

PAPER · INFOGRÁFICO

Abrindo a Caixa-Preta da Inferência em LLMs

Anatomia do Transformer por Dentro, da Entrada ao Token de Saída

Aridio Silva

Pesquisador Independente · @aridiosilva
ORCID 0009-0008-2411-6995



Do texto ao token: 9 etapas

1 Tokenização

Texto vira números (IDs)

2 Embeddings + posição

IDs viram vetores com ordem

3 Self-attention

Cada token consulta os demais

4 Multi-head + máscaras

Várias relações em paralelo

5 Add & Norm

Estabilidade em profundidade

6 FFN

Transformação não linear por token

7 Empilhamento

Refinamento camada a camada

8 Pesos e treino

O que o modelo aprendeu

9 Saída

Logits → softmax → palavra

O que este paper responde

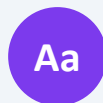
“O banco estava lotado”

Como o modelo decide, sem regras explícitas, se “banco” é uma agência ou um assento?



Objetivo

Abrir a “caixa-preta”: conceito, mecanismo, técnica, exemplo numérico e referência para cada etapa da inferência.



Convenções

n = tamanho da sequência; d_model = dimensão do modelo; d_k = dimensão de cada cabeça. Referências no formato (Autor, ano).

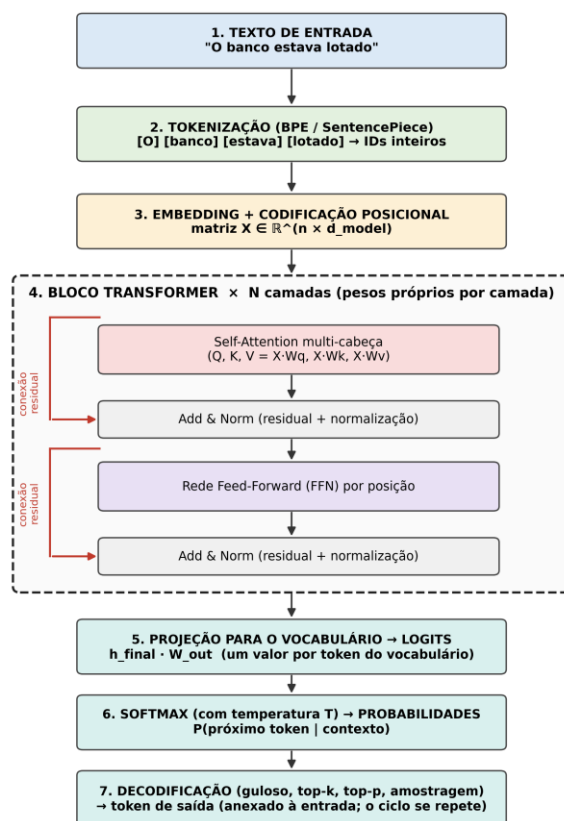


Aviso numérico

Os números são didáticos: dimensões pequenas, pesos à mão ou aleatórios, calculados por código. Não são medições de um modelo real.

O fluxo completo da inferência

Figura 1 - Fluxo completo da inferência em um Transformer (visão geral)



- **Inferência:** executar o modelo treinado, com pesos fixos, para gerar uma saída.
- **Ciclo autorregressivo:** prevê o próximo token, escolhe um, anexa à entrada e repete.
- **Forward pass:** uma passagem completa pelas camadas.
- **Famílias:** somente codificador (BERT), somente decodificador (GPT) e codificador-decodificador (original).
- **Treino ≠ inferência:** o treino define os pesos; a inferência apenas os usa.

Etapas, técnicas e referências

Etapas	Técnica principal	Referência-chave
1. Tokenização	BPE, SentencePiece	Sennrich et al., 2016; Kudo & Richardson, 2018
2. Embedding e posição	Matriz de embeddings; senoidal ou RoPE	Mikolov et al., 2013; Vaswani et al., 2017; Su et al., 2021
3. Self-attention	Produto escalar escalonado e softmax	Bahdanau et al., 2015; Vaswani et al., 2017
4. Multi-head	Subespaços de dimensão d_k	Vaswani et al., 2017; Voita et al., 2019
5. Add & Norm	Residual; LayerNorm, RMSNorm	He et al., 2016; Ba et al., 2016; Zhang & Sennrich, 2019
6. FFN	MLP com ReLU, GELU ou SwiGLU	Geva et al., 2021; Shazeer, 2020
7. Empilhamento	N camadas	Tenney et al., 2019
8. Saída	Softmax, temperatura, top-k, top-p	Holtzman et al., 2020

Problema resolvido e custo da ausência (1/2)

Etapa	Problema que resolve	Sem ela
Tokenização	A rede só processa números; o vocabulário precisa ser finito	Sem entrada numérica; palavras raras sem representação ou sequências longas demais
Embedding	IDs inteiros são rótulos arbitrários, sem similaridade	One-hot esparso de dimensão $ V $, sem similaridade entre tokens
Codificação posicional	A atenção não usa a ordem das palavras	Atenção sem máscara não distingue ordens diferentes (Vaswani et al., 2017)
Self-attention	Cada token precisa de informação dos demais	Cada token seria lido isoladamente: “banco” teria sempre o mesmo vetor
Multi-head	Várias relações ao mesmo tempo	Uma só média ponderada por camada (Vaswani et al., 2017)

Problema resolvido e custo da ausência (2/2)

Etapa	Problema que resolve	Sem ela
Add & Norm	Sinal e gradiente atravessam muitas camadas	Treino instável ou degradado (He et al., 2016; Ba et al., 2016)
FFN	A atenção só combina Values; falta transformação por token	Sem ~2/3 dos parâmetros da camada (Geva et al., 2021)
Empilhamento	Relações complexas exigem composição em passos	Só relações diretas; induction heads exigem ≥ 2 camadas (Olsson et al., 2022)
Treino	Pesos aleatórios não produzem linguagem	Circuito sem comportamento útil, como nos exemplos
Saída	Um vetor precisa virar um token	Sem distribuição nem palavra escolhida

Tokenização: texto vira números

1

Conceito

O texto vira uma sequência de tokens e cada token, um ID inteiro. O vocabulário é finito e fixado antes do treino.

2

Por que subpalavras

Palavras inteiras deixam termos novos sem representação; caracteres geram sequências longas. Subpalavras equilibram os dois.

3

Por que é necessária

Sem IDs não há entrada para a rede; sequências longas encarecem a atenção (custo $\propto n^2$).

Frase-guia, 4 palavras = 4 tokens (IDs ilustrativos)

0

→ 12

banco

→ 4871

estava

→ 905

lotado

→ 23019

A tokenização não desambigua: o ID de “banco” é o mesmo nos dois sentidos.

