Laboratório de Engenharia de Sistemas e Robótica

UFPB

Modelos de Linguagem Larga (LLM)

4. Customização de Modelos

Prof.: Alisson Brito (alisson@ci.ufpb.br)

LASER

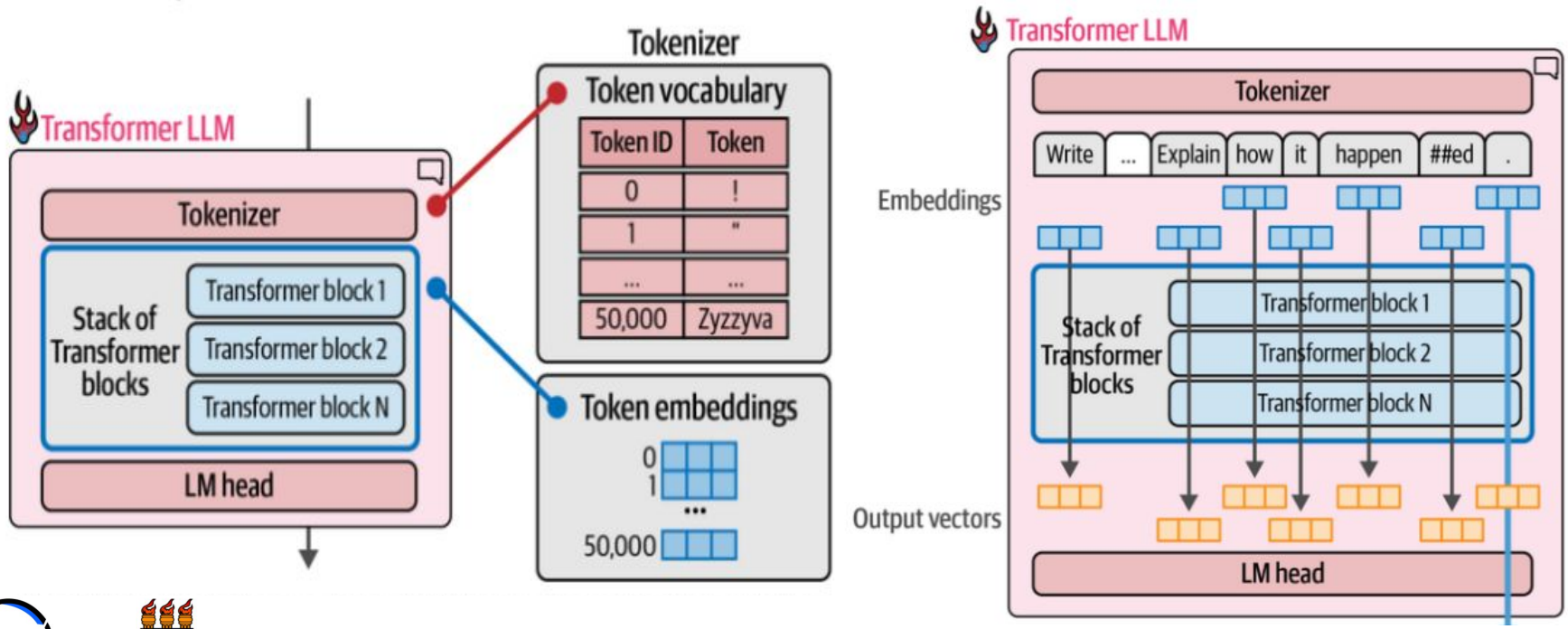
Laboratório de Engenharia de Sistemas e Robótica

UFPB

Recapitulando: Arquitetura Transformers

- **Embeddings**: convertem cada palavra/token em um vetor numérico denso que captura significado semântico.
- **Positional Encoding**: adiciona informações de posição para indicar a ordem na sequência.
- **Encoder**: lê a sequência de entrada e cria representações internas.
- **Decoder**: gera a sequência de saída (usado em tradução e geração de texto).
- **Mecanismo de atenção (Attention):**
    - **Self-Attention**: cada token compara-se com todos os outros tokens da mesma sequência para inferir quais informações são importantes.
    - **Masked Self-Attention:** similar ao Self-Attention mas com uma máscara que restringe influência dos tokens futuros nos cálculos de atenção.
    - **Multi-Head Attention**: usa várias atenções em paralelo, cada uma focando em diferentes relações entre palavras.

Tokenização e Paralelismo



Processamento Paralelo

- Diferente de RNNs, que processam token por token, Transformers processam toda a sequência de uma vez.
- Isso acelera muito o treinamento e permite lidar com sequências longas.

```
Frase: "O gato dorme"

RNN (sequencial): O → gato → dorme
Transformer (paralelo): [O, gato, dorme] → processa todos de uma vez
```

Conceituação e embasamento

Customização é o processo de **adaptar um modelo de linguagem genérico** para que ele **atenda exatamente às suas necessidades ou de um domínio específico** de forma que:

- Entenda melhor o seu contexto;
- Fale no tom ou estilo desejado;
- Responda com maior precisão às tarefas que você precisa.

Conceituação e embasamento

Formas de utilização:

- **Transfer Learning:** reaproveita o conhecimento do modelo base.
- **Prompt Engineering:** dá instruções claras para orientar o comportamento do modelo.
- **Fine-tuning:** treina o modelo com dados específicos para tarefas como QA, sumarização ou geração de texto.

Transfer Learning

Transfer Learning é uma técnica em que **um modelo pré-treinado em uma tarefa ou grande base de dados é reaproveitado para outra tarefa diferente ou mais específica**. Ou seja, o modelo **já sabe muita coisa sobre linguagem** e você só precisa **adaptá-lo para o seu objetivo**.

Como Funciona

1. **Pré-treinamento:**
   - O modelo aprende padrões gerais de linguagem (gramática, semântica, contexto).
   - Exemplo: GPT, BERT, T5, Llama.

2. **Transferência:**
   - Você reaproveita o conhecimento prévio para uma nova tarefa.
   - Pode ser aplicado em QA, sumarização, geração de texto ou qualquer domínio específico.

Prompt Engineering

Prática de **escrever instruções ou prompts cuidadosamente estruturados** para **orientar o comportamento de um modelo de linguagem**.

Em vez de re-treinar o modelo, você **controla a saída apenas fornecendo informações ou instruções bem formuladas**.

Técnicas principais

1. **Zero-shot:** O modelo recebe apenas a **instrução**, sem exemplos prévios.
2. **Few-shot:** O modelo recebe **alguns exemplos de entrada-saída** antes da pergunta real.
3. **Chain-of-Thought (CoT):** Incentiva o modelo a **raciocinar passo a passo** antes de chegar à resposta final.

Fine-tuning

O **Fine-tuning** (ou ajuste fino) é o processo de **re-treinar um modelo de linguagem já existente** usando um **conjunto de dados próprio**. Esse processo ensina o modelo a **focar em uma tarefa específica** ou **falar de um domínio de conhecimento**.

