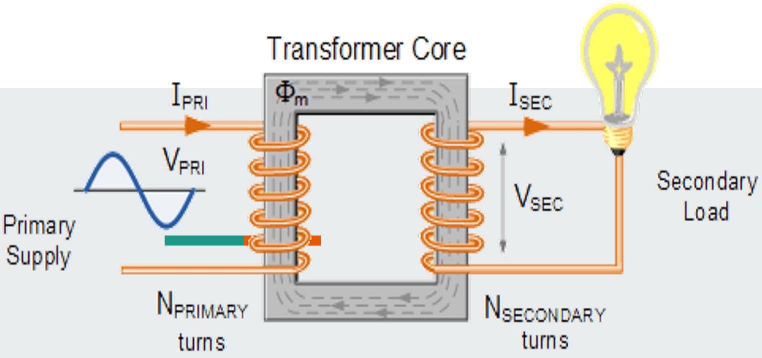
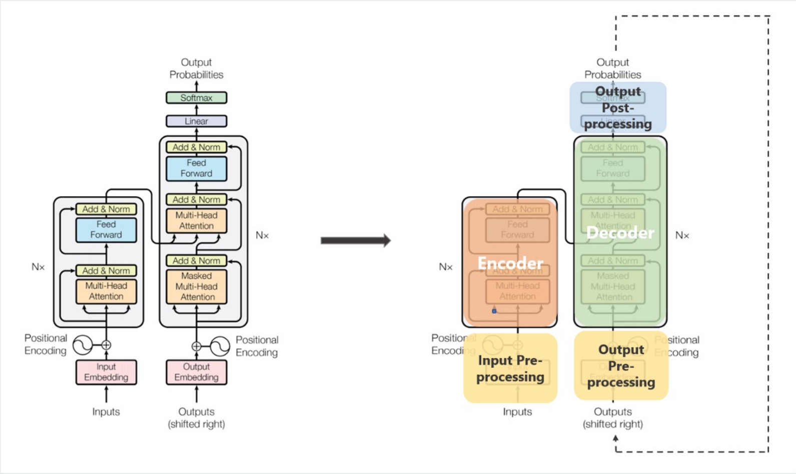


Attention is all you need

Pradeep Kumar Ragu Chanthar
Srinivasa Sai Deepak Varukol



Background

- Feed forward neural network
- Recurrent neural network
- Encoding and decoding modeling
- Attention mechanism
- softmax

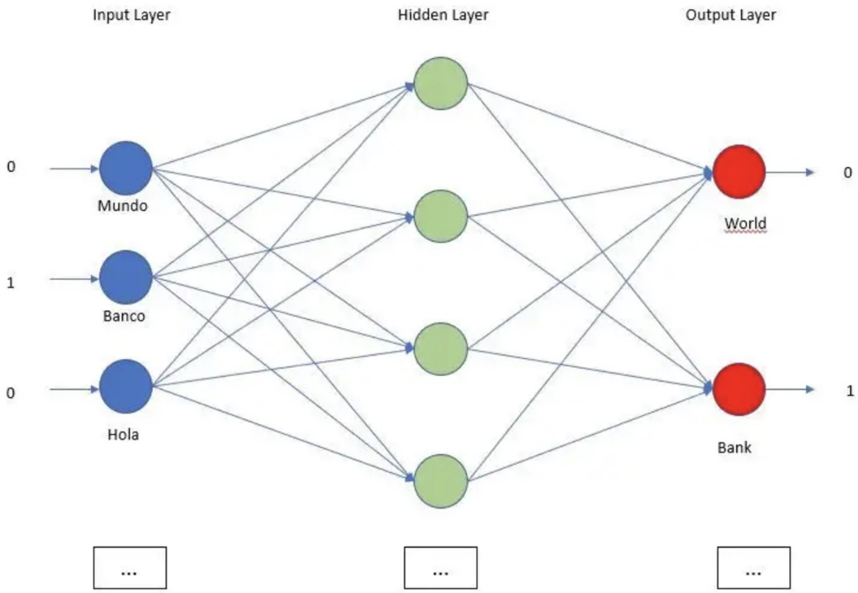
Feed Forward Neural Network (FFNN)

- First and simple neural network

- The information moves in only one direction which is forward from the input nodes, through the hidden nodes (if any) and to the output nodes so there are no cycles or loops in the network

- Using FFNN we can train a model that receives a Spanish word and give you the equivalent in English. For every Spanish word the model receives, it outputs an English one. But can we train a model to do translation of spanish sentence to english or vice versa?

FFNN example



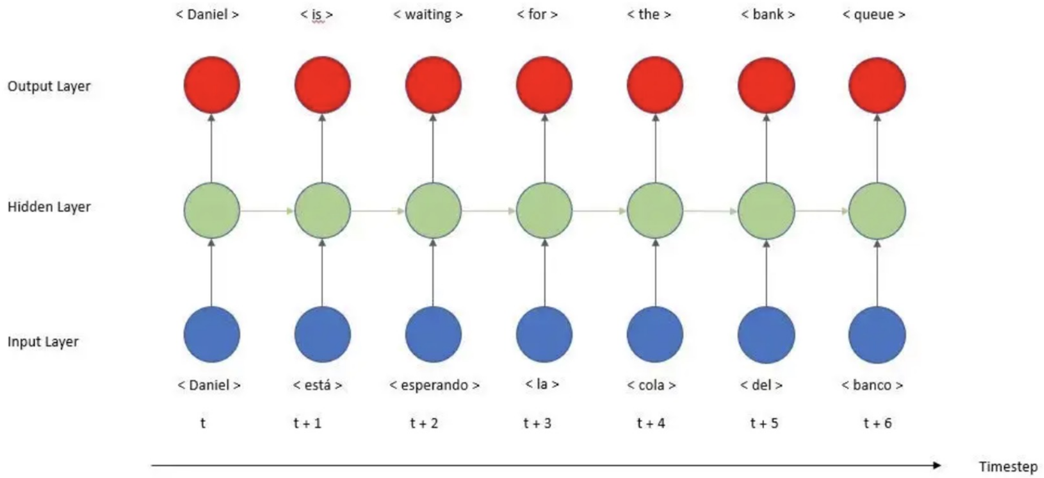
Feed forward neural network structure to translate incoming spanish words

