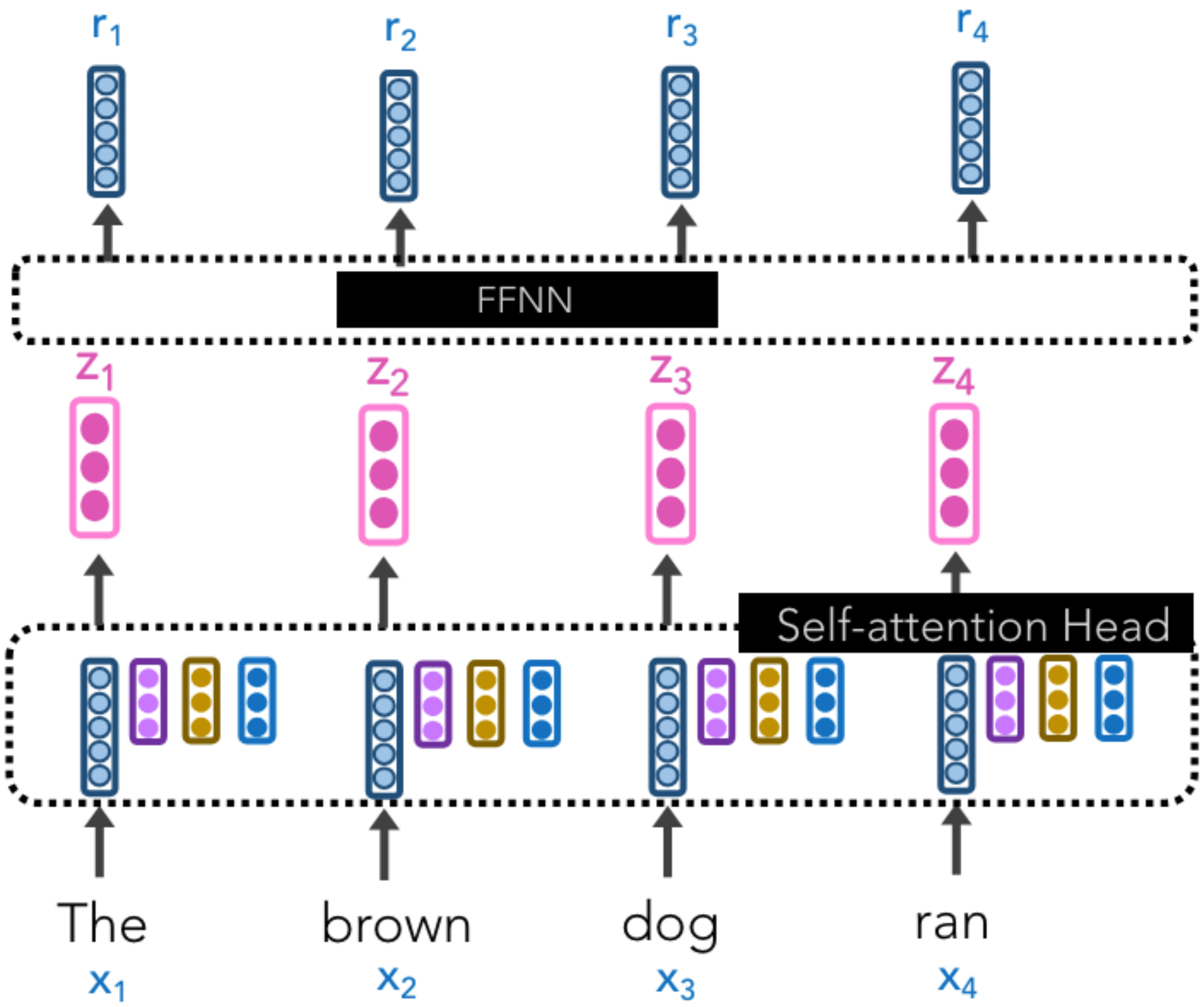
Transformer Encoder



Comprises of:

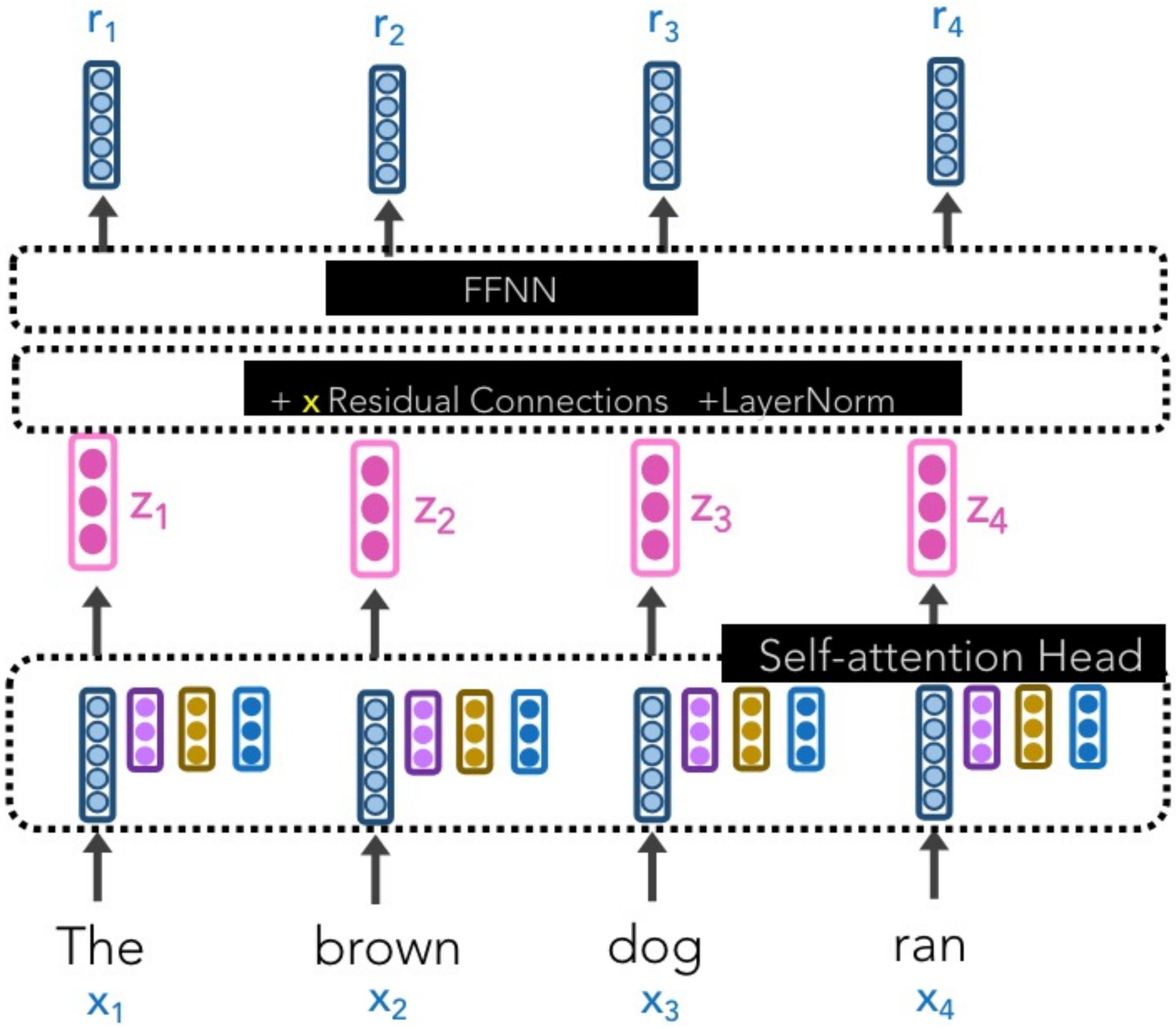
1. Multi-head attention (Parallelized self-attention)
2. Feed forward layers (FFNN) – MLP
3. Layer Norm
4. Residual Connections
5. Positional Encoding

