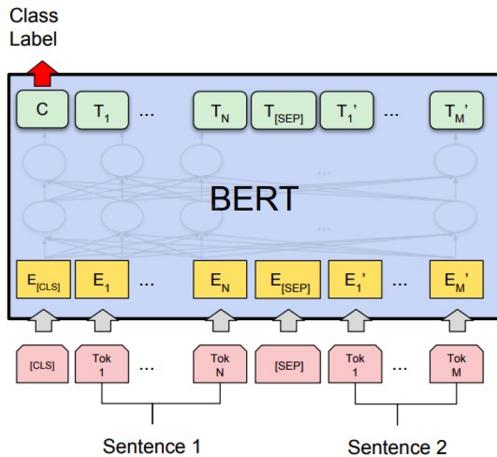


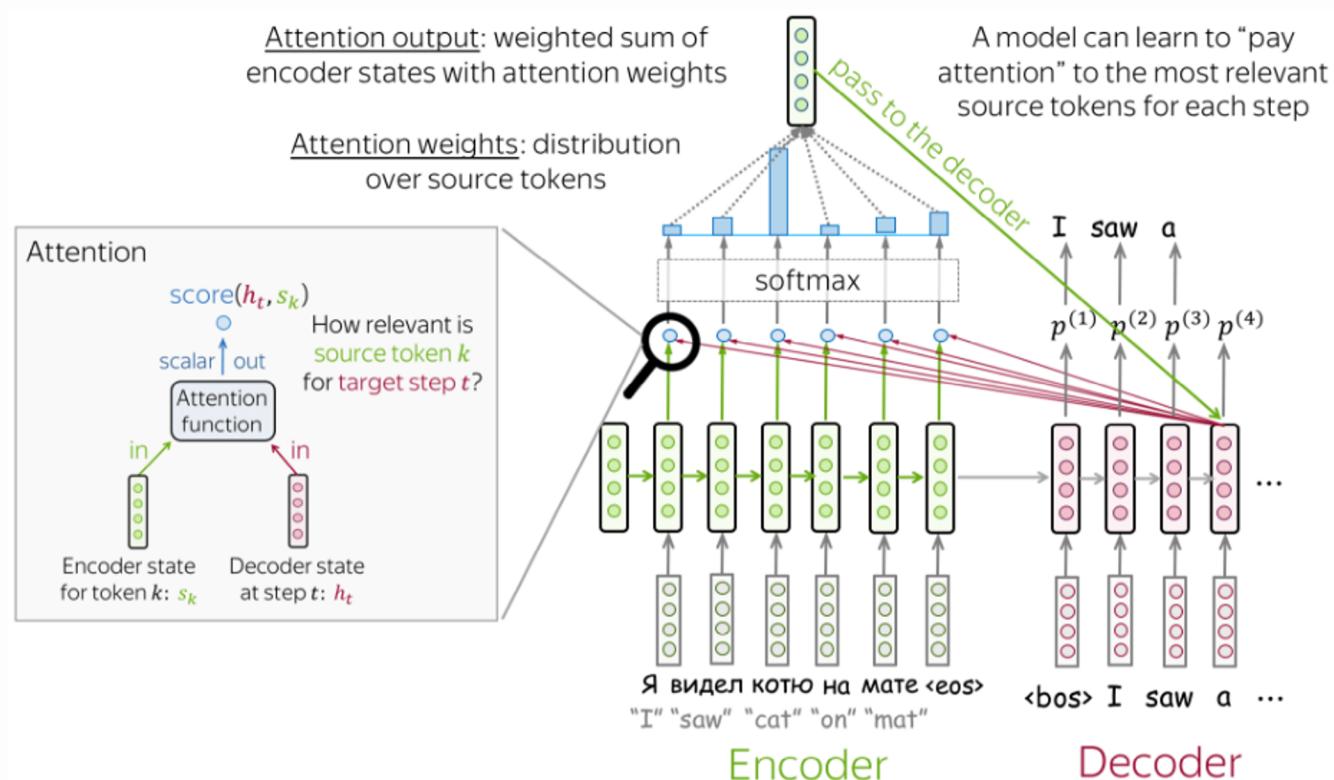
# Seminar 9: transformers

Actually, AI does not exist. It is just ML.

