Inteligência Artificial 2

Aula 13: A Mecânica Transformer

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Inteligência Artificial 2

Sumário

Onde Paramos?

Na aula anterior, construímos o **pipeline de ingestão** (ETL) para alimentar bancos vetoriais:

- Extração de PDFs com PyPDF e Apache Tika.
- Chunking com overlap para preservar contexto semântico.
- Carga no ChromaDB com metadados rastreáveis.

A Pergunta de Hoje

Temos vetores e busca semântica. Mas **como o modelo** entende linguagem? Qual é a arquitetura **por trás** do GPT?

Objetivos da Sessão

Ao final desta aula, você será capaz de:

1. **Explicar** por que RNNs/LSTMs falharam em escalar para textos longos.
2. **Descrever** o mecanismo de Self-Attention e sua complexidade $O(n^2)$.
3. **Interpretar** a equação $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$.
4. **Diferenciar** Encoder, Decoder e modelos Encoder-Decoder.
5. **Conectar** a arquitetura Transformer com GPT, BERT e T5.

Pré-requisito

Multiplicação de matrizes (Álgebra Linear) e Softmax (Aula 3) — hoje elas formam o coração da IA moderna.

Como IAs "Liam" antes de 2017

- O estado da arte do Processamento de Linguagem Natural era dominado por arquiteturas conhecidas como **RNNs** (Redes Neurais Recorrentes) e **LSTMs** (*Long Short-Term Memory*).

Como IAs "Liam" antes de 2017

- O estado da arte do Processamento de Linguagem Natural era dominado por arquiteturas conhecidas como **RNNs** (Redes Neurais Recorrentes) e **LSTMs** (*Long Short-Term Memory*).
- Estas arquiteturas processavam a linguagem através de *Loops* fechados e temporais.

Como IAs "Liam" antes de 2017

- O estado da arte do Processamento de Linguagem Natural era dominado por arquiteturas conhecidas como **RNNs** (Redes Neurais Recorrentes) e **LSTMs** (*Long Short-Term Memory*).
- Estas arquiteturas processavam a linguagem através de *Loops* fechados e temporais.
- O modelo lia o texto **em estrita sequência temporal**, uma palavra de cada vez, da esquerda para a direita, simulando a leitura humana e acumulando um "estado mental" (Hidden State).

Os Dois Gargalos Fatais do Modelo Sequencial

1. Degradação de Memória

Como o estado interno é sobrescrito a cada nova palavra, o modelo sofria com a perda de gradiente (*Vanishing Gradient*). Num texto de 300 palavras, ao ler a última palavra, a rede simplesmente **esquecia** quem era o sujeito na primeira linha.

2. O Gargalo de Hardware

Hardware moderno (GPUs/TPUs) adora **paralelismo**. Porém, para processar a palavra t, a RNN exige o resultado da palavra $t-1$. É um processo estritamente bloqueante. Treinar uma RNN com bilhões de parâmetros levaria séculos.

O Marco Zero da IA Moderna

- No fim de 2017, uma equipe do **Google Brain** publicou o paper *Attention is All You Need*.

O Marco Zero da IA Moderna

- No fim de 2017, uma equipe do **Google Brain** publicou o paper *Attention is All You Need*.
- A proposta arquitetural chamava-se **Transformer**. O impacto deste paper na engenharia de software e na humanidade não tem precedentes modernos.

O Marco Zero da IA Moderna

- No fim de 2017, uma equipe do **Google Brain** publicou o paper *Attention is All You Need*.
- A proposta arquitetural chamava-se **Transformer**. O impacto deste paper na engenharia de software e na humanidade não tem precedentes modernos.
- **A tese disruptiva:** Abandone as Redes Recorrentes (RNNs) e Convolucionais. Substitua tudo por um único mecanismo matemático de atenção global.

Quebra de Paradigma: Paralelismo Máximo

- O Transformer **não lê a frase em sequência**.

Quebra de Paradigma: Paralelismo Máximo

- O Transformer **não lê a frase em sequência**.
- Ele joga **todas** as 500 palavras do parágrafo dentro da placa de vídeo ao mesmo exato tempo.

