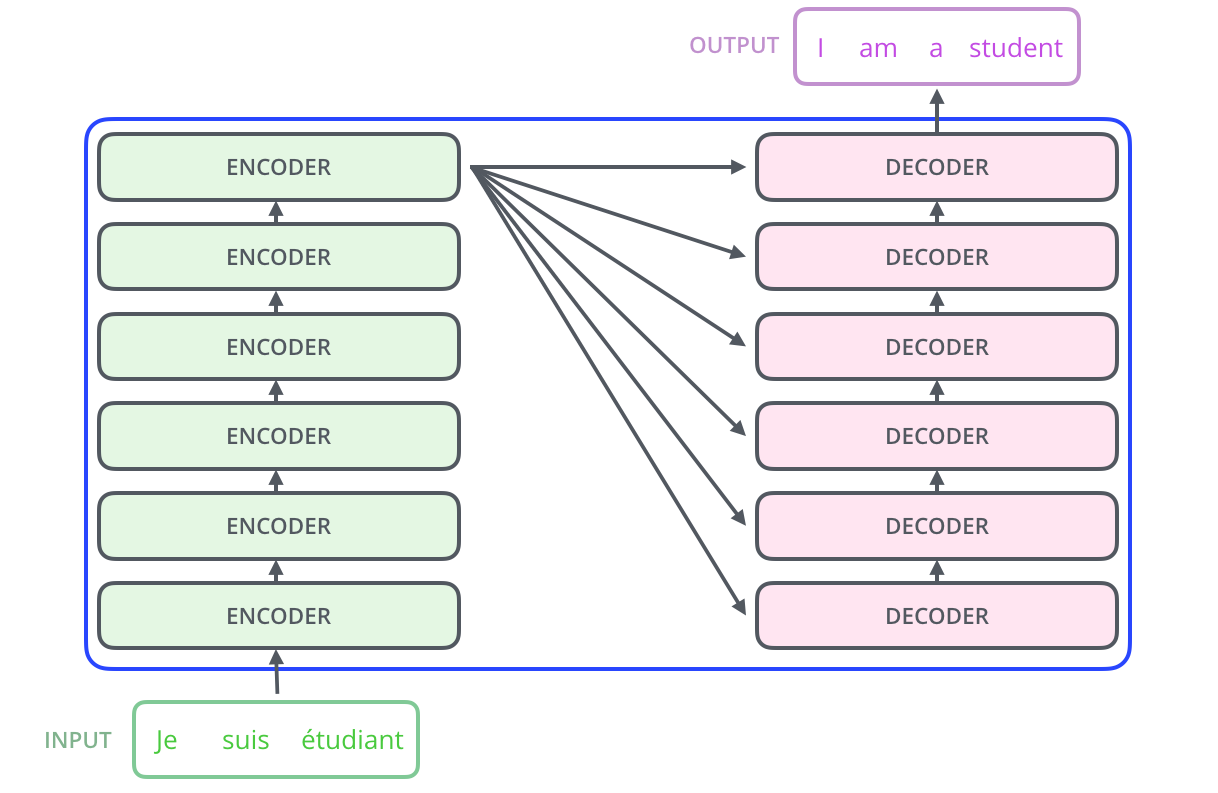
**Transformers**

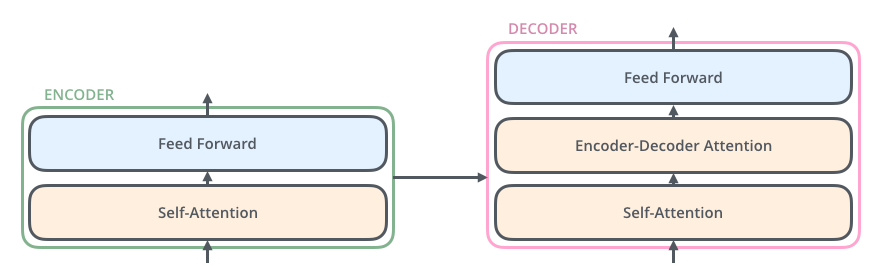
Model that uses attention to boost speed with which these models can be trained.

“Attention is All You Need” paper proposed Transformers.

