Attention Is All You Need

Introducing Transformer Networks

Credit Where Credit is Due

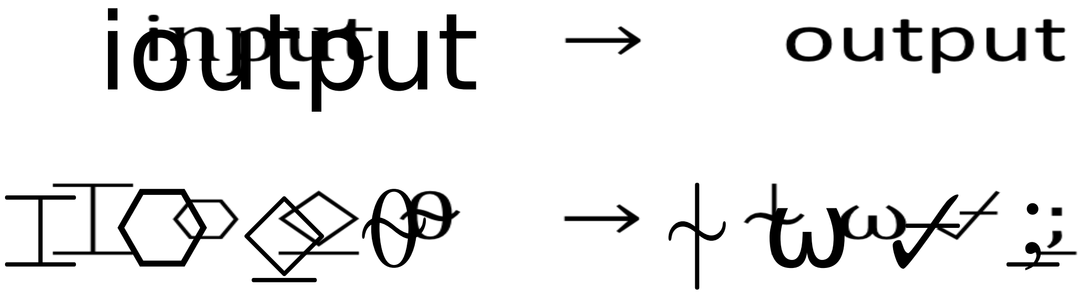
- Many of the diagrams in my slides were taken from Jay Alammar's "Illustrated Transformer" post (http://jalammar.github.io/illustrated-transformer/)
- I highly recommend reading it and/or the original paper for clarifications

Agenda

- The problem
- Some prior solutions and their weaknesses
- Deep dive into Transformer Networks
  - Architecture
  - Comparisons

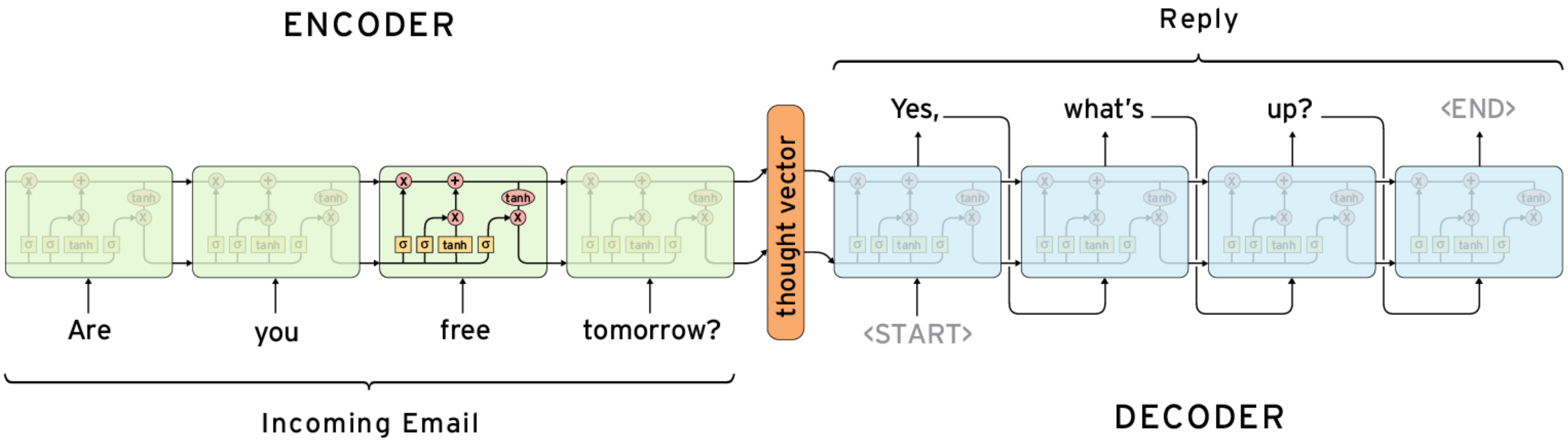
The Sequence to Sequence Problem (seq2seq)

- Map input sequences to output sequences
- Some examples:
    - Machine translation
    - Part of speech tagging
    - Text summarization
    - Chatbots



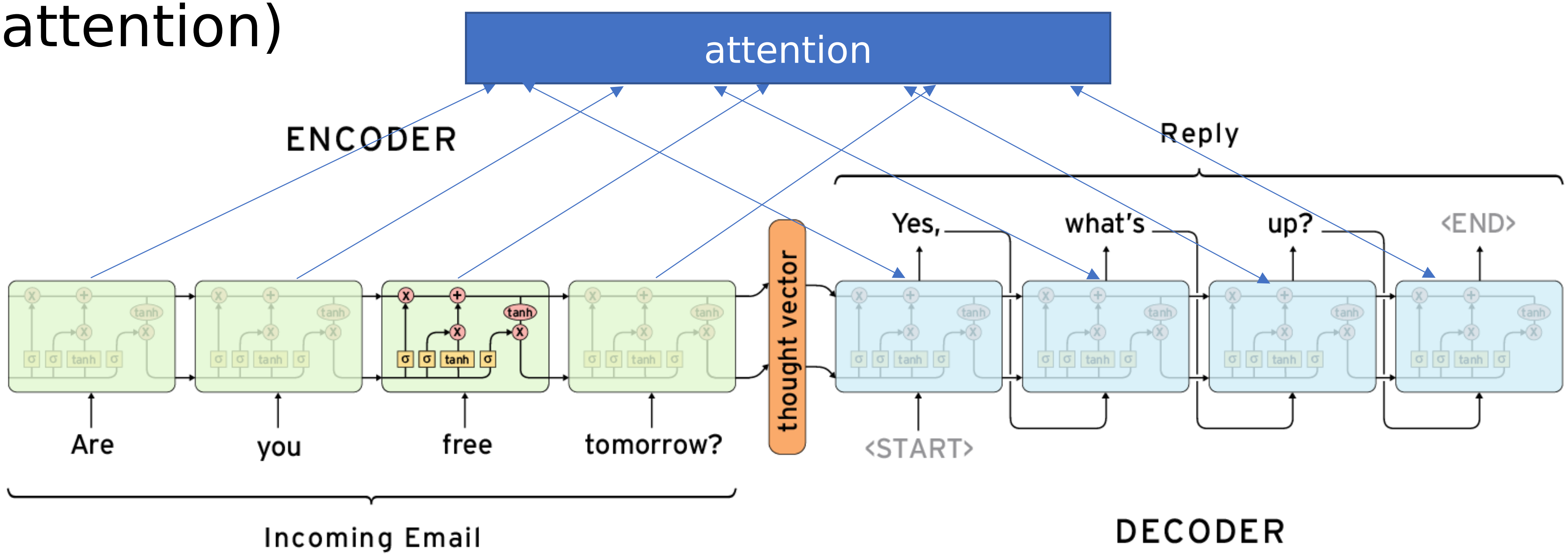
The Classic Solution

- Encoder-Decoder RNNs with fixed context



Introducing Attention

- Encoder-Decoder RNNs with more flexible context (i.e. attention)

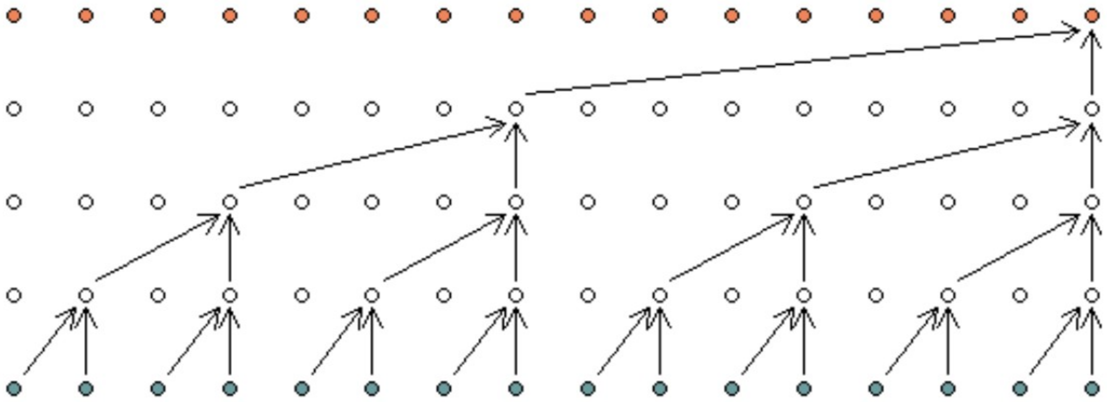


The Thing About RNNs

- Recurrent computation is sequential!
- This prevents parallelization **within** training examples
- Memory constraints limit how much parallelization can take place **across** training examples
- The number of operations required to relate signals from arbitrary input or output positions is O(n)

