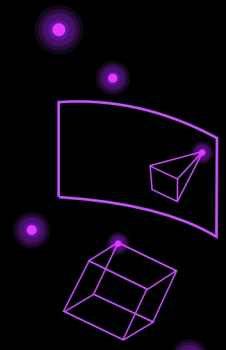
Grandes Modelos de Linguagens com APIs

Autoria:

Sávio Salvarino Teles de Oliveira
David O'Neil Campos Ferreira
Lucas Wanderley Alexandria Alves



Organizadores:

Deborah Silva Alves Fernandes
Renata Dutra Braga
Taciana Novo Kudo
Cristiane Bastos Rocha Ferreira
Arlindo Rodrigues Galvão Filho

Cegraf UFG

Grandes Modelos de Linguagens com APIs

Autoria:
Sávio Salvarino Teles de Oliveira
David O'Neil Campos Ferreira
Lucas Wanderley Alexandria Alves

Organizadores:
Deborah Silva Alves Fernandes
Renata Dutra Braga
Taciana Novo Kudo
Cristiane Bastos Rocha Ferreira
Arlindo Rodrigues Galvão Filho

Cegraf UFG

2025

Revisão Técnica
Rafael Divino Ferreira Feitosa
Rafael Teixeira Sousa

Revisão Editorial
Ana Laura de Sene Amâncio Zara Brisolla

Capa
Iuri Vaz Miranda

Editoração Eletrônica
Layane Grazielle Souza Dias
Luma Wanderley de Oliveira

https://doi.org/10.5216/OLI.gra.ebook.978-85-495-1085-3/2025

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Oliveira, Sávio Salvarino Teles de
Grandes modelos de linguagens com APIs [livro eletrônico] / Sávio Salvarino Teles de Oliveira ; organizadores Deborah Silva Alves Fernandes...[et al.]. -- Goiânia, GO : Cegraf UFG, 2025.
PDF

Outros organizadores: Renata Dutra Braga, Taciana Novo Kudo, Cristiane Bastos Rocha Ferreira, Arlindo Rodrigues Galvão Filho.
Bibliografia.
ISBN 978-85-495-1085-3

1. Ciência da Computação 2. Informática 3. Linguagem de programação (Computadores) 4. Linguagem e línguas I. Fernandes, Deborah Silva Alves. II. Braga, Renata Dutra. III. Kudo, Taciana Novo. IV. Ferreira, Cristiane Bastos Rocha. V. Galvão Filho, Arlindo Rodrigues. VI. Título.

25-257989 CDD-005.133

Índices para catálogo sistemático:

1. Linguagem de programação : Computadores : Processamento de dados 005.133

Eliane de Freitas Leite - Bibliotecária - CRB 8/8415

Grandes Modelos de Linguagens com APIs

Instituições responsáveis

Universidade Federal de Goiás (UFG)

Centro de Competência Embrapii em Tecnologias Imersivas, denominado AKCIT (Advanced Knowledge Center for Immersive Technologies)

Centro de Excelência em Inteligência Artificial (CEIA)

Instituições financiadoras

Empresa Brasileira de Pesquisa e Inovação Industrial (Embrapii)

Governo do Estado de Goiás

Empresas parceiras do AKCIT

Apoio

Universidade Federal de Goiás (UFG)

Pró-Reitoria de Pesquisa e Inovação (PRPI-UFG)

Instituto de Informática (INF-UFG)

Lista de Abreviaturas e Siglas

API	*Application Programming Interface* - Interface de Programação de Aplicação
CI/CD	*Continuous Integration and Continuous Delivery* - Integração Contínua e Entrega Contínua
CRM	*Customer Relationship Management* - Gestão de Relacionamento com o Cliente
DevOps	Desenvolvimento e Operações
Embrapii	Empresa Brasileira de Pesquisa e Inovação Industrial
FAQs	*Frequently Asked Questions* - Perguntas Frequentes
GB	Gigabytes
GPT	*Generative Pre-trained Transformer* - Transformer Generativo Pré-treinado
GPU	*Graphics Processing Unit* - Unidade de Processamento Gráfico
HTTP	*Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto
LCAP	*Low-code Application Platform* - Plataforma de Aplicação de Baixo Código
LLM	*Large Language Model* - Grandes Modelos de Linguagem
NLP	*Natural Language Processing* - Processamento de Linguagem Natural
RAG	*Retrieval-Augmented Generation* - Geração Aumentada por Recuperação
RFP	*Request for Proposal* - Solicitação de Proposta
SSO	*Single Sign-On* - Autenticação Única

TI	Tecnologia da Informação
UFG	Universidade Federal de Goiás
VWoA	*Volkswagen of America* - Volkswagen da América

Lista de Figuras e Tabelas

Sumário

Apresentação

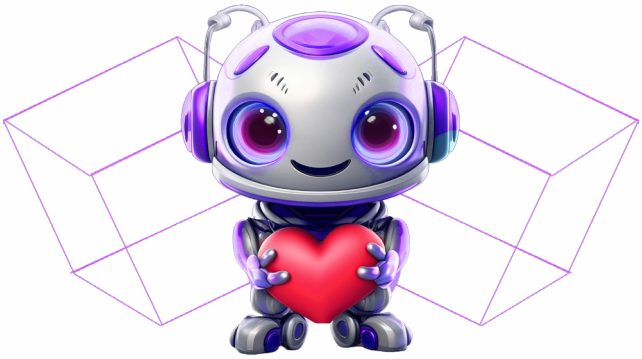
Prezado(a) Participante,

Seja bem-vindo(a) ao Microcurso "Grandes Modelos de Linguagem com Interfaces de Programação de Aplicações (APIs)". Este Microcurso faz parte da Coleção Formação e Capacitação do Centro de Competências Imersivas, uma parceria entre a Empresa Brasileira de Pesquisa e Inovação Industrial (Embrapii) e a Universidade Federal de Goiás (UFG).

A oferta deste microcurso visa suprir a crescente demanda por profissionais capacitados em lidar com os avanços e as aplicações da inteligência artificial (IA), especificamente no campo do Processamento de Linguagem Natural (*Natural Language Processing* [NLP]) e dos Grandes Modelos de Linguagem (*Large Language Models* [LLM]).

Neste curso, exploraremos os conceitos acerca dos LLMs e como eles podem ser acessados e utilizados por meio de APIs. Os LLMs representam um avanço significativo no campo do NLP, permitindo a criação de aplicações mais sofisticadas e capazes de compreender e gerar linguagem humana de forma mais natural e contextualizada.

Ao longo deste *ebook*, você aprenderá sobre os fundamentos dos LLMs, como eles funcionam e como podem ser integrados em aplicações práticas usando APIs. Abordaremos também plataformas de baixo código (*low-code*) que facilitam o desenvolvimento de soluções baseadas em LLMs, e exploraremos casos de uso reais para inspirar suas próprias aplicações.



Desejamos um excelente estudo!

Grandes Modelos de Linguagens com APIs

Unidade I

Introdução aos Grandes Modelos de Linguagem utilizando API

Unidade I - Introdução aos LLMs Utilizando API

1.1 Fundamentos de LLMs Utilizando APIs

Grandes Modelos de Linguagem (*Large Language Models* [LLMs]) são modelos de IA treinados em conjuntos massivos de dados de texto. Eles aprendem a reconhecer, traduzir, prever e gerar texto e código de forma impressionante, imitando a maneira como humanos se comunicam e interagem com a linguagem.

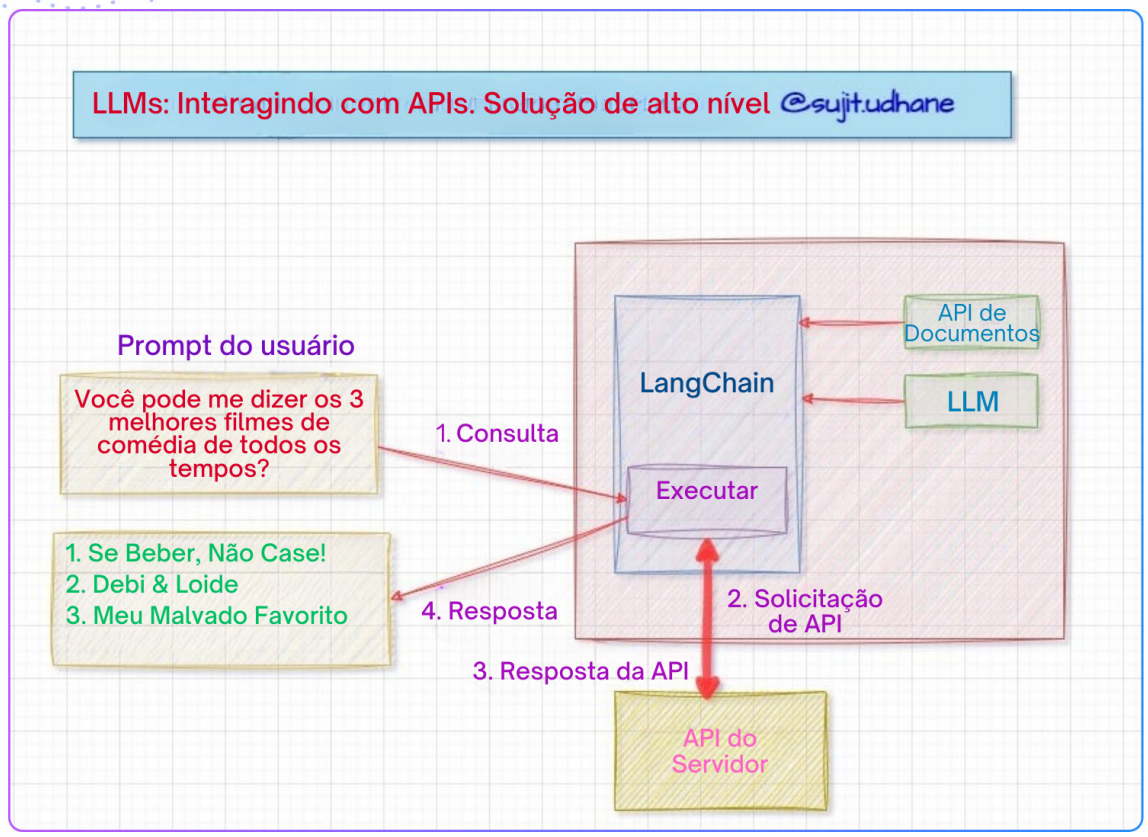
Os LLMs se baseiam em redes neurais profundas, especificamente em uma arquitetura chamada *Transformer*. Essa arquitetura permite que o modelo processe sequências de dados (como palavras em uma frase) de forma eficiente, considerando o contexto de cada elemento na sequência. Durante o treinamento, o modelo aprende a prever a próxima palavra em uma sequência, dado o contexto das palavras anteriores. Esse processo iterativo em um conjunto massivo de dados permite que o modelo desenvolva uma compreensão da linguagem.

Ao utilizar uma API, você pode acessar as capacidades desses modelos de maneira simples, enviando dados de entrada e recebendo respostas processadas pelo modelo. API significa "*Application Programming Interface*" (Interface de Programação de Aplicações). No contexto dos LLMs, as APIs são formas padronizadas de acessar e utilizar esses modelos por meio da internet.

As APIs de LLMs funcionam como uma ponte entre sua aplicação e o modelo de linguagem. Elas permitem que você envie solicitações (geralmente texto) ao modelo e receba respostas, sem precisar que você hospede ou mantenha o LLM. O processo básico de utilização de um LLM por meio de uma API envolve:

- **Autenticação**: obter credenciais de acesso à API.
- **Preparação da solicitação**: formatar o texto ou *prompt* a ser enviado.
- **Envio da solicitação**: fazer uma chamada HTTP à API com o *prompt*.
- **Processamento pelo LLM**: o modelo processa os dados, aplicando algoritmos de NLP para gerar uma resposta.
- **Recebimento da resposta**: obter e processar a resposta do LLM.
- **Integração**: utilizar a resposta em sua aplicação, conforme necessário.

Figura 1 - Processo de consulta à Interface de Programação de Aplicações (API) de um *Large Language Model* (LLM)



Fonte: traduzida de Udhane (2021).

Na Figura 1, é ilustrado um processo de alto nível que envolve o uso de um LLM e como realizar consultas e obter respostas de servidores de API. O processo começa com uma **consulta de entrada**, ou seja, uma pergunta ou comando que o usuário fornece. No exemplo da Figura 1, a pergunta feita ao LLM é: "Quais são os três melhores filmes de comédia de todos os tempos?". Essa consulta é enviada para o ***APIChain***, que é uma parte do *LangChain*, uma biblioteca que facilita a integração de modelos de linguagem com APIs. O papel do *APIChain* é intermediar a comunicação entre o LLM e a API externa.

Após receber a consulta, o *LangChain*, com base na documentação da API disponível (denotada como "API DOCS" na Figura 1), gera um **pedido de API**. Esse pedido é enviado a um servidor de API externo, que contém a informação desejada, nesse caso, uma lista de filmes de comédia.

O servidor de API processa o pedido e retorna uma **resposta da API** contendo os dados solicitados. A resposta geralmente está formatada de maneira que o LLM consiga interpretá-la facilmente. Essa etapa é essencial, pois é quando as informações solicitadas, como os títulos dos filmes de comédia, chegam ao LLM.

Após a resposta da API ser recebida, o *LangChain* e o LLM processam os dados, transformando-os em uma resposta amigável ao usuário. No exemplo da Figura 1, o sistema retorna a lista dos filmes:

"1. *When Harry Met Sally*";

"2. *Dumb and Dumber*" e

"3. *Despicable Me*".

Essa resposta final é apresentada ao usuário, completando o ciclo de interação.

Essa integração entre LLMs e APIs, demonstrada na Figura 1, permite que os modelos de linguagem acessem informações que não estão diretamente armazenadas neles. Em vez de tentar gerar todas as respostas de forma autônoma, o LLM utiliza as APIs como fontes externas de conhecimento, tornando o processo muito mais eficiente e dinâmico. O *LangChain*, nesse contexto, serve como o "intermediário", gerenciando como as consultas são feitas e como as respostas são formatadas.

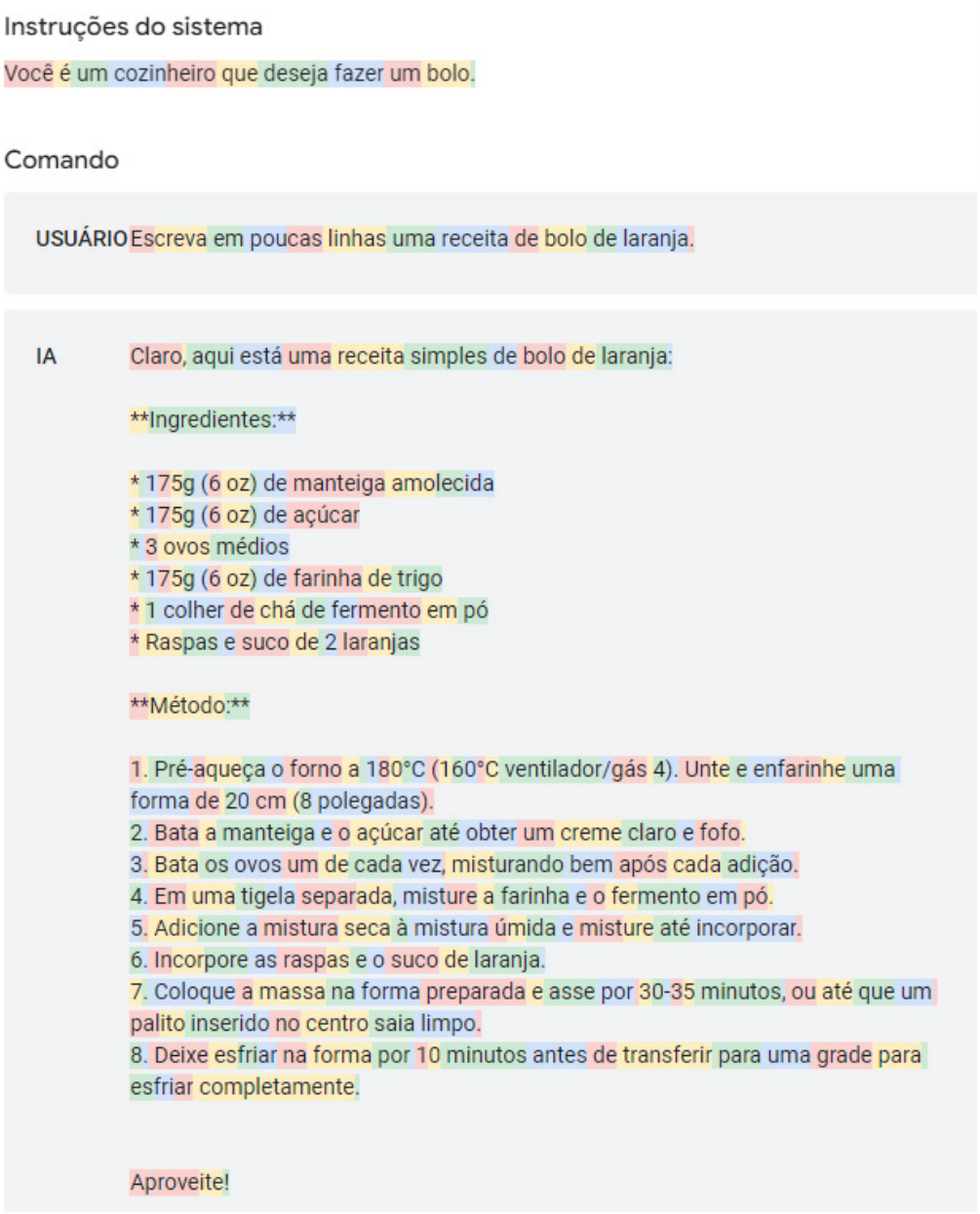
1.1.1 Tokenização em LLMs

Quando você envia uma requisição a uma API de LLM, o texto que você fornece (seja uma pergunta, um comando ou um texto para ser processado) é primeiro dividido em *tokens* pela API. A sequência de *tokens* resultante é então enviada ao modelo para processamento. Por isso, quando usamos um LLM, como os oferecidos pela OpenAI®, Google® ou Anthropic®, é comum ouvirmos falar de "*tokens*" em vez de palavras. Isso pode parecer confuso à primeira vista, mas dividir palavras em *tokens* é uma técnica fundamental para o funcionamento dos modelos de linguagem (Zhao *et al.*, 2023).

Tokens são as menores unidades de texto que um modelo de linguagem pode processar (Shanahan, 2024). Eles podem ser palavras inteiras, partes de palavras ou até mesmo caracteres individuais, dependendo da complexidade do texto e da língua que está sendo analisada.

Na Figura 2, é apresentado um exemplo do Gemini® do Google®, no qual a ferramenta disponibiliza o número de *tokens* utilizados pelo usuário durante a interação com o modelo. Na Figura 2, também é mostrado como as palavras foram divididas em *tokens*, que são destacados em cores diferentes na imagem. Observe, por exemplo, que a palavra "cozinheiro" foi dividida em dois *tokens*: "cozin" (cor azul na imagem) e "heiro" (cor vermelha na imagem).

Figura 2 - Exemplo de tokenização de texto no Gemini®



Fonte: gerado por Gemini®, em outubro de 2024.

Por que dividir em *tokens* é importante?