Transformers are used for Machine Translation where a sentence in one language is converted to a sentence in other language.



The Encoding component is a stack of encoders and Decoding component is stack of decoders.



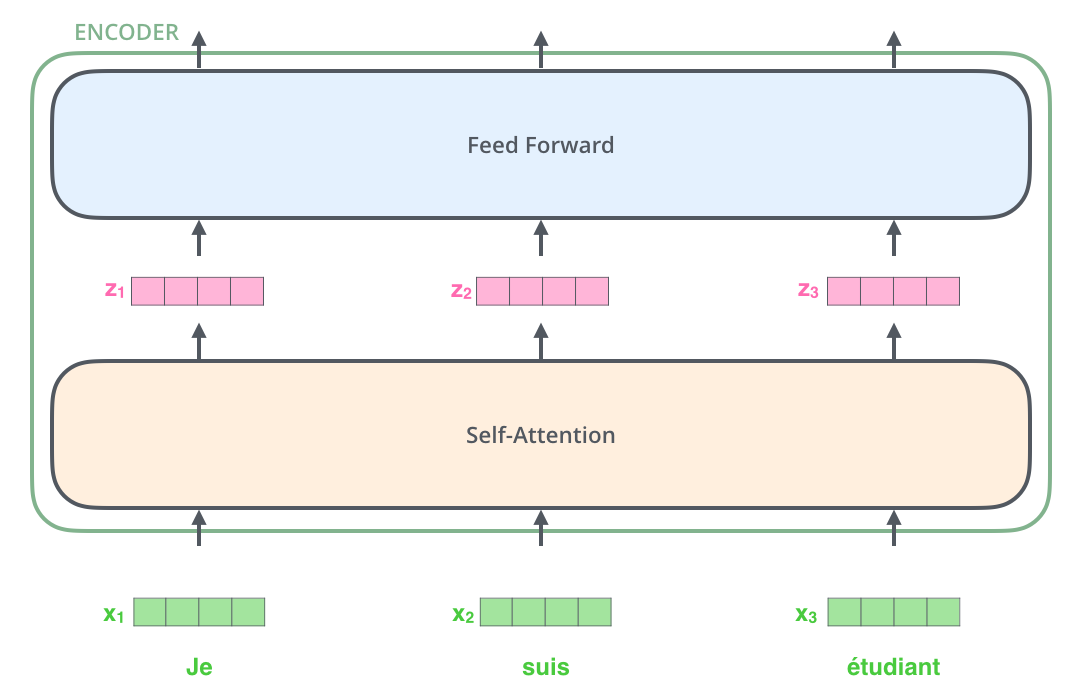
Encoders are all identical in nature but do not share same weights.

Self-Attention layer – a layer that helps the encoder look at other words in the input sentence as it encodes a specific word. 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.

But in decoder a separate “Encoder-Decoder Attention” layer which focus on relevant part of the input sequence.

Transition of Input:

Each input token will be converted to an Embedding vector of size 512. So, the encoder receives list of vectors. This Embedding happens only at bottom most encoder while other encoders take output from previous encoders as input and no further embedding. The word in each position flows through its own path.

