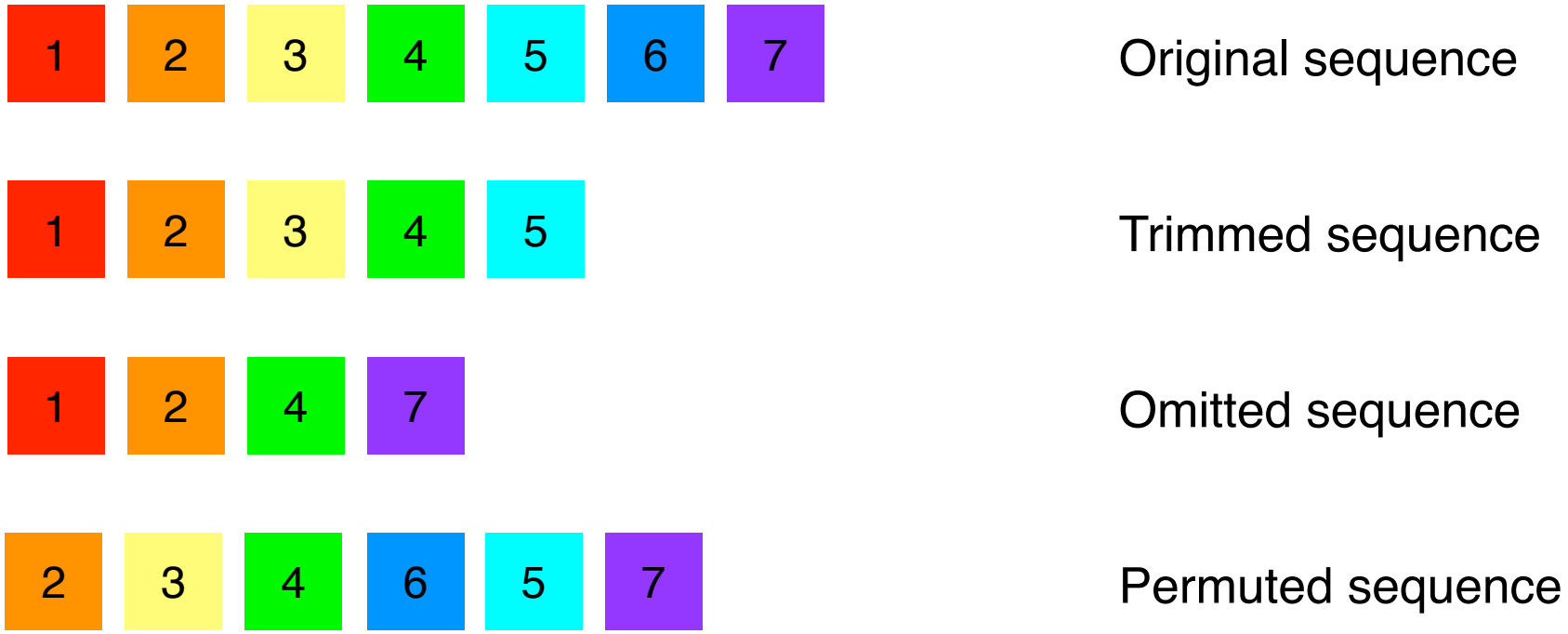
Deep Learning Basics

Lecture 8: Transformer

최성준 (고려대학교 인공지능학과)

Sequential Model

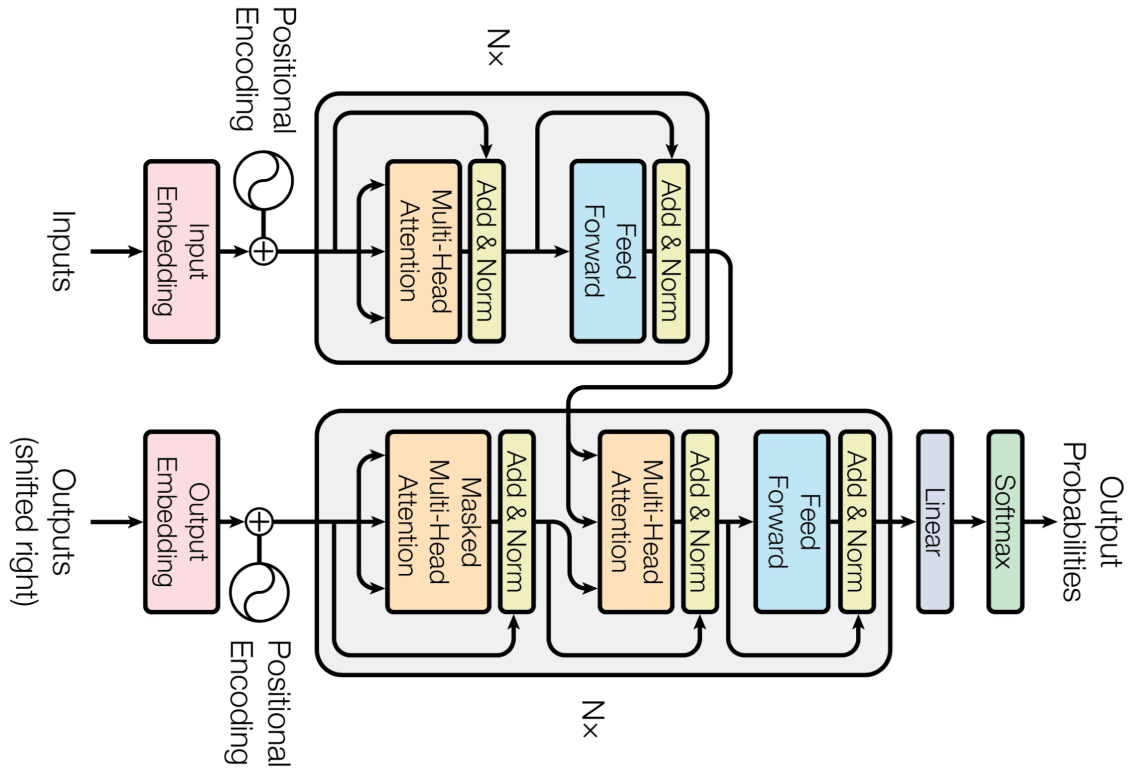
- What makes sequential modeling a hard problem to handle?



Transformer

Attention is All You Need, NIPS, 2017

Transformer



- **Transformer** is the first sequence transduction model based entirely on attention.

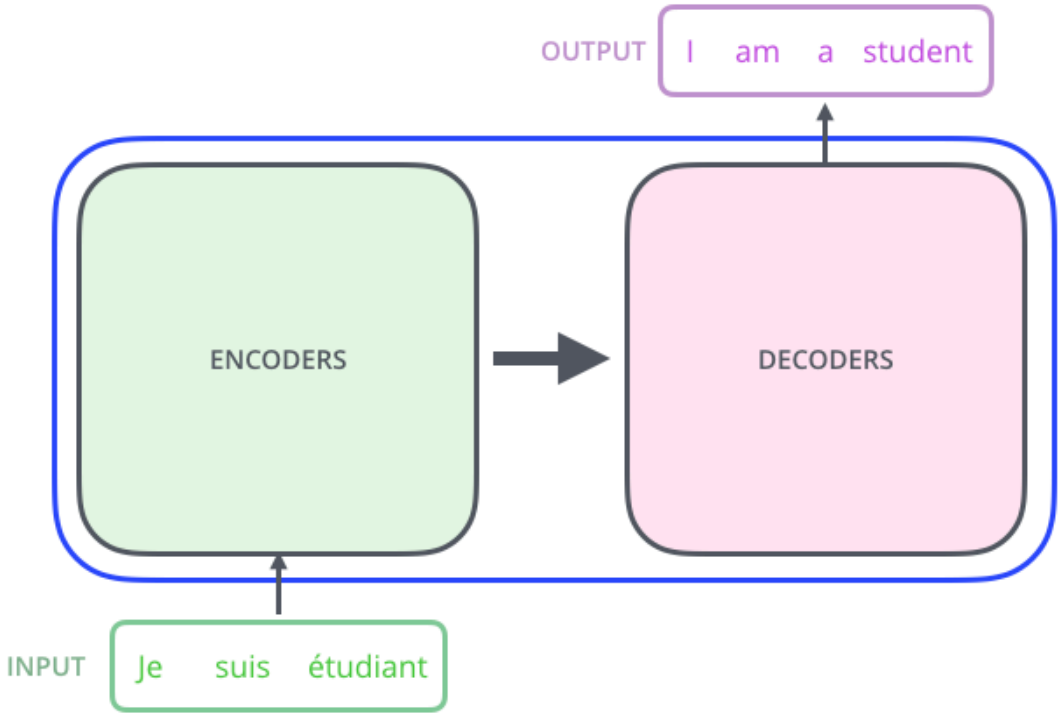
https://arxiv.org/abs/1706.03762

Transformer

- From a bird's-eye view, this is what the **Transformer** does for machine translation tasks.

Jay Alammar
Visualizing machine learning one concept at a time.
@JayAlammar on Twitter. YouTube Channel
http://jalammar.github.io/illustrated-transformer/

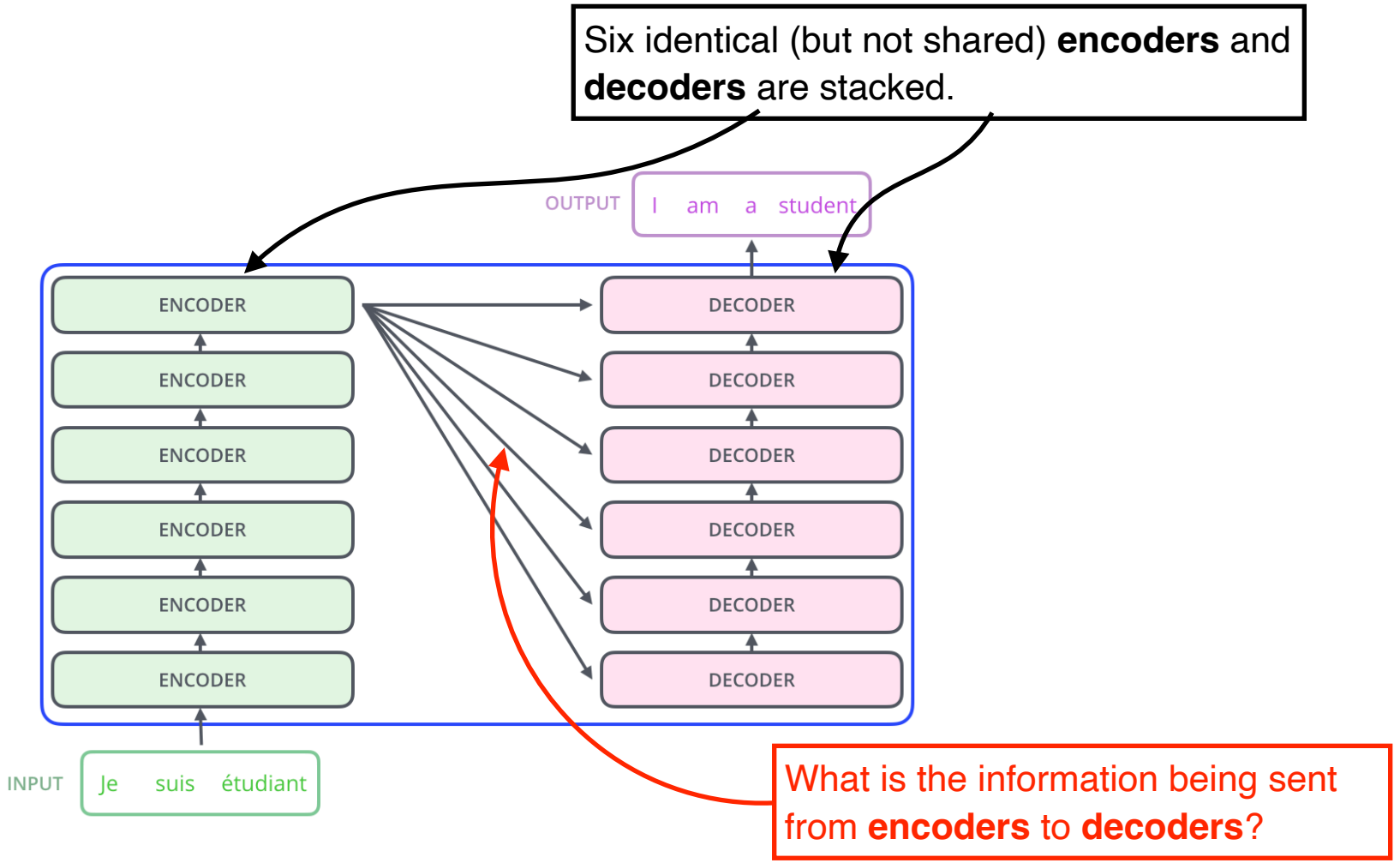
Transformer



- If we glide down a little bit, this is what the **Transformer** does.

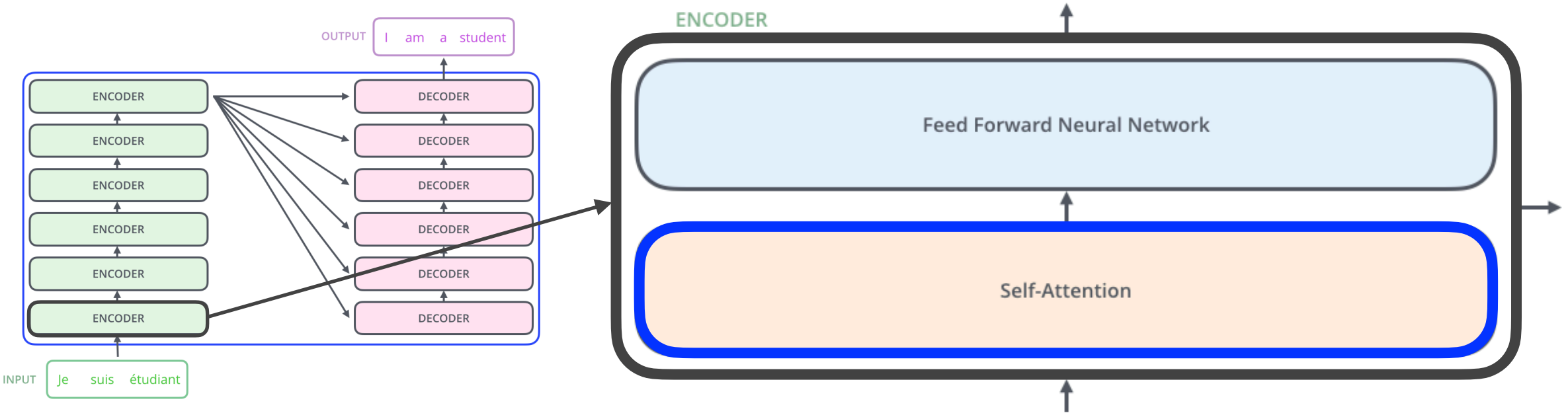
http://jalammar.github.io/illustrated-transformer/

Transformer



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

Transformer



- The **Self-Attention** in both encoder and decoder is the cornerstone of Transformer.

