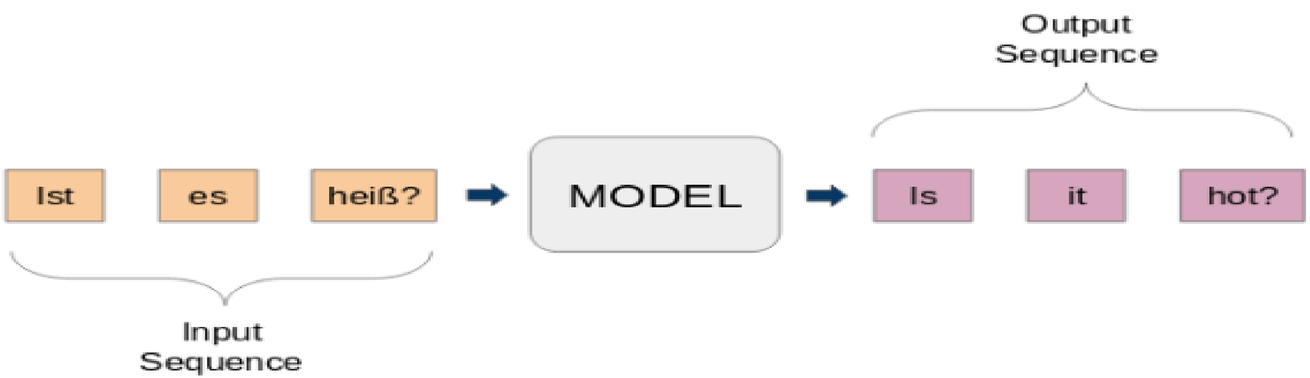
Neural Machine Translation

Language Translation

The objective is to convert a German sentence to its English counterpart using a Neural Machine Translation (NMT) system.

$(\text{Es regnet draußen})_{\text{German}}$ ➡ $(\text{It's raining outside})_{\text{English}}$

Sequence-to-Sequence (Seq2Seq) Modeling



Source:https://medium.com/analytics-vidhya/a-must-read-nlp-tutorial-on-neural-machine-translation-the-technique-powering-google-translate-c5c8d97d7587

Model Components

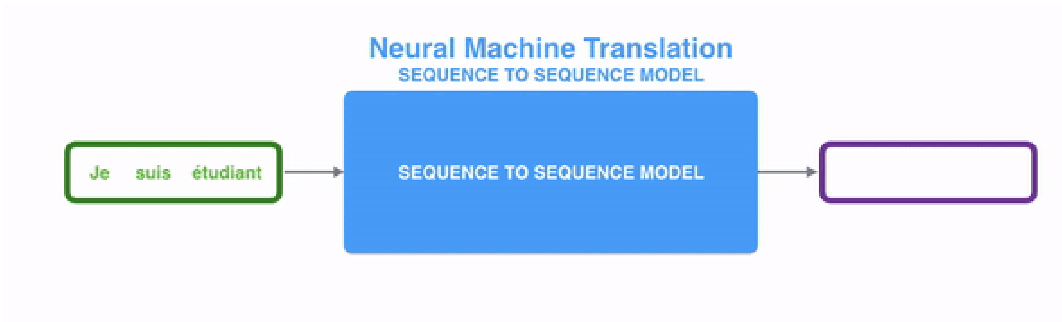
A typical seq2seq model has 2 major components —

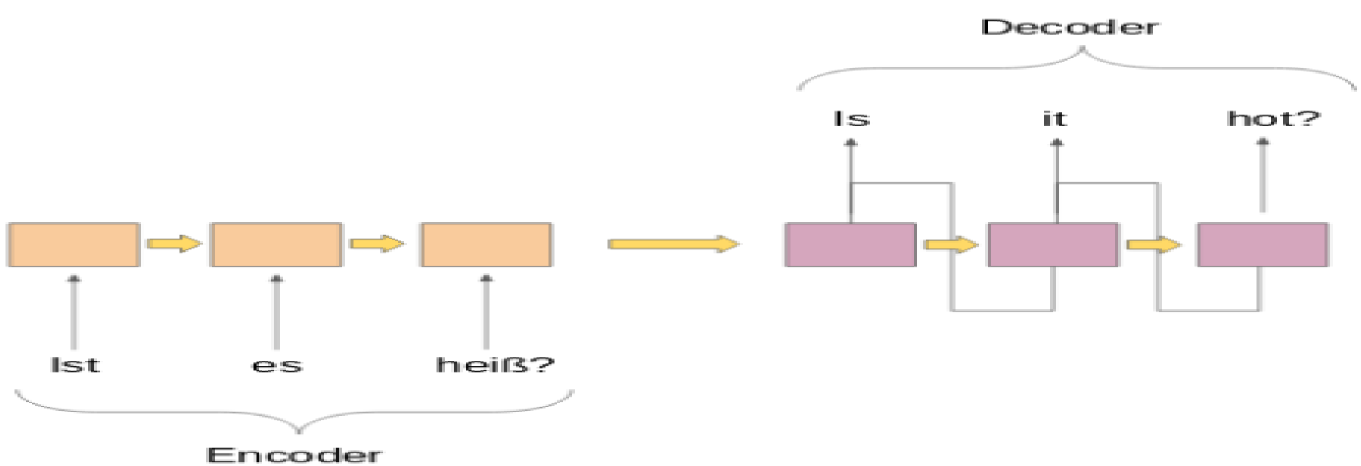
a) an encoder

b) a decoder

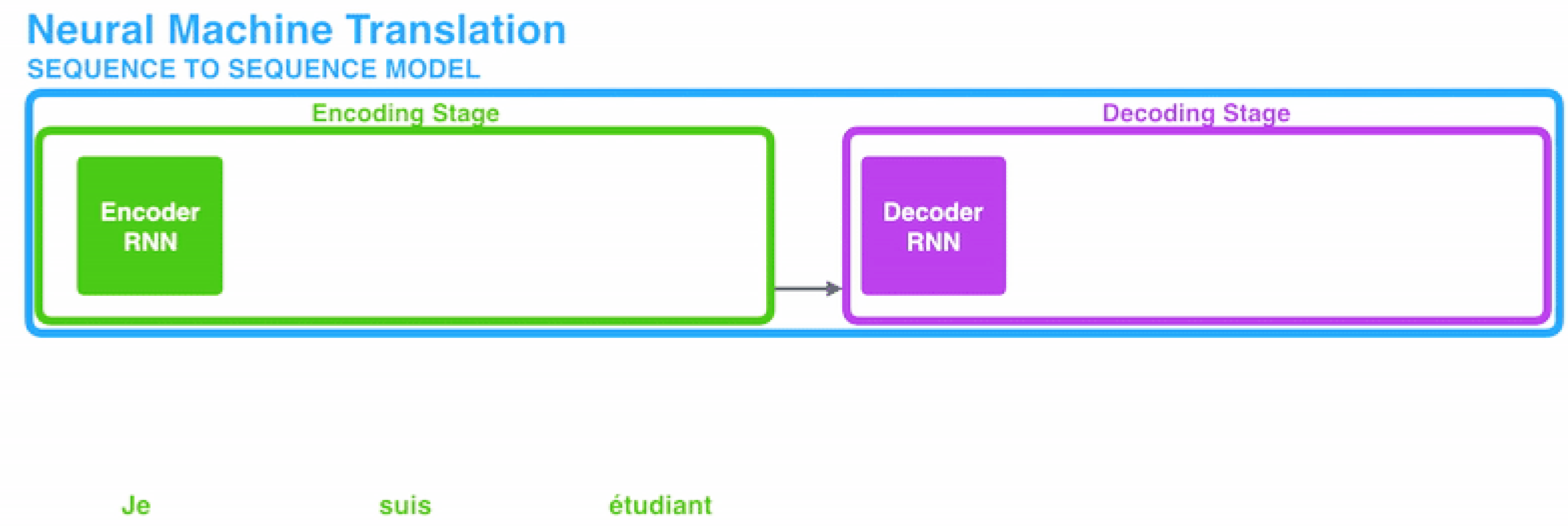
Both these parts are essentially two different RNN/ LSTM models combined into one giant network:

Encoder-Decoder





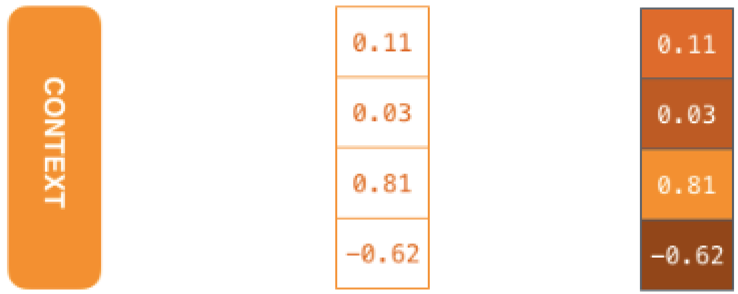
Encoder - Decoder



The encoder processes each item in the input sequence, it compiles the information it captures into a vector (called the context). After processing the entire input sequence, the encoder send the context over to the decoder, which begins producing the output sequence item by item

Source: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Context vector

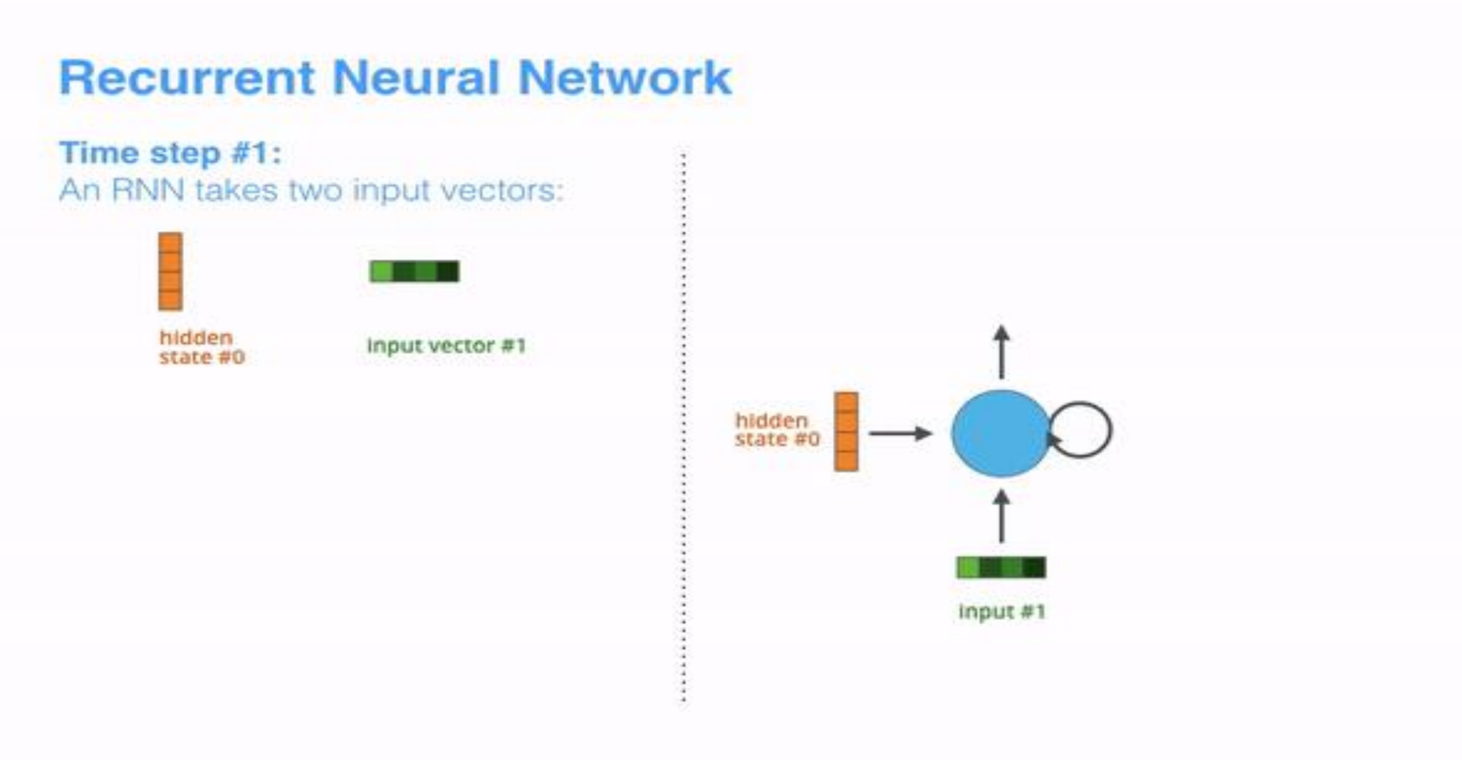


The context is a vector in the case of machine translation which basically represents the context information in a given sentence

You can set the size of the context vector when you set up your model. It is basically the number of hidden units in the encoder RNN. These visualizations show a vector of size 4, but in real world applications the context vector would be of a size like 256, 512, or 1024.

Source: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

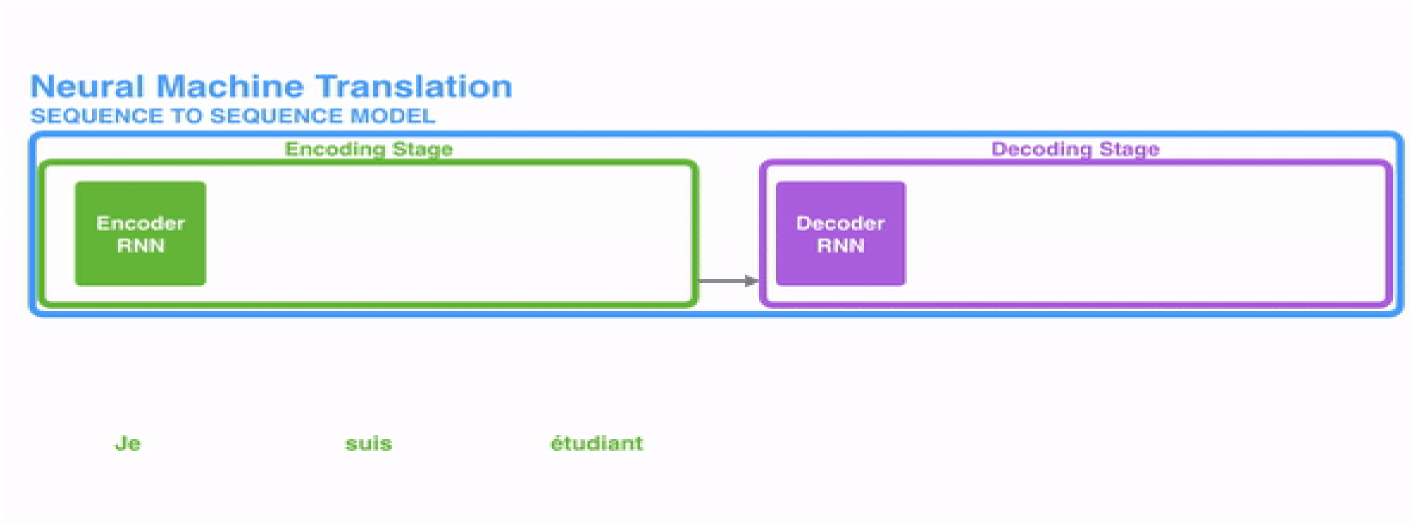
How it works?



