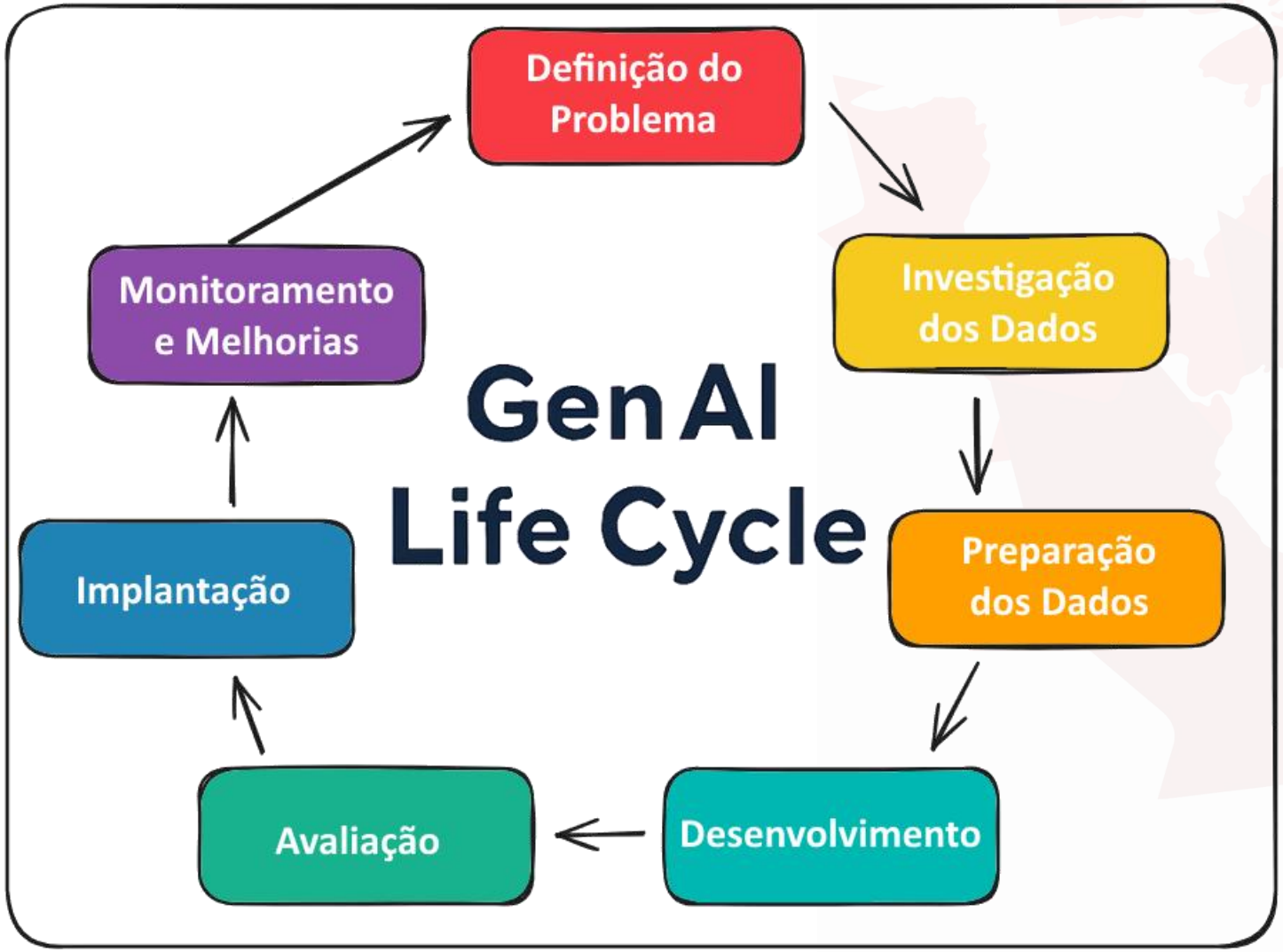
I²A²

institut d'intelligence artificielle appliquée

i2a2.academy

RAG (Retrieval-Augmented Generation)

Ciclo de vida da IA Generativa



Ciclo de vida da IA Generativa

Definição do Problema

- Gerar definição precisa do problema de negócio
- Compreender o contexto (regras de negócio)
- Estabelecer objetivos claros
- Avaliar potencial impacto
- **Definir critérios de sucesso**

Investigação dos Dados

- Compreensão e obtenção dos dados → insumos para o contexto
- Usar RAG?
- Governança de dados

Ciclo de vida da IA Generativa

Preparação dos Dados

- Limpar, formatar e estruturar os dados
- OCR?
- Incorporar a banco vetorial?
- Tratar imagens?