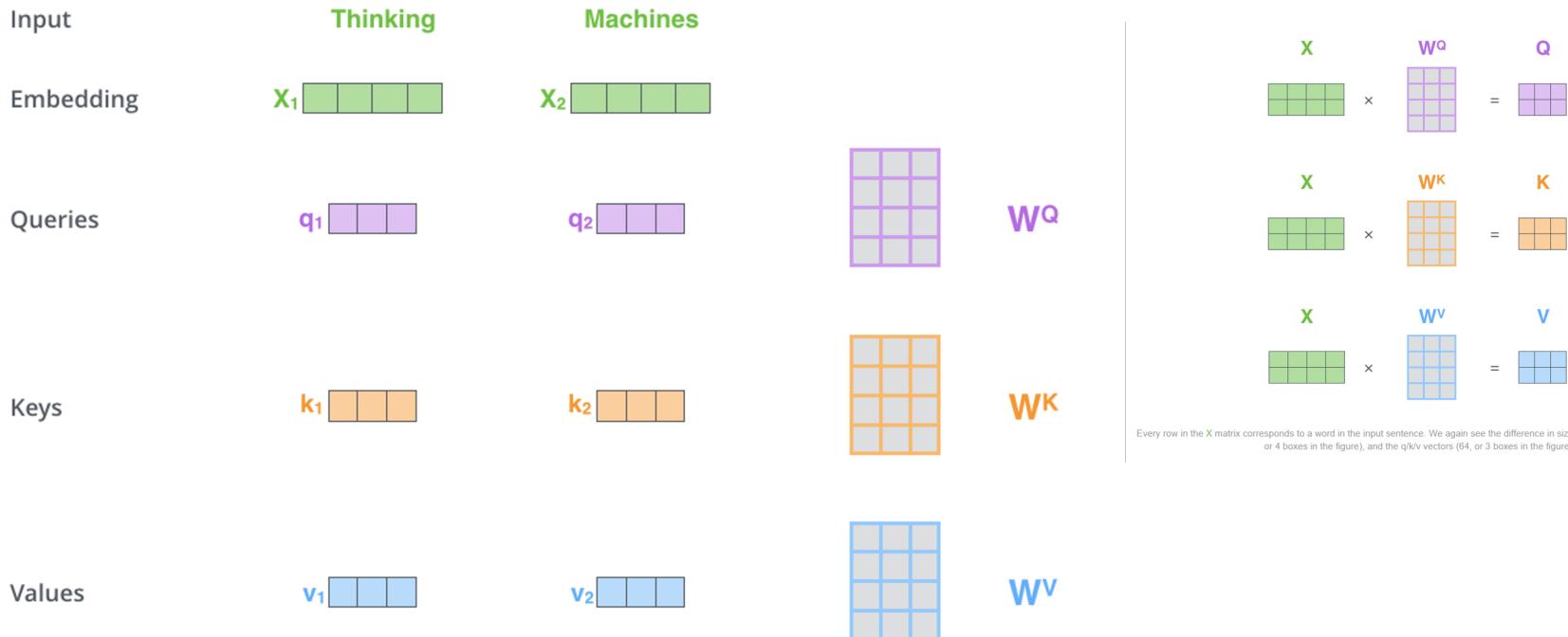


99% toxic comment

# Attention



# Self-Attention



Multiplying  $X_1$  by the  $W^Q$  weight matrix produces  $q_1$ , the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

# Self-Attention

$$\begin{matrix} \textcolor{green}{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \textcolor{purple}{W^Q} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \textcolor{purple}{Q} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} \textcolor{green}{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \textcolor{orange}{W^K} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \textcolor{orange}{K} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} \textcolor{green}{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \textcolor{blue}{W^V} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \textcolor{blue}{V} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

Every row in the  $\textcolor{green}{X}$  matrix corresponds to a word in the input sentence. We again see the difference in size of the embedding vector (512, or 4 boxes in the figure), and the q/k/v vectors (64, or 3 boxes in the figure)

# Self-Attention

$$\text{softmax} \left( \frac{\begin{matrix} Q \\ \times \\ K^T \end{matrix}}{\sqrt{d_k}} \right) = Z$$

The diagram illustrates the self-attention calculation in matrix form. It shows the multiplication of the Query matrix (Q) and the transpose of the Key matrix (K<sup>T</sup>) divided by the square root of the dimension d<sub>k</sub>. The resulting matrix is then passed through a softmax function to produce the attention weights matrix Z.

The diagram shows three 2x3 matrices labeled Q, K<sup>T</sup>, and V. Matrix Q is purple, K<sup>T</sup> is orange, and V is blue. Below them is a large bracket containing the softmax function and the multiplication operation. A horizontal line connects the multiplication operation to the division by  $\sqrt{d_k}$ . Below this, an equals sign is followed by a pink matrix labeled Z, which has the same dimensions as Q and K<sup>T</sup>.

The self-attention calculation in matrix form

# Multi-Head-Attention

1) This is our input sentence\*

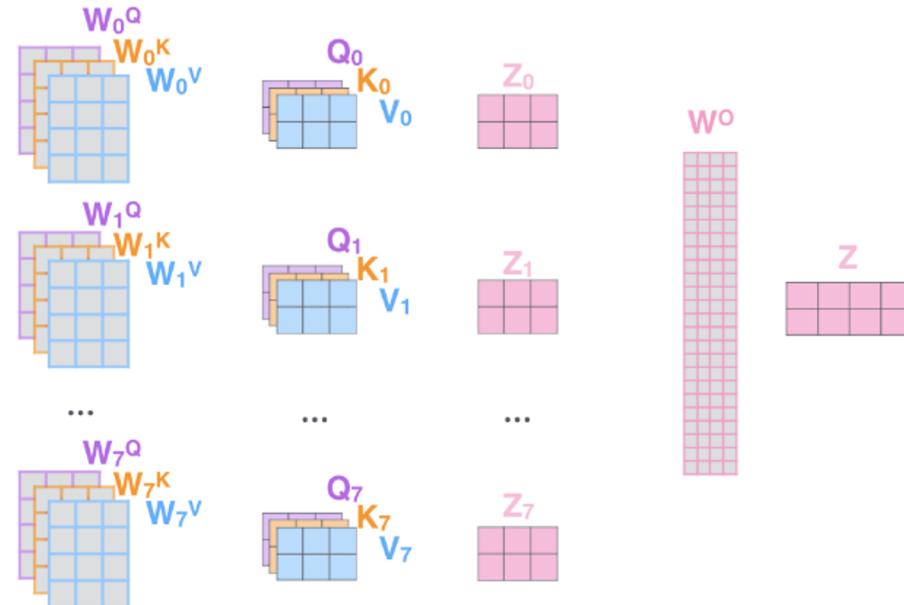
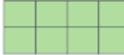
2) We embed each word\*