The next RNN step takes the second input vector and hidden state #1 to create the output of that time step

Source: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

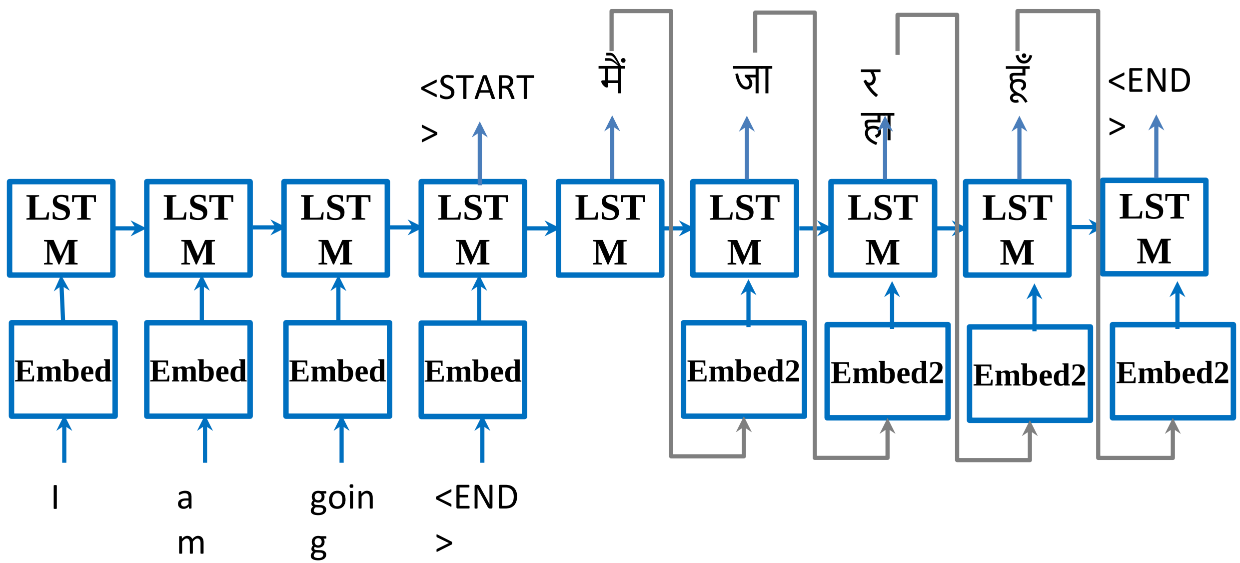
How it works?



- Encoder or decoder is that RNN/LSTM processing its inputs and generating an output for that time step.
- Since the encoder and decoder are both same units, each time step one of the units does some processing, updates its hidden state based on its inputs and previous inputs it has seen
- The decoder also maintains a hidden states that it passes from one time step to the next

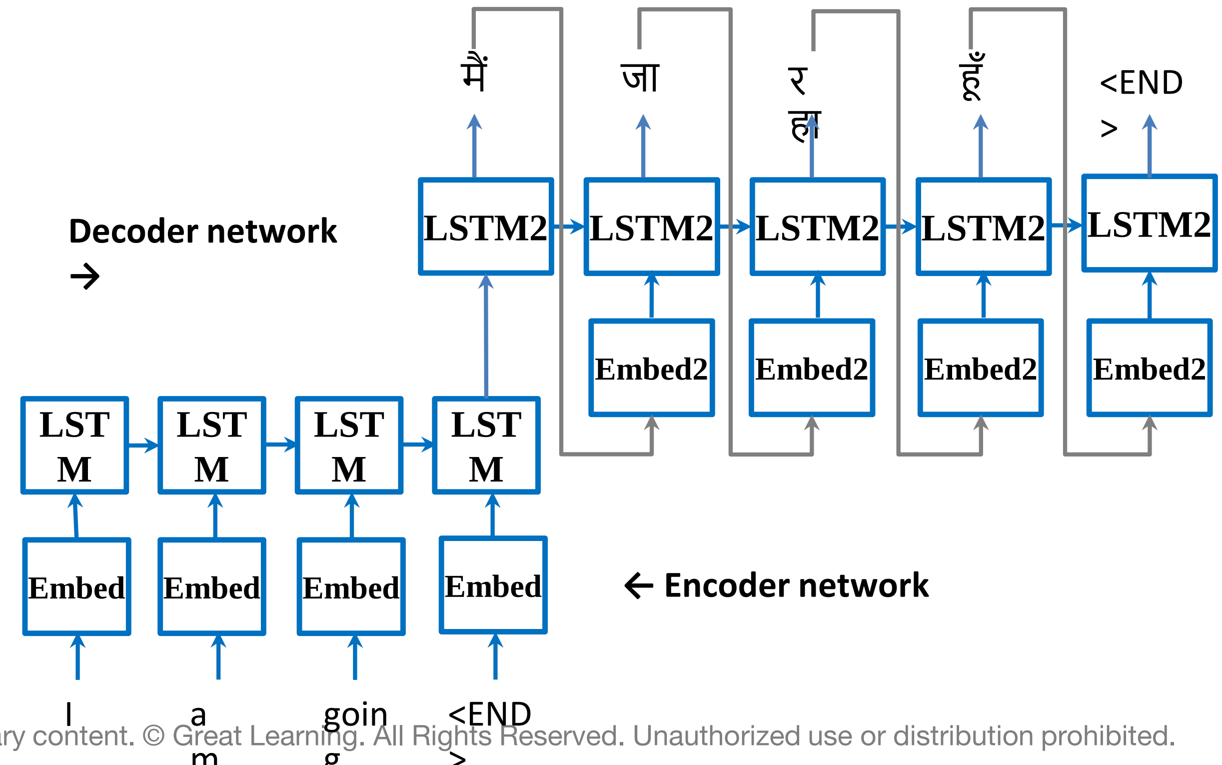
Machine translation - with LSTM units

- One could also feed in the output to the next instance input to predict a coherent structure



Machine translation

- In actuality, one can also use separate LSTMs pre-trained on two different languages



Lets look into the implementation using Keras in Jupyter Notebook!!

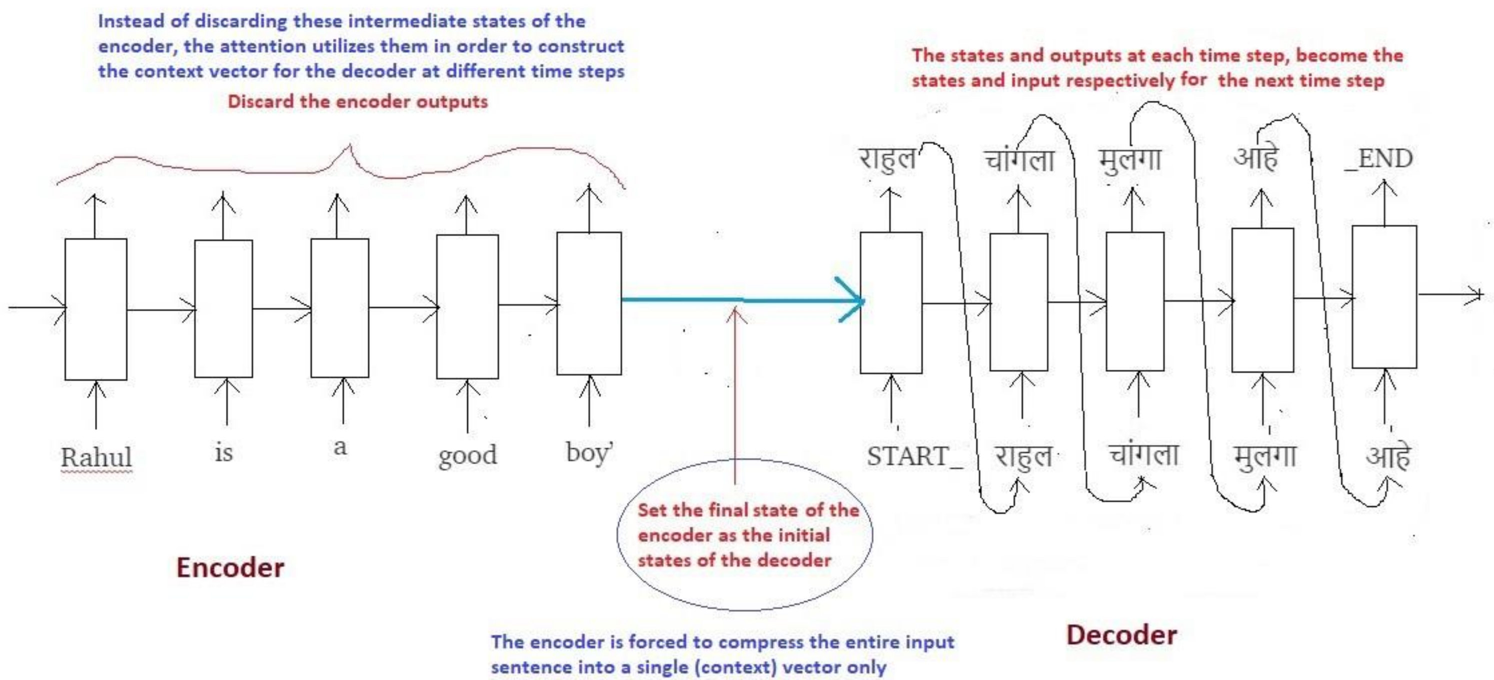
Attention Models

What's wrong with seq2seq models?

The seq2seq models is normally composed of an encoder-decoder architecture, where the encoder processes the input sequence and encodes/compresses/summarizes the information into a context vector (also called as the "thought vector") of a fixed length.

A critical and apparent disadvantage of this fixed-length context vector design is the incapability of the system to remember longer sequences.

Seq2Seq Model



Concept of Attention

When you predict “राहुल”, it's obvious that this name is the result of the word “Rahul” present in the input English sentence regardless of the rest of the sentence. We say that while predicting “राहुल”, we pay more attention to the word “Rahul” in the input sentence.

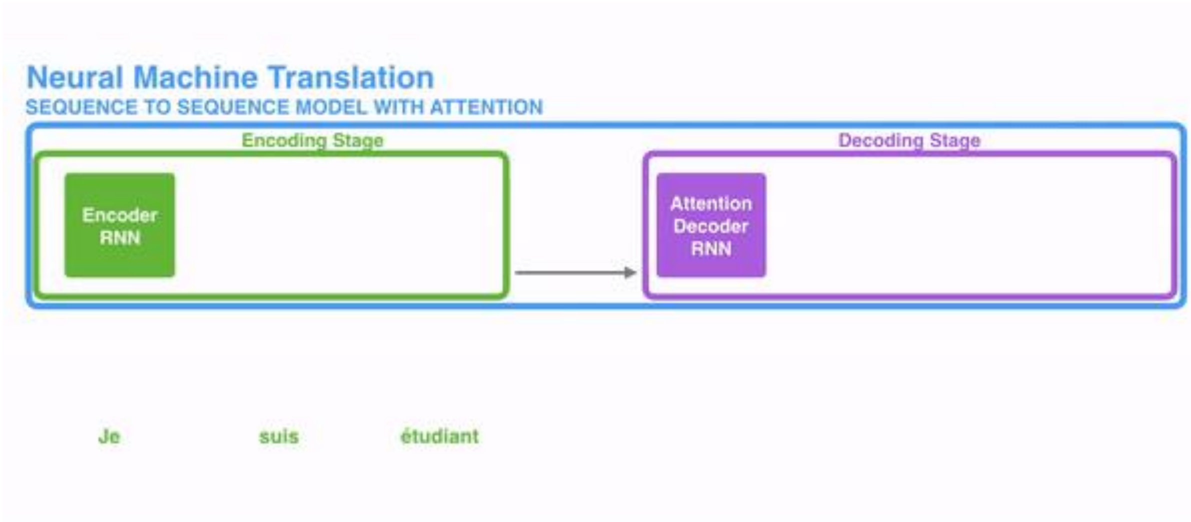
Similarly while predicting the word “चांगला”, we pay more attention to the word “good” in the input sentence and so on.

Hence the name “ATTENTION”.

The central idea behind Attention

- As human beings we are quickly able to understand these mappings between different parts of the input sequence and corresponding parts of the output sequence. However it’s not that straight-forward for neural networks to automatically detect these mappings.
- Thus the Attention mechanism is developed to “learn” these mappings through Gradient Descent and Back-propagation.
- The central idea behind Attention is not to throw away those intermediate encoder states but to utilize all the states in order to construct the context vectors required by the decoder to generate the output sequence.