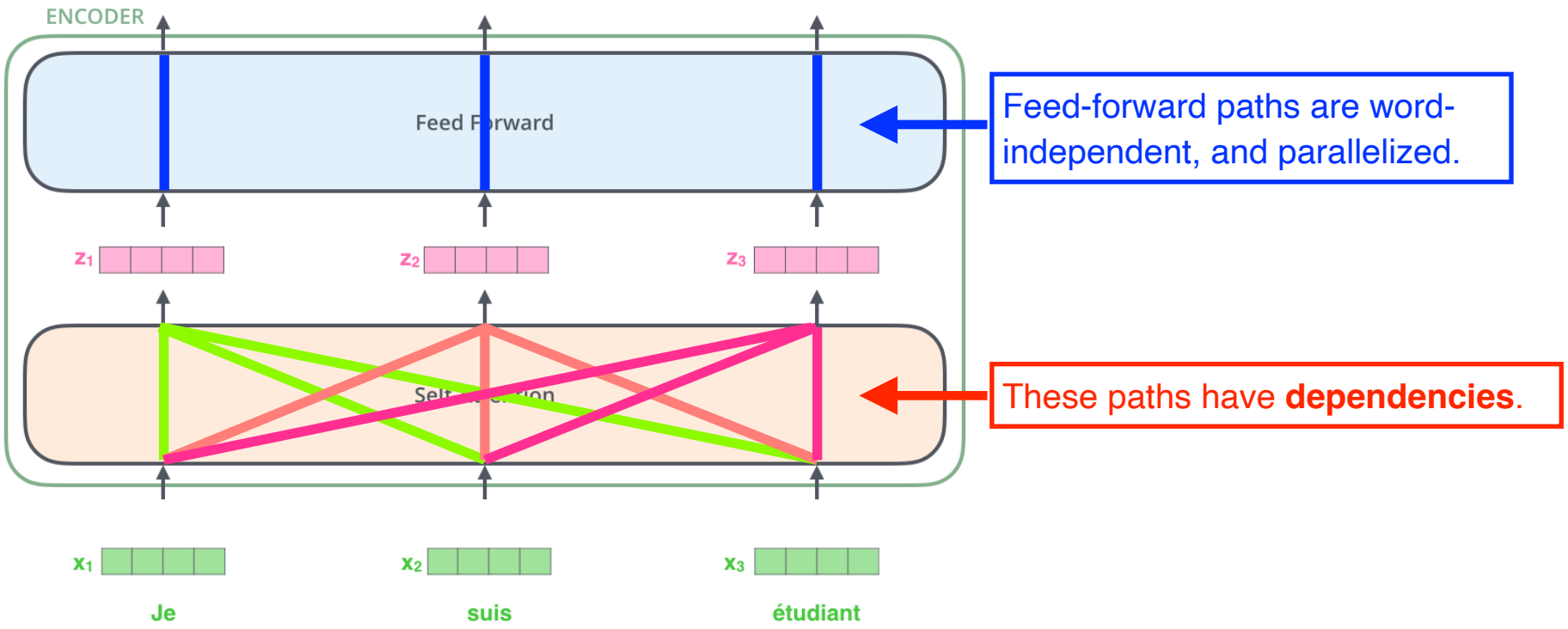
http://jalammar.github.io/illustrated-transformer/

Transformer

- First, we represent each word with some embedding vectors.

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

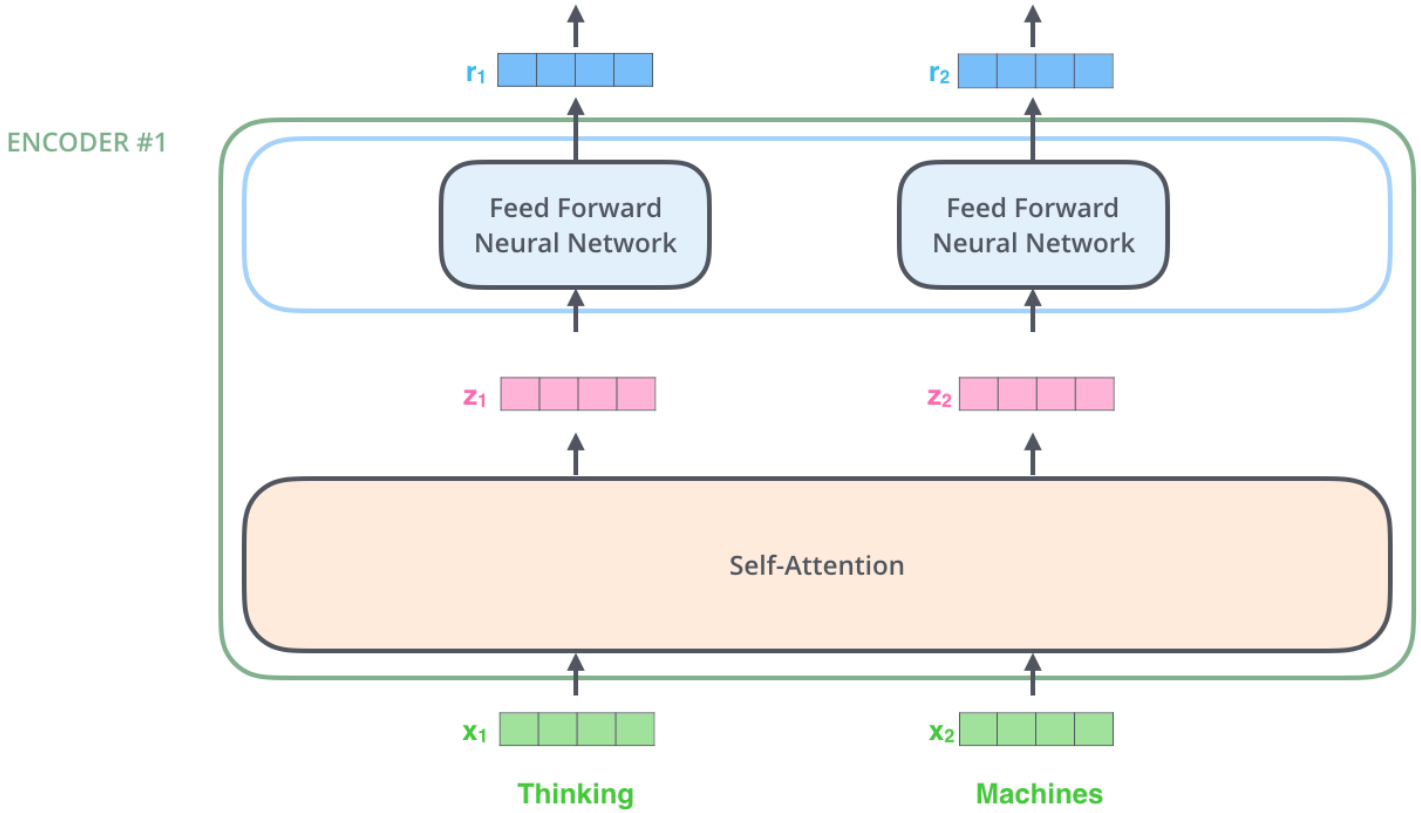
Transformer



- Then, Transformer encodes each word to feature vectors with **Self-Attention**.

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

Transformer

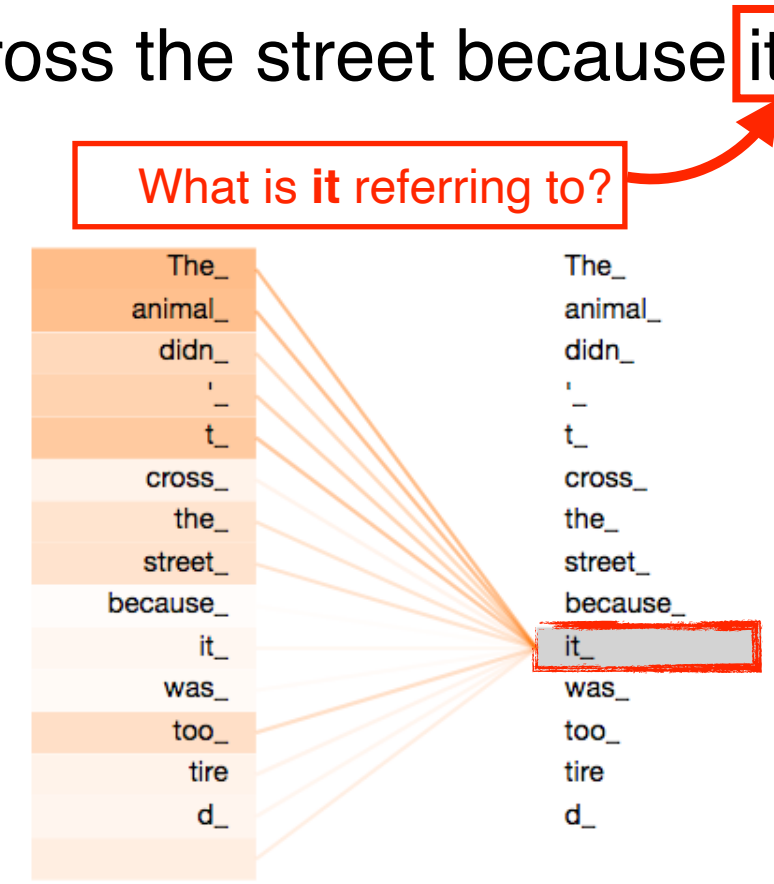


- Suppose we are encoding two words:
  - **Thinking** and **Machines**.

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

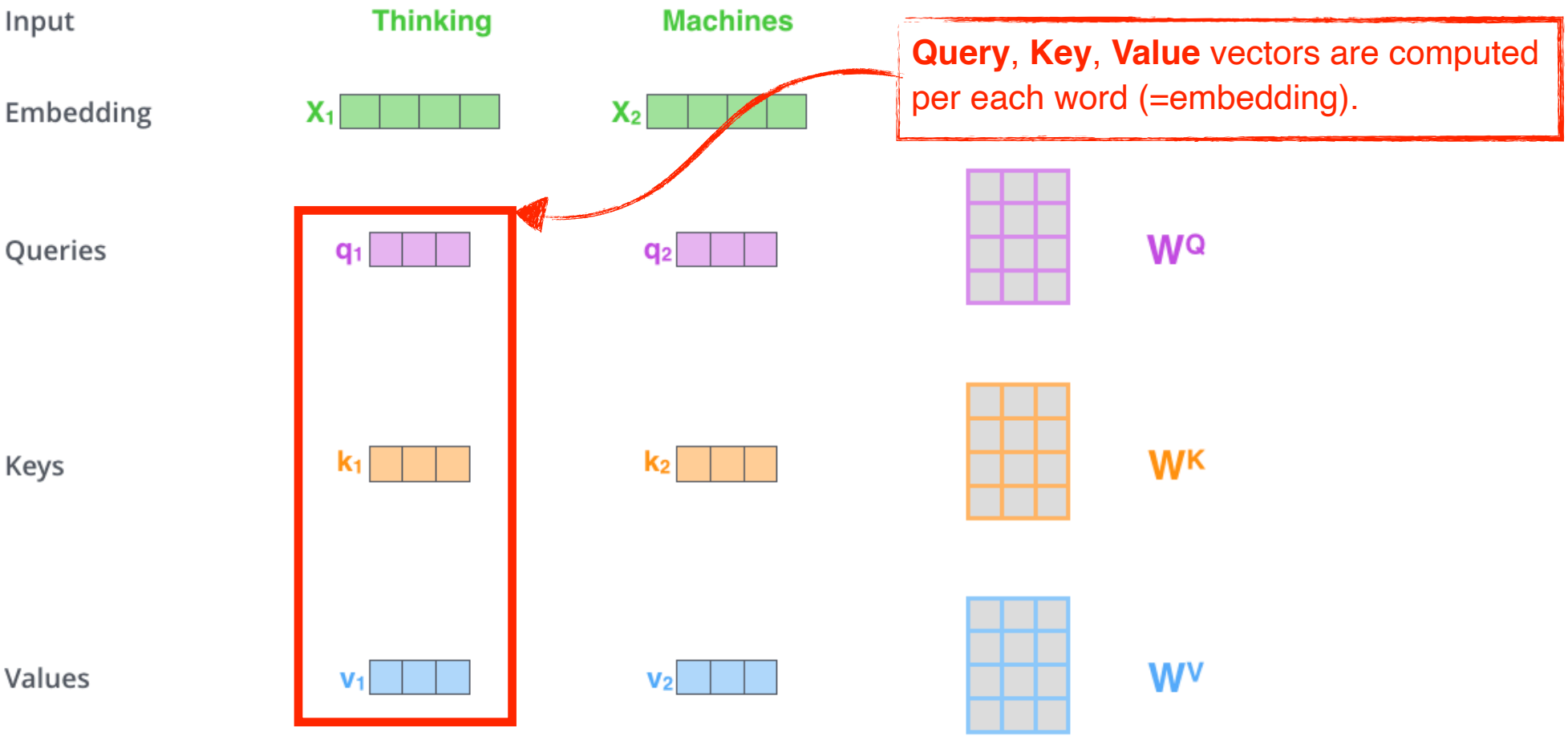
Transformer

- **Self-Attention** at a high level
  - The animal didn't cross the street because it was too tired.