Cada idioma tem palavras de diferentes tamanhos e complexidades. Se os modelos de linguagem receberem palavras inteiras, é difícil garantir uma forma uniforme de processamento de textos variados. Ao dividir as palavras em *tokens*, os modelos conseguem lidar melhor com diferentes tipos de textos, variando desde palavras simples a expressões longas e complexas.

Outro desafio é que muitos idiomas têm prefixos, sufixos e radicais que se repetem com frequência. Ao transformar o texto em *tokens*, o modelo pode reconhecer padrões e semelhanças mais facilmente. Por exemplo, palavras como "inteligente" e "inteligência" compartilham a raiz "intelig", que pode ser reutilizada, permitindo uma maior eficiência no processamento.

Diferentes idiomas têm regras gramaticais e de composição de palavras bastante distintas. Por exemplo, idiomas como o alemão ou o finlandês frequentemente possuem palavras compostas muito longas e seria complicado para um modelo processar essas palavras inteiras. Ao dividir essas palavras em *tokens*, os modelos conseguem lidar com uma variedade maior de idiomas e contextos.

Na linguagem humana, novas palavras e termos técnicos surgem frequentemente. Como os modelos não são treinados com cada palavra possível, essa técnica ajudaria a tratar palavras desconhecidas dividindo-as em partes que já conhecem. Por exemplo, se um modelo não conhece a palavra "criptomoeda", ele pode dividi-la em "cripto" e "moeda", compreendendo o contexto sem precisar conhecer a palavra inteira previamente.

Por fim, processar símbolos menores é mais eficiente do que processar frases ou parágrafos inteiros. A tokenização reduz a complexidade computacional, permitindo que os modelos processem grandes volumes de texto de forma mais rápida e com menos recursos computacionais.

1.2 Comparativo de Plataformas de APIs para LLMs

Nesta seção, exploraremos as plataformas de APIs para LLMs, comparando suas características, custos e capacidades para auxiliar na escolha da melhor opção. Diversas plataformas oferecem acesso a LLMs por meio de APIs, cada uma com suas particularidades. Cada plataforma de API de LLM oferece diferentes modelos com capacidades variadas. Alguns modelos são capazes de gerar respostas mais precisas e contextualmente adequadas, enquanto outros podem ser mais limitados, mas apresentam a vantagem de consumir menos recursos computacionais.

Na Tabela 1, é apresentada uma análise de modelos das principais empresas com APIs comerciais de LLMs: OpenAI®, Google® e Anthropic®. A comparação é baseada em parâmetros como a janela de contexto (capacidade de processar informações de entrada), número máximo de *tokens* de saída (capacidade de resposta), custo por milhão de *tokens* e a performance medida em *rankings* como o Chatbot Arena® (essa medida será explicada na Seção 1.2.3).

Tabela 1 - Comparação de Grandes Modelos de Linguagem (LLMs) via Programação de Aplicações (APIs)

Empresa	Modelo	Janela de Contexto	Máximo de tokens de saída	Custo / 1M de tokens	Rank no Chatbot Arena
OpenAI	o1-preview	128.800 tokens	32.768 tokens	$15.00	1
OpenAI	GPT-4o	128.800 tokens	16.384 tokens	$2.50	7
OpenAI	GPT-4o-mini	128.800 tokens	16.384 tokens	$0.15	8
OpenAI	GPT-3.5-turbo	16.385 tokens	4.096 tokens	$0.50	75
Google	Gemini-1.5-Pro-002	2M tokens	8.192 tokens	$3.50	3
Google	Gemini-1.5-Flash-002	1M tokens	8.192 tokens	$0.07	8
Anthropic	Claude 3.5 Sonnet	200.000 tokens	8.192 tokens	$3.00	8
Anthropic	Claude 3 Opus	200.000 tokens	4.986 tokens	$15.00	20

Nota: comparação realizada em setembro/2024. Fonte: autoria própria.

1.2.1 Janela de Contexto

A janela de contexto representa a quantidade máxima de *tokens* que o modelo pode processar em uma única interação. Quanto maior a janela de contexto, mais informações o modelo poderá processar de uma só vez, o que é útil para tarefas que envolvem textos longos ou discussões complexas. Um valor maior de janela de contexto permite que o modelo se "lembre" de mais detalhes da conversa ou do documento que está sendo processado, sem perder informações importantes ao longo do tempo.

As APIs de LLMs geralmente impõem limites no número de *tokens* que podem ser processados em uma única requisição. Esses limites se aplicam tanto à entrada (o texto que você envia) quanto à saída (a resposta do modelo). Exceder esses limites resulta em erros e o texto precisa ser truncado ou dividido em múltiplas requisições. Você deve entender os limites de *tokens* da API que você está utilizando para otimizar o uso e evitar problemas. Na Tabela 1, por exemplo, observa-se uma grande variação entre os modelos, com o Gemini-1.5-Pro-002® suportando um contexto bem maior (2 milhões de símbolos) que o GPT-3.5-turbo® (16.385 símbolos).

1.2.2 Máximo de *Tokens* de Saída

A resposta do LLM, que pode ser um texto gerado, uma classificação, uma tradução, etc., também é gerada na forma de *tokens*. A API converte essa sequência de *tokens* de volta para texto legível antes de enviá-la de volta para o usuário. O parâmetro de "**máximo de *tokens* de saída**" se refere ao tamanho máximo da resposta que o modelo pode gerar. Modelos com maior limite de saída são mais adequados para tarefas que exigem respostas longas e detalhadas.

O modelo **o1-preview®** (Tabela 1) pode gerar até **32.768 *tokens*** em uma única resposta, o que o torna melhor para tarefas que requeiram grandes quantidades de saída de texto. Por exemplo, ele seria uma boa escolha para aplicações como geração de conteúdos longos ou respostas detalhadas em consultas complexas. Os modelos com capacidade de gerar **8.192 *tokens*** são suficientes para a maioria das aplicações, como assistentes virtuais avançados, redação de conteúdos de média extensão ou respostas em interações de suporte ao cliente. Os modelos **Claude 3 Opus**® e **GPT--3.5-turbo**®, com capacidade de gerar até **4.096 *tokens***, são soluções limitadas para geração de respostas longas.

1.2.3 Chatbot Arena®

O Chatbot Arena® é uma plataforma que avalia o desempenho de diferentes modelos em cenários de uso prático (Chiang, 2024). A classificação indica o desempenho relativo de cada modelo em termos de qualidade e coerência das respostas em cenários de conversação. Um *ranking* menor indica melhor desempenho em qualidade de respostas, compreensão do contexto e fluidez na geração de texto. Embora seja um indicador útil para comparação, é importante lembrar que a performance de um modelo pode variar dependendo da tarefa específica.

Na Tabela 1, podemos observar que o modelo **o1-preview®** está colocado no primeiro lugar do *ranking*. Com uma janela de **128.000 *tokens***, ele tem uma enorme capacidade de manter o contexto em conversas longas, o que melhora a fluidez das respostas e evita que informações anteriores sejam "esquecidas". O **o1-preview®** é capaz de raciocinar sobre tarefas complexas e solucionar problemas desafiadores, destacando-se pela capacidade de gerar respostas altamente coerentes e informativas em cenários complexos, o que justifica sua alta classificação.

O **Google Gemini-1.5-Pro-002®** ocupa o **3° lugar** no *ranking* do Chatbot Arena®, sendo o modelo de melhor desempenho da Google®. Com uma janela de 2 milhões de *tokens*, este é o modelo com a maior capacidade de contexto da Tabela 1. Isso o

torna excelente para diálogos extensos e aplicações que necessitam de muito contexto, como análise de grandes documentos.

O **OpenAI GPT-4o®** é classificado em **7° lugar**, o que o coloca como um dos melhores modelos disponíveis com uma ótima relação custo-benefício. Assim como o **o1-preview®**, ele possui uma janela de **128.000 *tokens***, permitindo que ele processe informações consideráveis de contexto, mas por um custo muito mais acessível do que o **o1-preview®** (**$2,50/1M de *tokens***). A posição de **7° lugar** se justifica pelo equilíbrio entre custo, desempenho e capacidade de contexto, tornando-o uma opção mais acessível para quem busca alta qualidade nas respostas dentro das soluções da OpenAI®.

Os modelos **GPT-4o-mini®** e **Gemini-1.5-Flash-002®** são classificados em **8° lugar** no Chatbot Arena®. Ambos são modelos mais acessíveis financeiramente que o **GPT-4o®** e **Gemini-1.5-Pro-002®**, respectivamente. Apesar de serem uma versão "menor", a qualidade das respostas geradas é bastante próxima aos modelos maiores, o que ajuda a justificar sua alta classificação no *ranking*. O fator custo-benefício claramente influencia o ranqueamento, tornando-os ótimas opções para quem precisa de alta capacidade de processamento com um orçamento apertado.

O **Anthropic Claude 3.5 Sonnet®** também está empatado na **8ª posição**, oferecendo um bom equilíbrio entre ética no uso da IA, qualidade e custo. Com uma janela de **200.000 *tokens***, é superior à maioria dos modelos, exceto os da Google®, o que lhe dá uma boa capacidade para manter conversas longas e processar grandes volumes de texto. A sua classificação reflete o bom equilíbrio entre qualidade e custo, além de um foco em segurança e confiabilidade.

Os modelos **Claude 3 Opus®** (**20° lugar**) e **GPT-3.5-turbo®** (**75° lugar**) não oferecem um desempenho competitivo em comparação com outros modelos, o que reflete diretamente na classificação. Além disso, as empresas Anthropic® e Google® disponibilizam modelos mais baratos e com melhor desempenho em tarefas de NLP, tornando-os opções não adequadas para o cenário atual. A Anthropic® divulgou que irá lançar, até o fim de 2024, o modelo **Claude 3.5 Opus®** com desempenho bem superior que o **Claude 3 Opus®** (Anthropic, 2024).

1.2.4 Custo por 1 Milhão de *Tokens*

O custo por 1 milhão de *tokens* é um dos fatores mais importantes, principalmente para desenvolvedores e empresas que precisam processar grandes volumes de dados sem exceder orçamentos. O custo de utilização das APIs de LLMs é frequentemente baseado no número de *tokens* processados. Modelos com maior janela de

contexto ou capacidade de saída tendem a ser mais caros, mas oferecem maior precisão e detalhamento, enquanto modelos mais simples são mais econômicos.

Na Tabela 1, é mostrado o custo por 1 milhão de *tokens* processados, tanto na entrada quanto na saída. Em termos de custo, o **GPT-4o-mini®** ($0,15/1M *tokens*) e o **Gemini-1.5-Flash-002®** ($0,07/1M *tokens*) são as opções mais acessíveis. É importante notar aqui que o preço do **Gemini-1.5-Flash-002®** é $0,07/1M *tokens* quando o *prompt* de entrada para o modelo é menor que 128.000 *tokens*, pois para *prompts* maiores que esse tamanho o valor se torna $0,15/1M *tokens*, igualando o valor do GPT-4o-mini®. Os dois modelos possuem baixo custo e também são muito eficientes, podendo entregar resultados com menor latência para os usuários finais, se comparados com os modelos mais caros. Ou seja, podem ser ótimas escolhas para quem busca modelos com bons resultados, baratos e com bom desempenho computacional.

É interessante notar que o **Claude 3.5 Sonnet®** possui custo cinco vezes menor e opera com o dobro da velocidade do **Claude 3 Opus®**. O mesmo cenário acontece na OpenAI®, onde o modelo **GPT-4o-mini®** possui custo mais de três vezes menor que **GPT-3.5-turbo®**, mas com janela de contexto maior e ranqueamento melhor em tarefas de NLP. Esse aumento de desempenho, combinado com preços econômicos, tornam os modelos **Claude 3.5 Sonnet®** e **GPT-4o-mini®** a escolha ideal em relação aos modelos **Claude 3 Opus®** e **GPT-3.5-turbo®**.

Os modelos **GPT-4o®**, **Gemini-1.5-Pro-002®** e **Claude 3.5 Sonnet®** possuem custos financeiros similares. O **Gemini-1.5-Pro-002®** é o modelo mais caro entre os três ($3,50/1M *tokens*), mas possui melhor desempenho em tarefas de NLP, além de possuir janela de contexto muito maior (2 milhões de *tokens*) que os outros dois modelos.

O modelo **o1-preview®** possui o maior custo entre todos ($15,00/1M *tokens*), mas com alta capacidade de solução de tarefas complexas de NLP, refletindo sua liderança no ranqueamento do Chatbot Arena. O modelo **o1-preview®** foi projetado para simular o processo de pensamento humano, permitindo que ele dedique mais tempo ao processamento antes de gerar uma resposta. Essa abordagem é uma novidade em relação aos modelos anteriores, que frequentemente priorizavam a velocidade em detrimento da profundidade do raciocínio. Mesmo com o custo elevado, ele é o melhor para aplicações onde a qualidade e o desempenho são importantes.

1.3 Estudos de Casos de Soluções com APIs para LLMs

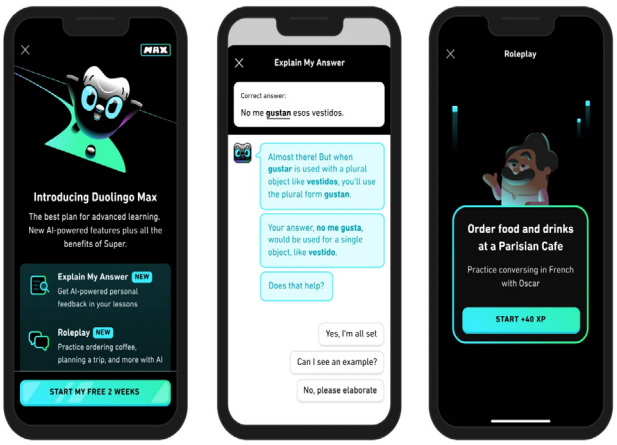
Vamos trazer três estudos de casos de soluções com APIs para LLMs: o Duolingo®, uma plataforma popular para aprendizagem de idiomas, a Volkswagen®, montadora de carros alemã, e o GitLab®, que é uma plataforma líder em gerenciamento de código com tecnologia de IA.

1.3.1 Duolingo®

O Duolingo® demonstra como LLMs, em particular o GPT-4®, podem ser integrados para criar experiências de aprendizado mais personalizadas (OpenAI, 2024). O Duolingo® oferece suporte a 40 idiomas em mais de 100 cursos. Os alunos avançam de exercícios simples de vocabulário para estruturas de frases complicadas com toques e deslizamentos em seus telefones.

O Duolingo recorreu ao GPT-4® da OpenAI® para avançar o produto com dois novos recursos (Figura 3): *Role Play*, um parceiro de conversação de IA, e *Explain my Answer*, que fornece explicações detalhadas sobre erros e acertos, adaptando a explicação ao nível do aluno.

Figura 3 - Aplicativo Duolingo® no uso da inteligência artificial para suporte ao aluno



Fonte: OpenAI (2024).

Anteriormente, o Duolingo® tentava "bater um papo" com os alunos por meio de conversas com *script* incorporando cenários clássicos como pedir comida, conhecer alguém pela primeira vez ou comprar uma passagem aérea. Mas, o GPT® permite

que o aluno interaja com personagens virtuais dentro do aplicativo, simulando diálogos em cenários da vida real. O GPT-4® permite que esses personagens respondam de forma dinâmica e contextualizada às falas do aluno, adaptando-se a diferentes estilos de comunicação e oferecendo uma experiência imersiva e personalizada.

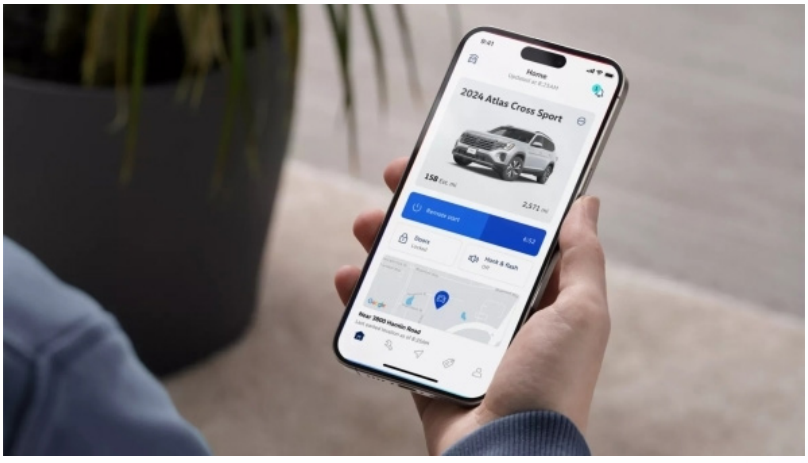
O GPT-4® também é utilizado para aprimorar as explicações de respostas em outras partes da plataforma. Quando um aluno comete um erro em um exercício, ele pode solicitar uma explicação. O GPT-4® analisa a resposta incorreta e gera uma explicação clara e concisa, adaptada ao nível de proficiência do aluno. Isso auxilia na identificação de padrões de erro e na assimilação das regras gramaticais.

A equipe do Duolingo® vê o potencial do GPT-4® para fornecer uma experiência de aprendizado mais eficaz e envolvente do que nunca, o que deve melhorar os resultados da aprendizagem. A simulação de conversas e as explicações detalhadas, impulsionadas pelo GPT-4®, permitem que os alunos pratiquem suas habilidades de comunicação de forma mais natural e aprofundem sua compreensão da estrutura do idioma. O Duolingo®, neste caso, serve como um exemplo de como a IA pode ser utilizada para democratizar o acesso à educação de alta qualidade.

1.3.2 Volkswagen®

A Volkswagen da América® (VWoA®) anunciou a implementação de IA generativa em seu aplicativo móvel myVW® (Figura 4), por meio de uma parceria estratégica com o Google Cloud® (Volkswagen of America, 2024). A montadora está aproveitando os recursos de IA e aprendizado de máquina do Google Cloud®, além da experiência da consultoria Google Cloud® para impulsionar o novo Assistente Virtual myVW®. O assistente de IA é gratuito e está disponível para cerca de 120 mil proprietários dos modelos Atlas® e Atlas Cross Sport® da Volkswagen®.

Figura 4 - Aplicativo myVW® integrado ao Gemini®



Fonte: Volkswagen of America (2024).

O objetivo é fornecer aos proprietários de veículos acesso intuitivo a informações e serviços sobre seus carros. Utilizando o Google Cloud Vertex AI®, o assistente fornece informações sobre os veículos, respondendo a perguntas como "Como trocar um pneu furado?" e explicando alertas do painel ao apontar a câmera do *smartphone*. Essa parceria combina dados do carro com modelos Gemini®, melhorando a experiência do cliente ao centralizar manuais, perguntas frequentes FAQs) e guias em um único lugar.