Transformer Encoder – Feed Forward Layers



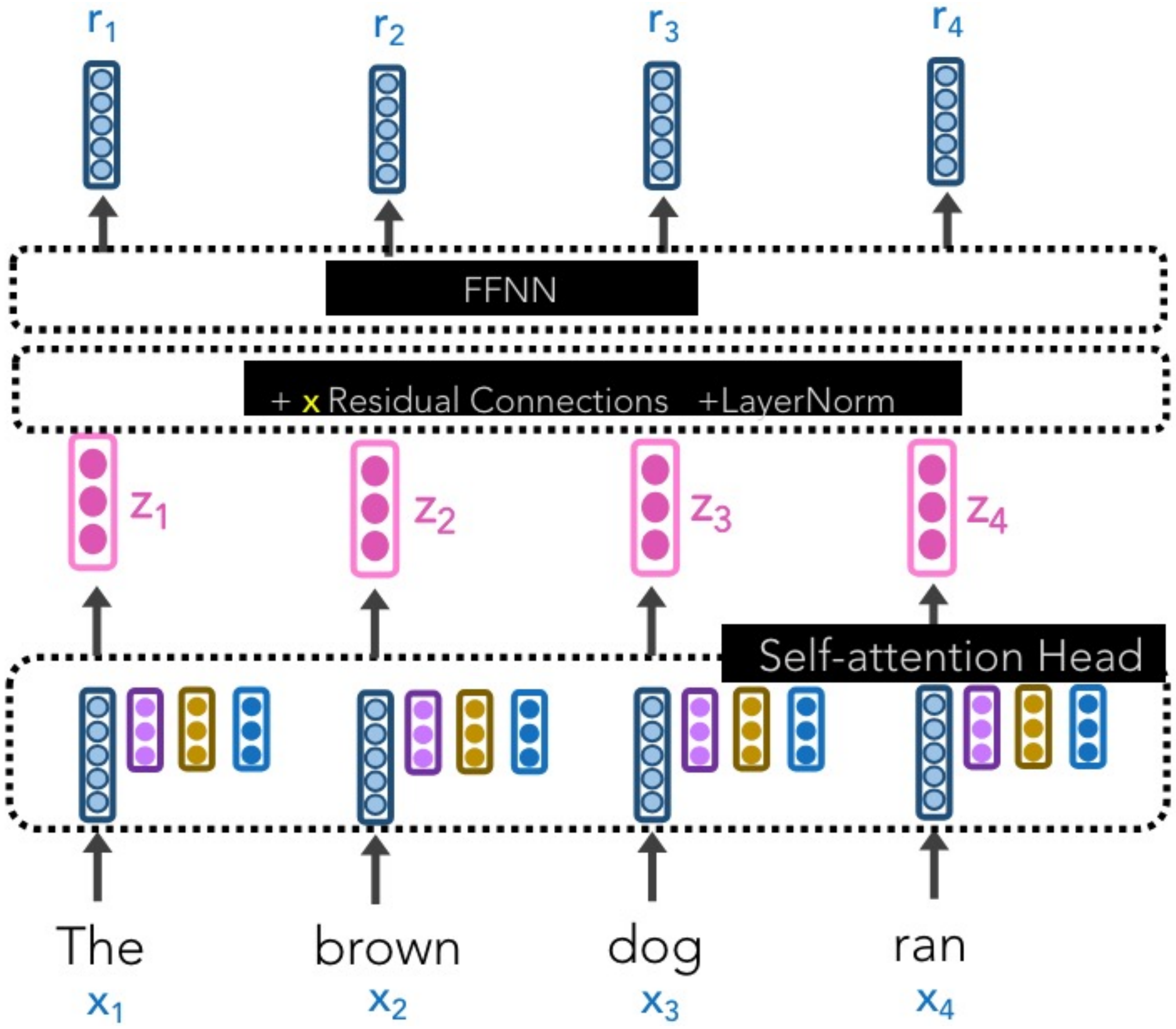
Feed-forward layers are basically standard MLPs which take the output from the attention head as the input and output relevant vectors

Transformer Encoder – Residual Connection



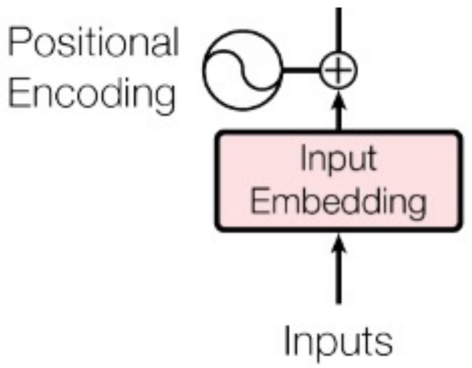
Residual connection ensures that the information is passed through layers and helps mitigate the problem of gradient vanishing which occurs during understanding long context

Transformer Encoder – Layer Norm



Layer Norm stabilizes the network and allow proper gradient flow

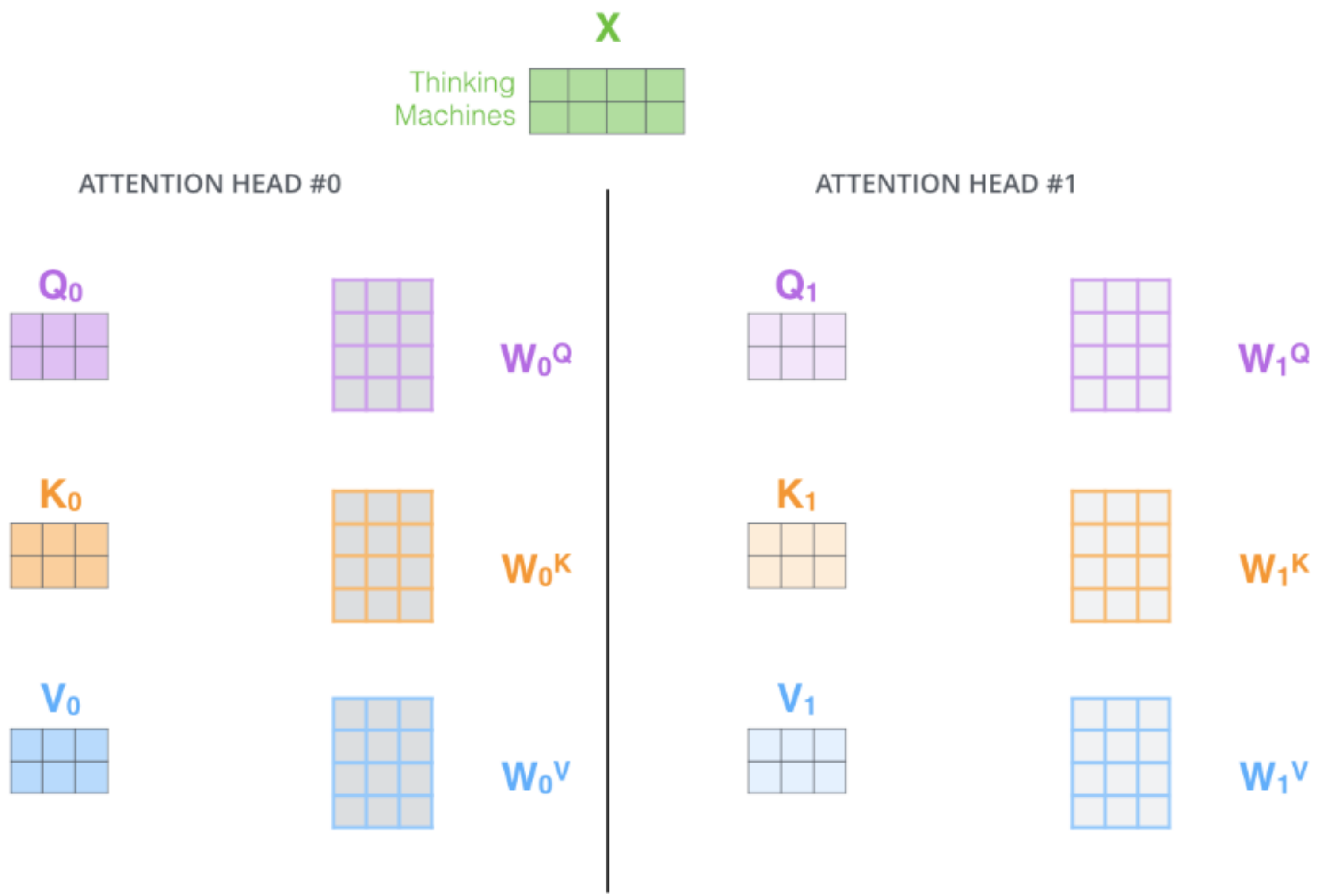
Transformer Encoder – Positional Encoding



