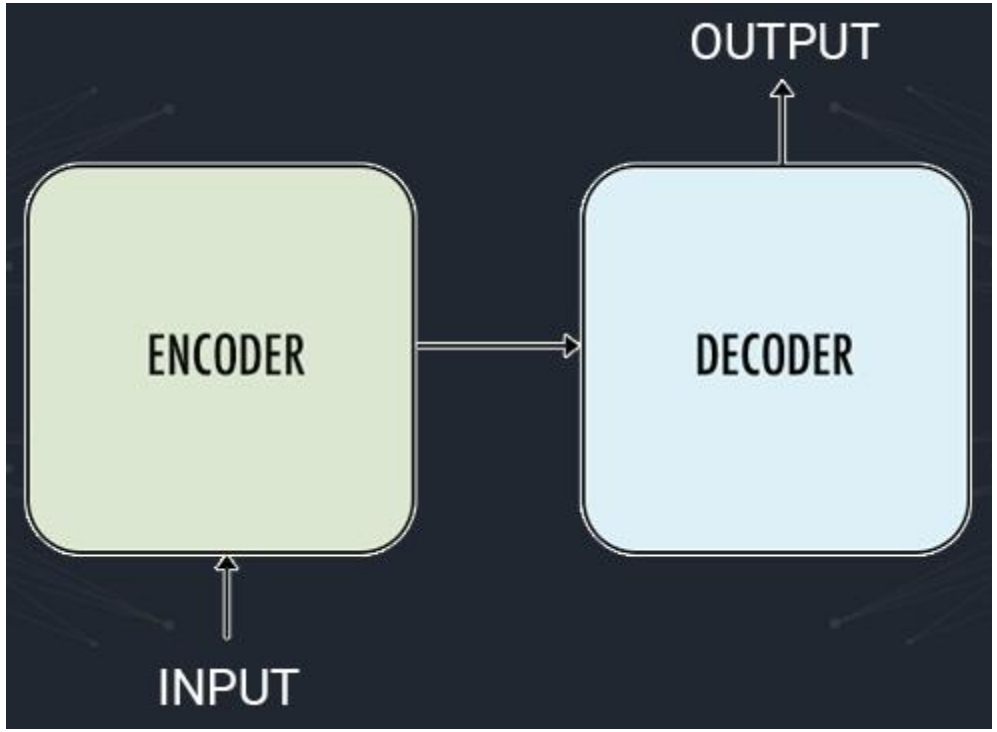
Seq2Seq: turn sequence in another sequence



Sequence to Sequence (often abbreviated to seq2seq) models is a special class of Recurrent Neural Network architectures that we typically use (but not restricted) to solve complex Language problems like Machine Translation, Question Answering, creating Chatbots, Text Summarization, etc.

Machine Language Translation

Les modèles de séquence sont super puissants → Sequence Model → *Sequence models are super powerful*

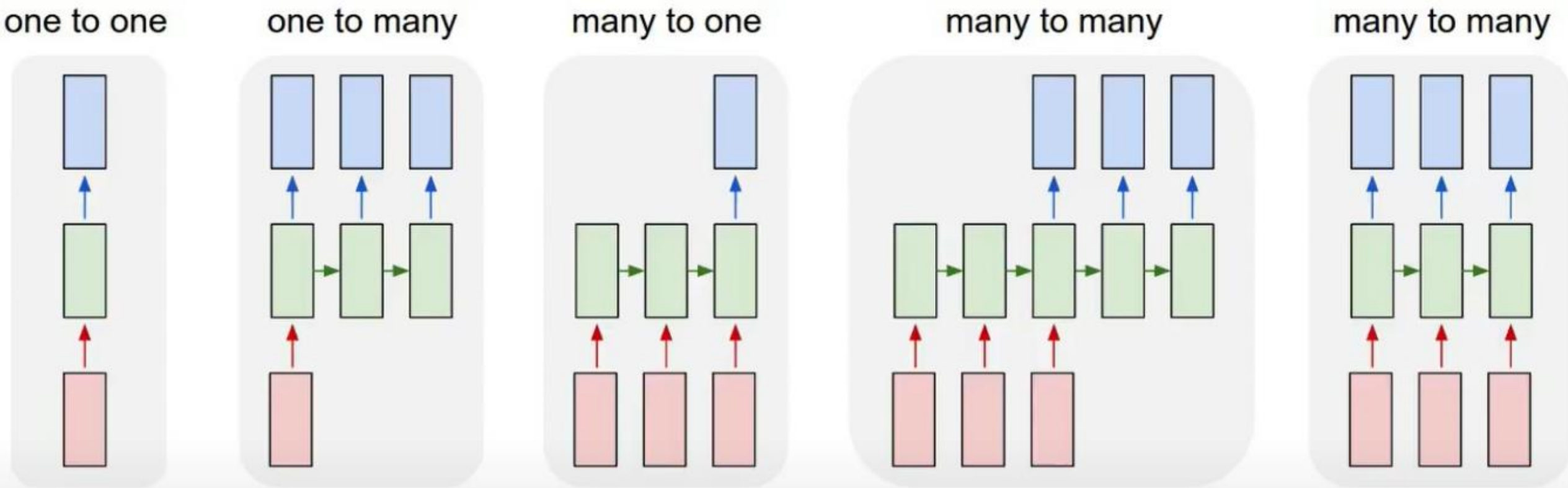
Text Summarization

A strong analyst have 6 main characteristics. One should master all 6 to be successful in the industry :
1. .....
2. .....

→ Sequence Model → *6 characteristics of successful analyst*

Chatbot

How are you doing today? → Sequence Model → *I am doing well. Thank you. How are you doing today?*



French

Er liebte zu essen

<X1> <X2> <X3> <X4>

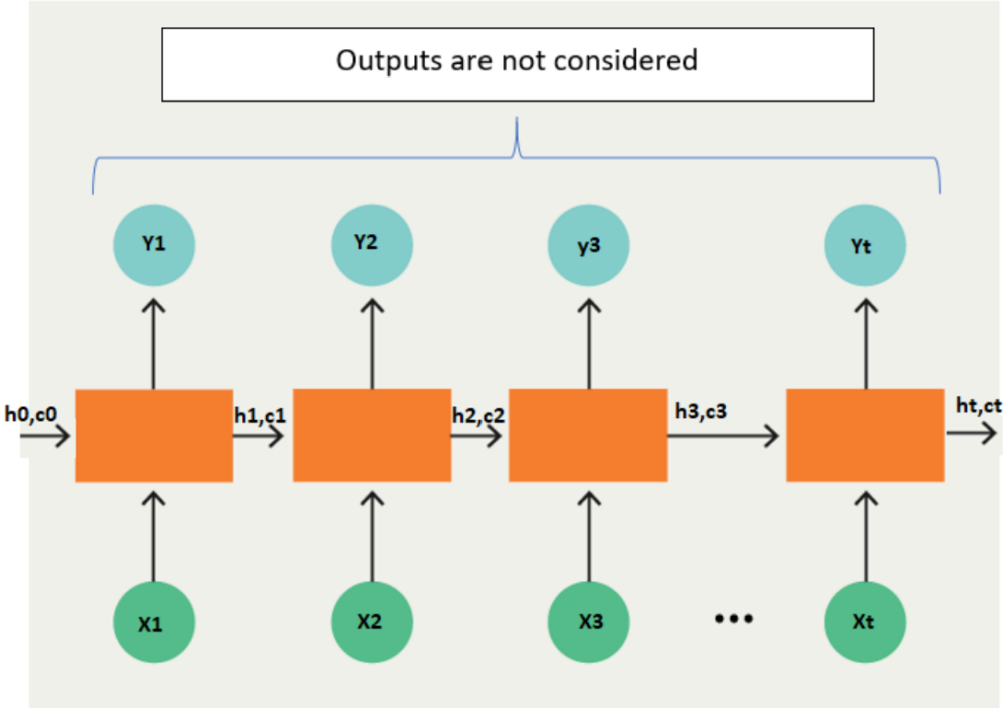
He loved to eat

<y1> <Y2> <Y3> <Y4>

Encoder :

- Both encoder and the decoder are LSTM models (or sometimes GRU models)
- Encoder reads the input sequence and summarizes the information in something called the **internal state vectors** or **context vector** (in case of LSTM these are called the hidden state and cell state vectors). We discard the outputs of the encoder and only preserve the internal states. This context vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions.
- The hidden states *h_i* are computed using the formula:

$$h_t = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$



Decoder :

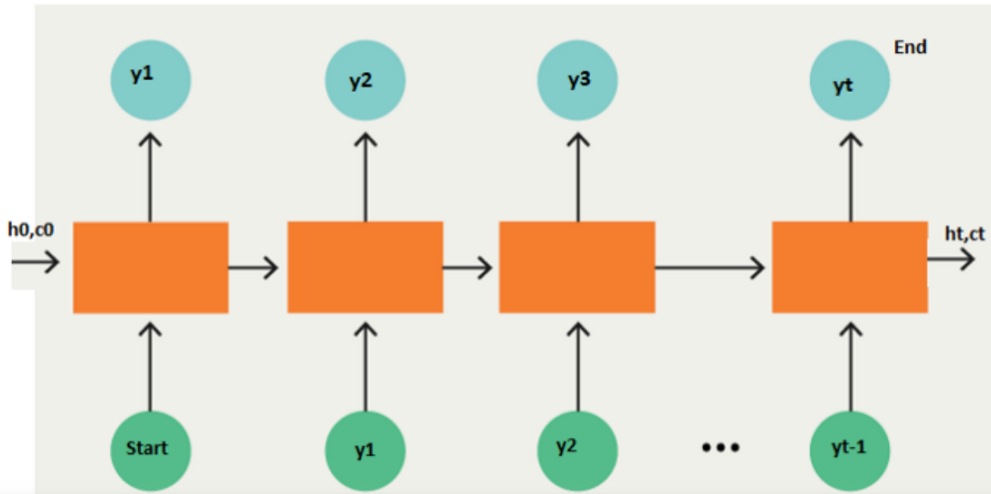
- The decoder is an LSTM whose initial states are initialized to the final states of the Encoder LSTM, i.e. the context vector of the encoder's final cell is input to the first cell of the decoder network. Using these initial states, the decoder starts generating the output sequence, and these outputs are also taken into consideration for future outputs.
- A stack of several LSTM units where each predicts an output y_t at a time step t.
- Each recurrent unit accepts a hidden state from the previous unit and produces and output as well as its own hidden state.
- Any hidden state *h_i* is computed using the formula:

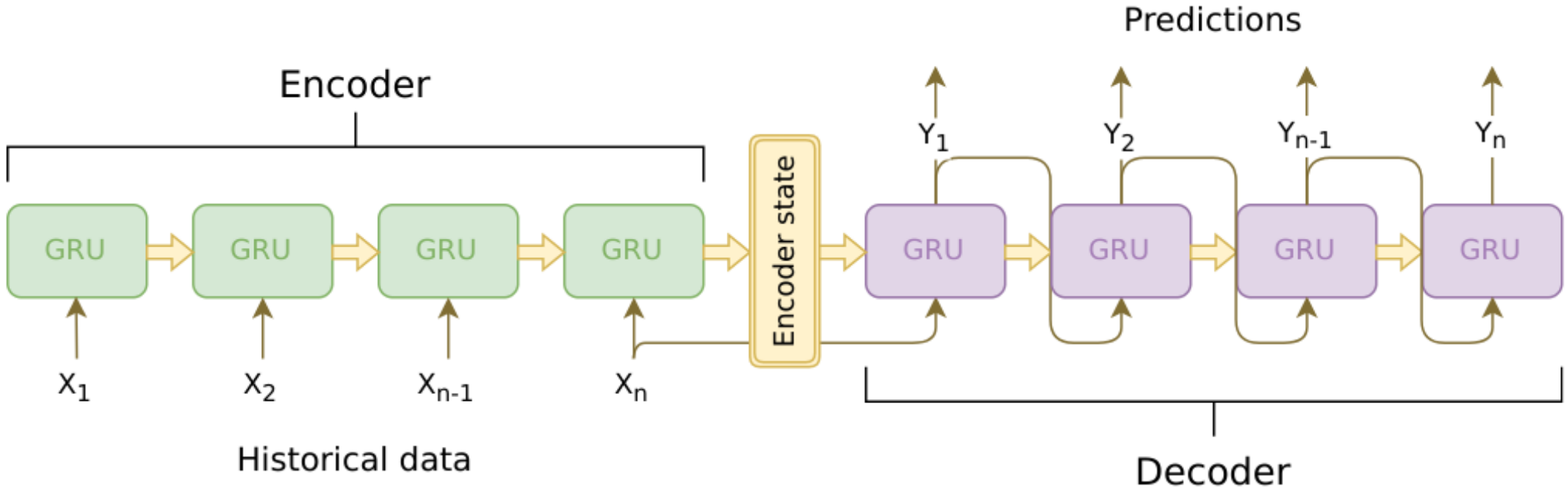
$$h_t = f(W^{(hh)} h_{t-1})$$

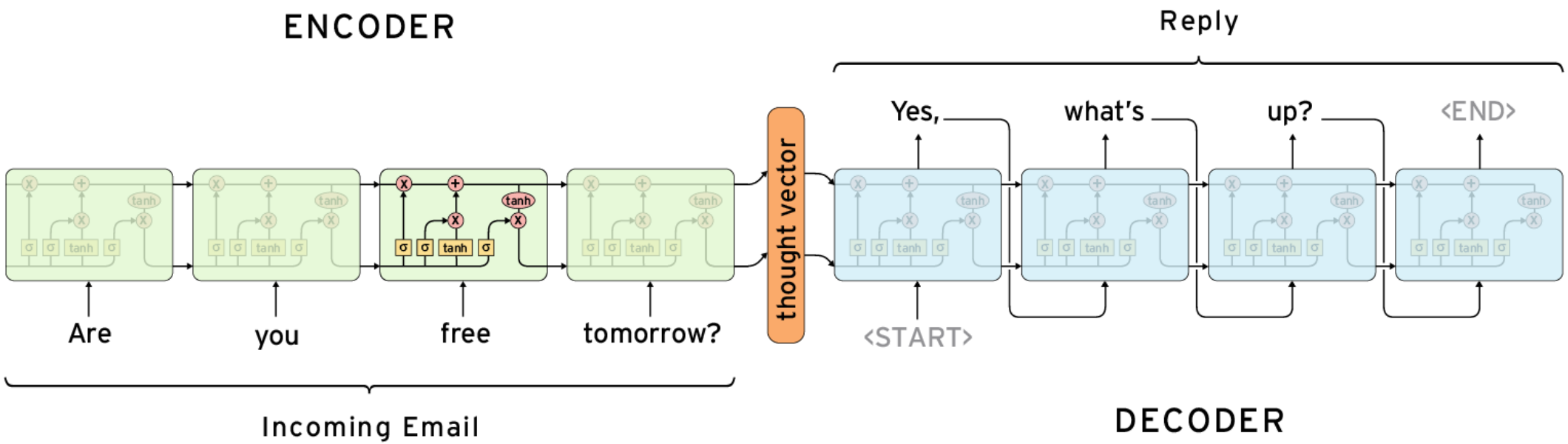
- The output *y_t* at time step *t* is computed using the formula:

$$y_t = softmax(W^S h_t)$$

- We calculate the outputs using the hidden state at the current time step together with the respective weight W(S). Softmax is used to create a probability vector which will help us determine the final output (e.g. word in the question-answering problem).

