Lecture 07:
Machine Translation

Overview

1. Introduction to machine translation
2. Neural machine translation:
    1. seq2seq model
    2. Attention

Machine Translation (MT)

- **Machine Translation (MT)** is the task of translating a sentence *x* from one language (the source language) to a sentence *y* in another language (the target language).

x: *L'homme est né libre, et partout il est dans les fers*

y: *Man is born free, but everywhere he is in chains*

- 1950s: rule-based using bilingual dictionaries
- 1990 – 2010s: Statistical machine translation
- 2010s: Neural machine learning

Statistical Machine Translation (SMT)

SMT aims at learning the probabilistic model from data

- Suppose we want to translate French -> English
- We need to find the *best English sentence* y, given a *French sentence* x

$$\underset{y}{\operatorname{argmax}}\, P(y|x) = \underset{y}{\operatorname{argmax}}\, P(x|y)P(y)$$

Bayes Rule

Translation Model

Models how words and phrases should be translated (*fidelity*). Learnt from parallel data.

Language Model

Models how to write good English (*fluency*). Learnt from monolingual data.

Statistical Machine Translation (SMT)

How do we learn the translation model $P(x|y)$?

- large corpus of parallel text (French/English)
- Rewrite the translation model

$$P(x|y) \approx P(x, a|y)$$

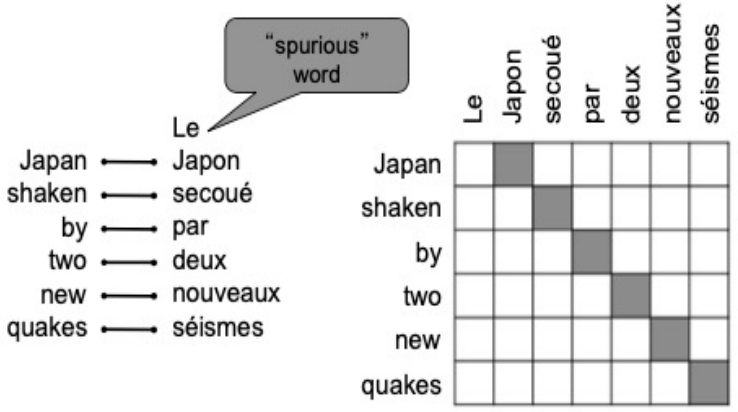
where a is an alignment

- an alignment is a word-level correspondence between French sentence x and English sentence y

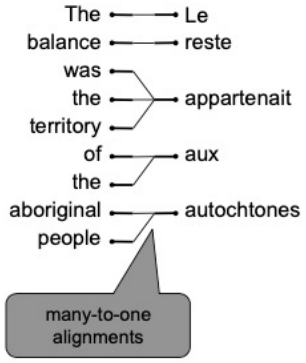
What is alignment?

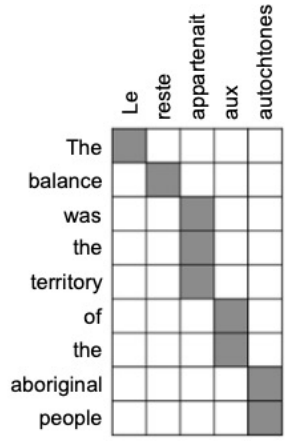
- Correspondence between particular words in the translated sentence pair
- Alignment can be very complex

Some words have no counterpart

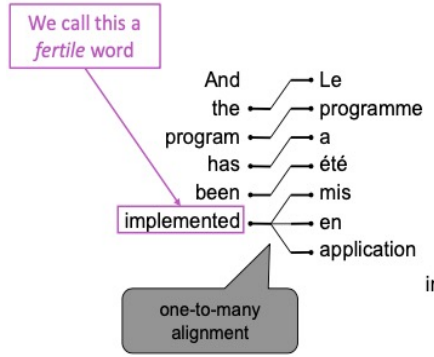


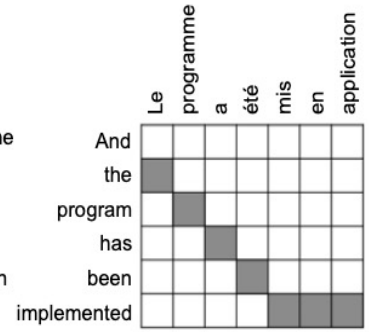
Alignment can be many-to-one





Alignment can be one-to-many





- Fertile words: words with no single equivalent between corresponding language
- Alignment can be many-many (phrase level)

The Mathematics of Statistical Machine Translation: Parameter Estimation", Brown et al, 1993. http://www.aclweb.org/anthology/J93-2003

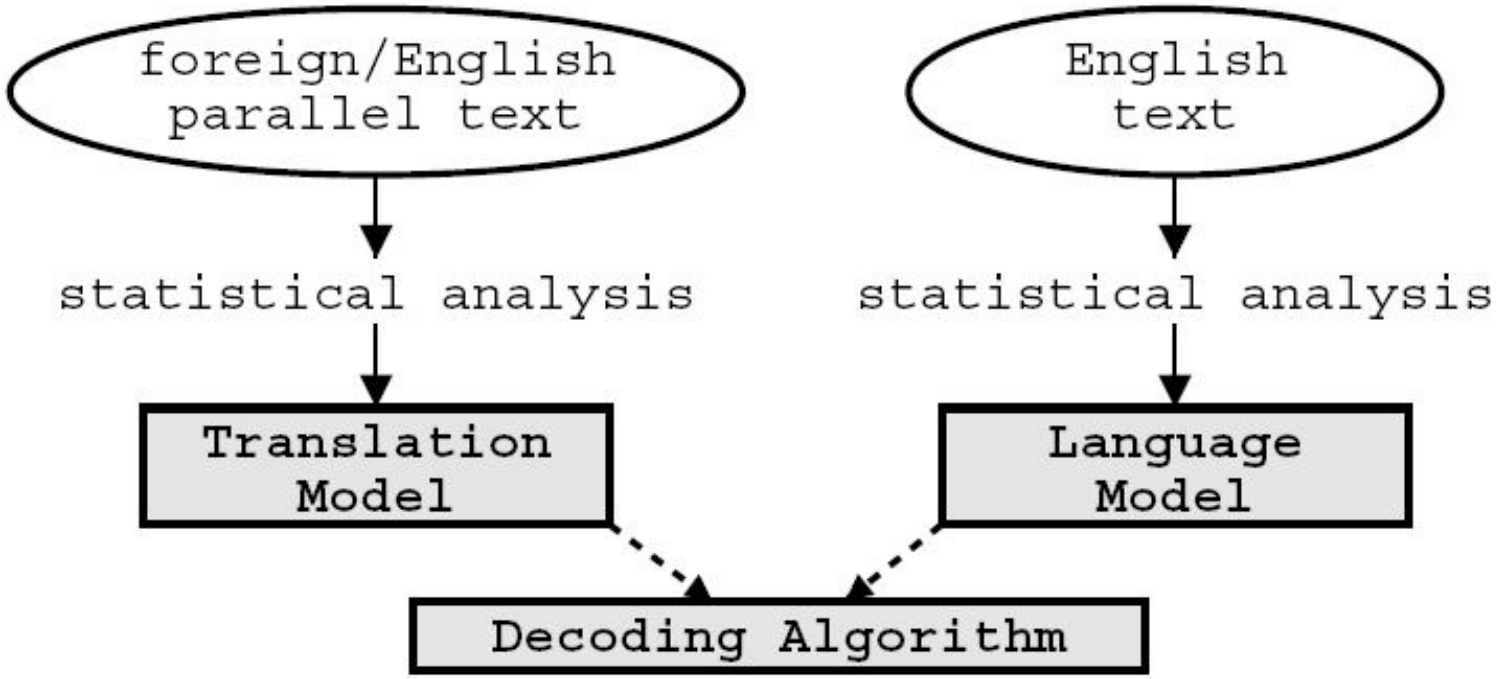
Learning alignment for SMT

We learn the alignment $P(x, a|y)$ as a combination of many factors including

- Probability of particular words aligning (can depend on position in sentence)
- Probability of particular words having particular fertility (number of corresponding words
  - What's the probability of a French word having 3 corresponding English word

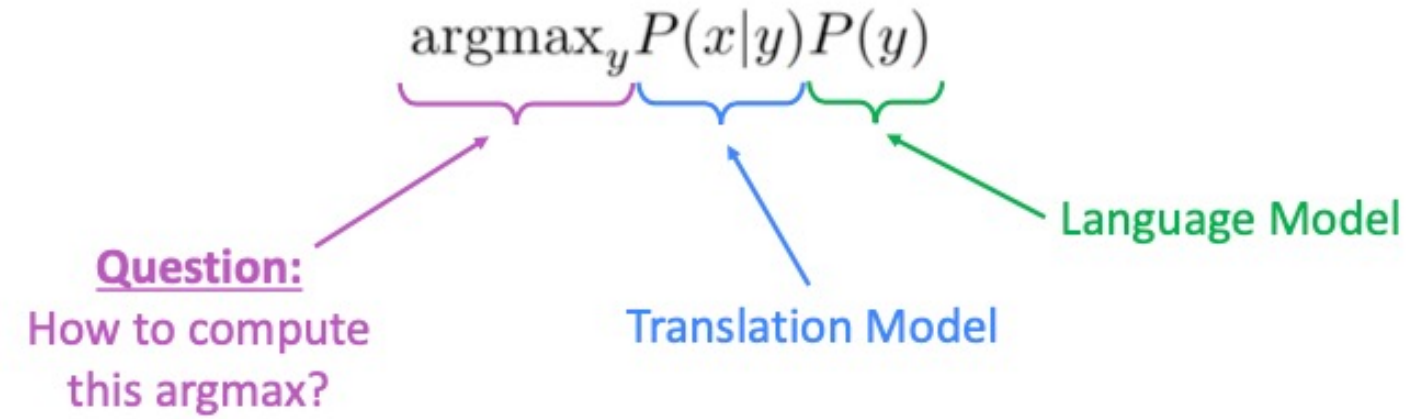
Components of a SMT model

- Components: Translation model, language model, decoder



Slide from Koehn 2008

Learning alignment for SMT



- Enumerate every possible *y* and calculate the probability?
  - Too expensive!
- **Solution:** Use a heuristic search algorithm to search for the best translation, discarding hypotheses that are too low-probability
  - This process is called *decoding*

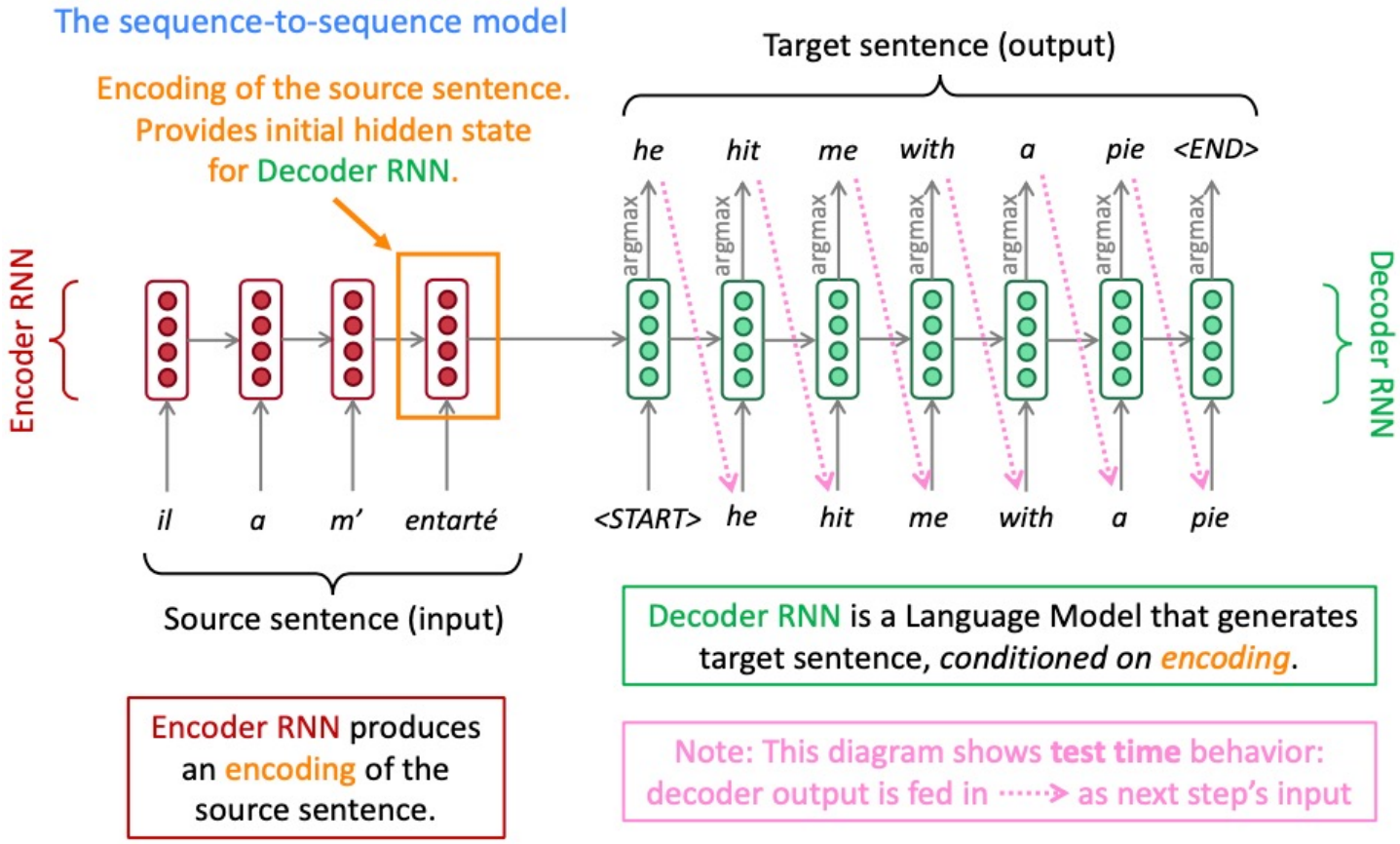
Statistical Machine Translation (SMT)

- The best systems were extremely complex
  - Hundreds of important details
  - Systems had many separately-designed subcomponents
  - Lots of feature engineering
    - Need to design features to capture particular language phenomena
  - Require compiling and maintaining extra resources
    - Like tables of equivalent phrases
  - Lots of human effort to maintain
    - Repeated effort for each language pair!

