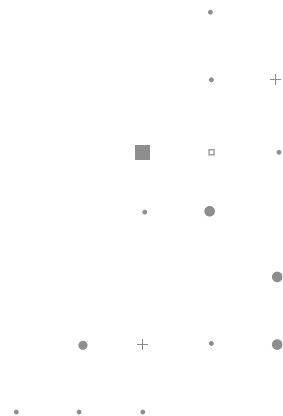




FIAP



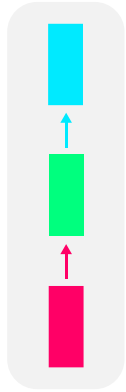


MACHINE LEARNING E DEEP LEARNING

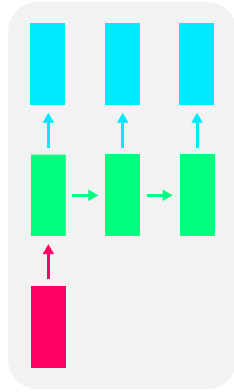


REDES NEURAIS RECORRENTES - SEQUÊNCIA DE PROCESSOS

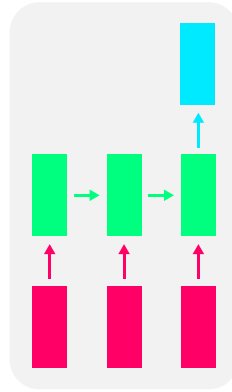
One to one



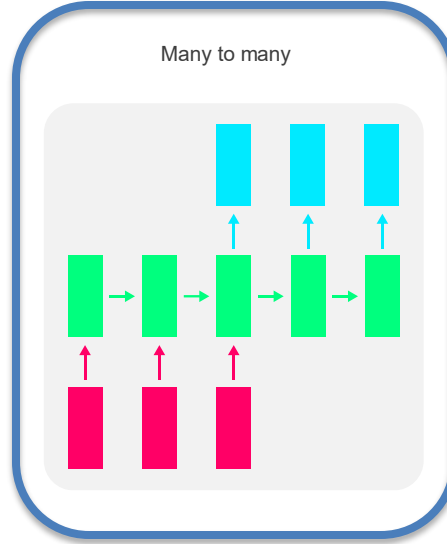
One to many



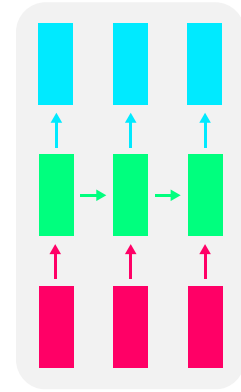
Many to one



Many to many



Many to many



- Muitos para muitos - por exemplo, Tradução por Máquina;
- Sequência de palavras -> sequência de palavras.

COMO FUNCIONA UM LLM?

Language modeling

Imagine the following task: Predict the next word in a sequence

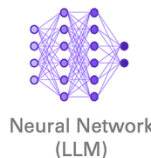
[The cat likes to sleep in the _____] → What word comes next?

Can we frame this as a ML problem? Yes, it's a classification task.

Now we have (say)
~50,000 classes (i.e.
words)

[The cat likes to sleep in the]

Input



Neural Network
(LLM)

Word	Probability
ability	0.002
bag	0.071
box	0.085
...	...
zebra	0.001

Output

COMO FUNCIONA UM LLM?

Massive training data

We can create **vast amounts of sequences** for training a language model

● Context ● Next Word ● Ignored

[The cat likes to sleep in the]
[The cat likes to sleep in the]
[The cat likes to sleep in the]
[The cat likes to sleep in the]
[The cat likes to sleep in the]

We do the same with much **longer sequences**. For example:

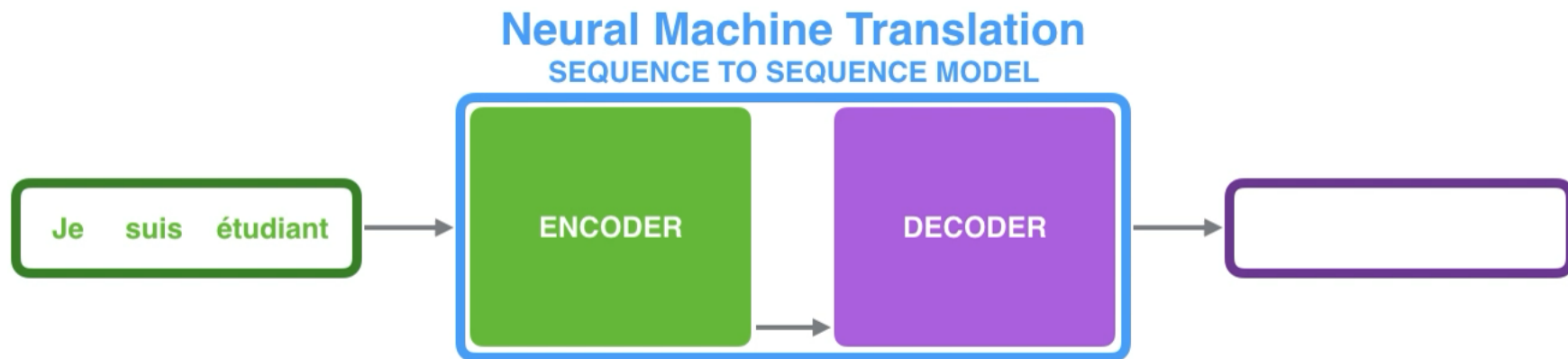
A language model is a probability distribution over sequences of words. [...] Given any sequence of words, the model predicts the next ...

Or also with **code**:

```
def square(number):  
    """Calculates the square of a number."""  
    return number ** 2
```

And as a result - the model becomes incredibly good at **predicting the next word** in any sequence.

REDES NEURAIS RECORRENTES - SEQUÊNCIA DE PROCESSOS (seq2seq)



Fonte: Jay Alammar - Visualizing machine learning one concept at a time (2018).

REDES NEURAIS RECORRENTES – SEQ2SEQ

Word Embedding?

Input

Je
suis
étudiant

0.901	-0.651	-0.194	-0.822
-0.351	0.123	0.435	-0.200
0.081	0.458	-0.400	0.480



REDES NEURAIS RECORRENTES – SEQ2SEQ

Recurrent Neural Network

Time step #1:

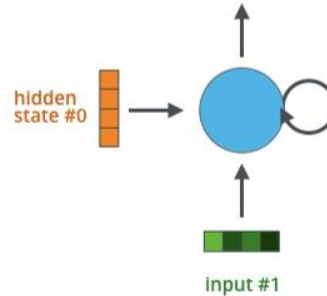
An RNN takes two input vectors:



hidden state #0

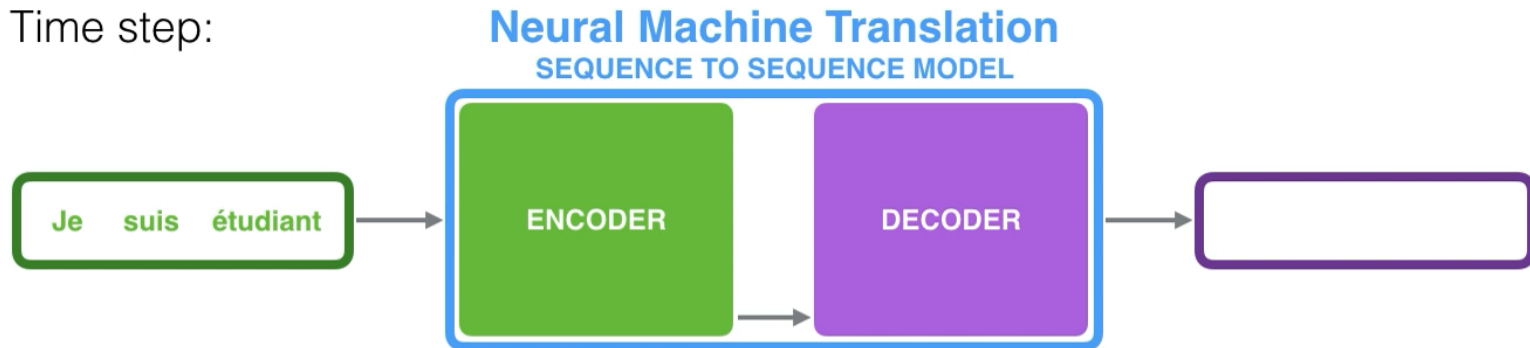


input vector #1



REDES NEURAIS RECORRENTES – SEQ2SEQ

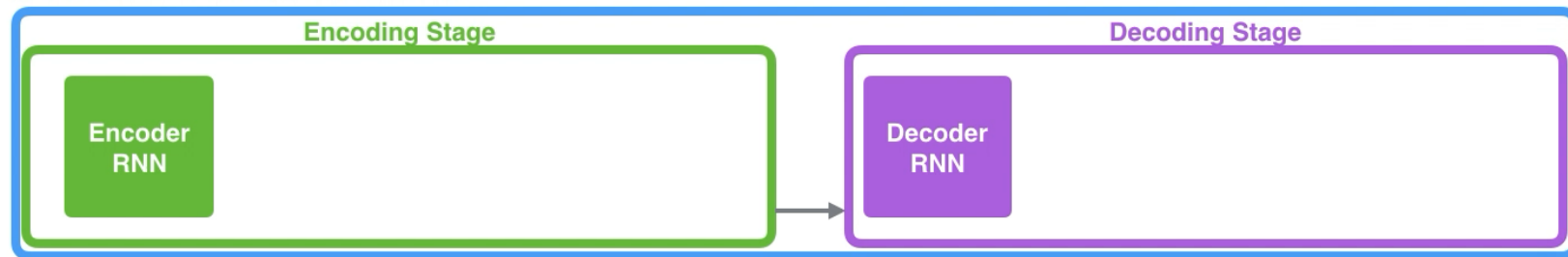
Time step:



REDES NEURAIS RECORRENTES – SEQ2SEQ

Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL

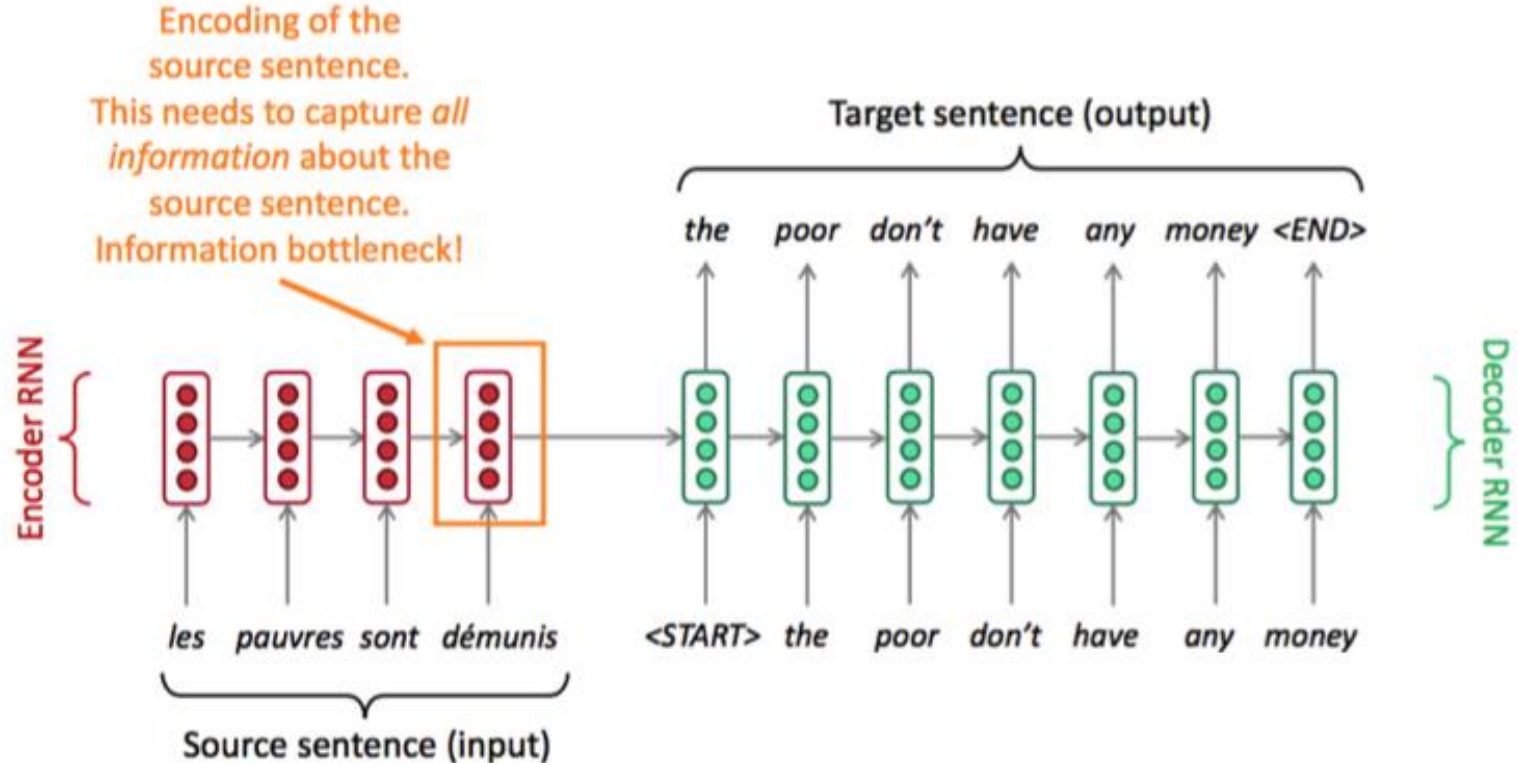


Je

suis

étudiant

REDES NEURAIS RECORRENTES – SEQ2SEQ



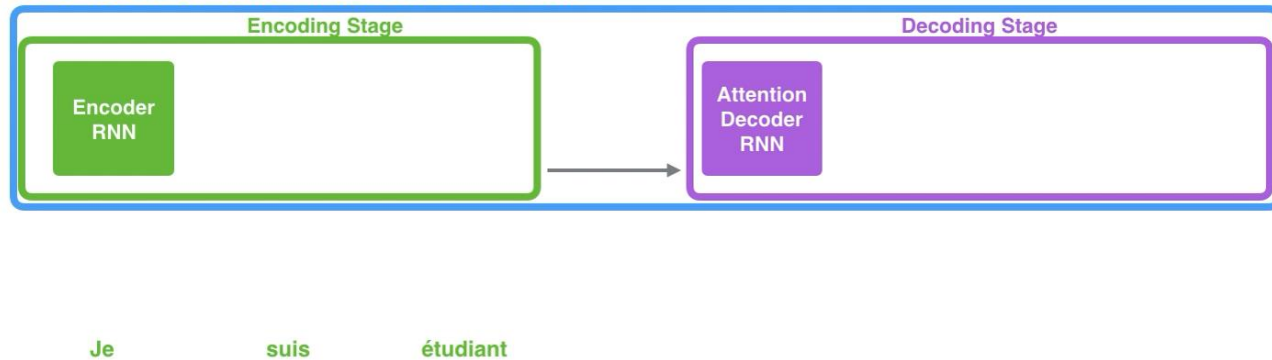
REDES NEURAIS RECORRENTES – SEQ2SEQ

O **vetor de contexto** acabou se tornando um gargalo para esse tipo de modelo. Isso dificultava para os modelos lidarem com sentenças longas. Uma solução foi proposta por Bahdanau et al., em 2014, e Luong et al., em 2015. Esses artigos introduziram e refinaram uma técnica chamada “**Atenção**”, que melhorou significativamente a qualidade dos sistemas de tradução automática. A atenção permite que o modelo foque nas partes relevantes da sequência de entrada conforme necessário.