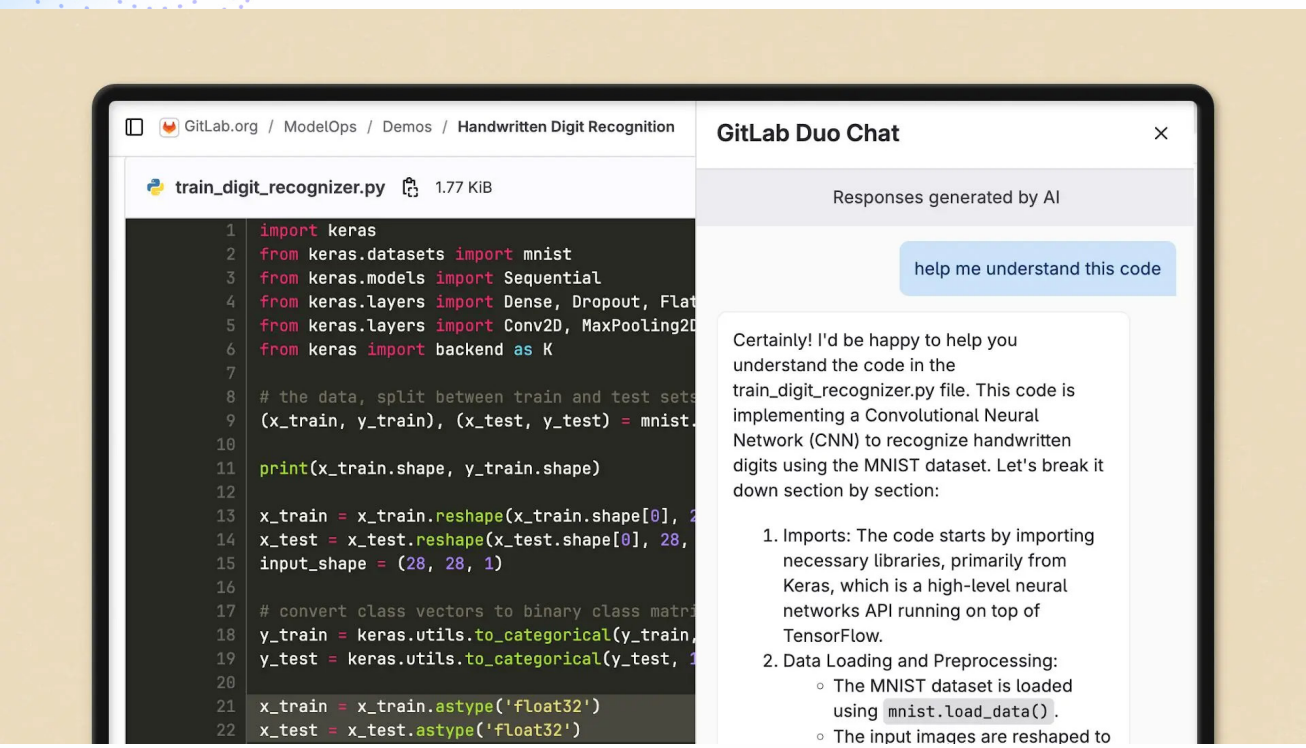
Com a combinação do Vertex AI® do Google Cloud® e do BigQuery®, a Volkswagen® é capaz de ajustar os modelos de linguagem do Gemini® em várias fontes de dados, incluindo manuais do proprietário do veículo, FAQs dos clientes, artigos da central de ajuda, vídeos oficiais da Volkswagen® no YouTube® e guias passo a passo. A Volkswagen® também aproveitou a expertise da Google Cloud® durante todo o processo de desenvolvimento do aplicativo, do *design* à implantação. Esse suporte garante que o aplicativo não apenas atenda aos requisitos legais específicos e aos altos padrões da Volkswagen®, mas também ofereça uma experiência centrada no usuário.

1.3.3 GitLab®

O GitLab® adotou o Claude® para capacitar suas equipes a criar conteúdo de alta qualidade, obter análises sobre os dados e acelerar os ciclos de vendas com o produto GitLab Duo Chat® (Linz; Alves, 2024) (Figura 5). A implementação do Claude® resultou em ganhos substanciais para o GitLab®, incluindo:

- **Alta satisfação**: 98% dos membros da equipe do GitLab® pesquisados relataram estar satisfeitos ou muito satisfeitos com o Claude for Work®.
- **Aumento de produtividade**: ganhos de produtividade entre 25% e 50%.
- **Economia de tempo**: redução significativa no tempo gasto em respostas a *Request for Proposal* (RFPs) e geração de conteúdo.
- **Melhoria na colaboração**: maior colaboração e compartilhamento de habilidades entre as equipes.
- **Maior complexidade de tarefas**: membros da equipe mais propensos a assumirem tarefas complexas.

Figura 5 - Aplicativo GitLab Duo Chat® integrado ao Claude®



Fonte: Linz e Alves (2024).

O GitLab Duo Chat® é uma ferramenta de IA integrada à plataforma GitLab®, projetada para aumentar a produtividade em todo o ciclo de vida do desenvolvimento de *software*. Lançada em 2024, ela oferece funcionalidades como explicação de código, refatoração e geração de testes automatizados. Sua interface natural melhora a colaboração e acelera o *onboarding* de novos desenvolvedores. A solução equilibra eficiência e segurança, permitindo personalização e uso em ambientes protegidos, sem comprometer a privacidade dos dados.

As seguintes funcionalidades do GitLab Duo® estão acessíveis via *chat*:

- **Escrita e explicação de código**: auxilia desenvolvedores a escreverem e entenderem código desconhecido;
- **Refatoração de código**: permite melhorar e modernizar códigos existentes; e
- **Geração de testes**: automatiza a criação de testes para funções e métodos, ajudando equipes a identificarem erros mais cedo.

O GitLab Duo® é mais do que apenas um *chatbot*. Ele representa a visão do GitLab para a integração da IA em todo o ciclo de vida *DevOps* (Shanahan, 2024). Outras funcionalidades do Duo® incluem sugestões de código em tempo real, análise

de causa raiz para falhas em *pipelines* de integração contínua e entrega contínua (CI/CD) e assistência para resolução de vulnerabilidades de segurança.

Com o *GitLab Duo*®, a privacidade do código se torna prioridade. Segundo o *GitLab*, os dados da organização não são usados para treinar os modelos de IA (Linz; Alves, 2024). O *GitLab* mantém um Centro de Transparência de IA, detalhando como a ética e a transparência são mantidas em seus recursos baseados em IA (Zhao, 2023).

O GitLab Duo Chat® representa um passo importante na direção de um desenvolvimento de *software* mais eficiente e inteligente. Ao integrar a IA diretamente na plataforma, o GitLab® capacita os desenvolvedores a se concentrarem em tarefas criativas e estratégicas, enquanto as tarefas repetitivas e complexas são automatizadas ou simplificadas com a ajuda da IA. À medida que a IA continua a evoluir, podemos esperar que o Duo Chat® se torne uma ferramenta ainda mais importante para desenvolvedores de todos os níveis.

SAIBA MAIS...

Para se aprofundar nos detalhes dos modelos de linguagem e plataformas de APIs discutidos nessa Unidade, você pode explorar as páginas oficiais das empresas que os desenvolvem. Lá você encontrará informações atualizadas sobre os modelos, suas capacidades, preços e exemplos de uso.

- OpenAI®:
  - Modelos: https://platform.openai.com/docs/models
  - Preços da API: https://openai.com/api/pricing/
- **Google®:**
  - Modelos: https://ai.google.dev/gemini-api/docs/models/gemini
  - Preços da API: https://ai.google.dev/pricing

- Anthropic®:
  - Modelos: https://docs.anthropic.com/en/docs/about-claude/models
  - Preços da API: https://www.anthropic.com/pricing

PARA RELEMBRAR...

Nesta Unidade, aprendemos os fundamentos da utilização de LLMs por meio de APIs. Vimos como os LLMs funcionam, como a tokenização funciona no processamento de texto e como utilizar APIs para acessar as capacidades dos modelos. Além disso, comparamos diferentes plataformas de APIs de LLMs, considerando fatores como janela de contexto, máximo de *tokens* de saída, custo e performance. Lembre-se dos principais pontos abordados:

- **LLMs**: modelos de IA treinados em conjuntos massivos de dados de texto, capazes de realizar diversas tarefas de NLP.
- **APIs**: interfaces de programação que permitem acessar e utilizar LLMs de forma simples e padronizada.
- **Tokenização**: processo de dividir o texto em unidades menores (*tokens*) para facilitar o processamento pelos modelos.
- **Janela de contexto**: quantidade máxima de *tokens* que o modelo pode processar em uma única interação.
- **Custo**: fator importante a ser considerado ao escolher uma plataforma de API de LLM.

AKCIT
CENTRO DE COMPETÊNCIA EMBRAPII EM TECNOLOGIAS IMERSIVAS

Grandes Modelos de Linguagens com APIs

Unidade II

Serviços comerciais para Grandes Modelos de Linguagem (LLMs)

Unidade II - Serviços Comerciais para LLMs

2.1 Visão Geral de Serviços Comerciais de API

Diversas empresas oferecem acesso a LLMs via APIs, cada uma com suas particularidades em termos de modelos, funcionalidades, custos e suporte. Esses serviços funcionam como uma ponte, conectando seus aplicativos e sistemas a LLMs, sem a necessidade de gerenciar a complexa infraestrutura por trás deles.

Uma das principais vantagens de utilizar um serviço comercial de API é a simplicidade. Em vez de se preocupar com treinamento, otimização e manutenção de um LLM, você pode concentrar seus esforços naquilo que realmente importa: construir seu aplicativo. As APIs fornecem uma interface clara e bem documentada, permitindo a integração com qualquer linguagem de programação. Por exemplo, imagine que você precise adicionar um recurso de *chatbot* ao seu *site* de *e-commerce*. Com uma API de LLM, basta enviar as perguntas dos usuários para o serviço e receber as respostas geradas pelo modelo, sem ter que lidar com os detalhes de NLP.

Outro benefício é a escalabilidade, já que os serviços comerciais de APIs são projetados para lidar com grandes volumes de solicitações, garantindo que seu aplicativo continue funcionando sem problemas, mesmo em momentos de pico de demanda. Por exemplo, você pode desenvolver um aplicativo de tradução que precisa processar milhares de frases por minuto e utilizar um serviço comercial de API que permite que o aplicativo se adapte facilmente a essas flutuações, sem a necessidade de investir em infraestrutura adicional.

Ao comparar a execução de um LLM localmente com o uso de APIs, o custo é um fator a ser analisado com cuidado. A princípio, APIs podem parecer mais acessíveis, pois eliminam o investimento inicial em *hardware*. Contudo, os custos de uso, cobrados por solicitação ou por *token*, podem acumular rapidamente, tornando-se mais caros em longo prazo, especialmente com uso intenso. Por outro lado, investir em *hardware* para executar um LLM localmente representa um gasto inicial significativo, principalmente se forem necessárias unidades de processamento gráfico (GPUs) potentes. Porém, em longo prazo, essa opção pode ser mais econômica, já que não há cobranças por uso. A decisão ideal depende do seu orçamento, da frequência de uso e das suas necessidades específicas.

A dependência do provedor da API também pode ser um fator a considerar. Alterações nas políticas de uso ou interrupções no serviço podem afetar o funcionamento do seu aplicativo. Por fim, a segurança dos dados é algo que deve ser analisado com cuidado. Ao enviar informações para um serviço externo, é importante certificar-se de que o provedor da API possui políticas de segurança para proteger seus dados.

2.2 Plataformas Comerciais para LLMs Proprietários

Nos últimos anos, várias plataformas comerciais surgiram para fornecer acesso a LLMs de maneira acessível e flexível. Essas plataformas oferecem APIs que permitem integrar IA a diversos tipos de aplicações, sem a necessidade de treinar modelos. Entre os modelos mais conhecidos no mercado estão o OpenAI®, Gemini®, Grok® e Claude®, que apresentam diferentes abordagens e especializações para resolver problemas reais em negócios e tecnologia.

Essas plataformas possuem um ponto em comum: **os LLMs são proprietários da empresa e não são divulgados ao público em geral**. Nesta seção, exploraremos essas plataformas, suas características, benefícios e como elas podem ser usadas para implementar soluções de IA.

Todos os exemplos de códigos serão apresentados utilizando a biblioteca sugerida por cada empresa (Google®, OpenAI® e Anthropic®), porém, você poderá executar chamadas às APIs dos modelos dessas empresas, utilizando bibliotecas como LangChain® (LangChain, 2024) e LlamaIndex® (LlamaIndex, 2024).

2.2.1 OpenAI®

OpenAI® é uma das plataformas mais conhecidas e oferece acesso aos seus modelos GPT®, como o GPT-4®, além de modelos para geração de código, imagens e *embeddings*. Uma das principais vantagens do OpenAI® é a capacidade de lidar com diversas tarefas em linguagem natural, seja para compreensão de texto, geração de linguagem ou até mesmo tradução.

A OpenAI® também oferece um serviço de *fine-tuning*, onde empresas podem ajustar os modelos para contextos específicos, aumentando a precisão em casos de uso mais especializados. Outra vantagem da plataforma é a documentação detalhada, tornando a integração e o uso dos modelos mais fáceis até para desenvolvedores iniciantes. Você pode seguir o tutorial da OpenAI® para adquirir sua chave de API, caso queira executar o código abaixo.

Exemplo de uso:

Aqui está um exemplo de código em Python® que utiliza a API da OpenAI® para criar um assistente virtual personalizado. Esse assistente é capaz de entender consultas complexas e gerar respostas contextuais, podendo ser aplicado em diferentes áreas, como saúde, educação ou suporte técnico. Você pode personalizar o código:

- O contexto pode ser ajustado para diferentes áreas, como "finanças", "jurídico", "*marketing*", ou qualquer outra especialização que você precise.
- O modelo utilizado é o `gpt-4o-mini®`, mas você pode ajustá-lo para outras versões, como o `gpt-4o®`, dependendo da sua necessidade.
- A configuração da temperatura permite ajustar o quão criativas ou precisas as respostas serão.

Código:

Instale a biblioteca **openai** na versão 1.51.2:

```
Unset
pip install openai==1.51.2
```

Depois de instalada a biblioteca, execute o código no Google Colab®:

```
Python
import openai


# Chave de API da OpenAI - Substitua pela sua chave
api_key = 'SUA_CHAVE_API'


#Configura o cliente da OpenAI
client = openai.OpenAI(api_key = api_key)


# Exemplo de uso para um assistente de saúde
consulta_usuario = "Meu computador não liga, o que posso fazer?"
# Faz a construção do código que irá chamar a api
resposta = client.chat.completions.create(
  model="gpt-4o-mini",
```

continua

```
  messages=[
    {"role": "user", "content": f"Responda de forma clara e
objetiva a seguinte pergunta: {consulta_usuario}"}
  ]
)

# Imprime a resposta
print("Resposta do Assistente:",
resposta.choices[0].message.content)
```

2.2.2 Google®

A plataforma Gemini®, do Google®, oferece uma suíte de modelos de linguagem com diferentes capacidades, desde modelos mais leves para dispositivos móveis até modelos maiores para tarefas complexas. A plataforma inclui ferramentas para ajuste fino, desenvolvimento de *prompts* e integração com outros serviços do Google Cloud®. A API do Gemini® permite que os desenvolvedores integrem LLMs em aplicações que demandam NLP em grande escala, seja para análise de sentimentos, automação de conteúdo ou tradução automática.

Uma das principais características do Gemini® é sua integração com o Google Cloud®, que permite uma utilização em conjunto com outros serviços de infraestrutura em nuvem. Isso proporciona grande flexibilidade, escalabilidade e segurança para empresas que já usam a Google Cloud Platform® em suas operações.

No exemplo a seguir, você vai precisar de uma chave do Gemini®. Você pode gerar sua chave no Gemini® neste *link*, caso queira executar o código.

Exemplo de Uso:

Este exemplo demonstra como usar o Gemini para criar um assistente virtual especializado em suporte técnico, capaz de responder a perguntas sobre instalação e manutenção de computadores. Você pode personalizar o código:

- *Modelos*: você pode experimentar com outros modelos Gemini disponíveis, alterando o parâmetro *model* na chamada *text.generate_text*.
- *Temperatura*: ajuste o parâmetro *temperature* para controlar a criatividade do resumo.
- *max_output_tokens*: ajuste esse parâmetro para controlar o comprimento máximo da saída do modelo.

Código:

Instale a biblioteca `google-generativeai` na versão *0.8.3*:

```
Unset
pip install -q -U google-generativeai==0.8.3
```

Depois de instalada a biblioteca, execute o código no *Google Colab*:

```
Python
import google.generativeai as genai

# Configura a chave do Gemini - Substitua pela sua chave
genai.configure(api_key="SUA_CHAVE_API")

# Define o modelo do Gemini
model = genai.GenerativeModel('gemini-1.5-flash')

# Configuração da API da LLM
generation_config = genai.GenerationConfig(
        max_output_tokens=1000,
        temperature=0.2,
    )

# Pergunta do cliente ao chatbot
consulta_usuario = "Meu computador não liga, o que posso fazer?"

# Envio da pergunta e impressão da resposta
resposta = model.generate_content(
    f"Responda de forma clara e objetiva a seguinte pergunta: {consulta_usuario}",
    generation_config = generation_config
)
print(resposta.text)
```

2.2.3 Anthropic®

Claude, desenvolvido pela Anthropic, é um modelo de LLM desenhado com foco em segurança e comportamento ético. Uma das principais propostas da plataforma Claude é proporcionar uma abordagem mais segura e previsível para o uso de IA, especialmente em setores onde o comportamento dos modelos pode ter grandes consequências. A plataforma foi criada para mitigar riscos de comportamento indesejado dos modelos, usando princípios de IA responsável e ética.

Por isso, Claude é uma boa escolha para empresas que lidam com informações sensíveis, como no setor jurídico, financeiro ou governamental, onde a segurança e previsibilidade do modelo são importantes. A plataforma da Anthropic oferece acesso ao Claude por meio de APIs e permite o ajuste fino para diferentes tarefas.

Exemplo de uso:

Este exemplo demonstra como usar o Claude® para criar um assistente virtual especializado em responder a perguntas sobre manutenção de computadores. Você pode personalizar o código:

- ***Prompt***: define o papel do Claude® e fornece contexto para a pergunta.
- **Modelo**: você pode usar outros modelos Claude® disponíveis.
- ***max_tokens_to_sample***: controla o comprimento máximo da resposta.
- **Temperatura**: controla a criatividade da resposta.

Código:

Instale a biblioteca `anthropic` na versão 0.36.0:

```
Unset
pip install anthropic==0.36.0
```

Depois de instalada a biblioteca, execute o código no Google Colab®:

```
Python
from anthropic import Anthropic


# Inicia o cliente e configura a chave de API
client = Anthropic(
```

continua

```
    api_key="SUA_CHAVE_API",
)

# Pergunta do cliente ao chatbot
consulta_usuario = "Meu computador não liga, o que posso fazer?"

# Envio da pergunta
resposta = client.messages.create(
    model="claude-3-5-sonnet-20240620",
    max_tokens=1024,
    messages=[
        {"role": "user", "content": f"Responda de forma clara
e objetiva a seguinte pergunta: {consulta_usuario}"}
    ]
)
# Imprime a resposta
print(resposta.content)
```

2.3 Plataformas Comerciais com APIs para LLMs com Código Aberto

Diversas empresas estão construindo plataformas que facilitam o acesso e utilização de LLMs com código aberto (*open-source*) por meio de APIs. Essas plataformas oferecem uma alternativa à construção e manutenção de infraestrutura própria, principalmente para usuários iniciantes ou com recursos computacionais limitados. LLMs de código aberto precisam de grandes recursos de GPU para executarem. Por exemplo, o modelo **Meta Llama 3.1 405B com FP16**® requer mais de 800 GB de memória de GPU, o que está disponível em ambientes com supercomputadores, caros e com manutenção complexa.