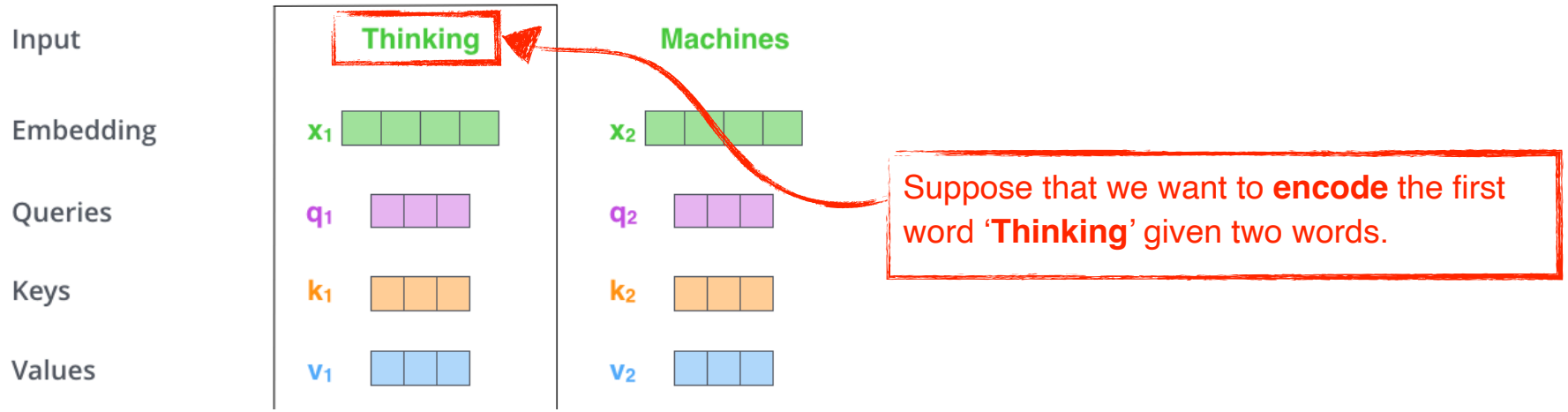
http://jalammar.github.io/illustrated-transformer/

Transformer



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

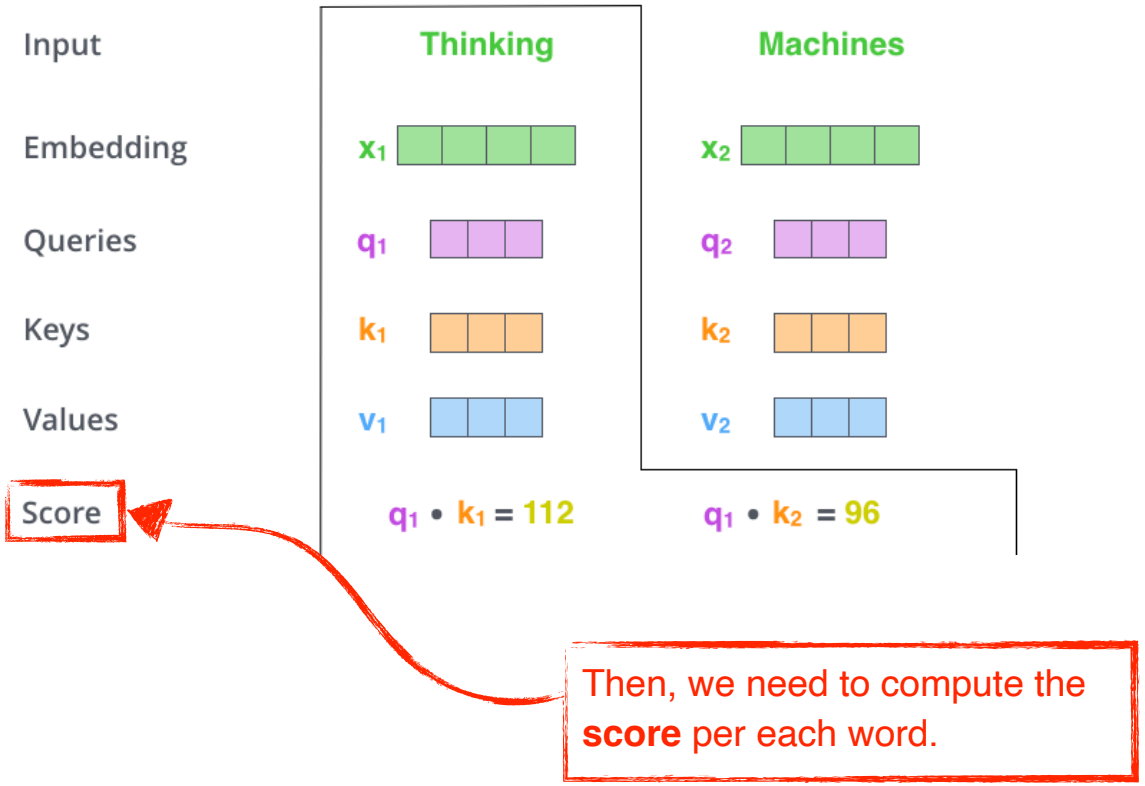
Transformer



- Suppose we are encoding the first word: '**Thinking**' given '**Thinking**' and '**Machines**'.

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

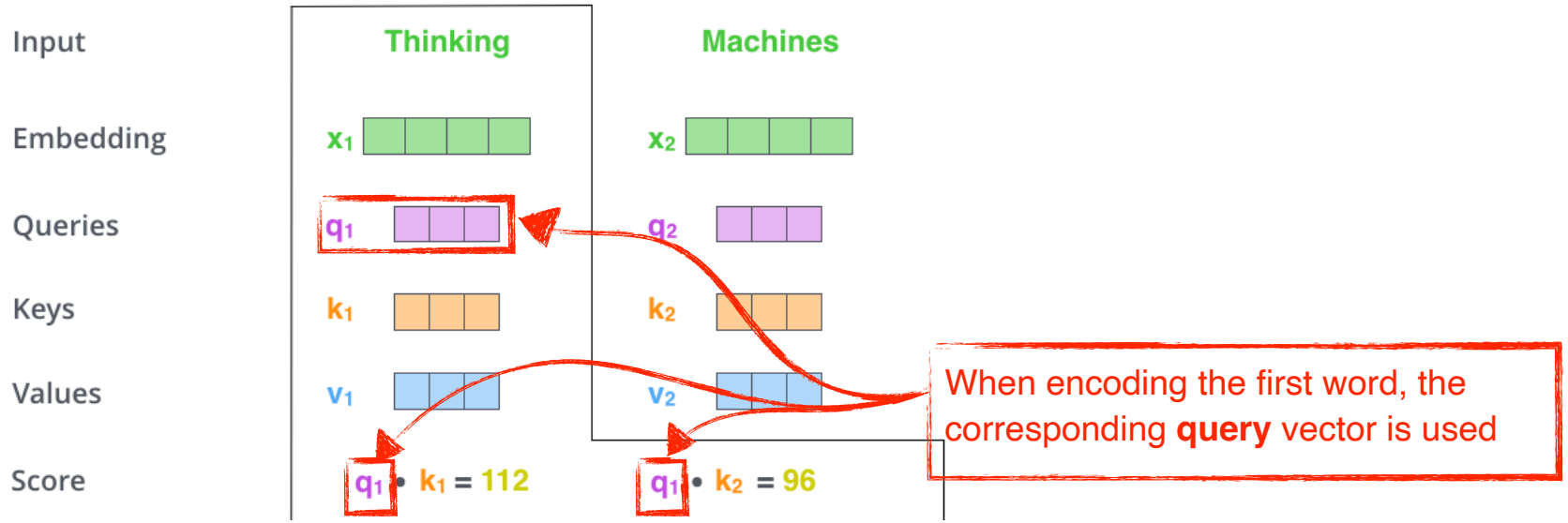
Transformer



- Suppose we are encoding the first word: '**Thinking**' given '**Thinking**' and '**Machines**'.

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

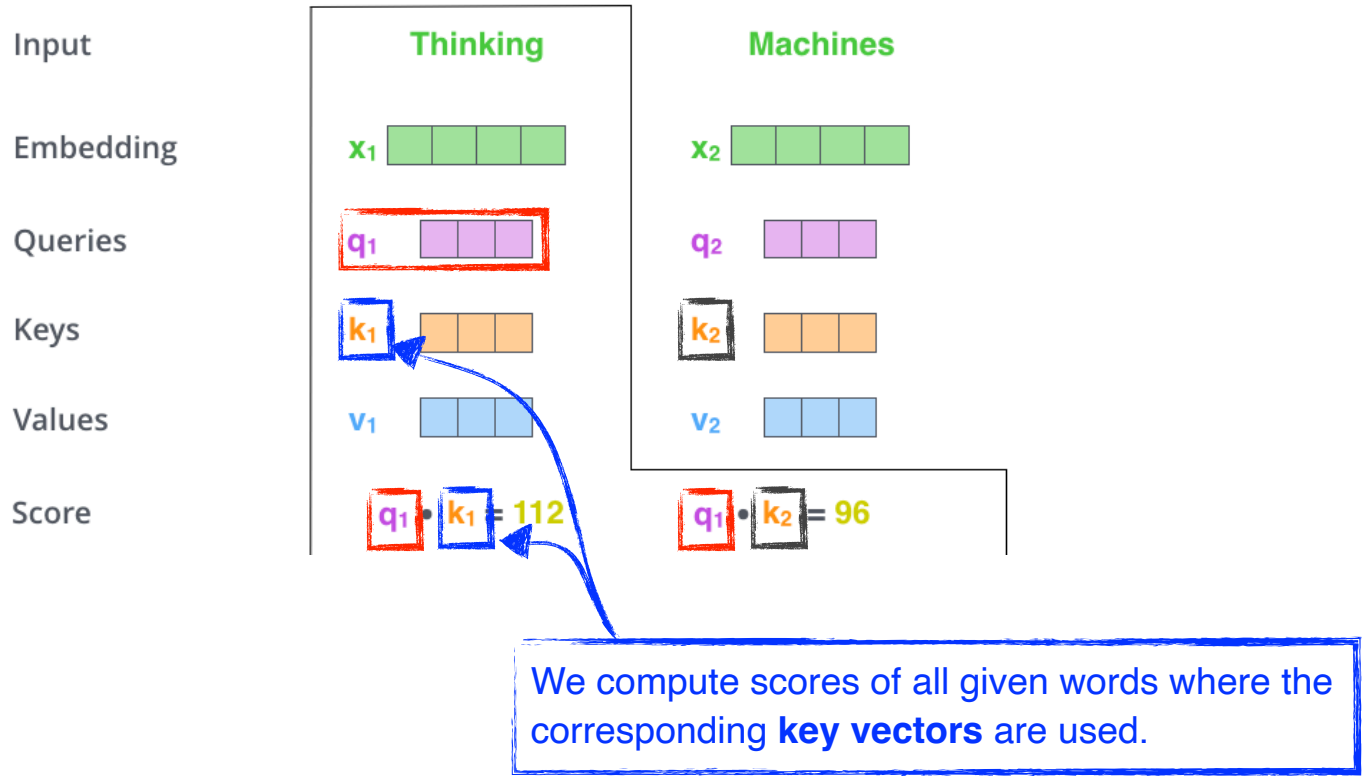
Transformer



- Suppose we are encoding the first word: '**Thinking**' given '**Thinking**' and '**Machines**'.

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

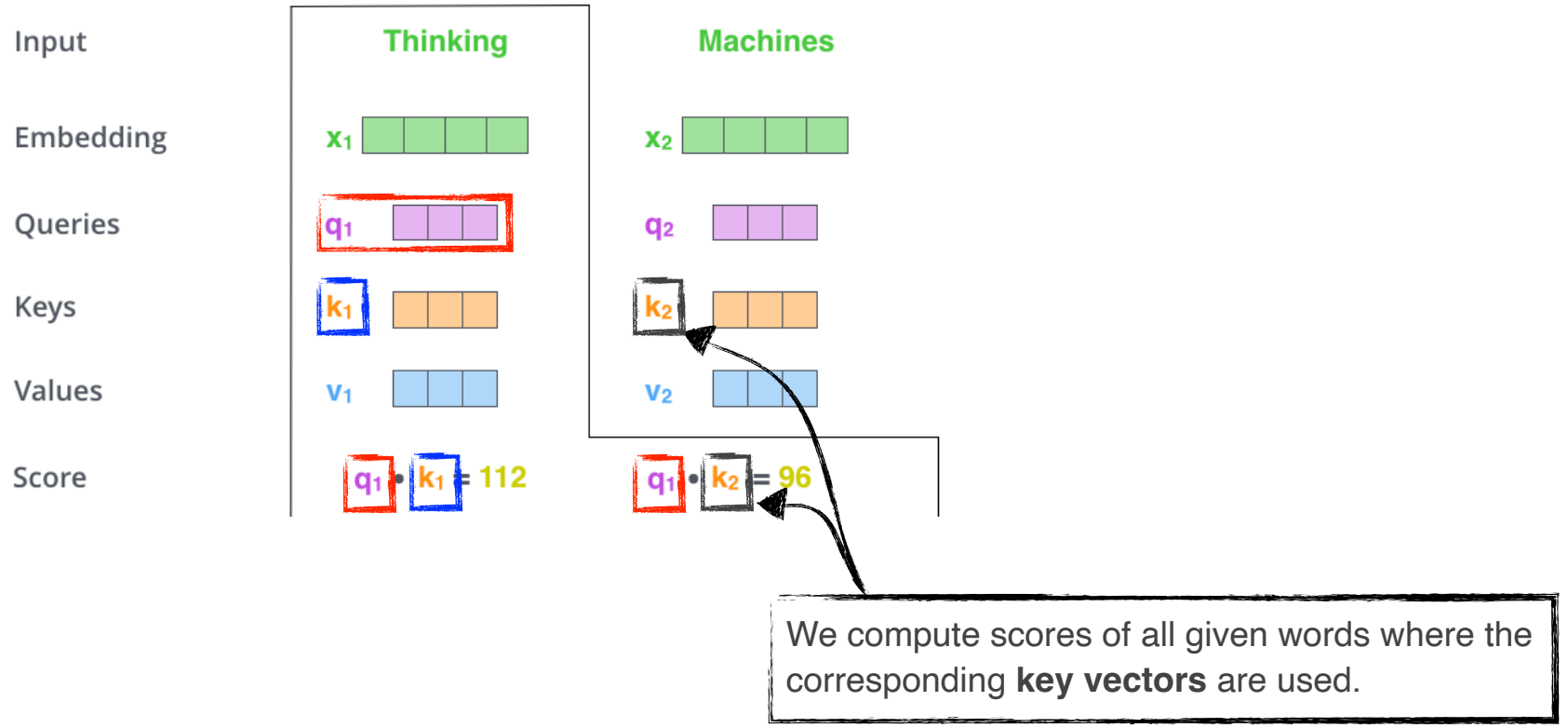
Transformer



- Suppose we are encoding the first word: '**Thinking**' given '**Thinking**' and '**Machines**'.