Recurrent Neural Network (RNN)

- Like any other neural network model recurrent neural networks utilize training data to learn but they are distinguished by their “memory” as they take information from prior inputs to influence the current input and output

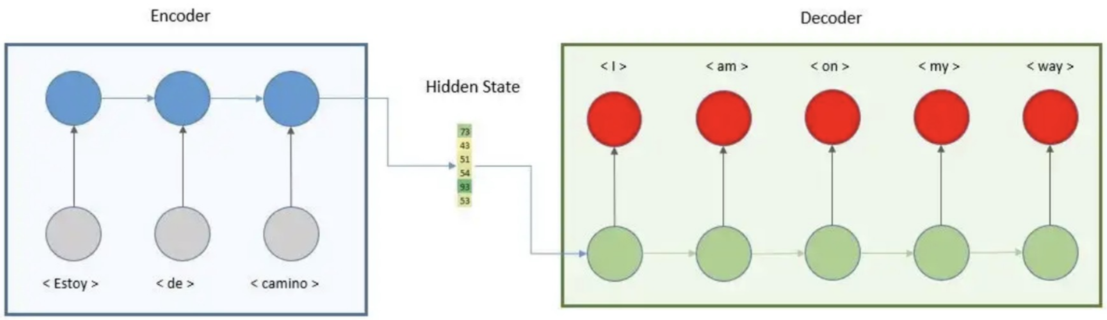
- While traditional deep neural networks assume that inputs and outputs are independent of each other, The output of recurrent neural networks depend on the prior elements within the sequence. While future events would also be helpful in determining the output of a given sequence

RNN example



Recurrent neural network structure to translate incoming spanish words

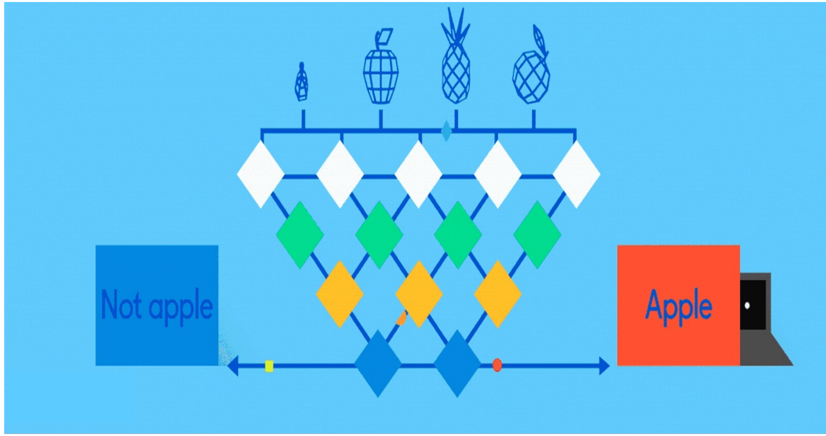
Encoder and decoder model



Activation function

- Activation functions are functions used in a neural network to compute the weighted sum of inputs and biases, which is in turn used to decide whether a neuron can be activated or not

- It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).



Motivation

- RNN was used to analyzing the language whether it for translation or text summarization or text generation

- RNN takes each word from the input one at a time and process the output sequentially

- One of the major drawback of RNN was not handling not large sequence of words. Another one is slow to train because it couldn’t parallelize the process

Transformer

- Transformer is a neural network architecture that aims to solve tasks sequence-to-sequence while easily handling long-distance dependencies

- The input sequence can be passed parallelly so that GPU can be used effectively and the speed of training can also be increased.

Difference between RNN and Transformer

RNN

Transformer is better

RNN encoder

vector

RNN decoder

Transformer

Transformer is best

Transformer encoder

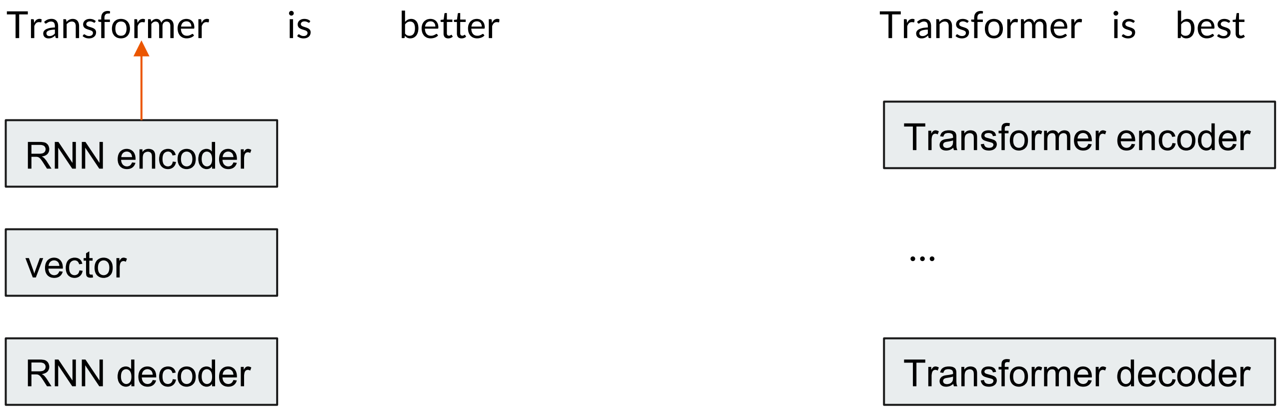
...

Transformer decoder

Difference between RNN and Transformer

RNN

Transformer



Difference between RNN and Transformer

RNN

Transformer is better

RNN encoder

vector

RNN decoder

Transformer

Transformer is best

Transformer encoder

...

Transformer decoder

Difference between RNN and Transformer

RNN

Transformer is better

RNN encoder

vector

RNN decoder

Transformer

Transformer is best

Transformer encoder

...