Quebra de Paradigma: Paralelismo Máximo

- O Transformer **não lê a frase em sequência**.
- Ele joga **todas** as 500 palavras do parágrafo dentro da placa de vídeo ao mesmo exato tempo.
- As multiplicações de tensores ocorrem em paralelo total (O(1) para sequência).

Quebra de Paradigma: Paralelismo Máximo

- O Transformer **não lê a frase em sequência**.
- Ele joga **todas** as 500 palavras do parágrafo dentro da placa de vídeo ao mesmo exato tempo.
- As multiplicações de tensores ocorrem em paralelo total (O(1) para sequência).
- **O Problema Resultante:** Se o modelo ingere todas as palavras de uma vez como um "saco de dados", como ele sabe que a palavra X veio antes da palavra Y? A estrutura de tempo sumiu!

O Resgate do Tempo: Positional Encoding

- Para resolver a ausência de ordem, o Transformer introduz o **Positional Encoding** (Codificação Posicional).

O Resgate do Tempo: Positional Encoding

- Para resolver a ausência de ordem, o Transformer introduz o **Positional Encoding** (Codificação Posicional).
- Antes de enviar o *Embedding* (o vetor da palavra) para a rede, somamos a ele um pequeno vetor de ondas harmônicas (senos e cossenos de diferentes frequências).

O Resgate do Tempo: Positional Encoding

- Para resolver a ausência de ordem, o Transformer introduz o **Positional Encoding** (Codificação Posicional).
- Antes de enviar o *Embedding* (o vetor da palavra) para a rede, somamos a ele um pequeno vetor de ondas harmônicas (senos e cossenos de diferentes frequências).
- Essa "tatuagem matemática" avisa ao modelo exatamente em que coordenada (posição na frase) aquele vetor se encontrava originalmente.

Encoder vs Decoder: As Ramificações

O *paper* original do Transformer possuía duas metades: O Encoder e o Decoder. Com o tempo, a indústria se fragmentou e especializou as duas:

Encoder vs Decoder: As Ramificações

O *paper* original do Transformer possuía duas metades: O Encoder e o Decoder. Com o tempo, a indústria se fragmentou e especializou as duas:

- **Encoder-Only (Ex: BERT):** Usam Auto-Atenção bidirecional absoluta. Excelentes para *ler* e classificar textos (análise de sentimentos, buscas vetoriais). Não geram texto novo.

Encoder vs Decoder: As Ramificações

O *paper* original do Transformer possuía duas metades: O Encoder e o Decoder. Com o tempo, a indústria se fragmentou e especializou as duas:

- **Encoder-Only (Ex: BERT):** Usam Auto-Atenção bidirecional absoluta. Excelentes para *ler* e classificar textos (análise de sentimentos, buscas vetoriais). Não geram texto novo.
- **Decoder-Only (Ex: GPT, LLaMA):** Usam Auto-Atenção mascarada (só olham para trás). Péssimos classificadores por natureza, mas reis absolutos na "Geração Autoregressiva" (prever a próxima palavra).

Encoder vs Decoder: As Ramificações

O *paper* original do Transformer possuía duas metades: O Encoder e o Decoder. Com o tempo, a indústria se fragmentou e especializou as duas:

- **Encoder-Only (Ex: BERT):** Usam Auto-Atenção bidirecional absoluta. Excelentes para *ler* e classificar textos (análise de sentimentos, buscas vetoriais). Não geram texto novo.
- **Decoder-Only (Ex: GPT, LLaMA):** Usam Auto-Atenção mascarada (só olham para trás). Péssimos classificadores por natureza, mas reis absolutos na "Geração Autoregressiva" (prever a próxima palavra).
- **Encoder-Decoder (Ex: T5, BART):** A fusão original. O Encoder lê um texto na língua A, e o Decoder cospe na língua B. Perfeitos para Tradução e Resumo de textos.

Leis de Escala (Scaling Laws)

- Pesquisadores notaram um fenômeno assustador nas arquiteturas Transformer: O modelo se torna mais "inteligente" matematicamente apenas adicionando mais camadas e alimentando-o com mais dados.

