Desenvolvimento e implantação dos LLMs

"A IA generativa é a chave para resolver alguns dos maiores problemas do mundo, como mudanças climáticas, pobreza e doenças. Ela tem o potencial de tornar o mundo um lugar melhor para todos".

Mark Zuckerberg[37]

Esta seção aborda os principais aspectos do processo de desenvolvimento e implantação de LLMs. Ela examina os principais componentes, como dados e arquitetura de modelos, bem como os estágios de pré-treinamento, fine-tuning e implementação. Além disso, discute os principais desafios e considerações que precisam ser levados em conta para garantir um desenvolvimento ético, robusto e alinhado com os objetivos da organização.

Principais aspectos do desenvolvimento dos LLMs

O desenvolvimento de um LLM é um processo complexo que envolve vários componentes e decisões críticas. A seguir, apresentamos os principais componentes do desenvolvimento dos LLMs que precisam ser compreendidos e alguns aspectos fundamentais dos mesmos.

Dados

Os dados são a base sobre a qual os LLMs são construídos, e sua qualidade, diversidade e representatividade têm um impacto direto sobre o desempenho e as tendências do modelo resultante. A abordagem dos desafios relacionados à propriedade intelectual, à qualidade dos dados e ao pré-processamento é essencial para o desenvolvimento de LLMs robustos, imparciais e precisos. À medida que as regulações e as práticas recomendadas nesse campo evoluem, é provável que vejamos maior ênfase no uso responsável e transparente dos dados no treinamento de LLM.

Alguns aspectos importantes dos dados de treinamento do LLM são:

- Corpus de treinamento[38]: os LLMs são treinados com grandes corpus de dados, geralmente extraídos da Internet, que incluem bilhões de palavras e abrangem uma ampla gama de domínios e gêneros, como livros, artigos de notícias, sites, mídias sociais e muito mais. Esses corpus enormes permitem que os LLMs aprendam padrões e representações de linguagem em grande escala, o que lhes dá uma capacidade sem precedentes de entender e gerar textos coerentes e contextualizados. Por exemplo, corpus comuns para treinamento incluem BookCorpus[39], Gutenberg[40], Wikipedia[41] e CodeParrot[42].

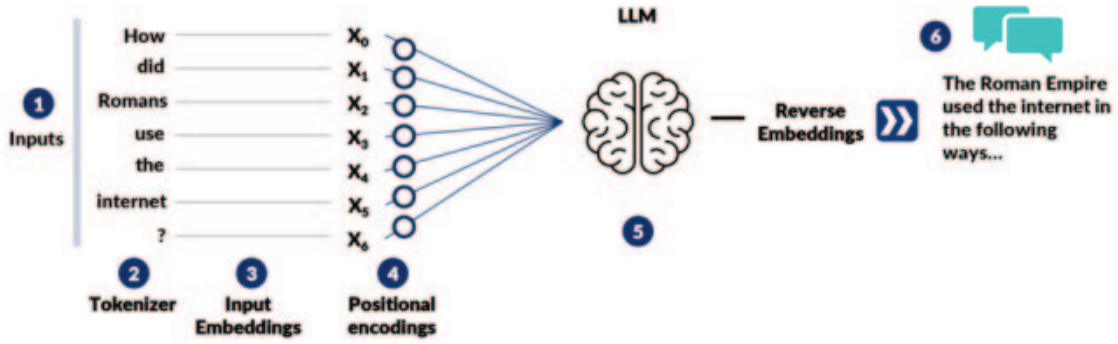
- Propriedade intelectual e direitos autorais[43]: a extração e o uso de dados da Internet para o treinamento de LLMs levanta desafios relacionados à propriedade intelectual e aos direitos autorais. Muitos desses dados são protegidos por direitos autorais, e seu uso sem permissão ou compensação adequada pode ser problemático. O AI Act na Europa aborda essa questão impondo novos requisitos aos desenvolvedores de LLM, como a obrigação de divulgar as fontes de dados usadas e a obtenção das licenças necessárias.

- Qualidade e representatividade dos dados[44]: como qualquer modelo, um LLM será tão bom quanto os dados usados em seu treinamento. Se os dados forem de baixa qualidade, tendenciosos ou não representativos, o modelo poderá herdar esses problemas e gerar resultados imprecisos, injustos ou inadequados. Portanto, é fundamental garantir que os corpus de treinamento sejam diversificados, equilibrados e representem adequadamente diferentes grupos demográficos[45], opiniões e perspectivas.

- Iniciativas de dados de alta qualidade[46]: Algumas iniciativas recentes se concentram na criação de LLM com menos parâmetros, mas com dados de maior qualidade, como corpus de treinamento menores, mas cuidadosamente

[37] Mark Zuckerberg (n. 1984), cofundador y CEO de Facebook y de Meta, una de las mayores compañías de redes sociales, tecnología e inteligencia artificial del mundo.
[38] Liu (2024).
[39] Soskek (2019).
[40] Project Gutenberg (2024).
[41] Wikipedia Dumps (2024).
[42] Hugging Face Datasets (2024).
[43] Li (2024), Chu (2023).
[44] Alabdulmohsin (2024).
[45] Yogarajan (2023).
[46] Sachdeva (2024).

Figura 5. Estágios do processamento de dados de entrada do LLM..



selecionados e filtrados [47], que incluem conteúdo de alta qualidade, como livros, artigos científicos e publicações respeitadas. Por exemplo, esses filtros podem ser limitados a um único idioma ou a um setor ou área temática, o que reduz drasticamente o tamanho do corpus. Essa estratégia pode resultar em LLMs com melhor desempenho e menos viés do que os modelos treinados em dados massivos não filtrados.seleccionados y filtrados.

- Pré-processamento e rotulagem de dados[48]: antes de treinar ou fazer o fine-tuning de um LLM, os dados devem ser pré-processados e, em alguns casos, como o fine-tuning supervisionado ou o uso de um conjunto de dados específico, rotulados. O pré-processamento envolve a limpeza e a formatação dos dados[49], a remoção de ruídos e de erros e a aplicação de técnicas como tokenização e normalização (por exemplo, LayerNorm[50] para Transformers).

Tokenização e encoding

A tokenização refere-se ao processo de dividir um texto em unidades menores chamadas "tokens", que são as unidades processadas pelo LLM durante o treinamento e a inferência de resposta. Esses tokens podem ser palavras, partes de uma palavra (por exemplo, lemas) ou caracteres. Por exemplo, uma das maneiras mais simples de gerar tokens é separar o corpus de acordo com os espaços entre as palavras. O encoding é o processo de representar essas unidades de texto em formato numérico para que possam ser processadas pelo modelo.