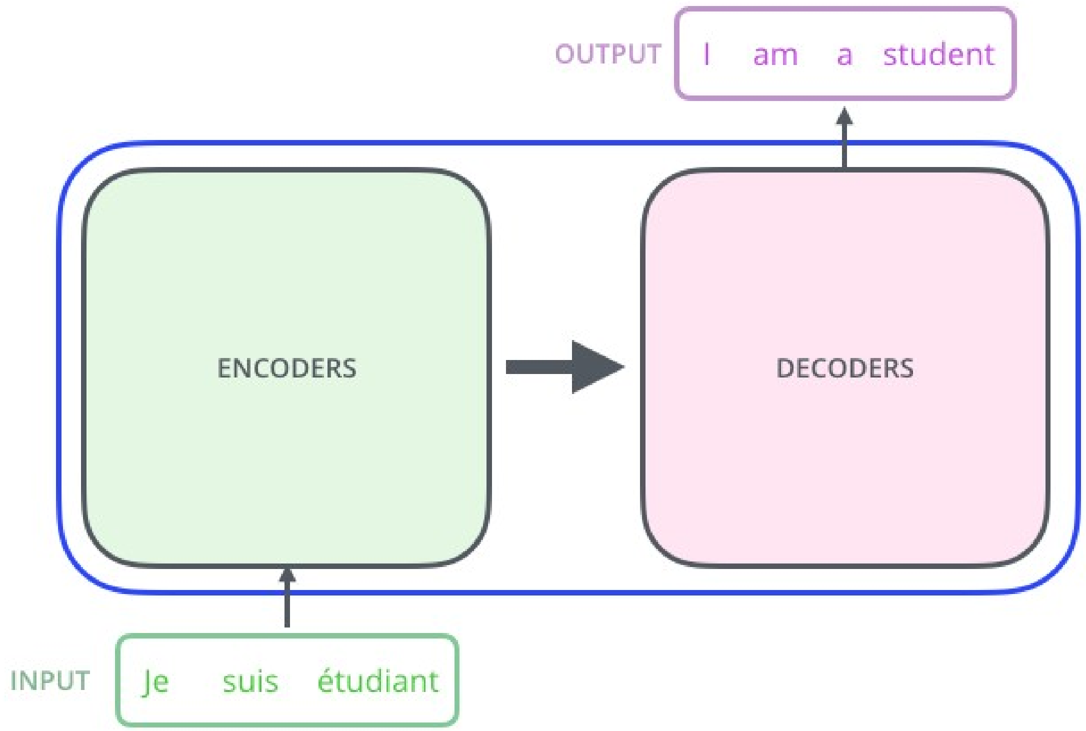
Reducing Sequential Computation

- Other approaches to seq2seq:
  - ConvS2S
  - ByteNet
- Using convolution to compute representations in parallel for all tokens
- The number of operations required to relate signals from arbitrary input or output positions still grows with sequence length



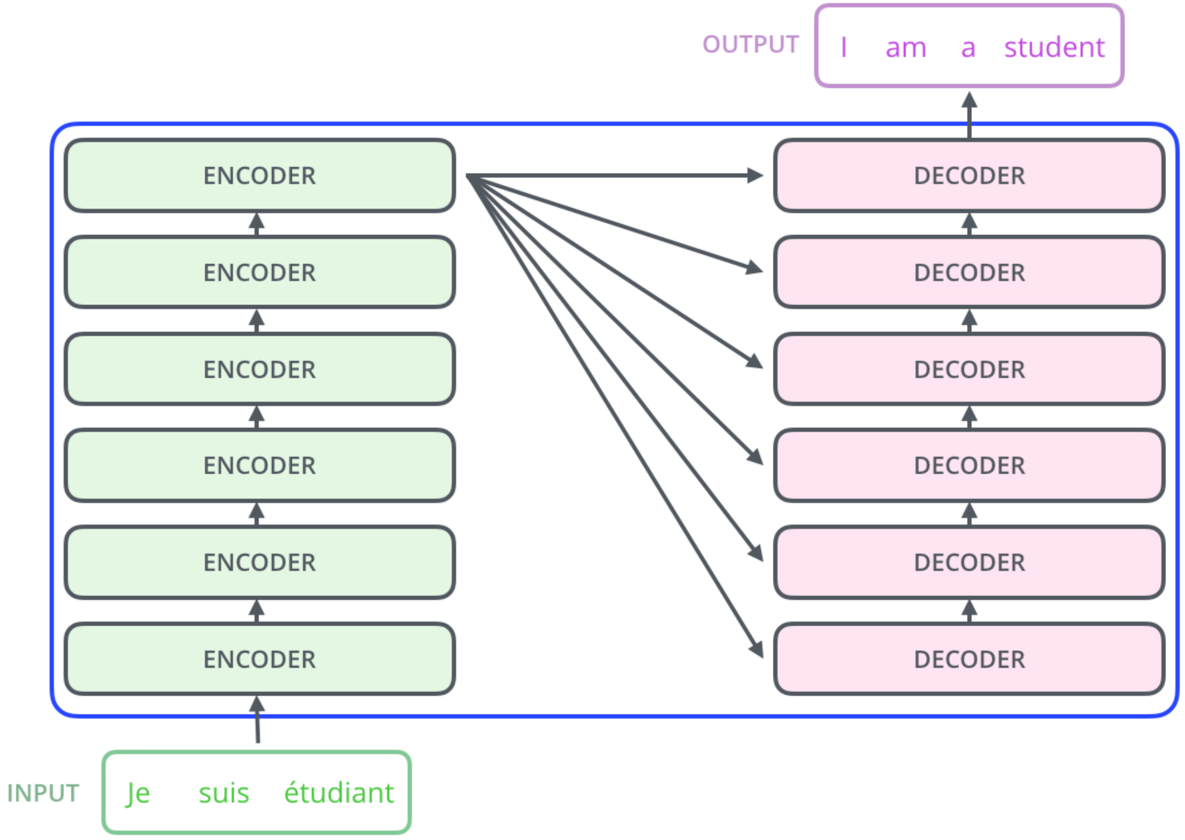
The Transformer Network

- Follows an encoder-decoder architecture bu without recurrence or convolution



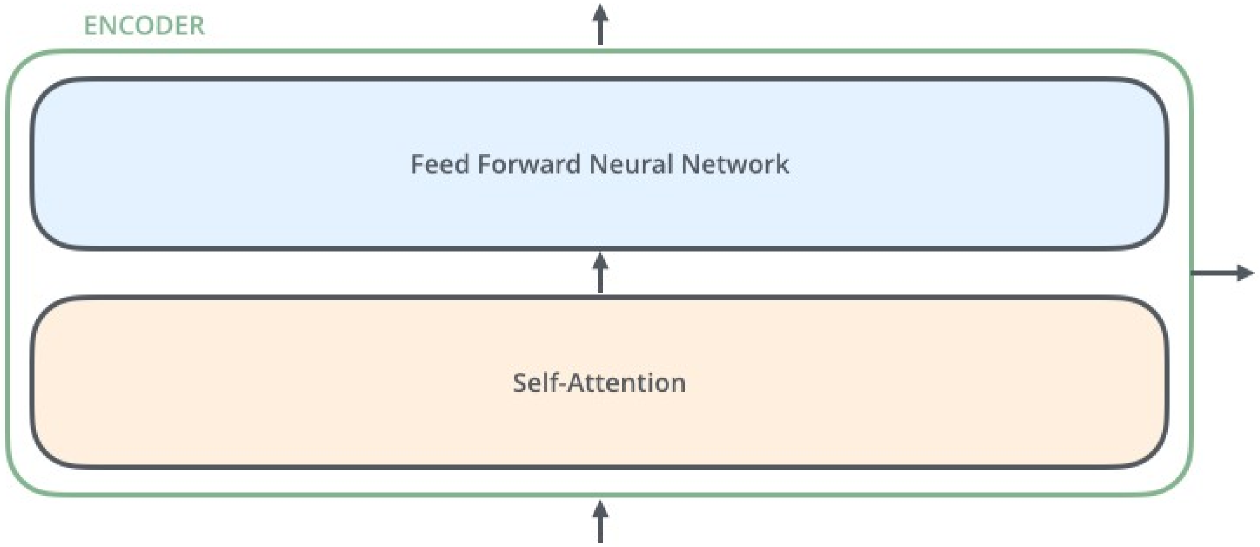
The Transformer Network

- The encoder and the decoder each consist of a fixed number of layers
- The final output of the encoder is used as input for each layer in the decoder



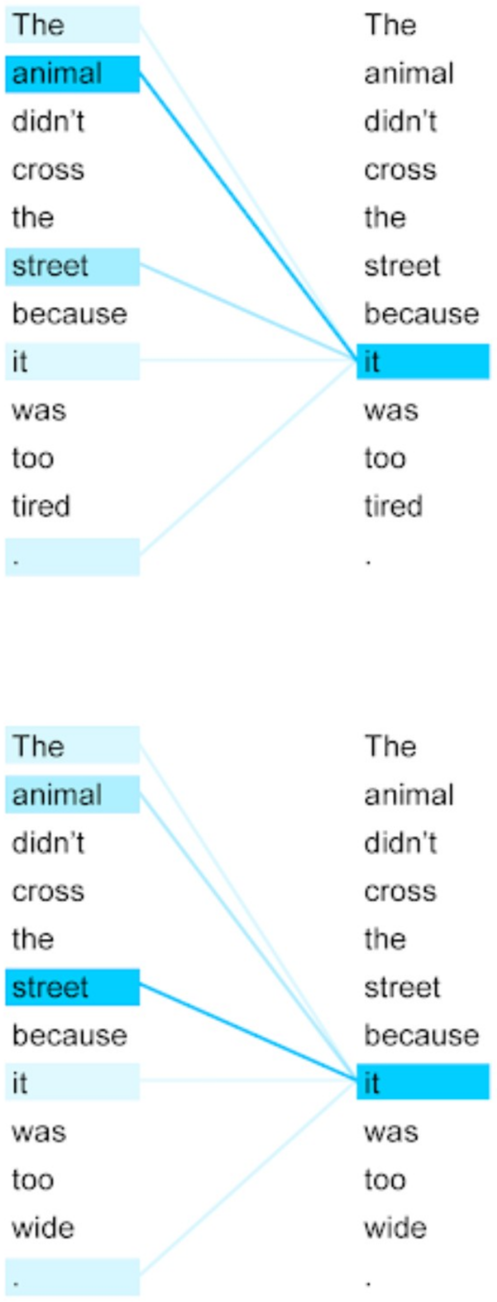
Encoder Layers

- Each layer in the encoder consists of 2 sublayers
- A Self-attention sublayer and a pointwise feedforward network