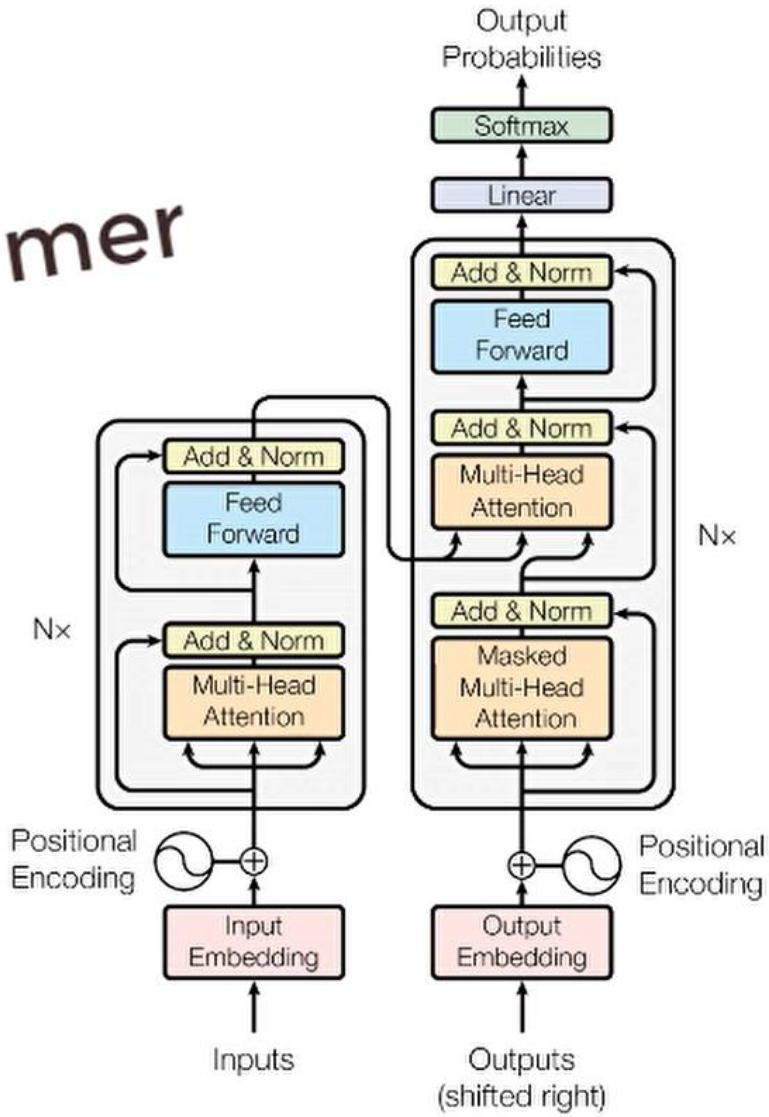



Predictions
Encoder
GRU
GRU
GRU
GRU
Encoder state
GRU
GRU
GRU
GRU
Y_1
Y_2
Y_{n-1}
Y_n
X_1
X_2
X_{n-1}
X_n
Historical data
Decoder


ENCODER
Reply
Yes,
what's
up?
<END>
thought vector
Are
you
free
tomorrow?
<START>
Incoming Email
DECODER

LSTM Vs Transformer

Transformer



Attention Is All You Need

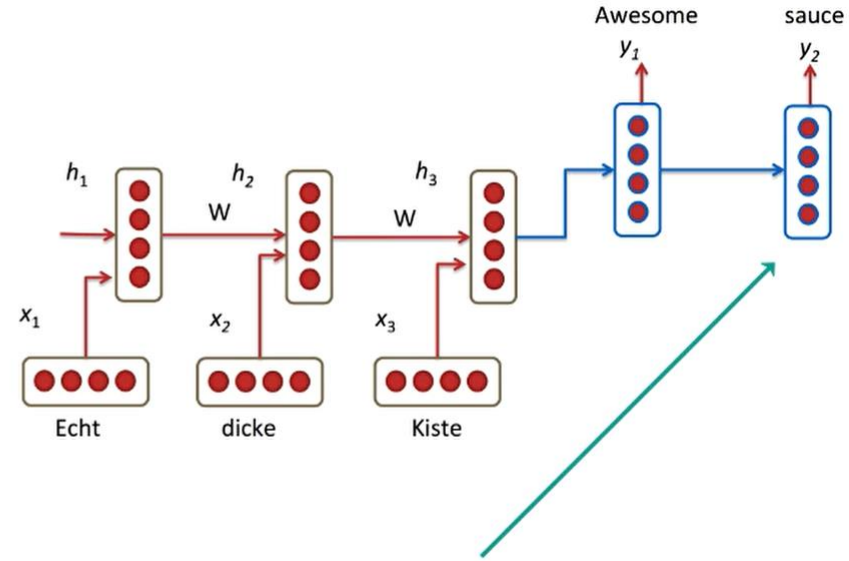
Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

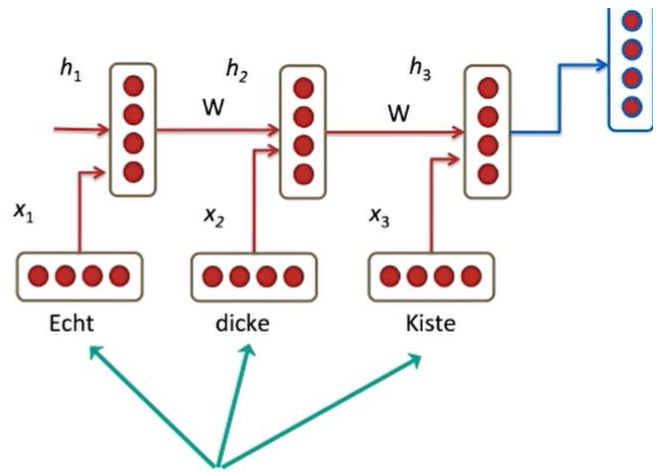
LSTM Networks

1. Slow



Timestep 5

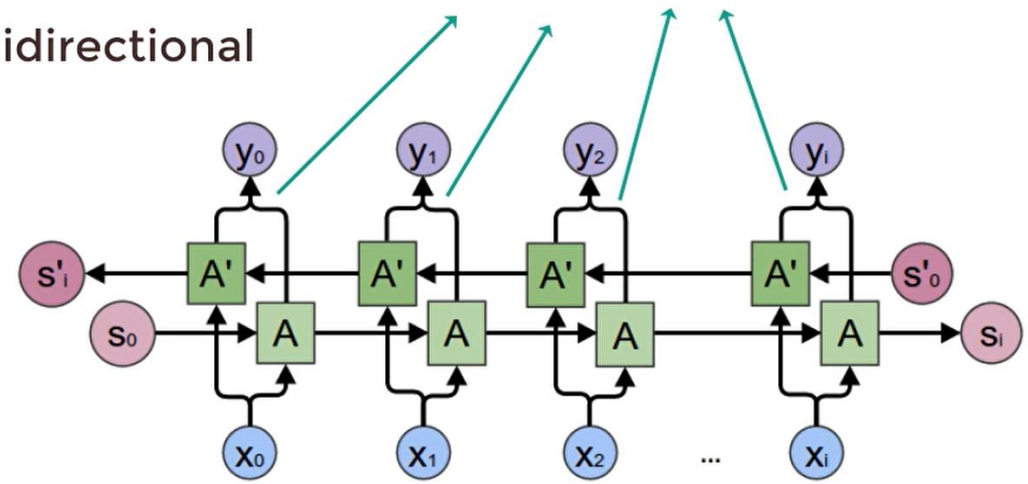
2. Not truly Bidirectional



True meaning of source words not entirely captured

Bidirectional

Simple concatenation

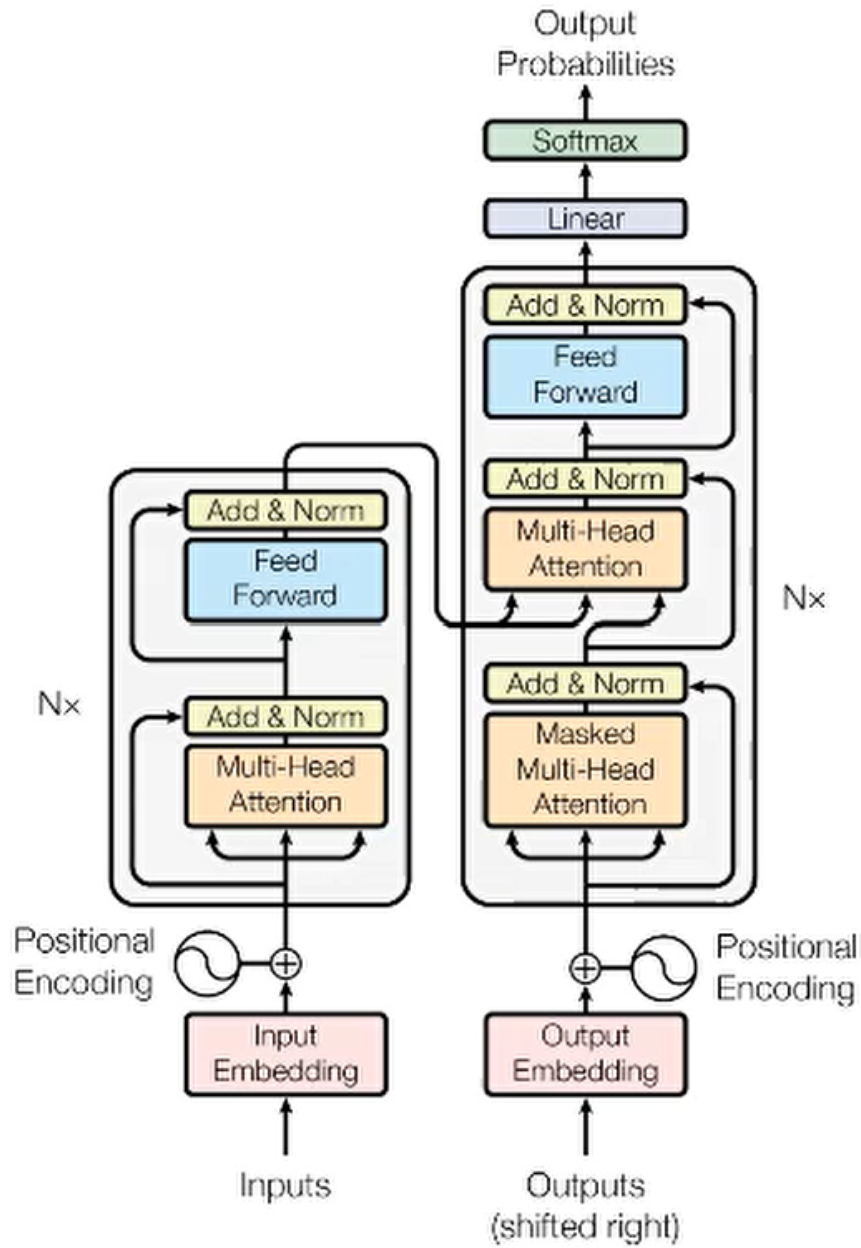


Transformer

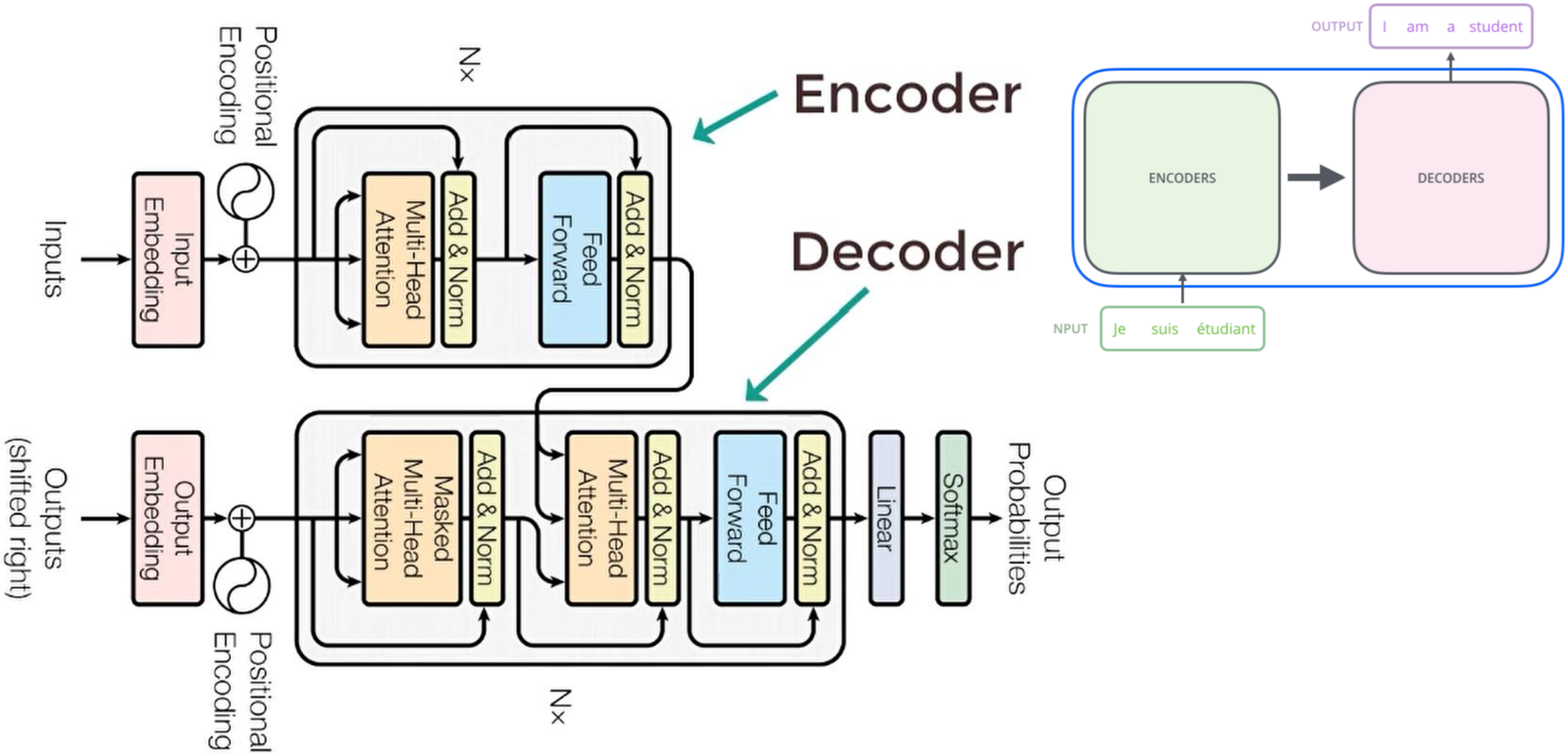
1. ~~Slow~~ Faster

2. ~~Not truly Bidirectional~~

   Deeply Bidirectional



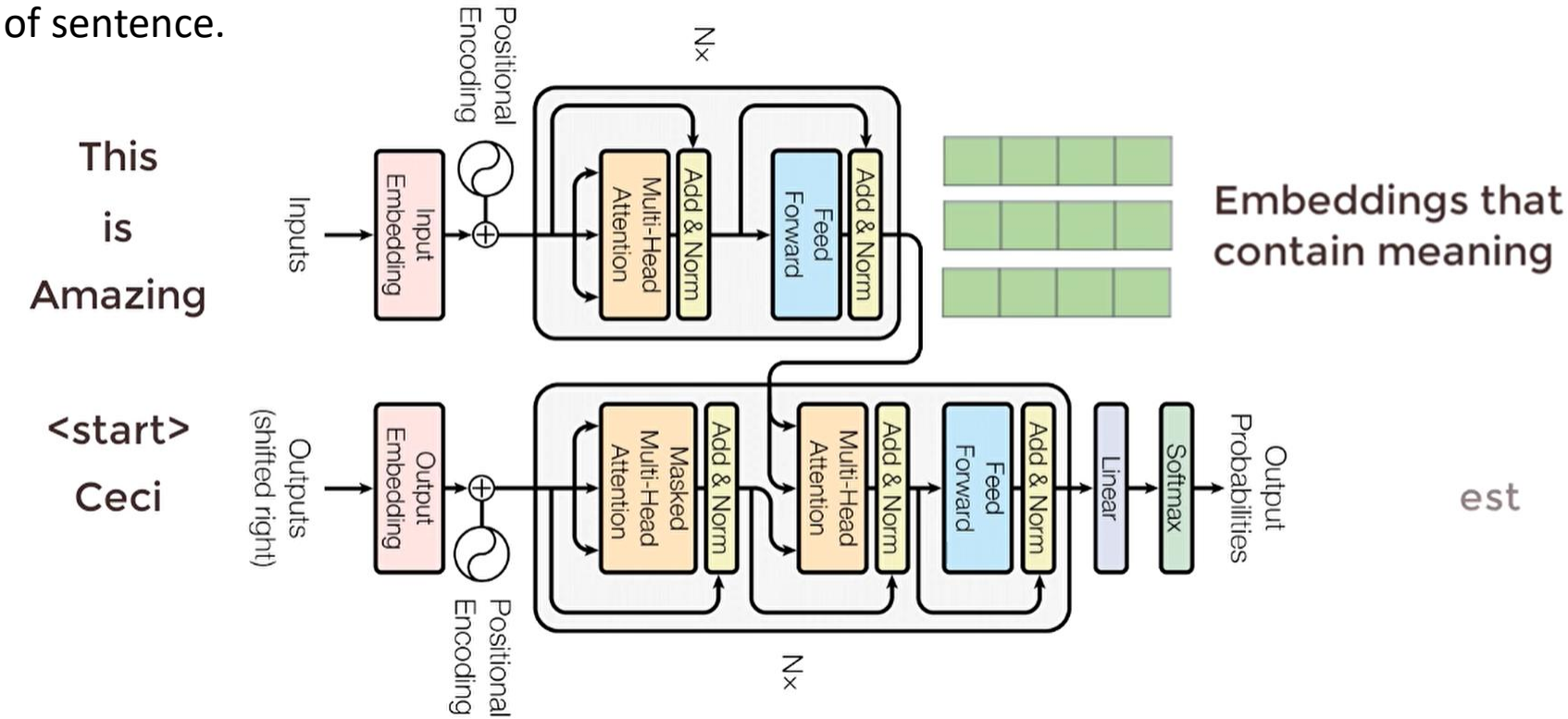
Transformer Flow



ES. TRANSLATION ENGLISH-FRENCH

ENCODER= Take english words simultaneously and generate embeddings for every word simultaneously : These embeddings are VECTORS that encapslute the meaning of the words, similar words have similar number inside the vectors.