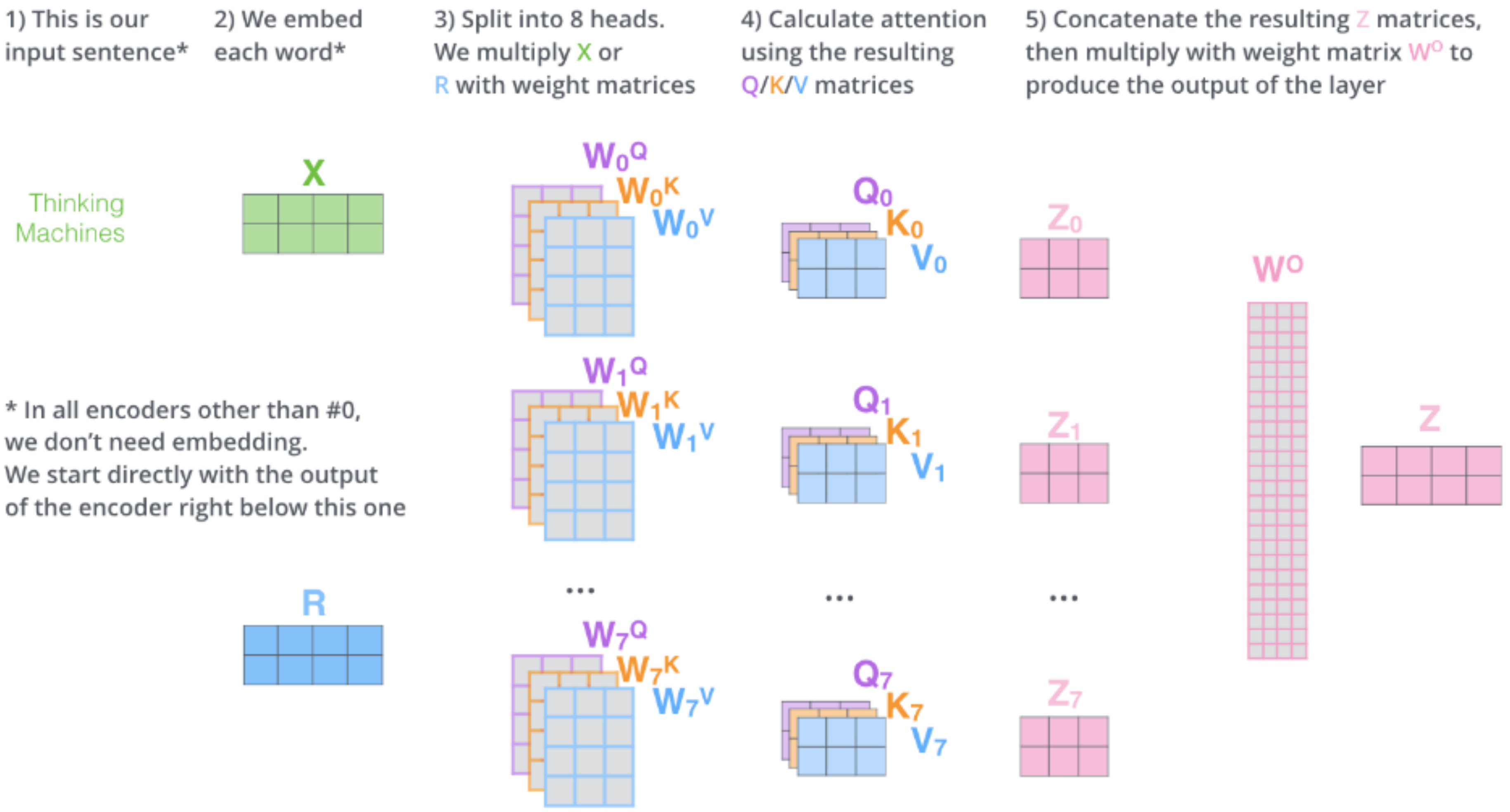
Since the transformer model processes the sequence parallelly, positional information has to be encoded in the input tokens to inject information about their relative or absolute position in the sequence

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

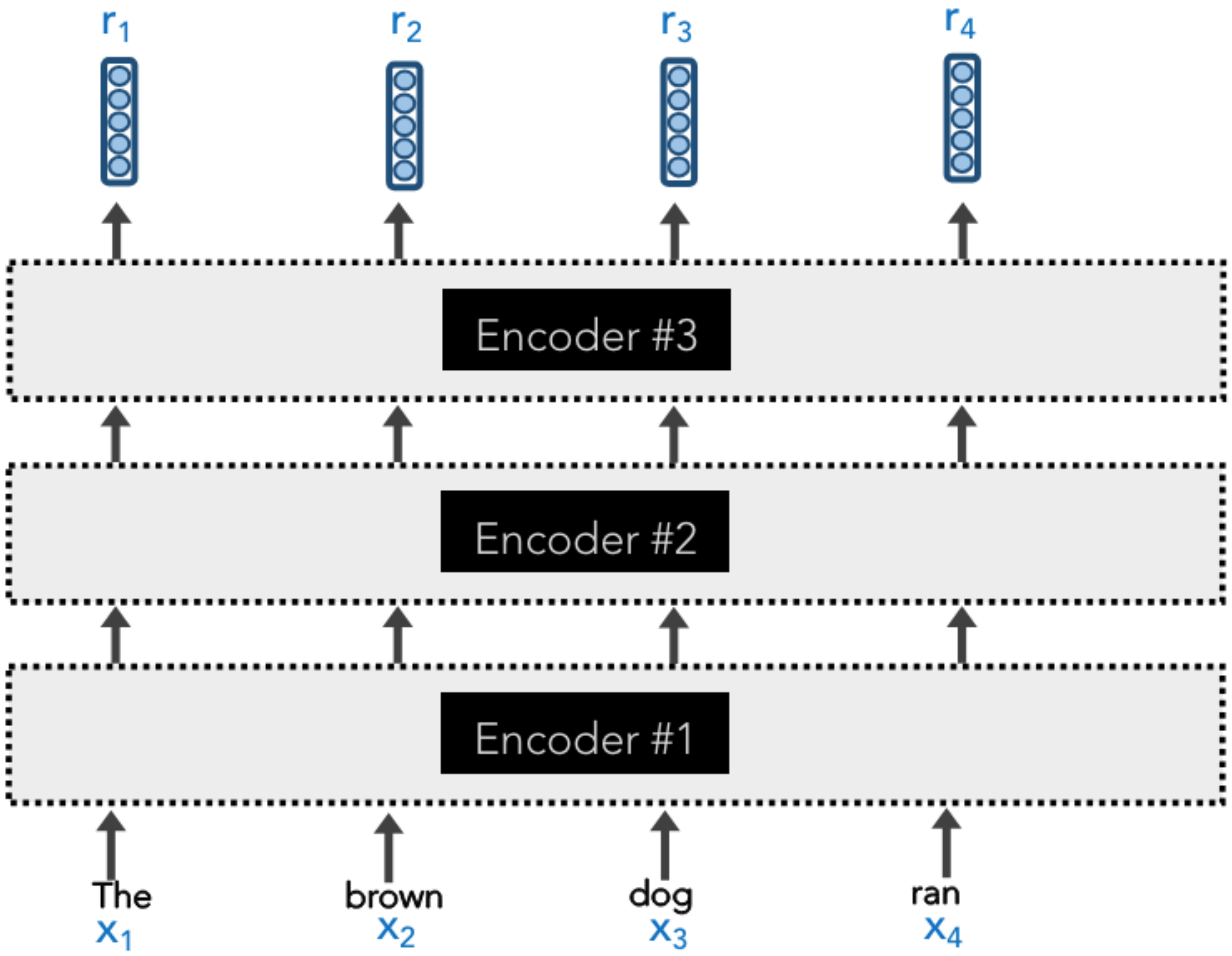
Transformer Encoder – Multi-head attention