REDES NEURAIS RECORRENTES – SEQ2SEQ COM ATTENTION

Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL WITH ATTENTION



Primeiro, o **encoder** passa muito mais dados para o **decoder**. Em vez de passar apenas o último estado oculto da etapa de codificação, o **encoder** passa **todos os estados ocultos** (**hidden states**) para o **decoder**.

Attention at time step 4



REDES NEURAIS RECORRENTES – SEQ2SEQ COM ATTENTION

Em **segundo** lugar, um **decoder** com atenção realiza uma etapa extra antes de produzir sua saída. Para focar nas partes da entrada que são relevantes para o passo atual de decodificação, o **decoder** faz o seguinte:

- Observa o conjunto de **hidden states** (estados ocultos) do **encoder** que recebeu – cada estado oculto do encoder está mais associado a uma determinada palavra na sentença de entrada.
- Atribui uma pontuação para cada **hidden state** (vamos ignorar, por enquanto, como essa pontuação é calculada).
- Multiplica cada estado oculto por sua pontuação após aplicar softmax, amplificando os **hidden states** com pontuações altas e suprimindo os **hidden states** com pontuações baixas.

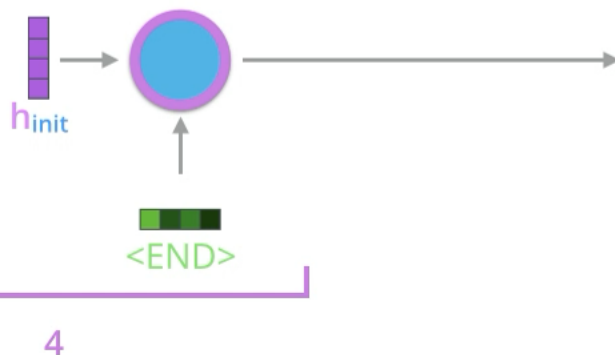
Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL WITH ATTENTION

Encoding Stage



Attention Decoding Stage



REDES NEURAIS RECORRENTES – SEQ2SEQ COM ATTENTION

Terceiro Passo:

A RNN com **atenção** do **decoder** recebe o **embedding** do token **<END>** e um estado oculto inicial do **decoder**.

A RNN processa suas entradas, produzindo uma saída e um novo vetor de estado oculto (**h4**). A saída é descartada.

- **Etapas de Atenção:** Usamos os estados ocultos do **encoder** e o vetor **h4** para calcular um vetor de contexto (**C4**) para este passo.
- Concatenamos **h4** e **C4** em um único vetor.
- Passamos esse vetor por uma rede neural MLP (treinada conjuntamente com o modelo).
- A saída da rede neural MLP indica a palavra de saída para este passo de tempo.
- Repetimos para os próximos passos de tempo.

• • • • • +
• • • • • •

Encoder
hidden
state

Je

hidden
state #1

suis

hidden
state #2

étudiant

hidden
state #3



• • • • •
• • • • •
• + •
+ •

ATENÇÃO

É TUDO QUE VOCÊ PRECISA

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noan@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

Lukasz Kaiser*
Google Brain
lukaszkaizer@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.



















Xiv:1706.03762v5 [cs.CL] 6 Dec 2017

TRANSFORMERS



POSITIONAL ENCODING

Embedding

Token	Token Embedding ?	Positional Encoding ?	
This	 id 1212	+	 position 0 = 
Data	 6060	+	 1 = 
Science	 5800	+	 2 = 
class	 1398	+	 3 = 
is	 318	+	 4 = 
awesome	 7427	+	 5 = 

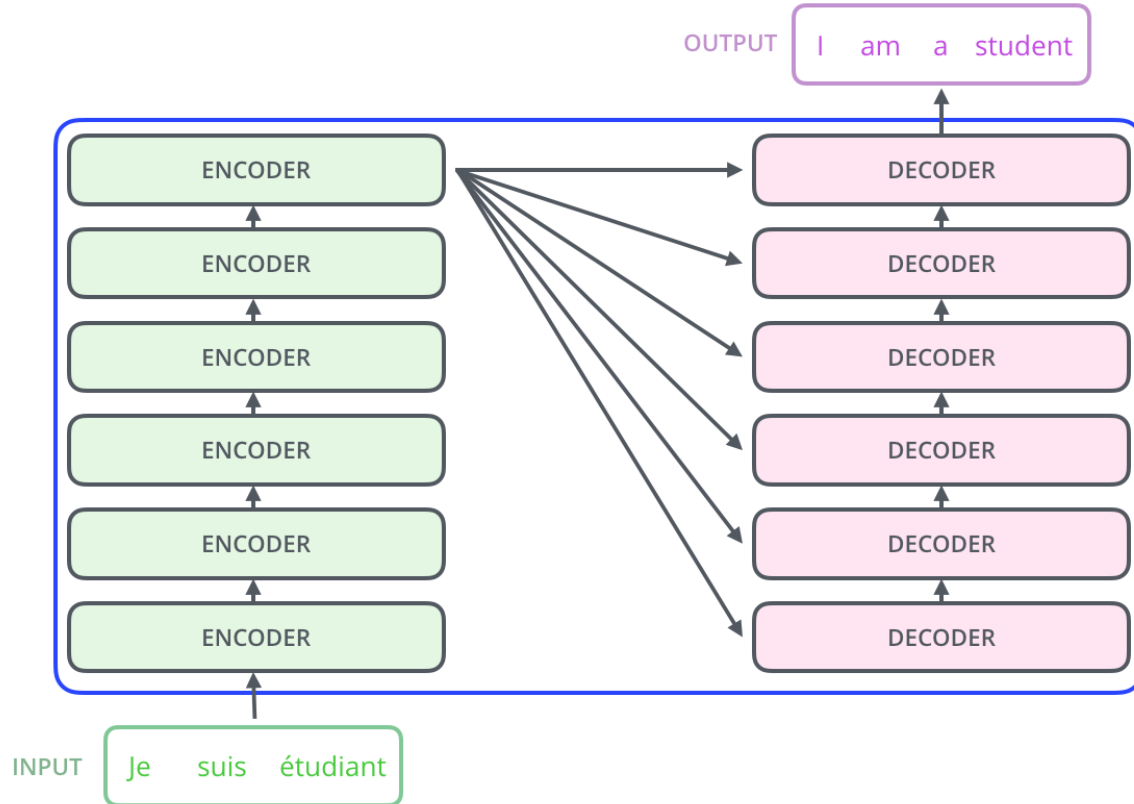
POSITIONAL ENCODING

Positional Encoding é um dos componentes mais importantes do Transformer, pois resolve a falta de informação posicional intrínseca, uma vez que o modelo não processa a sequência de maneira sequencial como uma LSTM. O Positional Encoding introduz informações sobre a ordem das palavras na sequência para que o modelo possa entender a posição relativa dos tokens.