Transformer decoder

Difference between RNN and Transformer

RNN

Transformer is better

RNN encoder

vector

RNN decoder

Transformer

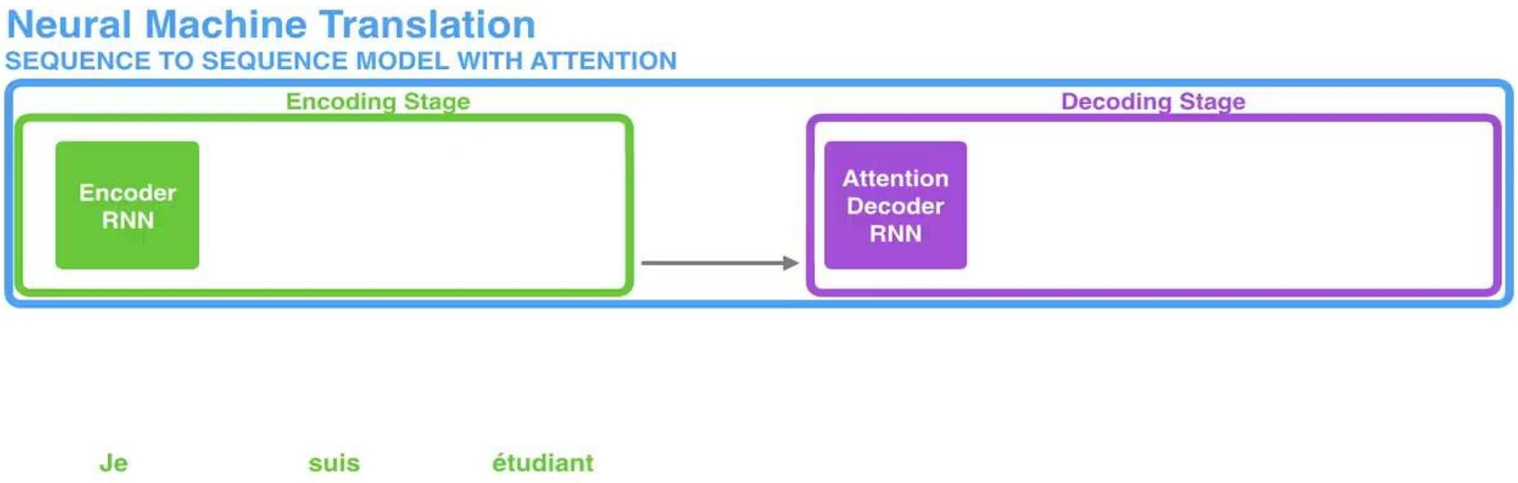
Transformer is best

Transformer encoder

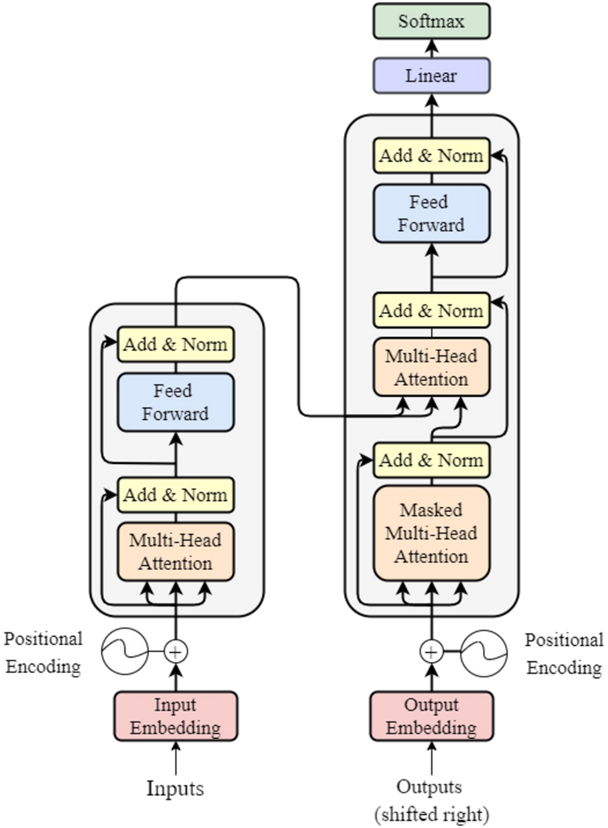
...

Transformer decoder

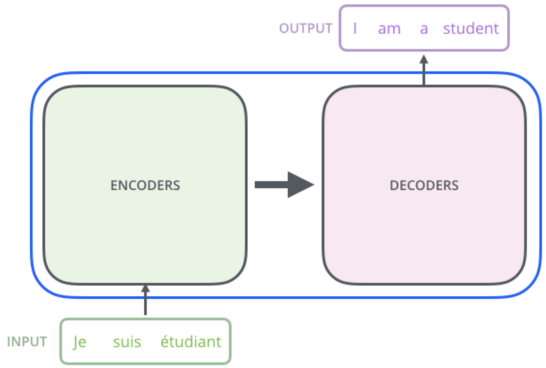
Attention

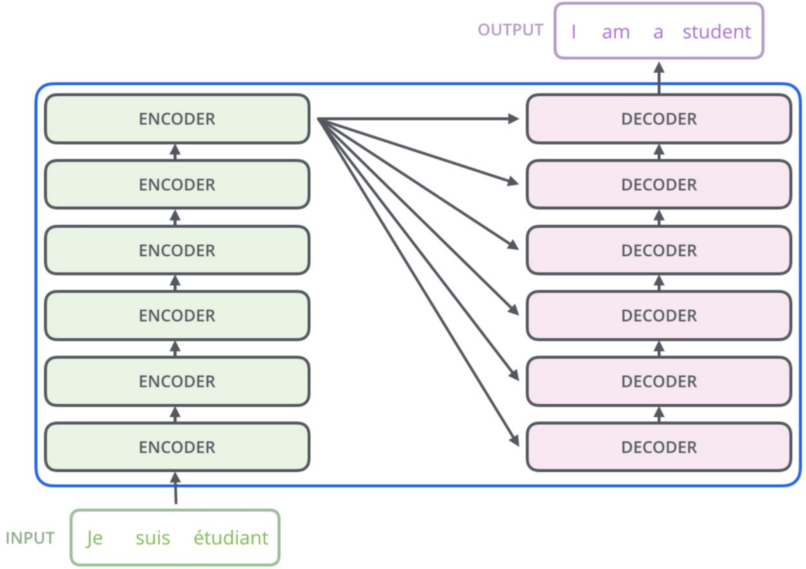

Neural Machine Translation
SEQUENCE TO SEQUENCE MODEL WITH ATTENTION
Encoding Stage
Decoding Stage
Encoder RNN
Attention Decoder RNN
Je
suis
étudiant