As plataformas abstraem a complexidade da configuração, otimização e execução dos modelos, permitindo que os desenvolvedores se concentrem na integração e aplicação dos LLMs em seus projetos. Em vez de investir tempo e recursos para configurar um servidor potente e instalar todas as dependências necessárias para rodar um modelo LLM, você pode simplesmente usar a API de uma plataforma especializada e começar a desenvolver sua solução.

Essas plataformas normalmente oferecem uma variedade de modelos pré-treinados, permitindo que os usuários escolham aquele que melhor se adapta às suas

necessidades, considerando fatores como tamanho, desempenho e tarefa específica. Além disso, algumas plataformas oferecem recursos adicionais, como ajuste fino (*fine-tuning*) dos modelos com dados específicos do usuário, monitoramento do uso e ferramentas para análise de performance.

Pense, por exemplo, num pesquisador que está trabalhando com um conjunto de dados específico de textos médicos. Ele pode usar uma plataforma que permite o ajuste fino de um LLM com seus dados, resultando em um modelo mais preciso para a tarefa em questão.

Existem diversas plataformas que oferecem acesso a LLMs com código aberto via API, cada uma com suas particularidades e vantagens. Vamos explorar apenas alguns exemplos:

- **Together.ai**®: disponibiliza uma API que permite acesso a vários LLMs de código aberto. A plataforma é fácil de usar e disponibiliza uma variedade de modelos, desde modelos menores para tarefas mais simples até modelos de grande porte com alto desempenho (Together AI, 2024).
- **Groq**®: adota uma abordagem diferente, focando em *hardware* especializado para acelerar a execução de modelos LLMs. A plataforma oferece uma API que permite aos usuários executar modelos de código aberto em seus *chips* otimizados, obtendo, assim, um ganho significativo de performance.
- **Hugging Face**®: uma das plataformas mais populares, Hugging Face® oferece APIs para uma vasta gama de modelos de código aberto, desde LLMs até modelos de visão computacional. A plataforma permite tanto o uso de modelos hospedados em sua infraestrutura quanto a opção de rodá-los localmente (Hugging Face, 2024).
- **Fireworks.ai®:** oferece uma plataforma especializada em hospedar e servir LLMs, permitindo que desenvolvedores criem aplicações interativas e de baixa latência. Ela oferece serviços de ajuste fino dos modelos para tarefas específicas com um custo baixo em relação a outros provedores (Fireworks AI, 2024).
- **Google**®: disponibiliza uma solução, denominada Model Garden®, que oferece uma variedade de modelos, desde os voltados para tarefas específicas, como tradução e classificação de texto, até LLMs e multimodais. Ele possui uma biblioteca de modelos abertos que o usuário pode escolher para colocar em produção (Google Cloud, 2024).
- **AWS**®: a Amazon® disponibiliza um serviço (Amazon Bedrock®) totalmente gerenciado que oferece várias opções de modelos de base de alta performance das principais empresas de IA, como Anthropic®, Cohere®, Meta®, Mistral AI® e Amazon®, por meio de uma única API, além de um amplo conjunto de recursos necessários para criar aplicações de IA generativa com segurança, privacidade e IA responsável (Amazon Web Services, 2024).

2.3.1 Exemplo de Código com Together.AI®

O objetivo da Together.ai® é fornecer uma infraestrutura que centralize modelos de IA de código aberto, facilitando o acesso e a execução desses modelos em escala. A plataforma opera com o princípio de colaboração aberta, onde a comunidade de IA pode compartilhar e utilizar modelos e dados. Ela disponibiliza modelos de linguagem que foram desenvolvidos e disponibilizados com código aberto por diferentes grupos de pesquisa e empresas. Esses modelos estão prontos para uso e podem ser acessados por meio de APIs.

A Together.ai® facilita o acesso aos modelos por meio de uma interface de API unificada. Com poucas linhas de código, qualquer desenvolvedor pode fazer chamadas a LLMs para realizar tarefas como tradução automática, resumo de textos, classificação de sentimentos ou geração de conteúdo. A API permite que desenvolvedores de diferentes níveis de habilidade possam rapidamente integrar a IA em suas aplicações, sem a necessidade de entender os detalhes técnicos de como os modelos são treinados e executados.

Além de fornecer modelos pré-treinados, a Together.ai® oferece recursos para ajuste fino com dados específicos do usuário. Isso permite adaptar os modelos a tarefas e domínios específicos, melhorando sua performance em cenários particulares. Por exemplo, um pesquisador que precisa de um modelo especializado em linguagem médica pode usar a Together.ai® para ajustar um modelo pré-treinado com um conjunto de dados de textos médicos, obtendo um modelo mais preciso para sua aplicação.

Exemplo de uso:

Aqui iremos apresentar um exemplo utilizando o modelo **Llama 3.1 8B Turbo**® executando na plataforma da Together.ai® para criar um assistente virtual especializado em responder a perguntas sobre manutenção de computadores. Nesse exemplo, a plataforma fica responsável por executar e manter o modelo em operação.

Código:

Instale a biblioteca `together` na versão 1.3.1:

```
Unset
pip install together==1.3.1
```

Depois de instalada a biblioteca, execute o código no Google Colab®:

```
Python
import os
from together import Together

# Inicia o cliente e configura a chave de API
client = Together(api_key='SUA_CHAVE_API')

# Pergunta do cliente ao chatbot
consulta_usuario = "Meu computador não liga, o que posso fazer?"

# Envio da pergunta
response = client.chat.completions.create(
    model="meta-llama/Meta-Llama-3.1-8B-Instruct-Turbo",
    messages=[{"role": "user", "content": f"Responda de forma
clara e objetiva a seguinte pergunta: {consulta_usuario}"}],
    max_tokens=512,
    temperature=0.7
)

# Imprime a resposta
print(response.choices[0].message.content)
```

2.4 Casos de Uso

Os serviços comerciais de APIs de LLMs podem ser aplicados em diversas situações do mundo real, desde a automação de atendimento ao cliente até a análise de dados em larga escala. A flexibilidade dessas APIs permite que empresas e desenvolvedores criem soluções escaláveis de forma rápida, aproveitando o potencial da IA sem precisar construir e treinar modelos do zero. Nesta seção, vamos explorar três casos de uso em que serviços comerciais de LLMs podem ser integrados para resolver problemas práticos.

Além disso, forneceremos exemplos de código que mostram como cada uma dessas soluções pode ser implementada utilizando a API do modelo Gemini®. Para saber mais sobre como trabalhar com a API Gemini®, consulte o tutorial Python® (Google AI for Developers, 2024a).

Se você é novo em modelos de IA generativa, talvez queira dar uma olhada no guia de conceitos (Google AI for Developers, 2024b) e na visão geral da API Gemini® (Google AI for Developers, 2024c). Em todos os exemplos a seguir, você vai precisar de uma chave do Gemini®. Você pode gerar sua chave no Gemini® neste *link*.

2.4.1 Suporte Automatizado ao Cliente

Empresas que lidam com grandes volumes de perguntas repetitivas, como lojas de *e-commerce*, podem beneficiar-se da automação do atendimento ao cliente. Um assistente virtual pode ser configurado para responder automaticamente às perguntas mais comuns, como status de pedidos, política de devolução e informações sobre produtos.

Exemplo:

Um cliente pergunta: "Como posso redefinir minha senha?". O *chatbot*, usando a API do Gemini®, pode entender a pergunta e responder com instruções passo a passo para redefinir a senha.

Código:

Instale a biblioteca google.generativeai por meio do comando a seguir para executar o código:

```
Unset
pip install -q -U google-generativeai==0.8.3
```

```
Pyhton
import google.generativeai as genai
import os

# Configura as chaves de acesso ao Gemini
os.environ["GOOGLE_API_KEY"] = "SUA_CHAVE_API"
```

continua

```
genai.configure(api_key=os.environ["GOOGLE_API_KEY"])

# Define o modelo do Gemini
model = genai.GenerativeModel('gemini-1.5-flash')

# Pergunta do cliente ao chatbot
pergunta_cliente = "Como posso redefinir minha senha?"

# Configuração da API da LLM
generation_config = genai.GenerationConfig(
        max_output_tokens=1000,
        temperature=0.2,
    )

# Envio da pergunta e recebimento da resposta
resposta = model.generate_content(
    pergunta_cliente,
    generation_config = generation_config
)
print(resposta.text)
```

2.4.2 Análise de Sentimentos em Redes Sociais

Empresas podem monitorar a percepção do público sobre seus produtos ou serviços nas redes sociais utilizando APIs de LLMs para realizar a análise de sentimentos. Isso permite identificar o *feedback* positivo ou negativo de clientes, ajudando a ajustar estratégias de comunicação ou *marketing*.

Exemplo:

Uma empresa monitora menções à sua marca no X® (antigo Twitter®). A API do Gemini® pode analisar o sentimento de cada postagem (antigo *tweet*), classificando-a como positiva, negativa ou neutra.

Código:

Instale a biblioteca google.generativeai por meio do comando a seguir para executar o código:

```
Unset
pip install -q -U google-generativeai==0.8.3
```

```
Pyhton
import google.generativeai as genai
import os

# Configura as chaves de acesso ao Gemini
os.environ["GOOGLE_API_KEY"] = "SUA_CHAVE_API"
genai.configure(api_key=os.environ["GOOGLE_API_KEY"])

# Define o modelo do Gemini
model = genai.GenerativeModel('gemini-1.5-flash')

# Pergunta do cliente ao chatbot
texto = "Adorei o novo produto! Super recomendo."

# Prompt a ser enviado ao modelo
prompt = f"""
    Classifique o sentimento do seguinte texto como Positivo, Negativo ou Neutro:
    "{texto}"
    """
# Configuração da API da LLM
generation_config = genai.GenerationConfig(
        max_output_tokens=1000,
        temperature=0 # Temperatura zero para respostas mais objetivas
    )

# Envio da pergunta e recebimento da resposta
resposta = model.generate_content(
    prompt,
```

continua

```
    generation_config = generation_config
)
print(resposta.text)
```

Notebook Colab

Resumo do Colab

Neste colab, exploraremos como os modelos de LLM podem ser aplicados em problemas comerciais, abrangendo plataformas como OpenAI, Gemini da Google, e a Together. Vamos aprender a obter as API keys dessas ferramentas, entender a configuração de parâmetros como a temperatura e ver exemplos práticos de uso. A partir disso, discutiremos como esses modelos podem ser integrados a serviços comerciais, com um enfoque em como resolver desafios reais do comércio, ajudando empresas a otimizar processos e oferecer soluções inovadoras.

Google Gemini

O Gemini, da Google, é uma plataforma de inteligência artificial avançada que oferece modelos altamente eficientes para análise e processamento de informações. Ele tem uma grande vantagem de estar totalmente integrado com as ferramentas da da nuvem do Google, Google Cloud Platform.

Preço da API do Google Gemini

Atualmente (em 2024), o Google possui uma possibilidade de acesso gratuito a API do Gemini com algumas limitações de acesso. Por exemplo, para o modelo Gemini 1.5 Flash temos os seguintes limites:

1. 15 RPM (requisições por minuto)
2. 1 milhão TPM (tokens por minuto)
3. 1,500 RPD (requisições por dia)

Você pode encontrar mais detalhes na página do Google.

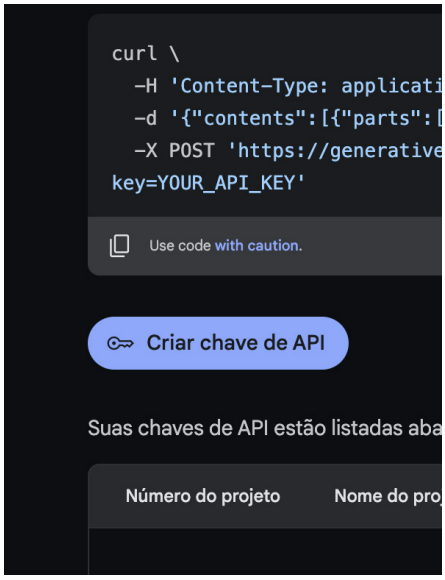
Como Obter a Chave da Google Cloud e Instanciar o Modelo Gemini

Este guia apresenta um passo a passo minimalista para obter a chave de API da Google Cloud e instanciar o modelo Gemini.

1. Acessar o AI Studio da Google

» Acesse o Google AI Studio.

» Clique em **"Criar chave de API"**



2. Crie a chave a partir de um novo projeto

» Aceite os termos de uso.

3. Copie a chave de API

Agora vamos instanciar nosso modelo

```
[ ]!pip install -q -U google-generativeai==0.8.3
```

O código abaixo importa userdata do Google Colab para manipular dados do usuário no ambiente Colab.

```
[ ]from google.colab import userdata
   import os

   GOOGLE_API_KEY = userdata.get('GOOGLE_API_KEY')
```

continua

```
os.environ["GOOGLE_API_KEY"] = userdata.get('GOOGLE_API_KEY')  # Define a variável de ambiente GOOGLE_API_KEY com a chave de API do Google, recuperada do Colab.
```

```
[ ]import google.generativeai as genai

  # Configura a chave do Gemini - Substitua pela sua chave
  genai.configure(api_key=GOOGLE_API_KEY)

  # Define o modelo do Gemini
  model = genai.GenerativeModel('gemini-1.5-flash')

  # Configuração da API da LLM
  generation_config = genai.GenerationConfig(
          max_output_tokens=1000,
          temperature=0.2,
  )

  # Pergunta do cliente ao chatbot
  consulta_usuario = "Meu computador não liga, o que posso fazer?"

  # Envio da pergunta e impressão da resposta
  resposta = model.generate_content(
      f"Responda de forma clara e objetiva a
  seguinte pergunta: {consulta_usuario}",
      generation_config = generation_config
  )
  print(resposta.text)
```

Verifique a fonte de alimentação, os cabos, o botão liga/desliga e a tomada. Se nada funcionar, procure um técnico.

Alterando o código para Langchain

Vamos mudar o código para invocar a API do Google utilizando a biblioteca Langchain. Então, ao invés de usar diretamente a biblioteca do Google, iremos usar a biblioteca langchain-google-genai que facilita a interação com a API do Google.

```
[ ]!pip install -q -U langchain==0.3.7 langchain-google-genai==2.0.6
```

```
   ━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 11.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━ 41.3/41.3 kB 1.7 MB/s eta 0:00:00
```

O código abaixo realiza a mesma operação do código utilizando a biblioteca com a classe genai.GenerativeModel. Mas, perceba que o número de linhas de código é menor:

```
[ ]from langchain_google_genai import ChatGoogleGenerativeAI

  llm = ChatGoogleGenerativeAI(model="gemini-1.5-flash")

  consulta_usuario = "Meu computador não liga, o que posso fazer?"
  result = llm.invoke(f"Responda de forma clara e objetiva
  a seguinte pergunta: {consulta_usuario}")

  print(result.content)
```

```
Verifique a tomada, o cabo de força e o interruptor do computador.  Se nada funcionar, verifique fusíveis ou disjuntores. Se ainda assim não ligar, pode haver um problema interno e necessitará de reparo profissional.
```

Nesse caso, que tal alterarmos o max_tokens?

Vamos entender como ele pode te ajudar na hora de limitar uma saída. Mesmo prompt do código anterior, mas com um limite de 10 tokens.

```
[ ]from langchain_google_genai import ChatGoogleGenerativeAI

  llm = ChatGoogleGenerativeAI(model="gemini-1.5-flash", temperature=0.7,
  max_tokens=10)

  consulta_usuario = "Meu computador não liga, o que posso fazer?"
  result = llm.invoke(f"Responda de forma clara e objetiva
  a seguinte pergunta: {consulta_usuario}")

  print(result.content)
```

Verifique a tomada, o cabo de força e

A resposta fica estranha, né?

Parece que não faz muito sentido utilizar esse parâmetro, né...? Porém, imagine que você precisa usar um LLM para ler uma pergunta de um usuário e você SÓ pode responder com **SIM** ou com **NÃO**.

```
[ ]from langchain_google_genai import ChatGoogleGenerativeAI

  llm = ChatGoogleGenerativeAI(model="gemini-1.5-flash", temperature=0.7,
  max_tokens=1)

  consulta_usuario = "Olá, bom dia. Vocês tem acesso ao meu banco de
  dados que é privado? Me responda explicando o porque você não tem"

  result = llm.invoke(f"Responda de forma clara e objetiva a seguinte
  pergunta: {consulta_usuario}")

  print(result.content)
```

Não

Exercício 1

Neste exercício, você aprenderá a utilizar uma API para acessar o Gemini utilizando Langchain e adaptar o código para diferentes finalidades. Você está aprendendo a usar a API **`langchain_google_genai`** para acessar o LLM Gemini. O código fornecido abaixo demonstra como enviar uma consulta simples sobre um problema de computador e receber uma resposta.

Objetivo:

Este exercício visa familiarizar você com o uso de APIs para interagir com LLMs, especificamente com o Gemini, e a explorar o impacto dos parâmetros temperature e max_tokens na resposta gerada.

Tarefa:

1. **Modifique a consulta do usuário:** altere a variável consulta_usuario para solicitar os passos para fazer uma receita de bolo. Seja específico no tipo de bolo que você deseja (ex: "bolo de chocolate com cobertura de morango").
2. **Ajuste a temperatura:** experimente diferentes valores para o parâmetro temperature (ex: 0.5, 1.5, 2.0). Observe como a variação da temperatura afeta a criatividade e a aleatoriedade da resposta do LLM.
3. **Ajuste o número máximo de tokens:** modifique o parâmetro max_tokens (ex: 500, 1500, 2000). Analise como o limite de tokens influencia o comprimento e a completude da resposta gerada.

```
[ ]from langchain_google_genai import ChatGoogleGenerativeAI

  llm = ChatGoogleGenerativeAI(model="gemini-1.5-flash", temperature=1.0,
  max_tokens=1000)

  consulta_usuario = "Meu computador não liga, o que posso fazer?"
  result = llm.invoke(f"Responda de forma clara e objetiva a seguinte
  pergunta: {consulta_usuario}")

  print(result.content)
```

```
Verifique a tomada, o cabo de força e o interruptor do computador.  Se o problema persistir, verifique as luzes indicadoras de energia e tente conectar outro dispositivo na tomada para testar a energia. Se nada funcionar, procure ajuda profissional.
```

OpenAI

A OpenAI oferece modelos poderosos como o GPT, que podem ser usados para diversas tarefas comerciais, como atendimento ao cliente, criação de conteúdo, automação de processos e análise de dados. Esses modelos são capazes de entender e gerar texto de forma natural, permitindo que empresas otimizem suas interações com os clientes, criando experiências mais personalizadas e eficientes.

API paga: a OpenAI cobra pelo uso da API. As políticas da API mudam constantemente e, por isso, é importante validar se a API está disponível com créditos para novos usuários. **Gostaria de reforçar que não é preciso executar este trecho do Colab com a API da OpenAI, caso não tenha interesse em investir em pagar pela API (verifique os preços no site da OpenAI)**. Entretanto, estamos deixando este código como referência para permitir o estudo de vocês com a API mais utilizada no mundo no campo de LLMs.

Como Obter a Chave da OpenAI e Instanciar Modelos LLMs

Este guia apresenta um passo a passo minimalista para obter a chave de API da OpenAI e instanciar modelos de linguagem.

1. Acessar a Conta OpenAI

- Faça login na sua conta OpenAI em https://platform.openai.com/.

2. Ir para Configurações

- No canto superior direito, clique no seu perfil e selecione "API Keys" ou vá diretamente para https://platform.openai.com/account/api-keys.