3) Split into 8 heads.  
We multiply  $X$  or  $R$  with weight matrices

4) Calculate attention using the resulting  $Q/K/V$  matrices

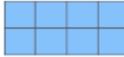
5) Concatenate the resulting  $Z$  matrices, then multiply with weight matrix  $W^o$  to produce the output of the layer

Thinking  
Machines

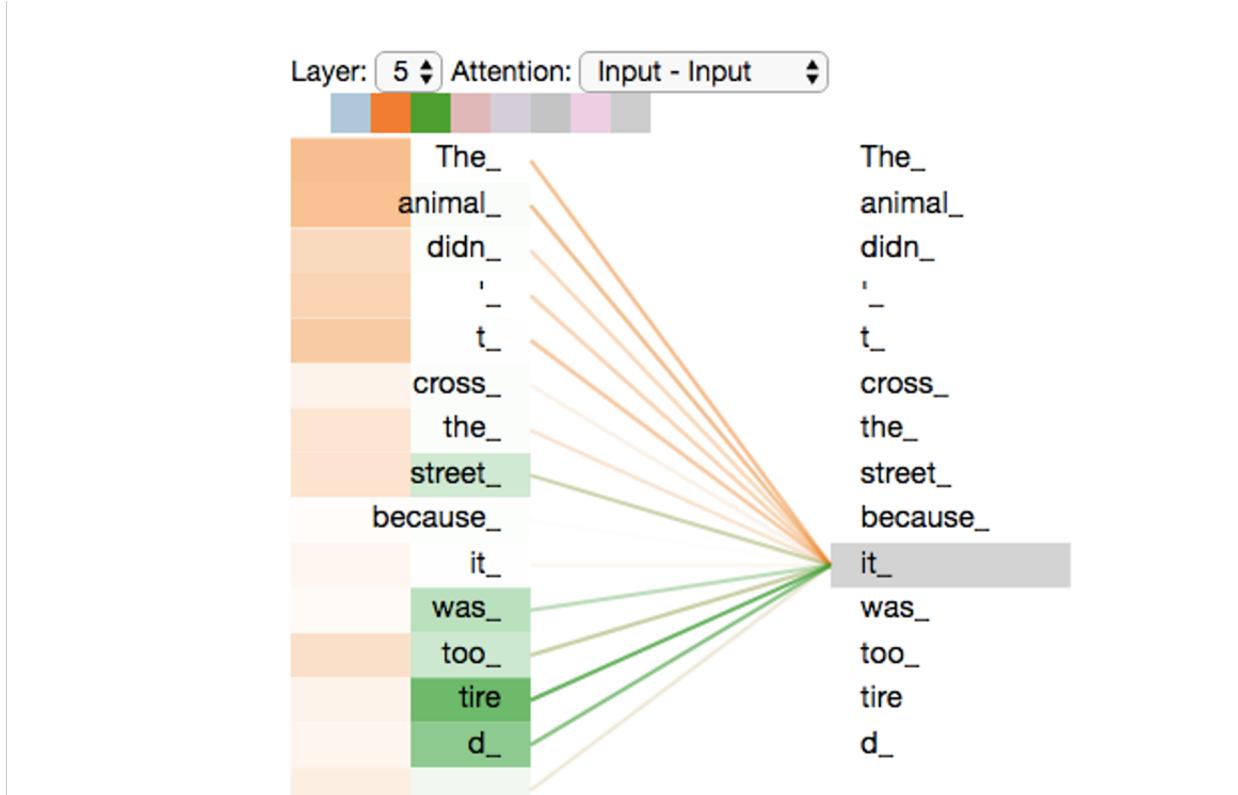


\* In all encoders other than #0, we don't need embedding.  
We start directly with the output of the encoder right below this one