Alguns pontos importantes sobre tokenização no LLM:

- Isso é feito com base no corpus de texto disponível, a fim de dividir o texto de origem em unidades menores de forma otimizada. O resultado final da tokenização é um encoding.
- As codificações têm um impacto significativo no desempenho do LLM[51], pois definem a unidade mínima de processamento que receberão e determinam o vocabulário ao qual o LLM tem acesso.
- Há diferentes algoritmos de encoding no mercado[52] que diferem na forma como dividem o texto com base em palavras, frases ou sentenças, uso de espaços, capitalização ou formatação, aparência de caracteres em diferentes idiomas ou erros presentes no texto.
- As principais codificações[53] usadas são BytePairEncoding, SentencePieceEncoding e WordPieceEncoding.

O resultado da tokenização é usado como ponto de partida no modelo de embedding.

Embedding

Embeddings são representações numéricas de palavras, frases, sentenças ou mesmo parágrafos que capturam seu significado semântico e as relações entre elas. Para isso, eles partem do corpus de entrada do LLM dividido em tokens. Elas são um componente fundamental dos LLMs e desempenham um papel crucial no pré-treinamento, no fine-tuning e no uso subsequente desses modelos.

Embeddings em LLMs:

- Eles são projetados para capturar as relações semânticas entre as palavras, de modo que as palavras com significados semelhantes tenham vetores semelhantes. Isso permite que o modelo compreenda a similaridade e as analogias entre palavras e conceitos.

[47] Tirumala (2023).
[48] Chen (2023).
[49] Wenzek (2019), Penedo (2023).
[50] Zhao (2023).
[51] Rejeleene (2024).
[52] Minaee (2024).
[53] Kudo (2018).

- Eles não são valores universais, mas variam entre modelos diferentes, dependendo do espaço vetorial em que são definidos.

- Eles são contextuais, o que significa que a representação de uma palavra pode variar de acordo com o contexto em que ela aparece. Isso possibilita a captura de nuances de significado e a desambiguação de palavras polissêmicas. Eles não são predefinidos, mas são aprendidos com dados de treinamento baseados no modelo de embeddings do LLM. Durante o pré-treinamento, o modelo ajusta os embeddings para maximizar sua capacidade de prever palavras no contexto (por exemplo, por meio de estruturas de embeddings, como SentenceTransformers). Entretanto, os embeddings por si só já são um modelo que precisa ser ajustado durante o processo.

Pré-treinamento

O pré-treinamento é um estágio fundamental no desenvolvimento do LLM, durante o qual os modelos adquirem conhecimento geral e profundo do idioma a partir de grandes quantidades de dados não rotulados. Embora esse processo seja computacionalmente intensivo e caro, ele permite que o modelo seja adaptado a uma ampla gama de tarefas.

O principal objetivo do pré-treinamento é que o modelo adquira um conhecimento amplo e profundo do idioma, incluindo sua estrutura, semântica, sintaxe e contexto. Durante esse processo, o LLM aprende a prever palavras ou fragmentos de texto (ou seja, tokens) com base no contexto circundante, o que lhe permite capturar relações e padrões linguísticos complexos. Esse conhecimento geral torna-se a base sobre a qual o modelo pode ser adaptado para tarefas específicas por meio do fine tuning.

Há várias técnicas populares para o pré-treinamento do LLM, como:

- Modelagem autorregressiva da linguagem ou modelagem unidirecional (por exemplo, modelagem autorregressiva[54]), que consiste em treinar o modelo para prever a próxima palavra ou fragmento de texto com base no contexto anterior. Essa tarefa permite que o modelo aprenda as probabilidades condicionais do idioma e gere um texto coerente. Exemplos são os modelos GPT e Claude.

[54]Devlin (2018), Liu (2022).

Tipos de embeddings

Os embeddings são usados em LLMs para estabelecer uma métrica que define a semelhança entre os significados das palavras e para incorporar informações sobre a posição das palavras em uma frase. Isso é fundamental, pois a ordem das palavras afeta o significado. Há três tipos principais de embeddings posicionais:

- *Embedding* posicional absoluto[1]: atribui a cada palavra – ou a cada unidade mínima de texto ou token – um vetor que representa sua posição exata na frase (por exemplo, primeira, segunda, terceira posição etc.).
- *Embedding* posicional relativa[2]: em vez de se basear em posições absolutas, ela representa a posição de uma palavra em relação a outras palavras (por exemplo, duas palavras antes, uma palavra depois etc.).
- *Embedding* posicional rotatório[3]: combina informações de posição absolutas e relativas, usando funções trigonométricas para criar representações vetoriais mais complexas.

Em um *transformer*, um *embedding* posicional simples para uma palavra em uma determinada posição pode ser representado matematicamente usando funções de seno e cosseno. Especificamente, um embedding posicional E para um token i com posição P pode ser representado matematicamente em sua forma mais simples como:

$$E\,(P, 2i) = \sin\frac{P}{10000^{\frac{2i}{d}}}$$

$$E\,(P, 2i+1) = \cos\frac{P}{10000^{\frac{2i}{d}}}$$

onde P é a posição do token na sequência de entrada e d é a dimensão da camada oculta do *transformer*.

A escolha do tipo de *embedding* posicional pode afetar o desempenho do LLM, pois determina a quantidade e o tipo de informações posicionais disponíveis para o modelo durante o treinamento.

[1]Vaswani (2017).
[2]Shaw (2018).
[3]Su (2021).

- O modelo não autorregressivo[55], usado em modelos como o Gemini, no qual a resposta não é obtida sequencialmente palavra por palavra, mas é transformada e refinada como um todo.

- Modelagem de linguagem mascarada[56], popularizada por modelos como o BERT, que consiste em mascarar aleatoriamente algumas palavras no texto de entrada e treinar o modelo para prever essas palavras mascaradas com base no contexto ao redor. Essa técnica permite o aprendizado bidirecional e uma melhor compreensão do contexto. Algumas arquiteturas de LLM (por exemplo, transformers bidirecionais) usam essa técnica.

- Modelagem de sequência para sequência[57] (por exemplo, seq2seq[58]), em que o modelo é treinado para gerar sequências de texto com base em outras sequências de entrada. Ele é usado em modelos como T5, BART ou ProphetNET.

