Natural Language Processing
Lecture 09

Qun Liu, Valentin Malykh
Huawei Noah's Ark Lab

Spring 2022
A course delivered at KFU, Kazan

Content

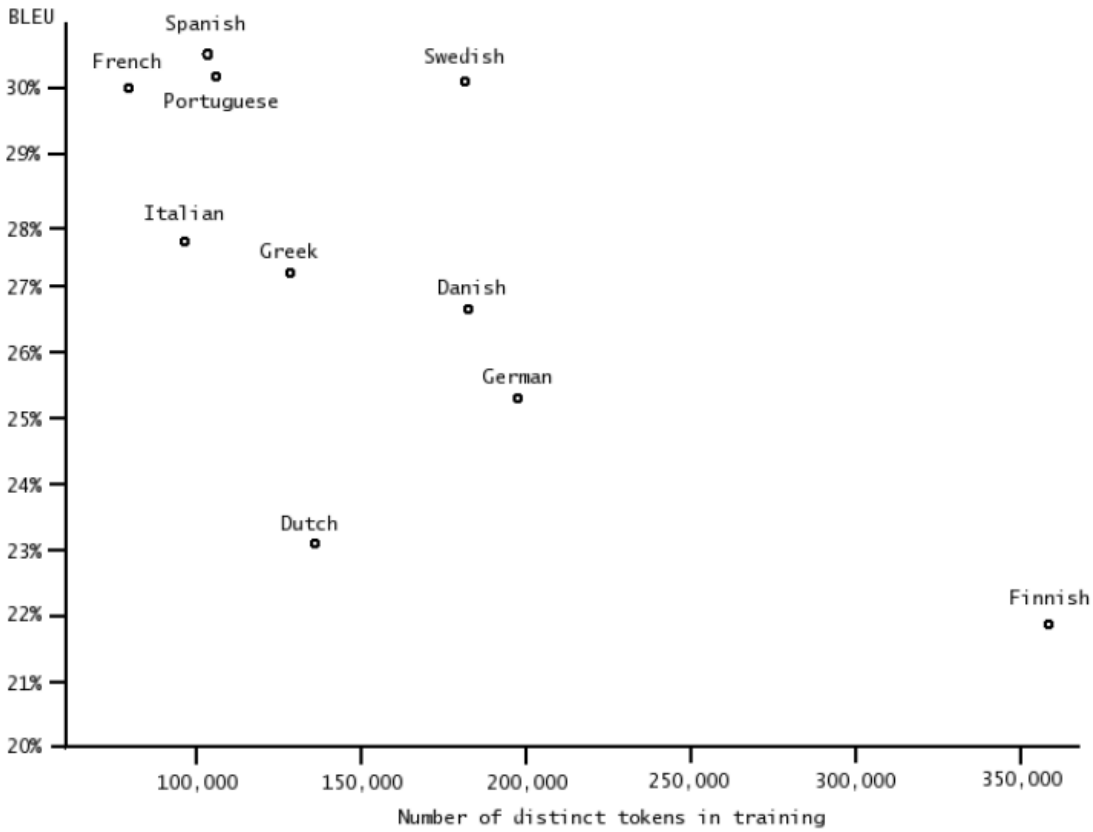
Content

Content

Open-vocabulary problem for NMT

- In NMT, due to the use of softmax in decoding, the output vocabulary is a closed set.
- Furthermore, a large size will make the decoding very slow because the softmax operation consumes huge computing resources.
- However, the vocabulary of a natural language is always an open set because new words emerge every day.
- Moreover, morphologically-rich languages such like finnish, Turkish, Arabic, etc. have very large vocabulary size compared with other languages.

Vocabulary Size vs. MT Performance



Vocabulary size vs. BLEU score when translating into English (which has about 65,000 distinct word forms) for SMT
Philipp Koehn, Europarl: A Parallel Corpus for Statistical Machine Translation, MT Summit 2005

Replace OOV words with the UNK symbol

- Practically, the vocabulary size of a NMT system is around 500k.
- In early NMT systems, all the out-of-vocabulary (OOV) words are replaced with a UNK symble (means UNKNOWN):
  - UNKs in source sentences will make the system not differentiat the rare words;
  - UNKs in the target sentences will make the system unreadable.

Content

Solutions to open-vocabulary problem

- Subword-level models
  - Byte-Pair Encoding (BPE)
  - Word Piece or Sentence Piece
- Character-level models

Bype-Pair Encoding (BPE)

- Originated from a compression algorithm
  - Replace a most frequent byte pair with a new byte
- BPE for NMT was proposed in 2016 and become very popular, not only in NMT but also in many other NN-based NLP tasks

> Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. ACL 2016.

Bype-Pair Encoding (BPE)

- Initialize **vocabulary** of subwords with all the basic characters
- Maintain a **dictionary** with all words and their frequencies calculated from the **corpus**
- Segment all the words in the **dictionary** into characters
- Repeat until the **vocabulary** size reaches a predefined limit:
  - Select the subword bigram with the highest frequency
  - Merge that bigram into a new subword and add it to the **vocabulary**
  - Update the segmented **dictionary** by replacing all the occurrence of that bigram with the newly added subword

BPE: build the vocabulary

- The dictionary:

l o w</w>	5
l o w e r</w>	2
n e w e s t</w>	6
w i d e s t</w>	3

- The vocabulary:

l o w</w> w e r</w> n s t</w> i d

Note: 'w</w>','r</w>' and 't</w>' are single characters which are different from 'w', 'r' and 't'.

BPE: build the vocabulary

- The dictionary:

l o w</w>	5
l o w e r</w>	2
n e w **es** t</w>	6
w i d **es** t</w>	3

- The vocabulary:

l o w</w> w e r</w> n s t</w> i d **es**

Note: 'w</w>','r</w>' and 't</w>' are single characters which are different from 'w', 'r' and 't'.

BPE: build the vocabulary

- The dictionary:

l o w</w>	5
l o w e r</w>	2
n e w **est**</w>	6
w i d **est**</w>	3

- The vocabulary:

l o w</w> w e r</w> n s t</w> i d es **est</w>**

Note: 'w</w>','r</w>' and 't</w>' are single characters which are different from 'w', 'r' and 't'.

BPE: build the vocabulary

- The dictionary:

lo w</w>	5
lo w e r</w>	2
n e w est</w>	6
w i d est</w>	3

- The vocabulary:

l o w</w> w e r</w> n s t</w> i d es est</w> **lo**

Note: 'w</w>','r</w>' and 't</w>' are single characters which are different from 'w', 'r' and 't'.

More about BPE

- Do deterministic longest piece segmentation of words
- Segmentation is only within words after tokenization
- Advantages:
  - Automatically build vocabulary from corpus
  - Language agnostic
  - Unsupervised
  - Trade off between text length and vocabulary size
- A trick:
  - treat 't' and 't</w>' as different characters to keep the word boundary information

WordPiece and SentencePiece

- An alternative solution for subword segmentation
- Proposed by Google in 2012:

> Schuster, Mike, and Kaisuke Nakajima. Japanese and korean voice search. ICASSP 2012.

- Instead of merging the bigram with highest frequence in BPE, WordPiece merge the bigram to maximizing the language model log likelihood of the corpus.
- SentencePiece is similar with WordPiece but applied on raw texts without tokenization, while whitespace is treated as a special character (_).

Subword-level NMT

- Subword-level NMT is almost the same as word-level NMT except for the preprocessing process for subword segmentation and the postprocessing process for subword combination.
- The subword combination is simple because the word boundary information is kept in the words.
- Subword-level NMT solves the open-vocabulary problem very well.
- Subword-level NMT outperforms word-level NMT significantly.
- Subword segmentation and combination has become a standard technique for NMT.

Content

RNN-based NMT Recap

- RNN-based NMT obtained great success:
  - Sequence-to-sequence model
  - RNN encoder and RNN decoder
  - Attention between target and source
  - Subword or character level encoding

Advantages of NMT

NMT provided a brand new paradigm for machine translation, which demonstrated huge advantages over previous approaches:

- NMT is a single model which is trained as a whole (end-to-end training), while an SMT system has many components each of which is trained against a separate object function.
- The translation quality of NMT are much better than that of SMT, especially in terms of fluency.
- NMT is good at learning from huge amount of data.
- Subword or character level NMT provides an elegant mechanism to deal with morphologically rich languages.

Improvement and extension of NMT

Research of NMT is exploding in recent years and great progress has been made:

- Transformer-based NMT
- Convolution-based NMT
- Multilingual NMT
- Multimodal NMT
- Unsupervised NMT

Transformer-based NMT has replaced RNN-based NMT and become a new state-of-the-art approach.

