Introdução a Large Language Models

Professor Thiago de Paulo Faleiros

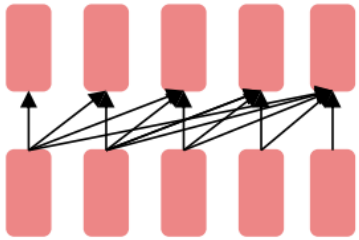
Large Language Models

- Lembrando do simples modelo n-grama
  - Atribua probabilidades para a sequência de palavras
  - Gerar textos por amostragem de possíveis próximas palavras
  - É treinada na contagem de vários textos
- LLM's são similares e diferente:
  - Atribui probabilidade para sequência de palavras
  - Gera a próxima palavra por amostragem
  - São treinadas para adivinhar a próxima palavra

Large Language Models

- Pré-treinado apenas para prever palavras
- Aprende muito conhecimento útil para a linguagem
- Desde que treinado em MUITO texto

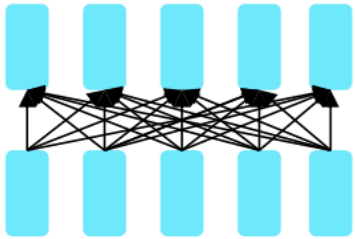
Três arquiteturas para LLM's



Decoders

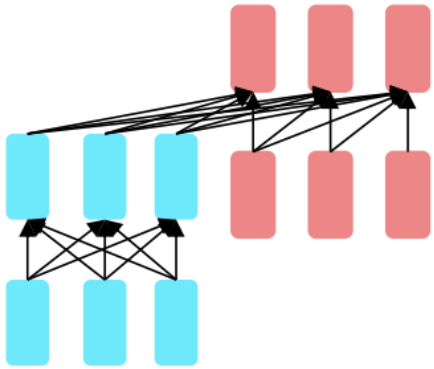
GPT, Claude, Llama

Mixtral



Encoders

BERT family, HuBERT



Encoder-decoders

Flan-T5, Whisper

Encoders

Muita variedade de arquiteturas e modelos!

- Popular: Masked Language Models (MLMs)
- BERT

- Treinado para predizer palavras via palavras vizinhas em ambos os lados
- São normalmente **finetuned** (treinado em dados supervisionados) para classificação.

Encoder-Decoders

- Treinado para mapear uma sequência em outra
  - Muito popular para:
    - traduação de máquina
    - reconhecimento de fala (mapeamento do acústico para palavras)

Grande ideia!

Várias tarefas podem ser transformadas na tarefa de predição de palavras

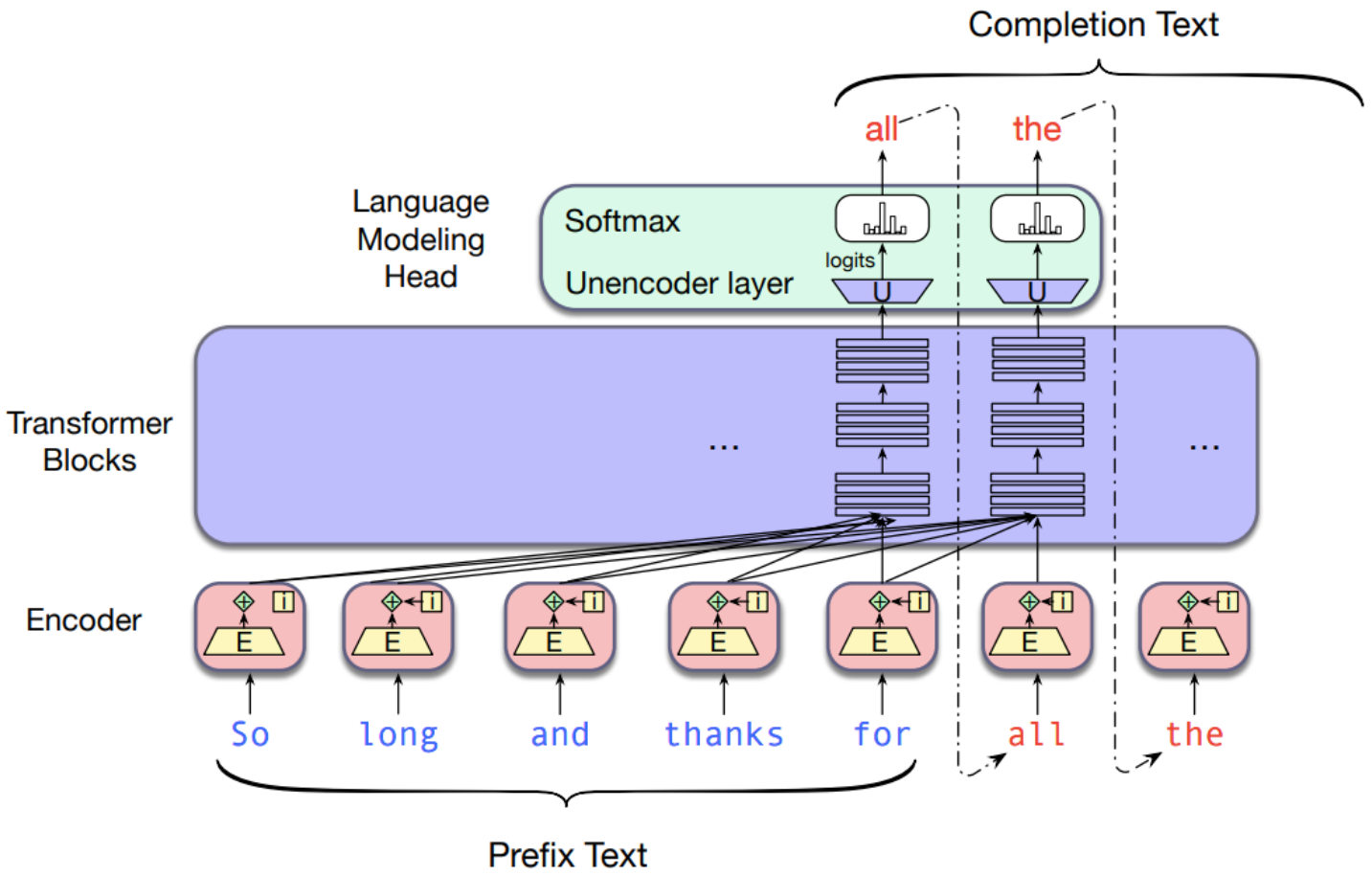
Essa aula: decoder-only

também chamado de:

- Causal LLMs
- Autoregressive LLMs
- Left-to-right LLMs

Prediz a palavra da esquerda para a direita

Geração condicional: Gerando texto condicionado na palavra anterior



Várias tarefas de NLP

Análise de sentimentos: “I like Jackie Chan”

1. Damos ao modelo de linguagem a seguinte string:

   The sentiment of the sentence “I like Jackie Chan” is:

2. E veremos qual palavra vem em seguida:

$$P(\textit{positive}|\text{The sentiment of the sentence “I like Jackie Chan” is:})$$

$$P(\textit{negative}|\text{The sentiment of the sentence “I like Jackie Chan” is:})$$

Enquadrando muitas tarefas como geração condicional

QA: “Who wrote The Origen of Species”

- Damos ao modelo de linguagem a seguinte string:

Q: Who wrote the book “The Origin of Species”? A:

- E veja qual palavra vem em seguida:

P(w | Q: Who wrote the book “The Origin of Species”? A:)

- E iterar:

P(w | Q: Who wrote the book “The Origin of Species”? A: Charles)

Sumarização

Original

The only thing crazier than a guy in snowbound Massachusetts boxing up the powdery white stuff and offering it for sale online? People are actually buying it. For $89, self-styled entrepreneur Kyle Waring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says.

But not if you live in New England or surrounding states. "We will not ship snow to any states in the northeast!" says Waring's website, ShipSnowYo.com. "We're in the business of expunging snow!"

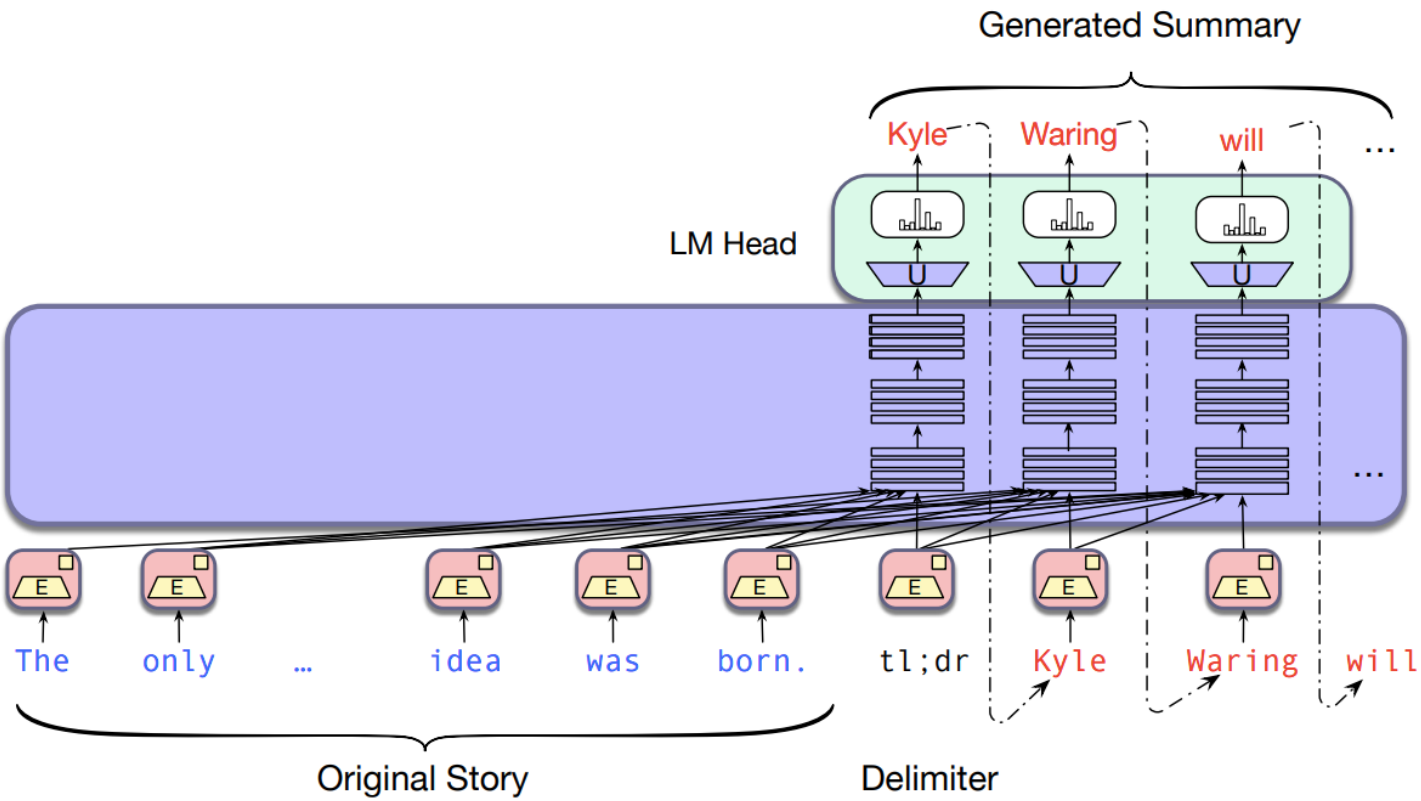
His website and social media accounts claim to have filled more than 133 orders for snow – more than 30 on Tuesday alone, his busiest day yet. With more than 45 total inches, Boston has set a record this winter for the snowiest month in its history. Most residents see the huge piles of snow choking their yards and sidewalks as a nuisance, but Waring saw an opportunity.