Transformer Encoder – Multi-head attention

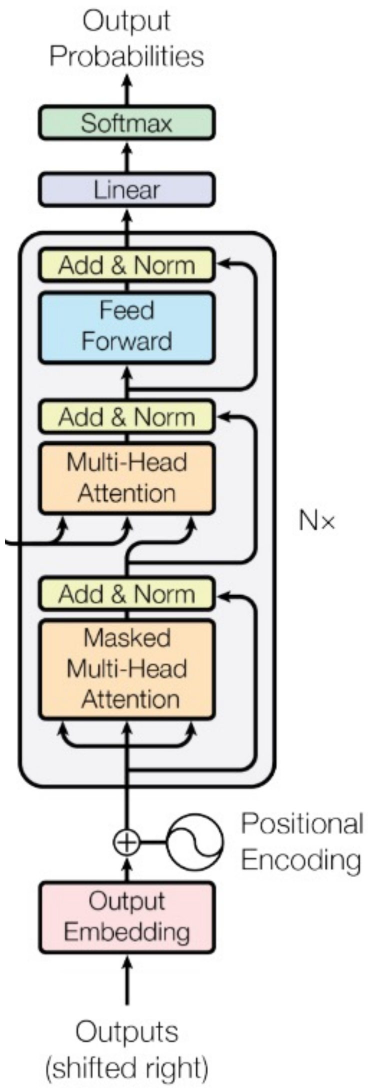


Transformer Encoder



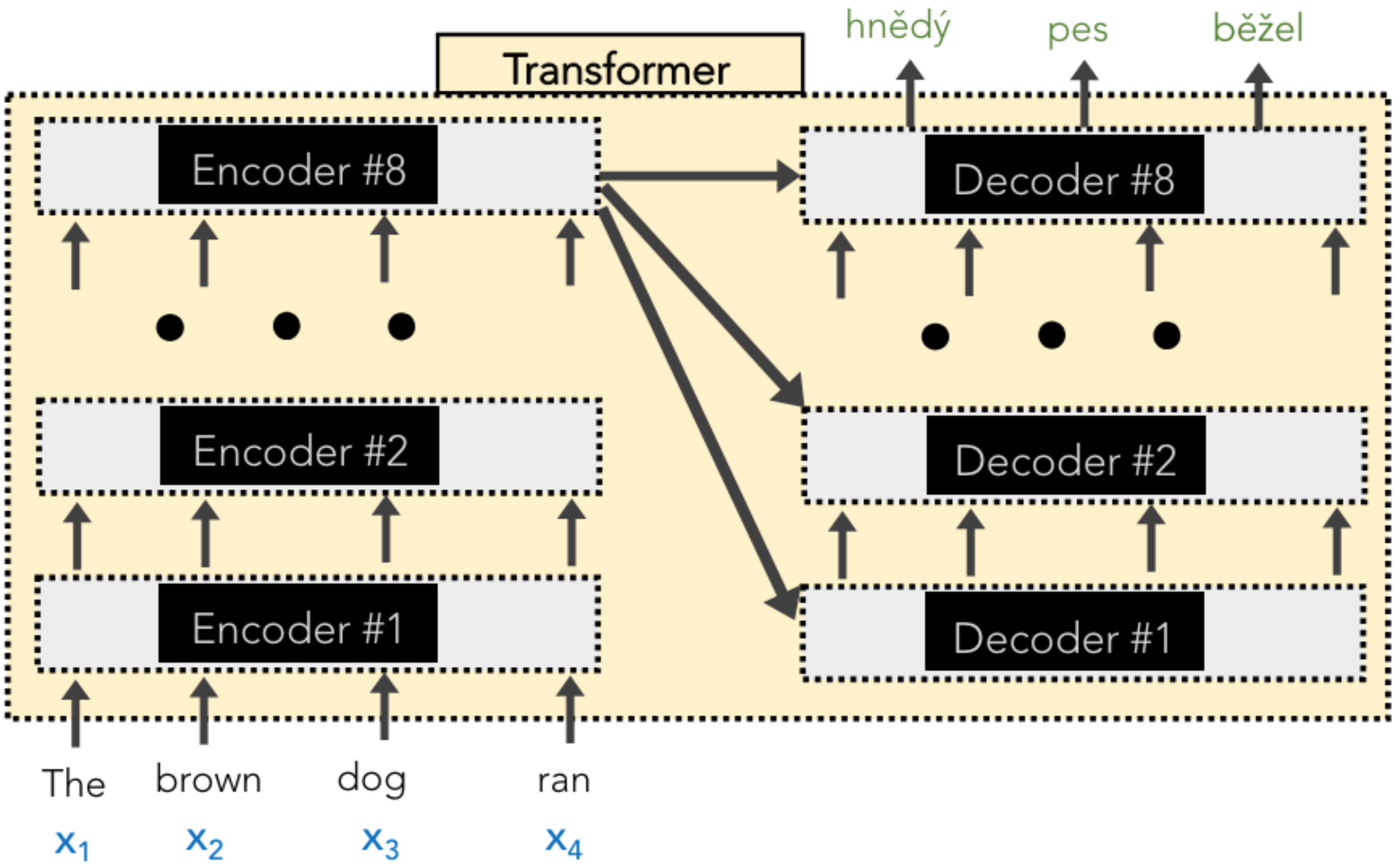
Generally, a transformer network will consist of several encoder layers stacked upon each other yielding more complex non-linear representation of the input

Transformer Decoder



- Similar to the transformer encoder
- Output from the transformer encoder acts as the key and value to the decoder

