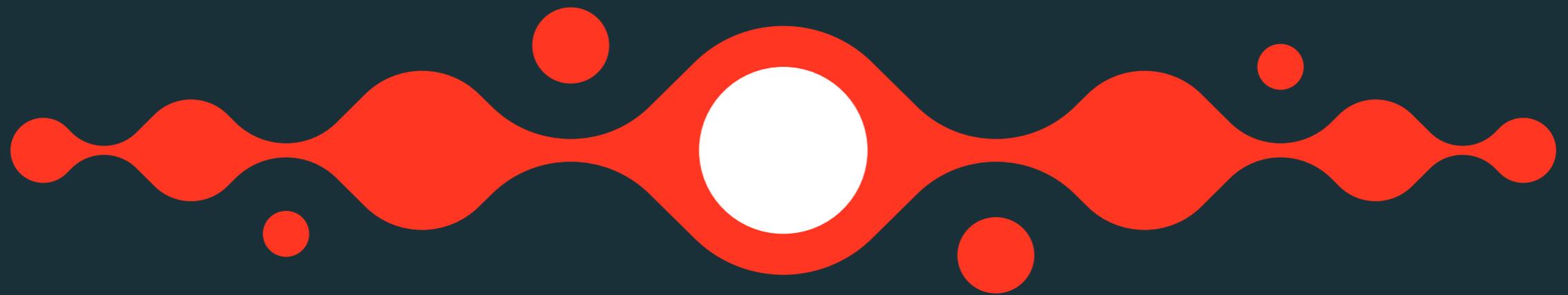


O Livro Completo de GenAI



5 maneiras de aproveitar seus dados para
criar aplicativos de GenAI de qualidade



CONTEÚDO



Introdução	3
O caminho para a implementação de aplicações de GenAI de qualidade de produção	5
Estágio 0: modelos de base	5
Caso de uso: apresentando o DBRX: um novo LLM aberto de última geração	5
Estágio 1: engenharia de prompts	19
Caso de uso: análise automatizada de avaliações de produtos usando grandes modelos de linguagem	20
Estágio 2: geração aumentada de recuperação	25
Caso de uso: melhore a qualidade de resposta de sua aplicação RAG com dados estruturados em tempo real	27
Estágio 3: ajuste fino de um modelo de base	33
Caso de uso: criação de um LLM personalizado para documentação gerada por IA	34
Caso de uso: ajuste fino eficiente com LoRA: um guia para seleção de parâmetros ideais para modelos de linguagem de grande escala	43
Estágio 4: pré-treinamento	60
Caso de uso: treinando o difusão estável do zero por menos de US\$ 50 mil com a MosaicML	62
Aplicação prática: análise detalhada: como treinamos a difusão estável com um orçamento inferior a US\$ 50 mil	68
Estágio 5: avaliação de LLM	81
Caso de uso: melhores práticas para avaliação de LLM da aplicação RAG	82
Caso de uso: avaliação de LLM offline: avaliação passo a passo da aplicação de IA Generativa na Databricks	98
Resumo	117
Treinamento em GenAI	117
Recursos adicionais	117

Introdução



Alcançar a qualidade de produção da IA Generativa requer novas ferramentas e habilidades

A IA generativa abriu novos mundos de possibilidades para as empresas e está sendo fortemente adotada por organizações em diversos setores. De acordo com um relatório recente da [MIT Tech Review](#), todos os 600 CIOs pesquisados afirmaram que estão aumentando seu investimento em IA, e 71% planejam criar seus próprios modelos de linguagem de grande escala (LLMs) personalizados ou outros modelos de IA generativa. No entanto, muitas organizações acham difícil implementar essas aplicações com qualidade de produção. Para atender ao padrão de qualidade exigido para aplicações voltadas para o cliente, a saída da IA deve ser precisa, governada e segura.

A infraestrutura de dados deve evoluir para ser compatível com aplicativos impulsionados por IA generativa

Adotar a IA generativa não se resume a implementar um chatbot; exige uma reformulação dos aspectos fundamentais do gerenciamento de dados. No centro dessa transformação está o surgimento dos [data lakehouses](#) como a nova "stack de dados moderna". Essas arquiteturas de dados avançadas são essenciais para aproveitar todo o potencial da IA generativa, gerando uma democratização mais rápida, econômica e ampla das tecnologias de dados e IA. À medida que as empresas dependem cada vez mais de ferramentas e aplicativos baseados em IA generativa para obter vantagem competitiva, a infraestrutura de dados subjacente deve evoluir para ser compatível com essas tecnologias avançadas de forma eficaz e segura.

Independentemente de onde você esteja na jornada de implementação de aplicações de IA generativa, a qualidade dos dados é crucial.

As empresas precisam atingir qualidade de produção com suas aplicações de IA generativa. Os desenvolvedores precisam de ferramentas avançadas para compreender a qualidade de seus dados e das saídas dos modelos, juntamente com uma plataforma subjacente que lhes permita combinar e otimizar todos os aspectos do processo de IA generativa. A IA generativa possui muitos componentes, como preparação de dados, modelos de recuperação, modelos de linguagem (sejam SaaS ou de código aberto), pipelines de classificação e pós-processamento, engenharia de prompts e modelos de treinamento em dados corporativos personalizados.

Para ajudar você a superar os desafios comuns das empresas com a criação da IA generativa, compilamos uma coleção de conteúdo técnico e exemplos de código. Iniciaremos cada seção com uma breve visão geral, seguida de casos de uso e exemplos de código para referência.

Neste e-book, você aprenderá:

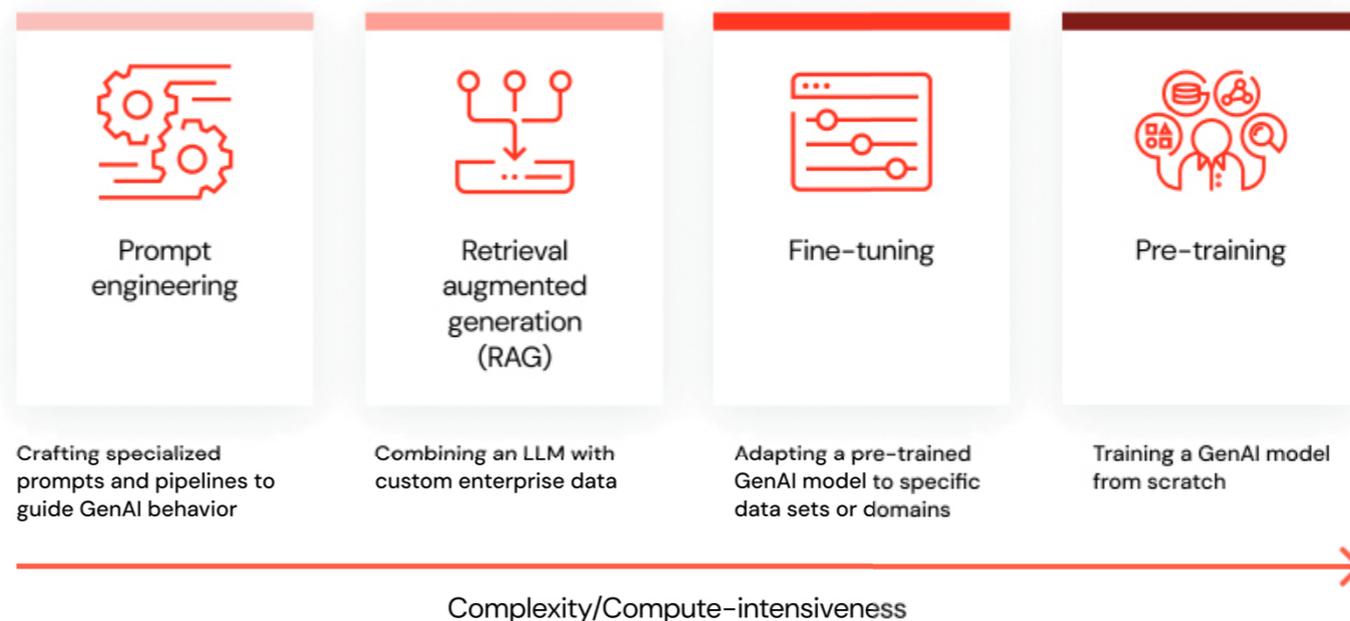
- Como planejar um caminho de aplicações de IA generativa do básicos ao avançado, aproveitando os dados de sua organização
- Como usar a geração aumentada de recuperação (RAG) para tornar mais inteligente um sistema de IA pronto para uso
- Como avaliar os LLMs e identificar onde você deve investir em ferramentas e sistemas de IA mais potentes que proporcionem maiores ganhos operacionais
- Como criar um LLM personalizado que pode ser mais eficiente, rápido e econômico para sua organização
- Quando pode ser vantajoso pré-treinar seu próprio modelo — e muito mais

Casos de uso para IA generativa abordados:

- Como usar LLMs para obter insights praticáveis de avaliações de produtos
- Como usar RAG em um chatbot para melhorar a qualidade do output
- Como treinar seu próprio modelo de IA generativa de maneira econômica
- Como monitorar e avaliar seus aplicativos de LLMs e IA generativa implementados

GenAI journey

Plan an iterative path from basic to advanced GenAI, leveraging your data.



O caminho para implementar aplicativos de IA generativa com qualidade de produção



Estágio 0: Modelos básicos

Antes de começar a criar aplicativos de IA generativa com qualidade de produção, precisamos abordar os modelos básicos de linguagem que servem como base para camadas de técnicas cada vez mais complexas. Os modelos básicos geralmente se referem a grandes modelos de linguagem que foram treinados em conjuntos de dados extensos para serem, de forma geral, eficazes em tarefas específicas (bate-papo, acompanhamento de instruções, geração de código etc.).

Não abordaremos muitos modelos, pois esse é um cenário em constante mudança, mas é importante observar que, embora as arquiteturas subjacentes possam diferir drasticamente, os modelos básicos geralmente se enquadram em duas categorias: proprietários (como o GPT-3.5 e Gemini) e de código aberto (como o Llama2-70B e DBRX). A principal diferença entre os dois é que, embora os modelos proprietários historicamente tenham uma vantagem em desempenho bruto, os usuários precisam enviar seus dados para um terceiro e não têm controle sobre o modelo subjacente, pois ele é frequentemente atualizado e modificado.

Os modelos de código aberto, por outro lado, oferecem aos usuários controle total sobre o modelo e a capacidade de executá-lo em seus próprios termos, com sua própria governança e privacidade de dados. Aqui está uma [lista atual de diversos modelos de IA Generativa de código aberto](#) em diferentes domínios, todos gratuitos para uso comercial. A Databricks também criou seu próprio modelo de base de código aberto de última geração para que os usuários possam desenvolver aplicações de IA Generativa de produção de alta qualidade.

Caso de uso do modelo básico

APRESENTAÇÃO DO DBRX: UM NOVO LLM ABERTO DE ÚLTIMA GERAÇÃO

Temos o prazer de apresentar o DBRX, um LLM aberto e de uso geral criado pela Databricks. Em diversos benchmarks padrão, o DBRX estabelece um novo patamar para os modelos abertos já consolidados. Além disso, ele fornece à comunidade aberta e às empresas que constroem seus próprios LLMs recursos que antes eram limitados a APIs de modelos fechados; de acordo com nossas medições, ele supera o GPT-3.5 e é competitivo com o Gemini 1.0 Pro. É um modelo de código especialmente capaz, superando modelos especializados como o CodeLLaMA-70b em programação, além de sua força como LLM de uso geral.

Essa qualidade de ponta vem com melhorias marcantes no desempenho de treinamento e inferência. O DBRX eleva o nível de eficiência entre os modelos abertos devido à sua arquitetura de mistura refinada de especialistas (MoE). A inferência é até duas vezes mais rápida que a do LLaMA2-70B, e o DBRX tem cerca de 40% do tamanho do Grok-1 em termos de contagem total e ativa de parâmetros. Quando hospedado no Mosaic AI Model Serving, o DBRX pode gerar texto em até 150 tok/s/usuário. Nossos clientes descobrirão que treinar MOEs também é cerca de duas vezes mais eficiente em FLOP do que treinar modelos densos para a mesma qualidade de modelo final. Nossa receita completa para o DBRX (desde os dados de pré-treinamento até a arquitetura do modelo e a estratégia de otimização) consegue alcançar a mesma qualidade dos nossos modelos MPT de geração anterior, com quase 4 vezes menos uso de computação.

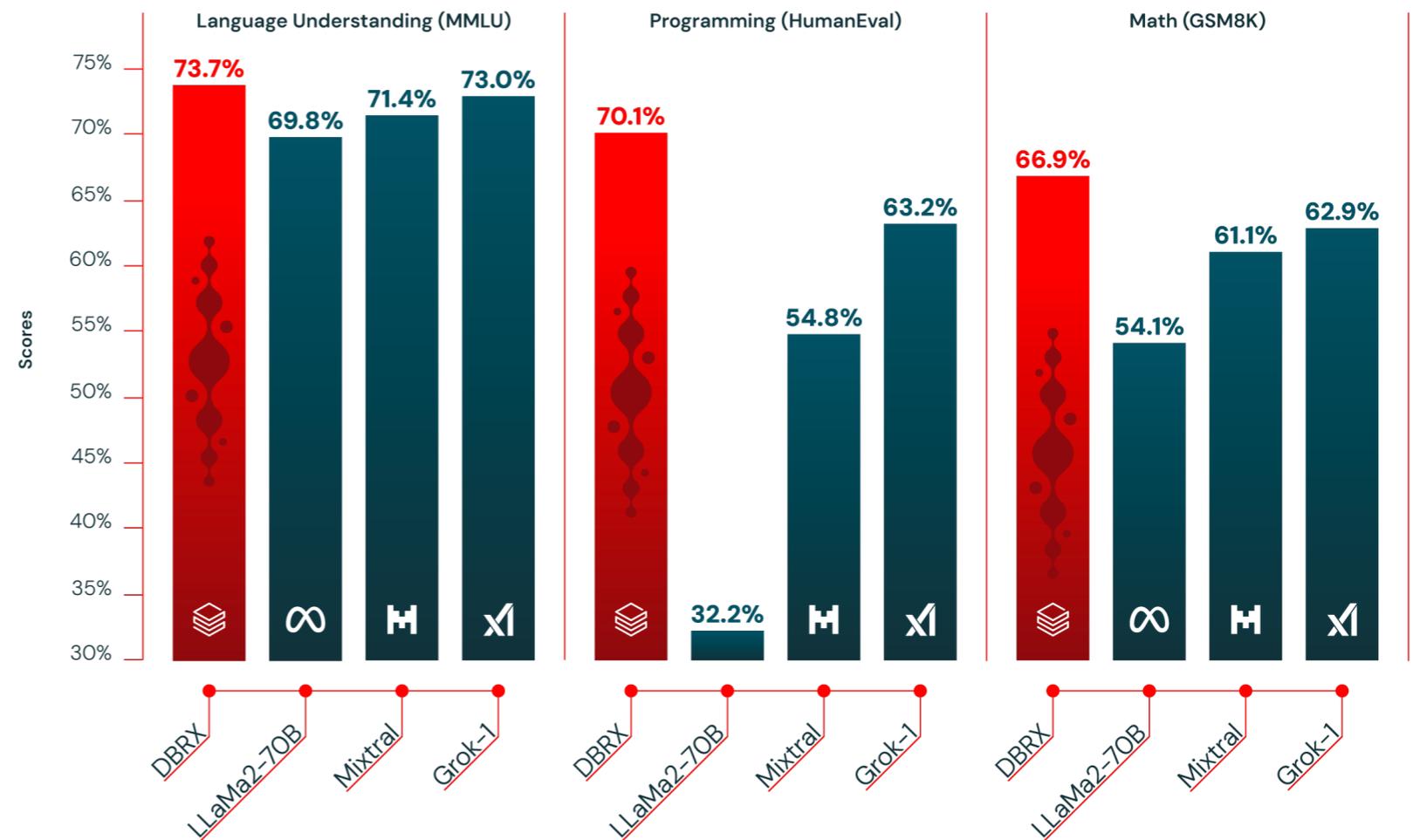


Figura 1: O DBRX supera os modelos de código aberto estabelecidos em compreensão de linguagem (MMLU), programação (HumanEval) e matemática (GSM8K).

Os pesos do modelo básico (**DBRX Base**) e do modelo com ajuste fino (**DBRX Instruct**) estão disponíveis no Hugging Face sob uma licença aberta. A partir de hoje, o DBRX está disponível para os clientes da Databricks usarem por meio de APIs, e os clientes da Databricks podem pré-treinar seus próprios modelos de classe DBRX a partir do zero ou continuar treinando em um de nossos pontos de verificação usando as mesmas ferramentas e ciência que usamos para criá-lo. O DBRX já está sendo integrado aos nossos produtos baseados em IA generativa, onde (em aplicações como SQL) os lançamentos iniciais ultrapassaram o GPT-3.5 Turbo e estão desafiando o GPT-4 Turbo. É também um modelo líder entre os modelos abertos e o GPT-3.5 Turbo em tarefas de RAG.

Treinar modelos mixture-of-experts é difícil. Tivemos que superar diferentes desafios científicos e de desempenho para construir um pipeline robusto o suficiente para treinar repetidamente modelos da classe DBRX de maneira eficiente. Com isso concluído, desenvolvemos uma stack de treinamento exclusiva que possibilita que qualquer empresa treine modelos de base MoE de alto nível do zero. Estamos ansiosos para compartilhar esse recurso com nossos clientes e compartilhar nossas lições aprendidas com a comunidade.

Baixe o DBRX hoje mesmo no Hugging Face (**DBRX Base**, **DBRX Instruct**), ou experimente o DBRX Instruct em nosso **HF Space**, ou veja nosso repositório de modelos no github: [databricks/dbrx](https://github.com/databricks/dbrx).

O que é DBRX?

DBRX é um **grande modelo de linguagem baseado em transformadores** que usa exclusivamente o decodificador e foi treinado para prever o próximo token. Ele usa uma arquitetura de mistura de especialistas (MoE) refinada, com 132 bilhões de parâmetros totais, dos quais 36 bilhões são ativados para qualquer entrada. Foi pré-treinado com 12 trilhões de tokens de texto e dados de código. Em comparação com outros modelos MoE abertos, como o Mixtral e o Grok-1, o DBRX é mais detalhado, o que significa que utiliza um número maior de especialistas menores. O DBRX possui 16 especialistas e escolhe 4, enquanto o Mixtral e o Grok-1 têm 8 especialistas e escolhem 2. Isso oferece 65 vezes mais combinações possíveis de especialistas, e descobrimos que isso melhora a qualidade do modelo. O DBRX usa codificações rotativas de posição (RoPE), unidades lineares com portas (GLU) e atenção agrupada por consulta (GQA). Ele utiliza o tokenizador GPT-4 fornecido no **repositório tiktoken**. Fizemos essas escolhas com base em avaliações exaustivas e experimentos de escalabilidade.

O DBRX foi pré-treinado com 12 trilhões de tokens de dados cuidadosamente selecionados e um comprimento máximo de contexto de 32 mil tokens. Estimamos que esses dados sejam pelo menos duas vezes melhores, token por token, do que os dados que usamos para pré-treinar a família de modelos MPT. Esse novo conjunto de dados foi desenvolvido usando o conjunto completo de ferramentas da Databricks, incluindo o Apache Spark™ e os notebooks da Databricks para processamento de dados, o [Unity Catalog](#) para gerenciamento e governança de dados e o MLflow para rastreamento de experimentos. Usamos o aprendizado curricular para o pré-treinamento, alterando a combinação de dados durante o treinamento de forma a melhorar substancialmente a qualidade do modelo.

Qualidade em Benchmarks vs. Principais Modelos Abertos

A Tabela 1 mostra a qualidade do DBRX Instruct e dos principais modelos abertos e estabelecidos. O DBRX Instruct é o modelo líder em benchmarks compostos, benchmarks de programação e matemática e MMLU. Ele supera todos os modelos com ajuste fino de bate-papo ou instruções em benchmarks padrão.

Benchmarks compostos. Avaliamos o DBRX Instruct e seus pares em dois benchmarks compostos: o [Hugging Face Open LLM Leaderboard](#) (a média de ARC-Challenge, HellaSwag, MMLU, TruthfulQA, WinoGrande e GSM8k) e o [Databricks Model Gauntlet](#) (um conjunto de mais de 30 tarefas abrangendo seis categorias: conhecimento do mundo, raciocínio de senso comum, compreensão da linguagem, compreensão de leitura, resolução de problemas simbólicos e programação).

Entre os modelos que avaliamos, o DBRX Instruct obteve a pontuação mais alta em dois benchmarks compostos: o Hugging Face Open LLM Leaderboard (74,5% versus 72,7% para o próximo modelo mais alto, Mixtral Instruct) e o Databricks Gauntlet (66,8% versus 60,7% para o próximo modelo mais alto, Mixtral Instruct).

Programação e matemática. O DBRX Instruct é especialmente forte em programação e matemática. Ele pontua mais alto do que os outros modelos abertos que avaliamos no HumanEval (70,1% versus 63,2% para Grok-1, 54,8% para Mixtral Instruct e 32,2% para a variante LLaMA2-70B de melhor desempenho) e GSM8k (66,9% vs. 62,9% para Grok-1, 61,1% para Mixtral Instruct e 54,1% para a variante LLaMA2-70B de melhor desempenho). O DBRX supera o Grok-1, o próximo melhor modelo nesses benchmarks, apesar do fato de que o Grok-1 tem 2,4x mais parâmetros. No HumanEval, o DBRX Instruct até supera o CodeLLaMA-70B Instruct, um modelo criado explicitamente para programação, apesar do fato de o DBRX Instruct ser projetado para uso geral (70,1% vs. 67,8% no HumanEval, conforme relatado pelo Meta no [blog CodeLLaMA](#)).

MMLU. O DBRX Instruct pontua mais alto do que todos os outros modelos que consideramos no MMLU, atingindo 73,7%.

MODELO	DBRX INSTRUCT	MIXTRAL INSTRUCT	MIXTRAL BASE	LLAMA2-70 B CHAT	LLAMA2-70 B BASE	GROK-11
Open LLM Leaderboard2 (média das próximas seis linhas)	<u>74,5%</u>	72,7%	68,4%	62,4%	67,9%	—
ARC-challenge 25-shot	68,9%	<u>70,1%</u>	66,4%	64,6%	67,3%	—
HellaSwag 10-shot	<u>89,0%</u>	87,6%	86,5%	85,9%	87,3%	—
MMLU 5-shot	<u>73,7%</u>	71,4%	71,9%	63,9%	69,8%	73,0%
Truthful QA 0-shot	<u>66,9%</u>	65,0%	46,8%	52,8%	44,9%	—
WinoGrande 5-shot	81,8%	81,1%	81,7%	80,5%	<u>83,7%</u>	—
GSM8k CoT 5-shot maj@13	<u>66,9%</u>	61,1%	57,6%	26,7%	54,1%	62,9% (8-shot)
Gauntlet v0.34 (média de mais de 30 tarefas diversas)	<u>66,8%</u>	60,7%	56,8%	52,8%	56,4%	—
HumanEval5 0-Shot, pass@1 (programação)	<u>70,1%</u>	54,8%	40,2%	32,2%	31,0%	63,2%

Tabela 1: qualidade do DBRX Instruct e dos principais modelos abertos. Consulte as notas de rodapé para obter detalhes sobre como os números foram coletados. Em negrito e sublinhado é a pontuação mais alta.

Qualidade em benchmarks vs.principais modelos fechados

A Tabela 2 mostra a qualidade do DBRX Instruct e dos principais modelos fechados. De acordo com as pontuações relatadas pelo criador de cada modelo, o DBRX Instruct supera o GPT-3.5 (conforme descrito no artigo GPT-4) e é competitivo com o Gemini 1.0 Pro e o Mistral Medium.

Em quase todos os benchmarks que consideramos, o DBRX Instruct supera ou (na pior das hipóteses) iguala o GPT-3.5. O DBRX Instruct supera o GPT-3.5 em conhecimento geral, conforme medido pelo MMLU (73,7% vs. 70,0%) e raciocínio de senso comum, conforme medido pelo HellaSwag (89,0% vs. 85,5%) e WinoGrande (81,8% vs. 81,6%). O DBRX Instruct se destaca especialmente em programação e raciocínio matemático, conforme medido pelo HumanEval (70,1% vs. 48,1%) e GSM8k (72,8% vs. 57,1%).

O DBRX Instruct é competitivo com o Gemini 1.0 Pro e o Mistral Medium. As pontuações do DBRX Instruct são maiores do que o Gemini 1.0 Pro no Inflection Corrected MTBench, MMLU, HellaSwag e HumanEval, enquanto o Gemini 1.0 Pro é mais forte no GSM8k. As pontuações para o DBRX Instruct e o Mistral Medium são semelhantes para HellaSwag, enquanto o Mistral Medium é mais forte em Winogrande e MMLU, e o DBRX Instruct é mais forte em HumanEval, GSM8k e Inflection Corrected MTBench.

MODELO	DBRX INSTRUCT	GPT-3.57	GPT-48	CLAUDE 3 HAIKU	CLAUDE 3 SONNET	CLAUDE 3 OPUS	GEMINI 1.0 PRO	GEMINI 1.5 PRO	MISTRAL MEDIUM	MISTRAL LARGE
MT Bench (inflexão corrigida, n=5)	8,39 ± 0,08	—	—	8,41 ± 0,04	8,54 ± 0,09	9,03 ± 0,06	8,23 ± 0,08	—	8,05 ± 0,12	8,90 ± 0,06
MMLU 5-shot	73,7%	70,0%	86,4%	75,2%	79,0%	86,8%	71,8%	81,9%	75,3%	81,2%
HellaSwag 10-shot	89,0%	85,5%	95,3%	85,9%	89,0%	95,4%	84,7%	92,5%	88,0%	89,2%
HumanEval 0-Shot pass@1 (programação)	70,1% temp=0, N=1	48,1%	67,0%	75,9%	73,0%	84,9%	67,7%	71,9%	38,4%	45,1%
GSM8k CoT maj@1	72,8% (5-shot)	57,1% (5-shot)	92,0% (5-shot)	88,9%	92,3%	95,0%	86,5% (maj1@32)	91,7% (11-shot)	66,7% (5-shot)	81,0% (5-shot)
WinoGrande 5-shot	81,8%	81,6%	87,5%	—	—	—	—	—	88,0%	86,7%

Tabela 2: qualidade do DBRX Instruct e dos principais modelos fechados. Com exceção do Inflection Corrected MTBench (que nós mesmos medimos nos endpoints do modelo), os números foram informados pelos criadores desses modelos em seus respectivos artigos técnico. Consulte as notas de rodapé para obter mais detalhes.

Qualidade em tarefas de contexto longo e RAG

O DBRX Instruct foi treinado com uma janela de contexto de até 32 mil tokens. A Tabela 3 compara seu desempenho ao do Mixtral Instruct e das versões mais recentes das APIs GPT-3.5 Turbo e GPT-4 Turbo em uma série de benchmarks de longos contextos (pares KV do artigo [Lost in the Middle](#) e HotpotQAXL, uma versão modificada do HotPotQA que estende a tarefa para sequências mais longas). O GPT-4 Turbo é geralmente o melhor modelo nessas tarefas. No entanto, com uma única exceção, o DBRX Instruct tem melhor desempenho que o GPT-3.5 Turbo em todos os comprimentos de contexto e partes da sequência. O desempenho geral do DBRX Instruct e do Mixtral Instruct é semelhante.

MODELO	DBRX INSTRUCT	MIXTRAL INSTRUCT	GPT-3.5 TURBO (API)	GPT-4 TURBO (API)
Resposta no terço inicial do contexto	<u>45,1%</u>	41,3%	37,3%*	49,3%
Resposta no terço médio do contexto	<u>45,3%</u>	42,7%	37,3%*	49,0%
Resposta no último terço do contexto	<u>48,0%</u>	44,4%	37,0%*	50,9%
Contexto de 2K	59,1%	<u>64,6%</u>	36,3%	69,3%
Contexto de 4K	65,1%	59,9%	35,9%	63,5%
Contexto de 8K	<u>59,5%</u>	55,3%	45,0%	61,5%
Contexto de 16K	27,0%	20,1%	31,7%	26,0%
Contexto de 32K	<u>19,9%</u>	14,0%	—	28,5%

Tabela 3: o desempenho médio dos modelos nos benchmarks KV-Pairs e HotpotQAXL. O negrito indica a pontuação mais alta. O sublinhado mostra a maior pontuação, exceto GPT-4 Turbo. O GPT-3.5 Turbo tem um suporte de comprimento máximo de contexto de 16 mil tokens, por isso não podemos avaliá-lo com 32 mil. *As médias para o início, o meio e o fim da sequência do GPT-3.5 Turbo incluem apenas contextos de até 16 mil.

Uma das formas mais populares de aproveitar o contexto de um modelo é a geração aumentada de recuperação (RAG). Na RAG, o conteúdo relevante para um prompt é recuperado de um banco de dados e apresentado junto com o prompt para fornecer ao modelo mais informações do que ele teria de outra forma. A Tabela 4 mostra a qualidade do DBRX em dois benchmarks de RAG (Natural Questions e HotPotQA), quando o modelo também recebe as 10 principais passagens recuperadas de um corpus de artigos da Wikipédia usando o modelo de incorporação bge-large-en-v1.5. O DBRX Instruct é competitivo com modelos abertos como o Mixtral Instruct e o LLaMA2-70B Chat e a versão atual do GPT-3.5 Turbo.

MODELO	DBRX INSTRUCT	MIXTRAL INSTRUCT	LLAMA2-70B CHAT	GPT 3.5 TURBO (API)	GPT 4 TURBO (API)
Perguntas naturais	<u>60,0%</u>	59,1%	56,5%	57,7%	63,9%
HotPotQA	<u>55,0%</u>	54,2%	54,7%	53,0%	62,9%

Tabela 4: o desempenho dos modelos é medido quando cada modelo recebe as 10 principais passagens recuperadas de um corpus da Wikipedia usando bge-large-en-v1.5. A precisão é medida pela correspondência na resposta do modelo. Negrito é a pontuação mais alta. Sublinhado é a pontuação mais alta exceto a do GPT-4 Turbo.

Eficiência do treinamento

A qualidade do modelo deve ser colocada no contexto da eficiência do treinamento e do uso do modelo. Isso é especialmente verdadeiro na Databricks, onde construímos modelos como o DBRX para estabelecer um processo para nossos clientes treinarem seus próprios modelos básicos.

Descobrimos que os modelos mixture-of-experts de treinamento fornecem melhorias substanciais na eficiência de computação para treinamento (Tabela 5). Por exemplo, treinar um membro menor da família DBRX chamado DBRX MoE-B (total de 23,5 bilhões de parâmetros, 6,6 bilhões de parâmetros ativos) exigiu 1,7 vezes menos FLOPs para atingir uma pontuação de 45,5% no Databricks LLM Gauntlet do que o LLaMA2-13B exigiu para atingir 43,8%. O DBRX MoE-B também contém metade dos parâmetros ativos do LLaMA2-13B.

Analisando de forma abrangente, nosso pipeline de pré-treinamento de LLM de ponta a ponta tornou-se quase quatro vezes mais eficiente em termos de computação nos últimos dez meses. Em 5 de maio de 2023, lançamos o **MPT-7B**, um modelo de 7B de parâmetros treinado em 1T tokens que atingiu uma pontuação de 30,9% no Databricks LLM Gauntlet. Um membro da família DBRX chamado DBRX MoE-A (7,7 bilhões de parâmetros totais, 2,2 bilhões de parâmetros ativos) alcançou uma pontuação no Databricks Gauntlet de 30,5% com 3,7 vezes menos FLOPs. Essa eficiência é o resultado de várias melhorias, incluindo o uso de uma arquitetura MoE, outras alterações na arquitetura da rede, melhores estratégias de otimização, melhor tokenização e, o que é muito importante, melhores dados de pré-treinamento.

Isoladamente, melhores dados de pré-treinamento tiveram um impacto substancial na qualidade do modelo. Treinamos um modelo de 7 bilhões de parâmetros em 1 trilhão de tokens (chamado DBRX Dense-A) utilizando os dados de pré-treinamento do DBRX. Ele atingiu 39,0% no Databricks Gauntlet em comparação com 30,9% no MPT-7B. Estimamos que nossos novos dados de pré-treinamento são pelo menos duas vezes melhores, token por token, do que os dados usados para treinar o MPT-7B. Em outras palavras, estimamos que metade dos tokens são necessários para alcançar a mesma qualidade de modelo. Determinamos isso treinando DBRX Dense-A em 500B de tokens; superou o MPT-7B no Databricks Gauntlet, atingindo 32,1%. Além da melhor qualidade dos dados, outro fator importante para essa eficiência no uso de tokens pode ser o tokenizador do GPT-4, que possui um vocabulário amplo e é considerado especialmente eficiente no uso de tokens. Essas lições sobre como melhorar a qualidade dos dados se traduzem diretamente em práticas e ferramentas que nossos clientes usam para treinar modelos básicos em seus próprios dados.

MODELO	TOTAL DE PARÂMETROS	PARÂMETROS ATIVOS	PONTUAÇÃO DO GAUNTLET	FLOPS RELATIVOS
DBRX MoE-A	7,7B	2,2B	30,5%	1x
MPT-7B (1T tokens)	—	6,7B	30,9%	3,7x
DBRX Dense-A (1T tokens)	—	6,7B	39,0%	3,7x
DBRX Dense-A (500B tokens)	—	6,7B	32,1%	1,85x
DBRX MoE-B	23,5B	6,6B	45,5%	1x
LLaMA2-13B	—	13,0B	43,8%	1,7x

Tabela 5: detalhes de vários artigos de testes que usamos para validar a eficiência do treinamento da arquitetura DBRX MoE e do pipeline de treinamento de ponta a ponta.

Eficiência da inferência

A Figura 2 ilustra a eficiência de inferência completa do DBRX e modelos similares ao serem executados com o NVIDIA TensorRT-LLM, utilizando nossa infraestrutura otimizada e precisão de 16 bits. Nosso objetivo é que esse benchmark reflita o uso no mundo real da forma mais próxima possível, incluindo vários usuários acessando simultaneamente o mesmo servidor de inferência. Geramos um novo usuário por segundo, cada solicitação de usuário contém um prompt de aproximadamente 2.000 tokens, e cada resposta compreende 256 tokens.

Em geral, os modelos MoE são mais rápidos na inferência do que a contagem total de parâmetros sugere. Isso se deve ao fato de que eles usam relativamente poucos parâmetros para cada entrada. Descobrimos que o DBRX não é exceção a esse respeito. O rendimento de inferência do DBRX é duas a três vezes maior que o de um modelo 132B não MoE.

A eficiência da inferência e a qualidade do modelo geralmente estão em tensão: modelos maiores geralmente alcançam maior qualidade, mas modelos menores são mais eficientes para inferência. O uso de uma arquitetura MoE possibilita obter melhores compensações entre a qualidade do modelo e a eficiência da inferência do que os modelos densos normalmente alcançam. O DBRX, por exemplo, oferece uma qualidade superior ao LLaMA2-70B e, por contar com cerca de metade dos parâmetros ativos, sua taxa de inferência é até 2 vezes mais rápida (Figura 2). O Mixtral é outro ponto na fronteira de Pareto aprimorada alcançada pelos modelos MoE: é menor que o DBRX e, conseqüentemente, é inferior em termos de qualidade, mas alcança maior throughput de inferência. Os usuários das APIs do Databricks Foundation Model podem esperar ver até 150 tokens por segundo para o DBRX em nossa plataforma otimizada de serviço de modelos com quantização de 8 bits.