NMT with attention mechanism

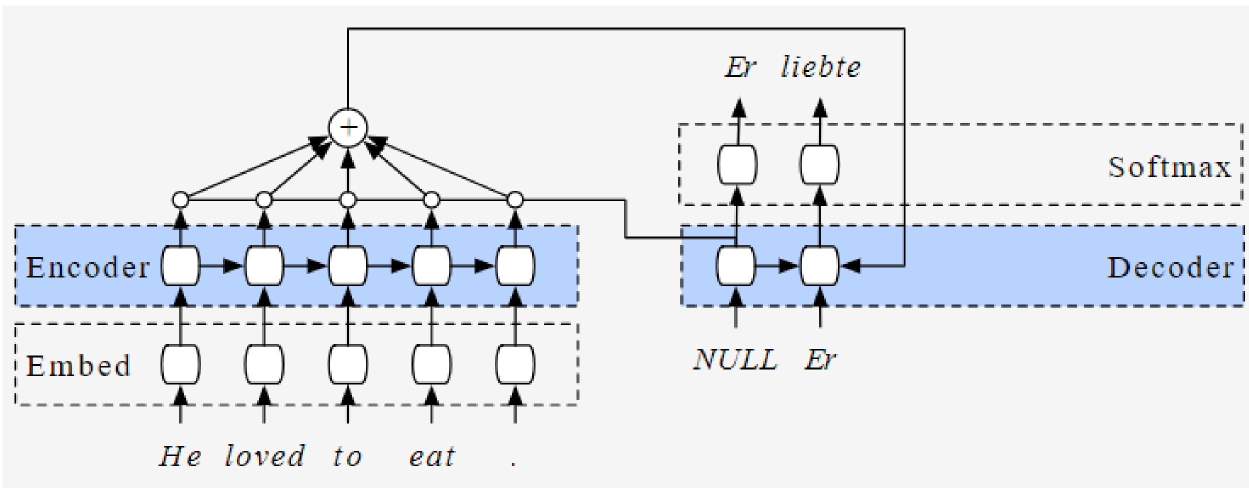


An attention model differs from a classic sequence-to-sequence model in two main ways:

First, the encoder passes a lot more data to the decoder. Instead of passing the last hidden state of the encoding stage, the encoder passes *all* the hidden states to the decoder:

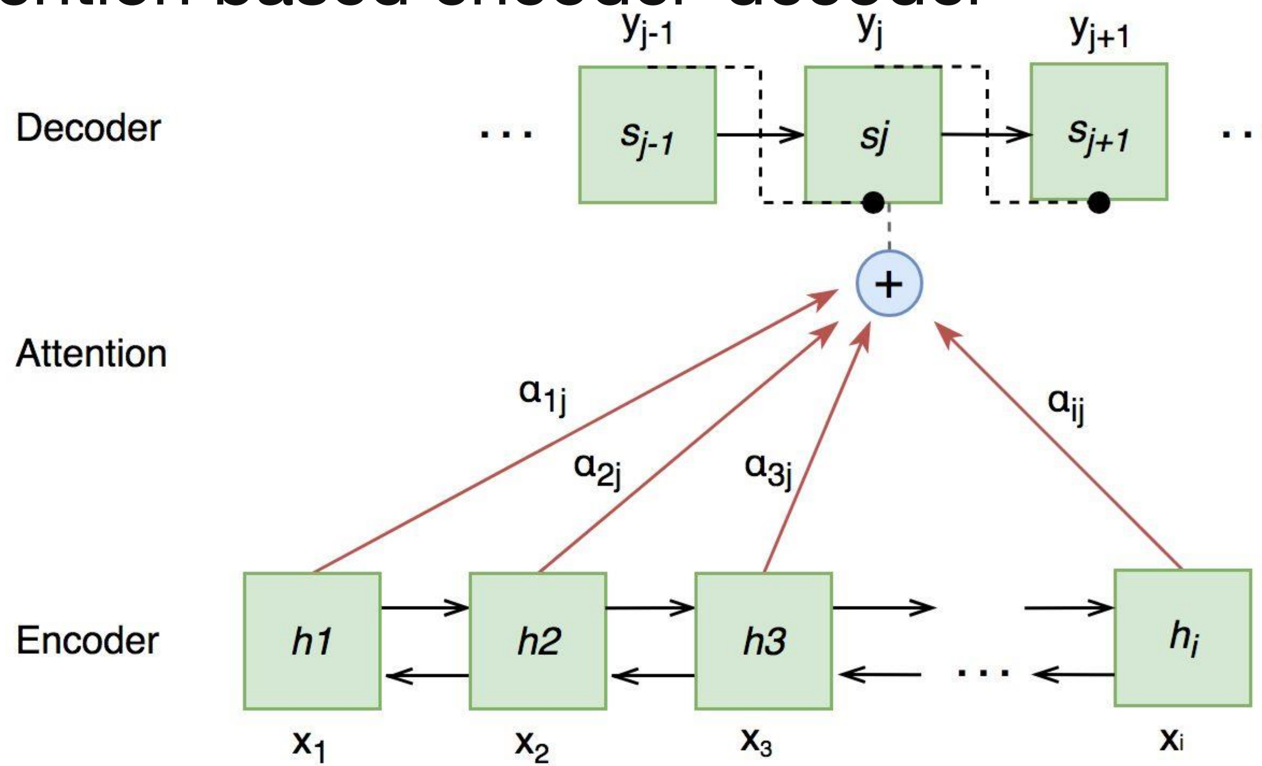
Source: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Attention based encoder-decoder



Source: https://smerity.com/articles/2016/google_nmt_arch.html

Attention based encoder-decoder



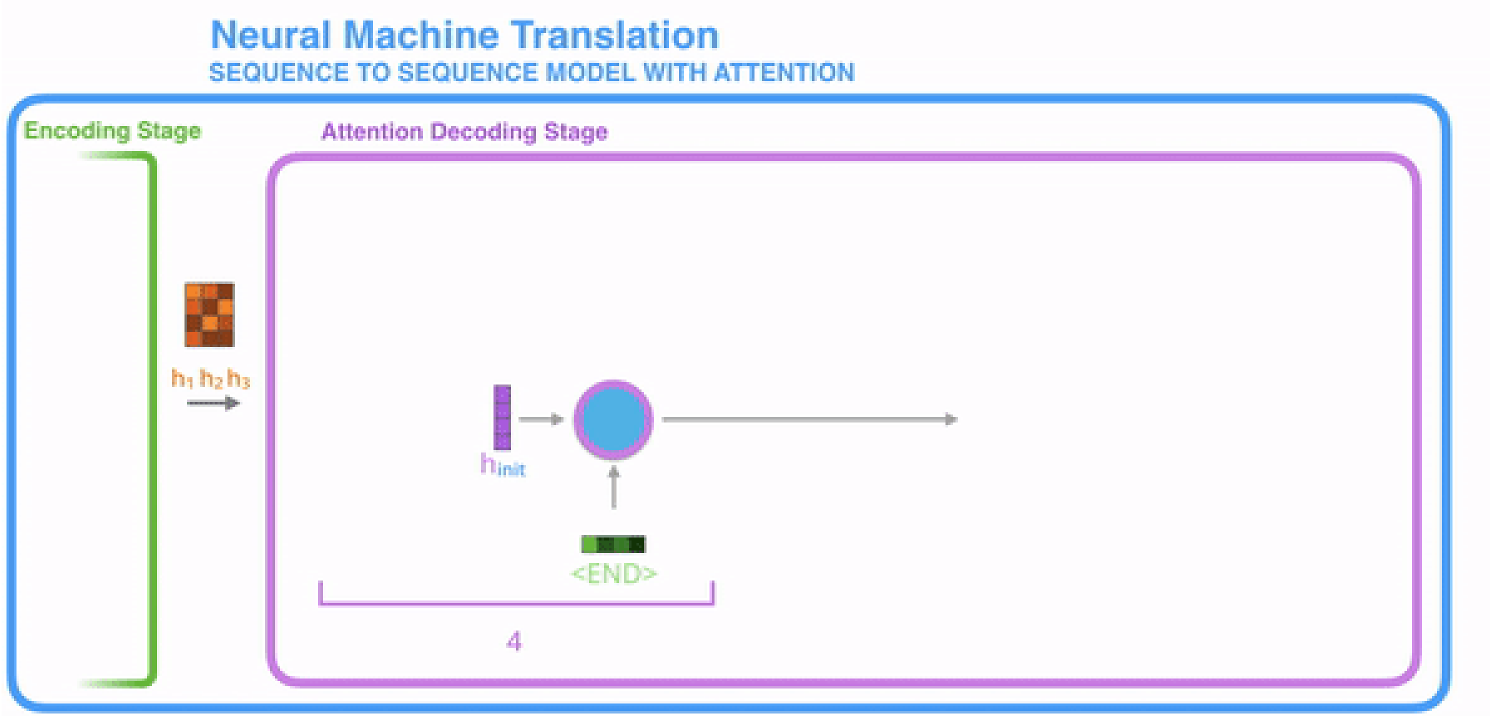
Attention mechanism

Attention at time step 4

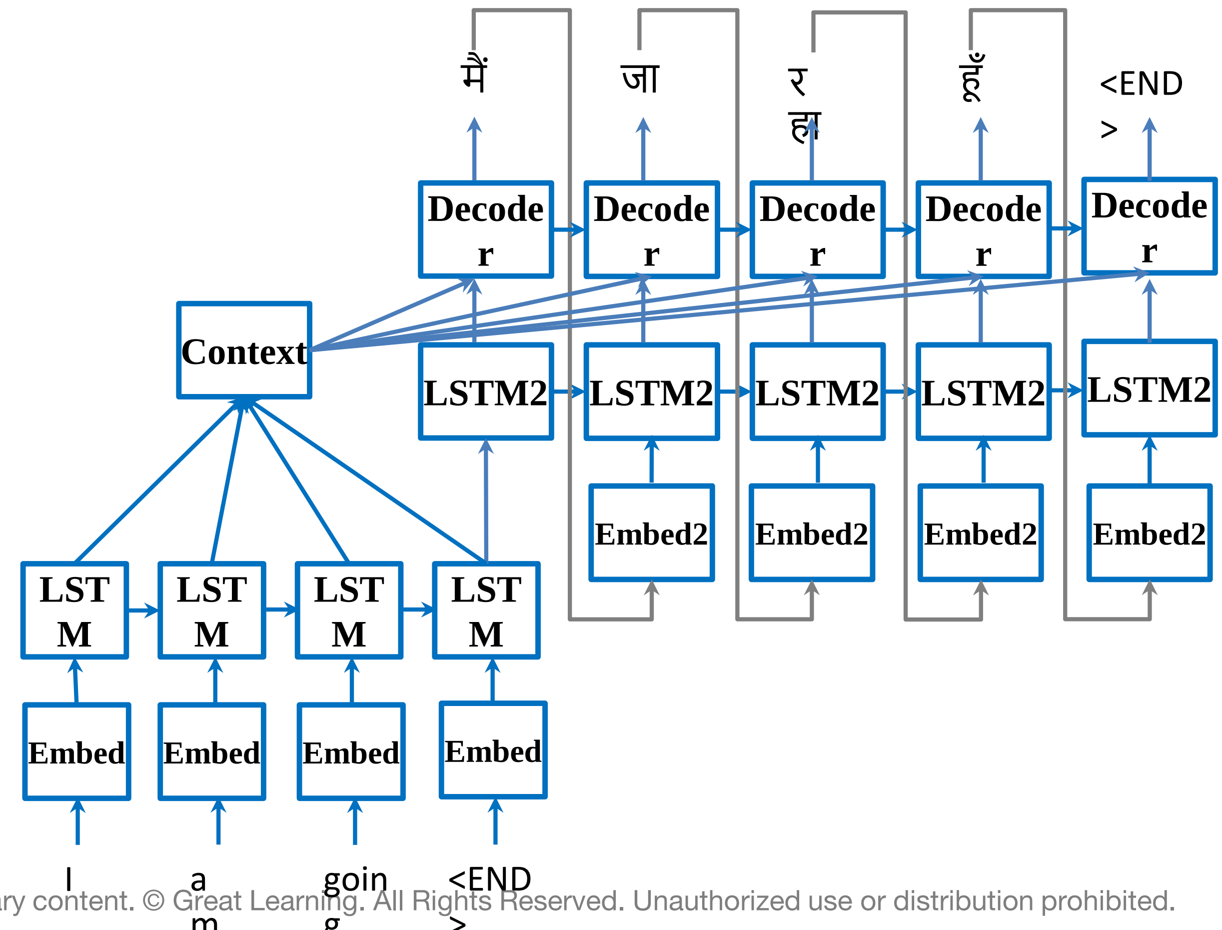
Attention decoder does an extra step before producing its output. In order to focus on the parts of the input that are relevant to this decoding time step, the decoder does the following:

1. Look at the set of encoder hidden states it received – each encoder hidden states is most associated with a certain word in the input sentence
2. Give each hidden states a score (let’s ignore how the scoring is done for now)
3. Multiply each hidden states by its softmaxed score, thus amplifying hidden states with high scores, and drowning out hidden states with low scores

Attention mechanism



NMT with attention - Total architecture

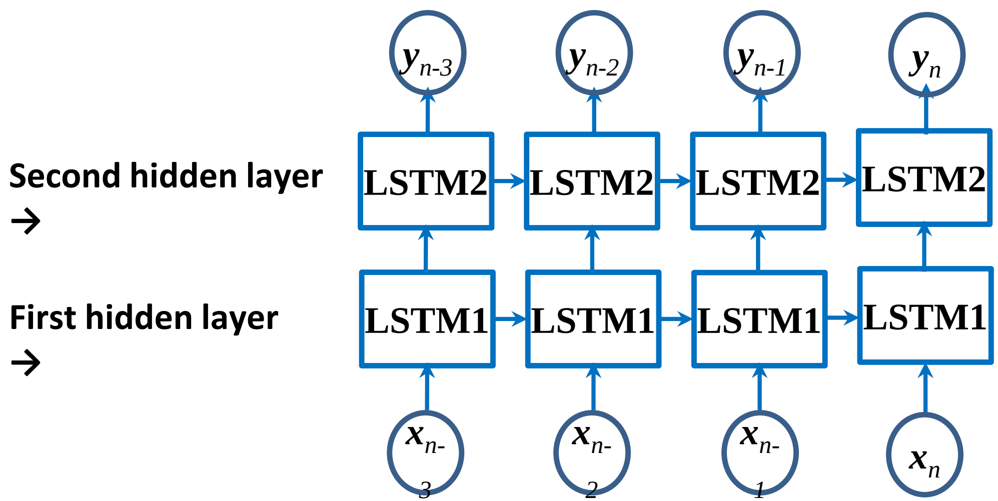


Lets look into the implementation using Keras in Jupyter Notebook!!