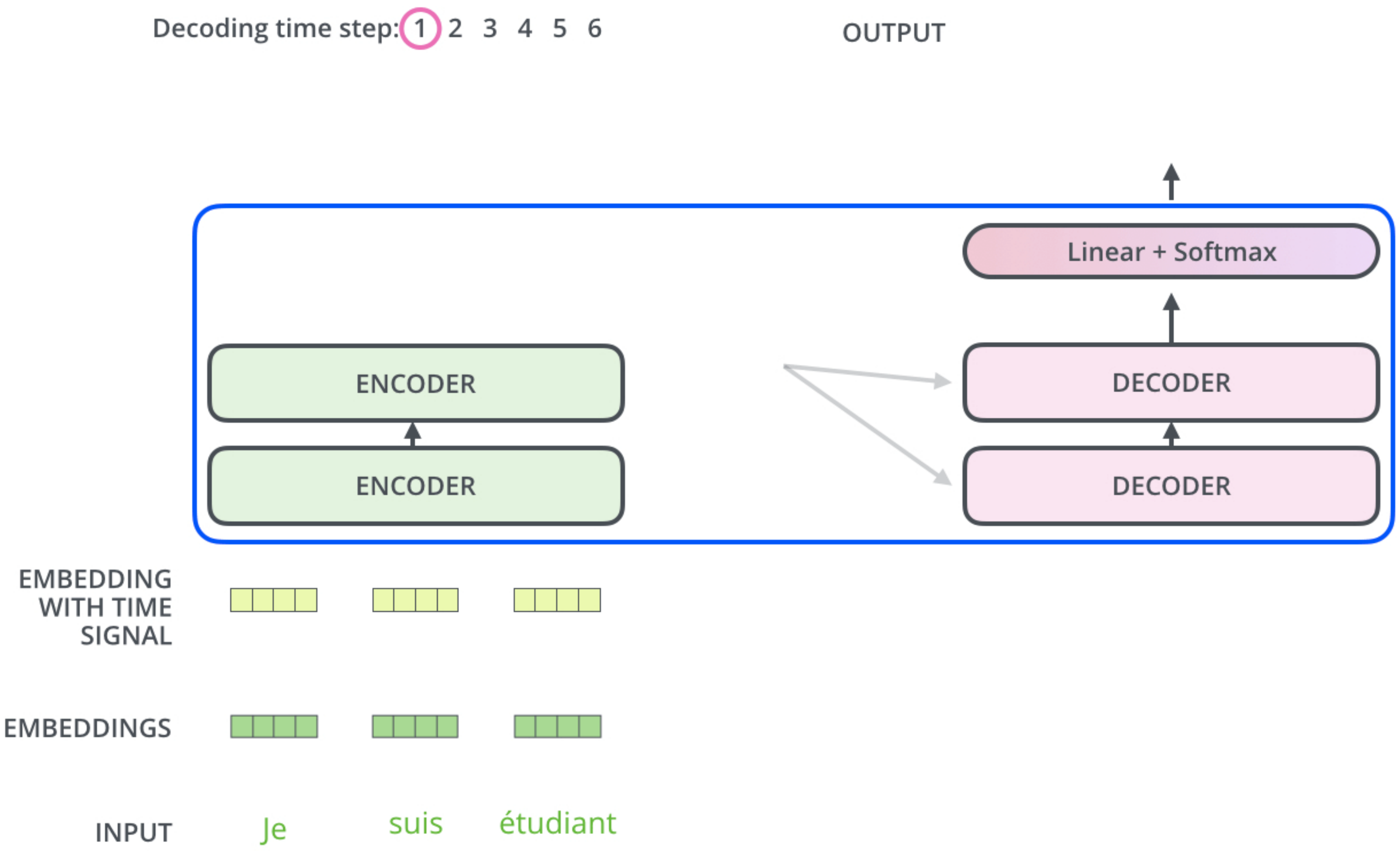
Transformer Decoder



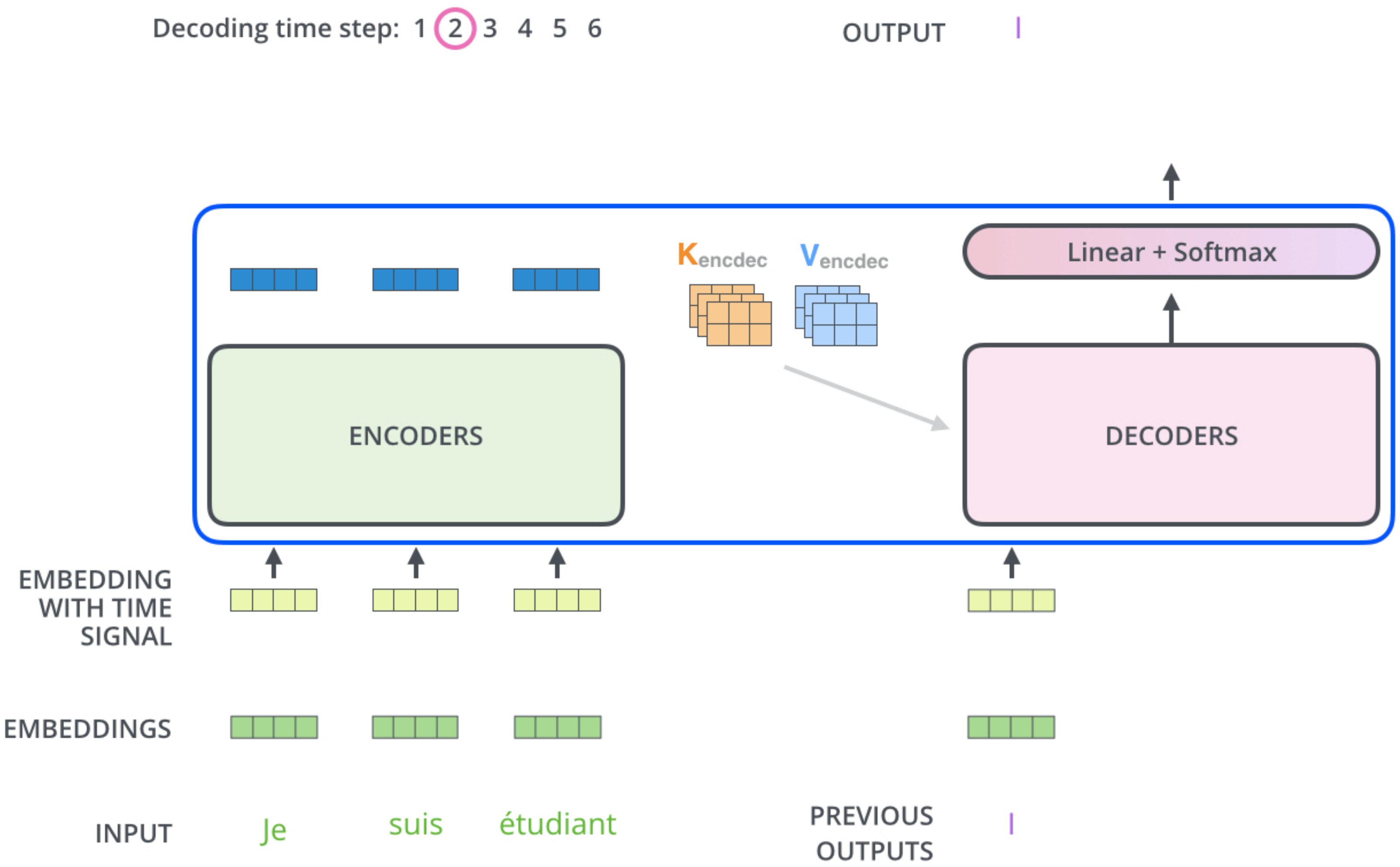
Transformer encoders produce contextualized embeddings of each token