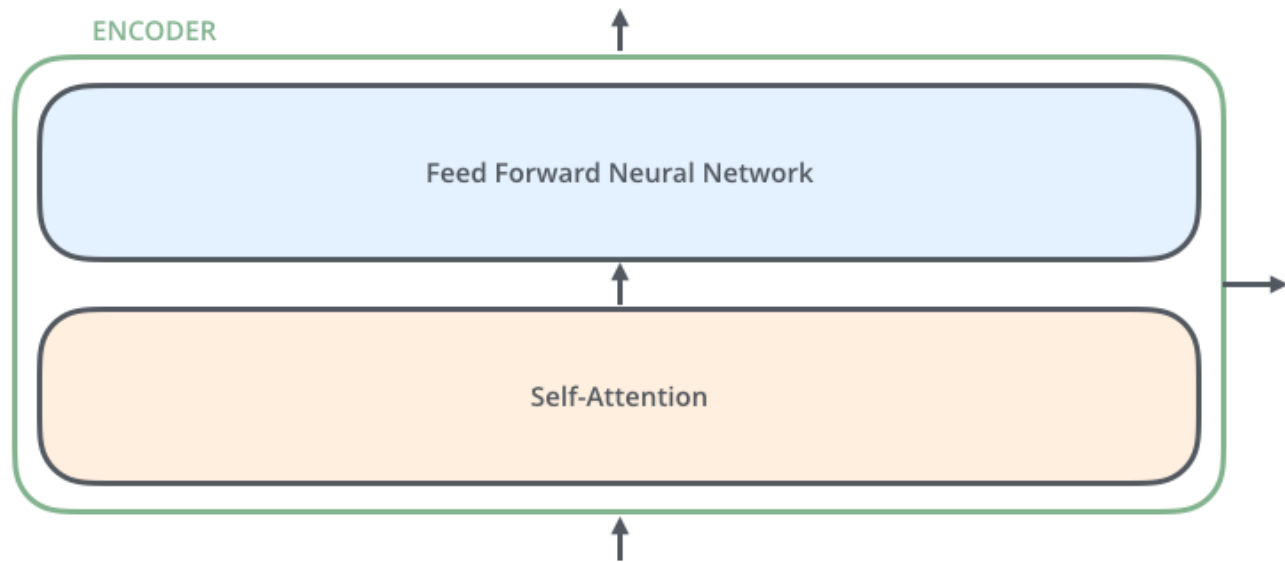
Por que Precisamos de Positional Encoding?

- Diferente das redes LSTM, que processam as palavras uma de cada vez, o Transformer processa todas as palavras ao mesmo tempo (em paralelo).
- Sem um componente posicional, o Transformer não saberia a ordem das palavras na sequência. Por exemplo, “o gato preto” e “preto o gato” teriam a mesma representação se não soubéssemos a ordem.