3. Gerar Nova Chave

- Clique em "+ Create new secret key" para gerar uma nova chave de API.
- **Importante**: Copie a chave gerada, pois não poderá visualizá-la novamente.

Agora vamos instanciar nosso modelo

Para isso, nós precisamos conectar no cliente da openai atavés de uma API (Interface de Programação de Aplicação), que nada mais é que uma forma de conectar serviços através da WEB.

```
[ ]!pip install openai==1.57.0
```

```
Collecting openai==1.57.0
  Downloading openai-1.57.0-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from openai==1.57.0) (3.7.1)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from openai==1.57.0) (1.9.0)
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from openai==1.57.0) (0.28.0)
Requirement already satisfied: jiter<1,>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from openai==1.57.0) (0.8.0)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from openai==1.57.0) (2.10.3)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from openai==1.57.0) (1.3.1)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages (from openai==1.57.0) (4.66.6)
Requirement already satisfied: typing-extensions<5,>=4.11 in /usr/local/lib/python3.10/dist-packages (from openai==1.57.0) (4.12.2)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai==1.57.0) (3.10)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai==1.57.0) (1.2.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->openai==1.57.0) (2024.8.30)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->openai==1.57.0) (1.0.7)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages (from httpcore==1.*->httpx<1,>=0.23.0->openai==1.57.0) (0.14.0)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai==1.57.0) (0.7.0)
Requirement already satisfied: pydantic-core==2.27.1 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai==1.57.0) (2.27.1)
```

```
Downloading openai-1.57.0-py3-none-any.whl (389 kB)
   ━━━━━━━━━━━━━━━━━━━━ 389.9/389.9 kB 9.0 MB/s eta 0:00:00
Installing collected packages: openai
  Attempting uninstall: openai
    Found existing installation: openai 1.54.5
    Uninstalling openai-1.54.5:
      Successfully uninstalled openai-1.54.5
Successfully installed openai-1.57.0
```

A biblioteca da OpenAI faz esse serviço mais rapidamente para nós, ela antecipa alguns código que nós baixamos quando instalamos ela e nos permite com poucas linhas de código fazer aplicações robustas.

Nesse trecho de código estamos conectando na OpenAI, simulando uma consulta do usuário e mandando para o modelo da OpenAI na forma de uma mensagem de um "user".

```
[ ]
from google.colab import userdata
import os

OPENAI_KEY = userdata.get('OPENAI_KEY')
os.environ["OPENAI_API_KEY"] = userdata.get('OPENAI_KEY')  # Define a variável de ambiente GOOGLE_API_KEY com a chave de API do Google, recuperada do Colab.
```

```
import openai

#Configura o cliente da OpenAI
client = openai.OpenAI(api_key = OPENAI_KEY)

# Exemplo de uso para um assistente de saúde
consulta_usuario = "Meu computador não liga, o que posso fazer?"

# Faz a construção do código que irá chamar a api
resposta = client.chat.completions.create(
```

continua

```
  model="gpt-4o-mini",
  messages=[
    {"role": "user", "content": f"Responda de forma clara e objetiva a seguinte pergunta: {consulta_usuario}"}
  ]
)

# Imprime a resposta
print("Resposta do Assistente:", resposta.choices[0].message.content)
```

Resposta do Assistente: Se o seu computador não liga, siga estas etapas para solucionar o problema:

1. **Verifique a fonte de energia**: Assegure-se de que o cabo de alimentação está conectado corretamente à tomada e ao computador.

2. **Testar a tomada**: Tente usar outra tomada ou um dispositivo diferente na mesma tomada para garantir que ela está funcionando.

3. **Verifique o botão de liga/desliga**: Às vezes, o botão pode estar com problemas. Pressione-o firmemente.

4. **Remova dispositivos externos**: Desconecte todos os dispositivos (mouse, teclado, impressora, etc.) e tente ligar o computador novamente.

5. **Observe sinais de vida**: Veja se as luzes do computador acendem ou se há barulhos (ventiladores, HD).

6. **Verifique RAM e cabos internos**: Se você se sentir confortável, abra o computador e verifique se a RAM está bem encaixada e se os cabos internos estão conectados corretamente.

7. **Teste com outra fonte de energia**: Se possível, tente usar outra fonte de energia para ver se o problema é com a fonte atual.

8. **Procure por problemas de superaquecimento**: Verifique se não há acúmulo de poeira nos ventiladores.

Se, após essas etapas, o computador ainda não ligar, pode ser necessário procurar um técnico especializado.

Nesta célula de código, estamos interagindo com um modelo de linguagem grande chamado GPT-4o-mini através da API do Google AI. Este código envia uma pergunta para o modelo GPT-4o-mini, configura alguns parâmetros que influenciam na geração da resposta e, por fim, exibe a resposta gerada pelo modelo.

Passo a passo:

1. **Inicialização da requisição:** `client.chat.completions.create(...)` inicia o processo de envio de uma requisição para a API.
2. **Definição do modelo:** `model="gpt-4o-mini"` especifica qual modelo de linguagem será utilizado para gerar a resposta.
3. **Envio da mensagem:** `messages=[{"role": "user", "content": f"Responda de forma clara e objetiva a seguinte pergunta: {consulta_usuario}"}]` define a mensagem que será enviada ao modelo. A variável `consulta_usuario` deve conter a pergunta que queremos que o modelo responda.
4. **Parâmetros de controle:**
   - » **temperature:** controla a "criatividade" da resposta. Valores mais altos resultam em respostas mais diversificadas e menos previsíveis.
   - » **max_tokens:** limita o tamanho da resposta em número de tokens (palavras ou subpalavras).
   - » **top_p:** define a probabilidade acumulada para a seleção de palavras durante a geração da resposta (nucleus sampling).

```
[ ]resposta = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "user", "content": f"Responda de forma clara
e objetiva a seguinte pergunta: {consulta_usuario}"}
    ],
    temperature=0.7,  # Controla a aleatoriedade da resposta. Valores
mais altos (até 1.0) tornam a resposta mais criativa; valores mais
baixos (próximos a 0) tornam a resposta mais focada e conservadora.
    max_tokens=150,  # Limita o número máximo de tokens
(palavras e partes de palavras) que a resposta pode ter.
Isso ajuda a controlar o comprimento da resposta.
    top_p=1.0  # Usa a técnica de amostragem chamada nucleus
sampling. Valores próximos a 1.0 significam que todas as
palavras do vocabulário estão disponíveis para a amostragem,
enquanto valores mais baixos restringem as opções.
)
```

continua

```
# Imprime a resposta
print("Resposta do Assistente:", resposta.choices[0].message.content)
```

```
Resposta do Assistente: Se o seu computador não liga, siga estes passos:

1. **Verifique a fonte de energia**:
   - Certifique-se de que o cabo de alimentação está conectado corretamente na tomada e no computador.
   - Teste outra tomada ou use um outro cabo de alimentação, se disponível.

2. **Verifique indicadores de luz**:
   - Veja se há luzes acesas no computador (como a luz do botão de energia ou no painel frontal). Se não houver, pode ser um problema de energia.

3. **Desconecte periféricos**:
   - Remova todos os dispositivos externos (impressoras, pen drives, etc.) e tente ligar o computador novamente.

4. **Tente um reinício for
```

Alterando para utilizar Langchain

Vamos mudar o código para invocar a API da OpenAI utilizando a biblioteca Langchain. Então, ao invés de usar diretamente a biblioteca da OpenAI, iremos usar a biblioteca langchain-openai que facilita a interação com a API da OpenAI.

```
[ ]!pip install -q -U langchain-openai==0.2.11
```

```
   ━━━━━━━━━━━━━━━━━━━━ 50.7/50.7 kB 2.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 16.0 MB/s eta 0:00:00
```

```
[ ]from langchain_openai import ChatOpenAI

  llm = ChatOpenAI(model="gpt-4o-mini")

  consulta_usuario = "Meu computador não liga, o que posso fazer?"
```

continua

```
result = llm.invoke(f"Responda de forma clara e objetiva
a seguinte pergunta: {consulta_usuario}")


print(result.content)
```

Se o seu computador não liga, você pode seguir estas etapas para tentar diagnosticar o problema:

1. **Verifique a fonte de energia**:

   - Certifique-se de que o cabo de alimentação está corretamente conectado à tomada e ao computador.

   - Tente usar uma tomada diferente.

2. **Inspecione o botão de ligar**:

   - Verifique se o botão de ligar está funcionando corretamente. Às vezes, pode ficar preso ou danificado.

3. **Verifique a luz indicadora**:

   - Veja se alguma luz LED acende no computador. Isso pode indicar se está recebendo energia.

4. **Remova periféricos**:

   - Desconecte todos os dispositivos externos (impressoras, USB, etc.) e tente ligar o computador novamente.

5. **Teste com outra fonte de energia**:

   - Se possível, teste o computador com uma fonte de energia diferente ou utilize um multímetro para verificar se a fonte está funcionando.

6. **Verifique o hardware interno**:

   - Se você se sentir confortável, abra o gabinete e verifique se todos os cabos internos e componentes (como memória RAM e placa gráfica) estão bem conectados.

7. **Escute os sinais sonoros**:

   - Preste atenção a qualquer sinal sonoro que o computador emita ao tentar ligar. Os bipes podem indicar problemas específicos.

8. **Considere problemas de superaquecimento**:

   - Se o computador estava desligando repentinamente antes de não ligar, pode ser um problema de superaquecimento. Verifique se as ventoinhas estão funcionando.

Se, após todas essas etapas, o computador ainda não ligar, pode ser necessário procurar a ajuda de um técnico especializado.

Exercício 2 - Explorando parâmetros de temperatura, tokens e top_p no GPT

Objetivo:

Entender como os parâmetros `temperature`, `max_tokens` e `top_p` influenciam a resposta do modelo GPT-4 através de experimentação.

Lembre que a API da OpenAI é paga e, portanto, este exercício não é obrigatório. Desenvolva o exercício caso tenha acesso à API do GPT e queira praticar com essa API.

Instruções:

Modifique o prompt na variável `prompt` para direcionar a ação do modelo para solicitar os passos para fazer uma receita de bolo. Seja específico no tipo de bolo que você deseja (ex: "bolo de chocolate com cobertura de morango").

Adicione os parâmetros abaixo:

1. **Temperatura:**

» Modifique o valor da `temperature` no código para 0.0, 0.5 e 1.0.

» Para cada valor, execute o código com o mesmo prompt (defina um prompt interessante no início do código).

» Observe como a resposta do modelo varia em termos de criatividade e aleatoriedade.

2. **Máximo de Tokens:**

» Modifique o valor de `max _ tokens` no código para 50, 150 e 300.

» Execute o código com o mesmo prompt para cada valor.

» Compare o tamanho (número de palavras) das respostas geradas.

3. **Top_p (Nucleus Sampling):**

» Modifique o valor de `top _ p` no código para 0.2, 0.5 e 1.0.

» Execute o código com o mesmo prompt para cada valor.

» Observe como a diversidade e a coerência das respostas são afetadas.

```
[ ]from langchain_openai import ChatOpenAI

  llm = ChatOpenAI(model="gpt-4o-mini")
```

continua

```
consulta_usuario = "Meu computador não liga, o que posso fazer?"
prompt = f"Responda de forma clara e objetiva a seguinte pergunta: {consulta_usuario}"

result = llm.invoke(prompt)

print(result.content)
```

Se o seu computador não liga, siga estes passos:

1. **Verifique a fonte de energia**: Certifique-se de que o cabo de alimentação está conectado corretamente à tomada e ao computador. Teste a tomada com outro aparelho.

2. **Inspecione o botão de ligar**: Pressione o botão de ligar por alguns segundos. Às vezes, pode ser necessário pressioná-lo mais de uma vez.

3. **Remova dispositivos externos**: Desconecte todos os dispositivos externos (impressoras, pen drives, etc.) e tente ligar o computador novamente.

4. **Verifique os LEDs e sons**: Observe se há luzes acesas ou se o computador faz algum som (como o ventilador). Isso pode indicar se ele está recebendo energia.

5. **Teste com outra fonte de energia**: Se possível, use um cabo de alimentação diferente ou teste em outra tomada.

6. **Verifique a bateria (em laptops)**: Se for um laptop, remova a bateria (se possível) e conecte apenas o cabo de alimentação. Tente ligar sem a bateria.

7. **Limpeza interna**: Se você se sentir confortável, abra o computador e verifique se há poeira acumulada ou conexões soltas.

8. **Consulte um técnico**: Se nada funcionar, pode ser um problema de hardware. Nesse caso, é melhor procurar um profissional.

Siga esses passos e veja se consegue resolver o problema.

Desafio Prático: AKCIT Pizzas

Imagine que você é dono de uma pizzaria, vamos dar um nome a ela...

E você percebeu que nem sempre consegue ou sanar todas as dúvidas do seus clientes ou nem mesmo consegue responder os pedidos deles.

Porém, você como um aspirante a especialista em NLP, lembrou que pode utilizar LLM para suprir essa sua carência. Nisso, você desenvolveu um assistente para sua pizzaria, que vai instruir seu cliente e mitigar suas percas.

Primeiro, criar um uma base conteúdo para servir de contexto para nosso LLM. O código abaixo apresenta um conteúdo de um FAQ de uma pizzaria que iremos utilizar como contexto para o LLM conseguir fornecer o resultado final.

```
[ ]# Criando perguntas e respostas para a pizzaria AKCIT Pizzas.
   faq_content = """1. Qual é o horário de funcionamento da AKCIT Pizzas?
   Estamos abertos todos os dias das 11h às 23h.

   2. Vocês fazem entrega?
   Sim, oferecemos entrega em toda a cidade. Você pode
   fazer o pedido pelo nosso site ou aplicativo.

   3. Qual é o tempo médio de entrega?
   O tempo médio de entrega é de 30 a 45 minutos,
   dependendo da localização.

   4. Quais são os métodos de pagamento aceitos?
   Aceitamos cartões de crédito, débito e
   pagamentos em dinheiro na entrega.

   5. Vocês têm opções vegetarianas?
   Sim, temos várias opções vegetarianas no nosso cardápio,
   incluindo pizzas de legumes e queijos especiais.

   6. Posso personalizar minha pizza?
   Claro! Você pode escolher os ingredientes e a
   massa da sua pizza ao fazer o pedido.

   7. Vocês oferecem promoções?
   Sim, temos promoções semanais e descontos
   especiais em pizzas grandes e combos.

   8. Quais são os tamanhos de pizza disponíveis?
```

continua

Oferecemos pizzas em tamanhos pequeno, médio e grande.

9. Vocês têm pizza sem glúten?

Sim, temos opções de massa sem glúten para atender a clientes com restrições alimentares.

10. Qual é a pizza mais popular da AKCIT?

A pizza "AKCIT Especial", com pepperoni, queijo e azeitonas, é uma das mais populares entre nossos clientes.

11. Vocês têm alguma opção de sobremesa?

Sim, oferecemos sobremesas como tortas, sorvetes e bolos caseiros.

12. É possível fazer um pedido antecipado?

Sim, você pode fazer um pedido antecipado pelo nosso site.

13. Vocês entregam em áreas mais distantes da cidade?

Entregamos em diversas áreas, mas pode haver uma taxa adicional para locais mais distantes.

14. Posso alterar ou cancelar meu pedido?

Sim, você pode alterar ou cancelar seu pedido até 10 minutos após a confirmação.

15. Quais são as opções de bebidas disponíveis?

Oferecemos refrigerantes, sucos, água e opções de bebidas alcoólicas.

16. Vocês têm algum programa de fidelidade?

Sim, temos um programa de fidelidade que oferece descontos para clientes frequentes.

17. Vocês aceitam pedidos por telefone?

Sim, você pode fazer seu pedido pelo telefone durante nosso horário de funcionamento.

18. Como posso acompanhar meu pedido?

continua

Após fazer o pedido, você receberá um código de rastreamento por SMS ou e-mail para acompanhar a entrega.

19. Vocês oferecem catering para eventos?

Sim, oferecemos serviços de catering para festas e eventos. Entre em contato para mais informações.

20. Qual é o prazo de entrega em horários de pico?

Durante horários de pico, o prazo de entrega pode aumentar para até 60 minutos.

21. Vocês têm promoções para aniversários?

Sim, oferecemos uma pizza grátis para aniversários, mediante solicitação.

22. Como posso fazer uma reclamação?

Você pode entrar em contato conosco pelo telefone ou e-mail para registrar sua reclamação.

23. Quais ingredientes estão disponíveis para personalização?

Temos uma variedade de ingredientes, incluindo diferentes queijos, carnes, vegetais e molhos.

24. Vocês têm opções de pizza vegana?

Sim, temos pizzas veganas feitas com ingredientes livres de produtos animais.

25. Vocês têm um menu de crianças?

Sim, oferecemos opções de pizza em tamanho menor e pratos especiais para crianças.

26. Como vocês garantem a qualidade dos ingredientes?

Trabalhamos com fornecedores locais e selecionamos ingredientes frescos para garantir a qualidade.

27. Vocês têm algum combo especial?

continua

Sim, temos combos que incluem pizza, bebida e sobremesa a preços promocionais.

28. Como posso saber sobre novas promoções?

Você pode se inscrever em nossa newsletter ou seguir nossas redes sociais para receber atualizações sobre promoções.

29. É possível fazer pedidos para retirada?

Sim, você pode fazer pedidos para retirada diretamente em nossa loja.

30. Vocês oferecem pizzas sazonais?

Sim, temos pizzas sazonais que mudam de acordo com a época do ano e ingredientes disponíveis.

31. Quais são os ingredientes da pizza "AKCIT Especial"?

A pizza "AKCIT Especial" é feita com pepperoni, queijo mozzarella, azeitonas e molho de tomate.

32. Vocês têm opções de massa recheada?

Sim, oferecemos opções de massa recheada com queijo e outros ingredientes.

33. Como posso receber descontos em pedidos?

Fique atento às nossas promoções e ofertas especiais, que são divulgadas regularmente.

34. Vocês entregam em condomínios?

Sim, entregamos em condomínios, mas pedimos que informe o nome e número do apartamento.

35. Qual é a pizza mais picante do cardápio?

A "Pizza Jalapeño" é a mais picante do nosso cardápio, perfeita para quem gosta de um sabor intenso.

36. Vocês oferecem pizza de café da manhã?

Sim, temos opções de pizza de café da manhã

continua

com ingredientes como ovos e bacon.

37. Posso fazer um pedido em grupo?

Sim, oferecemos descontos especiais para pedidos em grupo ou para eventos.

38. Vocês têm opções de pizzas sem lactose?

Sim, podemos fazer pizzas sem lactose, utilizando queijos especiais para atender a essa necessidade.

39. Vocês oferecem informações nutricionais sobre os pratos?

Sim, as informações nutricionais estão disponíveis em nosso site, junto com o cardápio.

40. Qual é a sua política de devolução?

Se o pedido não estiver conforme o solicitado, entre em contato e faremos o possível para resolver o problema.

41. Vocês têm alguma opção de pizza com frutos do mar?

Sim, oferecemos pizzas com camarões e outros frutos do mar, dependendo da disponibilidade.

42. Como faço para ser um entregador da AKCIT Pizzas?

Você pode enviar seu currículo pelo nosso site ou entrar em contato diretamente com a equipe de recursos humanos.

43. Vocês têm algum aplicativo para pedidos?

Sim, temos um aplicativo disponível para download, onde você pode fazer pedidos e acompanhar promoções.