- Personaliza o modelo para **tarefas ou contextos específicos**.
- Aumenta a **precisão das respostas**.
- Reduz a dependência de *prompts* complexos.
- É a forma de **criar modelos sob medida** (customização).

Exemplo: pegar um modelo como o GPT e treiná-lo com perguntas e respostas sobre Direito ou Educação. O modelo já sabe a língua, mas ainda precisa aprender o assunto.

Fine-tuning em tarefas específicas

1. QA (Pergunta e Resposta)

- Ensina o modelo a **responder perguntas** com base em um **conteúdo específico**.
- Útil para criar **assistentes de atendimento**, **tutoriais automatizados** ou **FAQ inteligentes**
- Exemplo: o modelo responde perguntas sobre um manual ou uma disciplina específica.

```
Você é um especialista em Direito Educacional. Responda a seguinte pergunta com precisão, usando informações do material fornecido.

No fine-tuning, a “Pergunta” e a “Resposta” fariam parte do dataset de treino.
```

Fine-tuning em tarefas específicas

2. Sumarização

- Ensina o modelo a gerar **resumos coerentes e concisos** a partir de textos maiores.
- Útil para **resumos de artigos, relatórios ou atas**.
- Exemplo: transformar um artigo científico longo em um resumo de 3–5 linhas.

```
Você é um assistente que resume textos de forma clara e objetiva.

Texto: <passar o texto desejado>.

Resumo: <O modelo gera um resumo coerente e conciso de um texto maior>.
```

Fine-tuning em tarefas específicas

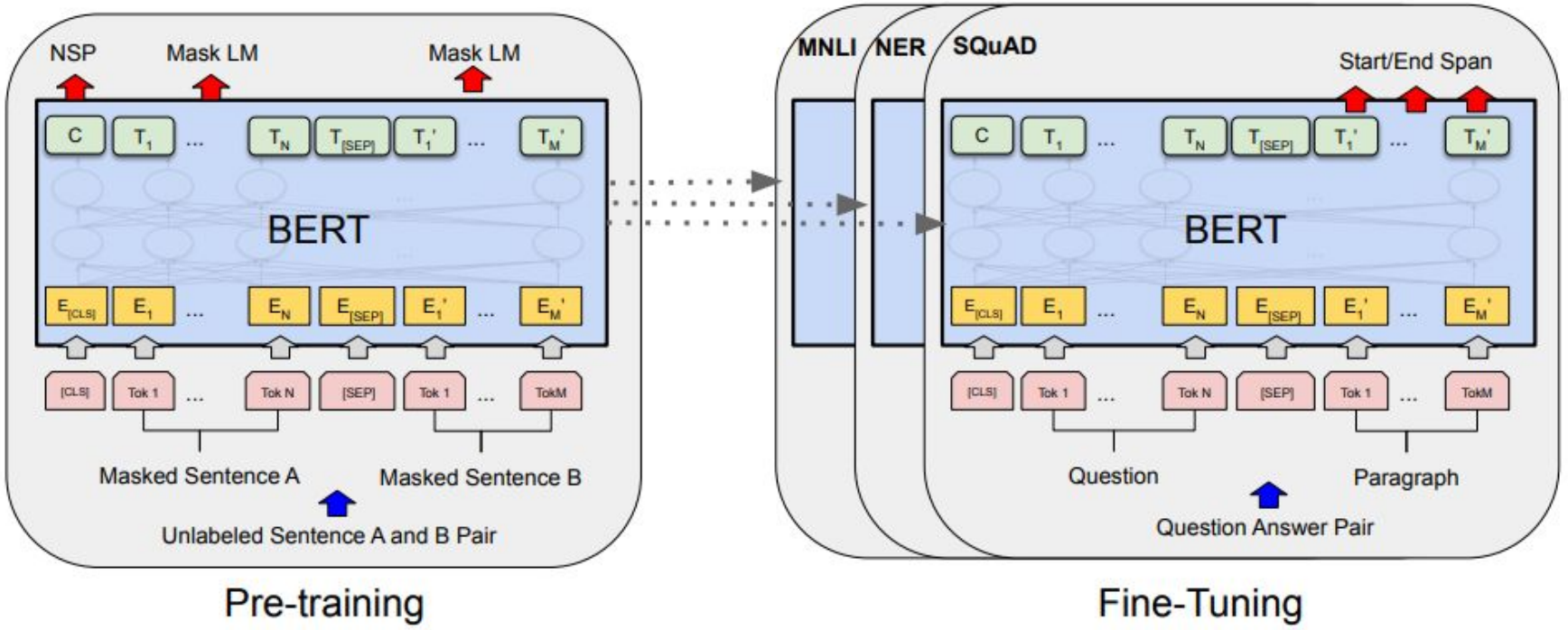
3. Geração de Texto

- O modelo aprende a criar **textos novos**, seguindo um **determinado estilo, tom ou formato**.
- Útil para **roteiros, posts, e-mails ou textos criativos**
- Exemplo: gerar e-mails formais ou postagens em tom descontraído, mantendo coerência e estilo.

Treinamento de LLMs

- Treinar LLMs significa ensinar uma rede neural gigante a entender e gerar linguagem natural, expondo-a a enormes quantidades de texto e ajustando seus parâmetros para prever a próxima palavra.
- O **Transformer** é a arquitetura central, usando **self-attention** para que cada palavra considere todas as outras na frase, capturando contexto e relações de longo alcance. Dessa forma, o modelo aprende padrões de gramática, significado e contexto para gerar respostas coerentes.
- Existem formas de treinar LLMs:
- No **pre-training**, ele aprende padrões gerais de linguagem com grandes volumes de texto usando Transformers e self-attention.
- No **fine-tuning**, o modelo é ajustado com dados específicos para tarefas concretas, aproveitando o conhecimento adquirido no pre-training.

Etapas do Treinamento

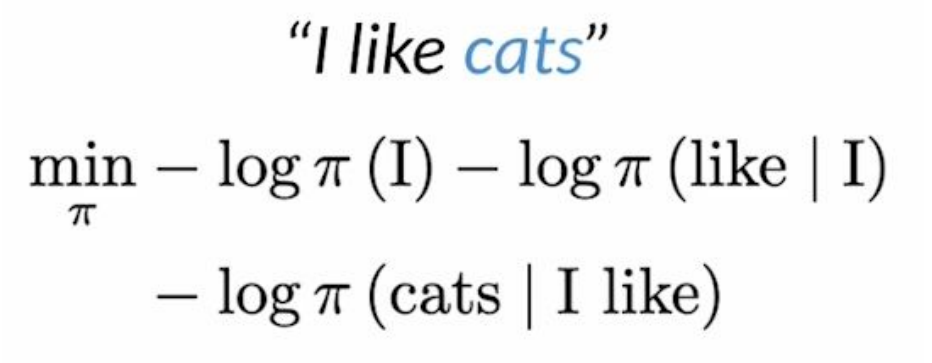


Etapas do Treinamento: **Pre-training**

- É a primeira fase do treinamento de uma LLM.
- O modelo é exposto a enormes quantidades de texto **geral e diversa**, sem foco em tarefas específicas.
- Ele "aprende" **padrões gerais de linguagem**, "gramática", "significado" e relações contextuais usando a arquitetura Transformer e a self-attention.
- O objetivo é que o modelo adquira conhecimento da(s) linguagem(ns) que está sendo treinado.