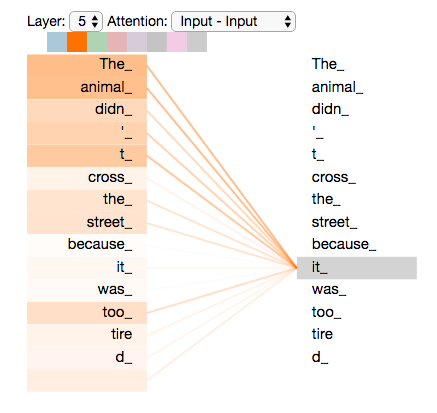


**Self-Attention:**

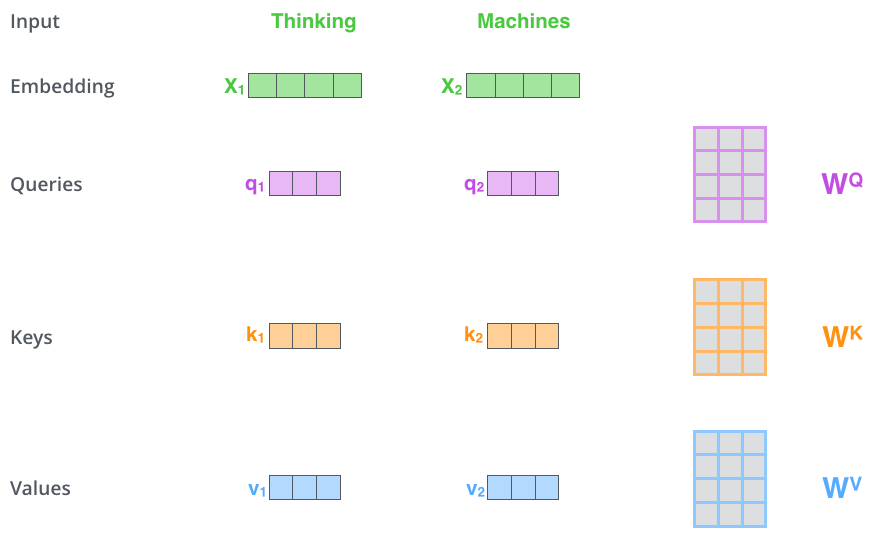
Look at the sentence.

“*The animal didn’t cross the street because It was too tired*”

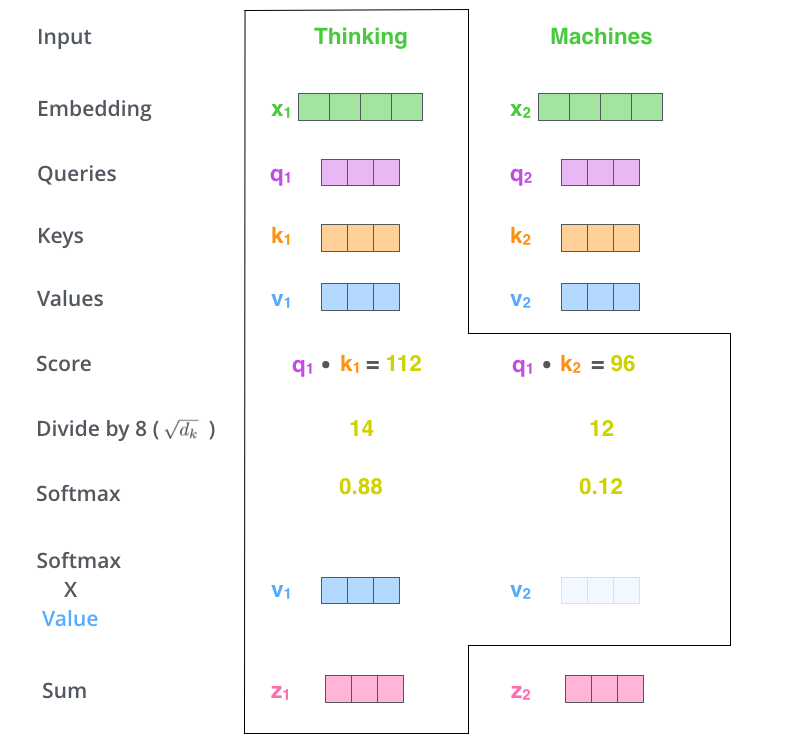
Here ‘it’ refers to animal. But how can a machine understand that. So, Self-Attention uses to understand the relevance of one word w.r.t other words.



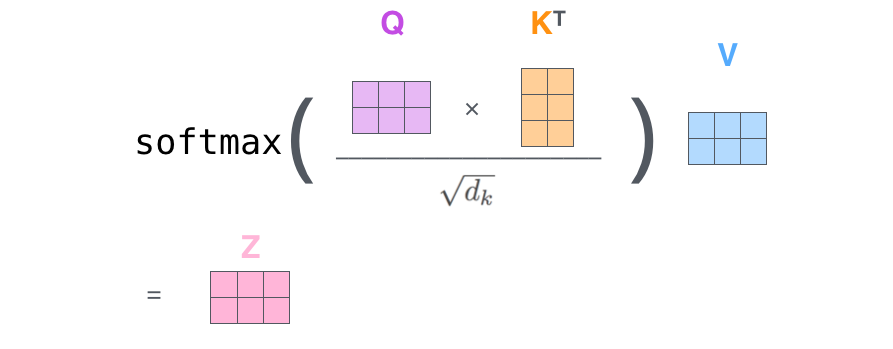
Firstly, we generate Query Vector, Key Vector, Value Vector by multiplying Embeddings of words with Query Matrix, Key Matrix, Value Matrix. Here embeddings size is 512 and each query, key, value vectors is 64, this is because we use “**Multiheaded attention**”.