Advanced LSTM structures

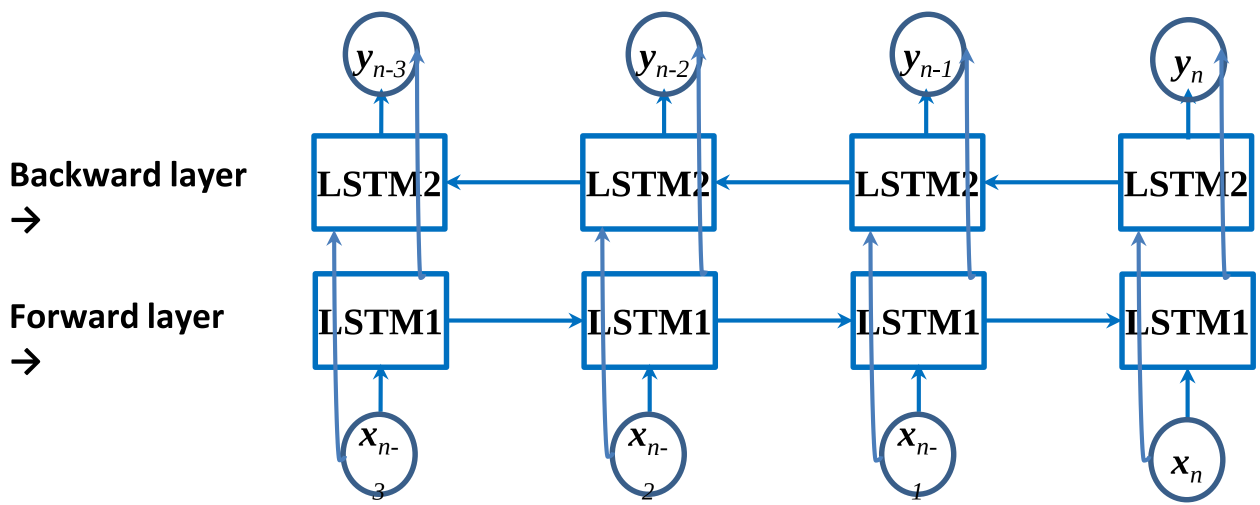
Multi-layer LSTM

- More than one hidden layer can be used



Bi-directional LSTM

- Many problems require a reverse flow of information as well
- For example, POS tagging may require context from future words



Some problems in LSTM and its troubleshooting

- Inappropriate model
  - Identify the problem: One-to-many, many-to-one etc.
  - Loss only for outputs that matter
  - Separate LSTMs for separate languages

- High training loss
  - Model not expressive
    - Too few hidden nodes
    - Only one hidden layer

- Overfitting
  - Model has too much freedom
    - Too many hidden nodes
    - Too many blocks
    - Too many layers
    - Not bi-directional

References

https://towardsdatascience.com/intuitive-understanding-of-attention-mechanism-in-deep-learning-6c9482aecf4f

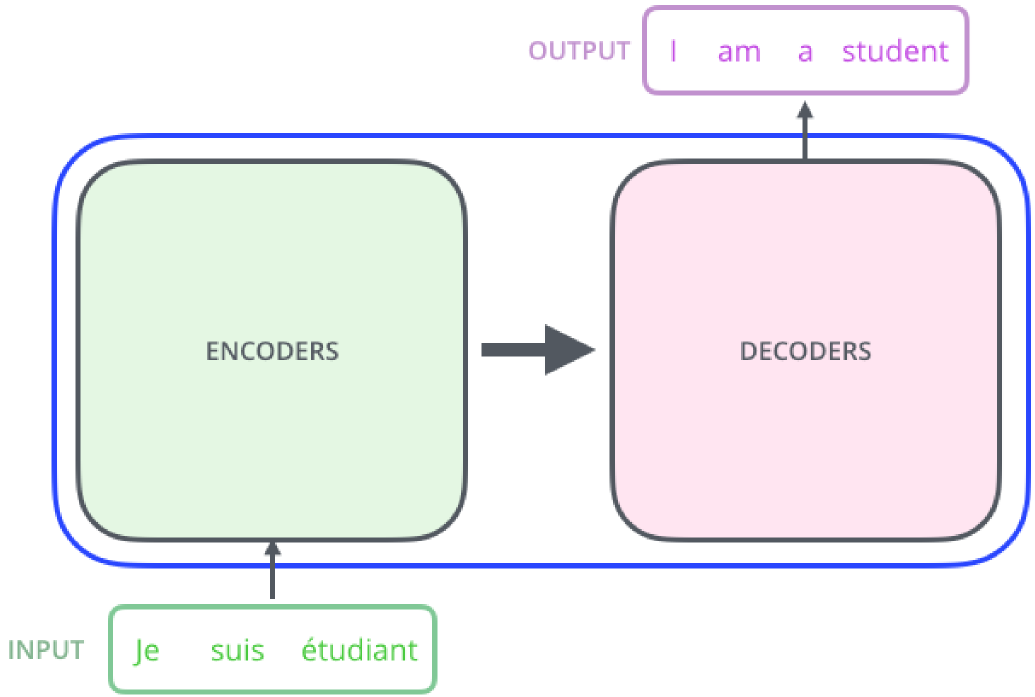
https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/

Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification -https://www.aclweb.org/anthology/P16-2034

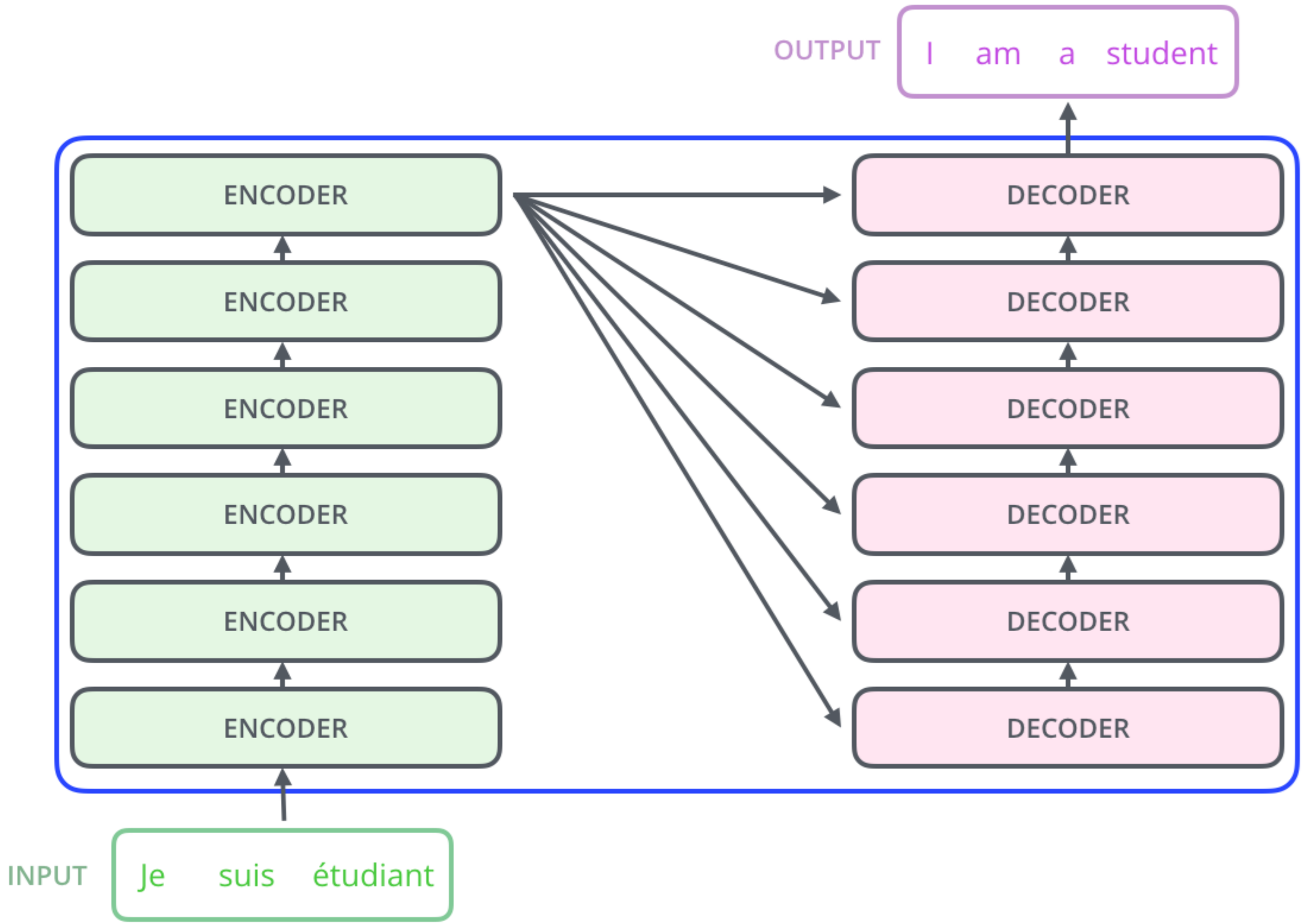
http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

APPENDIX
TRANSFORMER, BERT and ELMO

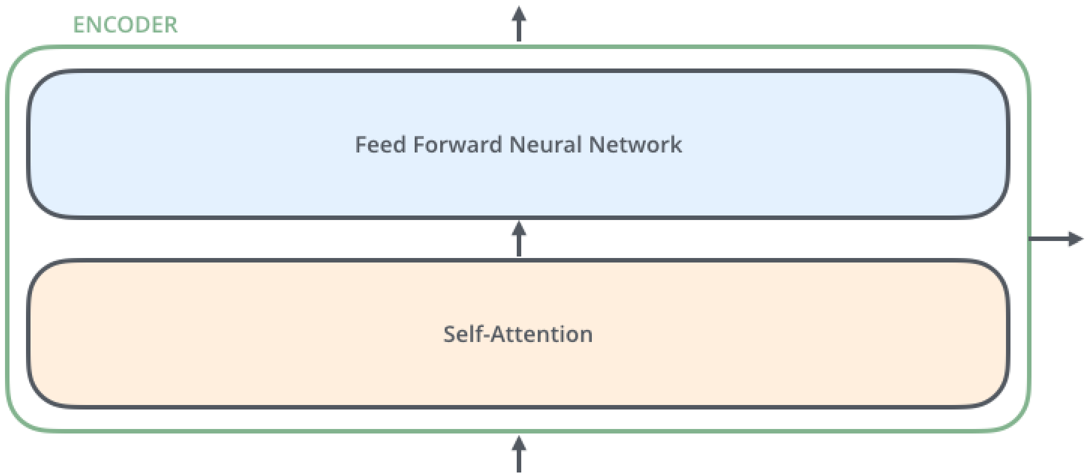
What is a Transformer…?



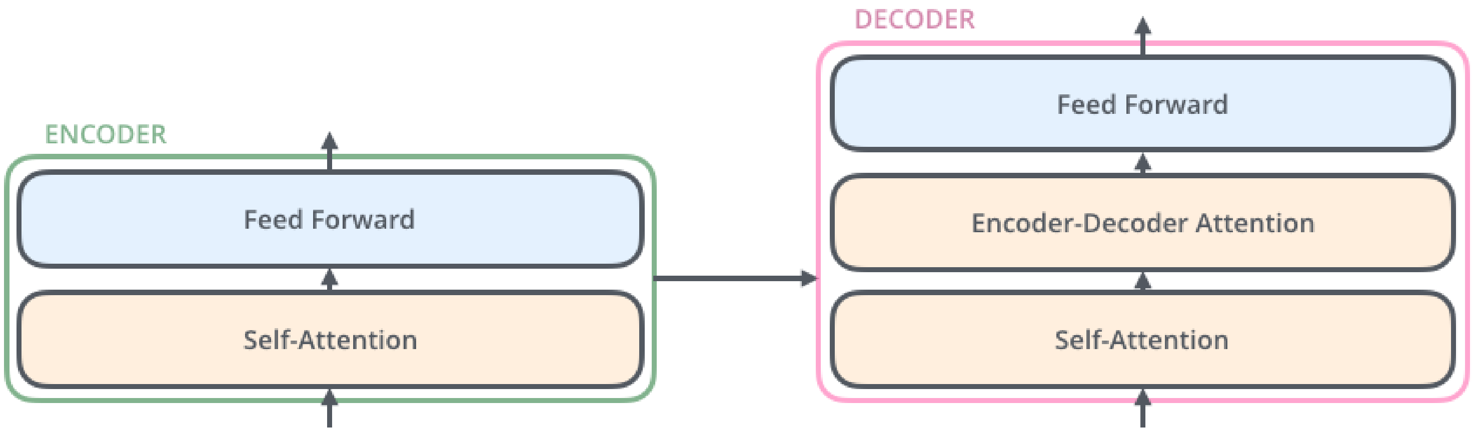
What is a Transformer…?



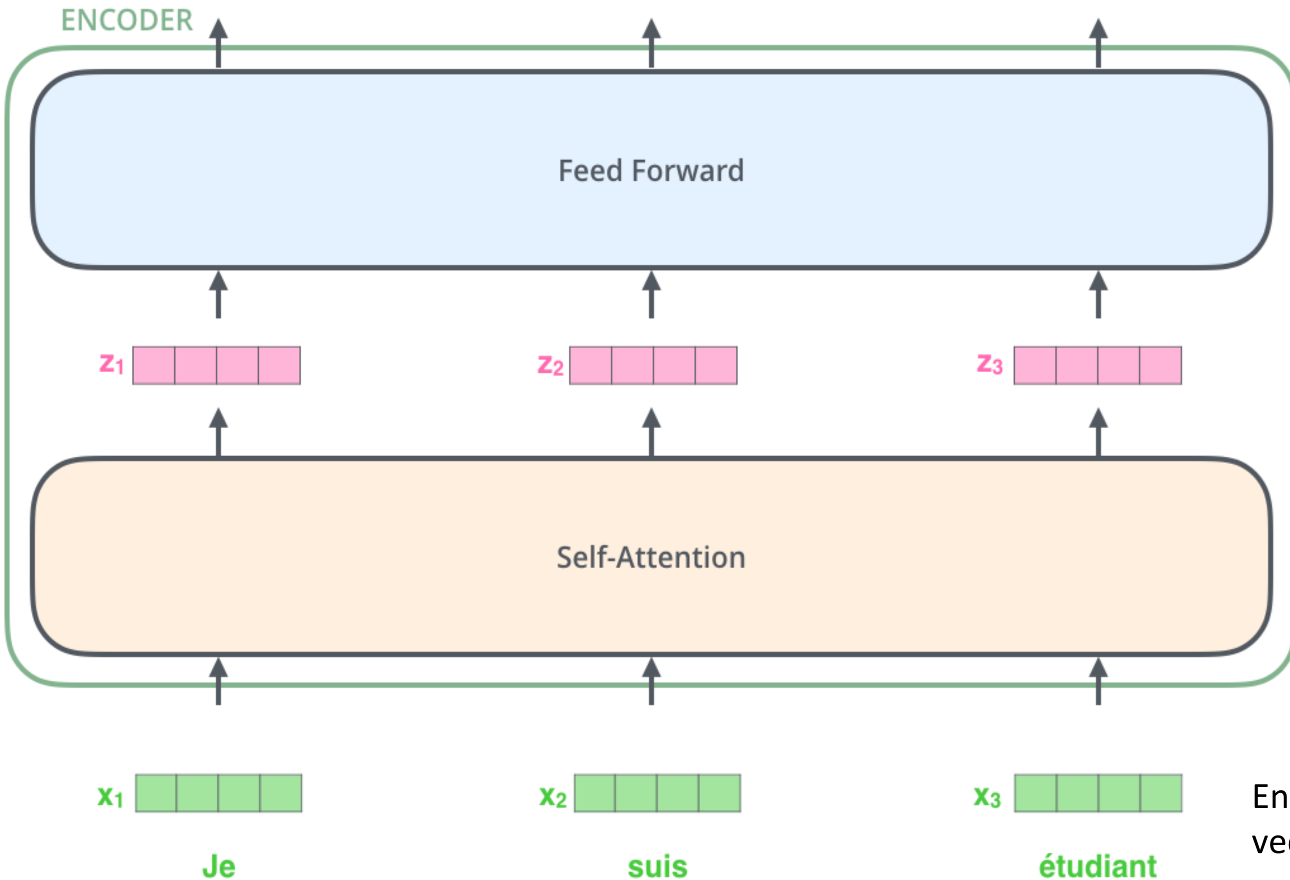
What is a Transformer…?