According to Boston.com, it all started a few weeks ago, when Waring and his wife were shoveling deep snow from their yard in Manchester-by-the-Sea, a coastal suburb north of Boston. He joked about shipping the stuff to friends and family in warmer states, and an idea was born. [...]

Summary

Kyle Waring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says. But not if you live in New England or surrounding states.

LLM's para sumarização (usando tl;dr)



Decoding and Sampling

Essa tarefa de escolher uma palavra para geração baseada na probabilidade do modelo é chamada decoding.
O método mais comum para decoding em LLM's: sampling.
Amostragem da distribuição das palavras:

- escolha uma palavra aleatória de acordo a probabilidade atribuída pelo modelo

Depois de cada token iremos amostrar as palavras para gerar de acordo a probabilidade condicionada as escolhas anteriores,

- Um modelo de linguagem baseado em transformer irá dar a probabilidade

Amostragem aleatória

$$
\begin{aligned}
&\mathrm{i} \leftarrow 1 \\
&w_i \sim \mathrm{p}(w) \\
&\textbf{while } w_i \text{ != EOS} \\
&\quad \mathrm{i} \leftarrow \mathrm{i} + 1 \\
&\quad w_i \sim \mathrm{p}(w_i \mid w_{<i})
\end{aligned}
$$

Amostragem aleatória não funciona muito bem

Embora a amostragem aleatória gere principalmente palavras sensatas e de alta probabilidade,
Há muitas palavras estranhas e de baixa probabilidade na cauda da distribuição
Cada uma é de baixa probabilidade, mas somadas constituem uma grande parte da distribuição
Então elas são escolhidas o suficiente para gerar frases estranhas

Fatores na amostragem de palavras: qualidade e diversidade

Enfatiza palavras de alta probabilidade

- + qualidade: mais acurado, coerente, e factual,
- - diversidade: repetitivo, entendiante

Enfatizando palavras de probabilidade mediana

- + diversidade: mais criatividade, diverso
- menos factual, incoerente

Top-k Sampling:

1. Escolha o número de k palavras
2. Para cada palavra no vocabulário V, use o modelo de linguagem para computar a probabilidade dessa palavra dado o contexto $p(w_t|w_{<t})$
3. Ordene as palavras por probabilidade, mantenha apenas as top k palavras mais prováveis
4. Renormalize os escores das k palavras para legitimizar a distribuição de probabilidade
5. Amostre aleatoriamente uma palavra dessas restantes k palavras mais prováveis e acordo com sua probabilidade.

Top-p Sampling:

Problema com top-k: k é fixo, então pode cobrir quantidades muito diferentes de massa de probabilidade em diferentes situações
Ideia: Em vez disso, mantenha o p percentual superior da massa de probabilidade
Dada uma distribuição $P(w_t|w_{<t})$, o vocabulário top-p $V(p)$ é o menor conjunto de palavras tal que

$$\sum_{w \in V(p)} P(w|w_{<t}) \geq p$$

Amostragem por temperatura

Remodele a distribuição em vez de truncá-la
Intuição da termodinâmica,

- um sistema em alta temperatura é flexível e pode explorar muitos estados possíveis,
- um sistema em temperatura mais baixa provavelmente explorará um subconjunto de estados de menor energia (melhores).

Na amostragem de baixa temperatura, ($\tau \leq 1$)) nós suavemente

- aumentamos a probabilidade das palavras mais prováveis
- diminuímos a probabilidade das palavras raras.

Amostragem por temperatura

Divide o logit pelo parâmetro de temperatura τ antes de passar pelo softmax

- Em vez de

$$y = softmax(u)$$

- Fazemos

$$y = softmax\left(\frac{u}{\tau}\right)$$

Amostragem por temperatura

$$y = softmax\left(\frac{u}{\tau}\right)$$

Por que isso funciona?

- Quando τ é próximo de 1 a distribuição não muda muito
- O valor baixo de τ é o maior escore sendo passado pelo softmax
- Softmax pressiona valores altos para 1 and valores baixos para 0.
- Entradas grandes aumenta o valor de palavras com alta probabilidade e diminui para baixas probabilidades, fazendo a distribuição mais “gulosa”.
- A medida que τ se aproxima de 0, os mais prováveis se aproximam de 1

Pré-treinamento

A grande ideia que fundamenta todo o incrível desempenho dos modelos de linguagem

Primeiramente pré-treine um modelo Transformer em uma quantidade enorme de texto

Então aplique isso para a próxima tarefa.

Algoritmo de treinamento “Self-supervised”

Apenas treine ele para predizer a próxima palavra!

1. Pegue um corpus de texto
2. A cada passo t:
    1. pergunte para o modelo predizer a próxima palavra
    2. treine o modelo usando gradiente descendente para minimizar o erro na predição

“Self-supervised” se refere a utilização da próxima palavra como rótulo

Intuição do treinamento de modelo de linguagem: loss

- A mesma função de perda: **cross-entropy loss**
  - Queremos que o modelo atribua uma alta probabilidade para palavra corretas w
  - = queremos que a perca seja alta se o modelo atribua baixa probabilidade para w
- CE Loss: A probabilidade logarítmica negativa que o modelo atribui para a correta próxima palavra
  - Se o modelo atribui uma probabilidade muito baixa para w
  - Nós movemos os pesos do modelo na direção para aumentar a probabilidade de w