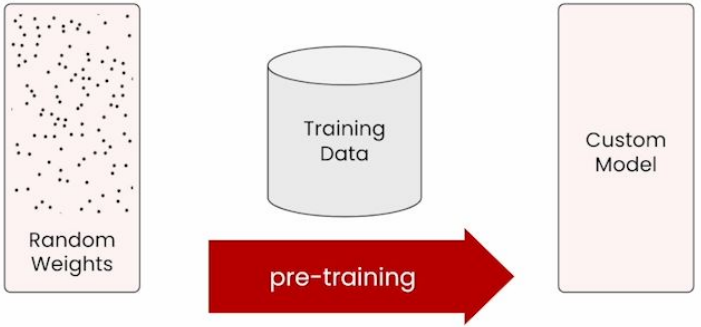
Pre-training

- Não supervisionado
- Grande corpus de texto não rotulado
  - Wikipedia, Common Craw, Github etc
  - Geralmente mais de 2T de tokens
- Trabalha minimizando probabilidade do log negativo para cada token, condicionado aos tokens anteriores
  -

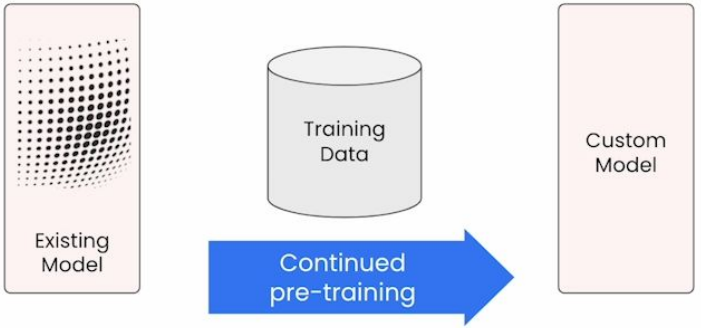


Cenários de **pré-treinamento**

- *Pre-training from scratch* - ou pré-treinamento inicial

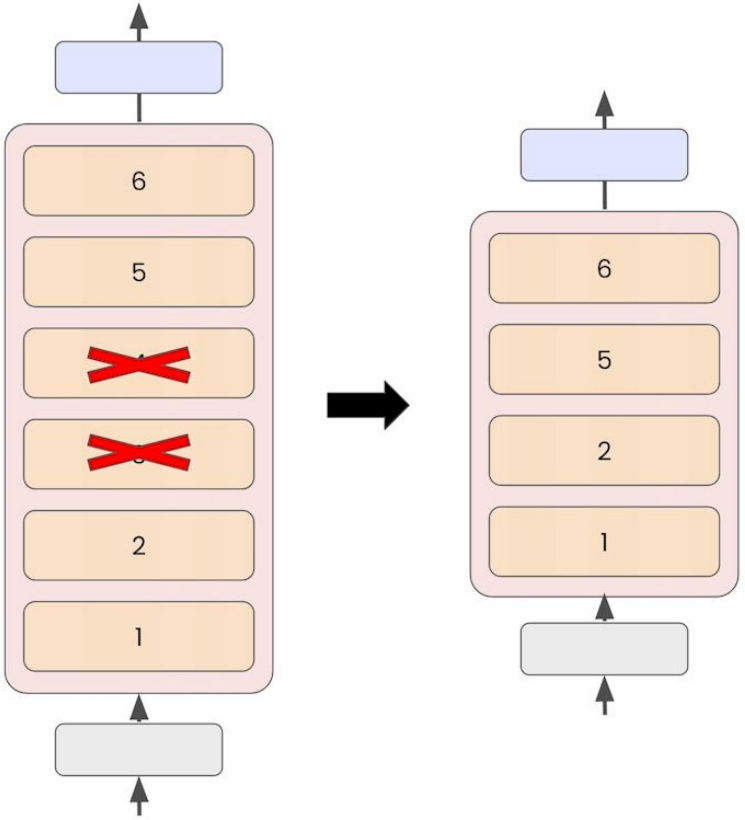


- Pré-treinamento continuado



Estratégias de pré-treinamento **continuado**

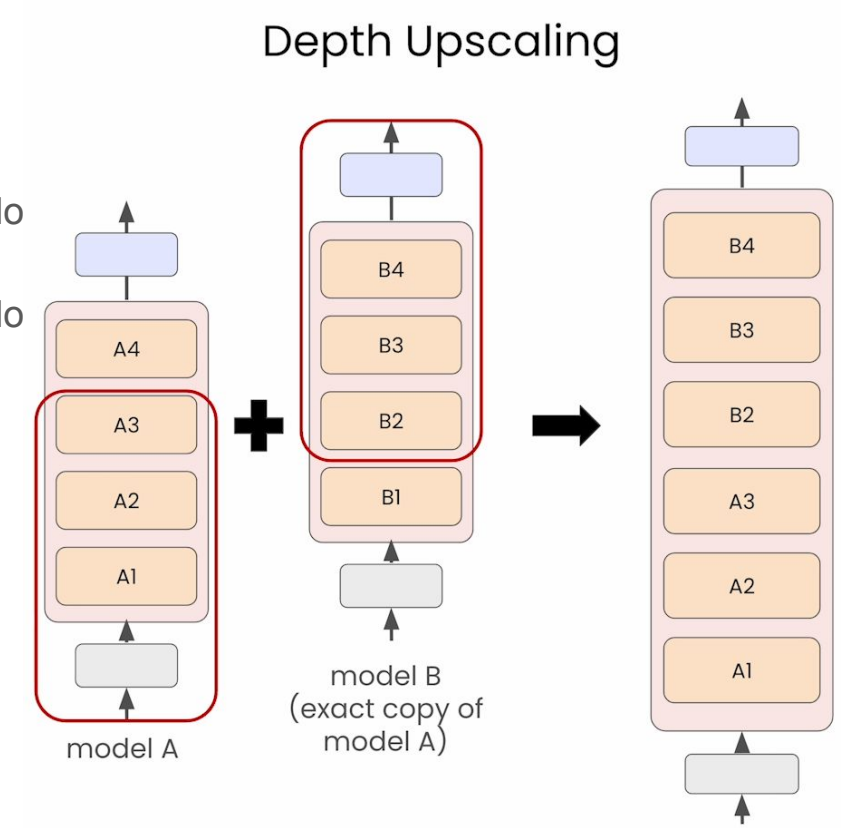
- Downscaling
  - Seleciona-se um modelo existente
  - São descartadas algumas camadas
  - Só em seguida é feito um novo treinamento dos pesos
- Usado quando se pretende pré-treinar um modelo final com tamanho menor que o original