Leis de Escala (Scaling Laws)

- Pesquisadores notaram um fenômeno assustador nas arquiteturas Transformer: O modelo se torna mais "inteligente" matematicamente apenas adicionando mais camadas e alimentando-o com mais dados.
- Isso cunhou as **Leis de Escala**. Diferente das LSTMs que estagnavam rapidamente, o *Loss* do Transformer cai em uma curva logarítmica previsível conforme dobramos o processamento (compute).

Leis de Escala (Scaling Laws)

- Pesquisadores notaram um fenômeno assustador nas arquiteturas Transformer: O modelo se torna mais "inteligente" matematicamente apenas adicionando mais camadas e alimentando-o com mais dados.
- Isso cunhou as **Leis de Escala**. Diferente das LSTMs que estagnavam rapidamente, o *Loss* do Transformer cai em uma curva logarítmica previsível conforme dobramos o processamento (compute).
- É essa lei que motivou o Google, OpenAI e Anthropic a gastarem dezenas de bilhões de dólares construindo os maiores Supercomputadores da história em 2024.

O Mecanismo de Atenção

- O núcleo fundamental da arquitetura que substituiu a leitura sequencial foi o **Self-Attention** (Auto-Atenção).

O Mecanismo de Atenção

- O núcleo fundamental da arquitetura que substituiu a leitura sequencial foi o **Self-Attention** (Auto-Atenção).
- Pense em uma sala cheia de pessoas (as palavras da frase). O Self-Attention permite que **cada pessoa converse com todas as outras pessoas da sala simultaneamente** para descobrir com quem ela tem maior afinidade de sentido.

O Mecanismo de Atenção

- O núcleo fundamental da arquitetura que substituiu a leitura sequencial foi o **Self-Attention** (Auto-Atenção).
- Pense em uma sala cheia de pessoas (as palavras da frase). O Self-Attention permite que **cada pessoa converse com todas as outras pessoas da sala simultaneamente** para descobrir com quem ela tem maior afinidade de sentido.
- Ao fazer isso, o Transformer tece uma teia de correlações matemáticas densas, entendendo a frase através da **topologia de suas relações transversais**.

A Metáfora do Banco de Dados: Q, K e V

Para calcular essa afinidade, o *Self-Attention* projeta cada palavra em 3 vetores menores chamados **Q, K e V**.

A Metáfora do Banco de Dados: Q, K e V

Para calcular essa afinidade, o *Self-Attention* projeta cada palavra em 3 vetores menores chamados **Q, K e V**.

1. **Query (Q) - A Pergunta:** O que eu (esta palavra atual) estou procurando para dar sentido a mim mesma no contexto?

A Metáfora do Banco de Dados: Q, K e V

Para calcular essa afinidade, o *Self-Attention* projeta cada palavra em 3 vetores menores chamados **Q, K e V**.

1. **Query (Q) - A Pergunta:** O que eu (esta palavra atual) estou procurando para dar sentido a mim mesma no contexto?
2. **Key (K) - A Chave:** O que eu tenho a oferecer de informação estrutural para outras palavras que estejam procurando?

A Metáfora do Banco de Dados: Q, K e V

Para calcular essa afinidade, o *Self-Attention* projeta cada palavra em 3 vetores menores chamados **Q, K e V**.

1. **Query (Q) - A Pergunta:** O que eu (esta palavra atual) estou procurando para dar sentido a mim mesma no contexto?
2. **Key (K) - A Chave:** O que eu tenho a oferecer de informação estrutural para outras palavras que estejam procurando?
3. **Value (V) - O Valor:** Qual é o meu conteúdo semântico real e profundo (o meu significado base)?

Exemplo Prático: Resolvendo Ambiguidades

O Desafio da Coreferência

Frase: *"O* **animal** *não atravessou a* **rua** *porque* **ele** *estava cansado."*

Exemplo Prático: Resolvendo Ambiguidades

O Desafio da Coreferência

Frase: *"O* **animal** *não atravessou a* **rua** *porque* **ele** *estava cansado."*

- Quem estava cansado? A rua ou o animal? Humanos usam o contexto para deduzir que ruas não se cansam.