Desenvolvimento

- Selecionar a LLM adequada
- Treinar um novo modelo
- Reduzir o Modelo
- Fine-Tuning
- RAG
- Engenharia de prompts
- Cloud ou Edge

Ciclo de vida da IA Generativa

Avaliação

- Testes rigorosos → correção, legibilidade, desempenho e confiabilidade.
- Avaliação para a escolha do melhor modelo de LLM

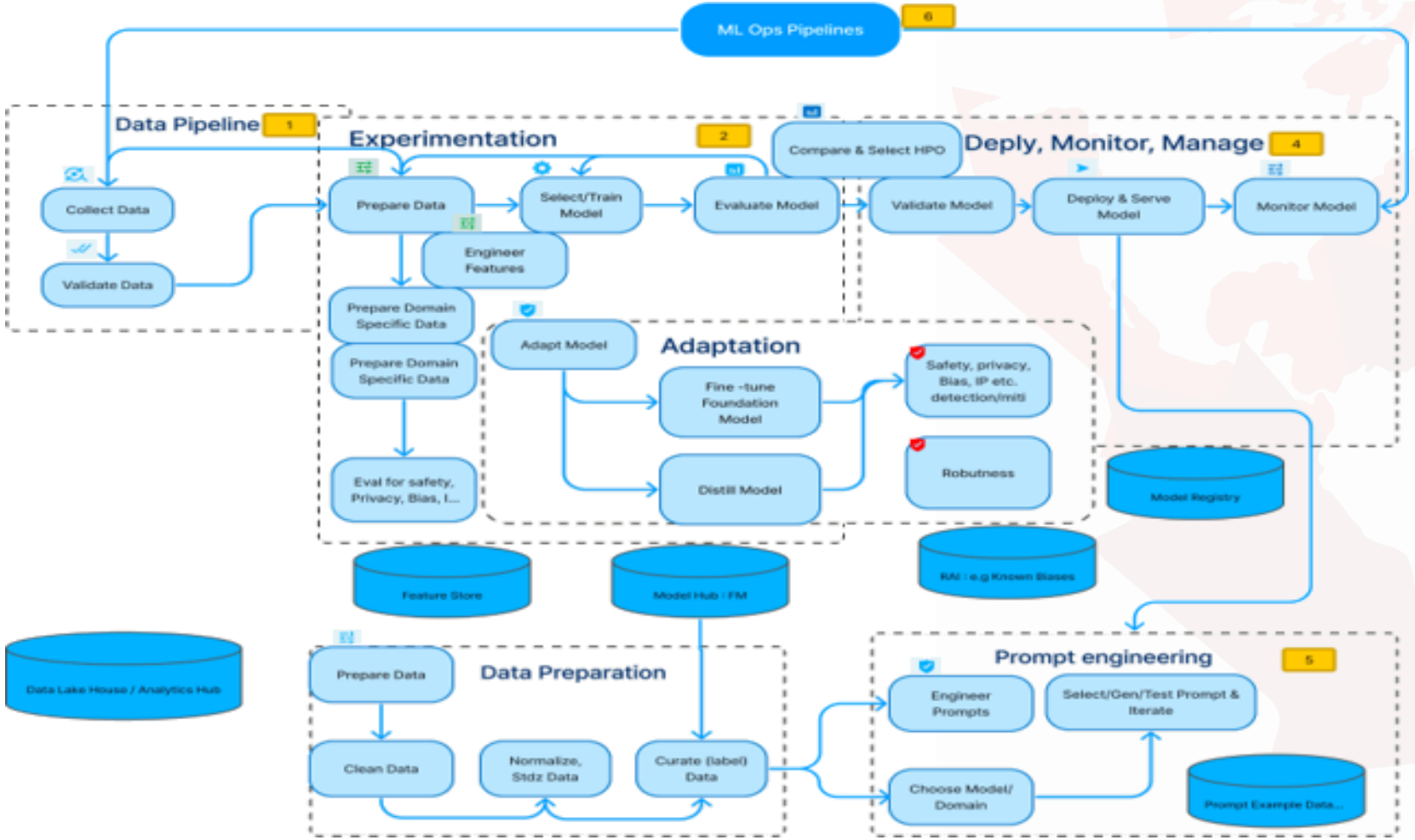
Implantação

- Configurar a infraestrutura
- Definir serviços
- Estrutura para coleta de métricas e feedback

Monitoramento e Melhorias

- Coletar métricas e feedback
- Avaliar necessidades de melhoria
- Identificar mudanças no ambiente

Ciclo de vida da IA Generativa



Fonte: https://www.linkedin.com/pulse/life-cycle-generative-ai-dr-rvs-praveen-ph-d-zqfpc/

