Attention Models: da motivação às variantes modernas

Intuição geométrica, formulação matemática e aplicações

Rodrigo Petrus Domingues, Felipe Grael e Vivian

17 de agosto de 2025

Table of Contents

Redes Neurais Recorrentes

1 Motivação e Histórico

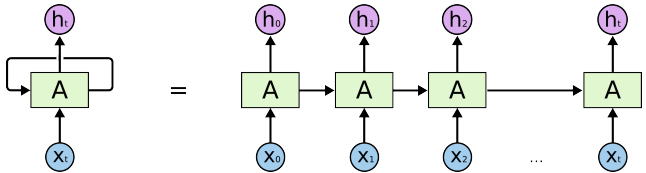


Figura: Redes Neurais Recorrentes [1]

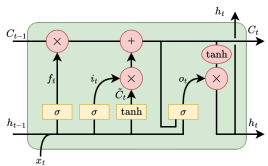
- Bem adaptadas para dados sequenciais como séries temporais e texto
- Diferentes tipos de modelos: LSTM, GRU
- Diferentes arquiteturas: simples, bidirecional, encoder-decoder

[1] Figura de Christopher Olah

Tipos de camadas RNN

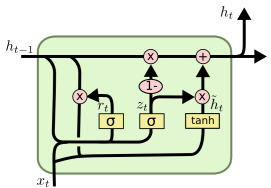
1 Motivação e Histórico

LSTM



$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \qquad i_t = \sigma(W_i[h_{t-1}, x_t] + b_i),$$
$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c), \quad C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t,$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \qquad h_t = o_t \odot \tanh(C_t)$$

GRU



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t])$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

LSTM como baseline para NMT

- Encoder LSTM lê $(x_1, \ldots, x_n)$ e produz estados h_t; o contexto é o último estado $c = h_n$.
- Decoder LSTM gera $(y_1, \ldots, y_m)$ condicionado a c.
- **Gargalo**: toda a informação comprimida em $c = h_n$.
  - Processamento **sequencial** $\Rightarrow$ baixa paralelização.
  - **Dependências longas** ainda são difíceis (mesmo com portas).
  - **Gargalo do contexto** (vetor único) degrada qualidade em frases longas.

Atenção aditiva [1]

$$e_{ij} = a(s_{i-1}, h_j), \quad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \quad c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

- h_j: estado oculto do encoder na posição j (palavra x_j).
- s_{i-1}: estado do decoder no passo anterior (y_{i-1}).
- e_{ij}: escore de alinhamento entre h_j e s_{i-1} (via rede feedforward).
- α_{ij}: pesos normalizados (softmax) $\rightarrow$ distribuem a atenção sobre os h_j.
- c_i: vetor de contexto dinâmico usado para prever y_i.

Intuição: O decoder calcula, em cada passo, um mapa de atenção sobre os estados do encoder, decidindo onde focar.

Integração com encoder bidirecional e decoder

- O **encoder** é uma RNN bidirecional:

$$h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$$

  Cada h_j contém contexto passado e futuro da palavra x_j.
- O vetor de contexto c_i é construído a partir desses estados bidirecionais.
- O **decoder** (RNN unidirecional) atualiza seu estado com:

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

  - Usa o estado anterior s_{i-1} - O símbolo anterior y_{i-1} - O contexto dinâmico c_i
- Assim, a cada passo, o decoder combina memória interna + contexto dinâmico para prever y_i.

Resultado: Resolve o gargalo do vetor fixo único (h_n) e permite traduções mais fiéis em frases longas.

Atenção multiplicativa [2]

Três variantes de scoring:

$$e_{ij} = v^\top \tanh(W[s_j; h_i]) \quad \text{(concat, similar ao Bahdanau)}$$
$$e_{ij} = s_j^\top W h_i \quad \text{(general)}$$
$$e_{ij} = s_j^\top h_i \quad \text{(dot)}$$

- s_j: query → estado oculto do decoder.
- h_i: key/value → estado do encoder.
- **Concat:** aproxima-se da atenção aditiva de Bahdanau.
- **General:** bilinear, mais expressivo (aprende W).
- **Dot:** mais simples e rápido (nenhum parâmetro extra).

Nota: Podemos reinterpretar em termos modernos como $Q = s_j, K = h_i, V = h_i$.

Global vs Local Attention [2]

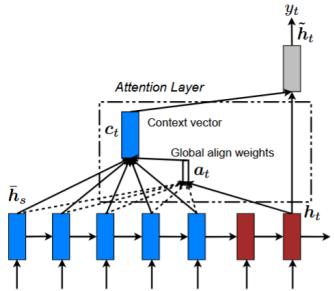


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector $\boldsymbol{a}_t$ based on the current target state $\boldsymbol{h}_t$ and all source states $\bar{\boldsymbol{h}}_s$. A global context vector $\boldsymbol{c}_t$ is then computed as the weighted average, according to $\boldsymbol{a}_t$, over all the source states.

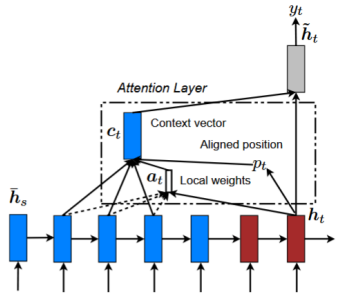


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights $\boldsymbol{a}_t$ are inferred from the current target state $\boldsymbol{h}_t$ and those source states $\bar{\boldsymbol{h}}_s$ in the window.

Self-Attention: dependências em paralelo

$$\mathrm{Att}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \quad Q = XW_Q,\ K = XW_K,\ V = XW_V.$$

- Calcula relações *entre todos os tokens* da mesma sequência, **em paralelo**.
- Multi-head:

$$\mathrm{MHA}(X) = \mathrm{Concat}(H_1,\ldots,H_h)\,W_O, \quad H_r = \mathrm{softmax}\left(\frac{Q_r K_r^\top}{\sqrt{d_k}}\right)V_r.$$

- Comparativo: RNN/LSTM exige n passos sequenciais; self-attention faz um passo paralelo com custo $\mathcal{O}(n^2)$.

Arquitetura pré-transformer

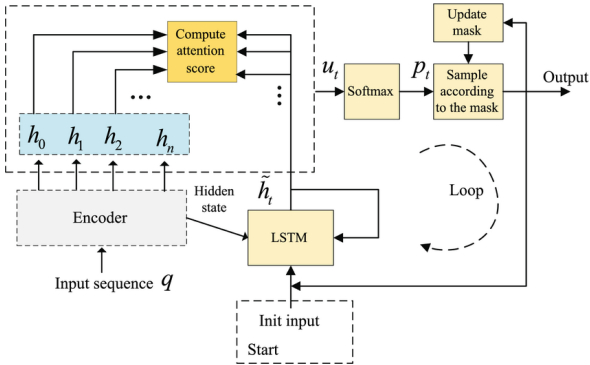


Figura: Exemplo de arquitetura pré-transformer (aditiva) [3]

Table of Contents

2 Transformers

Attention Is All You Need [4]: nascendo o Transformer

2 Transformers

- **Remove** completamente a recorrência (sem LSTM).
- **Positional encodings** preservam ordem:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right).$$

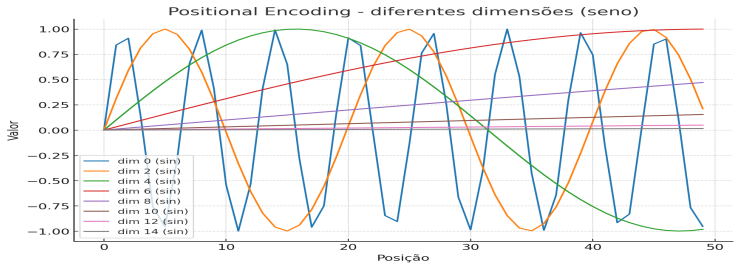


Figura: Exemplo de codificação posicional senoidal [4]

Bloco Transformer e arquitetura

- Cada **bloco Transformer** (pré-norm, forma comum):

$$Y = X + \mathrm{MHA}(\mathrm{LN}(X)),$$
$$Z = Y + \mathrm{FFN}(\mathrm{LN}(Y)), \quad \mathrm{FFN}(u) = W_2\,\phi(W_1 u + b_1) + b_2,$$

- Empilha-se vários blocos de atenção+FFN $\Rightarrow$ **arquitetura Transformer**.
- **Máscara causal** (para LMs) impede olhar o futuro:

$$\mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right), \quad M_{ij} = \begin{cases} 0, & j \le i \\ -\infty, & j > i \end{cases}$$

Arquiteturas Transformer e Aplicações

Encoder–Decoder
(Transformer original, 2017)

- Tradução automática
- Sumarização
- Diálogo
- Captioning

Encoder-only
(BERT, RoBERTa, DistilBERT)

- Classificação de texto
- NER (entidades)
- QA (extração de trechos)
- Análise semântica

Decoder-only
(GPT, LLaMA, etc.)

- Modelos de linguagem
- Geração de texto
- Completamento de prompts
- Story generation

Arquiteturas Transformer: encoder e decoder [4]

2 Transformers

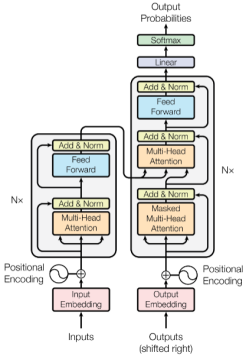


Figura: Transformer Encoder e Decoder

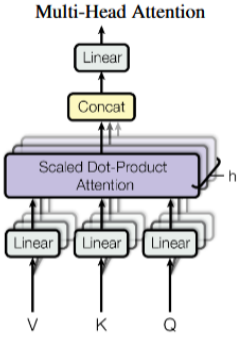


Figura: Multi-Head Attention

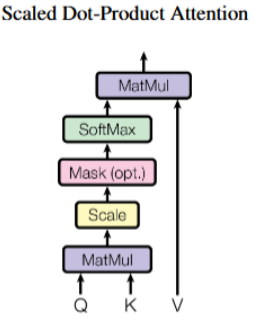


Figura: Multi-Head Attention

Resumo da evolução dos modelos

Linha do tempo (síntese):

- LSTM encoder–decoder: contexto único $c = h_n$ (gargalo).
- LSTM + **atenção** (Bahdanau/Luong): alívio do gargalo.
- **Self-attention**: dependências longas em paralelo.
- **Transformer**: atenção + posição + FFN; várias camadas empilhadas; sem LSTM.

Table of Contents

3 Embeddings & Interpretações

Word/Subword Embeddings

3 Embeddings & Interpretações

- Estáticos (Word2Vec, GloVe) vs. Contextuais (ELMo, BERT)
- Subword (BPE/Unigram) para robustez morfológica
- Análogos em outras modalidades: patches (ViT), time2vec (TS), node2vec (grafos)

Atenção: projeções e compatibilidade

3 Embeddings & Interpretações

- **Projeções lineares**:
  $Q = XW_Q,\ K = XW_K,\ V = XW_V.$
- **Compatibilidade**: $e_i = \frac{\langle q, k_i \rangle}{\sqrt{d}}$ (ou cosseno se normalizar).
- **Pesos**:
  $\alpha_i = \mathrm{softmax}(e_i) \Rightarrow \alpha_i \geq 0,\ \sum_i \alpha_i = 1.$

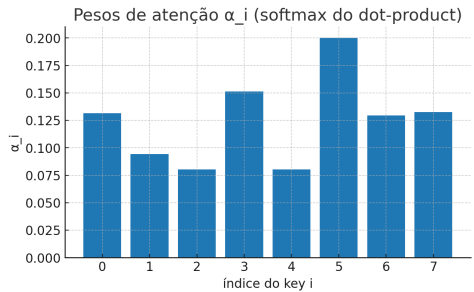
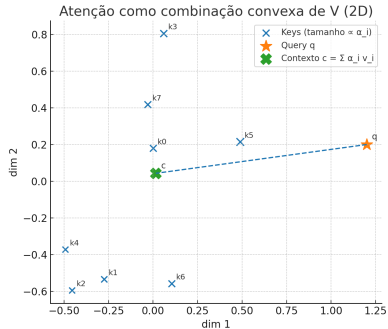


Figura: Pesos α_i (softmax do dot-product)

Interpretação Geométrica: contexto = barycenter

- **Contexto**: $c = \sum_i \alpha_i v_i \in \mathrm{conv}\{v_i\} \Rightarrow$ combinação *convexa* dos values.
- **Geometria**: níveis de igual peso são hiperplanos ortogonais a q; pesos crescem exponencialmente com o alinhamento entre q e k_i.



Multi-head: múltiplas projeções/métricas

3 Embeddings & Interpretações

- Cada head aplica $(W_Q^{(h)}, W_K^{(h)}, W_V^{(h)})$ e induz uma métrica interna distinta: $\langle x, y \rangle_h = x^\top W_Q^{(h)} {W_K^{(h)}}^\top y$.
- Heads diferentes → diferentes pesos $\alpha^{(h)}$ e contextos $c^{(h)}$; a saída concatena/combina esses subespaços.

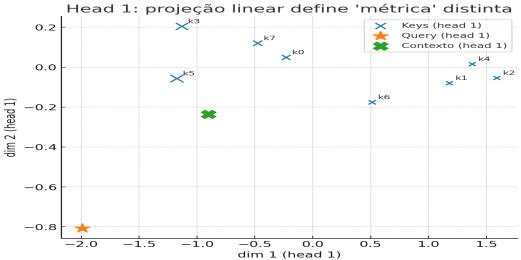


Figura: Head 1

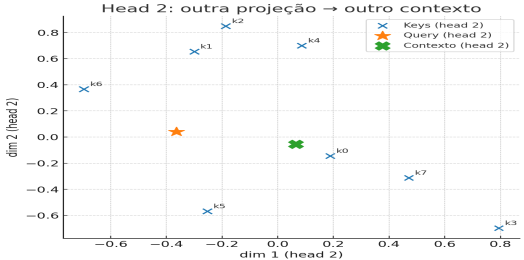


Figura: Head 2

Interpretação Matemática: Self-Attention

Fórmula central (scaled dot-product)

$$\mathrm{Att}(Q,K,V) \;=\; \mathrm{softmax}\!\left(\frac{QK^{\top}}{\sqrt{d_k}}\right) V$$

- Máscara causal/atencional conforme a tarefa
- Complexidade $\mathcal{O}(n^2)$ em tempo/memória
- Gradientes e saturação da softmax

Positional Encodings

3 Embeddings & Interpretações

- Absolutos senoidais vs. aprendidos
- Relativos e RoPE (rotary): melhor generalização/composição
- Impacto em extrapolação e contextos longos

Table of Contents

4 Treino & Hiperparâmetros

Práticas de Treino

4 Treino & Hiperparâmetros

- AdamW, warmup + decaimento; label smoothing quando aplicável
- Regularização: dropout, stochastic depth, weight decay
- AMP/mixed precision, grad clipping, checkpointing
- Dados: curriculum, masking, augmentation (TS/ViT/GAT)

Hiperparâmetros Essenciais

- Profundidade, d_{model}, #heads, d_{ff}, dropout
- Comprimento de contexto, batch size, *LR schedule*
- Específicos: tokenização (LM), *patch size* (ViT), janela/patch (TS)

Qual tamanho ideal? (Scaling)

- **Leis de escala [5]:** desempenho cresce com ↑ dados, parâmetros e compute, até saturar.
- **Trade-offs:** muitos parâmetros + poucos dados → overfitting; muitos dados + poucos parâmetros → subutilização.
- **Tokens vs. parâmetros:** ideal quando #tokens ≈ múltiplos de #parâmetros [6].
- **Contexto:** janelas maiores ajudam, mas ganhos saturam.
- **Regra prática:** dimensione conforme dataset + compute + budget de inferência.

Resumo: não há tamanho ótimo universal — depende do equilíbrio entre dados, parâmetros e recursos disponíveis.

Table of Contents

5 Séries Temporais

Atenção em Séries Temporais

5 Séries Temporais

- **Codificação temporal:** absolutos/relativos; *Time2Vec*; embeddings de calendário (hora/dia/sazonalidade).
- **Exógenas + cross-attention:** integrar variáveis externas (clima, preços, feriados) à série-alvo.
- **Contextos longos:** *patching* (janelas), sparsity e hashing; ganhos de janelas muito grandes tendem a saturar.
- **Por que atenção:** foca nos trechos relevantes do histórico e alinha padrões não estacionários/irregulares.
- **Tarefas:** previsão (forecasting), imputação (faltantes) e detecção de anomalias.

Modelos recentes (com código)

5 Séries Temporais

- Informer (AAAI'21) — atenção esparsa p/ sequências longas.
- Autoformer (NeurIPS'21) — auto-correlação + decomposição.
- PatchTST (ICLR'23) — *patching* / channel-independent.
- TimesNet (ICLR'23) — modelagem de variação temporal 2D.
- TSDiffusion (2023) - difusão para séries temporais.

Table of Contents

6 Language Models

Language Models (LM)

- **Atenção causal** (*masked self-attention*): máscara triangular impede olhar o futuro.
- **Objetivo autoregressivo**: minimizar $-\sum_t \log p(x_t \mid x_{<t})$ (*next-token*).
- **Pré-treino** (corpus amplo, tarefa genérica) **vs. fine-tuning** (tarefa/estilo): *instruction tuning* (FLAN), RLHF (InstructGPT).
- **PEFT** (ajuste eficiente): *LoRA*, *Adapters* (congelar base + poucos params).
- **Métricas**: *perplexity* (surpresa média por token) e *downstream* (ex.: SuperGLUE, MMLU).

Language Models — Implementações (GitHub)

- karpathy/nanoGPT **(2022)** — GPT minimalista
- huggingface/transformers **(2019)** — biblioteca SOTA
- EleutherAI/gpt-neox **(2022)** — treino LLMs em larga escala
- meta-llama/llama **(2023)** — código/pesos Llama
- harvardnlp/annotated-transformer **(2018)** — Transformer anotado
- huggingface/trl **(2020)** — SFT, PPO/DPO, RLHF
- DeepSpeed-Chat **(2023)** — pipeline RLHF
- OpenRLHF/OpenRLHF **(2023)** — RLHF escalável
- huggingface/peft **(2023)** — PEFT (LoRA, QLoRA, etc.)
- adapter-hub/adapters **(2020)** — biblioteca de Adapters
- microsoft/LoRA **(2021)** — implementação LoRA oficial
- google-research/FLAN **(2021)** — dados p/ instruction tuning

Table of Contents

7 Vision Transformer

Vision Transformer (ViT)

- Imagem $\rightarrow$ *patches* + `[CLS]` token
- Posicionais 2D; augmentations (RandAug, Mixup/CutMix)
- Transfer: *linear probe* vs. *fine-tune*

Table of Contents

8 Graph Attention

Graph Attention Networks (GAT)

8 Graph Attention

Coeficientes de Atenção (um cabeçalho)

$$\alpha_{ij} = \mathrm{softmax}_j\big(\mathrm{LeakyReLU}\big(a^\top [Wh_i \,||\, Wh_j]\big)\big)$$

- Multi-head; sobre-*smoothing* e escalabilidade
- Heterógrafos e atenção relacional

Table of Contents

9 Referências Bibliográficas

Referências Bibliográficas

9 Referências Bibliográficas

[1] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[2] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1412–1421, 2015.

[3] S. Gu and Y. Zhuang, “Method for solving constrained 0-1 quadratic programming problems based on pointer network and reinforcement learning,” *Neural Computing and Applications*, vol. 35, 2022.

Referências Bibliográficas

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[5] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020.

[6] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, J. Noland, K. Millican, G. v. d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, “Training compute-optimal large language models,” *arXiv preprint arXiv:2203.15556*, 2022.