- Pré-treinamento contrastivo[59], usado em modelos como CLIP e ALIGN[60], envolve o treinamento do modelo para identificar pares de texto-imagem semanticamente relacionados, o que permite que ele aprenda representações multimodais e transfira conhecimento entre diferentes modalidades[61].

O pré-treinamento de um LLM é um processo computacionalmente intensivo que exige enormes quantidades de dados, tempo e recursos de hardware. Os maiores modelos podem ter cerca de 1 trilhão (10^{12}) de parâmetros e exigem milhares de GPUs de última geração para semanas ou meses de treinamento. Isso torna o pré-treinamento extremamente caro e acessível apenas para algumas empresas e organizações no mundo com os recursos necessários.

Quantificação

Durante o treinamento do LLM, os pesos dos neurônios são ajustados para fazer previsões mais precisas. Em geral, esses pesos são armazenados como números de alta precisão, o que pode resultar em modelos grandes e caros do ponto de vista computacional.

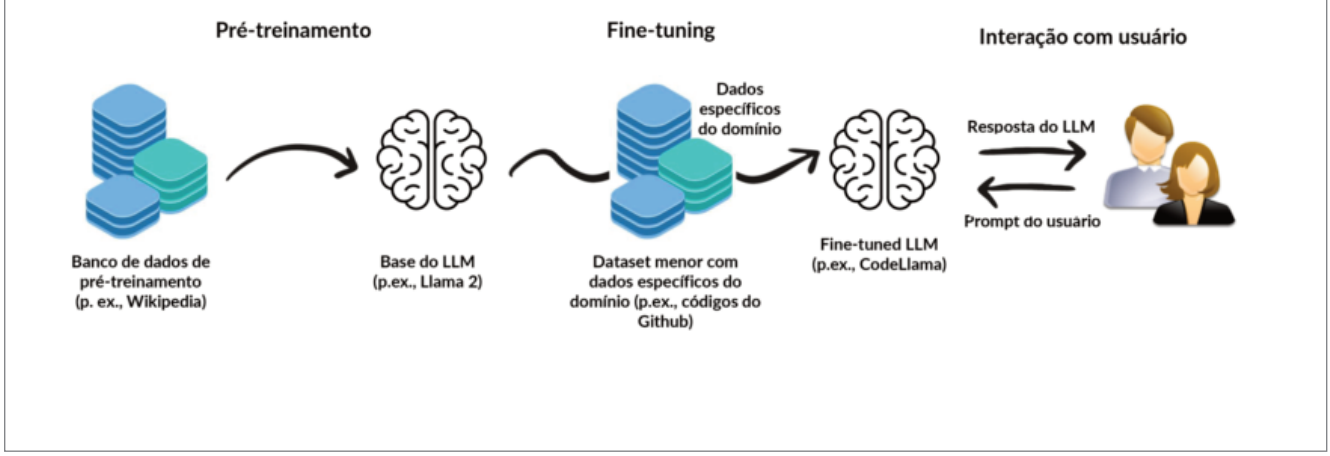
A quantização pós-treinamento é uma técnica[62] que permite que a precisão dos parâmetros do modelo seja reduzida sem afetar significativamente o desempenho do modelo. Por exemplo, as redes neurais que armazenam seus parâmetros usando números de ponto flutuante de 32 bits podem passar a usar apenas 16 bits ou 8 bits, dependendo do tipo de quantização. Isso resulta em modelos menores e mais rápidos, pois exigem menos memória e podem executar operações de forma mais eficiente com o hardware adequado.

Recentemente, houve uma tendência de desenvolver modelos de linguagem de pequena escala (SLMs), ou mesmo os chamados "tiny LLM[63]", modelos que mantêm alto desempenho apesar de seu tamanho muito menor. Esses modelos compactos são obtidos por meio de uma combinação de técnicas, incluindo a quantização pós-treinamento.

Com a aplicação hábil dessas técnicas, os SLMs e os tiny LLMs estão, em alguns casos, alcançando desempenho comparável ao de modelos muito maiores[64], o que os torna atraentes para aplicativos com restrições de recursos computacionais ou de memória.

[55] Xu (2021).
[56] Devlin (2019), Sinha (2021).
[57] Lee (2022).
[58] Sutskever (2014).
[59] Zeng (2023).
[60] Jia (2021).
[61] Cui (2022).
[62] Li (2024).
[63] Tian (2024).
[64] Fu (2024).

Figura 6. Fine-tuning de un LLM.



Fine-tuning, instruction-tuning y RAG

O fine-tuning é o processo de adaptação de um LLM pré-treinado a uma tarefa específica usando um conjunto de dados menor. Essa técnica permite aproveitar o conhecimento geral adquirido durante o pré-treinamento e especializá-lo para obter alto desempenho na tarefa-alvo.

O principal objetivo do fine-tuning (Fig. XX) é adaptar um LLM pré-treinado a uma tarefa específica, como classificação de sentimentos, resposta a perguntas, tradução automática ou geração de resumos. Durante esse processo, o modelo aprende a usar seu conhecimento geral de linguagem e a aplicá-lo de forma eficaz ao domínio específico e aos requisitos da tarefa em questão. Os LLMs disponíveis comercialmente, sejam eles proprietários ou de código aberto, geralmente são pré-treinados (e, portanto, de uso geral), mas não receberam fine-tuning, o que os adaptaria a uma finalidade específica.

O fine-tuning oferece vários benefícios significativos:

- **Aproveita o conhecimento prévio:** ao começar com um modelo pré-treinado, o fine-tuning permite aproveitar o vasto conhecimento geral do idioma adquirido durante o pré-treinamento, o que acelera o aprendizado e melhora o desempenho específico da tarefa.
- **Requer menos dados e recursos:** em comparação com o treinamento do zero, o fine-tuning requer muito menos dados rotulados e recursos computacionais, tornando-o mais acessível e econômico para uma ampla gama de organizações e aplicativos.
- **Permite a especialização:** o fine-tuning permite que os LLMs sejam adaptados a domínios e tarefas específicos, resultando em modelos altamente especializados e eficazes para aplicações específicas.
- **Facilita a transferência de aprendizado:** modelos fine-tuned podem receber fine-tuning adicionais para tarefas relacionadas, permitindo a transferência de aprendizado e a criação de modelos ainda mais especializados com relativamente poucos dados adicionais.

Apesar de seus benefícios, o fine-tuning também apresenta alguns desafios:

- **Super-especialização**[65]: se o modelo for fine-tuned em um conjunto de dados muito específico, ele poderá perder parte de sua generalização e ter um desempenho ruim com dados desconhecidos ou ligeiramente diferentes.