Estratégias de pré-treinamento **continuado**

- Upscaling
  - Duplica-se um mesmo modelo em duas instâncias (A e B)
  - Seleciona-se um conjunto de camadas do modelo A
  - Seleciona-se um conjunto de camadas do modelo B
  - Faz-se uma montagem do novo modelo

- O modelo final será maior que o original



Pre-training: Caso de Uso

- Exemplo de parâmetros de Continued Pre-training
  - Técnica Upscaled a partir de 2 Mistral 7B
  - Expandindo "repertório" do LLM

Depth-up Scaling of Mistral-7B		
Model	**Name**	SOLAR-10.7B-v1.0
	Model type	Decoder
	Initial weights	2 X mistralai/Mistral-7B-v0.1
	# parameters	10.7B
	Context size	4096
	Scaling	**Upsized**
Training	**Overview**	Mistral-7B-v0.1 → Depth-Up Scaling → SOLAR-10.7B-v1.0
	Data	- **Total 1T** tokens trained - 100% unsupervised
	Hyper-parameters	- learning_rate: 1e-5 - sequence_length: 4k - batch_size: 4k - weight_decay: 0.1
	Price	~ $1M

Pre-training: Caso de Uso

- Caso de refinamento de modelo
  - Continued Pre-training com datasets em idioma Koreano

Korean language extension of SOLAR-10.7B		
Model	**Name**	SOLAR-10.7B-v1.0-enko
	Model type	Decoder
	Base Mode:	upstage/SOLAR-10.7B-v1.0 (English)
	# parameters	10.7B
	Context size	4096
	Scaling	same size
Training	**Overview**	SOLAR-10.7B-v1.0 → Korean language extension → SOLAR-10.7B-v1.0-enko
	Data	- Total 200B tokens trained - 100% unsupervised
	Hyper-parameters	- learning_rate: 1e-5 - sequence_length: 4k - batch_size: 4k - weight_decay: 0.1
	Price	~ $0.2M

Etapas do Treinamento: **Fine-tuning**

- Partindo de um **modelo base**, ou seja pré-treinado para prever o próximo token, é aplicada nova técnica de treinamento para adicionar novas habilidades

- **Conjuntos de dados menores e mais específicos**.

- O fine-tuning pode “refinar” parâmetros aprendidos, ou mesmo adicionar novos, para que o modelo se comporte conforme tarefa desejada, sem perder o conhecimento geral adquirido no pre-training.

Etapas do Treinamento: **Fine-tuning**

- Exemplos de capacidades:
  - *Instruct/Chat Model*: capacidade de seguir instruções ou manter um diálogo
  - *Reasoning*: capacidade do LLM de quebrar problemas complexos em etapas lógicas e intermediárias antes de fornecer a resposta final
  - *Tool-Use* ou *Function Calling* (nativo): capacidade de um LLM de identificar a intenção do usuário e, em vez de responder diretamente, gerar um código ou uma chamada de função para uma ferramenta externa

Construindo LLMs

Pré-treinamento

Inicialização do modelo

Aprendizado a partir de grande corpus de textos (geralmente não supervisionado)

Base Model

Predicts next word / token

Pós-treinamento com Fine-Tuning

Aprendizado a partir de dados curados

Instruct / Chat Model

Respond to instructions

Q: What is the capital of France?
A: The capital of France is Paris.

Mudando comportamentos ou melhorando habilidades

Customized Model

Specialized in certain domain or have specific behaviors

Q: Write me a SQL query for
A: SELECT * FROM …

Full vs PEFT vs Head Extension

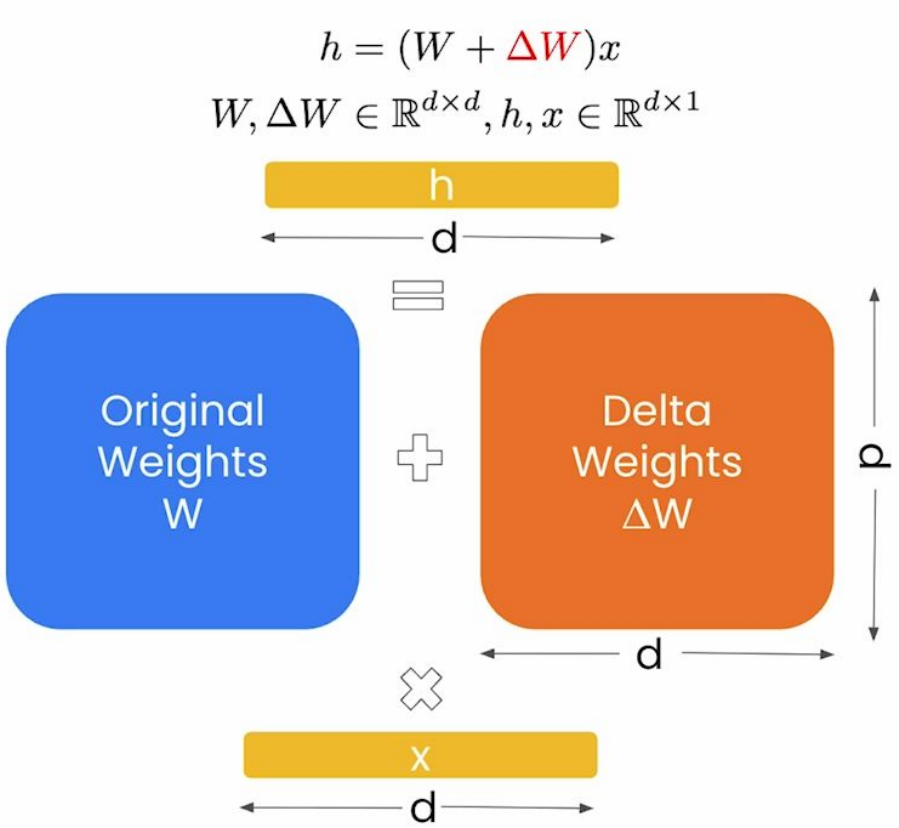
- Full Fine-tuning
  - Conceitualmente bem direto: ajustar todos os pesos originais W ao adicionar o ajuste ΔW em cada passo de otimização.