What is a Transformer…?

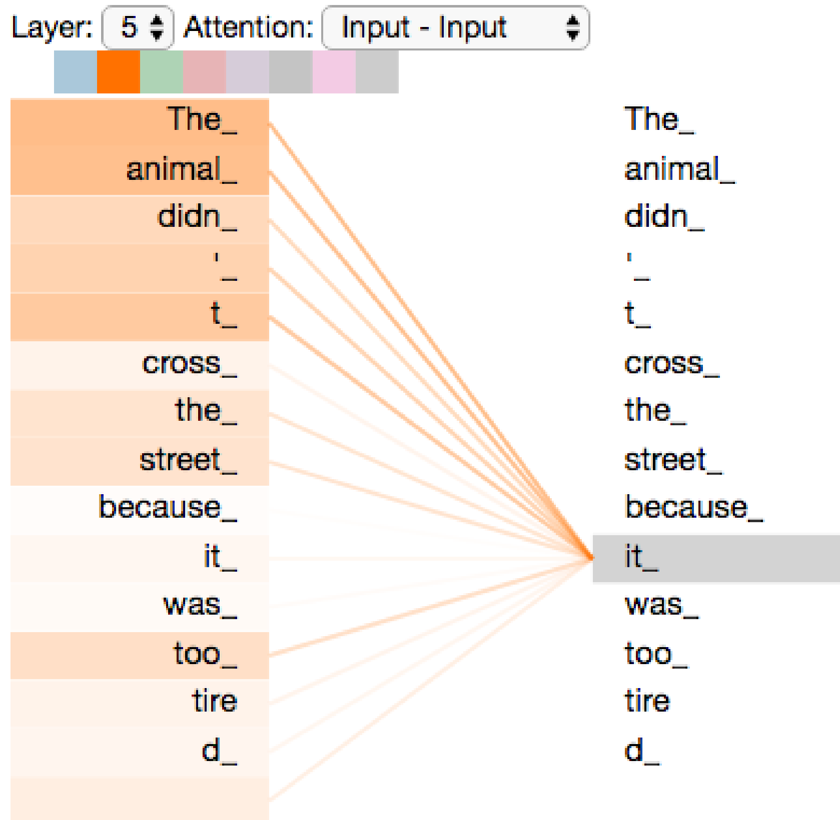


What is a Transformer…?