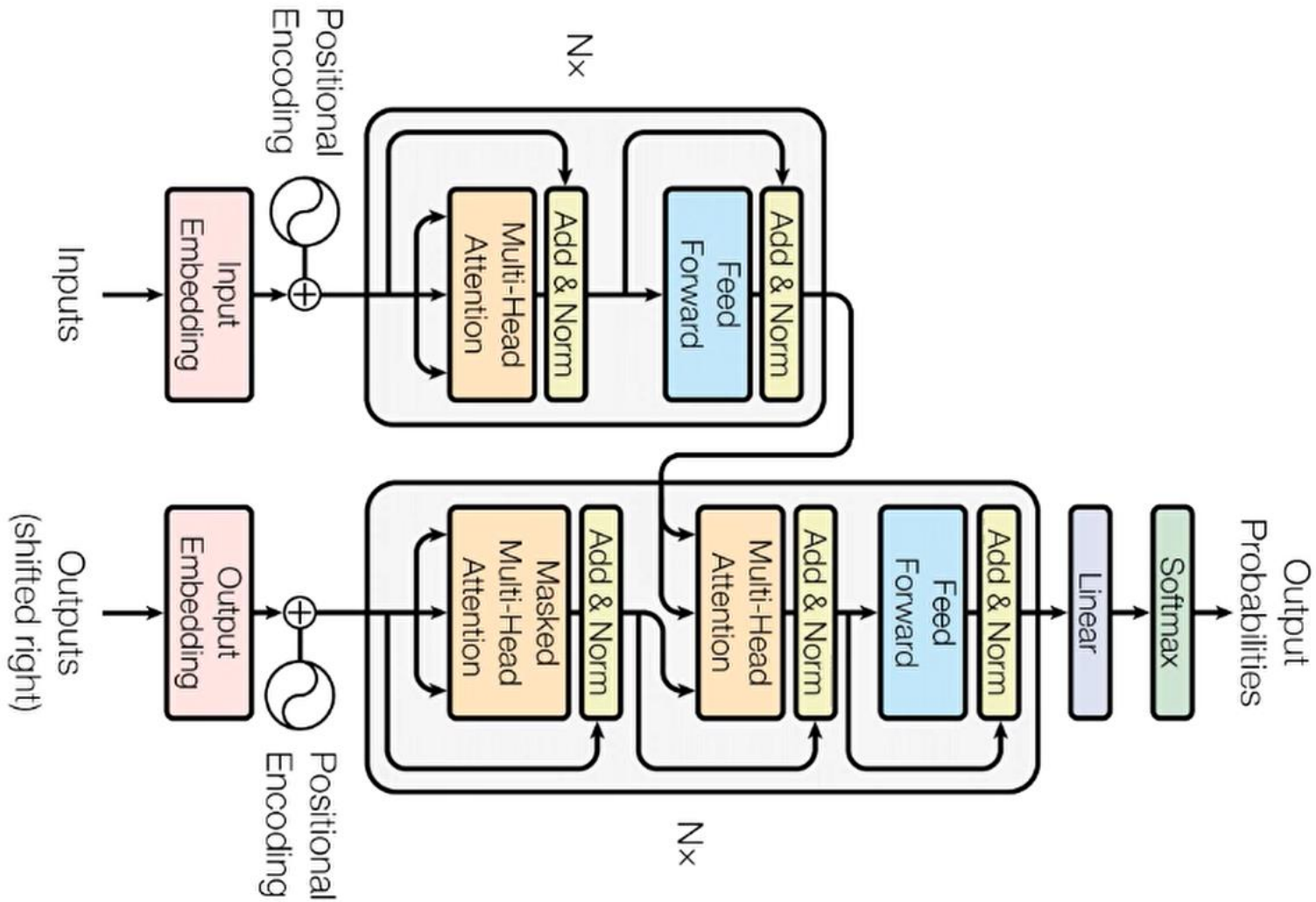
DECODER= Take embeddings from Encoder, and the first generate translate french word and use to generate the next french word. Keep the generate words at the time till the end of sentence.



WE SEE THE SEPARTION IN TASK:

ENCODER LEARN : WHAT IS ENGLISH, WHAT IS GRAMMER, WHAT IS CONTEXT

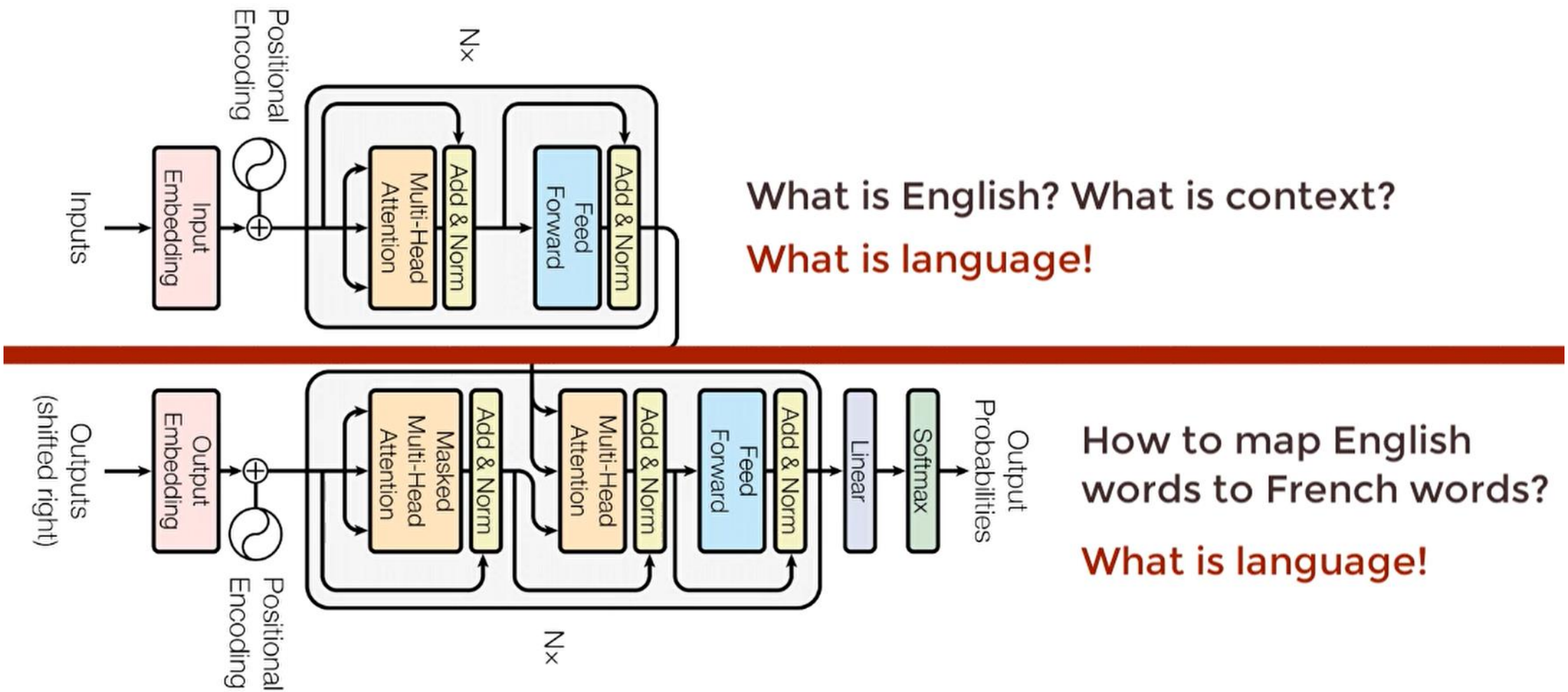
DECODER LEARN : WHAT ENGLISH WORD TO ENGLISH -→MAPPING



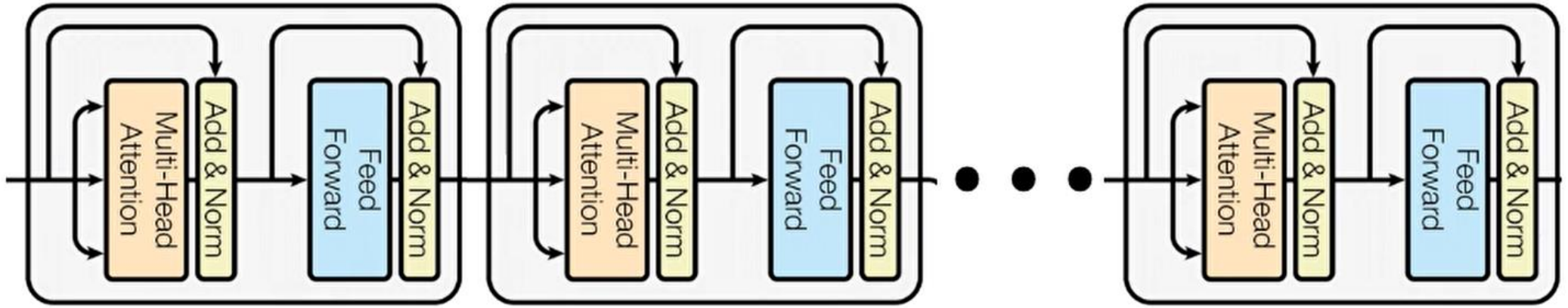
ENCODER

DECODER

IN BOTH PART DECODER AND ENCODER
UNDERSTAND THE LANGUAGE!!



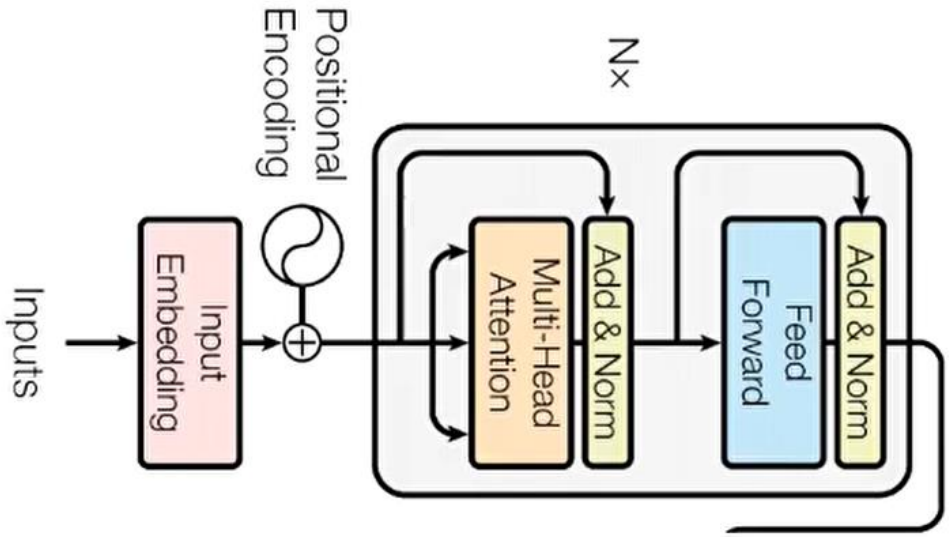
IF WE SET THE ENCODER:
We have BERT structure

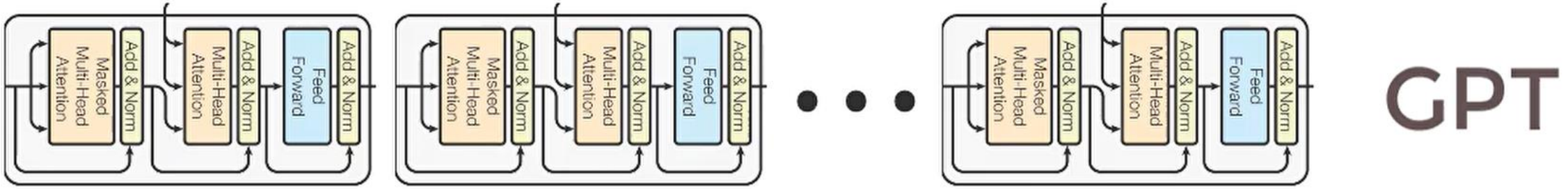


BERT

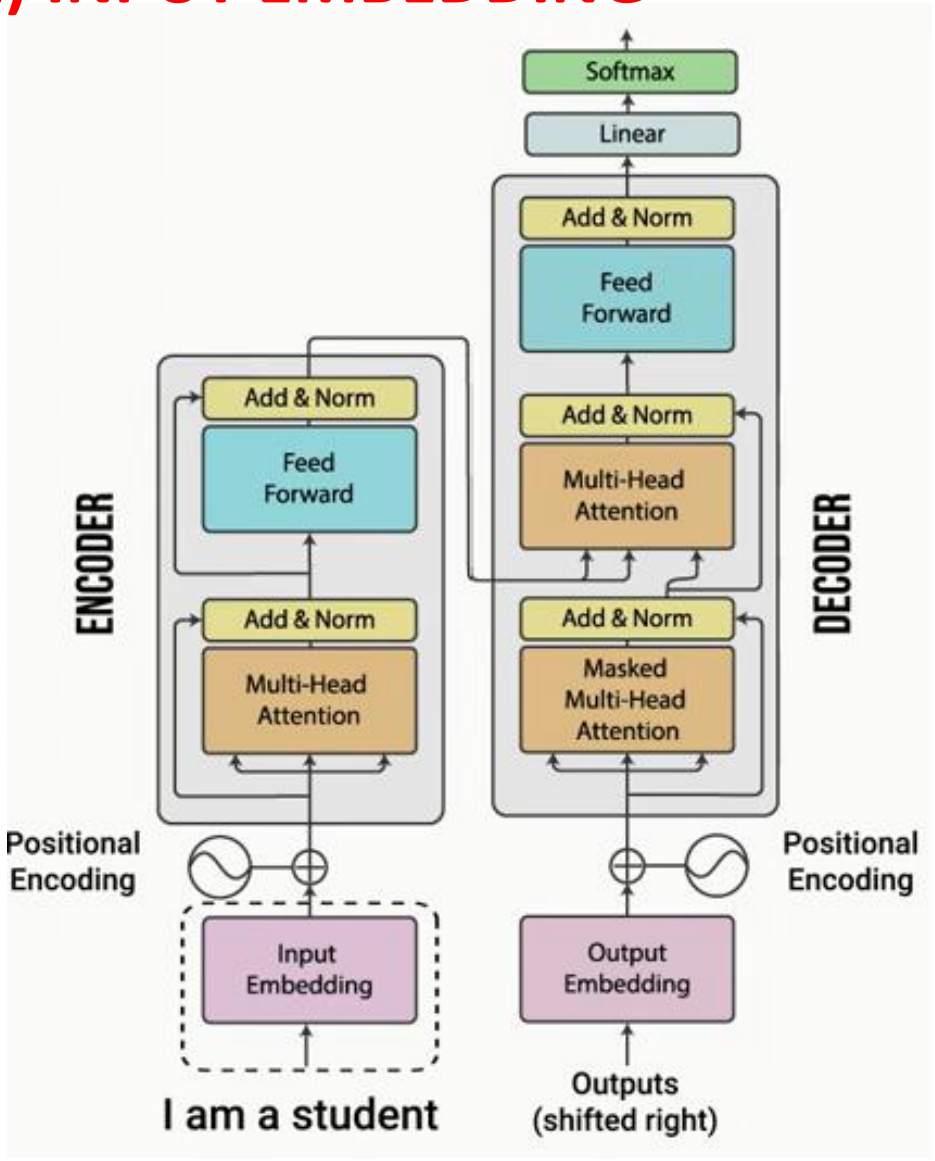
Bidirectional **E**ncoder **R**epresentation from **T**ransformers