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

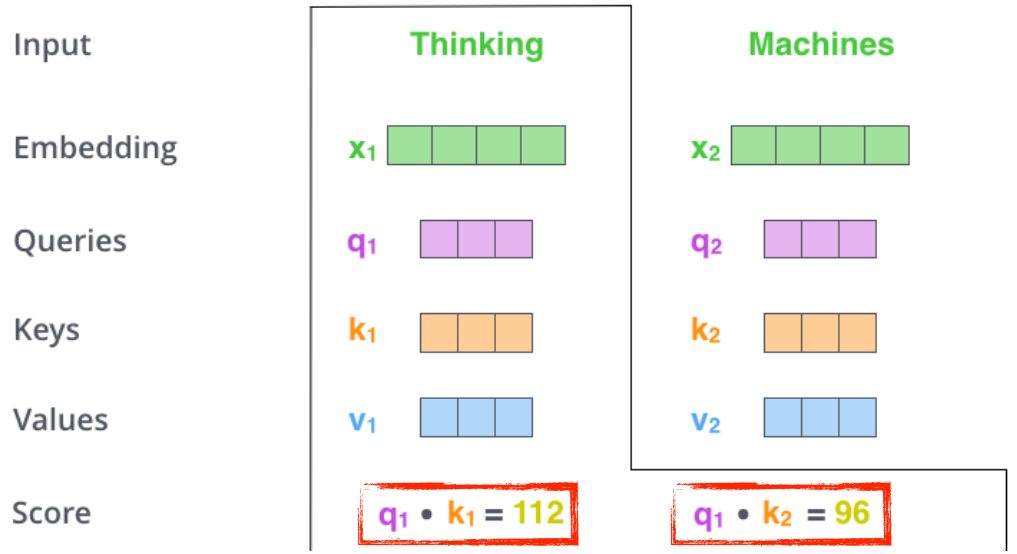
Transformer



- Suppose we are encoding the first word: '**Thinking**' given '**Thinking**' and '**Machines**'.

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

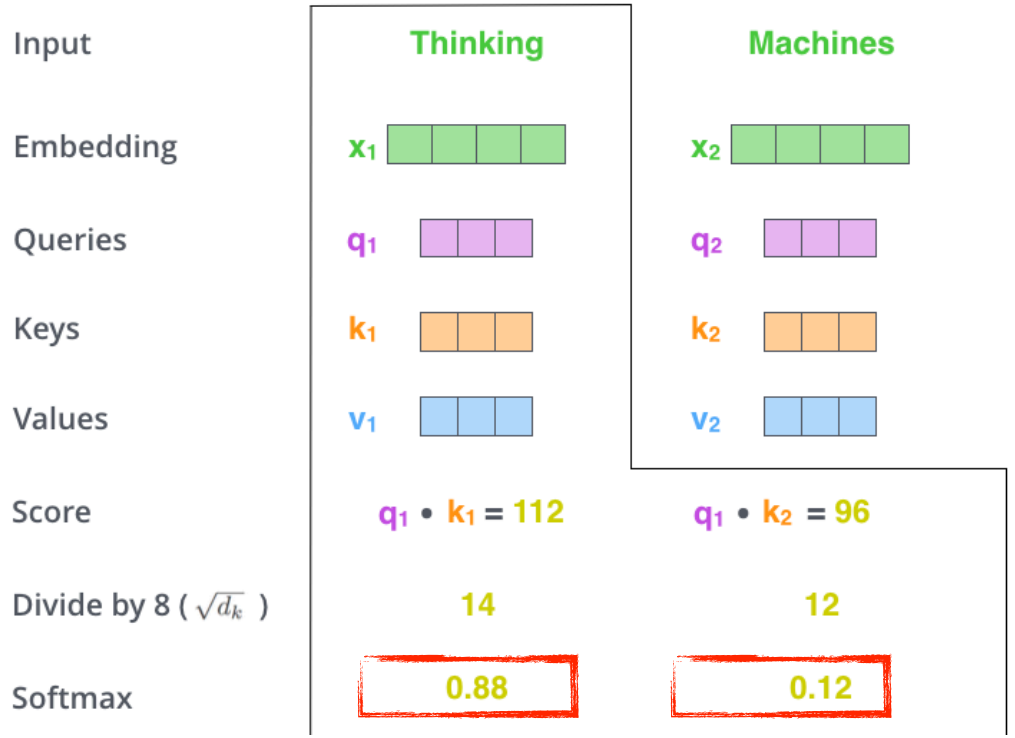
Transformer



- We compute the **scores** of each word with respect to '**Thinking**'.

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

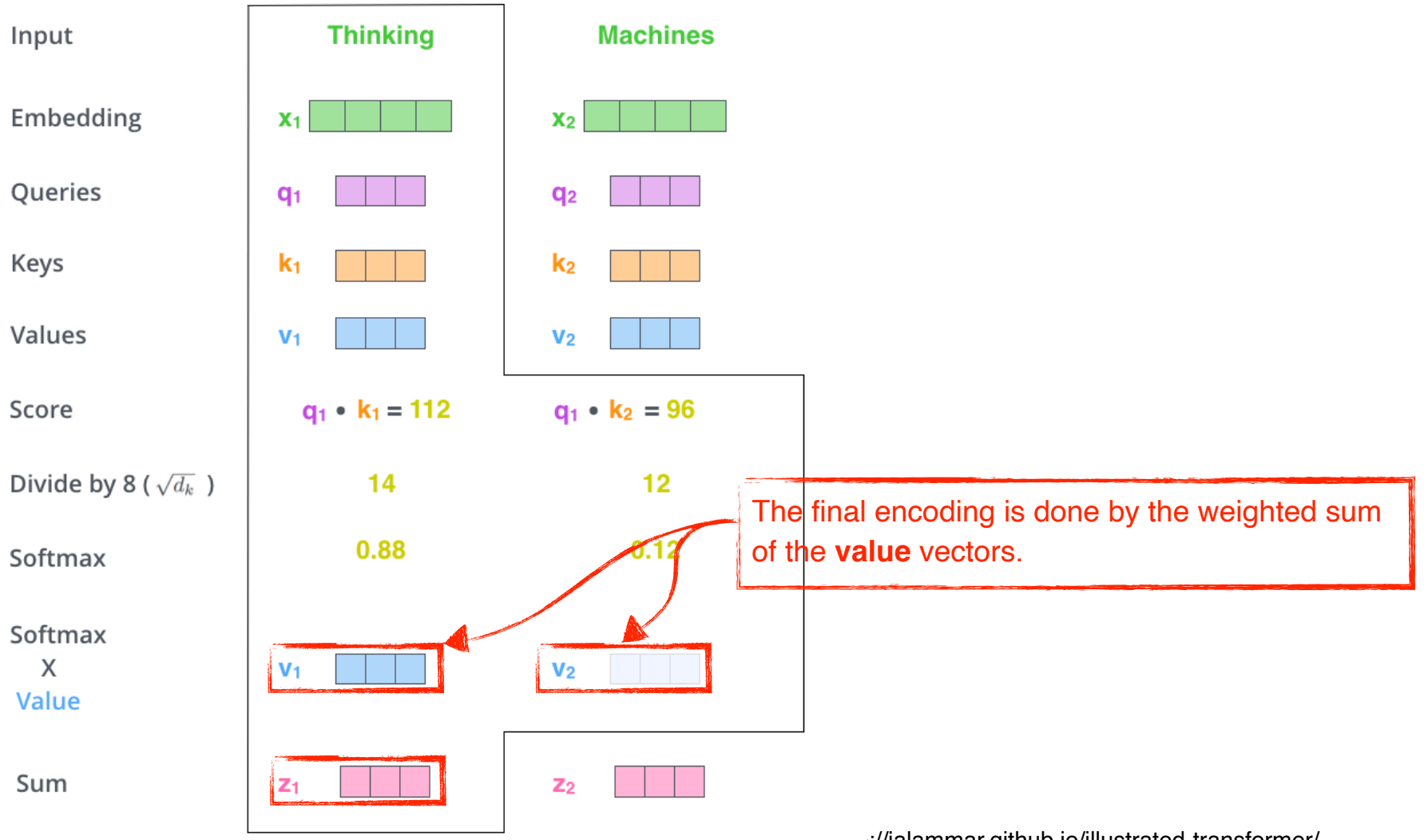
Transformer



- Then, we compute the **attention weights** by scaling followed by softmax.

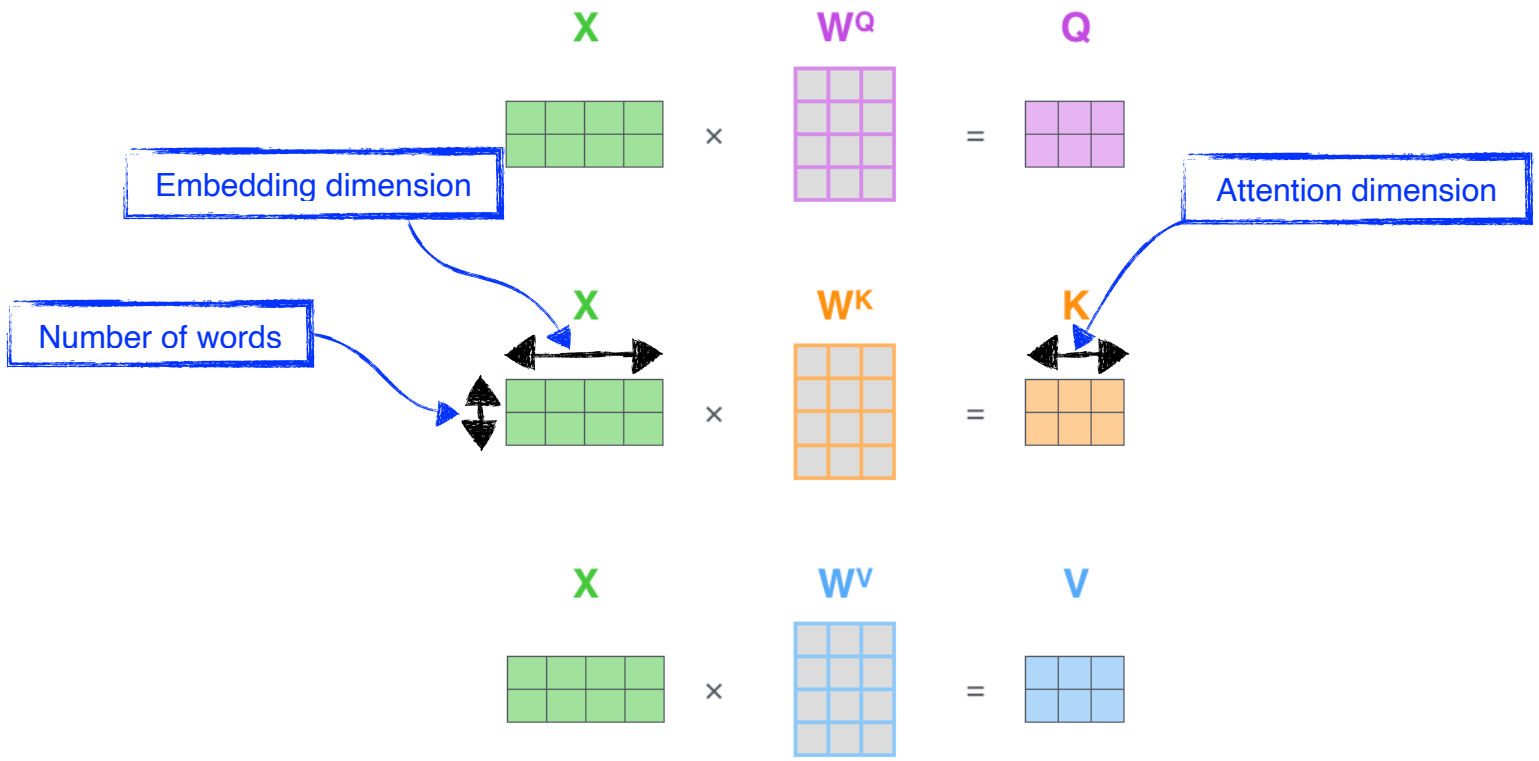
http://jalammar.github.io/illustrated-transformer/

Transformer



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

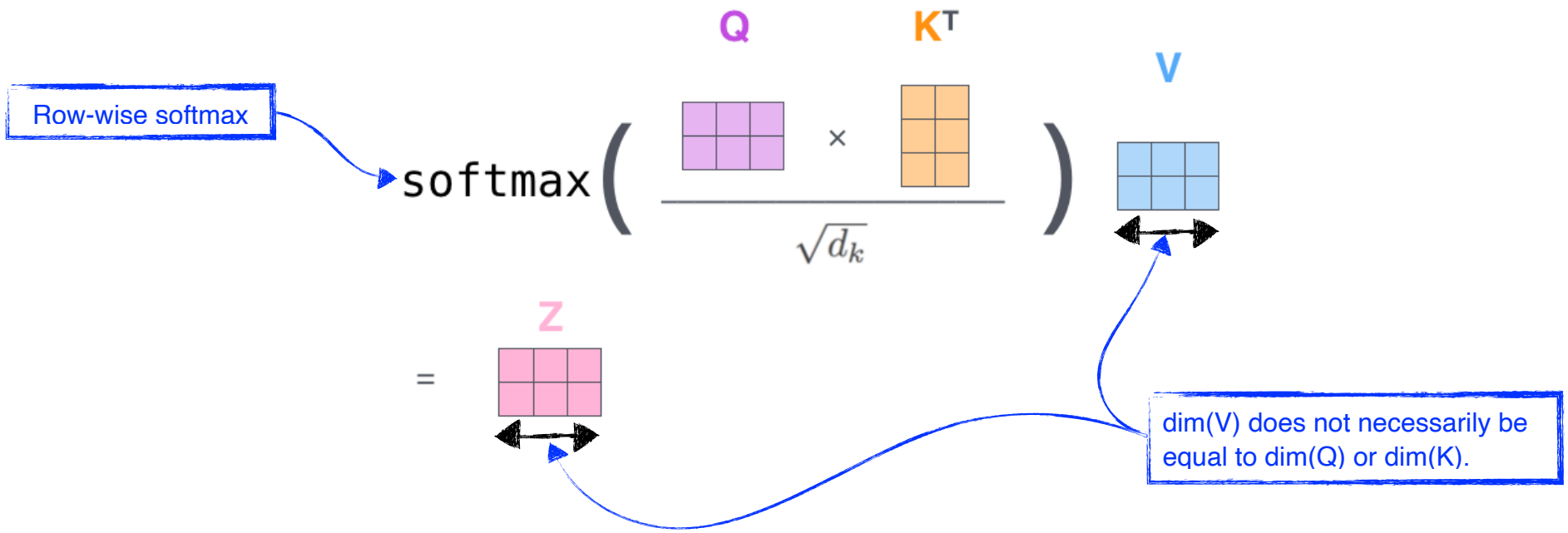
Transformer



- Calculating Q, K, and V from X in a matrix form.