Neural Machine Translation

NMT – Sequence-to-sequence model

- NMT technique to model Machine Translation with a *single neural network*
- The neural network architecture is called seq2seq and it involves *two* RNNs.



Sequence-to-sequence model

- The sequence-to-sequence model is an example of a **Conditional Language Model**.
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are *also* conditioned on the source sentence *x*

- NMT directly calculates $P(y|x)$ while SMT breaks it down in to smaller parts:

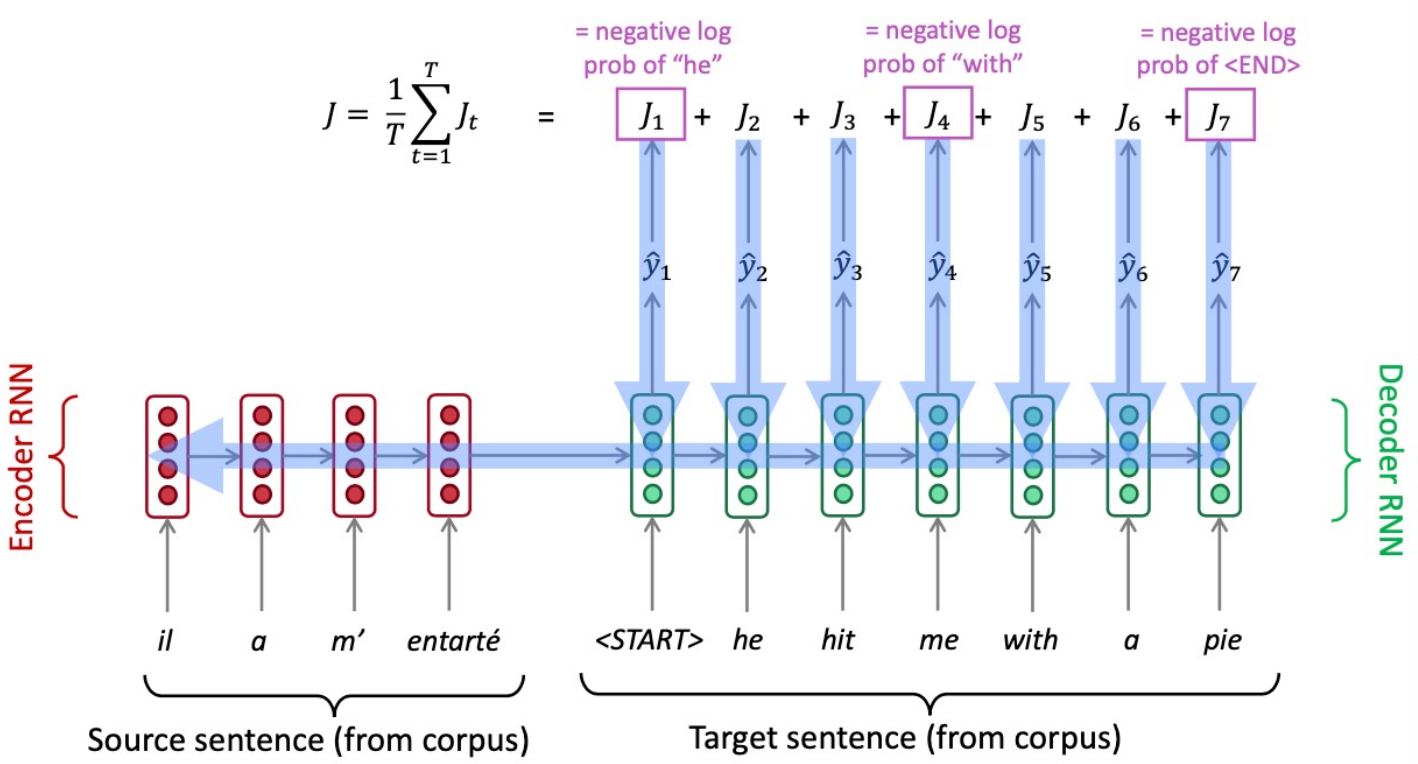
$$P(y|x) = P(y_1|x)P(y_2|y_1, x)P(y_3|y_2, y_1, x), \ldots, \underbrace{P(y_3|y_1, \ldots, y_{T-1}, x)}$$

Probability of next target word, given target words so far and source sentence *x*

Training neural machine translation system

You will need two embedding vectors

- One for the source language and another for the target language



Seq2seq is optimized as a **single system.**
Backpropagation operates "*end-to-end*".

Training can be done end-to-end so that the weights ares optimized together for optimal performance.

Decoding in NMT

NMT models require and encoder and a decoder

- Greedy decoding:
  - Generates target sentence by taking the argmax on each step of the decoder.
  - Has no way to undo decision

- Input: *il a m'entarté* *(he hit me with a pie)*
- → *he* ____
- → *he hit* ____
- → *he hit a* ____ (whoops! no going back now…)

Decoding in NMT

- Exhaustive search decoding
  - Find translation that maximizes $P(y|x)$
  - Try computing all possible sequences y (too expensive)
    - At each time step we are tracking V^t possible partial translations

- Beam search decoding
  - On each step of decoder keep track of the k most probable partial translation (hypothesis). K is the beam size
  - Beam search is not guaranteed to find optimal solution
  - More efficient than exhaustive search!

Advantages of NMT

Compared to SMT, NMT has many advantages:

- Better performance
  - More fluent
  - Better use of context
  - Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No subcomponents to be individually optimized
- Requires much less human engineering effort
  - No feature engineering
  - Same method for all language pairs

Disadvantages of NMT

Compared to SMT:

- NMT is less interpretable
  - Hard to debug

- NMT is difficult to control
  - For example, can’t easily specify rules or guidelines for translation
  - Safety concerns!

Evaluating Machine Translation

BLEU (**Bi**lingual **E**valuation **U**nderstudy)

- BLEU compares the machine-written translation to one or several human-written translation(s), and computes a similarity score based on:
  - *n*-gram precision (usually for 1, 2, 3 and 4-grams)
  - Plus a penalty for too-short system translations

- BLEU is useful but imperfect
  - There are many valid ways to translate a sentence
  - So a good translation can get a poor BLEU score because it has low *n*-gram overlap with the human translation

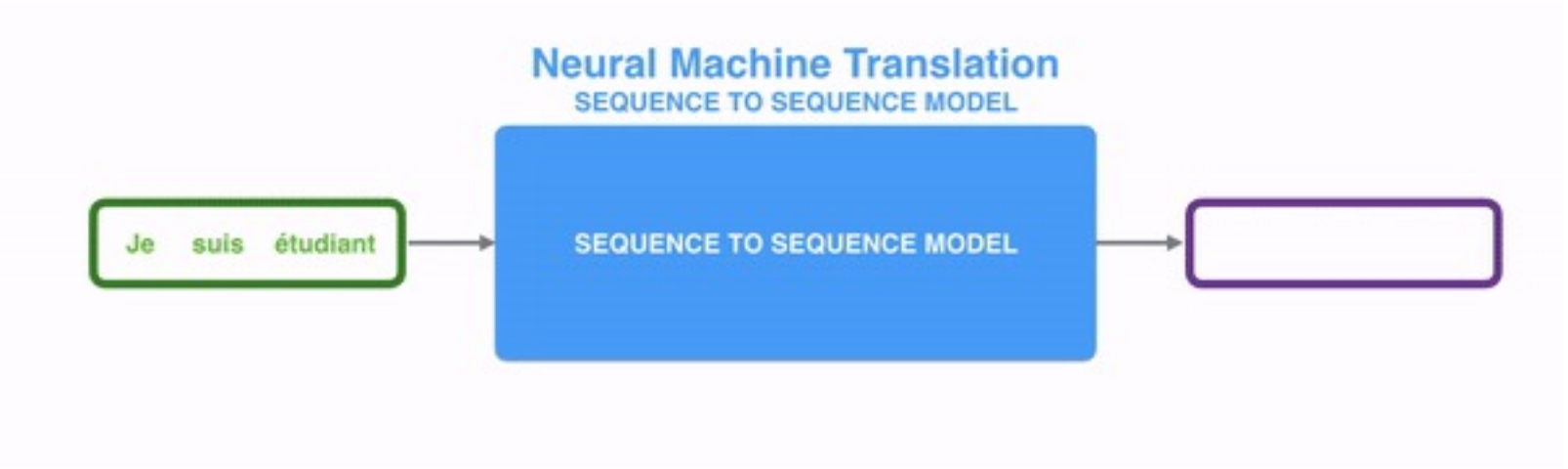
Difficulties in current machine translation

- Out-of-vocabulary words
- Domain mismatch between train and test data
- Maintaining context over longer text
- Low-resource language pairs
- NMT picks up biases in training data (gender, racial etc)

Attention

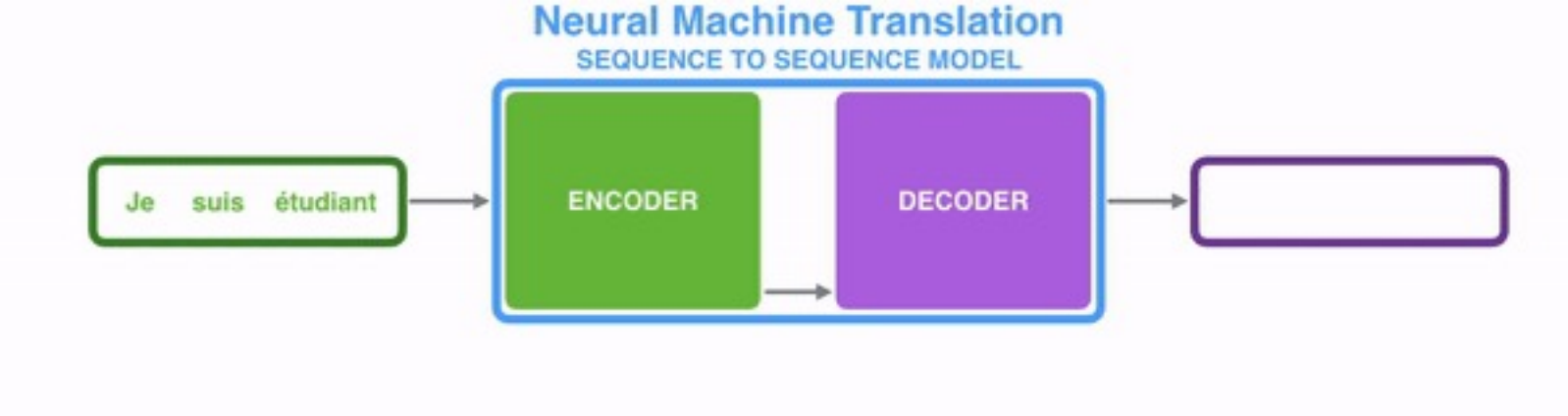
Sequence-to-Sequence (seq2seq) model

- A **sequence-to-sequence** model is a neural network model that takes a sequence as input and outputs another sequence.
  - Sequence can be of items (words, letters, features of an images…etc)
  - In neural machine translation a sequence is a series of words processed one after another and the output is also a sequence



Decoder – Encoder architecture

- A seq2seq model is composed of an encoder and a decoder
  - Encoder processes each item in the input sequence, it compiles the information it captures into a vector (called the context).
    - Dimension of context vector is the number of hidden units in the encoder RNN
  - Decoder takes in the context and produce the output sequence item by item.