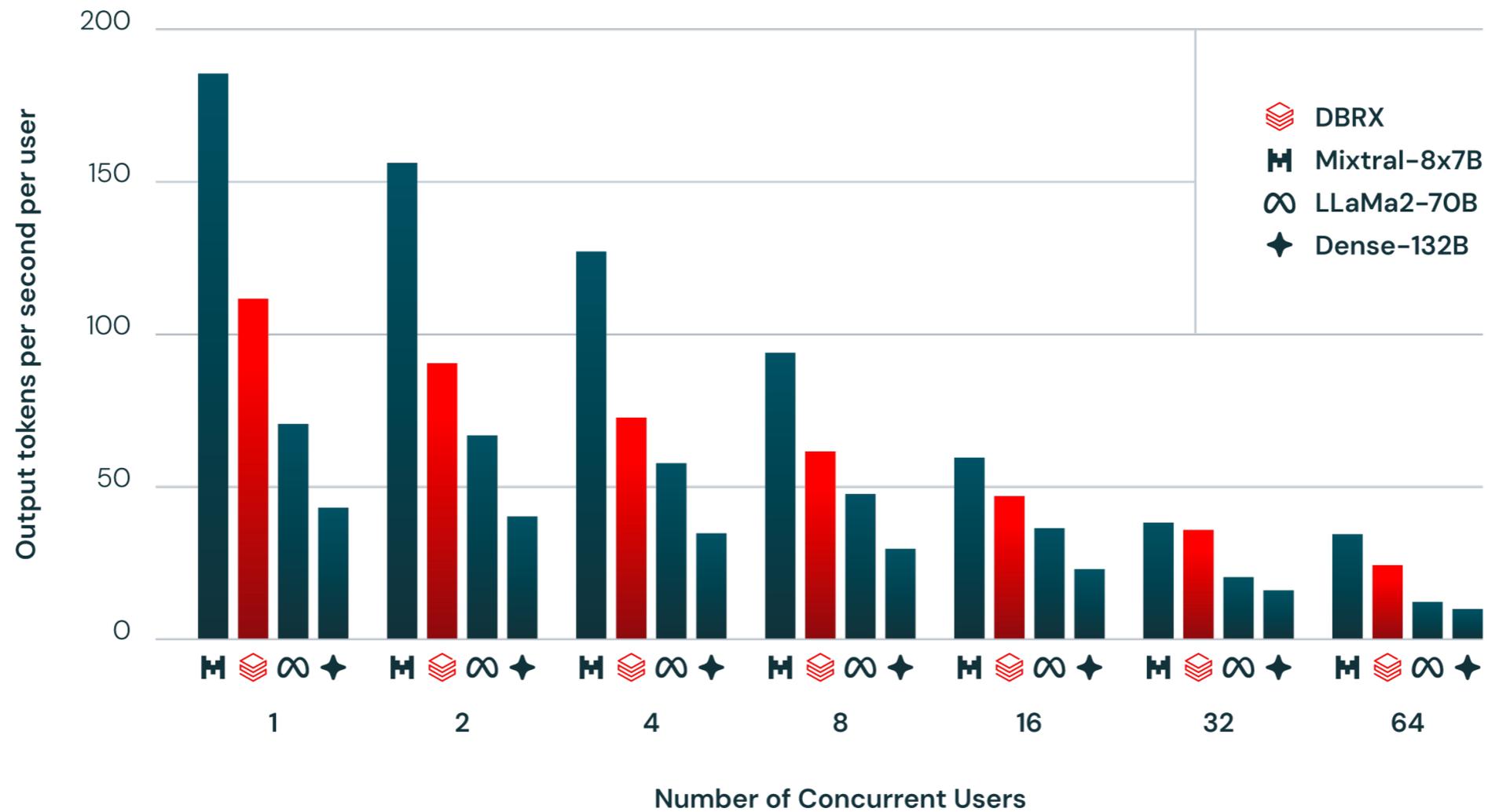


Figura 2: throughput de inferência para várias configurações de modelo em nossa infraestrutura de serviço otimizada usando o NVIDIA TensorRT-LLM com precisão de 16 bits e os melhores sinalizadores de otimização que podemos encontrar. Os modelos são executados em tensor-paralelo em todo o nó. O prompt de entrada contém aproximadamente 2.000 tokens de prompt, e geramos 256 tokens de saída. Um novo usuário surge a cada segundo.

Como criamos o DBRX

O DBRX foi treinado em 3072 NVIDIA H100s conectadas por 3,2 Tbps de Infiniband. O principal processo de construção do DBRX, incluindo pré-treinamento, pós-treinamento, avaliação, red-teaming e refinamento, ocorreu ao longo de três meses. Foi a continuação de meses de pesquisa científica, desenvolvimento de datasets e experimentos de escalabilidade, sem mencionar anos de desenvolvimento de LLM na Databricks, que inclui os projetos [MPT](#) e [Dolly](#), além dos milhares de modelos que construímos e levamos à produção com nossos clientes.

Para criar o DBRX, aproveitamos o mesmo conjunto de ferramentas da Databricks que estão disponíveis para nossos clientes. Gerenciamos e controlamos nossos dados de treinamento usando o Unity Catalog. Exploramos esses dados usando o recém-adquirido [Lilac AI](#). Processamos e limpamos esses dados usando notebooks Apache Spark™ e Databricks. Treinamos o DBRX usando versões otimizadas de nossas bibliotecas de treinamento de código aberto: [MegaBlocks](#), [LLM Foundry](#), [Composer](#) e [Streaming](#). Gerenciamos o treinamento e o ajuste fino de modelos em larga escala em milhares de GPUs usando nosso serviço de treinamento Mosaic AI. Registramos nossos resultados usando o [MLflow](#). Coletamos feedback humano para melhorias de qualidade e segurança por meio do Mosaic AI Model Serving and Inference Tables. Experimentamos manualmente o modelo usando o Databricks Playground. Descobrimos que as ferramentas da Databricks são as melhores da categoria para cada uma de suas finalidades e nos beneficiamos do fato de que todas elas faziam parte de uma experiência de produto unificada.

Comece com o DBRX na Databricks

Se você deseja começar a trabalhar com o DBRX imediatamente, é fácil fazer isso com as Databricks Mosaic AI [Foundation Model APIs](#). Você pode começar rapidamente com nossa precificação de pagamento conforme o uso e consultar o modelo na interface de chat do [AI Playground](#). Para aplicações em produção, oferecemos uma opção de throughput provisionado para garantir desempenho, suporte a modelos ajustados e segurança e conformidade adicionais. Para hospedar o DBRX de forma privada, você pode baixar o modelo do [Databricks Marketplace](#) e [implementar o modelo no Model Serving](#).

Conclusões

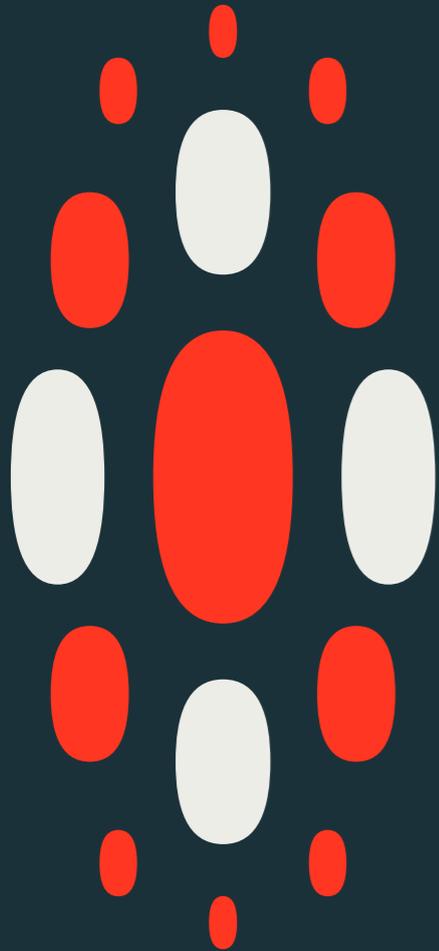
Na Databricks, acreditamos que toda empresa deve ter a capacidade de controlar seus dados e seu destino no mundo emergente da IA generativa. O DBRX é um pilar central da nossa próxima geração de produtos de IA generativa, e estamos ansiosos pela jornada emocionante que aguarda nossos clientes enquanto aproveitam os recursos do DBRX e as ferramentas que usamos para construí-lo. No ano passado, treinamos milhares de LLMs com nossos clientes. O DBRX é apenas um exemplo dos modelos poderosos e eficientes que estão sendo construídos na Databricks para uma ampla gama de aplicações, desde recursos internos até casos de uso ambiciosos para nossos clientes.

Como acontece com qualquer novo modelo, a jornada com o DBRX é apenas o começo, e o melhor trabalho será feito por aqueles que o desenvolverem: empresas e a comunidade aberta. Este também é apenas o começo do nosso trabalho no DBRX, e você deve esperar muito mais por vir.

Contribuições

O desenvolvimento do DBRX foi liderado pela equipe do **Mosaic**, que criou anteriormente a família de modelos MPT, em colaboração com dezenas de engenheiros, advogados, especialistas em compras e finanças, gerentes de programas, profissionais de marketing, designers e outros colaboradores de toda a Databricks. Somos gratos aos nossos colegas, amigos, familiares e à comunidade por sua paciência e apoio nos últimos meses.

Ao criar o DBRX, nos apoiamos nos gigantes da comunidade aberta e acadêmica. Ao disponibilizar o DBRX de forma aberta, esperamos retribuir à comunidade, na expectativa de que juntos possamos desenvolver tecnologias ainda mais avançadas no futuro. Com isso em mente, reconhecemos com gratidão o trabalho e a colaboração de **Trevor Gale** e seu projeto **MegaBlocks** (o orientador de doutorado do Trevor é o CTO da Databricks, Matei Zaharia), a equipe **PyTorch** e o projeto **FSDP, NVIDIA** e o projeto **TensorRT-LLM**, a equipe e o projeto **vLLM, EleutherAI** e seu projeto de avaliação **LLM**, Daniel Smilkov e Nikhil Thorat na **Lilac AI** e todos os nossos amigos no **Allen Institute for Artificial Intelligence (AI2)**.



Estágio 1: Engenharia de Prompts

Muitas empresas ainda permanecem nos estágios fundamentais da adoção da tecnologia de IA generativa. Eles não têm nenhuma estratégia abrangente de IA implementada, nenhum caso de uso claro a ser seguido e nenhum acesso a uma equipe de cientistas de dados e outros profissionais que possam ajudar a orientar a jornada de adoção da IA pela empresa.

Se sua empresa também é assim, um bom ponto de partida é um LLM pronto para uso. Embora esses LLMs não tenham a experiência específica de domínio dos modelos de IA personalizados, a experimentação pode ajudá-lo a traçar seus próximos passos. Seus funcionários podem criar **prompts e fluxos de trabalho** especializados para orientar seu uso. Seus líderes podem entender melhor os pontos fortes e fracos dessas ferramentas, bem como uma visão mais clara de como pode ser o sucesso inicial na IA. Sua organização pode usar coisas como o **Databricks AI Playground** para descobrir onde investir em ferramentas e sistemas de IA mais poderosos, que geram ganhos operacionais mais significativos, e até mesmo usar **LLMs como juiz** para ajudar a avaliar as respostas.

APLICAÇÕES PRÁTICAS DA TECNOLOGIA DE IA GENERATIVA

Vamos nos aprofundar em um caso de uso convincente que ilustra o poder da engenharia de prompts com LLMs prontos para uso. Considere o desafio que muitas empresas enfrentam: examinar grandes quantidades de avaliações de produtos para obter insights práticos. Sem uma equipe dedicada de cientistas de dados ou uma estratégia clara de IA, essa tarefa pode parecer assustadora. No entanto, aproveitar a flexibilidade dos LLMs por meio da engenharia de prompts oferece uma solução simples.

Caso de uso de engenharia de prompts

Análise automatizada de avaliações de produtos usando grandes modelos de linguagem

Acompanhe o feedback do cliente em escala

Confira nossos [Aceleradores de soluções de LLM para varejo](#) para obter mais detalhes e baixar os notebooks.

Embora a IA conversacional tenha atraído muita atenção da mídia nos últimos meses, os recursos dos grandes modelos de linguagem (LLMs) vão muito além das interações conversacionais. É nesses recursos menos proeminentes, como resposta a consultas, resumo, classificação e pesquisa que muitas organizações estão encontrando oportunidades imediatas para turbinar sua força de trabalho e melhorar as experiências dos clientes.

O potencial dessas aplicações é impressionante. Segundo uma [estimativa](#), os LLMs (e outras tecnologias de IA generativa) poderiam, em um futuro próximo, lidar com tarefas que hoje ocupam de 60% a 70% do tempo dos funcionários. Por meio do aumento, [vários estudos](#) mostraram que o tempo para concluir várias tarefas realizadas por trabalhadores do conhecimento, como pesquisa de fundo, análise de dados e redação de documentos, pode ser reduzido pela metade. E [outros estudos](#) ainda mostraram que o uso dessas tecnologias pode reduzir drasticamente o tempo para novos trabalhadores atingirem a produtividade total.

Mas antes que esses benefícios possam ser totalmente obtidos, as organizações devem primeiro [repensar](#) o gerenciamento dos ativos de informações não estruturados dos quais esses modelos dependem e encontrar maneiras de mitigar os problemas de viés e precisão que afetam seu output. É por isso que tantas organizações estão atualmente concentrando seus esforços em aplicativos internos focados, onde um escopo limitado oferece oportunidades para um melhor acesso à informação, e a supervisão humana pode servir como uma verificação para resultados errôneos. Esses aplicativos, alinhados com os principais recursos que já residem na organização, têm o potencial de agregar valor real e imediato, enquanto os LLMs e as tecnologias que os apoiam continuam a evoluir e amadurecer.

O RESUMO DE AVALIAÇÕES DE PRODUTOS PODERIA SER MELHORADO

Para ilustrar o potencial de uma abordagem mais focada na adoção de LLMs, consideramos uma tarefa bastante simples e comum realizada em muitas organizações de varejo on-line: o resumo de avaliações de produtos. Atualmente, a maioria das organizações emprega uma equipe de funcionários de tamanho modesto para ler e digerir o feedback dos usuários em busca de insights que podem ajudar a melhorar o desempenho de um produto ou identificar problemas relacionados à satisfação do cliente.

O trabalho é importante, mas está longe de ser atraente. Um trabalhador lê uma avaliação, faz anotações e passa para a próxima. As avaliações individuais que exigem uma resposta são sinalizadas, e um resumo do feedback de várias avaliações é compilado para análise pelos gerentes de produtos ou de categorias.

Esse é um tipo de trabalho que está pronto para a automação. O volume de avaliações que chegam a um site significa que as partes mais detalhadas desse trabalho geralmente são realizadas em um subconjunto limitado de produtos em janelas variáveis, dependendo da importância do produto. Em organizações mais sofisticadas, regras que detectam o curso ou linguagem imprópria e modelos que estimam o sentimento do usuário ou classificam as avaliações com base em experiências positivas, negativas ou neutras podem ser aplicadas para ajudar a identificar o conteúdo problemático e chamar a atenção do revisor para ele. Mas, de qualquer forma, muita coisa se perde simplesmente porque não conseguimos alocar pessoas suficientes para resolver o problema, e essas pessoas tendem a ficar entediadas ou cansadas com a monotonia do trabalho.

GRANDES MODELOS DE LINGUAGEM PODEM AUTOMATIZAR A ANÁLISE DE REVISÕES DE PRODUTOS

Ao usar um LLM, questões de escala e consistência podem ser facilmente resolvidas. Tudo o que precisamos fazer é levar as avaliações de produtos ao modelo e perguntar:

- Quais são os três principais pontos de feedback negativo encontrados nessas avaliações?
- Quais recursos nossos clientes mais gostam nesse produto?
- Os clientes sentem que estão recebendo valor suficiente do produto em relação ao que estão sendo solicitados a pagar?
- Existe alguma avaliação especialmente negativa ou que esteja usando linguagem imprópria?

Em segundos, podemos ter uma resposta organizada, permitindo que nossos gerentes de produtos se concentrem em responder aos problemas em vez de simplesmente detectá-los.

Mas e quanto ao problema de precisão e vieses? Os padrões para identificar imprecisões e vieses nos resultados do LLM estão evoluindo, assim como as técnicas para garantir que os resultados estejam alinhados com as expectativas da organização, e o ajuste fino dos modelos usando conteúdo aprovado pode ajudar muito a garantir que os modelos tenham preferência por gerar conteúdo que esteja, pelo menos, alinhado com a forma como a organização prefere se comunicar.

Essa é uma maneira prolixa de dizer que ainda não existe uma solução ideal para o problema. Mas, quando comparado com onde estamos em relação a processos conduzidos por humanos e modelos mais simplistas ou abordagens baseadas em regras, espera-se que os resultados sejam melhores ou, no mínimo, não piores do que os que experimentamos atualmente. E, dado que esses resumos de revisões são para consumo interno, o impacto de um modelo errôneo poder ser facilmente gerenciado.

VOCÊ PODE CRIAR UMA SOLUÇÃO PARA ISSO HOJE

Para demonstrar exatamente como esse trabalho pode ser executado, criamos um [acelerador de soluções](#) para resumir as análises de produtos. Isso se baseia fortemente em um [blog publicado anteriormente](#) de Sean Owen, que abordou alguns dos principais desafios técnicos do ajuste de um LLM na plataforma Databricks. Para o acelerador, estamos usando o [Amazon Product Reviews Dataset](#), que contém 51 milhões de avaliações geradas por usuários sobre dois milhões de livros distintos, pois fornece acesso a uma ampla variedade de conteúdo de revisores e apresenta um desafio de dimensionamento que muitas organizações reconhecerão.

Imaginamos um cenário no qual uma equipe de gerentes de produtos receba feedback de clientes por meio de avaliações on-line. Essas avaliações são importantes para identificar problemas que talvez precisem ser resolvidos em relação a um item específico e para direcionar os futuros livros a serem oferecidos pelo site. Sem o uso da tecnologia, essa equipe tem dificuldade de ler todo o feedback e resumi-lo em um conjunto de notas viável. Como resultado, eles limitam sua atenção apenas aos itens mais críticos e só consegue processar o feedback de forma esporádica.

Mas usando a Databricks, eles são capazes de configurar um pipeline para coletar feedback de uma variedade maior de produtos e resumi-los regularmente. Reconhecendo que os produtos com classificação positiva provavelmente destacam os pontos fortes desses livros, enquanto os produtos com classificação mais baixa provavelmente se concentram em seus pontos fracos, eles separam essas avaliações com base nas classificações fornecidas pelos usuários e encarregam um LLM de extrair diferentes conjuntos de informações de cada categoria de alto nível de avaliações.

As métricas resumidas são fornecidas para permitir aos gerentes de produtos uma visão geral do feedback recebido e são corroboradas por resumos mais detalhados gerados pelo LLM (Figura 1).



Figura 1: Métricas resumidas e detalhes em tópicos extraídos de avaliações de usuários utilizando um LLM

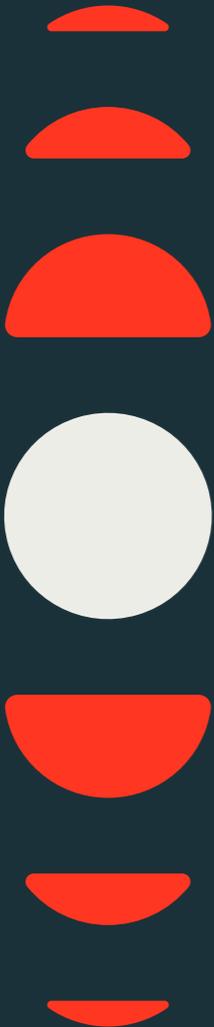
A DATABRICKS AGRUPA TODOS OS COMPONENTES DE UMA SOLUÇÃO

O cenário demonstrado acima depende do uso de um LLM. Nos meses anteriores, o uso desse LLM exigia acesso a infraestruturas computacionais especializadas, mas com os avanços na comunidade de código aberto e os investimentos na plataforma Databricks, agora podemos executar o LLM em nosso ambiente local da Databricks.

Nesse cenário específico, a confidencialidade dos dados não foi um fator motivador para essa escolha. Em vez disso, descobrimos que o volume de avaliações a serem processadas inclinou as escalas de custo para o uso da Databricks, o que nos permitiu reduzir cerca de um terço do custo da implementação de uma solução similar usando um serviço terceirizado.

Além disso, descobrimos que, ao implementar nossa própria infraestrutura, conseguimos expandir o ambiente para um processamento mais rápido, realizando até 760.000 avaliações por hora em um teste sem precisar nos preocupar com as restrições impostas por um serviço externo. Embora a maioria das organizações não tenha a necessidade de escalar até esse nível, é bom saber que ele está disponível, caso seja necessário.

No entanto, essa solução vai além de um simples LLM. Para agrupar toda a solução, precisávamos desenvolver um fluxo de trabalho de processamento de dados para receber as avaliações que chegam, prepará-las para envio ao modelo e capturar a saída do modelo para análise posterior. Como uma plataforma de dados unificada, a Databricks nos fornece os meios para lidar com requisitos de engenharia de dados e ciência de dados sem replicação de dados. E quando terminarmos de processar as avaliações, nossos analistas podem usar suas ferramentas preferidas para consultar o resultado e tomar decisões de negócios. Por meio da Databricks, temos acesso a toda a gama de recursos para construirmos uma solução alinhada às necessidades de nossa empresa.



Estágio 2: Geração Aumentada de Recuperação

A geração aumentada de recuperação (RAG) permite que você traga recursos de conhecimento suplementar para tornar um sistema de IA pronto para uso mais inteligente. O RAG não mudará o comportamento subjacente do modelo, mas **melhorará a qualidade** e a precisão das respostas.

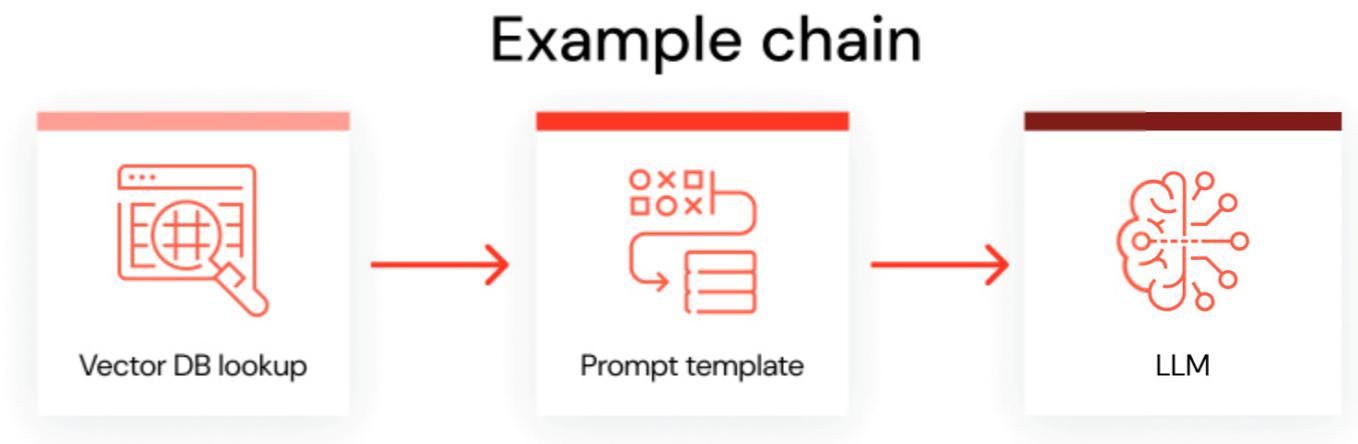
No entanto, neste ponto, sua empresa não deve fazer upload de seus dados "de missão crítica". Em vez disso, o processo RAG normalmente envolve quantidades menores de informações não confidenciais.

Por exemplo, a inclusão de um manual do funcionário pode permitir que seus trabalhadores comecem a fazer perguntas ao modelo subjacente sobre a política de férias da organização. O upload de manuais de instruções pode ajudar a impulsionar um chatbot de serviço. Com a capacidade de consultar tickets de suporte usando IA, os agentes de suporte podem obter respostas mais rapidamente; no entanto, inserir dados financeiros confidenciais para que os funcionários consultem o desempenho da empresa é provavelmente um passo longe demais.

Para começar, sua equipe deve primeiro consolidar e limpar os dados que pretende usar. Com o RAG, é fundamental que sua empresa armazene os dados em tamanhos apropriados para os modelos downstream. Muitas vezes, isso exige que os usuários os dividam em segmentos menores.

Portanto, você deve procurar uma ferramenta como o **Databricks Vector Search**, que permite que os usuários configurem rapidamente seu próprio banco de dados vetorial. E como ele é governado pelo Unity Catalog, é possível implementar controles granulares para garantir que os funcionários acessem apenas os conjuntos de dados para os quais têm credenciais.

Por fim, você pode conectar esse endpoint a um LLM. Uma ferramenta como o Databricks MLflow ajuda a centralizar o gerenciamento dessas APIs.



Entre os benefícios do **RAG** estão redução de alucinações, respostas mais atualizadas e precisas e melhor inteligência específica do domínio. Os modelos assistidos por RAG também são uma abordagem mais econômica para a maioria das organizações.

Embora o RAG ajude a melhorar os resultados dos modelos de negócios, ainda há muitas limitações para o uso do RAG. Se a sua empresa não está conseguindo os resultados desejados, é hora de partir para soluções mais robustas, mas ir além dos modelos suportados por RAG geralmente exige um comprometimento muito maior. A personalização adicional custa mais e exige muito mais dados.

É por isso que é fundamental que as organizações primeiro desenvolvam uma compreensão básica de como usar LLMs. Ao explorar ao máximo o desempenho dos modelos disponíveis antes de avançar, você e sua equipe de liderança podem identificar com mais precisão onde direcionar os recursos.

Aprimore o desempenho de modelos de IA prontos para uso com o RAG

Vamos explorar um caso de uso prático que demonstra como os dados estruturados em tempo real podem melhorar significativamente a qualidade da resposta de seus aplicativos RAG. Este exemplo mostrará como a integração de informações dinâmicas pode transformar a eficácia e a aplicabilidade da IA em suas operações de negócios.

Caso de uso do RAG

Melhore a qualidade de resposta de seu aplicativo RAG com dados estruturados em tempo real

por [Mani Parkhe](#), [Aakrati Talati](#), [Sue Ann Hong](#), [Craig Wiley](#), [Chenen Liang](#) e [Mingyang Ge](#)

A **geração aumentada de recuperação (RAG)** é um mecanismo eficiente para fornecer dados relevantes como contexto em aplicativos de IA generativa. A maioria dos aplicativos RAG normalmente usa índices de vetores para pesquisar contexto relevante de dados não estruturados, como documentação, wikis e tickets de suporte. Ontem, anunciamos o Databricks Vector Search Public Preview, que ajuda exatamente com isso. No entanto, a qualidade da resposta de IA generativa pode ser aprimorada aumentando esses contextos baseados em texto com dados estruturados relevantes e personalizados. Imagine uma ferramenta GenAI em um site de varejo onde os clientes perguntam: "Onde está meu pedido recente?" Essa IA deve entender que a consulta é sobre uma compra específica e, em seguida, reunir informações de remessa atualizadas para itens de linha, antes de usar LLMs para gerar uma resposta. O desenvolvimento desses aplicativos escaláveis exige um trabalho substancial, integrando tecnologias para lidar com dados estruturados e não estruturados com recursos de IA generativa.

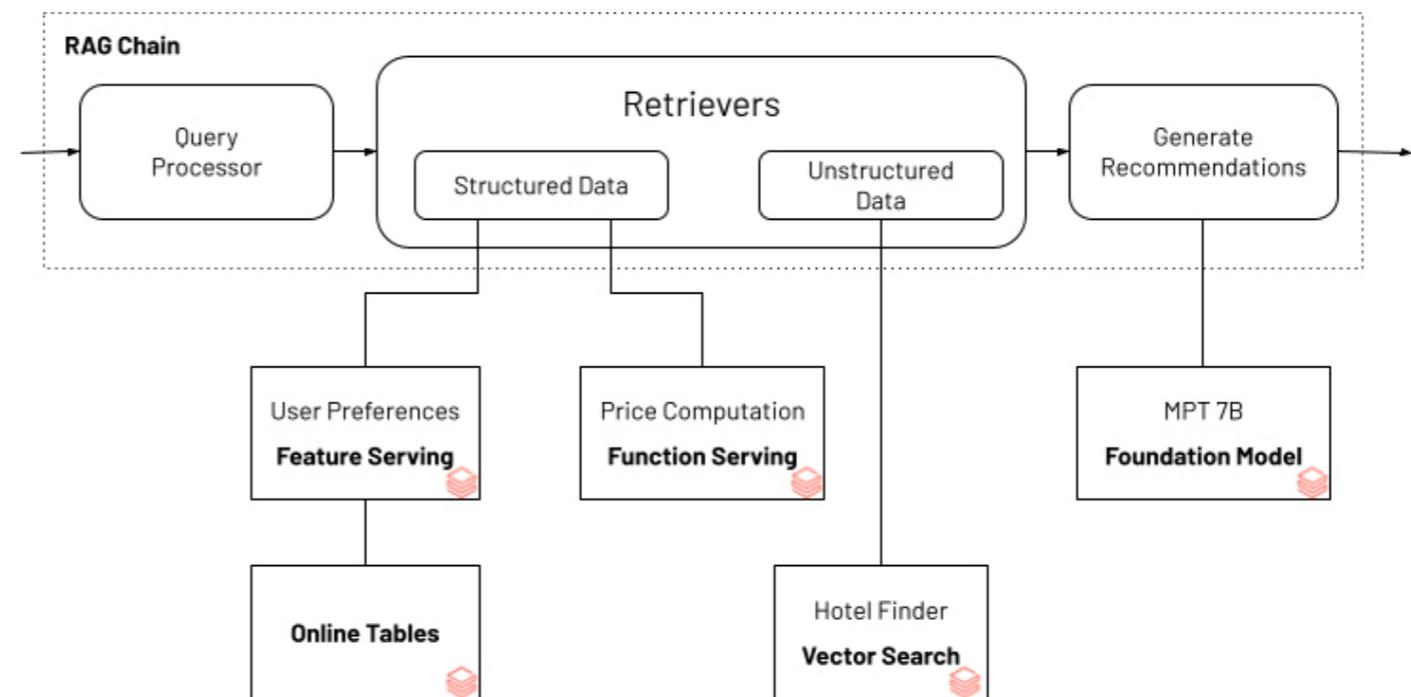
Temos o prazer de anunciar a prévia pública do Databricks Feature & Function Serving, um serviço em tempo real de baixa latência projetado para fornecer dados estruturados da Data Intelligence Platform da Databricks. Você pode acessar instantaneamente recursos de ML pré-computados, bem como realizar transformações de dados em tempo real, servindo qualquer função Python do Unity Catalog. Os dados recuperados podem ser usados em mecanismos de regras em tempo real, ML clássico e aplicativos de IA generativa.

O uso do Feature and Function Serving ([AWS](#)) ([Azure](#)) para dados estruturados em coordenação com o Databricks Vector Search ([AWS](#)) ([Azure](#)) para dados não estruturados simplifica significativamente a produção de aplicativos de IA generativa. Os usuários podem criar e implementar esses aplicativos diretamente na Databricks e contar com pipelines de dados existentes, governança e outros recursos corporativos. Os clientes da Databricks em vários setores estão usando essas tecnologias junto com estruturas de código aberto para criar aplicativos de IA generativa poderosos, como os descritos na tabela abaixo.

SETOR	CASO DE USO
Varejo	<ul style="list-style-type: none"> Recomendações de produtos/classificação de pesquisas usando as preferências do usuário, histórico de pesquisa, localização etc. Pesquisa de produtos baseada em imagens e metadados Gerenciamento e previsão de estoque usando dados de vendas, tendências sazonais e análise de mercado/da concorrência
Educação	<ul style="list-style-type: none"> Planos de aprendizado personalizados com base em erros passados, tendências históricas e coortes Classificação automatizada, feedback, acompanhamentos e relatórios de progresso Filtragem de conteúdo para dispositivos emitidos
Serviços financeiros	<ul style="list-style-type: none"> Aplicativos de linguagem natural para analistas e investidores correlacionarem chamadas e relatórios de ganhos com inteligência de mercado e tendências históricas Análise de fraudes e riscos Gerenciamento personalizado de patrimônio, planejamento de aposentadoria, análise hipotética e próximas melhores ações
Viagens e hospitalidade	<ul style="list-style-type: none"> Chatbots para interações personalizadas com clientes e recomendações de viagens personalizadas Planejamento dinâmico de rotas usando condições meteorológicas, padrões de tráfego ao vivo e dados históricos Otimização dinâmica de preços usando análise da concorrência e preços baseados na demanda
Cuidados de saúde e ciências biológicas	<ul style="list-style-type: none"> Envolvimento de pacientes/membros e resumos de saúde Aplicativos de suporte para atendimento personalizado, decisões clínicas e coordenação de cuidados Resumo de relatórios de P&D, análise de ensaios clínicos, reaproveitamento de medicamentos
Seguros	<ul style="list-style-type: none"> Avaliação de riscos para subscrição de hipotecas usando texto e dados estruturados sobre propriedades e bairros Use chatbots para perguntas sobre apólices, riscos e análises hipotéticas Automação do processamento de sinistros
Tecnologia e Manufatura	<ul style="list-style-type: none"> Manutenção prescritiva e diagnóstico de equipamentos usando instruções guiadas Deteção de anomalias em stream de dados ao vivo em comparação com estatísticas históricas Análise automatizada para produção diária/análise de turnos e planejamento futuro
Mídia e entretenimento	<ul style="list-style-type: none"> Descoberta e recomendações de conteúdo no aplicativo, e-mail personalizado e marketing digital Localização de conteúdo Experiências de jogo personalizadas e análises de jogos

FORNECIMENTO DE DADOS ESTRUTURADOS PARA APLICATIVOS RAG

Para demonstrar como os dados estruturados podem ajudar a melhorar a qualidade de um aplicativo de IA generativa, usamos o exemplo a seguir para um chatbot de planejamento de viagens. O exemplo mostra como as preferências do usuário (exemplo: “vista para o mar” ou “ideal para famílias”) podem ser combinadas com informações não estruturadas obtidas sobre hotéis para pesquisar correspondências de hotéis. Normalmente, os preços dos hotéis mudam dinamicamente com base na demanda e na sazonalidade. Uma calculadora de preços incorporada ao aplicativo de IA generativa garante que as recomendações estejam dentro do orçamento do usuário. O aplicativo de IA generativa que alimenta o bot usa o Databricks Vector Search e o Databricks Feature and Function Serving, que servem como blocos de construção para atender às preferências personalizadas do usuário e às informações orçamentárias e de hotéis necessárias usando a API de agentes da LangChain.



Você pode encontrar o [notebook completo](#) para esse aplicativo RAG Chain, conforme mostrado acima. Esse aplicativo pode ser executado localmente no notebook ou implementado como um endpoint acessível por uma interface de usuário de chatbot.

ACESSE SEUS DADOS E FUNÇÕES COMO ENDPOINTS EM TEMPO REAL

Com a **engenharia de recursos no Unity Catalog**, você já pode usar qualquer tabela com uma chave primária para fornecer recursos para treinamento e serviço. O Databricks Model Serving é compatível com o uso de **funções Python para computar recursos sob demanda**. Criados usando a mesma tecnologia disponível nos bastidores do Databricks Model Serving, os endpoints de recursos e funções podem ser usados para acessar qualquer recurso pré-computado ou computá-lo sob demanda. Com uma sintaxe simples, você pode definir uma função de especificação de recurso no Unity Catalog, que pode codificar o gráfico acíclico direcionado para calcular e servir recursos como um endpoint REST.

```
1 from databricks.feature_engineering import (
2     FeatureFunction,
3     FeatureLookup,
4     FeatureEngineeringClient,
5 )
6
7 features = [
8     # Procure as colunas "latitude" e "longitude" da tabela "restaurants" no UC usando a entrada "restaurant_id" como
9     # chave
10    FeatureLookup(
11        table_name="main.default.restaurants",
12        lookup_key="restaurant_id",
13        features=["latitude", "longitude"]
14    ),
15    # Calcule um novo recurso chamado "distance" usando o restaurante e a localização atual do usuário
16    FeatureFunction(
17        udf_name="main.default.distance",
18        output_name="distance",
19        # Vincule o parâmetro da função com a entrada de outros recursos ou da solicitação.
20        input_bindings={"user_latitude": "user_latitude", "user_longitude": "user_longitude",
21                        "restaurant_latitude": "latitude", "restaurant_longitude": "longitude"},
22    ),
23 ]
24
25 fe = FeatureEngineeringClient()
26
27 # Crie uma especificação de recurso com os recursos listados acima.
28 # O FeatureSpec pode ser acessado no UC como uma função.
29 fe.create_feature_spec(
30     name="main.default.restaurant_features",
31     features=features,
32 )
```

Essa função de especificação de recurso pode ser servida em tempo real como um endpoint REST. Todos os endpoints estão acessíveis na aba de navegação à esquerda do Serving, incluindo features, funções, modelos personalizados treinados e modelos fundamentais. Provisione o endpoint usando essa API.

```
1 from databricks.feature_engineering.entities.feature_serving_endpoint import (
2     ServedEntity,
3     EndpointCoreConfig,
4 )
5
6 fe.create_feature_serving_endpoint(
7     name="restaurant-features",
8     config=EndpointCoreConfig(
9         served_entities=ServedEntity(
10            feature_spec_name="main.default.restaurant_features",
11            workload_size="Small",
12            scale_to_zero_enabled=True
13        )
14    )
15 )
```

O endpoint também pode ser criado usando um fluxo de trabalho da interface do usuário, conforme mostrado no gráfico a seguir

The screenshot shows the Databricks Catalog Explorer interface. The left sidebar displays a catalog tree with 'feature_serving' selected. The main panel shows the details for 'restaurant_feature_spec', including its definition and function metadata.