IF WE SET THE DECODER:
We have GPT structure

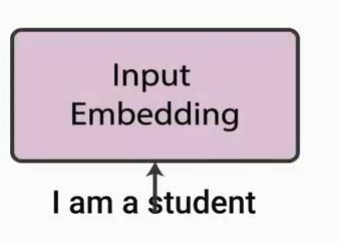




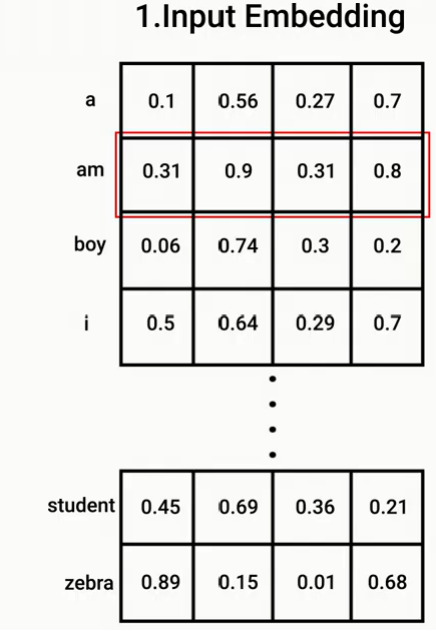
1) INPUT EMBEDDING



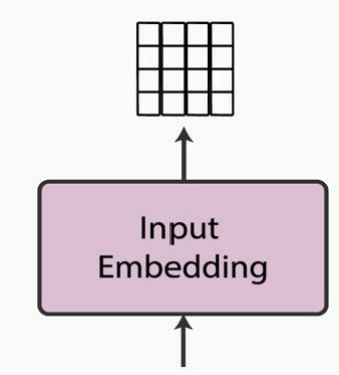
1) We pass the input sequence into thw word embedding layer



Word Embedding layer is a matrix with numerical representation for each word in the sentence and retrun the output



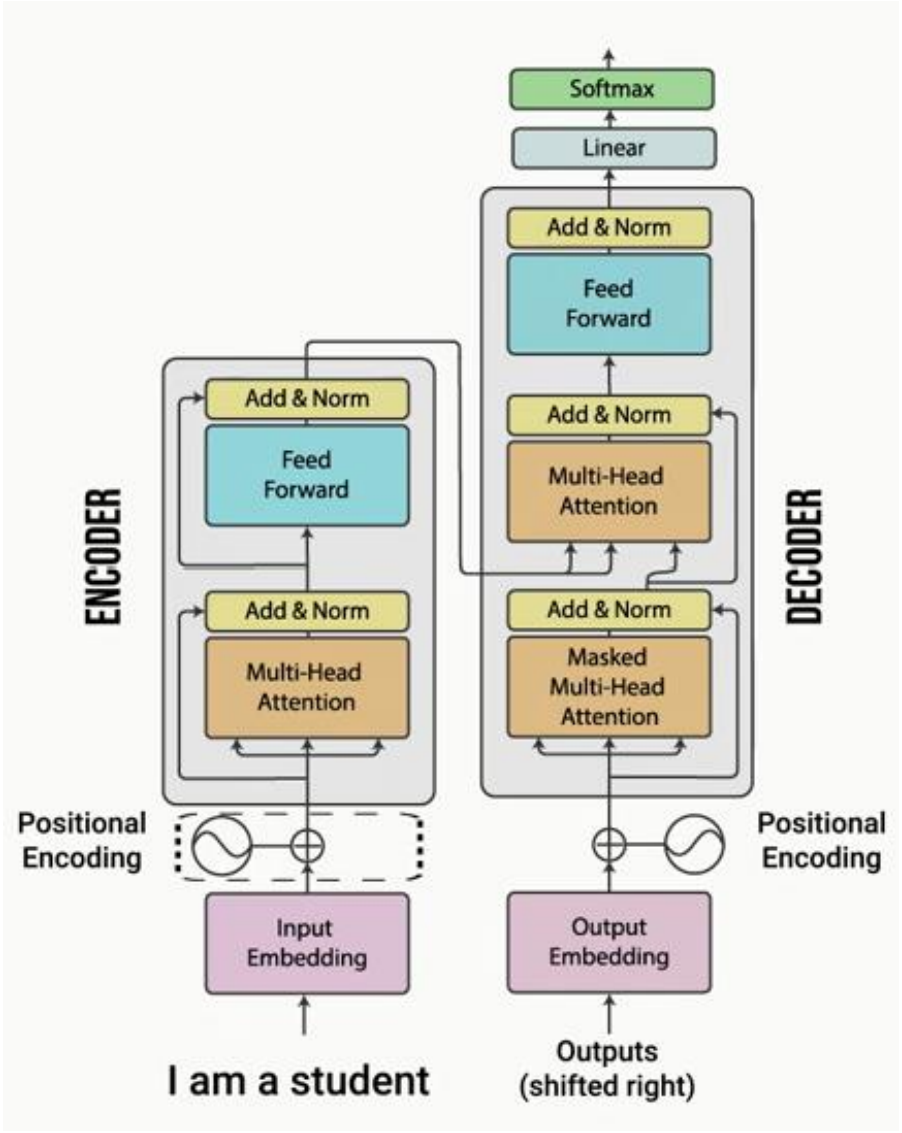
Return the matrix to the output




King
Queen
Woman
Princess
Royalty
Masculinity
Femininity
Age
...
0.99
0.99
0.05
0.7
0.99
0.05
0.93
0.6
0.02
0.01
0.999
0.5
0.98
0.02
0.94
0.1

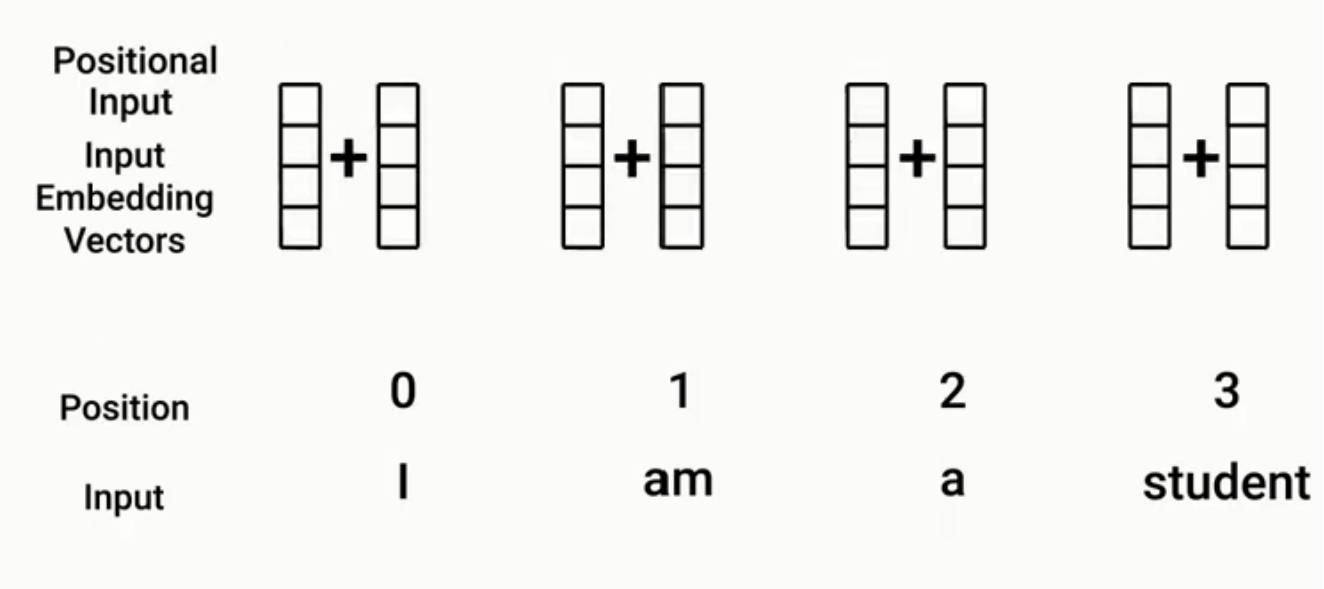
2) POSITIONAL ENCODING

The scope is to enject the information about the POSITION of each word into its embedding representation



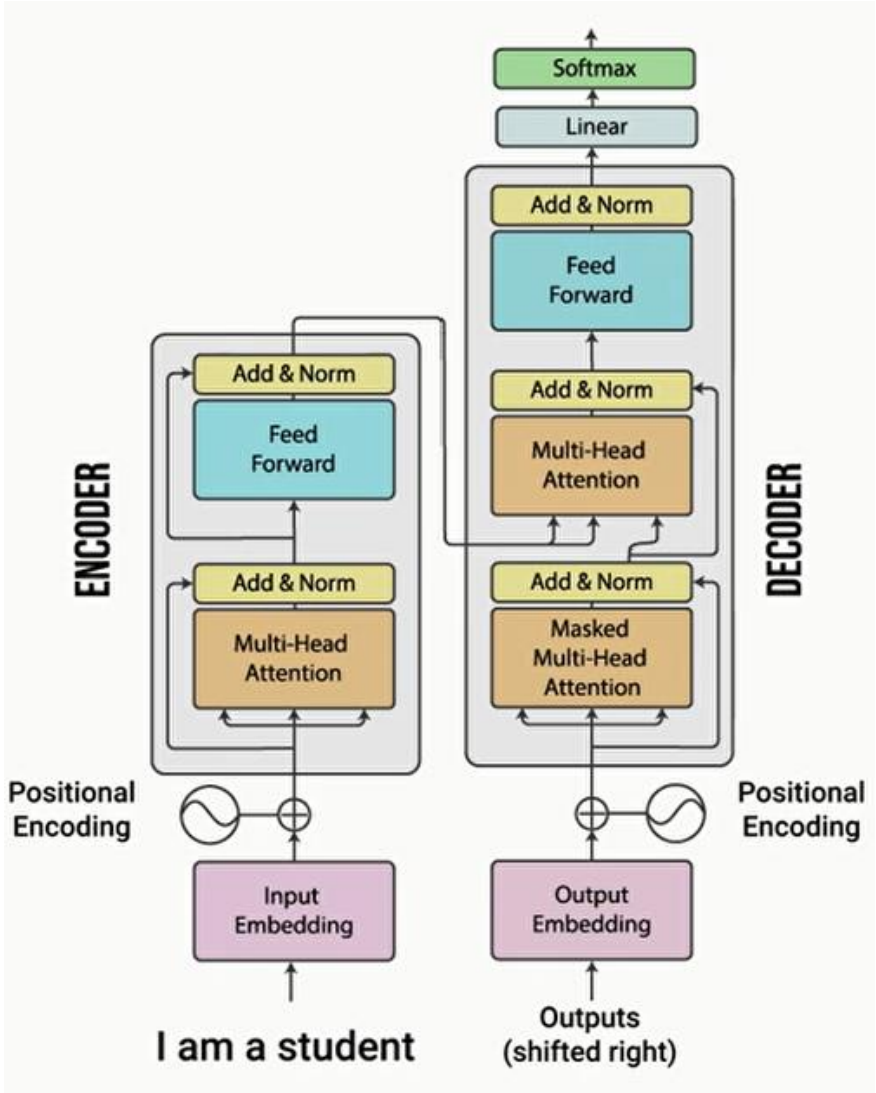
$$\mathrm{PE}_{(\mathrm{pos},2\mathrm{i})} = \sin(\mathrm{pos}/10000^{2\mathrm{i}/\mathrm{dmodel}})$$

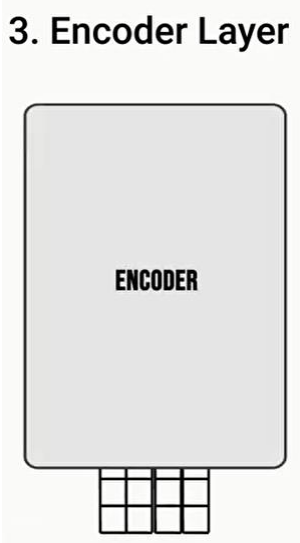
$$\mathrm{PE}_{(\mathrm{pos},2\mathrm{i}+1)} = \cos(\mathrm{pos}/10000^{2\mathrm{i}/\mathrm{dmodel}})$$

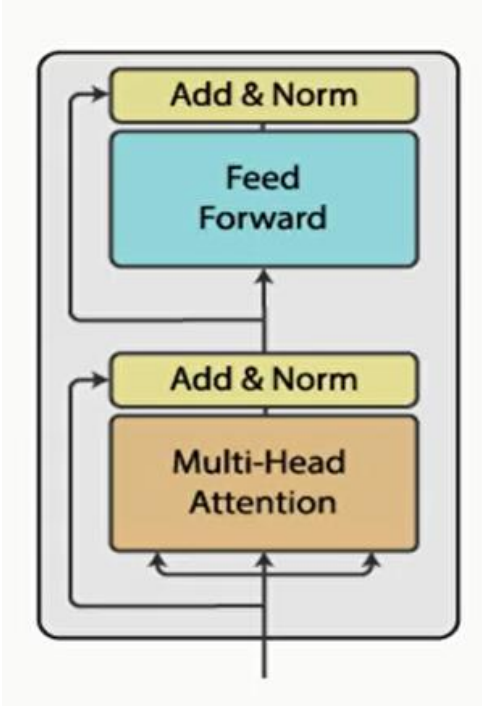


3) ENCODER LAYER

This positional input embedded matrix pass into the Transformer Encoder which contain two submodels:





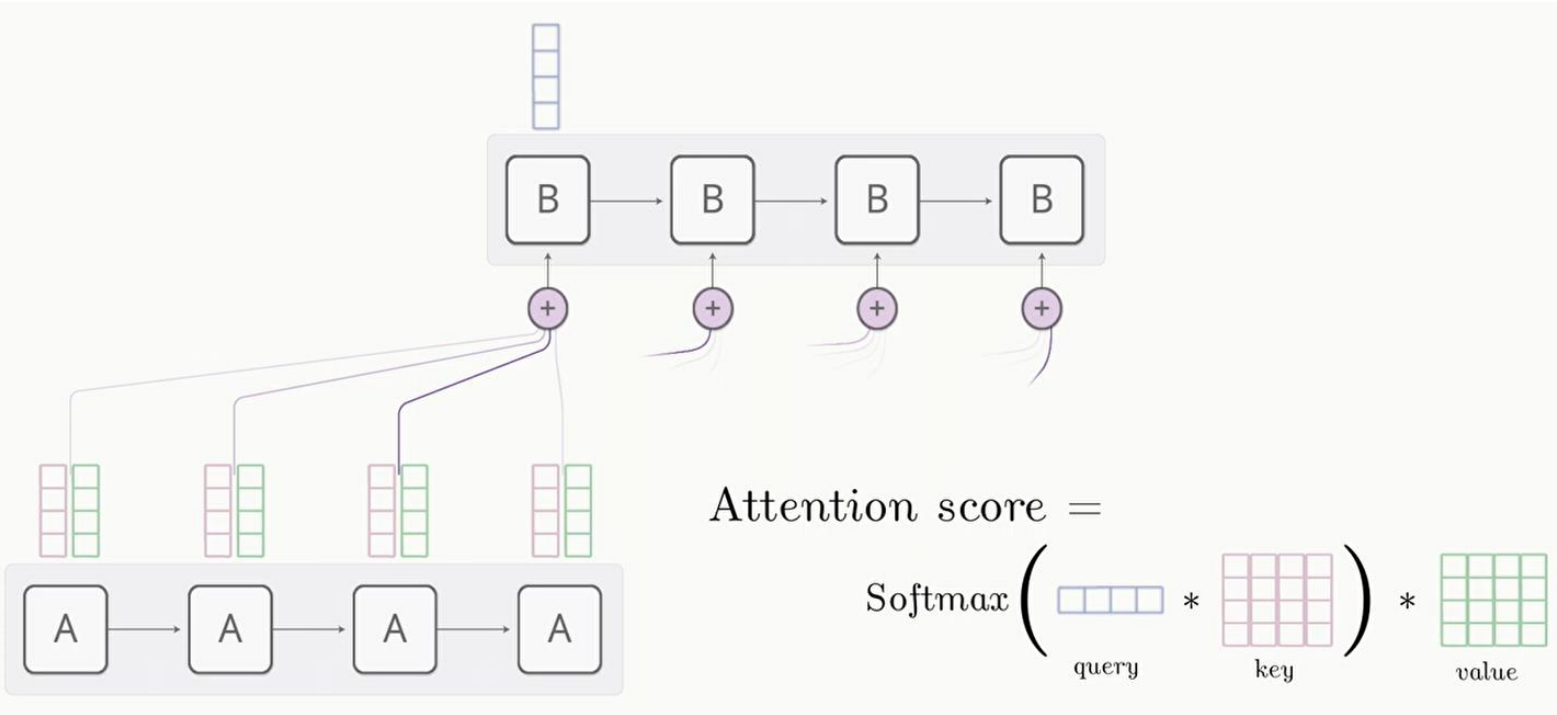


Multi Head Attention model is followed by fully connected feed foward network

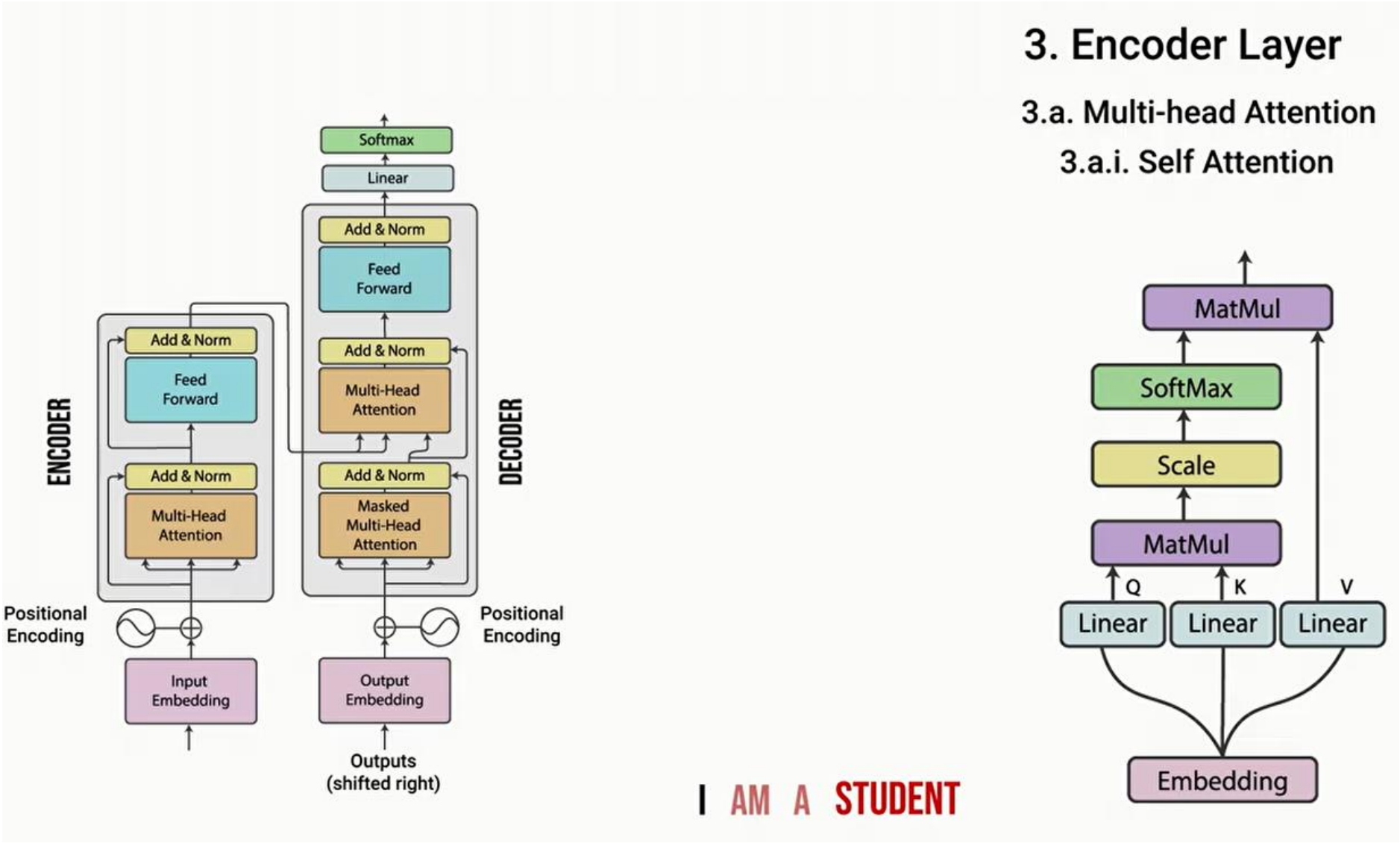
Both have a residual connection follow by leyer normalization

3a. Multi-head Attention

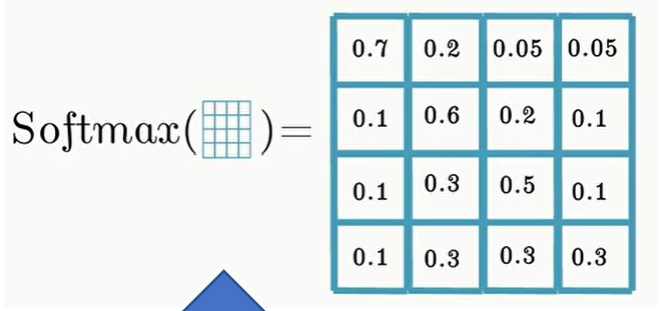
When using attention in regular sequence model: once the encoder is done processing the input sequence it passes on this output to the decoder which then tries to focus on which of the input representation vectors are most relevant to its current time step output and to do this it would create an **attention query vector** for that time step



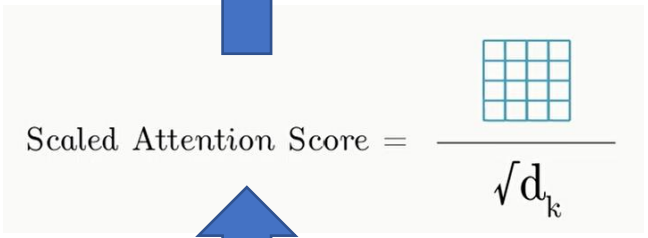
Use the encoder output as its attention key and value vectors and then compute an **attention score** using these query key and value vectors and this type of attention is known as regular or vanilla attention while in the transformer encoder a slight variation of this vanilla attention is used and this is known as **self-attention** which in this case it simply allows the encoder compute the relevancy of each word in the sentence to all the other words in the same sentence.



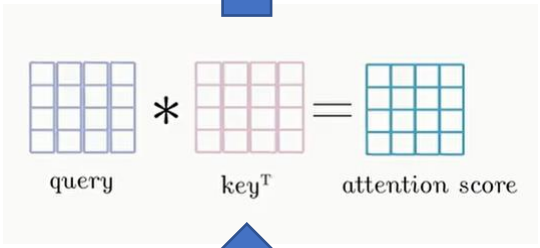
The query key and value vectors all come from the same sequence so when encoding the input sentence self attention allows the decoder learned that the word “student” is most relevant to the word “I” to compute the self attention.



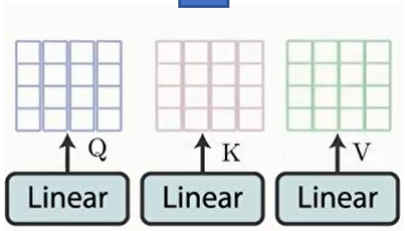
5) These scores are passed into a soft max layer which returns probability values between 0 and 1. Finally this SOFTMAX attention scores are used to multiply the value vector to produce a weighted output vector



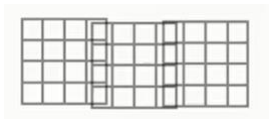
4) The scoring matrix then gets scaled down by dividing by the square root of the dimension of the key vector and this is to allow for more stable gradients as multiplying these values several times can have exploding effects



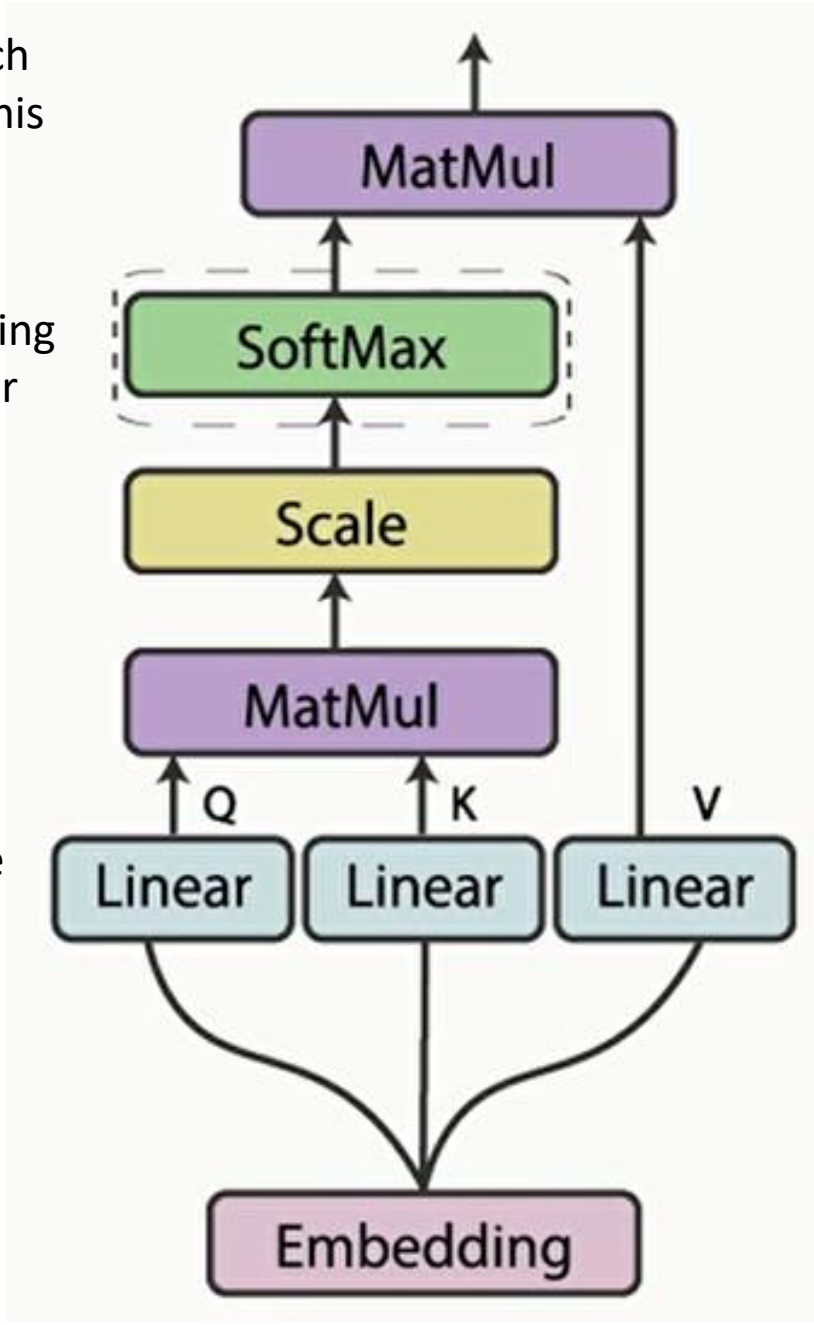
3) Dot multiplication is applied to the query and the transpose of the key to produce a scoring matrix and this scoring matrix is what determines the relevance of a word to other words in the matrix the higher the score the more relevant the word is