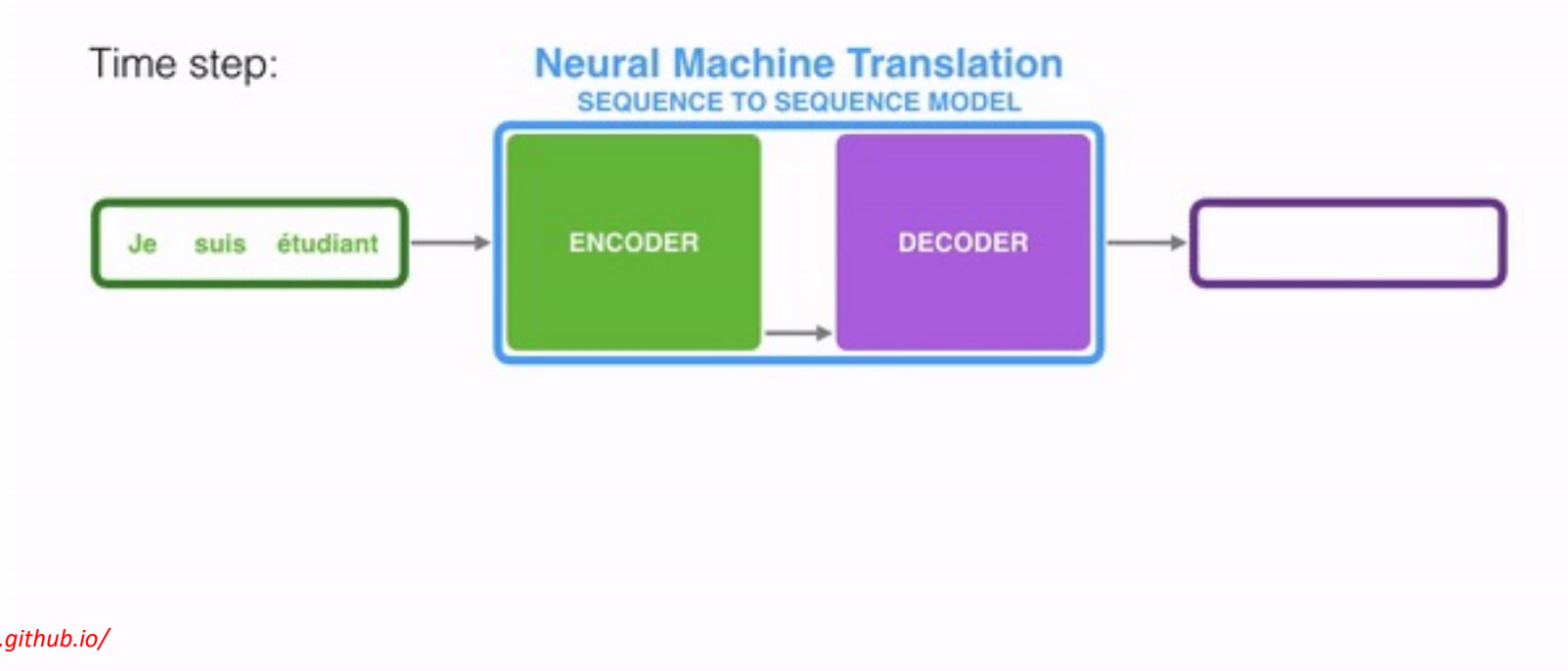


The encoder and decoder architecture for machine translation are both recurrent neural networks

Credit: https://jalammar.github.io/

Decoder – Encoder architecture

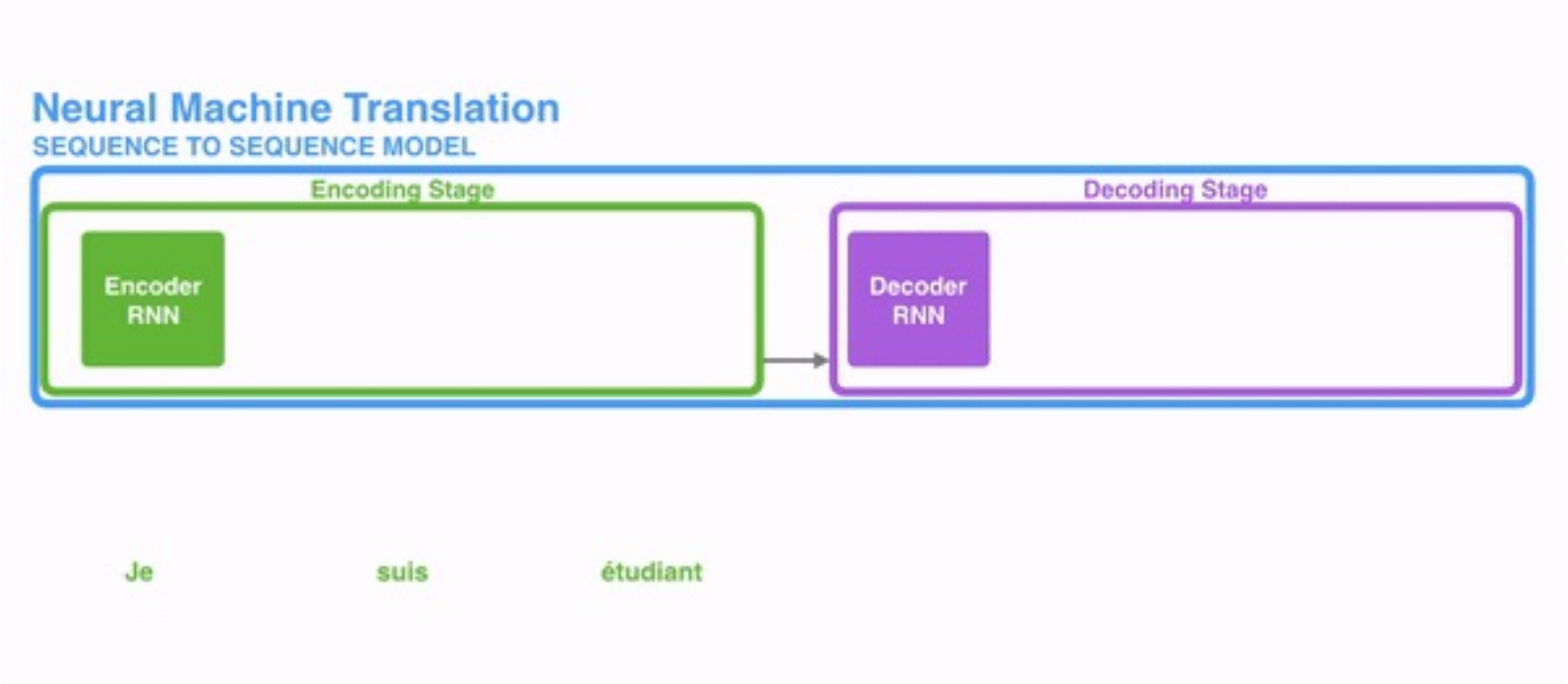
- Both encoder and decoder are RNNs
  - At timestep t one of the RNNs does some processing
  - it updates its hidden state $h^{(t)}$ based on its current inputs $x^{(t)}$ and previous inputs $h^{(t-1)}$ and $x^{(t-1)}$.
  - Hidden state of the encoder is the context we pass along
  - decoder also contain a hidden states that it passes from one time step to the next



Credit: https://jalammar.github.io/

Decoder – Encoder architecture

- An unrolled view of the encoder-decoder architecture
  - Encoder or decoder often consist of stacked (multi-layer) RNN models



Credit: https://jalammar.github.io/

Decoder – Encoder architecture

- Decoder – encoder bottleneck
  - The context vector turned out to be a bottleneck for these types of models.
    - *Context vector size is pre-defined*
  - It is challenging for the models to deal with long sequences
- **Solution:**
  - Attention (Bahdanau et al., 2014 and Luong et al., 2015)
  - Attention head allows the model to focus on the relevant parts of the input sequence

Attention

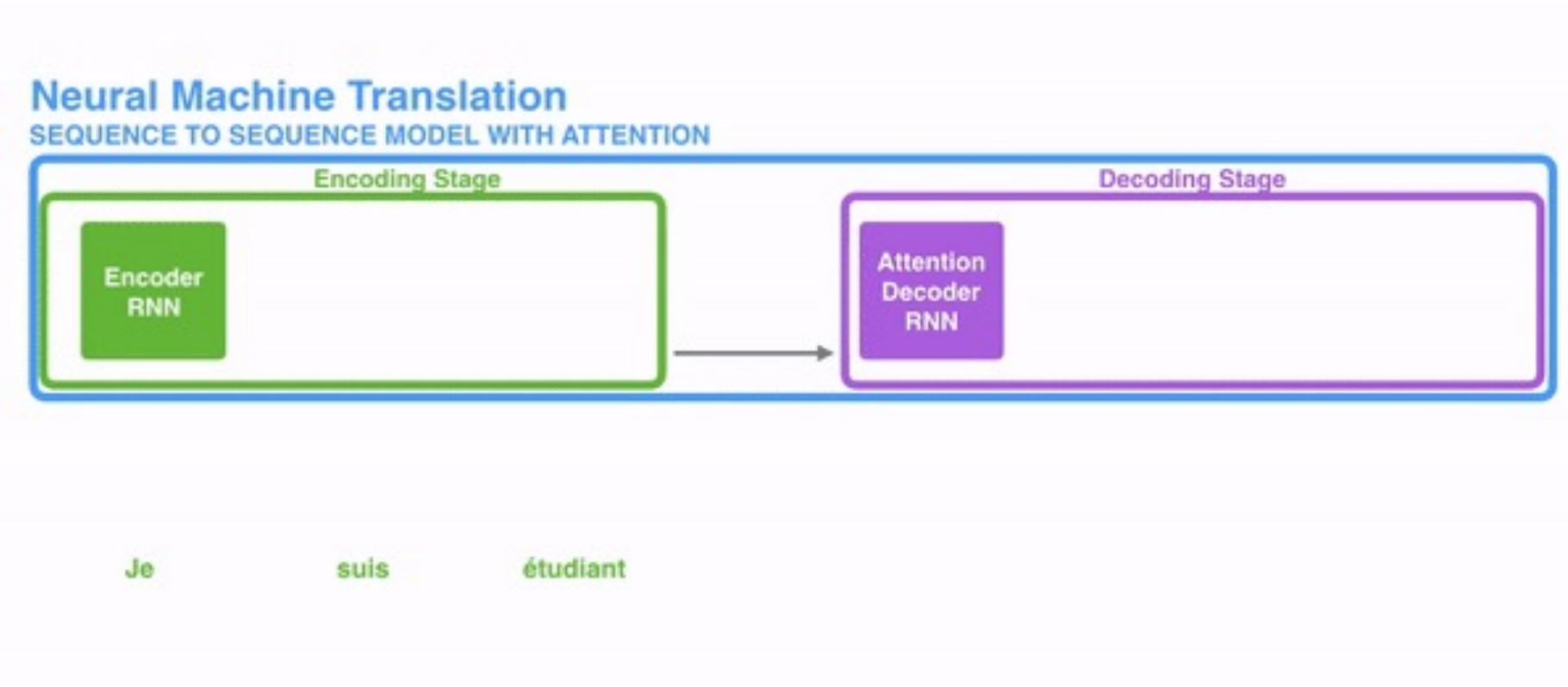
"One important property of human perception is that one does not tend to process a whole scene in its entirety at once. Instead humans focus attention selectively on parts of the visual space to acquire information when and where it is needed, and combine information from different fixation over time to build up an internal representation of the scene, guiding future eye movements and decision making."