Transformer-based NMT

- Proposed by Google in 2017:

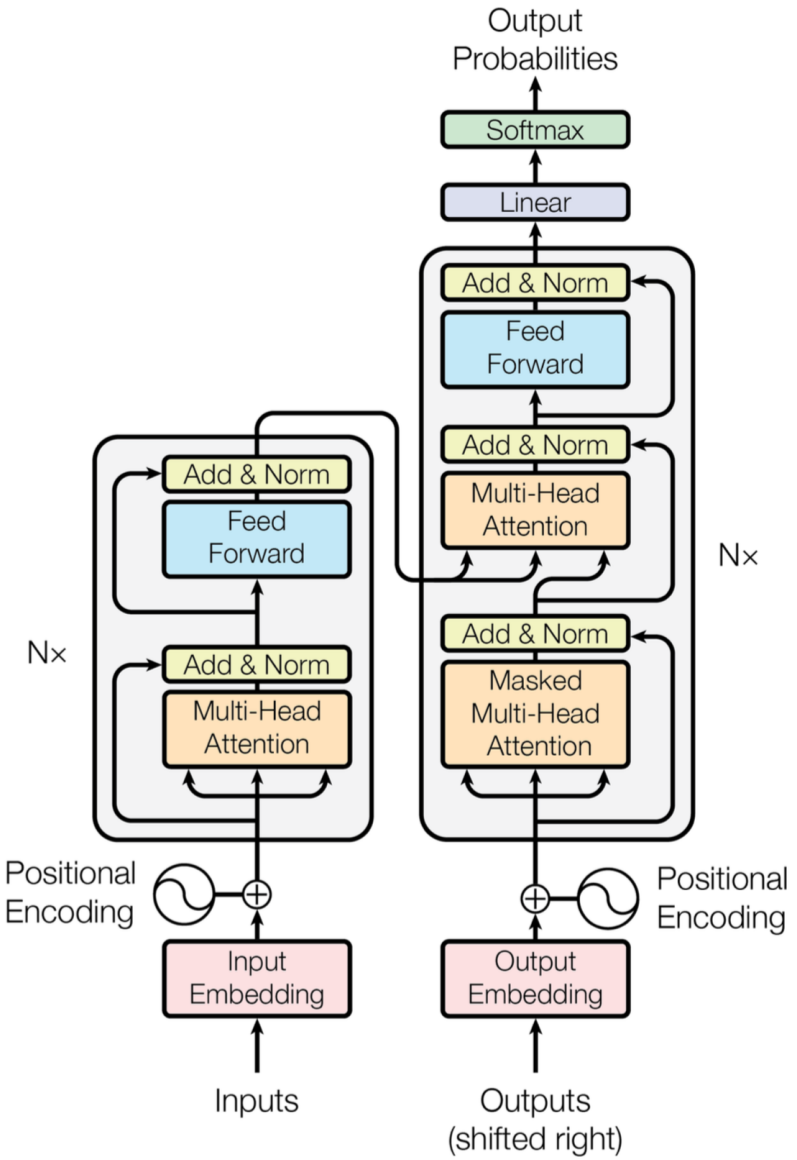
> Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. NIPS 2017.

- Transformer is a new neural network architecture based soly on an attention mechanism, dispensing with recurrence and convolutions entirely.
- Transformer-based NMT is the state-of-the-art of MT technologies.

Transformer NMT architecture

- Transformer NMT adopts a sequence-to-sequenct architecture.
- In this section, we use figures from Jay Alammar's blog to illustrate the transformer NMT.

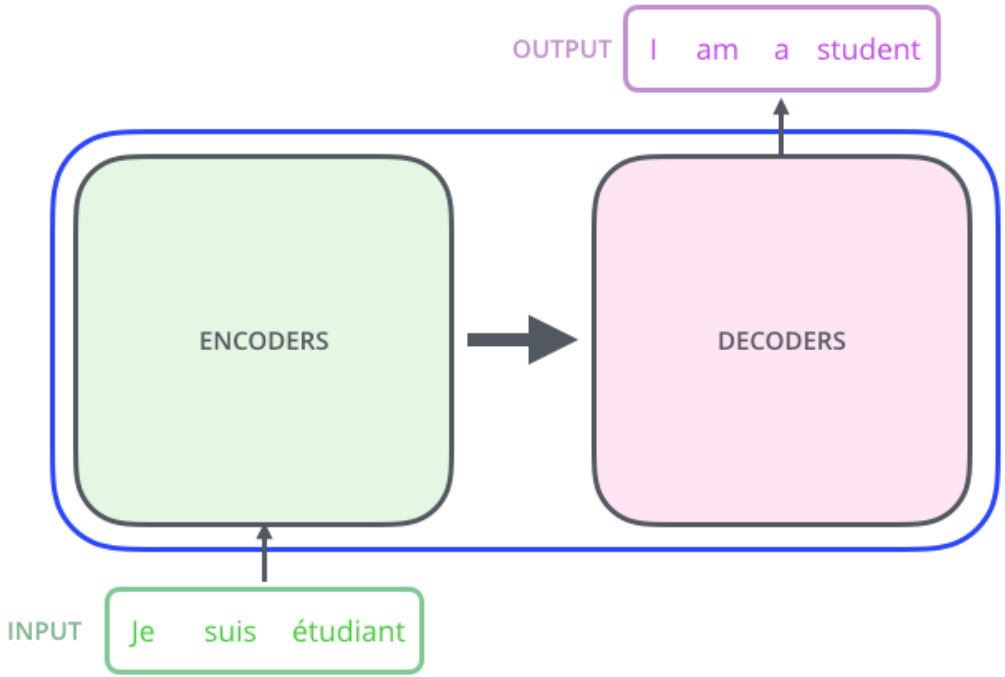
  `http://jalammar.github.io/illustrated-transformer/`



Content

A High-Level Look

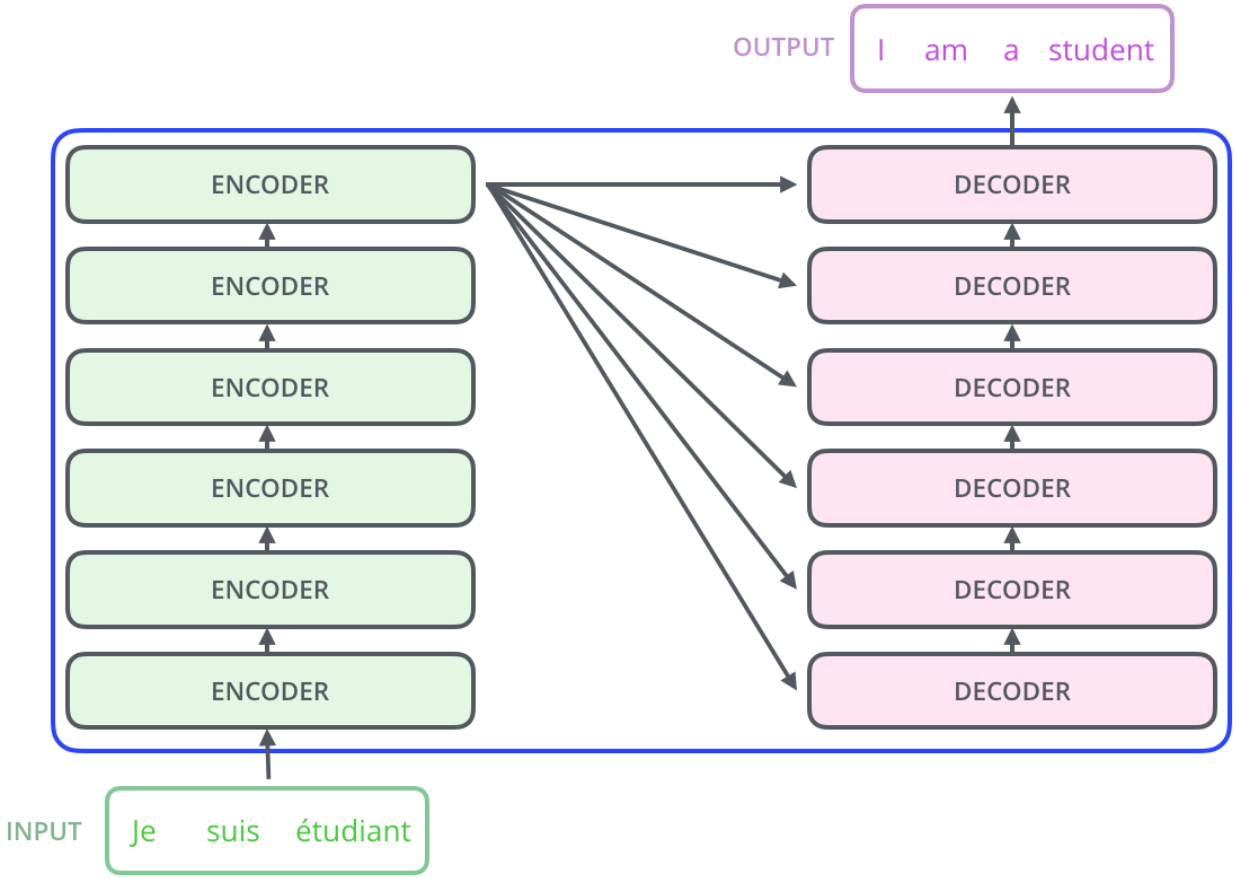
Transformer NMT is a seq2seq model.



A High-Level Look

The encoding component is a stack of encoders (6 in this paper).

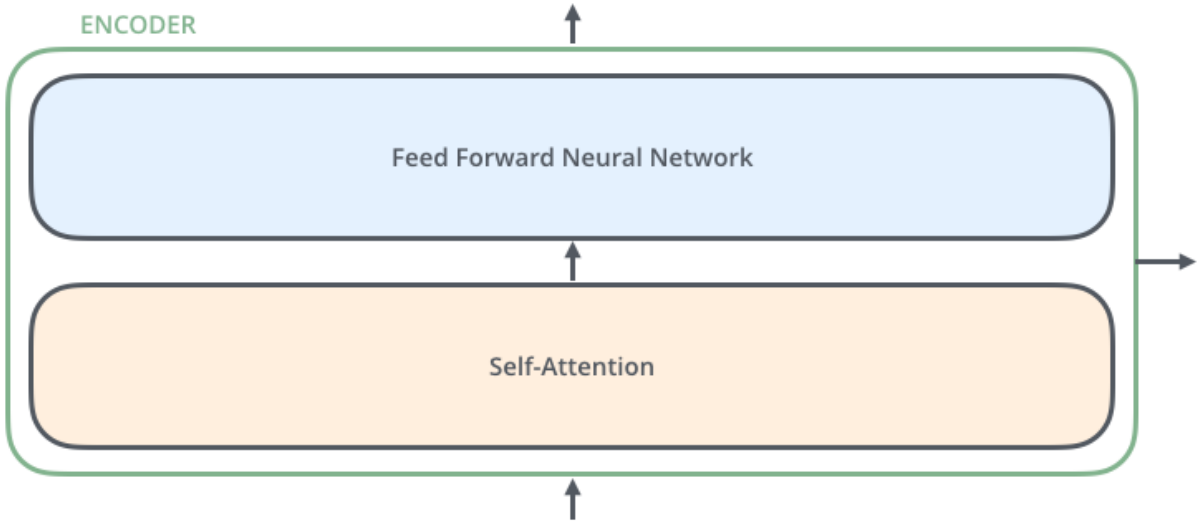
The decoding component is also a stack of decoders of the same number.