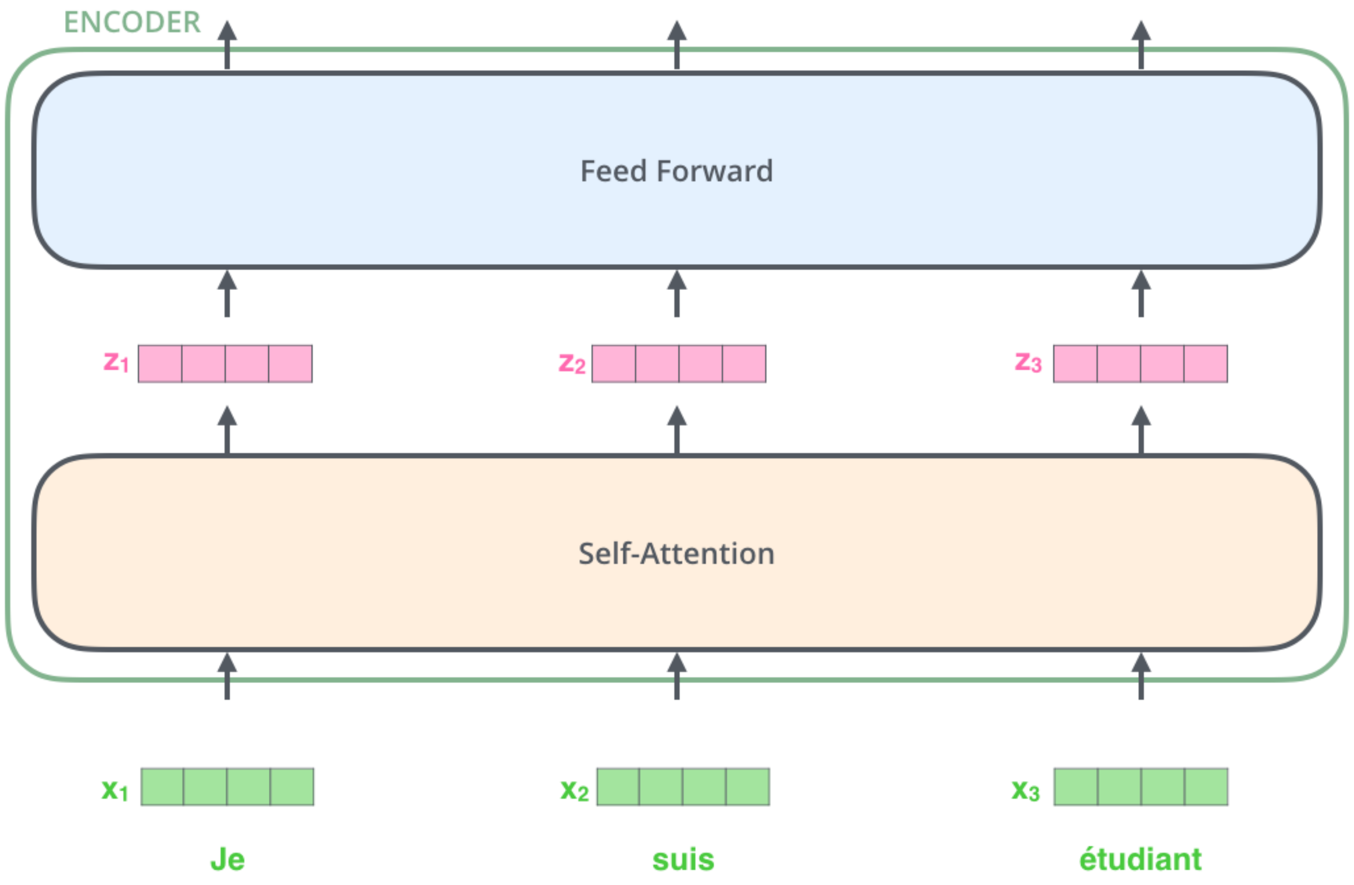
Self-Attention

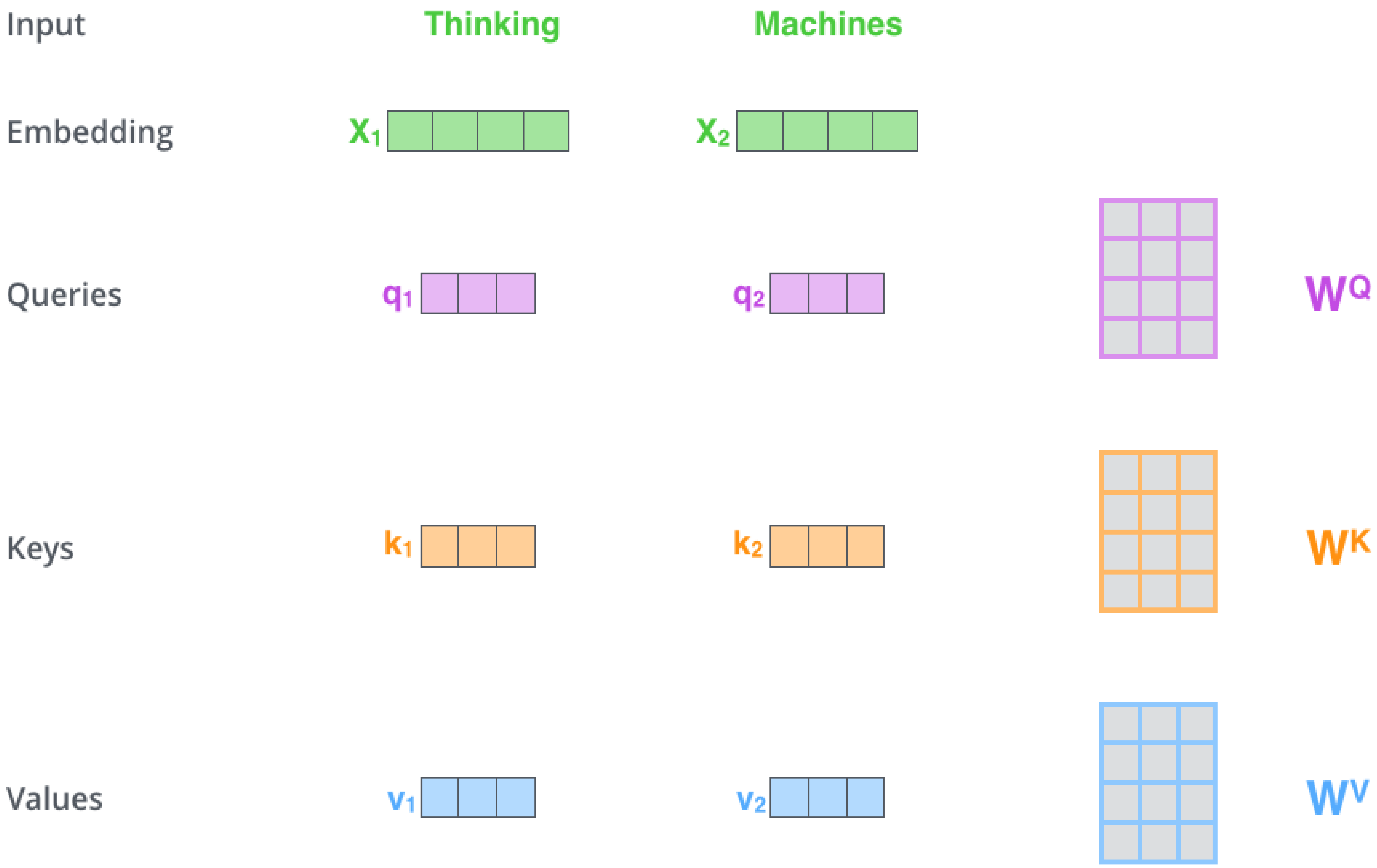
- A way of computing a representation of an input sequence by relating the elements of the input sequence with each other
- In computing the representation for a given element of a sequence, use the other the other elements in the sequence