- Parameter Efficient Fine-Tuning
  - Quantidade de parâmetros novos (treináveis) é drasticamente menor do que o número total de parâmetros

- Task Head ou Head Extension
  - Adição de uma ou mais camadas de saída no topo de um modelo pré-treinado

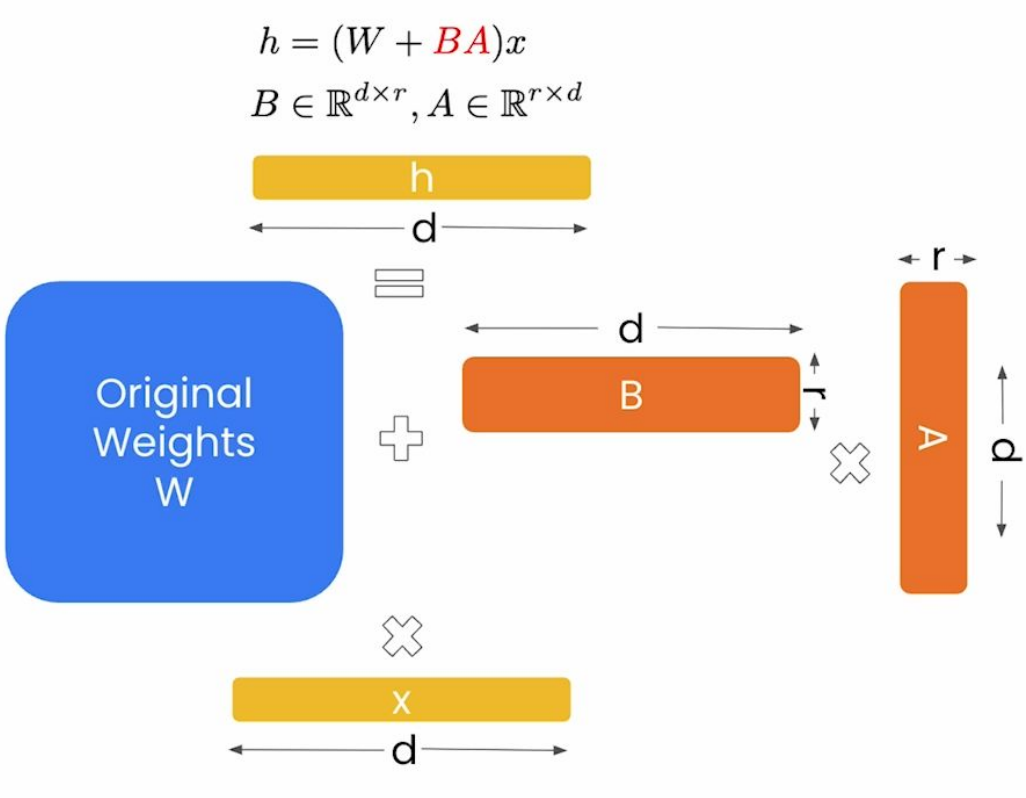
Full Fine-tuning

- **x** é a entrada e **h** a saída
- **W, ΔW ∈ Rd×d**: As matrizes de pesos (Original e Delta) têm **d linhas** e **d colunas**.
- **h, x ∈ Rd×1**: Os vetores de entrada (x) e saída (h) têm **d linhas** e **1 coluna** (um vetor coluna de dimensão d)
- Ou seja, todos os parâmetros podem ser alterados no treinamento!
  - Nem sempre é benéfico pois pode "esquecer" conteúdos aprendidos
  - Custo



Parameter Efficient Fine-Tuning (PEFT)

- **(W+BA)**: Representa os **pesos ajustados**.
- O ajuste (ΔW) do modelo anterior é agora substituído pelo produto matricial **BA**.
- **r** é um número **muito menor** do que d (a dimensão total). Por exemplo, se d for 1024 (1 milhão de parâmetros em W), r pode ser apenas 4 ou 8.
- **LoRA**
  - **LoRA Learns Less and Forgets Less** (https://arxiv.org/pdf/2405.09673)



Parameter Efficient Fine-Tuning (PEFT)

- Apenas A e B são treinados e atualizados
- A matriz original W é congelada
- O número de parâmetros a serem armazenados e atualizados (os de A e B) é minúsculo (muitas vezes menos de 1%) em comparação com os parâmetros totais em W

Task Head

- Técnica mais tradicional e inicial de adaptação de modelos pré-treinados
- O modelo pré-treinado é congelado (seus pesos W originais não são atualizados)
- Um novo conjunto de camadas lineares ou densas (o head) é adicionado, e apenas esses novos pesos são treinados para a nova tarefa

Task Head

- Exemplo: Um modelo de linguagem pré-treinado, que gera representações de texto, adicionando um "Task Head" para classificar sua saída em positivo ou negativo
  - Apenas os pesos desse novo head de classificação seriam treinados
  - O modelo base serve apenas como um extrator de características

- Especialmente comum com modelos como o BERT (e seus antecessores/sucessores como RoBERTa, XLNet, etc.)

Task Head

Tarefa	Tipo de Cabeça (Camada Densa Final)
Classificação de Texto	Uma camada densa seguida por uma função de ativação Softmax para obter probabilidades.
Geração de Texto	Uma camada densa seguida por um decoder (em arquiteturas encoder-decoder).
Resposta a Perguntas	Uma cabeça que calcula as probabilidades da resposta no texto.

Possibilidades de combinação Task Head

Conceito	Método	Onde o "Head" se Encaixa
Full Fine-tuning	$h=(W+\Delta W)x$	Se você usar Full Fine-tuning, o corpo do modelo (W) e o Task Head (se adicionado) é treinado.
PEFT (LoRA)	$h=(W+BA)x$	Geralmente, no LoRA, você treina o Task Head (se houver) e adiciona módulos LoRA (BA) às camadas internas (como as camadas de atenção) do corpo do modelo.
Congelamento com Task Head	$h=Wx$ (apenas W é usado, mas a saída vai para o novo head)	O Task Head é frequentemente usado sozinho em uma técnica chamada Linear Probing ou Fine-tuning da última camada, onde o W é totalmente congelado, e apenas o novo head é ajustado.