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

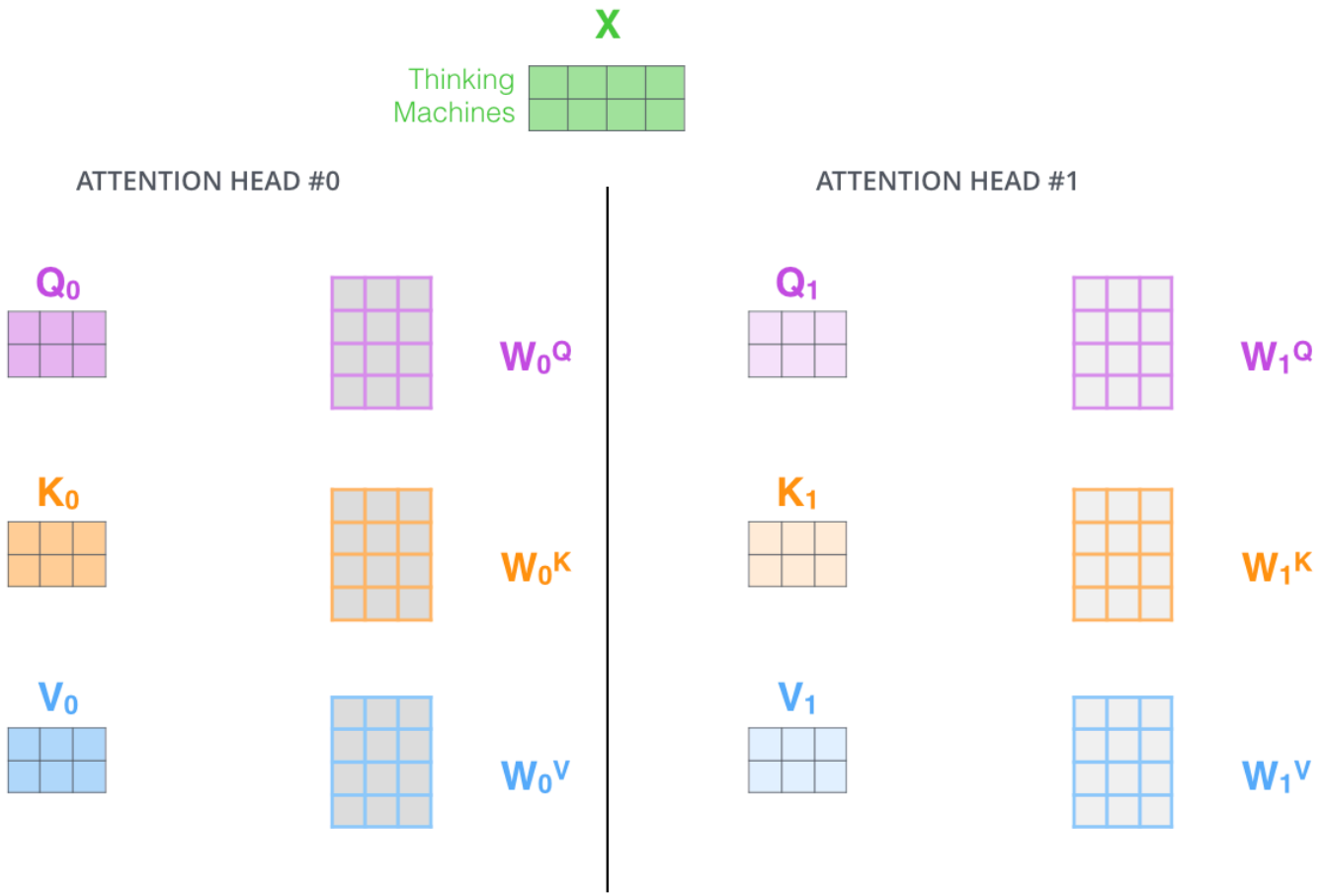
Transformer



- Calculating Q, K, and V from X in a matrix form.

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

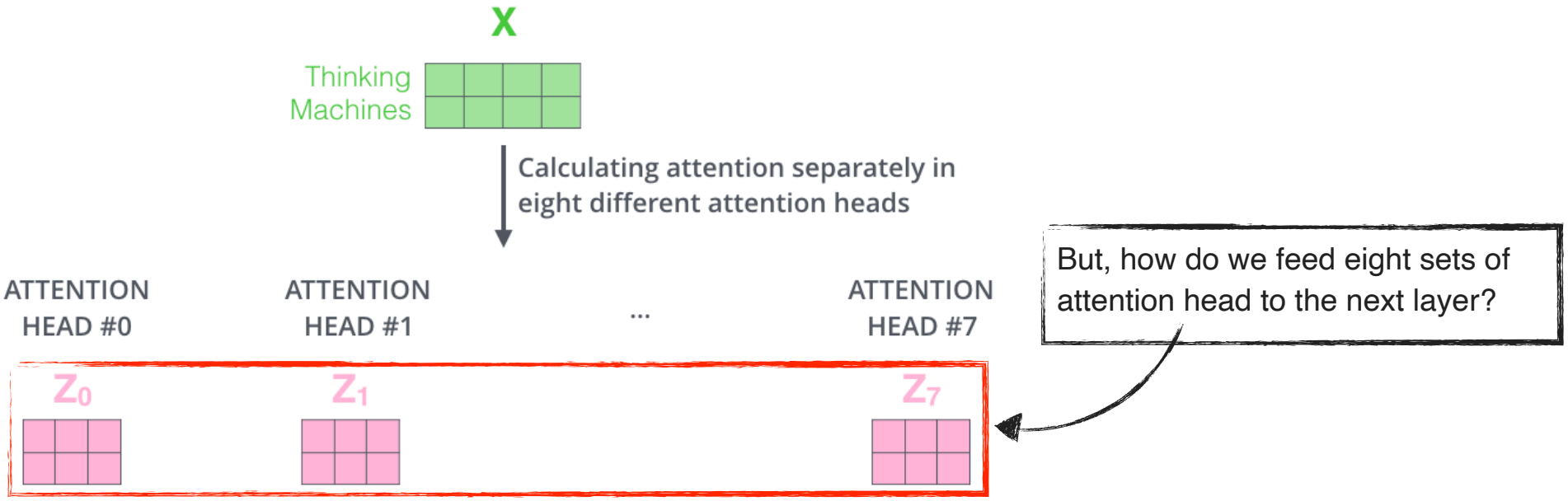
Transformer



- Multi-headed attention (MHA) allows Transformer to focus on different positions.

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

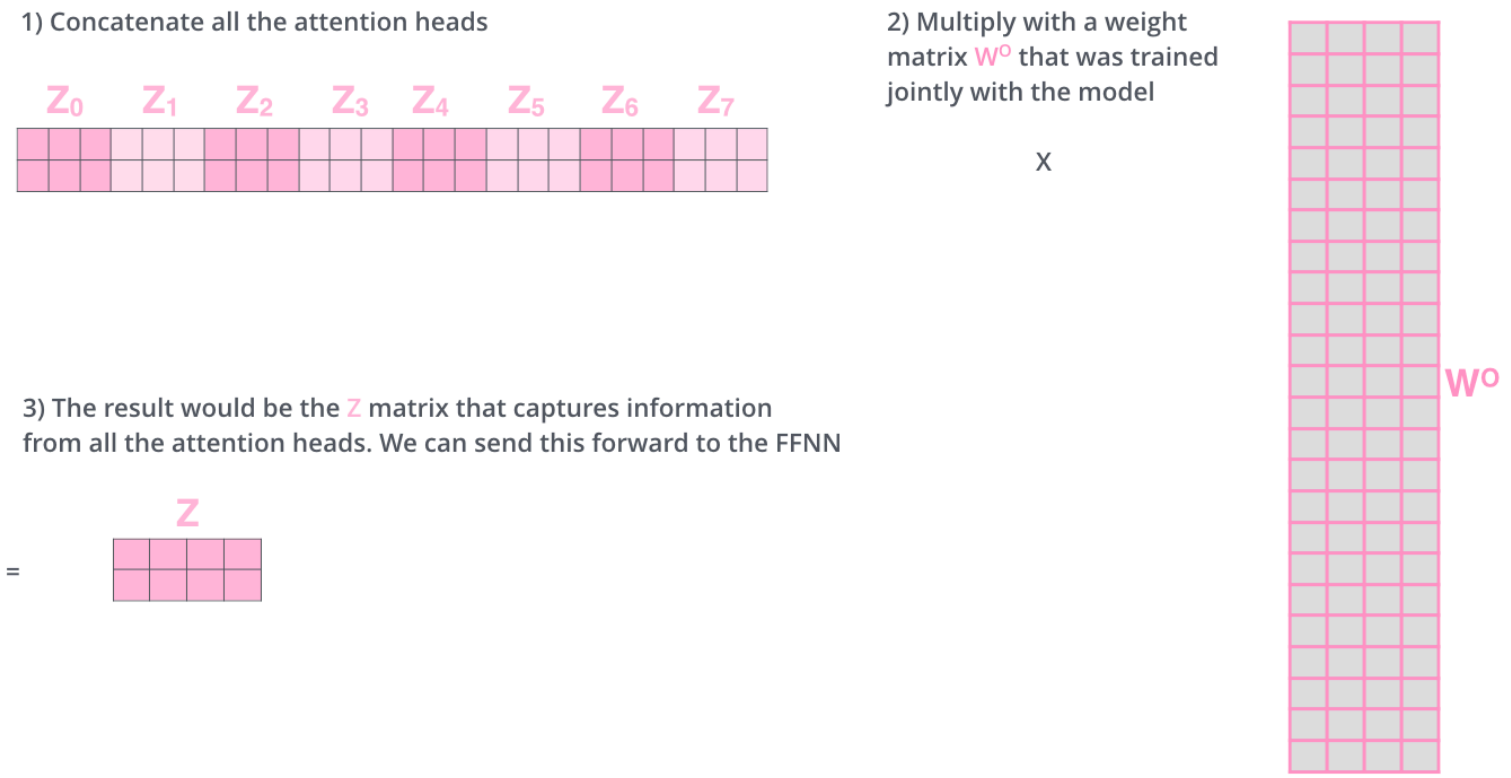
Transformer



- If eight heads are used, we end up getting eight different sets of encoded vectors (**attention heads**).

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

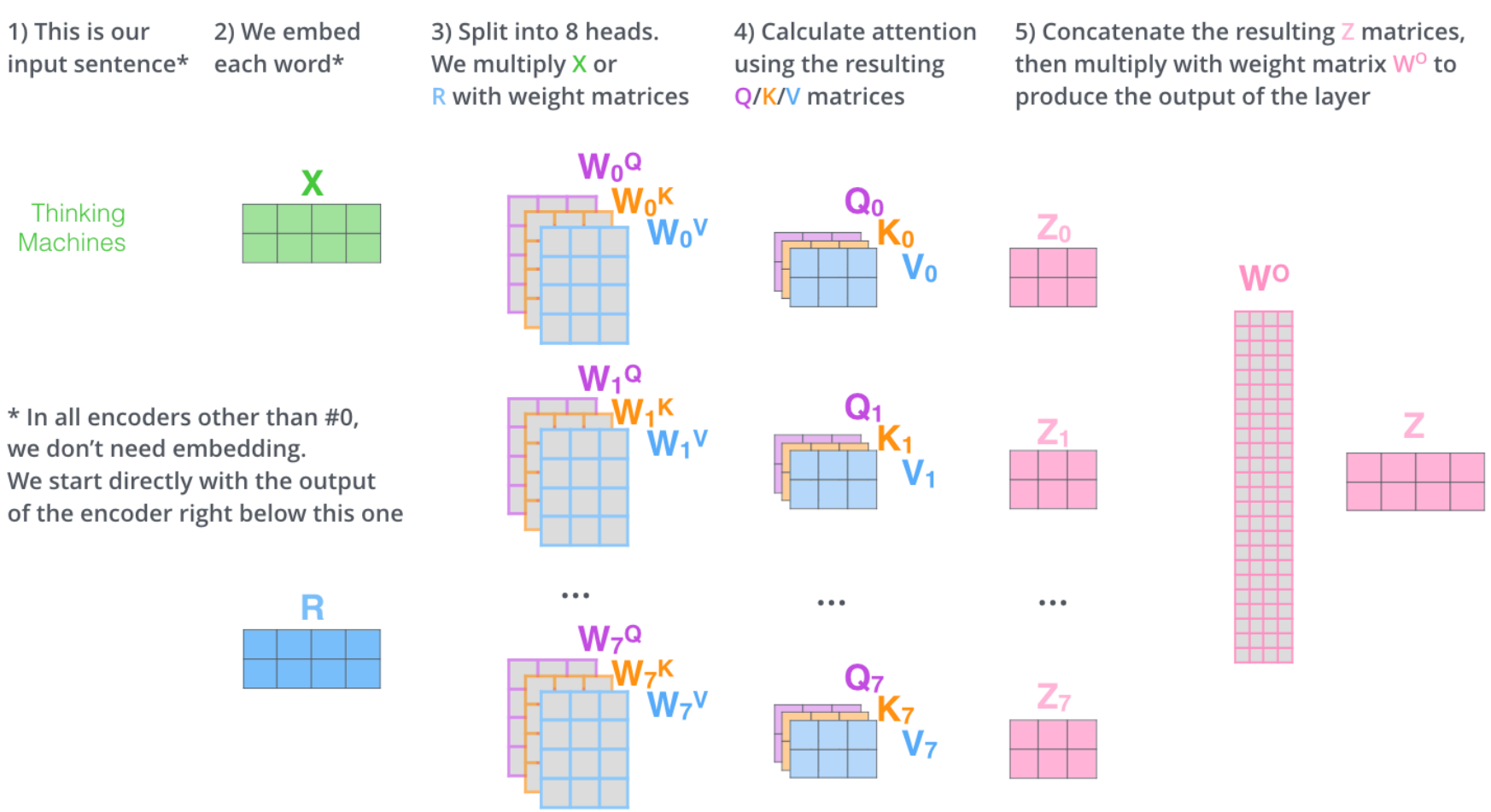
Transformer



- We simply pass them through additional (learnable) linear map.