The Architecture of Transformer

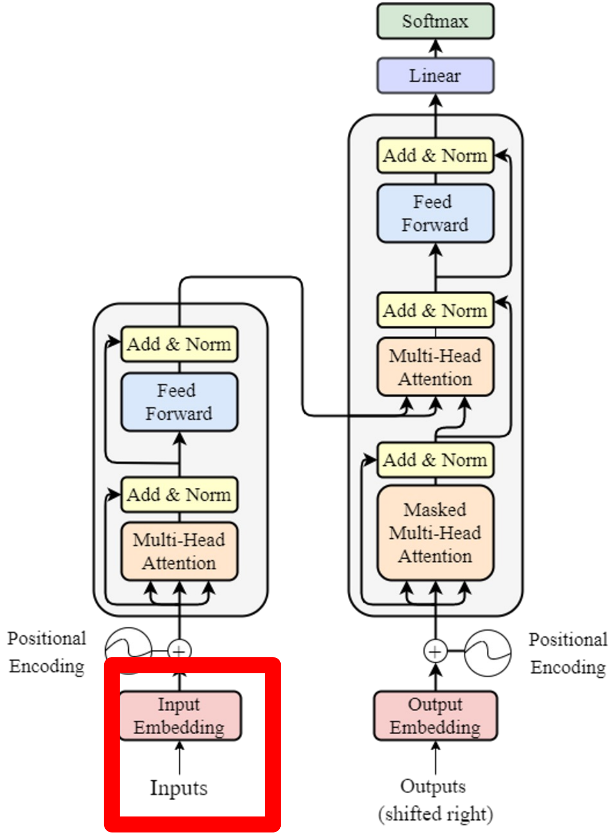


The Architecture of Transformer

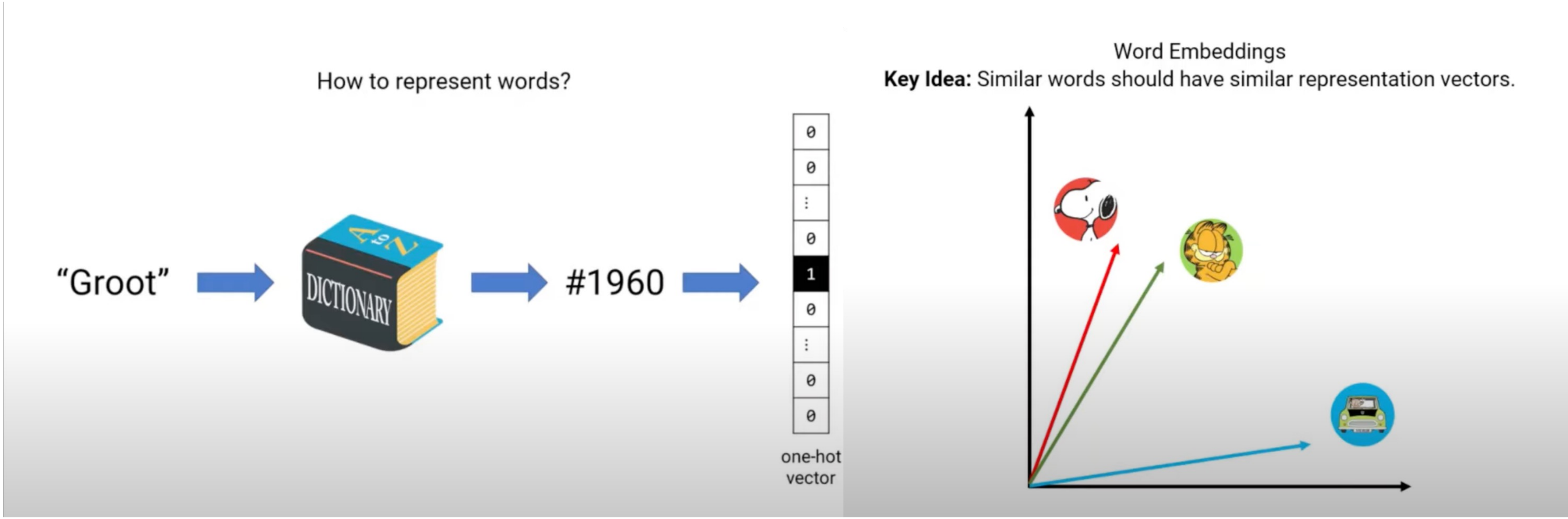




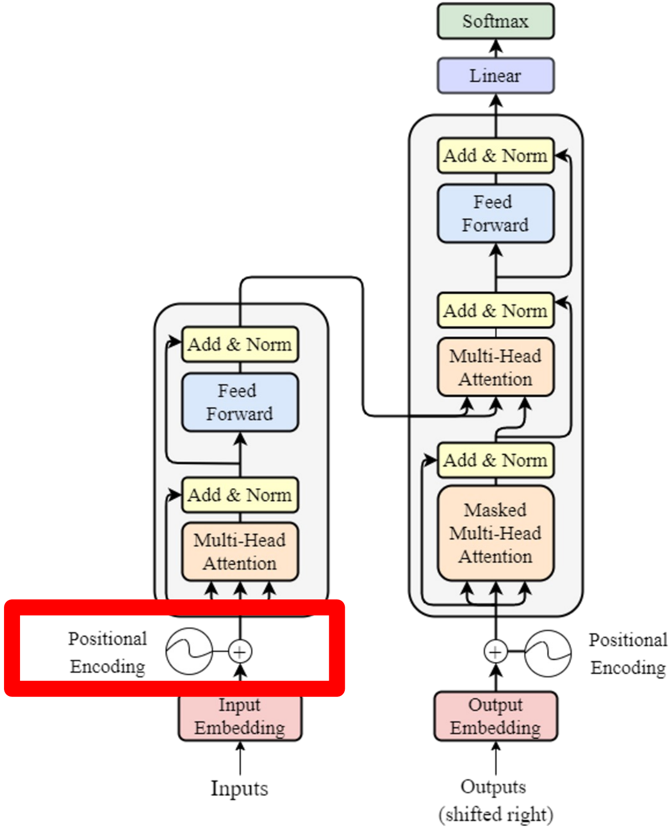
The Architecture of Transformer



Input Embedding

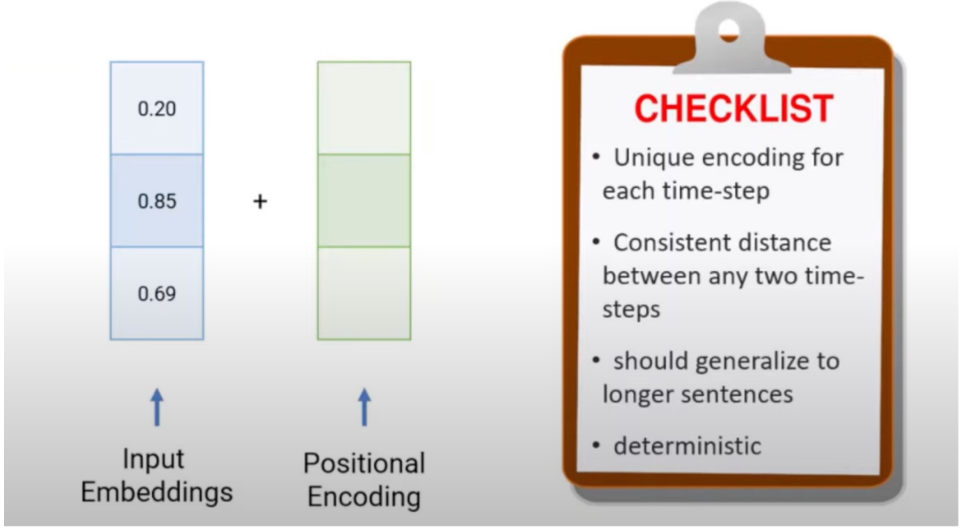