Fine-Tuning

- Necessário definir Método, Biblioteca e Avaliação
- Métodos de pós-treinamento (fine-tuning)
  - Supervised Fine-Tuning
  - Direct Preference Optimization
  - Online Reinforcement Learning
  - Etc
- Bibliotecas
  - Huggingface TRL
  - OpenRLHF
  - veRL
  - Nemo RL
  - Etc

Supervised Fine-Tuning (SFT)

- Supervised ou Imitation Learning
- Trabalha com pares de **Prompt** e **Resposta**
- Corpus necessário contém em média 1K a 1T de tokens
  - A qualidade dos dados importa muito mais do que a quantidade
  - Corpus de **1K** de alta qualidade é mais efetivo que corpus de **1M** com misto de qualidade
- O erro é calculado apenas sobre os tokens da resposta

$$\min_{\pi} -\log \pi \left(\text{Response} \mid \text{Prompt}\right)$$

SFT - Casos de Uso

- Habilitar novo comportamento
  - Transformar modelo pré-treinado → instruct model
  - Transformar modelo em reasoning model
  - Habilitar Tools nativamente em um modelo
  - Etc
- Melhorar habilidades existentes a partir de outros modelos
  - Modelo maior gera dados sintéticos para treinar respostas desejadas em modelos menores, destilando assim novas capacidades
  - A partir do dataset de prompts são obtidas as respostas de um modelo "ideal"
  - As respostas são organizadas com respectivos prompts para o pós-treinamento
  - Após o fine-tuning o modelo menor conseguirá imitar o modelo maior "ideal"

Direct Preference Optimization (DPO)

- Trabalha com tuplas de **Prompt** e **Resposta Boa** e **Resposta Ruim**
  - Prompt: "Como treinar um LLM?"
  - Resposta Boa: "Para treinar um LLM …"
  - Resposta Ruim: "Desculpe, não …"
- Corpus necessário contém em média 1K a 1T de tokens
- Função de erro um pouco mais sofisticada
  - Recompensa respostas próximas à "Resposta Boa"
  -

$$\min_{\pi} -\log \sigma \left(\beta \left(\log \frac{\pi(\text{Good R} \mid \text{Prompt})}{\pi_{\text{ref}}(\text{Good R} \mid \text{Prompt})} - \log \frac{\pi(\text{Bad R} \mid \text{Prompt})}{\pi_{\text{ref}}(\text{Bad R} \mid \text{Prompt})} \right) \right)$$

Online Reinforcement Learning

- Trabalha com **Prompt** e **Função de Recompensa**
  - Prompt: "Como treinar um LLM?"
  - Resposta do modelo: "Para treinar um LLM …"
  - Resultado da Função de Recompensa: "0.94"
- Corpus necessário contém em média 1K a 10M de tokens
- A função de erro maximiza o valor obtido pela **Função de Recompensa**
  - A função deve calcular um valor para a resposta, de forma que maiores valores representam melhores respostas

$$\max_{\pi} \ \text{Reward}(\text{Prompt}, \text{Response}(\pi))$$

2. Escala e Avaliação

- Pré-processamento de datasets massivos.
- Treinamento distribuído em larga escala.
- Métricas de avaliação

Pré-processamento de dados

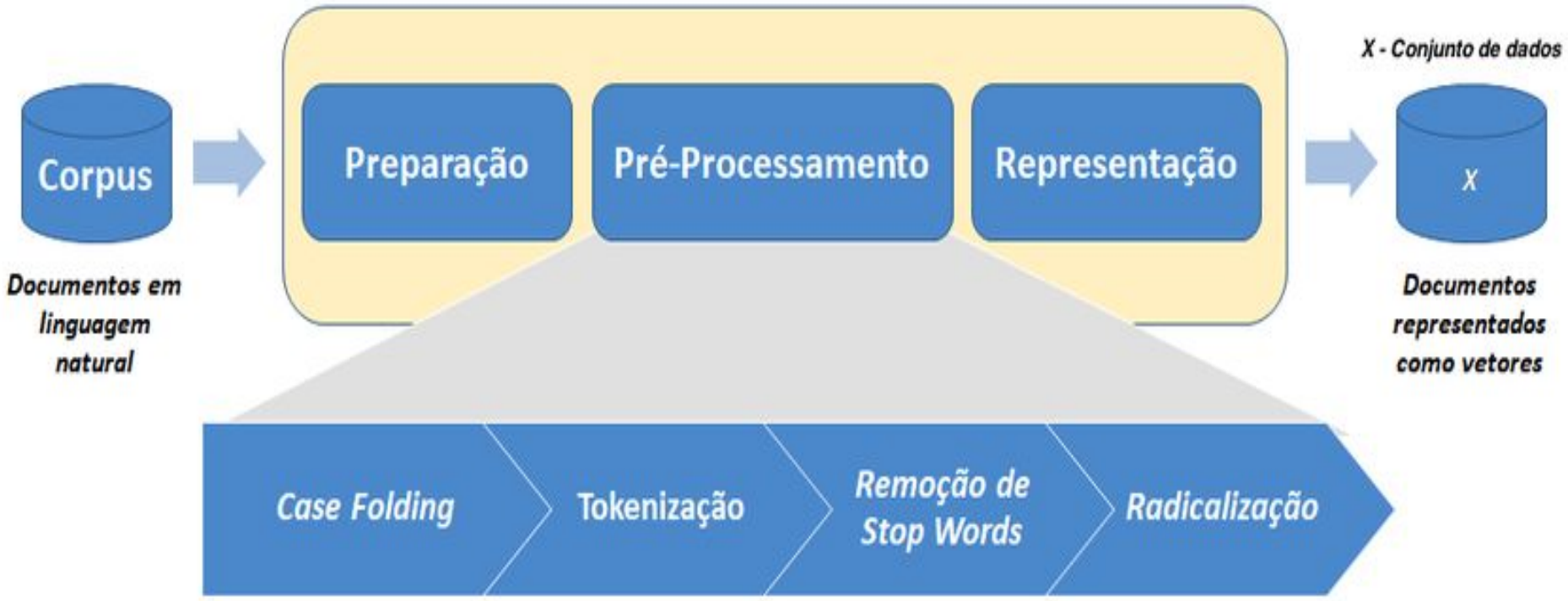
O **pré-processamento de datasets massivos** para LLMs é o conjunto de técnicas e etapas aplicadas a grandes volumes de dados textuais (ou multimodais) antes do treinamento de modelos de linguagem

Seu objetivo de **limpar, normalizar, filtrar e estruturar** os dados de forma que eles sejam **consistentes, relevantes e eficientes para o aprendizado do modelo**.

Em outras palavras:

Transformar dados brutos em um formato adequado para que o LLM consiga aprender padrões linguísticos, sem ser prejudicado por duplicatas, erros, ruído ou informações sensíveis.

Etapas para pré-processamento em linguagem natural.



Características do pré-processamento em LLMs:

Escalabilidade: precisa lidar com datasets que podem ter **centenas de GB a TBs de texto**.

Qualidade: filtra dados que são irrelevantes, duplicados, tóxicos ou de baixa qualidade.

Uniformidade: normaliza textos, tokeniza e divide em formatos compatíveis com o modelo.