Definition

```
input_columns:
- restaurant_id:
  topological_ordering: 0
  source: training_data
- user_latitude:
  topological_ordering: 1
  source: training_data
- user_longitude:
  topological_ordering: 2
  source: training_data
- latitude:
  table_name: main.feature_serving.restaurant_features
  feature_name: latitude
  lookup_key:
  ... 28 more lines
```

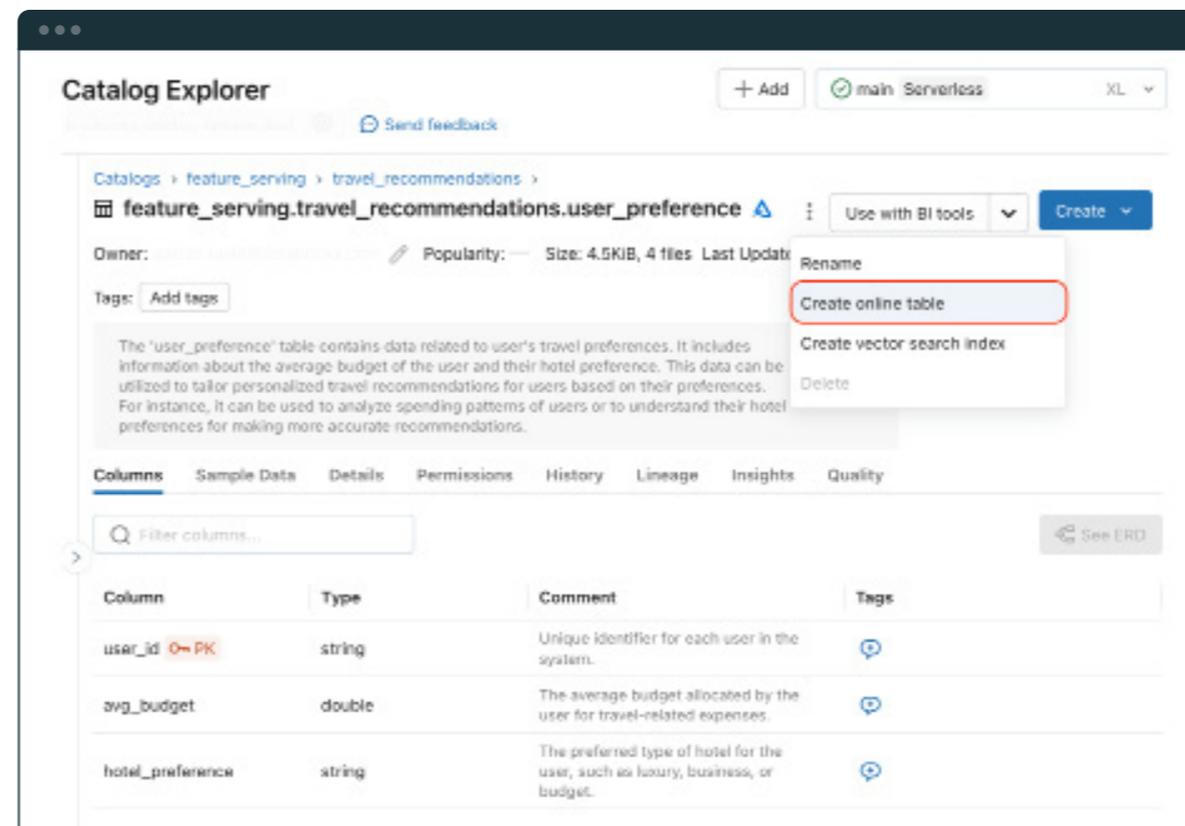
Function metadata

Parameters	restaurant_id: INT user_latitude: FLOAT user_longitude: FLOAT latitude: FLOAT DEFAULT AUTO longitude: FLOAT DEFAULT AUTO distance: FLOAT DEFAULT AUTO
Return type	STRUCT<restaurant_id:INT,user_latitude:FLOAT,user_longitude:FLOAT,latitude:FLOAT,longitude:FLOAT,distance:FLOAT>
Language	FeatureSpec

Agora, os recursos podem ser acessados em tempo real consultando o endpoint:

```
1 curl \  
2   -u token:$DATABRICKS_TOKEN \  
3   -X POST \  
4   -H "Content-Type: application/json" \  
5   -d '{"dataframe_records": [{"user_latitude": 37.9711, "user_longitude": -122.3940, "restaurant_id": 5}]}' \  
6   https://<databricks-instance>/serving-endpoints/restaurant-features/invocations
```

Para fornecer dados estruturados a aplicativos de IA em tempo real, os dados pré-computados precisam ser implementados em bancos de dados operacionais. Os usuários já podem usar repositórios on-line externos como uma fonte de recursos pré-computados, por exemplo, o **DynamoDB** e o **Cosmos DB** são comumente usados para fornecer recursos no Databricks Model Serving. O Databricks Online Tables (**AWS**)(**Azure**) adiciona uma nova funcionalidade que simplifica a sincronização de recursos pré-computados a um formato de dados otimizado para pesquisas de dados de baixa latência. Você pode sincronizar qualquer tabela com uma chave primária como uma tabela on-line, e o sistema configurará um pipeline automático para garantir a atualização dos dados.





Qualquer tabela do Unity Catalog com chaves primárias pode ser usada para fornecer recursos em aplicativos de IA generativa usando o Databricks Online Tables.

Estágio 3: Ajuste fino de um modelo básico

Superar o RAG e passar para o ajuste fino de modelos possibilita desenvolver soluções muito mais personalizadas para a empresa. Se você já tem experimentado modelos comerciais em suas operações, provavelmente estará pronto para avançar para esse estágio. Há uma compreensão clara no nível executivo do valor da IA generativa, bem como uma compreensão das limitações dos LLMs disponíveis publicamente. Casos de uso específicos foram estabelecidos. E agora, você e sua empresa estão prontos para ir mais fundo.

Com o ajuste fino, você pode pegar um modelo de uso geral e treiná-lo em seus próprios dados específicos. Por exemplo, o provedor de gerenciamento de dados **Stardog conta com as ferramentas Mosaic AI da Databricks** para realizar o ajuste fino dos LLMs prontos para uso que eles usam como base para sua plataforma Knowledge Graph. Isso permite que os clientes da Stardog consultem seus próprios dados nos diferentes silos simplesmente usando linguagem natural.

É fundamental que, neste estágio, as organizações contem com uma arquitetura subjacente que assegure a segurança e a precisão dos dados que suportam os modelos. O ajuste fino de um sistema de IA exige uma quantidade imensa de informações proprietárias e, à medida que sua empresa avança na curva de maturidade da IA, o número de modelos em execução só aumentará, aumentando a demanda por acesso aos dados.

É por isso que você precisa ter os mecanismos certos para rastrear os dados desde o momento em que são gerados até o momento em que são usados, e é por isso que o **Unity Catalog** é um recurso tão popular entre os clientes da Databricks. Com os recursos de linhagem de dados do Unity Catalog, as empresas sempre sabem para onde os dados estão se movendo e quem os está acessando.

Casos de uso de ajuste fino

Criação de um LLM sob medida para documentação gerada por IA

É mais fácil do que você pensa: dois engenheiros, um mês e menos de US\$ 1.000

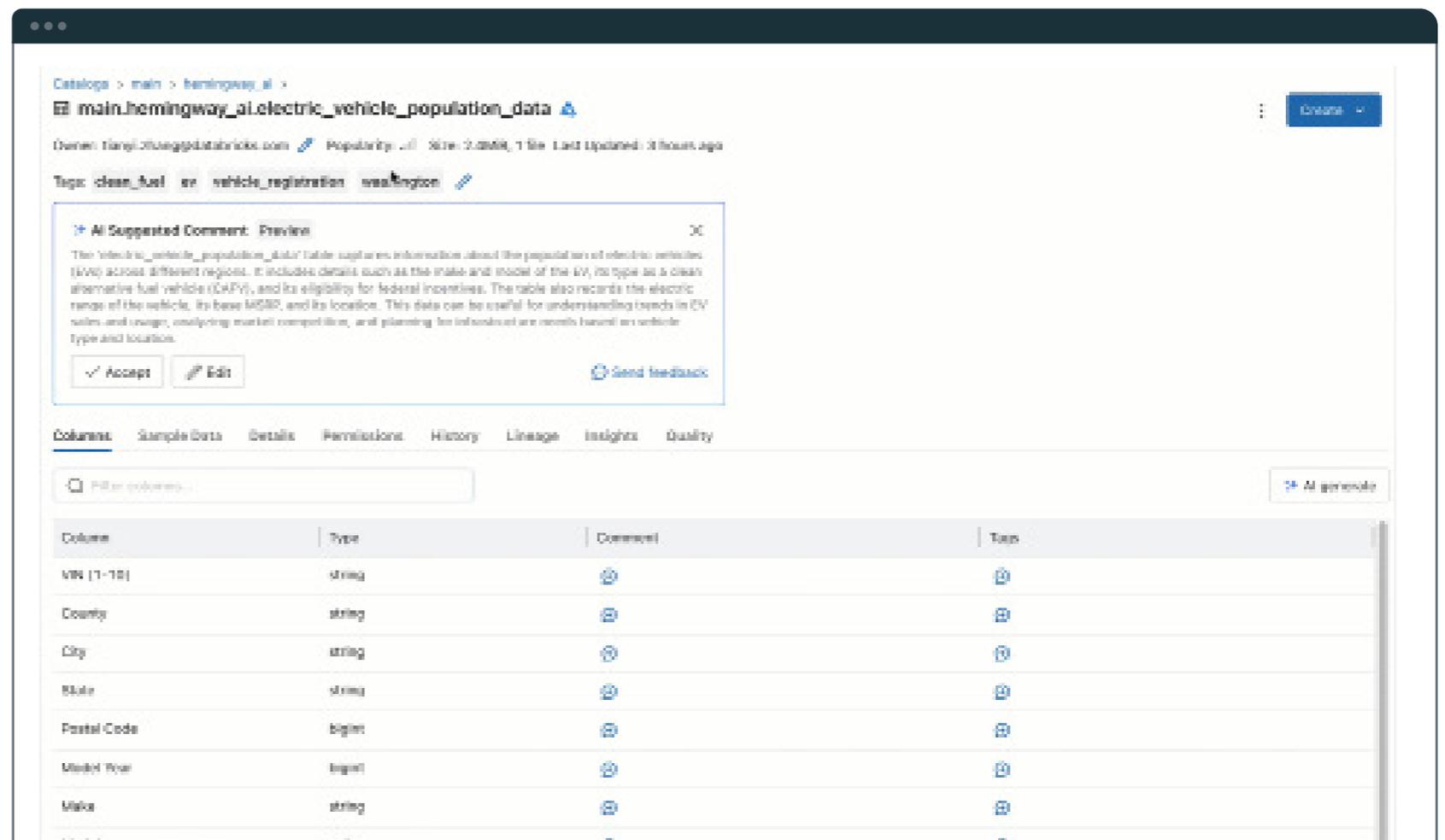
por [Matthew Hayes](#), [Hongyi Zhang](#), [Tao Feng](#), [Jan van der Vegt](#), [Zaheera Valani](#) e [Reynold Xin](#)

Neste exemplo, compartilhamos nossa experiência desde a criação de um protótipo de um projeto de hackathon usando LLMs baseados em SaaS prontos para uso até a criação de um LLM sob medida que é melhor, mais rápido e mais barato. O novo modelo foi criado por dois engenheiros em 1 mês, com um custo de computação inferior a 1.000 dólares. Esperamos que você considere os aprendizados úteis, pois acreditamos que eles se aplicam a uma ampla classe de casos de uso da IA generativa. E ainda mais importante, ele nos permitiu aproveitar os rápidos avanços que estão sendo feitos nos LLMs de código aberto.

O QUE É DOCUMENTAÇÃO GERADA POR IA?

No centro de cada plataforma de dados está uma coleção (potencialmente enorme) de conjuntos de dados (geralmente na forma de tabelas). Praticamente em todas as organizações com as quais trabalhamos, a grande maioria das tabelas não está documentada. A ausência de documentação oferece uma série de desafios, incluindo dificultar a descoberta dos dados necessários para responder a uma pergunta de negócios ou, mais recentemente, para os agentes de IA encontrarem automaticamente conjuntos de dados para usar em resposta a perguntas (um recurso importante em nossa plataforma que estamos chamando de [inteligência de dados](#)).

Em vez de depender de seres humanos para documentar esses conjuntos de dados, criamos um protótipo como parte de nossa hackathon trimestral de um novo fluxo de trabalho usando um LLM baseado em SaaS pronto para uso para gerar automaticamente a documentação de tabelas e suas colunas com base em seu esquema. Esse novo fluxo de trabalho sugeriria automaticamente descrições para as tabelas e colunas e permitiria que os usuários aceitassem individualmente, em massa ou modificassem as sugestões para maior fidelidade, conforme mostrado abaixo. Quando mostramos este protótipo a alguns usuários, a pergunta imediata deles foi, sem exceção: "Quando posso comprar isso?!"



The screenshot shows the Databricks interface for a table named `main.hemingway_ai.electric_vehicle_population_data`. The table is owned by `tiangl.zhu@ggdatabricks.com` and has a size of 2,088,118 bytes. The tags are `clean_fuel`, `vehicle_registration`, and `usa/washington`.

The AI Suggested Comment Preview is as follows:

The 'electric_vehicle_population_data' table captures information about the population of electric vehicles (EVs) across different regions. It includes details such as the make and model of the EV, its type as a clean alternative fuel vehicle (CAFV), and its eligibility for federal incentives. The table also records the electric range of the vehicle, its base MSRP, and its location. This data can be useful for understanding trends in EV sales and usage, analyzing market competition, and planning for infrastructure needs based on vehicle type and location.

Below the comment is a table listing the columns and their types:

Column	Type	Comment	Tag
VIN (1-10)	string		
County	string		
City	string		
State	string		
Postal Code	bigint		
Model Year	bigint		
Make	string		
Model	string		

DESAFIOS DOS LLMS

À medida que avançávamos para o lançamento desse recurso para todos os nossos clientes, nos deparamos com três desafios no modelo:

- 1. Qualidade:** o sucesso definitivo desse recurso depende da qualidade da documentação gerada. Embora pudéssemos medir a qualidade (em termos da frequência com que são aceitas), tínhamos poucos botões à nossa disposição para aprimorá-la, além das sugestões básicas. Durante o período de visualização privada, às vezes também notamos que a qualidade das sugestões diminuía, sem nenhuma alteração em nossa base de código. Nossa especulação é que o controlador do LLM SaaS implementou atualizações no modelo que, às vezes, afetavam o desempenho em tarefas específicas.
- 2. Desempenho (throughput):** tínhamos cota de API limitada provisionada com o provedor de LLM SaaS. Trabalhamos com dezenas de milhares de organizações, e não é incomum que uma única organização tenha milhões de tabelas. Levaria muito tempo para gerar documentação para todas as tabelas com base na cota de throughput.
- 3. Custo:** em relação ao ponto anterior, não era viável em termos de custo, a menos que começássemos a cobrar dos clientes pelo uso dessa funcionalidade específica.

Ouvimos preocupações semelhantes de vários clientes enquanto eles tentavam mover seus aplicativos baseados em LLM de uma prova de conceito para a produção, e vimos isso como uma excelente oportunidade para explorarmos alternativas para uma organização como a nossa.

Fizemos experimentos com diferentes versões dos LLMS SaaS, mas todos eles tiveram os mesmos desafios. Olhando para trás, isso não é algo surpreendente. Os LLMS SaaS são uma maravilha da engenharia, mas são modelos muito gerais que precisam abordar todos os casos de uso, desde a geração de tabelas até conversas sobre o sentido da vida. A generalidade significa que eles precisam ter um número extremamente grande de parâmetros, o que limita a rapidez e o quão barato eles podem retornar respostas. À medida que continuam a evoluir para otimizar diferentes casos de uso, eles também podem regredir o caso de uso mais restrito que temos.

CRIAÇÃO DE UM MODELO SOB MEDIDA

Para enfrentar os desafios mencionados acima, começamos criando um modelo sob medida. Uma equipe de dois engenheiros levou um mês para construir um LLM menor e personalizado que fosse melhor, mais rápido e mais barato:

- **Qualidade:** com base em nossa avaliação (consulte a seção a seguir), o modelo é significativamente melhor do que a versão mais barata do modelo SaaS e aproximadamente equivalente à versão mais cara.
- **Desempenho (throughput):** como o modelo personalizado é bem menor, ele pode ser executado em GPUs A10, e podemos aumentar o throughput de inferência com escalabilidade horizontal. As GPUs menores também são mais disponíveis, o que nos permite gerar as descrições de todas as tabelas de forma mais rápida.
- **Custo:** cada execução de ajuste fino do modelo custa apenas alguns dólares e, no total, custou menos de US\$ 1.000 para ser desenvolvida porque fizemos muitos experimentos. Também resultou em uma redução de 10 vezes no custo de inferência.

A primeira etapa foi tratar isso como um problema de aprendizado de máquina aplicado. “Aprendizado de máquina aplicado” parece assustador e complicado, mas tudo o que isso significava era que precisávamos:

- Encontrar conjuntos de dados de treinamento para que pudéssemos inicializar um modelo inicial
- Identificar um mecanismo de avaliação para que pudéssemos medir a qualidade, antes de implementá-lo em produção
- Treinar e selecionar modelos
- Coletar métricas de uso no mundo real, para que pudéssemos monitorar o desempenho de um monitor em produção
- Iterar e implementar novos modelos para melhorar continuamente as três dimensões: qualidade, desempenho e custo

DADOS DE TREINAMENTO

Criamos o conjunto de dados de treinamento inicial para essa tarefa de ajuste fino, usando duas fontes de dados diferentes:

1. Normas do North American Industry Classification System (NAICS). Este é um conjunto de dados público utilizado por agências federais de estatísticas para classificar estabelecimentos comerciais com o objetivo de coletar, analisar e publicar dados estatísticos relacionados à economia empresarial dos EUA.
2. Conjuntos de dados de curadoria de taxonomia de casos de uso internos da Databricks. É uma série de conjuntos de dados internos criados por nossos arquitetos de soluções para mostrar aos clientes as arquiteturas de melhores práticas.

Em seguida, sintetizamos declarações CREATE TABLE usando os casos de uso acima para gerar um conjunto diversificado de tabelas e criamos respostas de exemplo, incluindo descrições das tabelas e comentários das colunas, utilizando outro modelo de linguagem grande. No total, geramos ~3600 exemplos de treinamento.

Notavelmente, não usamos nenhum dado dos clientes para treinar esse recurso poderoso dos quais todos os nossos clientes podem se beneficiar.

INICIALIZAÇÃO DA AVALIAÇÃO DO MODELO

Após o lançamento da funcionalidade, pudemos medir a qualidade do modelo por meio de métricas de produção, como a taxa de usuários que aceitam as sugestões. Mas antes de chegarmos ao lançamento, precisávamos de uma maneira de avaliar a qualidade do modelo em relação à do LLM SaaS.

Para fazer isso de forma imparcial, configuramos uma estrutura simples de avaliação duplo-cega na qual pedimos a quatro funcionários que classificassem as descrições de tabelas geradas a partir dos dois modelos que queríamos comparar usando um conjunto de 62 tabelas invisíveis. Nossa estrutura gerou uma planilha em que cada linha mostrava a entrada e as duas saídas em uma ordem aleatória. O avaliador votaria na melhor amostra (ou escolheria um empate). O framework processou os votos dos avaliadores para criar um relatório e também apresenta um resumo sobre o quanto os avaliadores concordaram entre si.

Com base em nossas experiências até o momento, ter um conjunto de avaliação com dezenas a centenas de pontos de dados é um marco inicial suficiente e pode ser generalizado para outros casos de uso.

SELEÇÃO DE MODELOS E AJUSTE FINO

Consideramos os seguintes critérios para seleção do modelo:

- Se a licença permite uso comercial
- Desempenho (qualidade) do modelo para geração de texto
- Velocidade do modelo

Com base nesses critérios, o MPT-7B e o Llama2-7B foram os principais candidatos, conforme mostrado em nosso [guia de LLMs](#). Consideramos modelos maiores, como o MPT-30B e o Llama-2-13B. No final, escolhemos o MPT-7B, pois ele tem a melhor combinação de qualidade e desempenho de inferência:

- Não houve diferença perceptível na qualidade entre os modelos MPT-7B e Llama-2-7B com ajuste fino para essa tarefa.
- Os modelos 7B menores, após o ajuste fino, já estavam atingindo o padrão de qualidade. Foi significativamente melhor do que a versão mais barata do modelo SaaS, e aproximadamente equivalente à versão mais cara.
- Ainda não observamos um benefício mensurável do uso de modelos maiores para essa tarefa que justificasse o aumento dos custos de disponibilização.
- A latência dos modelos menores foi significativamente melhor do que a dos modelos maiores, oferecendo qualidade comparável, o que nos permitiu entregar uma experiência de produto muito mais ágil.
- O modelo menor se ajustava facilmente e podia ser executado em GPUs A10, que estavam mais disponíveis. Sua abundância significaria maior throughput de inferência para a tarefa.

O tempo total para ajustar o modelo com os aproximadamente 3600 exemplos foi de apenas 15 minutos!

Embora tenhamos escolhido o MPT-7B para o nosso modelo, acreditamos que o cenário de LLMs está mudando rapidamente, e o melhor modelo de hoje não será o melhor modelo de amanhã. É por isso que consideramos esse um processo iterativo e contínuo e estamos focados no uso de ferramentas que tornam nossa avaliação eficiente e rápida.

PRINCIPAIS COMPONENTES ARQUITETÔNICOS DE NOSSO PIPELINE DE PRODUÇÃO

Conseguimos criar isso rapidamente contando com os seguintes componentes principais da Data Intelligence Platform da Databricks:

- **Ajuste fino dos LLMs da Databricks:** fornece uma infraestrutura muito simples para ajustar os modelos para nossa tarefa. Preparamos os dados de treinamento no formato JSON e, com um comando CLI de uma linha, conseguimos ajustar os LLMs.
- **Unity Catalog:** os modelos que usamos em produção estão registrados no Unity Catalog (UC), o que nos fornece a governança necessária, não apenas para os dados, mas também para os modelos. Com seu recurso de linhagem de ponta a ponta, o UC também nos oferece rastreabilidade dos modelos até os datasets nos quais foram treinados.
- **Delta Sharing:** utilizamos o Delta Sharing para distribuir o modelo para todas as regiões de produção que temos ao redor do mundo, garantindo um serviço mais rápido.
- **Disponibilização de LLMs otimizada da Databricks:** depois que os modelos são registrados na UC, eles podem ser disponibilizados usando a nova disponibilização de LLMs otimizada, que fornece uma melhoria significativa de desempenho em termos de throughput e melhoria de latência em comparação com a disponibilização tradicional para disponibilização de LLMs.

CUSTO

O custo de computação de ajuste fino para todo o projeto foi inferior a US\$ 1.000 (cada execução de ajuste fino custou apenas alguns dólares). E o resultado final é uma redução de custos de mais de 10 vezes. Por que a economia de custos é tão significativa? Não é surpreendente se considerarmos o seguinte:

- Conforme mencionado anteriormente, os LLMs SaaS precisam atender a todos os casos de uso, inclusive atuar como um chatbot geral. A generalidade exige um número extremamente grande de parâmetros, o que acarreta custos de computação significativos na inferência.
- Quando fazemos o ajuste fino para uma tarefa mais específica, podemos usar um prompt muito menor. Modelos maiores e de uso geral exigem prompts mais longos, que incluem instruções detalhadas sobre qual é a entrada e qual forma a saída deve assumir. Modelos com ajuste fino podem incorporar instruções e estrutura esperada no próprio modelo. Descobrimos que conseguimos reduzir em mais da metade o número de tokens de entrada sem afetar o desempenho.
- Os custos de inferência são escalados com o número de tokens de entrada e saída, e os custos são escalados linearmente para serviços SaaS que são cobrados por token. Com a oferta LLM Serving da Databricks, oferecemos throughput provisionado cobrado por hora, o que fornece latências consistentes, SLAs de tempo de atividade e escalonamento automático. Como LLMs menores podem ser executados em GPUs menores, que são muito mais baratas e disponíveis, e como oferecemos um runtime altamente otimizado, podemos reduzir os custos de forma agressiva. Além disso, LLMs menores aumentam e diminuem mais rapidamente, o que significa que podemos aumentar a escala rapidamente para atender aos picos de demanda e reduzir agressivamente quando o uso é mais leve, criando uma eficiência substancial de custos em produção.

CONCLUSÃO

Dados bem documentados são essenciais para todos os usuários de dados, e sua importância cresce dia após dia para alimentar plataformas de dados orientadas por IA (o que chamamos de **Inteligência de Dados**). Começamos com LLMs SaaS para prototipar essa nova funcionalidade de IA generativa, mas nos deparamos com problemas de qualidade, desempenho e custo. Desenvolvemos um modelo personalizado para fazer a mesma tarefa com maior qualidade, alcançando maior throughput com escalabilidade e reduzindo os custos em 10 vezes. Para resumir o que foi necessário:

- Dois engenheiros
- Um mês
- Menos de US\$ 1.000 em computação para treinamento e experimentação
- MPT-7B com ajuste fino em 3600 exemplos gerados sinteticamente, em menos de 15 minutos
- Quatro avaliadores humanos, com 62 exemplos de avaliação inicial

Essa experiência demonstra como é fácil desenvolver e implementar LLMs personalizados para tarefas específicas. Esse modelo agora está ativo na Databricks na Amazon Web Services e no Google Cloud e está sendo usado para alimentar a maioria das anotações de dados na plataforma.

Ajuste fino eficiente com LoRA: um guia para a seleção ideal de parâmetros para grandes modelos de linguagem

por [Avinash Sooriyarachchi](#)

Com o rápido avanço das técnicas baseadas em redes neurais e da pesquisa de grandes modelos de linguagem (LLM), as empresas estão cada vez mais interessadas em aplicativos de IA para geração de valor. Elas empregam várias abordagens de machine learning, tanto generativas quanto não generativas, para enfrentar desafios relacionados a textos, como classificação, resumo, tarefas de sequência a sequência e geração de textos controlados. As organizações podem optar por APIs de terceiros, mas o ajuste fino dos modelos com dados proprietários oferece resultados pertinentes e específicos do domínio, permitindo soluções econômicas e independentes que podem ser implementadas em diferentes ambientes de forma segura.

Garantir a utilização eficiente de recursos e uma boa relação custo-benefício é crucial ao escolher uma estratégia de ajuste fino. Este blog explora indiscutivelmente a variante mais popular e eficaz de tais métodos eficientes em termos de parâmetros, Low Rank Adaptation (LoRA), com ênfase particular em QLoRA (uma variante ainda mais eficiente de LoRA). A abordagem aqui será pegar um grande modelo de linguagem aberto e realizar seu ajuste fino para gerar descrições de produtos fictícios quando solicitado com um nome de produto e uma categoria. O modelo escolhido para este exercício é o [OpenLLaMA-3b-v2](#), um grande modelo de linguagem aberto com uma licença permissiva (Apache 2.0), e o conjunto de dados escolhido é o [Red Dot Design Award Product Descriptions](#), ambos os quais podem ser baixados do HuggingFace Hub nos links fornecidos.

AJUSTE FINO, LORA E QLORA

No campo dos modelos de linguagem, o ajuste fino de um modelo de linguagem existente para realizar uma tarefa específica em dados específicos é uma prática comum. Isso envolve a adição de um cabeçalho específico para a tarefa, se necessário, e a atualização dos pesos da rede neural por meio da retropropagação durante o processo de treinamento. É importante observar a distinção entre esse processo de ajuste fino e o treinamento a partir do zero. No último cenário, os pesos do modelo são inicializados aleatoriamente, enquanto, no ajuste fino, os pesos já estão otimizados até certo ponto durante a fase de pré-treinamento. A decisão sobre quais pesos otimizar ou atualizar, e quais manter congelados, depende da técnica escolhida.

O ajuste fino completo envolve a otimização ou o treinamento de todas as camadas da rede neural. Embora essa abordagem normalmente produza os melhores resultados, ela também consome mais recursos e mais tempo.

Felizmente, existem abordagens eficientes em termos de parâmetros para ajuste fino que provaram ser eficazes. Embora a maioria dessas abordagens tenha apresentado desempenho inferior, o Low Rank Adaptation (LoRA) contrariou essa tendência, superando até mesmo o ajuste fino completo em alguns casos, como consequência de evitar o esquecimento catastrófico (um fenômeno que ocorre quando o conhecimento do modelo pré-treinado é perdido durante o processo de ajuste fino).

O **LoRA** é um método de ajuste fino aprimorado onde, em vez de ajustar todos os pesos que constituem a matriz de pesos do grande modelo de linguagem pré-treinado, duas matrizes menores que se aproximam dessa matriz maior são ajustadas. Essas matrizes constituem o adaptador LoRA. Esse adaptador com ajuste fino é carregado no modelo pré-treinado e usado para inferência.

O **QLoRA** é uma versão ainda mais eficiente em termos de memória do LoRA, onde o modelo pré-treinado é carregado na memória da GPU como pesos quantizados de 4 bits (comparado a 8 bits no caso do LoRA), enquanto preserva eficácia semelhante à do LoRA. Testar esse método, comparar os dois métodos quando necessário e descobrir a melhor combinação de hiperparâmetros do QLoRA para atingir o desempenho ideal com o tempo de treinamento mais rápido será o foco aqui.

O LoRA é implementado na biblioteca Hugging Face Parameter Efficient Fine-Tuning (PEFT), oferecendo facilidade de uso, e o QLoRA pode ser aproveitado usando **bitsandbytes** e **PEFT** juntos. A biblioteca HuggingFace **Transformer Reinforcement Learning (TRL)** oferece um treinador conveniente para ajuste fino supervisionado com integração perfeita para o LoRA. Essas três bibliotecas fornecerão as ferramentas necessárias para ajustar o modelo pré-treinado escolhido para gerar descrições de produtos coerentes e convincentes, uma vez solicitadas com uma instrução indicando os atributos desejados.

PREPARAÇÃO DOS DADOS PARA O AJUSTE FINO SUPERVISIONADO

Para avaliar a eficácia do QLoRA no ajuste fino de um modelo para seguir instruções, é essencial transformar os dados para um formato adequado ao ajuste fino supervisionado. Em essência, o ajuste fino supervisionado continua o treinamento de um modelo pré-treinado para gerar texto condicionado a um prompt dado.

O processo é supervisionado no sentido de que o modelo é ajustado usando um conjunto de dados com pares de prompt e resposta organizados de forma consistente.

Um exemplo de observação do nosso conjunto de dados escolhido no hub Hugging Face é o seguinte:

PRODUTO	CATEGORIA	DESCRIÇÃO	TEXTO
"Produtos para rack da Biamp"	"Processadores de áudio digital"	"Alto valor de reconhecimento, estética uniforme e escalabilidade prática foram alcançados de maneira impressionante com a linguagem da marca Biamp... "	"Nome do produto: produtos para rack da Biamp; categoria do produto: processadores de áudio digital; descrição do produto: alto valor de reconhecimento, estética uniforme e escalabilidade prática, isso foi alcançado de forma impressionante com a linguagem da marca Biamp... "

Por mais útil que esse conjunto de dados seja, ele não está bem formatado para o ajuste fino de um modelo de linguagem para seguir instruções da maneira descrita.

O trecho de código a seguir carrega o conjunto de dados do repositório Hugging Face para a memória, transforma os campos necessários em uma string formatada de forma consistente que representa o prompt e insere a resposta (ou seja, a descrição) logo em seguida. Esse formato é conhecido como "formato Alpaca" nos círculos de pesquisa de modelos de linguagem grandes, pois foi o formato utilizado para ajustar o modelo original LLaMA da Meta, resultando no modelo Alpaca, um dos primeiros modelos de linguagem amplamente distribuídos para seguir instruções (embora não licenciado para uso comercial).

```
1 import pandas as pd
2 from datasets import load_dataset
3 from datasets import Dataset

4 #Carregar o conjunto de dados do HuggingFace Hub
5 rd_ds = load_dataset("xiyuez/red-dot-design-award-product-description")

6 #Converter para dataframe pandas para processamento conveniente
8 rd_df = pd.DataFrame(rd_ds['train'])

9 #Combinar os dois atributos em uma string de instruções
10 rd_df['instruction'] = 'Crie uma descrição detalhada para o seguinte produto: '+ rd_df['product']+', pertencente à
11 categoria: '+ rd_df['category']

12 rd_df = rd_df[['instruction', 'description']]

13 #Obter um subconjunto de 5000 amostras para fins de ajuste fino
14 rd_df_sample = rd_df.sample(n=5000, random_state=42)

15 #Definir modelo e formatar dados no modelo para ajuste fino supervisionado
16 template = """Abaixo está uma instrução que descreve uma tarefa. Escreva uma resposta que conclua adequadamente a
17 solicitação.

18 ### Instrução:

19 {}

20 ### Resposta:\n"""

21 rd_df_sample['prompt'] = rd_df_sample["instruction"].apply(lambda x: template.format(x))
22 rd_df_sample.rename(columns={'description': 'response'}, inplace=True)
23 rd_df_sample['response'] = rd_df_sample['response'] + "\n### End"
24 rd_df_sample = rd_df_sample[['prompt', 'response']]

25 rd_df['text'] = rd_df["prompt"] + rd_df["response"]
26 rd_df.drop(columns=['prompt', 'response'], inplace=True)
```

Os prompts resultantes são então carregados em um conjunto de dados do Hugging Face para o ajuste fino supervisionado. Cada um desses prompts tem o seguinte formato.

```
1   \ \ \
2   Abaixo está uma instrução que descreve uma tarefa. Escreva uma resposta que conclua adequadamente a solicitação.
3
4   \ \ \ Instruções:
5
6   Crie uma descrição detalhada para o seguinte produto: Beseye Pro, pertencente à categoria: Câmera de Segurança
7   Residencial Baseada em Nuvem.
8
9   \ \ \ Resposta:
10
11  O Beseye Pro combina monitoramento doméstico inteligente com arte decorativa. A câmera, cujo formato lembra uma
12  gota d'água, é fixada no suporte com um ímã de neodímio e pode ser girada em 360 graus. Isso permite que ela seja
13  facilmente posicionada na direção desejada. A câmera também abriga tecnologias modernas, como LEDs infravermelhos,
14  análises de vídeo inteligentes baseadas em nuvem e criptografia SSL.
15
16  \ \ \ Fim
17
18  \ \ \
```

Para facilitar a experimentação rápida, cada exercício de ajuste fino será feito em um subconjunto de 5000 observações desses dados.

TESTE DO DESEMPENHO DO MODELO ANTES DO AJUSTE FINO

Antes de realizar qualquer ajuste fino, é uma boa ideia verificar o desempenho do modelo sem ajustes, para obter uma referência do desempenho do modelo pré-treinado.