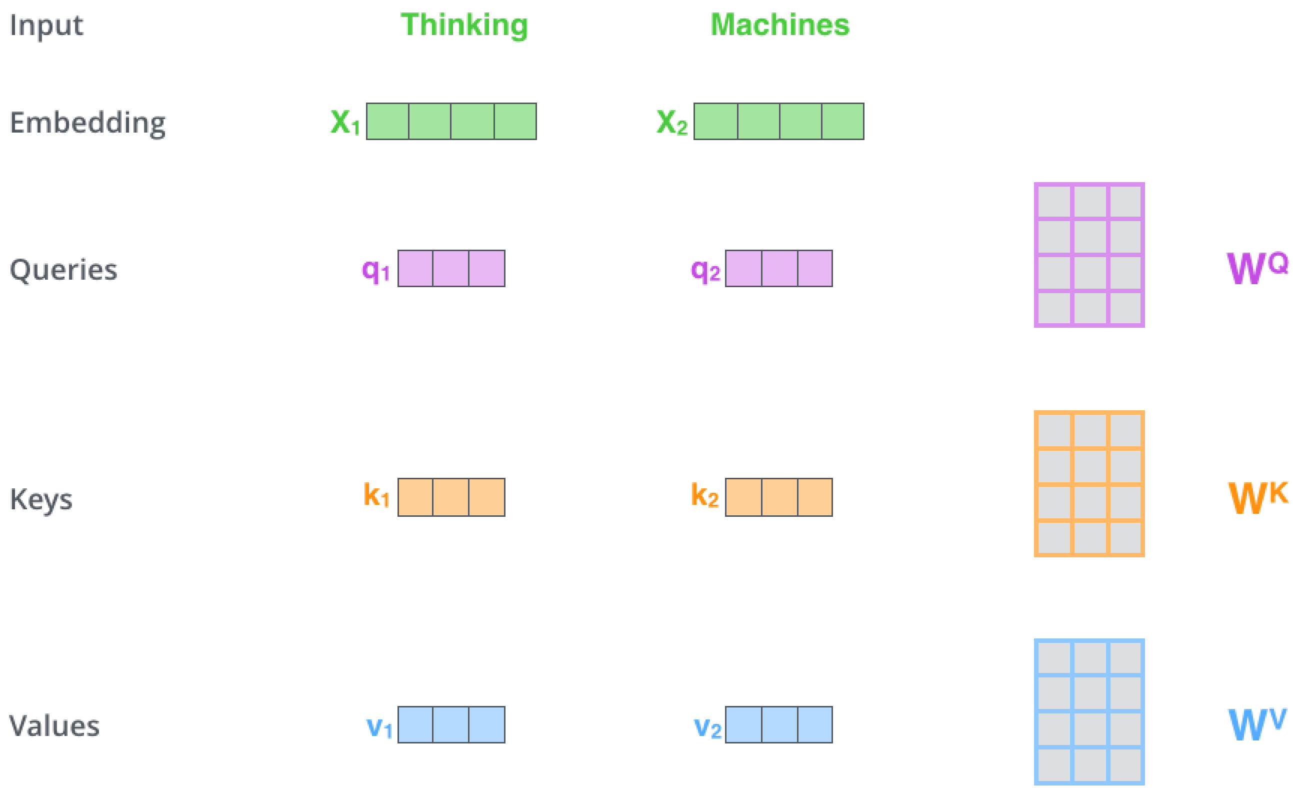


Encoding int0 512-d vectors

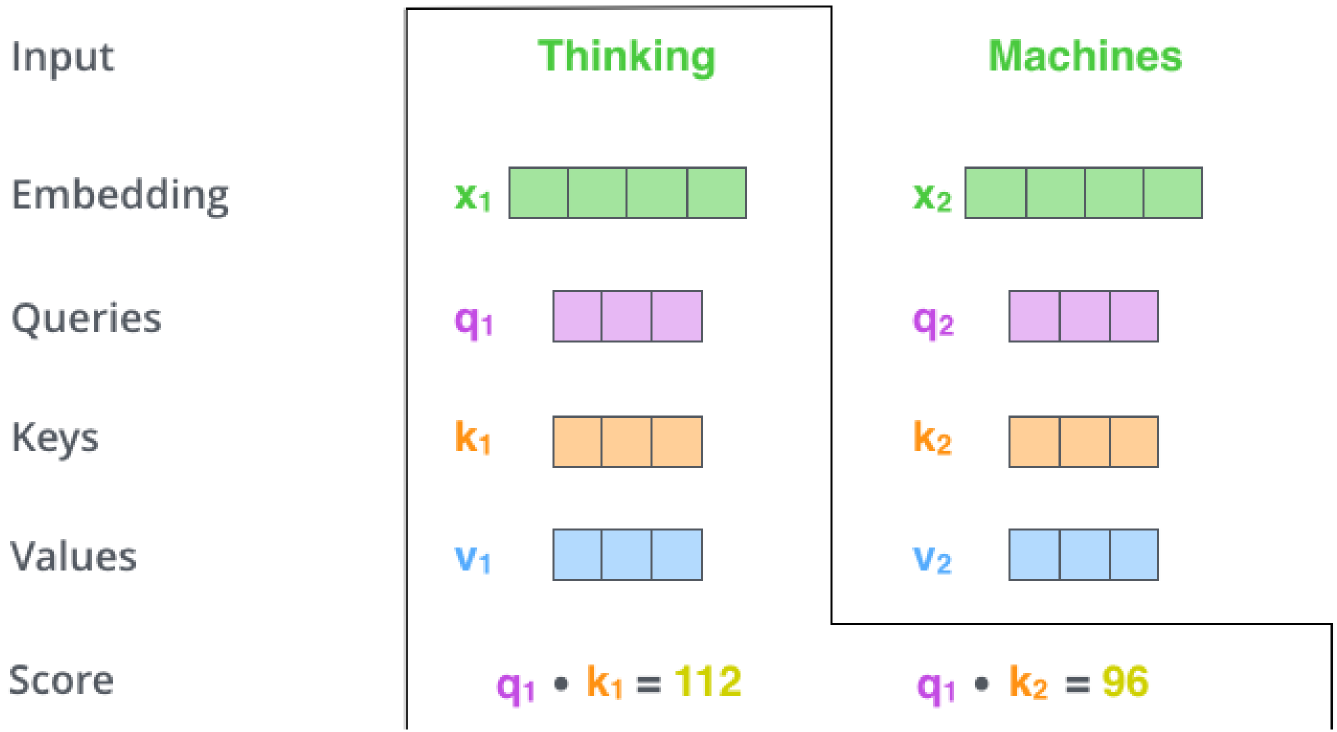
What is a Transformer…? Self Attention…



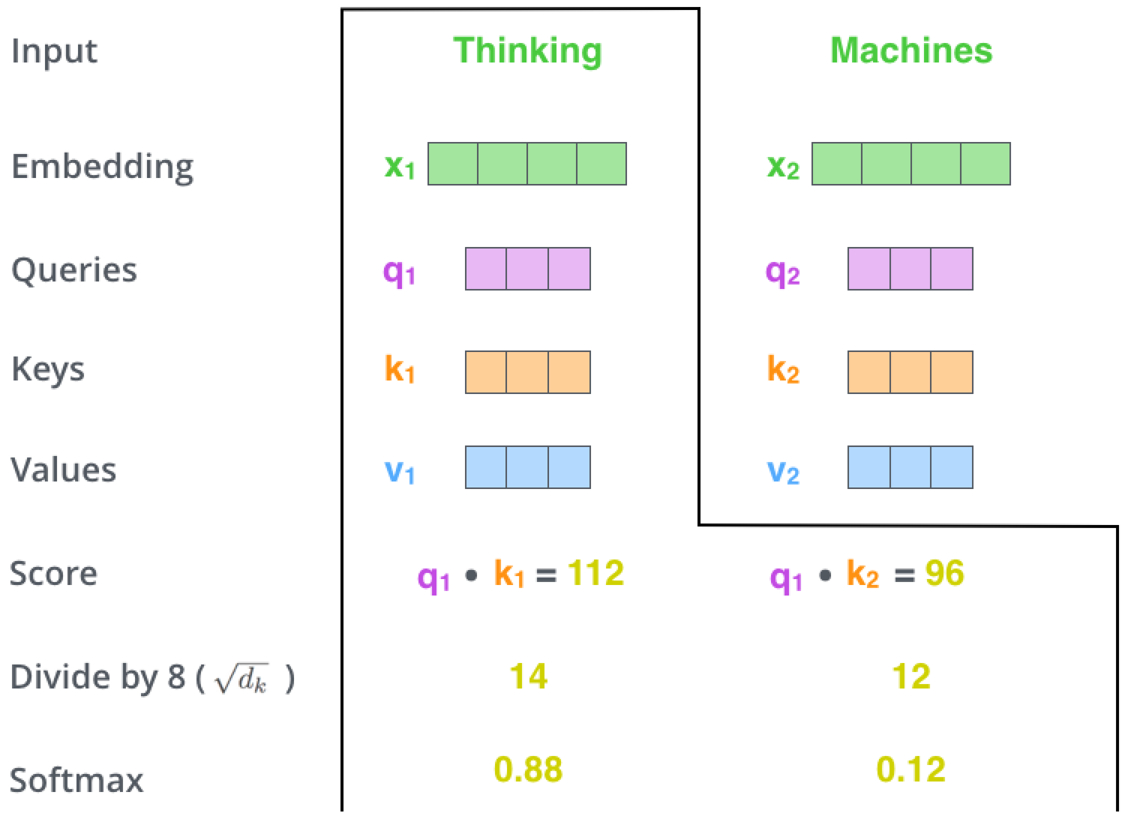
Self Attention in Detail…



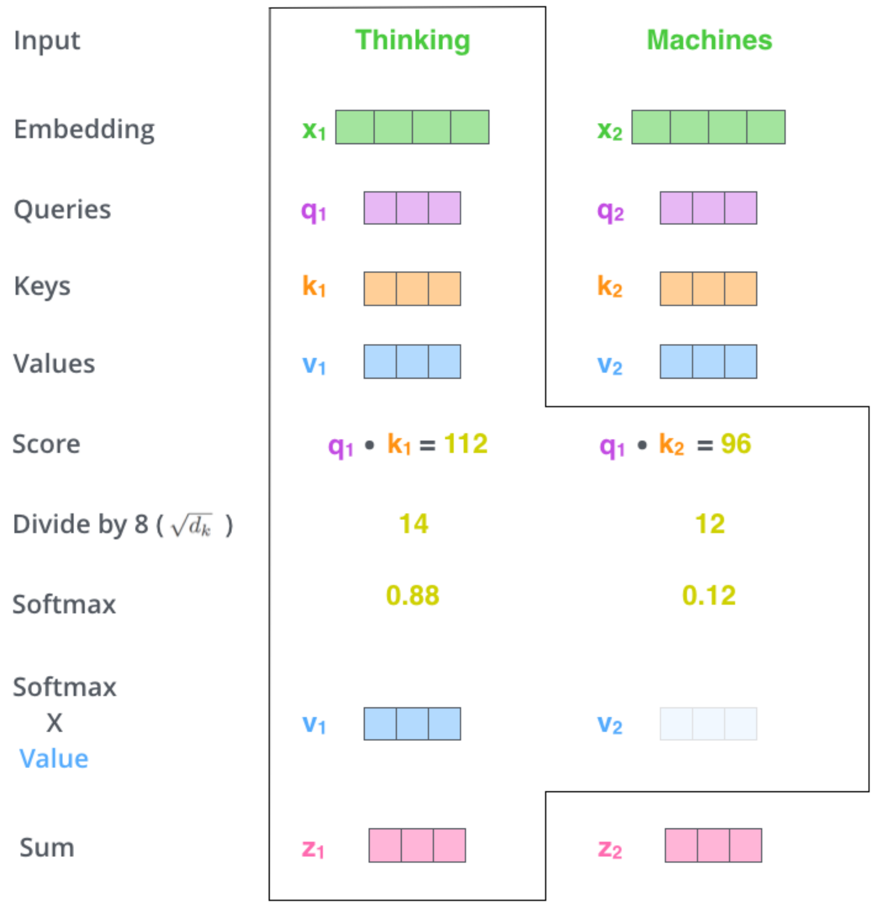
Self Attention in Detail…



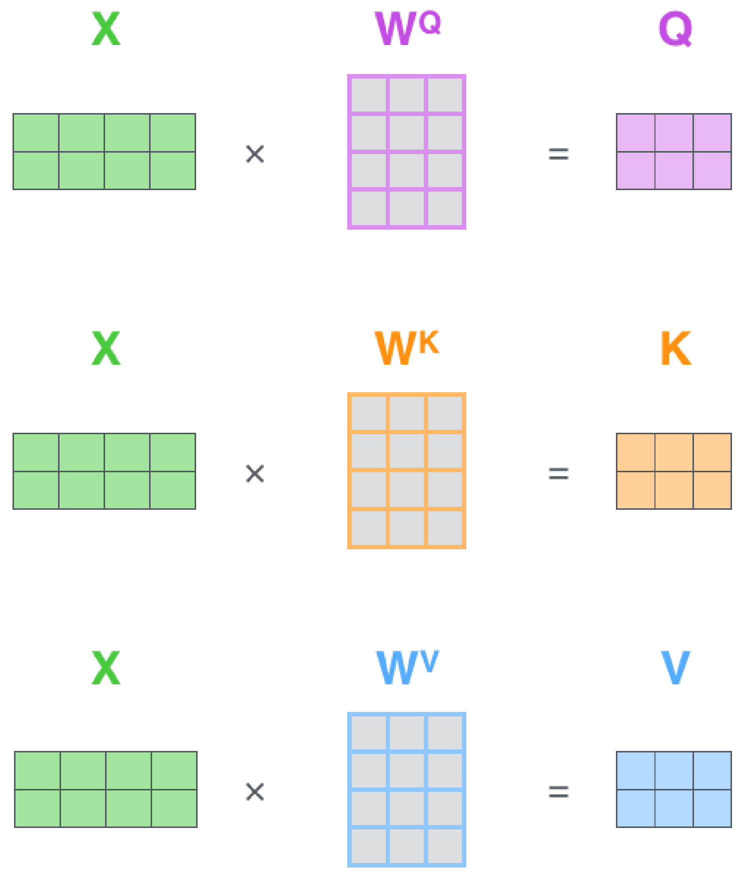
Self Attention in Detail…



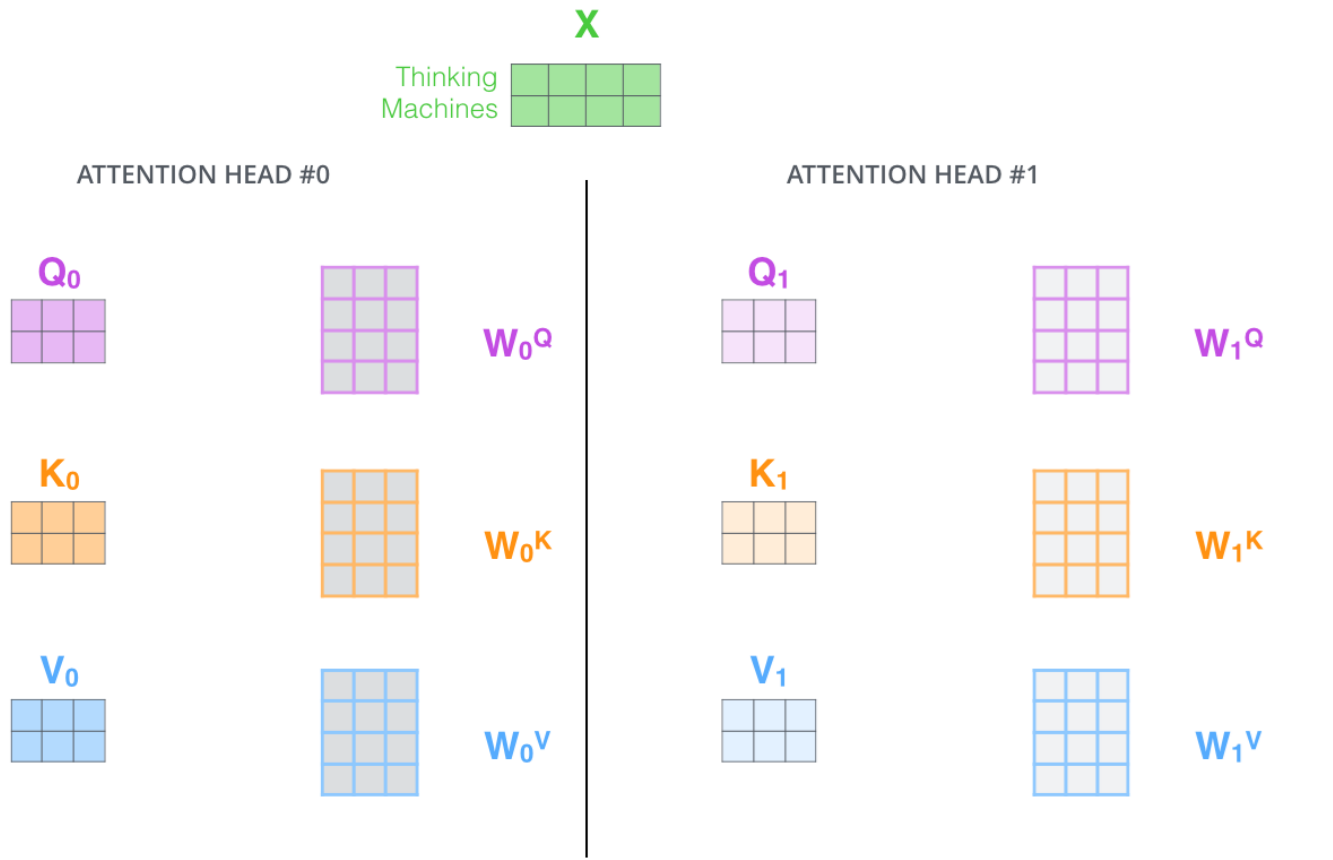
Self Attention in Detail…



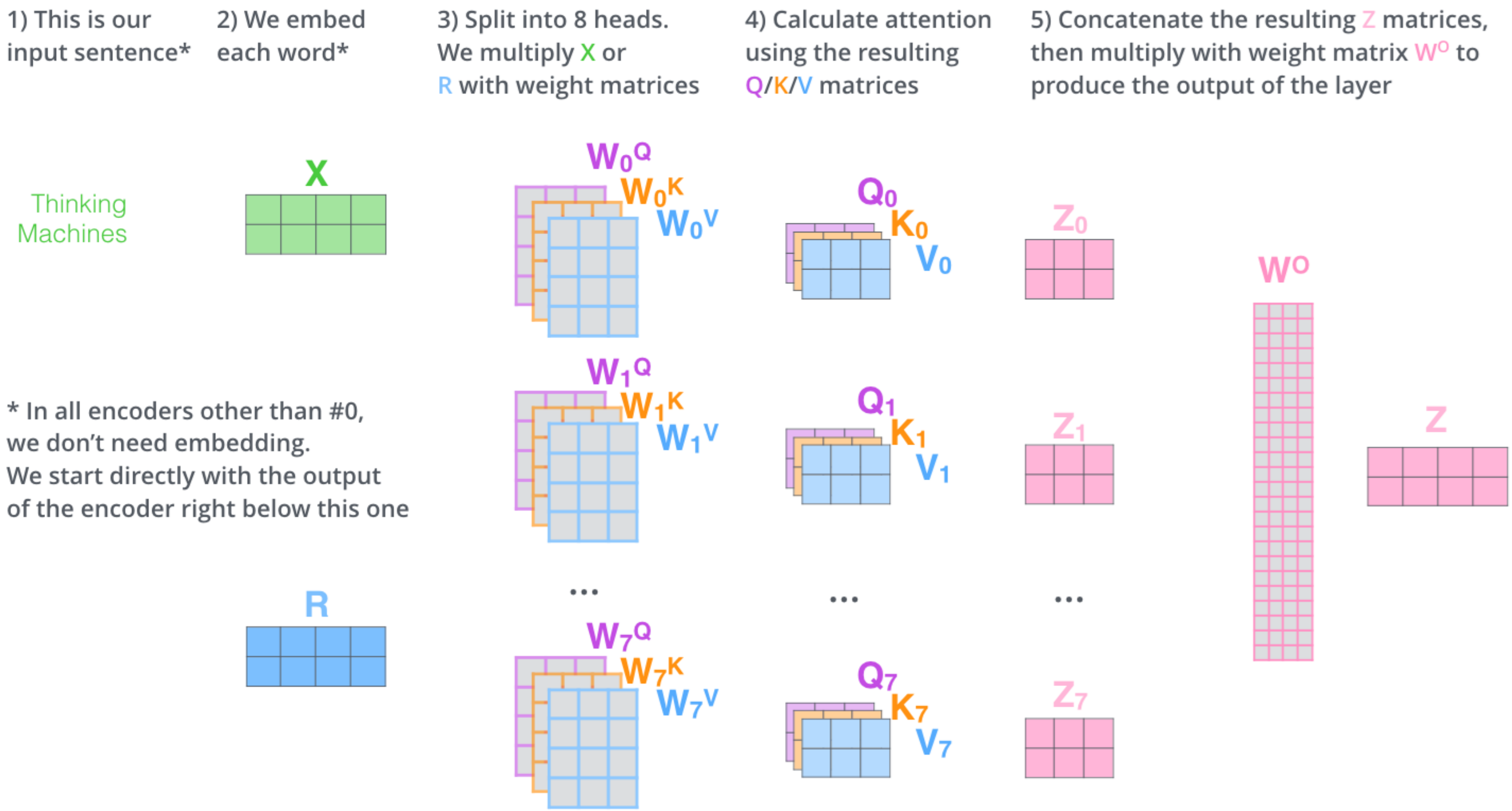
Matrix Calculation of Self-Attention



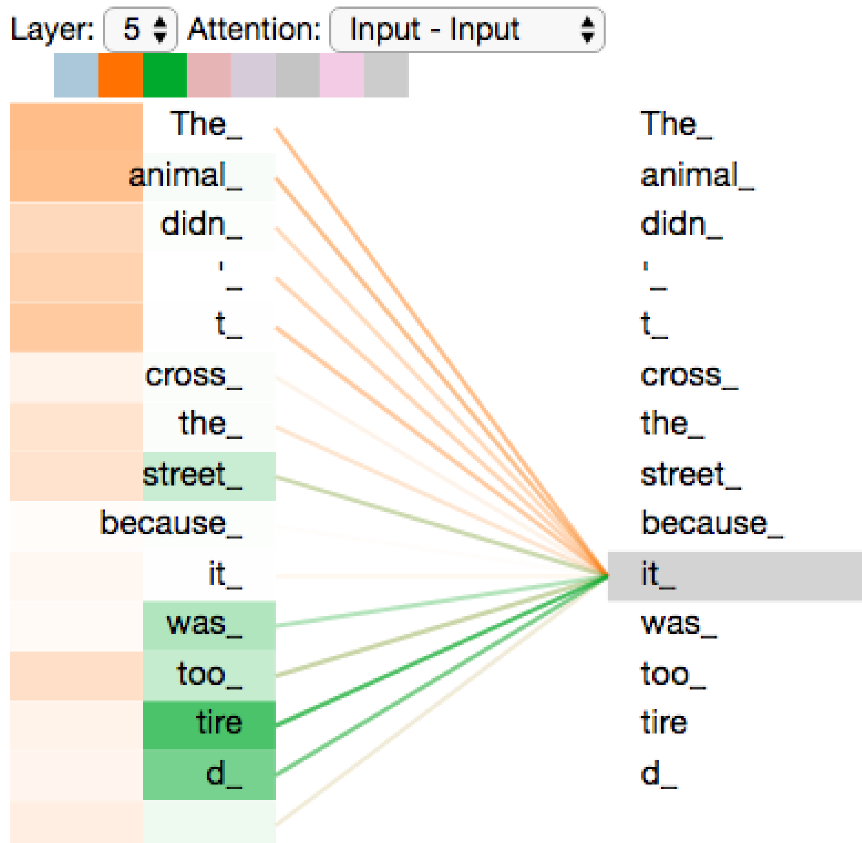
The Beast with Many Heads…



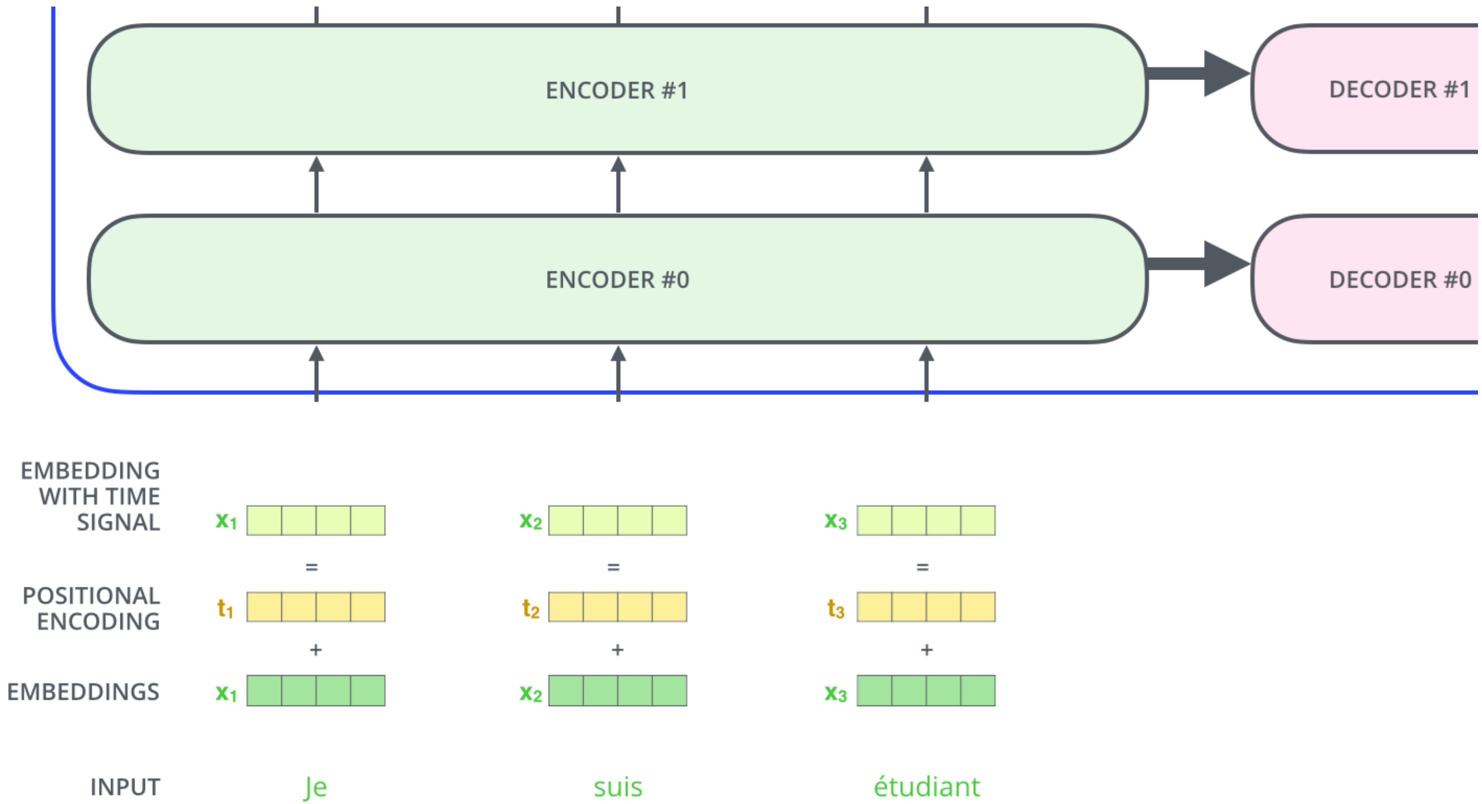