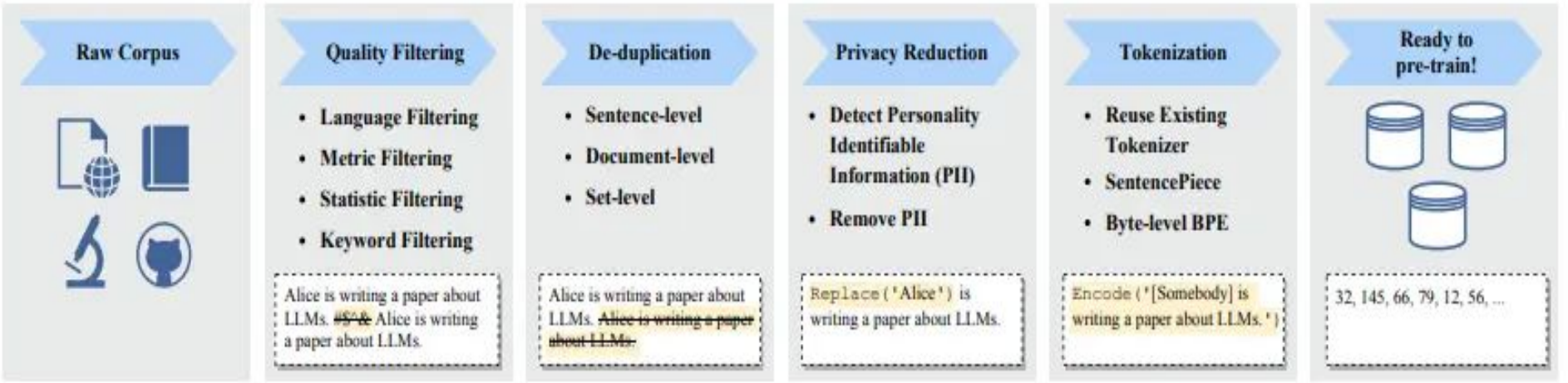
Eficiência de treinamento: organiza os dados para permitir **leitura rápida e streaming** durante o treino.

Principais Desafios em Datasets para LLMs

1. Volume massivo de dados
2. Qualidade e ruído
3. Viés e toxicidade
4. Heterogeneidade
5. Dados sensíveis ou privados
6. Balanceamento e representatividade
7. Eficiência de leitura e tokenização

Técnicas de Pré-Processamento

As técnicas de pré-processamento têm como objetivo **preparar dados brutos para que um LLM consiga aprender de forma eficiente e segura**. Em grandes volumes de dados, textos vêm de fontes heterogêneas, com formatos diferentes, ruído, erros e conteúdos repetidos ou problemáticos. Sem essas técnicas, o modelo pode aprender padrões incorretos, gerar respostas imprecisas ou até reproduzir vieses.



Técnicas de Pré-Processamento

- **Limpeza e normalização:** eliminam erros, caracteres estranhos, códigos HTML, espaços extras, emojis inconsistentes ou diferenças de caixa de texto, garantindo que os dados fiquem consistentes.
- **Filtragem e detecção de duplicatas:** removem textos repetidos, tóxicos ou com informações sensíveis, preservando a qualidade e a segurança do dataset.
- **Tokenização e chunking:** transformam textos em unidades (tokens) compreensíveis pelo modelo e dividem textos longos em blocos menores, mantendo o contexto, para que o treinamento seja eficiente.
- **Balanceamento e organização:** asseguram diversidade de idiomas, domínios e estilos, evitando que o modelo fique enviesado para determinados tipos de texto
- **Otimização e escalabilidade:** preparam os dados em formatos que permitem leitura rápida e processamento distribuído, essencial quando lidamos com datasets de centenas de GB ou TB.

Treinamento distribuído em larga escala

Quando enfrentamos tarefas gigantescas, uma estratégia eficiente é **dividi-las em subtarefas e executá-las em paralelo**. Isso economiza tempo e torna o trabalho complexo **realizável**.

O treinamento de **LLMs (Large Language Models)** envolve bilhões ou trilhões de parâmetros, exigindo **massiva capacidade computacional e memória**.

Por isso, não é viável treinar esses modelos em um único GPU ou máquina.

O **treinamento distribuído** permite dividir a carga entre múltiplos dispositivos ou nós, acelerando o processo e possibilitando lidar com datasets massivos.

Principais conceitos e técnicas

Data Parallelism (Paralelismo de Dados):

- Cada nó recebe uma **porção diferente do dataset** e mantém uma cópia completa do modelo.
- Gradientes são sincronizados entre os nós a cada passo.
- Útil quando o modelo cabe em cada dispositivo, mas o dataset é enorme.

Model Parallelism (Paralelismo de Modelo):

- O modelo é **dividido entre vários dispositivos**, cada um processando uma parte da rede.
- Necessário quando o modelo é **grande demais para caber em um único GPU**.

Principais conceitos e técnicas

Pipeline Parallelism:

- O modelo é dividido em estágios sequenciais, e cada estágio é alocado em um dispositivo.
- Permite que diferentes mini-batches avancem em paralelo pelos estágios, aumentando a utilização do hardware.

ZeRO e técnicas de otimização de memória:

- Dividem os gradientes, estados do otimizador e parâmetros entre múltiplos GPUs, permitindo treinar modelos maiores sem exigir memória individual gigantesca.

Treinamento híbrido:

- Combina data parallelism + model/pipeline parallelism, adaptando-se à arquitetura do cluster e ao tamanho do modelo.

Benefícios do treinamento distribuído

Escalabilidade: permite treinar modelos com bilhões de parâmetros.

Eficiência: reduz tempo de treinamento de semanas para dias ou horas.

Flexibilidade: possibilita trabalhar com datasets massivos, que não caberiam em uma única máquina.

Métricas de avaliação

As métricas de avaliação têm como objetivo **medir a qualidade e a eficácia de um modelo de linguagem** após o treinamento. Como os LLMs podem gerar texto de forma aberta e flexível, é importante usar métricas que capturem **precisão, coerência, relevância e segurança** das respostas.

- Ao efetuar pós-treinamen

Métricas de avaliação

As métricas de avaliação para LLMs podem ser divididas em três categorias:

- **Similaridade:** comparam respostas do modelo com textos de referência.
  Ex.: BLEU, ROUGE, METEOR, BERTScore.
- **Probabilidade / Modelos internos:** avaliam confiança do modelo ou qualidade da sequência. Ex.: **Perplexidade**; quanto menor, melhor a predição.
- **Rank / Recuperação:** medem se o modelo prioriza respostas corretas.
  Ex.: **Mean Reciprocal Rank (MRR)**, que avalia a posição da primeira resposta correta retornada pelo modelo.

Similaridade

Pontuação BLEU

Avalia correspondência de n-gramas em tradução, o quão próximo o resultado se assemelha à Verdade Fundamental, ou seja, mede o quanto um texto gerado pela máquina se parece com um ou mais textos de referência escritos por humanos

$$\text{Pontuação BLEU} = BP * \exp(\sum(peso_i * \log(p_i)))$$

Onde

> BP = *Penalidade de Brevidade*
>
> $peso_i$ = *peso atribuído à precisão de cada n-grama*
>
> p_i = *Precisão para cada n-grama*

Exemplo Pontuação BLEU

Frase de referência: “O gato está no tapete.”
Frase candidata: “O gato sentou no tapete.”