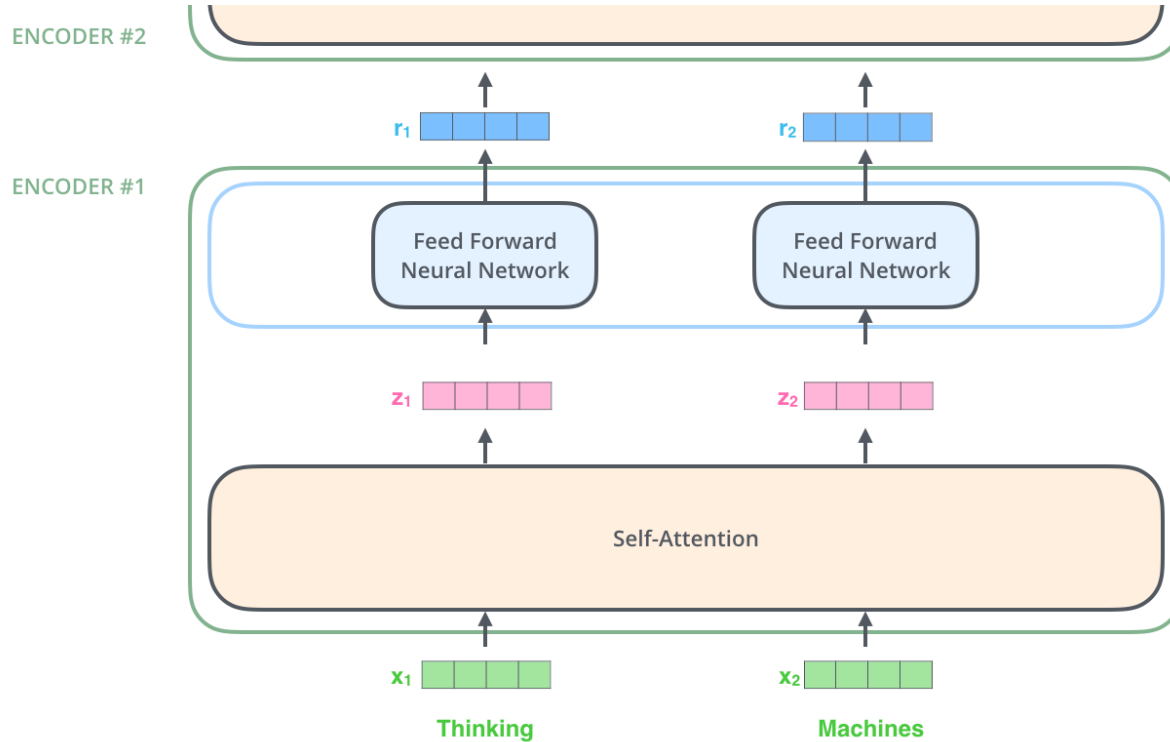
TRANSFORMERS



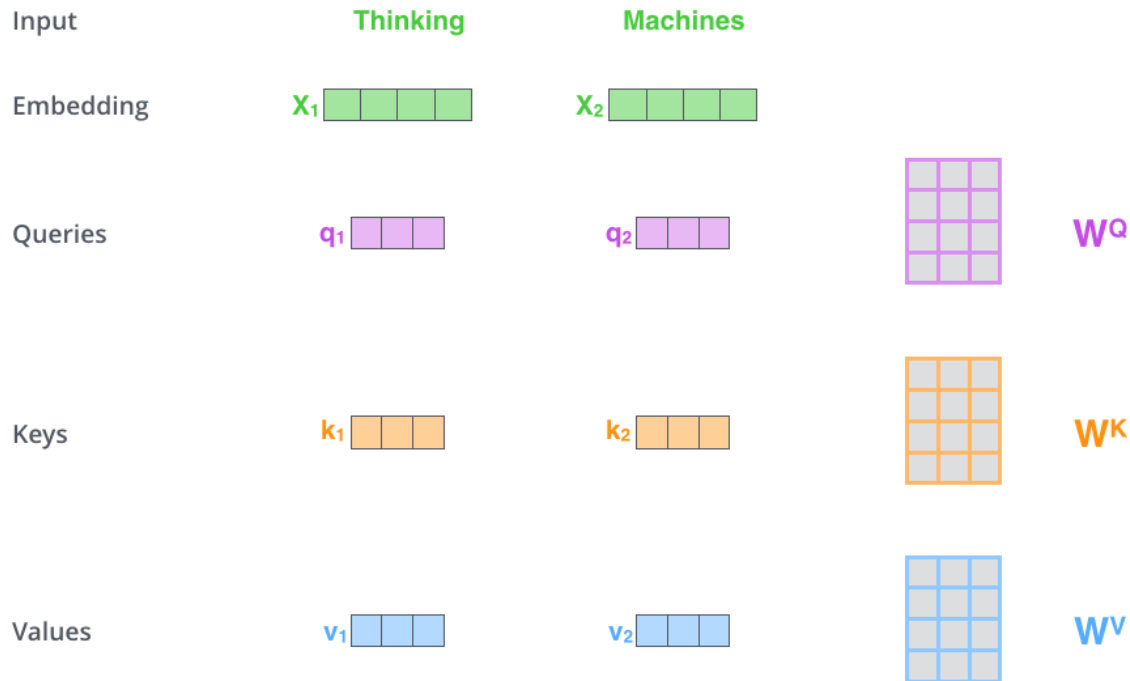
TRANSFORMERS – SELF-ATTENTION



TRANSFORMERS – SELF-ATTENTION



TRANSFORMERS – SELF-ATTENTION



Multiplicar x_1 pela matriz de pesos W^Q produz q_1 , o vetor de “query” associado a essa palavra. Assim, acabamos criando uma projeção de “query”, “key” e “value” para cada palavra na sentença de entrada.

TRANSFORMERS – SELF-ATTENTION

O **primeiro passo** para calcular a **self-attention** é criar três vetores a partir de cada um dos vetores de entrada do encoder (neste caso, o **embedding** de cada palavra). Assim, para cada palavra, criamos um vetor de **Query** (Consulta), um vetor de **Key** (Chave) e um vetor de **Value** (Valor). Esses vetores são criados multiplicando o **embedding** por três matrizes que foram treinadas durante o processo de treinamento.

Observe que esses novos vetores têm uma dimensão menor do que o vetor de **embedding**. A dimensionalidade deles é 64, enquanto os vetores de **embedding** e de entrada/saída do encoder têm dimensionalidade de 512. Eles **não precisam** ser menores; isso é uma escolha de arquitetura para tornar o cálculo da atenção multi-cabeça (principalmente) constante.

TRANSFORMERS – SELF-ATTENTION

Each token's embedding vector is transformed into three vectors: **Query (Q)**, **Key (K)**, and **Value (V)**. These vectors are derived by multiplying the input embedding matrix with learned weight matrices for **Q**, **K**, and **V**. Here's a web search analogy to help us build some intuition behind these matrices:

- **Query (Q)** is the search text you type in the search engine bar. This is the token you want to *"find more information about"*.
- **Key (K)** is the title of each web page in the search result window. It represents the possible tokens the query can attend to.
- **Value (V)** is the actual content of web pages shown. Once we matched the appropriate search term (Query) with the relevant results (Key), we want to get the content (Value) of the most relevant pages.

TRANSFORMERS – SELF-ATTENTION

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ($\sqrt{d_k}$)