The Architecture of Transformer

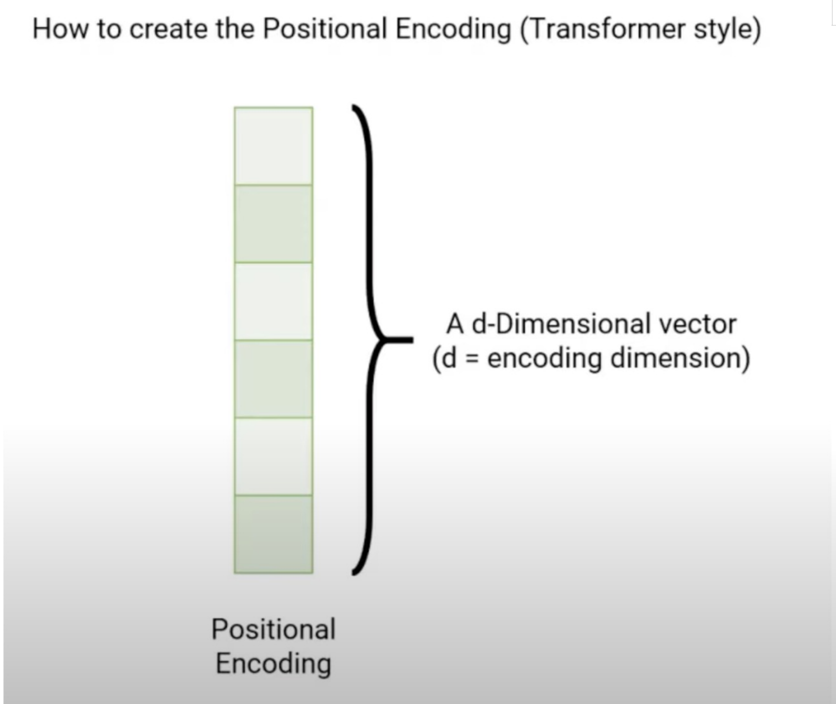


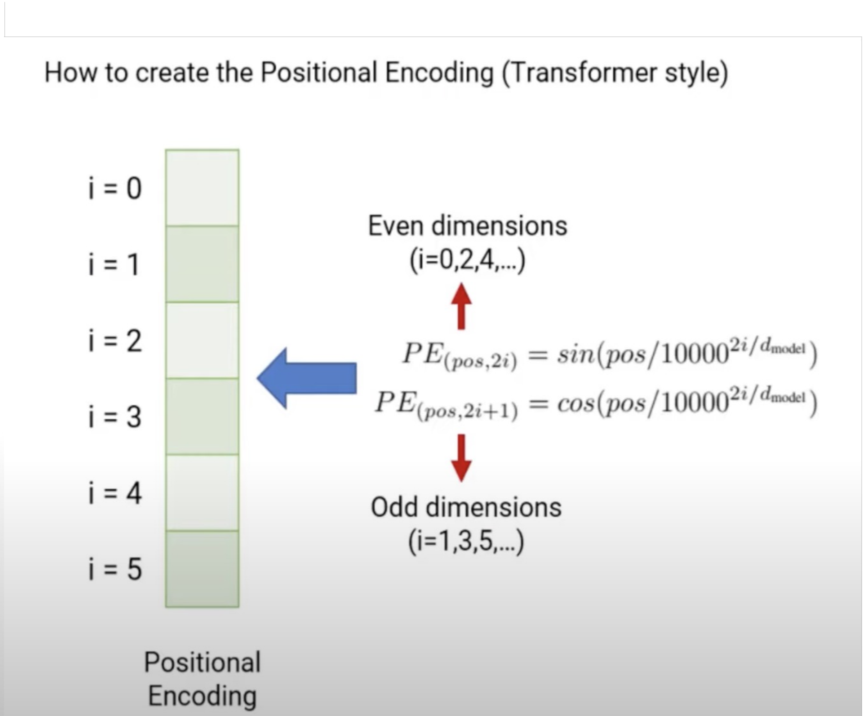
Positioning encoding

How to add positional information?
Key Idea: add a new vector containing positional info to current vectors.