$R$



# Self-Attention



# Transformer

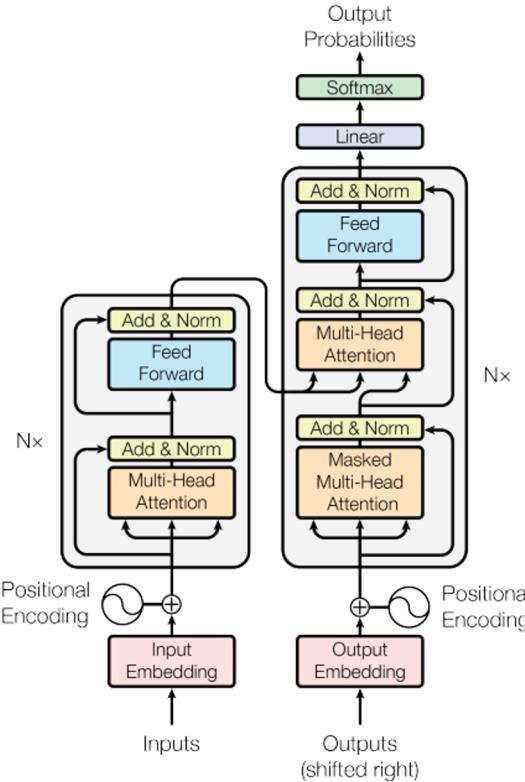
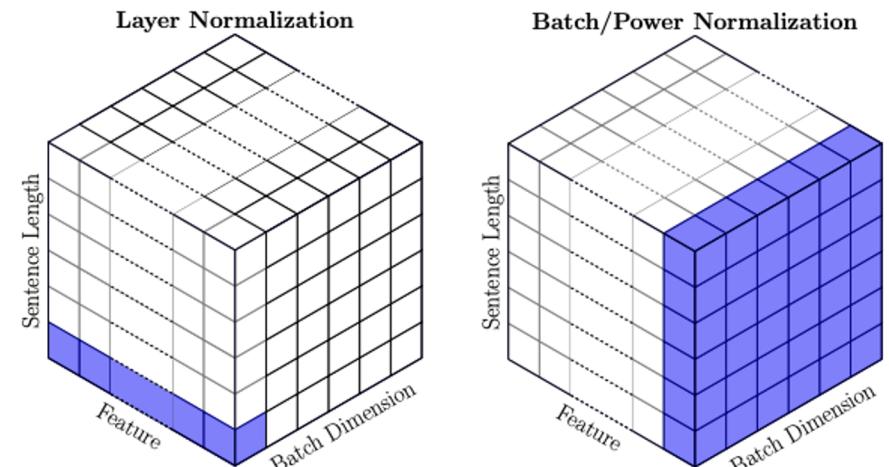
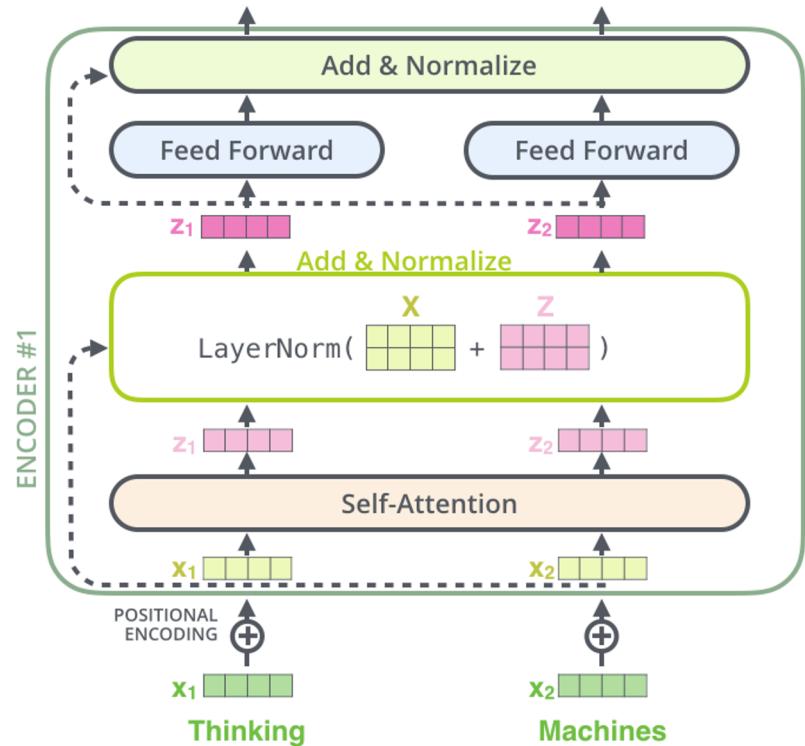
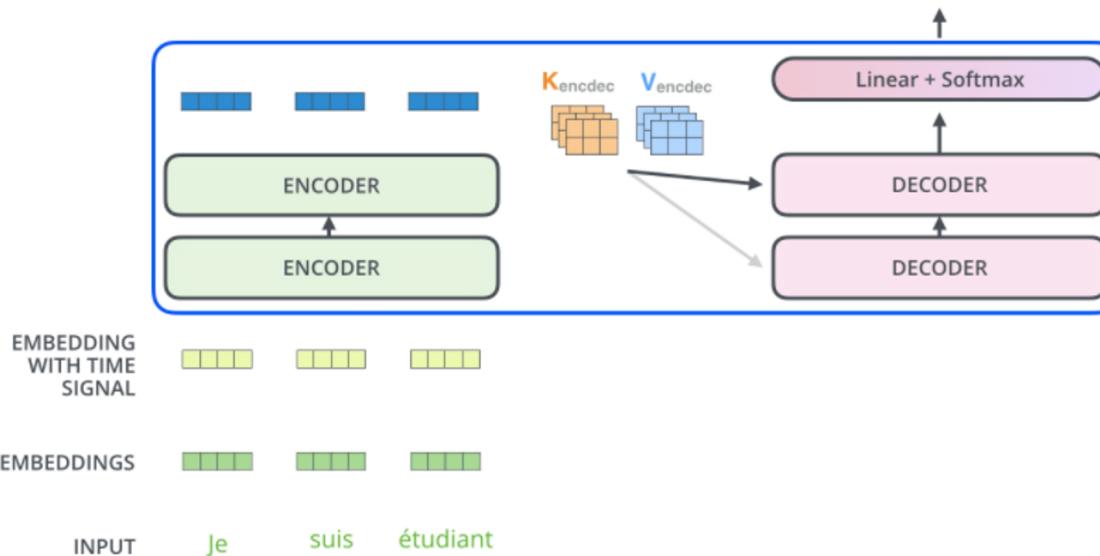


Figure 1: The Transformer - model architecture.