- Recurrent Models of visual Attention

Credit: https://jalammar.github.io/

Attention

- Attention model differs from a classic seq2seq model in two main ways
  1. the encoder passes *all* the hidden states to the decoder



Credit: https://jalammar.github.io/

Attention

Attention model differs from a classic seq2seq model in two main ways

1. the encoder passes *all* the hidden states to the decoder
2. To focus on the relevant parts of the input decoder does the following
   i. Evaluate each encoder hidden states – each encoder hidden states is most associated with a certain word in the input sentence.
   ii. Assign a score to each hidden states
   iii. Multiply each hidden states by its softmaxed score, thus amplifying hidden states with high scores, and drowning out hidden states with low scores

Credit: https://jalammar.github.io/

Attention

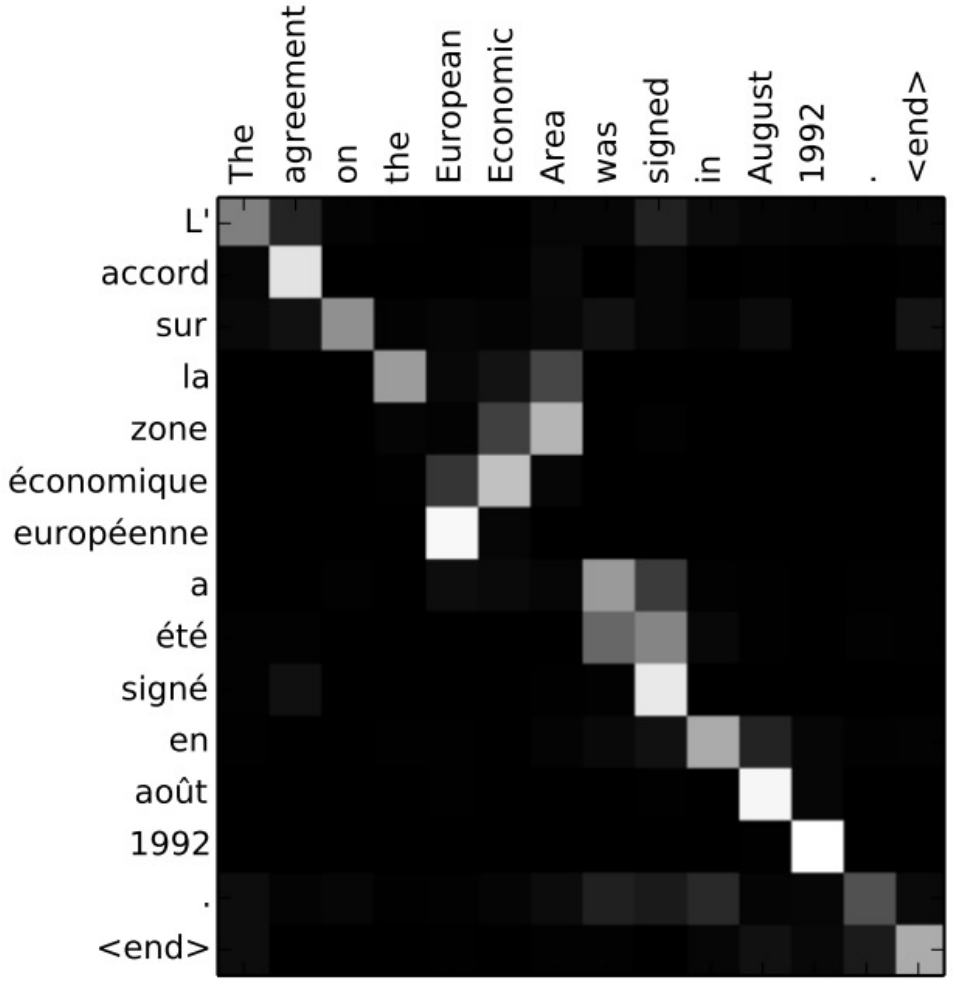
Scoring is done at each timestep on the decoder side

Credit: https://jalammar.github.io/

Attention

Example of the how precise the attention mechanism can be.

- The model learns how to align words in the language pair
- You can see how the model paid attention correctly
  - *European Economic Area* ⇔ *zone européenne économique*
    - -reverse order in French
  - Every other word in the sentence is in similar order.

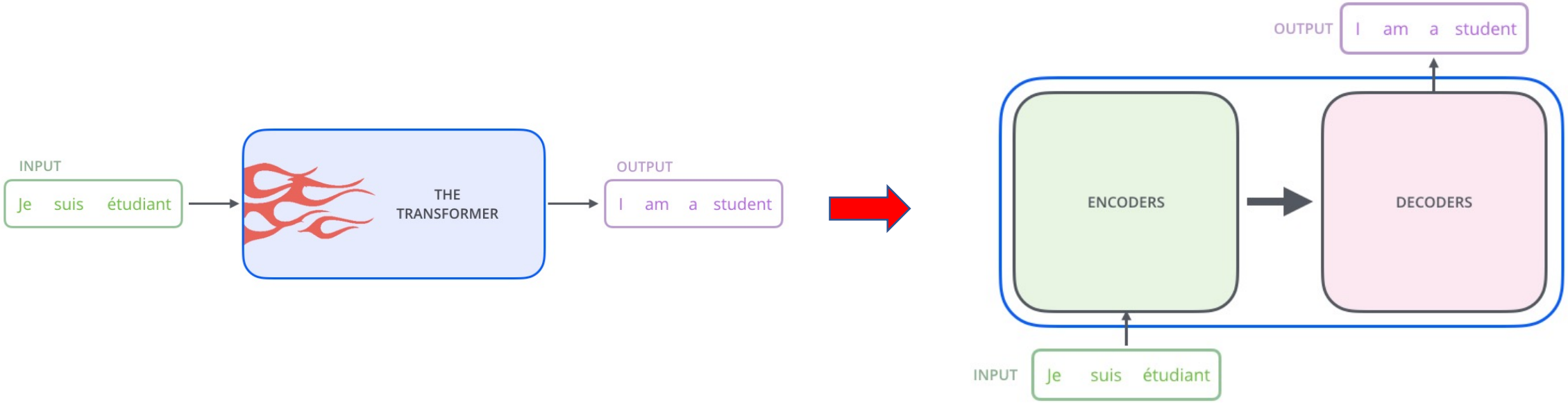


Transformers

Transformers

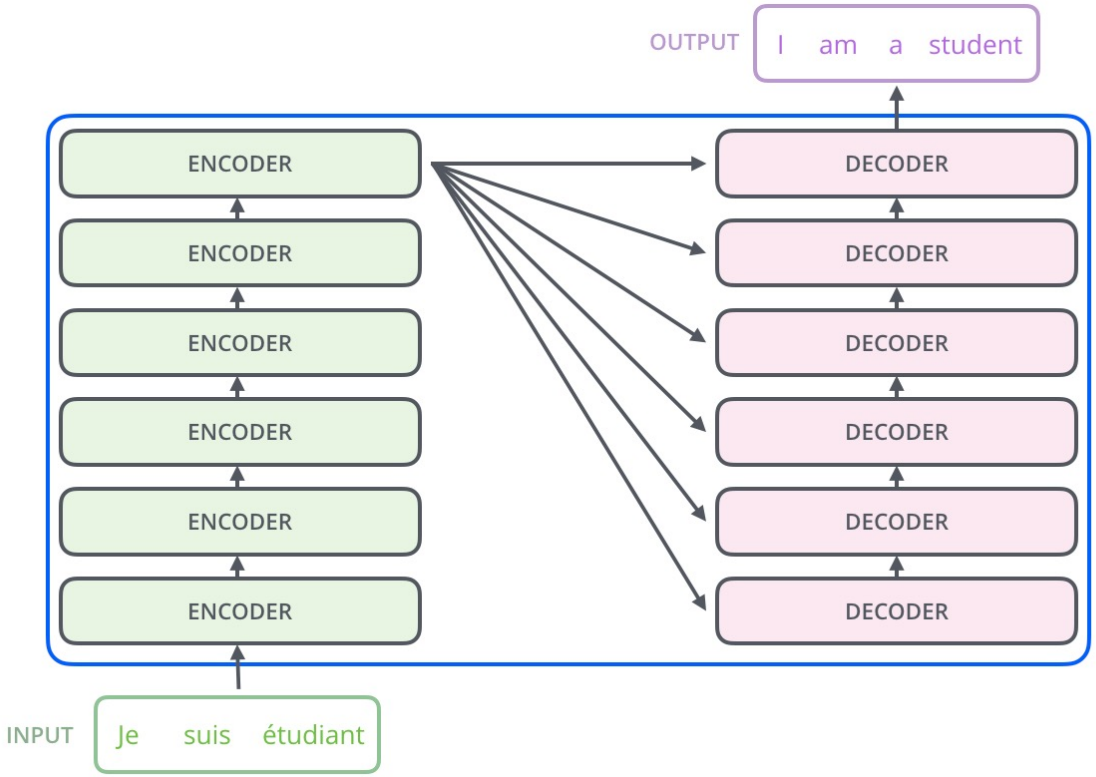
- **The Transformer** – proposed by Vaswani A., 2017 (Attention is all you need)
  - uses attention to boost the speed with which these models can be trained.
  - The biggest benefit of The Transformer is the speed of computation due to parallelization.

- In a machine translator setting transformer model takes input in. one language and outputs its translation in another.

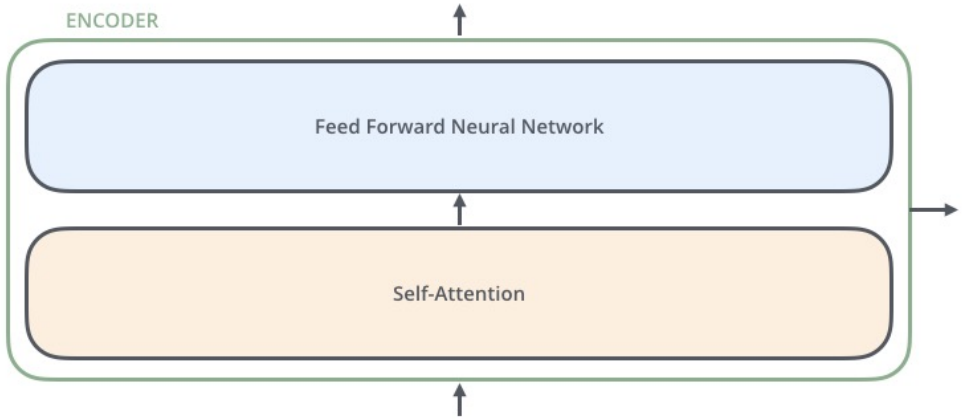


Transformer – Architecture

- Architecture ⇔ stacked encoder – decoder architecture
  - The encoding component is a stack of encoders (the original paper stacks six of them on top of each other)
  - The decoding component is a stack of decoders of the same number as the encoder

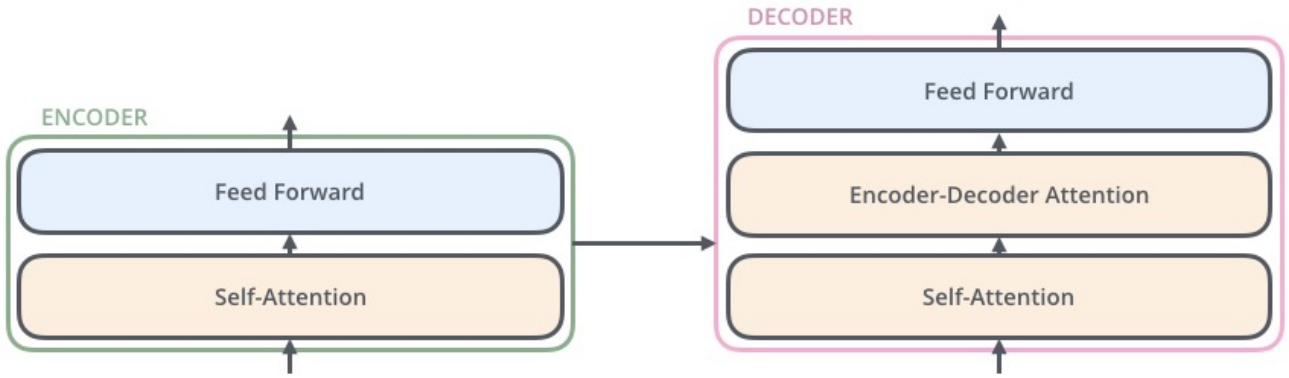


The encoders are all identical in structure
- yet they do not share weights
- Each one is broken down into two sub-layers



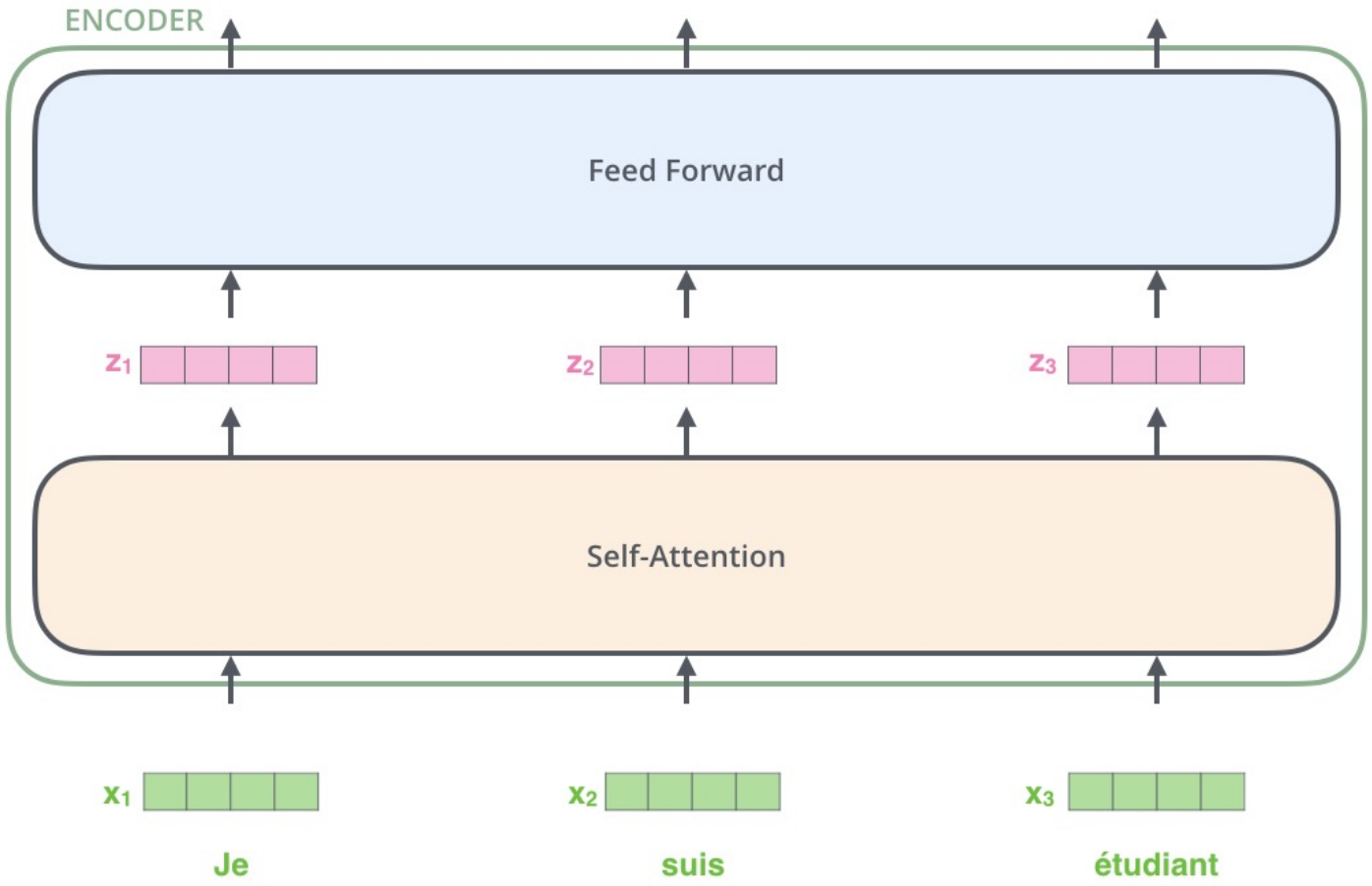
Transformer – Architecture

- Architecture ⇔ stacked encoder – decoder architecture
  - The encoder's inputs first flow through a self-attention layer
  - 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.