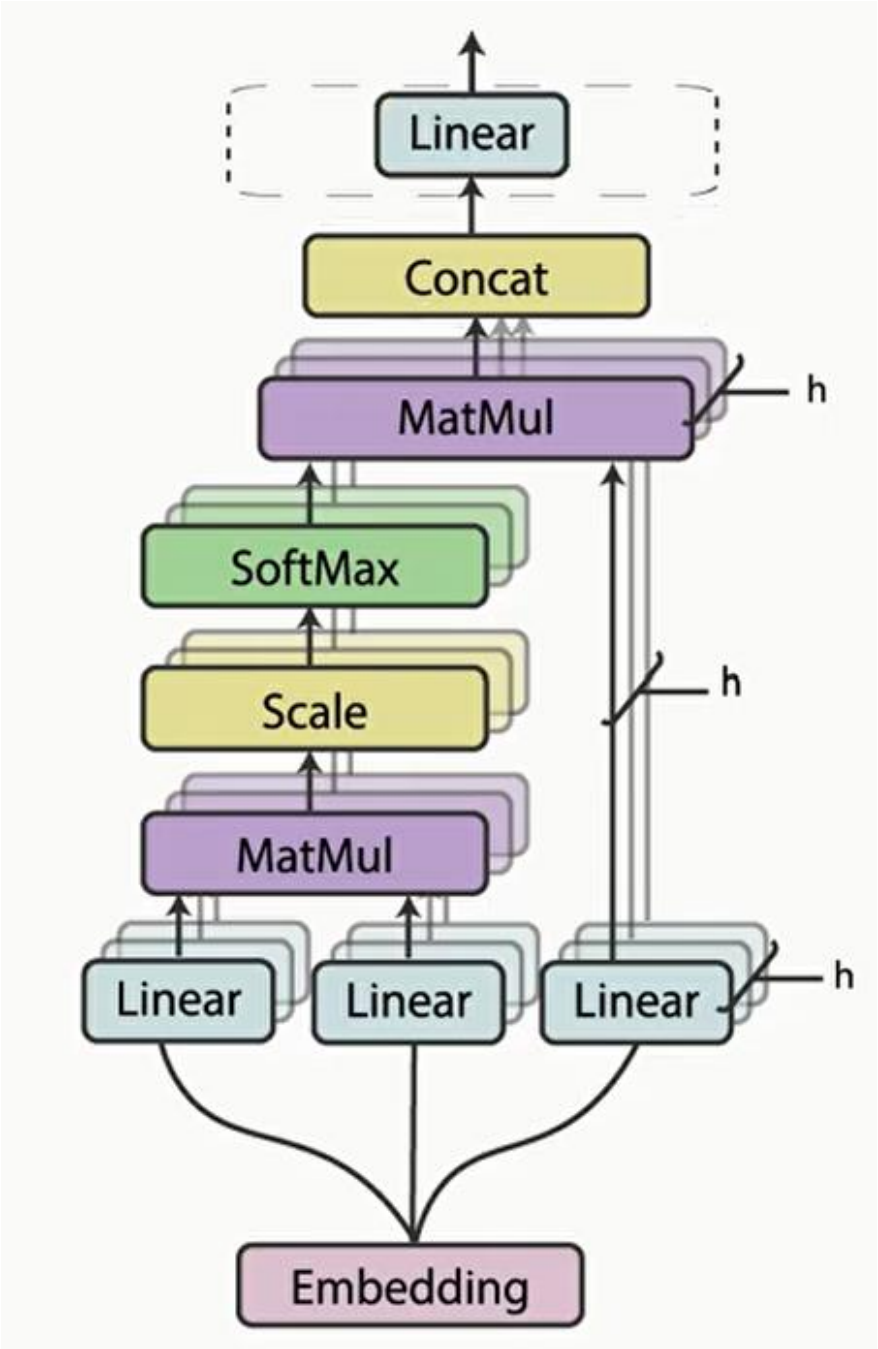


2) Create the ENCODER attention query Q key K and value V vectors

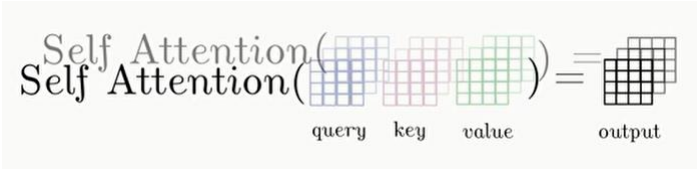


1) The positional input embeddings are fed into three distinct linear layers

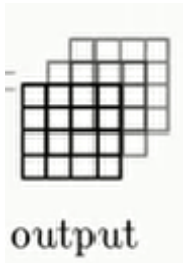


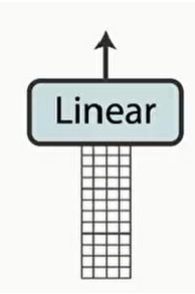


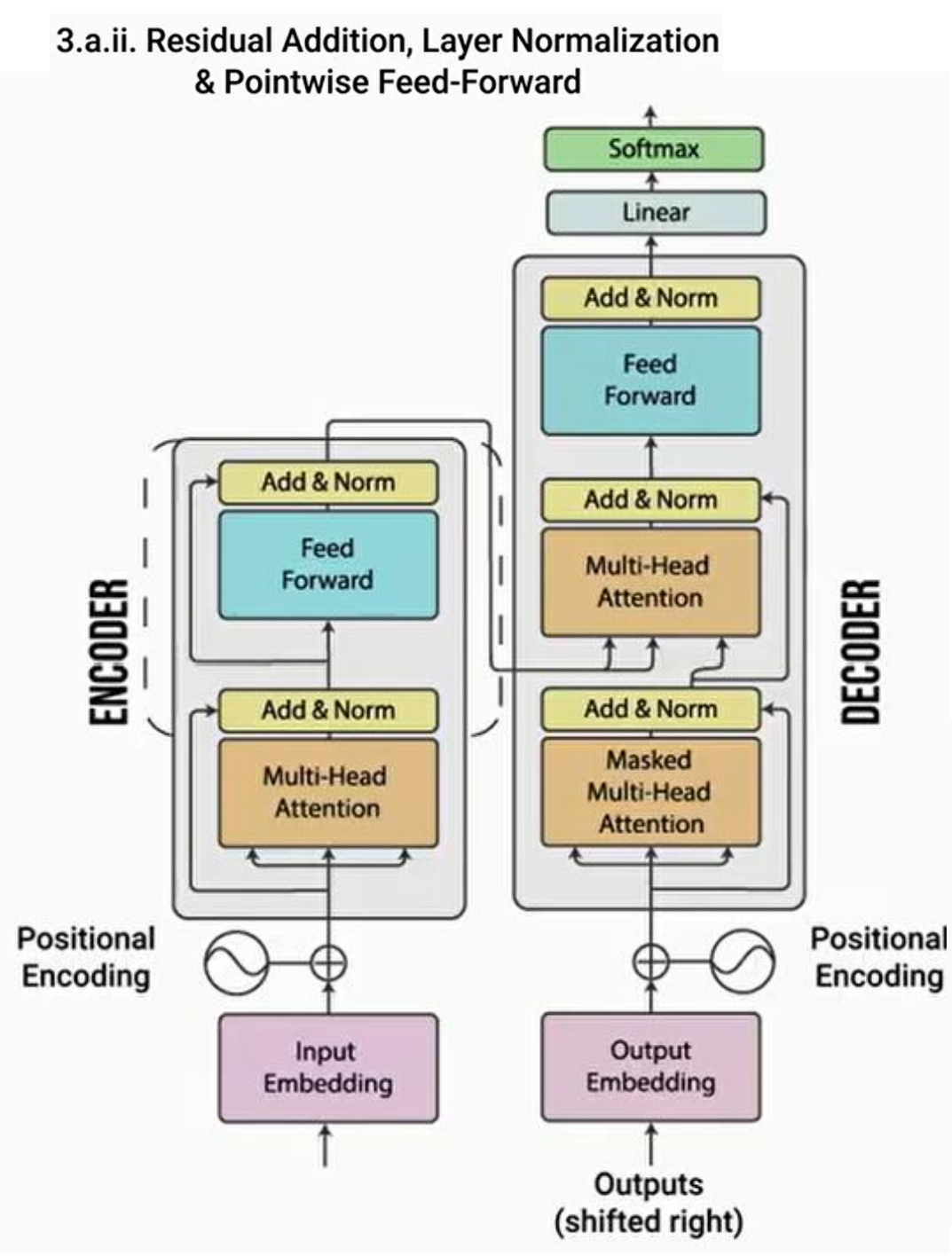
To account for the fact that a word can mean different things to different neighbors the transformer combines the output of several of these self-attention models and this is where the name multi-head attention comes from

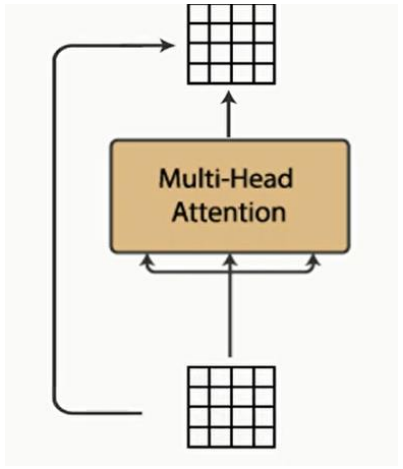


This combination is done by copying the query key and value matrices "h" number of times computing the self-attention operation over each copy concatenating each attention output together and then linearly transforming it into the expected dimensions

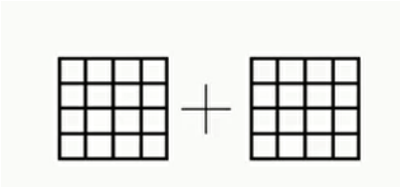




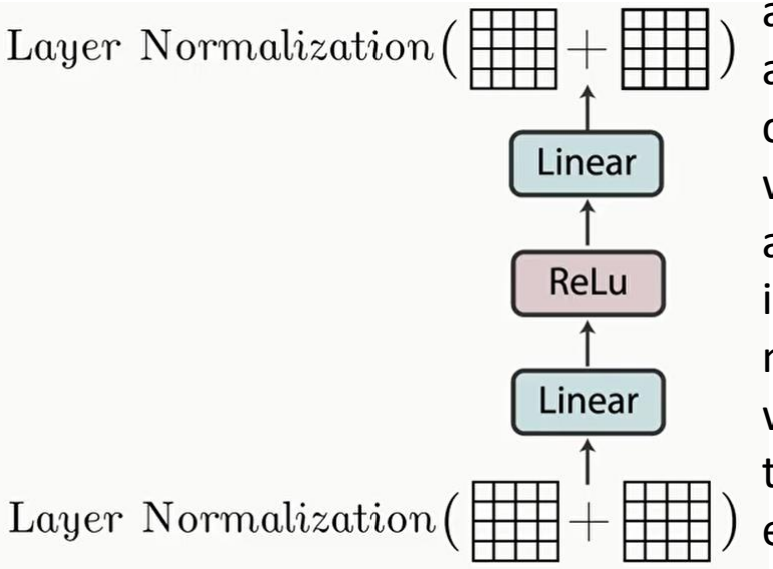




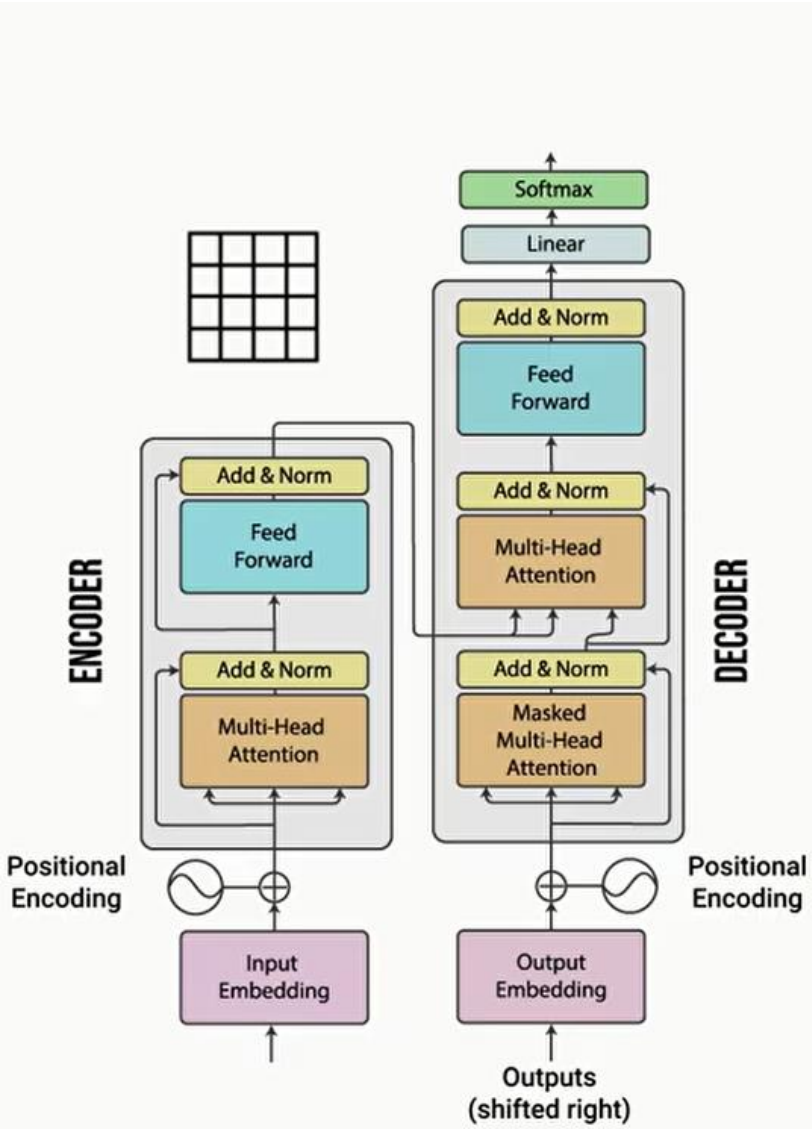
once the motor head attention computation is done a residual connection is added to its output

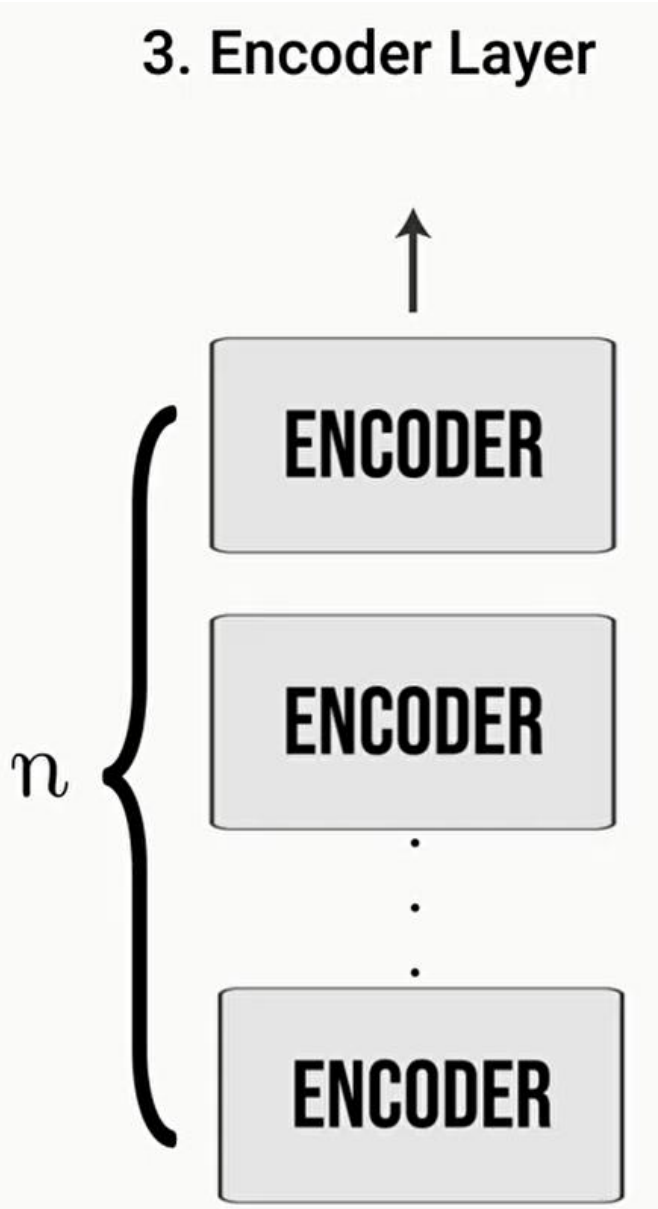


which means that we simply add the input of the layer together with the output of the layer to allow gradients flow through the net directly without having to pass through non-linear functions



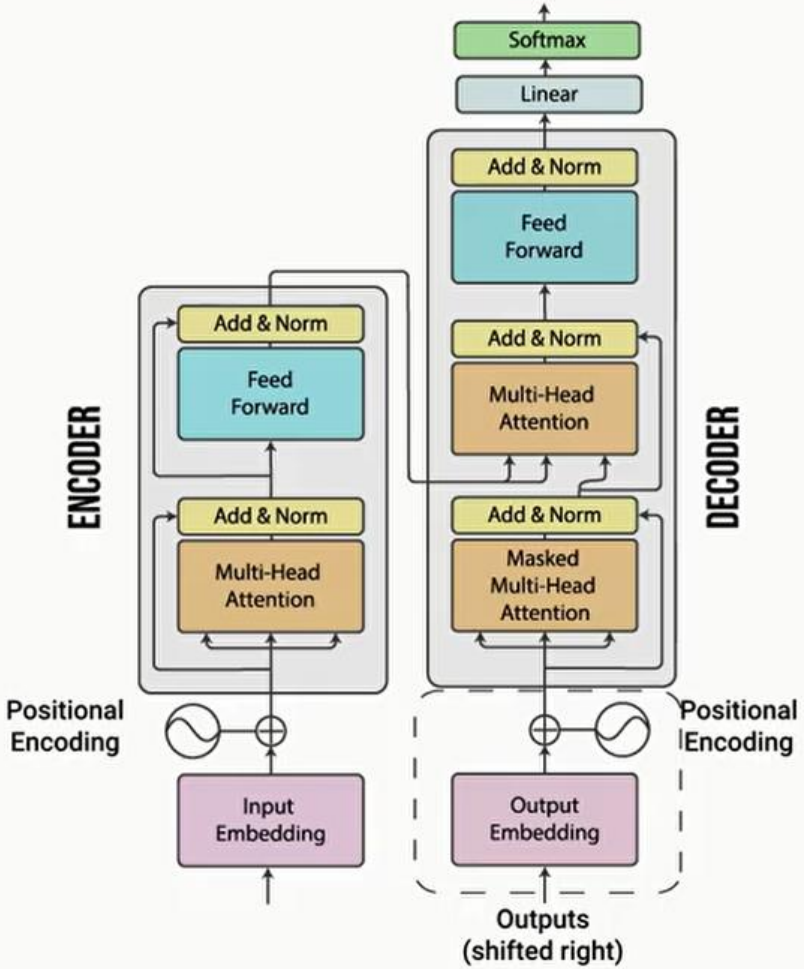
and layer normalization is then applied, then gets feed into a fully connected feed forward network which consists of **two Linear Layers** and **Relu** activation in between and is then residually connected and normalized and that pretty much wraps up all of the operations that take place in the transformer encoder layer

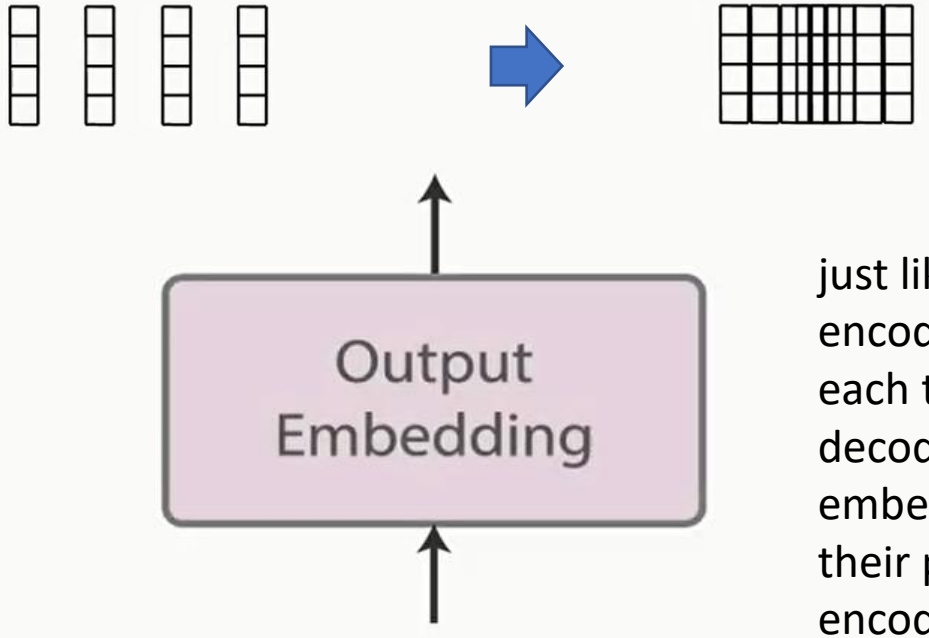




This encoder can be stacked up n times to further encode the inputs and thereby improve the predictive power of the network once the encoding phase is done the output is passed to the decoder whose job is to generate text sequences and also has a similar sub-layer structure to the encoder which are two multi-head attention layers…