[65] Wang (2024).

Treinando um LLM: funções de perda

Os LLMs, como outros modelos de aprendizagem profunda, aprendem ajustando seus parâmetros para minimizar uma função de perda. Essa função mede a diferença entre as previsões do modelo e os resultados esperados, orientando o modelo para um melhor desempenho.

A escolha da função de perda depende do tipo de tarefa para a qual o LLM está sendo treinado. Por exemplo, para um modelo que prevê a próxima palavra em uma frase (modelagem de linguagem autorregressiva), uma função comum é a entropia cruzada. Essa função compara a distribuição de probabilidade das palavras previstas pelo modelo com a distribuição real observada nos dados de treinamento.

Matematicamente, a função de perda de entropia cruzada para um modelo autorregressivo pode ser expressa como uma soma dos logaritmos negativos das probabilidades atribuídas às palavras corretas em cada posição da sequência.

Especificamente, dada uma função de perda, como a entropia cruzada, e uma tipologia de treinamento, como a modelagem de linguagem autorregressiva, é possível definir a função de perda a ser minimizada como:

$$f_L(\varphi) = \sum_{i=1}^{N} -\log P(x_i \mid x_i, 1 \ldots N, \varphi)$$

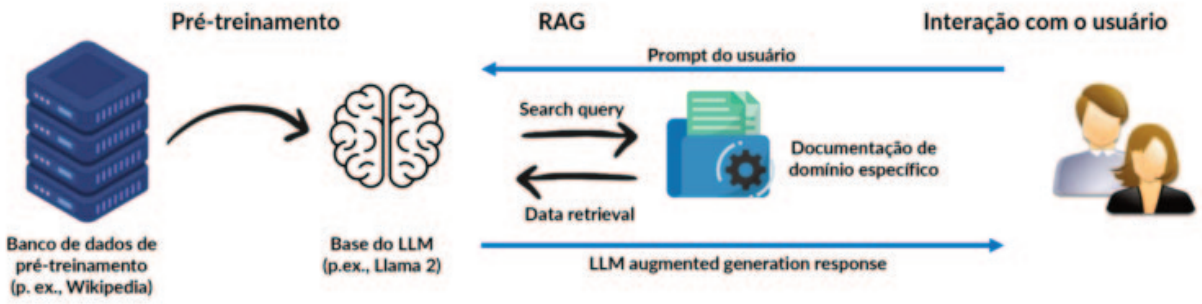
em que φ representa os parâmetros do modelo, i refere-se ao número de tokens em uma determinada sequência com N tokens, P é a probabilidade de prever o token i como uma função da sequência x de tokens anteriores.

Durante o fine-tuning dos embeddings do modelo, funções de perda especializadas podem ser usadas para ajustar as representações vetoriais das palavras. As opções mais populares incluem:

- Perda de similaridade de cosseno: ajusta as incorporações para que palavras semelhantes tenham vetores mais semelhantes.
- Perda de erro quadrático médio: minimiza a diferença quadrática entre os embeddings previstos e esperados.
- Perda de classificação negativa múltipla: associa os embeddings de palavras relacionadas de modo que elas fiquem mais próximas do que as de palavras não relacionadas.
- Tripleto, Matryoshka ou perda contrastiva: variantes mais avançadas que consideram as relações entre trios ou grupos de embeddings.

A seleção cuidadosa da função de perda é fundamental para treinar LLMs eficazes e eficientes que possam capturar as nuances da linguagem natural.

Figura 7. Funcionamiento del RAG.



- **Esquecimento catastrófico**[66]: durante o fine-tuning, é possível que um modelo esqueça o conhecimento crítico aprendido anteriormente.

- **Instabilidade**[67]: o processo de fine-tuning pode ser sensível a fatores como inicialização de pesos, hiperparâmetros e seleção de dados, o que pode levar a resultados inconsistentes ou variações no desempenho.

- **Herança de vieses**[68]: os modelos que receberam fine-tuning podem herdar e ampliar os vieses presentes nos dados de pré-treinamento e de fine-tuning, o que exige consideração e mitigação cuidadosas.

Há vários tipos de fine-tuning a serem selecionados, dependendo de quanto o modelo inicial precisa ser modificado para se adequar a uma tarefa em um domínio mais específico. Os principais métodos são:

- ***Fine-tuning supervisionado***[69]: rrequer conjuntos de dados de entrada e resposta rotulados do LLM, que são usados para melhorar sua resposta a tarefas específicas. Um método popular de fine-tuning supervisionado é o chamado instruction-tuning[70], que consiste em alinhar as respostas do modelo às expectativas de seus usuários por meio de interações com o modelo.

- **Aprendizagem por reforço:** métodos baseados na aprendizagem por reforço que se concentram em melhorar a qualidade da resposta do LLM, neste caso, com base no feedback do usuário ou em modelos de recompensa (por exemplo, otimização direta de preferências[71]).

- Fine-tuning não supervisionado[72]: sesse é um método que não exige conjuntos de dados rotulados, mas se baseia no retreinamento do modelo com as mesmas metodologias usadas durante o pré-treinamento (por exemplo, prever o próximo token).

- ***Parameter efficient***[73] ***fine-tuning* (PEFT):** outros métodos de fine-tuning visam aumentar a eficiência e reduzir o esforço necessário para treinar novamente o modelo. Por exemplo, técnicas baseadas em LoRA[74] (*low-rank adaptation*) como QLoRA ou LongLoRA[75], que permitem o fine-tuning do modelo sem a necessidade de modificar seus pesos e armazenam o conhecimento aprendido durante o processo de fine-tuning em parâmetros adicionais do modelo.

Em muitos casos de uso de LLM, não é necessário empregar o fine-tuning para melhorar seus recursos em um domínio específico. A retrieval-augmented generation[76] (RAG) é uma técnica que melhora o desempenho do LLM por meio do uso de fontes de conhecimento externas ao modelo.

As técnicas RAG (Fig. 7) funcionam por meio da busca de documentos em um banco de dados que se assemelham ou se referem ao prompt de entrada. Essa pesquisa e seus resultados são adicionados à geração da resposta do LLM para enriquecê-la, fornecendo um contexto específico.

[66]Luo (2024).
[67]Zhang (2024).
[68]Zhang (2024).
[69]Ovadia (2024).
[70]Zhang (2023).
[71]Rafailov (2023).
[72]Zhou (2023).
[73]Xu (2023).
[74]Dettmers (2023).
[75]Chen (2023).
[76]Lewis (2020) y Neelakantan (2022).