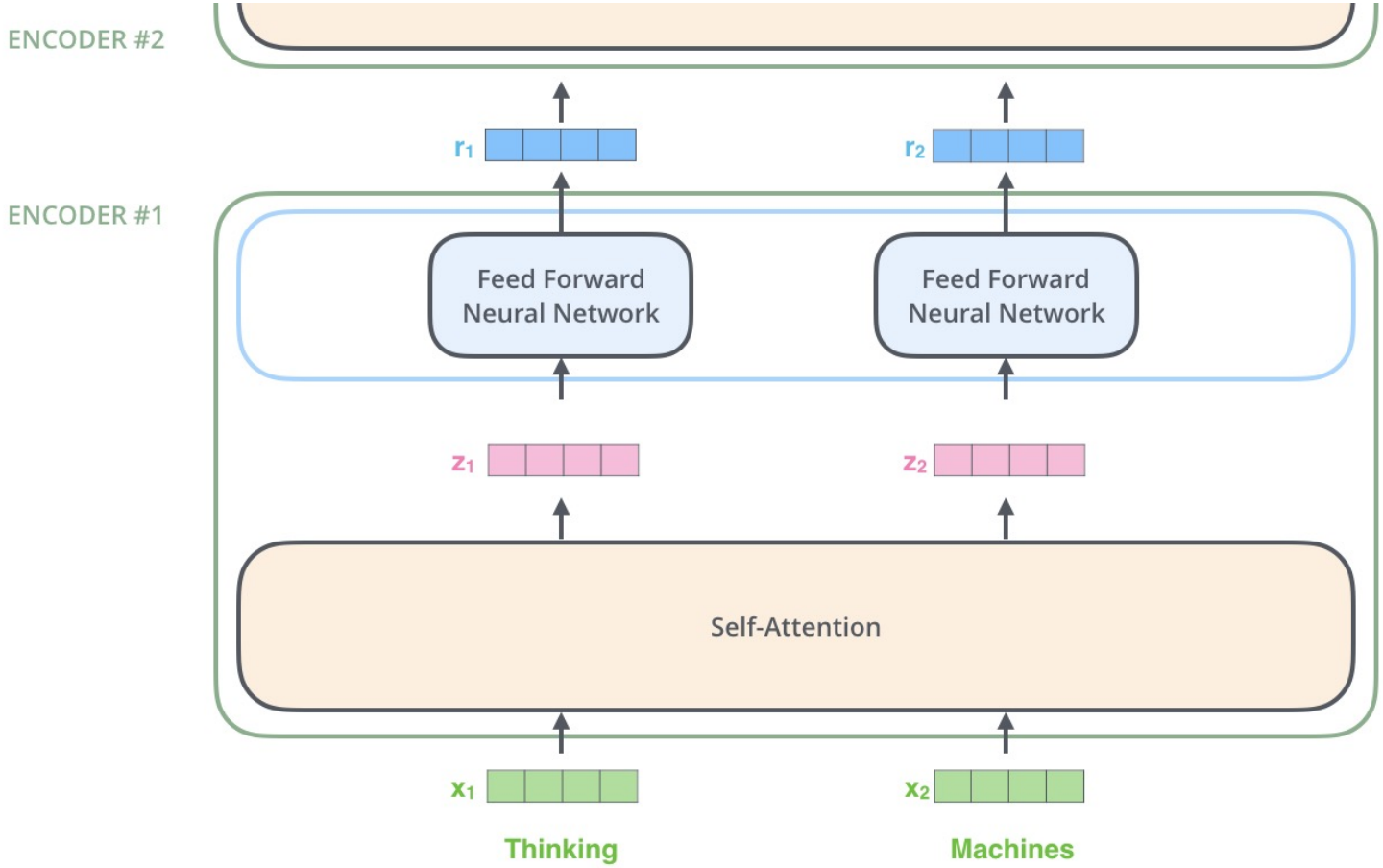
How the transformer works

- Similar to all NLP applications convert input word into a word embedding vectors



Transformer – 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

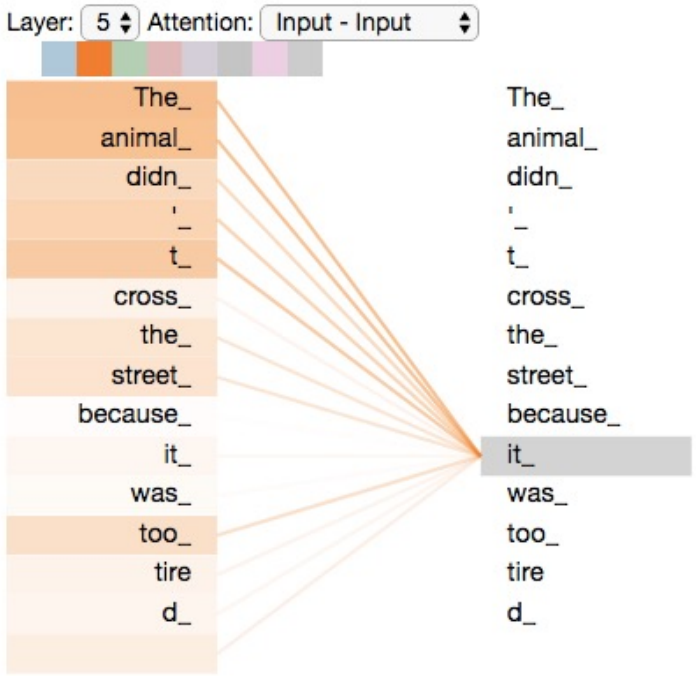


Transformer – Self Attention

Input sentence: *"The animal didn't cross the street because it was too tired."*

- What does "*it*" in this sentence refer to?

- Self-attention allows the transformer model to associate "*it*" with "*animal*".

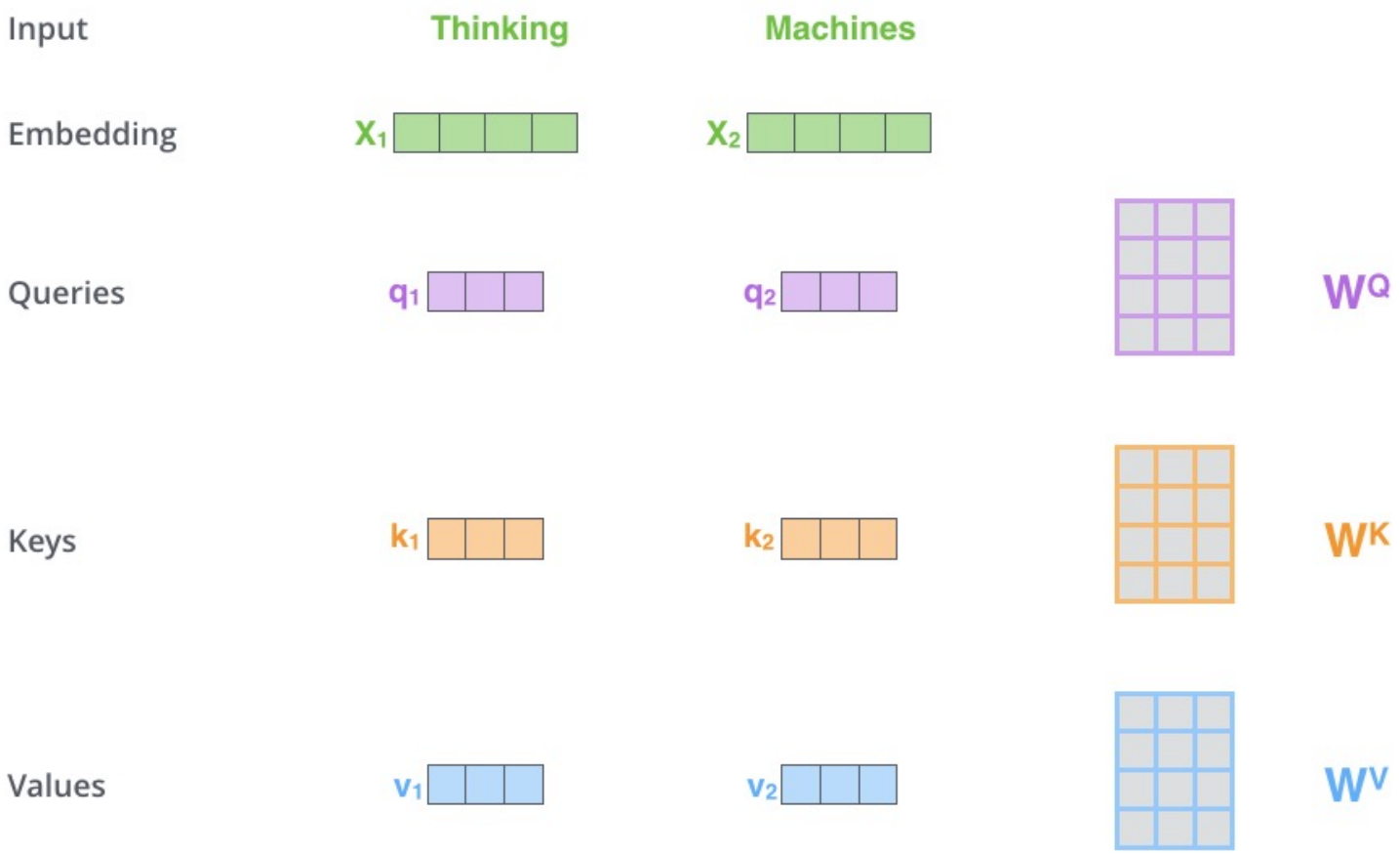


https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb

How to calculate self Attention

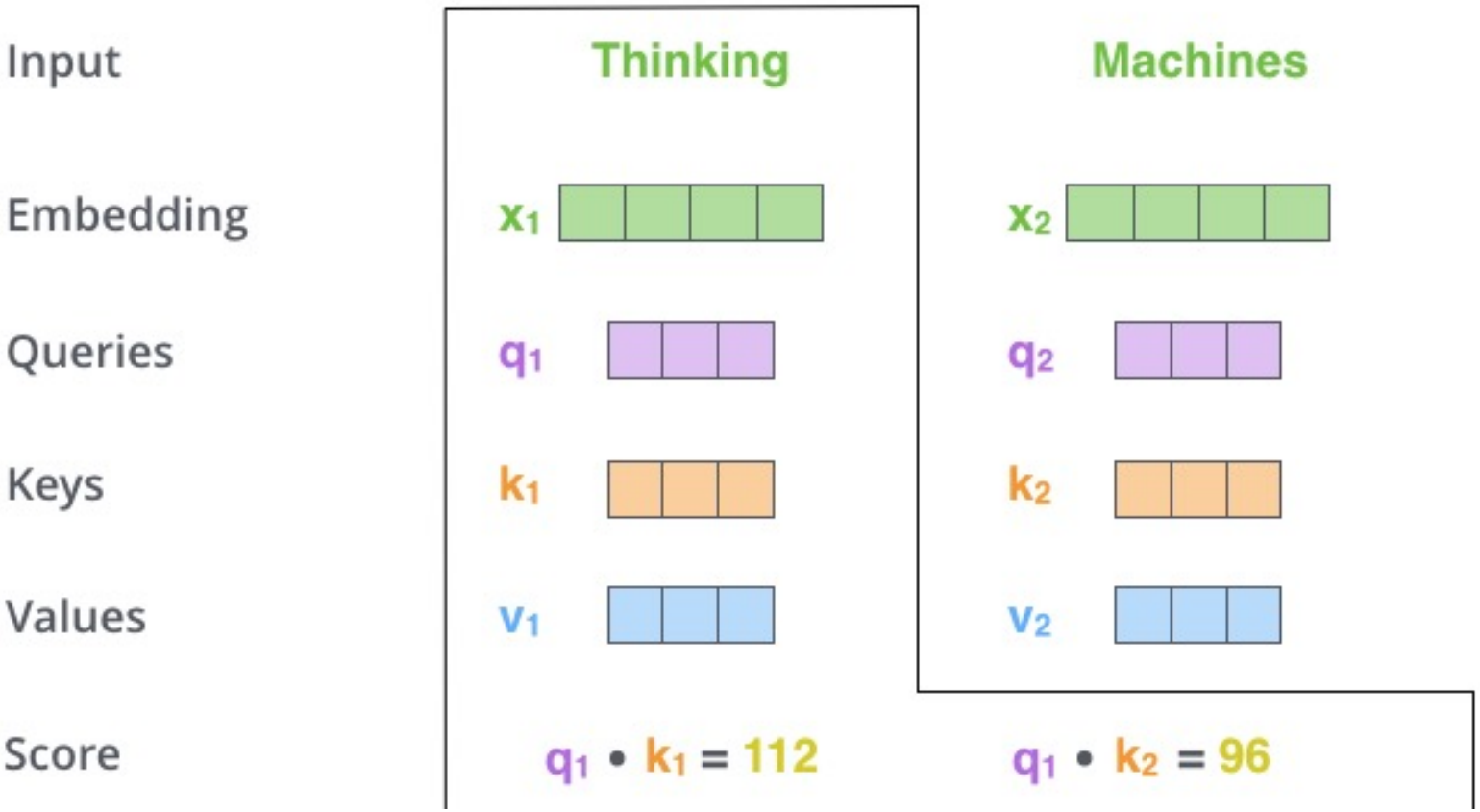
Step 1: create 3 vectors from each encoder's input vector