Softmax

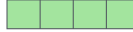
Softmax

X

Value

Sum

Thinking

x_1 

q_1 

k_1 

v_1 

$q_1 \cdot k_1 = 112$

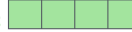
14

0.88

v_1 

z_1 

Machines

x_2 

q_2 

k_2 

v_2 

$q_1 \cdot k_2 = 96$

12

0.12

v_2 

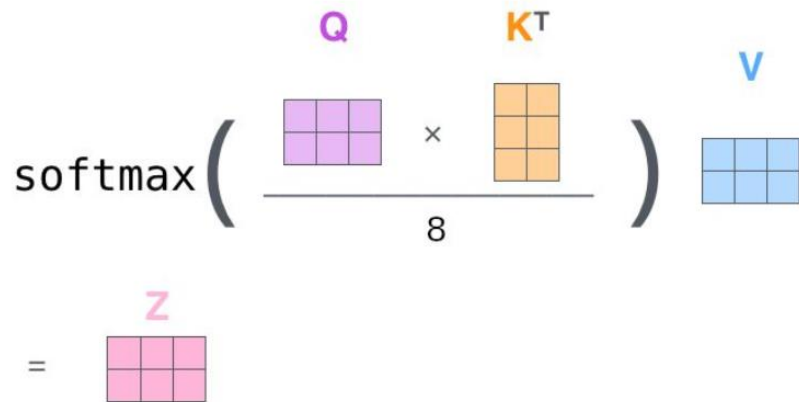
z_2 

TRANSFORMERS – SELF-ATTENTION

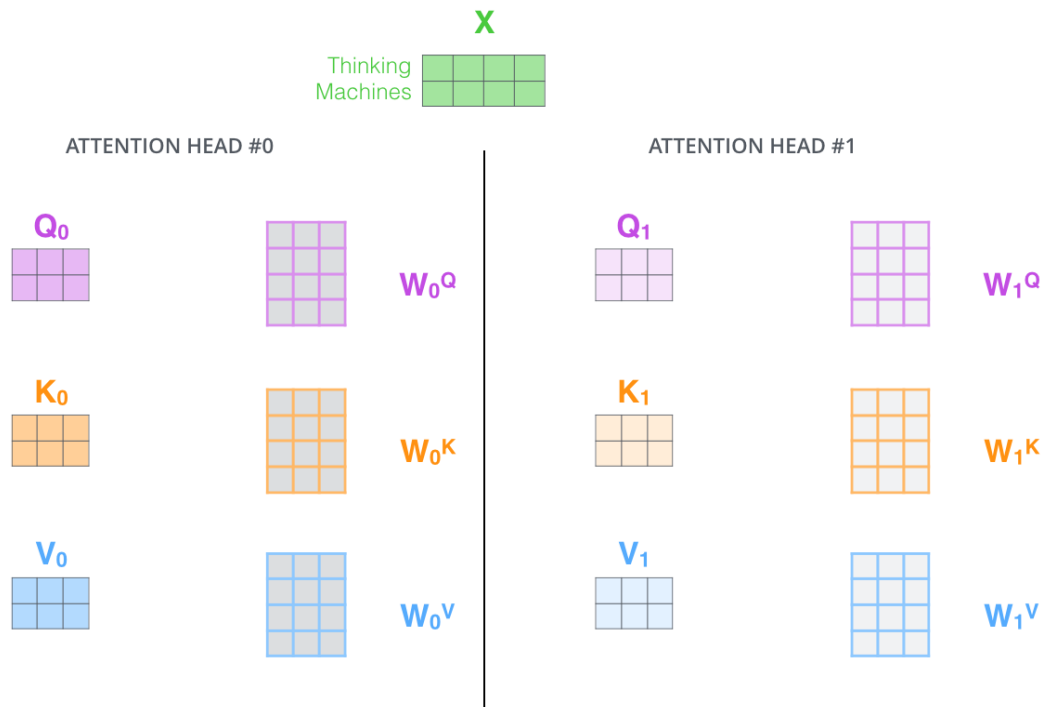
$$X \times W^Q = Q$$


$$X \times W^K = K$$


$$X \times W^V = V$$

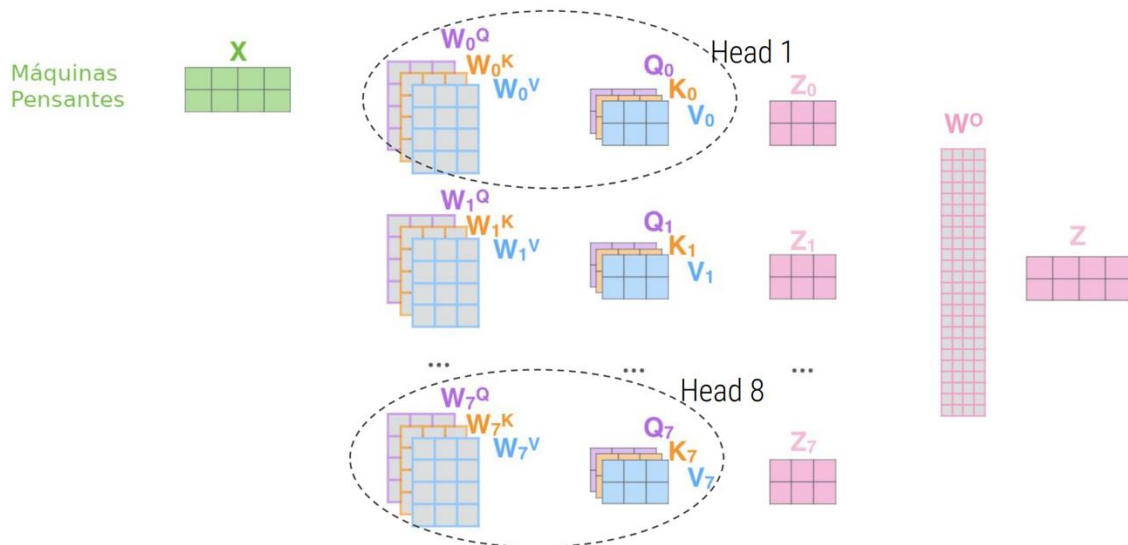

$$\text{softmax}\left(\frac{Q \times K^T}{8}\right) \times V = Z$$


TRANSFORMERS – MULT-HEADED ATTENTION



TRANSFORMERS – MULT-HEADED ATTENTION

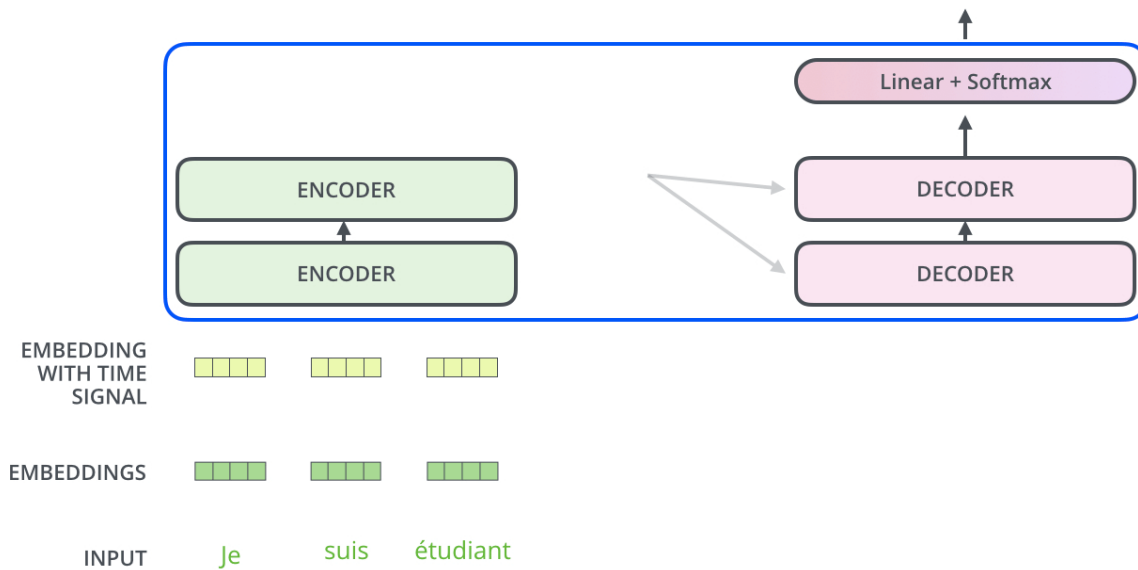
- 1) Entrada
- 2) Entrada codificada
- 3) Dividir em 8 cabeças e multiplicar X por cada uma delas.
- 4) Calcular atenção usando as matrizes $Q/K/V$
- 5) Concatenar as matrizes Z_i e multiplicar o resultado com uma matriz de pesos W^O para produzir a saída.