Implementação e uso

Depois de treinado e validado, o LLM deve ser implantado em um ambiente de produção para uso em aplicações reais. Isso envolve a integração do modelo aos sistemas e fluxos de trabalho existentes, bem como a criação de interfaces e APIs para interagir com ele.

Esse processo envolve vários aspectos importantes, incluindo integração e monitoramento.

Integração em sistemas e fluxos de trabalho

- **Infraestrutura**[77]: Os LLMs geralmente são modelos grandes e de computação intensiva, exigindo uma infraestrutura robusta para sua implementação. Isso pode envolver o uso de hardware especializado, como GPUs ou TPUs, e plataformas de computação em nuvem otimizadas para executar com eficiência o processo de inferência.

- **Interfaces e APIs**[78]: Para facilitar o uso do LLM em aplicativos e serviços, é necessário desenvolver interfaces e APIs que permitam que outros sistemas interajam com o modelo de forma eficiente e segura. Isso pode incluir endpoints, bibliotecas de clientes em várias linguagens de programação e interfaces gráficas de usuário para usuários não técnicos.

- **Integração com outros componentes:** em muitos casos, os LLMs fazem parte de um sistema maior que inclui outros componentes, como bancos de dados, serviços de processamento de linguagem natural e aplicativos para usuários finais. A integração suave e eficiente do LLM com esses componentes é fundamental para garantir o desempenho ideal e a experiência do usuário.

Monitoramento e manutenção

- **Monitoramento do desempenho**[79]: uma vez implementado, é essencial monitorar de perto o desempenho do LLM em condições reais. Isso envolve o rastreamento de métricas, como latência, taxa de transferência, precisão e uso de recursos, e a definição de limites de consumo e custos de recursos, além de alertas para detectar e tratar qualquer degradação ou anomalia.

- **Atualização e retreinamento**[80]: À medida que novos dados se tornam disponíveis ou áreas de melhoria são identificadas, pode ser necessário atualizar ou retreinar o LLM. Isso requer um processo bem definido para coletar e preparar novos dados, realizar o fine-tuning e implementar a versão atualizada do modelo sem interrupções de serviço.

- **Gestão de versões**[81]: Com atualizações e aprimoramentos contínuos, é importante manter um rigoroso controle de versões do LLM e de seus componentes associados. Isso facilita a reprodutibilidade, a depuração e a capacidade de reverter para versões anteriores, se necessário.

Como se pode ver, o desenvolvimento e a implantação do LLM é um processo complexo e multifacetado que exige a consideração cuidadosa de vários aspectos, desde a seleção e a preparação dos dados até a implementação e o uso responsável do modelo. Um entendimento completo dos principais componentes, como pré-treinamento, fine-tuning e embedding, bem como a conscientização dos desafios e riscos associados, é essencial para a realização de todo o potencial da LLM de forma ética, sustentável e econômica, bem como alinhada aos objetivos de cada organização.

[77] Wan (2024).
[78] Abhyankar (2024).
[79] Goyal (2024).
[80] Lester (2021).
[81] Banerjee (2023).

Arquitetura do LLM

A arquitetura do LLM refere-se à estrutura e à organização das redes neurais que compõem esses modelos. A escolha da arquitetura e de seus componentes tem um impacto significativo sobre o desempenho, a eficiência e os recursos do LLM. Esta seção explorará as principais arquiteturas usadas em LLMs e suas características, vantagens e limitações.

Transformers: o estado da arte em LLM

Os transformers, apresentados[82] em 2017, stornaram-se a arquitetura dominante para LLMs. Diferentemente das arquiteturas anteriores baseadas em redes neurais recorrentes (RNNs) ou redes neurais convolucionais (CNNs), os transformers dependem exclusivamente de mecanismos de atenção para processar e gerar sequências de texto (Fig. 8).

A arquitetura do transformer consiste em dois componentes principais: o codificador (encoder) e o decodificador (decoder), e há transformers com somente codificador, somente decodificador ou ambos os componentes. O codificador processa a sequência de entrada e gera uma representação contextual para cada token, enquanto o decodificador gera a sequência de saída a partir da representação do codificador e das previsões anteriores.

O segredo dos transformers é o mecanismo de atenção, que permite que o modelo preste atenção a diferentes partes da sequência de entrada (atenção do codificador) e a previsões anteriores (atenção do decodificador) para gerar a próxima palavra ou token. Isso permite capturar dependências de longo prazo e gerar sequências coerentes.

Os transformers também introduzem o conceito de atenção multicabeça (multi-head attention), em que vários mecanismos de atenção operam em paralelo, permitindo que o modelo capture diferentes tipos de relações e padrões nos dados.

A arquitetura do transformer demonstrou excelente desempenho em uma ampla gama de tarefas de processamento de linguagem natural e foi adotada pela maioria dos LLMs de última geração.

Variantes e extensões de transformers

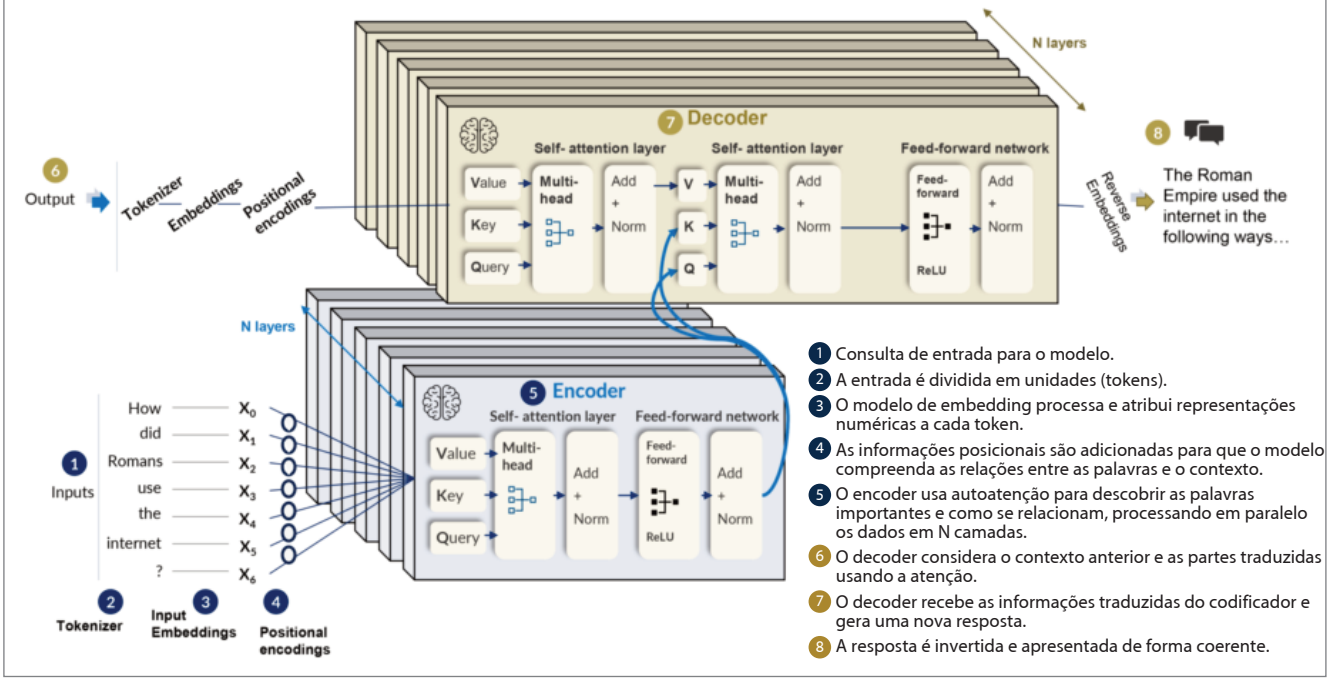
Desde a introdução dos transformers, inúmeras variantes e extensões foram propostas para melhorar sua eficiência, escalabilidade e recursos de modelagem.

