

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Departamento de Informática Aplicada

Aula 12: Redes neurais para textos [3]

Prof. Dennis Giovani Balreira



INF01221 - Tópicos Especiais em Computação XXXVI:
Processamento de Linguagem Natural



Conteúdo

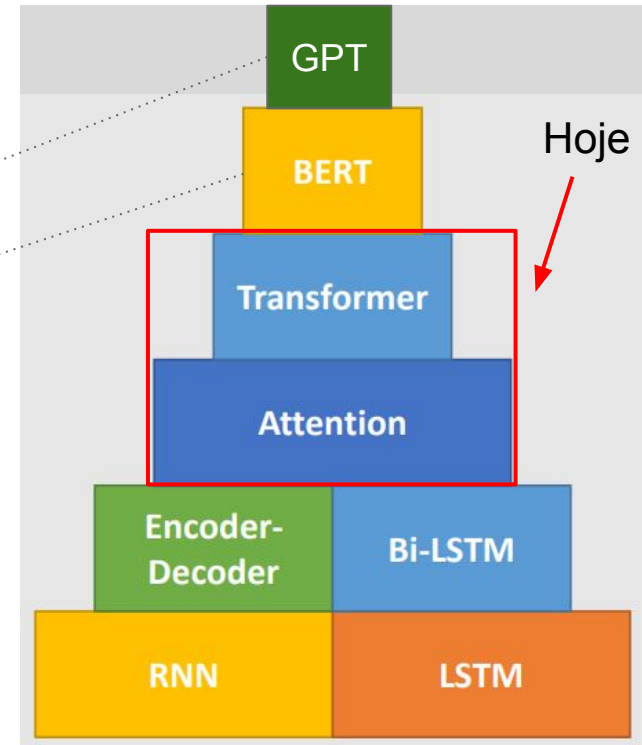
- Redes neurais para textos [3]
 - Transformer

Onde estamos em PLN?

- Algoritmos tradicionais
 - Predominantes entre o final dos anos 1990 até ~2016
 - BoW features + Aprendizado de Máquina
- Embeddings fixas + Deep Learning
 - Predominates de ~2014 até ~2019
 - Word2vec, Glove, FastText + LSTM
- Embeddings contextuais + Large Language Models
 - Estado da arte em diversas tarefas
 - BERT, GPT, etc.

Onde estamos em PLN?

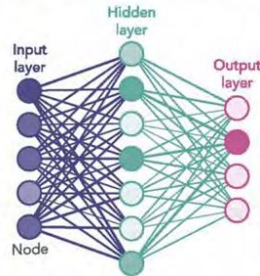
- Para chegar nos modelos de linguagem de larga escala (*Large Language Models - LLMs*), precisamos vencer a montanha de *deep learning para texto*



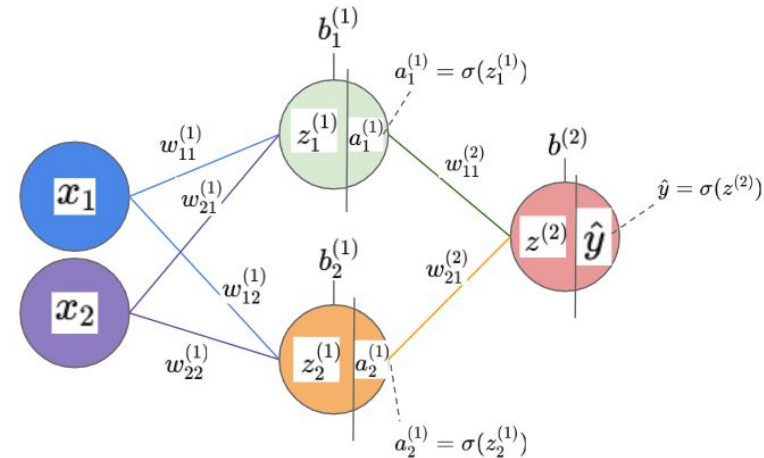
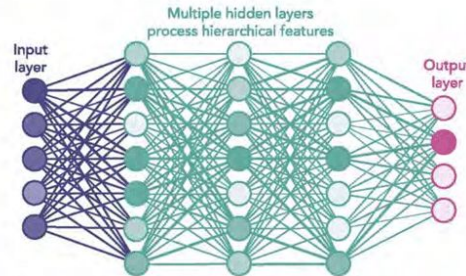
Redes Neurais: Revisão

- Modelos computacionais inspirados na estrutura e funcionamento do cérebro humano
- Redes **shallow vs. deep**:
 - Redes mais profundas têm melhor desempenho que redes superficiais
 - Mas apenas até certo limite
 - Após X camadas, o desempenho “estabiliza”

SHALLOW NEURAL NETWORK



DEEP NEURAL NETWORK



Função de ativação: $\sigma(z)$

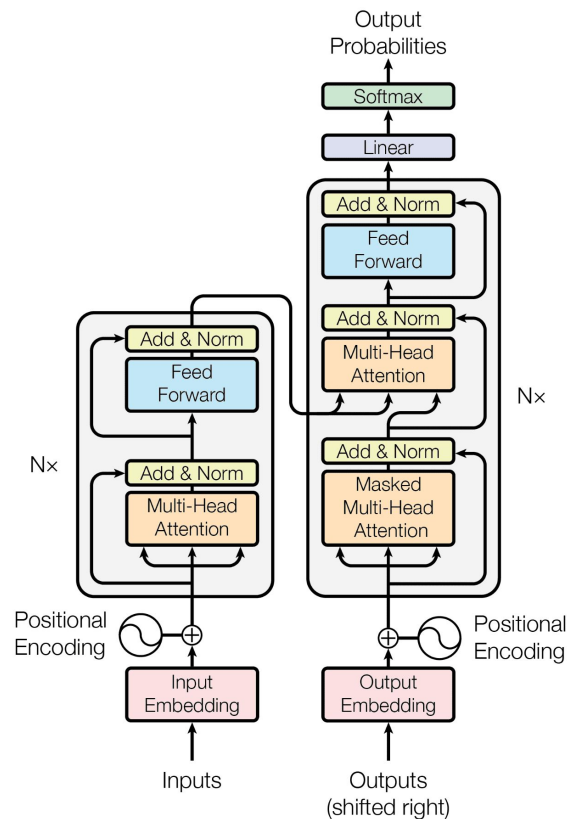
Saída esperada: y

Redes Neurais para textos

- Redes neurais para textos e dados sequenciais permitem capturar padrões e estruturas complexas a partir do contexto e sequência dos dados
- As seguintes arquiteturas são especializadas para texto e dados sequenciais:
 - Redes Neurais Recorrentes (RNN)
 - Long-short Term Memory (LSTM)
 - Bi Long-short Term Memory (Bi-LSTM)
 - Encoder-decoder
 - Transformer

Transformer

- Arquitetura que permite **processamento em paralelo** utilizando **encoder-decoder** e mecanismo de *self-attention*

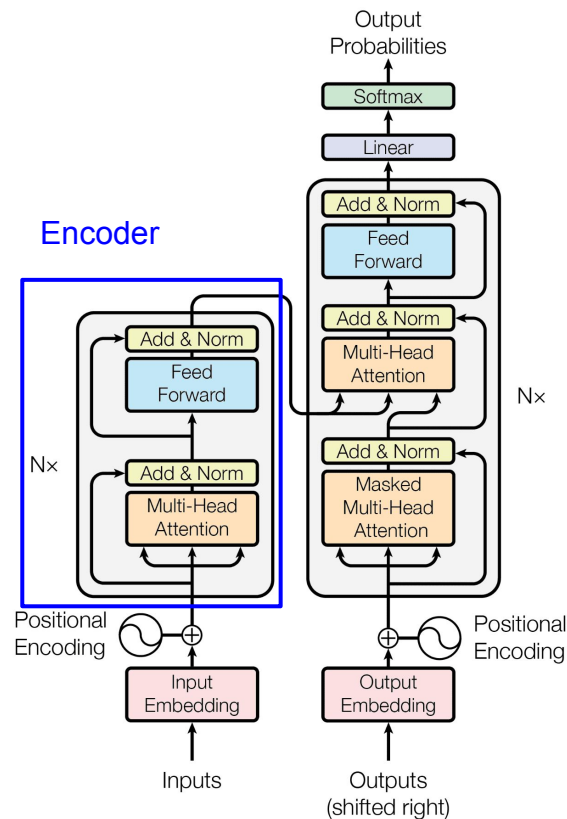


Transformer

- Arquitetura que permite **processamento em paralelo** utilizando **encoder-decoder** e mecanismo de *self-attention*

Encoder:

- Transforma sequência de entrada em uma representação (embedding) contextualizada, capturando relações entre todas as palavras de entrada
- Cada token (palavra) recebe um embedding contextual próprio



Transformer

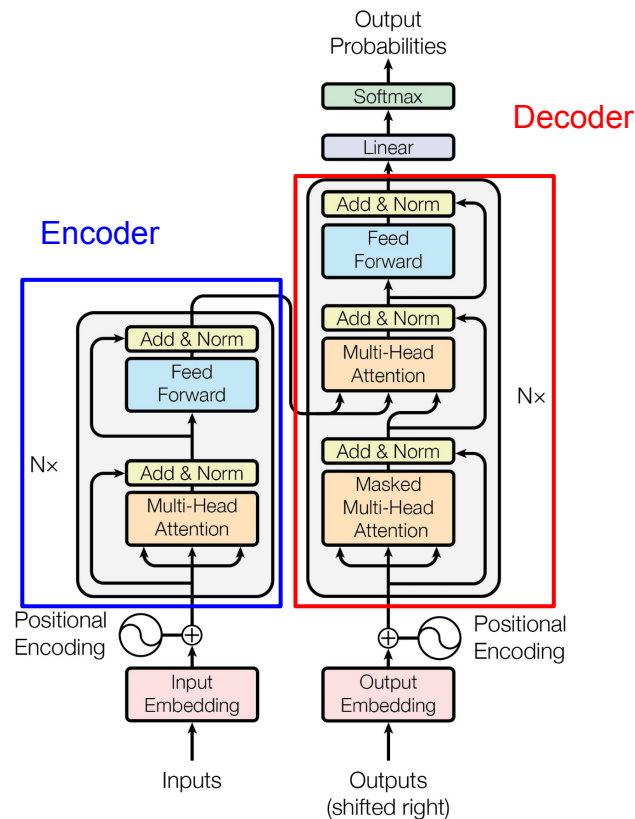
- Arquitetura que permite **processamento em paralelo** utilizando **encoder-decoder** e mecanismo de *self-attention*

Encoder:

- Transforma sequência de entrada em uma representação (embedding) contextualizada, capturando relações entre todas as palavras de entrada
- Cada token (palavra) recebe um embedding contextual próprio

Decoder:

- Gera sequência de saída token por token, usando (i) a saída do encoder e (ii) a sequência de saída gerada até o momento



Transformer

- Arquitetura que permite **processamento em paralelo** utilizando **encoder-decoder** e mecanismo de **self-attention**
 - Apresentado no paper *Attention is all you need* (Vaswani et al. - Google, 2017)

Attention is all you need

A Vaswani, N Shazeer, N Parmar... - Advances in neural ..., 2017 - proceedings.neurips.cc

... the number of **attention** heads and the **attention** key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head **attention** is 0.9 ...

☆ Salvar 99 Citar Citado por ? Artigos relacionados Todas as 73 versões >>

Acesso em 15/04/2025.