Cross-entropy loss para modelo de linguagem

CE Loss: diferença entre a correta distribuição de probabilidade e a predita

$$L_{CE} = -\sum_{w \in V} y_t[w] \log \hat{y}_t[w]$$

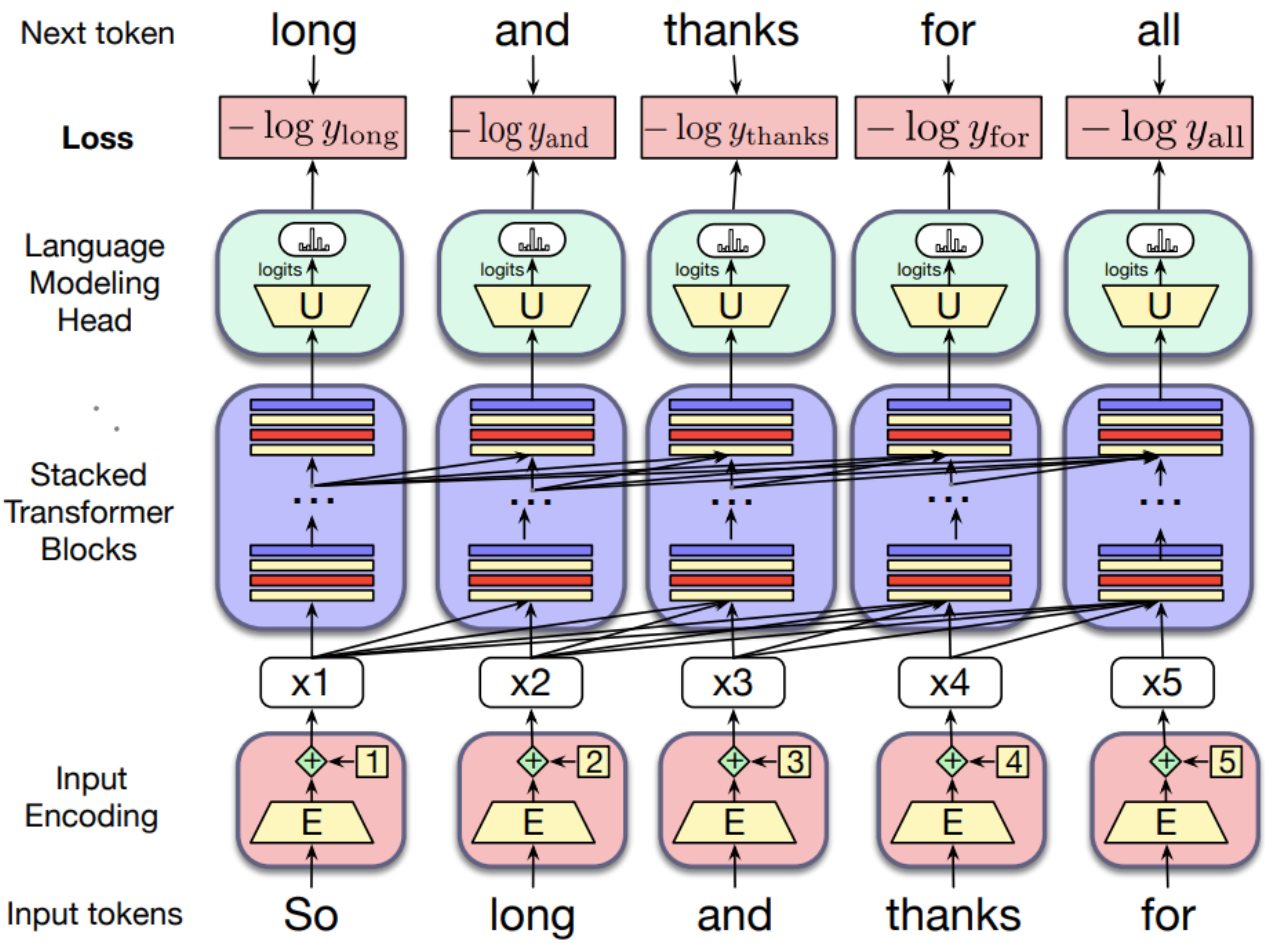
A distribuição correta y_t conhece a próxima palavra, então é 1 para a palavra correta e 0 para as outras
Assim, na soma, todos os termos são multiplicados por zero exceto um: o logarítmo da probabilidade do modelo atribuído para a correta próxima palavra.

$$L_{CE}(\hat{y}_t, y_t) = -\log \hat{y}_t[w_{t+1}]$$

Teacher forcing

- Para cada token t, o modelo vê os tokens corretos $w_{1:t}$
  - Computa o loss (-log probability) para o próximo token w_{t+1}
- Na próxima posição $t+1$ nós ignoramos o que o modelo preveu para w_{t+1}
  - Em vez disso, nós pegamos a palavra correta w_{t+1}, e adiciona ao contexto