- Uma variante popular é o transformer bidirecional, que permite que o modelo atenda tanto ao contexto esquerdo quanto ao direito de cada token. Isso é obtido por meio do uso de um alvo de MLM (masked language modelling, modelagem de linguagem mascarada) de pré-treinamento, em que alguns tokens são mascarados aleatoriamente e o modelo deve prevê-los com base no contexto ao redor.

- Outra variante é o transformer generativo, como o GPT, que usa uma abordagem de modelagem de linguagem unidirecional. Isso permite a geração de texto eficiente e consistente, pois o modelo só pode atender ao contexto esquerdo de cada token.

[82] Vaswani (2017).

Figura 8. Operação de um transformer.



Prompt engineering em LLMs: princípios e melhores práticas

O *Prompt engineering* se refere ao processo de projetar e otimizar os prompts (entradas de texto) para obter os melhores resultados possíveis dos LLMs. Essa disciplina emergente contém vários princípios e práticas recomendadas que permitem a exploração de todos os recursos desses modelos. Esses princípios incluem:

- Seja claro e específico: as instruções dadas ao modelo devem indicar explicitamente o formato, a extensão e o nível de detalhes esperados na resposta. Por exemplo, em vez de simplesmente pedir "Analise a situação financeira da empresa X", é melhor dar uma instrução como "Escreva um relatório de 1.000 palavras sobre a situação financeira da empresa X, abrangendo sua lucratividade, liquidez, solvência e perspectivas futuras".
- Divisão de tarefas complexas: é útil dividir os problemas em subtarefas que sejam mais gerenciáveis para os LLMs. Por exemplo, ao invés de pedir "Desenvolva um plano estratégico para a empresa Y", podem ser solicitadas subtarefas como "Realize uma análise SWOT da empresa Y", "Defina os principais objetivos estratégicos para Y", "Proponha iniciativas para atingir cada objetivo", etc.
- Fornecer exemplos ilustrativos (few-shot learning): alguns exemplos bem escolhidos podem ajudar muito a comunicar a tarefa desejada. Digamos que se queira gerar propostas de valor para produtos; dois exemplos poderiam ser dados: "Nosso software de CRM permite que as equipes de vendas fechem negócios 50% mais rápido" e "Nosso aplicativo de bem-estar ajuda os funcionários a reduzir o estresse e aumentar sua produtividade em 25%".
- Peça um raciocínio passo a passo: instruir o LLM a verbalizar seu processo de pensamento geralmente leva a resultados mais robustos. Isso é especialmente útil para tarefas analíticas ou de solução de problemas em negócios. Por exemplo: "Descreva passo-a-passo como você calcularia o ROI deste projeto de investimento".
- Solicite as referências utilizadas: instrua o LLM a indicar em seu processo de argumentação as referências aos documentos que utilizou, incluindo citações do texto original ao qual ele tem acesso.
- Pedir ao LLM para adotar uma persona: antes da tarefa principal, você pode primeiro instruir o modelo a adotar uma determinada função, tom ou estilo. Por exemplo: "Aja como um analista financeiro especializado e forneça uma avaliação objetiva da empresa X". Isso ajuda a orientar seu comportamento.
- Aproveitar o conhecimento externo: o fornecimento de informações adicionais permite complementar a base de conhecimento do LLM. Por exemplo, para responder a perguntas sobre um setor específico, os relatórios setoriais relevantes podem ser recuperados e inseridos no modelo.
- Iterar e refinar sistematicamente: a avaliação contínua do desempenho do modelo permite identificar áreas de melhoria e ajustar os prompts de acordo. Métricas quantitativas e julgamentos qualitativos de especialistas no domínio podem orientar esse processo iterativo.

Ao aplicar esses princípios de engenharia imediata, é estatisticamente comprovado que os LLMs fornecem um resultado mais preciso e confiável.

Considerando tudo isso, uma sugestão ruim para um LLM escrever uma coluna sobre engenharia imediata seria: "Escreva um artigo sobre engenharia imediata".

E uma boa sugestão para escrever essa coluna seria:

> "Aja como um especialista em inteligência artificial e escreva uma coluna de divulgação de 600 palavras sobre os princípios fundamentais do prompt engineering para obtenção dos melhores resultados dos LLMs. Estruture a coluna com uma introdução curta e envolvente, de 4 a 5 parágrafos que abordem os pontos principais (ser específico, dividir as tarefas, dar exemplos...) e uma conclusão com os benefícios da aplicação destas técnicas. Use um tom informativo, mas rigoroso, adequado para um público de negócios. Inclua exemplos concretos para ilustrar as ideias".

Fontes: Guia de prompt engineering da OpenAI[1], suporte do Anthropic Claude Opus e elaboração própria.

[1]OpenAI (2024).

- Também foram propostas extensões para tornar os transformers mais eficientes e dimensionáveis, como o transformer esparso, que usa atenção esparsa para reduzir a complexidade computacional, e o transformer comprimido, que usa técnicas de compressão para reduzir o tamanho do modelo.