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

Lukasz Kaiser*
Google Brain
lukaszkaier@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.

Transformer

- Arquitetura que permite **processamento em paralelo** utilizando **encoder-decoder** e mecanismo de **self-attention**
 - Apresentado no paper *Attention is all you need* (Vaswani et al. - Google, 2017)

Attention is all you need

[A Vaswani](#), [N Shazeer](#), [N Parmar](#)... - *Advances in neural ...*, 2017 - [proceedings.neurips.cc](#)

... the number of **attention** heads and the **attention** key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head **attention** is 0.9 ...

☆ Salvar 99 Citar **Citado por 175987** Artigos relacionados Todas as 73 versões >>

Acesso em 15/04/2025.



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

Lukasz Kaiser*
Google Brain
lukaszkaier@google.com

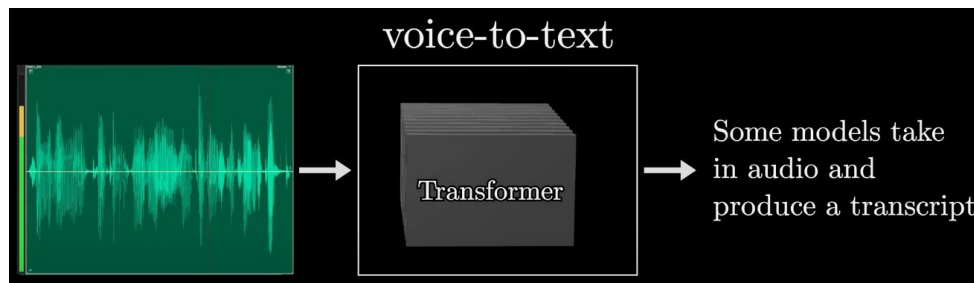
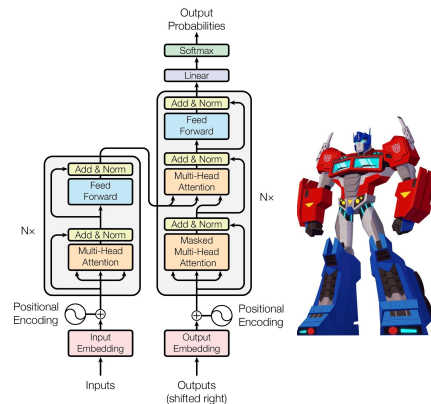
Illia Polosukhin*[‡]
illia.polosukhin@gmail.com

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.

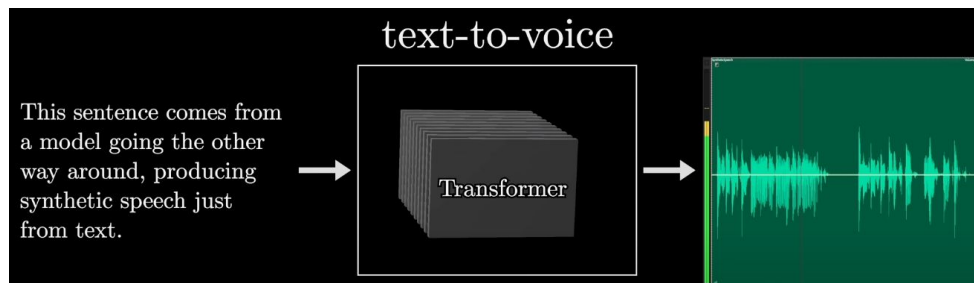
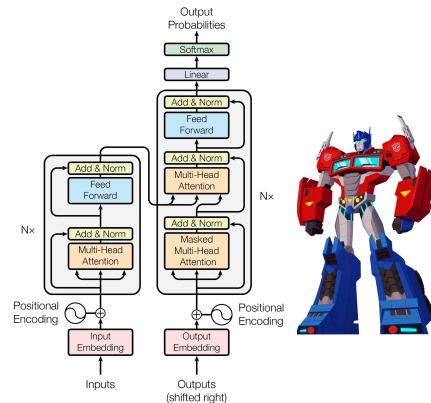
Transformer

- Arquitetura que permite **processamento em paralelo** utilizando **encoder-decoder** e mecanismo de **self-attention**
 - Apresentado no paper *Attention is all you need* (Vaswani et al. - Google, 2017)
- Diversos modelos usam:
 - Voice-to-text



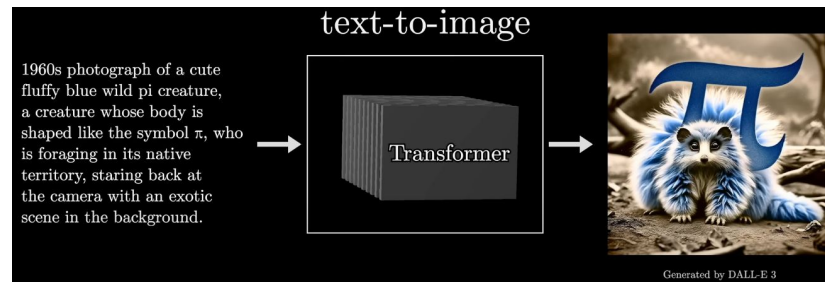
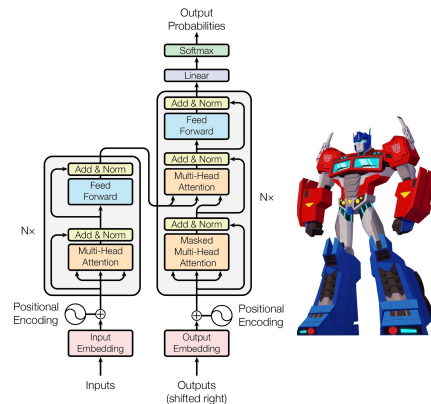
Transformer

- Arquitetura que permite **processamento em paralelo** utilizando **encoder-decoder** e mecanismo de **self-attention**
 - Apresentado no paper *Attention is all you need* (Vaswani et al. - Google, 2017)
- Diversos modelos usam:
 - Voice-to-text
 - Text-to-voice



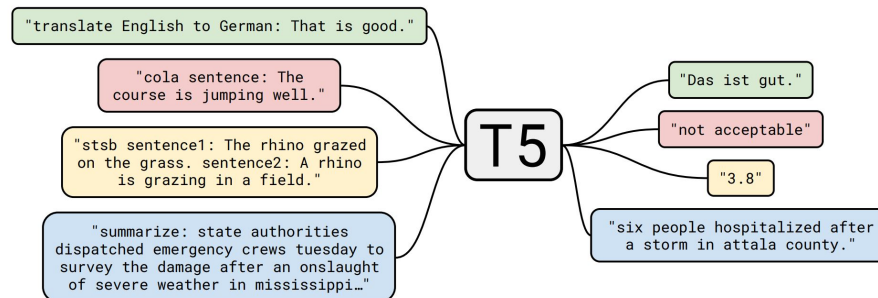
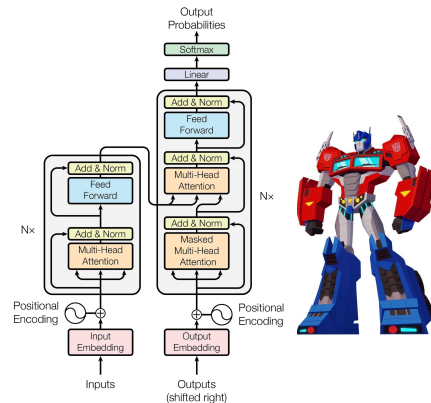
Transformer

- Arquitetura que permite **processamento em paralelo** utilizando **encoder-decoder** e mecanismo de **self-attention**
 - Apresentado no paper *Attention is all you need* (Vaswani et al. - Google, 2017)
- Diversos modelos usam:
 - Voice-to-text
 - Text-to-voice
 - Text-to-image



Transformer

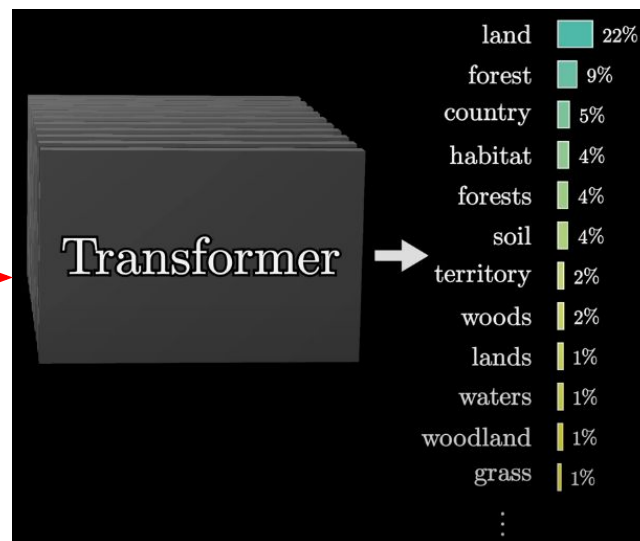
- Arquitetura que permite **processamento em paralelo** utilizando **encoder-decoder** e mecanismo de **self-attention**
 - Apresentado no paper *Attention is all you need* (Vaswani et al. - Google, 2017)
- Diversos modelos usam:
 - Voice-to-text
 - Text-to-voice
 - Text-to-image
 - Text-to-text**



Transformer

- Arquitetura que permite **processamento em paralelo** utilizando **encoder-decoder** e mecanismo de **self-attention**
 - Apresentado no paper *Attention is all you need* (Vaswani et al. - Google, 2017)
- Diversos modelos usam:
 - Voice-to-text
 - Text-to-voice
 - Text-to-image
 - **Text-to-text**
- LLMs generativas: prever próxima palavra

Behold, a wild pi creature,
foraging in its native _____



Transformer

- Resolvem dependência de longo prazo

Harry Potter was a highly unusual boy in many ways. For one thing, he hated the summer holidays more than any other time of year. For another, he really wanted to do his homework but was forced to do it in secret, in the dead of night. And he also happened to be a wizard.

It was nearly midnight, and he was lying on his stomach in bed, the blankets drawn right over his head like a tent, a flashlight in one hand and a large leather-bound book (A History of Magic by Bathilda Bagshot) propped open against the pillow. Harry moved the tip of his eagle-feather quill down the page, frowning as he looked for something that would help him write his essay, "Witch Burning in the Fourteenth Century Was Completely Pointless discuss."

The quill paused at the top of a likely-looking paragraph. Harry Pushed his round glasses up the bridge of his nose, moved his flashlight closer to the book, and read:

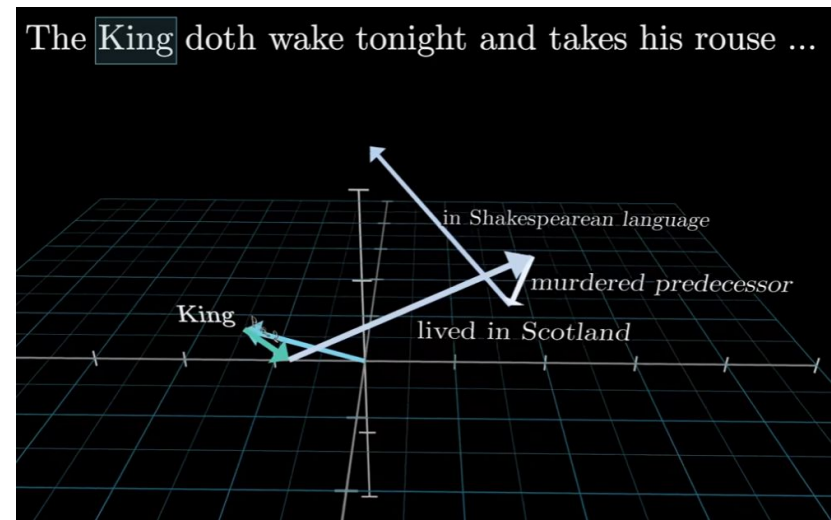
Non-magic people (more commonly known as Muggles) were particularly afraid of magic in medieval times, but not very good at recognizing it. On the rare occasion that they did catch a real witch or wizard, burning had no effect whatsoever. The witch or wizard would perform a basic Flame Freezing Charm and then pretend to shriek with pain while enjoying a gentle, tickling sensation. Indeed, Wendelin the Weird enjoyed being burned so much that she allowed herself to be caught no less than fiftyseven times in various disguises.