# Transformer (encoder block)

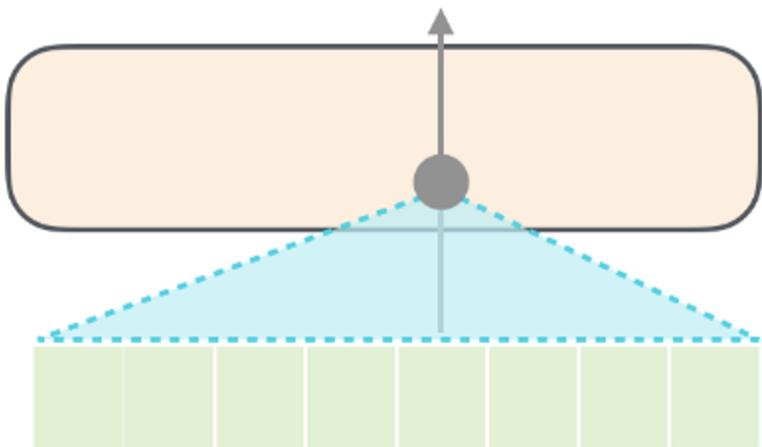


# Transformer (Decoder block)

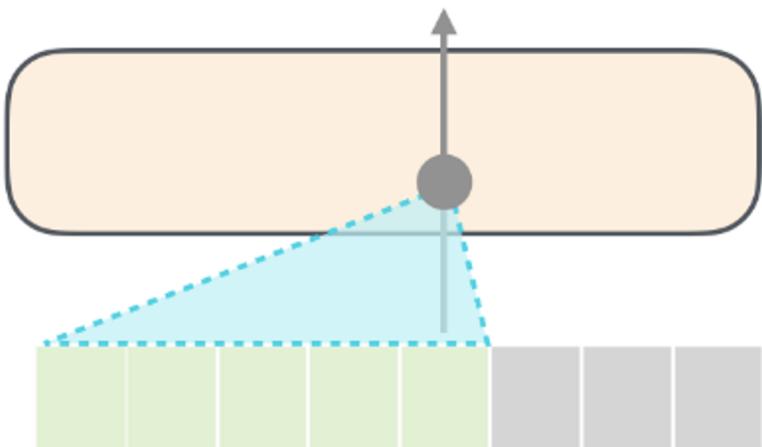


# Self-Attention

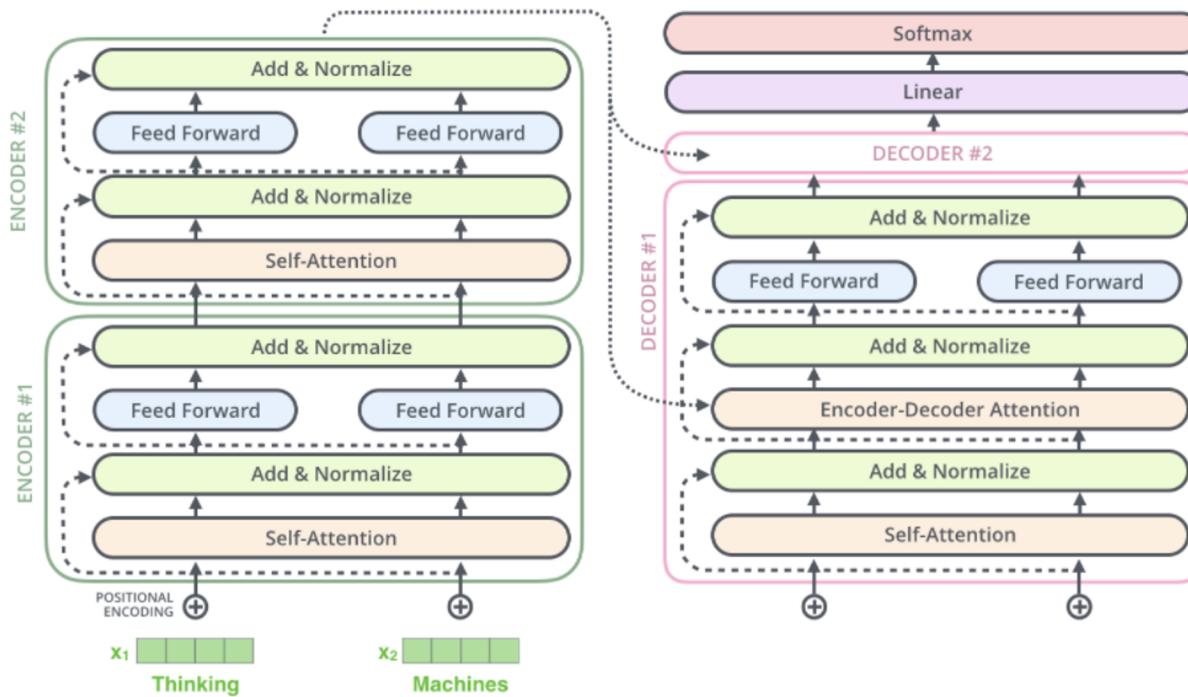