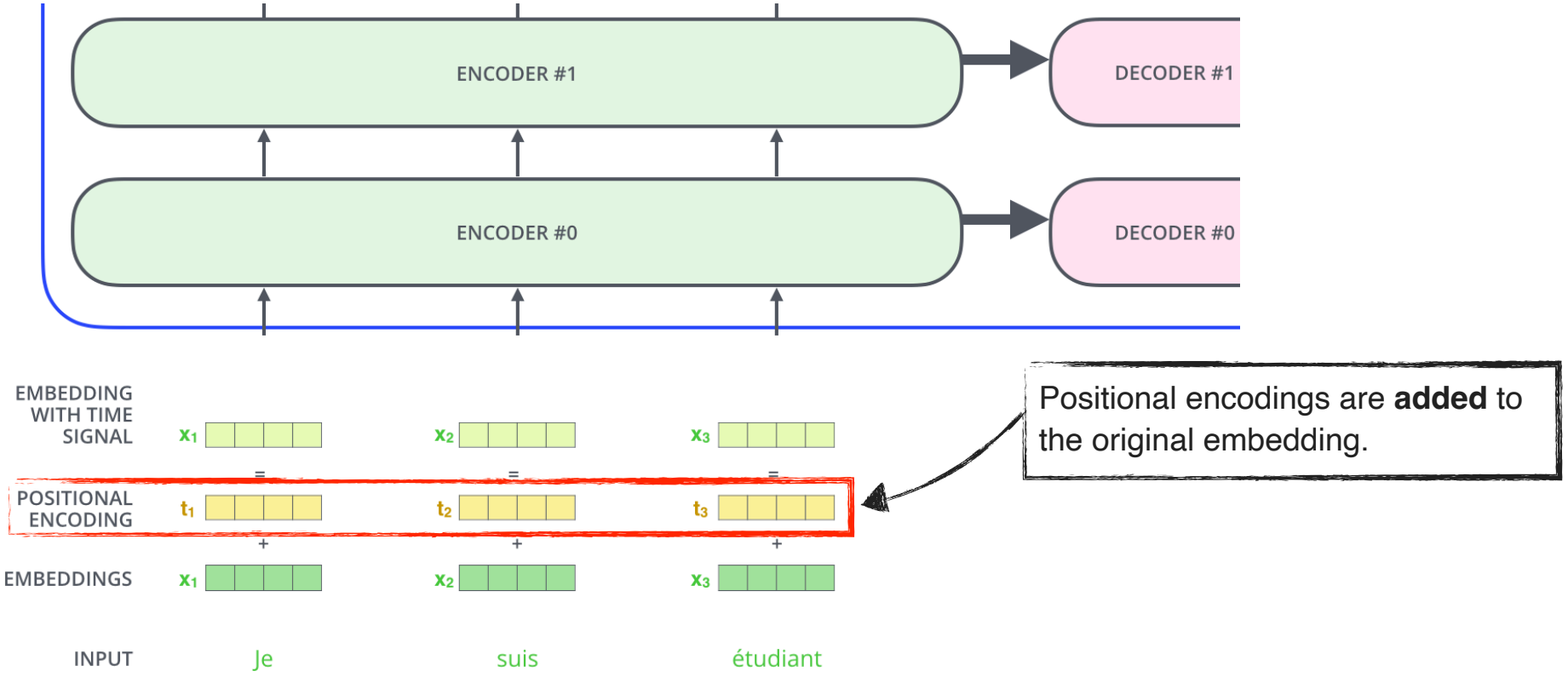
http://jalammar.github.io/illustrated-transformer/

Transformer



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

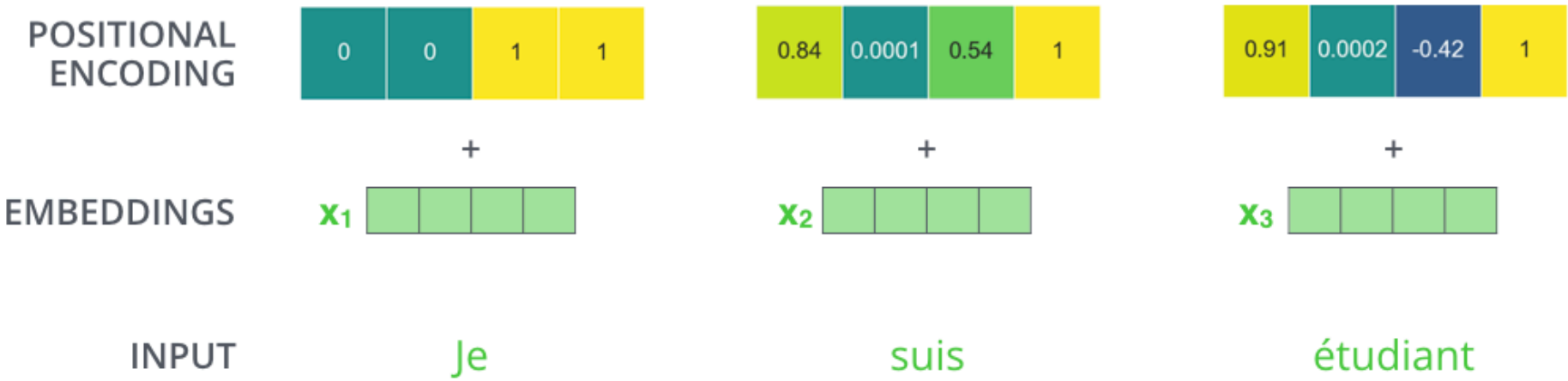
Transformer



- Why do we need positional encoding?

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

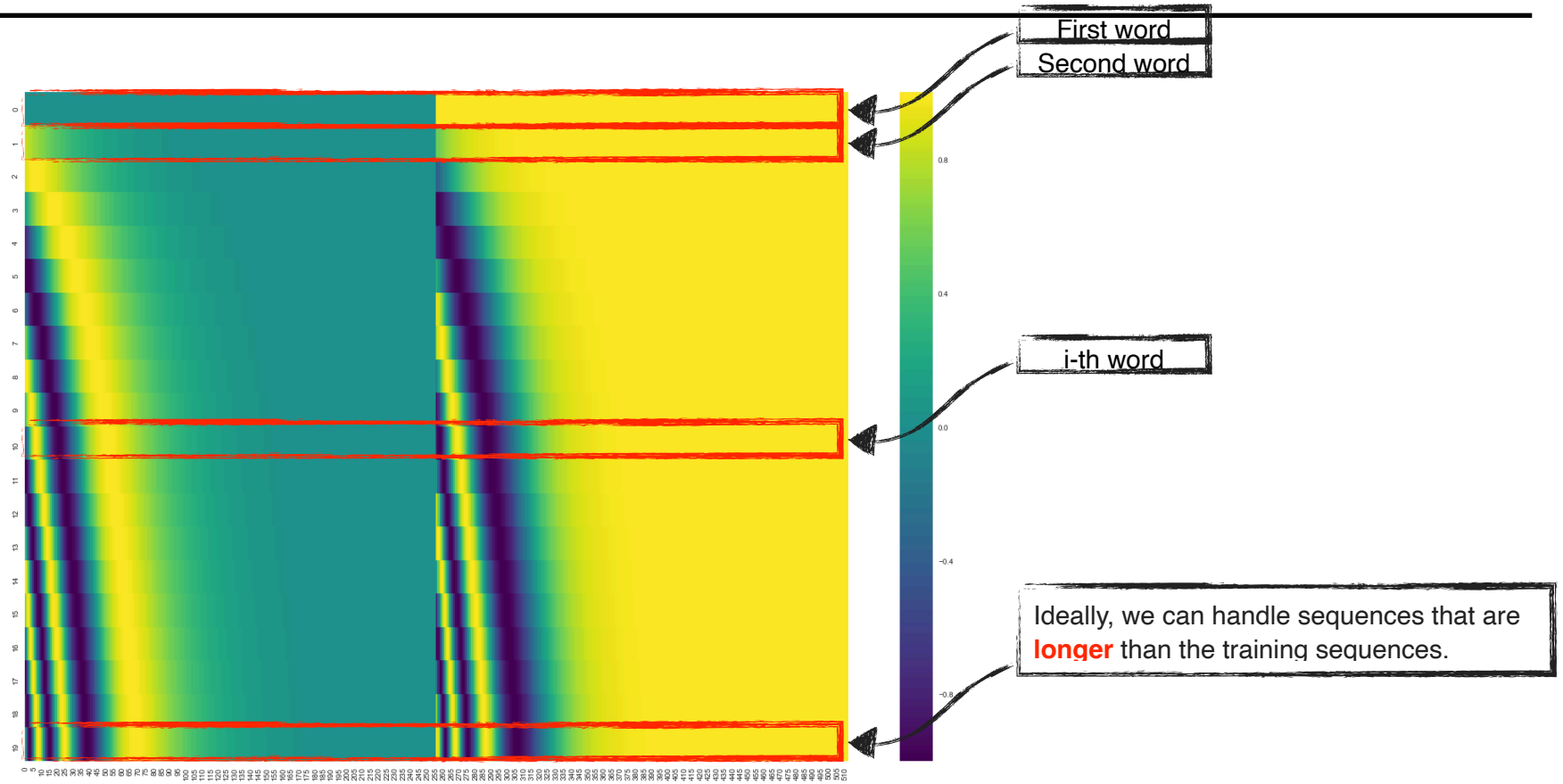
Transformer



- This is the case for 4-dimensional encoding.

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

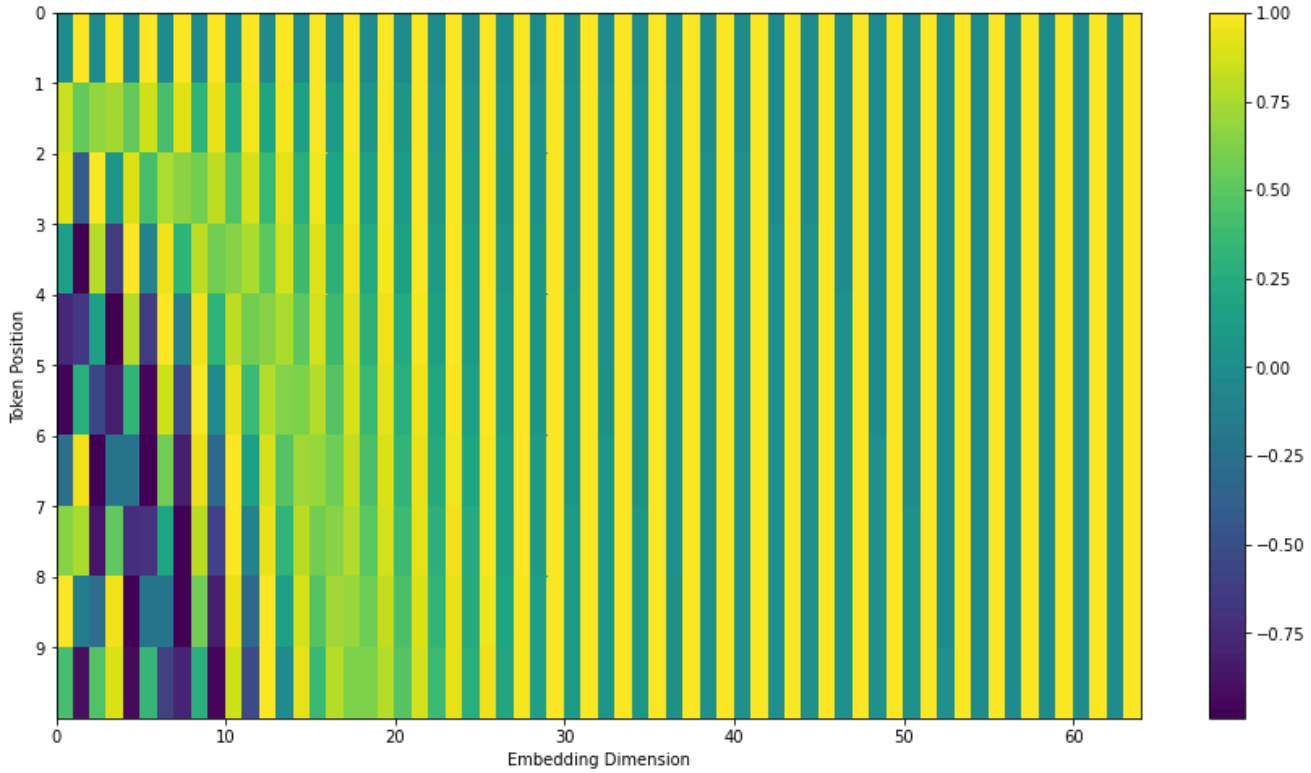
Transformer



- This is the case for 512-dimensional encoding.