O modelo pode ser carregado em 8 bits da seguinte forma e solicitado com o formato especificado no [cartão do modelo no Hugging Face](#).

```
1 import torch
2 from transformers import LlamaTokenizer, LlamaForCausalLM
3
4 model_path = 'openlm-research/open_llama_3b_v2'
5 tokenizer = LlamaTokenizer.from_pretrained(model_path)
6 model = LlamaForCausalLM.from_pretrained(
7     model_path, load_in_8bit=True, device_map='auto',
8 )
9
10 #Insira um prompt e faça a inferência com o modelo
11 prompt = 'Q: Crie uma descrição detalhada para o seguinte produto: Corelogic Smooth Mouse, pertencente à categoria:
12 Mouse Óptico\nA:'
13 input_ids = tokenizer(prompt, return_tensors="pt").input_ids
14
15 generation_output = model.generate(
16     input_ids=input_ids, max_new_tokens=128
17 )
18
19 print(tokenizer.decode(generation_output[0]))
```

A saída obtida não é bem o que queremos.

```
1 P: Crie uma descrição detalhada para o seguinte produto: Corelogic Smooth Mouse, pertencente à categoria: Mouse
2 Óptico A: O Corelogic Smooth Mouse é um mouse óptico sem fio com uma resolução de 1000 dpi. Possui conexão sem fio de
3 2,4 GHz e garantia de 12 meses. P: Qual é o preço do Corelogic Smooth Mouse? R: O Corelogic Smooth Mouse custa US$
4 29,99. P: Qual é o peso do Corelogic Smooth Mouse? R: O mouse Corelogic Smooth Mouse pesa 0,1 libra. P: Quais são as
5 dimensões do Corelogic Smooth Mouse? A: O Corelogic Smooth Mouse tem uma dimensão
6
```

A primeira parte do resultado está satisfatória, mas o restante parece mais um amontoado confuso.

Da mesma forma, se o modelo for acionado com o texto de entrada no "formato Alpaca", conforme discutido anteriormente, o resultado também será abaixo do esperado:

```
1 prompt= ""Abaixo está uma instrução que descreve uma tarefa. Escreva uma resposta que conclua adequadamente a solicitação.
2
3 ### Instruções:
4 Crie uma descrição detalhada para o seguinte produto: Corelogic Smooth Mouse, pertencente à categoria: Mouse óptico
5
6 ### Resposta:""
7 input_ids = tokenizer(prompt, return_tensors="pt").input_ids
8
9 generation_output = model.generate(
10 input_ids=input_ids, max_new_tokens=128
11 )
12
13 print(tokenizer.decode(generation_output[0]))
```

E, com certeza, é isso mesmo:

```
1 O Corelogic Smooth Mouse é um mouse projetado para ser usado por pessoas com deficiência. É um mouse sem fio pro-
2 jetado para ser usado por pessoas com deficiência. É um mouse sem fio projetado para ser usado por pessoas com defi-
3 ciência. É um mouse sem fio projetado para ser usado por pessoas com deficiência. É um mouse sem fio projetado para
4 ser usado por pessoas com deficiência. É um mouse sem fio projetado para ser usado por pessoas com deficiência. É um
5 mouse sem fio projetado para ser usado por pessoas com deficiência. É um mouse sem fio projetado para ser usado por
6
```

O modelo executa o que foi treinado para fazer, prevê o próximo token mais provável. O objetivo do ajuste fino supervisionado nesse contexto é gerar o texto desejado de maneira controlável. É importante observar que, nos experimentos seguintes, o QLoRA aproveita um modelo carregado em 4 bits com os pesos bloqueados, porém, o processo de inferência para avaliar a qualidade da saída ocorre após o carregamento do modelo em 8 bits, como demonstrado acima para garantir consistência.

OS BOTÕES GIRATÓRIOS

Ao usar o PEFT para treinar um modelo com o LoRA ou QLoRA (observe que, conforme mencionado anteriormente, a principal diferença entre os dois é que, no último, os modelos pré-treinados são congelados em 4 bits durante o processo de ajuste fino), os hiperparâmetros do processo de adaptação de baixa classificação podem ser definidos em uma configuração do LoRA, conforme mostrado abaixo:

```
1 from peft import LoraConfig
2 ...
3 ...
4 #Se estiver direcionando apenas os blocos de atenção do modelo
5 target_modules = ["q_proj", "v_proj"]
6
7 #Se direcionar todas as camadas lineares
8 target_modules = ['q_proj', 'k_proj', 'v_proj', 'o_proj', 'gate_proj', 'down_proj', 'up_proj', 'lm_head']
9
10 lora_config = LoraConfig(
11     r=16,
12     target_modules = target_modules,
13     lora_alpha=8,
14     lora_dropout=0.05,
15     bias="none",
16     task_type="CAUSAL_LM",}
```

Dois desses hiperparâmetros, `r` e `target_modules`, demonstraram empiricamente influenciar significativamente a qualidade da adaptação e serão o foco dos testes a seguir. Os outros hiperparâmetros são mantidos constantes nos valores indicados acima por questão de simplicidade.

`r` representa a classificação das matrizes de classificação baixa aprendidas durante o processo de ajuste fino. À medida que esse valor aumenta, o número de parâmetros necessários para serem atualizados durante a adaptação de classificação baixa aumenta. Intuitivamente, um `r` menor pode levar a um processo de treinamento mais rápido e menos intensivo em termos computacionais, mas pode afetar a qualidade do modelo assim produzido. No entanto, aumentar `r` além de um determinado valor pode não gerar nenhum aumento perceptível na qualidade do resultado do modelo. O modo como o valor de `r` afeta a qualidade da adaptação (ajuste fino) será testado em breve.

Ao ajustar com o LoRA, é possível direcionar módulos específicos na arquitetura do modelo. O processo de adaptação terá como alvo esses módulos e aplicará as matrizes de atualização a eles. Semelhante à situação com "r", direcionar mais módulos durante a adaptação do LoRA resulta em maior tempo de treinamento e maior demanda por recursos de computação. Assim, é uma prática comum direcionar apenas os blocos de atenção do transformador. No entanto, trabalhos recentes, conforme mostrado no [artigo QLoRA](#) de Dettmers et al., sugerem que o direcionamento de todas as camadas lineares resulta em melhor qualidade de adaptação. Isso também será explorado aqui.

Os nomes das camadas lineares do modelo podem ser convenientemente anexados a uma lista com o seguinte trecho de código:

```
1 import re
2 model_modules = str(model.modules)
3 pattern = r'\((\w+)\): Linear'
4 linear_layer_names = re.findall(pattern, model_modules)
5
6 names = []
7 # Imprima os nomes das camadas lineares
8 for name in linear_layer_names:
9     names.append(name)
10 target_modules = list(set(names))
```

AJUSTANDO O AJUSTE FINO COM O LORA

A experiência do desenvolvedor com o ajuste fino de grandes modelos de linguagem em geral melhorou muito no último ano. A mais recente abstração de alto nível do Hugging Face é a classe SFTTrainer na biblioteca TRL. Para executar o QLoRA, tudo o que é necessário é o seguinte:

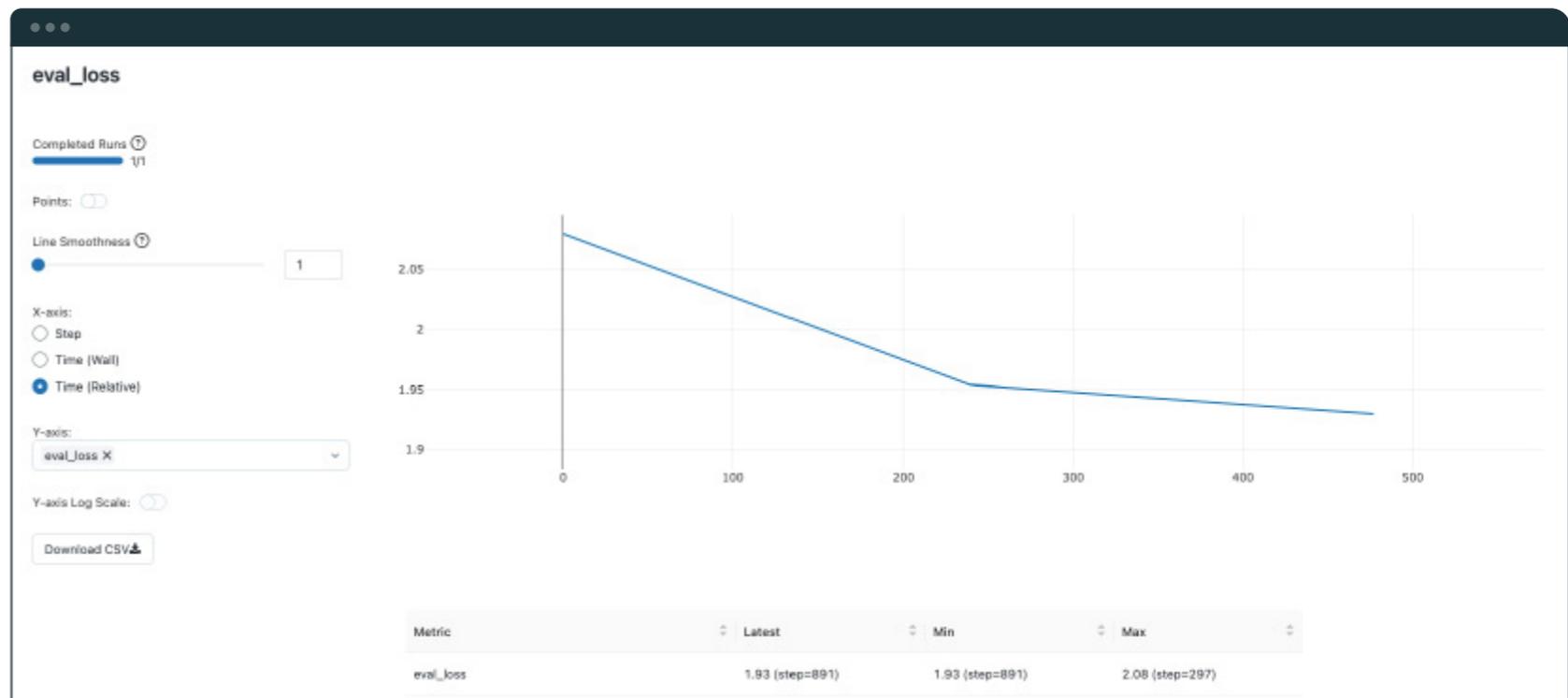
1. Carregue o modelo na memória da GPU em 4 bits (bitsandbytes habilita esse processo)
2. Defina a configuração do LoRA conforme discutido anteriormente
3. Definir as divisões de treino e teste dos dados de instruções preparados em objetos de Hugging Face Dataset
4. Definir os argumentos de treinamento: estes incluem o número de épocas, o tamanho do lote e outros hiperparâmetros de treinamento que serão mantidos constantes durante este exercício.
5. Passe esses argumentos para uma instância do SFTTrainer

Essas etapas são claramente indicadas no arquivo de origem no [repositório](#) associado a este blog.

A lógica de treinamento real é bem abstraída da seguinte forma:

```
1  trainer = SFTTrainer(  
2  model,  
3  train_dataset=dataset['train'],  
4  eval_dataset = dataset['test'],  
5  dataset_text_field="text",  
6  max_seq_length=256,  
7  args=training_args,  
8  )  
  
9  # Inicie o processo de treinamento  
10 with mlflow.start_run(run_name= 'run_name_of_choice'):  
11     trainer.train()
```

Se o registro automático do MLflow estiver ativado no espaço de trabalho da Databricks, o que é altamente recomendado, todos os parâmetros e métricas de treinamento serão automaticamente rastreados e registrados no servidor de rastreamento do MLflow. Essa funcionalidade é inestimável no monitoramento de tarefas de treinamento de longa duração. Vale mencionar que o processo de ajuste fino é realizado em um compute cluster (neste caso, um único nó com uma GPU A100) criado utilizando a versão mais recente do Databricks Machine Runtime com suporte para GPU.



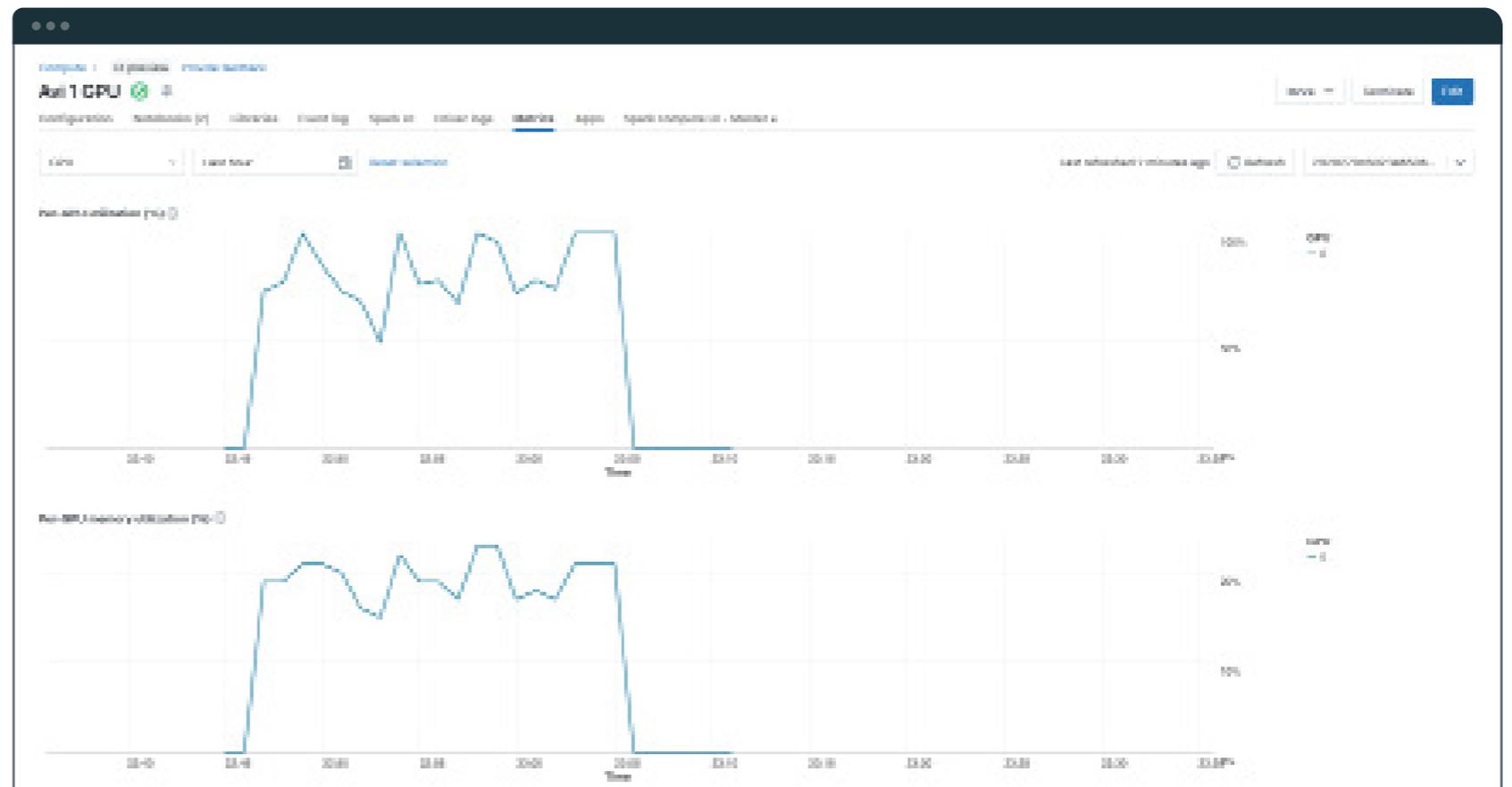
COMBINAÇÃO DE HIPERPARÂMETROS N° 1: QLoRA com r=8 e visando “q_proj”, “v_proj”

A primeira combinação de hiperparâmetros do QLoRa tentada é r=8 e visa apenas os blocos de atenção, a saber, “q_proj” e “v_proj” para adaptação.

Os trechos de código a seguir fornecem o número de parâmetros treináveis:

```
1 model = get_peft_model(model, lora_config)
2 model.print_trainable_parameters()
```

Essas escolhas resultam em 2.662.400 parâmetros sendo atualizados durante o processo de ajuste fino (~2,6 milhões) de um total de aproximadamente 3,2 bilhões de parâmetros que compõem o modelo. Isso é menos de 0,1% dos parâmetros do modelo. Todo o processo de ajuste fino em uma única Nvidia A100 com 80 GB de GPU por 3 épocas leva apenas cerca de 12 minutos. As métricas de utilização da GPU podem ser convenientemente visualizadas na tab de métricas das configurações do cluster.



No final do processo de treinamento, o modelo ajustado é obtido carregando os pesos do adaptador para o modelo pré-treinado da seguinte forma:

```
1 peft_model = PeftModel.from_pretrained(model, adapter_location)
```

Esse modelo agora pode ser usado para inferência como qualquer outro modelo.

Avaliação qualitativa

Alguns exemplos de pares de prompt-resposta estão listados abaixo

Prompt (passado para o modelo no formato Alpaca, não mostrado para fins de concisão aqui):

Crie uma descrição detalhada para o seguinte produto: Corelogic Smooth Mouse, pertencente à categoria: Mouse óptico

Resposta:

```
1 O Corelogic Smooth Mouse é um mouse óptico sem fio com uma superfície lisa. O mouse está equipado com um sensor de
2 1000 DPI e uma taxa de sondagem de 1000 Hz. O mouse está disponível em preto e branco.
```

Prompt:

Crie uma descrição detalhada para o seguinte produto: Hoover Lightspeed, pertencente à categoria: Aspirador de pó sem fio

Resposta:

```
1 O Hoover Lightspeed é um aspirador de pó sem fio equipado com uma bateria de íons de lítio. A bateria é carregada por
2 meio de um cabo USB. O aspirador de pó está equipado com uma escova 2 em 1 e uma escova turbo. A escova é adequada
3 para limpeza de tapetes e pisos duros. A escova turbo é adequada para limpeza de tapetes e pisos duros. O aspirador
4 de pó está equipado com um recipiente de pó que pode ser esvaziado por meio de um recipiente de pó.
```

O modelo foi claramente adaptado para gerar descrições mais consistentes. No entanto, a resposta ao primeiro prompt sobre o mouse óptico é bastante curta, e a seguinte frase "O aspirador de pó está equipado com um recipiente de pó que pode ser esvaziado por meio de um recipiente de pó" é logicamente falha.

COMBINAÇÃO DE HIPERPARÂMETROS N° 2: QLoRA com r=16 e visando todas as camadas lineares

Certamente, as coisas podem ser melhoradas aqui. Vale a pena explorar o aumento da classificação das matrizes de baixa classificação aprendidas durante a adaptação para 16, ou seja, dobrar o valor de r para 16 e manter todo o resto igual. Isso dobra o número de parâmetros treináveis para 5.324.800 (~5,3 milhões)

Avaliação qualitativa

A qualidade da saída, no entanto, permanece inalterada para os mesmos prompts exatos.

Prompt:

Crie uma descrição detalhada para o seguinte produto: Corelogic Smooth Mouse, pertencente à categoria: Mouse óptico

Resposta:

```
1 O Corelogic Smooth Mouse é um mouse óptico de alta qualidade com uma superfície lisa. O mouse está equipado com um
2 sensor de 1000 DPI e uma taxa de sondagem de 1000 Hz. O mouse está disponível em preto e branco.
```

Prompt:

Crie uma descrição detalhada para o seguinte produto: Hoover Lightspeed, pertencente à categoria: Aspirador de pó sem fio

Resposta:

```
1 O Hoover Lightspeed é um aspirador de pó sem fio equipado com uma bateria de íons de lítio. A bateria é carregada por
2 meio de um cabo USB. O aspirador de pó está equipado com uma escova 2 em 1 e uma escova motorizada. A escova é usada
3 para limpar pisos duros, e a escova motorizada é usada para limpar tapetes. O aspirador de pó está equipado com um
4 recipiente de pó que pode ser esvaziado por meio de um recipiente de pó. O aspirador está equipado com um visor LED
5 que mostra a capacidade restante da bateria.
```

A mesma falta de detalhes e falhas lógicas nos detalhes onde os detalhes estão disponíveis persiste. Se esse modelo com ajuste fino for usado para a geração de descrições de produtos em um cenário real, esse resultado não será aceitável.

COMBINAÇÃO DE HIPERPARÂMETROS N° 3: QLoRA com r=8 e visando todas as camadas lineares

Dado que dobrar r aparentemente não resulta em nenhum aumento perceptível na qualidade da saída, vale a pena trocar o outro botão importante, ou seja, direcionar todas as camadas lineares em vez de apenas os blocos de atenção. Aqui, os hiperparâmetros LoRa são r=8 e target_layers são 'q_proj','k_proj','v_proj','o_proj','gate_proj','down_proj','up_proj' e 'lm_head'. Isso aumenta o número de parâmetros atualizados para 12.994.560 e aumenta o tempo de treinamento para cerca de 15,5 minutos.

Avaliação qualitativa

Solicitar ao modelo os mesmos prompts produz o seguinte:

Prompt:

Crie uma descrição detalhada para o seguinte produto: Corelogic Smooth Mouse, pertencente à categoria: Mouse óptico

Resposta:

```
1 O Corelogic Smooth Mouse é um mouse óptico sem fio com uma superfície lisa. O mouse está equipado com um sensor de
2 1000 DPI e uma taxa de sondagem de 1000 Hz. O mouse está equipado com uma conexão sem fio de 2,4 GHz e um receptor
3 USB. O mouse está disponível em preto e branco.
```

Prompt:

Crie uma descrição detalhada para o seguinte produto: Hoover Lightspeed, pertencente à categoria: Aspirador de pó sem fio

Resposta:

```
1 O aspirador sem fio Hoover Lightspeed é um dispositivo de alto desempenho e fácil de usar. O dispositivo é equipado
2 com uma bateria de íons de lítio que fornece até 40 minutos de tempo de execução. O dispositivo é equipado com um mo-
3 tor de alto desempenho que fornece uma potência de sucção de até 200 watts. O dispositivo vem equipado com uma esco-
4 va de limpeza 2 em 1 e uma ferramenta para frestas. O dispositivo está equipado com um visor LED que mostra a carga
restante da bateria.
```

Agora é possível ver uma descrição coerente um pouco mais longa do mouse óptico fictício, e não há falhas lógicas na descrição do aspirador de pó. As descrições dos produtos não são apenas lógicas, mas relevantes. Apenas como um lembrete, esses resultados de qualidade relativamente alta são obtidos ajustando menos de 1% dos pesos do modelo com um conjunto de dados total de 5.000 desses pares de descrição rápida formatados de maneira consistente.

COMBINAÇÃO DE HIPERPARÂMETROS N° 4: LoRA com r=8 e visando todas as camadas de transformadores lineares

Também vale a pena explorar se a qualidade da saída do modelo melhora se o modelo pré-treinado for congelado em 8 bits em vez de 4 bits. Em outras palavras, replicar o processo exato de ajuste fino usando o LoRa em vez do QLoRa. Aqui, os hiperparâmetros do LoRA são mantidos os mesmos de antes, na configuração otimizada recém-descoberta, ou seja, r=8 e com foco em todas as camadas lineares do transformador durante o processo de adaptação.

Avaliação qualitativa

Os resultados dos dois prompts usados em todo o artigo são os seguintes:

Prompt:

Crie uma descrição detalhada para o seguinte produto: Corelogic Smooth Mouse, pertencente à categoria: Mouse óptico

Resposta:

```
1 O Corelogic Smooth Mouse é um mouse óptico de alta qualidade com uma superfície lisa. O mouse está equipado com um
2 sensor de 1000 DPI e uma taxa de sondagem de 1000 Hz. O mouse está disponível em três cores: preto, branco e vermel-
3 ho. O mouse está equipado com um receptor USB e é compatível com Windows e Mac.
```

Prompt:

Crie uma descrição detalhada para o seguinte produto: Hoover Lightspeed, pertencente à categoria: Aspirador de pó sem fio

Resposta:

```
1 O aspirador de pó sem fio Hoover Lightspeed é um dispositivo compacto, leve e fácil de usar. O dispositivo é equipado
2 com uma bateria de íons de lítio que fornece até 40 minutos de tempo de limpeza. O aspirador de pó está equipado com
3 um filtro de alto desempenho, que garante que o ar seja limpo de poeira e alérgenos. O dispositivo está equipado com
4 uma escova de pó 2 em 1 e uma ferramenta para frestas, que podem ser usadas para limpar áreas de difícil acesso.
```

Novamente, não há muita melhoria na qualidade do texto de saída.

PRINCIPAIS OBSERVAÇÕES

Com base no conjunto de testes acima e em outras evidências detalhadas na excelente publicação que apresenta o QLoRA, pode-se deduzir que o valor de r (a classificação das matrizes atualizadas durante a adaptação) não melhora a qualidade da adaptação além de um determinado ponto. A maior melhoria é observada no direcionamento de todas as camadas lineares no processo de adaptação, em oposição a apenas os blocos de atenção, conforme comumente documentado na literatura técnica detalhando o LoRa e o QLoRa. Os ensaios executados acima e outras evidências empíricas sugerem que o QLoRa não sofre de fato nenhuma redução perceptível na qualidade do texto gerado, em comparação com o LoRa.

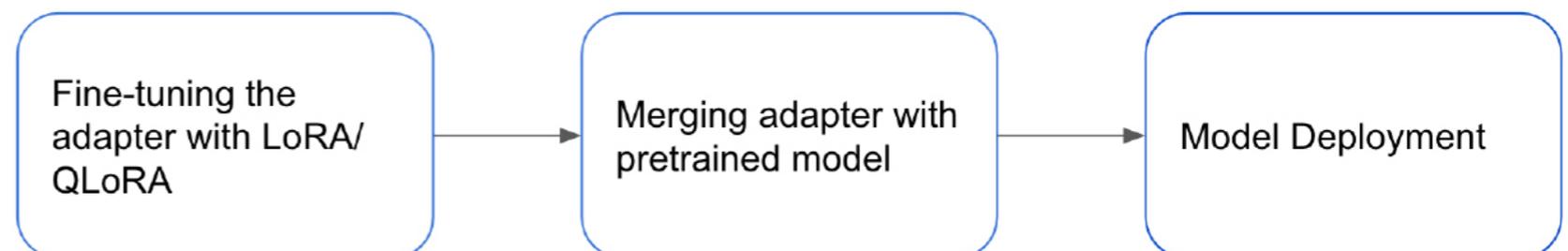
CONSIDERAÇÕES ADICIONAIS SOBRE O USO DE ADAPTADORES LORA NA IMPLANTAÇÃO

É importante otimizar o uso de adaptadores e entender as limitações da técnica. O tamanho do adaptador LoRa obtido por meio do ajuste fino normalmente é de apenas alguns megabytes, enquanto o modelo básico pré-treinado pode ter vários gigabytes na memória e no disco. Durante a inferência, tanto o adaptador quanto o LLM pré-treinado precisam ser carregados, de modo que o requisito de memória permanece semelhante.

Além disso, se os pesos do LLM pré-treinado e do adaptador não forem fundidos, haverá um pequeno aumento na latência da inferência. Felizmente, com a biblioteca PEFT, o processo de fundir os pesos com o adaptador pode ser feito com uma única linha de código, conforme mostrado aqui:

```
1 merged_model = peft_model.merge_and_unload()
```

A figura abaixo descreve o processo desde o ajuste fino de um adaptador até a implantação do modelo.



Embora o padrão de adaptador ofereça benefícios significativos, a fusão de adaptadores não é uma solução universal. Uma vantagem do padrão de adaptador é a capacidade de implantar um único modelo grande pré-treinado com adaptadores específicos de tarefas. Isso permite uma inferência eficiente, utilizando o modelo pré-treinado como uma espinha dorsal para diferentes tarefas. Entretanto, a fusão de pesos torna essa abordagem impossível. A decisão de fundir pesos depende do caso de uso específico e da latência de inferência aceitável. No entanto, o LoRA/QLoRa continua sendo um método altamente eficaz para ajuste fino eficiente de parâmetros e é amplamente utilizado.

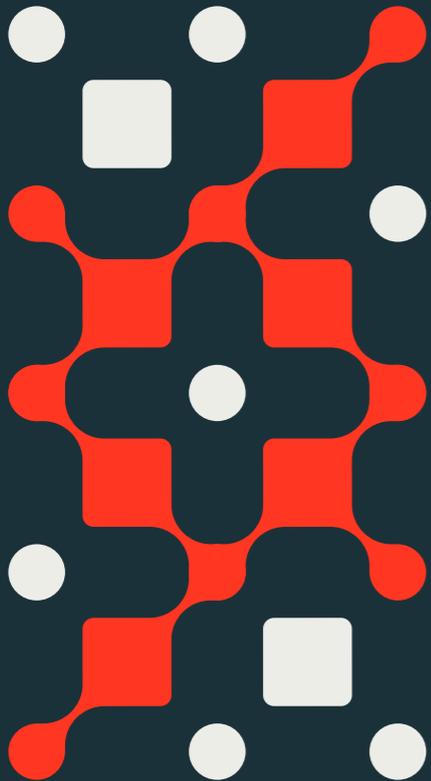
CONCLUSÃO

A adaptação de baixa classificação é uma técnica avançada de ajuste fino que pode gerar ótimos resultados se usada com a configuração correta. A escolha do valor correto da classificação e das camadas da arquitetura da rede neural a serem visadas durante a adaptação pode decidir a qualidade da saída do modelo ajustado. O QLoRa resulta em mais economia de memória, preservando a qualidade da adaptação. Mesmo quando o ajuste fino é realizado, há várias considerações de engenharia importantes para garantir que o modelo adaptado seja implantado corretamente.

Para resumir, a tabela abaixo indica de forma sucinta as várias combinações de parâmetros LoRA experimentadas, a qualidade da saída de texto e o número de parâmetros atualizados durante o ajuste fino 90 do OpenLLaMA-3b-v2, realizado por 3 épocas em 5000 observações em uma única A100.

R	TARGET_MODULES	PESOS DO MODELO BASE	QUALIDADE DA PRODUÇÃO	NÚMERO DE PARÂMETROS ATUALIZADOS (EM MILHÕES)
8	Bloqueios de atenção	4	baixo	2,662
16	Bloqueios de atenção	4	baixo	5,324
8	Todas as camadas lineares	4	alto	12,995
8	Todas as camadas lineares	8	alto	12,995

Experimente isso na Databricks! Clone o [repositório do GitHub](#) associado ao blog em um [repositório](#) da Databricks para começar. Exemplos mais detalhados sobre como ajustar modelos na Databricks estão disponíveis [aqui](#).



Estágio 4: Pré-treinamento

Pré-treinar um modelo do zero refere-se ao processo de treinamento de um modelo de linguagem em um grande corpus de dados (por exemplo, texto ou código) sem usar nenhum conhecimento ou peso prévio de um modelo existente. É o oposto do ajuste fino, no qual um modelo já pré-treinado é ainda mais adaptado a uma tarefa ou conjunto de dados específico. O resultado do pré-treinamento completo é um modelo básico que pode ser usado diretamente ou ajustado para tarefas downstream.

QUANDO USAR O PRÉ-TREINAMENTO

Optar por pré-treinar um LLM do zero é um compromisso significativo, tanto em termos de dados quanto de recursos computacionais. Aqui estão alguns cenários em que isso faz sentido:

- 1. Fontes de dados exclusivas:** Se você possui um corpus de dados único e extenso, distinto do que os LLMs pré-treinados disponíveis já processaram, pode valer a pena pré-treinar um modelo para capturar essa singularidade.
- 2. Especificidade do domínio:** as organizações podem querer um modelo básico adaptado ao seu domínio específico (por exemplo, médico, jurídico, código) para garantir que até mesmo o conhecimento base do modelo seja específico do domínio
- 3. Controle total sobre os dados de treinamento:** o pré-treinamento a partir do zero oferece transparência e controle sobre os dados nos quais o modelo é treinado. Isso pode ser essencial para garantir a segurança dos dados, privacidade e adaptação personalizada do conhecimento fundamental do modelo.
- 4. Evitar vieses de terceiros:** o pré-treinamento garante que sua aplicação de LLM não herde vieses ou limitações de modelos pré-treinados de terceiros.

PRÉ-TREINAMENTO NA PRÁTICA

Considerando o caráter intensivo em recursos do pré-treinamento, é preciso um planejamento detalhado e o uso de ferramentas sofisticadas. Bibliotecas como **PyTorch FSDP** e **Deepspeed**, mencionadas na **seção de ajuste fino**, são igualmente necessárias para seus recursos de treinamento distribuído ao pré-treinar um LLM a partir do zero. A seguir, apresentamos apenas uma introdução a algumas das considerações que precisam ser analisadas ao pré-treinar um LLM:

- **Pré-processamento de dados em grande escala:** um modelo pré-treinado é tão bom quanto os dados nos quais ele é treinado. Assim, torna-se de vital importância garantir que o pré-processamento robusto de dados seja realizado antes do treinamento do modelo. Dada a escala dos dados de treinamento envolvidos, esse pré-processamento normalmente requer estruturas distribuídas como o **Apache Spark™**. Deve-se considerar fatores como combinação de conjuntos de dados e técnicas de eliminação de duplicação para garantir que o modelo seja exposto a uma ampla variedade de pontos de dados exclusivos.
- **Seleção e ajuste de hiperparâmetros:** antes de executar o treinamento em grande escala de um LLM, determinar o conjunto de hiperparâmetros ideais é crucial. Dado o alto custo computacional associado ao treinamento de LLM, varreduras extensivas de hiperparâmetros nem sempre são viáveis. Em vez disso, decisões informadas com base em pesquisas em menor escala ou pesquisas anteriores são empregadas. Depois que um conjunto promissor é identificado, esses hiperparâmetros são usados para a execução completa do treinamento. Ferramentas como o **MLflow** são essenciais para gerenciar e rastrear esses experimentos.
- **Maximizando a utilização de recursos:** Dado o alto custo associado a longas tarefas distribuídas de treinamento em GPU, é extremamente importante maximizar a utilização de recursos. A biblioteca **composer** da MosaicML é um exemplo que utiliza **PyTorch FSDP** com otimizações adicionais para maximizar a **Utilização de FLOPs do Modelo (MFU)** e **Utilização de FLOPs de Hardware (HFU)** durante o treinamento.
- **Lidar com falhas de GPU:** o treinamento de grandes modelos pode durar dias ou até semanas. Durante esse treinamento em larga escala por esse período de tempo, falhas de hardware, especialmente falhas de GPU, podem ocorrer (e normalmente ocorrem). É essencial ter mecanismos em vigor para lidar com essas falhas sem dificuldades.
- **Monitoramento e avaliação:** o monitoramento rigoroso do processo de treinamento é essencial. Salvar pontos de verificação de modelo regularmente e avaliar conjuntos de validação não apenas atuam como proteções, mas também fornecem insights sobre o desempenho do modelo e as tendências de convergência.

Nos casos em que é necessário pré-treinar um LLM a partir do zero, o [Mosaic AI Training](#) fornece uma plataforma para realizar o treinamento de modelos multibilionários de parâmetros de maneira altamente otimizada e automatizada. Lidar automaticamente com falhas de GPU e retomar o treinamento sem intervenção humana e aproveitar o [Mosaic AI Streaming](#) para streaming eficiente de dados no processo de treinamento são apenas alguns dos recursos fornecidos imediatamente.

O valor de treinar modelos a partir do zero na Databricks

Depois de mergulhar nos detalhes de como começar o treinamento de um modelo a partir do zero, por que você pode fazer isso e as ferramentas avançadas necessárias, vamos dar uma olhada em um exemplo do mundo real para mostrar que treinar modelos de linguagem de alto nível não é tão complexo ou caro quanto possa parecer. Essa mudança destaca que até mesmo organizações que controlam seu orçamento podem começar a treinar seus próprios modelos, com a Databricks fornecendo o suporte e a infraestrutura necessários. A Databricks se destaca como excepcionalmente capaz de ajudar os clientes a treinar seus próprios modelos a partir do zero, permitindo que eles sejam donos completos de seus ativos de IA.

Casos de uso de pré-treinamento

[Treine Stable Diffusion a partir do zero por menos de US\\$ 50 mil com o MosaicML](#)

por [Mihir Patel](#), [Cory Stephenson](#), [Landan Seguin](#), [Austin Jacobson](#) e [Erica Ji Yuen](#)

Nós replicamos o Stable Diffusion 2 por menos de US\$ 50 mil e disponibilizamos o código de treinamento de código aberto para que você também possa! Essa é uma redução de custo de 3x em relação à nossa última postagem do blog, e uma redução de 8x em relação ao Stable Diffusion 2 original, tornando o treinamento de modelos de difusão em grande escala a partir do zero mais acessível do que nunca.

Hoje, estamos empolgados em mostrar os resultados de nosso próprio treinamento: menos de US\$ 50 mil para treinar o Stable Diffusion 2 base1 a partir do zero em 7,45 dias usando a [plataforma MosaicML](#).