44. Qual é o seu compromisso com a sustentabilidade?

Estamos comprometidos em utilizar embalagens recicláveis e trabalhar com fornecedores que compartilham valores sustentáveis.

45. Vocês têm algum menu especial para eventos corporativos?

Sim, oferecemos menus personalizados para eventos

continua

```
corporativos. Entre em contato para mais informações.

46. Vocês têm opções de pizzas doces?
Sim, oferecemos pizzas doces com nutella, frutas
e outros ingredientes saborosos.

47. Como posso deixar uma avaliação sobre o serviço?
Você pode deixar uma avaliação em nosso site ou nas redes
sociais, onde ficaremos felizes em receber seu feedback.

48. Vocês têm algum programa de indicações?
Sim, temos um programa de indicações onde você pode
ganhar descontos ao indicar novos clientes.

49. Quais são os ingredientes da pizza "Margherita"?
A pizza "Margherita" é feita com molho de tomate,
queijo mozzarella, manjericão fresco e azeite.

50. Vocês oferecem pizzas com ingredientes locais?
Sim, sempre que possível, utilizamos ingredientes locais
para apoiar a comunidade e garantir frescor.
"""
```

Em um chatbot, é importante manter um histórico da conversa para que o modelo possa entender o contexto e responder de forma coerente. A função da célula abaixo tem o objetivo de registrar cada interação (pergunta do usuário e resposta do modelo) em uma lista chamada `chat_history`.

Como funciona:

1. **Entrada:** recebe a `consulta _ usuario` (texto digitado pelo usuário) e a `resposta` do chatbot como strings.

2. **Armazenamento:**
   - adiciona um dicionário à lista `chat_history` representando a mensagem do usuário. Esse dicionário contém:
     - `"role": "user"` indica que a mensagem é do usuário.
     - `"parts": consulta_usuario` o texto da mensagem do usuário.
   - faz o mesmo para a resposta do modelo, mudando `"role"` para `"model"` e `"parts"` para a resposta.

```
[ ]def registrar_interacao(consulta_usuario: str, resposta: str):
      """Registra a interação entre o usuário e o modelo no formato de
  chat."""
      chat_history.append({"role": "user", "parts": consulta_usuario})
      chat_history.append({"role": "model", "parts": resposta})
```

Segundo, vamos escolher nosso modelo, que nesse caso vamos escolher o gemini da Google. O código abaixo define um chatbot simples que simula um assistente de atendimento da AKCIT Pizzas. Ele usa o modelo `gemini-1.5-flash` para gerar respostas às perguntas dos usuários.

Funcionamento:

1. **Armazenamento do histórico:** o código utiliza uma lista chamada `chat_history` para guardar o histórico da conversa, permitindo que o chatbot se lembre das interações anteriores.
2. **Geração de respostas:** a função `gerar_resposta` recebe a pergunta do usuário e a envia para o modelo de linguagem, juntamente com:
   - Instruções para o modelo se comportar como um atendente da AKCIT Pizzas.
   - Informações contextuais sobre o cardápio da pizzaria (opcional).
   - O histórico da conversa para fornecer contexto.
3. **Registro da interação:** após gerar a resposta, o código registra a pergunta do usuário e a resposta do modelo no `chat_history`.
4. **Retorno da resposta:** a função `gerar_resposta` retorna a resposta gerada pelo modelo, que pode então ser exibida ao usuário.

```
[ ]chat_history = []

  model = genai.GenerativeModel("gemini-1.5-flash")

  def gerar_resposta(consulta_usuario: str):
      """Gera uma resposta do modelo, registra a interação e atualiza o
  histórico de chat."""
       context = "\n".join([f"{msg['role']}: {msg['parts']}" for msg in
  chat_history])

      prompt = f"""
          Você é um assistente de atendimento personalizado para a AKCIT
  Pizzas.
          Aqui estão algumas perguntas frequentes no contexto do cardápio
  da AKCIT Pizzas:
          {faq_content}

          Aqui está o histórico de interações:
          {context}

          E aqui está a solicitação do nosso cliente:
          {consulta_usuario}
          """

      resposta = model.generate_content(
          prompt,
          generation_config=generation_config
      ).text

      registrar_interacao(consulta_usuario, resposta)

      return resposta
```

Abaixo realizamos várias perguntas ao chatbot para ver se o chatbot consegue interagir, armazenando o histórico.

```
[ ]consulta_usuario = "Eu gostaria de pedir uma pizza de mussarela grande."
   resposta = gerar_resposta(consulta_usuario)

   print(resposta)
```

Ok, gostaria de pedir uma pizza de mussarela grande. Para confirmar o seu pedido, preciso de algumas informações:

* **Qual o seu endereço de entrega?**

* **Qual o seu método de pagamento preferido? (Dinheiro, cartão de crédito ou débito)**

* **Você deseja adicionar alguma bebida ou sobremesa?**

Assim que me fornecer essas informações, posso finalizar seu pedido. O tempo estimado de entrega é de 30 a 45 minutos, podendo variar dependendo da sua localização. Em horários de pico, esse tempo pode chegar a 60 minutos.

```
[ ]consulta_usuario = "Adorei, quanto tempo demora?"
   resposta = gerar_resposta(consulta_usuario)

   print(resposta)
```

O tempo estimado de entrega é de 30 a 45 minutos, podendo variar dependendo da sua localização. Em horários de pico, esse tempo pode chegar a 60 minutos. Para te dar um tempo mais preciso, preciso do seu endereço de entrega. Você poderia me fornecer, por favor?

Na célula abaixo testamos se o modelo está usando o histórico da conversa para responder as perguntas dos usuários.

```
[ ]consulta_usuario = "Qual a pizza que eu pedi?"
   resposta = gerar_resposta(consulta_usuario)

   print(resposta)
```

Você pediu uma pizza de mussarela grande.

Deu para ver que nosso modelo não foi bem instruido. Isso é bom para vermos que não é mágica, tem toda uma estruturação e pensamento.

Você pode modificar a estrutura, fique a vontade para testar e ou adicionar outras técnicas como RAG.

Desafio : Modifique o escopo da AKCIT Pizzas.

Que tal tentar praticar o que aprendemos nesse colab? Mude o escopo da AKCIT Pizzas, implemente uma nova base de conhecimento, modifique os prompts e seja criativo na hora de criar novos serviços comerciais usando LLMs.

Aqui estão alguns exemplos reais de como LLMs podem ser usados em serviços comerciais:

1. **Geração de conteúdo para marketing:** LLMs podem gerar descrições de produtos, campanhas publicitárias e postagens para redes sociais de maneira eficiente, mantendo a consistência e o tom de voz da marca. A AKCIT Pizzas pode usar esse serviço para criar campanhas promocionais personalizadas para diferentes públicos.
2. **Análise de sentimento de clientes:** você pode analisar opiniões de clientes sobre a pizza e o serviço.
3. **Recomendações personalizadas:** você pode oferecer recomendações personalizadas de sabores e combinações.
4. **Gerenciamento de inventário inteligente:** prever a demanda de ingredientes com base nas tendências de compra, ajudando a gerenciar o estoque para reduzir desperdícios.

Essas são apenas algumas ideias. Podemos expandir a base de conhecimento da AKCIT Pizzas para incluir novos serviços que atendam ainda mais às necessidades dos clientes.

Referências Bibliográficas

Beginner's Guide to OpenAI API

Large Language Model (LLM) API: Full Guide 2024

Comprehensive Guide to Integrating Tools and APIs with Language Models

SAIBA MAIS...

Explore os *sites* oficiais das principais plataformas de LLMs para saber sobre os detalhes dos modelos e preços de API:

- OpenAI®: https://platform.openai.com/docs/models
- Google Gemini®: https://ai.google.dev/gemini-api/docs/models/gemini
- Anthropic Claude®: https://docs.anthropic.com/en/docs/about-claude/models

PARA RELEMBRAR...

Nessa Unidade, aprendemos a utilizar APIs comerciais para acessar LLMs.

- Os **LLMs** são modelos de IA que processam linguagem natural, realizando tarefas como compreensão de texto.
- **APIs** permitem integrar LLMs em aplicações, fornecendo interfaces simples para programadores.
- A **tokenização** quebra textos em unidades menores para processamento.
- A **janela de contexto** limita a quantidade de informação que pode ser processada em uma única interação.
- O **custo** das APIs deve ser considerado, balanceando com as necessidades de sua aplicação.
- Certifique-se de que o provedor da API possui **políticas de segurança** adequadas para proteger seus dados.

AKCIT
CENTRO DE COMPETÊNCIA EMBRAPII EM TECNOLOGIAS IMERSIVAS

Grandes Modelos de Linguagens com APIs

Unidade III

Plataformas *low-code* para LLMs

Unidade III - Plataformas *Low-code* para LLMs

3.1 Visão Geral de Plataformas *Low-code* para LLMs

A crescente complexidade dos LLMs apresenta desafios para desenvolvedores que buscam integrá-los em aplicativos. A necessidade de domínio de múltiplas linguagens de programação e *frameworks*, além da gestão de infraestrutura, pode tornar o processo caro e demorado.

Plataformas *low-code* (ferramentas que facilitam a criação de aplicações com a necessidade de escrita de pouco ou nenhum código) surgem como uma solução para mitigar essas dificuldades, oferecendo um ambiente de desenvolvimento simplificado, baseado em componentes visuais e interfaces intuitivas. A Gartner®, uma das mais respeitadas empresas de consultoria do mundo, prevê que, até 2026, desenvolvedores fora dos departamentos formais de tecnologia da informação (TI) representarão pelo menos 80% da base de usuários de ferramentas de desenvolvimento *low-code*, em comparação com 60% em 2021 (Gartner, 2022).

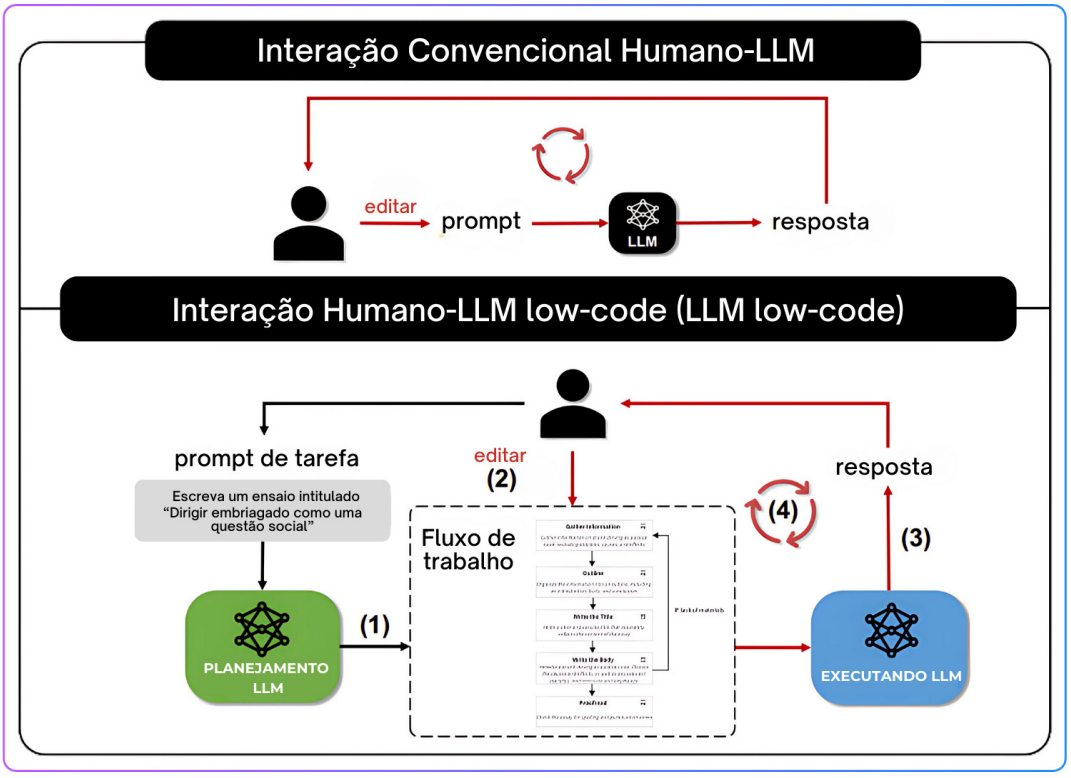
Uma das principais vantagens das plataformas *low-code* é a velocidade com que se pode desenvolver protótipos e produtos finais. Em vez de lidar diretamente com códigos complexos, os desenvolvedores utilizam componentes visuais, conectam APIs e configuram fluxos de trabalho de maneira intuitiva. Essa abordagem torna possível integrar LLMs a diferentes tipos de aplicativos sem a necessidade de conhecimentos profundos de IA ou de linguagens de programação específicas.

Na Figura 6, é apresentada a diferença entre a interação convencional e com soluções *low-code* entre humanos e as LLMs (Cai *et al*., 2023). Na abordagem convencional, o usuário envia o *prompt* com as instruções para o modelo e recebe a resposta. Quando existe a necessidade de construir soluções mais avançadas, fica a cargo do desenvolvedor criar códigos para *pipelines* (processos) mais complexos.

A interação com as LLMs utilizando soluções *low-code* podem ser realizadas por meio dos seguintes passos:

1. O usuário envia um *prompt* para um módulo de planejamento que gera um fluxo de trabalho altamente estruturado para tarefas complexas.
2. Os usuários podem editar o fluxo de trabalho, usando operações *low-code* predefinidas, que são todas realizadas por meio de operações de cliques, arrastar e soltar ou edição de texto.
3. Um módulo de execução das instruções do LLM gera respostas com base no fluxo de trabalho revisado.
4. Os usuários continuam refinando o fluxo de trabalho até que resultados satisfatórios sejam obtidos.

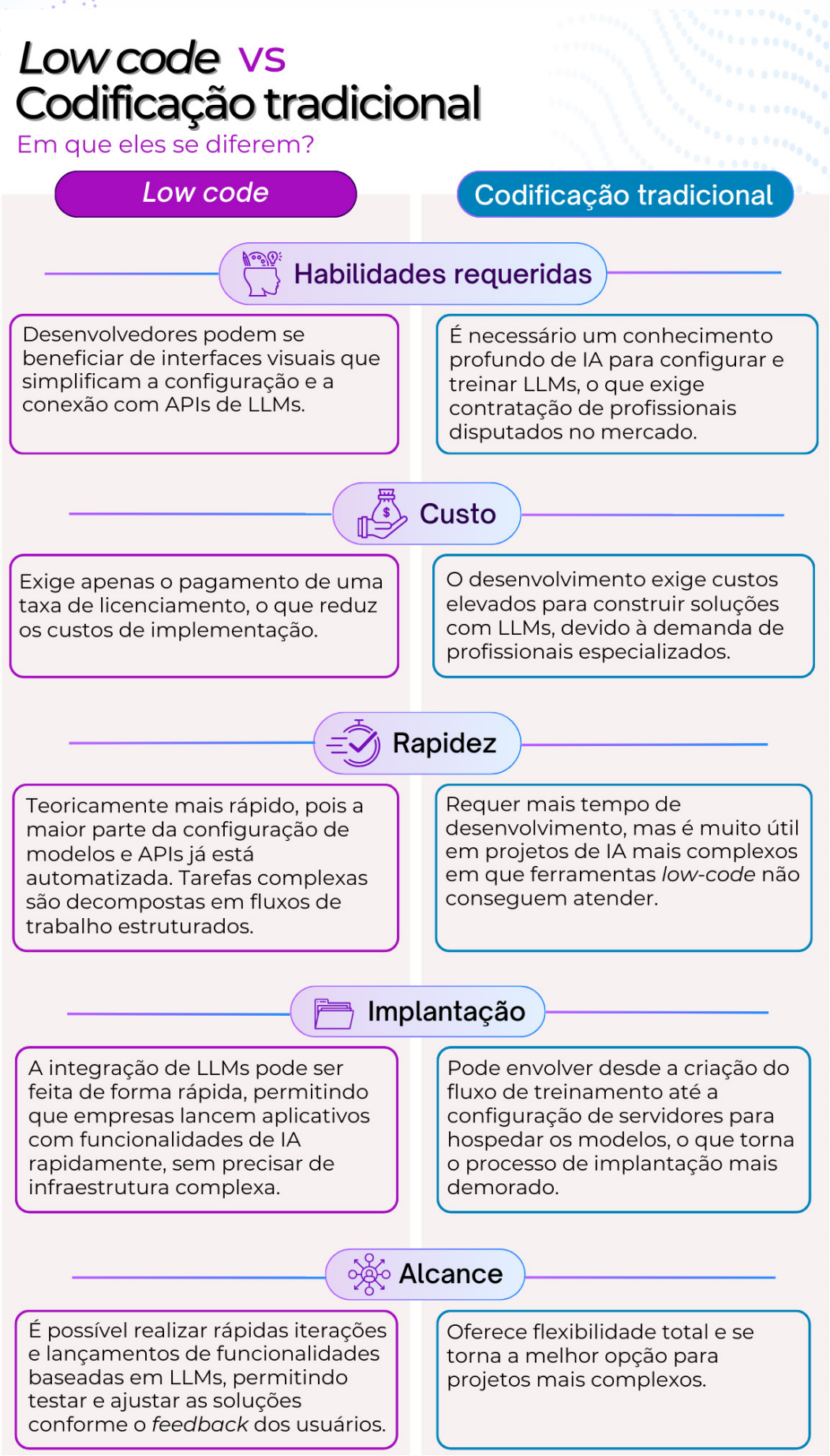
Figura 6 - Visão geral da interação humana com *Large Language Models* por meio de *low-code* (*Low-code* LLM) e sua comparação com a interação convencional



Fonte: Cai *et al*. (2023).

As plataformas *low-code* e a codificação tradicional oferecem abordagens distintas para o desenvolvimento de aplicativos, cada uma com suas próprias vantagens e desafios. No infográfico (Figura 7), a seguir, as principais diferenças entre a codificação tradicional e *low-code* são apresentadas, com base em cinco características: habilidades requeridas, custo, rapidez, implantação e alcance.

Figura 7 - *Low code versus* codificação tradicional semelhante



Nota: APIs: *Applications Processing Interface*. LLM: *Large Language Models*. IA: inteligência artificial.

Fonte: autoria própria.

O desenvolvimento *low-code* não substitui o desenvolvimento tradicional, mas é uma ferramenta que as empresas podem usar para aumentar suas capacidades de codificação e aliviar os departamentos de TI de tarefas repetitivas ou desnecessárias. O desenvolvimento *low-code* permite que os desenvolvedores trabalhem mais rápido e dediquem suas habilidades técnicas e de *full-stack* a projetos mais inovadores e personalizados.

Com o desenvolvimento *low-code*, os desenvolvedores podem iniciar projetos e adicionar funcionalidades básicas a páginas existentes ou criar novas páginas dentro do contexto de projetos maiores e mais complexos, evitando a necessidade de escrever código linha por linha. O desenvolvimento *low-code* pode expandir os recursos e ferramentas disponíveis para a equipe de TI, além de permitir que tarefas mais simples sejam repassadas para funcionários que não são da área de desenvolvimento.