**Self-Attention**



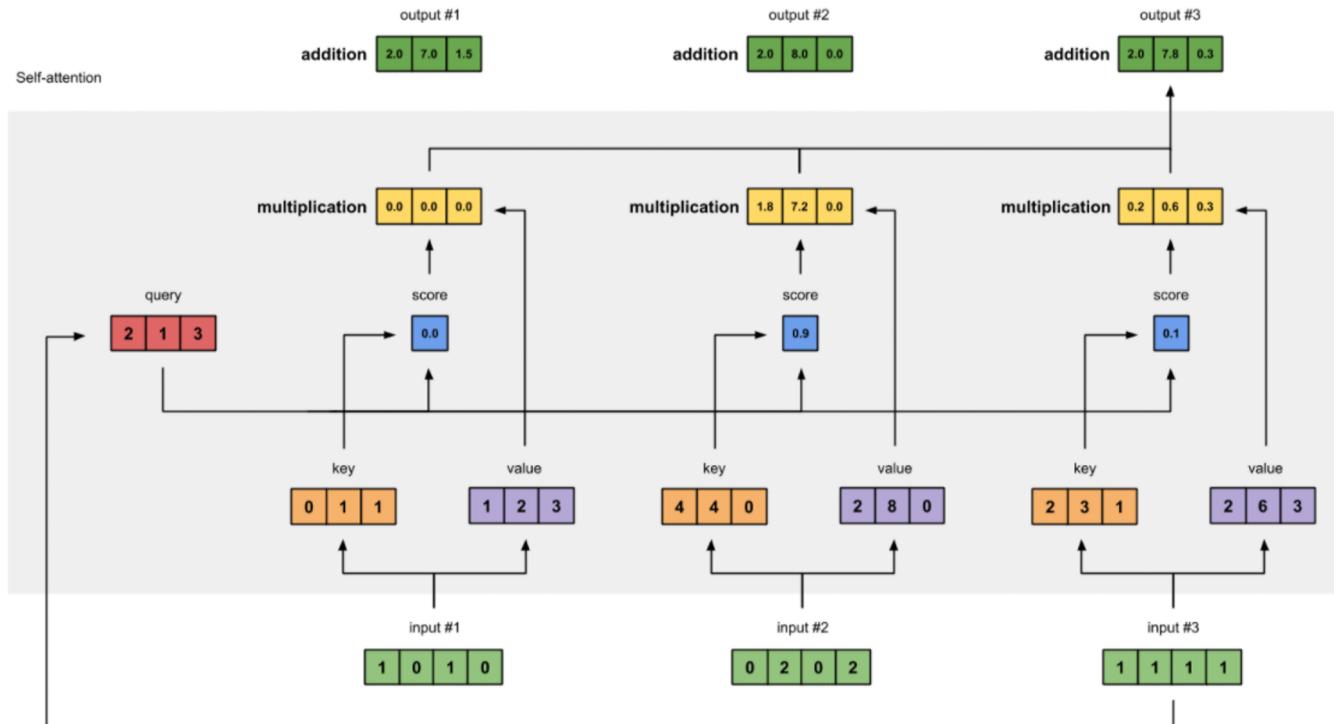
**Masked Self-Attention**



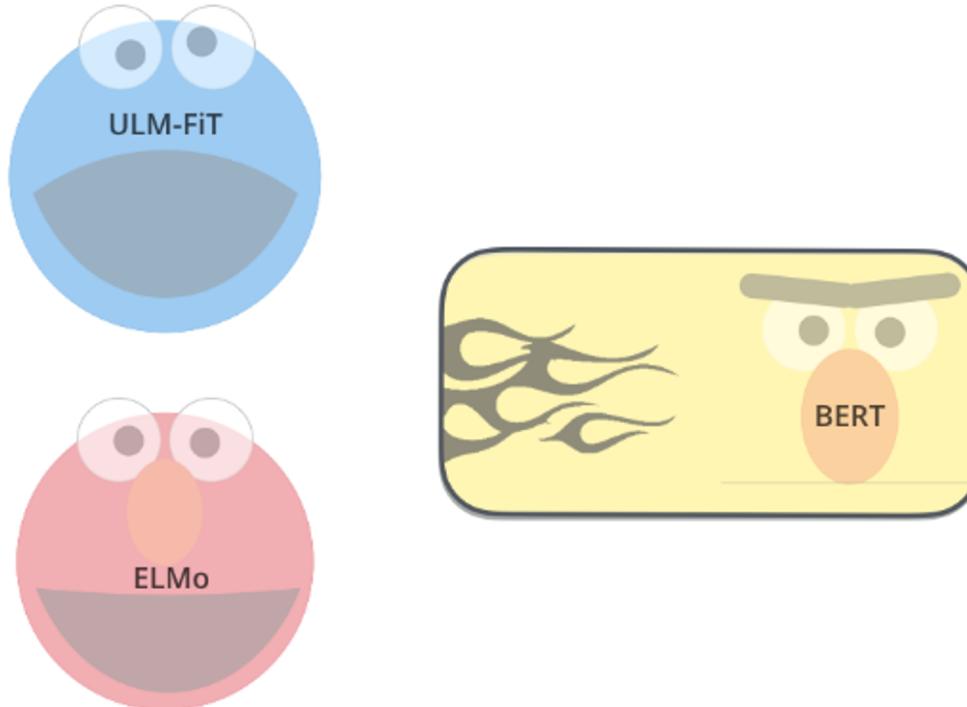
# Transformer (Full)



