

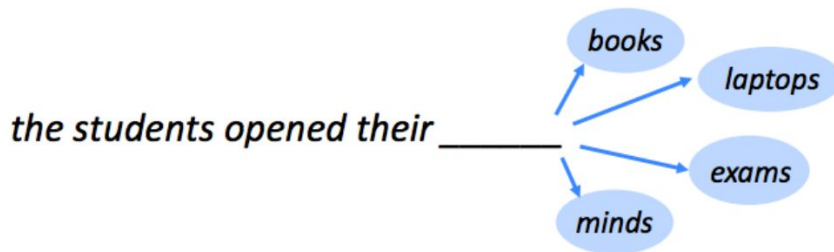
Processamento de Linguagem Natural

Modelos de Linguagem

Prof. Luciano Barbosa &
Prof. Johny Moreira
{luciano, jms5}@cin.ufpe.br

Objetivo Principal

- ❖ Tarefa de prever próximas palavras



- ❖ Computar a probabilidade da próxima palavra dada uma sequência

$$P(W_5 | W_1, W_2, W_3, W_4)$$

- ❖ Computar a probabilidade de uma sequência de palavras

$$P(W_1, W_2, W_3, W_4, W_5 \dots W_n)$$

Aplicação

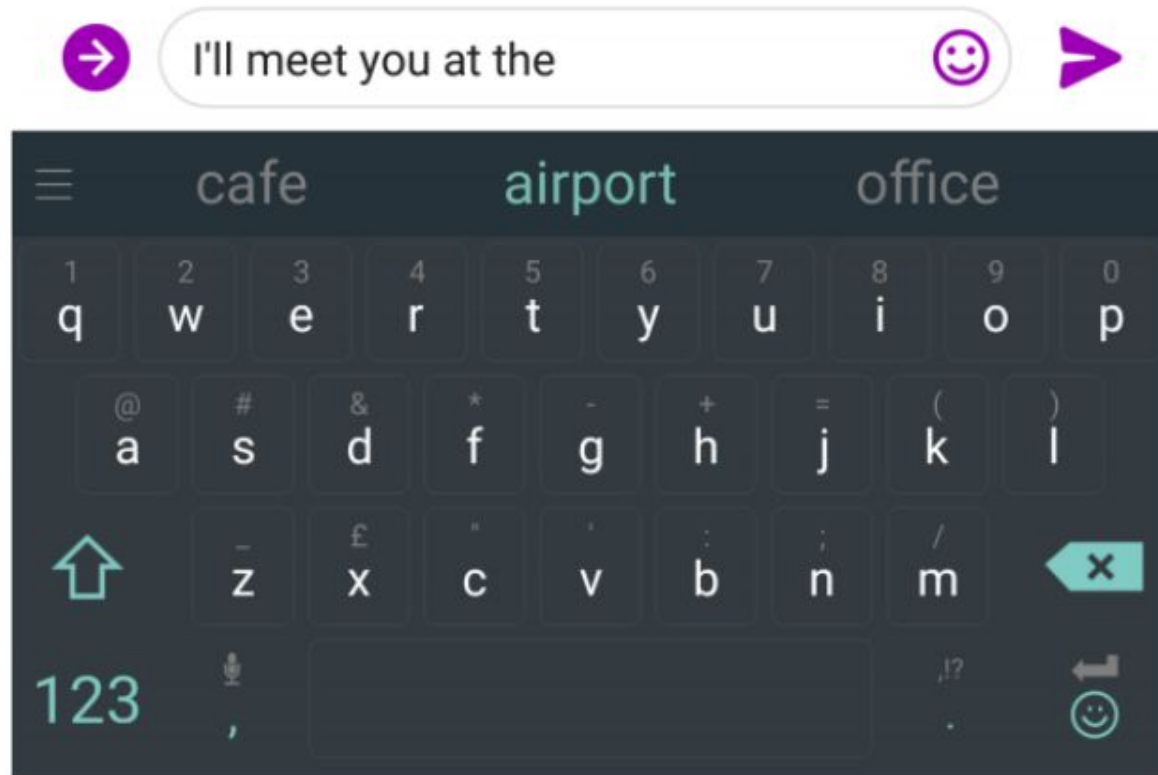


what is the |



what is the **weather**
what is the **meaning of life**
what is the **dark web**
what is the **xfl**
what is the **doomsday clock**
what is the **weather today**
what is the **keto diet**
what is the **american dream**
what is the **speed of light**
what is the **bill of rights**

Aplicação



Aplicação

- ❖ Gerar texto ou estimar a probabilidade de um texto
- ❖ Componente de várias tarefas de PLN
 - Corretor ortográfico
 - Reconhecedor de fala
 - Machine translation
 - Reconhecimento de escrita
 - Sumarização
 - Diálogo

Computando as Probabilidades

Ex: $P(\text{"its water is so transparent"})$

$$p(B|A) = \frac{P(A,B)}{P(A)} \longrightarrow P(A,B) = P(A) \times P(B|A)$$

$$P(A, B, C, D) = P(A) \times P(B|A) \times P(C|A,B) \times P(D|A,B,C)$$

$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1) \times P(x_2|x_1) \times P(x_3|x_1, x_2) \times P(x_n|x_1, x_2, x_3, \dots, x_{n-1})$$

$$\begin{aligned} P(\text{"its water is so transparent"}) &= P(\text{its, water, is, so, transparent}) = \\ &P(\text{its}) \times P(\text{water} | \text{its}) \times P(\text{is} | \text{its, water}) \times \\ &\quad P(\text{so} | \text{its, water, is}) \times P(\text{transparent} | \text{its, water, is, so}) \end{aligned}$$

Calculando as probabilidades

Baseado na frequência em um corpus de dados

$$P(\text{the | its water is so transparent that}) = \frac{\text{Count}(\text{its water is so transparent that the})}{\text{Count}(\text{its water is so transparent that})}$$

Problema: n-grams grandes são raros

N-grams: sequência de n palavras consecutivas

Calculando as probabilidades: Markov Assumption

Utilizando somente as k palavras mais próximas

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-k} \dots w_{i-1})$$

Utilizando bigramas (a palavra anterior)

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-1})$$

N-gram: trigrams, 4-grams, 5-grams

– Linguagem tem dependências de longa distância

Ex: *“The computer which I had just put into the machine room on the fifth floor crashed”*

Calculando as probabilidades:

Maximum Likelihood Estimate (Estimativa de Probabilidade Máxima)

$$p(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>

<s> Sam I am</s>

<s> I do not like green eggs and ham</s>

$$P(I | <s>) = \frac{2}{3} = .67$$

$$P(\text{Sam} | <s>) = \frac{1}{3} = .33$$

$$P(\text{am} | I) = \frac{2}{3} = .67$$

$$P(</s> | \text{Sam}) = \frac{1}{2} = 0.5$$

$$P(\text{Sam} | \text{am}) = \frac{1}{2} = .5$$

$$P(\text{do} | I) = \frac{1}{3} = .33$$

Calculando as probabilidades:

$$P(<s> \text{ I want english food } </s>) = P(I \mid <s>) \times P(\text{want} \mid I) \times \\ P(\text{english} \mid \text{want}) \times P(\text{food} \mid \text{english}) \times \\ P(</s> \mid \text{food})$$

$$P(<s> \text{ I want english food } </s>) = .000031$$

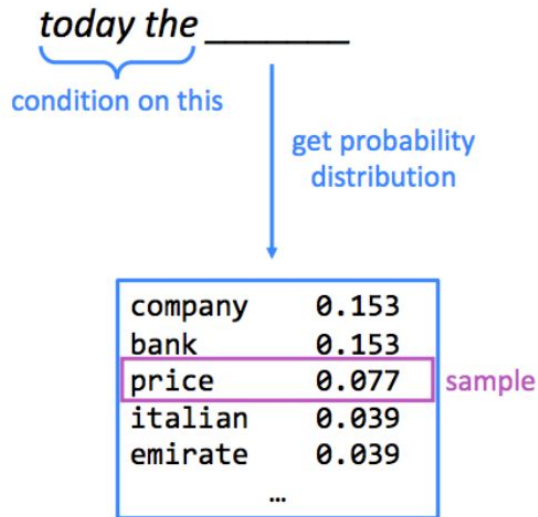
Cálculo em log

- Evitar overflow
- Adicionar é mais rápido que multiplicar

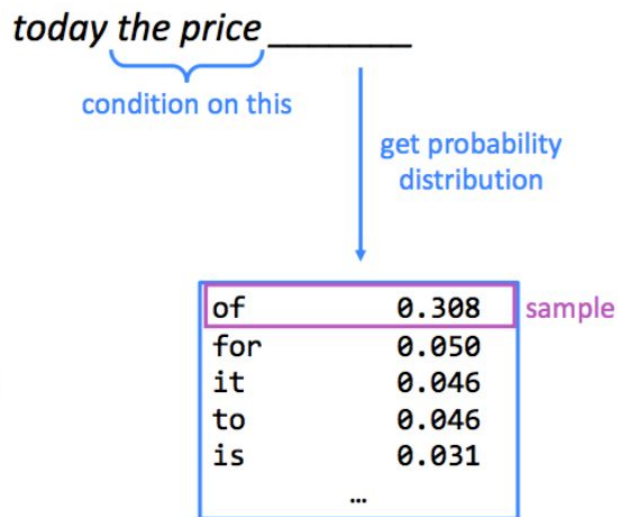
$$\log(p_1, p_2, p_3, p_4) = \log p_1 + \log p_2 + \\ \log p_3 + \log p_4$$

Gerando Sentenças

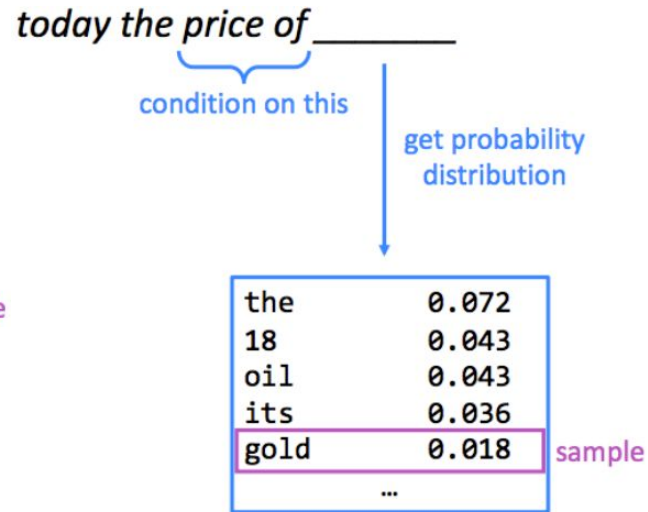
I



II

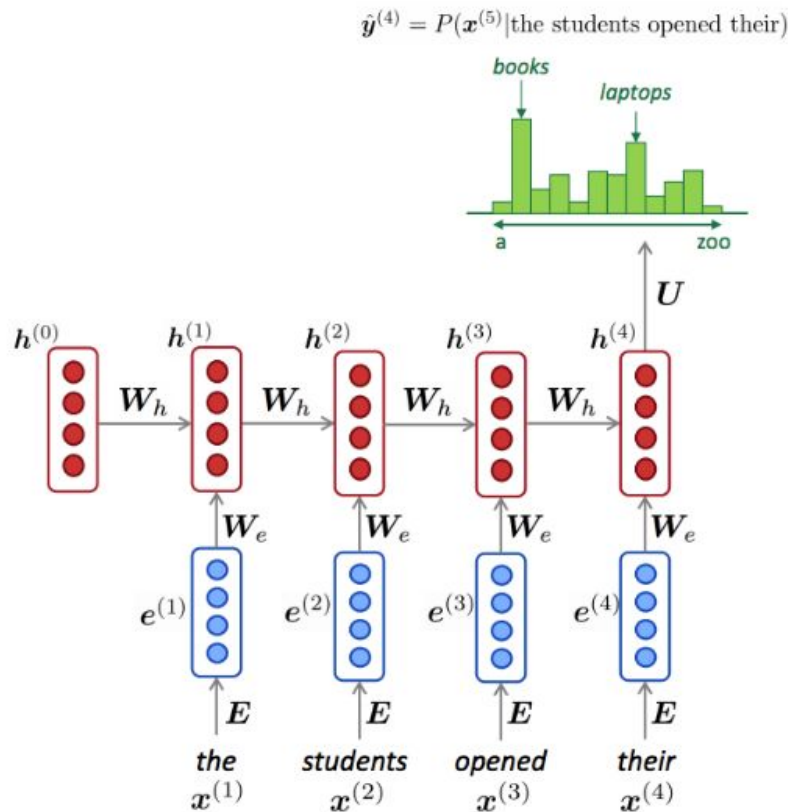


III



Modelos Neurais de Linguagem

Recurrent Neural Networks (RNN)



output distribution

$$\hat{y}^{(t)} = \text{softmax} \left(U h^{(t)} + b_2 \right) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma \left(W_h h^{(t-1)} + W_e e^{(t)} + b_1 \right)$$

$h^{(0)}$ is the initial hidden state

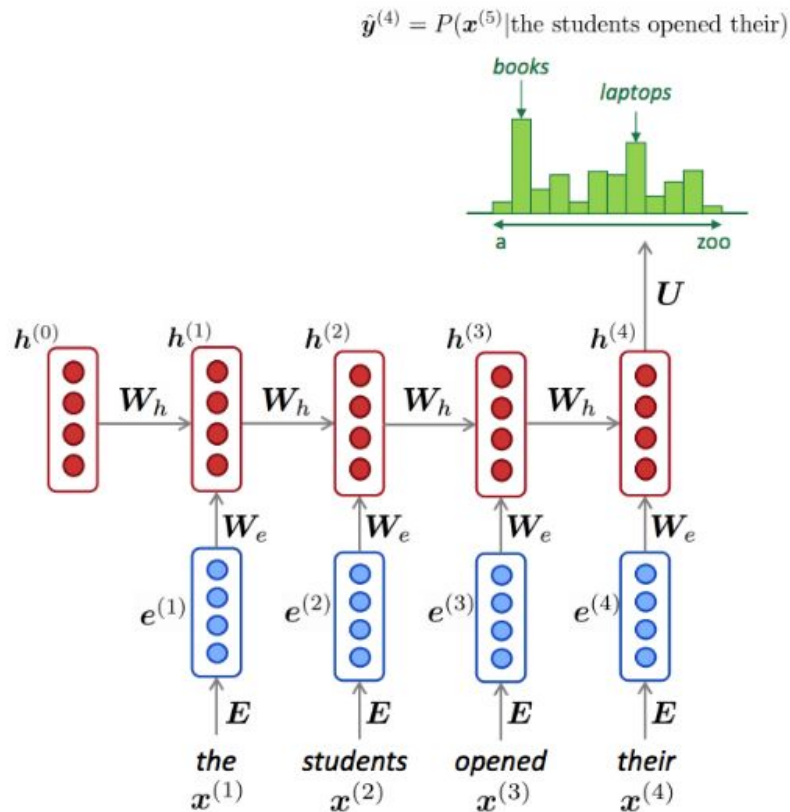
word embeddings

$$e^{(t)} = E x^{(t)}$$

words / one-hot vectors

$$x^{(t)} \in \mathbb{R}^{|V|}$$

Recurrent Neural Networks (RNN)



- ❖ **Vantagens**
 - Pode processar sequência de qualquer tamanho
 - Modelo não aumenta com o tamanho da sequência
 - Usa informação anterior
 - Mesmos pesos utilizados em cada passo
- ❖ **Desvantagens**
 - Lenta
 - Na prática, tem dificuldade em guardar informação de palavras muito anteriores

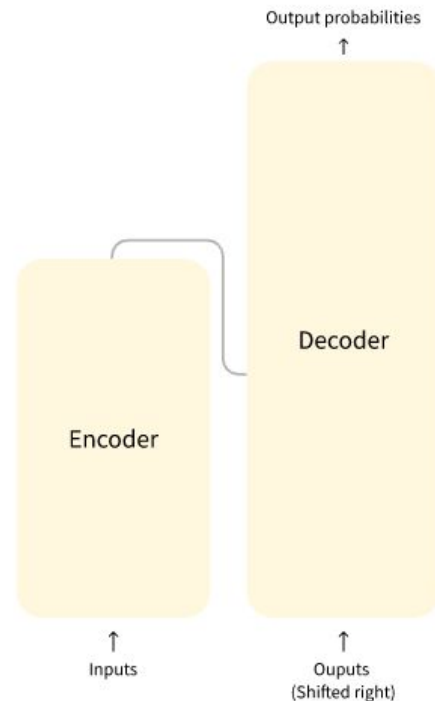
Transformers & Modelos de Linguagem Pré-treinados

Transformers

- ❖ É o estado-da-arte para o Processamento de Linguagem Natural
- ❖ É uma arquitetura de Deep Learning que visa resolver tarefas sequence-to-sequence com dependências de longo alcance
- ❖ Não usa RNN ou Convolução
- ❖ Não exige que os dados sequenciais sejam processados em ordem
- ❖ Permite maior paralelização
- ❖ Diferentes tarefas: classificação, extração, geração de texto e busca

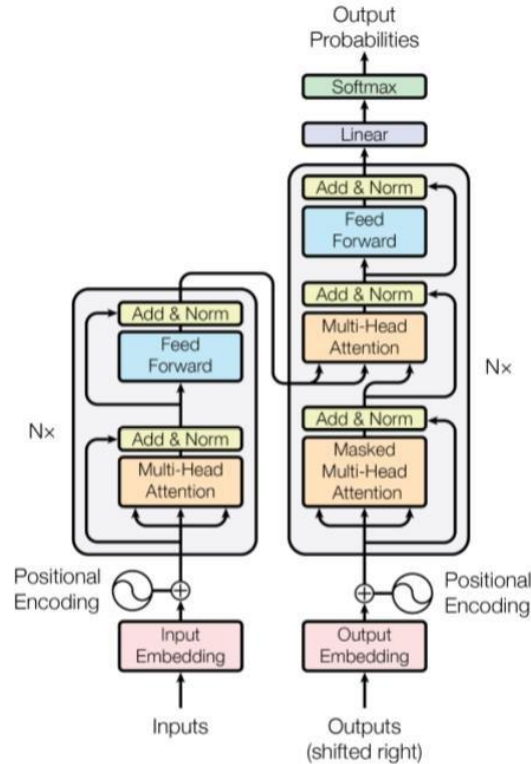
Transformers: Arquitetura

- Encoder: constrói uma representação da entrada
- Decoder: gera probabilidades baseado na saída do encoder e outras entradas



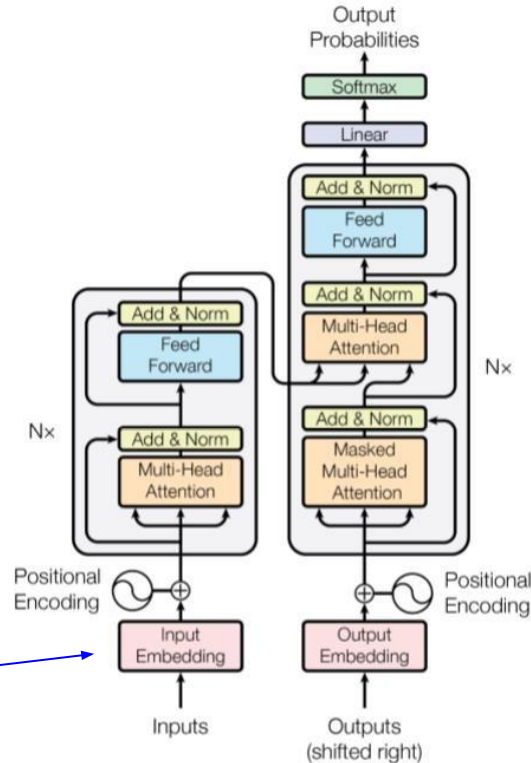
Fonte: <https://huggingface.co/course/chapter1/4>

Transformers








Transformers

Representação das palavras na
camada de entrada



Word Embedding Tradicional

- Vocabulário construído pelo conjunto de treinamento
- Palavras não presentes mapeadas para UNK

	word		vocab mapping	embedding
Common words	hat	→	pizza (index)	
	learn	→	tasty (index)	
Variations	taaaaasty	→	UNK (index)	
misspellings	laern	→	UNK (index)	
novel items	Transformerify	→	UNK (index)	

Subword Tokenization

- Crítico para línguas com muitas variações na estrutura das palavras
- Ex: Swahili

[illegible]

Subword Tokenization

- Palavras raras são quebradas em substrings
- Palavras frequentes são mantidas
- Tamanho do vocabulário é um parâmetro

Byte-pair Encoding

- Objetivo: representar o corpus com a menor quantidade de tokens
- Algoritmo:
 1. Inicia o vocabulário com todos os caracteres e símbolo “fim de palavra”
 2. Une dois tokens com maior frequência
 3. Decrementa a frequência dos dois tokens
 4. Repetir até o vocabulário desejado

BPE: Palavras no Corpus

WORD	FREQUENCY	WORD	FREQUENCY
deep </w>	3	build </w>	1
learning </w>	3	train </w>	1
the </w>	2	and </w>	1
models </w>	2	deploy </w>	1
Floydhub </w>	1	Build </w>	1
is </w>	1	models </w>	1
fastest </w>	1	in </w>	1
way </w>	1	cloud </w>	1
to </w>	1	Train </w>	1

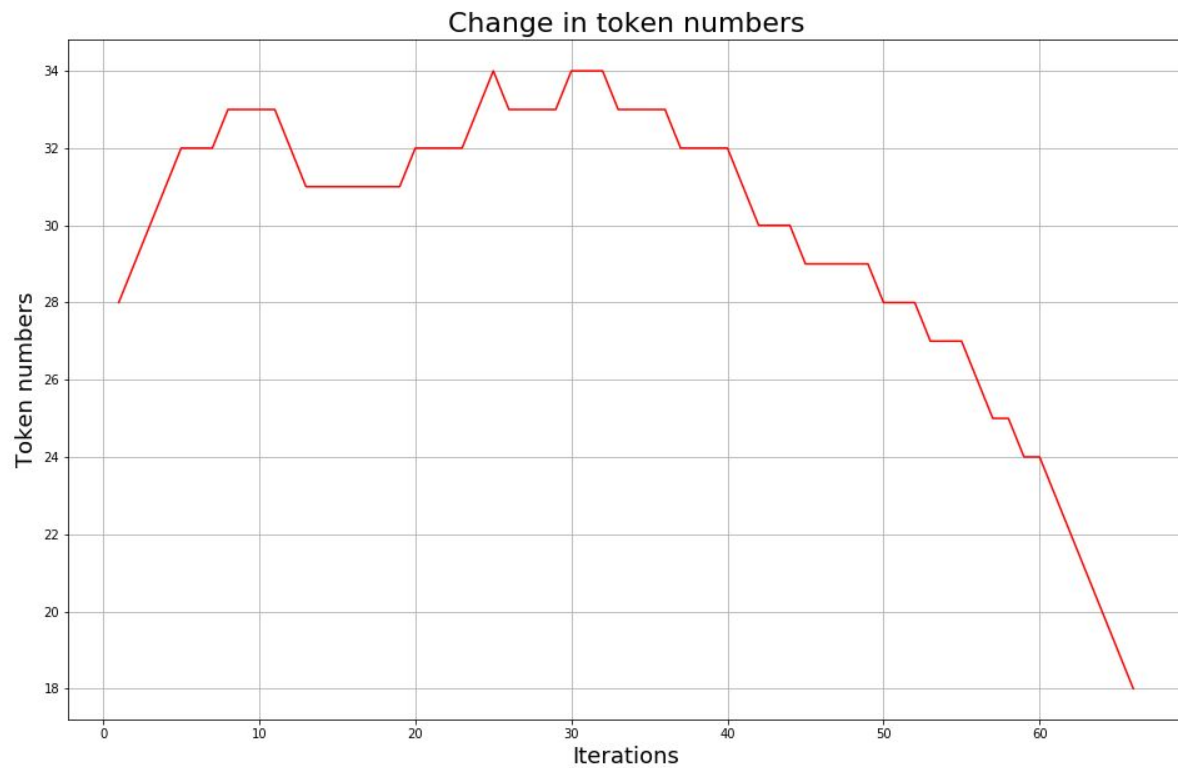
BPE: Caracteres no Corpus

NUMBER	TOKEN	FREQUENCY	NUMBER	TOKEN	FREQUENCY
1	</w>	24	15	g	3
2	e	16	16	m	3
3	d	12	17	.	3
4	l	11	18	b	2
5	n	10	19	h	2
6	i	9	20	F	1
7	a	8	21	H	1
8	o	7	22	f	1
9	s	6	23	w	1
10	t	6	24	,	1
11	r	5	25	B	1
12	u	4	26	c	1
13	p	4	27	T	1
14	y	3			

BPE: Merge

NUMBER	TOKEN	FREQUENCY	NUMBER	TOKEN	FREQUENCY
1	</w>	24	16	g	3
2	e	$16 - 7 = 9$	17	m	3
3	d	$12 - 7 = 5$	18	.	3
4	l	11	19	b	2
5	n	10	20	h	2
6	i	9	21	F	1
7	a	8	22	H	1
8	o	7	23	f	1
9	de	7	24	w	1
10	s	6	25	,	1
11	t	6	26	B	1
12	r	5	27	c	1
13	u	4	28	T	1
14	p	4			
15	y	3			

BPE



Wordpiece

- Similar ao BPE
- Algoritmo:
 1. Inicia o vocabulário com todos os caracteres e o símbolo de "fim da palavra"
 2. Une os tokens com maior score
 3. Diminui a frequência dos dois tokens
 4. Repete até o vocabulário desejado (por exemplo, tamanho de vocabulário pré-definido)

$$\text{score} = (\text{freq_of_pair}) / (\text{freq_of_first_element} \times \text{freq_of_second_element})$$

probabilidade maior de aparecer junto do que separado

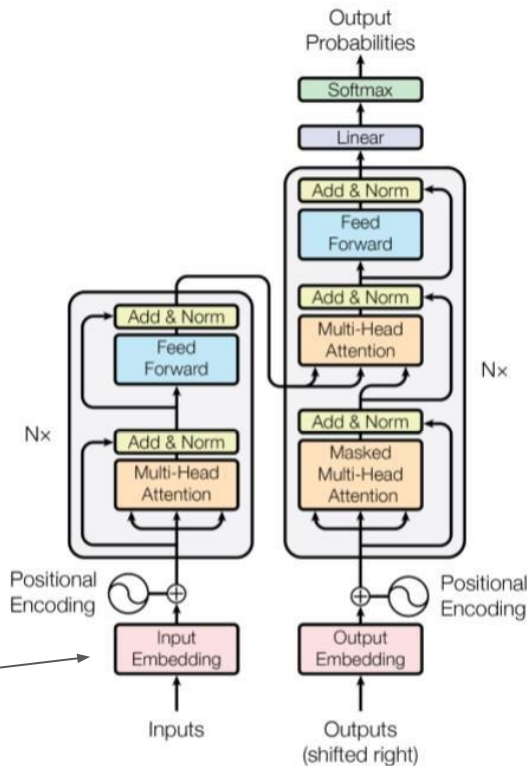
Wordpiece

Word	Token(s)
surf	['surf']
surfing	['surf', '##ing']
surfboarding	['surf', '##board', '##ing']
surfboard	['surf', '##board']
snowboard	['snow', '##board']
snowboarding	['snow', '##board', '##ing']
snow	['snow']
snowing	['snow', '##ing']

Transformers

Codificação de Posicionamento
das palavras

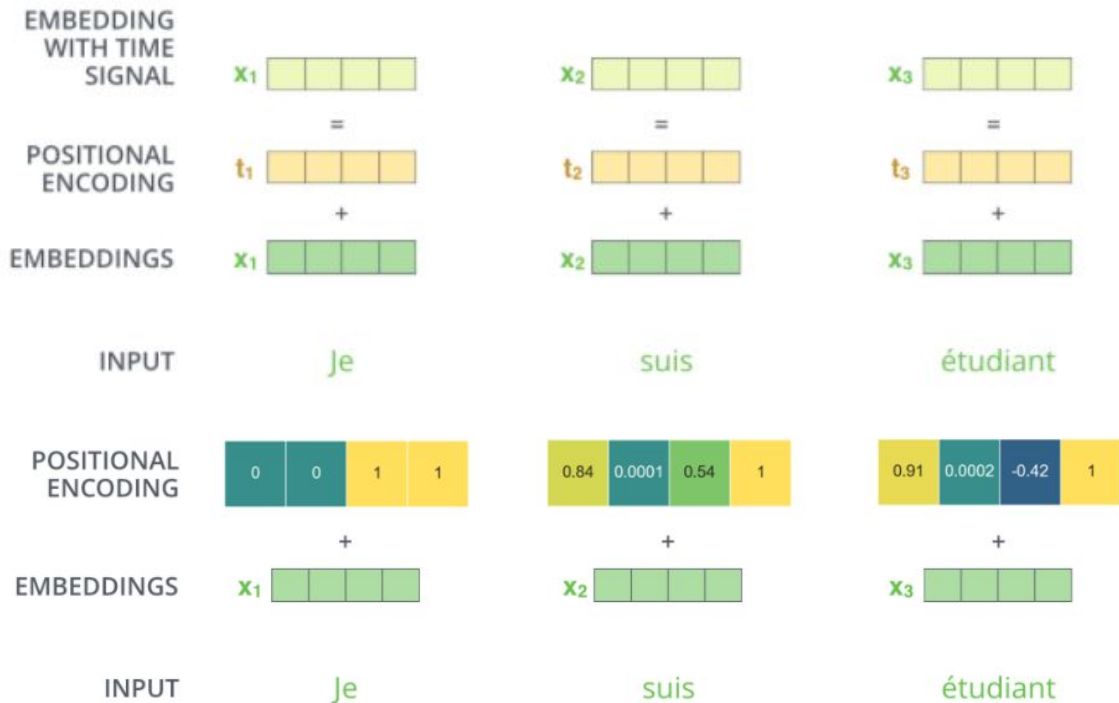
Representação das palavras na
camada de entrada



Positional Encoding

- ❖ Tokens são processados pelo Transformer de forma não ordenada
- ❖ Positional Encoding introduz ordem aos tokens que são manipulados pelo Transformer
- ❖ Assim, o modelo aprende representações diferentes para uma palavra dependendo da sua posição

Positional Encoding



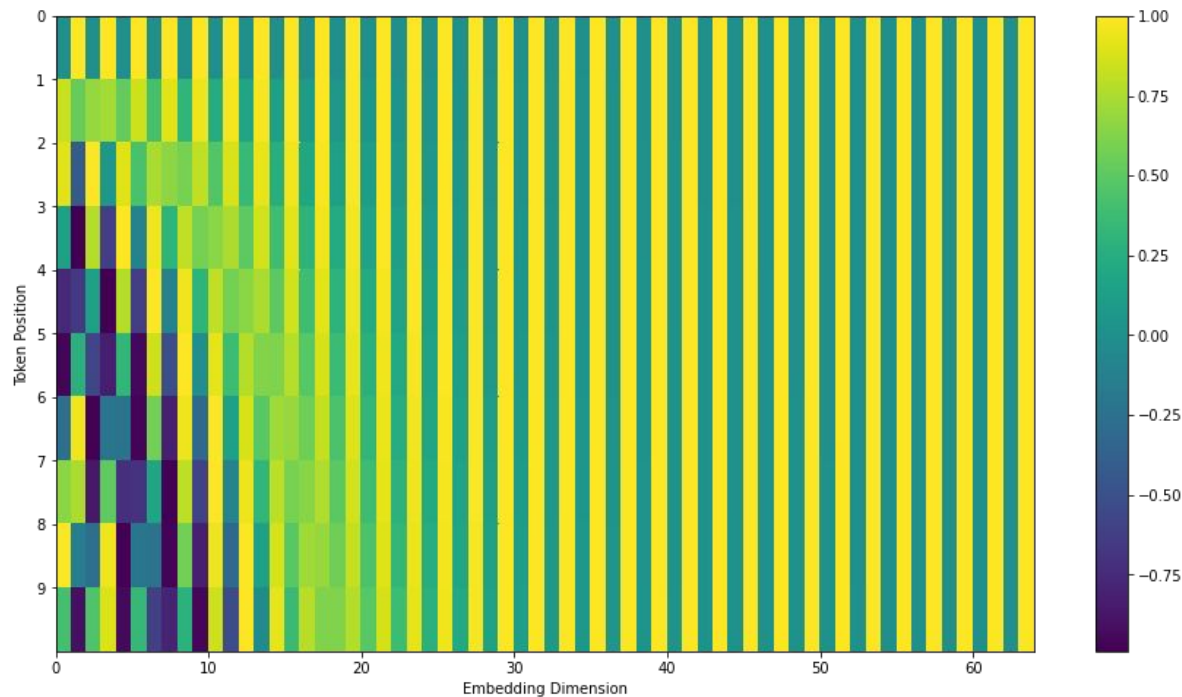
Cálculo do Posicional (Sinusoidal)

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

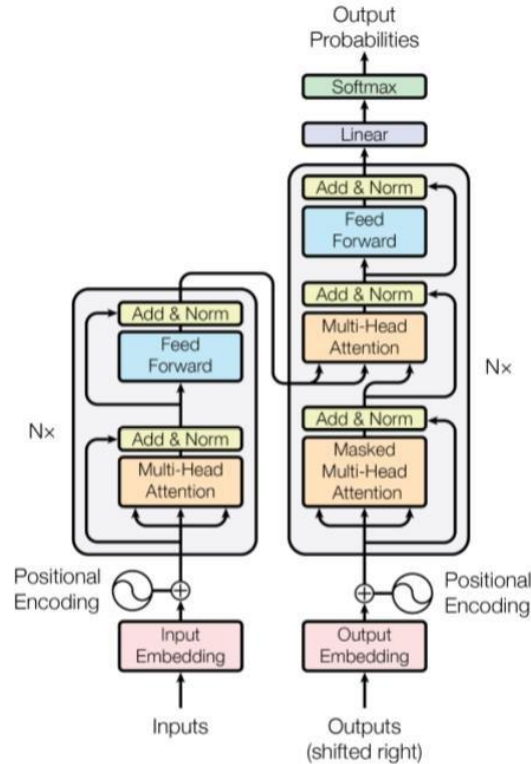
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

pos: posição do token na frase

i: dimensão no embedding

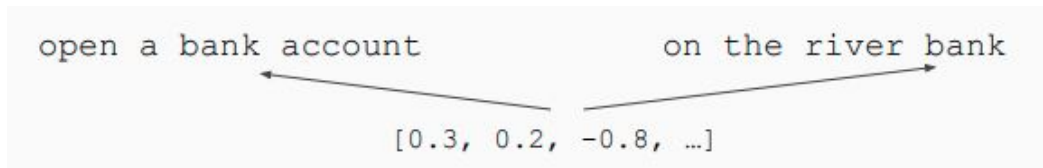


Transformers



Contextual Representations

- Limitação de word embeddings: mesma representação para significados diferentes



- Solução: aprender representações contextuais



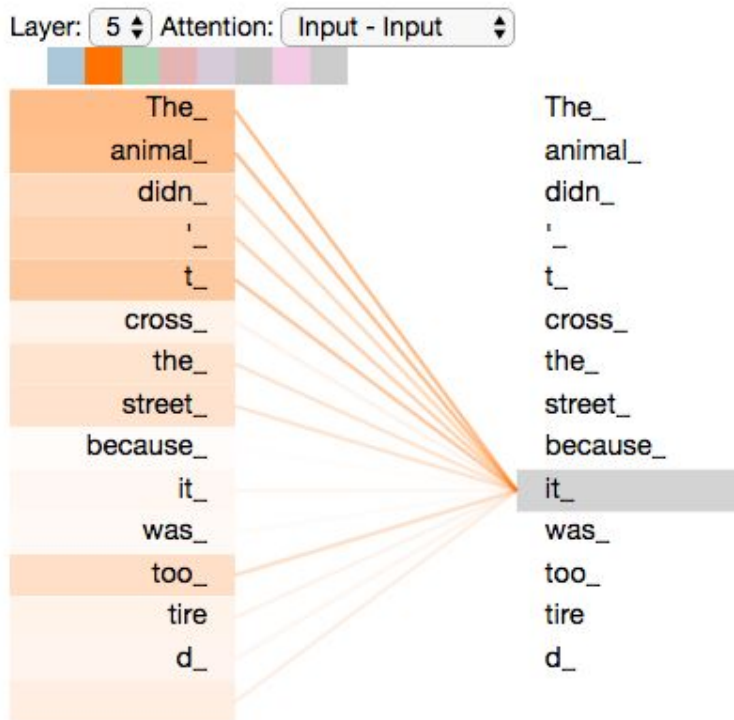
Self-Attention

“é um mecanismo de atenção que relaciona diferentes posições de uma única sequência para computar uma representação da sequência.”

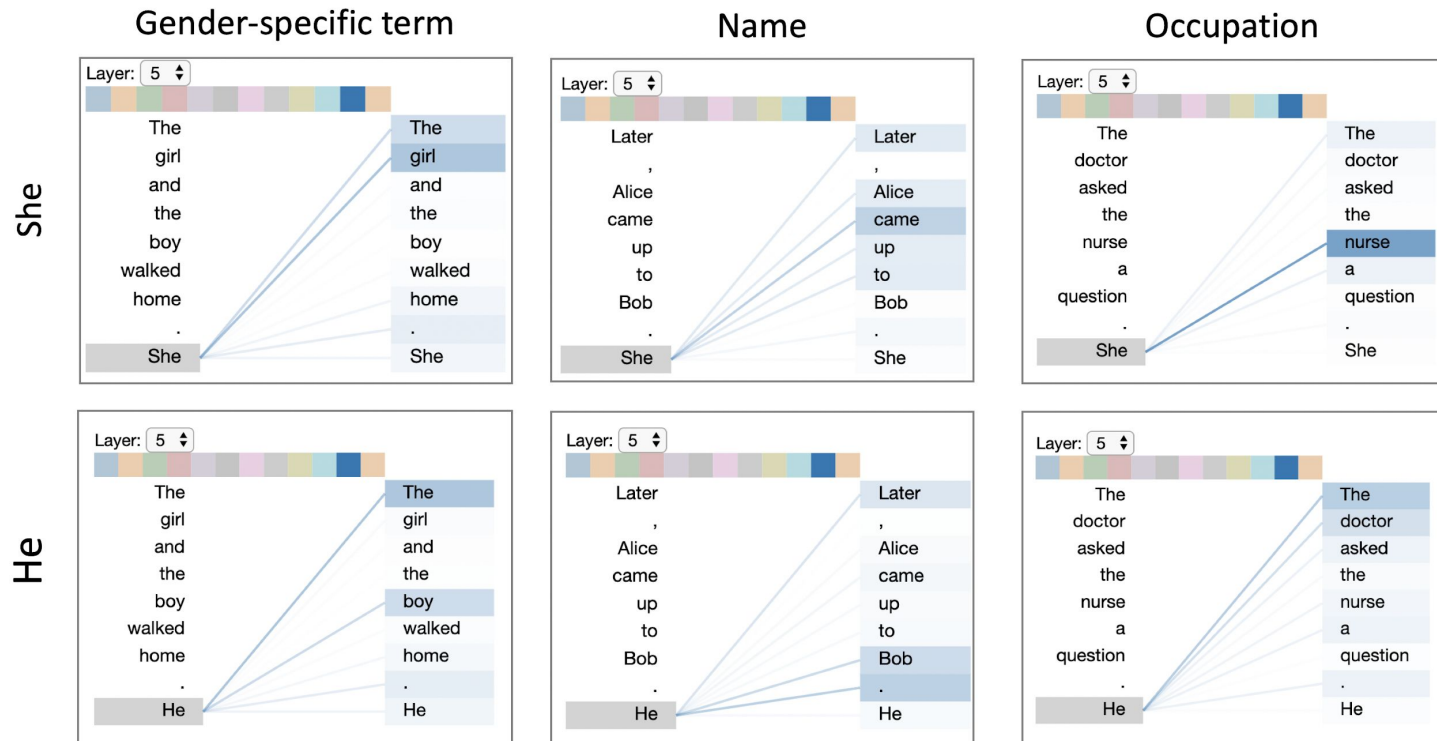
Ashish Vaswani et al., Google Brain.

Self-Attention

“The animal didn't cross
the street because **it** was
too tired”

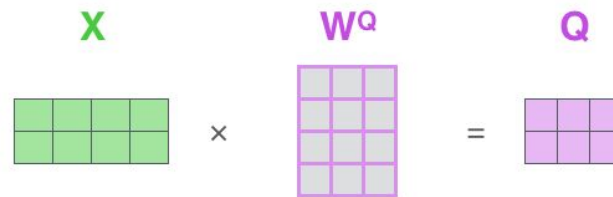


Exemplo de Self-Attention



Self-Attention

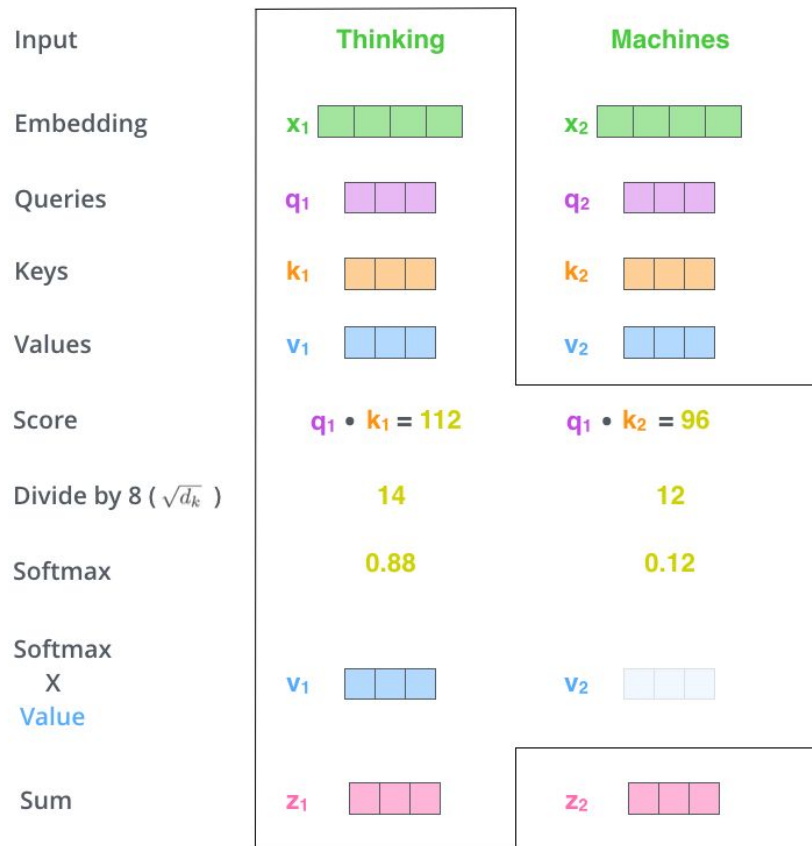
- Decompõe embedding da palavra em 3: Query, Key e Value



Self-Attention

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

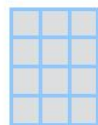
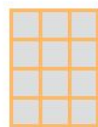
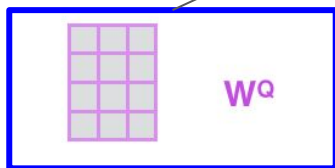
Calculando a autoatenção de “Thinking”



Self-Attention

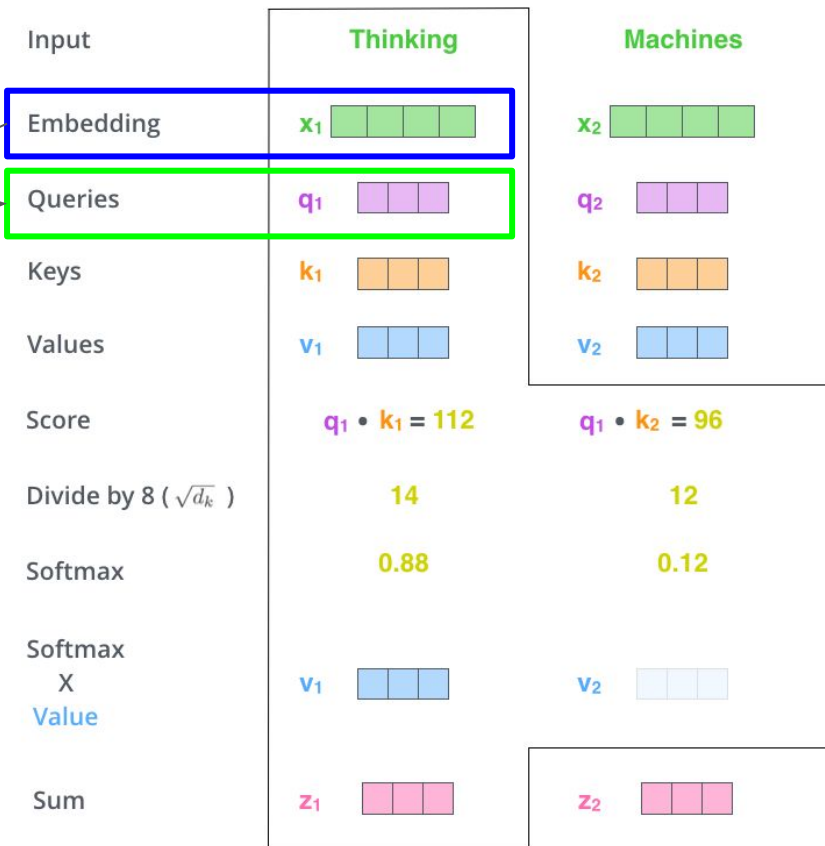
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}$$

Matrizes de peso aprendidas durante o treino



×

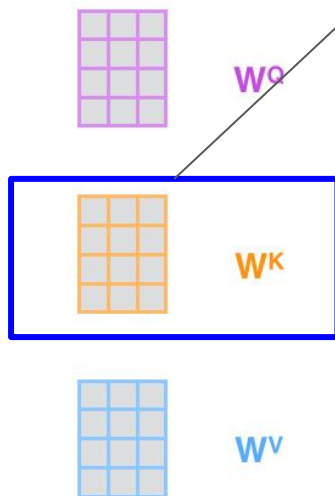
Calculando a autoatenção de “Thinking”



Self-Attention

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

Matrizes de peso aprendidas durante o treino



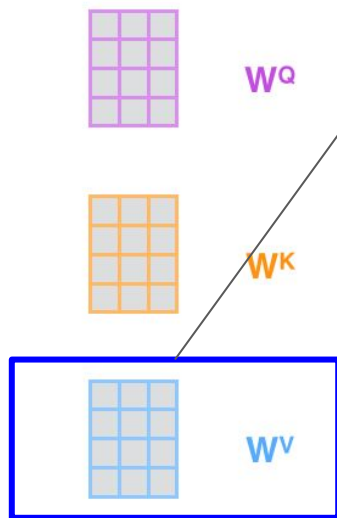
Calculando a autoatenção de “Thinking”

Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \cdot k_1 = 112$	$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12
Softmax X Value	v_1	v_2
Sum	z_1	z_2

Self-Attention

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

Matrizes de peso aprendidas durante o treino



Calculando a autoatenção de “Thinking”

Input	Thinking	Machines
Embedding	x₁	x ₂
Queries	q ₁	q ₂
Keys	k ₁	k ₂
Values	v₁	v ₂
Score	q ₁ • k ₁ = 112	q ₁ • k ₂ = 96
Divide by 8 (√d _k)	14	12
Softmax	0.88	0.12
Softmax X Value	v ₁	v ₂
Sum	z ₁	z ₂

Self-Attention

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

Determina a quantidade de atenção que deve ser dada a outras palavras da sentença enquanto codifica a palavra atual (“Thinking”)

Calculando a autoatenção de “Thinking”

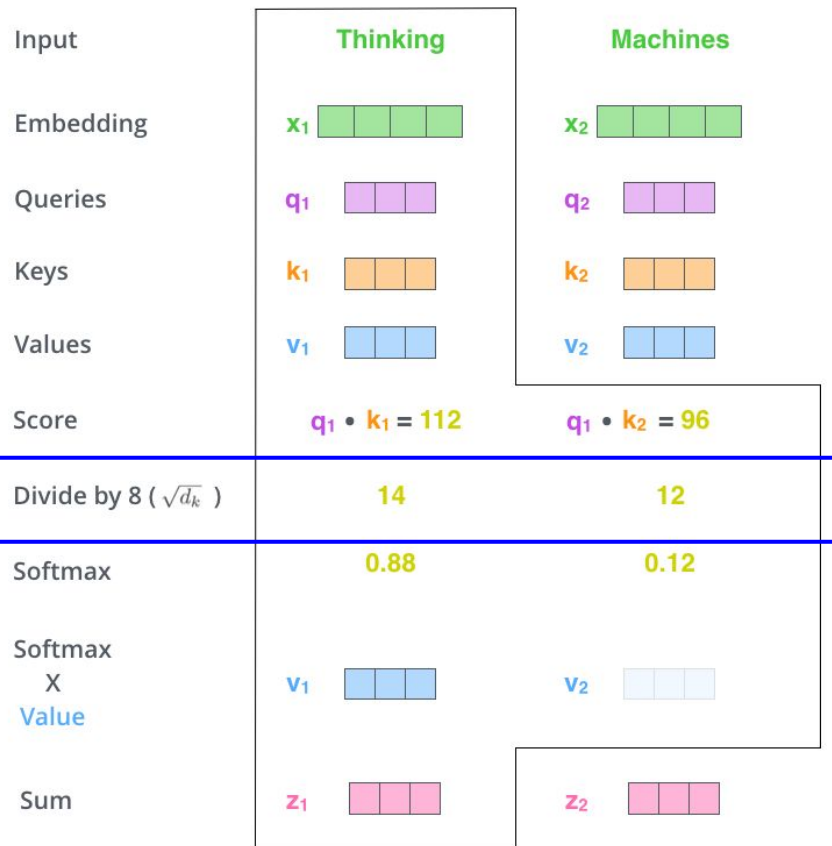
Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \cdot k_1 = 112$	$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12
Softmax X Value	v_1	v_2
Sum	z_1	z_2

Self-Attention

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

Divide o score pela raiz quadrada da dimensão (64)

Calculando a autoatenção de “Thinking”

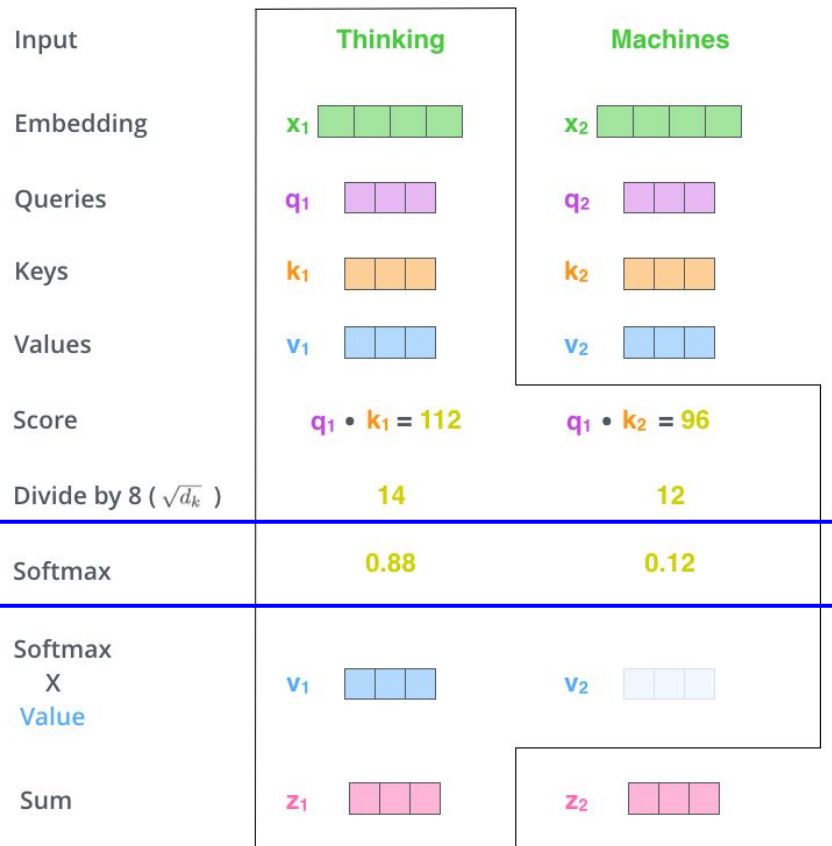


Self-Attention

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

Normaliza os scores para que sejam positivos e somem 1

Calculando a autoatenção de “Thinking”



Self-Attention

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

Calculando a autoatenção de “Thinking”

Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \cdot k_1 = 112$	$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12
Softmax X Value	v_1	v_2
Sum	z_1	z_2

Mantém intactos os valores das palavras que provêm atenção à palavra atual e diminui a relevância daquelas que não promovem.

Self-Attention

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

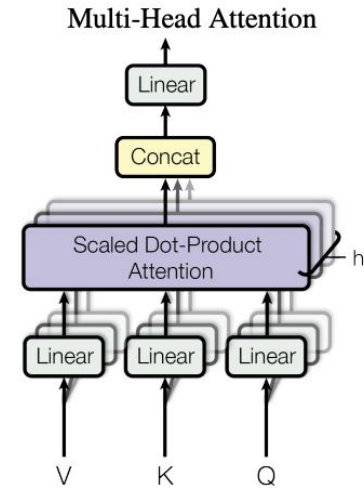
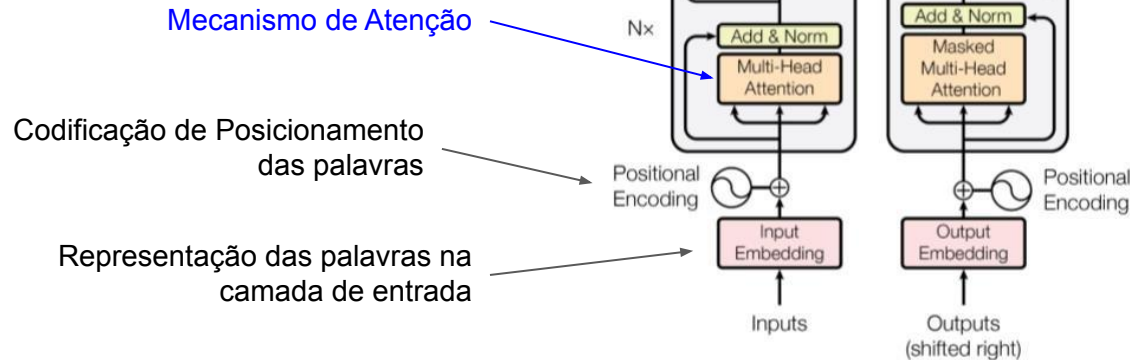
Soma os vetores

$$\text{output}_i = \sum_j \alpha_{ij} v_j$$

Calculando a autoatenção de “Thinking”

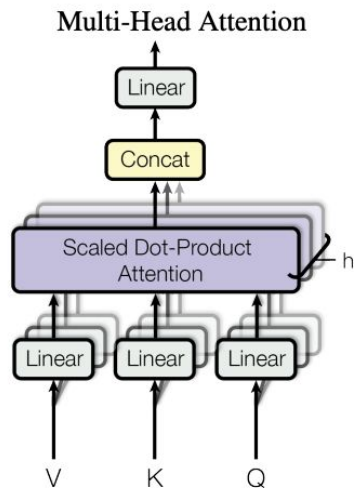
Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \cdot k_1 = 112$	$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12
Softmax X Value	v_1	v_2
Sum	z_1	z_2

Transformers: Multi-Head Attention



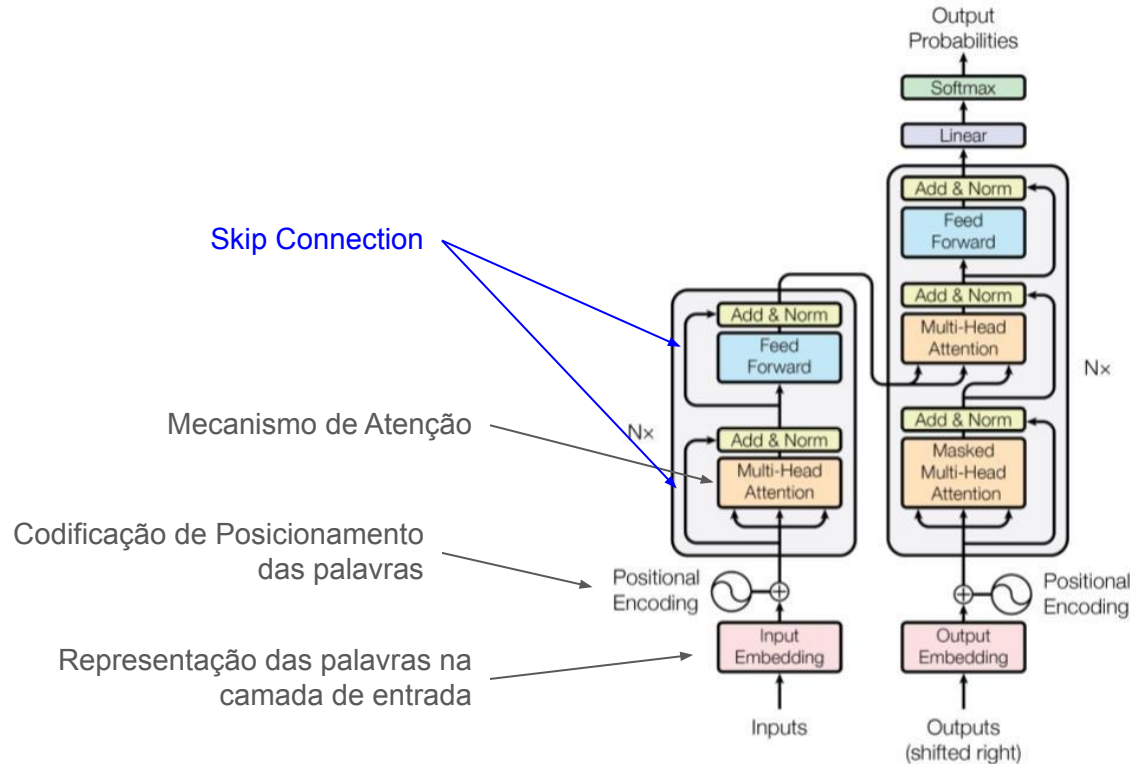
Múltiplas camadas de atenção rodando em paralelo

Transformers: Multi-head Attention

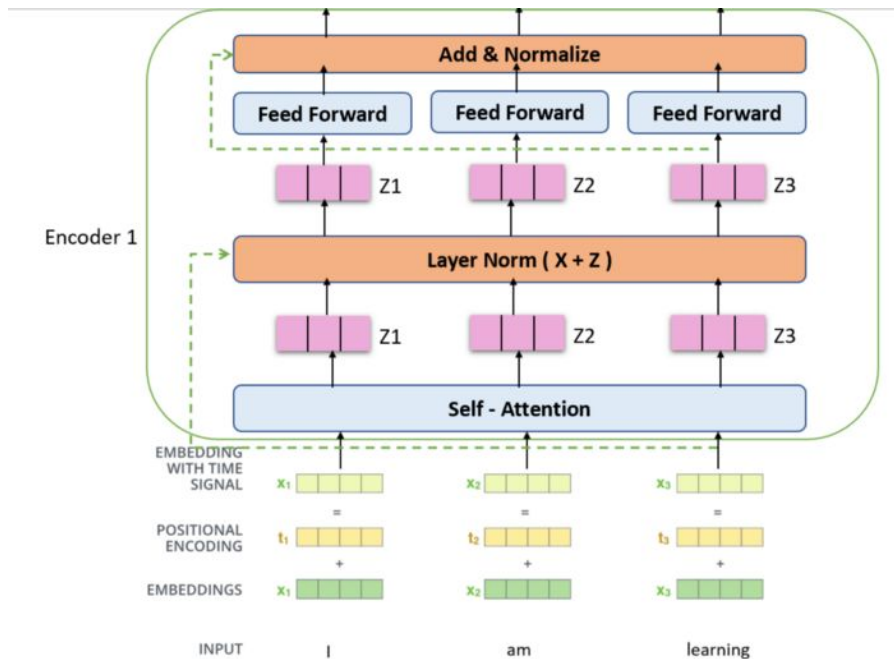


- ❖ Várias Camadas de Atenção rodando em paralelo
- ❖ Permite atender a diferentes partes da sequência de maneira diferente a cada vez.
- ❖ Melhor captura de informações posicionais, pois cada cabeça atenderá a diferentes segmentos da entrada.
- ❖ A combinação dessas cabeças de atenção provê uma representação mais robusta.
- ❖ Cada cabeça também irá capturar diferentes informações contextuais, correlacionando palavras de uma maneira única.

Transformers: Conexões Residuais

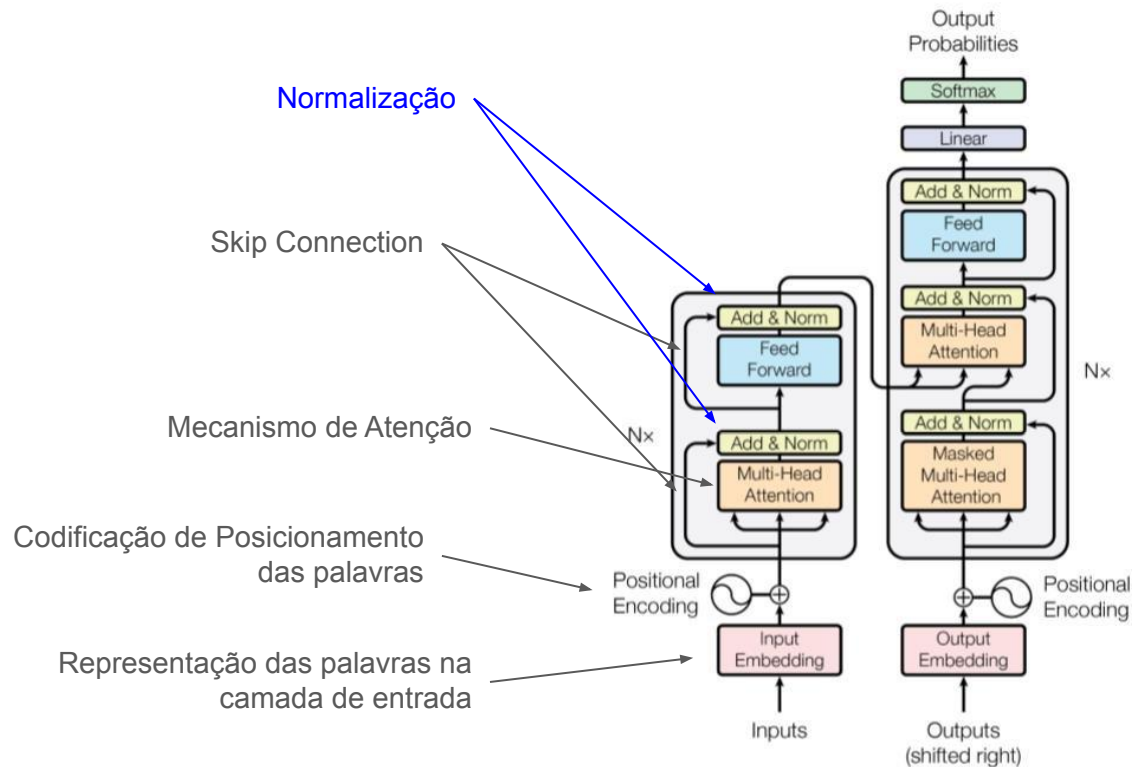


Skip Connection

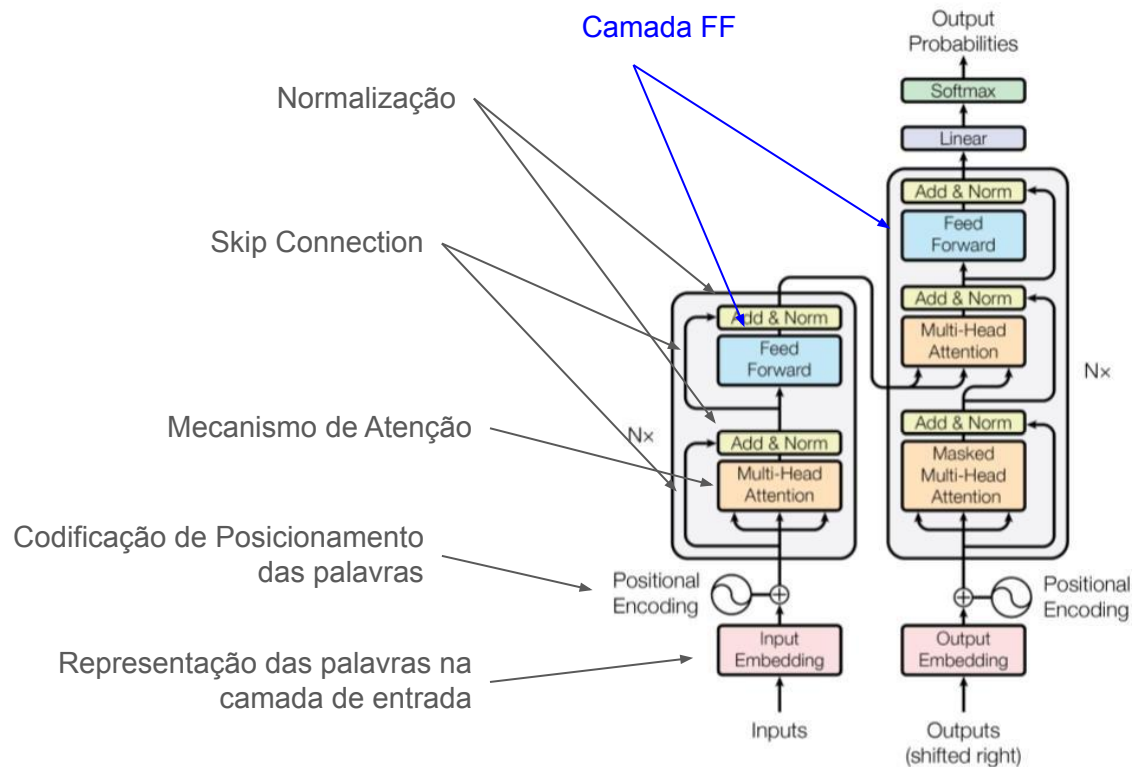


- ❖ Permite que as representações de diferentes níveis de processamento interajam.
- ❖ Mesma intuição da linguagem: combinação de diferentes ideias e contextos para uma compreensão mais ampla.
- ❖ As camadas posteriores têm acesso ao entendimento das camadas anteriores

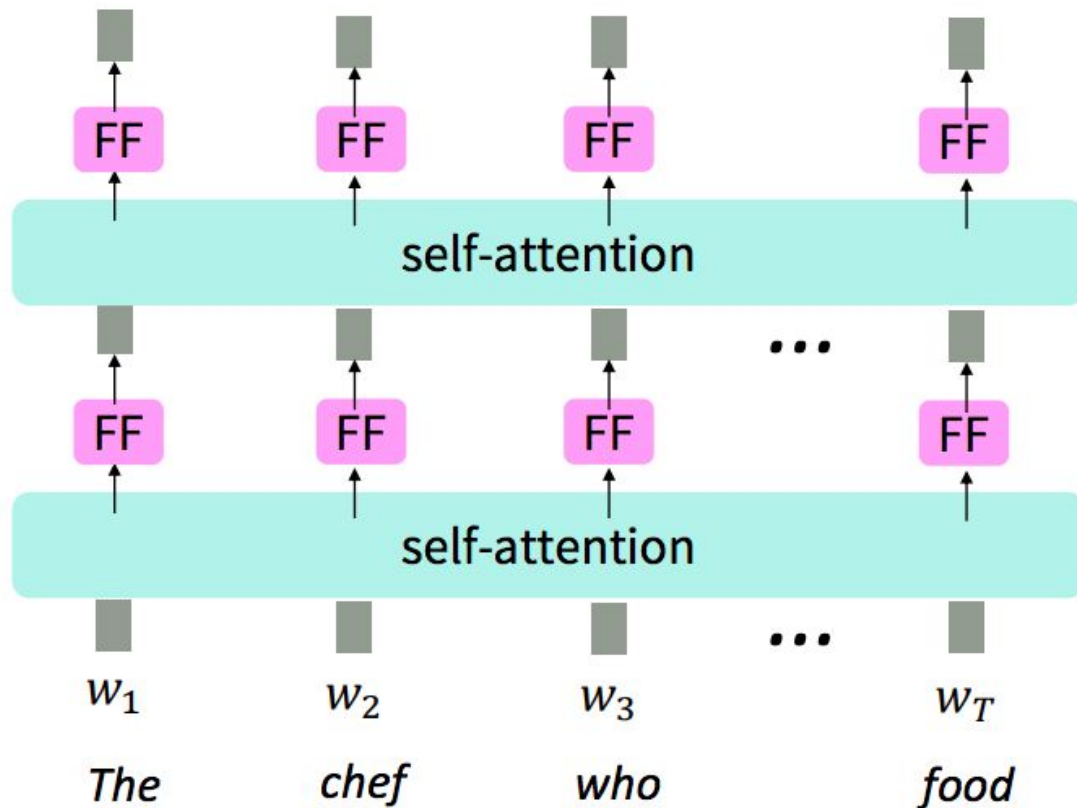
Transformers



Transformers



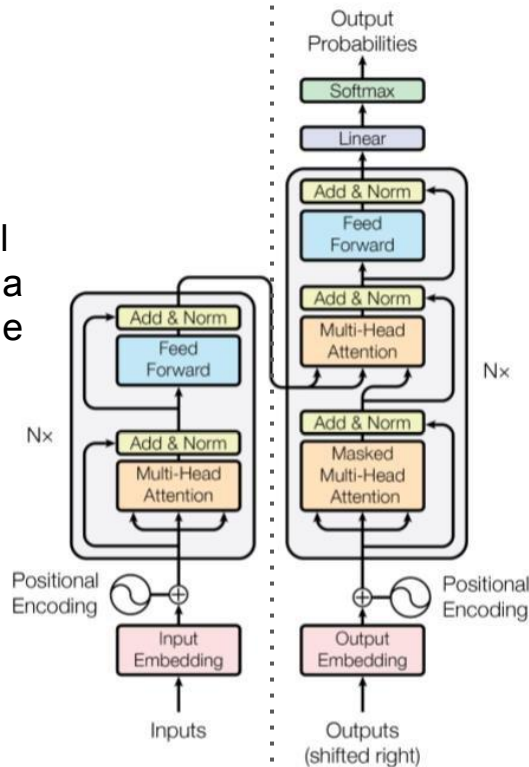
Adicionando Não-Linearidades



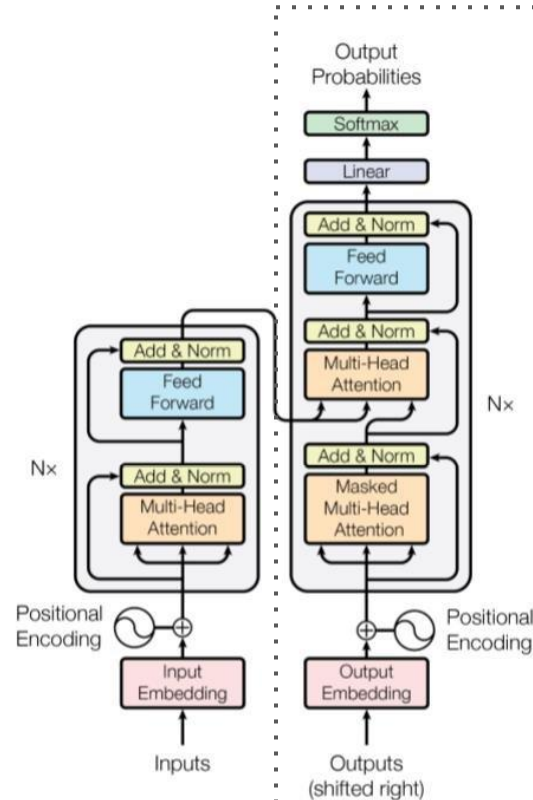
Transformers

Codificador (Encoder)

Encapsula a representação final da sentença de entrada e manda para o decodificador na forma de chave-valor (key e value)



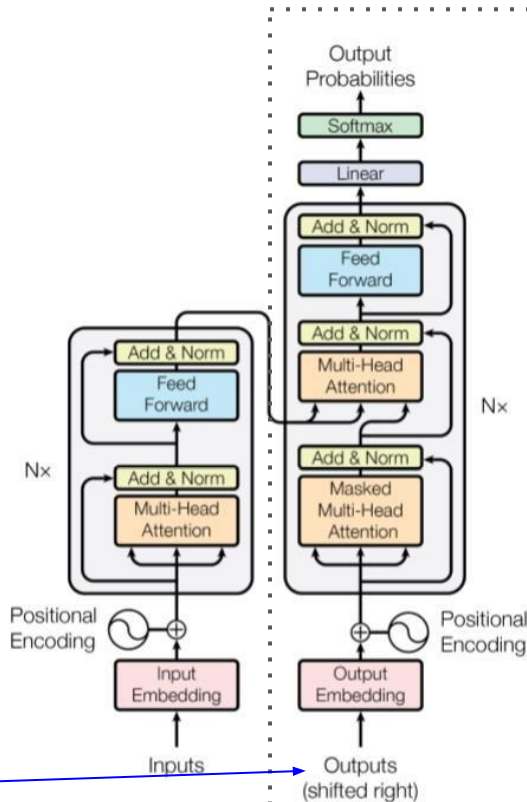
Transformers



Decodificador (Decoder)

- ❖ Processa a representação codificada.
- ❖ Determina o quão relacionada cada palavra da saída esperada está em relação às palavras da entrada

Transformers



Decodificador (Decoder)

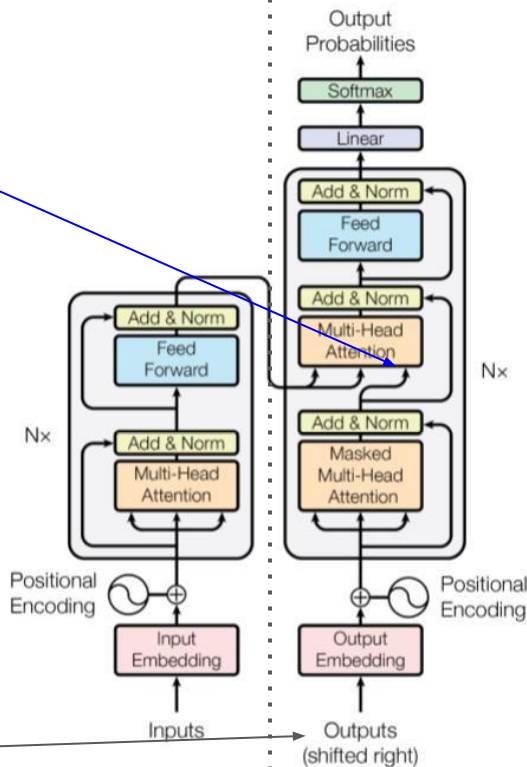
- ❖ Processa a representação codificada.
- ❖ Determina o quão relacionada cada palavra da saída esperada está em relação às palavras da entrada

Na geração de texto o output é a mesma sentença de entrada deslocada para a direita

Transformers

Após processamento da saída esperada. A matriz resultante é entregue como consulta

Na geração de texto o output é a mesma sentença de entrada deslocada para a direita



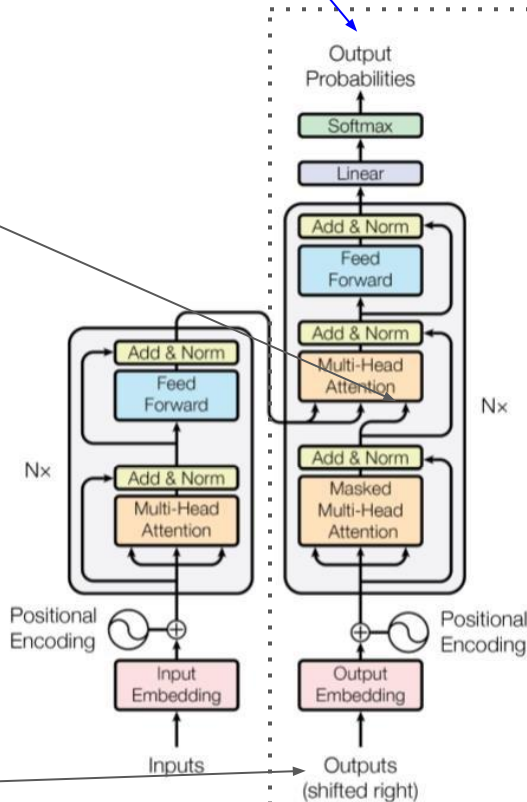
Decodificador (Decoder)

- ❖ Processa a representação codificada.
- ❖ Determina o quão relacionada cada palavra da saída esperada está em relação às palavras da entrada

Transformers

As probabilidades de saída predizem o próximo token na sentença de saída

Após processamento da saída esperada. A matriz resultante é entregue como consulta

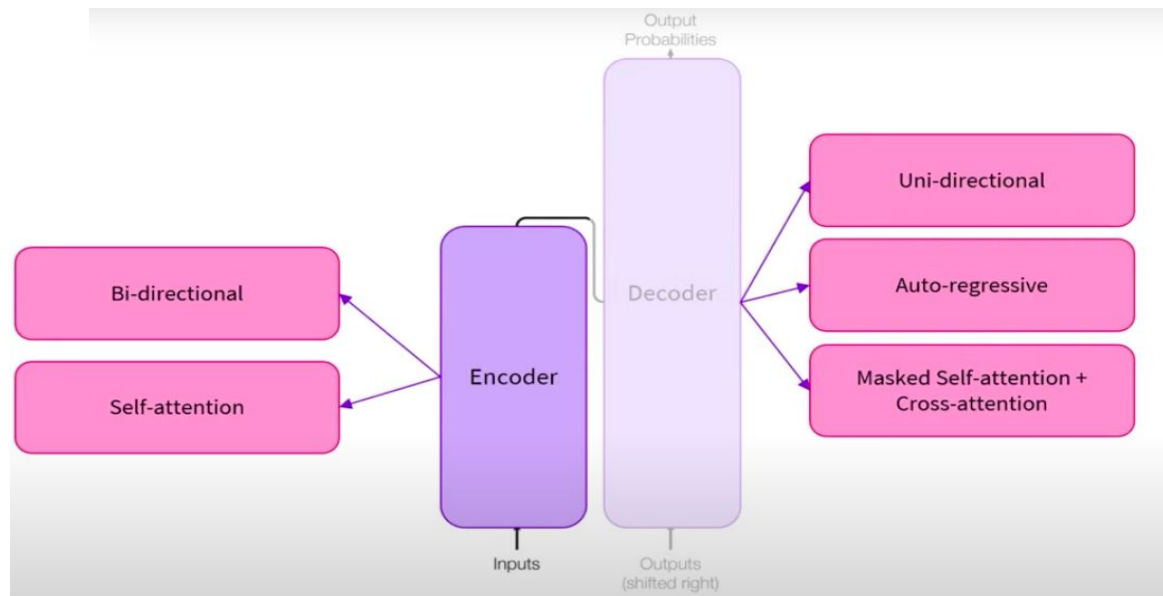


Decodificador (Decoder)

- ❖ Processa a representação codificada.
- ❖ Determina o quão relacionada cada palavra da saída esperada está em relação às palavras da entrada

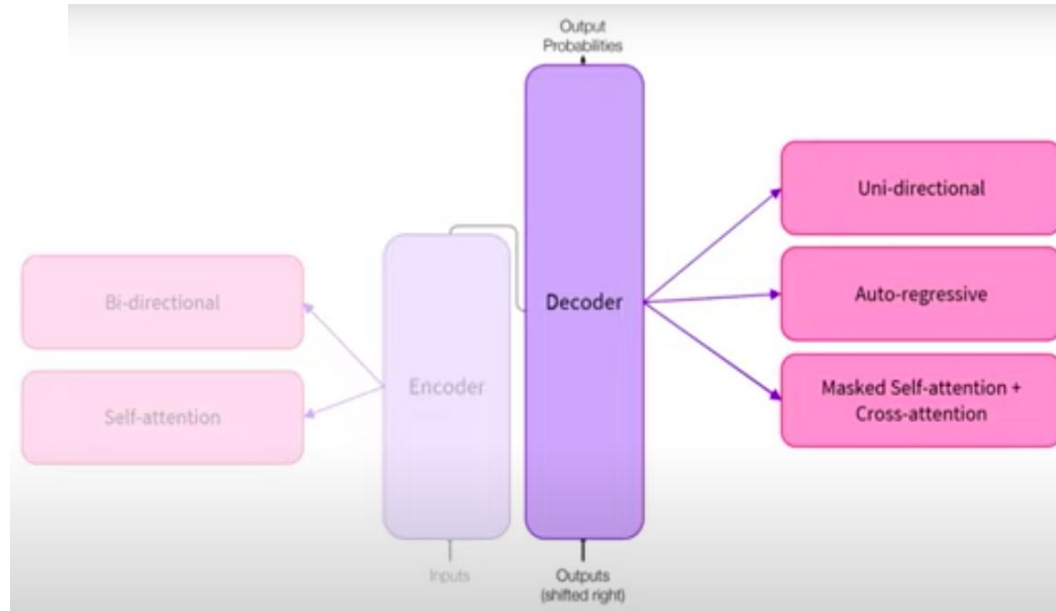
Na geração de texto o output é a mesma sentença de entrada deslocada para a direita

Transformers: Encoder



Fonte: <https://huggingface.co/course/chapter1/5>

Transformers: Decoder



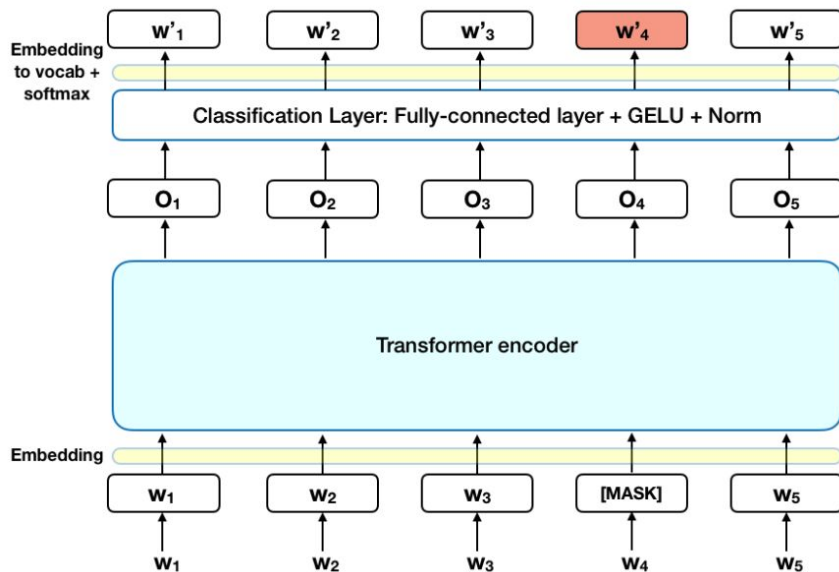
fonte: <https://huggingface.co/course/chapter1/6>

Transformers & Modelos de Linguagem Pré-treinados

Bidirectional Encoder Representations from Transformers

BERT

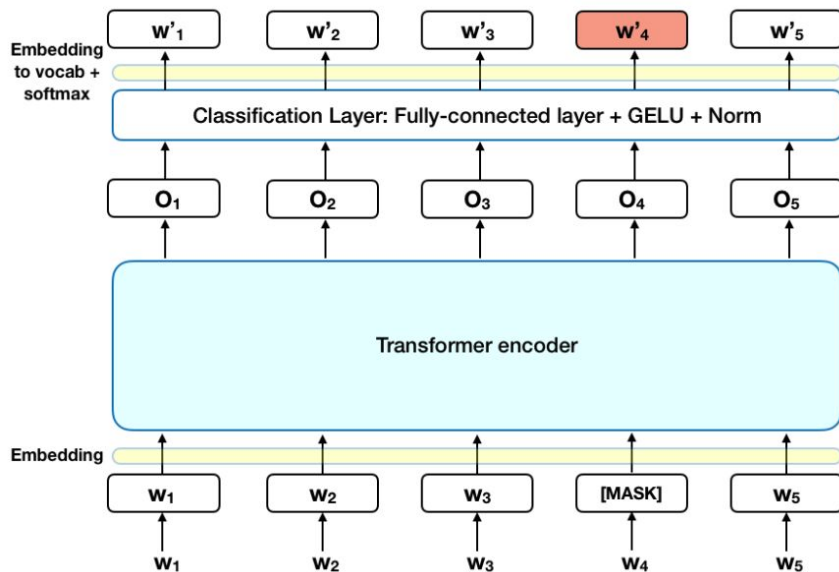
- ❖ Modelos de Linguagem Tradicionais: dependem da ordem sequencial de palavras
- ❖ BERT: É um modelo de linguagem contextual, pois mapeia a representação contextual bidirecional de toda a frase
- ❖ É um modelo pré-treinado para Processamento de Linguagem Natural
- ❖ Utiliza transformers mas, apenas o mecanismo codificador é necessário



Bidirectional Encoder Representations from Transformers

BERT

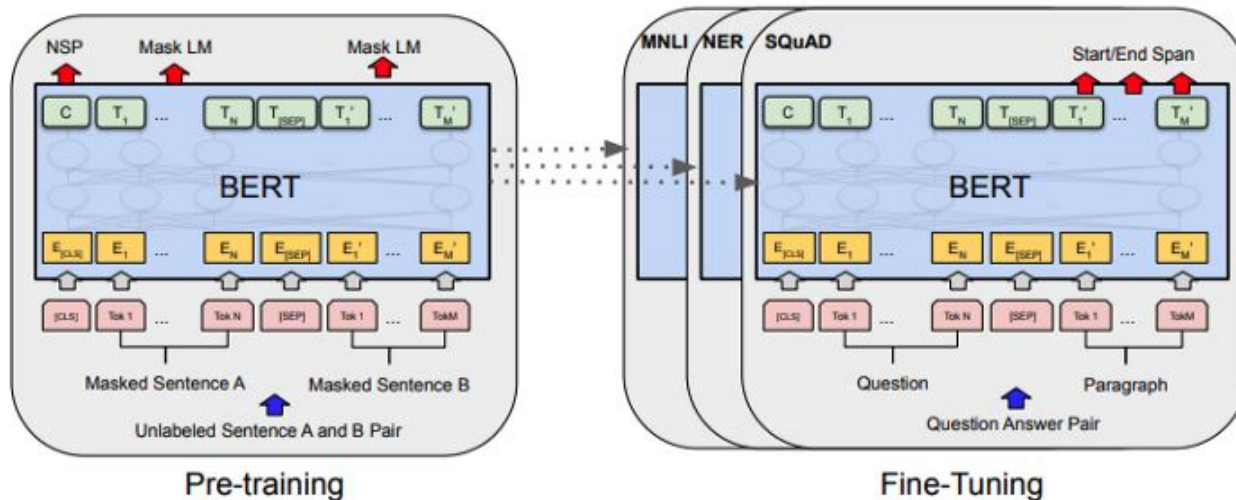
- ❖ Dados
 - Wikipedia (2.5B palavras) + BookCorpus (800M palavras)
- ❖ BERT-Base: 110M
- ❖ BERT-Large: 340M
- ❖ Treinado em 4 dias



BERT e Transfer Learning

❖ Duas etapas:

- Pre-treinamento do modelo
- Fine-tuning: treinar apenas algumas camadas do modelo pré-treinado (ou adicionar novas camadas) de forma a executar tarefas específicas



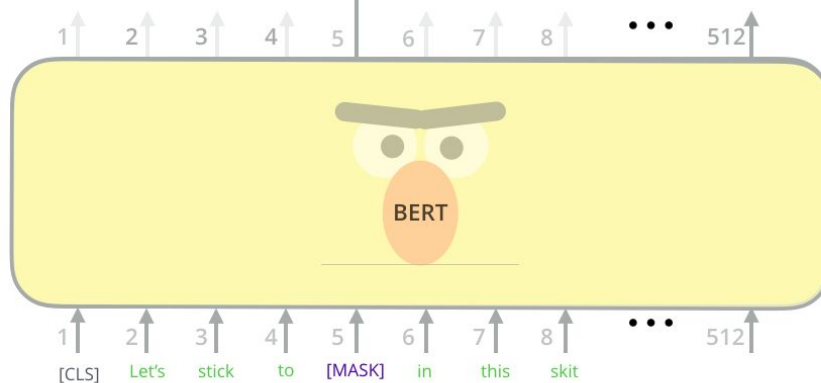
BERT pre-training: Masked Language Modeling

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

Possible classes:
All English words

0.1%	Aardvark
...	...
10%	Improvisation
...	...
0%	Zyzzzyva

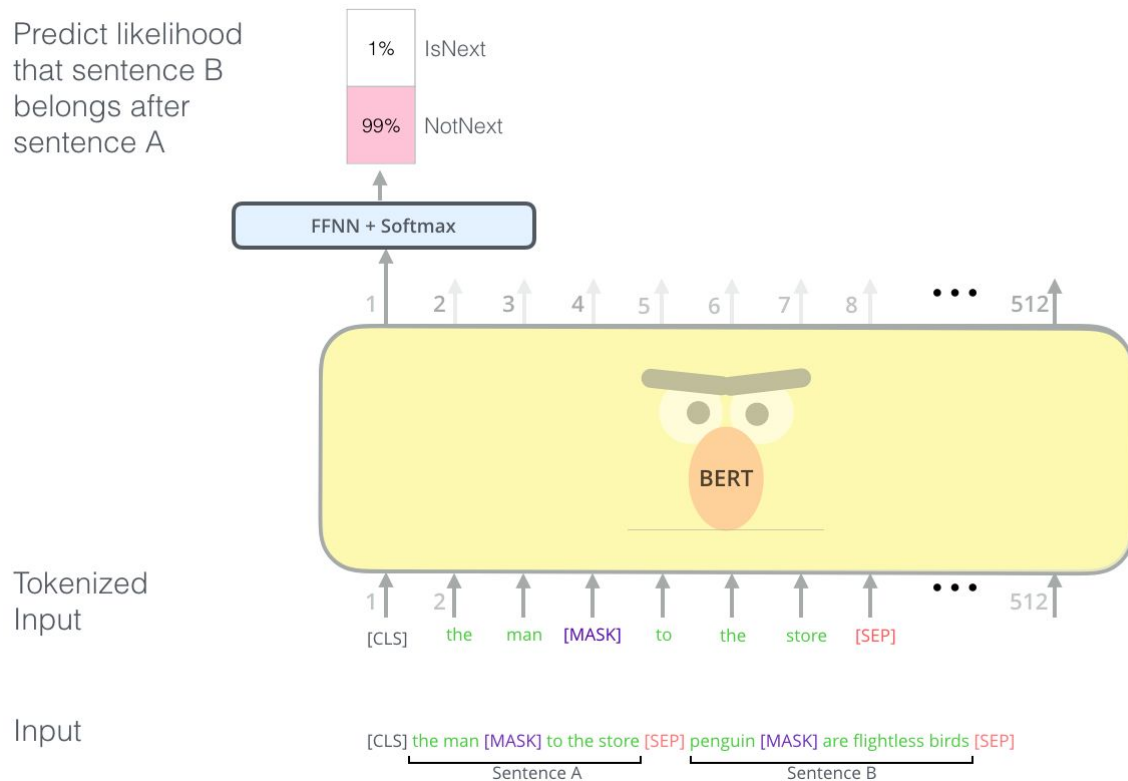
FFNN + Softmax



Randomly mask
15% of tokens

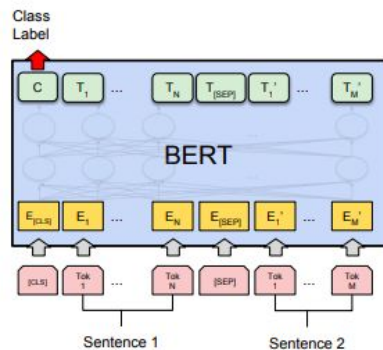
Input

BERT pre-training: Next Sentence Prediction

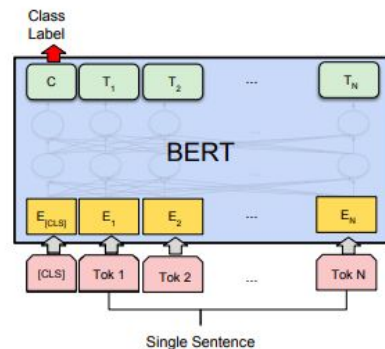


BERT: Como usar

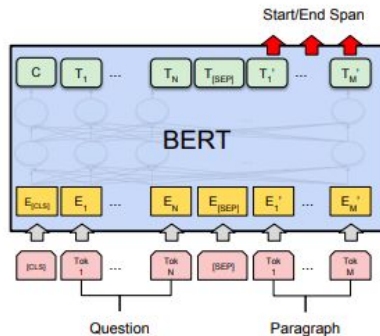
- ❖ Diversas tarefas
 - ❖ Como extrator de features
 - Feature de palavras: posição do token
 - Feature de sentenças: token especial [CLS]
- Pode ser usado para classificação



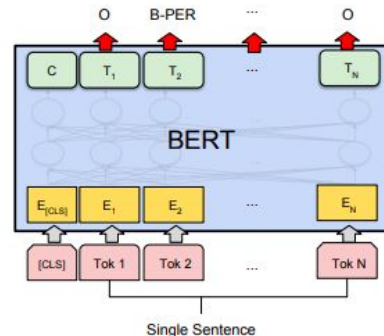
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1

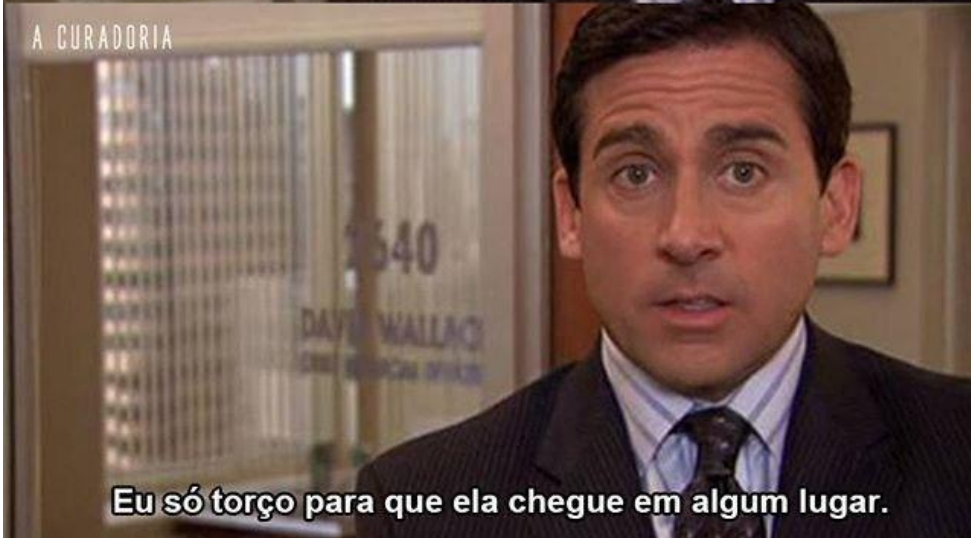
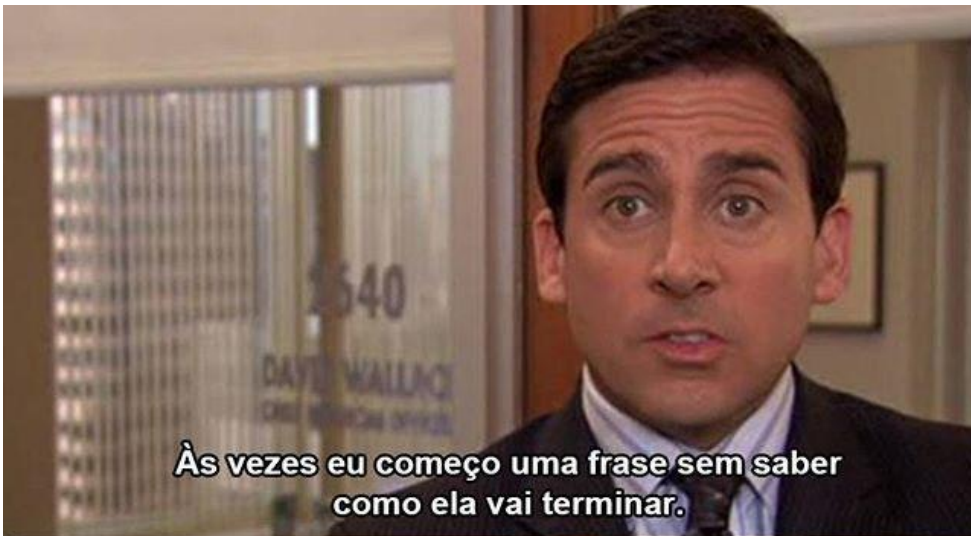


(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Variações



Generative Pretrained Transformer (GPT)

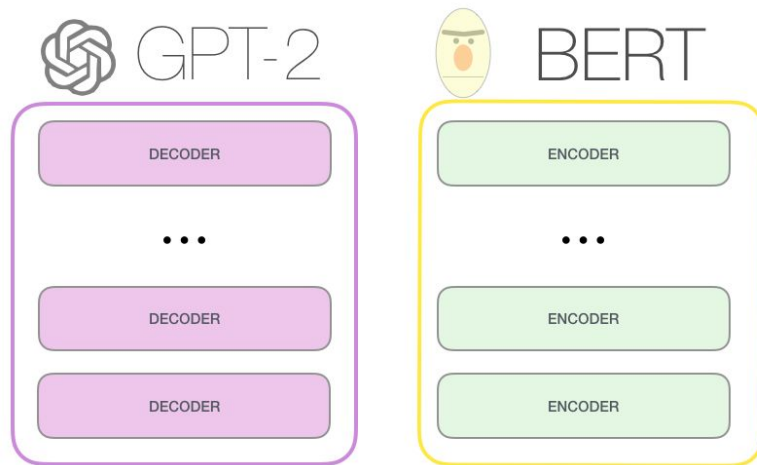


Generative Pretrained Transformer (GPT)

- ❖ OpenAI (2018)
<https://openai.com/blog/language-unsupervised/>
- ❖ Modelagem de Linguagem Não Supervisionada (Pré-treinamento)
- ❖ Processa o texto de forma unidirecional
- ❖ Transformer **decoder** com 12 camadas
- ❖ 768-dimensional hidden states, 3072-dimensional feed-forward
- ❖ **Byte-pair encoding** com 40.000 merges
- ❖ 117M parâmetros
- ❖ Treinado no BooksCorpus: mais de 7.000 livros (sentenças longas)

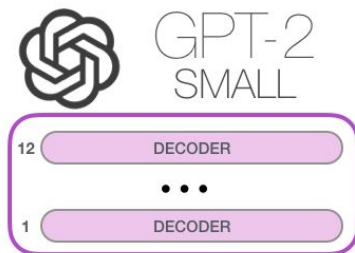
GPT-2

- ❖ Conjunto de dados maior e adição de mais parâmetros ao modelo
- ❖ WebText
 - 40 GB
 - 8 milhões de documentos
 - Removidos todos os artigos da Wikipedia pois muitos conjuntos de teste contém artigos da Wikipedia
- ❖ 1,5 bilhão de parâmetros (10x > GPT-1)
- ❖ 48 camadas
- ❖ 50.257 tokens
- ❖ Demo:
<https://demo.allennlp.org/next-token-lm>

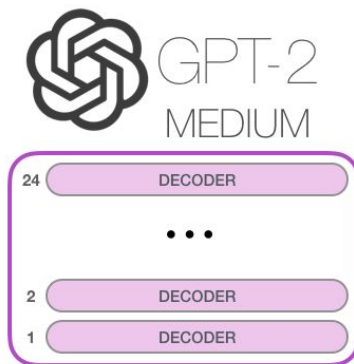


Fonte: <https://jalammar.github.io/illustrated-gpt2/>

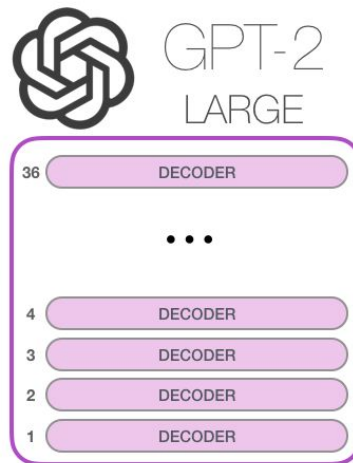
GPT-2



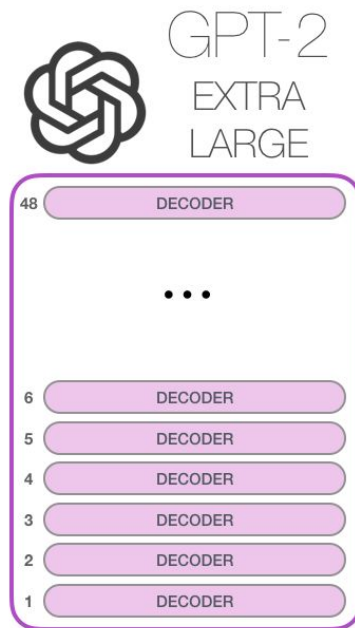
Model Dimensionality: 768



Model Dimensionality: 1024



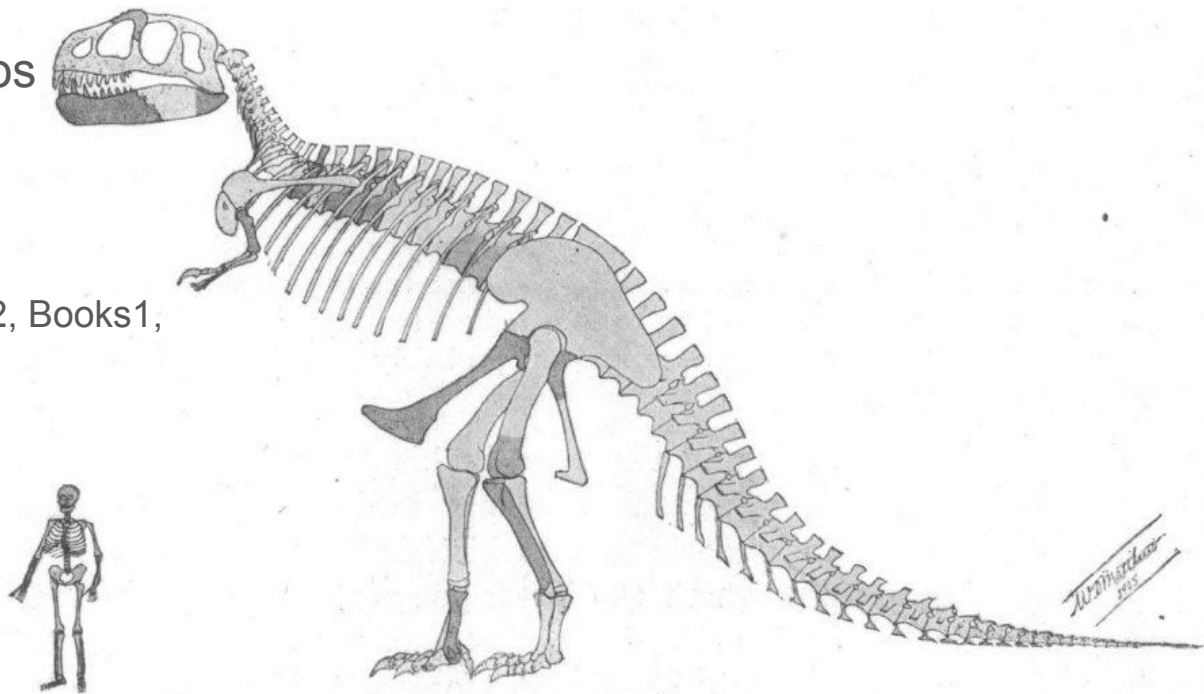
Model Dimensionality: 1280



Model Dimensionality: 1600

GPT-3

- ❖ 175 bilhões de parâmetros (100x > GPT-2)
- ❖ Corpus:
 - 45TB
 - Common Crawl, WebText2, Books1, Books2 e Wikipedia
- ❖ 96 camadas



GPT-2
1.5B Parameters

GPT-3
175B Parameters

GPT-3

In-context Learning

The three settings we explore for in-context learning

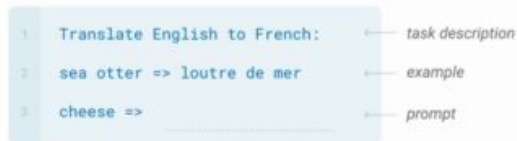
Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



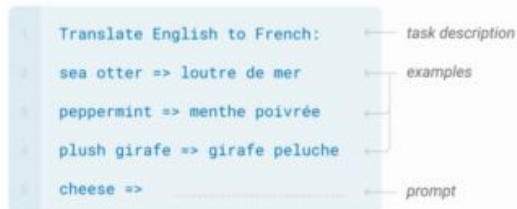
One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



Artigo Original:

<https://arxiv.org/pdf/2005.14165.pdf>

GPT-3: Algumas aplicações

- ❖ Q&A
- ❖ Resolução de esquemas
- ❖ Tradução
- ❖ Adição aritmética
- ❖ Decodificação de palavras
- ❖ Geração de artigos de notícias
- ❖ Aprendizagem e uso de palavras novas
- ❖ Programação
- ❖ API de acesso

<https://openai.com/api/>

GPT-3: Limitações

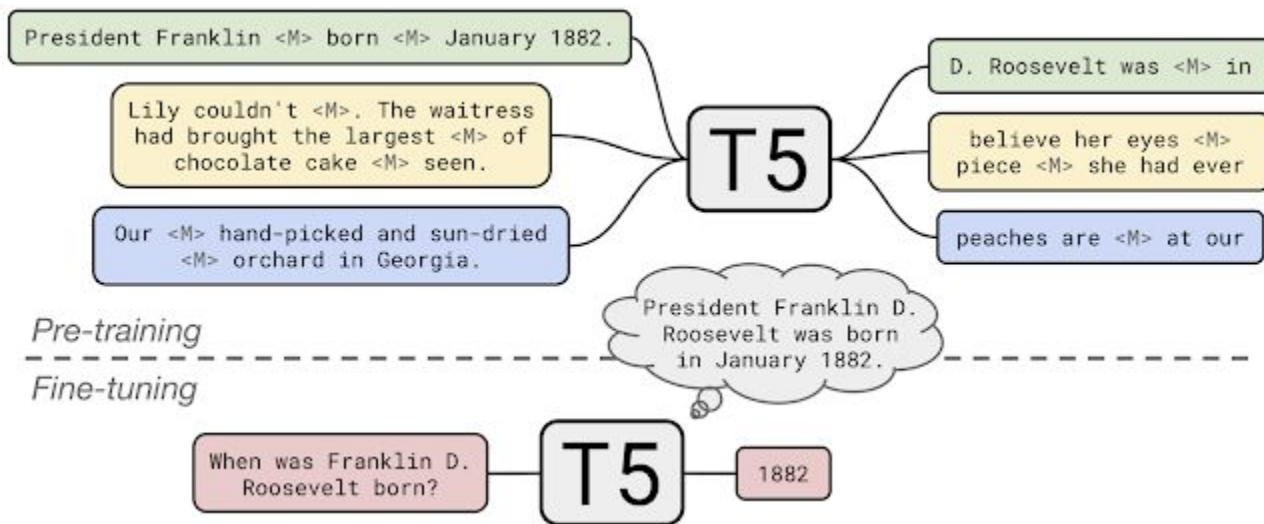
- ❖ Preconceitos da linguagem aprendidos do conjunto de treino: gênero, etnia, raça ou religião...
- ❖ Perda de coerência ao formular frases longas
- ❖ Repetição de sequências de texto indefinidamente
- ❖ Baixo desempenho em tarefas de inferência de linguagem natural, preencher os espaços em branco e tarefas de compreensão de leitura
- ❖ Problema para seguir instruções explícitas
- ❖ Alucinações: fatos inexistentes ou incorretos
- ❖ Desinformação
- ❖ Difícil de interpretar as suas tomadas de decisões

ChatGPT

- Treino incorpora aprendizagem por reforço a partir de feedback humano;
- Fine-tuning do modelo GPT-3 em um amplo conjunto de dados rotulado;
- Foram utilizadas entradas e prompts de usuários da plataforma da OpenAI API;
- Os rotuladores manuais precisaram criar exemplos de categorias de não abrangidas pelos usuários da OpenAI API.

Text-to-Text Transfer Transformer (T5)

- ❖ Encoder-decoder transformer
- ❖ Gera texto como saída para qualquer tarefa



Aula Prática

[Google Colab](#) - Modelo Bigram

[Google Colab](#) - Modelo BERT

GPT-3 -> [OpenAI](#)

ChatGPT -> [OpenAI](#)

Referências

Dan Jurafsky, James H. Martin. Speech and Language Processing. (3rd ed. Draft). 2021. Disponível em: <<https://web.stanford.edu/~jurafsky/slp3/7.pdf>>. Capítulo 6. Acesso em: 01 Setembro de 2022.

Dan Jurafsky, James H. Martin. Speech and Language Processing. (3rd ed. Draft). 2021. Disponível em: <<https://web.stanford.edu/~jurafsky/slp3/11.pdf>>. Capítulo 11. Acesso em: 01 Setembro de 2022.

<https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>

Tokenizers: How machines read. Disponível em: <<https://blog.floydhub.com/tokenization-nlp/>>. Acesso em: 01 Setembro de 2022.

Data Science Academy. Deep Learning Book, 2022. Disponível em: <<https://www.deeplearningbook.com.br/>>. Capítulos: 76 a 82. Acesso em: 01 Setembro. 2022.

Data Science Academy. Deep Learning Book, 2022. Disponível em: <<https://www.deeplearningbook.com.br/>>. Capítulos: 86 a 90. Acesso em: 01 Setembro. 2022.