Putting it all together…



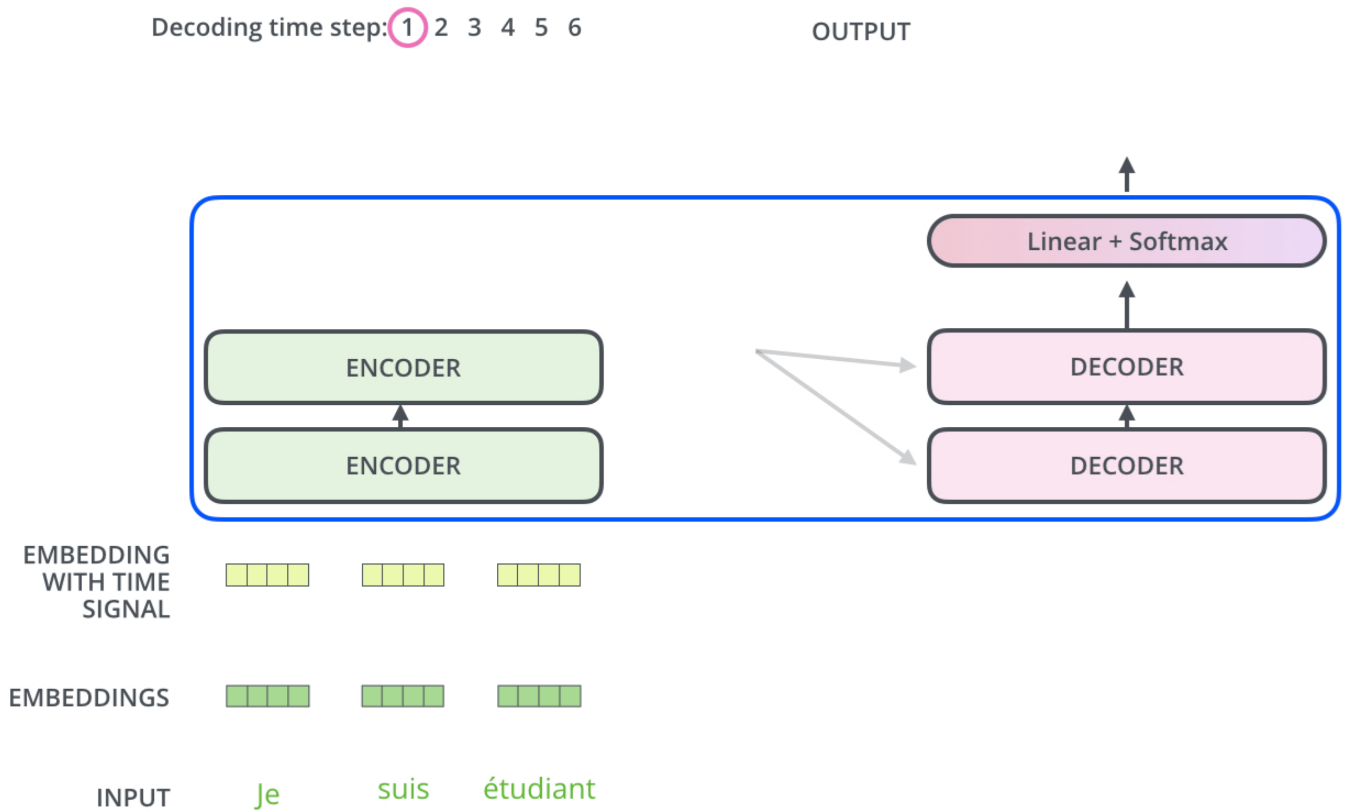
Going back to the Example…



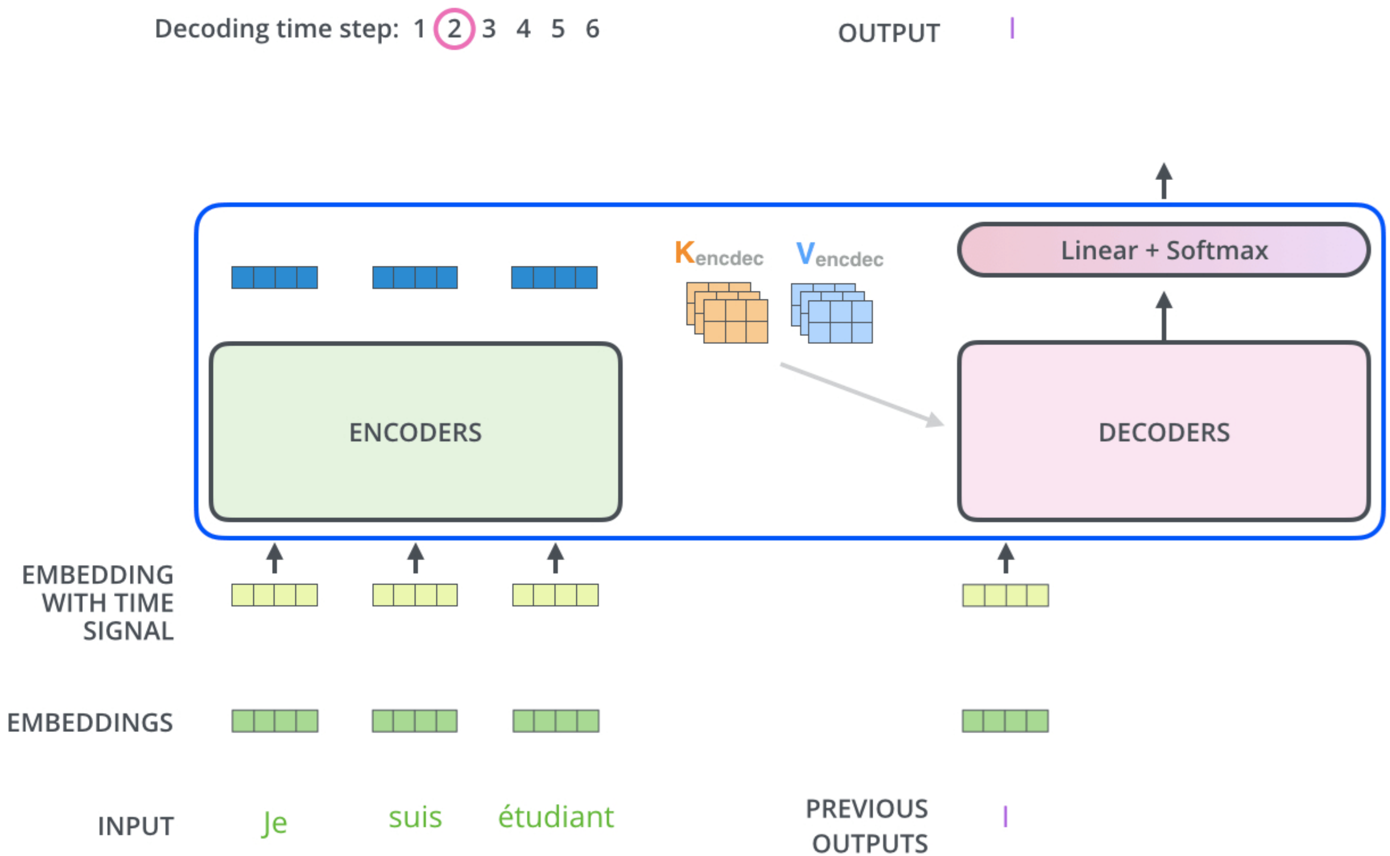
Positional Encoding…



The Decoder Side…



The Decoder Side…



Loss Function…

Output Vocabulary

WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

One-hot encoding of the word “am”

0.0	1.0	0.0	0.0	0.0	0.0

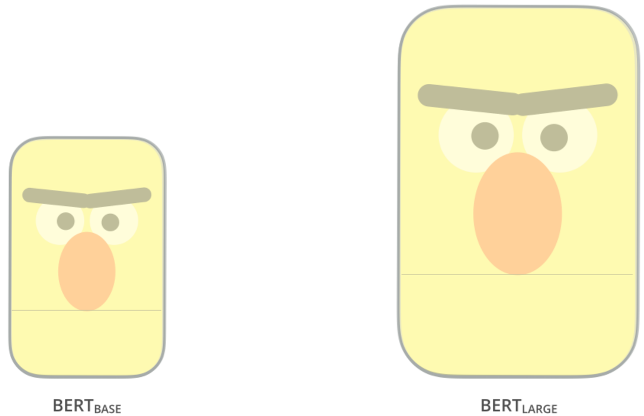
Untrained Model Output

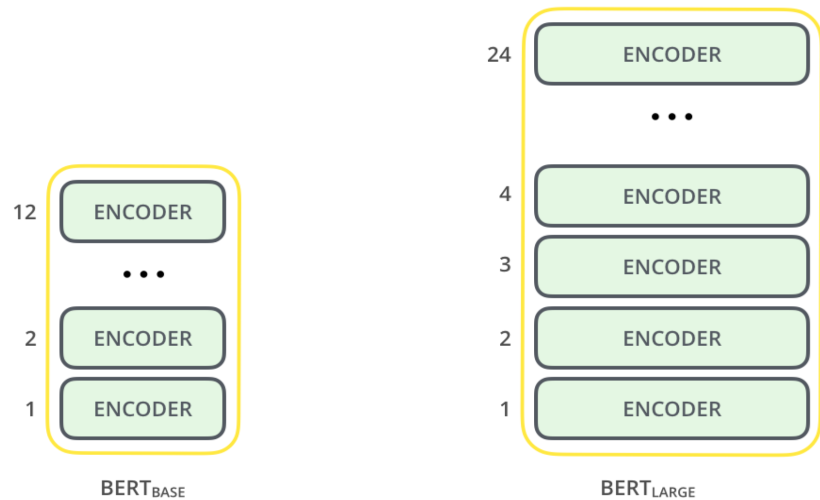
0.2	0.2	0.1	0.2	0.2	0.1

Correct and desired output

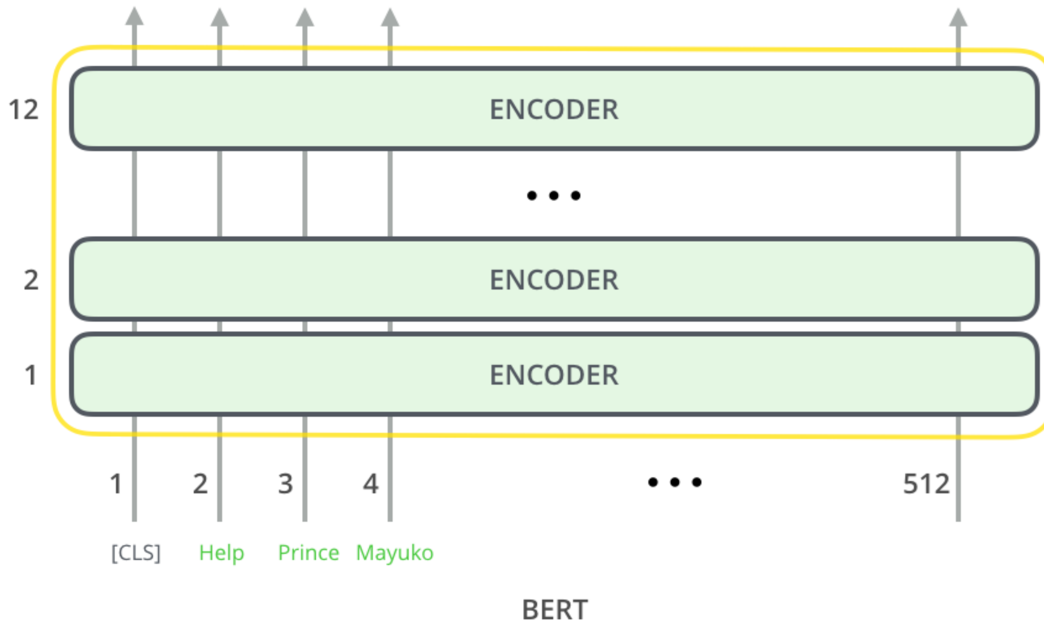
a	am	I	thanks	student	<eos>
0.0	0.0	0.0	1.0	0.0	0.0