3.2 Soluções Low-code para Construção de Aplicativos com Inteligência Artificial

No contexto da integração de LLMs, as plataformas *low-code* proporcionam um meio eficiente de conectar modelos de linguagem a outros recursos do aplicativo, tais como bases de dados, APIs externas e interfaces de usuário. Diversas plataformas oferecem integrações prontas com APIs de serviços de LLMs populares como o Google Cloud Vertex AI Platform®, o Azure OpenAI Services® e o Amazon SageMaker®. Essa integração facilita a comunicação entre os diferentes componentes dentro de um sistema construído na nuvem.

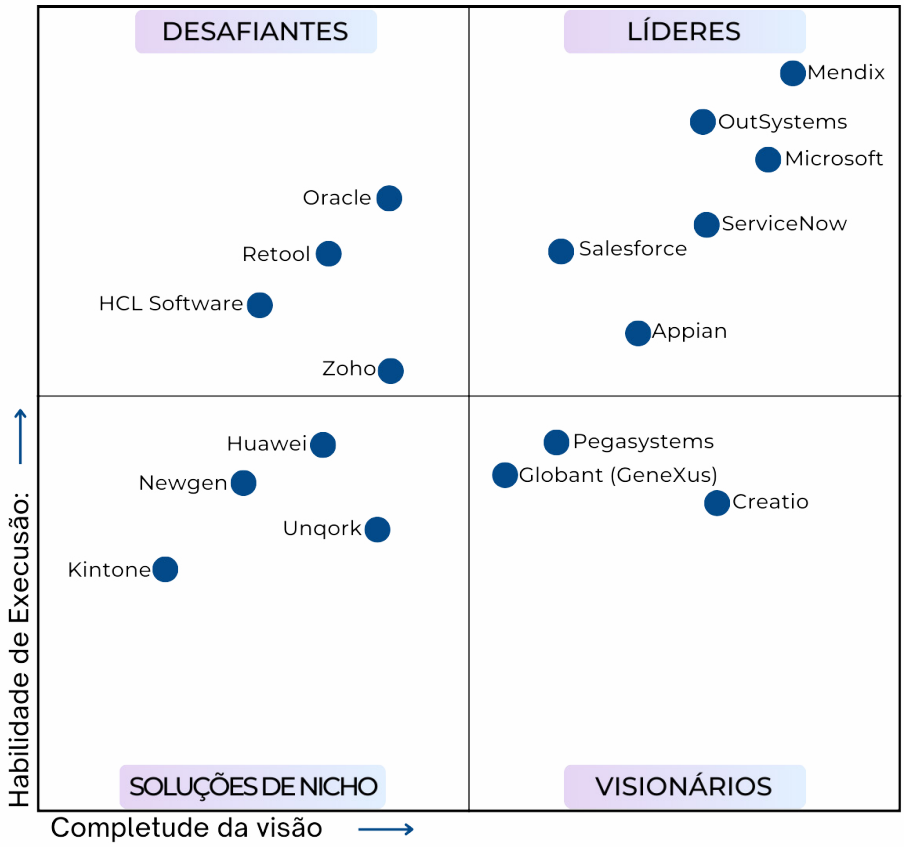
A empresa de consultoria Gartner® define plataformas de aplicativos de baixo código (*low-code application platforms* [LCAPs) como plataformas utilizadas para desenvolver e executar rapidamente aplicativos personalizados, abstraindo e minimizando o uso de linguagens de programação (Gartner, 2024). Segundo o relatório do Gartner, as LCAPs oferecem suporte para soluções escaláveis, com alto desempenho e disponibilidade, além de recuperação de desastres, segurança, acesso à API para (e de) serviços em nuvem corporativos e de terceiros, monitoramento de uso, acordos de nível de serviço e disponibilidade de suporte técnico e treinamento.

Na Figura 8, é apresentado o quadrante mágico da Gartner sobre plataformas *low-code* para construção de aplicações comerciais. Um quadrante mágico fornece o posicionamento gráfico competitivo de quatro tipos de provedores de tecnologia em mercados, nos quais o crescimento é alto e a diferenciação do provedor é distinta. Líderes executam bem em relação à sua visão atual e estão bem posicionados para o futuro. Visionários entendem para onde o mercado está indo ou têm uma visão para mudar as regras do mercado, mas não executam bem. As soluções de nicho concen-

tram-se com sucesso em um pequeno segmento ou não se concentram e não superam nem inovam mais do que os outros. Os desafiantes executam bem atualmente ou podem dominar um segmento grande, mas não demonstram uma compreensão da direção do mercado.

No relatório da Gartner®, é mostrado que as plataformas líderes no quadrante mágico estão aumentando consideravelmente o investimento em soluções para construção de aplicações com LLMs e IA generativa. A Gartner® destaca que uma das principais vantagens dessas plataformas *low-code* é a possibilidade de utilizar IA generativa de forma mais acessível, criando aplicativos customizados sem a necessidade de se aprofundar em codificação complexa.

Figura 8 - Quadrante mágico da Gartner para plataformas para aplicações *low-code*



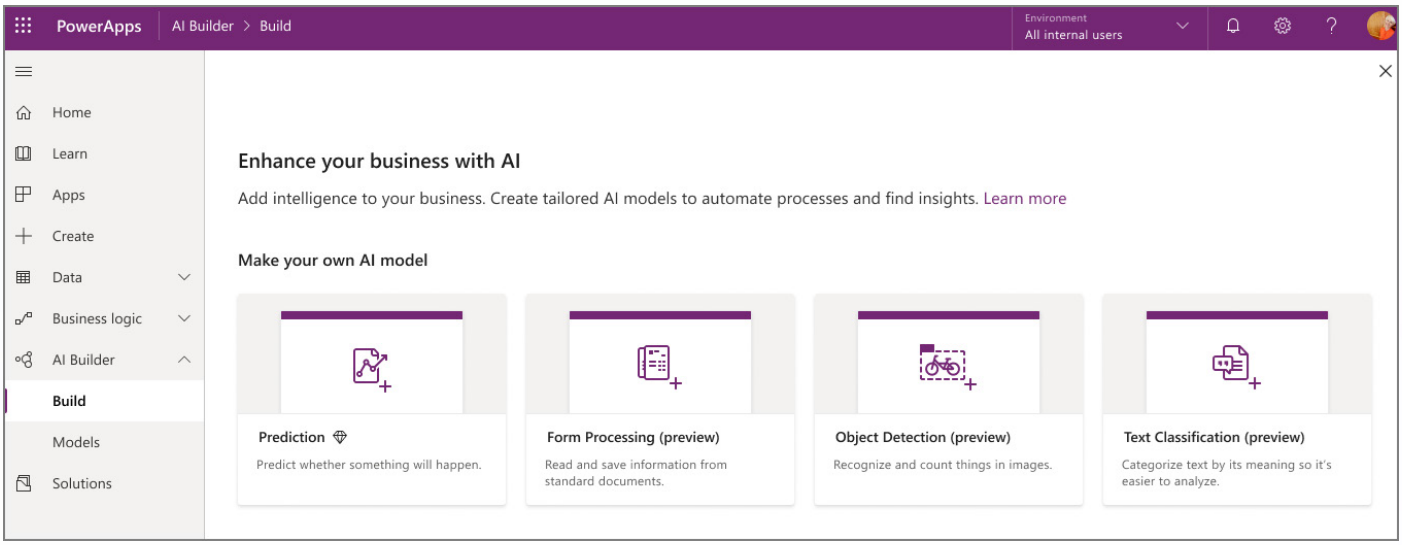
Fonte: Gartner (2024).

Um exemplo de sucesso é a solução da Microsoft®, denominada Power Apps®, que oferece um ambiente de desenvolvimento de aplicativos para criação de aplicativos personalizados, integrados com as soluções de IA da Microsoft®. O Power Apps® fornece uma plataforma que permite aos desenvolvedores profissionais interagir de modo programático com metadados e dados, aplicar lógica de negócios, criar conectores personalizados e integrar dados externos. Com ele, é possível inte-

grar sua aplicação com LLMs utilizando a ferramenta AI Builder®[1], apresentado na Figura 9. Com AI Builder®, você pode criar e usar modelos de IA que otimizam seus processos. Você pode usar um modelo predefinido que está pronto para uso ou criar um modelo personalizado que atende às suas necessidades.

O Power Apps® permite integrar com o Microsoft Pilot® para habilitar a construção de relatórios dentro dos aplicativos utilizando linguagem natural nas consultas. Outra vantagem do Power Apps® é a capacidade de integração com diversas ferramentas da Microsoft® e da Azure®, tais como o Power BI®. O Power BI® é uma ferramenta da Microsoft® de **análise de dados e criação de visualizações interativas**, que permite transformar dados em *insights* para facilitar a **tomada de decisões**.

Figura 9 - Ferramenta AI Builder® integrada ao Power Apps®



Fonte: Microsoft (2024).

3.3 Plataformas *Low-code* para Integração de LLMs

Existe outra categoria de plataformas *low-code* que facilita a construção de agentes baseados em LLMs, sem necessariamente se preocuparem na implantação do aplicativo final ao usuário. Essas plataformas oferecem diferentes níveis de flexibilidade e acessibilidade, tornando mais fácil para desenvolvedores (e até mesmo profissionais sem conhecimento técnico avançado) criar e gerenciar fluxos de trabalho com agentes de IA. Ao mesmo tempo, as soluções *low-code* oferecem uma abordagem amigável ao usuário, permitindo que essas tecnologias sejam aplicadas de maneira mais rápida e sem a necessidade de programação profunda.

1 "O AI Builder® é uma funcionalidade do Microsoft Power Platform® que pode ser usada para levar o poder da IA da Microsoft® para a organização, sem a necessidade de codificação ou de habilidades em ciência de dados" Fonte: https://learn.microsoft.com/pt-br/ai-builder/.

Na Tabela 2, são descritas as principais capacidades e aspectos relacionados às ferramentas de construção de soluções com LLMs. Nessa Tabela, as ferramentas Dify®, Langflow®, Flowise®, Google Vertex AI® e Microsoft Copilot Studio® são comparadas. Todas elas possuem interface visual de construção de fluxos de agentes e *Retrieval-Augmented Generation* (RAG). Essas ferramentas são avaliadas, segundo as seguintes características:

- **Abordagem de programação**: refere-se à forma como a ferramenta permite a interação com o usuário para construção de agentes. As abordagens podem incluir:
  - **Ferramenta visual**: oferece uma interface gráfica que permite aos usuários criar fluxos e interações de agentes arrastando e soltando elementos, sem a necessidade de programar diretamente.
  - **API**: A ferramenta também pode ser integrada a outros sistemas ou ser configurada por meio de APIs, oferecendo mais flexibilidade para programadores.
- **LLMs suportadas**: tipos de LLMs que a ferramenta pode integrar e utilizar:
  - **Principais LLMs de código aberto e comerciais**: a ferramenta oferece suporte tanto para modelos abertos e de código livre, quanto para modelos comerciais.
  - **Apenas modelos proprietários**: a ferramenta é limitada a utilizar apenas modelos desenvolvidos e disponibilizados por seus próprios fornecedores.
- **Preço:** indica se o projeto é de código aberto ou se está disponível apenas em planos comerciais. Em projetos com código aberto, algumas ferramentas como LangFlow® e Flowise® possuem também a opção de planos comerciais.
- **Suporte a RAG**: aponta se a plataforma tem suporte a construção de fluxos de RAG de forma visual.
- **Suporte a agentes**: capacidade da plataforma de criar agentes de IA que podem realizar tarefas automatizadas, responder a perguntas ou interagir com usuários de forma dinâmica.
- ***Single Sign-On* (SSO) e controle de acesso**: SSO permite que usuários acessem vários sistemas ou plataformas usando uma única credencial de *login* (por exemplo com contas do Google®), facilitando a gestão de identidades. Controle de acesso refere-se a ferramentas que permitem limitar quem pode visualizar, editar ou executar diferentes partes dos sistemas, garantindo a segurança e a privacidade dos dados.
- **Permite executar localmente**: indica se a ferramenta pode ser executada no ambiente local, em vez de depender de uma plataforma na nuvem.

» **Suporte à nuvem**: capacidade de a ferramenta ser hospedada e executada na nuvem, facilitando o acesso e a manutenção da plataforma.

Tabela 2 - Comparação das principais soluções *low-code* para construção de agentes

Funcionalidade	Dify	LangFlow	Flowise	Google Vertex AI	Microsoft Copilot Studio
Abordagem de Programação	Ferramenta visual e API	Ferramenta visual e API	Ferramenta visual	Ferramenta visual	Ferramenta visual
LLMs suportadas	Principais LLMs de código aberto e comerciais	Principais LLMs de código aberto e comerciais	Principais LLMs de código aberto e comerciais	Apenas modelos proprietários	Apenas modelos proprietários
Preço	Código aberto. Opção de planos comerciais	Código aberto	Código aberto. Opção de planos comerciais	Somente planos comerciais	Somente planos comerciais
Suporte a RAG	✓	✓	✓	✓	✓
Suporte a Agentes	✓	✓	✓	✓	✓
SSO e controle de acesso	✓	✗	✗	✓	✓
Permite executar localmente	✓	✓	✓	✗	✗
Suporte à nuvem	✓	✗	✓	✓	✓

Nota: APIs: *Applications Processing Interface*. LLM: *Large Language Models*. RAG: Retrieval-Augmented Generation. SSO: *Single Sign-On*. Fonte: autoria própria.

As plataformas como **Dify®**, **LangFlow®** e **Flowise®** oferecem soluções com código aberto, o que garante maior flexibilidade para os desenvolvedores que desejam personalizar seus agentes e executar localmente, além de contar com uma variedade de LLMs suportados, tanto com código aberto quanto comerciais. Ao contrário, soluções como o **Google Vertex AI®** e o **Microsoft Copilot Studio®** se restringem a modelos proprietários e planos exclusivamente comerciais, o que pode ser uma limitação para quem busca maior controle sobre a implementação dos agentes e não possuem recursos financeiros suficientes para iniciar a construção de agentes com soluções *low-code*.

As soluções com planos comerciais têm um bom suporte para funcionalidades como SSO e controle de acesso, o que garante que apenas usuários autorizados tenham acesso a determinados recursos, algo importante para grandes empresas que precisam gerenciar múltiplos usuários. No Dify®, por exemplo, autenticação via SSO só está disponível nos planos comerciais.

Além das ferramentas apresentadas na Tabela 2, existem diversas outras soluções *low-code* que habilitam a construção de soluções com LLMs. A plataforma Galadon® possui apenas planos comerciais e permite construir agentes de *chatbots* de forma visual (Galadon, 2024). Entre as ferramentas de código aberto, a LLMStack® permite construir fluxos de agentes de IA, possuindo opções na nuvem (LLMStack, 2024). Já a plataforma RAGFlow® é especializada na construção de *pipelines* de RAG de forma visual, permitindo que os agentes acessem bases de dados e documentos relevantes em tempo real para fornecer respostas melhores aos usuários. O Ragflow® facilita a integração com diversos serviços de dados, contendo um módulo otimizado para leitura de documentos externos (Ragflow, 2024).

O n8n® é uma plataforma de automação de *workflows* que permite a criação de soluções com integração de LLMs, como os oferecidos pela OpenAI® (N8N, 2024). Com seus nós pré-configurados para APIs e serviços, é possível construir agentes LLMs ao integrar fluxos que utilizam IA para tarefas como geração de texto, análise de dados ou automação de interações com usuários. Por exemplo, você pode configurar um *workflow* que receba *inputs* via formulário, processe esses dados com um modelo de IA para análise semântica ou elaboração de respostas, e envie os resultados automaticamente para *e-mails* ou sistemas de Gestão de Relacionamento com o Cliente (*Customer Relationship Management* [CRM]). A flexibilidade do n8n® facilita a orquestração de processos complexos, permitindo que os agentes LLMs sejam personalizados para diferentes necessidades operacionais, como atendimento ao cliente, suporte técnico e criação de conteúdos.

Uma ferramenta que se destaca também no cenário mundial é a **startup brasileira** CrewAI® (Crew AI, 2024). A CrewAI® é uma ferramenta que integra agentes de IA para executar tarefas autônomas e se destaca pela facilidade de configuração e execução de fluxos de trabalho complexos. Quando integrada em plataformas de *low-code*, ela permite que usuários, mesmo com pouca experiência em programação, criem, ajustem e implementem agentes de IA em seus *pipelines* de dados de forma visual e acessível, otimizando processos sem a necessidade de codificação detalhada.

3.4 Considerações Finais

Ao longo dessa Unidade, exploramos várias plataformas *low-code* que permitem a integração de LLMs em aplicações, cada uma com suas características, vantagens e desvantagens. As plataformas *low-code* para construção de agentes com IA, como Dify®, LangFlow® e Flowise®, oferecem soluções de código aberto com uma variedade de LLMs suportados, permitindo maior flexibilidade para personalização e execução local. Em contraste, Google Vertex AI® e Microsoft Copilot Studio® utilizam modelos proprietários e se restringem a planos comerciais, o que pode ser uma limitação para quem busca maior controle dos modelos.

As plataformas proprietárias geralmente oferecem suporte para funcionalidades como SSO e controle de acesso, essenciais para grandes empresas com múltiplos usuários. É importante ponderar esses fatores ao selecionar uma plataforma, considerando necessidades específicas de projeto, orçamento e requisitos de segurança.

Além das plataformas mencionadas, existem diversas outras alternativas no mercado *low-code* para integração de LLMs, como Galadon® para *chatbots*, LLMStack® para fluxos de agentes em nuvem, e RAGFlow® para pipelines de RAG. A escolha da melhor plataforma dependerá do caso de uso específico, recursos disponíveis e nível de personalização desejado.

As plataformas *low-code* democratizam o acesso a LLMs, permitindo que desenvolvedores com diferentes níveis de experiência criem aplicações inovadoras com IA. É importante pesquisar as opções disponíveis, experimentar e selecionar a que melhor se adapte às suas necessidades para aproveitar ao máximo o potencial dos LLMs em seus projetos. A sugestão é sempre estar em busca de novas ferramentas e rever a evolução das ferramentas já existentes para escolher aquela que melhor se alinhe aos seus objetivos e recursos disponíveis.

Saiba mais...

Consulte as seguintes plataformas *low-code*:

- Dify® e LangFlow® para soluções de código aberto para integração de LLMs.

- Google Vertex AI® e Microsoft Copilot Studio® para plataformas proprietárias.

- Explore outras soluções como Galadon® para *chatbots* visuais e RAGFlow® para facilitar a construção de *pipelines* de RAG.

Para relembrar...

Nessa Unidade, exploramos as plataformas *low-code* para LLMs. Vimos como essas plataformas podem simplificar o desenvolvimento de aplicações com LLMs, abstraindo a complexidade de código e infraestrutura. Discutimos as vantagens e desvantagens do uso de plataformas *low-code*, comparamos diferentes opções disponíveis e apresentamos exemplos de uso. Lembre-se dos principais pontos abordados:

- **Simplicidade**: as plataformas *low-code* oferecem interfaces visuais e intuitivas, permitindo que desenvolvedores sem conhecimento profundo de IA criem aplicações com LLMs.

- **Velocidade**: o desenvolvimento *low-code* acelera o processo de criação de protótipos e produtos finais, reduzindo o tempo de desenvolvimento.

- **Custo**: as plataformas *low-code* geralmente cobram uma taxa de licenciamento, mas eliminam a necessidade de investir em infraestrutura e contratar cientistas de dados.

- **Flexibilidade**: as plataformas *low-code* permitem que você se conecte a diferentes APIs de LLMs, escolha o modelo que melhor se adapte às suas necessidades e personalize o fluxo de trabalho.