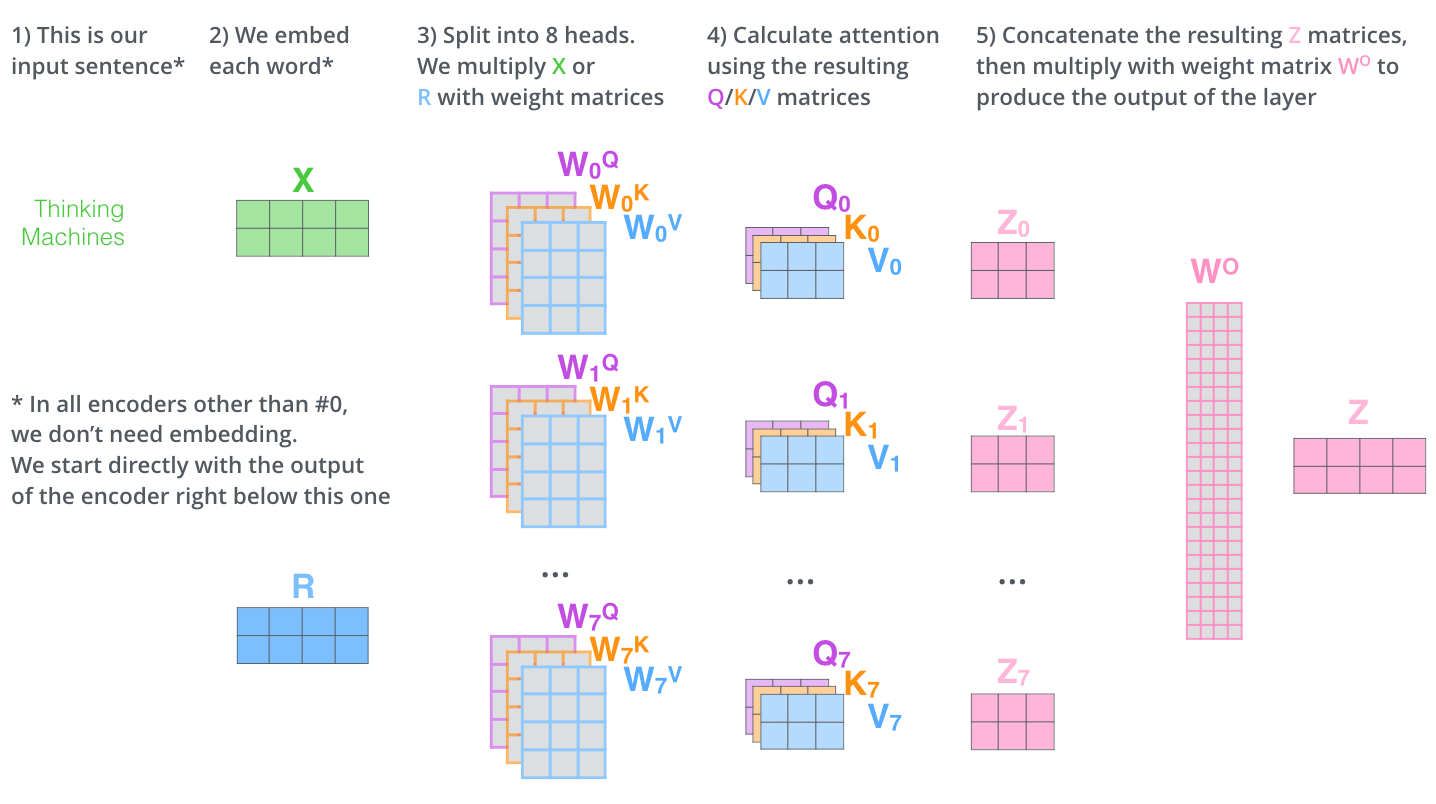
Second step is to find scores. For calculating self-attention for “Thinking” we need to do dot product of query vector of “Thinking” with all key vectors of other words to get scores and apply softmax to get sum of all scores to 1. Then we multiply value vectors w.r.t their scores and sum them to get resulting vector (Z). This vector is then send to Feed Forward network.