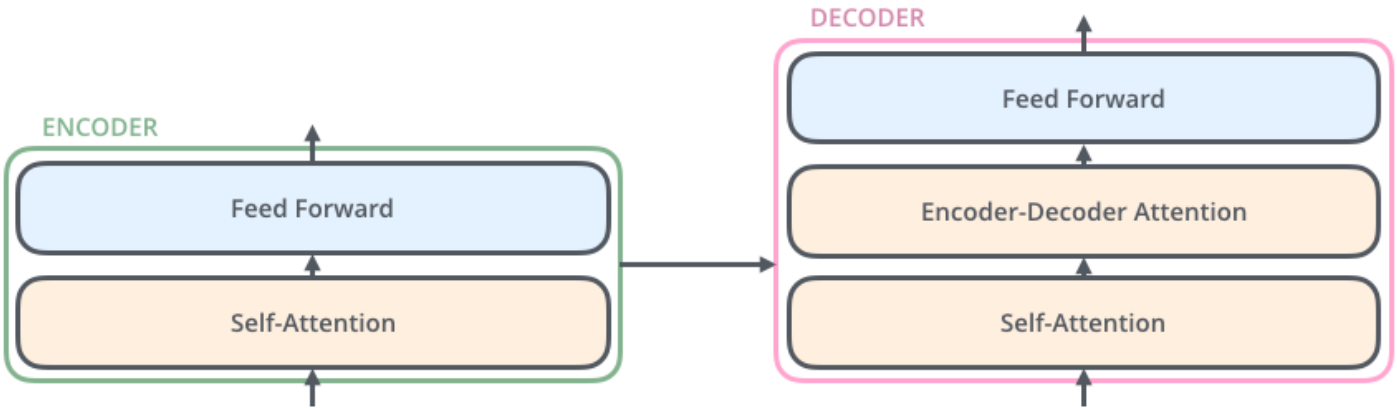


A High-Level Look

An encoder has two parts: FFNN and Self-Attention.

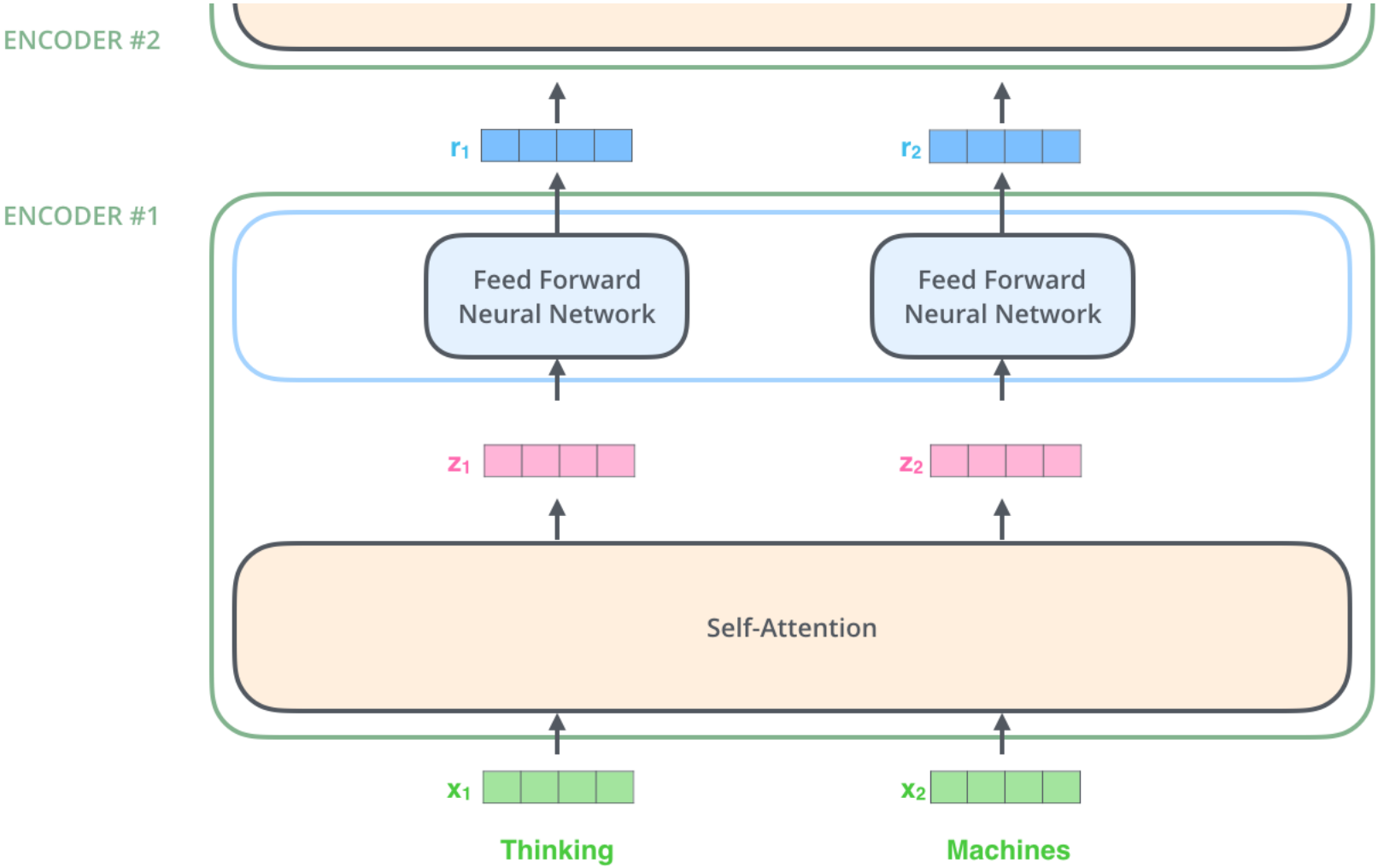
An decoder has one more part: Enc-Dec Attention.