Exemplo Prático: Resolvendo Ambiguidades

O Desafio da Coreferência

Frase: *"O* **animal** *não atravessou a* **rua** *porque* **ele** *estava cansado."*

- Quem estava cansado? A rua ou o animal? Humanos usam o contexto para deduzir que ruas não se cansam.
- O Transformer pega o **Query (Q)** da palavra **"ele"**. Esse vetor contém propriedades de "busca por um sujeito que sofra ação".

Exemplo Prático: Resolvendo Ambiguidades

O Desafio da Coreferência

Frase: *"O* **animal** *não atravessou a* **rua** *porque* **ele** *estava cansado."*

- Quem estava cansado? A rua ou o animal? Humanos usam o contexto para deduzir que ruas não se cansam.
- O Transformer pega o **Query (Q)** da palavra **"ele"**. Esse vetor contém propriedades de "busca por um sujeito que sofra ação".
- Ele cruza esse Q contra o **Key (K)** da palavra "rua" e contra o **K** da palavra "animal".

A Matemática do Self-Attention

- O cálculo de afinidade entre o Q de uma palavra e o K de outra é um simples **Produto Escalar (Dot Product)** entre vetores, que quantifica a similaridade trigonométrica.

A Matemática do Self-Attention

- O cálculo de afinidade entre o Q de uma palavra e o K de outra é um simples **Produto Escalar (Dot Product)** entre vetores, que quantifica a similaridade trigonométrica.
- Equação Fundamental do Scaled Dot-Product Attention:

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

A Matemática do Self-Attention

- O cálculo de afinidade entre o Q de uma palavra e o K de outra é um simples **Produto Escalar (Dot Product)** entre vetores, que quantifica a similaridade trigonométrica.
- Equação Fundamental do Scaled Dot-Product Attention:

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

- Multiplicamos todas as Perguntas (Q) por todas as Chaves (K^T).
- Dividimos pela dimensão ($\sqrt{d_k}$) para estabilizar os gradientes da placa de vídeo.
- Aplicamos o Softmax para converter os escores de distância em uma distribuição de **probabilidade (0.0 a 1.0)**.

O Papel do Value (V)

- O Softmax gerou uma matriz de pesos (Ex: *"A palavra 'ele' presta* 95% *de atenção em 'animal',* 4% *em 'cansado' e* 1% *no resto"*).

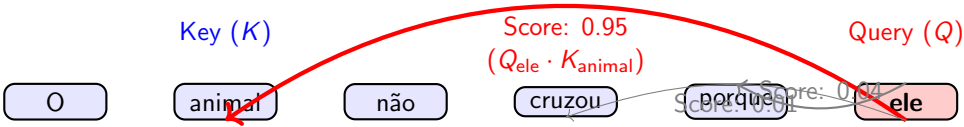
O Papel do Value (V)

- O Softmax gerou uma matriz de pesos (Ex: *"A palavra 'ele' presta* 95% *de atenção em 'animal',* 4% *em 'cansado' e* 1% *no resto"*).
- O último passo da equação é multiplicar esses percentuais pelos **Values (V)**.

O Papel do Value (V)

- O Softmax gerou uma matriz de pesos (Ex: *"A palavra 'ele' presta* 95% *de atenção em 'animal',* 4% *em 'cansado' e* 1% *no resto"*).
- O último passo da equação é multiplicar esses percentuais pelos **Values (V)**.
- Isso significa que a palavra original "ele" será sobrescrita e se tornará um novo vetor que absorve (injeta dentro de si mesmo) o *Embedding* da palavra "animal".
- Contexto não é mais um conceito abstrato: o **contexto agora está fundido na própria matriz da palavra**.

Representação Visual do Escalonamento



A Limitação de um Único Foco

- Se houvesse apenas um cálculo de Atenção (uma única tríade Q, K, V), a palavra prestaria atenção em apenas uma faceta da frase (ex: descobrir a gramática e as conexões de gênero).