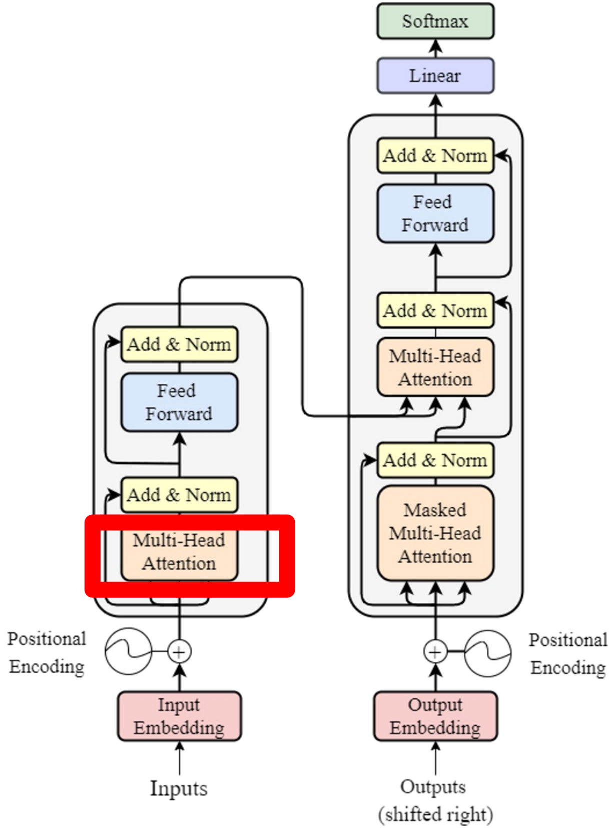


Positioning encoding





The Architecture of Transformer



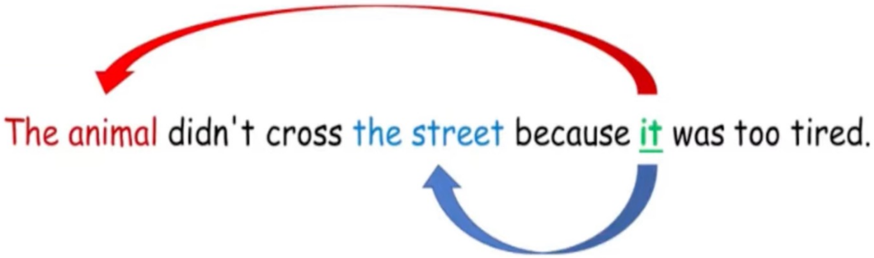
Attention mechanism

Mapping Queries and key-value pairs to output

Output - Weighted sum of the values

Attention functions

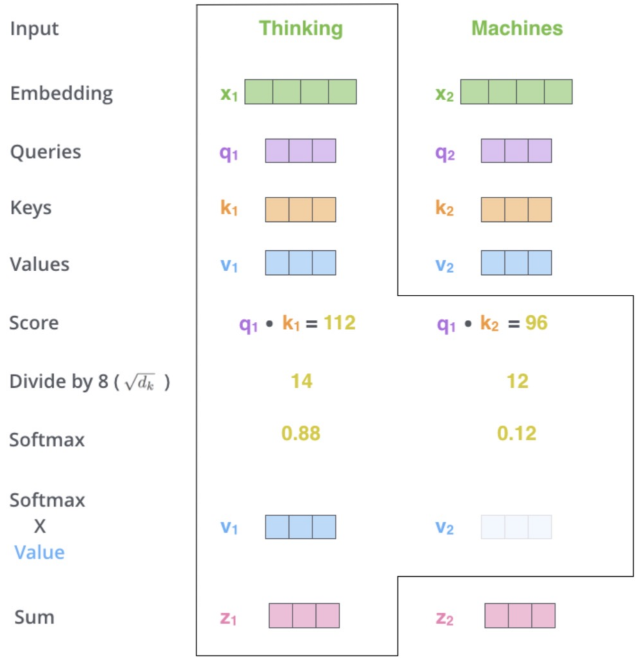
- Additive attention
- Dot product attention