Content

Self-Attention

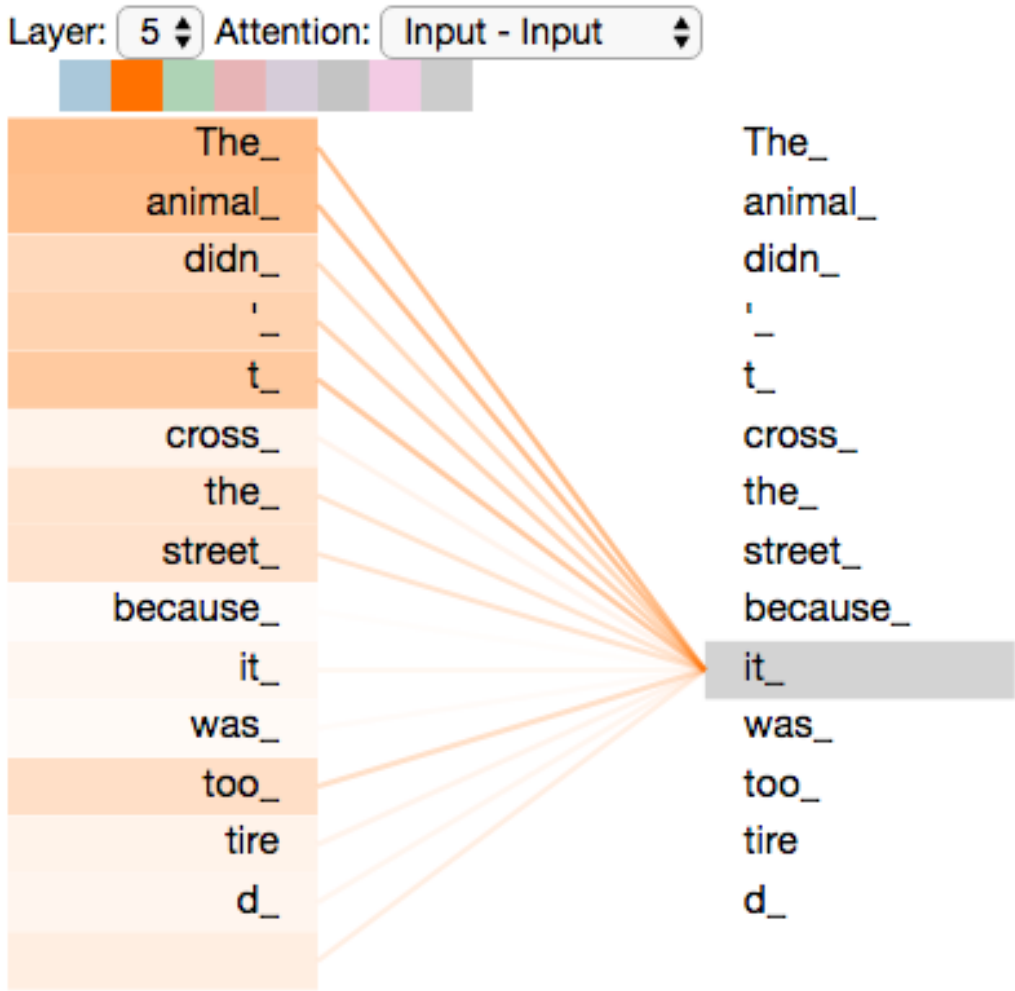


Self-attention: Visualization

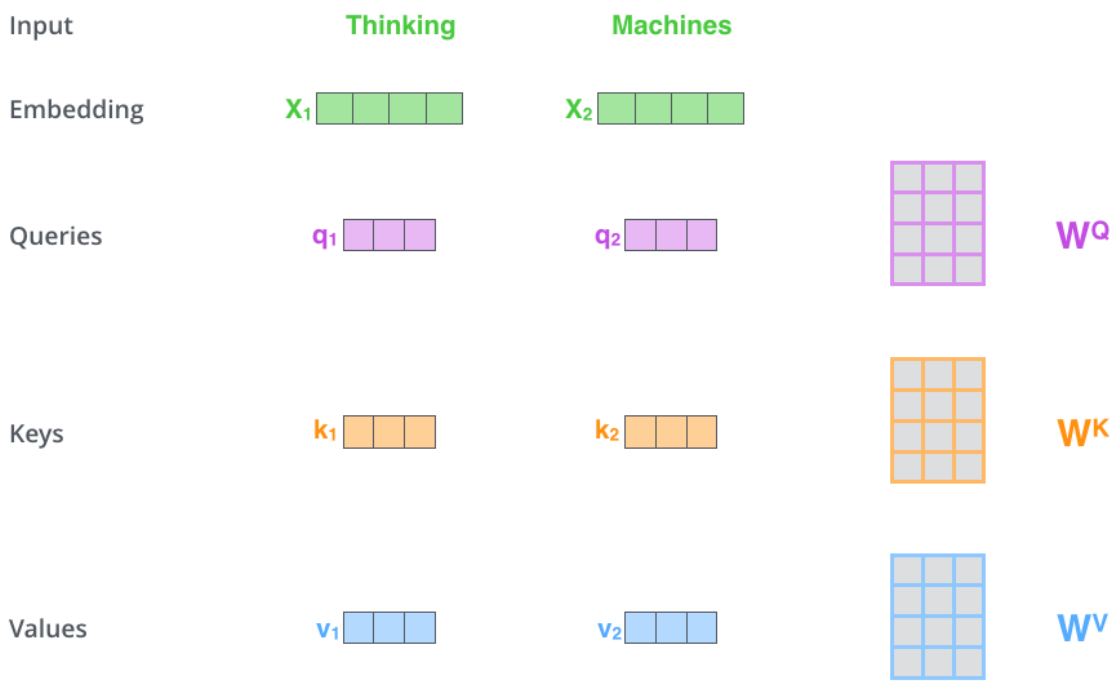
Example: `The animal didn't cross the street because it was too tired`

Associate `it` with `animal`

Look for clues when encoding

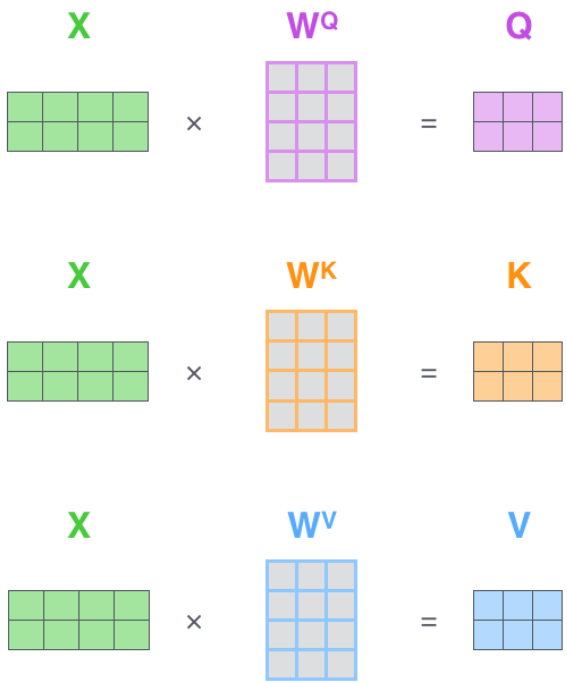


Self-attention: step 1 (create K,Q,V vectors)



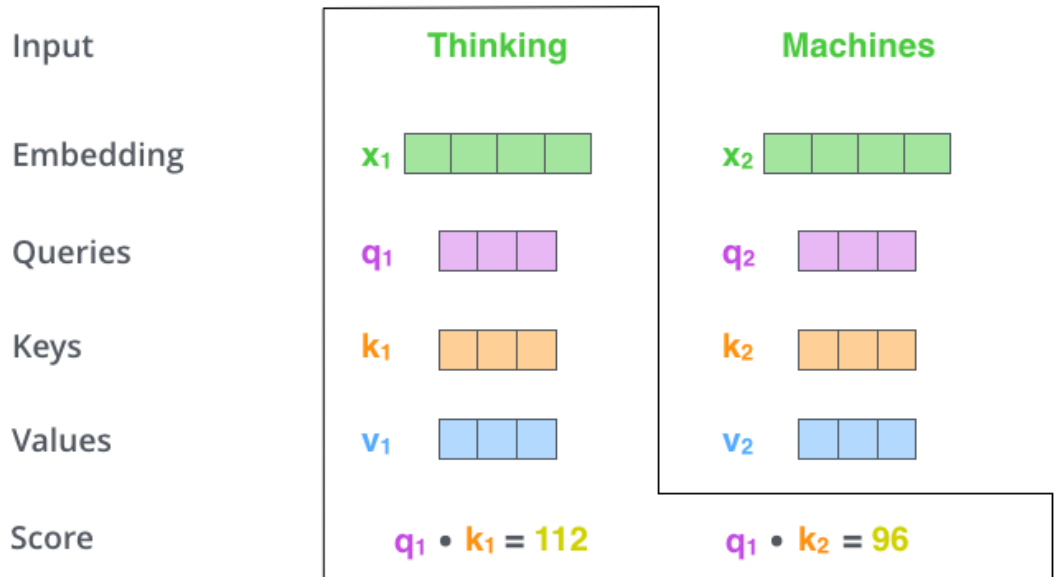
Multiplying x_1 by the W^Q weight matrix produces q_1, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

Self-attention: the calculation of K,Q,V vectors

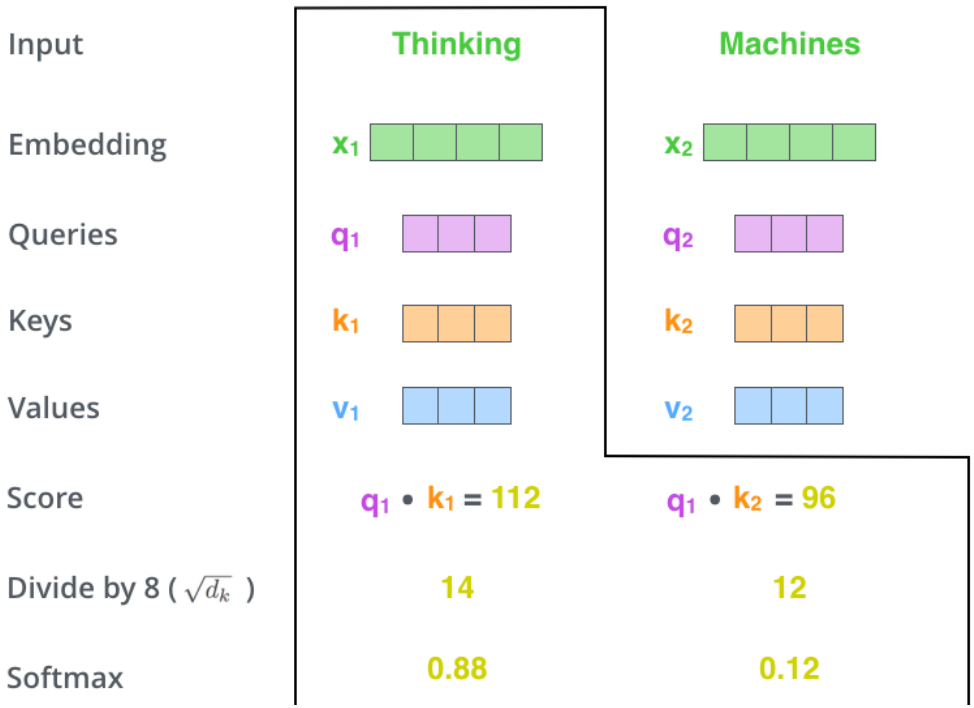


Every row in the X matrix corresponds to a word in the input sentence. We can see the difference in size of the embedding vector (4 boxes in the figure), and the $q/k/v$ vectors (3 boxes in the figure)