A Limitação de um Único Foco

- Se houvesse apenas um cálculo de Atenção (uma única tríade Q, K, V), a palavra prestaria atenção em apenas uma faceta da frase (ex: descobrir a gramática e as conexões de gênero).
- Contudo, a linguagem humana carrega **dimensões simultâneas**: Quem faz a ação? Onde ocorre? Qual o tom emocional? Existe sarcasmo?

A Limitação de um Único Foco

- Se houvesse apenas um cálculo de Atenção (uma única tríade Q, K, V), a palavra prestaria atenção em apenas uma faceta da frase (ex: descobrir a gramática e as conexões de gênero).
- Contudo, a linguagem humana carrega **dimensões simultâneas**: Quem faz a ação? Onde ocorre? Qual o tom emocional? Existe sarcasmo?
- Uma única matriz de peso de atenção seria forçada a englobar toda a complexidade da semântica humana e fracassaria.

A Solução: Multi-Head Attention

- Para entender contextos espessos, a arquitetura clona e bifurca o mecanismo em **Múltiplas Cabeças Independentes** (*Multi-Head Attention*).

A Solução: Multi-Head Attention

- Para entender contextos espessos, a arquitetura clona e bifurca o mecanismo em **Múltiplas Cabeças Independentes** (*Multi-Head Attention*).
- Um modelo moderno (como o GPT-3) utiliza quase 100 cabeças processando simultaneamente a mesma frase, cada uma sorteando pesos iniciais diferentes.

A Solução: Multi-Head Attention

- Para entender contextos espessos, a arquitetura clona e bifurca o mecanismo em **Múltiplas Cabeças Independentes** (*Multi-Head Attention*).
- Um modelo moderno (como o GPT-3) utiliza quase 100 cabeças processando simultaneamente a mesma frase, cada uma sorteando pesos iniciais diferentes.
- Cada cabeça se "especializa" naturalmente no treinamento:
  - A Cabeça 1 observa concordância verbal.
  - A Cabeça 4 observa adjetivos conectados a substantivos.
  - A Cabeça 8 caça conexões causais (Se... então).

A Concatenação Final

- Após as múltiplas cabeças processarem suas visões individuais do texto, o Transformer pega os vetores resultantes de todas elas e os **concatena** (junta-os lateralmente).

A Concatenação Final

- Após as múltiplas cabeças processarem suas visões individuais do texto, o Transformer pega os vetores resultantes de todas elas e os **concatena** (junta-os lateralmente).
- Esse vetor gigante passa por uma camada de Compressão Linear final (pesos densos normais) e por uma rede *Feed Forward* (FFN).

A Concatenação Final

- Após as múltiplas cabeças processarem suas visões individuais do texto, o Transformer pega os vetores resultantes de todas elas e os **concatena** (junta-os lateralmente).
- Esse vetor gigante passa por uma camada de Compressão Linear final (pesos densos normais) e por uma rede *Feed Forward* (FFN).
- O resultado é uma matriz altamente comprimida que contém o mapeamento absoluto da realidade linguística daquela sentença. O computador não só lê letras, ele compreende as abstrações semânticas daquele universo.

O Gargalo do Self-Attention: $O(N^2)$

- Todo esse poder tem um custo trágico de Engenharia de Software.

O Gargalo do Self-Attention: $O(N^2)$

- Todo esse poder tem um custo trágico de Engenharia de Software.
- Lembra que "Todas as palavras conversam com todas"? Na Matemática, o cruzamento de todo elemento com todo elemento cresce quadraticamente $\mathbf{O(N^2)}$.

O Gargalo do Self-Attention: $O(N^2)$

- Todo esse poder tem um custo trágico de Engenharia de Software.
- Lembra que "Todas as palavras conversam com todas"? Na Matemática, o cruzamento de todo elemento com todo elemento cresce quadraticamente **$O(N^2)$**.
- Se sua janela de contexto dobrar (ex: de 4000 para 8000 tokens), o consumo de VRAM e processamento da placa de vídeo não dobra... ele **quadruplica**.
- É por isso que Modelos com 1 Milhão de tokens de contexto (como o Gemini 1.5) exigiram avanços radicais na teoria (como o *Ring Attention*) para não derreter os servidores.