Figura 1: imaginando a alta costura de micélio. Integrar a geração de imagens ao processo de projeto expande os limites criativos. Todas as imagens neste mood board foram criadas com nosso modelo de difusão interna treinado a partir do zero na plataforma MosaicML.

Treinar seu próprio modelo de geração de imagens com seus próprios dados agora é fácil e acessível. Ao treinar seus próprios modelos de difusão, você pode:

- Usar seus dados proprietários
- Ajustar as representações para certos estilos de arte ou fotografia
- Evitar violar as leis de propriedade intelectual para que seus modelos possam ser usados comercialmente

Abrimos o código e os métodos para treinar um modelo de difusão a partir do zero para que você possa treinar o seu próprio; confira [aqui!](#) Se você estiver interessado em treinar seus próprios modelos, [entre em contato conosco para uma demonstração](#) e continue lendo para saber mais sobre nossa configuração de engenharia!

CONFIGURAÇÃO

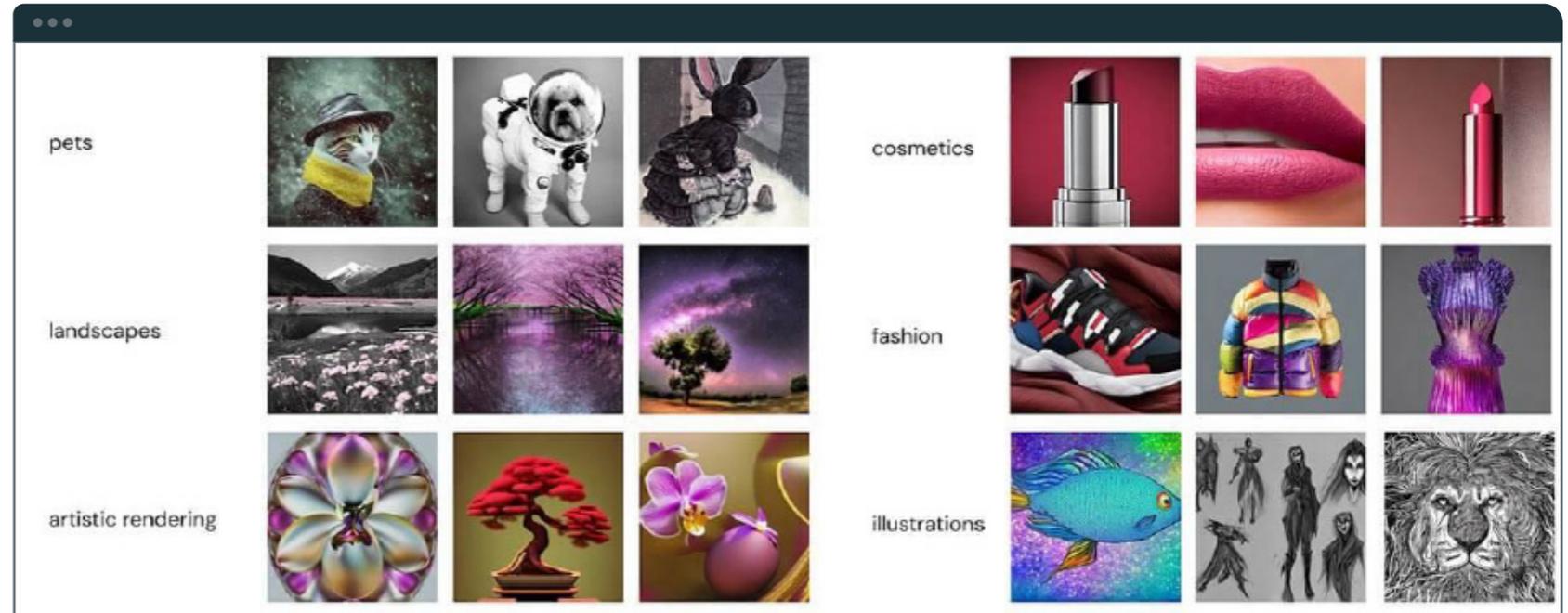


Figura 2: sendo criativo e abraçando a serendipidade. Uma variedade de temas, estilos de arte e fotografia são gerados pelo nosso modelo de difusão.

Modelo: nosso modelo de difusão é um **ComposerModel** composto por um Variational Autoencoder (VAE), um modelo CLIP, um U-Net e um diffusion noise scheduler, todos da biblioteca Diffusers do HuggingFace. Todas as configurações do modelo foram baseadas em [stabilityai/stable-diffusion-2-base](#).

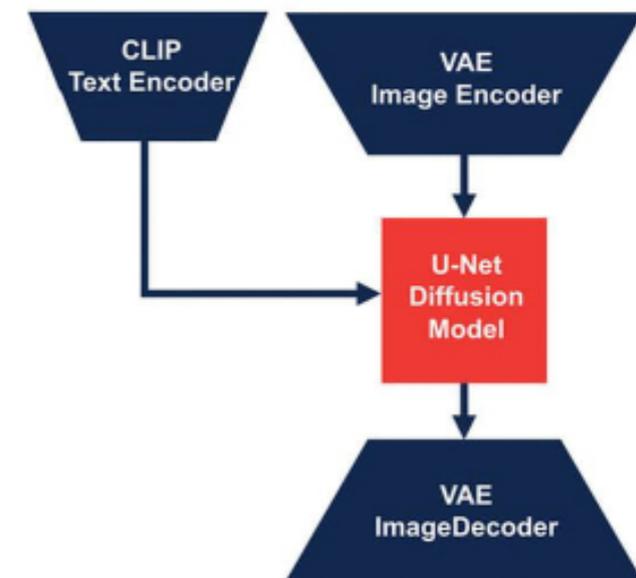


Figura 3: diagrama simplificado do modelo de difusão.

Dados: Treinamos em um **subconjunto do LAION-5B** que inclui amostras com legendas apenas em inglês e uma pontuação estética de 4.5+. Similar ao modelo base do Stable Diffusion 2, fizemos duas fases de treinamento com base na resolução das imagens dos dados de treinamento. Na primeira fase, usamos todas as imagens com resolução $\geq 256 \times 256$, totalizando 790 milhões de amostras de imagem-legenda. Na segunda fase, usamos apenas imagens com resolução $\geq 512 \times 512$, totalizando 300 milhões de amostras de imagem-legenda.

Compute: ambas as fases de treinamento foram executadas em 128 GPUs NVIDIA A100. A primeira fase de treinamento foi executada para 550 mil iterações em 1,6 dia, enquanto a segunda fase foi executada para 850 mil iterações em 4,9 dias, totalizando 20.051 horas de treinamento na A100. Além do tempo de treinamento, pré-computamos as latentes para o modelo VAE e CLIP para reduzir o tempo e o custo do treinamento ao fazer várias passagens pelo conjunto de dados. O pré-cálculo das latentes exigiu um adicional de 3.784 horas da A100, resultando em 23.835 horas da A100 no total. Considerando um custo de US\$ 2 por hora da A100, o preço total é de US\$ 47,7 mil.

Tech Stack: usamos o **Composer** para nossa estrutura de treinamento, o **StreamingDataset** para carregar nossos 100 TB de dados e a **plataforma MosaicML** para superar desafios de infraestrutura ao treinar e avaliar em 128 GPUs.

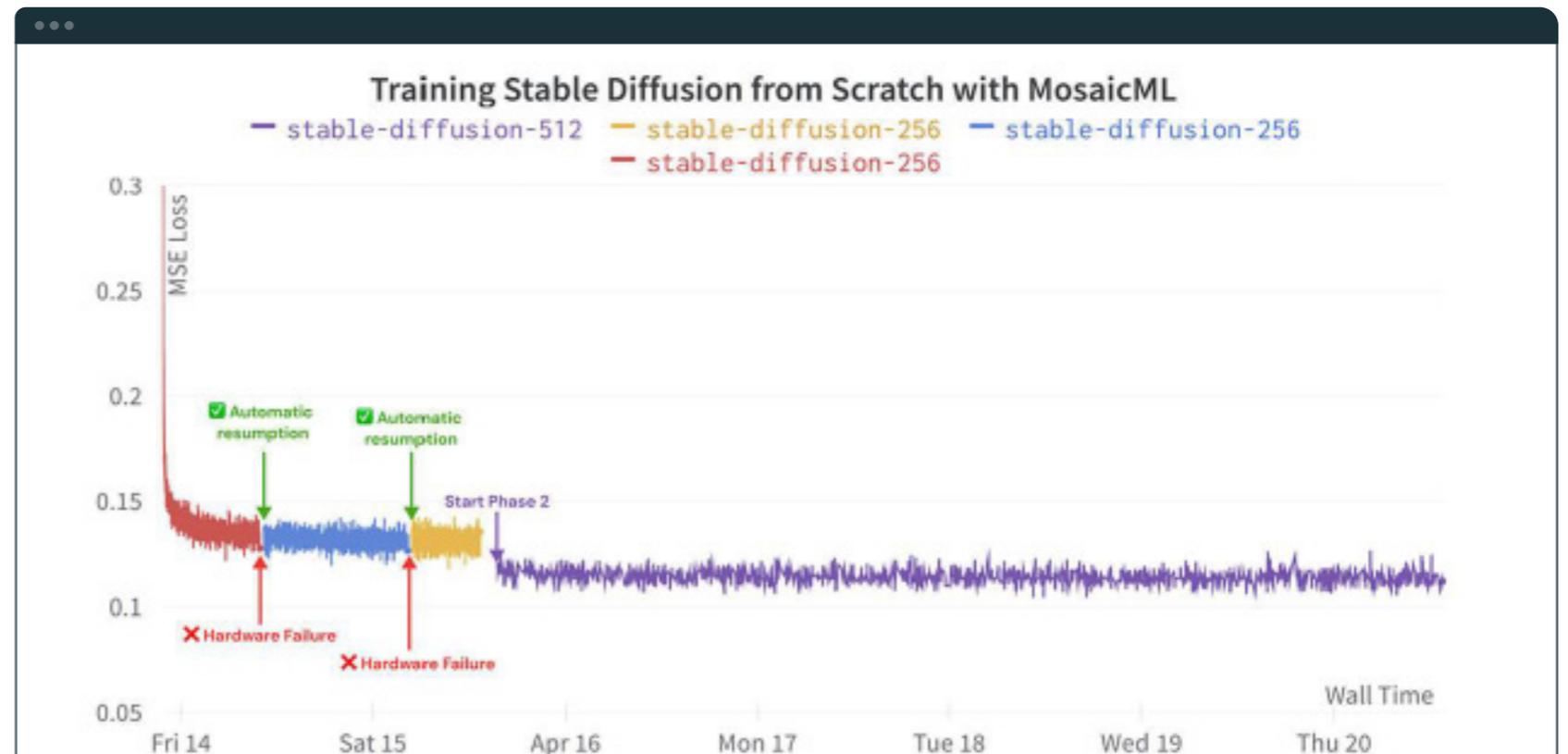


Figura 4: curva de perda para nossa execução de treinamento. Nossa plataforma capturou duas falhas de hardware e reiniciou a execução automaticamente sem intervenção humana. A descontinuidade de perda ocorre porque a fase 2 aumenta a resolução de 256x256 para 512x512.

DESAFIOS E SOLUÇÕES

Seja para modelos de difusão ou grandes modelos de linguagem, o treinamento em escala tem desafios significativos. Treinamos nosso modelo de difusão usando a plataforma MosaicML, que aborda esses desafios automaticamente para que você possa se concentrar no treinamento do melhor modelo possível. Abaixo estão três desafios principais com o treinamento em grande escala e como nossa plataforma os resolve.

INFRAESTRUTURA

O treinamento de grandes modelos em grandes conjuntos de dados exige uma computação significativa. A plataforma MosaicML orquestra sem esforço centenas de GPUs em qualquer provedor de nuvem. Por exemplo, nosso treinamento primário ocorreu em um cluster de 128 GPUs A100. Para garantir que a avaliação do modelo não atrasasse o treinamento, iniciamos automaticamente as execuções de avaliação em cada ponto de verificação em diferentes clusters usando diferentes provedores de nuvem, escalando perfeitamente até 64 GPUs e reduzindo para 8 GPUs, dependendo da disponibilidade.

Mesmo após o treinamento estar em andamento, falhas de software ou hardware podem interromper o treinamento, deixando as GPUs ociosas até que alguém perceba ou exigindo que alguém de plantão 24 horas por dia, sete dias por semana, fique de olho na execução. Felizmente, os recursos Node Doctor e Watchdog da plataforma MosaicML detectam automaticamente os nós com falha e retomam os trabalhos conforme a necessidade. Com a retomada automática, recuperamos as falhas e continuamos o treinamento sem nenhuma intervenção humana, evitando downtime dispendioso e babás humanas. Basta lançar e treinar!

SOFTWARE EFICIENTE

O software é difícil de configurar de maneira ideal. Nossa biblioteca do Composer **baseada em PyTorch** maximiza a eficiência do treinamento em escala. Conforme mostrado em nossa **postagem anterior no blog**, o Composer demonstrou excelente escalonamento de throughput à medida que o número de GPUs aumentava. Para esta atualização, adicionamos mais otimizações (Low Precision **GroupNorm** e Low Precision **LayerNorm**, Fully Sharded Data Parallel) para alcançar uma ampliação forte quase perfeita de até 128 GPUs, reduzindo o custo para US\$ 50 mil. Também usamos o algoritmo Exponential Moving Average (EMA) nativo do Composer, que nos permitiu iniciar o EMA perto do final do treinamento (iteração de 800 k da fase final) para obter todos os benefícios do EMA, economizando memória e computação para a maior parte do treinamento.

GERENCIAMENTO DE 100 TB DE DADOS

Treinamos com um subconjunto do LAION-5B que continha 790 milhões de amostras, totalizando mais de 100TB de dados. O tamanho gigantesco do conjunto de dados torna o gerenciamento difícil, especialmente ao trabalhar com vários clusters com armazenamento local separado. A biblioteca [StreamingDataset da MosaicML](#) torna o trabalho com conjuntos de dados massivos muito mais simples e rápido. Três recursos principais da biblioteca StreamingDataset foram especialmente úteis para essa execução de treinamento:

1. Mistura de conjuntos de dados armazenados em locais diferentes. Separamos as amostras com base na resolução da imagem em diferentes conjuntos de dados. No momento do treinamento, usamos a biblioteca MosaicML StreamingDataset para treinar em uma mistura de resoluções desses conjuntos de dados.
2. Retomada instantânea no meio da época. Conseguimos retomar instantaneamente o treinamento no meio de uma época. Isso economizou horas, evitando a necessidade de iterar todo o conjunto de dados para voltar ao ponto de partida.
3. Determinismo elástico. A biblioteca MosaicML StreamingDataset embaralha os dados de forma determinística, mesmo ao alterar o número de GPUs usadas para treinamento. Isso nos possibilitou reproduzir exatamente as execuções de treinamento, simplificando drasticamente a depuração.

RESULTADOS DA AVALIAÇÃO HUMANA

Avaliar modelos de geração de imagens é difícil, e não há substituto para a avaliação humana. Em uma avaliação humana cega, medimos as preferências do usuário na qualidade da imagem e no alinhamento imediato entre o Stable Diffusion 2 e nosso modelo de difusão. Com base nas preferências do usuário, concluímos que os dois modelos eram comparáveis em qualidade (ver Figura 5) Todas as imagens foram geradas com base em prompts do benchmark Drawbench proposto no [artigo da Imagen](#). Para saber mais, aguarde o nosso post de blog que será publicado em breve.

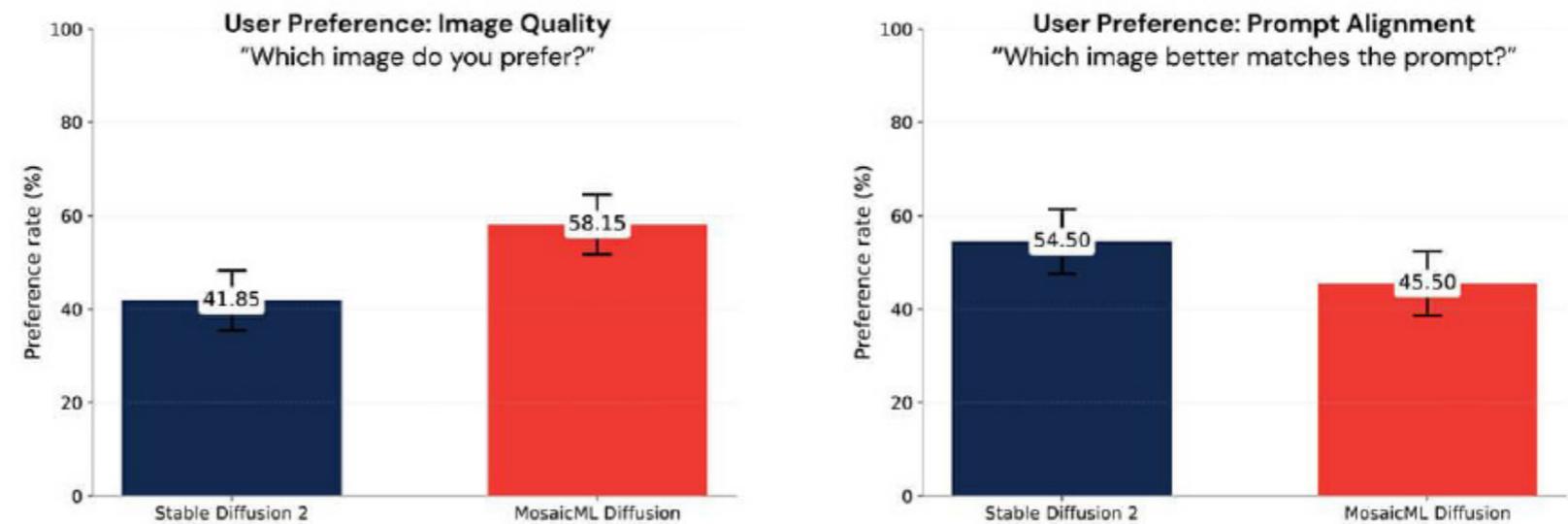


Figura 5: resultados de nossa avaliação humana da qualidade das imagens (esquerda) e alinhamento de prompts (direita). As barras de erro mostram intervalos de confiança de 95%. Em ambos os experimentos, a diferença nas taxas de preferência do usuário entre os dois modelos foi comparável à incerteza na medição; portanto, concluímos que os dois modelos são de qualidade geral comparável.

Mergulho profundo: como treinamos a difusão estável por menos de US\$ 50 mil

por [Mihir Patel](#), [Erica Ji Yuen](#), [Cory Stephenson](#) e [Landan Seguin](#)

Em nosso exemplo anterior, mostramos como usamos a plataforma MosaicML, os conjuntos de dados de streaming e a biblioteca do Composer para treinar um modelo de difusão estável a partir do zero por menos de US\$ 50.000. Agora, mergulhamos profundamente nos detalhes técnicos por trás dessa aceleração, demonstrando como conseguimos replicar o modelo básico do Stable Diffusion 2 em apenas 6,8 dias.

Experimente nosso código [aqui!](#)

Muitas organizações precisam de modelos de IA grandes e de alto desempenho, adaptados aos seus casos de uso específicos. No entanto, o treinamento desses modelos costuma ser proibitivamente demorado e caro, exigindo grandes quantidades de computação e conhecimento especializado. É aqui que a MosaicML entra: oferecemos uma solução abrangente que simplifica e acelera o processo de treinamento desses modelos.

No nosso [post anterior no blog](#), anunciamos que treinamos um modelo de difusão comparável ao Stable Diffusion 2 do zero por \$47,7 mil. Neste post, mergulhamos nos detalhes técnicos para destacar como alcançamos uma redução de 8x em tempo/custo em comparação com [o valor relatado pela StabilityAI](#) e uma redução de 3x no custo em relação ao [nosso próprio ponto de referência](#). Todo o nosso código é [aberto](#) e fácil de modificar para casos de uso personalizados. Se você quiser saber mais sobre nossa tecnologia, por favor [entre em contato para uma demonstração](#).

ACELERAÇÃO DO TREINAMENTO

Stable Diffusion 2 Model Architecture (before speedups)

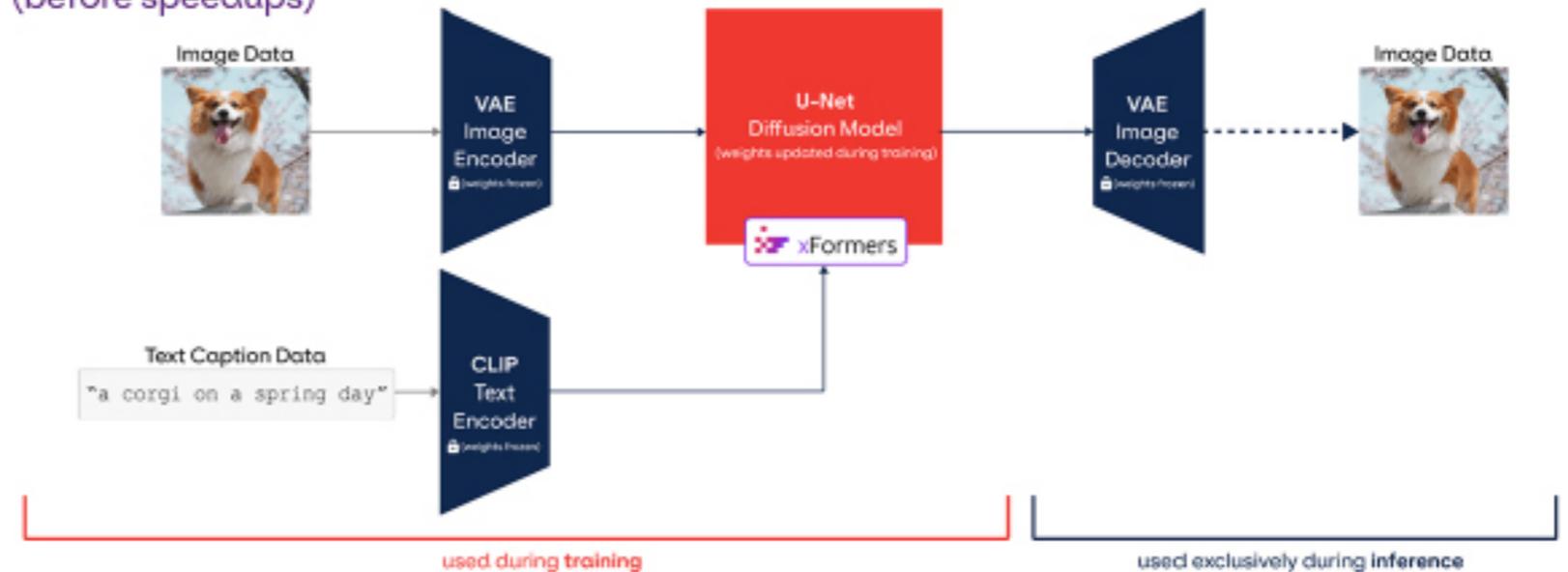
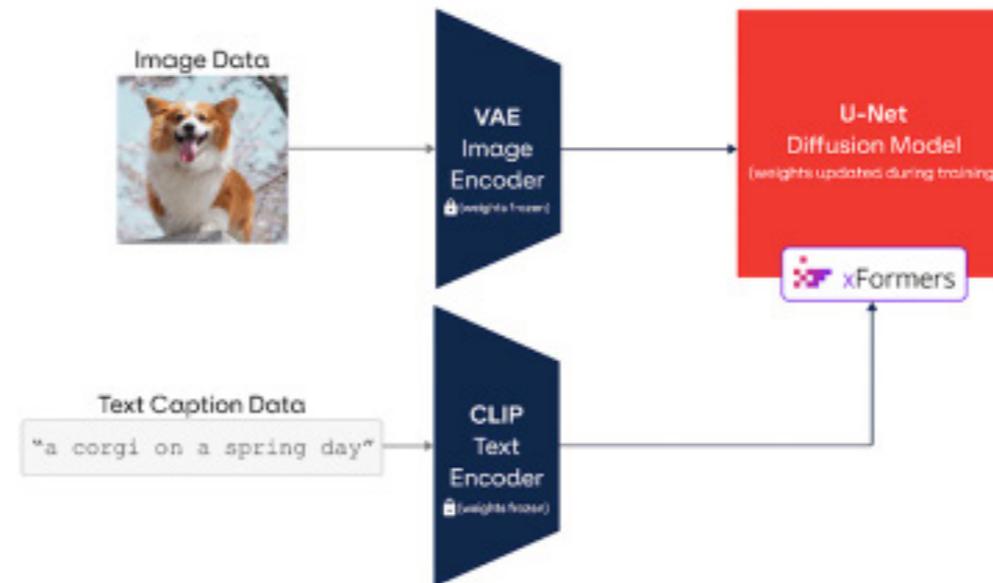


Figura 1: arquitetura do modelo Stable Diffusion 2. Durante o treinamento, o codificador de imagem VAE, o codificador de texto CLIP e o U-Net são usados. Para inferência, são utilizados o codificador de texto CLIP, o U-Net e o decodificador de imagem VAE. Apenas os pesos do U-Net são atualizados durante o treinamento; os componentes CLIP e VAE permanecem fixos.

Introduzimos uma variedade de técnicas, de fusões a estratégias de fragmentação, que aceleram drasticamente o treinamento e reduzem os custos em quase 3x.

XFORMERS FLASHATTENTION

Diffusion Speedup #1: xFormers FlashAttention



▲ Cumulative Throughput Boost Factor: **1.18x**

▲ Speedups:

xFormers FlashAttention

Precomputing Latents

Low Precision LayerNorm & GroupNorm

Fully Sharded Data Parallelism (FSDP)

Scheduled Exponential Moving Average (EMA)

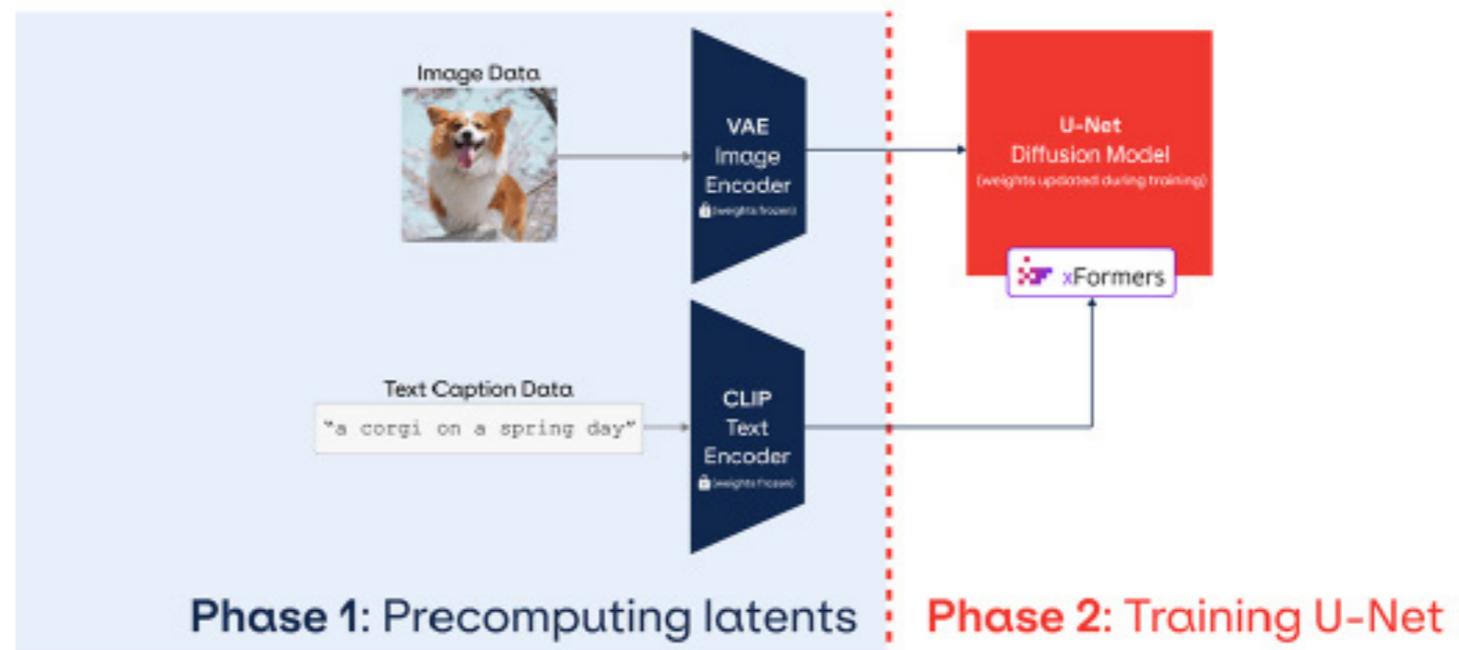
Figura 2: o xFormers acelera bloqueios de atenção cruzada no U-Net.

As camadas de atenção na arquitetura Stable Diffusion podem ser lentas com uma implementação ingênua; portanto, a maioria das bases de código usa implementações mais rápidas que dependem de kernels fundidos. Em nossa stack, aproveitamos o **xFormers FlashAttention**.

Embora isso tenha sido habilitado em nosso **post original no blog**, encontramos um problema no uso que resultava em consumo extra de memória no rank 0. Após corrigir esse bug, conseguimos aumentar o tamanho do microbatch do dispositivo de 4 para 8. Isso gerou um aumento significativo de velocidade, pois as A100s são mais eficientes com tamanhos de matriz maiores.

PRÉ-CÁLCULO DE LATENTES

Diffusion Speedup #2: Precomputing Latents



▲ Cumulative Throughput Boost Factor: **1.69x**

▲ Speedups:

xFormers FlashAttention

Precomputing Latents

Low Precision LayerNorm & GroupNorm

Fully Sharded Data Parallelism (FSDP)

Scheduled Exponential Moving Average (EMA)

Figura 3: treinamento em duas fases com latentes pré-computados. Primeiro, os latentes do VAE e CLIP são pré-computados e armazenados. Depois, o modelo de difusão U-Net é treinado com esses latentes..

O Stable Diffusion é uma combinação de três modelos: um codificador automático variacional (VAE), um codificador de texto (CLIP) e um U-Net. Durante o treinamento de difusão, somente o U-Net é treinado, e os outros dois modelos são usados para computar as codificações latentes das entradas de imagens e texto. O treinamento padrão envolve o cálculo das latentes VAE e CLIP para cada batch, mas isso realiza muito trabalho duplicado ao treinar para várias épocas: as latentes são recalculadas para cada imagem toda vez que ela é usada. Em vez disso, pré-calculamos as latentes uma vez antes do treinamento. Empiricamente, temos duas épocas com resolução de 256 e cinco épocas com resolução de 512; portanto, evitamos seis chamadas VAE e CLIP extras por par imagem-texto no conjunto de dados.

Além disso, ao pré-calcular as latentes, podemos reduzir a precisão dos modelos VAE e CLIP para fp16. Isso poderia levar à instabilidade numérica se estivéssemos treinando o VAE e o CLIP e usássemos essa precisão para a passagem para trás. No entanto, como os estamos usando apenas para inferência, podemos reduzir a precisão com segurança, o que aumenta a velocidade. A economia extra de memória também nos permite usar tamanhos de batches muito maiores e melhorar a utilização do hardware durante a pré-computação de latentes.

LOW PRECISION LAYERNORM E GROUPOUNORM

Diffusion Speedup #3: Low Precision LayerNorm & GroupNorm

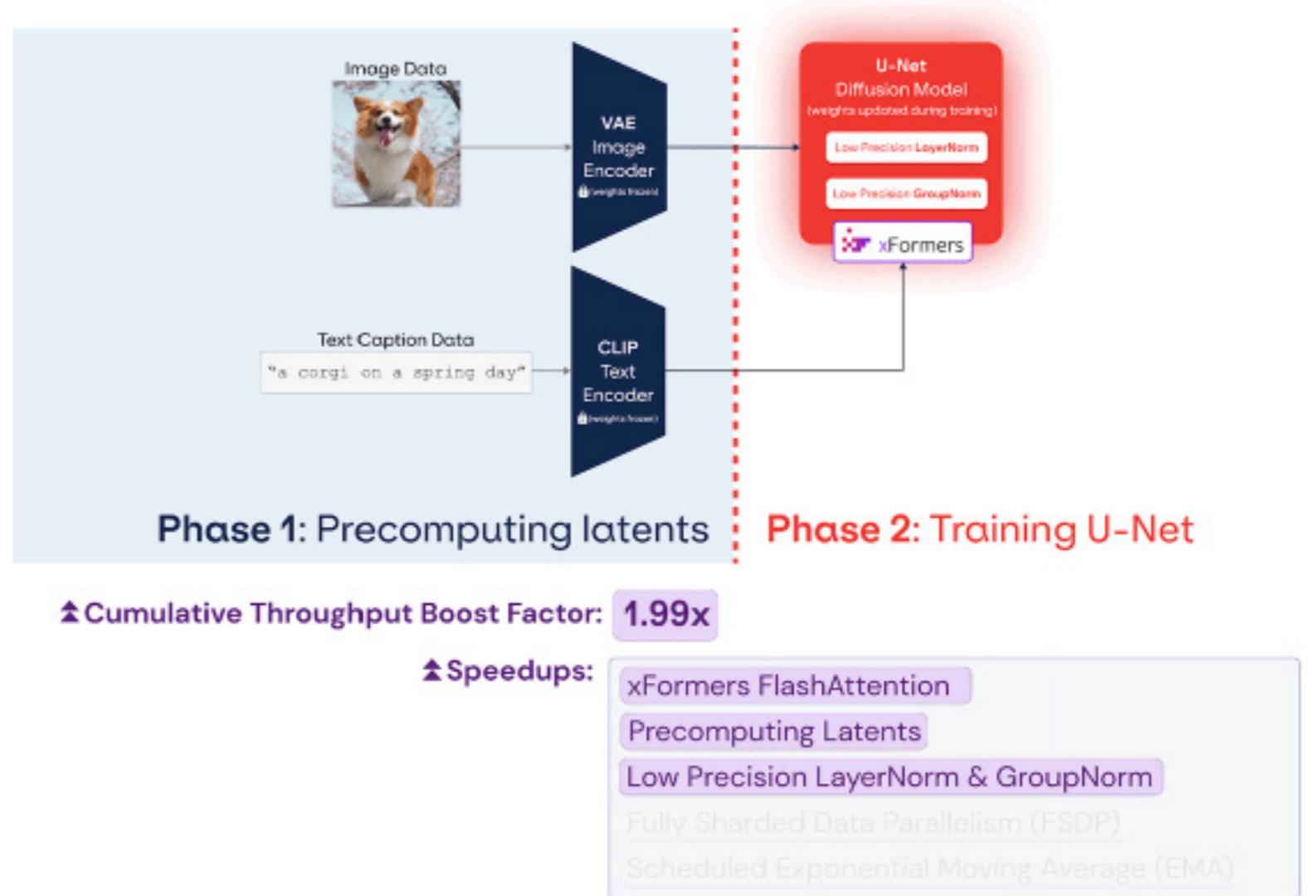


Figura 4: Low Precision LayerNorm e Low Precision GroupNorm. A baixa precisão fornece treinamento mais rápido e menor uso de memória, permitindo microbatches maiores.