TRANSFORMERS – DECODER

Decoding time step: 1 2 3 4 5 6

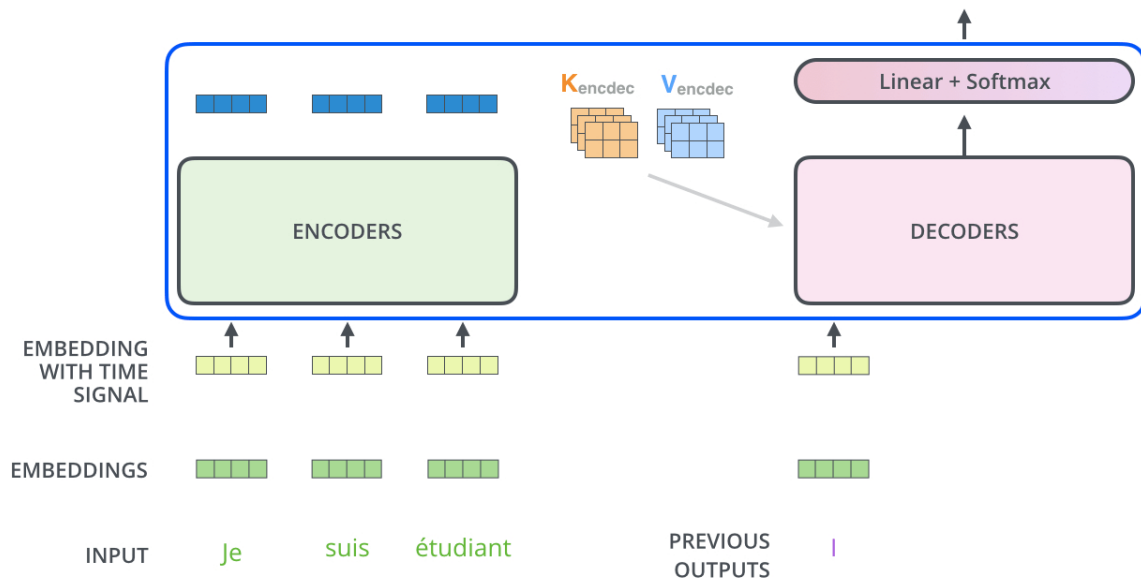
OUTPUT



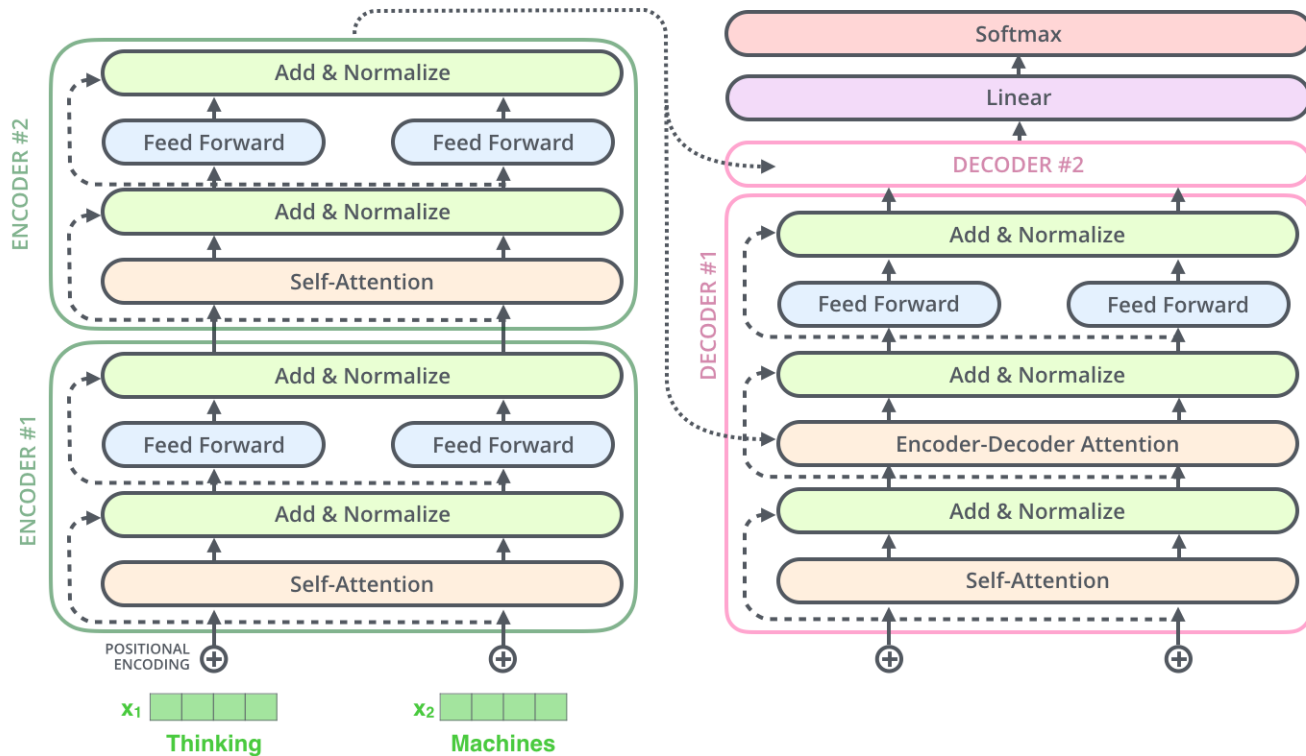
TRANSFORMERS – DECODER

Decoding time step: 1 2 3 4 5 6

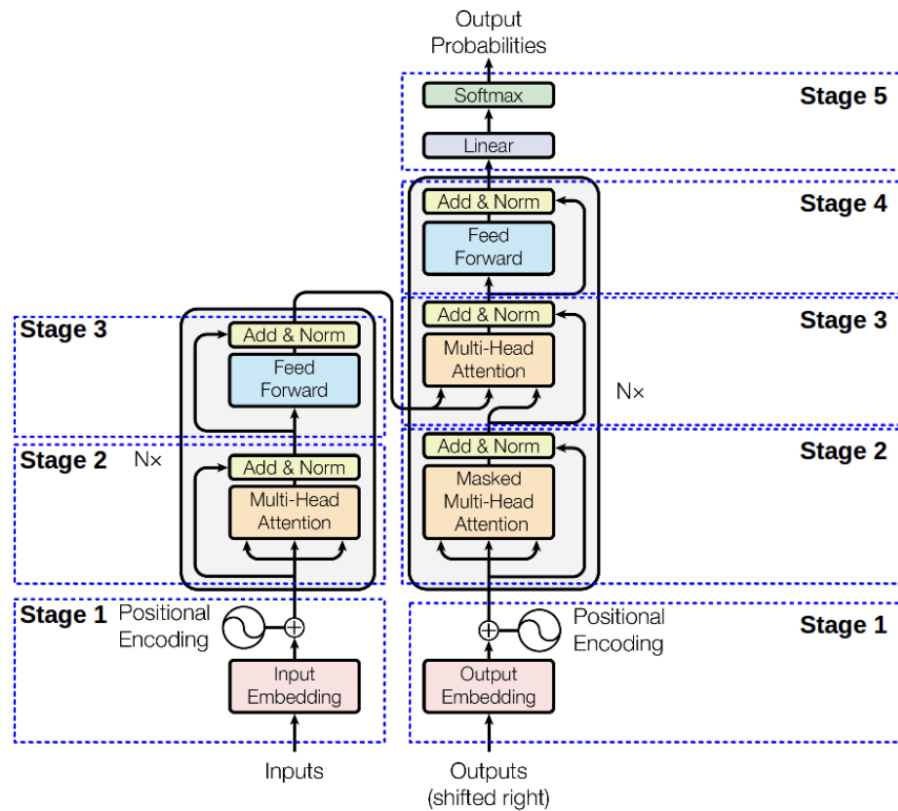
OUTPUT |



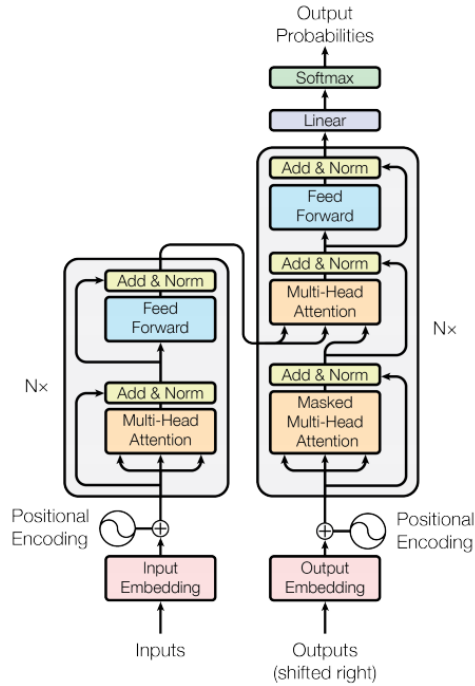
TRANSFORMERS – DECODER



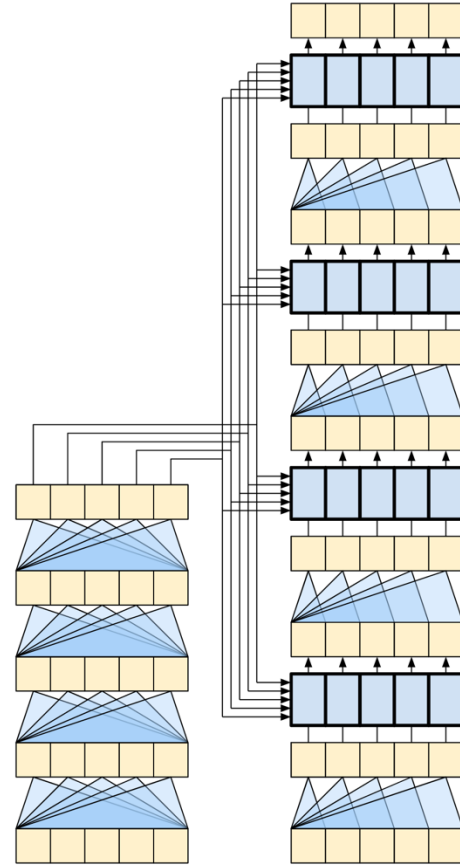
TRANSFORMERS – DECODER



TRANSFORMERS

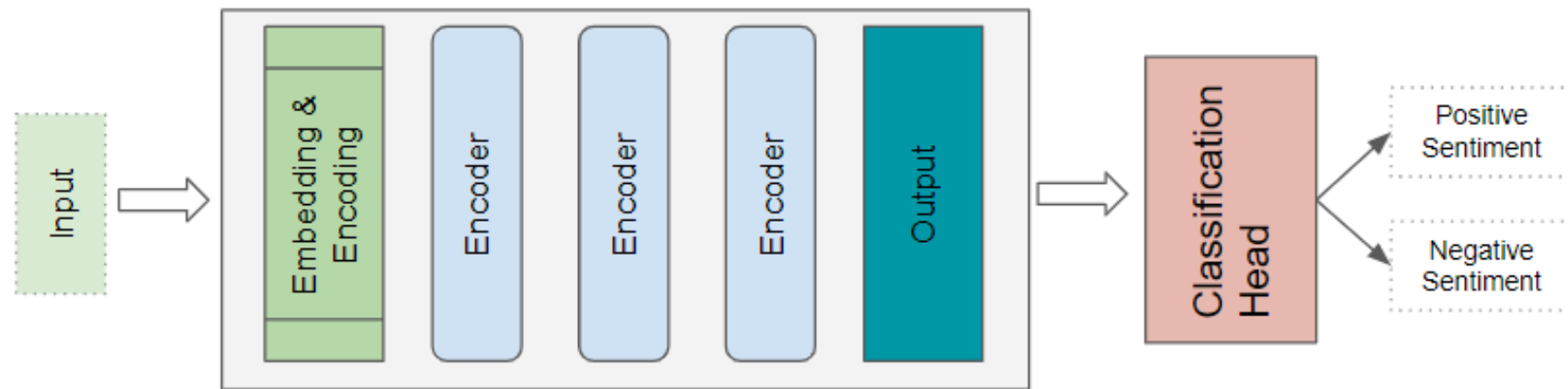


The original Transformer diagram



A representation of a 4-layer Transformer

TRANSFORMERS – CLASSIFICAÇÃO



TRANSFORMERS – ESTADO DA ARTE

GOOGLE BERT

Considerado um marco importante na história da IA;
Artigo “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” do Google AI:

- Publicado em 2018 na NAACL;
- Com cerca de 12 mil citações atualmente.

Implementa os modelos de atenção e a arquitetura encoder-decoder;
Foco principal: perguntas e respostas.



TRANSFORMERS – ESTADO DA ARTE

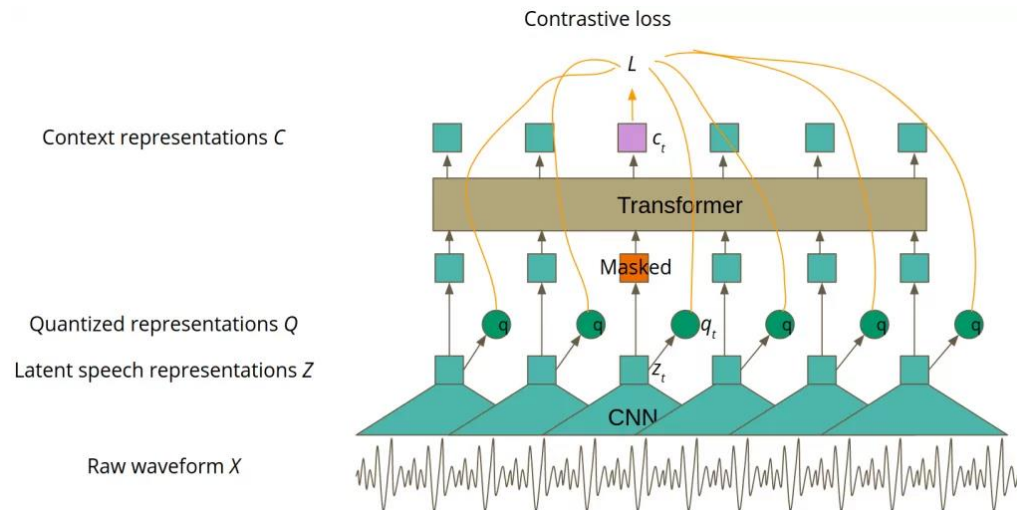