Treinando o Transformer

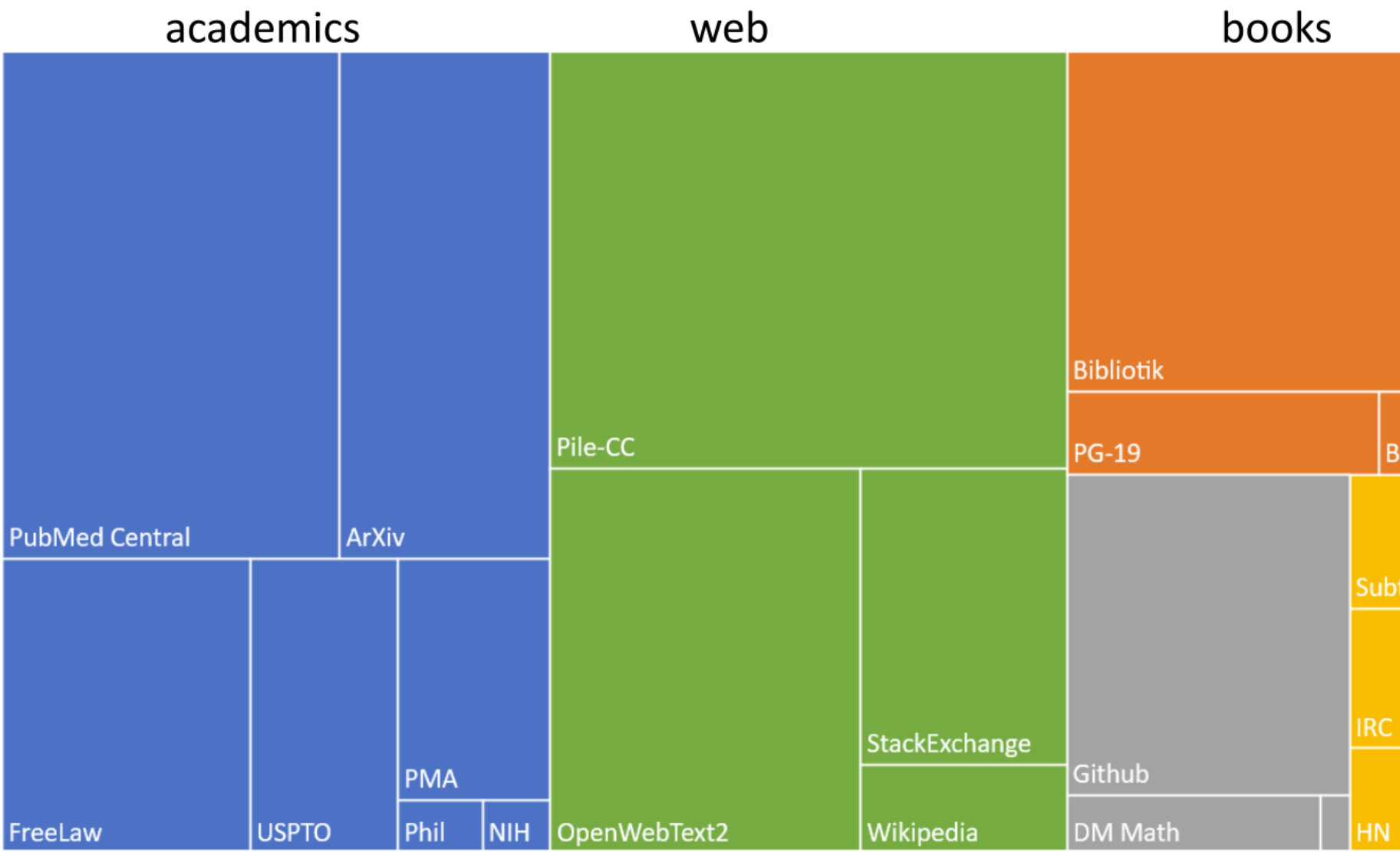


$$= \frac{1}{T}\sum_{t=1}^{T} L_{CE}$$

LLMs são treinados principalmente com dados da web

Rastreamento comum, instantâneos de toda a web produzidos pela organização sem fins lucrativos Common Crawl com bilhões de páginas
Colossal Clean Crawled Corpus (C4; Raffel et al. 2020), 156 bilhões de tokens de inglês, filtrados
O que há nele? Principalmente documentos de texto de patentes, Wikipedia e sites de notícias

A Pilha: um corpus de pré-treinamento



Filtros para qualidade e segurança

Qualidade é subjetiva

- Vários LLM's tentam extrair da Wikipedia, livros, websites particulares
- Precisa remover textos padrões e conteúdo adulto
- Desduplicação em muitos níveis (URLs, documentos, linhas)

Segurança é também subjetivo

- Detecção de conteúdos tóxicos é importante, apesar da dificuldade
- Pode errar no bloqueio de dialetos

O que o modelo aprende do prétreinamento

- Existem caninos em todos os lugares! Um cachorro na sala, e dois cachorro
- Não foi apenas grande, foi enorme
- O autor de "A Room of One's Own" é Virginia Woolf
- O doutor me disse que ele
- A raiz quadrada de 4 é 2

O texto contém enormes quantidades de conhecimento

O pré-treinamento em muito texto com todo esse conhecimento é o que dá aos modelos de linguagem sua capacidade de fazer tanto

Existem problemas com extração de dados na web

Copyright: muito do texto nesses conjuntos de dados é protegido por direitos autorais

- Não está claro se a doutrina de uso nos EUA permite esse uso
- Isso é uma questão legal ainda em aberto