- **Conta n-gramas em comum** (palavras ou grupos de palavras iguais nas duas frases);
- **Calcula a precisão** — quantos n-gramas do candidato aparecem nas referências;
- **Aplica um “penalizador de comprimento”** (brevity penalty), para evitar que frases curtas demais ganhem nota alta;
- **Combina os resultados** para gerar uma nota entre **0 e 1**.

Exemplo Pontuação METEOR

O **METEOR** foi criado como alternativa ao BLEU para avaliação de **tradução automática**.
Ele mede a sobreposição entre o texto gerado e a referência, mas vai além da correspondência exata de n-gramas:

- considera **sinônimos** (usando dicionários como WordNet),
- **radicais (stemming)**,
- e também a **ordem das palavras**.

A ideia é capturar melhor a **qualidade semântica** do texto, não só a similaridade superficial.

Frase de referência: “O gato está no tapete.”
Frase candidata: “O gato sentou no tapete.”

Mesmo que não haja bigramas exatos, reconhece ***“gato*** ↔ ***felino”*** **e** ***“tapete*** ↔ ***carpete”*** como sinônimos, resultando em um score relativamente alto, enquanto o BLEU daria quase zero.

Pontuação ROUGE

A **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation) é uma métrica usada para avaliar a **qualidade de textos gerados automaticamente**, como resumos ou traduções, comparando-os com textos de referência humanos.

- **ROUGE-N (n-gramas):** mede a sobreposição de n-gramas entre o texto gerado e o texto de referência (ex.: unigramas, bigramas).
- **ROUGE-L (Longest Common Subsequence - LCS):** avalia a **maior subsequência comum** entre os textos, considerando a estrutura da frase.
- **ROUGE-S (Skip-Bigram):** compara pares de palavras ou sequências descontínuas, capturando similaridades mais flexíveis.

Em geral:

Valores **mais altos** → maior similaridade → melhor desempenho do modelo.
Usada principalmente em **resumos automáticos**, **tradução de máquina** e **geração de texto**.

Pontuação BERTScore

O BERTScore utiliza um modelo BERT pré-treinado para avaliar a qualidade do texto gerado em comparação com um texto de referência. Ele mede a similaridade semântica entre os dois textos usando embeddings de BERT.

- Cada palavra do texto gerado é comparada com cada palavra do texto de referência em um **espaço vetorial semântico**.
- Calcula-se **precisão, recall e F1** com base nessas similaridades.
- Diferente de métricas tradicionais (ROUGE, BLEU), o BERTScore **leva em conta o significado**, e não apenas a correspondência literal de palavras.

Interpretação:

- Valores **mais altos** → maior semelhança semântica entre o texto gerado e o texto de referência.
- Útil para **resumos, tradução automática e geração de texto**, especialmente quando palavras diferentes podem ter **mesmo significado**.

Exemplo Pontuação BERTS

Texto de referência:
“O gato sentou no tapete.”

Etapa 1: Tokenização e Geração de Incorporação

- **Tokens de referência:** ["O", "gato", "sentado", "em", "o", "tapete", "."]
- **Tokens gerados pelo modelo:** ["A", "gato", "estava", "sentado", "em", "o", "tapete", "."]

Etapa 2: Incorporação da Geração com BERT

- Cada token é transformado em um **vetor de incorporação** pelo modelo BERT.
- **Tokens de referência:** ["O", "gato", "sentado", "em", "o", "tapete", "."]
- Vetores [v1, v2, v3, v4, v5, v6, v7]

Probabilidade

Probabilidade: Perplexidade

A **perplexidade** é uma métrica usada para avaliar modelos de linguagem, como aqueles que predizem a próxima palavra em uma frase. Ela mede o quão **surpreso ou confuso** o modelo fica ao tentar prever os próximos elementos de uma sequência de texto.

- Menor perplexidade → modelo prevê melhor a próxima palavra
- Maior perplexidade → mais confusão na previsão
- **Perplexidade = 1** → previsão perfeita
- **Perplexidade = 10** → média de 10 opções igualmente prováveis

Exemplo Perplexidade

Passo 1: Calcular probabilidades condicionais das palavras

- P(“O”) = 0,5
- P(“gato”|”O”) = 0,4
- P(“sentou”|”O gato”) = 0,3
- P(“sobre”|”O gato sentou”) = 0,4
- P(“o”|”O gato sentou em”) = 0,5
- P(“tapete”|”O gato sentou no”) = 0,6

Exemplo Perplexidade

Passo 2: Aplicar logaritmo e somar

$$\log(0,5) + \log(0,4) + \log(0,3) + \log(0,4) + \log(0,5) + \log(0,6) = abc$$

Passo 3: Calcular média e aplicar exponencial $\text{Perplexidade} = \exp\left(\frac{abc}{6}\right) = 2,275$

Portanto, Perplexidade=2,275, o que significa que o modelo precisa escolher entre cerca de 2,275 palavras para escolher a próxima palavra na sequência.

Rank ou Recuperação

Rank: Mean Reciprocal Rank (MRR)

A métrica de Classificação Recíproca Média (MRR) verifica a eficiência do LLM em colocar a resposta correta perto do topo da lista. Portanto, é uma boa métrica para tarefas de Classificação ou Recuperação.

Interpretação:

- Valor **próximo de 1** → o sistema retorna a resposta correta muito rapidamente (top da lista).
- Valor **mais próximo de 0** → a resposta correta aparece mais abaixo na lista ou não aparece.

MRR é muito usado em **buscadores, chatbots e sistemas de recomendação**, pois reflete a **eficiência do ranking de respostas**.

Exemplo MRR

Para cada consulta, calcula-se a **recíproca da posição da primeira resposta correta**:

$$\text{Reciprocal Rank} = \frac{1}{\text{posição da primeira resposta correta}}$$

O **MRR** é a média dessas recíprocas em todas as consultas:

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{posição}_i}$$

Se você fizer isso para várias perguntas, calcule o RR para cada uma e, em seguida, encontre a média. Essa média é a MRR. Se o assistente acertar com frequência nas primeiras posições, a MRR será alta, próxima de 1. Se isso acontecer raramente, a MRR será menor, próxima de 0.

Exemplo MRR

1. Pergunta: “Qual é a capital da França?”
Respostas do LLM:

1. Londres
2. **Paris**
3. Berlim

2. Cálculo:

- Resposta correta na 2ª posição; Logo a classificação Recíproca será: RR= 1/ Posição da resposta correta **RR = 1 / 2 = 0,5**

Avaliação de LLMs

Human Preferences for chat	**Chatbot Arena**
LLM as a judge for chat	Alpaca Eval MT Bench **Arena Hard V1 / V2**
Static Benchmarks for Instruct LLM	**LivecodeBench** **AIME 2024 / 2025** GPQA MMLU Pro IFEval
Function Calling & Agent	BFCL V2 / V3 NexusBench V1 / V2 **TauBench** **ToolSandbox**

Dúvidas?