# Self-Attention



# Better Embeddings



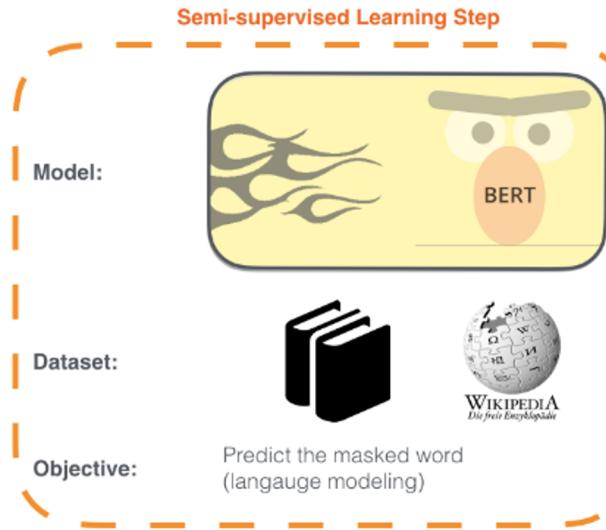
<https://jalammar.github.io/illustrated-bert/>

# Bert

Bidirectional Encoder Representations from Transformers

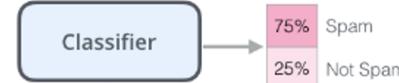
## 1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.



## 2 - Supervised training on a specific task with a labeled dataset.

### Supervised Learning Step



Model:  
(pre-trained  
in step #1)

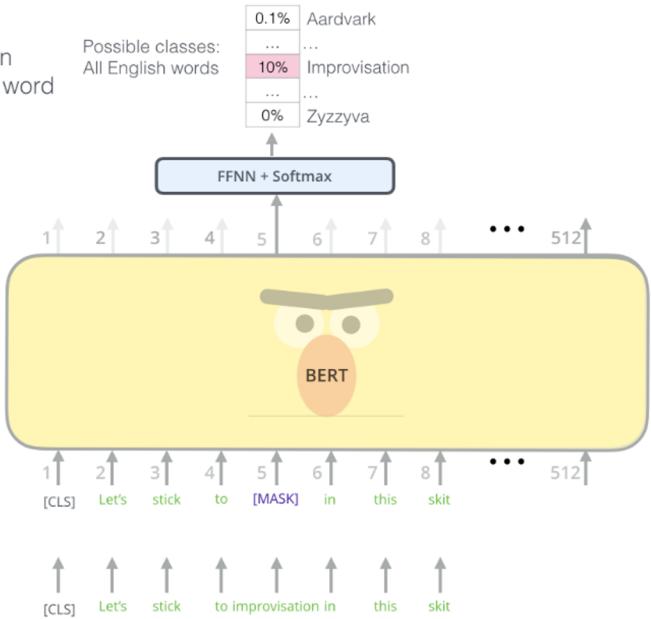


Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached...	Not Spam

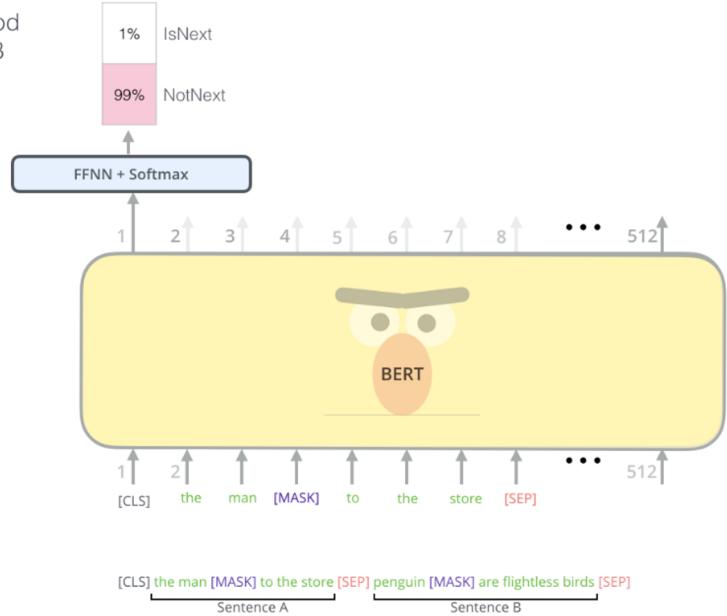
The two steps of how BERT is developed. You can download the model pre-trained in step 1 (trained on un-annotated data), and only worry about fine-tuning it for step 2. [Source for book icon].