Consentimento de dados

- Donos de websites pode indicar que não querem seus sites extraídos

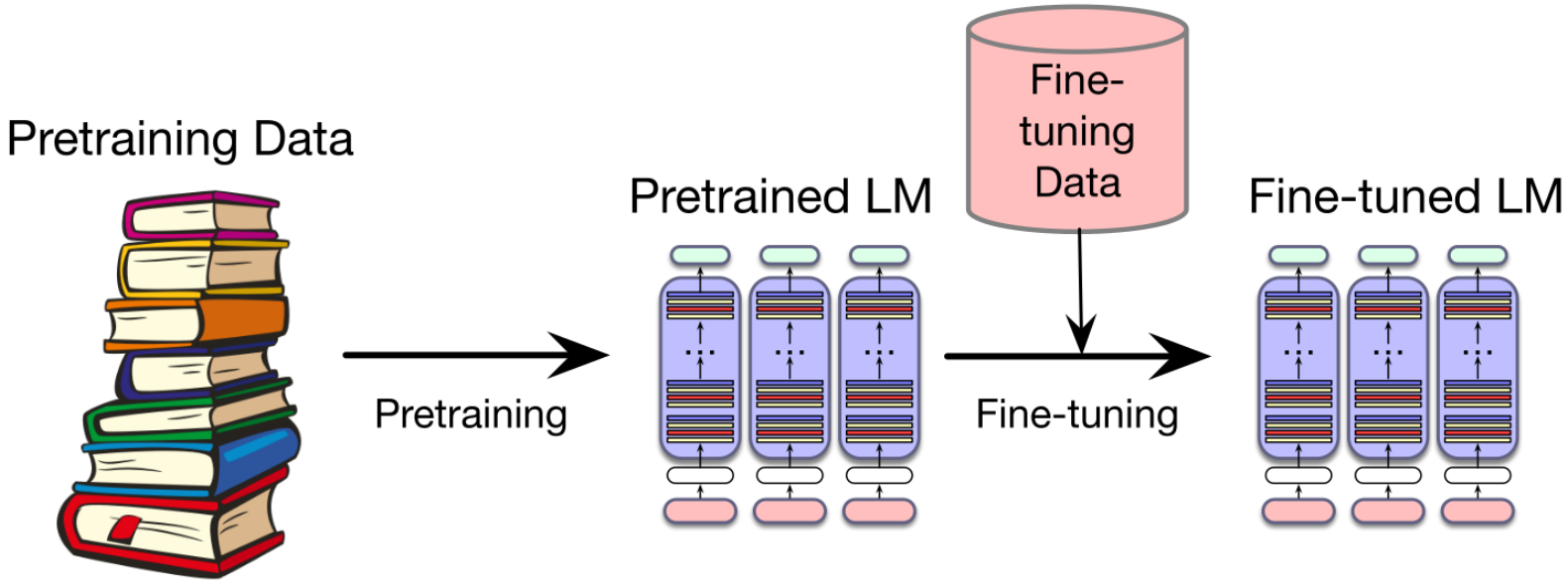
Privacidade

- Sites podem ter IP e telefones

Finetuning para adaptação para novos domínios

O que acontece se precisarmos que o LLM trabalhe bem em um domínio que não é conhecido no pré-treino?
Talvez para especificações médicas ou legais?
Ou talvez um modelo de linguagem multilingual precise ver mais dados em uma linguagem que era rara no pré-treinamento?

Finetuning


Pretraining Data
Pretraining
Pretrained LM
Fine-tuning Data
Fine-tuning
Fine-tuned LM

“Finetuning” significa 4 coisas diferentes

Será discutida apenas 1 aqui, e 3 nos próximos capítulos
Em todos os casos, finetunign significa:
Pegar um modelo pré-treinado e adaptar algum ou todos seus parâmetros para novos dados

Finetuning como “continuação do pré-treinamento”

- Treine ainda mais todos os parâmetros do modelo em novos dados
  - usando o mesmo método (previsão de palavras) e função de perda (perda de entropia cruzada) do pré-treinamento.
  - como se os novos dados estivessem no final dos dados de pré-treinamento
- por isso, às vezes, chamado de pré-treinamento contínuo

Avaliando LLM – Perplexidade

- Assim como no modelo n-gramas, usaremos a perplexidade para medir a qualidade do modelo de língua em prever palavras não vistas.
- A Perplexidade de um modelo θ em um conjunto de teste é o inverso da probabilidade que θ atribui para o conjunto de teste, normalizado pelo tamanho do conjunto de teste
- Para um conjunto de teste com n tokens $w_{1:n}$ a perplexidade é:

$$\begin{aligned} Perplexity_\theta(w_{1:n}) &= P_\theta(w_{1:n})^{-\frac{1}{n}} \\ &= \sqrt[n]{\frac{1}{P_\theta(w_{1:n})}} \end{aligned}$$

Por que Perplexidade?

- Por que não usar apenas a probabilidade do modelo no conjunto de teste?
- Probabilidade depende do tamanho do conjunto de teste
  - Probabilidade é menor para longas sequências de textos
  - Melhor: uma métrica que é por palavra, normalizada pelo tamanho
- Perplexidade é o inverso da probabilidade do conjunto de teste, normalizado pelo número de palavras
  (O inverso vem da definição original de perplexidade da taxa de entropia cruzada na teoria da informação)

  A probabilidade é da dimensão $[0, 1]$, perplexidade é da dimensão $[1, \infty]$.

Perplexidade

- Quanto maior a probabilidade da palavra em uma sequência, menor é a perplexidade
- Assim, quanto menor é perplexidade de um modelo nos dados, melhor o modelo
- Minimizando a perplexidade é o mesmo que maximizar a probabilidade

Perplexidade é sensível ao tamanho e ao tokenizador, então essa métrica será melhor utilizada para comparar LLM's que usam o mesmo tokenizador.

Vários outros fatores que avaliamos, como:

- Tamanho: Grandes modelos consomem muita GPU em tempo de treinamento e memória de armazenamento
- Usod e energia: Podemos mensurar KWh ou kilogramas de CO2 emitido
- Fairness: Benchmarks mendem esteriótipos de gênero e raça, ou diminui o desempenho para linguagens de ou sobre alguns grupos

Regras de escala

O desempenho do LLM depende de:

- Tamanho do modelo: número de parâmetros não contando embeddings
- O tamanho do conjunto de dados: quantidade de dados do treinamento
- Computação: Quantidade de computação
- Pode-se melhorar um modelo adicionando parâmetros (mais camadas, amplo contexto), mais dados, ou mais iterações de treinamento
- O desempenho de um grande modelo de linguagem (a perda) é dimensionado como uma lei de potência com cada um desses três

Leis de Escalas

Loss L com uma função no número de parâmetros N, tamanho do dataset D, orçamento C (se outros dois são constantes)

$$L(N) = \left(\frac{N_C}{N}\right)^{\alpha_N}$$
$$L(D) = \left(\frac{D_C}{D}\right)^{\alpha_D}$$
$$L(C) = \left(\frac{C_C}{C}\right)^{\alpha_C}$$

As leis de escala podem ser usadas inicialmente no treinamento para predizar qual a LOSS poderá ser se adicionármos mais dados ou aumentar o modelo.

Número de parâmetros N

Parâmetros não-embeddings

$$
\begin{aligned}
N \approx & 2 d n_{layer}(2 d_{attn} + d_{ff}) \\
\approx & 12 n_{layer} d^2 \\
& \text{(assuming } d_{attn} = \frac{d_{ff}}{4} = d\text{)}
\end{aligned}
$$

O GPT-3, com N=96 camadas e dimensionalidade D =12288, tem $12 \times 96 \times 122882 \approx 175$ bilhões de parâmetros.