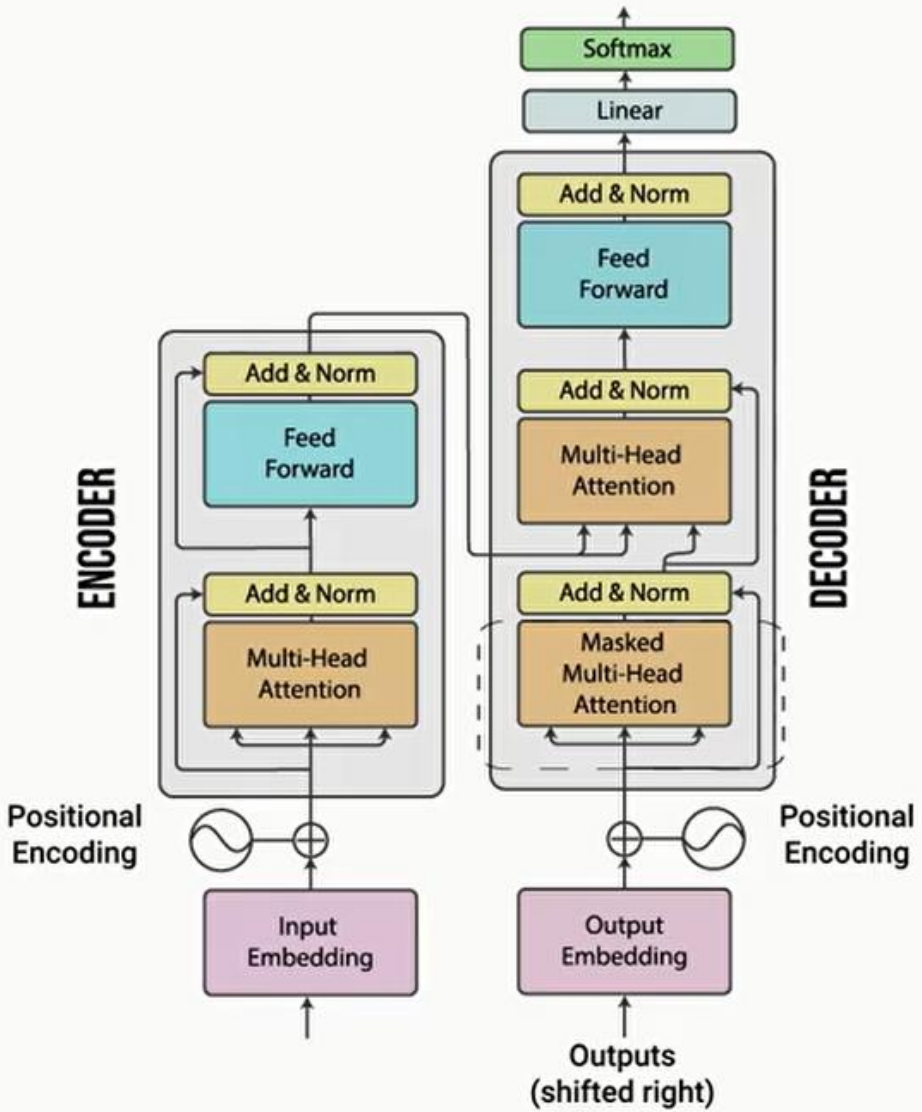
4. Output Embedding & Positional Encoding



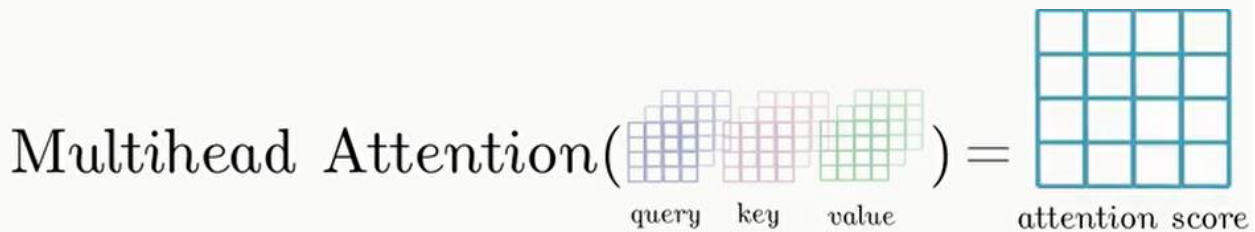


just like with the encoder inputs at each time step the decoder inputs are embedded added to their positional encodings transformed into the decoder attention

5. Decoder Layer



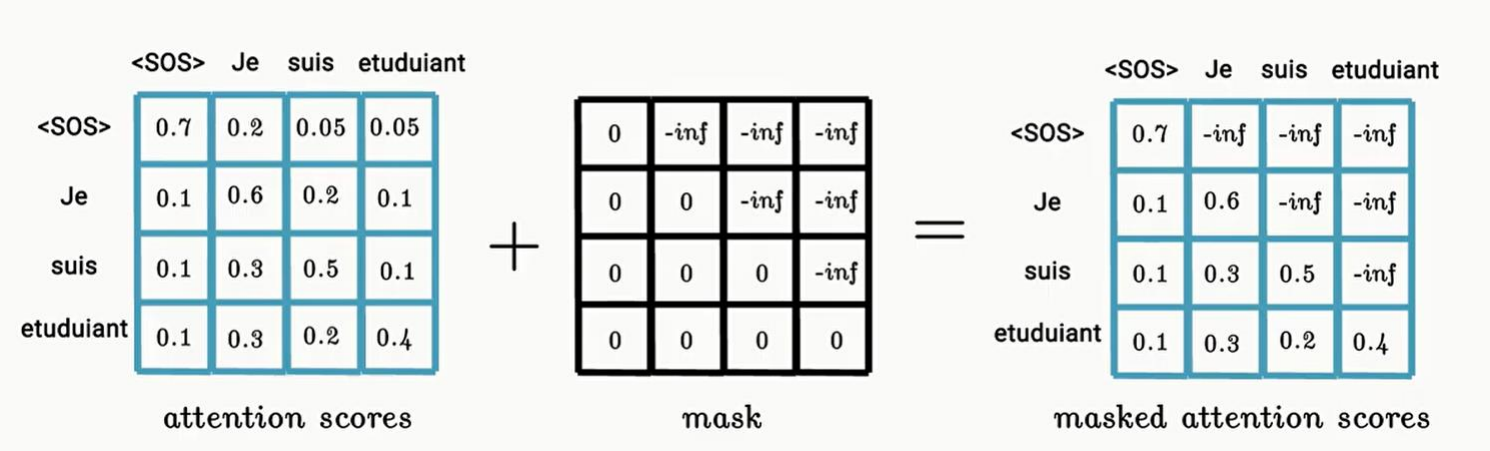
query Q key K and valid vectors V and then feed into the first multi-head attention layer this motorhead attention layer also does self-attention like we mentioned in the encoder layer by computing the attention scores for the decoder input but it does this with a slight twist since the decoder is auto regressive meaning that generates the output word by word it shouldn't be allowed to compute the attention scores on any future tokens



je suis etudiant

DECODER

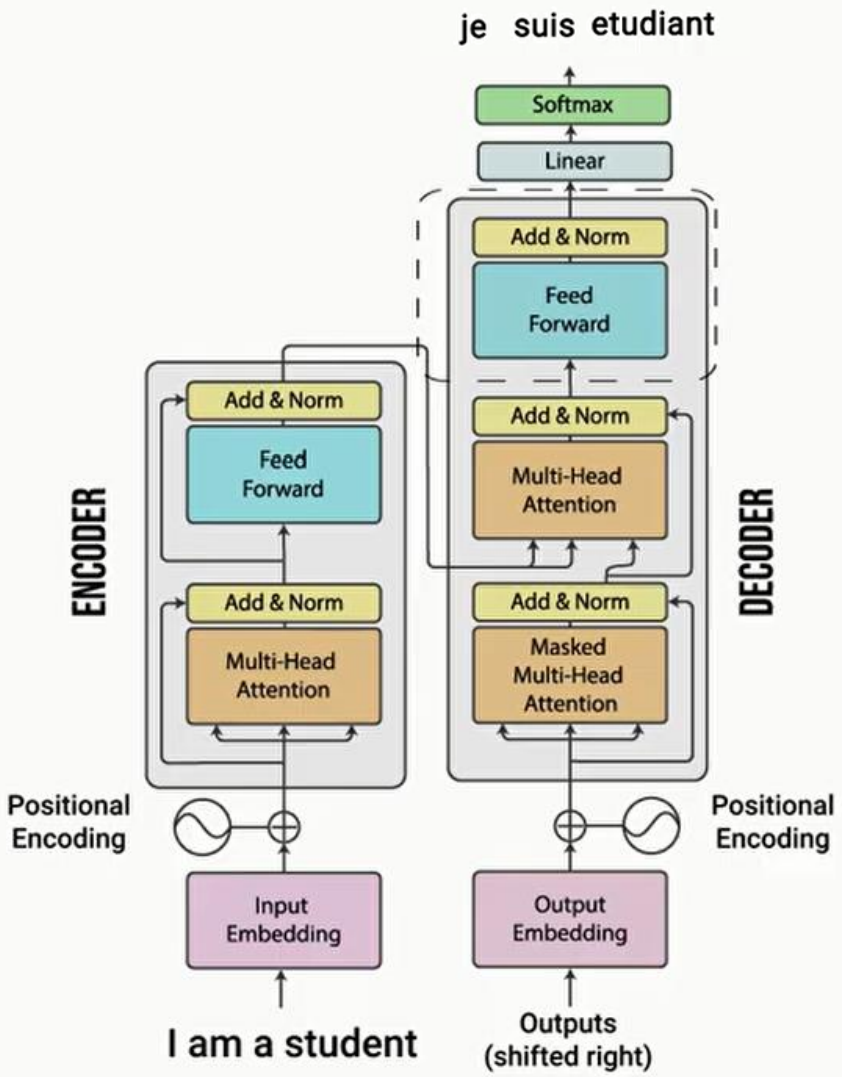
When computing attention scores for the word “suis” the scoring matrix should not have access to the word “etuduiant” because this is a future word to be generated next and instead it should only have access to itself and the words before it to achieve this the attention scores are masked by adding a matrix of zeros at infinities to the skilled scores before soft marks is applied and so this masking helps to disable all the elements above the diagonal of the attention score matrix and this is the only difference between the encoder and the decoder.



this is the only difference between the encoder and the decoder multi-head self-attention layers!!

all other steps concatenation linear transformation residual connection and layer normalization remain the same

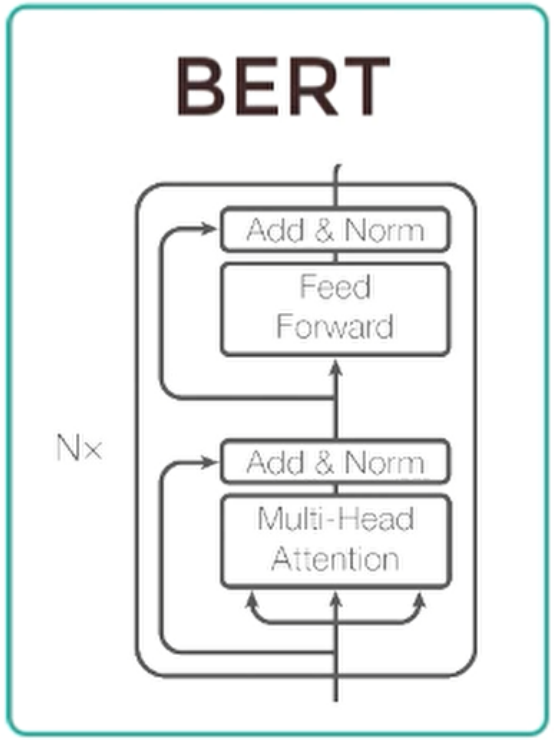
5. Decoder Layer



The output of this mask multi-header tension is then passed on to the second motor header tension layer to be used as its query vector while the encoder output serves as the key value vectors allowing the decoder to attend to appropriate places in the input sequence.

the final attention output vector is then passed to a fully connected feedforward network with residual connection and layer normalization and its output is then returned as the decoder output and just like in the encoding layer to boost its predictive power the decoder can be stacked n layers high with the output of each decoder bubbled up to the top decoder before being fed into a final decoding linear layer which acts as a classifier with a Softmax over the class vocabulary to return the predicted word and this coding process is continued till a special end of sequence token is reached

BERT



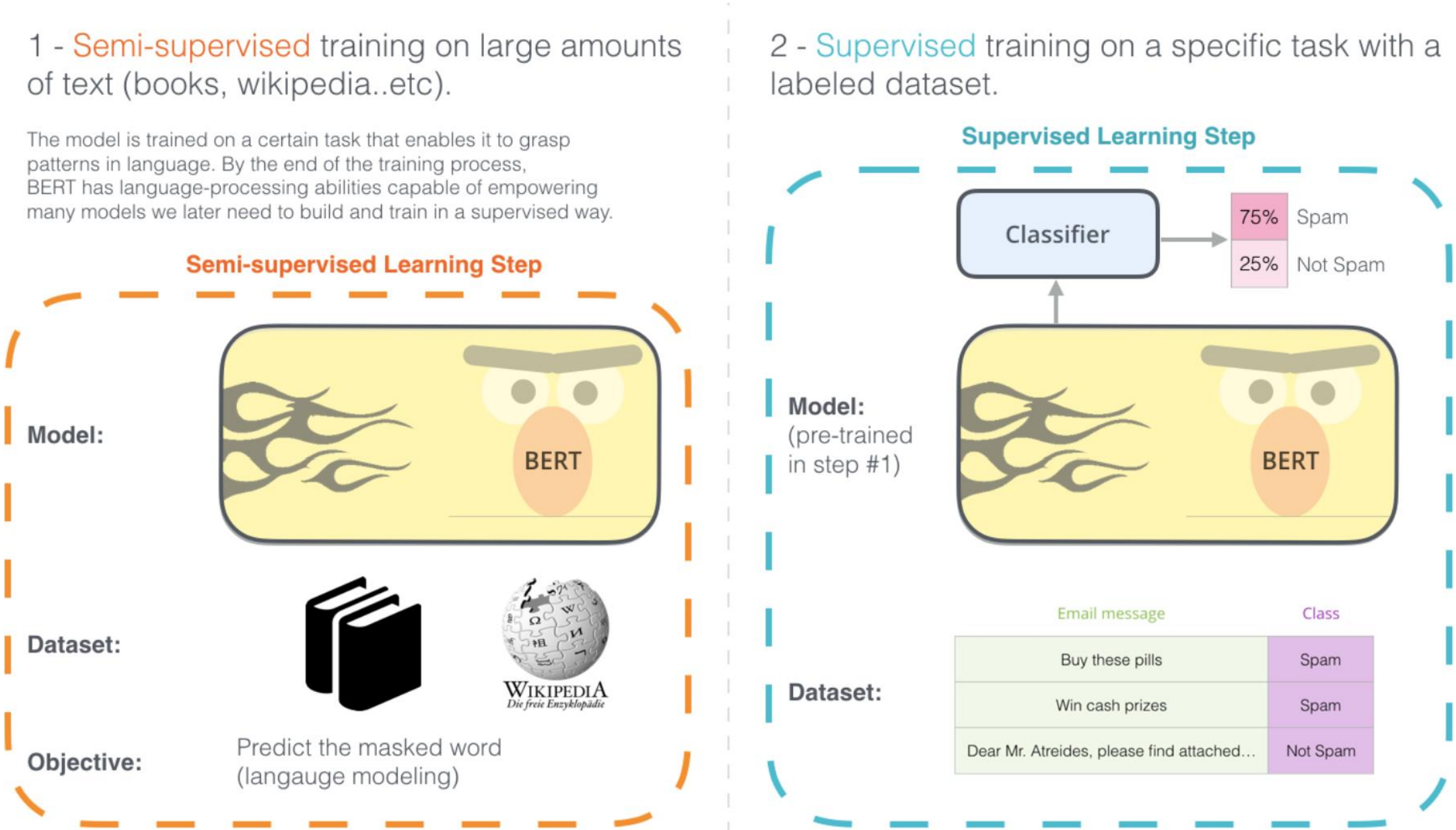
Problems to Solve

- Neural Machine Translation
- Question Answering
- Sentiment Analysis
- Text summarization

Needs Language understanding

How to solve Problems

- Pretrain BERT to understand langauge
- Fine tune BERT to learn specific task

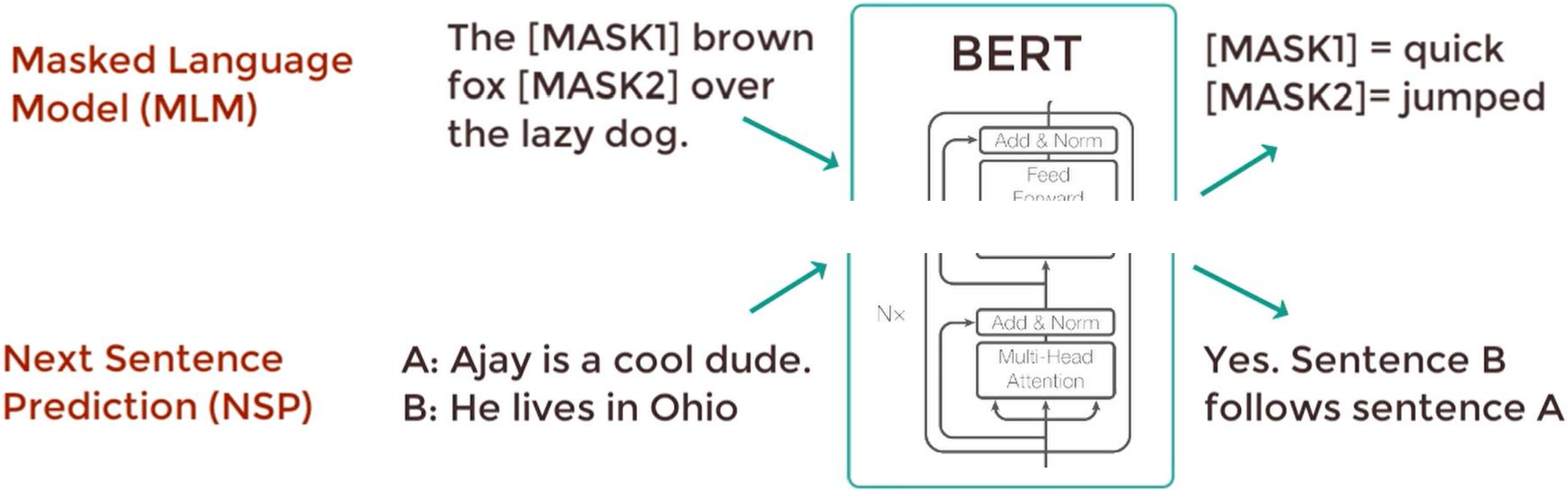


The two steps of how BERT is developed. You can download the model pre-trained in step 1 (trained on un-annotated data), and only worry about fine-tuning it for step 2. [Source for book icon].