Harry put his quill between his teeth and reached underneath his pillow for his ink bottle and a roll of parchment. Slowly and very carefully he unscrewed the ink bottle, dipped his quill into it, and began to write, pausing every now and then to listen, because if any of the Dursleys heard the scratching of his quill on their way to the bathroom, he'd probably find himself locked in the cupboard under the stairs for the rest of the summer.

The Dursley family of number four, Privet Drive, was the reason that Harry never enjoyed his summer holidays. Uncle Vernon, Aunt Petunia, and their son, Dudley, were Harry's only living relatives. They were Muggles, and they had a very medieval attitude toward magic. Harry's dead parents, who had been a witch and wizard themselves, were never mentioned under the Dursleys' roof. For years, Aunt Petunia and Uncle Vernon had hoped that if they kept Harry as downtrodden as possible, they would be able to squash the magic out of him. To their fury, they had been unsuccessful. These days they lived in terror of anyone finding out that Harry had spent most of the last two years at Hogwarts School of Witchcraft and Wizardry. The most they could do, however, was to lock away Harry's spellbooks, wand, cauldron, and broomstick at the start of the summer break, and forbid him to talk to the neighbors.

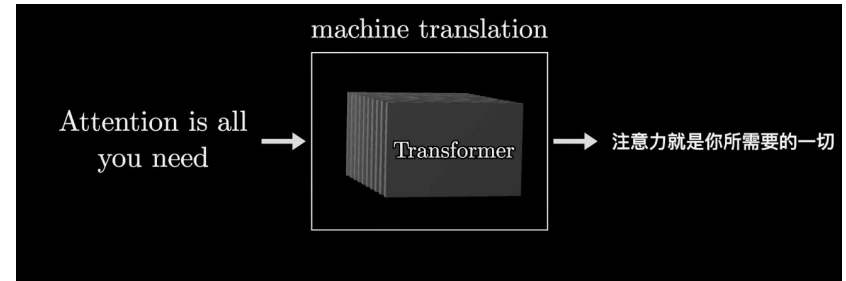
Transformer

- Resolvem dependência de longo prazo
- Intuição: vetores de word embeddings são “adaptados” pelo contexto



Transformer

- Originalmente, a “tarefa” visada pelo paper original foi tradução automática



Transformer

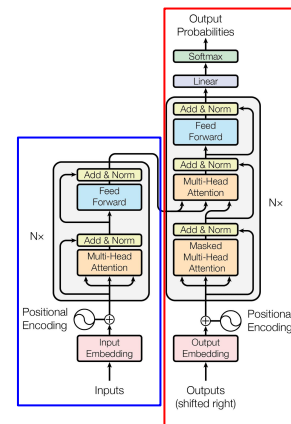
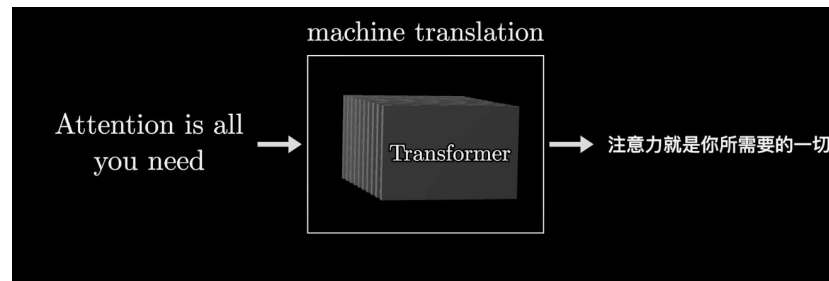
- Originalmente, a “tarefa” visada pelo paper original foi tradução automática
- Exemplo:
 - Encoder: lado inglês
 - Decoder: lado alemão

Fonte (Inglês): "The cat sleeps on the mat."

Tradução (Alemã): "Die Katze schläft auf der Matte."

- **Datasets:**

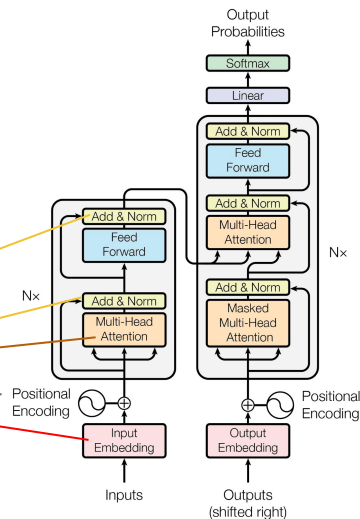
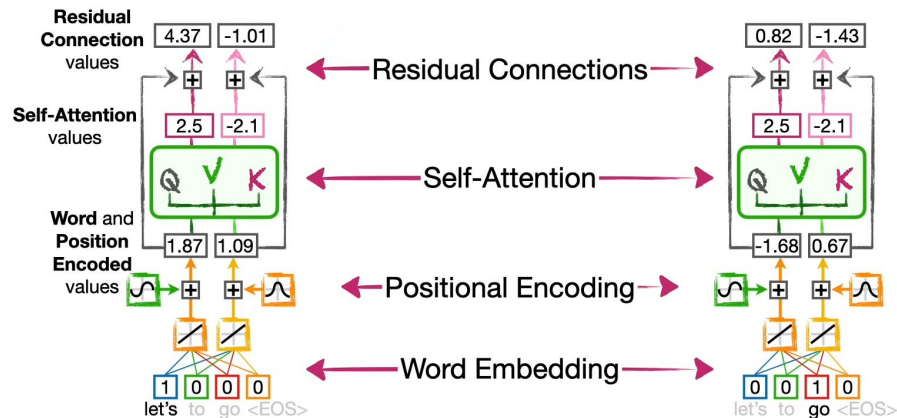
- **WMT 2014 English-to-German:** 4.5 milhões de pares de frases.
- **WMT 2014 English-to-French:** 36 milhões de pares.



Transformer

- Principais pontos:

- Input embedding
- Positional encoding
- Self-attention
- Residual connections



Train in parallel

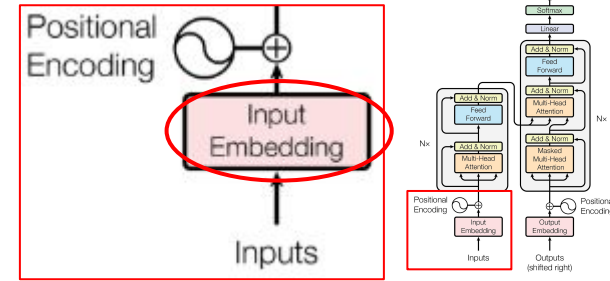
Encode the relationships among the words

Encode the positions of the words

Encode words into numbers

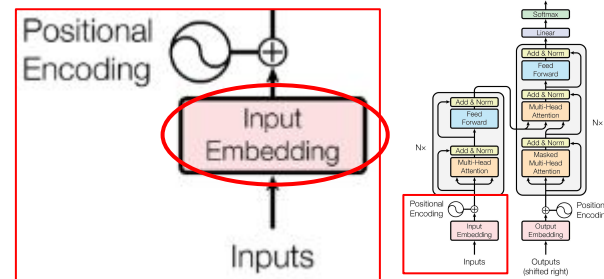
Inputs

- Entrada do modelo como texto (frase, parágrafo, etc.)



Input embedding

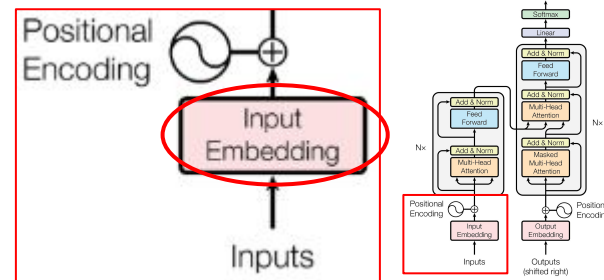
- Camada responsável por converter palavras (na verdade tokens) em vetores numéricos densos (embeddings)
- Cada palavra está associada a um embedding específico, iniciado aleatoriamente (mas com valores pequenos e controlados)



The	goal	of	our	model	is	to	predict	the	next	???
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
[5.4]	[9.5]	[0.2]	[1.2]	[4.5]	[3.6]	[3.1]	[6.5]	[1.9]	[9.7]	
[7.1]	[3.8]	[8.2]	[6.3]	[5.6]	[4.3]	[3.6]	[2.5]	[3.7]	[6.0]	
[6.0]	[7.8]	[7.7]	[1.4]	[0.2]	[6.9]	[5.6]	[4.6]	[8.1]	[7.3]	
[5.4]	[5.2]	[8.6]	[9.4]	[6.1]	[0.6]	[4.3]	[2.4]	[1.0]	[0.4]	
[4.2]	[5.6]	[9.7]	[5.2]	[6.1]	[6.6]	[9.8]	[1.6]	[8.3]	[2.8]	
[6.4]	[9.2]	[7.9]	[4.1]	[6.1]	[6.6]	[1.0]	[1.1]	[1.0]	[1.2]	
[⋮]	[⋮]	[⋮]	[⋮]	[⋮]	[⋮]	[⋮]	[⋮]	[⋮]	[⋮]	
[8.8]	[0.9]	[7.7]	[7.7]	[6.8]	[1.3]	[1.6]	[1.4]	[4.6]	[1.2]	

Input embedding

- Camada responsável por converter palavras (na verdade tokens) em vetores numéricos densos (embeddings)
- Cada palavra está associada a um embedding específico, iniciado aleatoriamente (mas com valores pequenos e controlados)
- É necessário prever embeddings para todas as palavras (tokens) do vocabulário