Transformer decoders generate new sequences of text

Transformer Decoder



Transformer Decoder

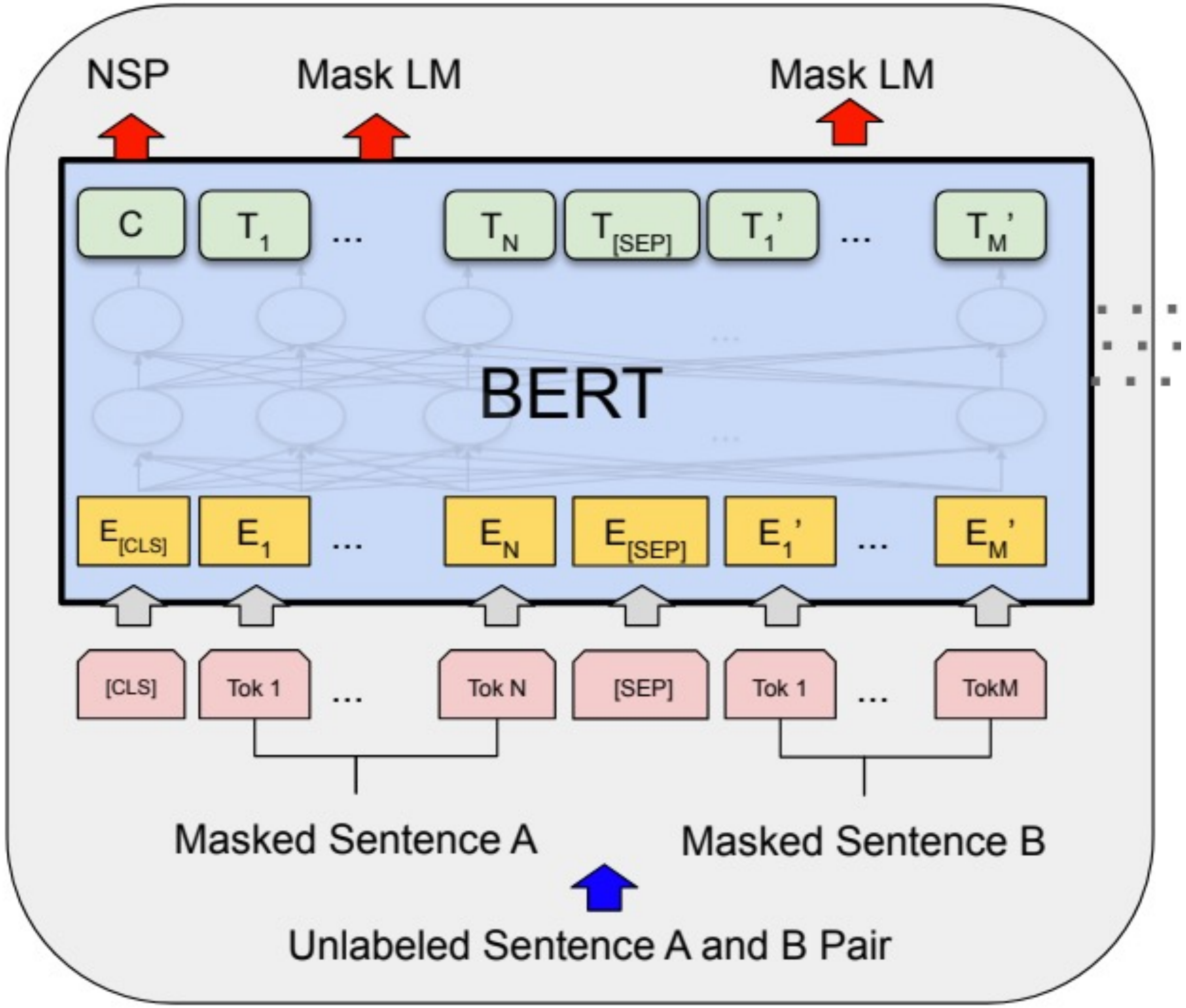


Transformers as general-purpose language models

- Textual data is everywhere online for free: web pages (news articles, Wikipedia, blogs)
- However, labeling large-scale data for specific tasks is laborious and expensive
- What can be done?

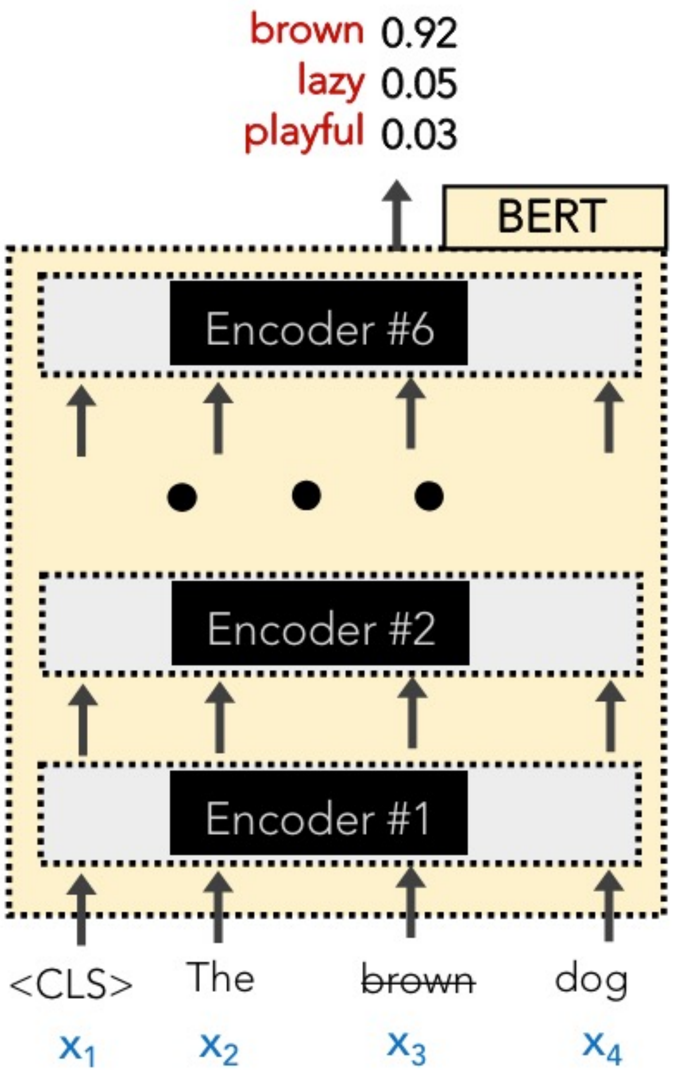
Train a general-purpose language model and adapt it to specific tasks as required

Bidirectional Encoder Representations from Transformers (BERT)



- A model with only transformer encoders
- A language model that builds rich representation
- Pre-trained with Masked Language modeling and Next-sentence prediction tasks

Bidirectional Encoder Representations from Transformers (BERT)