FACEBOOK WAV2VEC

o Wav2Vec é um modelo desenvolvido pelo Facebook AI Research (FAIR) para tarefas relacionadas ao processamento de fala e reconhecimento de fala. O modelo Wav2Vec foi projetado para extrair representações eficientes de áudio de forma não supervisionada, o que significa que ele pode aprender a representação sem a necessidade de rótulos ou transcrições para o áudio de treinamento.

TRANSFORMERS – ESTADO DA ARTE

FACEBOK WAV2VEC

O Wav2Vec tem sido utilizado em várias aplicações, incluindo reconhecimento de fala automático (ASR), onde o objetivo é transcrever áudio em texto. Modelos como o Wav2Vec podem ser fundamentais para melhorar o desempenho em tarefas de ASR, especialmente quando os dados de treinamento rotulados são limitados.



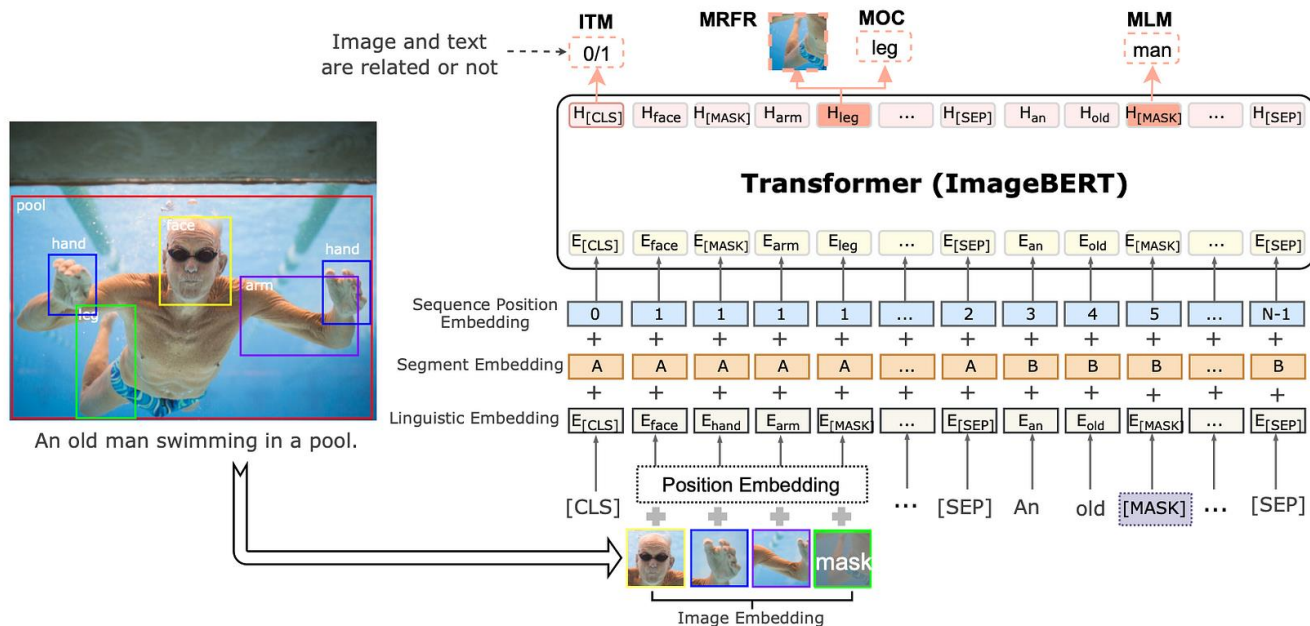
TRANSFORMERS – ESTADO DA ARTE

MICROSOFT IMAGEBERT

Projeto da Microsoft
(Jan, 2020);

Conceito de Image
Embedding;

Avanços rumo à sistemas
multi-canais.





FIAP