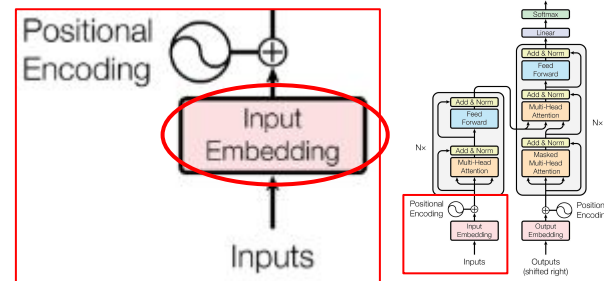


All words, ~ 50k

aah	aardvark	aartwolf	aargh	ab	aback	abacterial	abacus	abalone	abandon	...	zygoid	zygomatic	zygomorphic	zygosis	zygote	zygotic	zyne	zymogen	zymosis	zzz
+1.0	+4.3	+2.0	+0.9	-1.5	+2.9	-1.2	+7.8	+9.2	-2.3	...	+0.6	+1.3	+8.4	-8.5	-8.2	-9.5	+6.6	+5.5	+7.3	+9.5
+5.9	-0.8	+5.6	-7.6	+2.8	-7.1	+8.8	+0.4	-1.7	-4.7	...	-0.9	+1.4	-9.5	+2.3	+2.2	+2.3	+8.8	+3.6	-2.8	-1.2
+3.9	-8.7	+3.3	+3.4	-5.7	-7.3	-3.7	-2.7	+1.4	-1.2	...	-7.9	-5.8	-6.7	+3.0	-4.9	-0.7	-5.1	-6.8	-7.7	+3.1
-7.2	-6.0	-2.6	+6.4	-8.0	+6.7	-8.0	+9.4	-0.6	+9.4	...	+4.7	-9.1	-4.3	-7.5	-4.0	-7.5	-3.6	-1.7	-8.6	+3.8
+1.3	-4.6	+0.5	-8.0	+1.5	+8.5	-3.6	+3.3	-7.3	+4.3	...	-6.3	+1.7	-9.5	+6.5	-9.8	+3.5	-4.6	+4.7	+9.2	-5.0
+1.5	+1.8	+1.4	-5.5	+9.0	-1.0	+6.9	+3.9	-4.0	+6.2	...	+7.5	+1.6	+7.6	+3.8	+4.5	+0.0	+9.0	+2.9	-1.5	+2.1
-9.5	-3.9	+3.2	-4.2	+2.3	-1.4	-7.2	-4.0	+1.4	+1.8	...	+3.0	+3.0	-1.4	+7.9	-2.6	-1.3	+7.8	+6.1	+4.0	-7.9
+8.3	+4.2	+9.9	-6.9	+7.3	-6.7	+2.3	-7.4	+6.9	+6.1	...	-1.8	-8.5	+3.9	-0.9	+4.4	+7.3	+9.4	+7.0	-9.7	-2.8
:	:	:	:	:	:	:	:	:	:	...	:	:	:	:	:	:	:	:	:	:
-3.7	-2.0	-5.7	-6.2	+8.8	+4.7	-0.2	-5.4	-4.9	-8.8	...	-3.7	+3.9	-2.4	-6.3	-9.4	-8.6	+3.6	-0.9	+0.7	+7.9

Input embedding

- Camada responsável por converter palavras (na verdade tokens) em vetores numéricos densos (embeddings)
- Cada palavra está associada a um embedding específico, iniciado aleatoriamente (mas com valores pequenos e controlados)
- É necessário prever embeddings para todas as palavras (tokens) do vocabulário
- Palavras usam “one-hot encoding” implícito:
 - Em vez de criar vetores esparsos, cada palavra é associada diretamente à sua posição no vetor de vocabulário



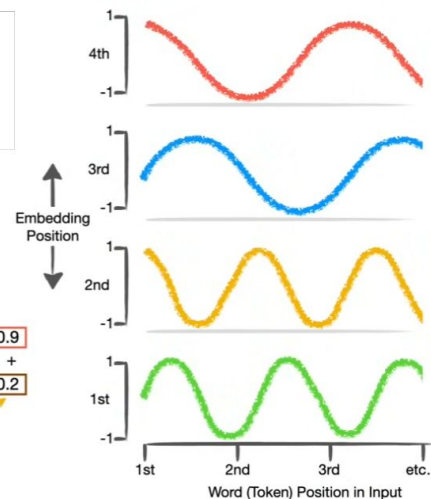
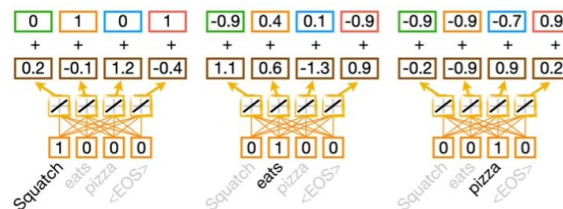
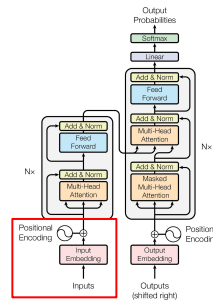
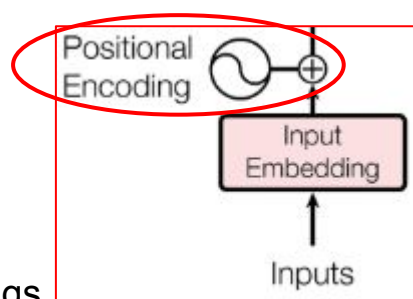
All words, ~ 50k

aah	aardvark	aartwolf	aargh	ab	aback	abacterial	abacus	abalone	abandon	...	zygoid	zygomorphic	zygomatic	zygote	zygotic	zyne	zymogen	zymosis	zzz	
+1.0	+4.3	+2.0	+0.9	-1.5	+2.9	-1.2	+7.8	+9.2	-2.3	...	+0.6	+1.3	+8.4	-8.5	-8.2	-9.5	+6.6	+5.5	+7.3	+9.5
+5.9	-0.8	+5.6	-7.6	+2.8	-7.1	+8.8	+0.4	-1.7	-4.7	...	-0.9	+1.4	-9.5	+2.3	+2.2	+2.3	+8.8	+3.6	-2.8	-1.2
+3.9	-8.7	+3.3	+3.4	-5.7	-7.3	-3.7	-2.7	+1.4	-1.2	...	-7.9	-5.8	-6.7	+3.0	-4.9	-0.7	-5.1	-6.8	-7.7	+3.1
-7.2	-6.0	-2.6	+6.4	-8.0	+6.7	-8.0	+9.4	-0.6	+9.4	...	+4.7	-9.1	-4.3	-7.5	-4.0	-7.5	-3.6	-1.7	-8.6	+3.8
+1.3	-4.6	+0.5	-8.0	+1.5	+8.5	-3.6	+3.3	-7.3	+4.3	...	-6.3	+1.7	-9.5	+6.5	-9.8	+3.5	-4.6	+4.7	+9.2	-5.0
+1.5	+1.8	+1.4	-5.5	+9.0	-1.0	+6.9	+3.9	-4.0	+6.2	...	+7.5	+1.6	+7.6	+3.8	+4.5	+0.0	+9.0	+2.9	-1.5	+2.1
-9.5	-3.9	+3.2	-4.2	+2.3	-1.4	-7.2	-4.0	+1.4	+1.8	...	+3.0	+3.0	-1.4	+7.9	-2.6	-1.3	+7.8	+6.1	+4.0	-7.9
+8.3	+4.2	+9.9	-6.9	+7.3	-6.7	+2.3	-7.4	+6.9	+6.1	...	-1.8	-8.5	+3.9	-0.9	+4.4	+7.3	+9.4	+7.0	-9.7	-2.8
:	:	:	:	:	:	:	:	:	:	...	:	:	:	:	:	:	:	:	:	:
-3.7	-2.0	-5.7	-6.2	+8.8	+4.7	-0.2	-5.4	-4.9	-8.8	...	-3.7	+3.9	-2.4	-6.3	-9.4	-8.6	+3.6	-0.9	+0.7	+7.9

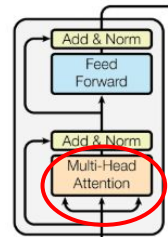
“abandon” possui ID=9

Positional encoding

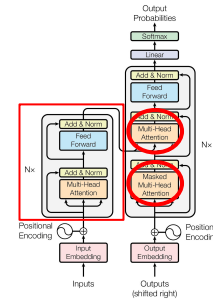
- **Vetores** que o Transformer adiciona aos embeddings das palavras para representar a **posição** de cada palavra na sequência
 - Soma de valores de funções **seno e cosseno**
 - Cada dimensão do embedding de cada token é somado com uma sequência de posições específicas
 - Funciona como uma espécie de “codificação” de posição única
- Os novos valores das funções são somados aos input embeddings



Self-attention mechanism



- Permite que **cada elemento de uma sequência** se relacione diretamente com **todos os outros elementos da mesma sequência**
 - O mecanismo de atenção é aplicado em todas as posições da sequência de entrada
 - Cada palavra "presta atenção" em todas as outras palavras da sequência, capturando relações de curto e longo alcance



João	João
levou	levou
o	o
material	material
em	em
seu	seu
carro	carro
mas	mas
ele	ele
não	não
era	era
um	um
bom	bom

Blue arrows indicate attention weights from the word 'carro' in the input sequence to the words 'João' and 'ele' in the output sequence.

Self-attention mechanism

- Attention possui três componentes: Q, K e V

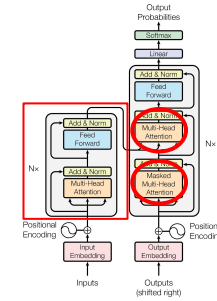
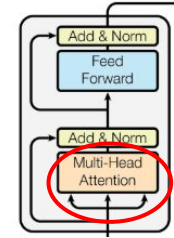
- (Q)uery
- (K)ey
- (V)alue

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

Garante que os valores
sejam normalizados
entre [0,1]

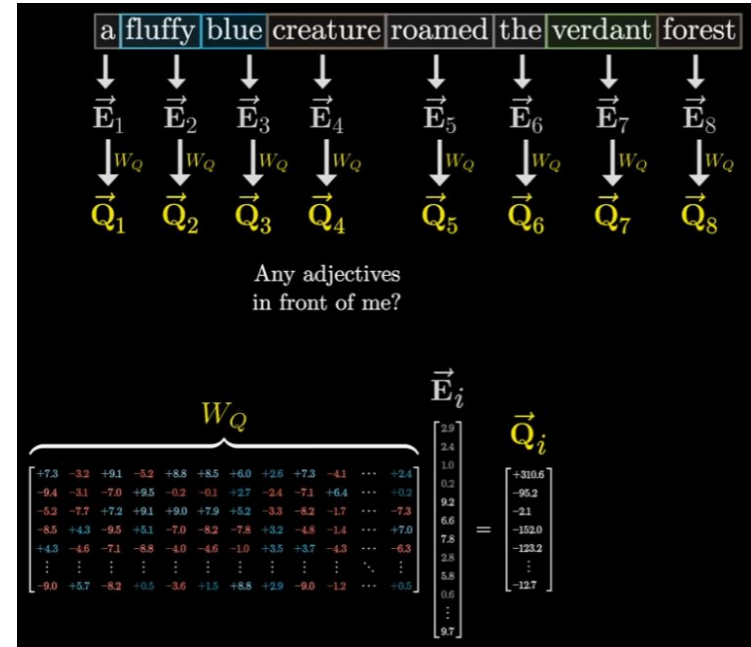
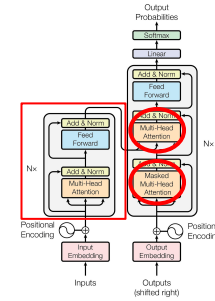
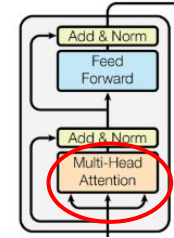
Dividir pela dimensão
da matriz K e Q

Evita que os scores
de atenção cresçam
demais



Self-attention mechanism

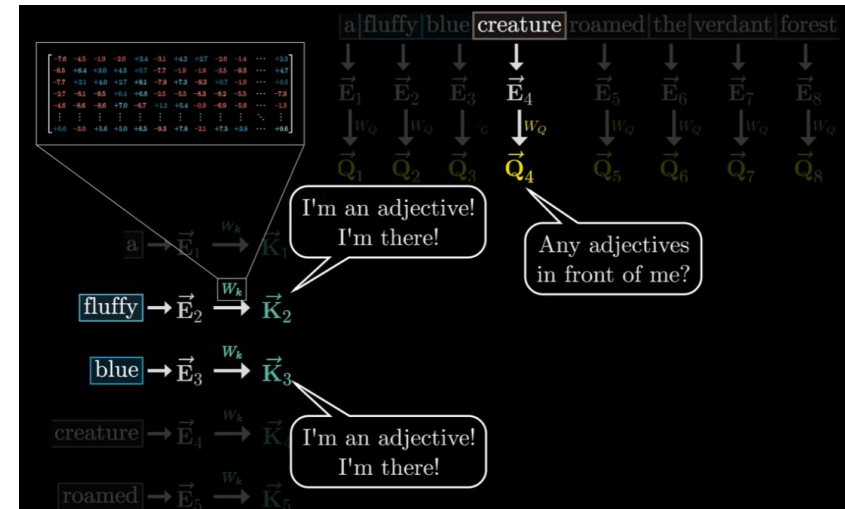
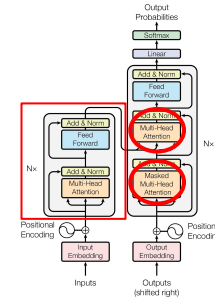
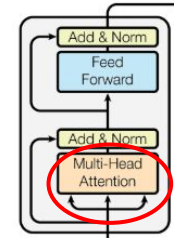
- Attention possui três componentes: Q, K e V
 - (Q)uery: Representa a "pergunta" que um token faz sobre os outros
 - (K)ey
 - (V)alue



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

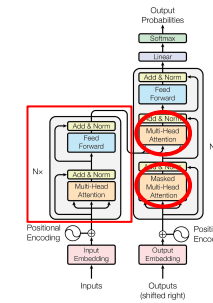
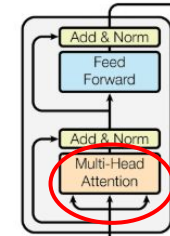
Self-attention mechanism

- Attention possui três componentes: Q, K e V
 - (Q)uery
 - (K)ey: Representa a “chave” que um token usa para responder às queries
 - (V)alue



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

Self-attention mechanism



- Attention possui três componentes: Q, K e V
 - (Q)uery
 - (K)ey: Representa a “chave” que um token usa para responder às queries
 - (V)alue

Queries e Keys definem um “espaço de similaridade” onde tokens com queries e keys alinhadas terão alta atenção

Essa similaridade é medida pelo dot product entre cada vetor K_i e Q_i

Maiores valores indicam maior similaridade entre pares (query,key)

“Attend to”

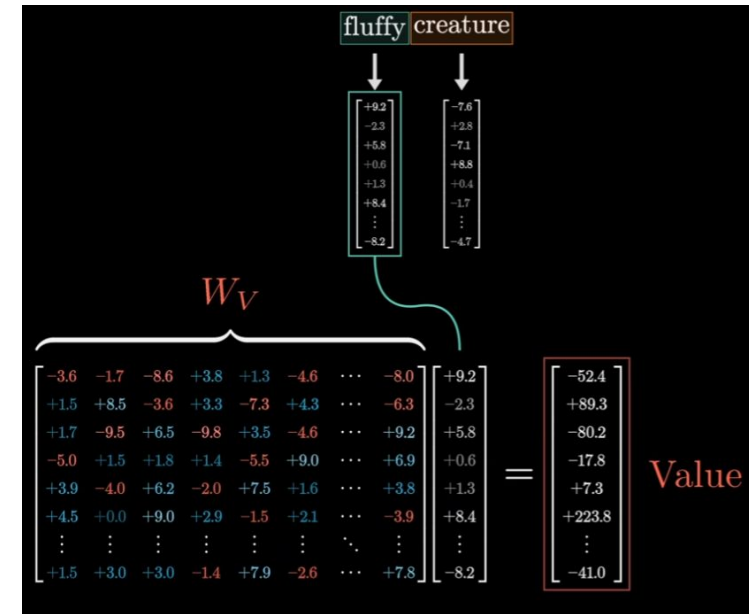
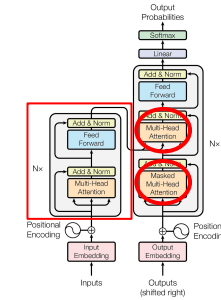
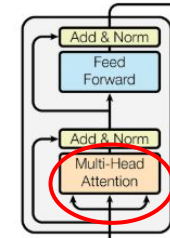
	a	fluffy	blue	creature
	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4
	\vec{Q}_1	\vec{Q}_2	\vec{Q}_3	\vec{Q}_4
a	$\vec{E}_1 \rightarrow \vec{E}_1 \xrightarrow{W_k} \vec{K}_1$	$\vec{K}_1 \cdot \vec{Q}_1$	$\vec{K}_1 \cdot \vec{Q}_2$	$\vec{K}_1 \cdot \vec{Q}_3$
fluffy	$\vec{E}_2 \rightarrow \vec{E}_2 \xrightarrow{W_k} \vec{K}_2$	$\vec{K}_2 \cdot \vec{Q}_1$	$\vec{K}_2 \cdot \vec{Q}_2$	$\vec{K}_2 \cdot \vec{Q}_3$
blue	$\vec{E}_3 \rightarrow \vec{E}_3 \xrightarrow{W_k} \vec{K}_3$	$\vec{K}_3 \cdot \vec{Q}_1$	$\vec{K}_3 \cdot \vec{Q}_2$	$\vec{K}_3 \cdot \vec{Q}_3$
creature	$\vec{E}_4 \rightarrow \vec{E}_4 \xrightarrow{W_k} \vec{K}_4$	$\vec{K}_4 \cdot \vec{Q}_1$	$\vec{K}_4 \cdot \vec{Q}_2$	$\vec{K}_4 \cdot \vec{Q}_3$

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

	a	fluffy	blue	creature	roamed	the	verdant	forest
	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8
	\vec{Q}_1	\vec{Q}_2	\vec{Q}_3	\vec{Q}_4	\vec{Q}_5	\vec{Q}_6	\vec{Q}_7	\vec{Q}_8
a	$\vec{E}_1 \rightarrow \vec{E}_1 \xrightarrow{W_k} \vec{K}_1$	$\vec{K}_1 \cdot \vec{Q}_1$	$\vec{K}_1 \cdot \vec{Q}_2$	$\vec{K}_1 \cdot \vec{Q}_3$	$\vec{K}_1 \cdot \vec{Q}_4$	$\vec{K}_1 \cdot \vec{Q}_5$	$\vec{K}_1 \cdot \vec{Q}_6$	$\vec{K}_1 \cdot \vec{Q}_7$
fluffy	$\vec{E}_2 \rightarrow \vec{E}_2 \xrightarrow{W_k} \vec{K}_2$	$\vec{K}_2 \cdot \vec{Q}_1$	$\vec{K}_2 \cdot \vec{Q}_2$	$\vec{K}_2 \cdot \vec{Q}_3$	$\vec{K}_2 \cdot \vec{Q}_4$	$\vec{K}_2 \cdot \vec{Q}_5$	$\vec{K}_2 \cdot \vec{Q}_6$	$\vec{K}_2 \cdot \vec{Q}_7$
blue	$\vec{E}_3 \rightarrow \vec{E}_3 \xrightarrow{W_k} \vec{K}_3$	$\vec{K}_3 \cdot \vec{Q}_1$	$\vec{K}_3 \cdot \vec{Q}_2$	$\vec{K}_3 \cdot \vec{Q}_3$	$\vec{K}_3 \cdot \vec{Q}_4$	$\vec{K}_3 \cdot \vec{Q}_5$	$\vec{K}_3 \cdot \vec{Q}_6$	$\vec{K}_3 \cdot \vec{Q}_7$
creature	$\vec{E}_4 \rightarrow \vec{E}_4 \xrightarrow{W_k} \vec{K}_4$	$\vec{K}_4 \cdot \vec{Q}_1$	$\vec{K}_4 \cdot \vec{Q}_2$	$\vec{K}_4 \cdot \vec{Q}_3$	$\vec{K}_4 \cdot \vec{Q}_4$	$\vec{K}_4 \cdot \vec{Q}_5$	$\vec{K}_4 \cdot \vec{Q}_6$	$\vec{K}_4 \cdot \vec{Q}_7$
roamed	$\vec{E}_5 \rightarrow \vec{E}_5 \xrightarrow{W_k} \vec{K}_5$	$\vec{K}_5 \cdot \vec{Q}_1$	$\vec{K}_5 \cdot \vec{Q}_2$	$\vec{K}_5 \cdot \vec{Q}_3$	$\vec{K}_5 \cdot \vec{Q}_4$	$\vec{K}_5 \cdot \vec{Q}_5$	$\vec{K}_5 \cdot \vec{Q}_6$	$\vec{K}_5 \cdot \vec{Q}_7$
the	$\vec{E}_6 \rightarrow \vec{E}_6 \xrightarrow{W_k} \vec{K}_6$	$\vec{K}_6 \cdot \vec{Q}_1$	$\vec{K}_6 \cdot \vec{Q}_2$	$\vec{K}_6 \cdot \vec{Q}_3$	$\vec{K}_6 \cdot \vec{Q}_4$	$\vec{K}_6 \cdot \vec{Q}_5$	$\vec{K}_6 \cdot \vec{Q}_6$	$\vec{K}_6 \cdot \vec{Q}_7$
verdant	$\vec{E}_7 \rightarrow \vec{E}_7 \xrightarrow{W_k} \vec{K}_7$	$\vec{K}_7 \cdot \vec{Q}_1$	$\vec{K}_7 \cdot \vec{Q}_2$	$\vec{K}_7 \cdot \vec{Q}_3$	$\vec{K}_7 \cdot \vec{Q}_4$	$\vec{K}_7 \cdot \vec{Q}_5$	$\vec{K}_7 \cdot \vec{Q}_6$	$\vec{K}_7 \cdot \vec{Q}_7$
forest	$\vec{E}_8 \rightarrow \vec{E}_8 \xrightarrow{W_k} \vec{K}_8$	$\vec{K}_8 \cdot \vec{Q}_1$	$\vec{K}_8 \cdot \vec{Q}_2$	$\vec{K}_8 \cdot \vec{Q}_3$	$\vec{K}_8 \cdot \vec{Q}_4$	$\vec{K}_8 \cdot \vec{Q}_5$	$\vec{K}_8 \cdot \vec{Q}_6$	$\vec{K}_8 \cdot \vec{Q}_7$

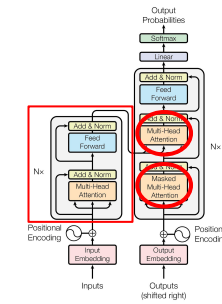
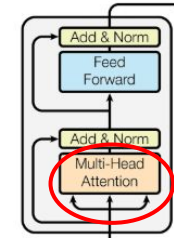
Self-attention mechanism

- Attention possui três componentes: Q, K e V
 - (Q)uery
 - (K)ey
 - (V)alue: Contém a informação “real” que será propagada após a atenção ser calculada



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

Self-attention mechanism



- Attention possui três componentes: Q, K e V
 - (Q)uery
 - (K)ey
 - (V)alue: Contém a informação “real” que será propagada após a atenção ser calculada

É responsável por carregar a informação do valor real (embeddings originais), que foi “perdida” pela atenção em QK

A informação dos embeddings E_i é multiplicada por cada vetor V_i , contendo seu “peso de origem”

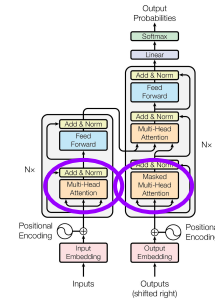
Estas alterações são somadas aos vetores originais

	a	fluffy	blue	creature	roamed	the	verdant	forest	
	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8	
a $\rightarrow \vec{E}_1 \xrightarrow{W_V} \vec{V}_1$	1.00 \vec{V}_1	0.00 \vec{V}_1	0.00 \vec{V}_1	0.00 \vec{V}_1	0.00 \vec{V}_1	0.00 \vec{V}_1	0.00 \vec{V}_1	0.00 \vec{V}_1	
fluffy $\rightarrow \vec{E}_2 \xrightarrow{W_V} \vec{V}_2$	0.00 \vec{V}_2	1.00 \vec{V}_2	0.00 \vec{V}_2	0.42 \vec{V}_2	0.00 \vec{V}_2	0.00 \vec{V}_2	0.00 \vec{V}_2	0.00 \vec{V}_2	
blue $\rightarrow \vec{E}_3 \xrightarrow{W_V} \vec{V}_3$	0.00 \vec{V}_3	0.00 \vec{V}_3	1.00 \vec{V}_3	0.58 \vec{V}_3	0.00 \vec{V}_3	0.00 \vec{V}_3	0.00 \vec{V}_3	0.00 \vec{V}_3	
creature $\rightarrow \vec{E}_4 \xrightarrow{W_V} \vec{V}_4$	0.00 \vec{V}_4	0.00 \vec{V}_4	0.00 \vec{V}_4	0.00 \vec{V}_4	0.00 \vec{V}_4	0.00 \vec{V}_4	0.00 \vec{V}_4	0.00 \vec{V}_4	
roamed $\rightarrow \vec{E}_5 \xrightarrow{W_V} \vec{V}_5$	0.00 \vec{V}_5	0.00 \vec{V}_5	0.00 \vec{V}_5	0.00 \vec{V}_5	0.01 \vec{V}_5	0.00 \vec{V}_5	0.00 \vec{V}_5	0.00 \vec{V}_5	
the $\rightarrow \vec{E}_6 \xrightarrow{W_V} \vec{V}_6$	0.00 \vec{V}_6	0.00 \vec{V}_6	0.00 \vec{V}_6	0.00 \vec{V}_6	0.99 \vec{V}_6	1.00 \vec{V}_6	0.00 \vec{V}_6	0.00 \vec{V}_6	
verdant $\rightarrow \vec{E}_7 \xrightarrow{W_V} \vec{V}_7$	0.00 \vec{V}_7	0.00 \vec{V}_7	0.00 \vec{V}_7	0.00 \vec{V}_7	0.00 \vec{V}_7	0.00 \vec{V}_7	1.00 \vec{V}_7	1.00 \vec{V}_7	
forest $\rightarrow \vec{E}_8 \xrightarrow{W_V} \vec{V}_8$	0.00 \vec{V}_8	0.00 \vec{V}_8	0.00 \vec{V}_8	0.00 \vec{V}_8	0.00 \vec{V}_8	0.00 \vec{V}_8	0.00 \vec{V}_8	0.00 \vec{V}_8	
	$\Delta \vec{E}_1$	$\Delta \vec{E}_2$	$\Delta \vec{E}_3$	$\Delta \vec{E}_4$	$\Delta \vec{E}_5$	$\Delta \vec{E}_6$	$\Delta \vec{E}_7$	$\Delta \vec{E}_8$	

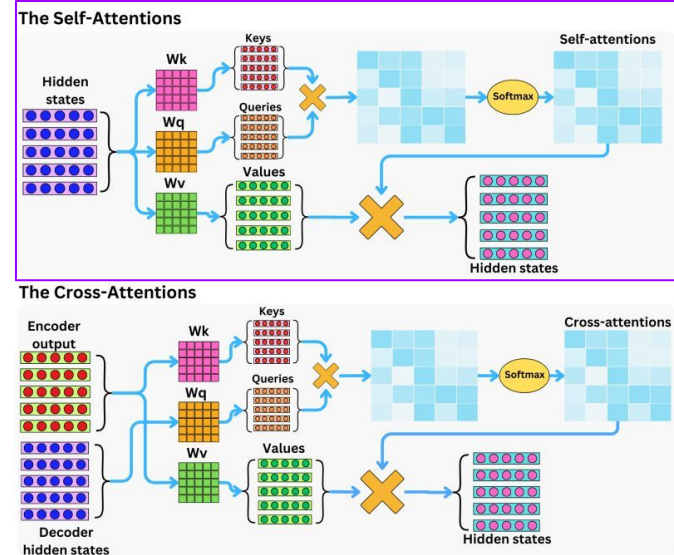
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{K^T Q}{\sqrt{d_k}}\right) V$$

Self-attention vs. cross-attention

- Self-attention:
 - Captura relações internas do texto, aprendendo dependências sintáticas e semânticas dentro da mesma frase
 - Q, K e V vêm da mesma sequência (texto de entrada)



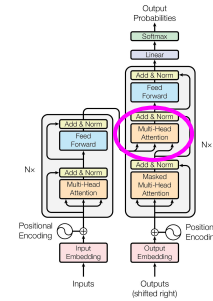
Self-Attention VS Cross-Attention TheAiEdge.io



https://www.linkedin.com/posts/damienbenveniste_what-is-the-difference-between-self-attention-activity-7211029906166624257-m0Wn

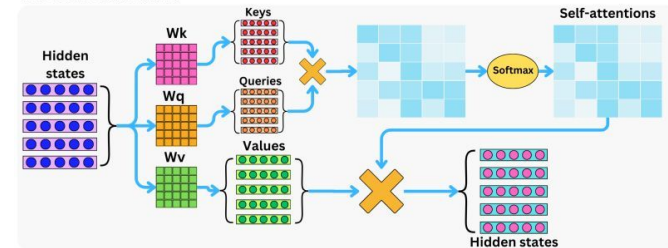
Self-attention vs. cross-attention

- **Self-attention:**
 - Captura relações internas do texto, aprendendo dependências sintáticas e semânticas dentro da mesma frase
 - Q, K e V vêm da mesma sequência (texto de entrada)
- **Cross-attention:**
 - Permite que o Decoder foque nas partes relevantes da entrada
 - Usado na 2a camada do Decoder
 - K e V vêm do encoder
 - Q vem do decoder

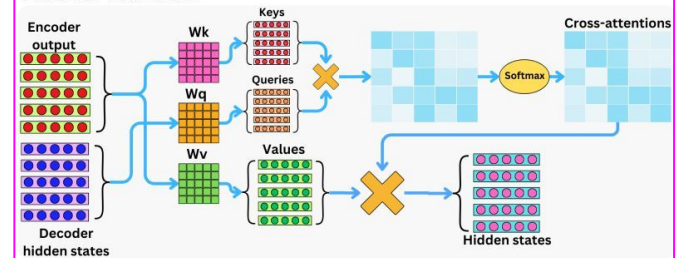


Self-Attention VS Cross-Attention TheAiEdge.io

The Self-Attentions



The Cross-Attentions



https://www.linkedin.com/posts/damienbenveniste_what-is-the-difference-between-self-attention-activity-7211029906166624257-m0Wn

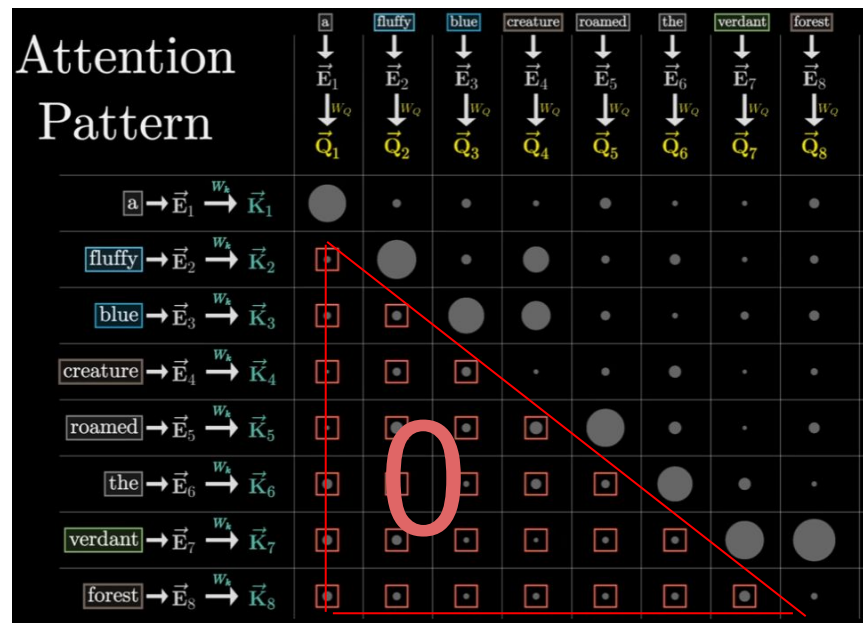
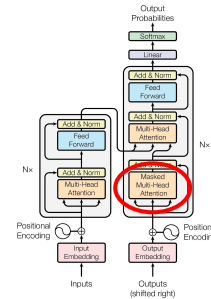
Masked attention

- Utilizado no Decoder para garantir que a geração de sequências seja autoregressiva (ou seja, o contexto só deve depender do “passado”)
- Aplica uma máscara para que parte do mecanismo de atenção de cada palavra “do futuro” não seja usado (softmax = 0)

Exemplo:

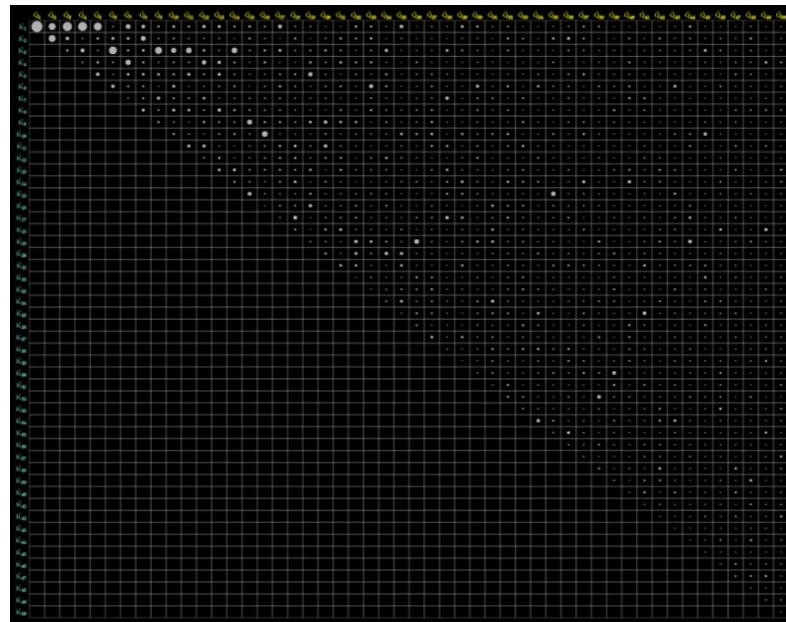
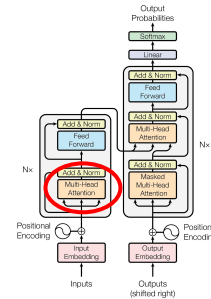
“O gato <?> feliz”.

Previsão de <?> deve levar em conta apenas “O gato”



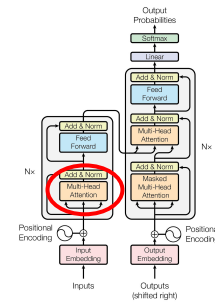
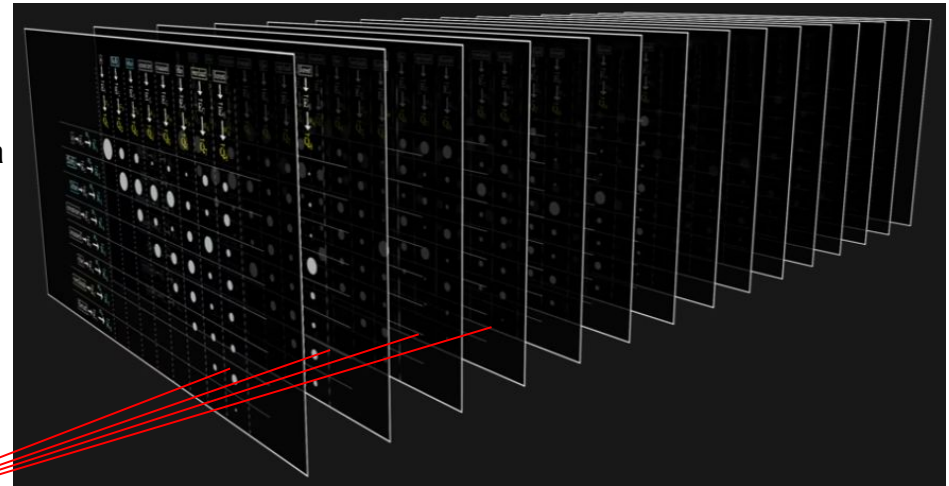
Tamanho do contexto

- Contexto é o número máximo de tokens que o modelo pode processar em uma sequência
- Tamanho da matriz de atenção é igual ao quadrado do tamanho do contexto C
- A matriz de atenção tem tamanho $C \times C$
- Se o número de tokens permitidos de entrada é 2048, a matriz é $2048 \times 2048 = 4.194.304$
- GPT-1: 512 tokens GPT-2: 1024 tokens
GPT-3: 2048 tokens GPT-4: 32k tokens



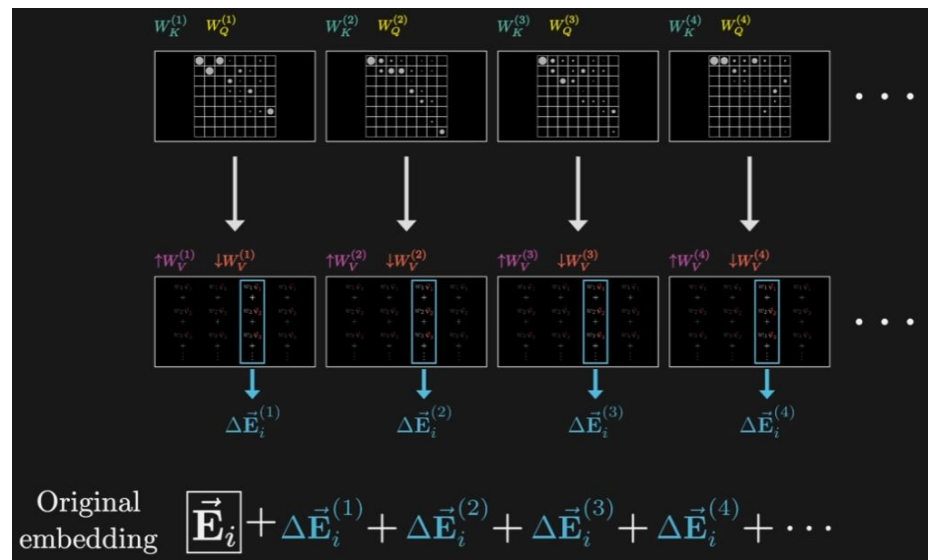
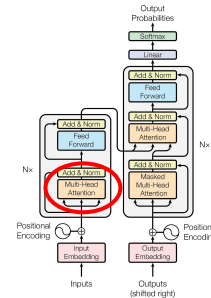
Multi headed attention

- Permite que o modelo capture diferentes tipos de relações (sintáticas, semânticas, locais/globais) em paralelo
 - Cada matriz attention possui seus próprios pesos Q, K, V
 - Ideia é capturar diferentes aspectos focando em relações não óbvias (uma para sujeito-verbo, outra para adjetivo-substantivo, outra para longo alcance, etc.)



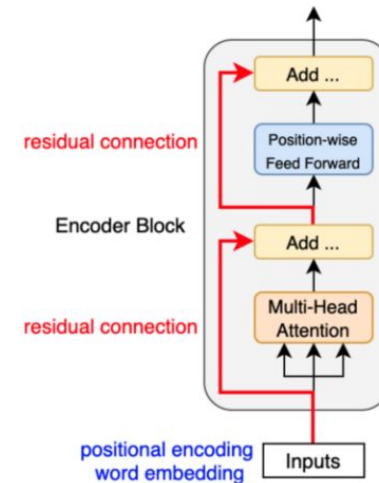
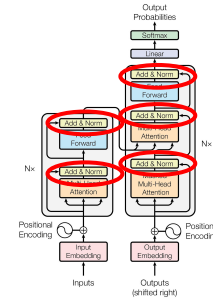
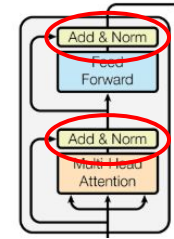
Multi headed attention

- Permite que o modelo capture diferentes tipos de relações (sintáticas, semânticas, locais/globais) em paralelo
 - Cada matriz attention possui seus próprios pesos Q, K, V
 - Ideia é capturar diferentes aspectos focando em relações não óbvias (uma para sujeito-verbo, outra para adjetivo-substantivo, outra para longo alcance, etc.)
- Cada embedding “novo” de cada token de entrada é dado pela soma de todos os valores encontrados em cada cabeça



Residual connections

- Atalhos em uma rede neural que **conectam** a saída de uma camada diretamente à saída de uma camada mais à frente, "**saltando**" uma ou mais camadas intermediárias
- Ajudam a mitigar o problema do *vanishing gradient* (valores próximos de zero)
 - Preservam o sinal durante a retropropagação
 - Garantem que a representação contextual dos tokens mantenha uma conexão com seu estado original
- São agregados com soma e normalização (add & norm)

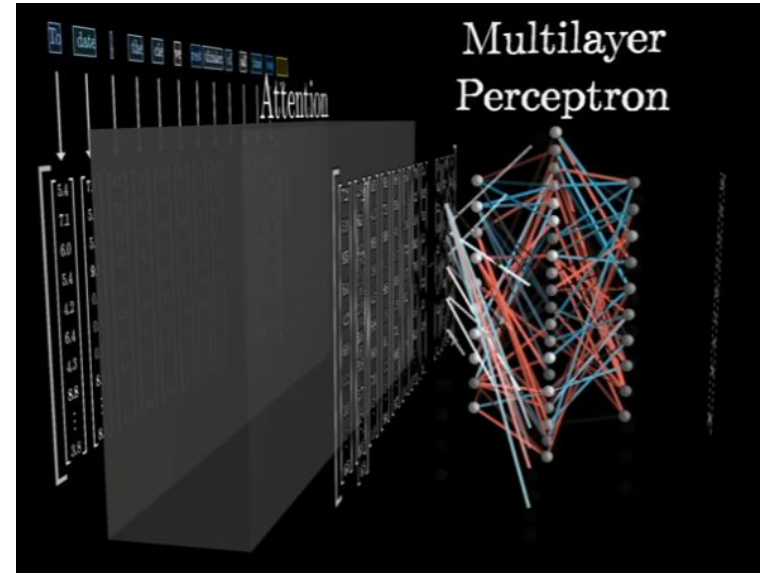
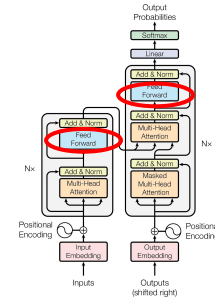
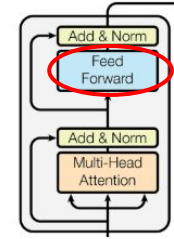


Mais detalhes em: <https://stats.stackexchange.com/questions/565196/why-are-residual-connections-needed-in-transformer-architectures>

Feedforward

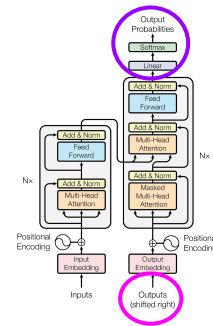
- Rede neural fully-connected (MLP) “padrão” de deep learning
- Aplicada independentemente a cada posição (token) da sequência (não considera os vizinhos)
- Utiliza função de ativação ReLU

$$\text{ReLU} \max(0, x)$$



Linear + softmax + output

- Linear: camada que transforma o embedding de cada token um valor (número) chamado de **logit**
- Cada logit não tem um significado direto isolado, mas quando comparado com os demais, quanto maior o logit, maior a chance daquela palavra ser escolhida
- Os logits são transformados em valores no intervalo $[0,1]$ pela função softmax, indicando “probabilidades” (output probabilities) **[Output probabilities]**
- Encoder é aplicado 1x e decoder Nx (a cada vez a entrada no outputs é deslocada) **[Output shifted right]**



$$\text{output} = x \cdot W^T + b$$

Exemplo:

Vocabulário $|V| = 10.000$ tokens

Embedding $|E_i| = 512$

Matriz de embeddings: $E_{\text{input}} \in \mathbb{R}^{10.000 \times 512}$

Matriz de pesos: $W \in \mathbb{R}^{10.000 \times 512}$

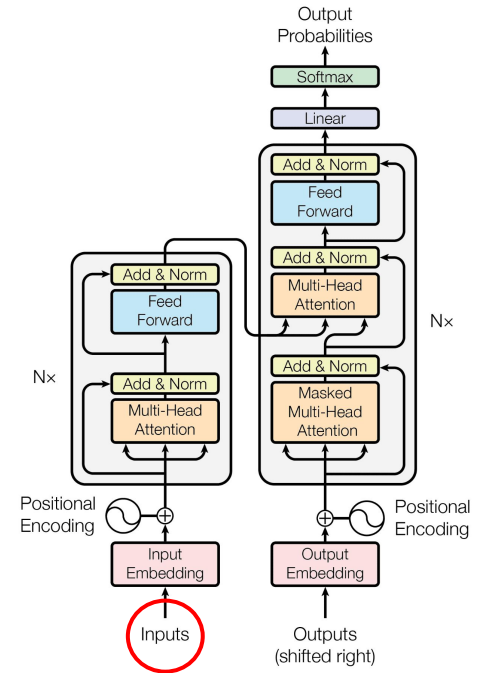
Transposta!

Logits: $\text{logits} \in \mathbb{R}^{10.000}$

Aleatória e ajustada por backpropagation

Exemplo (tradução automática)

- Para cada “frase” (texto):
 - Encoder:
 - Input: I love books



Exemplo (tradução automática)

- Para cada “frase” (texto):

- Encoder:

- Input: I love books

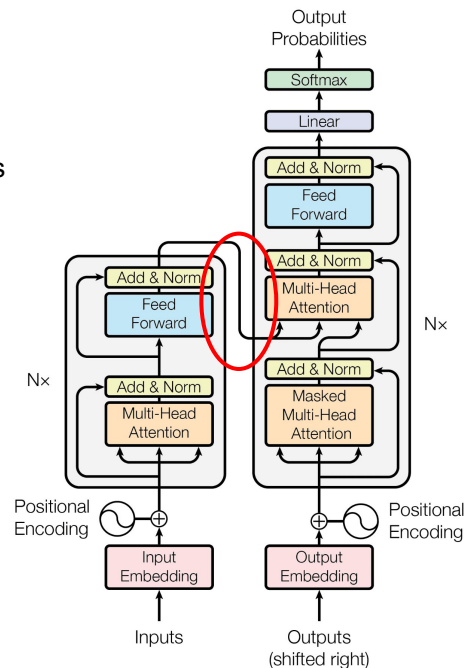
- Output:

- I: [0.1, 1.3, 6.4, -4.5 ... 3.2] (dimensão 512)

- love: [0.4, 5.3, 0.4, -1.5 ... 7.1] (dimensão 512)

- books: [1.0, 2.3, 1.4, -7.5 ... 4.2] (dimensão 512)

Este passo ocorre uma única vez antes do “decoder”



Exemplo (tradução automática)

- Para cada “frase” (texto):

- Encoder:

- Input: I love books

- Output:

- I: [0.1, 1.3, 6.4, -4.5 ... 3.2] (dimensão 512)

- love: [0.4, 5.3, 0.4, -1.5 ... 7.1] (dimensão 512)

- books: [1.0, 2.3, 1.4, -7.5 ... 4.2] (dimensão 512)

- Decoder ($t=1$):

- Input: [$\langle s \rangle$]

- Output:

- Eu: 0.1

- amo: 0.6

- livros: 0.2

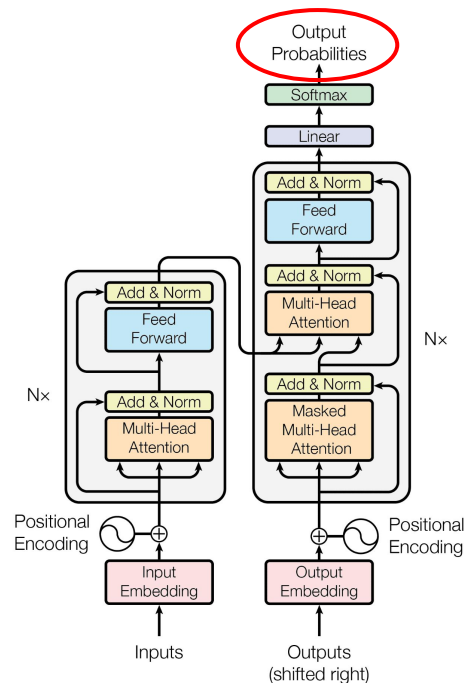
- ...

Objetivo é prever o próximo após $\langle s \rangle$, ou seja, Eu

Errou!

Cálculo da loss (o correto era Eu com 1.0)

Loss(t_1) = 0.9



Exemplo (tradução automática)

- Para cada “frase” (texto):

- Encoder:

- Input: I love books

- Output:

- I: [0.1, 1.3, 6.4, -4.5 ... 3.2] (dimensão 512)

- love: [0.4, 5.3, 0.4, -1.5 ... 7.1] (dimensão 512)

- books: [1.0, 2.3, 1.4, -7.5 ... 4.2] (dimensão 512)

- Decoder (**t=2**):

- Input: [<s>, Eu]

- Output:

- Eu: 0.1

- amo: 0.2**

- livros: 0.3

- ...

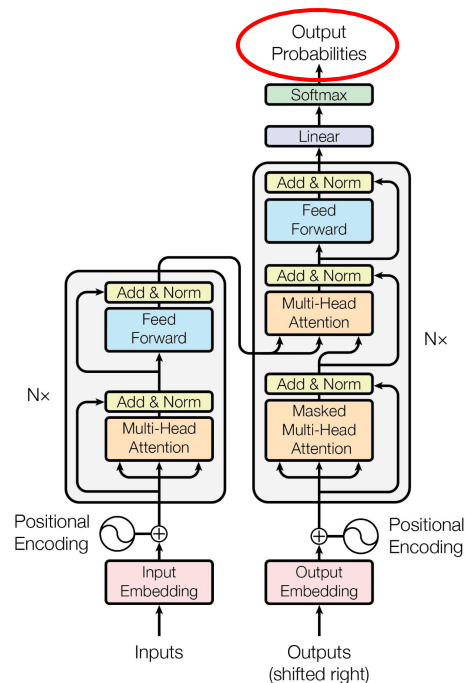
Objetivo é prever o próximo após Eu, ou seja, amo

Errou!

Cálculo da loss (o correto era amo com 1.0)

Loss(t1) = 0.9

Loss(t2) = 0.8



Exemplo (tradução automática)

- Para cada “frase” (texto):

- Encoder:

- Input: I love books

- Output:

- I: [0.1, 1.3, 6.4, -4.5 ... 3.2] (dimensão 512)

- love: [0.4, 5.3, 0.4, -1.5 ... 7.1] (dimensão 512)

- books: [1.0, 2.3, 1.4, -7.5 ... 4.2] (dimensão 512)

- Decoder ($t=T$):

- Input: [$\langle s \rangle$, Eu]

- Output:

- Eu: 0.1

- amo: 0.2

- livros: 0.3

- ...

Errou!

Cálculo da loss (o correto era amo com 1.0)

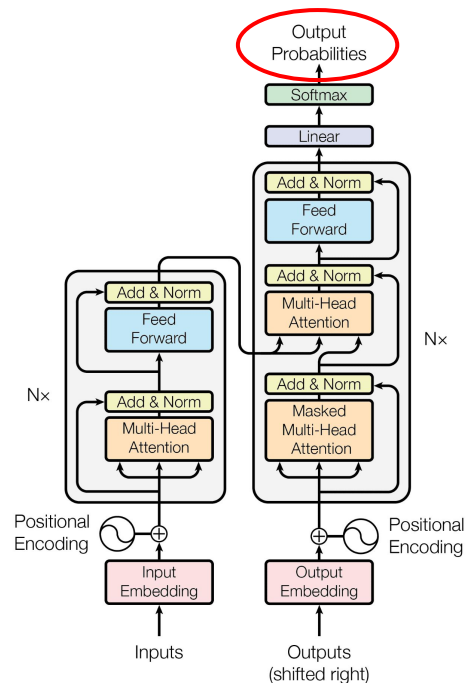
Loss(t_1) = 0.9

Loss(t_2) = 0.8

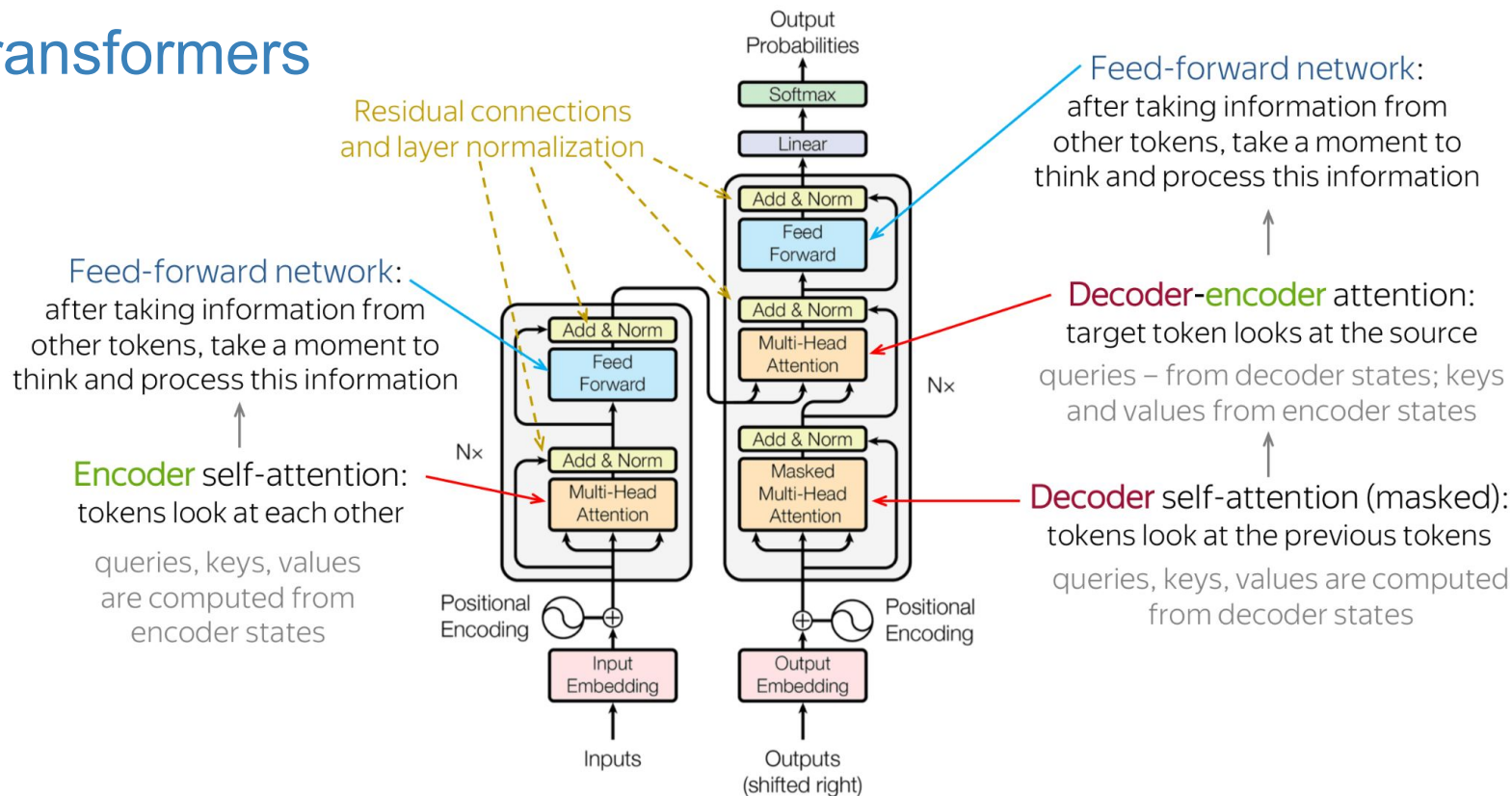
...

Loss total (soma ou média por token)

Backpropagation: calcula o gradiente da loss total, propaga os erros por todas as camadas



Transformers



Referências

- Understanding Transformers
 - <https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/>
- The illustrated Transformer
 - <https://jalammar.github.io/illustrated-transformer/>
- Canal StatQuest (YouTube): <https://www.youtube.com/@statquest>
- Canal 3Blue1Brown (YouTube): <https://www.youtube.com/@3blue1brown>

Próximas aulas

- Large Language Models [1]
 - Encoder-only (de compreensão - NLU)

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Departamento de Informática Aplicada

Obrigado pela atenção!
Dúvidas?

Prof. Dennis Giovani Balreira

(Material adaptado da Profa. Viviane Moreira e dos canais StatQuest e 3Blue1Brown)



INF01221 - Tópicos Especiais em Computação XXXVI:
Processamento de Linguagem Natural