Comparação com arquiteturas anteriores

Antes dos transformers, as arquiteturas dominantes para a modelagem de sequências eram as redes neurais recorrentes (RNN), como a Long Short-Term Memory (LSTM) e a Gated Recurrent Unit (GRU), e as redes neurais convolucionais (CNN).

- As RNNs podem capturar dependências de longo prazo em sequências, mas sofrem com problemas como o desaparecimento do gradiente e a dificuldade de paralelizar o treinamento. Além disso, as RNNs têm dificuldade de capturar dependências muito longas devido à sua natureza sequencial e ao uso de recorrências de intervalo constante.

- As CNNs podem capturar padrões locais em sequências e são eficientes do ponto de vista computacional, mas têm dificuldades para modelar dependências de longo prazo e exigem um tamanho de contexto fixo.

Em comparação, os transformers superam essas limitações usando mecanismos de atenção que podem capturar dependências de longo prazo de forma eficiente e paralela. Além disso, os transformers são mais flexíveis em termos de manipulação de sequências de comprimento variável e podem ser pré-treinados em grandes quantidades de dados não rotulados.

A arquitetura do transformer revolucionou o campo do LLM e permitiu avanços significativos em uma ampla gama de tarefas de processamento de linguagem natural. No entanto, ainda há desafios, como a escalabilidade, a interpretabilidade e a eficiência desses modelos. Com o avanço das pesquisas, é provável que surjam novas arquiteturas e técnicas que superem essas limitações e levem os LLMs a novos patamares de desempenho e capacidade.

LLMOps

MLOps *(Machine Learning Operations)* é uma metodologia e um conjunto de práticas projetadas para gerenciar o ciclo de vida completo dos modelos de aprendizado de máquina, desde o desenvolvimento e o treinamento até a implantação e a manutenção na produção.

Nos últimos anos, surgiu uma adaptação da metodologia MLOps especificamente voltada para LLMs, conhecida como LLMOps (Large Language Model Operations). Essa disciplina se concentra no gerenciamento eficiente de todo o ciclo de vida do LLM, desde o desenvolvimento e o treinamento até a implementação e a manutenção em ambientes de produção.

O LLMOps integra processos tradicionais de desenvolvimento de software com ferramentas e técnicas projetadas para enfrentar os desafios exclusivos apresentados por modelos de linguagem de grande escala. Alguns desses desafios incluem:

- **Gestão de grandes volumes de dados:** os LLMs exigem grandes quantidades de dados de treinamento, o que implica a necessidade de infraestruturas de armazenamento e processamento dimensionáveis e eficientes.

- **Dimensionamento de recursos computacionais:** o treinamento e a inferência de LLM exigem enormes recursos computacionais, tornando necessário o uso de técnicas de paralelização e distribuição, bem como a otimização do uso de hardware especializado, como GPUs e TPUs.

- **Monitoramento e manutenção:** uma vez implantados na produção, os LLMs devem ser monitorados de perto para detectar e corrigir problemas de desempenho, vieses, riscos, como alucinações, e degradação do modelo ao longo do tempo.

- **Controle de versões e reproducibilidade:** considerando o tamanho e a complexidade dos LLMs, é fundamental manter um controle de versões rigoroso e maximizar a reprodutibilidade de experimentos e resultados.

Para enfrentar esses desafios, o LLMOps conta com várias ferramentas e estruturas específicas, como MLFlow[83], CometML[84] e Weights & Biases[85]. Essas plataformas oferecem funcionalidades para rastreamento de experimentos, gestão de modelos, monitoramento de desempenho e colaboração em equipe.

Além disso, o LLMOps promove práticas como automação de processos, testes contínuos, documentação abrangente e governança de modelos. Isso não apenas melhora a eficiência e a qualidade do desenvolvimento dos LLMs, mas também garante seu uso ético e responsável.

Desafios

O desenvolvimento e a implementação de LLMs apresentam uma série de desafios significativos que precisam ser enfrentados para garantir seu uso responsável, ético e seguro. Esta seção explorará vários dos principais desafios para as organizações na implantação e no uso da LLM.

Vieses, alucinações e confiabilidade

Um dos maiores desafios para os LLMs é a presença de vieses e alucinações em seus resultados e previsões. Os vieses podem surgir de várias fontes, como vieses nos dados de treinamento, limitações das arquiteturas de modelos ou vieses humanos implícitos nas tarefas de anotação e avaliação. Por outro lado, as alucinações referem-se à geração de informações ou conteúdo que parecem plausíveis, mas não se baseiam em fatos reais ou no conhecimento adquirido durante o treinamento.

O viés no LLM pode se manifestar de várias maneiras, como a perpetuação de estereótipos de gênero, raça ou idade, a discriminação em tarefas de classificação ou a geração de conteúdo ofensivo ou inadequado. Esses vieses podem ter consequências graves, especialmente quando os LLMs são usados em aplicações sensíveis, como tomada de decisões jurídicas, financeiras ou médicas. As alucinações podem levar à disseminação de informações errôneas ou enganosas, o que pode ter um impacto negativo sobre a confiança dos usuários e a credibilidade dos aplicativos baseados em LLM.

Para enfrentar o desafio dos vieses, é necessário desenvolver técnicas robustas para detectar, medir e atenuar sua presença nos LLMs. Isso envolve a criação de conjuntos de dados de avaliação específicos para vieses, o uso de métricas de imparcialidade e a aplicação de técnicas de eliminação de vieses (debiasing) tanto no pré-treinamento quanto no fine-tuning. Além disso, é fundamental estabelecer processos contínuos de auditoria e monitoramento para garantir que os LLMs permaneçam imparciais ao longo do tempo.

[83] Zaharia (2018).
[84] CommetML: https://www.comet.com/
[85] Weights and biases: https://wandb.ai/site

Para lidar com as alucinações em LLMs, estão sendo desenvolvidos vários métodos que se concentram no aprimoramento dos dados de treinamento, na aplicação de técnicas robustas de regularização e no uso de feedback humano para ajustar as respostas do modelo. Além disso, estão sendo investigadas alterações arquitetônicas nos modelos para torná-los inerentemente menos propensos a alucinações. Os métodos de geração de texto e o contexto de entrada também podem ser otimizados para reduzir as alucinações. O monitoramento humano e a avaliação rigorosa são essenciais para detectar e corrigir informações imprecisas. Além disso, o desenvolvimento de ferramentas específicas, como modelos de avaliação de alucinação e técnicas de ofuscação, pode contribuir para melhorar a precisão dos LLMs.