KV Cache: O Segredo da Geração de Texto

- Quando um LLM (como o ChatGPT) responde a você, ele prevê a palavra $t + 1$.

KV Cache: O Segredo da Geração de Texto

- Quando um LLM (como o ChatGPT) responde a você, ele prevê a palavra $t + 1$.
- Para prever a $t + 2$, ele injeta a palavra gerada novamente na entrada e roda TUDO de novo. Recalcular a matriz Q e K do texto inteiro a cada sílaba travaria qualquer PC moderno.

KV Cache: O Segredo da Geração de Texto

- Quando um LLM (como o ChatGPT) responde a você, ele prevê a palavra $t + 1$.
- Para prever a $t + 2$, ele injeta a palavra gerada novamente na entrada e roda TUDO de novo. Recalcular a matriz Q e K do texto inteiro a cada sílaba travaria qualquer PC moderno.
- O segredo da inferência rápida se chama **KV Cache**. O Transformer armazena na memória RAM da placa de vídeo as chaves (K) e valores (V) do texto anterior. Assim, na sílaba seguinte, a rede neural só propaga o cálculo **daquela nova sílaba gerada**.

O Papel da Normalização e Conexões Residuais

- Em redes de centenas de camadas as "Tensões" dos números (gradientes) perdem força ou explodem ao infinito (*Exploding Gradients*).

O Papel da Normalização e Conexões Residuais

- Em redes de centenas de camadas as "Tensões" dos números (gradientes) perdem força ou explodem ao infinito (*Exploding Gradients*).
- O bloco de um Transformer incorpora engenharia clássica: **Residual Connections** (pular blocos enviando o sinal original diretamente para frente) e **Layer Normalization** (re-centrar os números em uma distribuição equilibrada).

O Papel da Normalização e Conexões Residuais

- Em redes de centenas de camadas as "Tensões" dos números (gradientes) perdem força ou explodem ao infinito (*Exploding Gradients*).
- O bloco de um Transformer incorpora engenharia clássica: **Residual Connections** (pular blocos enviando o sinal original diretamente para frente) e **Layer Normalization** (re-centrar os números em uma distribuição equilibrada).
- São esses pequenos detalhes da planta-baixa que garantem que redes gigantes de 70 Bilhões de parâmetros não matematicamente colapsem nos primeiros 10 minutos de treinamento.

Tokenização: A Queda da Palavra Inteira

- Antes da rede ver vetores, ela precisa converter o texto cru em IDs. Antigamente, dividíamos por palavras (ex: `split(" ")`).
- O problema: O dicionário ficava infinito. Como lidar com conjugações, erros de digitação e palavras compostas? O vocabulário estouraria a RAM e o modelo travaria.
- A solução moderna (usada no GPT-4 e LLaMA) é a Tokenização por Subpalavras (*Subword Tokenization*).

Byte-Pair Encoding (BPE)

- O algoritmo BPE varre o corpus de treinamento do modelo e começa fundindo caracteres sozinhos nos pares que mais ocorrem juntos.
- Palavras ultra comuns viram um único token ("the", "que", "IA").
- Palavras raras e neologismos são explodidos em sub-partes lógicas ("Otorrinolaringologista" $\rightarrow$ ["Otor", "rinolaringo", "logista"]).
- Assim, o modelo consegue ler e criar **qualquer** palavra do mundo (incluindo emojis) sem estourar seu limite de vocabulário fixo de $\sim$ 50.000 tokens.

O Paradoxo Posicional: De Vaswani ao RoPE

- No paper original, o Positional Encoding era **absoluto** e injetado via funções senoidais fixas (senos e cossenos).
- Hoje, os LLMs (LLaMA-3, Mistral) utilizam o **RoPE** (Rotary Position Embedding).
- O RoPE não apenas marca a posição absoluta, mas gira matematicamente os vetores no espaço, garantindo que a distância **relativa** entre duas palavras seja mantida não importa onde elas ocorram na frase. Isso foi vital para saltarmos janelas de contexto de 2.000 para 1.000.000 de tokens.