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

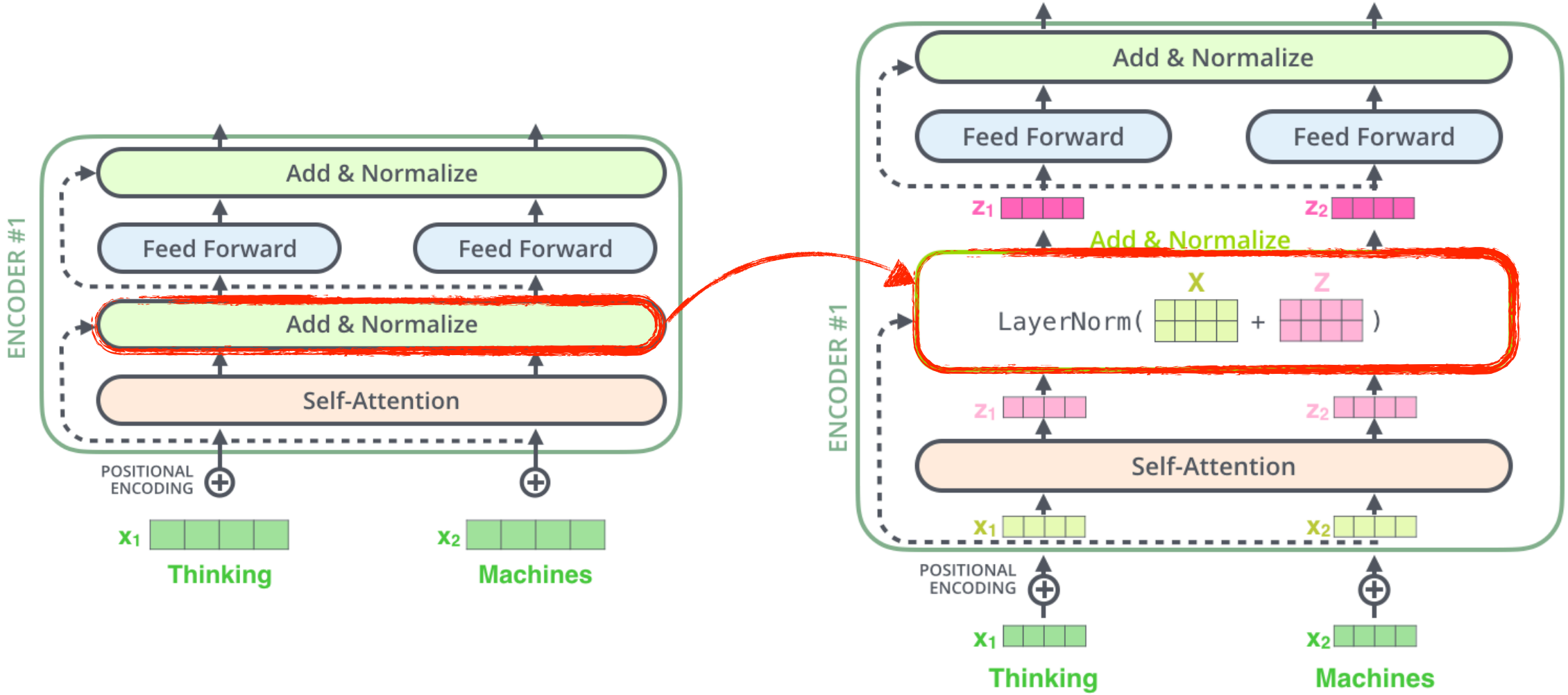
Transformer



- Recent (July, 2020) update on positional encoding.

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

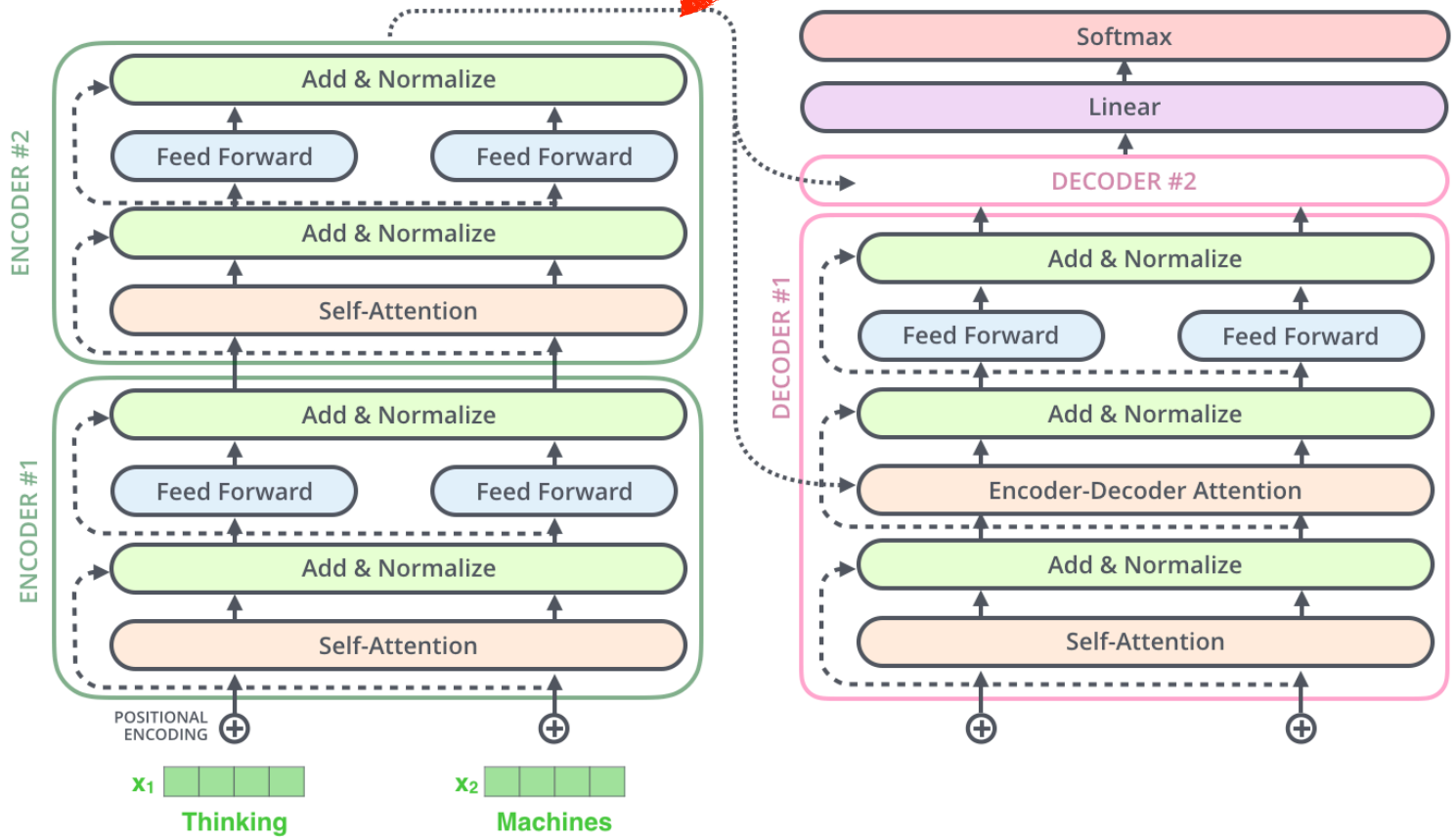
Transformer



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

Transformer

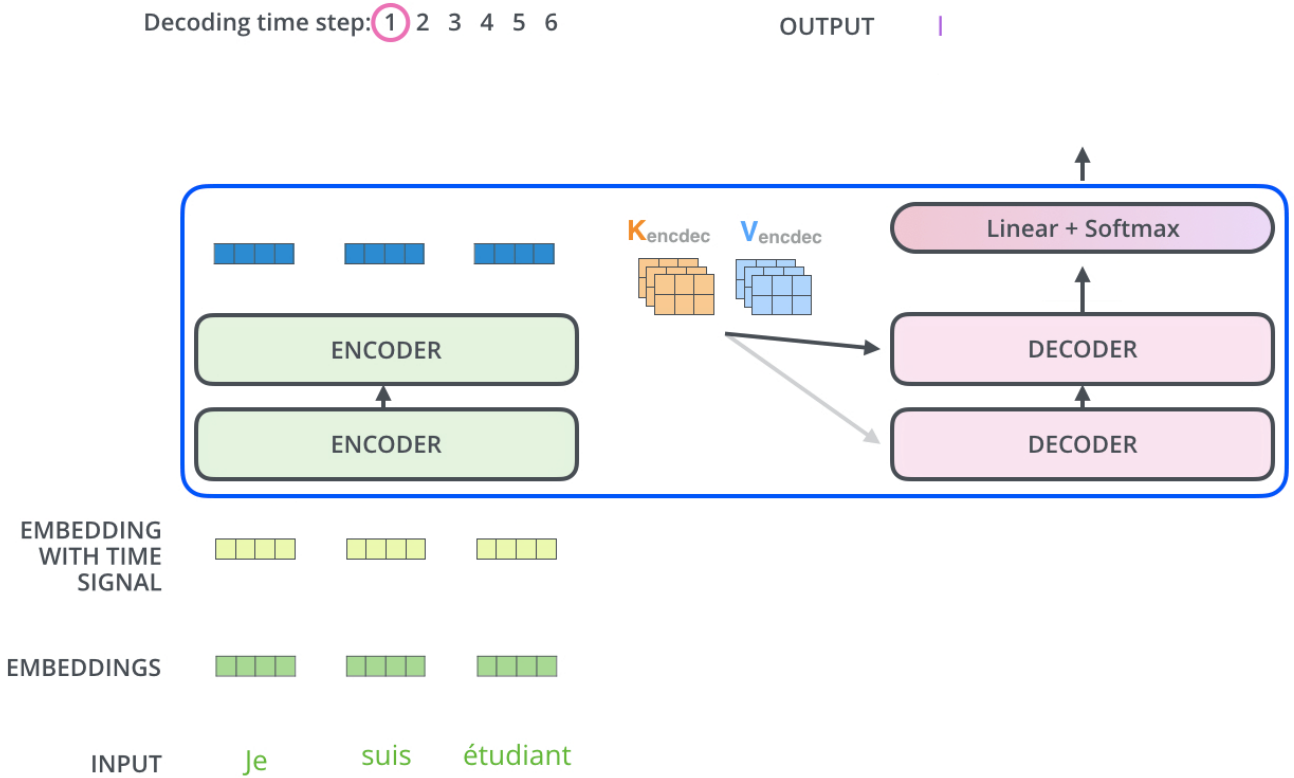
What is the information being sent from **encoders** to **decoders**?

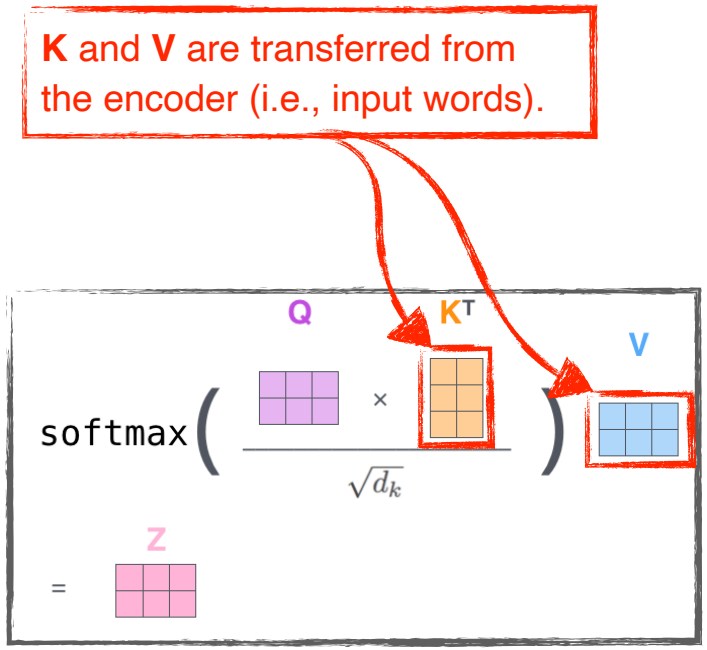


- Now, let's take a look at the **decoder** side.

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

Transformer

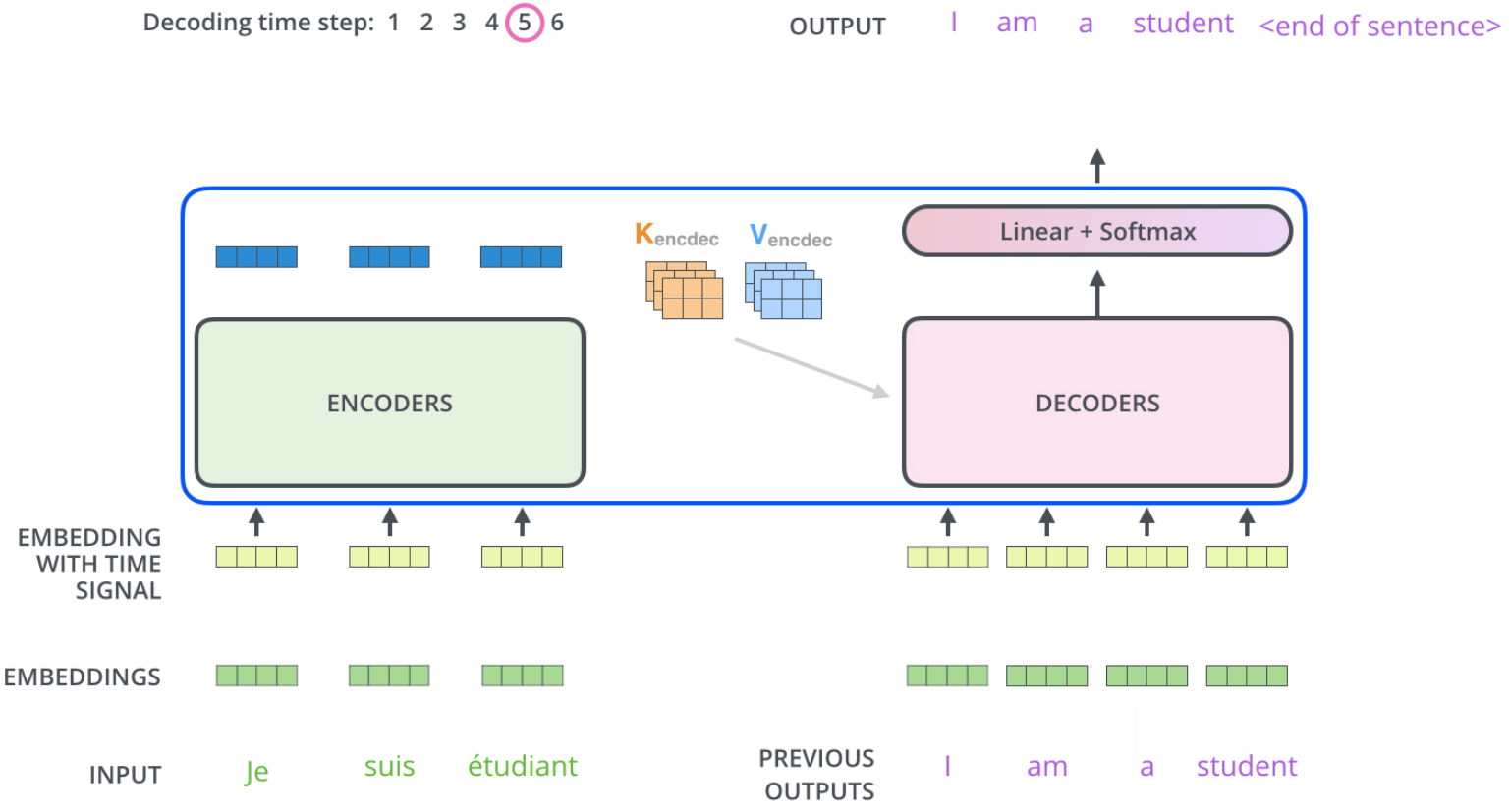




- **Transformer** transfers **key (K)** and **value (V)** of the topmost encoder to the decoder.