- **Escalabilidade**: verifique se a plataforma *low-code* escolhida oferece suporte à escalabilidade necessária para o seu projeto.

AKCIT
CENTRO DE COMPETÊNCIA EMBRAPII EM TECNOLOGIAS IMERSIVAS

Grandes Modelos de Linguagens com APIs

Unidade IV

Casos de uso de aplicações de LLMs

Unidade IV - Casos de Uso de Aplicações de LLMs

4.1 Instruções Sobre os Casos de Uso

Nesta seção, serão apresentados três estudos de caso práticos que demonstram a integração de LLMs com plataformas *low-code*. Em cada estudo de caso, um problema específico é descrito, a solução utilizando LLMs é apresentada e você será orientado por meio do processo de desenvolvimento, usando uma plataforma *low-code* de sua escolha. O objetivo é proporcionar uma experiência prática, permitindo que você explore as capacidades dos LLMs e aprenda como incorporá-los em aplicações do mundo real. Você pode escolher a plataforma *low-code* que mais se identificou, estando ela descrita ou não neste livro.

Lembre-se que a plataforma *low-code* serve como uma ferramenta para simplificar o desenvolvimento, minimizando a necessidade de codificação manual. Ao longo de cada caso de uso, será necessário explorar diferentes recursos da plataforma escolhida, como integração de dados, personalização de fluxos de trabalho e utilização de LLMs para automação de respostas, classificação de textos e personalização de recomendações.

Siga os requisitos de cada estudo de caso cuidadosamente e lembre-se de que a experimentação e o erro são parte do processo de aprendizado. À medida que avançar, você começará a perceber como pequenas mudanças nos fluxos podem alterar o comportamento da aplicação, um aprendizado importante no desenvolvimento com LLMs.

4.2 Estudo de Caso 1: Sumarização de Documentos Jurídicos em PDF

Imagine que uma editora de conteúdo jurídico precise de uma solução para processar grandes volumes de documentos PDF, como contratos, leis e pareceres, de forma rápida. O objetivo deste estudo de caso é criar uma aplicação que utilize uma LLM para realizar a sumarização automática desses documentos, extraindo os pontos principais e gerando um resumo conciso para que advogados possam revisá-los de maneira mais ágil.

Ao utilizar uma plataforma *low-code*, o desafio aqui será desenvolver uma aplicação que possa receber documentos PDF como entrada, processar o texto contido neles e utilizar uma LLM para gerar uma versão resumida que mantenha as informações mais relevantes. A interface deve ser simples, permitindo que os usuários façam o *upload* de arquivos PDF e recebam o resumo em tempo real.

A LLM deverá ser configurada para realizar a sumarização levando em consideração o tipo de documento. Por exemplo, para contratos, o foco deve estar em cláusulas importantes, datas e partes envolvidas; para leis, os principais artigos e disposições legais devem ser destacados. Você pode escolher um tipo de classes de documentos, tal como leis trabalhistas, e desenvolver usando um contexto menor.

Neste estudo de caso, você irá aprender a integrar uma LLM com um sistema que precisa lidar com a complexidade de documentos longos e variados, ao mesmo tempo que proporciona uma interface simples para o usuário final. O foco da atividade está em reduzir o tempo necessário para análise de grandes documentos jurídicos.

Implementação com plataforma *low-code*:

- **Interface do usuário**: utilize os componentes da plataforma *low-code* para criar uma interface de *upload* de PDF para o usuário final.
- **Processamento do PDF**: integre um componente ou biblioteca para extrair o texto do PDF carregado. Algumas plataformas *low-code* já possuem essa funcionalidade embutida.
- **Integração com LLM**: conecte a saída do extrator de texto à entrada do seu LLM escolhido. Configure o LLM para gerar um resumo. A plataforma *low-code* deve facilitar essa conexão por meio de conectores visuais ou APIs.
- **Exibição do resumo**: apresente o resumo gerado pelo LLM na interface do usuário. Considere adicionar opções para ajustar o tamanho do resumo ou destacar trechos importantes.

4.3 Estudo de Caso 2: Sistema de Suporte Jurídico Automatizado

Neste caso, você é solicitado a desenvolver uma aplicação para um escritório de advocacia que deseja oferecer um sistema de suporte automatizado para consultas jurídicas preliminares. A ideia é criar um assistente capaz de ajudar clientes a entenderem questões jurídicas básicas, como leis trabalhistas, direitos do consumidor ou contratos de aluguel, fornecendo orientações iniciais, sem a necessidade de uma consulta imediata com um advogado.

Ao usar a plataforma *low-code*, você deverá integrar uma LLM que seja capaz de interpretar as perguntas dos usuários em linguagem natural e fornecer respostas detalhadas, baseadas em legislação e jurisprudência. Um aspecto importante desse desafio é garantir que a LLM compreenda o contexto jurídico e possa diferenciar entre consultas de caráter geral e situações que exijam suporte especializado.

Implementação com plataforma *low-code*:

- **Interface de *chat***: utilize os componentes de interface da sua plataforma para criar uma interface de *chat*.
- **Integração com LLM**: conecte o LLM à interface de *chat*. A plataforma deve permitir a configuração de fluxos de conversa, onde as mensagens do usuário são enviadas para o LLM e as respostas são exibidas no *chat*.
- **Transferência para agente humano**: configure a lógica para transferir a conversa para um agente humano quando o *chatbot* não conseguir responder à pergunta do usuário ou quando a situação exigir intervenção humana.
- **Geração da resposta**: apresente a resposta da pergunta ao usuário na tela.

4.4 Estudo de Caso 3: Classificador de Sentimento de Avaliações do Cliente

Neste caso de uso, você desenvolverá uma aplicação para classificar o sentimento das avaliações de clientes deixadas em um *site* de compras online. A empresa deseja saber se as avaliações são positivas, negativas ou neutras, de modo a monitorar a satisfação dos clientes em tempo real e tomar decisões estratégicas com base nesses dados.

Utilizando a plataforma *low-code*, sua tarefa será criar uma aplicação que receba essas avaliações e as classifique automaticamente de acordo com o sentimento expresso pelos clientes. A LLM utilizada deverá ser capaz de interpretar a linguagem natural dos textos, que podem variar em tom e estilo, e atribuir uma classificação apropriada.

Aqui, o desafio será otimizar a precisão da LLM ao lidar com textos muitas vezes subjetivos, garantindo que a classificação de sentimentos reflita corretamente as emoções dos clientes.

Implementação com plataforma *low-code*:

- **Entrada de dados**: crie um mecanismo para coletar avaliações de clientes. Isso pode envolver a integração com plataformas de avaliação online, bancos de dados ou planilhas. Uma opção é criar uma interface para entrada da avaliação do cliente e saída da avaliação.

» **Integração com LLM**: conecte a saída do pré-processamento à entrada do LLM. Configure o LLM para realizar a classificação de sentimento.

» **Visualização de resultados**: exiba os resultados da classificação de sentimento em um formato visual na aplicação. Você decide como apresentar o resultado final da análise de sentimento ao usuário.

SAIBA MAIS...

Para se aprofundar nos casos de uso de LLMs apresentados nesta Unidade, você pode explorar os seguintes recursos:

- **Artigos e tutoriais**: pesquise por "casos de uso de LLMs", "aplicações de LLMs" ou "LLMs na prática" para encontrar exemplos e tutoriais específicos para cada área de aplicação. Alguns exemplos podem ser encontrados nas documentações das plataformas *low-code*:
  - Dify®: https://docs.dify.ai/learn-more/use-cases
  - Flowise®: https://flowiseai.com/#usecases
  - Microsoft®: https://www.microsoft.com/en-us/microsoft-copilot/blog/copilot-studio/transform-your-organization-with-custom-copilots/

PARA RELEMBRAR...

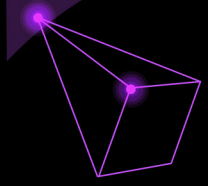
Nessa Unidade, apresentamos três estudos de caso que demonstram a integração de LLMs com plataformas *low-code*. Em cada caso de uso, um problema específico é apresentado, propõe-se uma solução utilizando LLMs e você é orientado por meio do processo de desenvolvimento. Lembre-se dos principais pontos abordados:

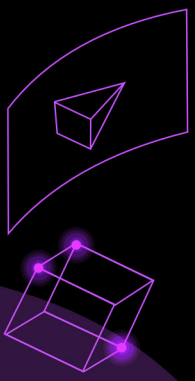
- **Sumarização de documentos**: como utilizar LLMs para resumir automaticamente grandes volumes de texto, extraindo as informações mais relevantes;
- **Sistema de suporte jurídico**: como criar um assistente virtual capaz de responder a perguntas jurídicas e fornecer orientações iniciais; e
- **Classificador de sentimento**: como utilizar LLMs para analisar o sentimento de textos e classificá-los como positivos, negativos ou neutros.

AKCIT
CENTRO DE COMPETÊNCIA EMBRAPII EM TECNOLOGIAS IMERSIVAS

Grandes Modelos de Linguagens com APIs

Unidade V
Encerramento





Unidade V - Encerramento

Ao longo das Unidades anteriores, exploramos os conceitos e a aplicação prática dos LLMs, focando no uso de APIs e plataformas *low-code*. Vimos como APIs comerciais permitem acessar LLMs de maneira prática, e como as plataformas *low-code* democratizam o desenvolvimento de soluções baseadas nesses modelos sem a necessidade de conhecimentos avançados de programação. Além disso, discutimos exemplos de casos de uso que ilustram a integração de LLMs em diversas aplicações do mundo real, desde suporte ao cliente automatizado até análise de sentimentos em redes sociais.

5.1 Sugestões para Ação Futura e Leitura Adicional

Para aqueles que desejam aprofundar o conhecimento sobre LLMs e continuar sua jornada, algumas sugestões são:

1. **Experimentação contínua:** utilize as plataformas mencionadas no curso, como Google Gemini®, OpenAI®, e as ferramentas *low-code* como Dify® ou LangFlow®, para criar seus próprios projetos práticos.
2. **Acompanhamento de tendências:** fique atento às inovações no campo da IA, como o desenvolvimento de novos modelos de LLMs e as melhorias em plataformas *low-code*.
3. **Leitura complementar:**
    - Explore a documentação oficial das plataformas utilizadas, como o LangFlow® ou o Google Cloud Vertex AI®, para entender melhor as funcionalidades e capacidades (Langflow, 2024; Google Cloud Vertex AI, 2024).
    - Estude a respeito dos conceitos de RAG, uma técnica cada vez mais importante para melhorar as capacidades dos LLMs, especialmente em contextos de busca e recuperação de informações.
    - Estudo sobre desenvolver e implantar soluções LLM, ponderando as implicações éticas. Alguns artigos apresentam boas sugestões de leitura sobre ética com LLMs para a área da saúde (Harrer, 2023; Ong, 2024).

5.2 Reflexões Finais sobre o Conteúdo

As ferramentas e tecnologias que envolvem LLMs estão em rápida evolução, e o futuro aponta para uma maior integração dessas soluções em diversas áreas de atuação. Seja no desenvolvimento de aplicativos comerciais, na educação ou na análise de grandes volumes de dados, o papel dos LLMs e das plataformas *low-code* continuará crescendo, facilitando a vida de desenvolvedores e empresas que buscam soluções com agilidade.

A escassez de profissionais especializados em NLP tem impulsionado a demanda por soluções que utilizam APIs ou plataformas *low-code*. Essas ferramentas permitem acelerar o desenvolvimento de projetos dentro das empresas, aumentando a eficiência e a produtividade das equipes de desenvolvimento ao facilitar a implementação de tecnologias avançadas, sem a necessidade de expertise técnica profunda.

Ao longo do Curso, percebemos que a simplicidade das interfaces *low-code* e a potência dos modelos de linguagem transformam a maneira como interagimos com a IA, tornando acessível o que antes era restrito a especialistas em IA.

Espero que esse livro tenha fornecido uma base em LLMs e suas aplicações. Acreditamos que o conhecimento que você adquiriu ao longo deste livro será uma base para impulsionar seu sucesso, seja desenvolvendo soluções ou aprimorando processos em sua área de atuação. A tecnologia está nas suas mãos!

Referências

AMAZON WEB SERVICES. Amazon Bedrock. **AWS**, 2024.Disponível em: https://aws.amazon.com/pt/bedrock/. Acesso em: 28 out. 2024.

ANTHROPIC. Claude 3.5: Sonnet. **Anthropic**, 2024. Disponível em: https://www.anthropic.com/news/claude-3-5-sonnet. Acesso em: 24 out. 2024.

CAI, Yuzhe et al. Low-code LLM: Graphical User Interface over Large Language Models. **arXiv preprint arXiv:2304.08103**, 2023.

CHIANG, Wei-Lin et al. Chatbot arena: An open platform for evaluating llms by human preference. **arXiv preprint arXiv:2403.04132**, 2024.

CREWAI. CrewAI – The Leading Multi-Agent Platform. **Crew AI,** 2024. Disponível em: https://www.crewai.com/. Acesso em: 28 out. 2024.

FIREWORKS AI. Fireworks AI. **Fireworks AI**, 2024. Disponível em: http://fireworks.ai. Acesso em: 28 out. 2024.

GALADON. Conversion Focused AI Live Chat - Outperform Your Sales Team. **Galadon**, 2024. Disponível em: https://www.galadon.com/. Acesso em: 28 out. 2024.

GARTNER. Gartner forecasts worldwide low-code development technologies market to grow 20 percent in 2023. **Gartner**, 2022. Disponível em: https://www.gartner.com/en/newsroom/press-releases/2022-12-13-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-20-percent-in-2023. Acesso em: 28 out. 2024.

GARTNER. Magic Quadrant for Cloud AI Developer Services. **Gartner**, 2024. Disponível em: https://www.gartner.com/en/documents/4843031. Acesso em: 28 out. 2024.

GOOGLE AI FOR DEVELOPERS. Tutorial: primeiros passos com a API Gemini. **Google AI for Developers**, 2024a. Disponível em: https://ai.google.dev/tutorials/python_quickstart. Acesso em: 27 out. 2024.

GOOGLE AI FOR DEVELOPERS. Sobre modelos generativos. **Google AI for Developers**, 2024b. Disponível em: https://ai.google.dev/docs/concepts. Acesso em: 27 out. 2024.

GOOGLE AI FOR DEVELOPERS. Visão geral da API Gemini. **Google AI for Developers**, 2024c. Disponível em: https://ai.google.dev/docs/gemini_api_overview. Acesso em: 27 out. 2024.

GOOGLE AI STUDIO. Obtenha sua chave de API. **Google AI Studio**, 2024. Disponível em: https://aistudio.google.com/app/apikey?hl=pt-br. Acesso em: 28 out. 2024.

GOOGLE CLOUD. Use open models with Vertex AI. **Google Cloud**, 2024. Disponível em: https://cloud.google.com/vertex-ai/generative-ai/docs/open-models. Acesso em: 27 out. 2024.

GOOGLE CLOUD. Vertex AI. **Google Cloud Vertex AI,** 2024. Disponível em: https://cloud.google.com/vertex-ai. Acesso em: 28 out. 2024.

GROQ. Groq is Fast AI Inference. **Groq**, 2024. Disponível em: https://groq.com/. Acesso em: 28 out. 2024.

HARRER, Stefan. Attention is not all you need: the complicated case of ethically using large language models in healthcare and medicine. **EBioMedicine**, v. 90, 2023.

HUGGING FACE. The AI community building the future. **Hugging Face**, 2024. Disponível em: https://huggingface.co/. Acesso em: 28 out. 2024.

LANGCHAIN. **LangChain**. 2024. Disponível em: https://www.langchain.com/. Acesso em: 27 nov. 2024.

LANGFLOW. Langflow – Crie sua aplicação de IA. **Langflow**, 2024. Disponível em: https://www.langflow.org/pt/. Acesso em: 28 out. 2024.

LINZ, T.; ALVES, L.. GitLab Duo Chat, your at-the-ready AI assistant, is now generally available. **GitLab INC**, 2024. Disponível em: https://about.gitlab.com/blog/2024/04/18/gitlab-duo-chat-now-generally-available/. Acesso em: 24 out. 2024.

LLAMAINDEX. **LlamaIndex**. 2024. Disponível em: https://www.llamaindex.ai/. Acesso em: 27 nov. 2024.

LLMSTACK. LLMStack – Plataforma de código aberto para construir agentes de IA, fluxos de trabalho e aplicações com seus dados. **LLMStack**, 2024. Disponível em: https://llmstack.ai/. Acesso em: 28 out. 2024.

MICROSOFT. Utilizar o AI Builder no Power Apps. Disponível em: https://learn.microsoft.com/pt-pt/power-apps/use-ai-builder. **Microsoft**, 2024. Acesso em: 28 out. 2024.

N8N. Secure, AI-native workflow automation. **N8N**, 2024. Disponível em: https://n8n.io/. Acesso em: 28 out. 2024.

ONG, Jasmine Chiat Ling et al. Ethical and regulatory challenges of large language models in medicine. **The Lancet Digital Health**, v. 6, n. 6, p. e428-e432, 2024.

OPENAI. API Keys. **OpenAI**, 2024. Disponível em: https://platform.openai.com/docs/guides/production-best-practices/api-keys. Acesso em: 27 out. 2024.

OPENAI. Duolingo. **OpenAI**, 2024. Disponível em: https://openai.com/index/duolingo/. Acesso em: 28 out. 2024.

RAGFLOW. RAGFlow – Build Generative AI into Your Business. **Ragflow**, 2024. Disponível em: https://ragflow.io/. Acesso em: 28 out. 2024.

SHANAHAN, M.. Talking about large language models. **Communications of the ACM**, v. 67, n. 2, p. 68-79, 2024.

TOGETHER AI. Together AI – The AI Acceleration Cloud. **Together AI**, 2024. Disponível em: http://together.ai. Acesso em: 27 out. 2024.

UDHANE, S.. Source code branching strategy and DevSecOps Pipeline at different stages of CI/CD. **Medium**, 2021. Disponível em: https://medium.com/codex/source-code-branching-strategy-and-devsecops-pipeline-at-different-stages-of-ci-cd-6e14f2cdf9c9. Acesso em: 17 jan. 2025.

VOLKSWAGEN OF AMERICA, INC. Volkswagen integra IA ao aplicativo móvel myVW com o Google Cloud. **Volkswagen of America,** 2014. Disponível em: https://media.vw.com/en-us/releases/1817. Acesso em: 24 out. 2024.

ZHAO, W. X. *et al*. A survey of large language models. **arXiv preprint arXiv:2303.18223**, 2023.

AKCIT

CENTRO DE COMPETÊNCIA EMBRAPII
EM TECNOLOGIAS IMERSIVAS

AKCIT CENTRO DE COMPETÊNCIA EMBRAPII EM TECNOLOGIAS IMERSIVAS

CEIA CENTRO DE EXCELÊNCIA EM INTELIGÊNCIA ARTIFICIAL

UFG UNIVERSIDADE FEDERAL DE GOIÁS

EMBRAPII

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

SEBRAE

GOVERNO DE GOIÁS O ESTADO QUE DÁ CERTO

FAPEG Fundação de Amparo à Pesquisa do Estado de Goiás

SOBRE O E-BOOK

Tipografia: Montserrat

Publicação: Cegraf UFG
Câmpus Samambaia, Goiânia - Goiás. Brasil. CEP 74690-900
Fone: (62) 3521-1358
https://cegraf.ufg.br