Explicabilidade e responsabilidade

Outro grande desafio dos LLMs é sua opacidade e falta de explicabilidade. Devido à sua complexidade e à natureza de suas arquiteturas, é difícil entender como esses modelos chegam aos seus resultados.

Essa falta de transparência gera problemas de responsabilidade, especialmente quando os LLMs são usados em contextos altamente sensíveis, em que as decisões têm um impacto significativo sobre os indivíduos (por exemplo, uso de LLMs na medicina, pesquisa farmacêutica, infraestrutura crítica ou acesso ao mercado de trabalho). Sem uma compreensão clara de como esses modelos funcionam, é difícil determinar a responsabilidade em caso de erros ou comportamento indesejado.

Para enfrentar esse desafio, é necessário desenvolver técnicas e ferramentas que permitam maior interpretabilidade e explicabilidade dos LLMs. Isso inclui métodos para visualizar e analisar os mecanismos internos de atenção, técnicas de atribuição para identificar as partes mais relevantes da entrada e abordagens para gerar explicações em linguagem natural das previsões do modelo.

Além disso, é importante estabelecer estruturas claras de prestação de contas que definam as responsabilidades dos desenvolvedores, implementadores e usuários de LLM, conforme proposto na Europa pelo AI Act. Isso pode envolver a criação de padrões e diretrizes para o desenvolvimento ético da LLM, mecanismos externos de monitoramento e auditoria e canais para que as partes interessadas manifestem suas preocupações.

Confidencialidade e proteção das informações

Os LLMs geralmente são treinados com grandes quantidades de dados que podem conter informações pessoais, sensíveis ou confidenciais. Além disso, quando implantados em aplicativos do mundo real, esses modelos podem ser expostos à entrada do usuário, que também pode incluir dados privados.

Isso representa desafios significativos de privacidade e segurança, pois os LLMs podem memorizar e reproduzir informações confidenciais de seus dados de treinamento ou ficar vulneráveis a ataques que tentam extrair dados privados por meio de consultas cuidadosamente elaboradas.

Para enfrentar esse desafio, é necessário desenvolver técnicas de preservação da privacidade no treinamento e na implantação do LLM (por exemplo, o Digger[86] para detectar informações protegidas, o uso de dados fictícios[87] durante o treinamento para detectar material protegido por direitos autorais).

Além disso, é fundamental estabelecer protocolos robustos de segurança e controle de acesso para proteger os LLMs e seus dados associados contra acesso não autorizado ou uso mal-intencionado. Isso pode envolver o uso de técnicas de autenticação e autorização, monitoramento de segurança e detecção de anomalias.

[86] Li (2024).
[87] Meeus (2024).

Uso racional de recursos

O treinamento e a implementação do LLM exigem grandes quantidades de recursos computacionais, armazenamento e energia. Com modelos que chegam a centenas de bilhões ou até trilhões de parâmetros, o custo financeiro e ambiental do desenvolvimento e da operação desses sistemas pode ser muito significativo[88].

Esse alto consumo de recursos apresenta desafios de eficiência, escalabilidade e sustentabilidade. Como a demanda por LLMs maiores e mais potentes continua a crescer, é necessário encontrar maneiras de otimizar seu desempenho e reduzir o consumo de recursos.

Para enfrentar esse desafio, várias direções de pesquisa estão sendo exploradas. Uma delas é o projeto de arquiteturas de modelos mais eficientes, como o uso de mecanismos de atenção esparsos ou técnicas de compressão que reduzem o tamanho e a complexidade computacional dos LLMs sem comprometer significativamente seu desempenho.

Também estão sendo feitas pesquisas para aprimorar o pré-treinamento contínuo[89]e o *fine-tuning* continuo[90], que visam integrar a capacidade de usar informações de diferentes domínios sem a necessidade de depender de um retreinamento extenso e caro com novos dados específicos. Também estão sendo feitos progressos no uso de sistemas inovadores e no projeto de algoritmos de IA ecológicos, que abordam os custos computacionais e ambientais associados à IA (por exemplo, o sistema GreenLightningAI da Qsimov Quantum Computing[91], desenvolve retreinamento incremental e oferece interpretabilidade direta).

Outra direção é o desenvolvimento de infraestruturas e plataformas de computação mais sustentáveis, como o uso de hardware especializado de baixo consumo de energia, sistemas de resfriamento mais eficientes e fontes de energia renováveis para alimentar os data centers onde os LLMs são treinados e implantados.

Além disso, é importante promover práticas de uso racional e compartilhado de recursos, como a reutilização e a adaptação de modelos pré-treinados, em vez de treinar novos modelos do zero para cada tarefa, e o compartilhamento de recursos e conhecimentos entre organizações e comunidades de pesquisa.

Outros desafios

Entre os muitos desafios adicionais que as organizações enfrentam no desenvolvimento, na implementação e no uso do LLM, vale a pena mencioná-los brevemente devido à sua importância:

- **Dependência e lock-in:** as organizações que dependem de LLMs fornecidos por terceiros podem enfrentar riscos de dependência e aprisionamento, especialmente se os modelos forem baseados em dados ou infraestrutura proprietários. É importante considerar estratégias de diversificação e planos de contingência.

- **Riscos de segurança e uso malicioso**[92]**:** Os LLMs podem ser vulneráveis a ataques adversários, como injeção de dados envenenados ou engenharia reversa. Além disso, eles podem ser usados de forma maliciosa para gerar desinformação, spam ou conteúdo enganoso. É essencial estabelecer medidas de segurança robustas e projetar os modelos com salvaguardas contra o uso indevido.

- **Questões de propriedade intelectual e licenciamento:** o uso do LLM levanta questões sobre propriedade intelectual e licenciamento de dados de treinamento, modelos e resultados gerados. Além disso, há o risco de roubo de informações ou dados pessoais de usuários que lançam consultas ao LLM implantado em nuvens de terceiros. A conformidade com a regulamentação e as estruturas éticas é necessária para equilibrar os direitos dos criadores, dos usuários e do interesse público e, no caso das organizações, para evitar riscos legais e de compliance.

- **Escalabilidade da arquitetura LLM**[93]: Um desafio adicional é a escalabilidade dos transformers à medida que o tamanho das sequências e dos modelos aumenta. Os mecanismos de atenção têm uma complexidade quadrática com relação ao comprimento da sequência, o que limita sua aplicação a sequências muito longas.

[88] iDanae 1T24 (2024).
[89] Yıldız (2024).
[90] Mehta (2023).
[91] iDanae 1T24 (2024).
[92] Pankajakshan (2024).
[93] Rae (2021).