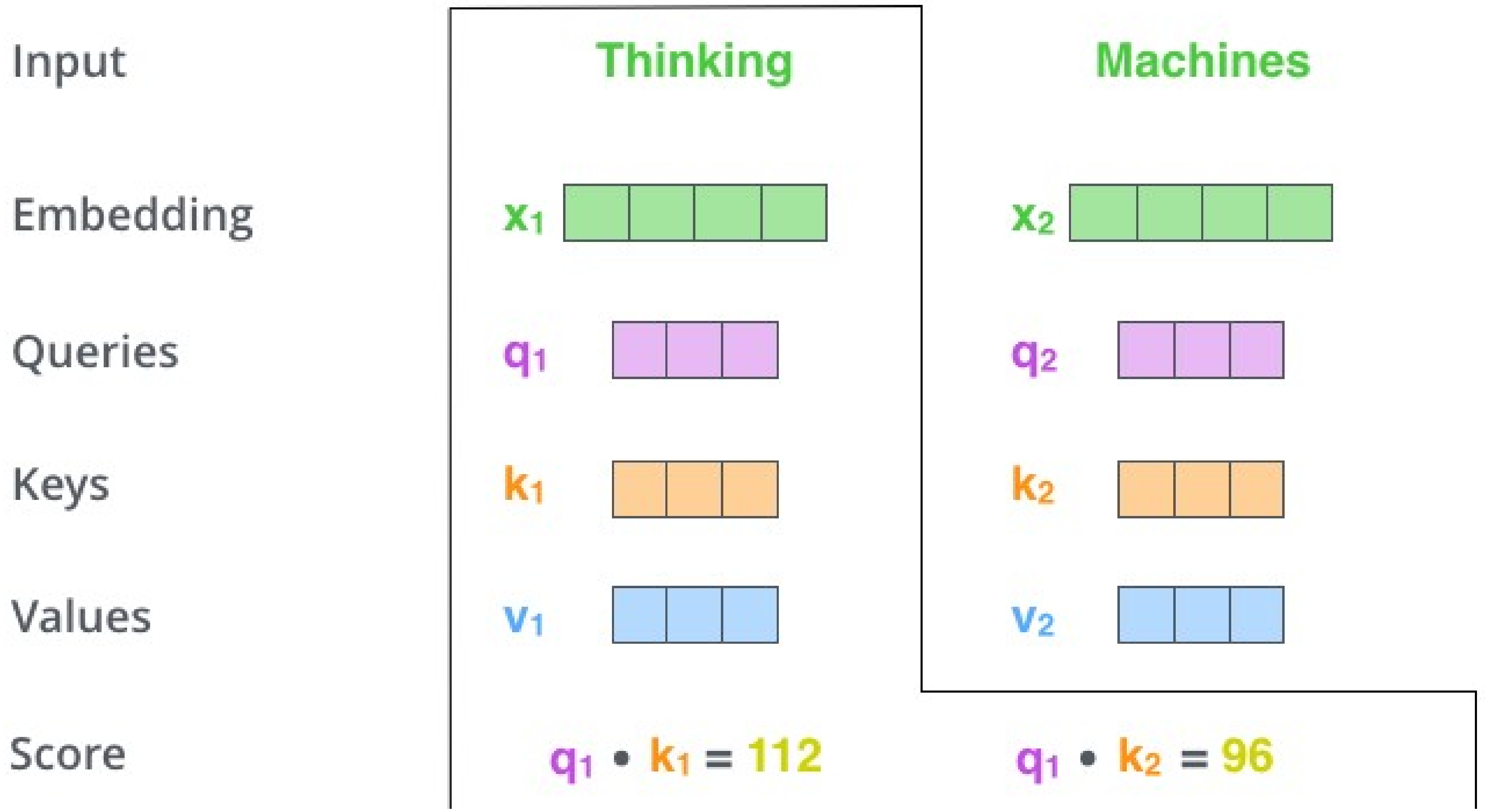
Self-Attention

- Self-attention sublayers receive a list of fixed length vectors as input, and produce output of the same dimension