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

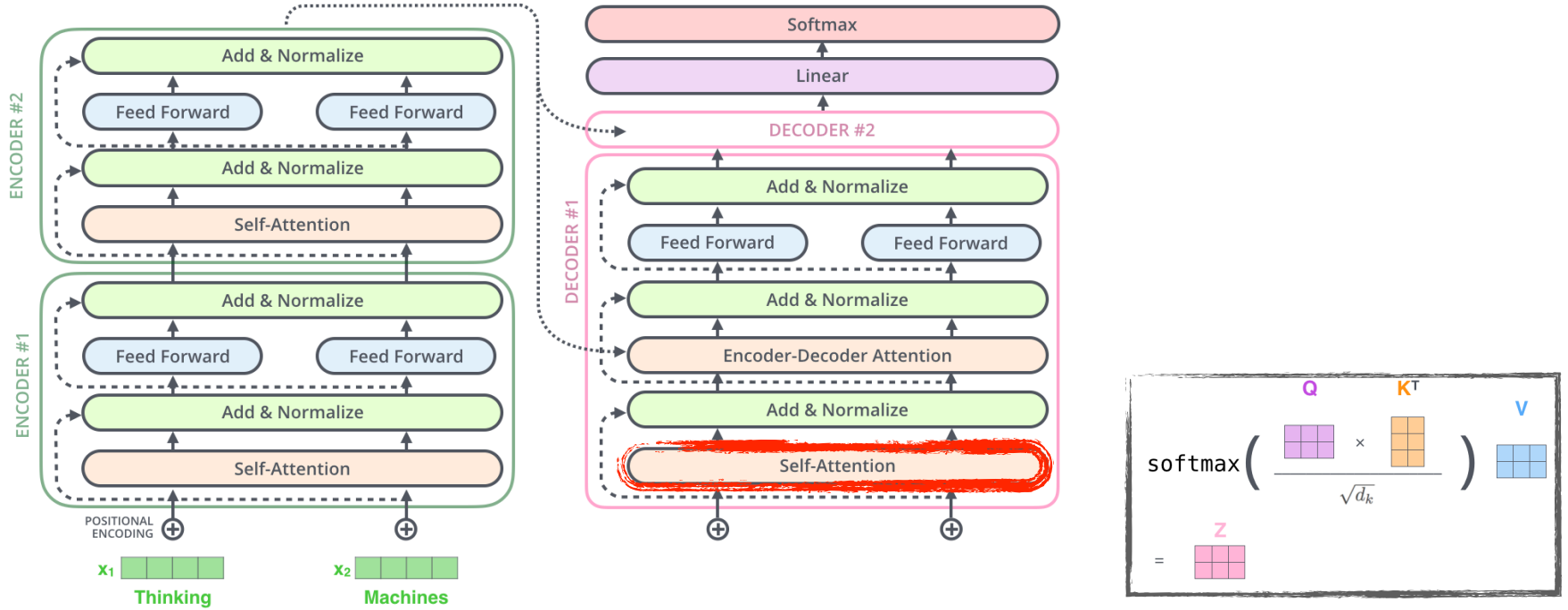
Transformer



- The output sequence is generated in an autoregressive manner.

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

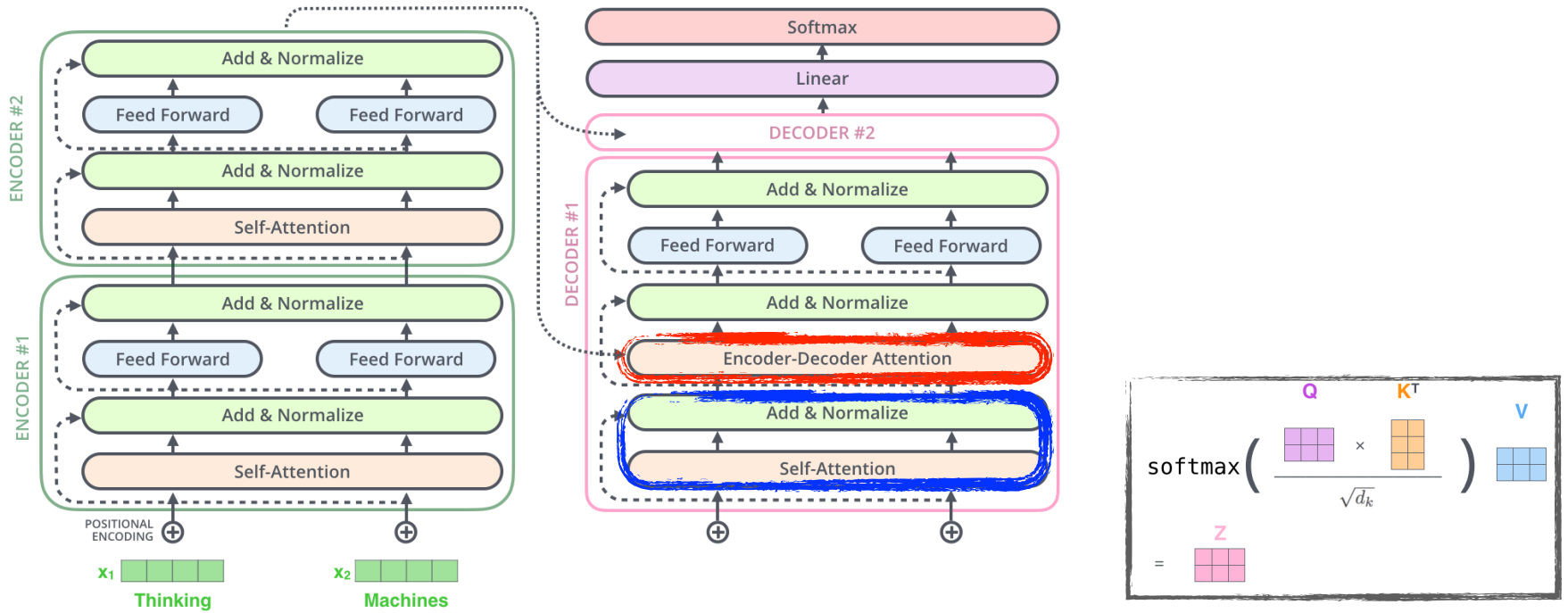
Transformer



- In the **decoder**, the **self-attention layer** is only allowed to attend to earlier positions in the output sequence which is done by masking future positions before the softmax step.

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

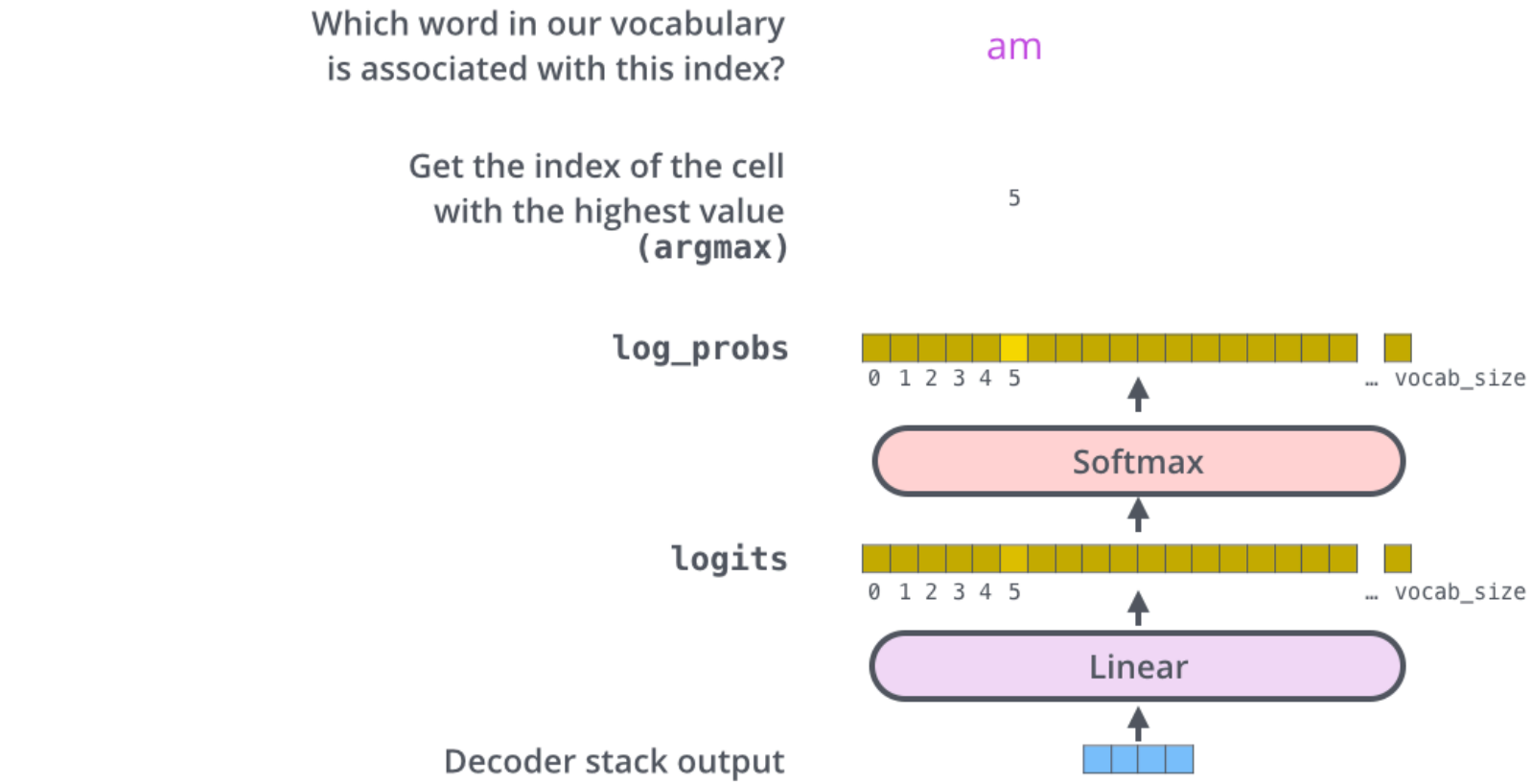
Transformer



- The "**Encoder-Decoder Attention**" layer works just like multi-headed self-attention, except it creates its **Queries** matrix from **the layer below it**, and takes the **Keys** and **Values** from the encoder stack.

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

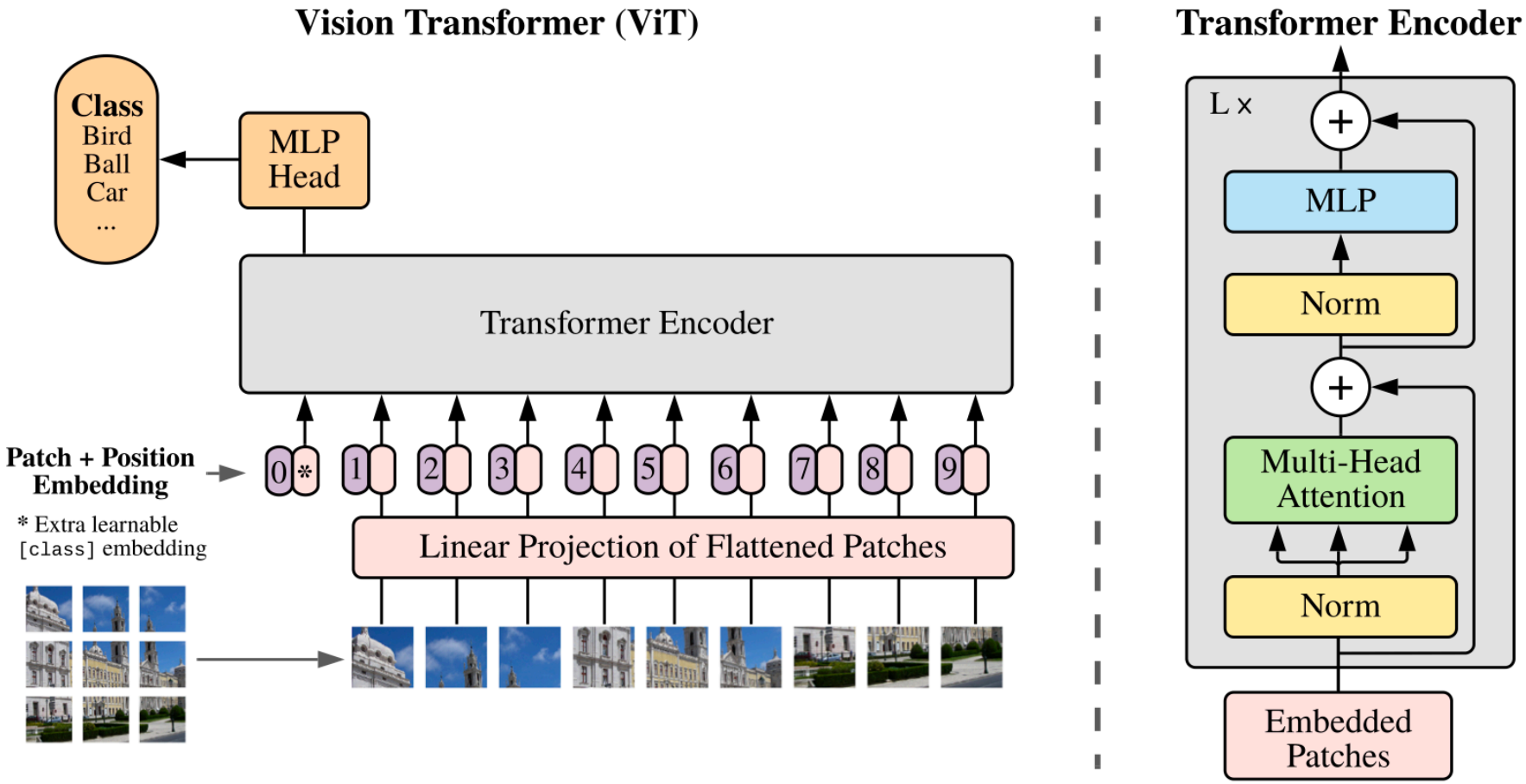
Transformer



- The final layer converts the stack of decoder outputs to the distribution over words.

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

Vision Transformer



https://arxiv.org/abs/2010.11929

DALL-E

An armchair in the shape of an avocado.

https://openai.com/blog/dall-e/

Thank you for listening