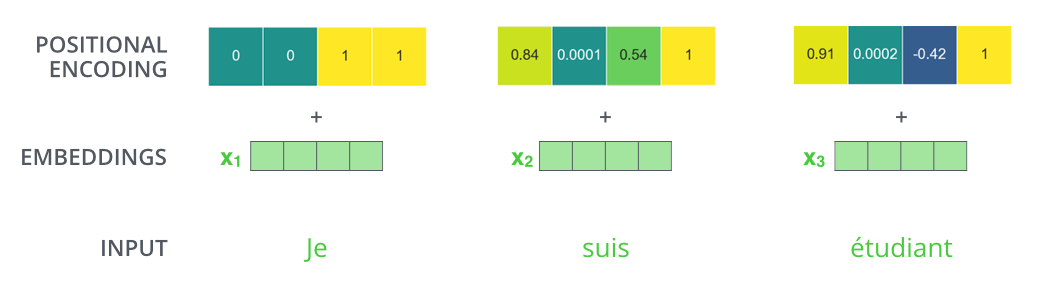
For general simple formula we can think it like.



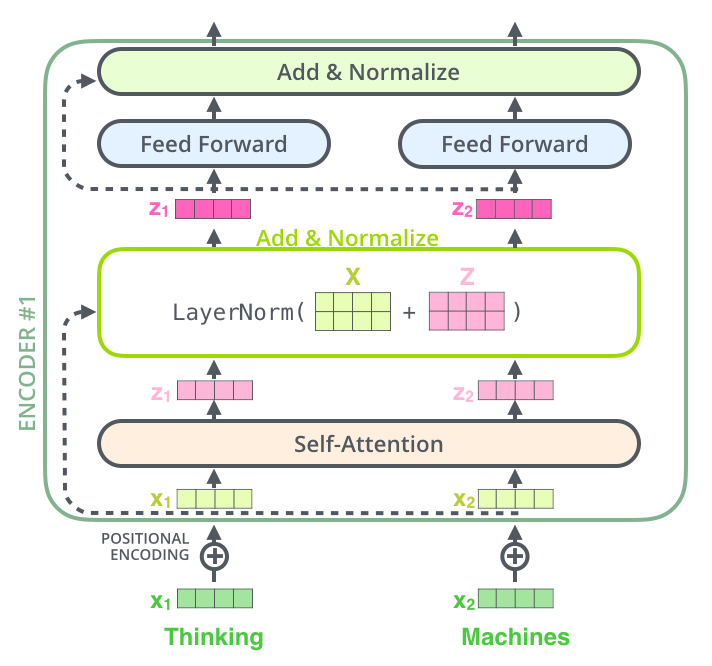
But Transformers use **“Multi-headed Attention”**. Transformers use 8 attention heads.



Next comes Positional encoding, in-order to order of the words we add positional vector to Embedding to get input vector.



Residuals:



Before sending output to further sub-layers we add input with output and normalize.

Decoder side:

At Decoder side almost everything is same, but we get output vectors as K, V encoded vectors from topmost encoder layer. They can be used in “Encoder-Decoder Attention Layer” in every decoder. The “Encoder-Decoder Attention” layer works just like multiheaded self-attention, except it creates its Queries matrix from the layer below it, and takes the Keys and Values matrix from the output of the encoder stack.



The Self-Attention in decoder is different from one in Encoder as it masks the Future Tokens so called Masked Self-Attention.

This gives a vector of 512 embedding-size. A Linear neural layer is applied to this to get a vector to project this to Vocab-size. Then a softmax is applied to this to get probabilities. Top probabilities can be considered. We can use top-k parameter to consider top-k probable words rather than fixing to top most probable word.