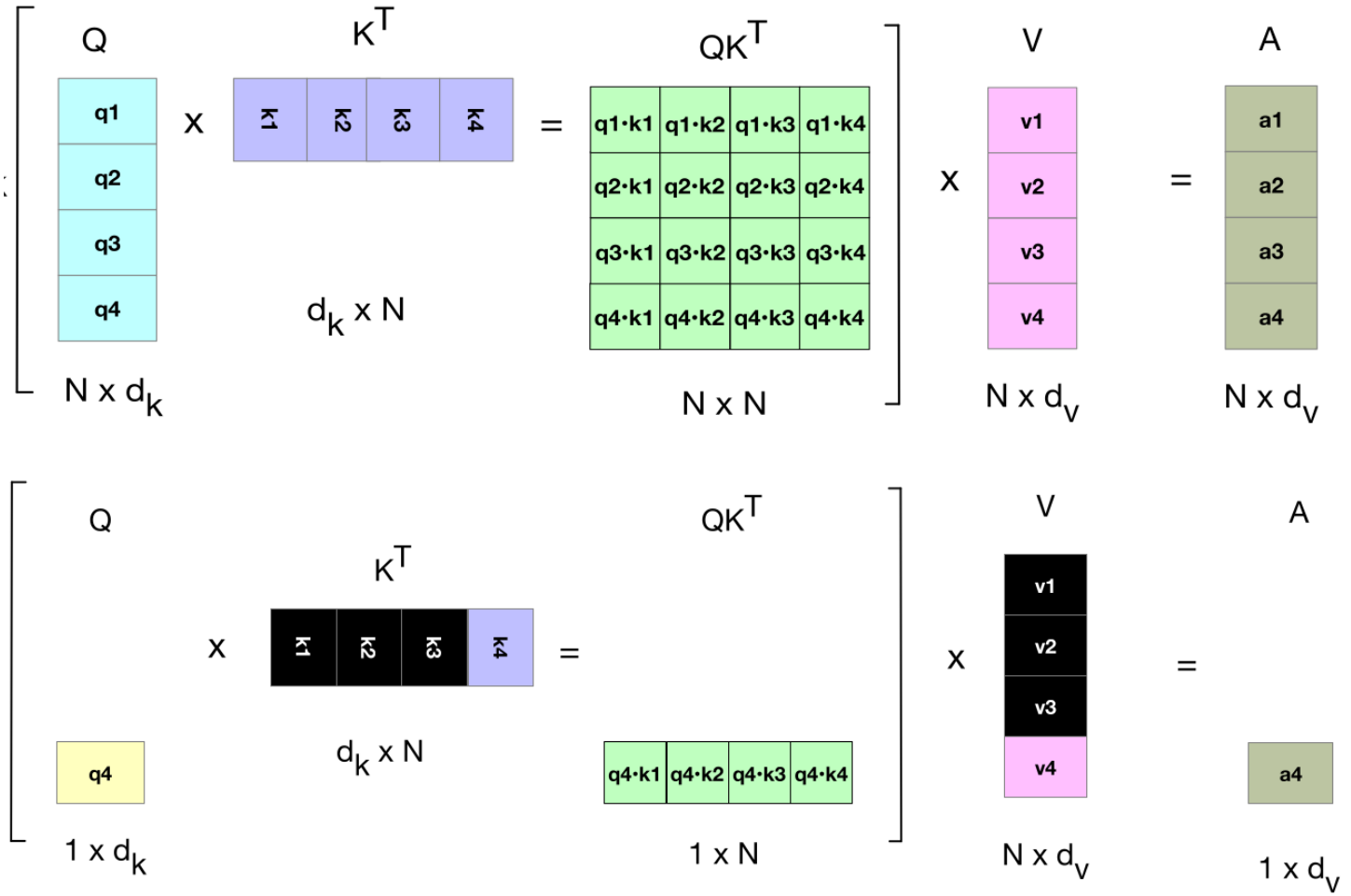
KV Cache

No treinamento, podemos computar a Atenção eficientemente em paralelo:

$$A = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

Mas não na inferência! O próximo token é gerado um por vez sequencialmente.
Para um novo token x, precisamos multiplicar por W^Q, W^K e por W^V para pegar a query, key e values.
Mas não queremos recomputar os vetores de key e value para os tokens anteriores $x_{<i}$.
Em vez disso, armazene os vetores de key e value na memória em um cache KV, e então apenas recuperamos os valores desse cache.

KV Cache


Q
K^T
QK^T
V
A
q1
q2
q3
q4
x
k1
k2
k3
k4
=
q1·k1 q1·k2 q1·k3 q1·k4
q2·k1 q2·k2 q2·k3 q2·k4
q3·k1 q3·k2 q3·k3 q3·k4
q4·k1 q4·k2 q4·k3 q4·k4
v1
v2
v3
v4
a1
a2
a3
a4
N x d_k
d_k x N
N x N
N x d_v
N x d_v
Q
K^T
QK^T
V
A
q4
k1
k2
k3
k4
q4·k1 q4·k2 q4·k3 q4·k4
v1
v2
v3
v4
a4
1 x d_k
d_k x N
1 x N
N x d_v
1 x d_v

Parameter-Efficient Finetuing (PEFT)

A adaptação para um novo domínio com treinamento contínue é um problema para os gigantes LLMs.

- Enorme número de parâmetros para treinamento
- Cada passo de um batch na descida do gradiente tem que propagar de volta para váriaos (VÁRIAS!!) camadas.
- Caro em poder de processamento, memória e tempo!

Em vez disso, **parameter-efficient fine tuning**(PEFT)

- Eficientemente selecione um subconjunto de parâmetros para atualizar quando for realizado o finetuning.
- Exemplo: congele alguns parâmetros
- E apenas atualize alguns parâmetros

LoRA (Low-Rank Adaptation)

- Transformers possuem várias matrizes densas
  - como W^Q, W^K, W^V, W^O
- Em vez de atualizar essas camadas durante o finetuning
  - congele essas camadas
  - Atualize uma aproximação com menos parâmetros

LoRA

- Considere a matrix W (dimensão $[N \times d]$) que precisa ser atualizada durante o finetune via gradiente descent.
  - Normalmente as atualizações são ΔW (dimensão $[N \times d]$).
- Em LoRA, as matrizes W são congeladas e atualize apenas uma decomposição de W.
  - A da dimensão $[N \times r]$
  - B da dimensão $[r \times d]$, r é bem pequeno (como 1 ou 2)
  - Isso é, durante o finetuning nós atualizamos A e B em vez de W.
  - Substitua $W + \Delta W$ com $W + BA$.