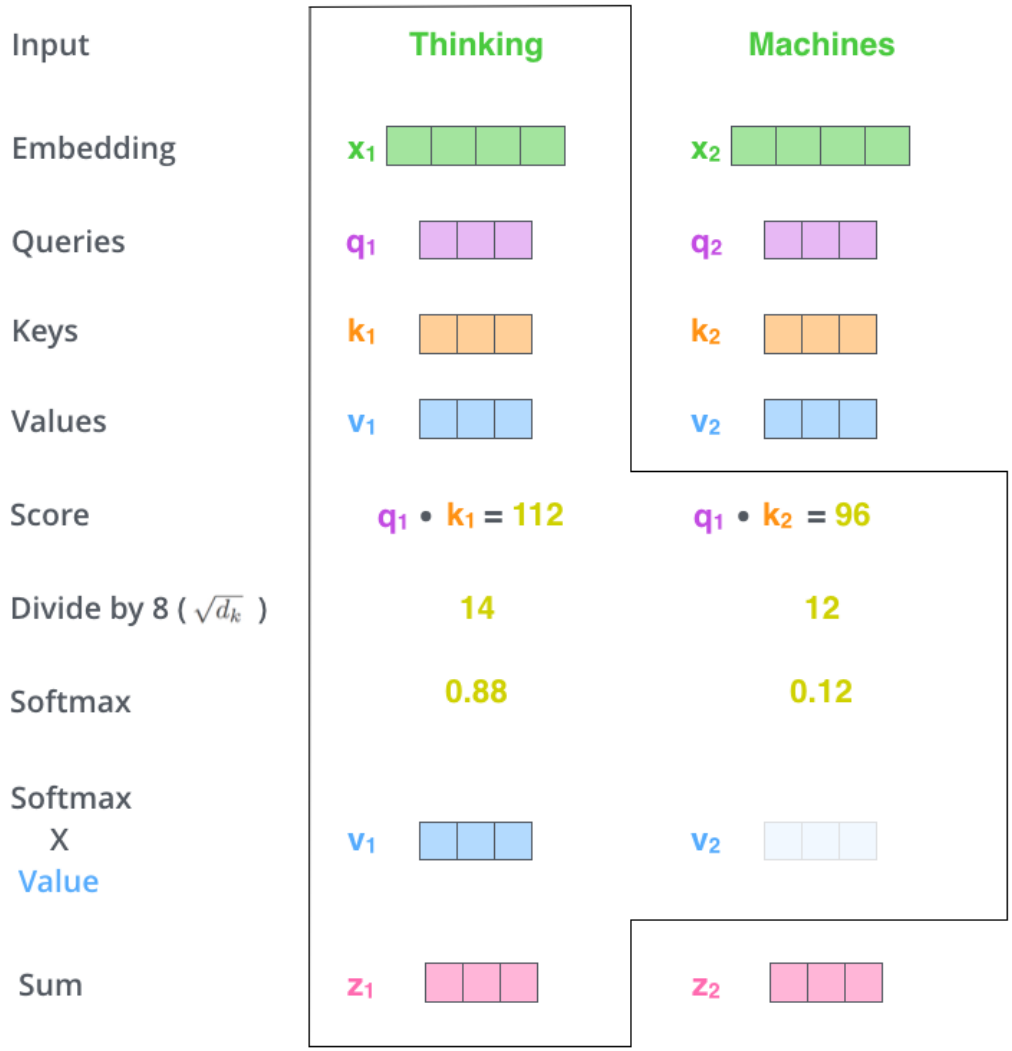
Self-attention: step 2 (calculate self-attention scores)



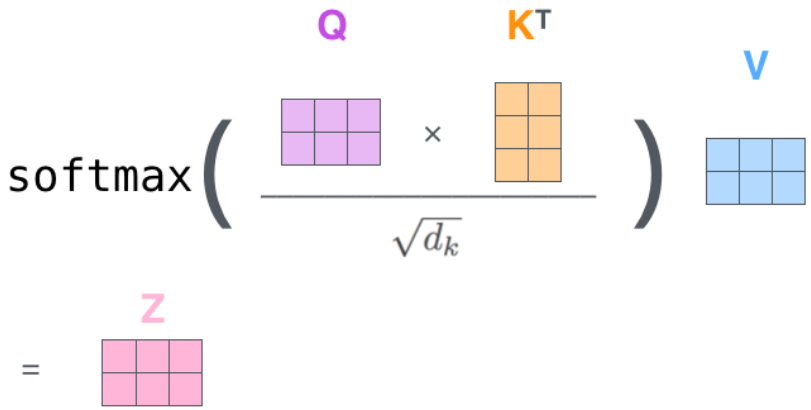
Self-attention: step 3 & 4 (normalize self-attention scores with softmax)



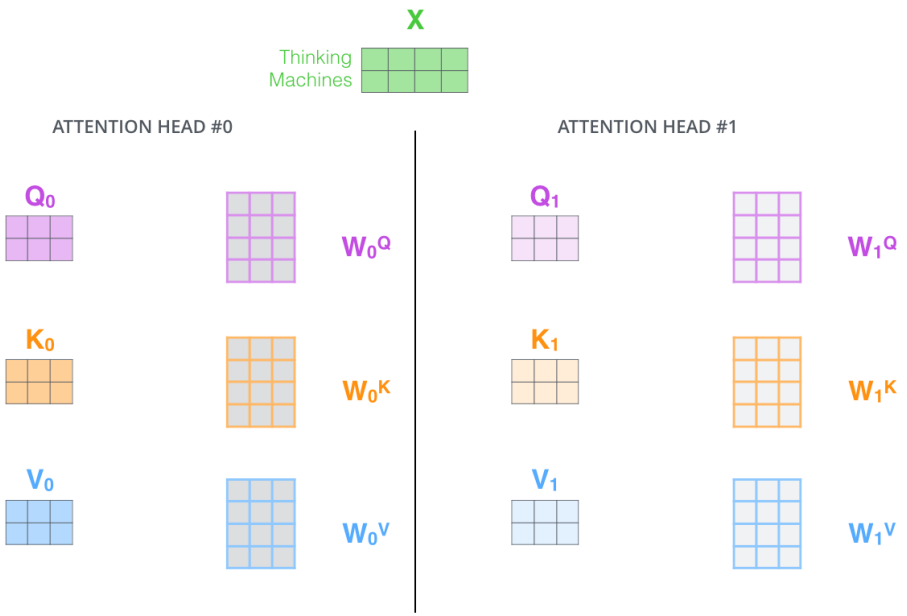
Self-attention: step 2 - 6 (weighted-sum of values from attented words)



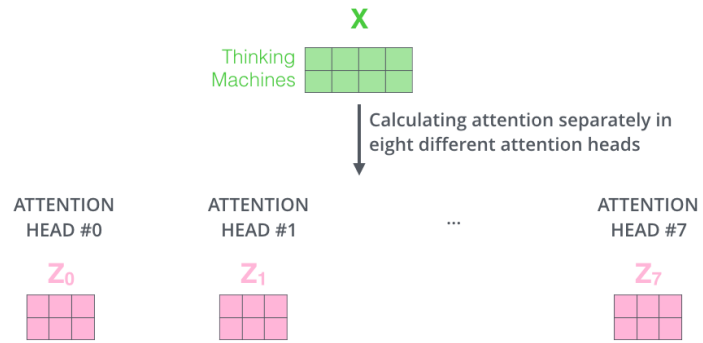
Self-attention: The self-attention calculation in matrix form



Multi-head attention



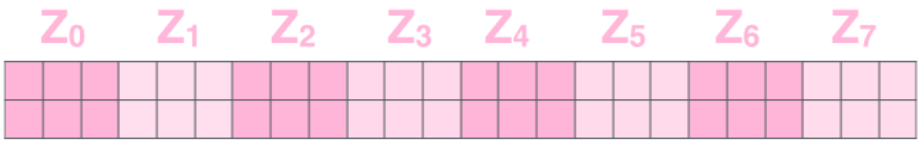
With multi-headed attention, we maintain separate $Q/K/V$ weight matrices for each head resulting in different $Q/K/V$ matrices.



We do the same self-attention calculation 8 times with different weight matrices, then we get 8 different Z matrices.

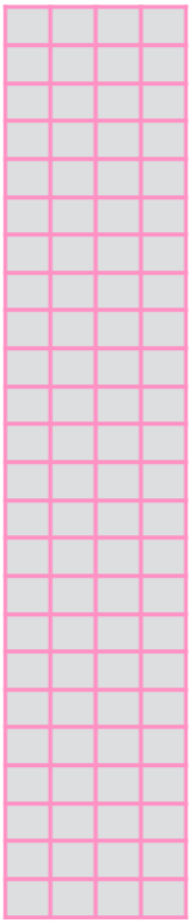
Multi-head attention

1) Concatenate all the attention heads

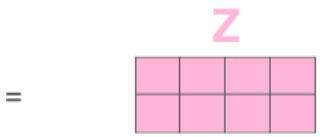


2) Multiply with a weight matrix W^O that was trained jointly with the model

X



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



Overall picture of multi-head self-attention

1) This is our input sentence*

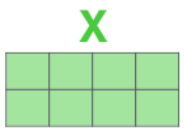
2) We embed each word*

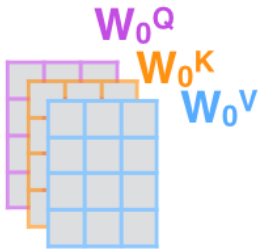
3) Split into 8 heads. We multiply X or R with weight matrices

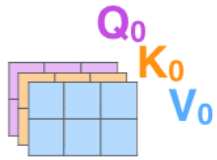
4) Calculate attention using the resulting Q/K/V matrices

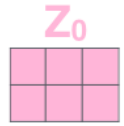
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

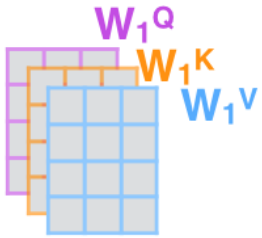
Thinking Machines

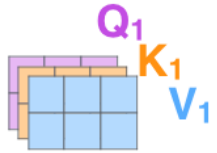


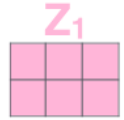




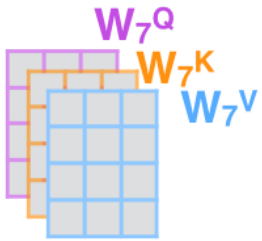


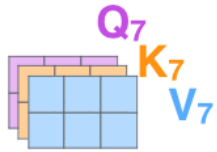


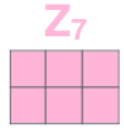


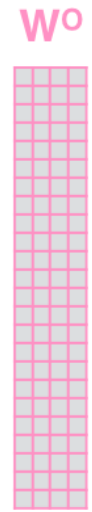


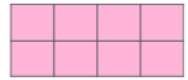
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one