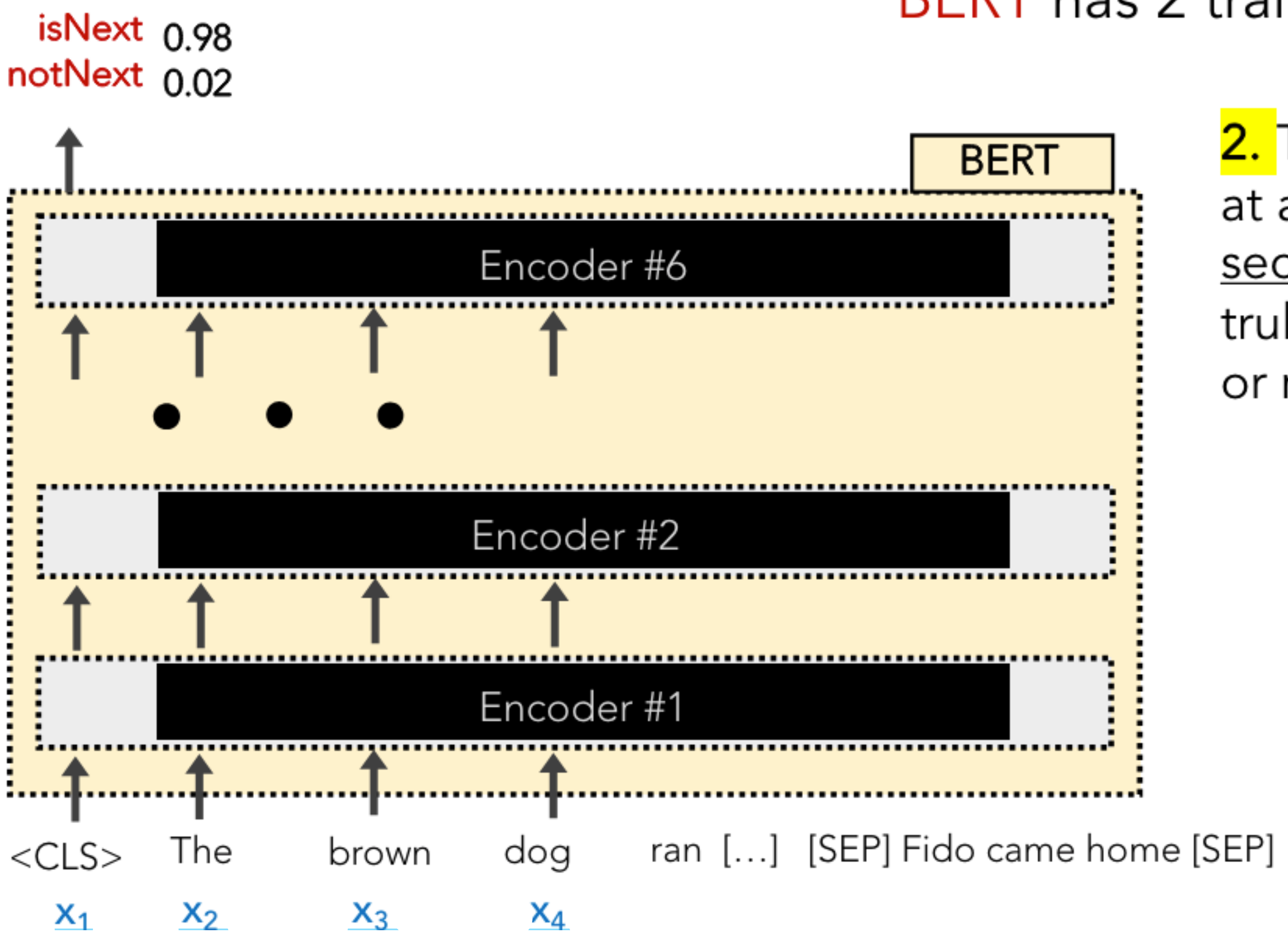
BERT has 2 training objectives:

1. Predict the **Masked word** (a la CBOW)

15% of all input words are randomly masked.

- 80% become [MASK]
- 10% become revert back
- 10% become are deliberately corrupted as wrong words

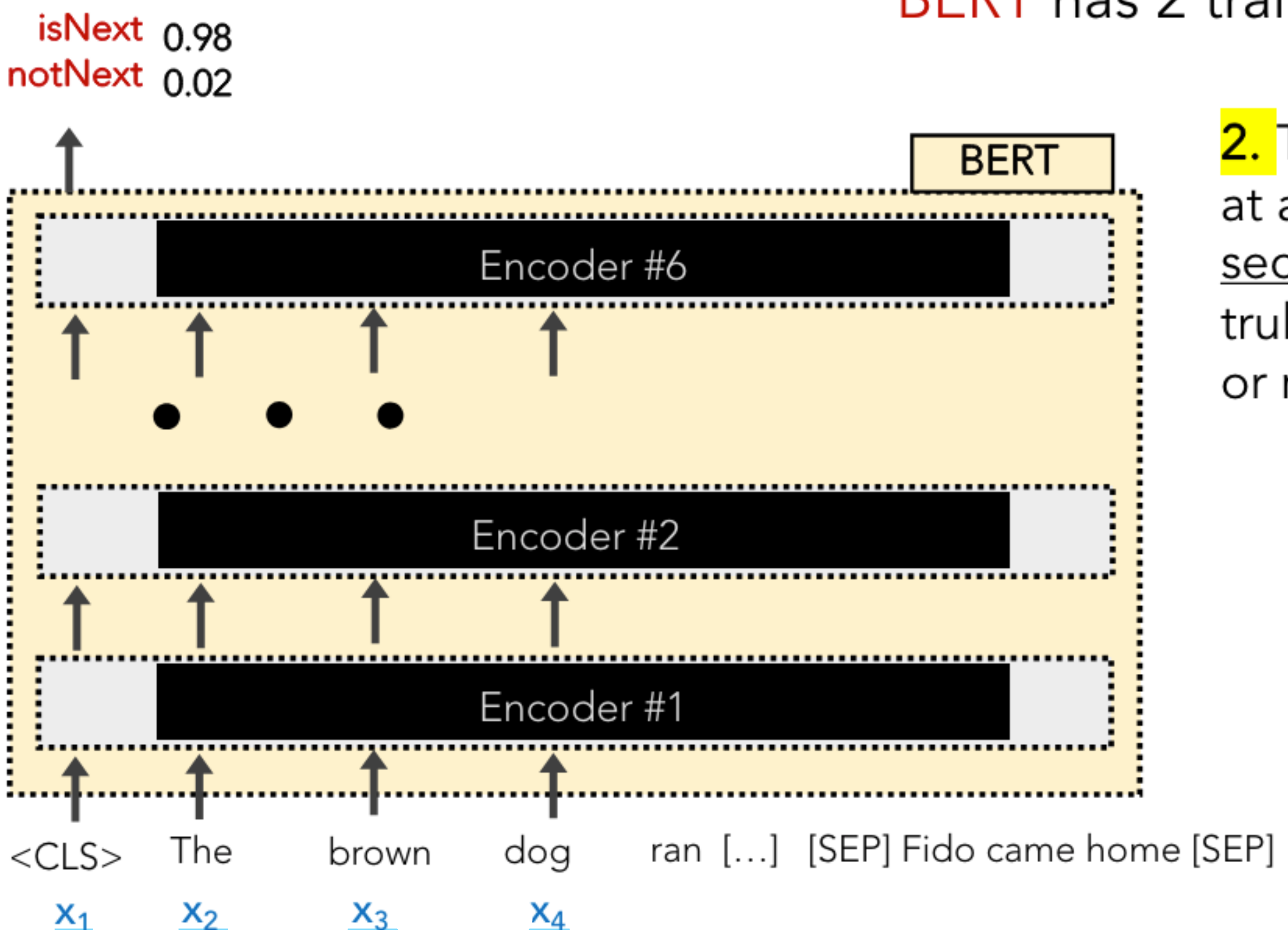
Bidirectional Encoder Representations from Transformers (BERT)



BERT has 2 training objectives:

2. Two sentences are fed in at a time. Predict the if the second sentence of input truly follows the first one or not.

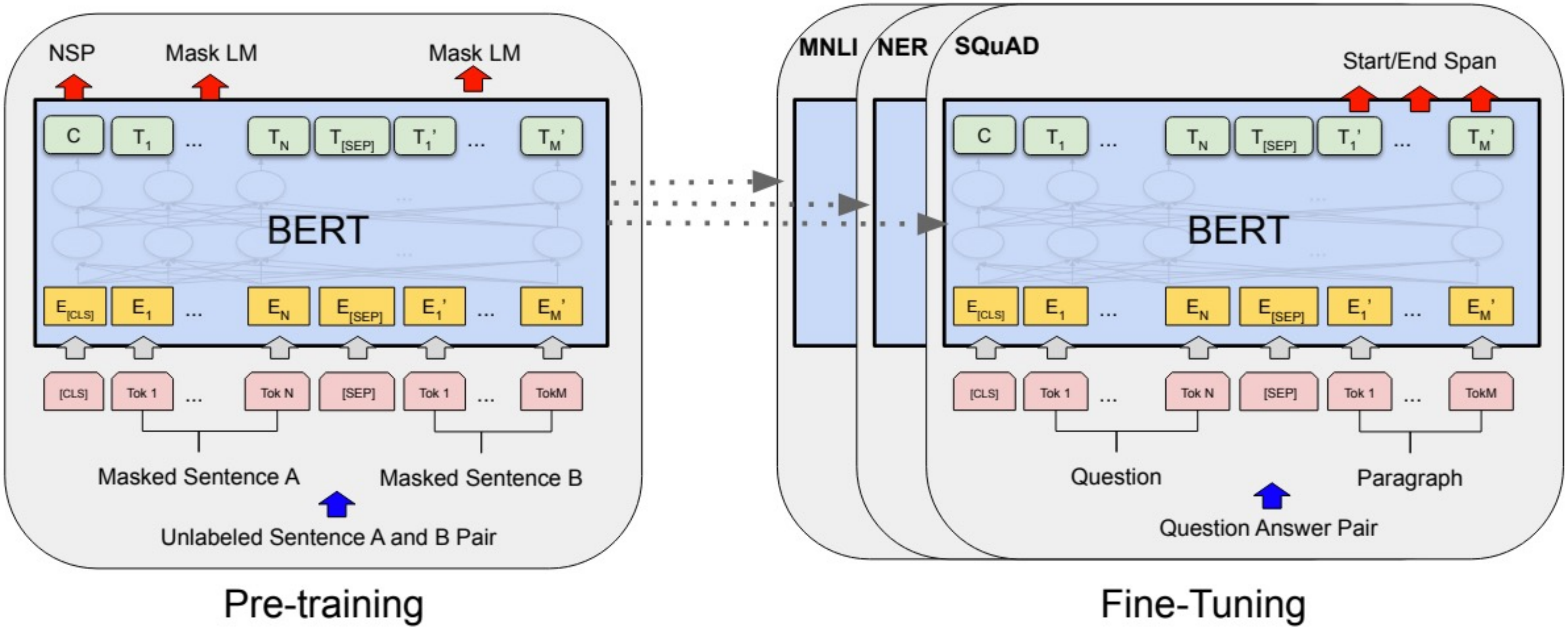
Bidirectional Encoder Representations from Transformers (BERT)



BERT has 2 training objectives:

2. Two sentences are fed in at a time. Predict the if the second sentence of input truly follows the first one or not.

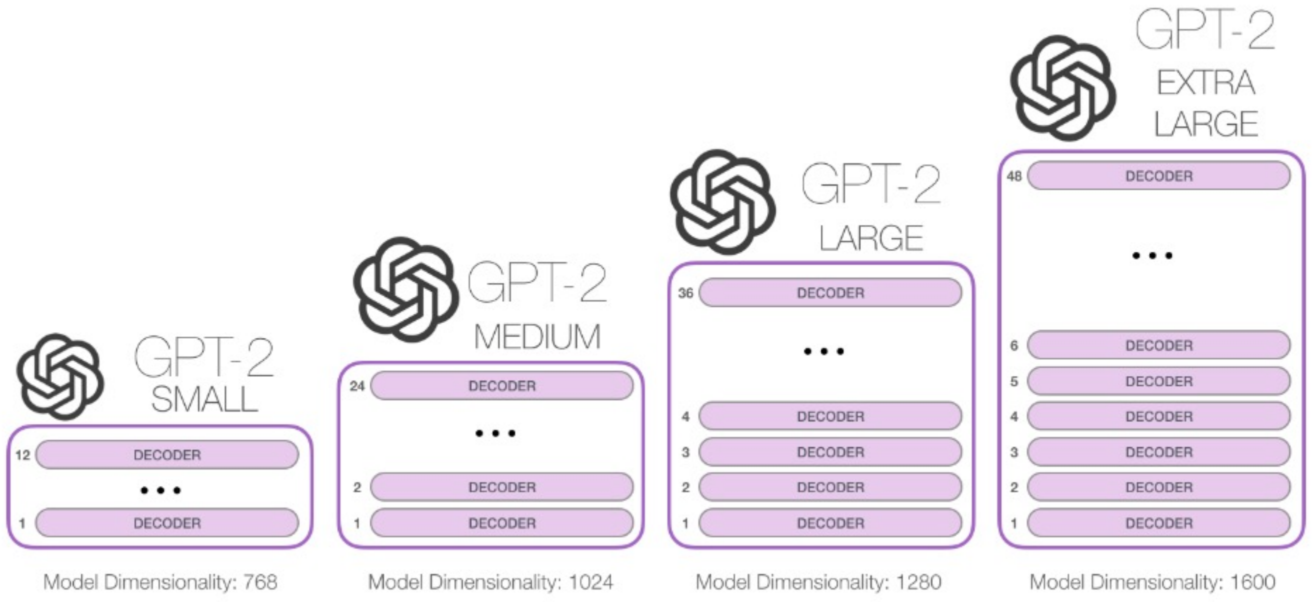
Bidirectional Encoder Representations from Transformers (BERT)



Generative Pre-trained Transformer (GPT)

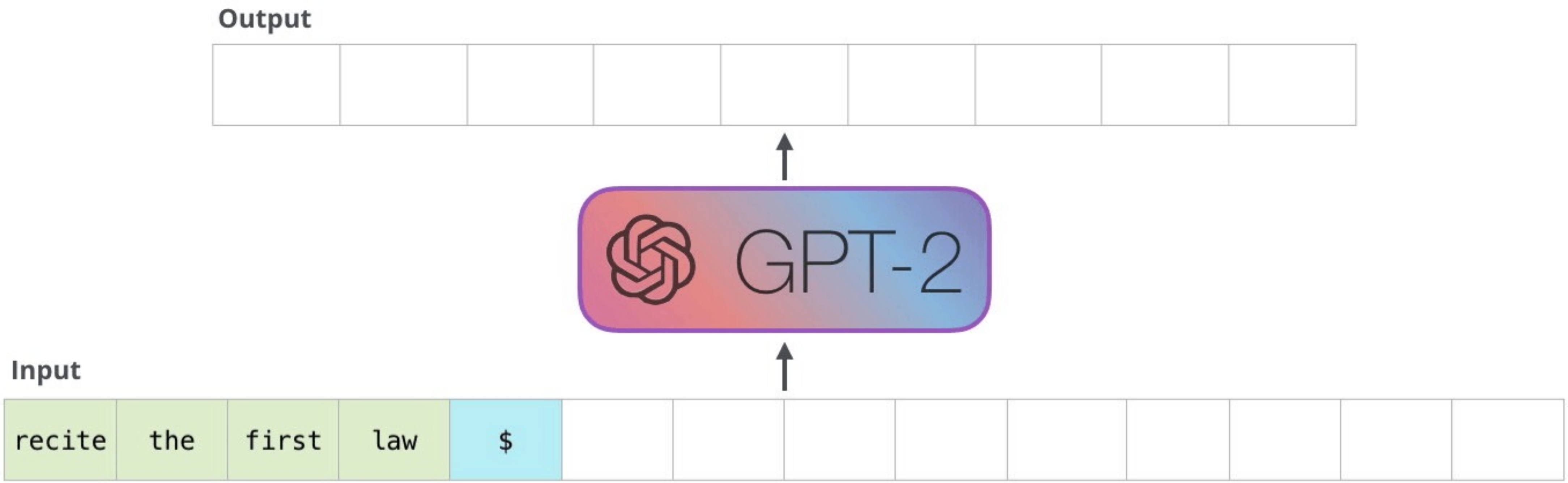
BERT doesn't generate new sequences as it only comprises of transformer encoders

GPT contains only transformer decoders, thus can generate new output sequences



Generative Pre-trained Transformer (GPT)

GPT learns by masking the future token of the given input sentence.



Let’s build a GPT !!