Self attention mechanism

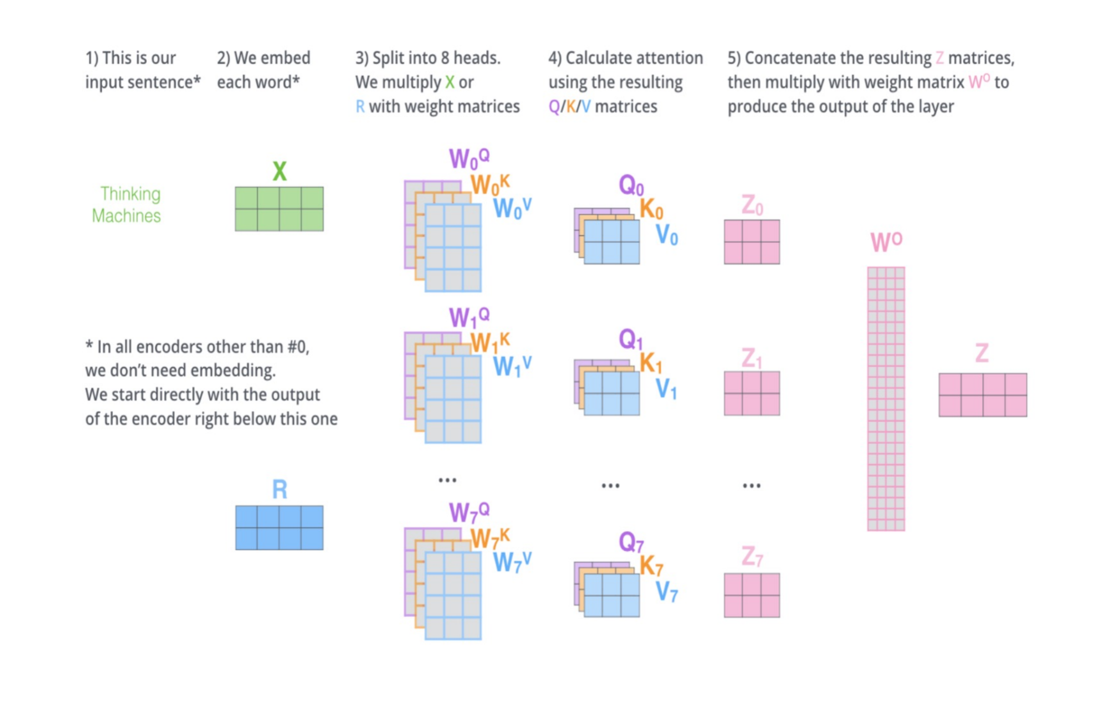
- We calculate the scores by multiplying query and key vectors
- We divide it with √dk (dimensions of key vectors)
- Then these results sent to softmax function, which indicates how much each word will be expressed at this position
- Multiply value vector with corresponding softmax score
- Sum of weighted value vectors

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

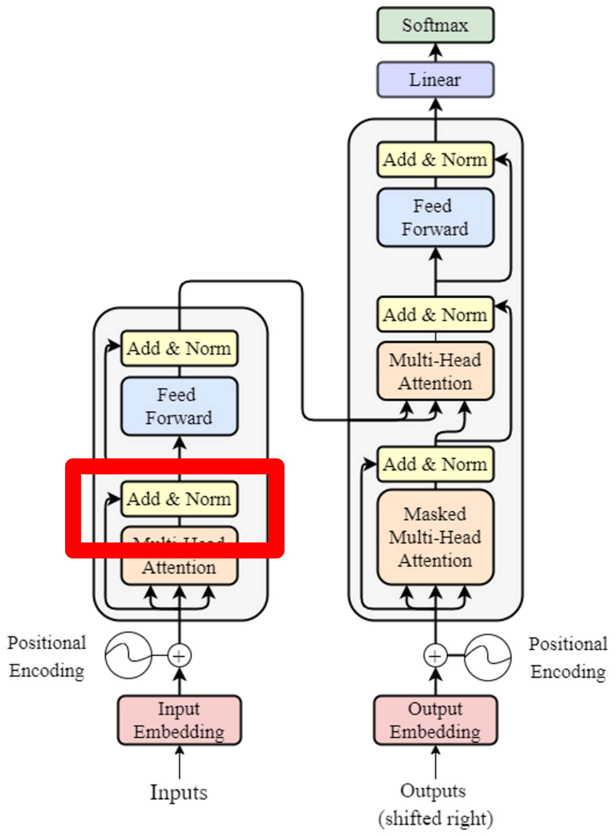


Multi head attention

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},...,\mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