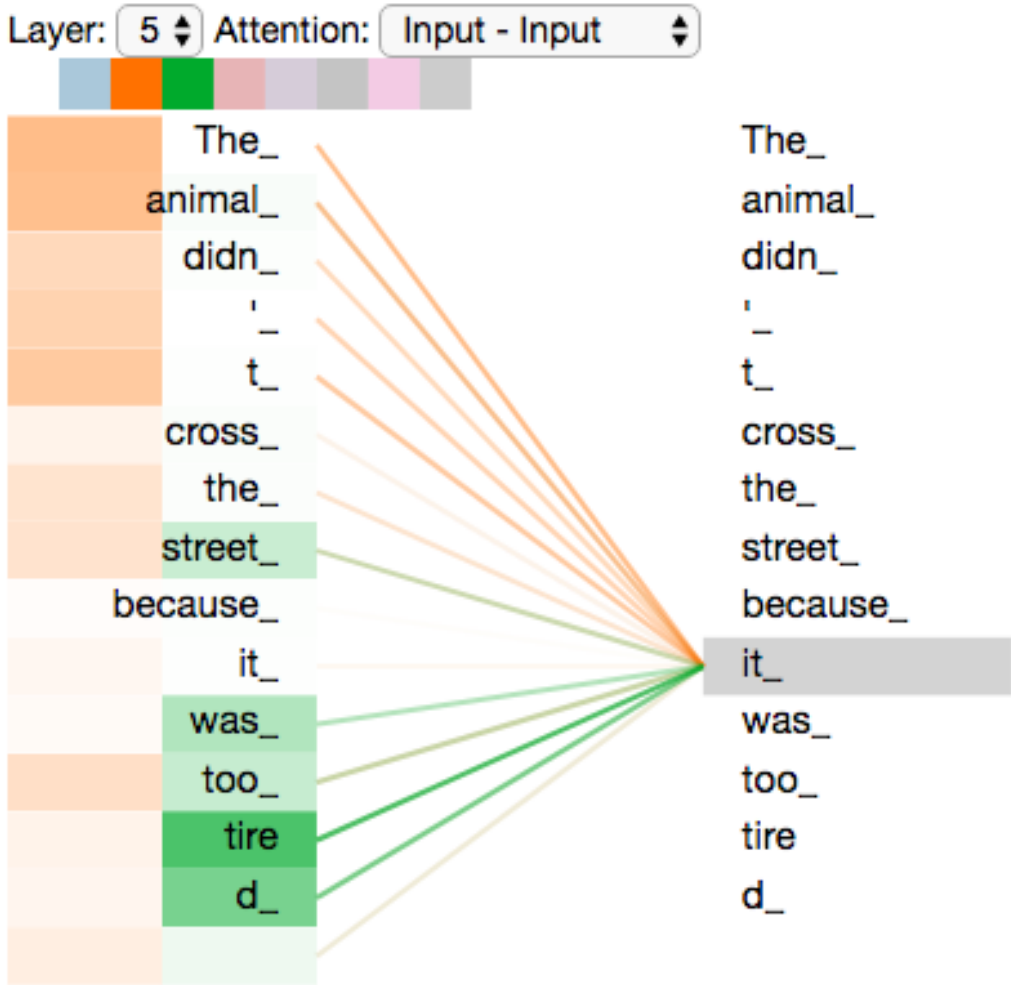






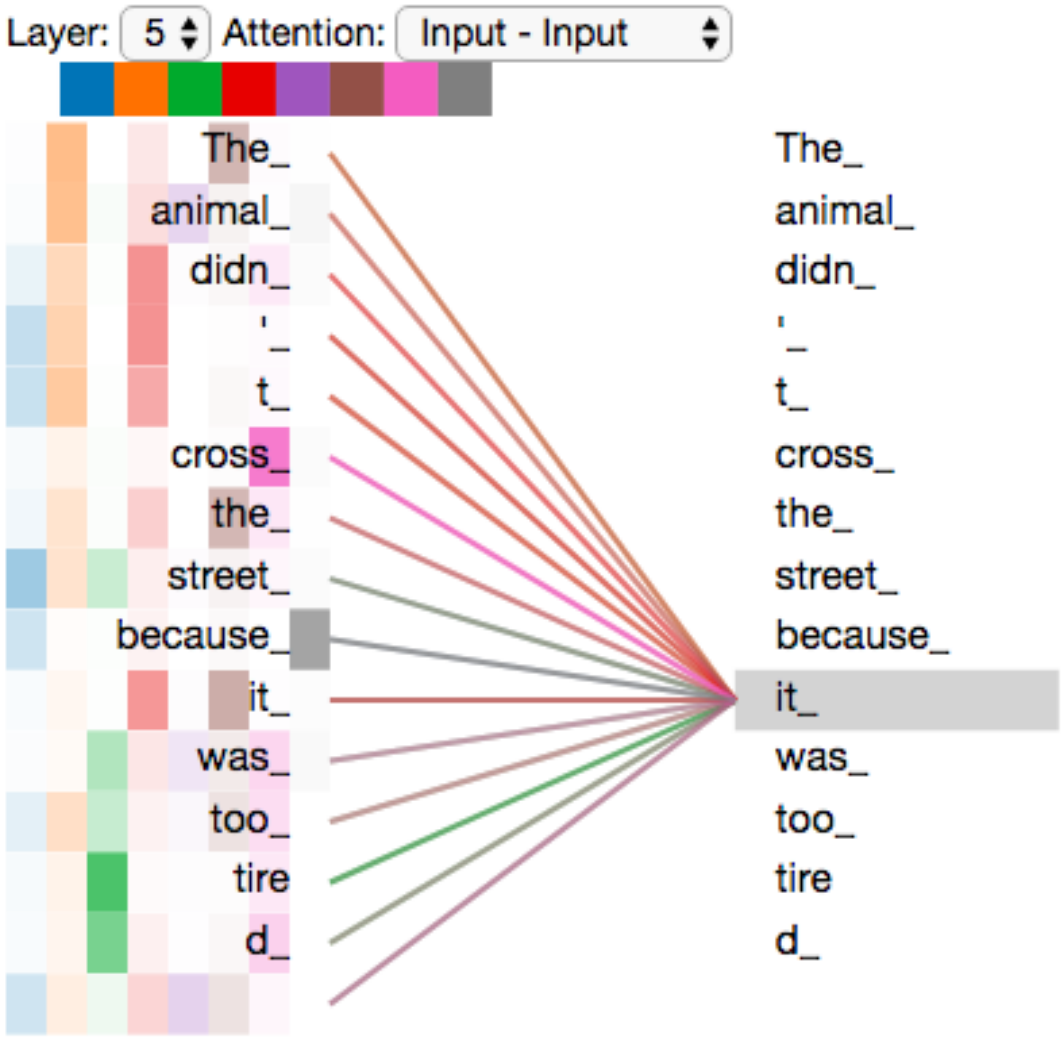
Self-attention: Visualization (Revisit with 2 heads)

As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" – in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".

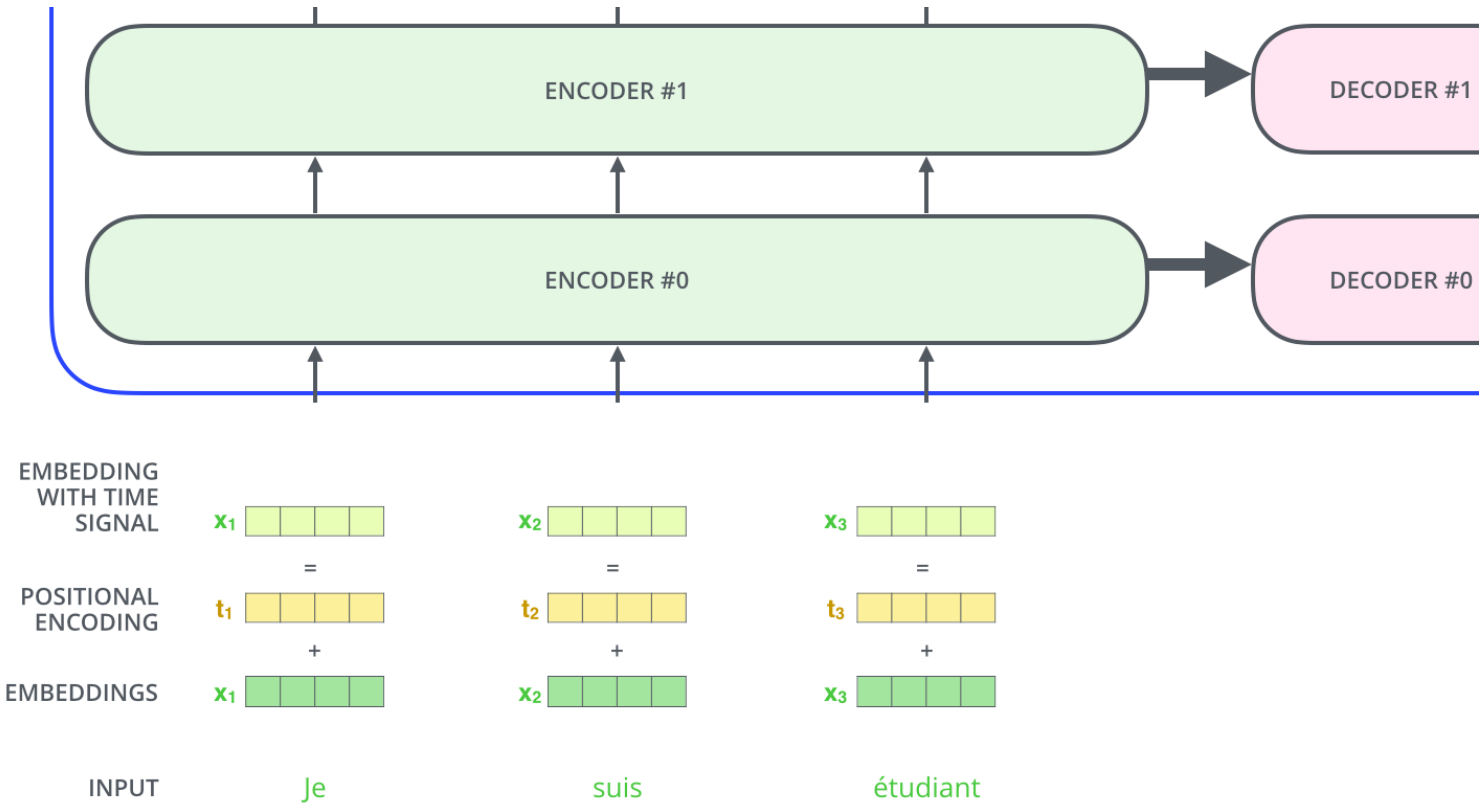


Self-attention: Visualization (Revisit with 8 heads)

If we add all the attention heads to the picture, however, things can be harder to interpret.