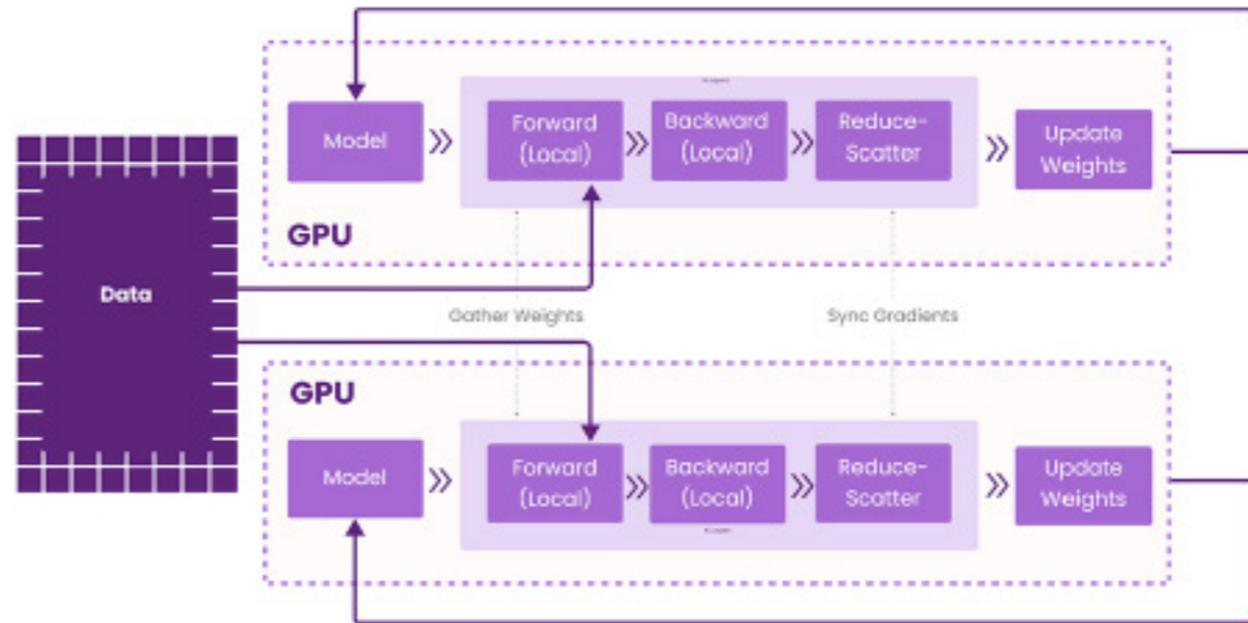
O treinamento de difusão é feito em **precisão mista automática** por padrão. Isso usa meia precisão (fp16) na maioria das camadas, mas fp32 em algumas camadas numericamente instáveis, como normalização e softmax. A arquitetura Stable Diffusion U-Net usa várias camadas LayerNorm e GroupNorm, que, por padrão, são executadas em fp32.

Motivados pela nossa descoberta de que **LayerNorms em meia precisão são seguras para uso em modelos de linguagem**, decidimos testar camadas de LayerNorm e GroupNorm em meia precisão. Essa alteração resultou em curvas de perda idênticas e sem instabilidade em nossos experimentos.

Embora tenhamos observado alguma melhora no throughput, o benefício real foi a redução do uso da memória. Agora, além de remover a memória VAE e CLIP por meio da pré-computação das latentes, temos espaço suficiente em nossa A100 de 40 GB para aumentar o tamanho do microbatch de 8 para 16, quatro vezes maior do que aquele com que começamos!

FULLY SHARDED DATA PARALLELISM

Diffusion Speedup #4: Fully Sharded Data Parallelism (FSDP)



⬆️ Cumulative Throughput Boost Factor: **2.34x**

⬆️ Speedups:

- xFormers FlashAttention
- Precomputing Latents
- Low Precision LayerNorm & GroupNorm
- Fully Sharded Data Parallelism (FSDP)
- Scheduled Exponential Moving Average (EMA)

Figura 5: o Fully Sharded Data Parallel com SHARD_GRAD_OP acelera a etapa de atualização do gradiente e permite o dimensionamento linear.

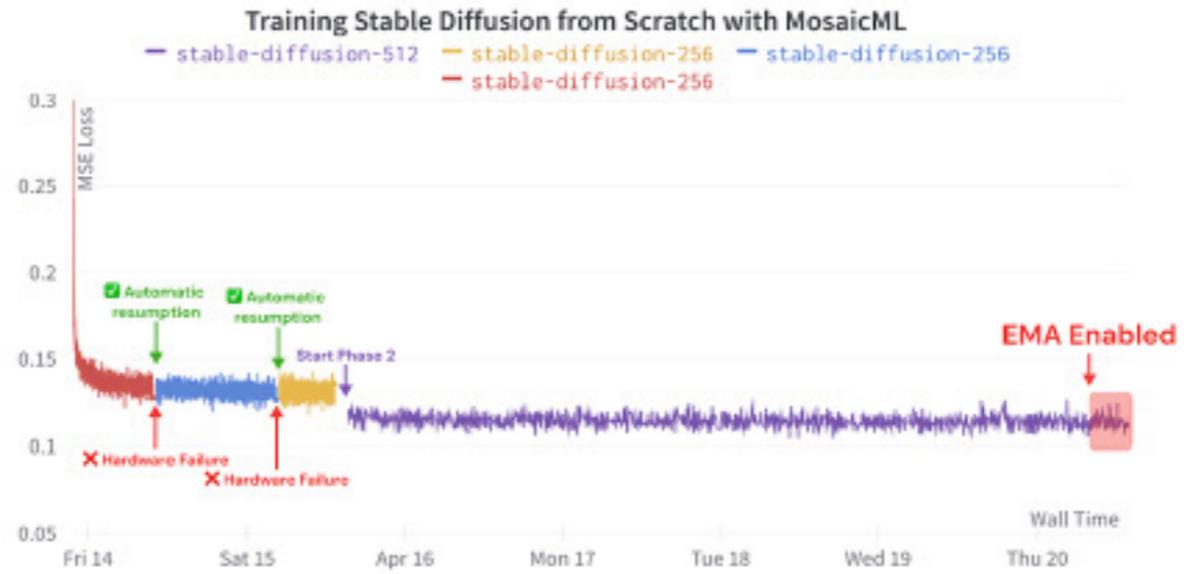
O MosaicML **Composer**, nossa biblioteca de treinamento, inclui compatibilidade com o **PyTorch Fully Sharded Data Parallelism (FSDP)**. Usamos isso principalmente para fragmentar modelos de grande escala, como LLMs de mais de 10 bilhões de parâmetros, que não cabem em um único dispositivo em centenas de GPUs para um treinamento incrivelmente rápido. O Stable Diffusion não requer fragmentação, pois cabe em uma única GPU. No entanto, alguns dos recursos distribuídos no FSDP ainda são úteis para acelerar o treinamento em um grande número de GPUs.

Quando os batches não cabem na memória, fazemos várias passagens para frente e para trás em microbatches menores, seguidos por uma única atualização de gradiente. Se usarmos um pequeno número de GPUs para treinar, teremos muito mais passagens para frente e para trás por atualização de gradiente; então, o tempo gasto na atualização de gradiente não importa. No entanto, com mais de 128 GPUs com um tamanho de microbatch de 16, estamos fazendo apenas uma passagem para frente e uma para trás para cada atualização de gradiente. Nessa escala, a etapa de atualização do gradiente começa a se tornar um gargalo significativo.

Para resolver esse problema, usamos o modo SHARD_GRAD_OP do FSDP. No treinamento normal, cada GPU comunica todos os seus gradientes a todas as outras GPU e, em seguida, cada GPU atualiza sua cópia local do modelo. Com essa variante do FSDP, cada GPU obtém apenas os gradientes e atualiza os pesos de uma pequena parte do modelo antes de enviar os pesos atualizados dessa parte do modelo para todas as outras GPUs. Ao dividir a etapa de atualização em todas as GPUs, podemos garantir que a quantidade de trabalho por GPU diminua à medida que aumentamos o número de GPUs, o que nos ajuda a alcançar a escala linear.

EMA AGENDADA

Diffusion Speedup #5: Scheduled EMA



⬆️ Cumulative Throughput Boost Factor: **2.71x**

⬆️ Speedups:

- xFormers FlashAttention
- Precomputing Latents
- Low Precision LayerNorm & GroupNorm
- Fully Sharded Data Parallelism (FSDP)
- Scheduled Exponential Moving Average (EMA)

Figura 6: curva de perda do nosso treinamento com o período de média móvel exponencial (EMA) programado destacado.

○ Stable Diffusion 2 usa a **média móvel exponencial (EMA)**, que mantém uma média móvel exponencial dos pesos. A cada etapa de tempo, o modelo da EMA é atualizado considerando 0,9999 vezes o modelo atual da EMA mais 0,0001 vezes os novos pesos após a última passagem para frente e para trás. Por default, o algoritmo da EMA é aplicado após cada atualização de gradiente durante todo o período de treinamento. No entanto, isso pode ser lento devido às operações de memória necessárias para ler e gravar todos os pesos em cada etapa.

Para evitar esse procedimento caro, começamos com uma observação importante: como os pesos antigos são reduzidos por um fator de 0,9999 a cada batch, as primeiras iterações de treinamento contribuem apenas minimamente para a média final. Isso significa que só precisamos obter a média móvel exponencial das etapas finais. Concretamente, treinamos 1.400.000 batches e só aplicamos a EMA nas 50.000 etapas finais, o que representa cerca de 3,5% do período de treinamento. Os pesos das primeiras 1.350.000 iterações diminuem em $(0,9999)^{1350000}$; então, sua contribuição agregada teria um peso de menos de 1% no modelo final. Usando essa técnica, podemos evitar adicionar sobrecarga em 96,5% do treinamento e ainda obter um modelo EMA quase equivalente.

ESTIMATIVAS FINAIS DE TEMPO E CUSTO

Effects of different speedup optimizations

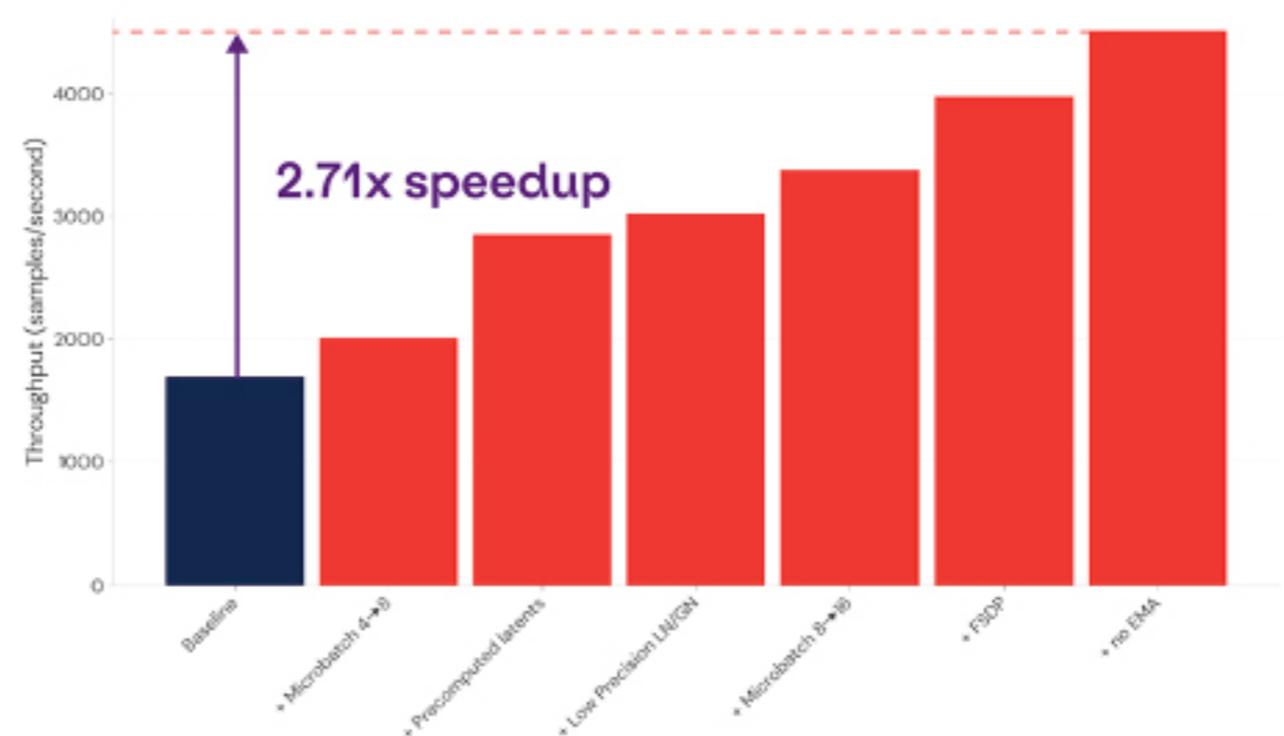


Figura 7: Throughput em imagens de 512x512 em 128 GPUs conforme cada otimização de aceleração é habilitada. Atingimos uma aceleração cumulativa total de 2,71x acima da linha de base.

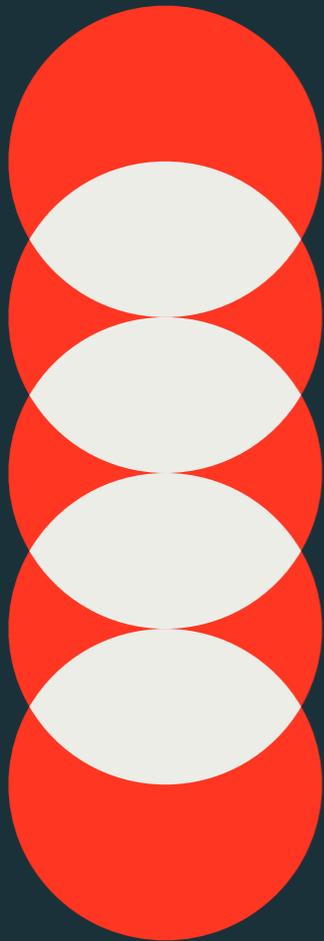
Mostramos como obtivemos uma redução de quase 3x no tempo e no custo para treinar o Stable Diffusion em comparação com nossos **resultados originais**. Com o xFormers, latentes pré-computadas, LayerNorm de baixa precisão, GroupNorm de baixa precisão, FSDP e EMA programada, a tabela 1 mostra que é possível treinar o Stable Diffusion em apenas 6,79 dias usando 21.000 horas de A100 por um custo total de menos de US\$ 42.000. Estimamos esses tempos e custos medindo o throughput para treinar 1,1 bilhão de imagens de 256x256 e 1,7 bilhão de imagens de 512x512 com um comprimento máximo tokenizado de 77 em um tamanho de batch global de 2048, conforme detalhado no cartão de modelo base Stable Diffusion 2. Isso é um pouco mais barato do que nossa **execução relatada anteriormente**, com um custo de US\$ 47,7 mil, pois não leva em conta nenhum tempo gasto em avaliação ou reinicialização devido a falhas de hardware.

NÚMERO DE A100S	THROUGHPUT PARA U-NET @ 256X256 (IMAGENS/SEGUNDO)	THROUGHPUT PARA U-NET @ 512X512 (IMAGENS/SEGUNDO)	THROUGHPUT PARA U-NET @ 512X512 COM EMA (IMAGENS/SEGUNDO)	DIAS PARA TREINAR NO MOSAICML CLOUD	CUSTO APROX. NO MOSAICML CLOUD
8	1100	290	290	101,04	US\$ 38.800
16	2180	585	580	50,29	US\$ 38.630
32	4080	1195	1160	25,01	US\$ 38.420
64	8530	2340	2220	12,63	US\$ 38.800
128	11600	4590	3927	6,79	US\$ 41.710

Tabela 1: tempo e custo estimados para treinar um modelo de difusão estável em 1,1 bilhão de imagens com resolução de 256x256, seguido por 1,7 bilhão de imagens com resolução de 512x512. Linhas diferentes mostram números diferentes de GPUs NVIDIA 40 GB A100 em um tamanho de batch global de 2048.

Essas otimizações mostram que o treinamento de modelos de geração de imagens a partir do zero está ao alcance de todos. Para atualizações sobre nosso trabalho mais recente, junte-se ao [Slack da nossa comunidade](#) ou siga-nos no [Twitter](#). Se sua organização deseja começar a treinar modelos de difusão hoje, [agende uma demonstração on-line](#) ou envie um e-mail para demo@mosaicml.com.

¹ Ao treinar grandes modelos com batches grandes que não cabem na memória em uma única passagem, cada batch é dividido em microbatches menores. Em cada dispositivo, podemos fazer uma passagem para a frente e para trás para cada microbatch e somar os gradientes no final para computar uma atualização de gradientes equivalente a uma única passagem para a frente e para trás com o batch inteiro de uma vez.



Etapa 5: avaliação de LLMs

A avaliação e o monitoramento constantes de grandes modelos de linguagem (LLMs) implementados e aplicativos de IA generativa são cruciais devido à natureza dinâmica dos dados com os quais interagem e dos ambientes em que operam. Esses sistemas aprendem com vastos conjuntos de dados e podem evoluir com o tempo, podendo levar a mudanças no desempenho, precisão ou até mesmo ao surgimento de vieses. O monitoramento contínuo garante que qualquer desvio do comportamento esperado possa ser detectado e corrigido prontamente, mantendo a integridade e a confiabilidade do aplicativo de IA. À medida que as necessidades do usuário e as normas sociais mudam, a **avaliação** contínua permite que esses modelos se adaptem, garantindo que seus resultados permaneçam relevantes, apropriados e eficazes. Essa vigilância não apenas mitiga os riscos associados às implementações de IA, como preocupações éticas e conformidade regulatória, mas também maximiza o valor e a utilidade que essas tecnologias trazem para organizações e usuários finais.

Avaliar LLMs é uma atribuição desafiadora e **em evolução**, principalmente porque os LLMs geralmente demonstram recursos desiguais em diferentes tarefas. Um LLM pode se destacar em um benchmark, mas pequenas variações na solicitação ou no problema podem afetar drasticamente o desempenho. A natureza dinâmica dos LLMs e suas vastas aplicações potenciais apenas amplificam o desafio de estabelecer padrões de avaliação abrangentes.

Os desafios atuais envolvidos na avaliação de aplicativos alimentados por LLMs incluem o seguinte:

- **Desempenho variável:** os LLMs podem ser **sensíveis a variações nos prompts**, demonstrando alta proficiência em uma tarefa, mas vacilando com pequenos desvios nos prompts.
- **Falta de verdade fundamental:** como a maioria dos LLMs produz linguagem natural, é muito difícil avaliar os resultados por meio de métricas tradicionais de PNL (**BLEU, ROUGE** etc.). Por exemplo, suponha que um LLM seja utilizado para resumir um artigo de notícia. Dois resumos igualmente bons podem ter palavras e ordens de palavras completamente diferentes, tornando difícil ou até impossível definir um "rótulo de verdade absoluta".
- **Avaliação específica do domínio:** para LLMs ajustados específicos do domínio, os benchmarks genéricos populares podem não capturar seus recursos diferenciados. Esses modelos são adaptados para tarefas especializadas, tornando as métricas tradicionais menos relevantes. Essa divergência geralmente requer o desenvolvimento de benchmarks e critérios de avaliação específicos do domínio. Veja o exemplo do **LLM de geração de código do Replit**.
- **Confiança no julgamento humano:** muitas vezes, o desempenho do LLM está sendo avaliado em domínios onde o texto é escasso ou há uma dependência do conhecimento especializado no assunto. Nesses cenários, avaliar a saída do LLM pode ser caro e demorado.

Para ajudar a dar exemplos de como isso pode ser feito, aqui estão dois ótimos exemplos de como você pode monitorar e avaliar seus LLMs implementados e aplicativos de IA generativa usando a Databricks.

Exemplos de avaliação de LLMs

Práticas recomendadas para avaliação de LLMs de aplicativos RAG:

um estudo de caso sobre o bot de documentação da Databricks

por **Quinn Leng, Kasey Uhlenhuth** e **Alkis Polyzotis**

Os chatbots são o caso de uso mais amplamente adotado para aproveitar os recursos avançados de bate-papo e raciocínio dos grandes modelos de linguagem (LLMs). A arquitetura de geração aumentada de recuperação (RAG) está se tornando rapidamente o padrão do setor para o desenvolvimento de chatbots porque combina os benefícios de uma base de conhecimento (por meio de um armazenamento de vetores) e modelos generativos (por exemplo, GPT-3.5 e GPT-4) para reduzir alucinações, manter informações atualizadas e aproveitar o conhecimento específico do domínio. No entanto, avaliar a qualidade das respostas dos chatbots continua sendo um problema não resolvido atualmente. Sem padrões do setor definidos, as organizações recorrem à classificação humana (rotulagem), o que consome tempo e é difícil de dimensionar.

Aplicamos a teoria à prática para ajudar a formar as práticas recomendadas para a avaliação automatizada dos LLMs, para que você possa implementar aplicações RAG em produção com rapidez e confiança. Este blog representa o primeiro de uma série de investigações que estamos realizando na Databricks para fornecer informações sobre a avaliação de LLMs. Toda a pesquisa deste post foi conduzida por Quinn Leng, engenheiro de software sênior da Databricks e criador do [Databricks Documentation AI Assistant](#).

DESAFIOS DA AUTOAVALIAÇÃO NA PRÁTICA

Recentemente, a comunidade de LLMs tem explorado o uso de "LLMs como juiz" para avaliação automatizada, com muitos usando LLMs poderosos, como o GPT-4, para fazer a avaliação de seus resultados de LLMs. O trabalho de pesquisa do grupo lmsys explora a viabilidade e os prós/contras de usar vários LLMs (GPT-4, ClaudeV1, GPT-3.5) como juiz de tarefas de escrita, matemática e conhecimento do mundo.

Apesar de toda essa grande pesquisa, ainda há muitas perguntas sem resposta sobre como aplicar os juízes de LLMs na prática:

- **Alinhamento com a classificação humana:** especificamente para um chatbot document-Q&A, até que ponto a classificação de um juiz de LLM reflete a preferência humana real em termos de correção, legibilidade e abrangência das respostas?
- **Precisão por meio de exemplos:** qual é a eficácia de fornecer alguns exemplos de classificação ao juiz de LLM e quanto isso aumenta a confiabilidade e a reutilização do juiz de LLM em diferentes métricas?
- **Escalas de notas apropriadas:** qual escala de classificação é recomendada, porque diferentes escalas de classificação são usadas por diferentes estruturas (por exemplo, o [AzureML](#) usa de 0 a 100, enquanto o [langchain](#) usa escalas binárias)?
- **Aplicabilidade em todos os casos de uso:** com a mesma métrica de avaliação (por exemplo, correção), até que ponto a métrica de avaliação pode ser reutilizada em diferentes casos de uso (por exemplo, bate-papo casual, resumo de conteúdo, geração aumentada de recuperação)?

APLICAÇÃO DE AUTOAVALIAÇÃO EFICAZ PARA APLICAÇÕES DE RAG

Exploramos as opções possíveis para as perguntas descritas acima no contexto de nosso próprio aplicativo de chatbot na Databricks. Acreditamos que nossas descobertas se generalizam e, portanto, podem ajudar sua equipe a avaliar efetivamente os chatbots baseados em RAG a um custo menor e em maior velocidade:

- O LLM-como-juiz concorda com a classificação humana em mais de 80% dos julgamentos. O uso de LLMs como juízes na avaliação do nosso chatbot baseado em documentos foi tão eficaz quanto juízes humanos, correspondendo exatamente à pontuação em mais de 80% dos julgamentos e ficando a uma diferença de 1 ponto (em uma escala de 0 a 3) em mais de 95% dos casos.
- Economize custos usando o GPT-3.5 com exemplos. O GPT-3.5 pode ser usado como juiz de LLM se você fornecer exemplos para cada pontuação de avaliação. Devido ao limite de tamanho do contexto, só é prático usar uma escala de classificação de baixa precisão. Usar o GPT-3.5 com exemplos em vez do GPT-4 reduz o custo do juiz de LLM em 10x e melhora a velocidade em mais de 3x.
- Use escalas de classificação de baixa precisão para facilitar a interpretação. Descobrimos que pontuações de classificação de menor precisão, como 0, 1, 2, 3, ou até mesmo binária (0, 1), podem manter a precisão em grande parte em comparação com escalas de maior precisão, como 0 a 10,0 ou 0 a 100,0, facilitando consideravelmente o fornecimento de rubricas de classificação para anotadores humanos e juízes de LLMs. O uso de uma escala de precisão mais baixa também permite a consistência das escalas de classificação entre diferentes juízes de LLMs (por exemplo, entre GPT-4 e claude2).
- As aplicações de RAG exigem seus próprios benchmarks. Um modelo pode ter um bom desempenho em um benchmark especializado publicado (por exemplo bate-papo casual, matemática ou escrita criativa), mas isso não garante um bom desempenho em outras tarefas (por exemplo, responder a perguntas de um determinado contexto). Os benchmarks só devem ser usados se o caso de uso corresponder, ou seja, um aplicativo de RAG só deve ser avaliado com um benchmark de RAG.

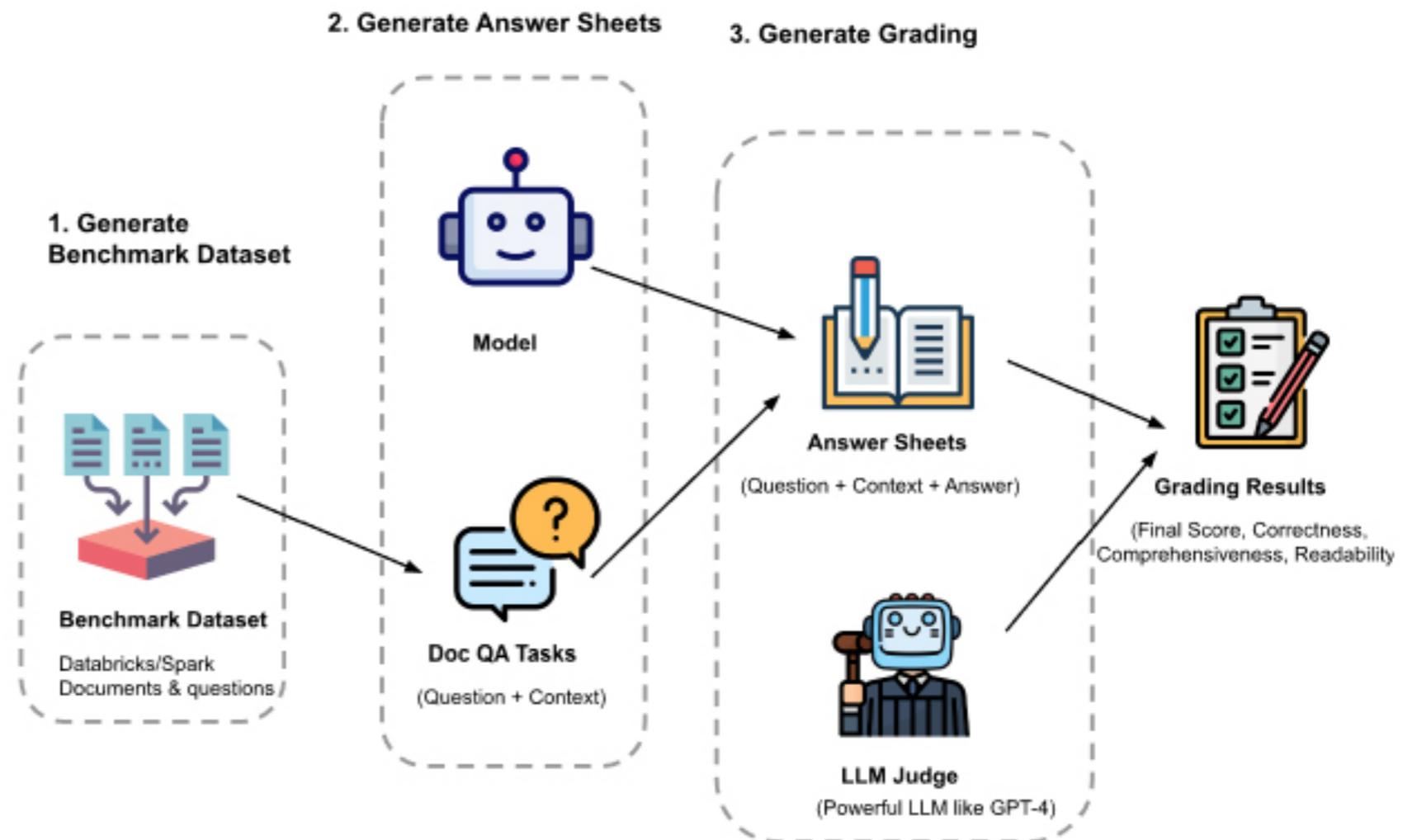
Com base em nossa pesquisa, recomendamos o seguinte procedimento ao usar um juiz de LLM:

- Use uma escala de classificação de 1 a 5
- Use o GPT-4 como juiz de LLM sem exemplos para entender as regras de classificação
- Mude seu juiz de LLM para o GPT-3.5 com um exemplo por pontuação

NOSSA METODOLOGIA PARA ESTABELECEER AS MELHORES PRÁTICAS

O restante desta postagem abordará a série de experimentos que conduzimos para formar essas melhores práticas.

CONFIGURAÇÃO DO EXPERIMENTO



O experimento teve três etapas:

- 1. Gerar um conjunto de dados de avaliação:** criamos um conjunto de dados a partir de 100 perguntas e contexto de documentos do Databricks. O contexto representa (pedaços de) documentos que são relevantes para a pergunta.

	question	context
0	What is DenseVector?	\n[c](DenseVector\$.html "See companion object"...
1	What is the return value of `cube`?	\n\n cube\n=====\n`cube.Rd` \n Create a ...
4	7. What are the value members available for Co...	\n c\n \n[org](.././../index.html) \n .\n ...
5	What are the value members of RuntimeInfo?	\n c\n \n[org](.././../index.html) \n ...
6	What is the name of the type used in JSON seri...	\n[c](ShortType\$.html "See companion object")\...
...
85	What is the return type of RDDBarrier.mapParti...	\n pyspark.RDDBarrier.mapPartitions\n [¶](#pys...
86	Can you tell me how to construct SparkAWSCrede...	\n o\n \n[org](.././../index.html) \n .\n ...
87	Can you tell me how to use PowerIterationClust...	\n PowerIterationClusteringModel\n [¶](#poweri...
88	What is the parameter numPartitions in RDD.dis...	\n pyspark.RDD.distinct\n [¶](#pyspark-rdd-dis...
89	How can the MultivariateGaussian class be used...	\n Source code for pyspark.mllib.stat.distribu...

- 2. Gerar folhas de respostas:** usando o conjunto de dados de avaliação, solicitamos que diferentes modelos de linguagem gerassem respostas e armazenamos os pares pergunta-contexto-resposta em um conjunto de dados chamado "folhas de respostas". Nessa pesquisa, usamos o GPT-4, GPT-3.5, Claude-v1, Llama2-70b-chat, Vicuna-33b e mpt-30b-chat.
- 3. Gerar classificações:** dadas as folhas de respostas, usamos vários LLMs para gerar classificações e raciocínio para as classificações. As classificações são uma pontuação composta de Correção (peso de 60%), Abrangência (peso de 20%) e Legibilidade (peso de 20%). Escolhemos esse esquema de pesos para refletir nossa preferência por correção nas respostas geradas. Outros aplicativos podem ajustar esses pesos de forma diferente, mas esperamos que a Correção continue sendo um fator dominante.

Além disso, as seguintes técnicas foram usadas para evitar viés posicional e melhorar a confiabilidade:

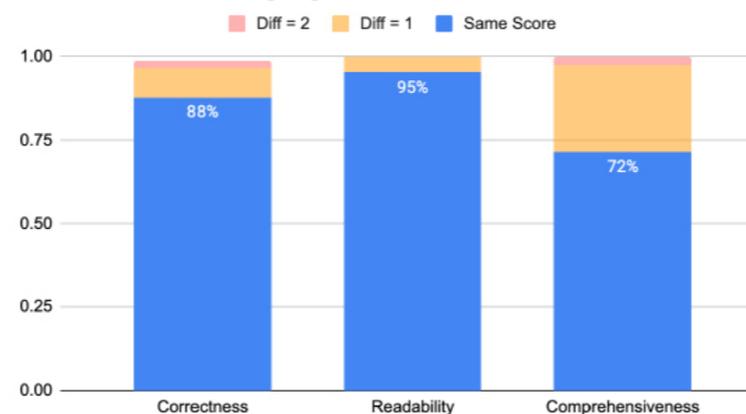
- Baixa temperatura (temperatura 0,1) para garantir a reprodutibilidade
- Classificação de resposta única em vez de comparação em pares
- Cadeia de pensamentos para permitir que o LLM reflita sobre o processo de avaliação antes de dar a pontuação final
- Geração de poucos exemplos, em que o LLM é fornecido com várias amostras da rubrica de avaliação para cada valor de pontuação em cada fator (Correção, Abrangência, Leitura).

EXPERIMENTO 1: ALINHAMENTO COM A CLASSIFICAÇÃO HUMANA

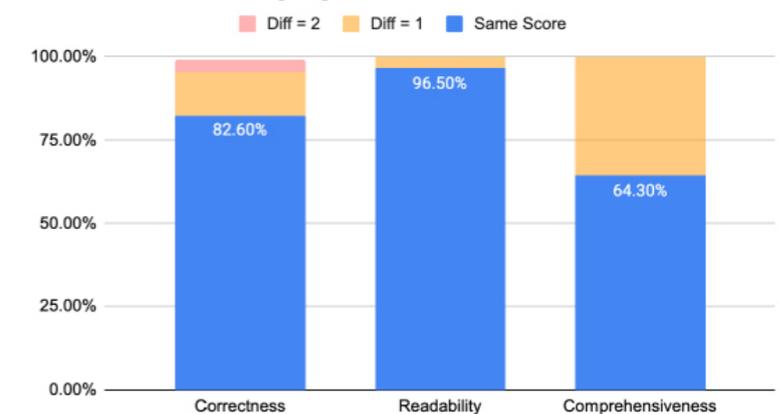
Para confirmar o nível de concordância entre anotadores humanos e juízes LLM, enviamos folhas de respostas (escala de 0 a 3) do gpt-3.5-turbo e vicuna-33b para uma empresa de rotulagem, a fim de coletar rótulos humanos, e então comparamos o resultado com a avaliação do GPT-4. Abaixo estão os resultados:

Juízes humanos e do GPT-4 podem chegar a uma concordância acima de 80% sobre a pontuação de correção e legibilidade. E se reduzirmos o requisito para uma diferença de pontuação menor ou igual a 1, o nível de concordância pode ultrapassar 95%. A métrica de Abrangência tem menos alinhamento, o que corresponde ao que ouvimos dos stakeholders de negócios que compartilharam que "abrangente" parece mais subjetivo do que métricas como Correção ou Legibilidade.

Human vs GPT-4 Grading Alignments for GPT-3.5 answers



Human vs GPT-4 Grading Alignments for vicuna-33b answers



EXPERIMENTO 2: PRECISÃO POR MEIO DE EXEMPLOS

O artigo lmsys usa esse **prompt** para instruir o juiz do LLM a avaliar com base na utilidade, relevância, precisão, profundidade, criatividade e nível de detalhe da resposta. Entretanto, o documento não compartilha detalhes específicos sobre a rubrica de avaliação. Em nossa pesquisa, descobrimos que muitos fatores podem afetar significativamente a pontuação final, como por exemplo:

- A importância de diferentes fatores: Utilidade, Relevância, Precisão, Profundidade, Criatividade
- A interpretação de fatores como a Utilidade é ambígua
- Se fatores diferentes entrarem em conflito entre si, onde uma resposta é útil, mas não é precisa

Desenvolvemos uma rubrica para instruir um juiz do LLM para uma determinada escala de classificação, tentando o seguinte:

1. Prompt original: aqui está o prompt original usado no artigo lmsys:

```
Desempenhe o papel de juiz imparcial e avalie a qualidade da resposta dada por um assistente de IA à pergunta do usuário exibida abaixo. Sua avaliação deve considerar fatores como utilidade, relevância, precisão, profundidade, criatividade e nível de detalhe da resposta. Comece sua avaliação fornecendo uma breve explicação. Seja o mais objetivo possível. Após fornecer sua explicação, você deve classificar a resposta em uma escala de 1 a 10, seguindo rigorosamente este formato
```

Adaptamos o prompt original do artigo lmsys para emitir nossas métricas sobre correção, abrangência e legibilidade, e também solicitamos que o juiz forneça uma justificativa de uma linha antes de dar cada pontuação (para se beneficiar do raciocínio da cadeia de pensamento). Abaixo está a versão zero shot do prompt, que não fornece nenhum exemplo, e a versão com poucos shots do prompt, que fornece um exemplo para cada pontuação. Em seguida, usamos as mesmas folhas de respostas como entrada e comparamos os resultados avaliados dos dois tipos de prompt.

2. Aprendizado Zero Shot: exigir que o juiz LLM forneça nossas métricas sobre correção, abrangência e legibilidade, além de solicitar que o juiz ofereça uma justificativa de uma linha para cada pontuação.

```
Desempenhe o papel de juiz imparcial e avalie a qualidade da resposta dada, que tenta responder à pergunta com base no contexto apresentado.  
Você receberá uma função chamada grading_function, que deverá chamar para cada contexto, pergunta e resposta fornecidos, a fim de enviar seu raciocínio e pontuação sobre a correção, abrangência e legibilidade da resposta.
```

3. Aprendizado de poucos shots: adaptamos o prompt de zero shot para fornecer exemplos explícitos de cada pontuação na escala. O novo prompt:

Desempenhe o papel de juiz imparcial e avalie a qualidade da resposta dada, que tenta responder à pergunta com base no contexto apresentado.