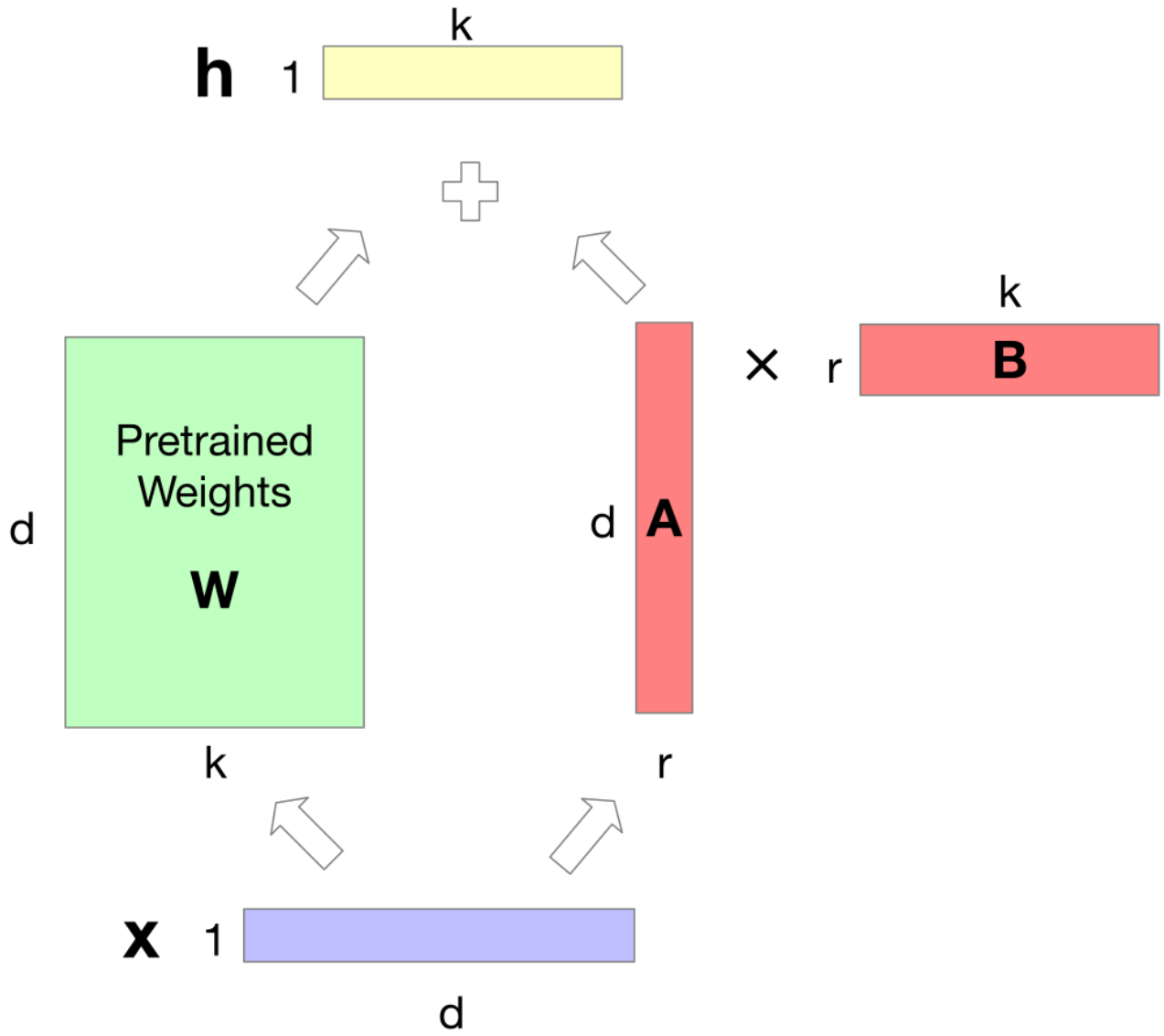
Forward pass: em vez de

$$h = xW$$

Fazemos

$$h = xW + xAB$$

LoRA



Hallucination

Chatbots May 'Hallucinate' More Often Than Many Realize

What Can You Do When A.I. Lies About You?

People have little protection or recourse when the technology creates and spreads falsehoods about them.

Air Canada loses court case after its chatbot hallucinated fake policies to a customer

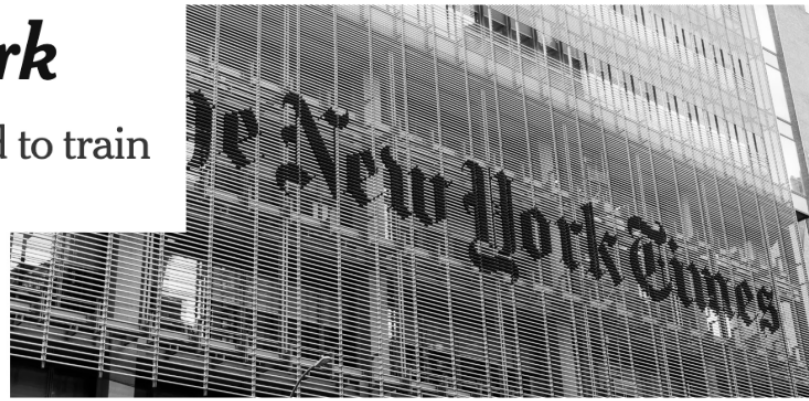
The airline argued that the chatbot itself was liable. The court disagreed.

Copyright

Authors Sue OpenAI Claiming Mass Copyright Infringement of Hundreds of Thousands of Novels

The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work

Millions of articles from The New York Times were used to train chatbots that now compete with it, the lawsuit said.



Privacy

Toxicity and Abuse

The New AI-Powered Bing Is Threatening Users.

Cleaning Up ChatGPT Takes Heavy Toll on Human Workers

Contractors in Kenya say they were traumatized by effort to screen out descriptions of violence and sexual abuse during run-up to OpenAI's hit chatbot

Misinformation

Chatbots are generating false and misleading information about U.S. elections