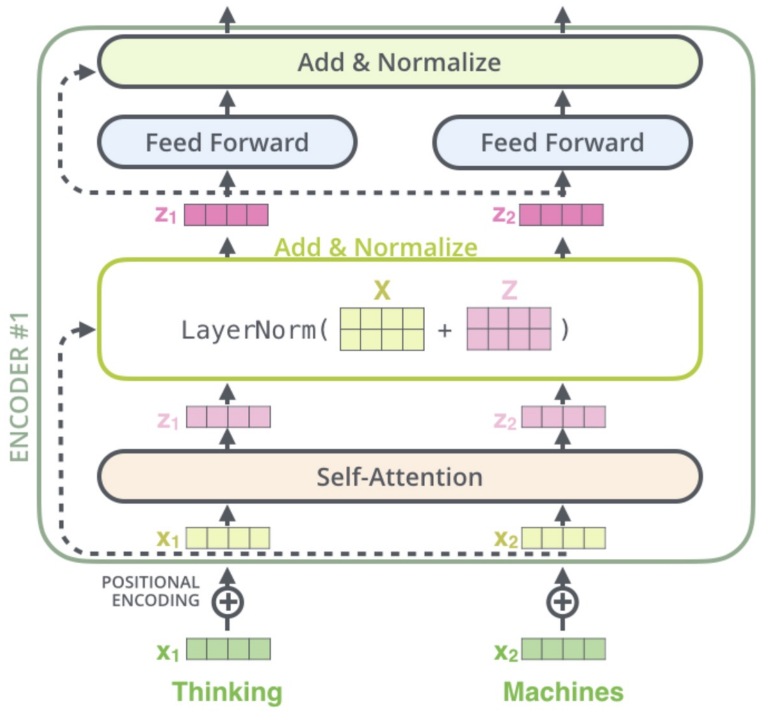


The Architecture of Transformer



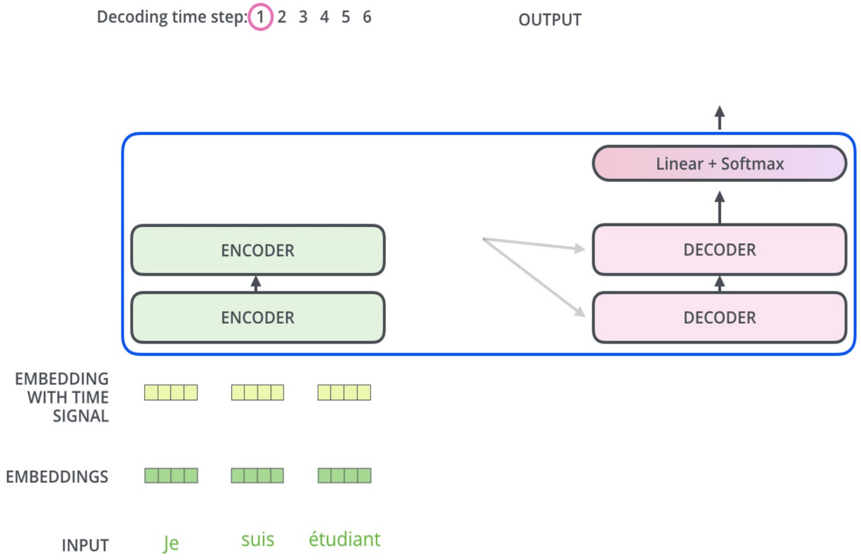
The Residuals

- Each sub-layer (self-attention, ffnn) in each encoder has a residual connection around it, and is followed by a *layer-normalization* step
- This goes for the sub-layers of the decoder as well



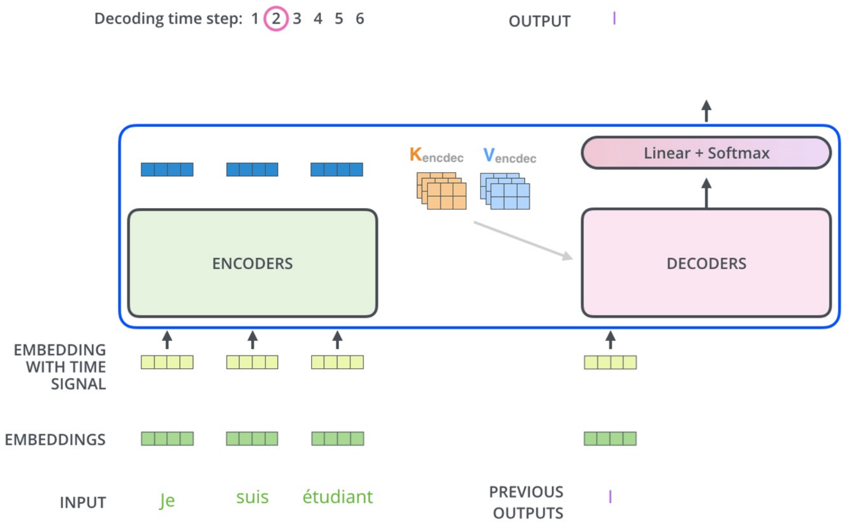
Decoder

- The encoder start by processing the input sequence. The output of the top encoder is then transformed into a set of attention vectors K and V.
- These are to be used by each decoder in its “encoder-decoder attention” layer which helps the decoder focus on appropriate places in the input sequence



Decoder

- The following steps repeat the process until a special symbol is reached indicating the transformer decoder has completed its output.
- The output of each step is fed to the bottom decoder in the next time step, and the decoders bubble up their decoding results just like the encoders did.
- And just like we did with the encoder inputs, we embed and add positional encoding to those decoder inputs to indicate the position of each word.



Decoder

Testing Mode

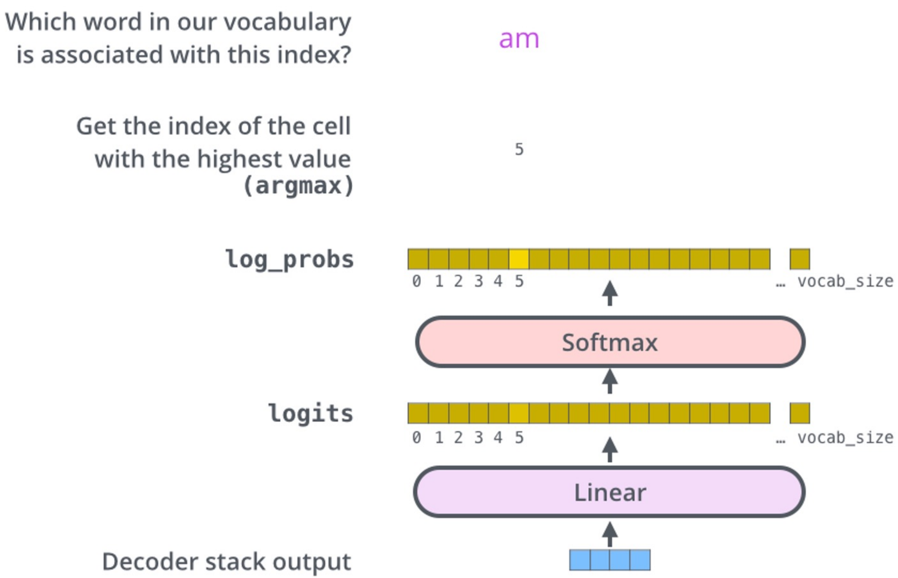
Training Mode

Decoder

The decoder stack outputs a vector of floats.

- The Linear layer is a simple fully connected neural network that projects the vector produced by the stack of decoders, into a much, much larger vector called a logits vector.

- The softmax layer then turns those scores into probabilities (all positive, all add up to 1.0). The cell with the highest probability is chosen, and the word associated with it is produced as the output for this time step.



Results

Model Differences

Model	N (Number of encoder/decoder blocks)	Dimensions of Model	Train steps	Parameters (x10^6)
Base	6	512	100K	65
Big	6	1024	300K	213

Results

English Constituency Parsing Results

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

Results

Translation Task

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Conclusion

- Understanding the Attention Mechanism in recurrent Encoder decoder network
- Introducing transformer: a sequence transduction model just needs attention to work
- New state of the art Language translator

Pros & Cons

Pros:

- State of the art technology
- Overcame the RNN shortcomings

Cons:

- Attention can only deal with fixed-length text strings. The text has to be split into a certain number of segments or chunks before being fed into the system as input
- This chunking of text causes context fragmentation. For example, if a sentence is split from the middle, then a significant amount of context is lost. In other words, the text is split without respecting the sentence or any other semantic boundary

Discussion Questions

- Why transformers have a fixed length context?

- Since long range dependency is not an issue, why segment have be short?