A Camada Esquecida: Feed-Forward (FFN)

- Nós focamos tanto na Atenção que esquecemos o resto do bloco Transformer.
- Após as cabeças de Atenção (Multi-Head Attention) cruzarem o contexto de toda a frase, a matriz resultante passa por uma rede densa gigantesca (*Feed-Forward Network*).
- **A Diferença:** A Atenção propaga a *Sintaxe e o Contexto*. O Feed-Forward é onde o modelo armazena os seus **Fatos e Conhecimento do Mundo** (Ex: "A capital da França é Paris").

Layer Normalization (Norm) Prática

- Ao multiplicar matrizes por 100 camadas seguidas, os valores dos tensores explodiriam (um peso 1.5 viria a ser alguns trilhões, e a GPU dispararia o erro `NaN`).
- A camada *LayerNorm* força os números a voltarem para o eixo (média 0, desvio padrão 1) ao final de cada bloco de Atenção.
- É o amortecedor que estabiliza o modelo colossal e impede a explosão nuclear matemática do treinamento de um ChatGPT.

Recapitulando - A Anatomia Completa

1. **Tokenizador (BPE):** Transforma texto longo em IDs de subpalavras otimizadas.
2. **Embeddings e RoPE:** Adiciona profundidade semântica (vetores) e noção posicional rotatória.
3. **Multi-Head Attention:** Usa Query, Key, e Value para cruzar a atenção entre os tokens.
4. **FFN (Feed Forward):** Aplica as memórias estáticas ("conhecimento do mundo").
5. **Norm e Residuais:** A engenharia que segura a matemática sem explodir o computador.

Roteiro do Laboratório 13

Missão: Dissecando as fiações internas de um Transformer visualmente usando `bertviz` no Colab.

Objetivo Central

Provar que a correferência de um pronome ("ele" ou "ela") acende cabos cerebrais diferentes de atenção dependendo unicamente das palavras neutras vizinhas da frase.

Lab Passo 1 e 2: Preparando a Entrada

- Usaremos a base de todo LLM moderno: a biblioteca `transformers` da HuggingFace.
- Baixaremos o modelo BERT e o seu respectivo Tokenizador para memória.

```
from transformers import AutoTokenizer, AutoModel
model_name = "bert-base-multilingual-cased"

tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name, output_attentions=True
    )
```

Lab Passo 3: Extraindo a Matriz de Atenção

- O detalhe principal é passar o argumento `output_attentions=True`. Isso obriga o modelo a não cuspir apenas o resultado, mas exportar os tensores intermediários de Q e K multiplicados de todas as cabeças e camadas!

```
texto = "O cachorro nao atravessou a rua porque ele estava cansado."
inputs = tokenizer(texto, return_tensors="pt")

# A magica acontece aqui:
outputs = model(**inputs)
attention = outputs.attentions  # Um tensor gigante tridimensional!
```

Lab Passo 4: O BertViz (Head View)

- Injetamos o tensor `attention` na biblioteca `bertviz`.
- O Colab abrirá um D3.js interativo (HTML) mostrando as linhas conectando as palavras.

```
from bertviz import head_view

# Renderiza as linhas de atencao de todos os tokens
tokens = tokenizer.convert_ids_to_tokens(inputs["input_ids"][0])
head_view(attention, tokens)
```

Lab Passo 5: Investigação Ativa

- Na última etapa, vocês mudarão a frase para **"A girafa não cruzou a rua porque ela estava cansada"**.
- Rastreiem a palavra **"ela"**. Notem como a linha grossa da atenção aponta para "girafa" na camada 8, cabeça 3.
- Isso prova que o "Contexto" dentro de um LLM não é magia negra: é simplesmente um peso matemático injetado dinamicamente durante a leitura!

Dúvidas?

Email: alessio@cefetmg.br
Aula: Segunda e Quarta-Feira, 7h as 8:40h (Sala 39)
WhatsApp: Dúvidas rápidas

Lembrete

Confira sua Issue no GitLab!

Próxima aula: Hugging Face Pipelines

100% da Aula 13 Concluída!