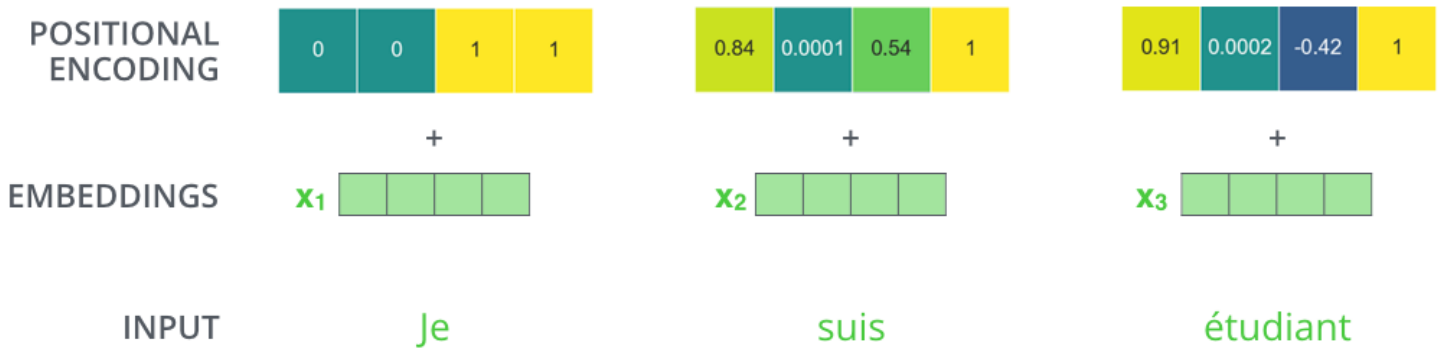
Positional Encoding: Representing Sequence Order



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.

Positional Encoding: Representing Sequence Order

If we assumed the embedding has a dimensionality of 4, the actual positional encodings would look like this:



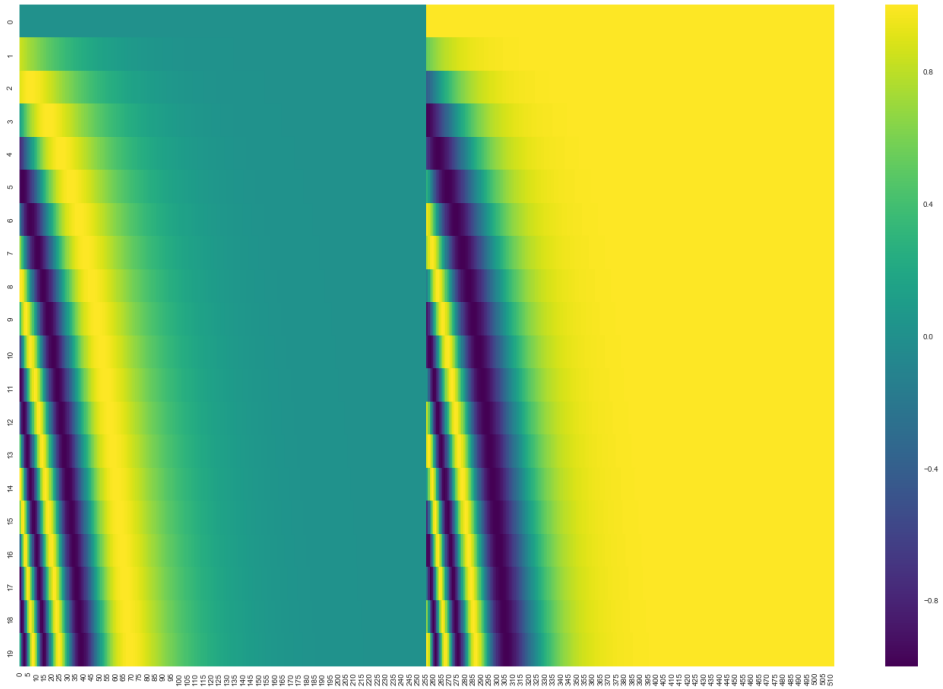
The formula for positional encoding is:

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

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

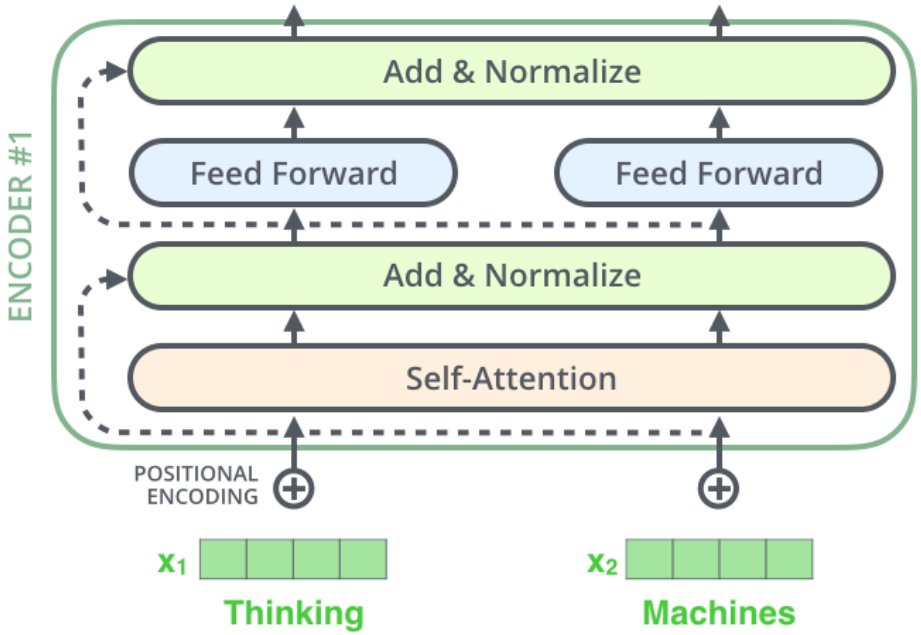
where *pos* is the position of the word in the input sequence, and *i* is the index of the dimension of the positional encoding vector.

Positional Encoding: Representing Sequence Order



A real example of positional encoding for 20 words (rows) with an embedding size of 512 (columns).