BERT learns language by training on two unsupervised task simultaneously :
- Masked Language Modeling
- Next Sentence Prediction

Masked Language Modeling: Takes in sentence with random words fill with MASKS

The goal is to output these MASK token's

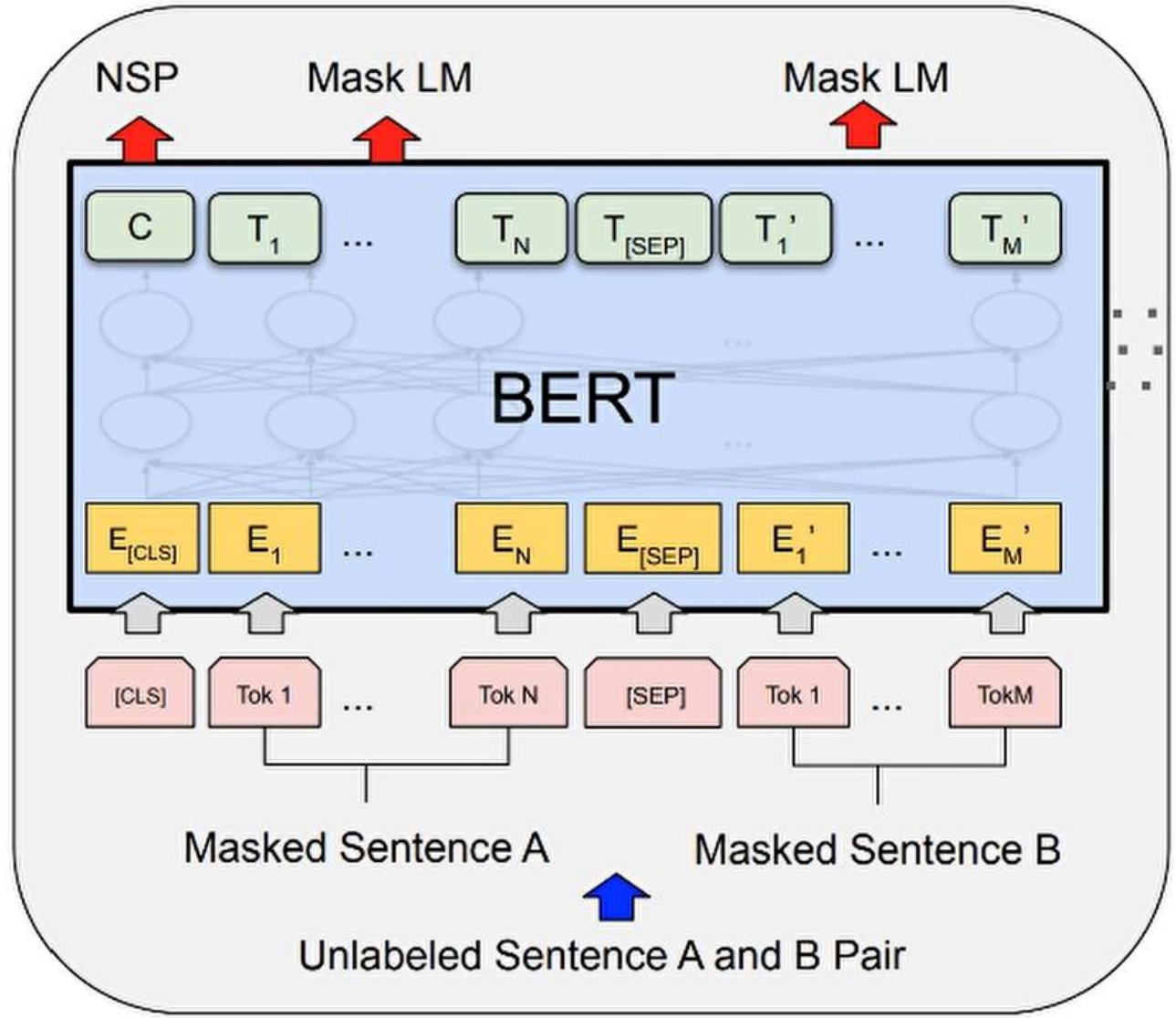


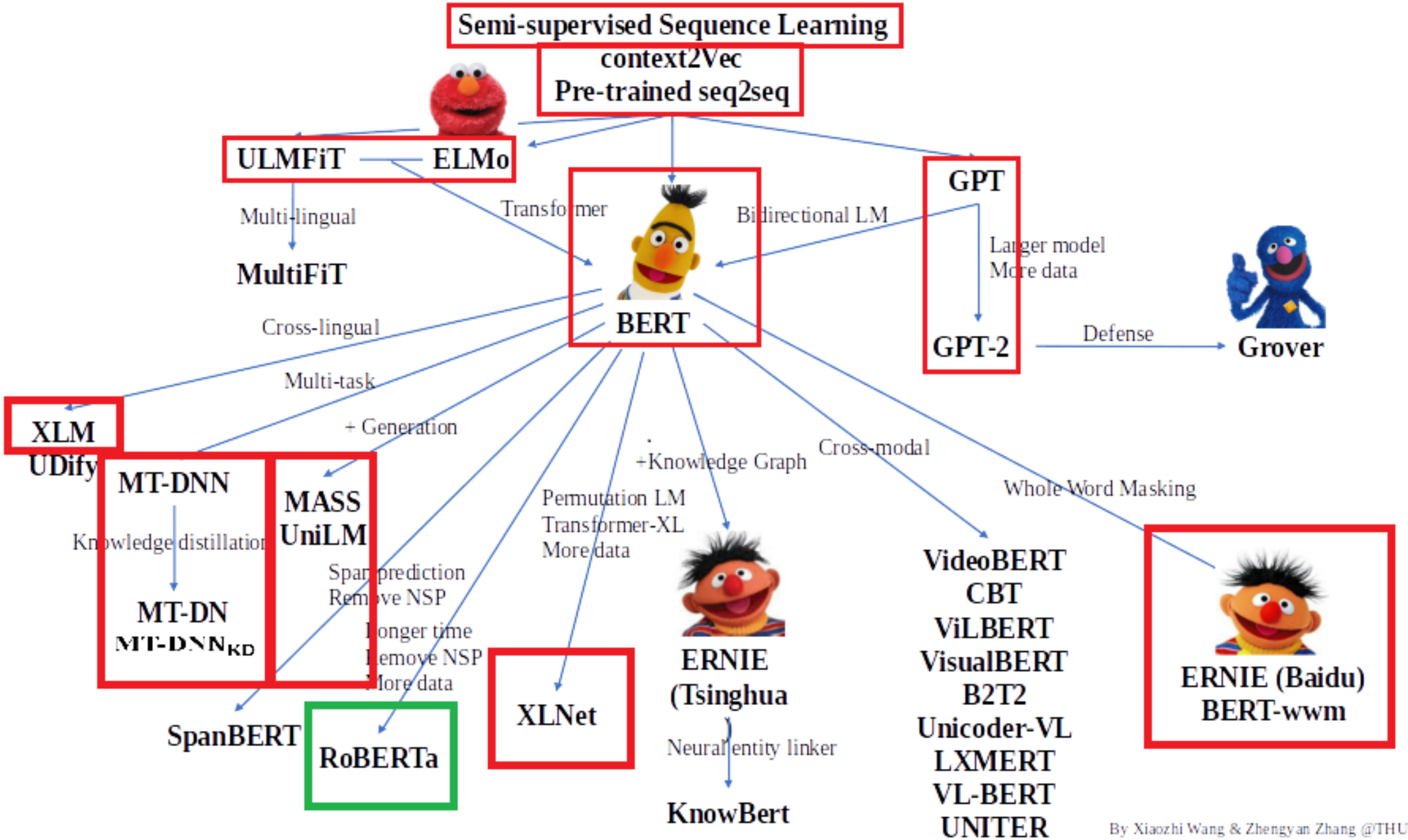
Pretraining (Pass 2)

Problems to train on simultaneously:

1. Masked Language Modeling (Mask LM)

2. Next Sentence Prediction (NSP)




Semi-supervised Sequence Learning
context2Vec
Pre-trained seq2seq
ULMFiT
ELMo
Multi-lingual
MultiFiT
Transformer
Bidirectional LM
BERT
GPT
Larger model
More data
GPT-2
Defense
Grover
Cross-lingual
Multi-task
XLM
UDify
+ Generation
MT-DNN
Knowledge distillation
MT-DN
MT-DNN_KD
MASS
UniLM
Span prediction
Remove NSP
Longer time
Remove NSP
More data
SpanBERT
RoBERTa
Permutation LM
Transformer-XL
More data
XLNet
+Knowledge Graph
ERNIE
(Tsinghua
Neural entity linker
KnowBert
Cross-modal
VideoBERT
CBT
ViLBERT
VisualBERT
B2T2
Unicoder-VL
LXMERT
VL-BERT
UNITER
Whole Word Masking
ERNIE (Baidu)
BERT-wwm
By Xiaozhi Wang & Zhengyan Zhang @THUNLP

What Is The T5 Transformer Model?

T5 was introduced in 2020 by the Google AI team and stands for Text-To-Text Transfer Transformer (5 Ts, or, in our case, T5). The main problem T5 addresses is the lack of systematic study comparing best practices in the field of NLP.

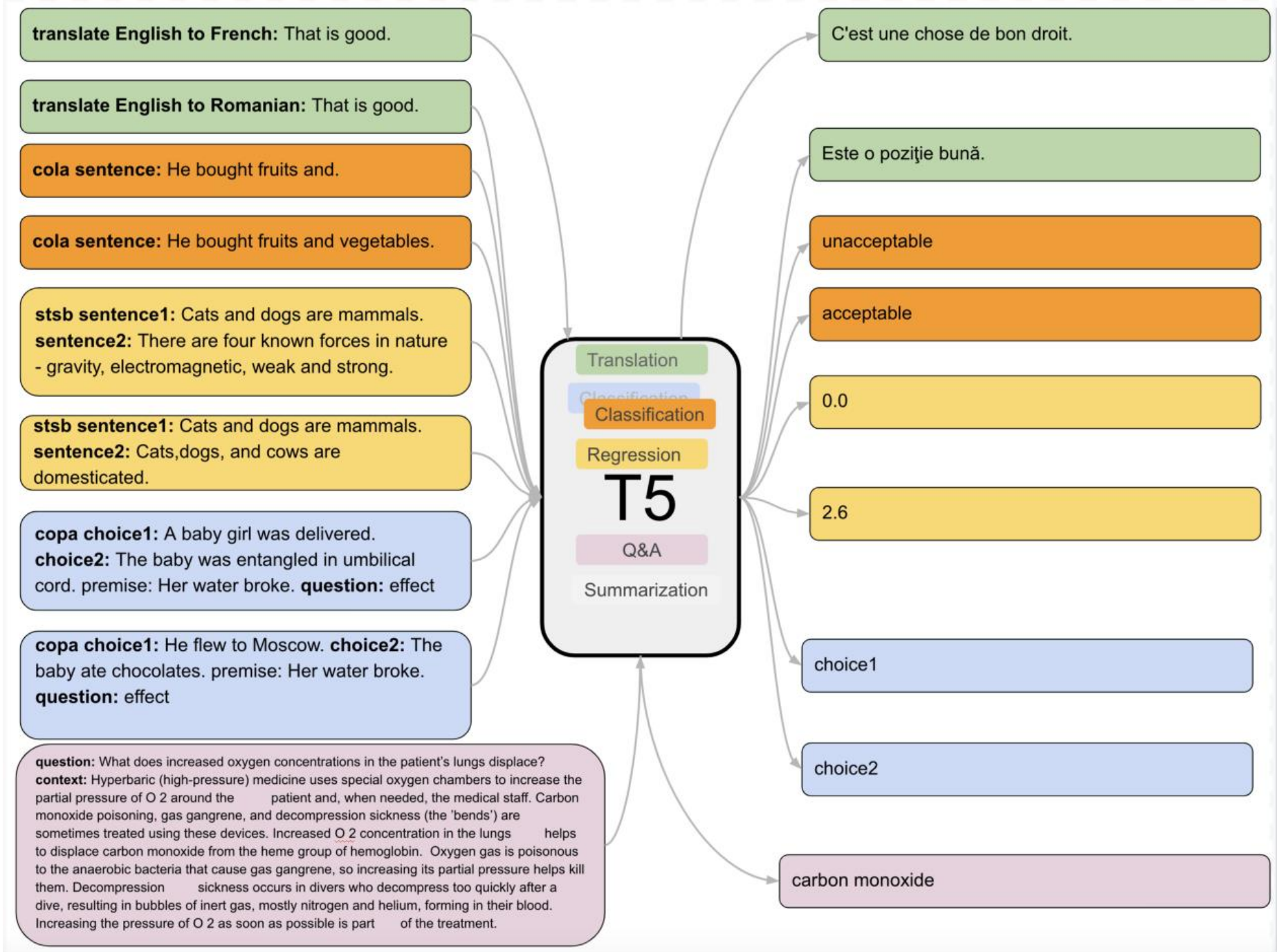
Most of the current SOTA models are derived from the Transformer architecture. The transformer was introduced in the legendary paper "Attention Is All You Need" by Vaswani et al., and had two main architectural blocks, namely Encoders and Decoders.

All the subsequent models had some sort of relation to these architectural blocks. Google's BERT only had Encoder blocks, OpenAI's GPT-2 only had Decoder blocks, etc.

With varying architectures, and various training datasets (Wikipedia, Wikipedia + Toronto Book Corpus), we cannot objectively compare these models and their SOTA results. After all, every model has varying pre-training objectives, unlabelled datasets, transfer approaches, and architectures.

T5 introduced the "Text-to-Text" framework, in which every NLP task (Translation, Classification, etc) has the same underlying structure in which text is fed as input to the model and text is produced as output. This means we can use the same model, same hyperparameters, and same loss function across all the tasks.

T5— Text-To-Text Transfer Transformer


translate English to French: That is good.
translate English to Romanian: That is good.
cola sentence: He bought fruits and.
cola sentence: He bought fruits and vegetables.
stsb sentence1: Cats and dogs are mammals. sentence2: There are four known forces in nature - gravity, electromagnetic, weak and strong.
stsb sentence1: Cats and dogs are mammals. sentence2: Cats,dogs, and cows are domesticated.
copa choice1: A baby girl was delivered. choice2: The baby was entangled in umbilical cord. premise: Her water broke. question: effect
copa choice1: He flew to Moscow. choice2: The baby ate chocolates. premise: Her water broke. question: effect
question: What does increased oxygen concentrations in the patient's lungs displace? context: Hyperbaric (high-pressure) medicine uses special oxygen chambers to increase the partial pressure of O 2 around the patient and, when needed, the medical staff. Carbon monoxide poisoning, gas gangrene, and decompression sickness (the 'bends') are sometimes treated using these devices. Increased O 2 concentration in the lungs helps to displace carbon monoxide from the heme group of hemoglobin. Oxygen gas is poisonous to the anaerobic bacteria that cause gas gangrene, so increasing its partial pressure helps kill them. Decompression sickness occurs in divers who decompress too quickly after a dive, resulting in bubbles of inert gas, mostly nitrogen and helium, forming in their blood. Increasing the pressure of O 2 as soon as possible is part of the treatment.
Translation
Classification
Regression
T5
Q&A
Summarization
C'est une chose de bon droit.
Este o poziţie bună.
unacceptable
acceptable
0.0
2.6
choice1
choice2
carbon monoxide