Quatro técnicas de subpalavra

1 BPE

Parte de caracteres e funde o par adjacente mais frequente. Compressão: Gage (1994); tradução: Sennrich et al. (2016). Byte-level no GPT.

2 WordPiece

Escolhe fusões pelo ganho de verossimilhança, não pela frequência (Schuster & Nakajima, 2012; Wu et al., 2016). Usada no BERT.

3 Unigram LM

Parte de um vocabulário grande e poda o que contribui menos (Kudo, 2018).

4 SentencePiece

Implementa BPE e Unigram sobre o texto bruto, sem pré-tokenização por espaços; independe de idioma (Kudo & Richardson, 2018).

BPE em um corpus didático

Corpus: low (5), lower (2), newest (6), widest (3); </w> marca o fim de palavra.

Passo	Par fundido	Freq.	Novo símbolo
1	e + s	9	es
2	es + t	9	est
3	est + </w>	9	est</w>
4	l + o	7	lo
5	lo + w	7	low
6	n + e	6	ne
7	ne + w	6	new
8	new + est</w>	6	newest</w>

newest</w>

“newest” vira um único token

w · i · d · est</w>

“widest” nunca foi vista inteira, mas reaproveita est</w>

Resultado calculado por código (empates pela ordem de ocorrência).

De ID a vetor denso

E

Embedding

Cada ID seleciona uma linha de W_E ($|V| \times d_{\text{model}}$): um vetor denso aprendido (Bengio et al., 2003; Mikolov et al., 2013).

512

d_{model} no Transformer original

≡

Estático

“banco” recebe o mesmo vetor em qualquer frase. A contextualização só acontece depois, na atenção.

768

d_{model} do BERT-base

?

Por que é necessária

O ID 4871 não é “parecido” com 4872. Vetores permitem similaridade e as somas e produtos seguintes.

12.288

d_{model} do GPT-3 (maior versão)

Codificação posicional: a ordem importa

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

Por quê? “o gato come o peixe” e “o peixe come o gato” têm os mesmos tokens; sem posição a atenção sem máscara os trata de forma equivalente (Vaswani et al., 2017).

1 Aprendidas

Tabela treinável (BERT, GPT-2)

2 Relativas

Atenção depende da distância
(Shaw et al., 2018)

3 RoPE

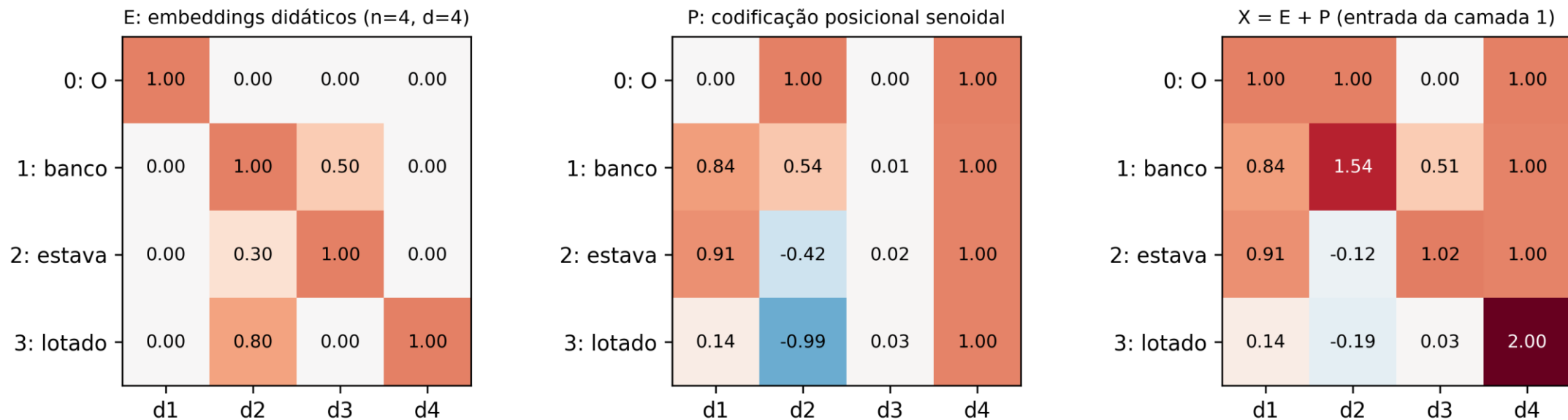
Rotaciona Q e K pela posição
(Su et al., 2021); usada no
LLaMA

4 ALiBi

Viés linear por distância;
extrapola contextos longos
(Press et al., 2022)

$X = E + P$ para “O banco estava lotado”

Figura 2 - Etapa 2: embeddings e codificação posicional (valores calculados; pesos didáticos)



© Aridio Silva · @aridiosilva · Outubro 2026 · ORCID 0009-0008-2411-6995

“banco” (posição 1): PE = [0,841; 0,540; 0,010; 1,000]; embedding [0; 1; 0,5; 0] → x = [0,841; 1,540; 0,510; 1,000]

$d_{model} = 4$ (didático; o original usa 512). X, de dimensão 4×4 , é a entrada da primeira camada.

Query, Key e Value: três papéis



$$\text{Query} = X \cdot W_q$$

O que este token procura.



$$\text{Key} = X \cdot W_k$$

O que este token oferece para ser encontrado.



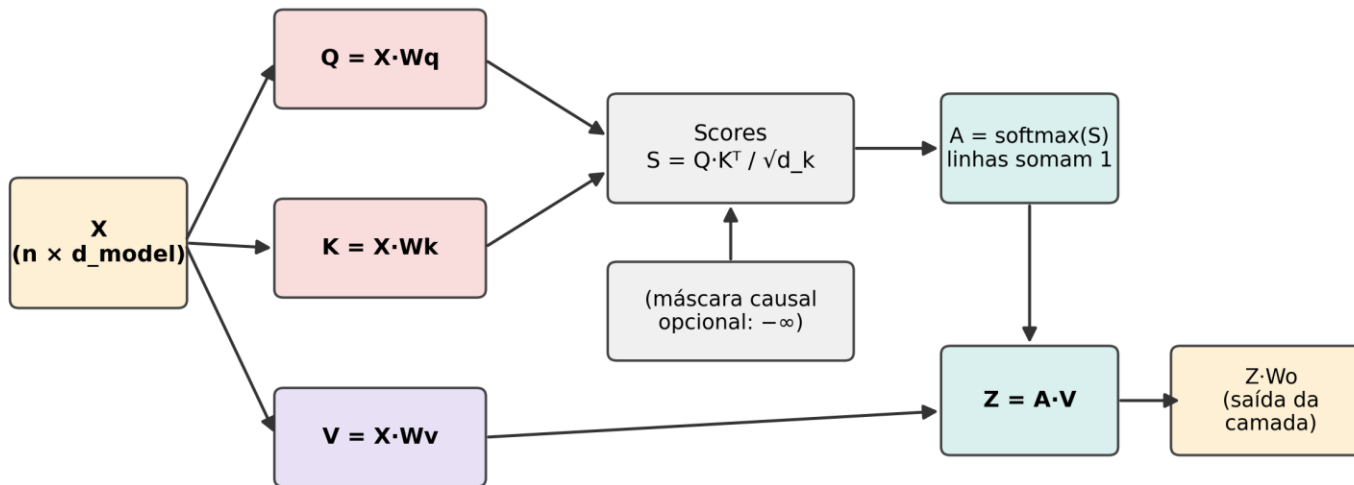
$$\text{Value} = X \cdot W_v$$

O conteúdo que o token repassa se for selecionado.

Por que é necessária. O sentido de uma palavra depende dos vizinhos, mas o embedding de entrada é estático. Cada token precisa consultar os demais, com pesos que dependam do conteúdo. Na atenção, quaisquer duas posições se ligam em um passo, e o cálculo é paralelizável (Vaswani et al., 2017).

Produto escalar escalonado

Figura 3 - Mecanismo de atenção por produto escalar escalonado (Vaswani et al., 2017)



© Aridio Silva · @aridiosilva · Outubro 2026 · ORCID 0009-0008-2411-6995

$$\text{softmax}(QK^T / \sqrt{d_k}) \cdot V$$

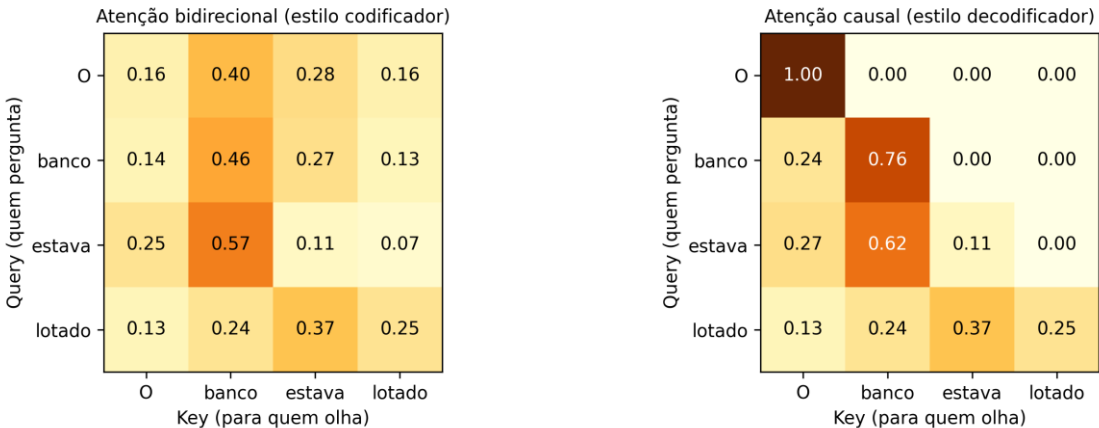
- QK^T : compatibilidade entre cada Query e cada Key.
- $\div \sqrt{d_k}$: evita que o softmax sature quando d_k é grande.
- **softmax**: pesos positivos que somam 1 por linha.
- $\times V$: média ponderada dos Values: o token contextualizado.

Atenção de “banco” em números

Passo (token “banco”)	O	banco	estava	lotado
Score q·k (QK ^T)	4,476	6,130	5,353	4,375
Score / √2	3,165	4,334	3,785	3,093
Peso (softmax)	0,143	0,459	0,265	0,133

Z = [1,043; 1,360]
saída para “banco”: média dos Values ponderada por 0,143 · 0,459 · 0,265 · 0,133

Figura 4 - Mapa de atenção softmax(QK^T/√d_k): cada linha soma 1,00



© Aridio Silva · @aridiosilva · Outubro 2026 · ORCID 0009-0008-2411-6995

Pesos escolhidos à mão: não leia significado linguístico no padrão. Num modelo treinado, Wq, Wk e Wv aprendem quais relações importam.

Várias relações em paralelo

$$\text{MultiHead}(X) = \text{Concat}(\text{head}^1, \dots, \text{head}^h) \cdot W^O$$

8 × 64

cabeças × d_k no Transformer base (d_model = 512)

1.048.576

pesos nas 4 projeções de atenção: $4 \times 512 \times 512$

h | d_model

h deve dividir d_model: 12 cabeças exigem d_model = 768 (BERT-base)

?

Por que é necessária

Uma só atenção dá uma única média por token e inibe subespaços diferentes; várias cabeças, em dimensão menor, não aumentam o custo total.

≈

O que se observa

Cabeças focam em delimitadores, posições fixas e relações sintáticas (Clark et al., 2019); muitas são redundantes (Voita et al., 2019; Michel et al., 2019).

Quem pode olhar para quem



Bidirecional

Cada posição vê todas as outras.
Regime do BERT (codificador).



Causal

Scores futuros recebem $-\infty$ e pesos 0.
Evita “ver” o token a prever
(vazamento).



Padding

Ignora posições vazias usadas para
igualar o tamanho dos lotes.

Linha de “banco”: [3,165; 4,334; 3,785; 3,093] → com máscara causal → [0,237; 0,763; 0; 0]

Em um modelo gerador, “banco” não pode olhar para “lotado”.

Inferência eficiente: MQA, GQA e KV cache



Multi-Query

Todas as cabeças compartilham um único par K e V (Shazeer, 2019).



Grouped-Query

Grupos de cabeças compartilham K e V: meio-termo entre qualidade e memória (Ainslie et al., 2023).



KV cache

Guarda os K e V já calculados e evita recomputar o passado a cada token (Pope et al., 2023).

Por que é necessária. Na geração, reler e recalcular todo o passado a cada token seria proibitivo; reduzir K e V diminui o cache e acelera a inferência.

Conexão residual e normalização

Residual

$y = x + f(x)$: cria um atalho para sinal e gradiente; cada subcamada aprende só uma correção (He et al., 2016).

LayerNorm / RMSNorm

Normalizam cada token nas `d_model` componentes (Ba et al., 2016). A RMSNorm omite a média (Zhang & Sennrich, 2019).

Pré-LN × Post-LN

Normalizar antes de cada subcamada (Pré-LN) treina de forma mais estável, sem aquecimento cuidadoso (Xiong et al., 2020).

Por que é necessária. Sem atalho, redes profundas degradam no treino; sem normalização, a escala dos valores deriva. A profundidade que dá capacidade ao modelo deixaria de ser treinável.

LayerNorm × RMSNorm para “banco”

Vetor	Valores (d_model = 4)
x (entrada)	[0,841; 1,540; 0,510; 1,000]
o (saída da atenção)	[1,074; 1,912; 0,537; 0,956]
r = x + o	[1,915; 3,453; 1,047; 1,956]
LayerNorm(r)	[−0,205; 1,572; −1,209; −0,158]
RMSNorm(r)	[0,846; 1,525; 0,462; 0,864]

$\mu = 2,093$

média de r

$\sigma^2 = 0,748$

variância de r

2,264

raiz do quadrático médio (RMS)

Após a LayerNorm: média 0 e variância 1. A RMSNorm mantém a proporção entre componentes, sem centralizar.

A rede feed-forward por token

$$FFN(x) = \max(0, x \cdot W^1 + b^1) \cdot W^2 + b^2$$



O que faz

A atenção mistura informação entre tokens; a FFN transforma a informação dentro de cada token: expande ($d_{ff} = 4 \times d_{model}$), aplica a não linearidade e comprime.



Por que é necessária

Na atenção, a saída é média de Values por projeções lineares; a única não linearidade é o softmax dos pesos. A FFN acrescenta transformação não linear por token.



Memória chave-valor

Colunas de $W1$ detectam padrões (chaves); linhas de $W2$ trazem a informação associada (valores). Hipótese de Geva et al. (2021).

$d_{model} = 512$



$d_{ff} = 2048 \cdot \text{ReLU}$



$d_{model} = 512$

Parâmetros e um cálculo didático

1.048.576

W1: 512 × 2048

1.048.576

W2: 2048 × 512

≈ 2/3

dos ≈ 3,15 milhões de pesos de uma camada estão na FFN (Geva et al., 2021)

Cálculo didático (d_model = 4, d_ff = 8, pesos aleatórios com semente 0)

Token	Entrada da FFN	Camada oculta (ReLU)	Saída da FFN
estava	[1,389; -0,901; -0,999; 0,511]	[0,907; 0,660; 0,330; 0; 0,623; 0; 1,541; 0,759]	[0,231; 0,954; 0,437; -0,291]
banco	[-0,205; 1,572; -1,209; -0,158]	[0; 0; 0; 0; 0; 0; 0; 0]	[0; 0; 0; 0]

“estava”: 6 de 8 unidades ativas. “banco”: todas zeradas pela ReLU, então a saída é 0 e o vetor atravessa a subcamada pelo caminho residual.

Empilhamento multicamadas

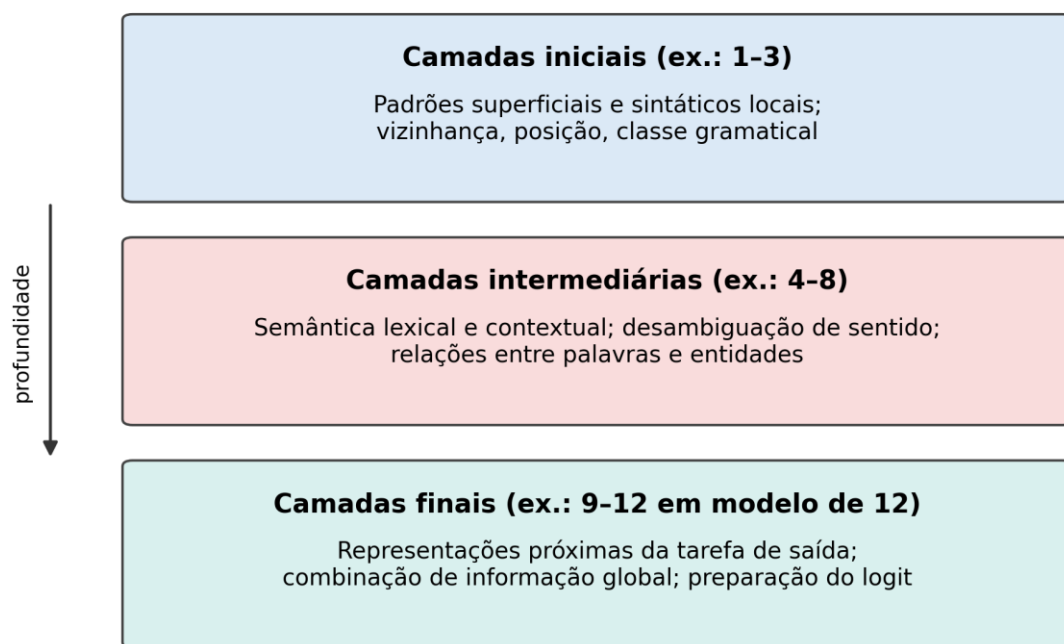
Modelo	Camadas	d_model	Cabeças	Parâmetros	Fonte
Transformer base	6 + 6	512	8	65 milhões	Vaswani et al., 2017
BERT-base	12	768	12	110 milhões	Devlin et al., 2019
BERT-large	24	1024	16	340 milhões	Devlin et al., 2019
GPT-3 (maior)	96	12.288	96	175 bilhões	Brown et al., 2020

Por que é necessária. Uma camada capta relações em um passo, mas muitas tarefas exigem composição: identificar, relacionar, inferir o sentido e só então prever. Algumas capacidades exigem ao menos duas camadas, como os induction heads (Elhage et al., 2021; Olsson et al., 2022). Mais camadas dão capacidade, mas custam mais: o número de camadas é escolha de projeto.

O que cada profundidade tende a representar

Figura 5 - Especialização aproximada por profundidade em Transformers empilhados

Esquema conceitual: tendência observada em estudos de sondagem em BERT (Tenney et al., 2019); as fronteiras entre camadas NÃO são exatas.



- **Sondagem:** camadas iniciais captam padrões locais; intermediárias, sintaxe; altas, semântica (Jawahar et al., 2019).
- **Pipeline clássico:** POS, parsing, entidades, papéis semânticos, correferência, nessa ordem (Tenney et al., 2019).
- **Refinamento:** a previsão do próximo token melhora camada a camada (Belrose et al., 2023).
- **Ressalva:** fronteiras variam por modelo, tarefa e método; “camada 6 resolve banco” é ilustração.

O objetivo: prever o próximo token

$$L = - (1/T) \sum^T \log p(x^t | x^{<t}) \quad PPL = e^L$$

3,806

perda se o alvo fosse “lotado” ($p = 0,022$)

0,473

perda se o alvo fosse “cheio” ($p = 0,623$)

$p - y$

gradiente da perda nos logits: $[-0,978; 0,114; 0,623; 0,094; 0,147]$ para o alvo “lotado”

?

Por que é necessária

Com pesos aleatórios a arquitetura não produz linguagem. O treino a transforma em modelo útil; o sinal vem do próprio texto, sem rótulos humanos.



BERT é diferente

Usa tokens mascarados e olha para os dois lados (Devlin et al., 2019). A desambiguação é estratégia para reduzir o erro de predição.

Como os pesos são ajustados



Retropropagação

Gradientes por regra da cadeia em todas as camadas (Rumelhart et al., 1986); necessária para milhões a bilhões de parâmetros.



Otimização

Adam (Kingma & Ba, 2015) com warm-up, decaimento, dropout e label smoothing (Vaswani et al., 2017); AdamW hoje (Loshchilov & Hutter, 2019).



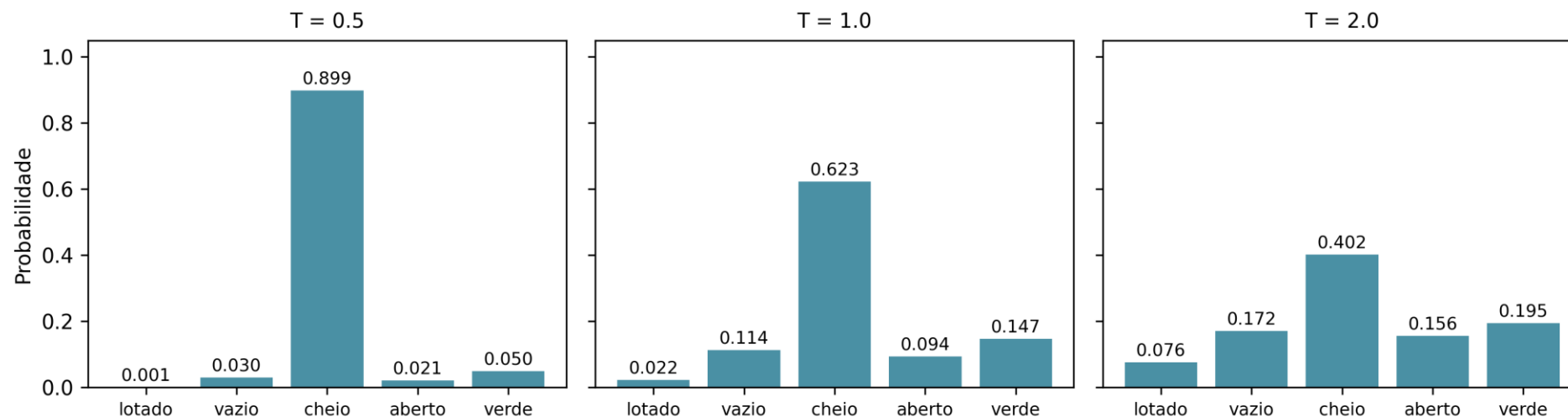
Escala

A perda cai em lei de potência com parâmetros, dados e computação (Kaplan et al., 2020); dados e parâmetros devem crescer juntos (Hoffmann et al., 2022).

Treino: todas as posições em paralelo (máscara causal). Inferência: um token por vez, com KV cache. Depois: pré-treino → fine-tuning → alinhamento (RLHF; Ouyang et al., 2022).

Logits, softmax e temperatura

Figura 6 - Efeito da temperatura sobre softmax(logits/T) (logits calculados; pesos de saída aleatórios)



© Aridio Silva · @aridiosilva · Outubro 2026 · ORCID 0009-0008-2411-6995

Logits (posição “estava”): lotado $-0,818$ · vazio $0,815$ · cheio $2,516$ · aberto $0,626$ · verde $1,070$

$T < 1$ concentra a probabilidade (mais determinístico); $T > 1$ a espalha. Pesos de saída aleatórios: em modelo treinado, “lotado” teria probabilidade alta.

Do vetor à palavra escolhida

Estratégia	Como funciona	No exemplo
Gulosa	Escolhe sempre o token mais provável	“cheio”
Amostragem	Sorteia conforme as probabilidades	lotado 0,022 ... cheio 0,623
Top-k (Fan et al., 2018)	Restringe aos k mais prováveis e renormaliza	k = 2: cheio 0,809 · verde 0,191
Top-p (Holtzman et al., 2020)	Menor conjunto com probabilidade acumulada $\geq p$	p = 0,9: {cheio, verde, vazio, aberto} $\rightarrow$ 0,637; 0,150; 0,117; 0,096
Beam search	Várias sequências candidatas; escolhe a de maior probabilidade total	—

Por que é necessária. Escolher sempre o mais provável tende a texto repetitivo; sortear sem restrição inclui tokens improváveis. Top-k e top-p equilibram os extremos.

Desambiguando “banco”

Brinquedo didático: 4 eixos [dinheiro, assento, pessoas, madeira]; atenção com $W_q = W_k = 2 \cdot I$, $W_v = I$; comparação por cosseno com dois protótipos de sentido.

Contexto	Vetor contextualizado	cos. instituição	cos. assento
Isolado: “banco”	[0,700; 0,700; 0; 0]	0,632	0,632
“O banco estava lotado de clientes”	[0,524; 0,351; 0,229; 0]	0,851	0,468
“O banco da praça estava velho de madeira”	[0,310; 0,509; 0,006; 0,175]	0,452	0,858

=

Isolado

Equidistante dos dois sentidos (0,632 × 0,632).

≠

Só mudam os vizinhos

O mecanismo é o mesmo: com clientes, aproxima-se da instituição; com praça e madeira, do assento.

Cuidados ao ler mapas de atenção

1 Atenção $\neq$ explicação

Pesos diferentes podem dar a mesma previsão; a atenção é só parte do cálculo (Jain & Wallace, 2019; Wiegreffe & Pinter, 2019).

2 Evidência empírica

Em modelos treinados, vetores da mesma palavra formam agrupamentos por sentido (Reif et al., 2019) e ficam mais específicos nas camadas altas (Ethayarajh, 2019).

3 Intervenção e circuitos

Saber o que o modelo “usa” exige experimentos de intervenção e análise de circuitos (Elhage et al., 2021; Olsson et al., 2022).

Ferramenta de visualização: Vig (2019).

Capacidades que ninguém programou

A Conceito

Propriedade do todo que não está em nenhum componente (Anderson, 1972). Nada no código fala de sentidos ou sintaxe.

W Com a escala

Habilidades ausentes em modelos menores e presentes em maiores (Wei et al., 2022a): few-shot, cadeia de pensamento.

S Debate

Parte dos saltos pode vir de métricas descontínuas (Schaeffer et al., 2023). O tema segue aberto.

G ≠ grokking

Grokking é generalização tardia no treino, em tarefas pequenas (Power et al., 2022; Nanda et al., 2023). Emergência por escala ocorre ao longo do tamanho do modelo.

B E o “banco”?

A desambiguação já aparece no BERT-base (110 milhões de parâmetros): não há limiar fixo de camadas. A capacidade vem do treino e cresce com a escala.

O que o documento não mostra

D Deste documento

- Exemplos com dimensões minúsculas, uma só cabeça, pesos à mão ou aleatórios.
- Omitidos: vieses, dropout, atenção cruzada, mixture of experts e alternativas à atenção, como modelos de espaço de estados (Gu & Dao, 2023).
- Detalhes de camadas, normalização e posição variam entre modelos.

M Do próprio mecanismo

- **Custo n^2** : contexto longo pode usar mal o meio do contexto (Liu et al., 2024).
- **Alucinação**: nenhum passo verifica a veracidade (Ji et al., 2023).
- **“Só estatística”?** Bender & Koller (2020) negam compreensão só por forma; Li et al. (2023) veem estados internos do domínio (Othello).
- **Emergência**: natureza dos saltos debatida.

O que levar daqui

- 1 Sem regra explícita
- 2 Cada etapa tem uma razão
- 3 Os pesos são o conhecimento
- 4 Emergência com cautela
- 5 Verifique as fontes

“Banco” é resolvido pelo contexto: atenção mistura vizinhos no vetor do token.

Números, vetores, ordem, contexto, estabilidade, não linearidade, profundidade e saída.

O treino ajusta pesos para prever o próximo token; a inferência só os usa.

Capacidades surgem do treino e da escala; o debate sobre saltos segue aberto.

12 de 64 referências conferidas em fonte; 14 links auditados. Os números são didáticos.

Termos e conceitos (1/8)

Termo	Definição	§
Adam / AdamW	Otimizadores com taxa adaptativa por parâmetro; o AdamW separa o decaimento de pesos.	10
ALiBi	Viés linear nos scores de atenção, proporcional à distância entre tokens.	4
Alucinação	Saída fluente, porém factualmente incorreta ou sem sustentação no contexto.	14
Atenção (attention)	Média ponderada de vetores, com pesos dados pela compatibilidade entre consulta e chaves.	5
Atenção cruzada	Queries de uma sequência (decodificador); Keys e Values de outra (codificador).	14
Autorregressivo	Gera um elemento por vez, condicionado aos já gerados.	2, 11
Backpropagation	Calcula gradientes da perda em todos os pesos, propagando o erro pela regra da cadeia.	10
Beam search	Busca em feixe: mantém várias sequências candidatas e escolhe a de maior probabilidade.	11
BERT	Modelo somente codificador, treinado com tokens mascarados e atenção bidirecional.	2, 10
BPE (Byte Pair Encoding)	Tokenização que funde iterativamente o par de símbolos adjacentes mais frequente.	3

Termos e conceitos (2/8)

Termo	Definição	§
Cabeça de atenção (head)	Uma das h atenções paralelas, com W_q , W_k e W_v próprias.	6
Camada (layer)	Um bloco Transformer: atenção, Add & Norm, FFN, Add & Norm.	9
Chain-of-thought	Prompt em que o modelo gera passos de raciocínio antes da resposta.	13
Codificação posicional	Vetor somado ao embedding para informar a posição do token.	4
Codificador / decodificador	Codificador lê com atenção bidirecional; decodificador gera com atenção causal.	2
Conexão residual	Soma da entrada de uma subcamada à sua saída: $y = x + f(x)$.	7
Contexto	Tokens que condicionam uma previsão (anteriores, ou toda a sequência se bidirecional).	5, 12
Cosseno (similaridade)	Alinhamento entre vetores: produto escalar dividido pelas normas, de -1 a 1 .	12
d_{model} , d_k , d_{ff}	Dimensão do modelo, de cada cabeça (d_{model}/h) e da camada oculta da FFN.	4, 6, 8
Decodificação	Regra que escolhe o token de saída (gulosa, top-k, top-p, feixe).	11

Termos e conceitos (3/8)

Termo	Definição	§
Dropout	Regularização que zera ativações aleatórias durante o treino.	10
Embedding	Vetor denso aprendido que representa um token (ou uma posição).	4
Emergência	Característica do sistema que surge da interação de muitos componentes, sem ser programada.	13
Entropia cruzada	Perda que mede o afastamento da distribuição prevista do token correto: $-\log p$.	10
Feed-forward (FFN)	Duas camadas lineares com não linearidade, aplicadas a cada posição.	8
Few-shot	Uso do modelo com poucos exemplos no próprio prompt, sem atualizar pesos.	13
Fine-tuning	Treino adicional, após o pré-treino, para uma tarefa ou comportamento.	10
Forward pass	Passagem da entrada pelas camadas até a saída, sem atualizar pesos.	2
GELU	Função de ativação suave, usada em BERT e GPT.	8
Gradiente	Derivadas da perda em relação aos pesos; indica como ajustá-los.	10

Termos e conceitos (4/8)

Termo	Definição	§
Grokking	Generalização tardia, muito depois da memorização, em tarefas algorítmicas pequenas.	13
Induction head	Circuito de duas cabeças que acha ocorrências passadas de um padrão e prevê a continuação.	9
Inferência	Execução do modelo treinado, com pesos fixos, para gerar uma saída.	2
Interpretabilidade	Métodos para entender o que o modelo computa e por quê.	12
Key (chave)	Projeção $X \cdot W_k$: o que cada token oferece para ser encontrado.	5
KV cache	Keys e Values já calculados, reaproveitados a cada novo token gerado.	6
Label smoothing	Suaviza o alvo no treino, reduzindo confiança extrema.	10
LayerNorm	Normalização por token nas dimensões, com escala γ e deslocamento β aprendidos.	7
Lei de escala	Relação regular (lei de potência) entre perda e parâmetros, dados e computação.	10, 13
Logit	Pontuação real, sem limites, por token do vocabulário, antes do softmax.	11

Termos e conceitos (5/8)

Termo	Definição	§
Logit lens / tuned lens	Projetam estados intermediários no vocabulário para ver a previsão por camada.	9
Máscara causal	Impede atender posições futuras, dando $-\infty$ aos scores correspondentes.	6
Masked language modeling	Tokens ocultos são reconstruídos a partir do contexto dos dois lados.	10
Mixture of experts (MoE)	Cada token ativa só um subconjunto de subredes FFN especializadas.	14
Multi-head attention	h atenções em paralelo em subespaços distintos, concatenadas e projetadas por W_o .	6
One-hot	Vetor com 1 na posição do token correto e 0 nas demais.	10
Parâmetro / peso	Número ajustável aprendido durante o treino.	10
Perplexidade (PPL)	Exponencial da perda média por token; mede a “surpresa” do modelo.	10
Pré-LN / Post-LN	Normalização antes (Pré-LN) ou depois (Post-LN) da soma residual.	7
Pré-treinamento	Fase inicial de treino em grande volume de texto bruto.	10

Termos e conceitos (6/8)

Termo	Definição	§
Probing (sondagem)	Classificadores simples sobre representações internas para ver o que contêm.	9
Produto escalar	Soma dos produtos componente a componente; mede alinhamento.	5
Query (consulta)	Projeção $X \cdot W_q$: o que cada token procura.	5
ReLU	Função de ativação $\max(0, x)$.	8
RLHF	Aprendizado por reforço com feedback humano, para alinhar a preferências.	10
RMSNorm	Normaliza pela raiz do valor quadrático médio, sem subtrair a média.	7
RoPE	Posição por rotação de Q e K, expressando posição relativa.	4
Self-attention	Atenção em que Queries, Keys e Values vêm da mesma sequência.	5
SentencePiece	Biblioteca de subpalavras (BPE ou Unigram) que opera sobre texto bruto.	3
Softmax	Converte números em probabilidades positivas que somam 1.	5, 11

Termos e conceitos (7/8)

Termo	Definição	§
SwiGLU	Ativação com porta (GLU) usando Swish, comum em modelos recentes.	8
Temperatura (T)	Divisor dos logits antes do softmax; controla a concentração da distribuição.	11
Token	Unidade do vocabulário (palavra, subpalavra, byte ou símbolo).	3
Token contextualizado	Vetor de um token após a atenção, com informação dos demais tokens.	5
Tokenização	Conversão de texto em sequência de tokens e IDs.	3
Top-k / Top-p (nucleus)	Amostragem restrita aos k mais prováveis ou ao menor conjunto com probabilidade acumulada p.	11
Transformer	Arquitetura baseada em atenção, sem recorrência nem convolução.	1, 2
Unigram LM	Tokenização que parte de um vocabulário grande e poda itens pouco úteis.	3
Value (valor)	Projeção $X \cdot W_v$: o conteúdo repassado por cada token.	5
Vocabulário	Conjunto finito de tokens reconhecidos pelo modelo.	3

Termos e conceitos (8/8)

Termo	Definição	§
Warm-up	Fase inicial do treino com taxa de aprendizado crescente.	7, 10
Weight tying	Compartilha a matriz de embeddings com a projeção de saída.	4, 11
WordPiece	Subpalavras com fusões escolhidas pelo ganho de verossimilhança.	3

Fontes e referências (1/8)

#	Referência	Link	✓ •
1	Ainslie et al. (2023) . GQA: Training generalized multi-query Transformer models from multi-head checkpoints. EMNLP 2023	arxiv.org/abs/2305.13245	
2	Anderson (1972) . More is different. Science, 177(4047), 393–396	doi.org/10.1126/science.177.4047.393	
3	Ba et al. (2016) . Layer normalization	arxiv.org/abs/1607.06450	
4	Bahdanau et al. (2015) . Neural machine translation by jointly learning to align and translate. ICLR 2015	arxiv.org/abs/1409.0473	
5	Belrose et al. (2023) . Eliciting latent predictions from Transformers with the tuned lens	arxiv.org/abs/2303.08112	✓ •
6	Bender et al. (2020) . Climbing towards NLU: On meaning, form, and understanding in the age of data. ACL 2020, p. 5185–5198	aclanthology.org/2020.acl-main.463	
7	Bengio et al. (2003) . A neural probabilistic language model. Journal of Machine Learning Research, 3, 1137–1155	www.jmlr.org/papers/v3/bengio03a.html	
8	Brown et al. (2020) . Language models are few-shot learners. NeurIPS 33	arxiv.org/abs/2005.14165	

✓ conferido em fonte • link auditado (demais: não verificados neste ambiente; ver nota da seção 16)

Fontes e referências (2/8)

#	Referência	Link	✓ •
9	Clark et al. (2019). What does BERT look at?. An analysis of BERT's attention. BlackboxNLP Workshop at ACL 2019, p. 276–286	arxiv.org/abs/1906.04341	•
10	Dao et al. (2022). FlashAttention: Fast and memory-efficient exact attention with IO-awareness. NeurIPS 35	arxiv.org/abs/2205.14135	
11	Devlin et al. (2019). BERT: Pre-training of deep bidirectional Transformers for language understanding. NAACL-HLT 2019, p. 4171–4186	arxiv.org/abs/1810.04805	
12	Elhage et al. (2021). A mathematical framework for Transformer circuits. Transformer Circuits Thread	transformer-circuits.pub/2021/framework/index.html	
13	Ethayarajh (2019). How contextual are contextualized word representations?. Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. EMNLP-IJCNLP 2019, p. 55–65	aclanthology.org/D19-1006	✓ •
14	Fan et al. (2018). Hierarchical neural story generation. ACL 2018, p. 889–898	arxiv.org/abs/1805.04833	
15	Gage (1994). A new algorithm for data compression. The C Users Journal, 12(2), 23–38	www.derczynski.com/papers/archive/BPE_Gage.pdf	
16	Ganguli et al. (2022). Predictability and surprise in large generative models. ACM FAccT 2022, p. 1747–1764	arxiv.org/abs/2202.07785	

✓ conferido em fonte • link auditado (demais: não verificados neste ambiente; ver nota da seção 16)

Fontes e referências (3/8)

#	Referência	Link	✓ •
17	Geva et al. (2021). Transformer feed-forward layers are key-value memories. EMNLP 2021, p. 5484–5495	aclanthology.org/2021.emnlp-main.446	✓ •
18	Gu et al. (2023). Mamba: Linear-time sequence modeling with selective state spaces	arxiv.org/abs/2312.00752	
19	He et al. (2016). Deep residual learning for image recognition. CVPR 2016, p. 770–778	arxiv.org/abs/1512.03385	
20	Hendrycks et al. (2016). Gaussian error linear units (GELUs)	arxiv.org/abs/1606.08415	
21	Hinton et al. (2015). Distilling the knowledge in a neural network	arxiv.org/abs/1503.02531	
22	Hoffmann et al. (2022). Training compute-optimal large language models. NeurIPS 35	arxiv.org/abs/2203.15556	
23	Holtzman et al. (2020). The curious case of neural text degeneration. ICLR 2020	arxiv.org/abs/1904.09751	
24	Jain et al. (2019). Attention is not explanation. NAACL-HLT 2019, p. 3543–3556	aclanthology.org/N19-1357	✓ •

✓ conferido em fonte • link auditado (demais: não verificados neste ambiente; ver nota da seção 16)

Fontes e referências (4/8)

#	Referência	Link	✓ •
25	Jawahar et al. (2019). What does BERT learn about the structure of language?. ACL 2019, p. 3651–3657	aclanthology.org/P19-1356	✓ •
26	Ji et al. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), art. 248	arxiv.org/abs/2202.03629	•
27	Kaplan et al. (2020). Scaling laws for neural language models	arxiv.org/abs/2001.08361	
28	Kingma et al. (2015). Adam: A method for stochastic optimization. ICLR 2015	arxiv.org/abs/1412.6980	
29	Kudo (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. ACL 2018, p. 66–75	arxiv.org/abs/1804.10959	
30	Kudo et al. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. EMNLP 2018 (System Demonstrations), p. 66–71	arxiv.org/abs/1808.06226	
31	Li et al. (2023). Emergent world representations: Exploring a sequence model trained on a synthetic task. ICLR 2023	arxiv.org/abs/2210.13382	
32	Liu et al. (2024). Lost in the middle: How language models use long contexts. Transactions of the ACL, 12, 157–173	aclanthology.org/2024.tacl-1.9	✓ •

✓ conferido em fonte • link auditado (demais: não verificados neste ambiente; ver nota da seção 16)

Fontes e referências (5/8)

#	Referência	Link	✓ •
33	Loshchilov et al. (2019). Decoupled weight decay regularization. ICLR 2019	arxiv.org/abs/1711.05101	
34	Michel et al. (2019). Are sixteen heads really better than one?. NeurIPS 32	arxiv.org/abs/1905.10650	
35	Mikolov et al. (2013). Efficient estimation of word representations in vector space	arxiv.org/abs/1301.3781	
36	Nanda et al. (2023). Progress measures for grokking via mechanistic interpretability. ICLR 2023	arxiv.org/abs/2301.05217	
37	Olsson et al. (2022). In-context learning and induction heads. Transformer Circuits Thread	arxiv.org/abs/2209.11895	
38	Ouyang et al. (2022). Training language models to follow instructions with human feedback. NeurIPS 35	arxiv.org/abs/2203.02155	
39	Pope et al. (2023). Efficiently scaling Transformer inference. Proceedings of Machine Learning and Systems, 5 (MLSys 2023)	arxiv.org/abs/2211.05102	✓ •
40	Power et al. (2022). Grokking: Generalization beyond overfitting on small algorithmic datasets	arxiv.org/abs/2201.02177	

✓ conferido em fonte • link auditado (demais: não verificados neste ambiente; ver nota da seção 16)

Fontes e referências (6/8)

#	Referência	Link	✓ ●
41	Press et al. (2022). Train short, test long: Attention with linear biases enables input length extrapolation. ICLR 2022	arxiv.org/abs/2108.12409	
42	Press et al. (2017). Using the output embedding to improve language models. EACL 2017, p. 157–163	arxiv.org/abs/1608.05859	
43	Radford et al. (2018). Improving language understanding by generative pre-training. OpenAI, relatório técnico	cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf	
44	Radford et al. (2019). Language models are unsupervised multitask learners. OpenAI, relatório técnico	cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf	
45	Reif et al. (2019). Visualizing and measuring the geometry of BERT. NeurIPS 32, p. 8594–8603	arxiv.org/abs/1906.02715	✓ ●
46	Rumelhart et al. (1986). Learning representations by back-propagating errors. Nature, 323, 533–536	doi.org/10.1038/323533a0	
47	Schaeffer et al. (2023). Are emergent abilities of large language models a mirage?. NeurIPS 2023 (v. 36)	arxiv.org/abs/2304.15004	✓ ●
48	Schuster et al. (2012). Japanese and Korean voice search. ICASSP 2012, p. 5149–5152	doi.org/10.1109/ICASSP.2012.6289079	

✓ conferido em fonte ● link auditado (demais: não verificados neste ambiente; ver nota da seção 16)

Fontes e referências (7/8)

#	Referência	Link	✓ •
49	Sennrich et al. (2016). Neural machine translation of rare words with subword units. ACL 2016, p. 1715–1725	arxiv.org/abs/1508.07909	
50	Shaw et al. (2018). Self-attention with relative position representations. NAACL-HLT 2018, p. 464–468	arxiv.org/abs/1803.02155	
51	Shazeer (2019). Fast Transformer decoding: One write-head is all you need	arxiv.org/abs/1911.02150	
52	Shazeer (2020). GLU variants improve Transformer	arxiv.org/abs/2002.05202	
53	Su et al. (2021). RoFormer: Enhanced Transformer with rotary position embedding	arxiv.org/abs/2104.09864	
54	Tenney et al. (2019). BERT rediscovers the classical NLP pipeline. ACL 2019, p. 4593–4601	aclanthology.org/P19-1452	✓ •
55	Touvron et al. (2023). LLaMA: Open and efficient foundation language models	arxiv.org/abs/2302.13971	
56	Vaswani et al. (2017). Attention is all you need. Advances in Neural Information Processing Systems 30 (NIPS 2017), p. 5998–6008	papers.neurips.cc/paper/7181-attention-is-all-you-need	✓ •

✓ conferido em fonte • link auditado (demais: não verificados neste ambiente; ver nota da seção 16)

Fontes e referências (8/8)

#	Referência	Link	✓ •
57	Vig (2019) . A multiscale visualization of attention in the Transformer model. ACL 2019 (System Demonstrations), p. 37–42	arxiv.org/abs/1906.05714	
58	Voita et al. (2019) . Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. ACL 2019, p. 5797–5808	arxiv.org/abs/1905.09418	
59	Wei et al. (2022a) . Emergent abilities of large language models. Transactions on Machine Learning Research	arxiv.org/abs/2206.07682	✓ •
60	Wei et al. (2022b) . Chain-of-thought prompting elicits reasoning in large language models. NeurIPS 35	arxiv.org/abs/2201.11903	
61	Wiegrefe et al. (2019) . Attention is not not explanation. EMNLP-IJCNLP 2019, p. 11–20	arxiv.org/abs/1908.04626	
62	Wu et al. (2016) . Google's neural machine translation system: Bridging the gap between human and machine translation	arxiv.org/abs/1609.08144	
63	Xiong et al. (2020) . On layer normalization in the Transformer architecture. ICML 2020 (PMLR 119)	arxiv.org/abs/2002.04745	
64	Zhang et al. (2019) . Root mean square layer normalization. NeurIPS 32	arxiv.org/abs/1910.07467	

✓ conferido em fonte • link auditado (demais: não verificados neste ambiente; ver nota da seção 16)