Coming to BERT…

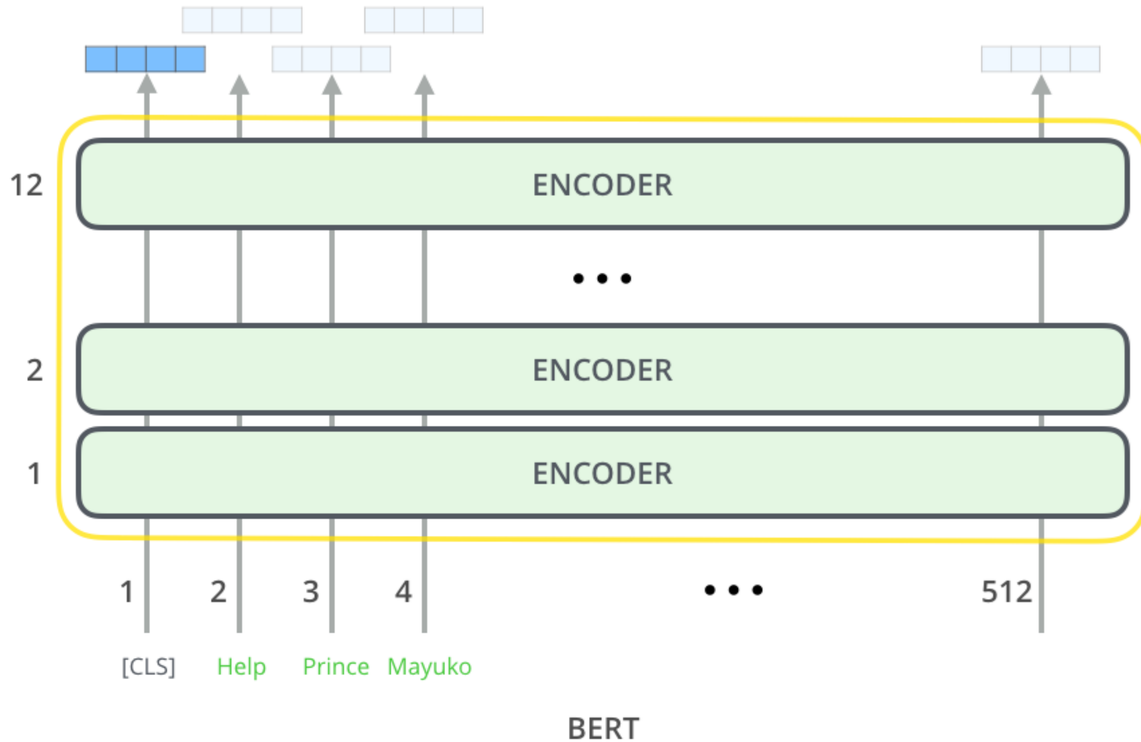




Coming to BERT…



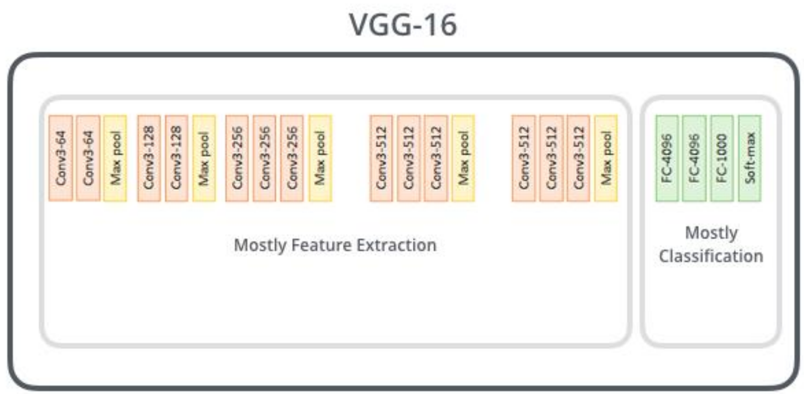
Coming to BERT, Model Outputs…



Coming to BERT, like CNNs

Input

Features



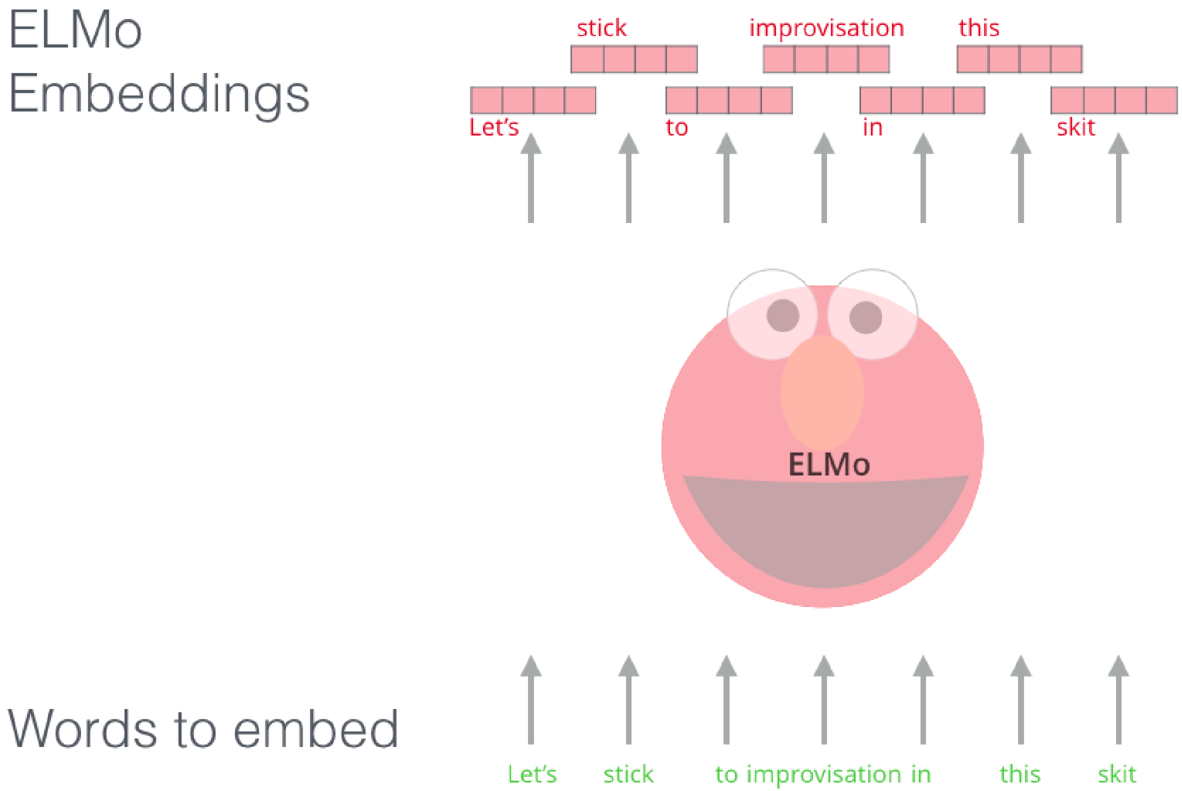
Output

Prediction

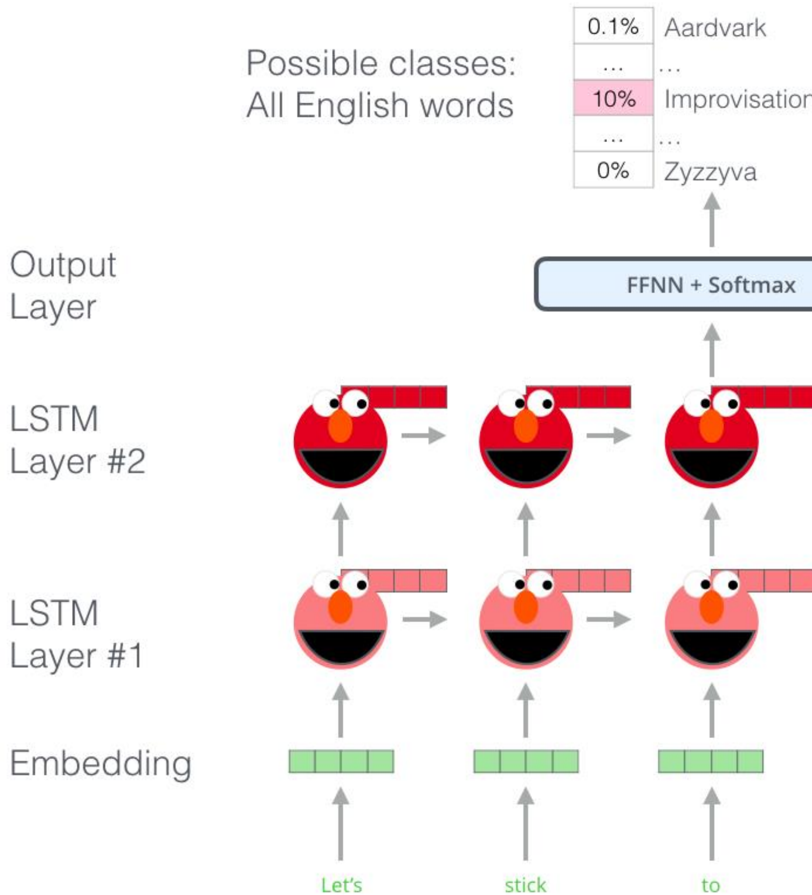
0.2%	Kit fox
0.1%	English setter
95%	Egyptian cat
1%	Great Dane
	...
0%	Hotdog

ELMo: Context Matters

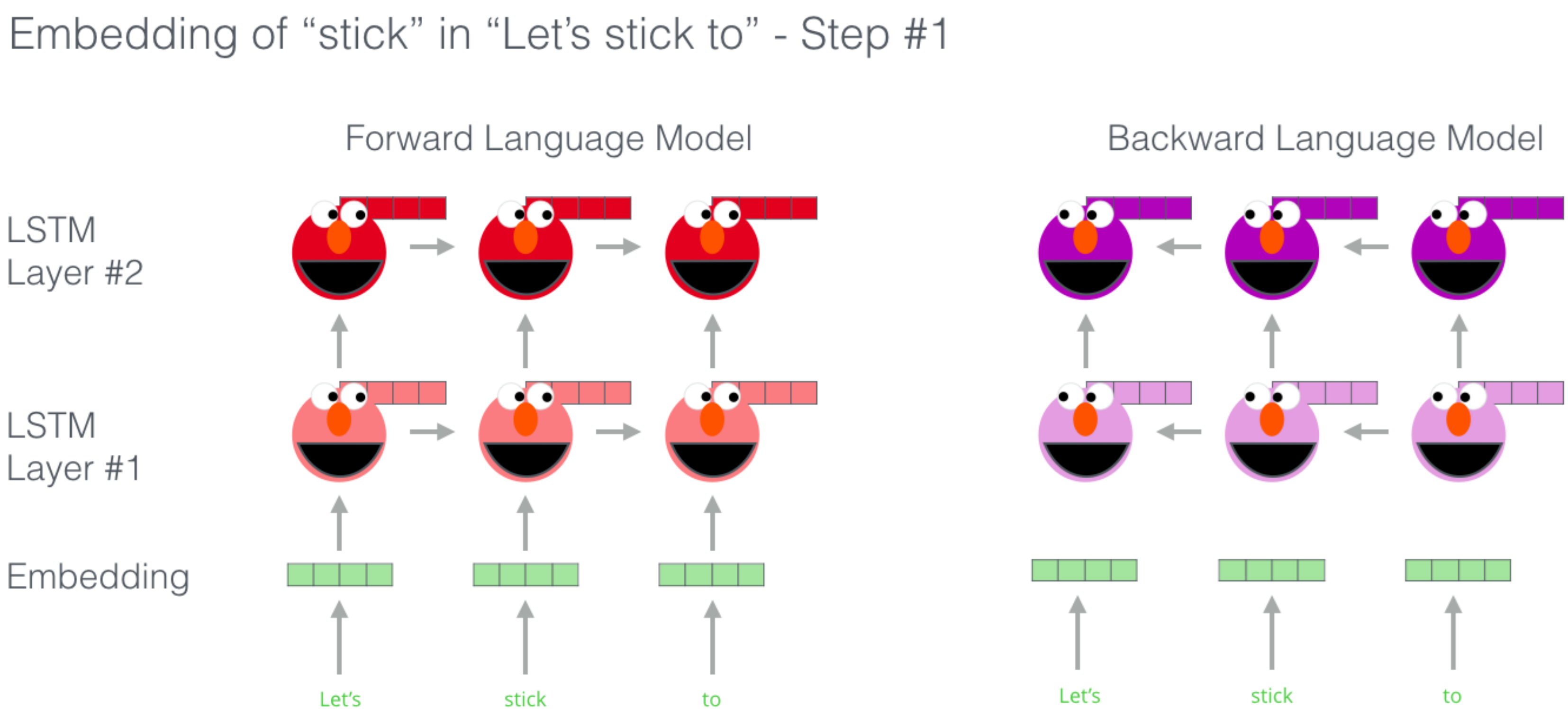
ELMo: Context Matters



ELMo's Secret…



ELMo’s Real Secret, Step 1…



ELMo’s Real Secret, Step 2…

Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers

2- Multiply each vector by a weight based on the task

x s_2

x s_1

x s_0

3- Sum the (now weighted) vectors

ELMo embedding of “stick” for this task in this context