- Query vector, Key vector, Value vector



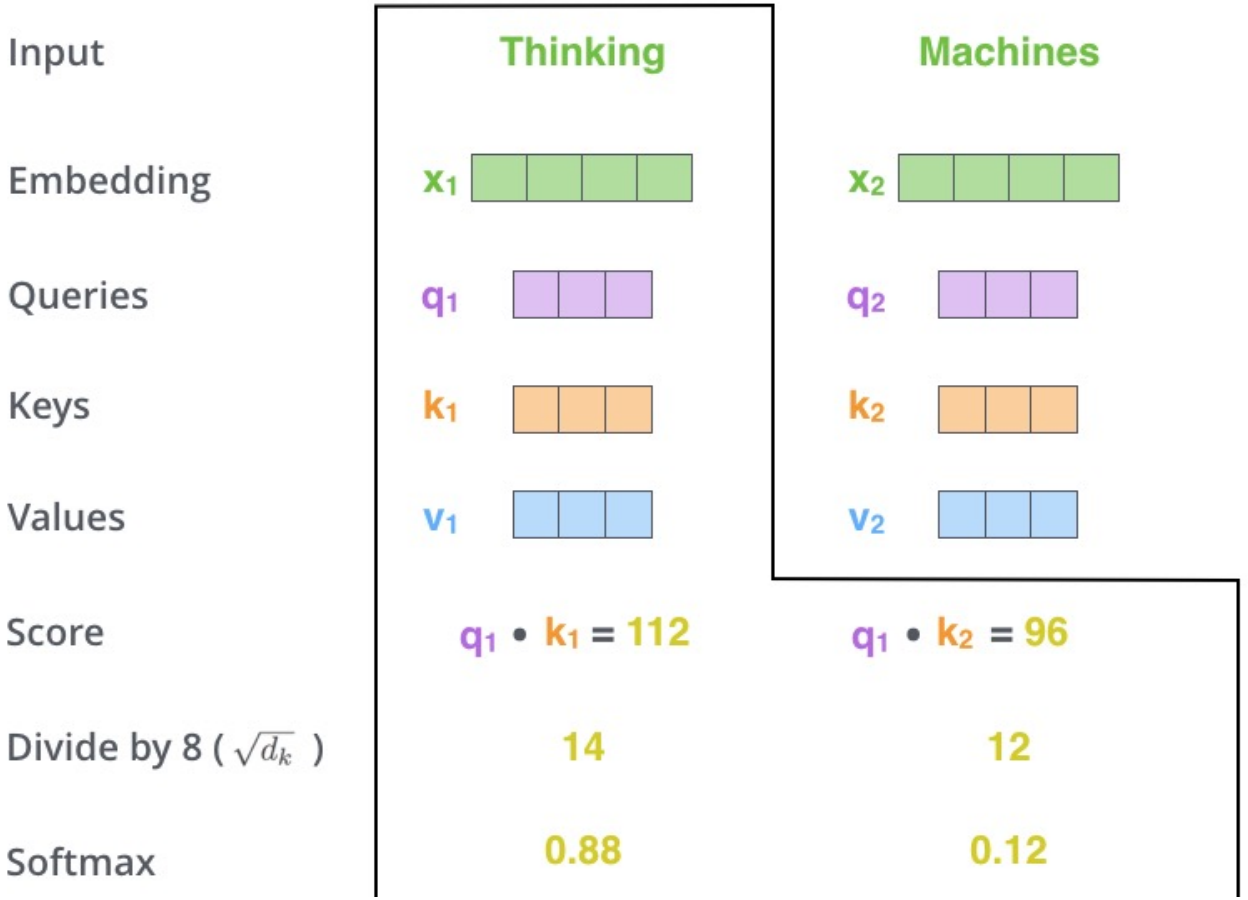
How to calculate self Attention

Step 2: calculate a self attention score



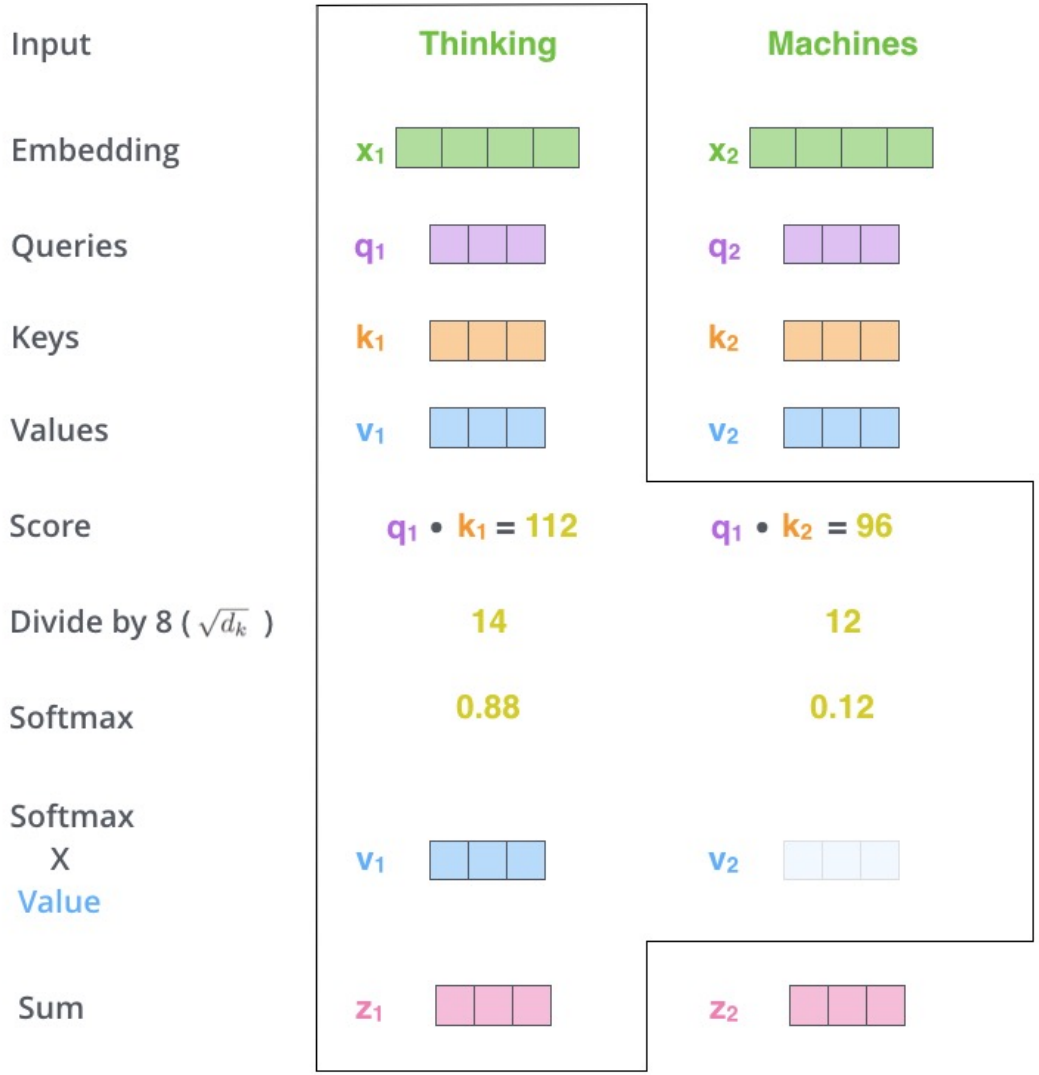
How to calculate self Attention

Step & 4: divide the core by 8 (hyperparameter to be tuned)



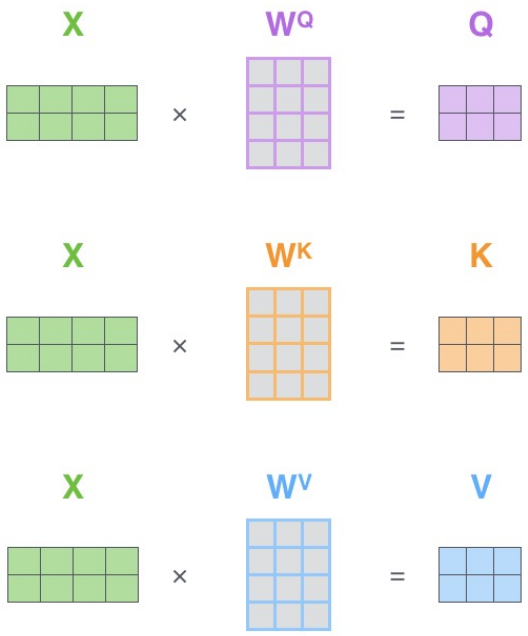
How to calculate self Attention

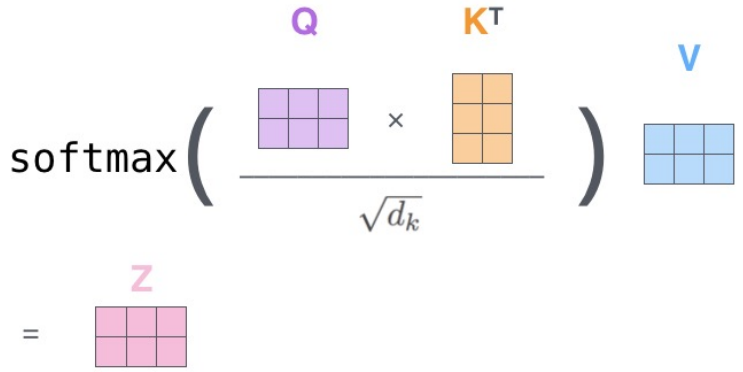
- **Step 5:** multiply each value vector by the SoftMax score
  - **Intuition:** to amplify relevant words and down grade irrelevant words
- **Step 6:** sum up the weighted value vectors.
  - This produces the output of the self-attention layer at this position (for the first word)



How to calculate self Attention – summary

- **First** calculate the Query, Key, and Value matrices.
  - by packing our embeddings into a matrix X and multiplying it by the weight matrices we've trained (W^Q, W^K, W^V).



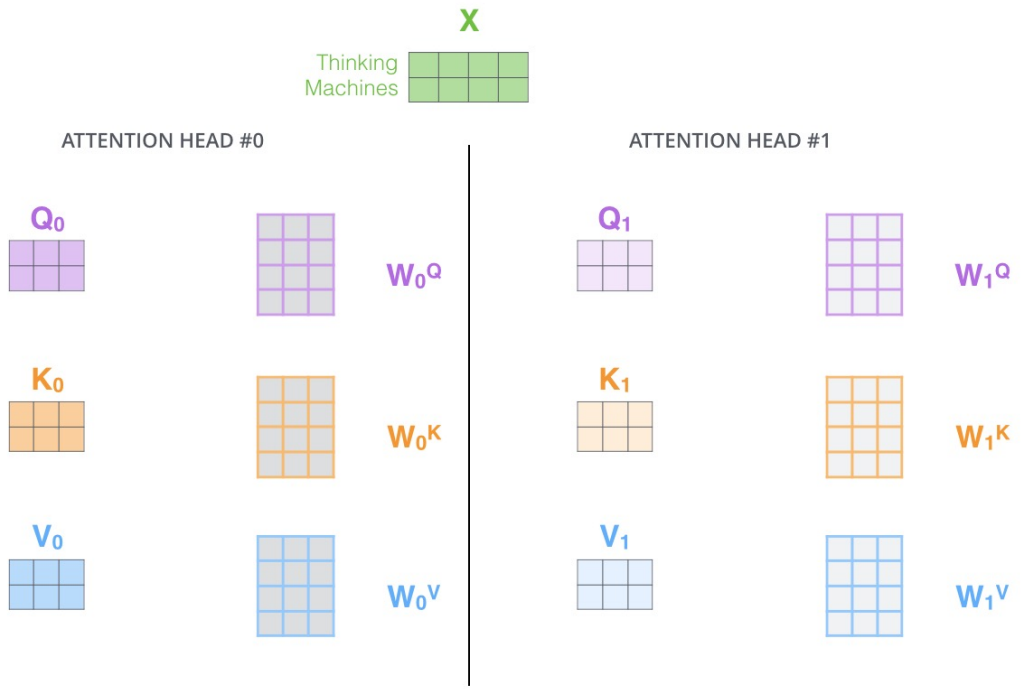


- **Finally** calcula tention layer
  - we can condense steps two through six in one formula to

Multi-headed Attention

Multi-headed attention improves the performance of the attention layer by allowing the following capabilities

1. It expands the model's ability to focus on different positions.
2. It gives the attention layer multiple "representation subspaces".