The residuals and layer normalization

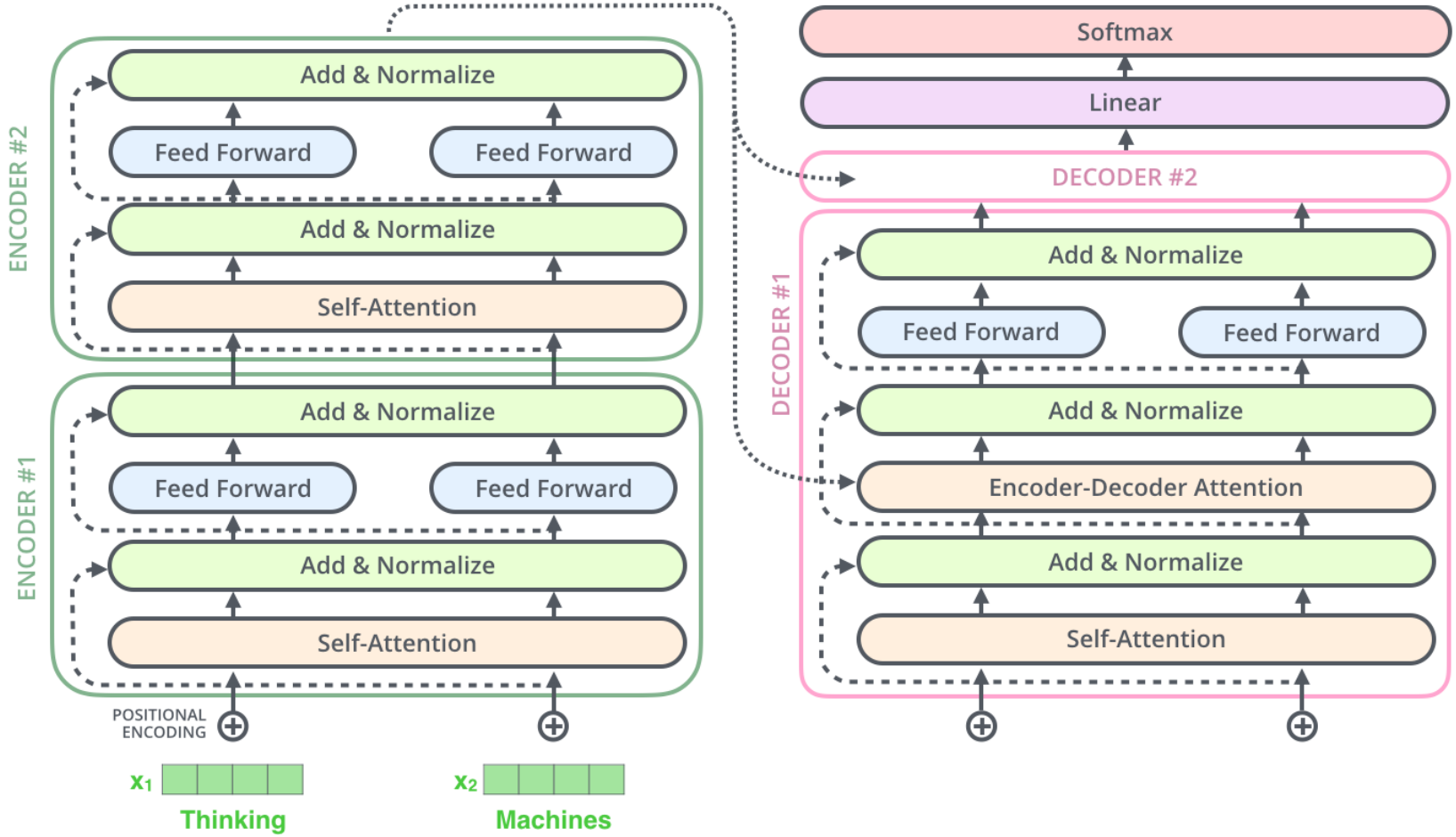


Each sub-layer (self-attention, ffnn) in each encoder has a residual connection around it, and is followed by a layer-normalization step.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Formula for the Feed Forward Network layer.

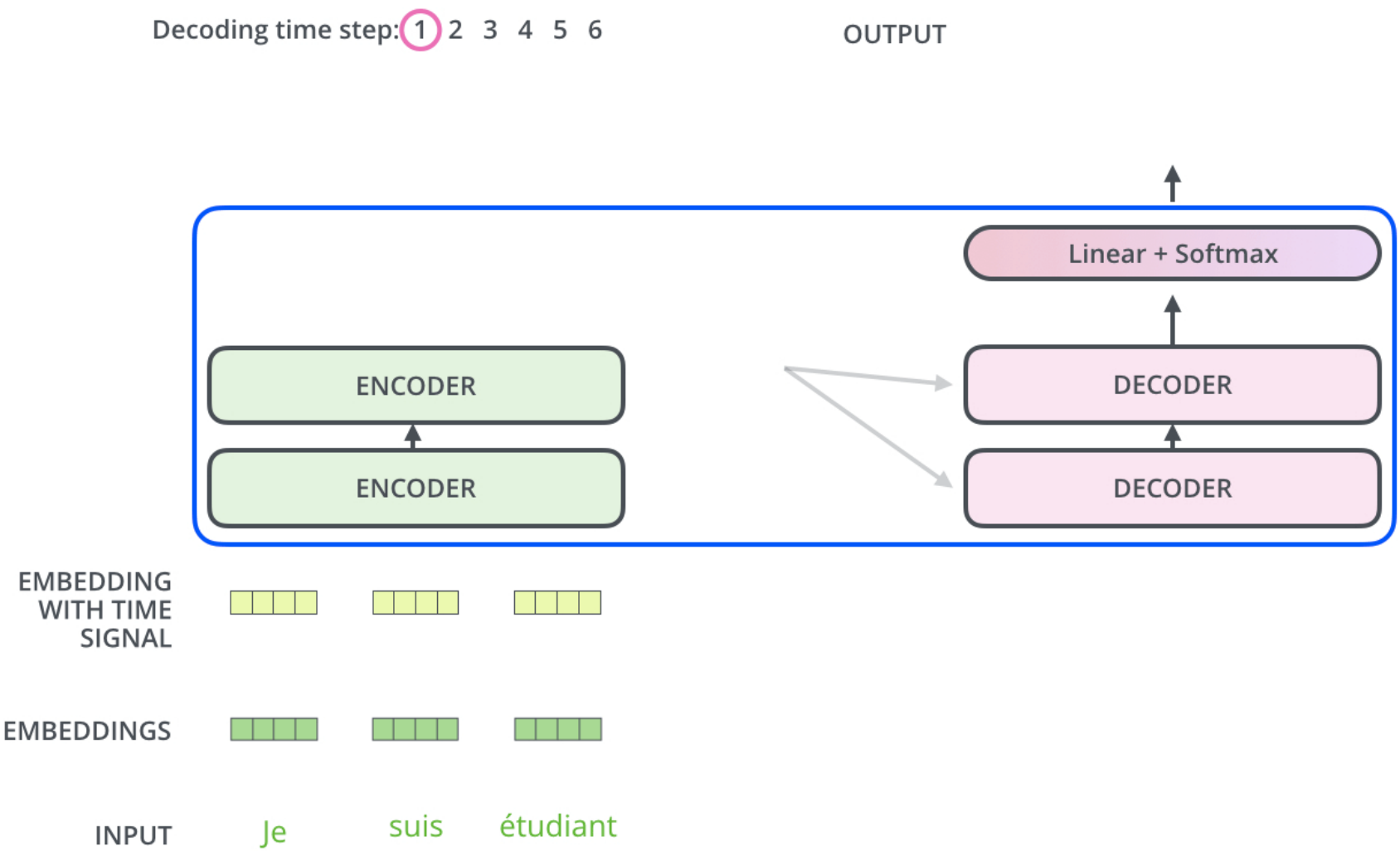
The residuals and layer normalizations



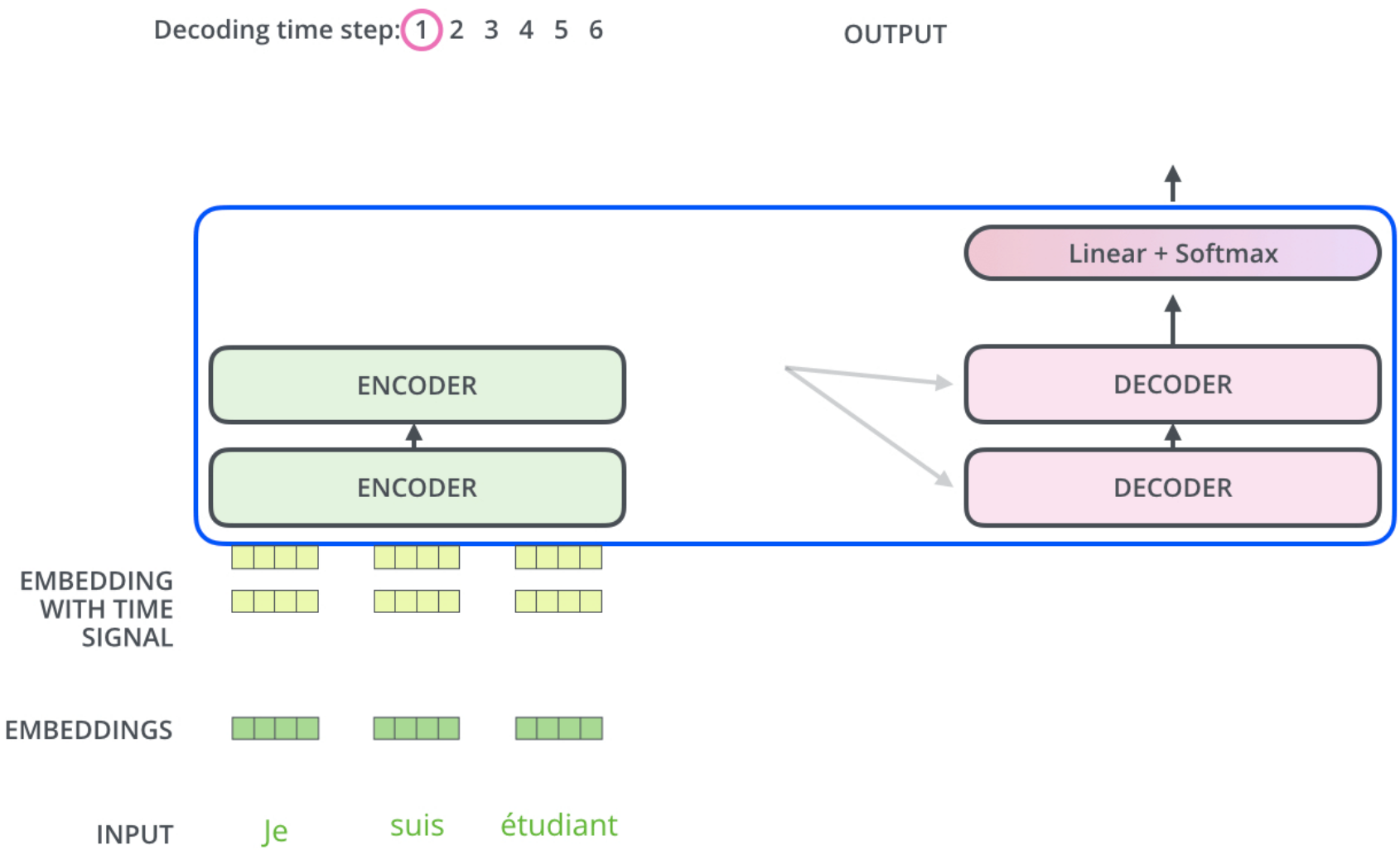
There are residual connections and layer normalizations for the sub-layers of the decoder as well.

Content

Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant