A revolução dos Transformers

Attention Is All You Need

Até 2017, a forma dominante de lidar com dados em sequência era por meio das **redes neurais recorrentes (RNNs)** e suas variantes, como **LSTM** e **GRU**. Essas redes processavam uma entrada de cada vez, mantendo uma espécie de memória interna que permitia relacionar o presente com o passado — por exemplo, lembrando palavras anteriores ao traduzir uma frase. Apesar de eficazes, elas apresentavam sérias limitações. A execução era **sequencial**, dificultando o paralelismo; a memória tendia a **esquecer dependências distantes**; e o treinamento era **lento e custoso**.

A busca por alternativas levou os pesquisadores da Google Brain a uma ideia ousada: eliminar completamente a recorrência. Assim nasceu o **Transformer**, apresentado no artigo *"Attention Is All You Need"* por Vaswani et al. (2017) . A inovação central foi substituir o processamento sequencial por um mecanismo de **atenção** capaz de aprender relações entre qualquer parte de uma sequência, sem depender da ordem temporal.

Atenção é Tudo o que Você Precisa

O princípio do Transformer é simples e poderoso: cada elemento da sequência deve ser capaz de "prestar atenção" a todos os outros. Essa operação, chamada de **self-attention**, faz com que o modelo aprenda quais palavras influenciam umas às outras, independentemente da distância. Assim, em "o banco da praça", a palavra "banco" aprende a se relacionar mais fortemente com "praça", enquanto em "o banco de dados" associa-se a "dados".

O mecanismo de atenção é descrito de forma matemática, mas sua intuição é acessível: cada palavra é representada por três vetores — **Query (Q)**, **Key (K)** e **Value (V)**.

- **Query** representa o que a palavra está procurando.
- **Key** representa a identidade de cada palavra.
- **Value** contém a informação a ser compartilhada.

O cálculo que determina o quanto cada palavra deve influenciar as demais é expresso pela equação:

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

Em termos práticos, isso corresponde a calcular a semelhança entre cada Query e todos os Keys, normalizar os resultados com *softmax* e, em seguida, somar os Values ponderados por esses pesos de atenção.

Um exemplo simples em PyTorch ilustra o conceito:

```python
import torch
import torch.nn.functional as F

def attention(Q, K, V):
    scores = Q @ K.T / (K.size(-1) ** 0.5)
    weights = F.softmax(scores, dim=-1)
    return weights @ V
```

Esse pequeno trecho de código resume a essência do Transformer: identificar o que deve ser relevante em cada contexto e combinar as informações de forma ponderada.

Multi-Head Attention: Diversidade de Perspectivas

Um único mecanismo de atenção capta apenas um tipo de relação entre palavras. Para capturar diferentes padrões — sintáticos, semânticos e posicionais — o Transformer utiliza **várias cabeças de atenção** que operam em paralelo. Cada uma observa a sequência sob um ângulo distinto, e o resultado combinado oferece uma compreensão muito mais rica do contexto.

O PyTorch oferece essa funcionalidade por meio da classe nn.MultiheadAttention:

```python
attn = torch.nn.MultiheadAttention(embed_dim=512, num_heads=8)
```

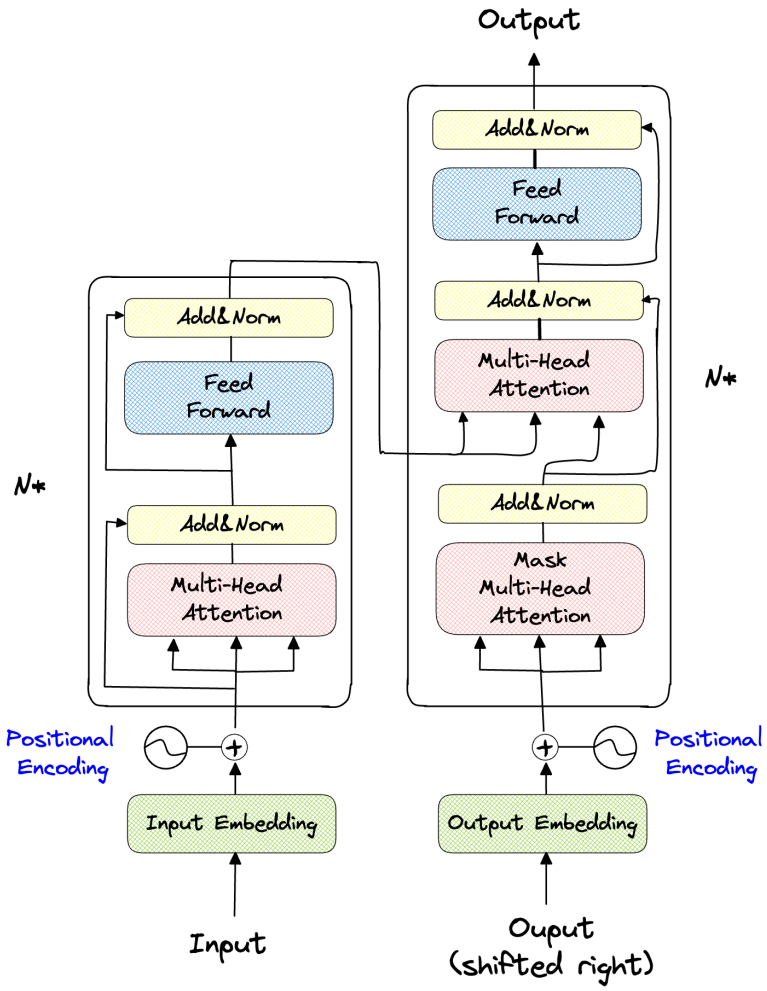
Essas múltiplas "cabeças" refletem a capacidade do modelo de analisar simultaneamente aspectos variados da mesma frase — por exemplo, uma cabeça pode focar em sujeito e verbo, enquanto outra identifica complementos e preposições.

Arquitetura: Encoder e Decoder

A estrutura do Transformer é organizada em duas partes principais: **encoder** e **decoder**. O **encoder** recebe a sequência de entrada e gera representações internas que sintetizam o significado de cada palavra em contexto. O **decoder**, por sua vez, utiliza essas representações para gerar a saída — como traduzir a frase para outro idioma. A **Figura 1** ilustra essa estrutura de forma esquemática, evidenciando o empilhamento de camadas e o papel dos mecanismos de atenção e normalização no fluxo de informação entre as duas partes.

O diagrama apresenta o fluxo completo de processamento da arquitetura proposta por Vaswani et al. (2017), com destaque para as camadas empilhadas de *Encoder* (à esquerda) e *Decoder* (à direita). Cada módulo contém mecanismos de atenção múltipla (*Multi-Head Attention*), redes totalmente conectadas (*Feed Forward*), normalização e conexões residuais (*Add & Norm*). As entradas e saídas são enriquecidas com *Positional Encoding*, garantindo que o modelo reconheça a ordem das sequências.

Figure 1 Arquitetura geral do modelo Transformer.



Fonte: modelo de ilustração Obsidian adaptado de *Attention Is All You Need* (Vaswani et al., 2017).

Cada uma dessas partes é composta por várias camadas idênticas, contendo três componentes: o mecanismo de atenção múltipla, uma rede totalmente conectada (*feed-forward network*) e uma combinação de normalização e conexões residuais. Essas conexões residuais permitem que o modelo aprenda de forma mais estável, reduzindo o risco de perda de gradiente. A profundidade das camadas, normalmente seis no encoder e seis no decoder, é o que confere ao modelo sua alta expressividade.

Positional Encoding: Dando Sentido à Ordem

Como o Transformer não processa palavras de maneira sequencial, ele precisa saber em que posição cada uma se encontra. Essa noção é introduzida por meio do **positional encoding**, um padrão matemático baseado em senos e cossenos de diferentes frequências. Cada posição da sequência é mapeada para um vetor único, que é somado ao embedding da palavra. Como indicado na Figura 1, tanto o encoder quanto o decoder recebem vetores de *embedding* somados ao *Positional Encoding*, o que fornece ao modelo uma noção da posição de cada token na sequência.

```python
import torch
import math

def positional_encoding(seq_len, d_model):
    pos = torch.arange(seq_len).unsqueeze(1)
    i = torch.arange(0, d_model, 2)
    pe = torch.zeros(seq_len, d_model)
    pe[:, 0::2] = torch.sin(pos / (10000 ** (i / d_model)))
    pe[:, 1::2] = torch.cos(pos / (10000 ** (i / d_model)))
    return pe
```

Essas ondas senoidais fornecem ao modelo uma "bússola posicional" contínua, permitindo que ele reconheça a ordem das palavras e até extrapole para comprimentos de sequência não vistos durante o treinamento.

Treinamento e Vantagens

O Transformer destacou-se pela **eficiência e paralelização**. Enquanto as RNNs processavam um token por vez, o Transformer podia processar toda a sequência simultaneamente. Essa característica reduziu o tempo de treinamento de dias para horas, com ganhos expressivos de desempenho. Segundo Vaswani et al., o modelo "base" superou os melhores sistemas de tradução da época, como o GNMT, consumindo apenas uma fração dos recursos computacionais .

Além da velocidade, a atenção é interpretável: é possível visualizar quais palavras influenciam outras, revelando estruturas sintáticas e semânticas emergentes — algo inédito nos modelos anteriores.

O Legado: BERT, GPT e Além

O artigo "Attention Is All You Need" inaugurou uma nova era ao apresentou um novo modelo. A arquitetura Transformer tornou-se a base de praticamente todos os avanços modernos em aprendizado de máquina. Modelos como **BERT** (voltado à compreensão de linguagem), **GPT** (para geração de texto), **T5** (para múltiplas tarefas) e **Vision Transformer (ViT)** (para imagens) derivam diretamente dessa proposta. Cada um adapta o mesmo princípio: aprender representações contextuais por meio de atenção em larga escala.

O impacto ultrapassou o campo do Processamento de Linguagem Natural. Hoje, Transformers são usados em visão computacional, bioinformática, música e até em modelos multimodais que integram texto, imagem e som.

Implementando um Mini-Transformer

Com as bibliotecas modernas, construir um Transformer tornou-se acessível. O PyTorch oferece implementações nativas que permitem experimentar rapidamente o funcionamento básico:

```python
import torch.nn as nn

model = nn.TransformerEncoder(
    nn.TransformerEncoderLayer(d_model=512, nhead=8),
    num_layers=6
)
```

Esse exemplo cria um encoder simples com seis camadas e oito cabeças de atenção, o suficiente para testar tarefas como classificação de texto ou tradução reduzida.

Dado todo este contexto, podemos concluir que o Transformer representou uma mudança de paradigma: ao substituir o processamento sequencial por um modelo de atenção paralela, tornou possível treinar redes neurais mais rápidas, profundas e interpretáveis. Essa arquitetura transformou o processamento de linguagem e redefiniu os rumos da inteligência artificial. A frase que dá nome ao artigo — *"Attention is all you need"* — acabou se tornando uma profecia tecnológica: em larga medida, a atenção realmente bastou para reinventar o aprendizado de máquina moderno.

Referência

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. **Attention Is All You Need**. arXiv, 2023. DOI: 10.48550/arXiv.1706.03762. Disponível em: http://arxiv.org/abs/1706.03762.