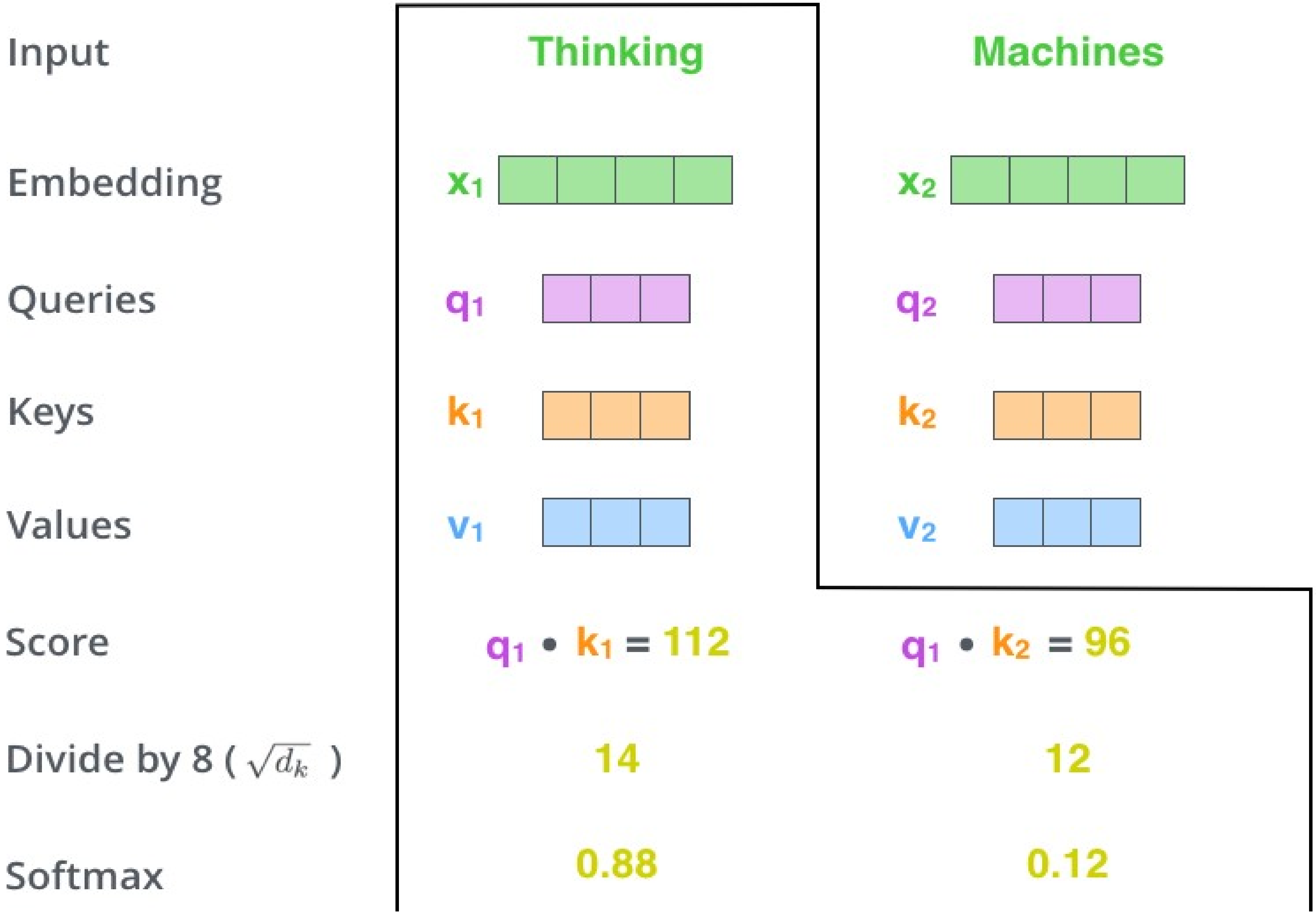


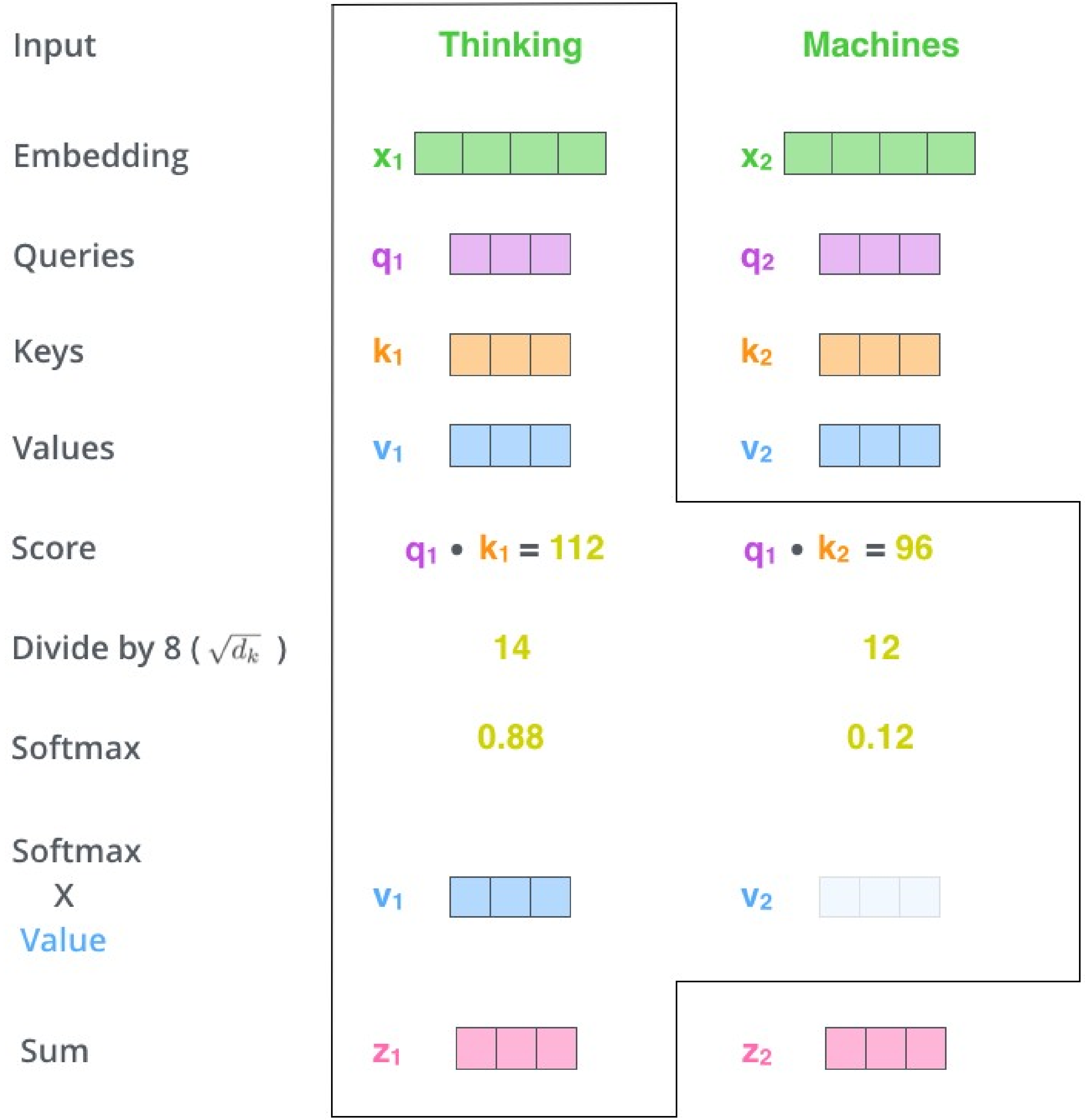


Input
Thinking
Machines
Embedding
X1
X2
Queries
q1
q2
WQ
Keys
k1
k2
WK
Values
v1
v2
WV



Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96

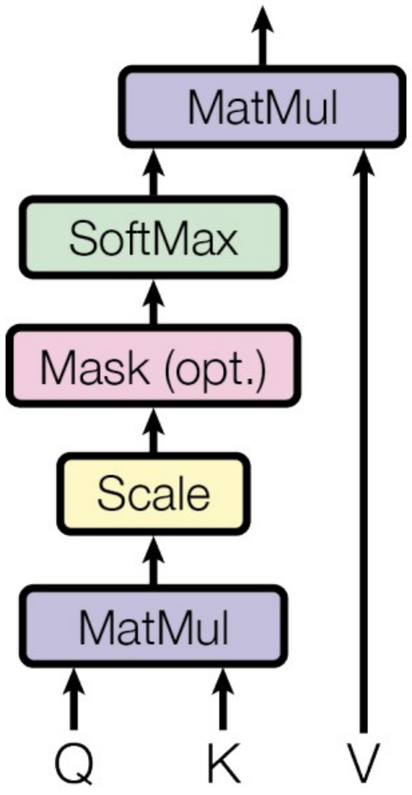

Input
Thinking
Machines
Embedding
x_1
x_2
Queries
q_1
q_2
Keys
k_1
k_2
Values
v_1
v_2
Score
$q_1 \cdot k_1 = 112$
$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12


Input
Thinking
Machines
Embedding
x_1
x_2
Queries
q_1
q_2
Keys
k_1
k_2
Values
v_1
v_2
Score
$q_1 \bullet k_1 = 112$
$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12
Softmax
X
Value
v_1
v_2
Sum
z_1
z_2

Self-Attention

- In a real implementation, we vectorize this computation

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

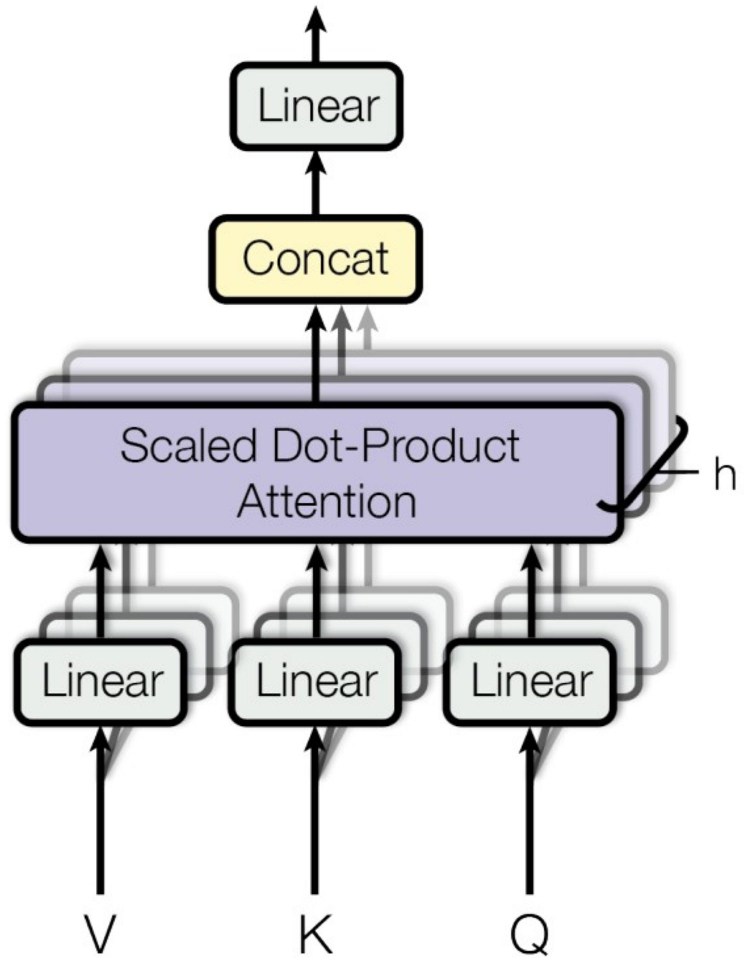


Multi-Head Self-Attention

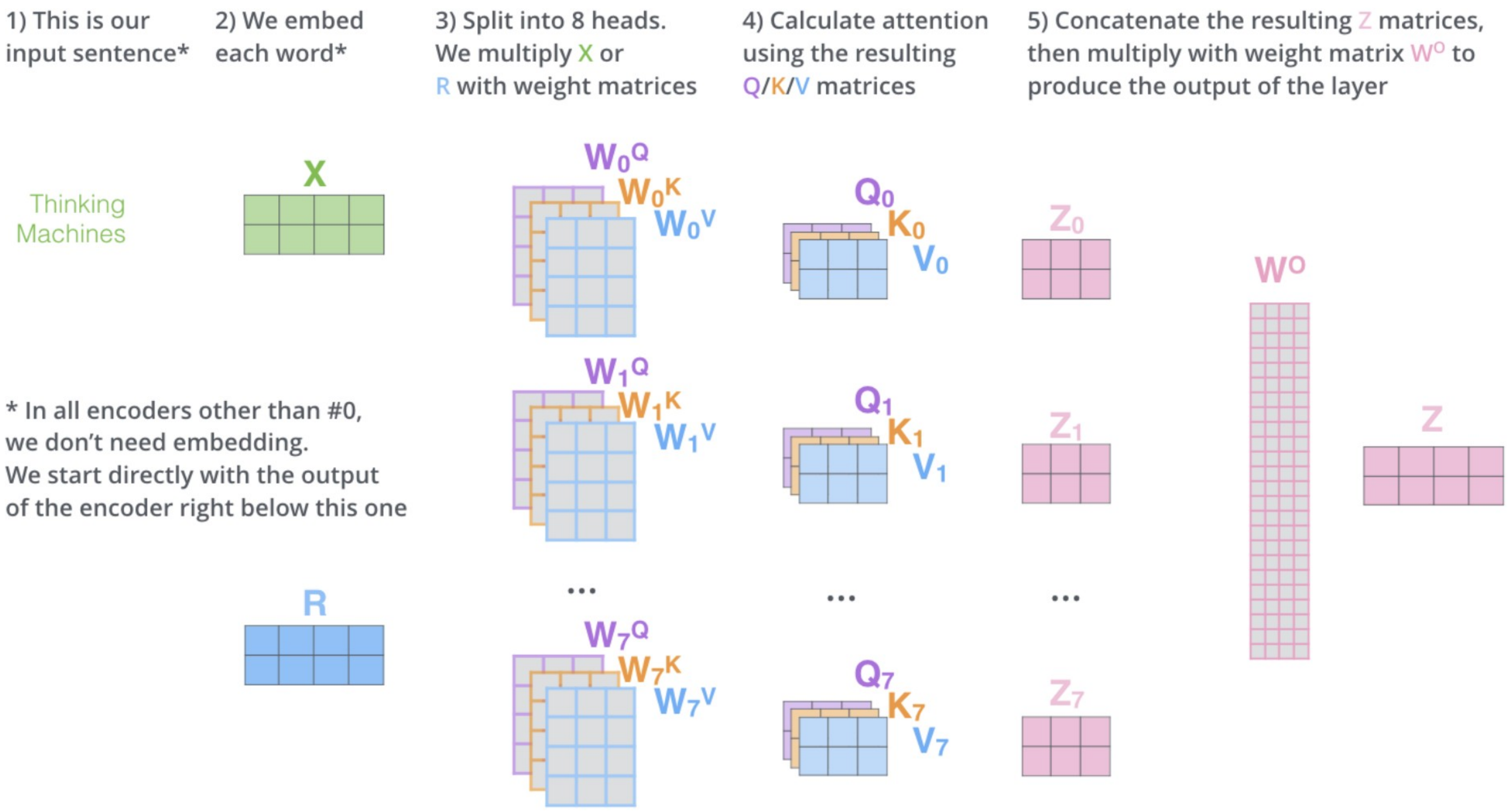
- The size of the input vectors is larger than the size of the elements of Q, K and V
- This doesn’t have to be the case, but is an architectural choice
- This results in a loss of information, but is made up for with something called multi-head attention