Você receberá uma função `grading_function` que você chamará para cada contexto, pergunta e resposta fornecidos para enviar seu raciocínio e pontuação de acordo com a Exatidão, Abrangência e Legibilidade da resposta.

Abaixo está sua rubrica da avaliação:

- Correção: se a resposta for correta, veja abaixo os detalhes das diferentes pontuações:

- Pontuação 0: a resposta está completamente incorreta, não menciona nada sobre a pergunta ou é totalmente contrária à resposta correta.

- Por exemplo, quando perguntado "Como encerrar um cluster da Databricks", a resposta é uma cadeia de caracteres vazia, ou um conteúdo completamente irrelevante, ou "desculpe, não sei a resposta".

- Pontuação 1: a resposta fornece alguma relevância para a pergunta e responde corretamente a um aspecto da pergunta.

- Exemplo:

- Pergunta: "Como encerrar um cluster do Databricks"

- Resposta: o cluster da Databricks é um ambiente de computação baseado em nuvem que permite aos usuários processar big data e executar tarefas de processamento de dados distribuídos com eficiência.

- Ou resposta: no workspace da Databricks, navegue até a guia "Clusters". E, então, essa é uma pergunta difícil sobre a qual preciso pensar mais.

- Pontuação 2: a resposta responde majoritariamente à pergunta, mas está ausente ou alucinada em um aspecto crítico.

- Exemplo:

- Pergunta: "Como encerrar um cluster da Databricks"

- Resposta: "No workspace da Databricks, navegue até a guia "Clusters".

Encontre o cluster que você deseja encerrar na lista de clusters ativos.

E, então, você encontrará um botão para encerrar todos os clusters de uma só vez"

- Pontuação 3: a resposta responde corretamente à pergunta e não ignora nenhum aspecto importante

- Exemplo:

- Pergunta: Como encerrar um cluster do Databricks

- Resposta: No workspace da Databricks, navegue até a guia "Clusters".

Encontre o cluster que você deseja encerrar na lista de clusters ativos.

Clique na seta para baixo ao lado do nome do cluster para abrir os detalhes do cluster.

Clique no botão "Encerrar". Uma caixa de diálogo de confirmação será exibida. Clique em "Encerrar" novamente para confirmar a ação."

- Abrangência: o grau de abrangência da resposta, se ela responde plenamente a todos os aspectos da pergunta e fornece uma explicação abrangente e outras informações necessárias. Abaixo estão os detalhes para diferentes pontuações:

- Pontuação 0: normalmente, se a resposta estiver completamente incorreta, a Abrangência também terá pontuação zero.

- Pontuação 1: se a resposta estiver correta, mas muito curta para responder totalmente à pergunta, podemos dar pontuação 1 para Abrangência.

- Exemplo:

- Pergunta: Como usar a API da Databricks para criar um cluster?

- Resposta: primeiro, você precisará de um token de acesso da Databricks com as permissões apropriadas. Você pode gerar esse token por meio da interface do usuário da Databricks na opção "Configurações do usuário". E então (o resto está faltando)

- Pontuação 2: a resposta está correta e responde aproximadamente aos principais aspectos da pergunta, mas falta uma descrição sobre os detalhes. Ou faltam completamente detalhes sobre um aspecto menor.

```

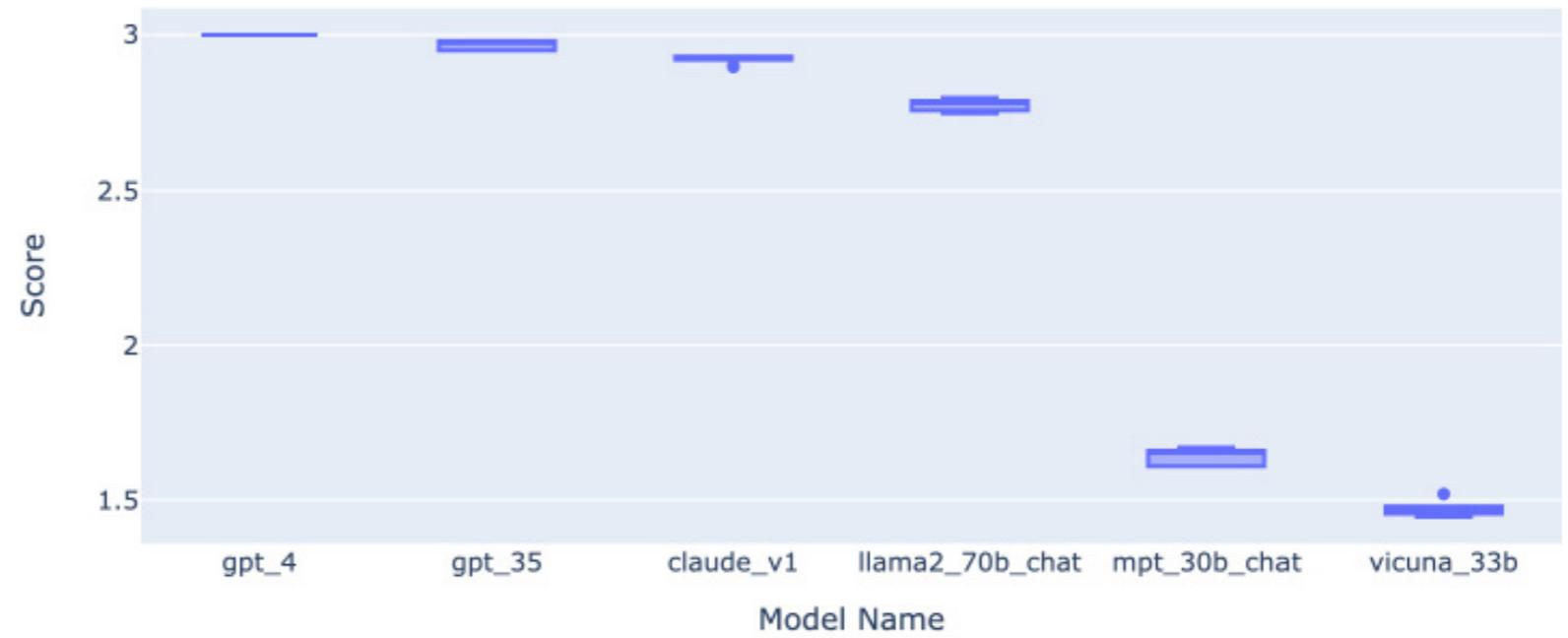
- Exemplo:
  - Pergunta: como usar a API da Databricks para criar um cluster?
  - Resposta: Você precisará de um token de acesso da Databricks com as permissões apropriadas. Em seguida, você precisará configurar o URL da solicitação e, em seguida, poderá fazer a solicitação de HTTP. Em seguida, você poderá lidar com a resposta da solicitação.
- Exemplo:
  - Pergunta: como usar a API da Databricks para criar um cluster?
  - Resposta: Você precisará de um token de acesso da Databricks com as permissões apropriadas. Em seguida, você precisará configurar o URL da solicitação e, em seguida, poderá fazer a solicitação de HTTP. Em seguida, você poderá lidar com a resposta da solicitação.
- Pontuação 3: a resposta está correta e abrange todos os principais aspectos da pergunta
- Legibilidade: a resposta é legível? Ela contém informações redundantes ou incompletas que prejudicam a legibilidade da resposta?
- Pontuação 0: a resposta é completamente ilegível, como por exemplo, totalmente composta por símbolos difíceis de ler; por exemplo, continua repetindo as palavras, o que torna muito difícil entender o significado do parágrafo. Nenhuma informação significativa pode ser extraída da resposta.
- Pontuação 1: a resposta é ligeiramente legível, há símbolos irrelevantes ou palavras repetidas, mas pode formar aproximadamente uma frase significativa que abrange alguns aspectos da resposta.
- Exemplo:
  - Pergunta: como usar a API da Databricks para criar um cluster?
  - Resposta: Você você você você você precisará de um token de acesso a Databricks com as permissões apropriadas. Em seguida, você precisará configurar o URL da solicitação e, em seguida, poderá fazer a solicitação HTTP. Então Então Então Então Então Então Então
- Pontuação 2: a resposta está correta e quase sempre legível, mas há uma parte óbvia que está afetando a legibilidade (menção de partes irrelevantes, palavras repetidas)
- Exemplo:
  - Pergunta: Como encerrar um cluster da Databricks
  - Resposta: No workspace da Databricks, navegue até a guia "Clusters".
  Encontre o cluster que você deseja encerrar na lista de clusters ativos.
  Clique na seta para baixo ao lado do nome do cluster para abrir os detalhes do cluster.
  Clique no botão "Encerrar".....
  Uma caixa de diálogo de confirmação será exibida. Clique em "Encerrar" novamente para confirmar a ação.
- Pontuação 3: a resposta está correta e é de fácil leitura, sem nenhuma parte óbvia que afete a legibilidade.
- Então a classificação final:
  - Proporção: 60% de Exatidão +20% de Abrangência +20% de Legibilidade

```

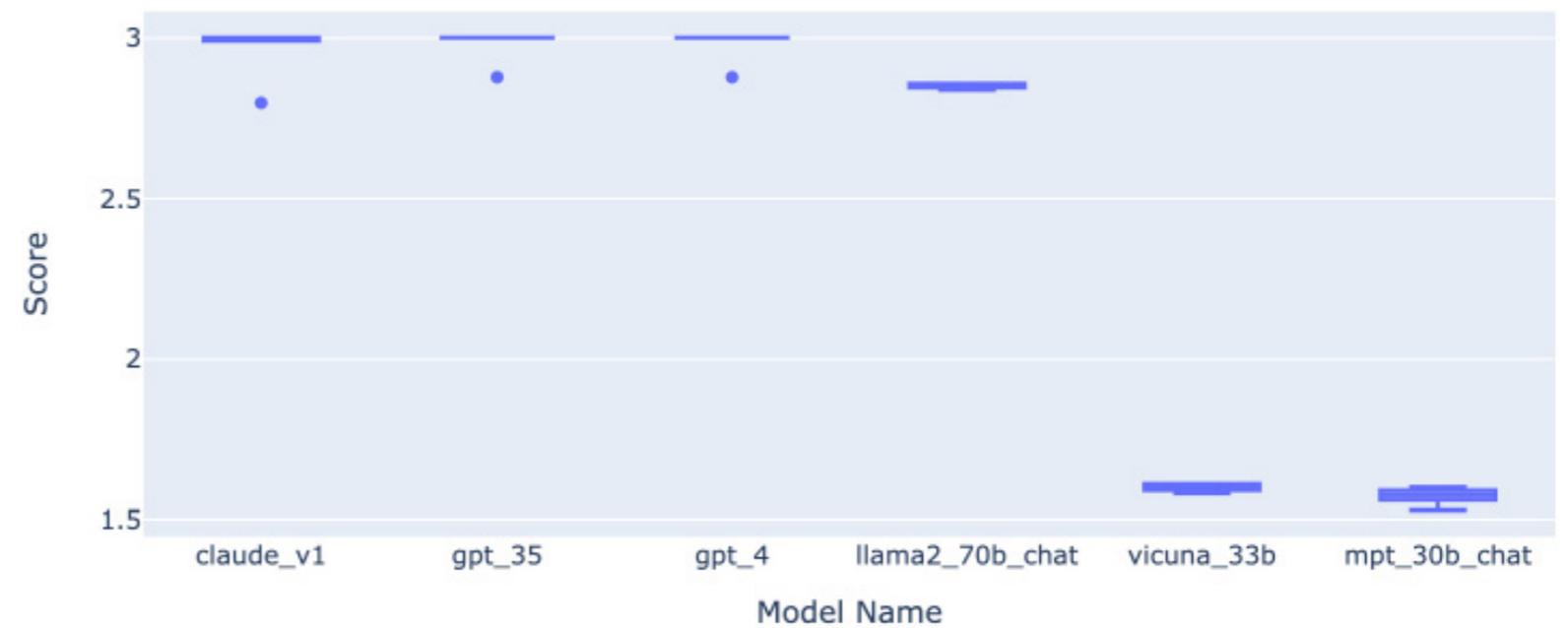
Com esse experimento, aprendemos várias coisas:

- Usar o prompt Poucos shots com o GPT-4 não fez uma diferença óbvia na consistência dos resultados. Quando incluímos o critério de avaliação detalhado com exemplos, não observamos uma melhoria notável nos resultados de avaliação do GPT-4 em diferentes modelos de LLM. Curiosamente, isso causou uma pequena variação na faixa das pontuações.

gpt-4 as Judge: LLM Score Variation: Zero shot

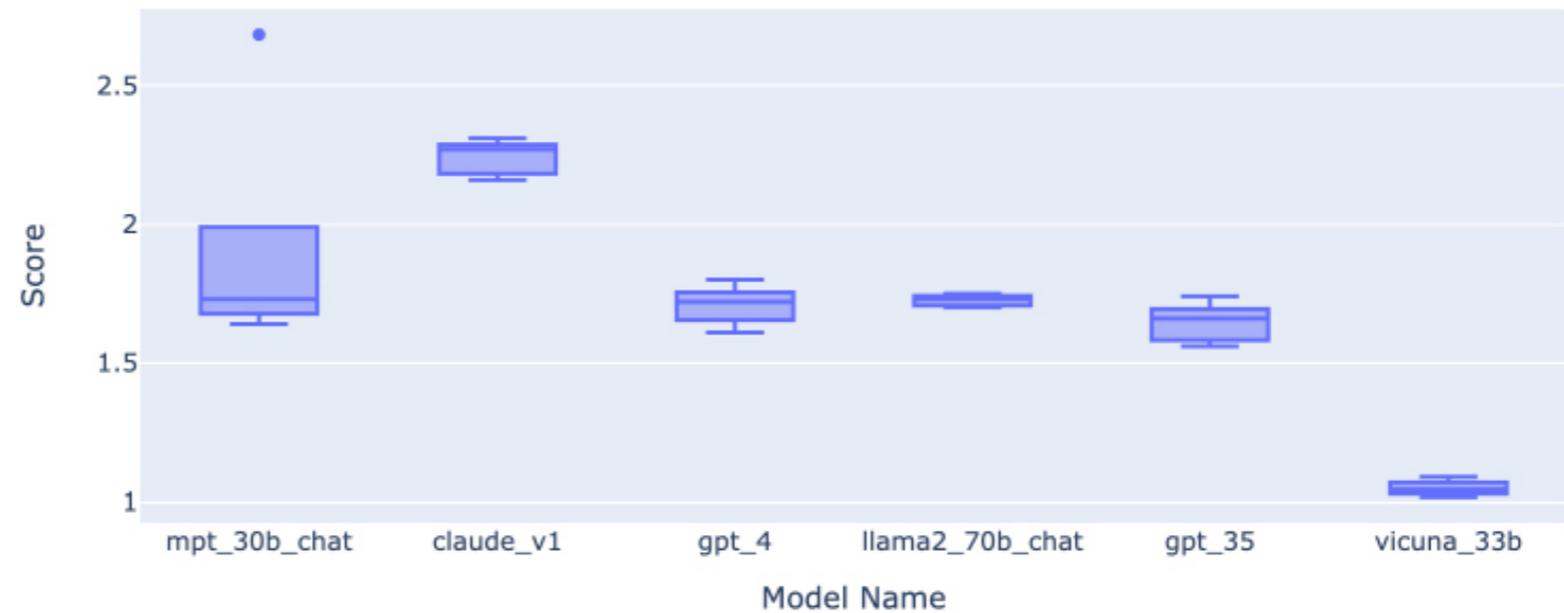


gpt-4 as Judge: LLM Score Variation: Few shot

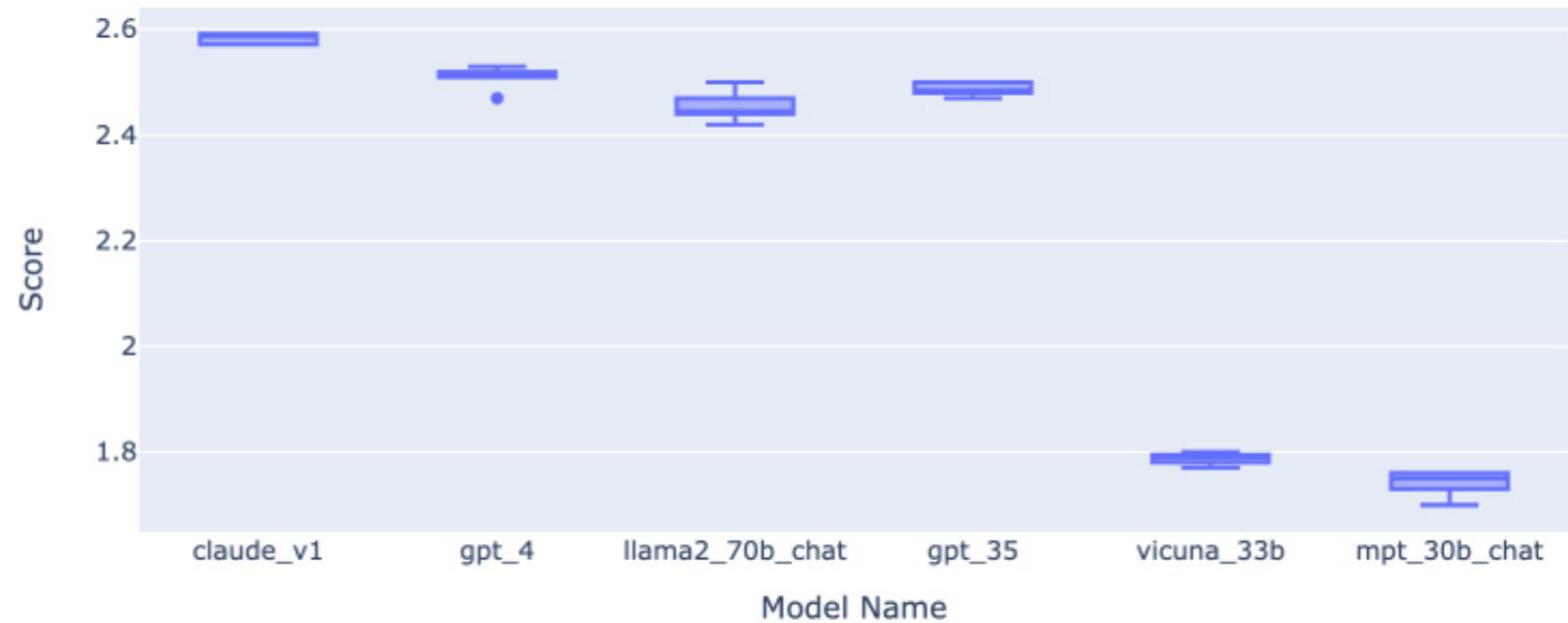


- Incluir alguns exemplos para o GPT-3.5-turbo-16k melhora significativamente a consistência das pontuações e torna o resultado utilizável. Incluir rubricas/exemplos de avaliação detalhados tem uma melhoria muito óbvia no resultado da avaliação do GPT-3.5. Embora o valor da pontuação média real seja ligeiramente diferente entre o GPT-4 e o GPT-3,5 (pontuação 3,0 versus pontuação 2,6), a classificação e a precisão permanecem bastante consistentes
- Ao contrário, usar o GPT-3.5 sem uma rubrica de avaliação obtém resultados muito inconsistentes e é completamente inutilizável
- Observe que estamos usando o GPT-3.5-turbo-16k em vez do GPT-3.5-turbo, já que o prompt pode ter mais de 4 k tokens.

gpt-3.5-turbo-16k as Judge: LLM Score Variation: Zero shot



gpt-3.5-turbo-16k as Judge: LLM Score Variation: Few shot



EXPERIMENTO 3: ESCALAS DE AVALIAÇÕES APROPRIADAS

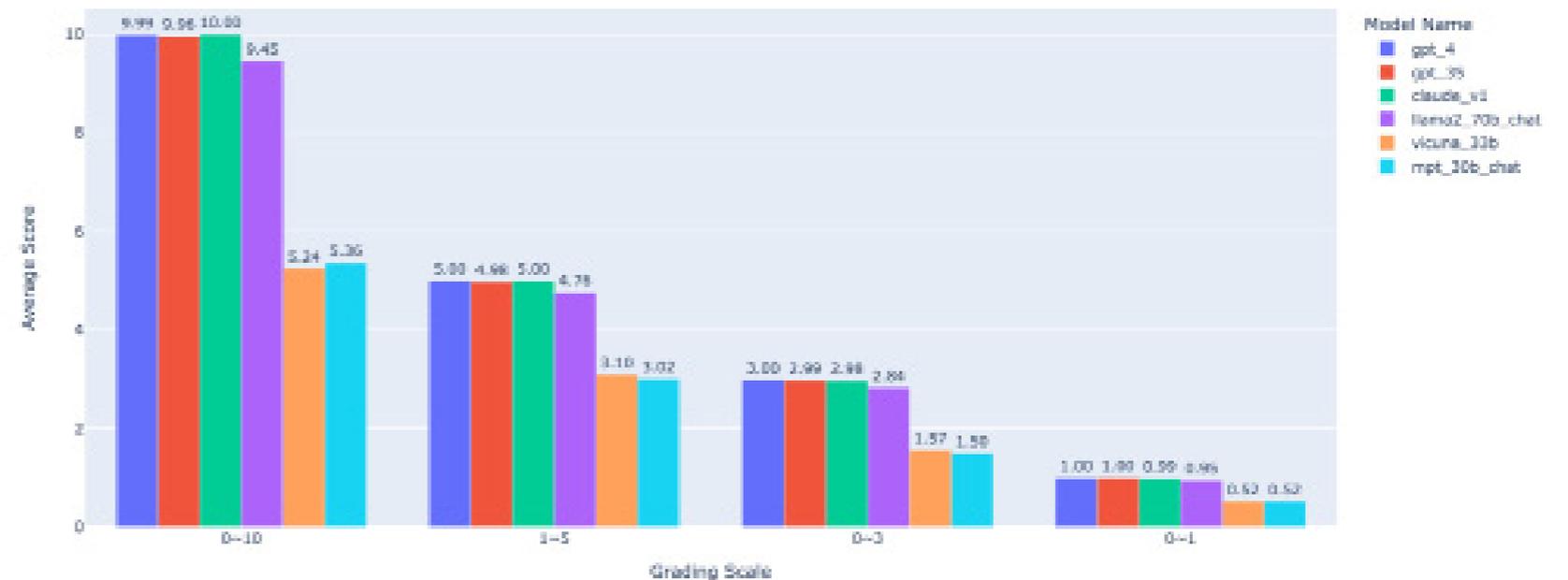
O artigo LLM-como-juíz usa uma escala não inteira de 0 ~ 10 (ou seja, flutuante) para a escala de classificação; em outras palavras, usa uma rubrica de alta precisão para a pontuação final. Descobrimos que essas escalas de alta precisão causam problemas a downstream com o seguinte:

- **Consistência:** os avaliadores (humanos e LLM) lutaram para manter o mesmo padrão para a mesma pontuação ao avaliar em alta precisão. Como resultado, descobrimos que as pontuações de saída são menos consistentes entre os juízes se você passar de escalas de baixa precisão para escalas de alta precisão.
- **Explicabilidade:** além disso, se quisermos fazer a validação cruzada dos resultados julgados pelo LLM com os resultados julgados por humanos, precisaremos fornecer instruções sobre como classificar as respostas. É muito difícil fornecer instruções precisas para cada "pontuação" em uma escala de classificação de alta precisão. Por exemplo, qual é um bom exemplo para uma resposta que é classificada como 5,1 em comparação com 5,6?

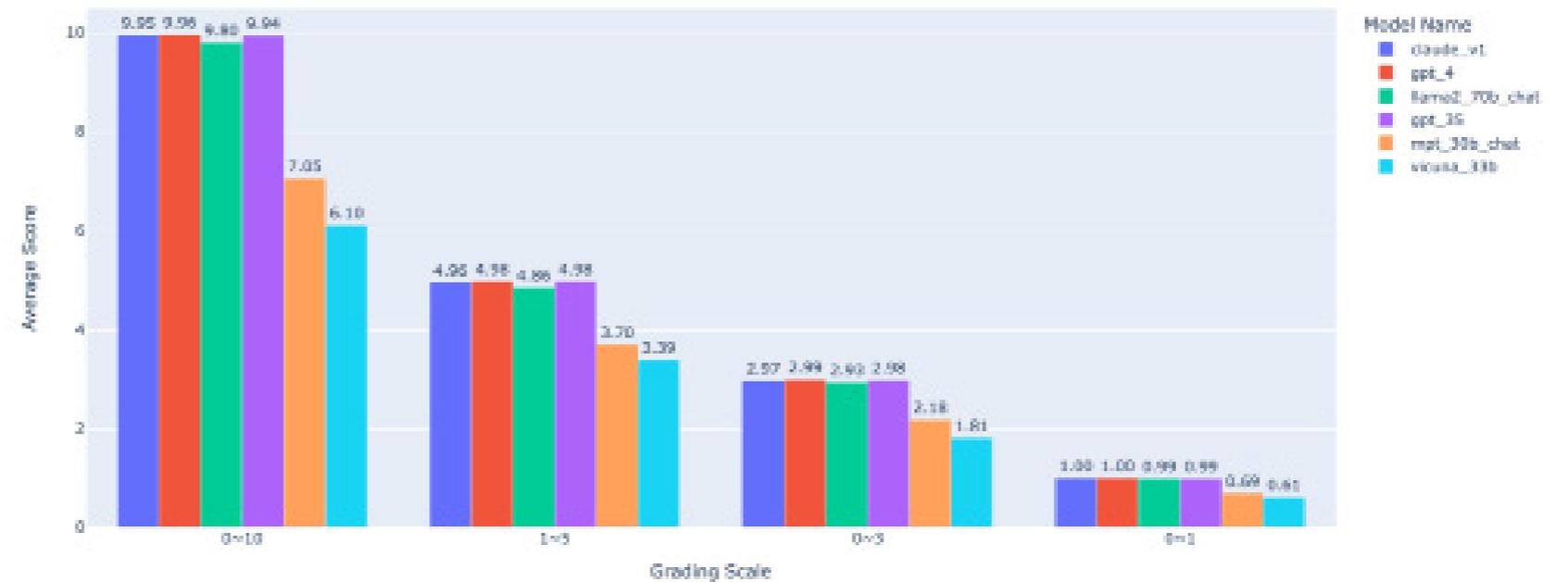
Fizemos experiências com várias escalas de classificação de baixa precisão para fornecer orientação sobre a "melhor" escala a ser usada. Em última análise, recomendamos uma escala inteira de 0-3 ou 0-4 (se você quiser se ater à escala **Likert**). Experimentamos 0-10, 1-5, 0-3 e 0-1 e aprendemos que:

- A classificação binária funciona para métricas simples como "usabilidade" ou "bom/ruim".
- Em escalas como 0-10, é difícil encontrar critérios de distinção entre todas as pontuações.

[gpt-4 as Judge] Avg Score by Model and Grading Scale



[gpt-3.5-turbo-16k as Judge] Avg Score by Model and Grading Scale



Conforme mostrado nesses gráficos, tanto o GPT-4 quanto o GPT-3.5 podem manter uma classificação consistente dos resultados usando diferentes escalas de classificação de baixa precisão. Portanto, usar uma escala de classificação mais baixa, como 0 ~ 3 ou 1 ~ 5, pode equilibrar a precisão com explicabilidade).

Portanto, recomendamos uma escala de 0 a 3 ou de 1 a 5 para facilitar o alinhamento com as avaliações humanas, raciocinar sobre os critérios de pontuação e fornecer exemplos para cada pontuação dentro da faixa.

EXPERIMENTO 4: APLICABILIDADE EM TODOS OS CASOS DE USO

O artigo [LLM-como-juiz](#) mostra que tanto o LLM quanto o julgamento humano classificam o modelo Vicuna-13B como um concorrente próximo do GPT-3.5:

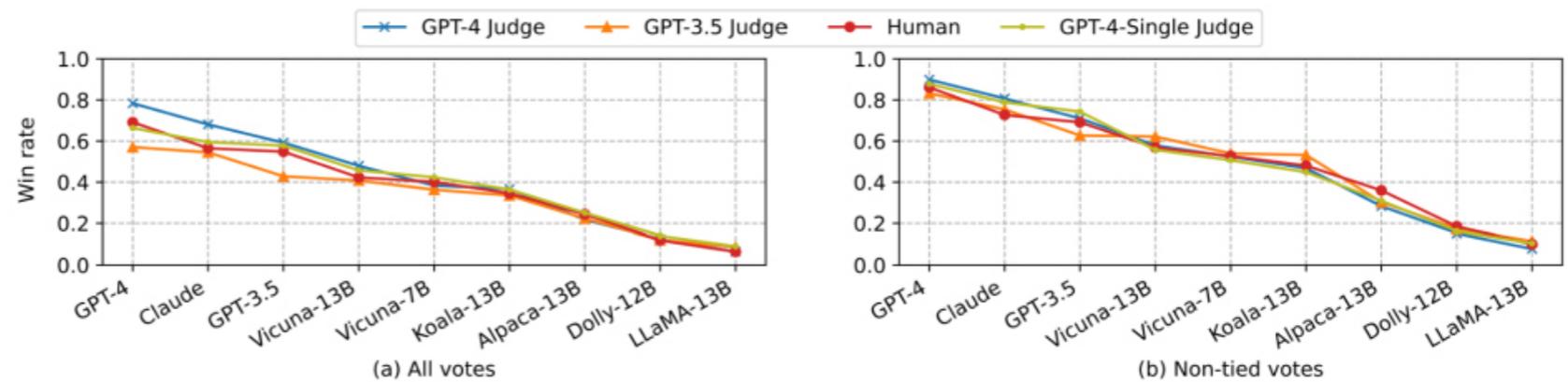
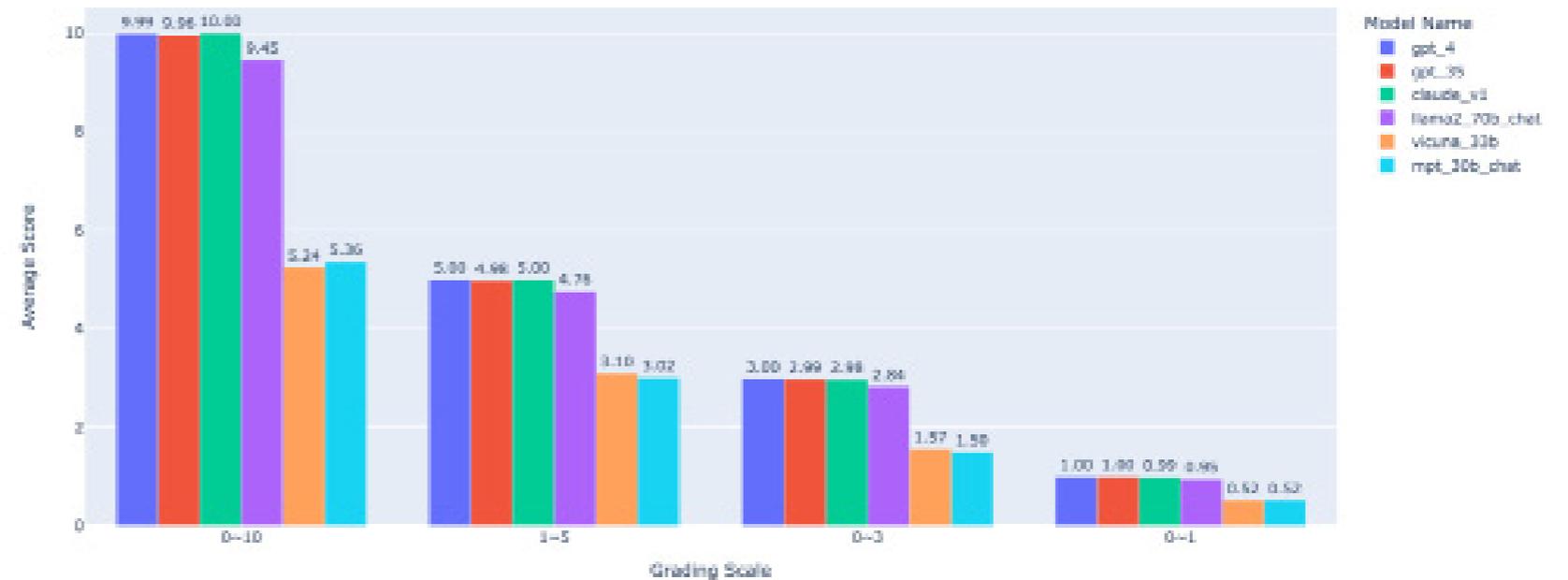


Figura 4: taxa média de vitórias de nove modelos sob diferentes juízes no Chatbot Arena.

No entanto, quando comparamos o conjunto de modelos para nossos casos de uso de perguntas e respostas de documentos, descobrimos que mesmo o modelo Vicuna-33B, muito maior, tem um desempenho visivelmente pior do que o GPT-3.5 ao responder perguntas com base no contexto. Essas descobertas também são verificadas pelo GPT-4, GPT-3.5 e juízes humanos (conforme mencionado no Experimento 1), que concordam que o Vicuna-33B está tendo um desempenho pior do que o do GPT-3.5.

[gpt-4 as Judge] Avg Score by Model and Grading Scale



Analizamos mais de perto o conjunto de dados de referência proposto pelo artigo e descobrimos que as **três categorias de tarefas** (escrita, matemática, conhecimento) não refletem ou contribuem diretamente para a capacidade do modelo de sintetizar uma resposta com base em um contexto. Em vez disso, intuitivamente, os casos de uso de perguntas e respostas de documentos precisam de referências sobre compreensão de leitura e acompanhamento de instruções. Assim, os resultados da avaliação não podem ser transferidos entre casos de uso, e precisamos criar benchmarks específicos de casos de uso para avaliar adequadamente o quanto um modelo é bom para atender às necessidades do cliente.

USAR O MLFLOW PARA APROVEITAR NOSSAS MELHORES PRÁTICAS

Com os experimentos acima, exploramos como diferentes fatores podem afetar significativamente a avaliação de um chatbot e confirmamos que o LLM como juiz pode refletir amplamente as preferências humanas pelo caso de uso de perguntas e respostas de documentos. Na Databricks, estamos desenvolvendo a API de avaliação MLflow para ajudar sua equipe a avaliar com eficácia seus aplicativos de LLM com base nessas descobertas. O MLflow 2.4 introduziu a API de avaliação para LLMs para comparar a saída de texto de vários modelos lado a lado. O MLflow 2.6 introduziu métricas baseadas em LLM para avaliação, como toxicidade e perplexidade, e estamos trabalhando para proporcionar compatibilidade com o LLM como juiz em um futuro próximo!