# Bert

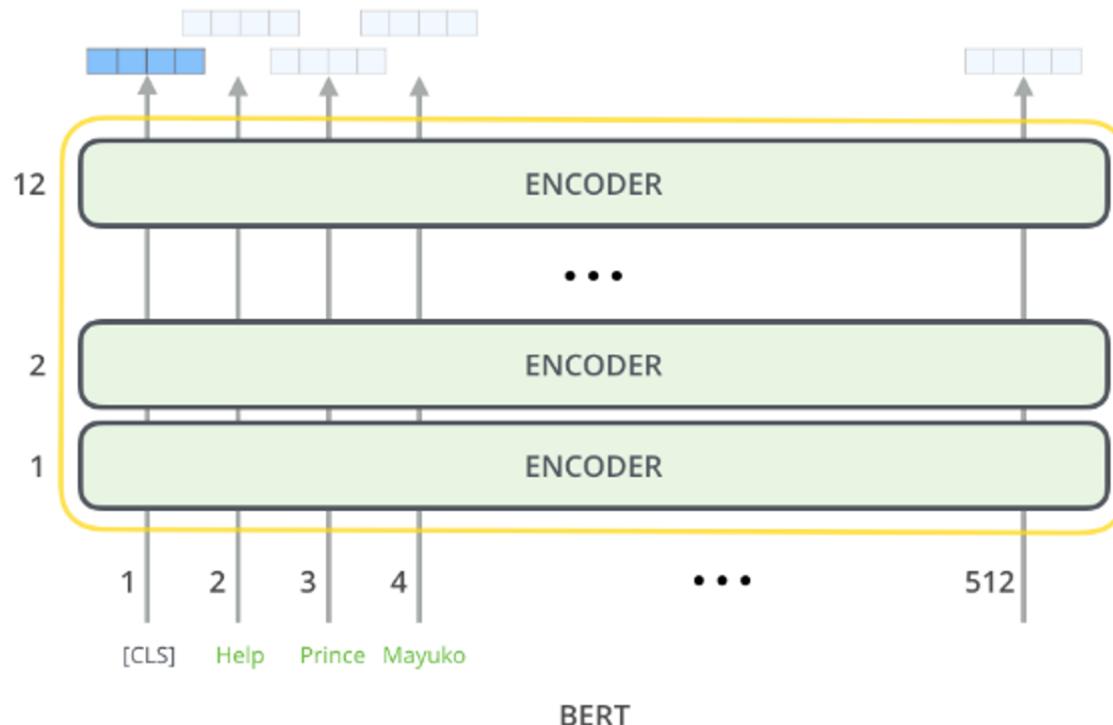
Use the output of the masked word's position to predict the masked word



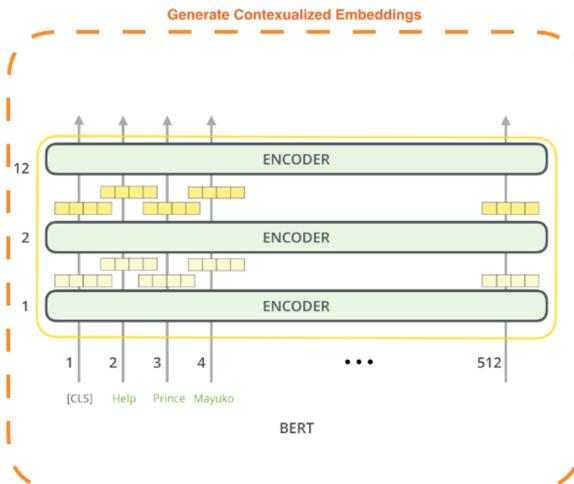
Predict likelihood that sentence B belongs after sentence A



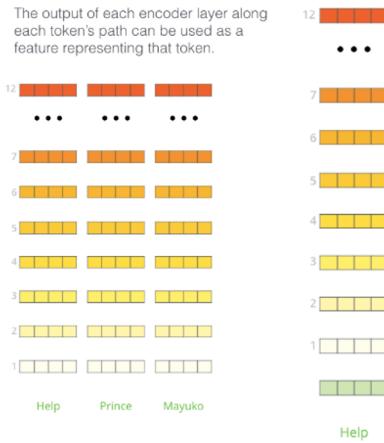
# Bert



# Bert



The output of each encoder layer along each token's path can be used as a feature representing that token.



But which one should we use?

What is the best contextualized embedding for “Help” in that context?  
For named-entity recognition task CoNLL-2003 NER

	Dev F1 Score
First Layer	91.0
Last Hidden Layer	94.9
Sum All 12 Layers	95.5
Second-to-Last Hidden Layer	95.6
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1

# GPT-3

Input Prompt:

Recite the first law of robotics



GPT-3

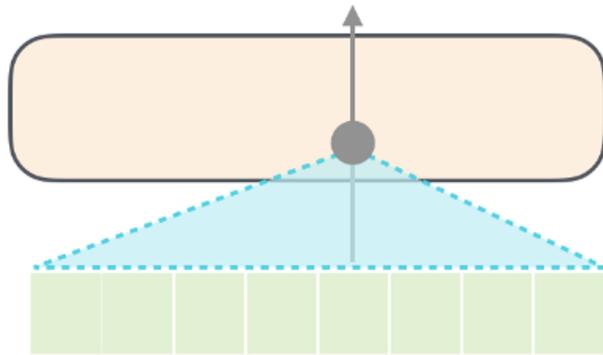


Output:

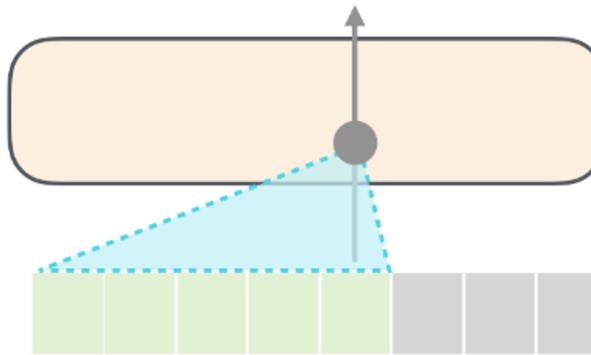
A robot may not injure a human  
being or, through inaction,  
allow a human being to  
come to harm.

# GPT-3

**Self-Attention**



**Masked Self-Attention**



# GPT-3