Multi-Head Self-Attention

- Instead of projecting the inputs only once, do it h times, where d_{model}=512 is the size of the inputs, d_k=d_v=64 is the size of the projections and h = d_{model}/d_k = 512/64 = 8

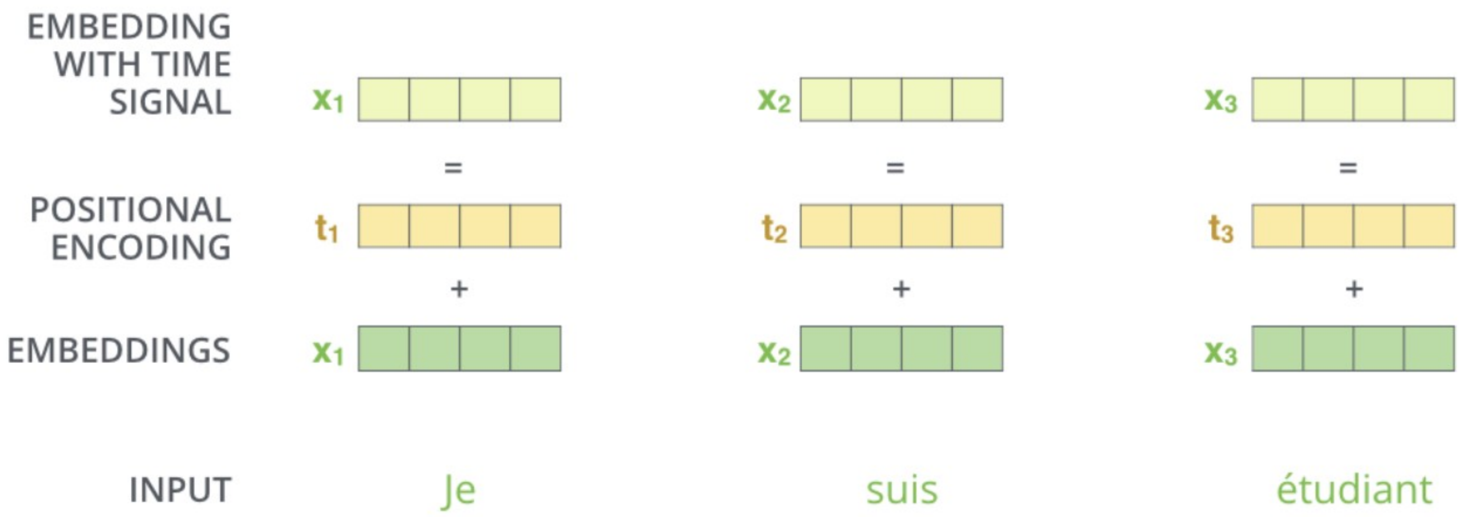


Multi-Head Self-Attention



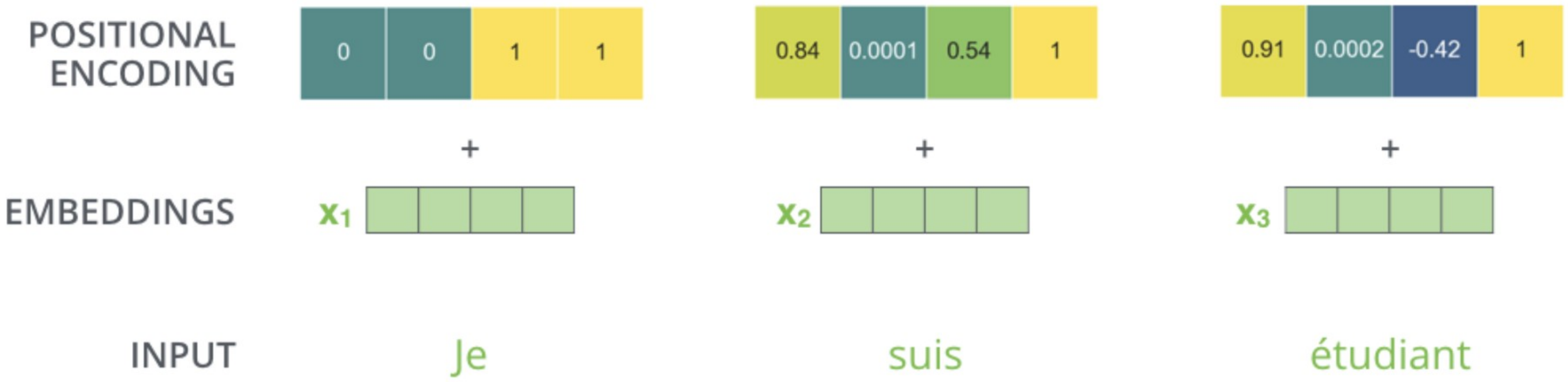
Positional Encoding

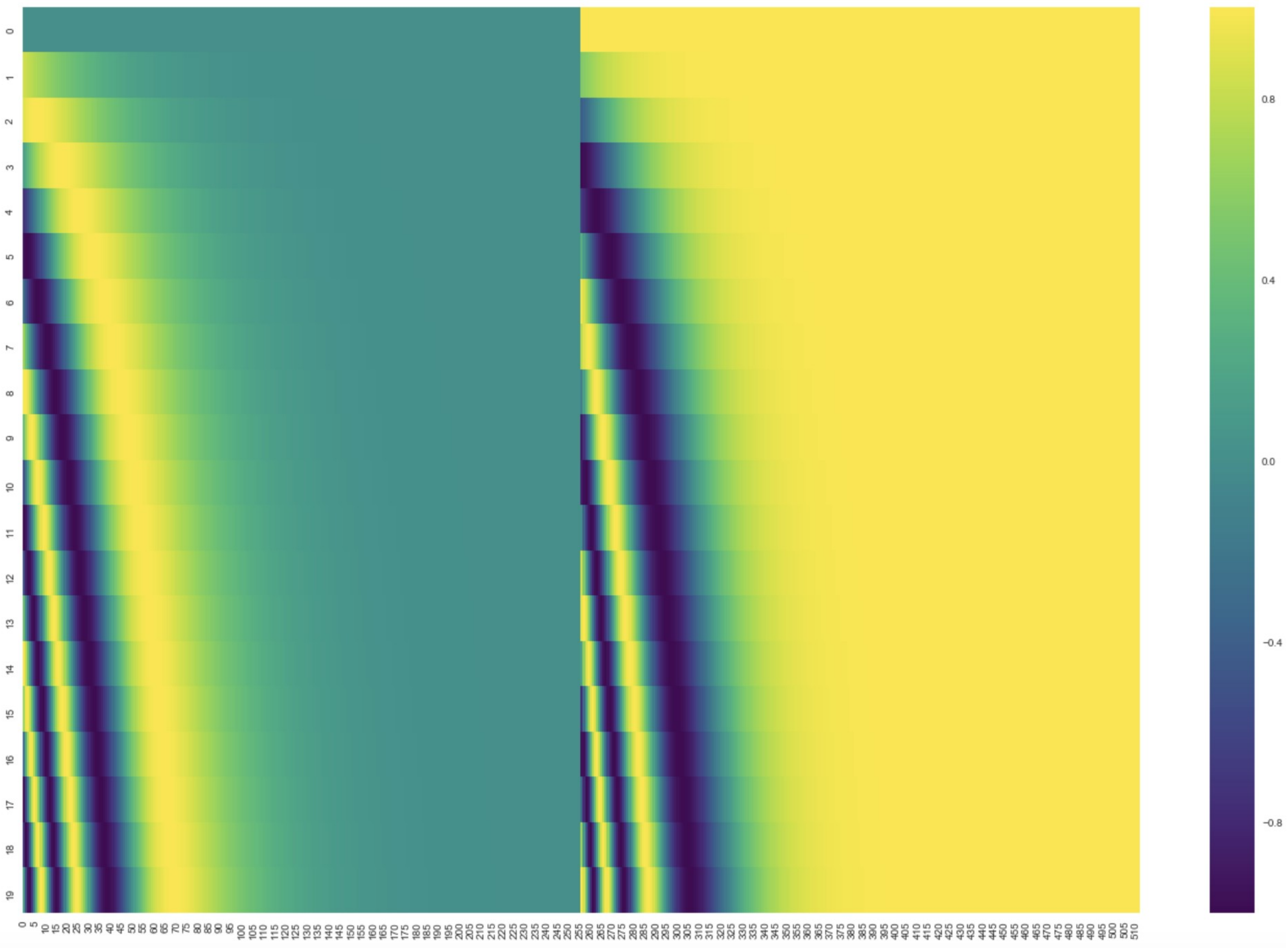
- So far we haven't seen any way for the encoder to know the order of the words
- A vector is added to each input embedding, which encodes the relative position of each token