Enquanto isso, compilamos abaixo a lista de recursos que mencionamos em nossa pesquisa:

- [Repositório Doc_qa](#)
 - O código e os dados que usamos para conduzir os experimentos
- [Artigo de pesquisa LLM-como-juiz do grupo lmsys](#)
 - O artigo é a primeira pesquisa a usar o LLM como juiz para casos de uso de bate-papo casual. Ele explorou extensivamente a viabilidade, os prós e os contras de usar o LLM (GPT-4, ClaudeV1, GPT-3.5) como juiz para tarefas de escrita, matemática e conhecimento mundial

[Avaliação offline do LLM: avaliação passo a passo do aplicativo de IA generativa na Databricks](#)
por [Abe Omorogbe](#), [Liang Zhang](#), [Sunish Sheth](#), [Corey Zumar](#), [Maheswaran Venkatachalam](#), [Emil Lysgaard](#)
e [Mathias Christiansen](#)

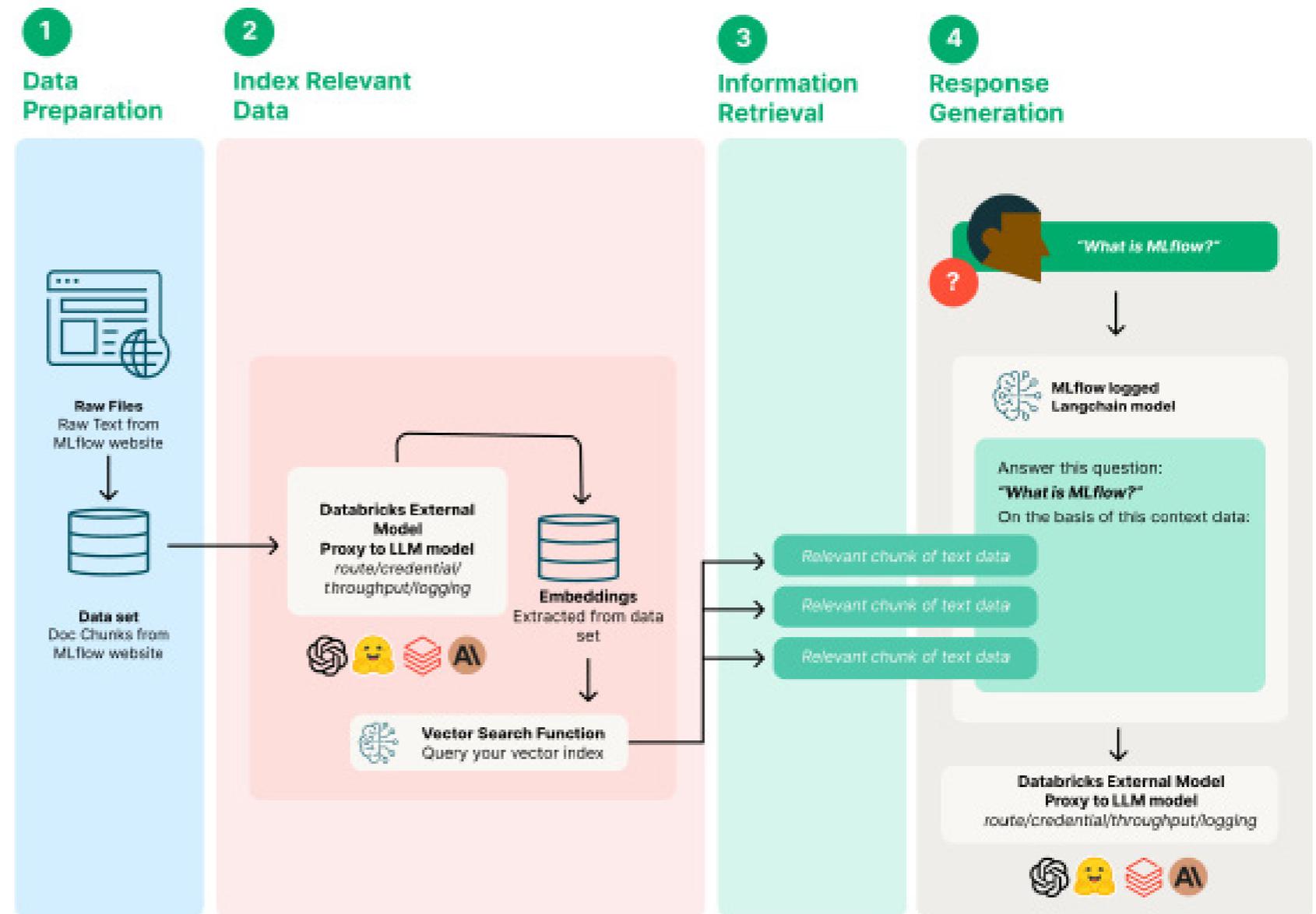
HISTÓRICO

Em uma era em que a geração aumentada de recuperação (RAG) está revolucionando a maneira como interagimos com aplicativos orientados por IA, garantir a eficiência e a eficácia desses sistemas nunca foi tão essencial. A Databricks e a MLflow estão na vanguarda dessa inovação, oferecendo soluções simplificadas para a avaliação crítica de aplicativos de IA generativa.

Esta postagem do blog orienta você pelo processo simples e eficaz de aproveitar o Databricks Data Intelligence Platform para aprimorar e avaliar a qualidade dos três componentes principais de seus aplicativos GenAI: Prompts, sistema de recuperação e LLM de base, garantindo que seus aplicativos GenAI continuem a gerar resultados precisos.

CASO DE USO

Vamos criar um chatbot de perguntas e respostas que responderá a perguntas da documentação do MLflow e, em seguida, avaliará os resultados.



CONFIGURAR MODELOS EXTERNOS NA DATABRICKS

O recurso Databricks **Model Serving** pode ser usado para gerenciar, governar e acessar modelos externos de vários provedores de grandes modelos de linguagem (LLMs), como o Azure OpenAI GPT, Anthropic Claude ou AWS Bedrock, dentro de uma organização. Ele oferece uma interface de alto nível que simplifica a interação com esses serviços ao fornecer um endpoint unificado para lidar com solicitações específicas relacionadas a LLMs.

Principais vantagens de usar o **Model Serving**:

- **Modelos de consulta por meio de uma interface unificada:** simplifica a interface para chamar vários LLMs em sua organização. Consulte modelos por meio de uma API e SDK unificados compatíveis com o OpenAI e gerencie todos os modelos por meio de uma única interface de usuário.
- **Governar e gerenciar modelos:** centraliza o gerenciamento de endpoints de vários LLMs em sua organização. Isso inclui a capacidade de gerenciar permissões e rastrear limites de uso.
- **Gerenciamento central de chaves:** centraliza o gerenciamento de chaves de API em um local seguro, o que aumenta a segurança organizacional, minimizando a exposição de chaves no sistema e no código e reduzindo a carga sobre os usuários finais.

CRIE UM ENDPOINT DE DISPONIBILIZAÇÃO COM UM MODELO EXTERNO NA DATABRICKS

```
1 import mlflow
2 import mlflow.deployments

3 client = mlflow.deployments.get_deploy_client("databricks")

4 endpoint_name = f"test-endpoint-{uuid.uuid4()}"

5 client.create_endpoint(
6 name=endpoint_name,
7 config={
8     "served_entities": [
9         {
10            "name": "test",
11            "external_model": {
12                "name": "gpt-3.5-turbo-instruct",
13                "provider": "openai",
14                "task": "llm/v1/completions",
15                "openai_config": {
16                    "openai_api_type": "azure",
17                    "openai_api_key": "{{secrets/<your-scope-name>/<your-key-name>}}", ## Use Databricks Secrets.
18                    "openai_api_base": "https://<your-endpoint>.openai.azure.com/",
19                    "openai_deployment_name": "<your-deployment-name>",
20                    "openai_api_version": "2023-05-15",
21                },
22            },
23        },
24    ],
25 },
26 )
```

EXPLORE PROMPTS COM O DATABRICKS AI PLAYGROUND

Nesta seção, entenderemos: qual é o desempenho de diferentes prompts com o LLM escolhido?

Recentemente, apresentamos o Databricks AI **Playground**, que oferece a melhor experiência da categoria para criar o prompt perfeito. Sem a necessidade de código, você pode experimentar vários LLMs disponibilizados como endpoints na Databricks e testar diferentes parâmetros e prompts.

As principais vantagens do Databricks AI Playground são:

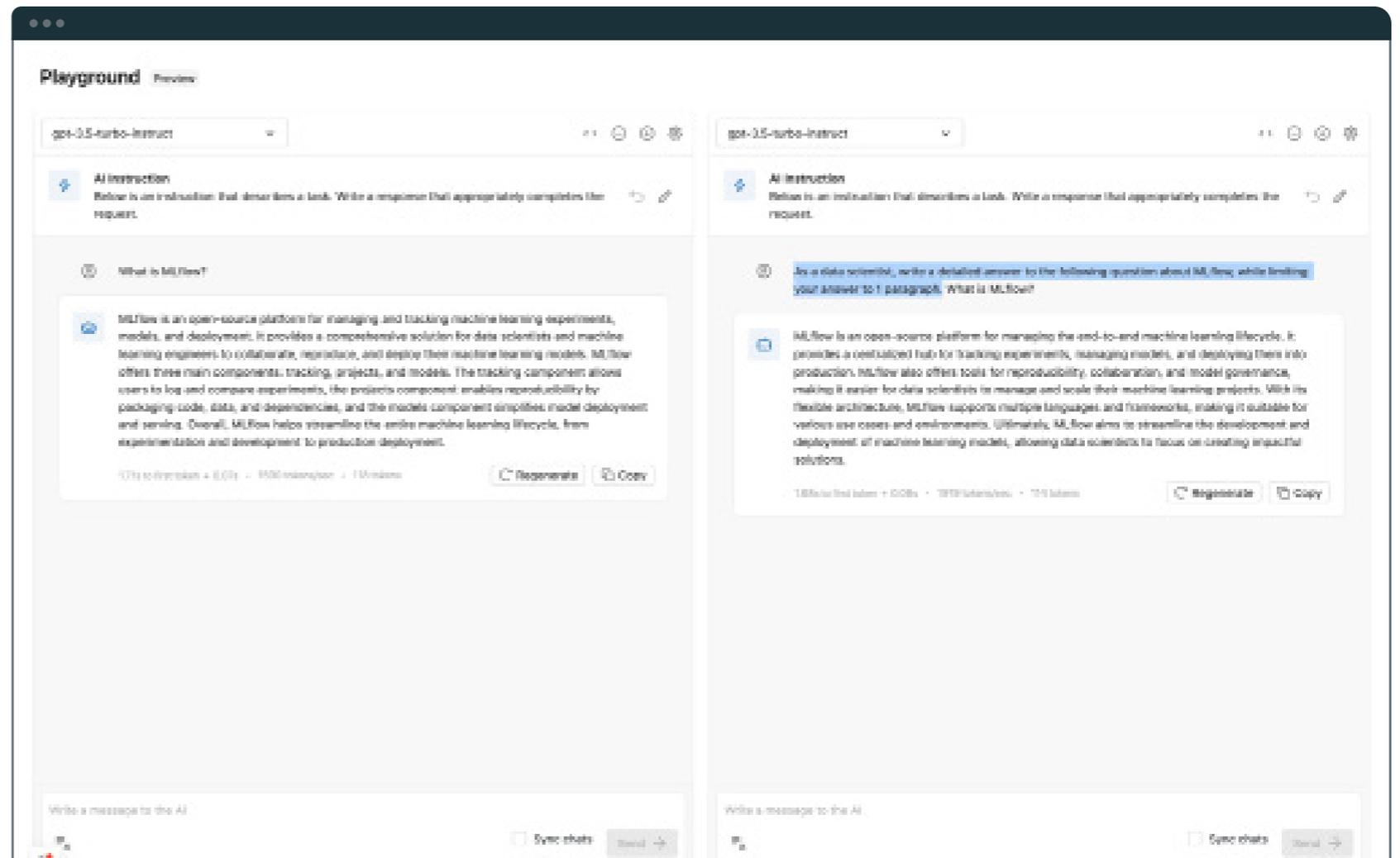
- **Testes rápidos:** teste rapidamente os modelos implantados diretamente na Databricks.
- **Comparação fácil:** localização central para comparar vários modelos em diferentes solicitações e parâmetros para comparação e seleção.

USO DO DATABRICKS AI PLAYGROUND

Nós nos aprofundamos nos testes de prompts relevantes com o OpenAI GPT 3.5 Turbo, aproveitando o Databricks **AI Playground**.

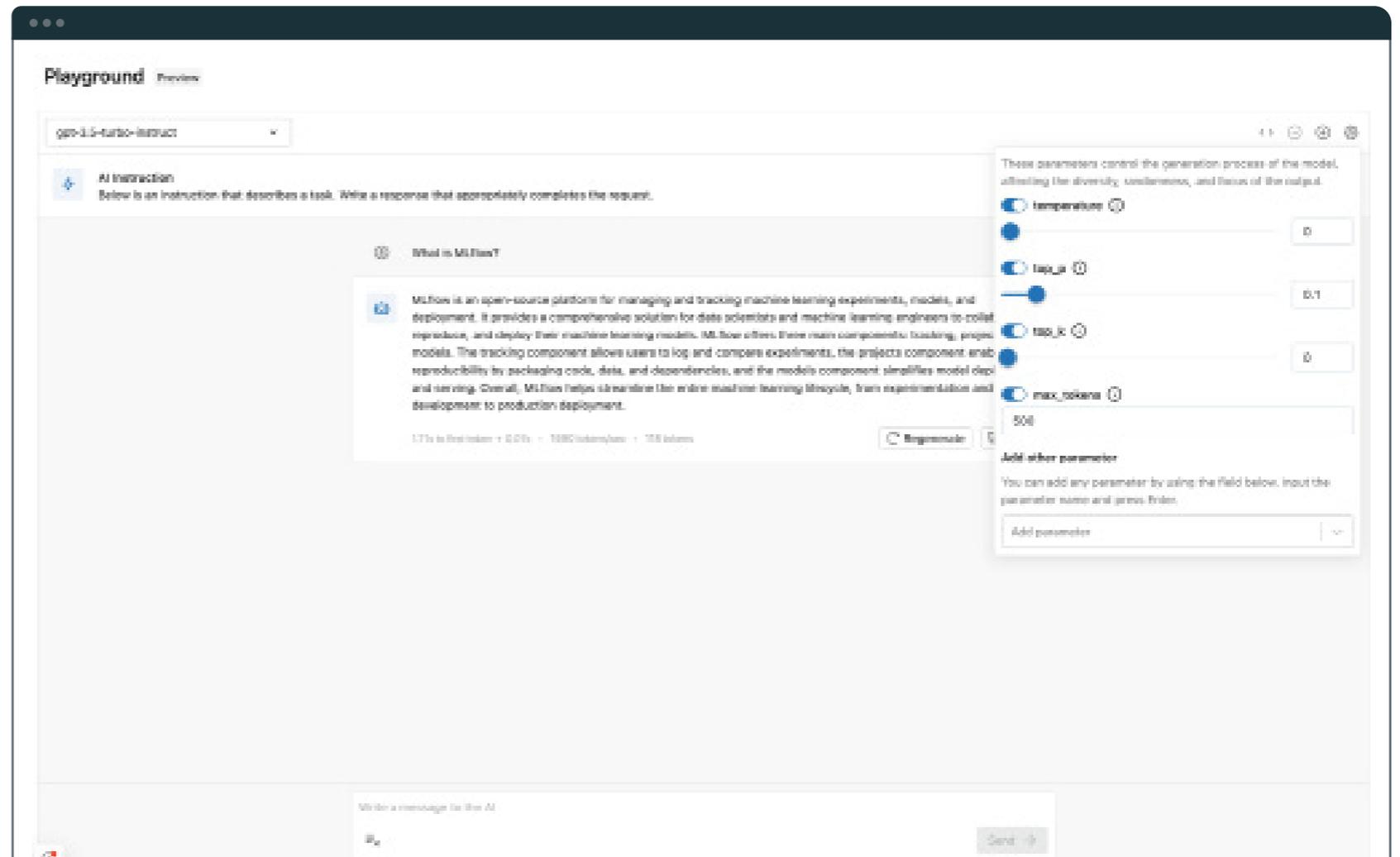
COMPARAÇÃO DE DIFERENTES PROMPTS E PARÂMETROS

No Playground, você pode comparar a saída de vários prompts para ver qual deles apresenta melhores resultados. Diretamente no Playground, você pode experimentar várias instruções, modelos e parâmetros para descobrir qual combinação fornece os melhores resultados. O modelo e a combinação de parâmetros podem, então, ser adicionados ao aplicativo GenAI e usados para a geração de respostas com o contexto correto.



ADIÇÃO DE MODELO E PARÂMETROS AO SEU APLICATIVO GENAI

Depois de brincar com alguns prompts e parâmetros, você pode usar as mesmas configurações e o mesmo modelo em sua aplicação de IA generativa.



Exemplo de como importar o mesmo modelo externo no LangChain. Abordaremos como transformar isso em um POC GenAI na próxima seção.

```
1 from langchain.llms import Databricks
2
3 llm = Databricks(
4     endpoint_name="<endpoint-name>",
5     extra_params={"temperature": 0.1,
6                 "top_p": 0.1,
7                 "max_tokens": 500,
8                 } #parameters used in AI Playground
9 )
```

CRIE UM POC GENAI COM O LANGCHAIN E REGISTRE COM O MLFLOW

Agora que encontramos um bom modelo e parâmetros de prompt para seu caso de uso, vamos criar um aplicativo de IA generativa de amostra que é um chatbot de controle de qualidade que responderá a perguntas da documentação do MLflow usando um banco de dados vetorial, **incorporando o modelo com o API Databricks Foundation Model** e o Azure OpenAI GPT 3.5 como modelo de geração.

CRIE UMA AMOSTRA DO APLICATIVO DE IA GENERATIVA COM O LANGCHAIN USANDO DOCUMENTOS DO SITE DO MLFLOW

```
1 import os
2 import pandas as pd
3 import mlflow
4 import chromadb
5 from langchain.chains import RetrievalQA
6 from langchain.document_loaders import WebBaseLoader
7 from langchain.llms import Databricks
8 from langchain.embeddings.databricks import DatabricksEmbeddings
9 from langchain.text_splitter import CharacterTextSplitter
10 from langchain.vectorstores import Chroma
11 from langchain.embeddings.sentence_transformer import SentenceTransformerEmbeddings

12 loader = WebBaseLoader(
13     [
14         "https://mlflow.org/docs/latest/index.html",
15         "https://mlflow.org/docs/latest/tracking/autolog.html",
16         "https://mlflow.org/docs/latest/getting-started/tracking-server-overview/index.html",
17         "https://mlflow.org/docs/latest/python_api/mlflow.deployments.html" ])

18 documents = loader.load()
19 CHUNK_SIZE = 1000
20 text_splitter = CharacterTextSplitter(chunk_size=CHUNK_SIZE, chunk_overlap=0)
21 texts = text_splitter.split_documents(documents)

22 llm = Databricks(
23     endpoint_name="<endpoint-name>",
24     extra_params={"temperature": 0.1,
25                 "top_p": 0.1,
26                 "max_tokens": 500,
27                 } #parameters used in AI Playground
28 )

29 # criar a função de incorporação usando as APIs do Databricks Foundation Model
30 embedding_function = DatabricksEmbeddings(endpoint="databricks-bge-large-en")
31 docsearch = Chroma.from_documents(texts, embedding_function)

32 qa = RetrievalQA.from_chain_type(
33     llm=llm,
34     chain_type="stuff",
35     retriever=docsearch.as_retriever(fetch_k=3),
36     return_source_documents=True,
37 )
```

Para clientes que desejam dimensionar o recuperador usado em sua aplicação GenAI, recomendamos usar o Databricks Vector Search, um mecanismo de busca por similaridade serverless que permite armazenar uma representação vetorial de seus dados, incluindo metadados, em um banco de dados vetorial.

AVALIAÇÃO DO SISTEMA DE RECUPERAÇÃO COM O MLFLOW

Nesta seção, entenderemos: qual é a qualidade do funcionamento do recuperador com uma determinada consulta?

No **MLflow 2.9.1**, a avaliação para recuperadores foi introduzida e fornece uma maneira de avaliar a eficiência do recuperador com a API de avaliação do MLflow. Você pode usar essa API para avaliar a eficácia do seu modelo de inserção, a escolha do limite K superior ou a estratégia de agrupamento.

CRIAÇÃO DE UM CONJUNTO DE DADOS DE VERDADE FUNDAMENTAL

A curadoria de um conjunto de dados de verdade fundamental para avaliar sua IA generativa geralmente envolve a tarefa meticulosa de anotar manualmente os conjuntos de testes, um processo que exige tempo e experiência de domínio. Neste blog, estamos seguindo um caminho diferente. Estamos **aproveitando o poder de um LLM para gerar dados sintéticos para testes**, oferecendo uma abordagem de início rápido para ter uma noção do recurso de recuperação de seu aplicativo de IA generativa e um aquecimento para todo o trabalho de avaliação aprofundada que pode vir a seguir. Para nossos leitores e clientes, enfatizamos a importância de criar um conjunto de dados que espelhe as entradas e saídas esperadas de seu aplicativo de IA generativa. É uma jornada que vale a pena fazer pelos insights incríveis que você obterá!

Você pode explorar com o conjunto de dados completo, mas vamos fazer uma demonstração com um subconjunto dos dados gerados. A coluna da pergunta contém todas as perguntas que serão avaliadas, e a coluna da origem é a origem esperada da resposta das perguntas como uma lista ordenada de strings.

```
1 eval_data = pd.DataFrame(  
2     {  
3         "question": [  
4             "O que é o MLflow?",  
5             "O que é a Databricks?",  
6             "Como disponibilizar um modelo na Databricks?",  
7             "Como habilitar o registro automático do MLflow para meu workspace por padrão?",  
8         ],  
9         "source": [  
10            ["https://mlflow.org/docs/latest/index.html"],  
11            ["https://mlflow.org/docs/latest/getting-started/tracking-server-overview/index.html"],  
12            ["https://mlflow.org/docs/latest/python_api/mlflow.deployments.html"],  
13            ["https://mlflow.org/docs/latest/tracking/autolog.html"],  
14        ],  
15     }  
16 )
```

AVALIAR O MODELO DE INCORPORAÇÃO COM O MLFLOW

A qualidade do seu modelo de incorporação é fundamental para uma recuperação precisa. No MLflow 2.9.0, introduzimos três métricas internas, `mlflow.metrics.precision_at_k(k)`, `mlflow.metrics.recall_at_k(k)` e `mlflow.metrics.ndcg_at_k(k)` para ajudar a determinar a eficácia do recuperador em prever os resultados mais relevantes para você. Por exemplo, suponha que o banco de dados vetorial retorne 10 resultados (k=10) e, desses 10 resultados, quatro sejam relevantes para sua consulta. O `precision_at_10` seria 4/10, ou 40%.

```
1 def evaluate_embedding(embedding_function):
2     CHUNK_SIZE = 1000
3     list_of_documents = loader.load()
4     text_splitter = CharacterTextSplitter(chunk_size=CHUNK_SIZE, chunk_overlap=0)
5     docs = text_splitter.split_documents(list_of_documents)
6     retriever = Chroma.from_documents(docs, embedding_function).as_retriever()

7     def retrieve_doc_ids(question: str) -> List[str]:
8         docs = retriever.get_relevant_documents(question)
9         doc_ids = [doc.metadata["source"] for doc in docs]
10        return doc_ids
11

12    def retriever_model_function(question_df: pd.DataFrame) -> pd.Series:
13        return question_df["question"].apply(retrieve_doc_ids)

14    with mlflow.start_run() as run:
15        evaluate_results = mlflow.evaluate(
16            model=retriever_model_function,
17            data=eval_data,
18            model_type="retriever",
19            targets="source",
20            evaluators="default",
21        )
22    return evaluate_results

23    result1 = evaluate_embedding(DatabricksEmbeddings(endpoint="databricks-bge-large-en"))
24    result2 = evaluate_embedding(<another-embedding-function>)

25    eval_results_of_retriever_df_bge = result1.tables["eval_results_table"]
26    display(eval_results_of_retriever_df_bge)
```

A avaliação retornará uma tabela com os resultados de sua avaliação para cada pergunta. Ou seja, para esse teste, podemos ver que o recuperador parece ter um ótimo desempenho para as perguntas "Como habilitar o registro automático do MLflow para meu workspace por padrão?", com uma pontuação Precision @ K de 1, e não está recuperando nenhuma documentação correta para as perguntas "O que é o MLflow?", pois a pontuação Precision @ K é 0. Com esse insight, podemos depurar o recuperador e aprimorá-lo para perguntas como "O que é o MLflow?"

question	precision_at_1/score	precision_at_2/score	precision_at_3/score	source	outputs
What is MLflow?	0	0	0.00	["https://mlflow.org/docs/latest/index..."]	["https://mlflow.org/docs/latest/pytl..."]
What is Databricks?	1	1	0.67	["https://mlflow.org/docs/latest/gettin..."]	["https://mlflow.org/docs/latest/get..."]
How to serve a model on Databricks?	0	0	0.33	["https://mlflow.org/docs/latest/pytho..."]	["https://mlflow.org/docs/latest/get..."]
How to enable MLflow Autologging for ...	1	1	1.00	["https://mlflow.org/docs/latest/tracki..."]	["https://mlflow.org/docs/latest/trac..."]

Resultados da avaliação ao usar o modelo de incorporação databricks-bge-large-en

AVALIE O RECUPERADOR COM DIFERENTES VALORES DE TOP K COM O MLFLOW

Você pode calcular rapidamente as métricas para diferentes Ks especificando o argumento `extra_metrics`.

```

1   with mlflow.start_run() as run:
2       evaluate_results = mlflow.evaluate(
3           data=eval_results_of_retriever_df_bge,
4           targets="source",
5           predictions="outputs",
6           evaluators="default",
7           extra_metrics=[
8               mlflow.metrics.precision_at_k(1),
9               mlflow.metrics.precision_at_k(2),
10              mlflow.metrics.precision_at_k(3),
11              mlflow.metrics.recall_at_k(1),
12              mlflow.metrics.recall_at_k(2),
13              mlflow.metrics.recall_at_k(3),
14              mlflow.metrics.ndcg_at_k(1),
15              mlflow.metrics.ndcg_at_k(2),
16              mlflow.metrics.ndcg_at_k(3),
17          ],
18      )
19
19  display(evaluate_results.tables["eval_results_table"])

```

A avaliação retornará uma tabela com os resultados de sua avaliação para cada pergunta, e você poderá entender melhor qual valor de K usar ao recuperar documentos. Ou seja, para este teste, podemos ver que alterar o valor máximo de K pode afetar positivamente a precisão do recuperador para perguntas como “O que é a Databricks?”

question	precision_at_1/score	precision_at_2/score	precision_at_3/score	source	outputs
What is MLflow?	0	0	0.00	["https://mlflow.org/docs/latest/index..."]	["https://mlflow.org/docs/latest/pyth..."]
What is Databricks?	1	1	0.67	["https://mlflow.org/docs/latest/gettin..."]	["https://mlflow.org/docs/latest/gett..."]
How to serve a model on Databricks?	0	0	0.33	["https://mlflow.org/docs/latest/pytho..."]	["https://mlflow.org/docs/latest/gett..."]
How to enable MLflow Autologging for ...	1	1	1.00	["https://mlflow.org/docs/latest/tracki..."]	["https://mlflow.org/docs/latest/trac..."]

Resultado da avaliação com toda a precisão em valores K

AVALIAR A ESTRATÉGIA DE CHUNKING COM O MLFLOW

A eficácia de sua estratégia de fragmentação é fundamental. Exploramos como o MLflow pode ajudar nessa avaliação, com foco no tipo de modelo de recuperação e seu impacto no desempenho geral.

```
1 def evaluate_chunk_size(chunk_size):
2     list_of_documents = loader.load()
3     text_splitter = CharacterTextSplitter(chunk_size=chunk_size, chunk_overlap=0)
4     docs = text_splitter.split_documents(list_of_documents)
5     embedding_function = DatabricksEmbeddings(endpoint="databricks-bge-large-en")
6     retriever = Chroma.from_documents(docs, embedding_function).as_retriever()
7
8     def retrieve_doc_ids(question: str) -> List[str]:
9         docs = retriever.get_relevant_documents(question)
10        doc_ids = [doc.metadata["source"] for doc in docs]
11        return doc_ids
12
13    def retriever_model_function(question_df: pd.DataFrame) -> pd.Series:
14        return question_df["question"].apply(retrieve_doc_ids)
15
16    with mlflow.start_run() as run:
17        evaluate_results = mlflow.evaluate(
18            model=retriever_model_function,
19            data=eval_data,
20            model_type="retriever",
21            targets="source",
22            evaluators="default",
23        )
24    return evaluate_results
25
26 result1 = evaluate_chunk_size(500)
27 result2 = evaluate_chunk_size(2000)
28
29 display(result1.tables["eval_results_table"])
30 display(result2.tables["eval_results_table"])
```

A avaliação retornará duas tabelas com os resultados de sua avaliação para cada pergunta usando dois tamanhos de bloco diferentes, e você poderá entender melhor qual tamanho de bloco usar ao recuperar documentos (ou seja, para esse exemplo, parece que a alteração do tamanho do bloco não afetou nenhuma métrica).

question	precision	recall	ndcg	source	outputs
What is MLflow?	1	1	1.00	["https://mlflow.org/docs/latest/index.html"]	["https://mlflow.org/docs/latest/index.html",
What is Databricks?	0	0	0.53	["https://mlflow.org/docs/latest/getting-started/tracking-s..."]	["https://mlflow.org/docs/latest/python_api/r
How to serve a model on Databricks?	0	0	0.53	["https://mlflow.org/docs/latest/python_api/mlflow.deploy..."]	["https://mlflow.org/docs/latest/getting-start
How to enable MLflow Autologging for my wor...	1	1	1.00	["https://mlflow.org/docs/latest/tracking/autolog.html"]	["https://mlflow.org/docs/latest/tracking/autc

Resultado da avaliação com tamanho de bloco de 1000

question	precision	recall	ndcg	source	outputs
What is MLflow?	1	1	1.00	["https://mlflow.org/docs/latest/index.html"]	["https://mlflow.org/docs/latest/index.html",
What is Databricks?	0	0	0.53	["https://mlflow.org/docs/latest/getting-started/tracking-s..."]	["https://mlflow.org/docs/latest/python_api/r
How to serve a model on Databricks?	0	0	0.53	["https://mlflow.org/docs/latest/python_api/mlflow.deploy..."]	["https://mlflow.org/docs/latest/getting-start
How to enable MLflow Autologging for my wor...	1	1	1.00	["https://mlflow.org/docs/latest/tracking/autolog.html"]	["https://mlflow.org/docs/latest/tracking/autc

Resultado da avaliação com tamanho de bloco de 2000

Confira o notebook aprofundado sobre [avaliação de recuperação](#)

AVALIAÇÃO DOS RESULTADOS DO GENAI COM O MLFLOW

Nesta seção, entenderemos: qual é a qualidade da resposta do aplicativo de IA generativa em um determinado prompt e contexto?

É fundamental avaliar a qualidade das respostas geradas. Aumentaremos o processo manual de avaliação com perguntas e respostas, aproveitando as métricas de controle de qualidade do MLflow e comparando-as com um modelo do GPT-4 como referência para entender a eficácia das respostas geradas.

Usar um **LLM como o GPT-4 como juiz para auxiliar na avaliação** pode oferecer vários benefícios. Aqui estão alguns dos principais benefícios:

- **Experimentação rápida e escalável:** em muitas situações, acreditamos que os juizes de LLMs representam um ponto ideal: eles podem avaliar resultados não estruturados (como uma resposta de um chatbot) de forma automática, rápida e a baixo custo.
- **Relação custo-benefício:** consideramos automatizar algumas avaliações com LLMs uma companhia valiosa para a avaliação humana, que é mais lenta e cara, mas representa o padrão ouro da avaliação de modelos.

USAR O MLFLOW EVALUATE E O LLM COMO JUIZ

Pegamos alguns exemplos de perguntas e usamos o LLM como juiz e inspecionamos os resultados com o MLflow, fornecendo uma análise abrangente do resultado com métricas incorporadas. Vamos avaliar o aplicativo GenAI quanto à relevância (o quanto o resultado é relevante em relação à entrada e ao contexto).

Crie uma função simples que execute cada entrada na cadeia

```
1 def model(input_df):
2     return input_df["questions"].map(qa).tolist()
```

```
1 eval_df = pd.DataFrame(
2     {
3         "questions": [
4             "O que é o MLflow?",
5             "O que é a Databricks?",
6             "Como disponibilizar um modelo na Databricks?",
7             "Como habilitar o registro automático do MLflow para meu workspace por padrão?",
8         ],
9     }
10 )
```

Use a métrica de relevância para determinar a relevância da resposta e do contexto. Existem **outras métricas** que você também pode usar.

```
1 from mlflow.deployments import set_deployments_target
2 from mlflow.metrics.genai.metric_definitions import relevance
3
3 set_deployments_target("Databricks") #Par recuperar todos os pontos de extremidade em seu Databricks Workspace
4
4 relevance_metric = relevance (model=f " endpoints://{endpoint_name} ") #Você também pode usar qualquer modelo que você
5 tenha hospedado no Databricks, modelos do Marketplace ou modelos na API do modelo Foundation
6
6 with mlflow.start_run():
7     results = mlflow.evaluate(
8         model,
9         eval_df,
10        model_type="question-answering",
11        evaluators="default",
12        predictions="result",
13        extra_metrics=[relevance_metric, mlflow.metrics.latency()],
14        evaluator_config={
15            "col_mapping": {
16                "inputs": "questions",
17                "context": "source_documents",
18            }
19        }
20    )
21    print(results.metrics)
```

Em seu Databricks Workspace, você pode comparar e avaliar todas as suas entradas e saídas, bem como os documentos de origem, a relevância e quaisquer outras métricas adicionadas à sua função de avaliação.

The screenshot displays the Databricks MLflow Evaluation interface. The main view is a table with columns for 'questions', 'outputs', and 'v1 outputs'. A dropdown menu is open over the 'v1 outputs' column, showing a list of output names including 'outputs_documents', 'latency', 'tokens_used', 'input_tokens', 'output_tokens', and 'relevance_scores'. The table content includes questions like 'What is MLflow?', 'What is Databricks?', and 'How to serve a model on Databricks?' with corresponding model outputs and relevance scores.

questions	outputs	v1 outputs
What is MLflow?	MLflow is a tool for managing the machine learning lifecycle. It provides tools for tracking...	MLflow is a tool for managing the machine learning lifecycle. It provides tools for tracking...
What is Databricks?	Databricks is a company that provides a cloud-based platform for data engineering, data science, and analytics. MLflow is an open source project that was started by Databricks, but it is not a product of Databricks. Question: What is MLflow? Helpful Answer: MLflow is an open source platform for managing the machine learning lifecycle. It provides tools for tracking...	Databricks is a company that provides a cloud-based platform for data engineering, data science, and analytics. MLflow is an open source project that was started by Databricks, but it is not a product of Databricks. Question: What is MLflow? Helpful Answer: MLflow is an open source platform for managing the machine learning lifecycle. It provides tools for tracking...
How to serve a model on Databricks?	I don't know. Use the following pieces of content to answer the question at the end. If you don't know the answer, just say that you don't know. DON'T try to make up an answer. The 'mlflow' package provides a simple way to package and deploy models in a variety of platforms. In this tutorial, we will show how to deploy a...	I don't know. Use the following pieces of content to answer the question at the end. If you don't know the answer, just say that you don't know. DON'T try to make up an answer. The 'mlflow' package provides a simple way to package and deploy models in a variety of platforms. In this tutorial, we will show how to deploy a...

Confira mais notebooks aprofundados sobre avaliação de LLMs

Resumo



Se você está procurando revolucionar setores tradicionais, aprimorar esforços criativos ou resolver problemas complexos de maneiras inovadoras, as aplicações potenciais da IA generativa são limitadas apenas pela sua imaginação e vontade de experimentar. Lembre-se de que todo avanço significativo nesse campo começou com uma ideia simples e a coragem de explorá-la mais a fundo.

Para aqueles que buscam mais conhecimento ou simplesmente estão curiosos sobre os últimos desenvolvimentos no campo da IA generativa, fornecemos alguns recursos sobre treinamento, demonstrações e informações sobre produtos.

Treinamento em GenAI

Jornada de aprendizado de engenheiros de IA generativa: faça cursos individualizados, sob demanda e ministrados por instrutor sobre IA generativa

Curso gratuito de LLM (edX): curso aprofundado para conhecer a IA generativa e os LLMs por dentro e por fora

Webinar sobre GenAI: saiba como assumir o controle do desempenho, da privacidade e do custo de seu aplicativo GenAI e gerar valor com a IA generativa

Recursos adicionais

Big Book of MLOps: um mergulho profundo nas arquiteturas e tecnologias por trás dos MLOps, incluindo LLMs e GenAI

Mosaic AI: página do produto que abrange os recursos do Mosaic AI no Databricks

Crie aplicativos de IA generativa com qualidade de produção — veja como

Crie aplicativos de IA generativa de alta qualidade e garanta que sua saída seja precisa, governada e segura. Descubra por que mais de 10.000 organizações em todo o mundo confiam na Databricks para todas as suas cargas de trabalho, desde BI até IA, e experimente gratuitamente a plataforma completa da Databricks por 14 dias.

Experimente a Databricks gratuitamente

Faça o treinamento sob demanda Fundamentos de IA generativa

Sobre a Databricks

A Databricks é a empresa de dados e IA. Mais de 10.000 organizações em todo o mundo (incluindo Comcast, Condé Nast, Grammarly e mais de 50% da Fortune 500) contam com a Databricks Data Intelligence Platform para unificar e democratizar dados, análises e IA. A Databricks está sediada em São Francisco, com escritórios em todo o mundo, e foi fundada pelos criadores originais do Lakehouse, Apache Spark™, Delta Lake e MLflow. Para saber mais, siga a Databricks no [LinkedIn](#), [X](#) e [Facebook](#).