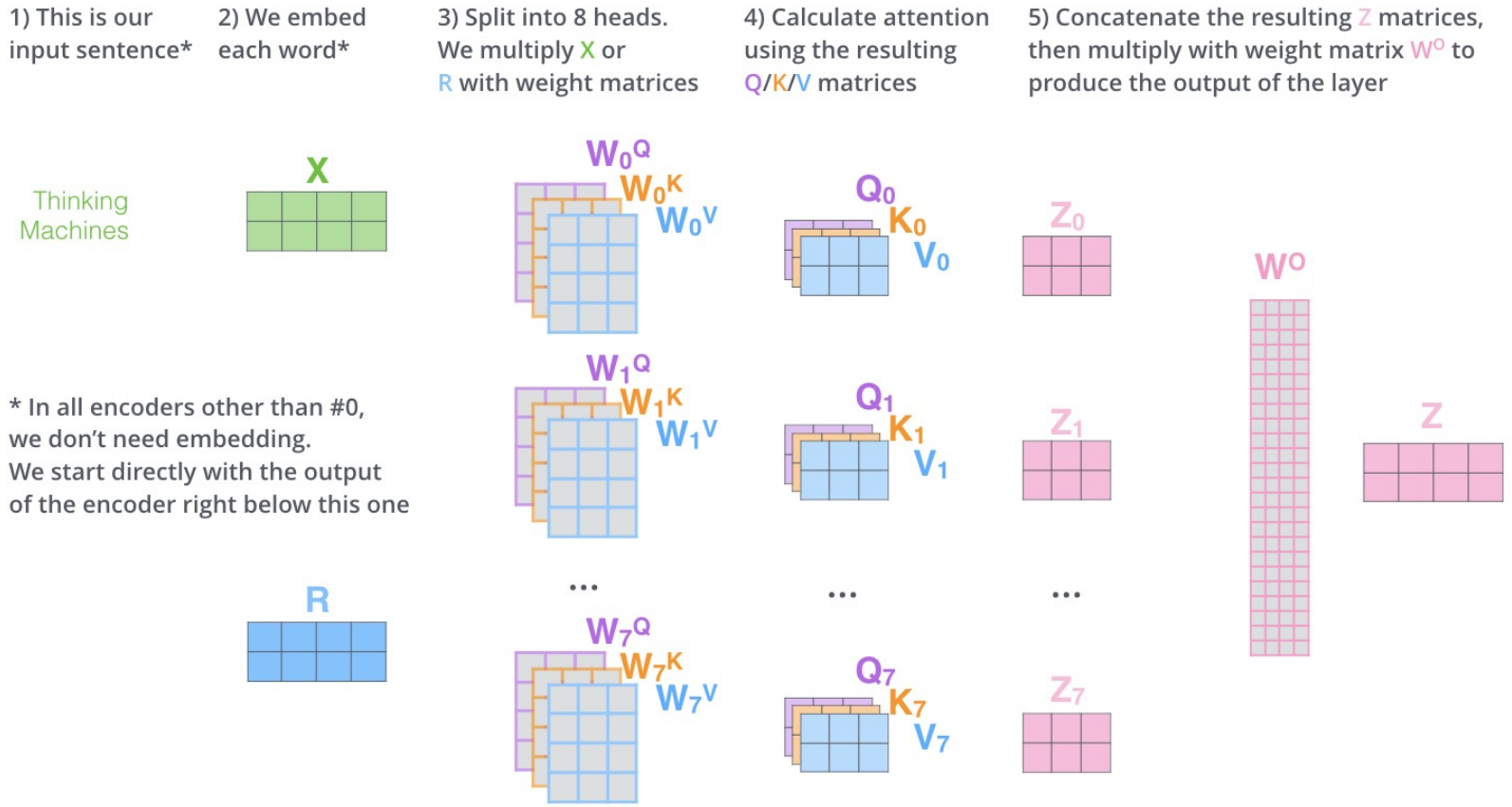


Note:
Multi-headed attention contain separate Q/K/V weight matrices for each head resulting in different Q/K/V matrices

Multi-headed Attention

Assume we have multi-headed attention with 8 self attention ⇔ eight *Z* matrices

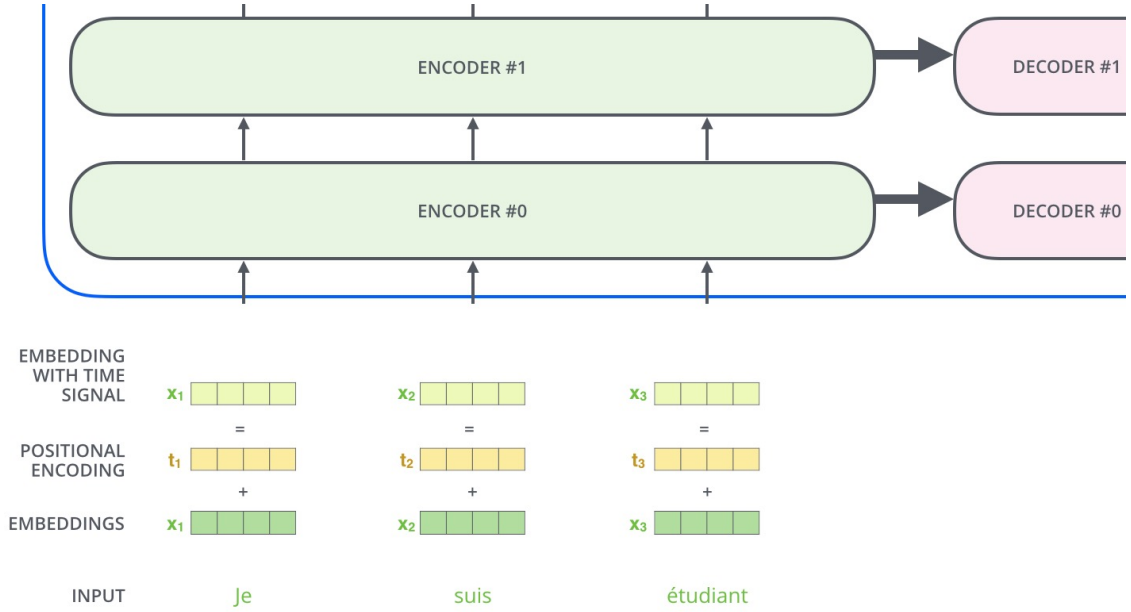
- the feedforward layer expects a single matrix



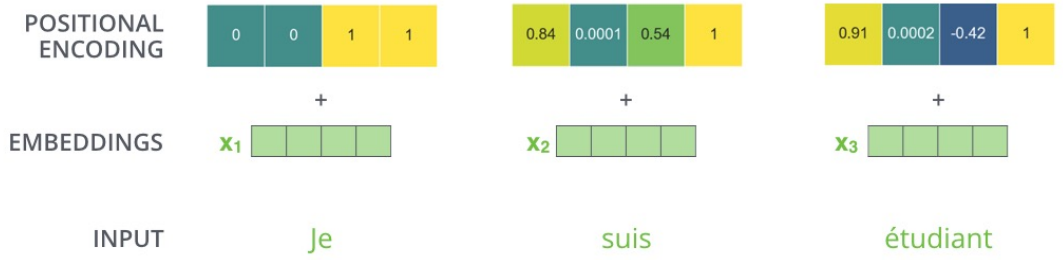
Positional encoding in Transformers

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.



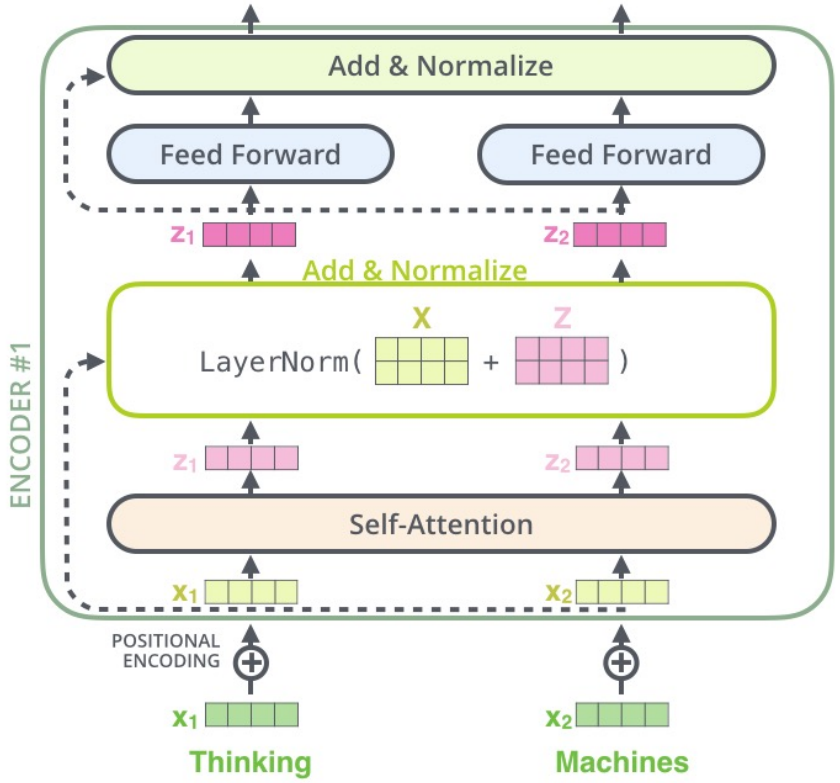
Example of positional encoding with a toy embedding size of 4



Residual connection

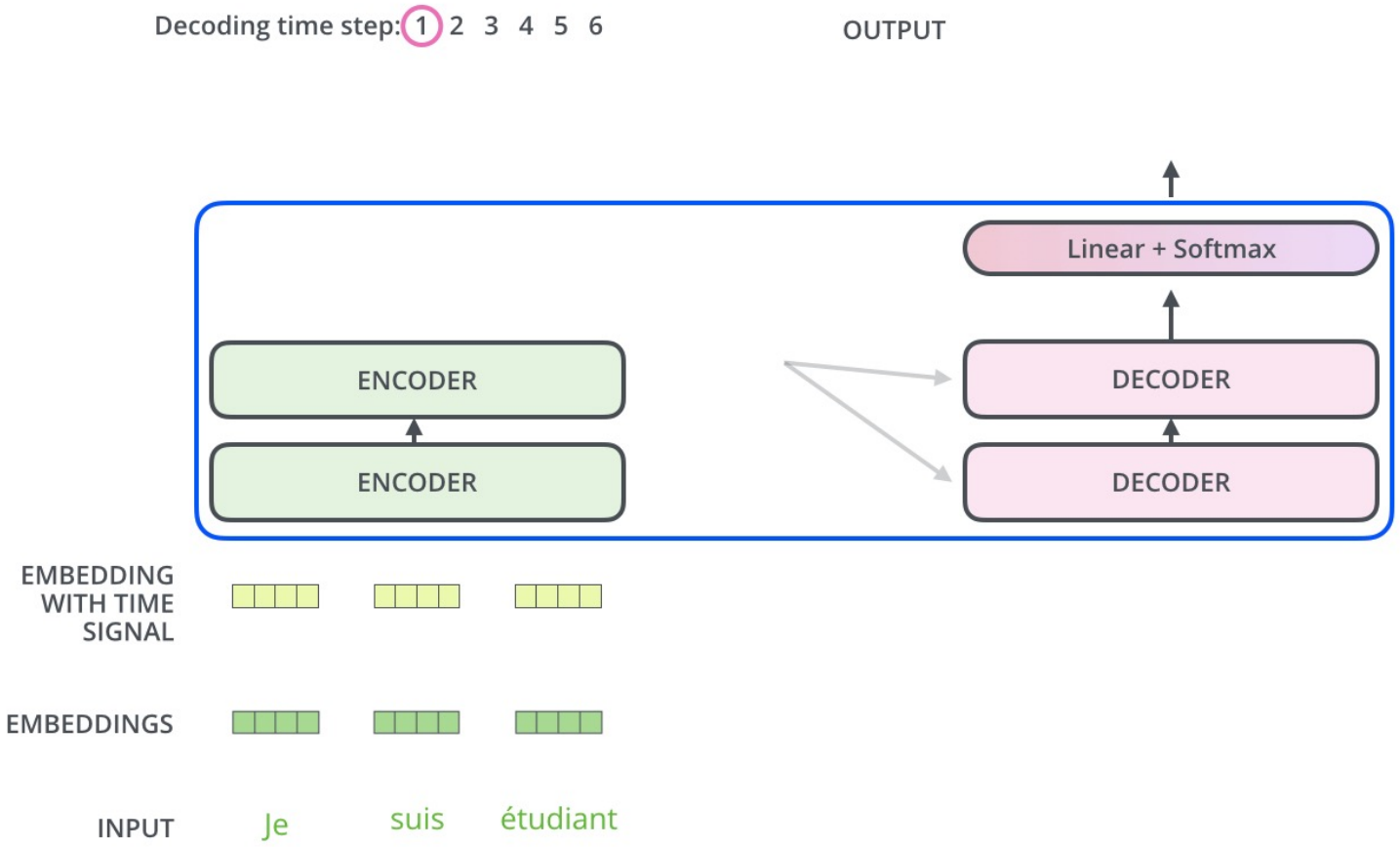
Residual connection:

- Each sub-layer (self-attention, feed forward neural network) in each encoder has a residual connection around it followed by a layer normalization



Decoder

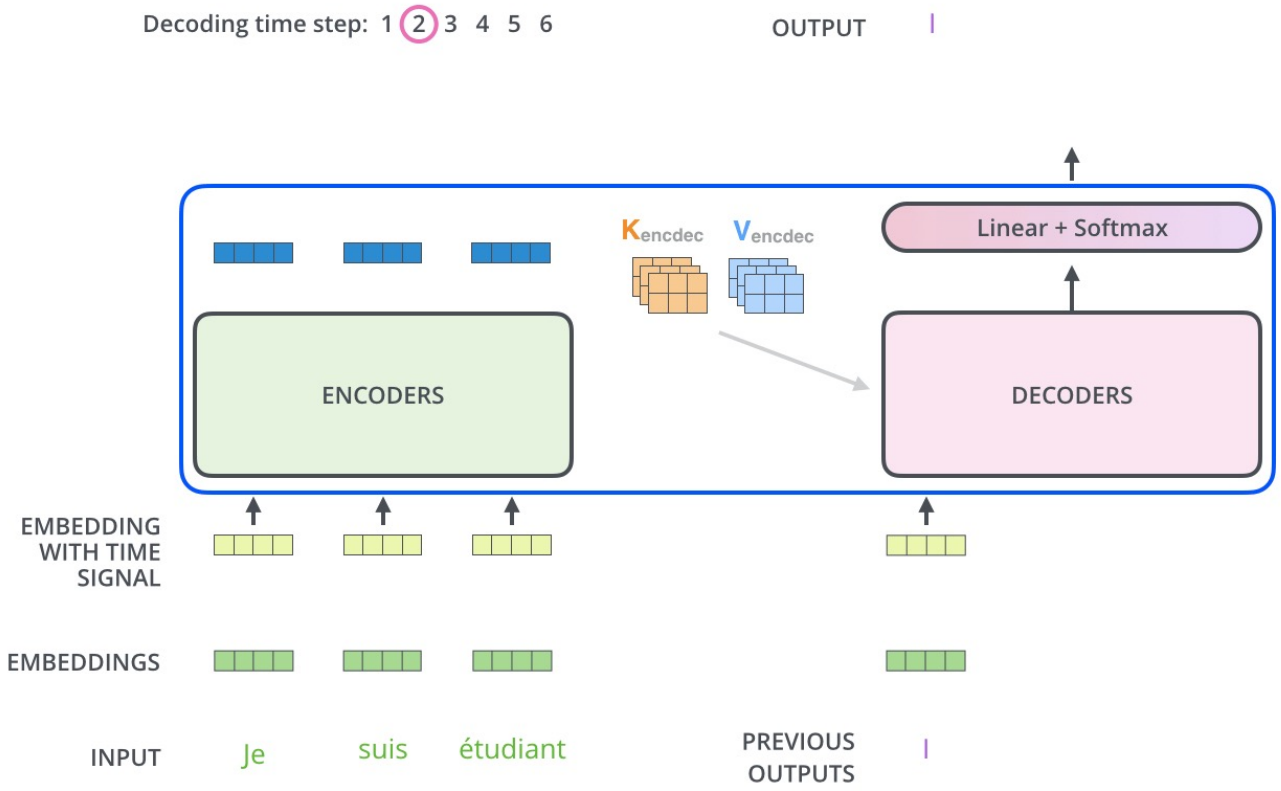
The output of the top encoder is used to as part input to each decoder



Decoder

Self attention layers in the decoder operate slightly different from that in the encoder

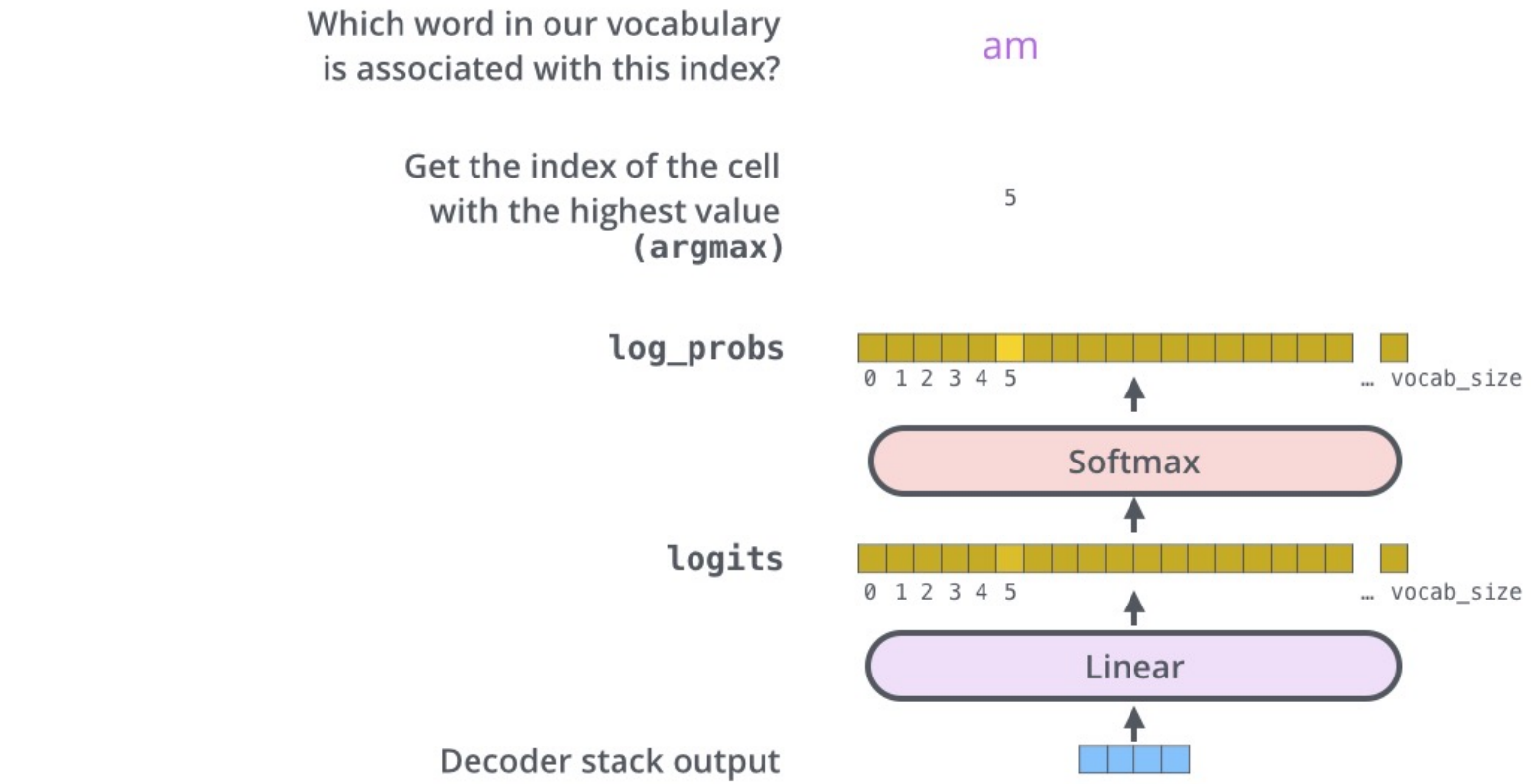
- self-attention layer is only allowed to attend to earlier positions in the output sequence.



Final Linear and SoftMax layer

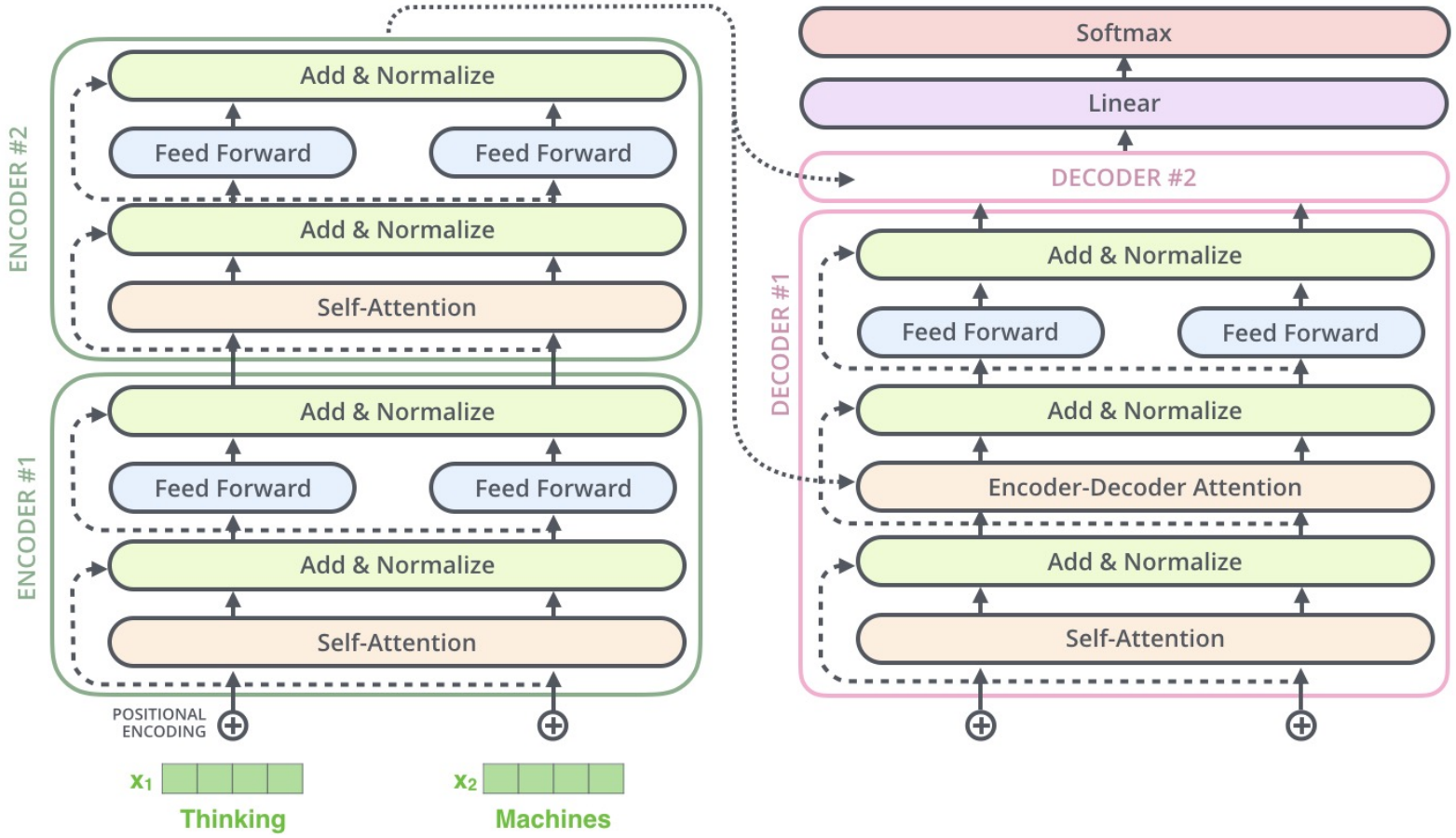
Final Linear Layer: **f**ully connected connected neural network that projects the vector produced by the stack of decoders into a logits vector.

SoftMax layer: convert the logit scores into a probability



Transformer architecture

Two Stacked encoder and decoders



Useful links

- Transformer model for language understanding
- Play with the Jupyter Notebook provided as part of the Tensor2Tensor
- Explore the Tensor2Tensor repo
- Huggingface
  - https://github.com/huggingface/transformers
  - https://huggingface.co/transformers/index.html