Positional Encoding

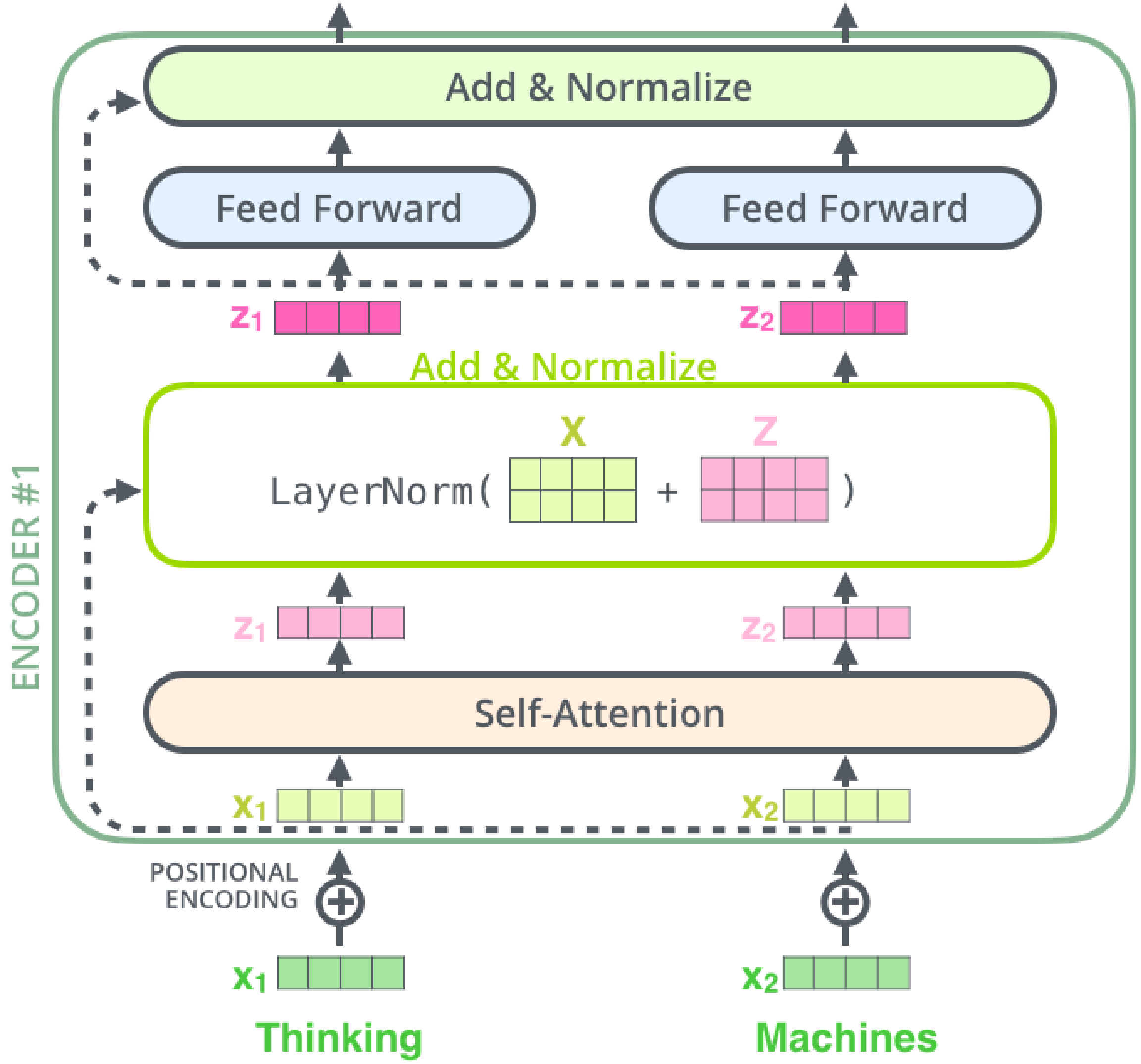
- These positional encoding vectors follow a specific pattern that the model learns and enables it to know the distance between words in the sequence




0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
0.8
0.4
0.0
−0.4
−0.8

Putting the Encoder Layer Together

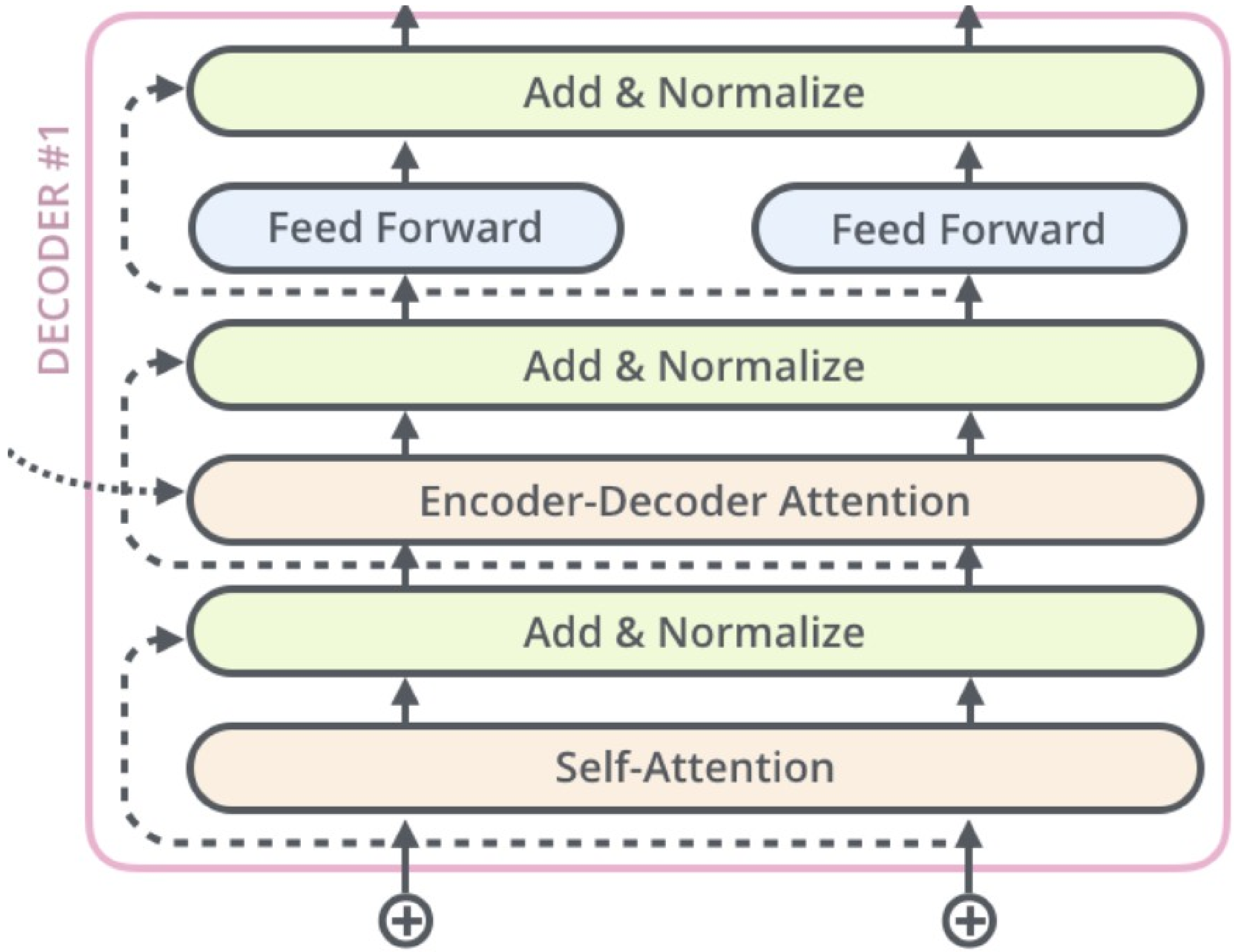
- Each sublayer is also wrapped in layer normalization and has a residual connection from its input



Decoder Layers

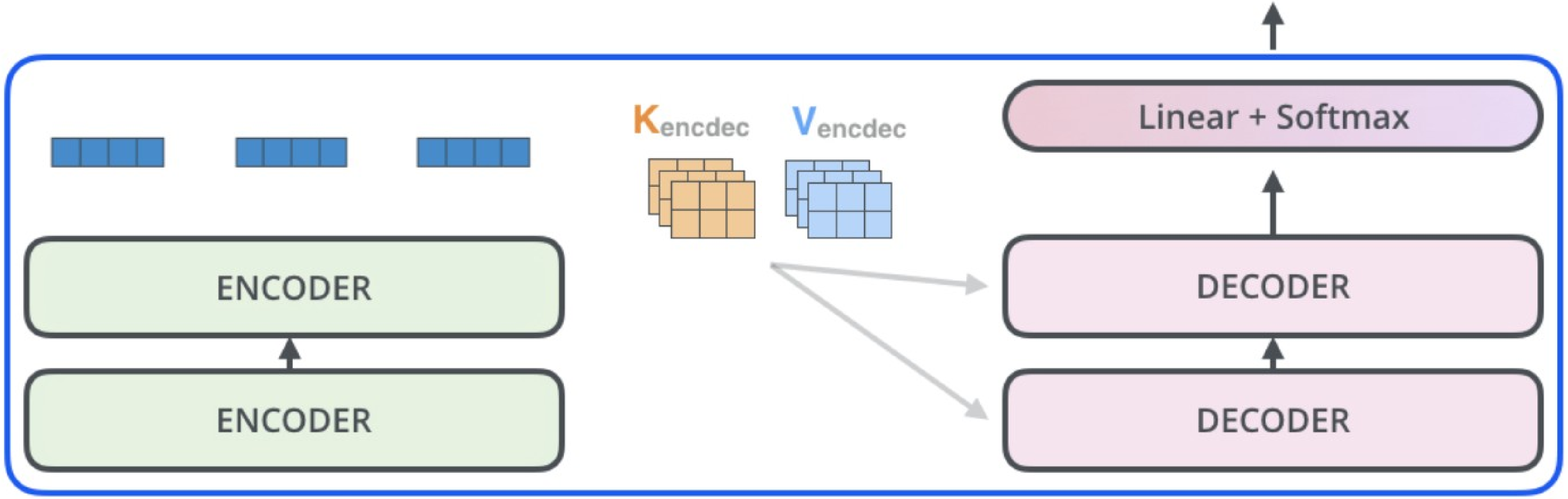
- Decoder layers are similar to encoder layers
- A 3rd sublayer is added
- Available outputs with positional encodings are provided as input

K,
V



The Decoder

- The output of the top encoder is fed into each decoder layer where it is projected as the familiar K and V
- Finally, there is a linear projection to a space of size equal to the output vocabulary, and a softmax to produce an output token



Comparing Transformer Nets to RNNs and CNNs

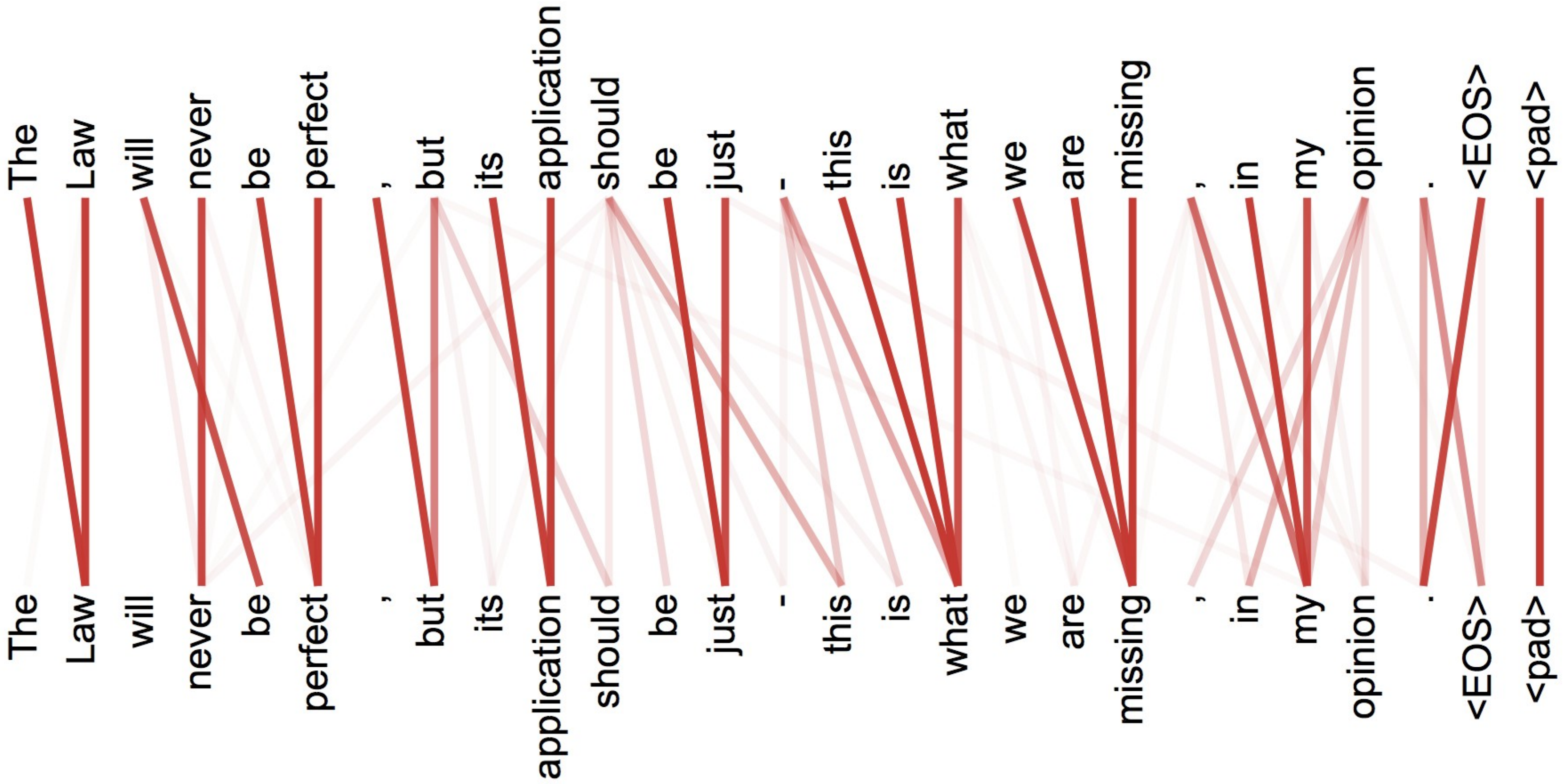
- Some relevant criteria:
  - Total computational complexity per layer
  - The amount of computation that can be parallelized, as measured by the minimum number of sequential operations required
  - The maximum path length between long-range dependences

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Discussion Questions

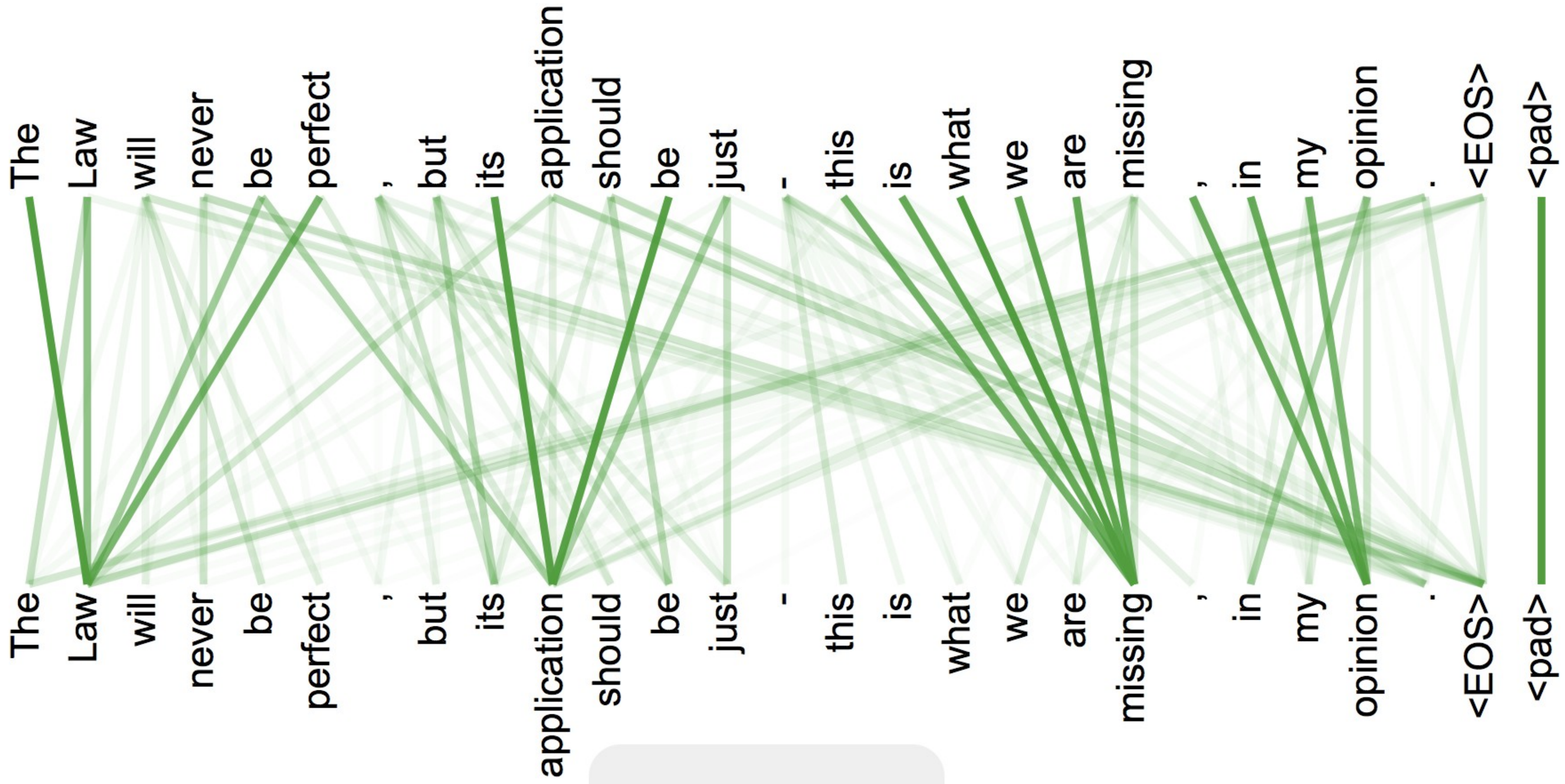
- What is the meaning of the query, key, value metaphor? Is it actually a good metaphor?
- What is the purpose of the “multi-head” part of multi-head attention?
- What is attention exactly? What principles are preserved across different kinds of attention?

Appendix



The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>

Appendix



The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>
The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>