Seleção do Modelo

Seleção do Modelo

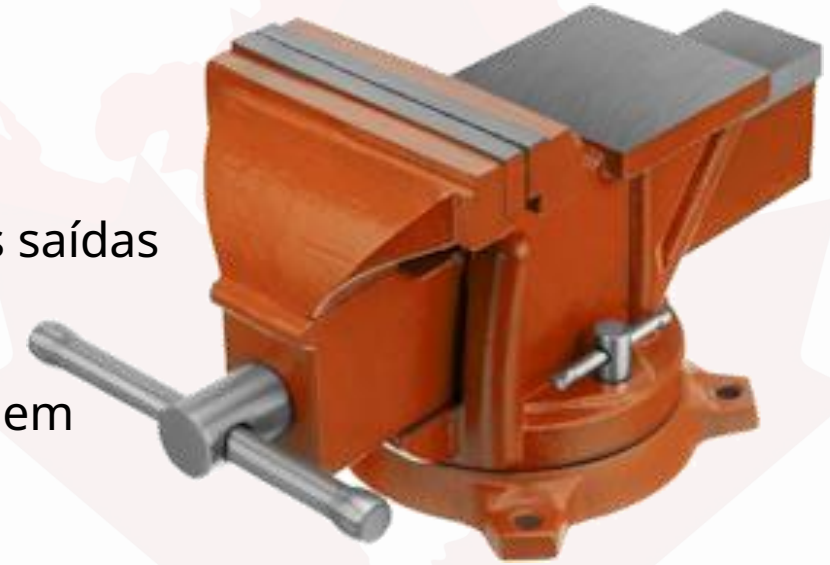
- Modelos pré treinados (Open AI GPT, Gemini, Llama, Mistral, etc)
- Modelos personalizados

Critérios para seleção:

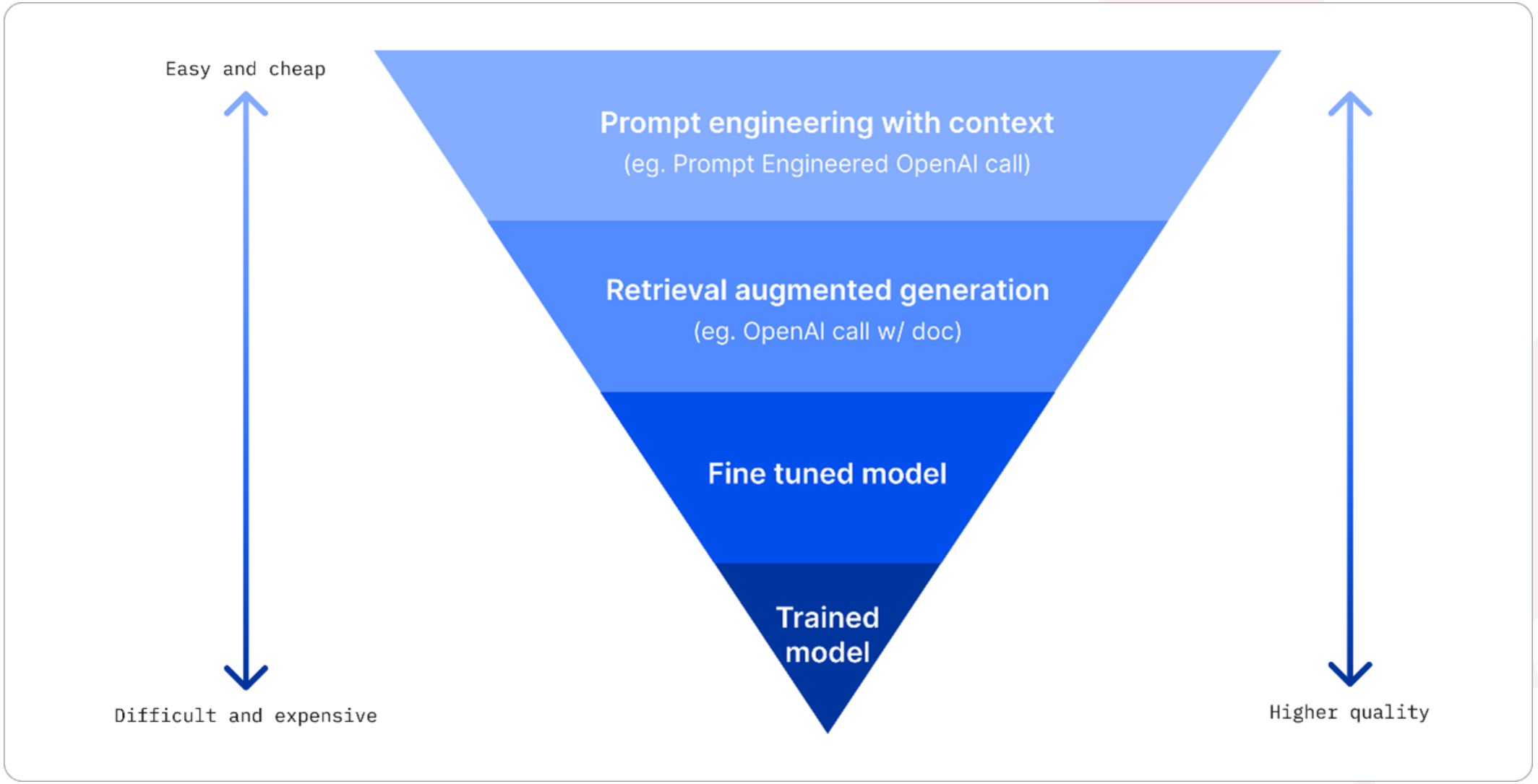
- Modalidade – somente texto, multimodal
- Tamanho – número de parâmetros
- Custo
- Precisão de desempenho
- Janela de Contexto
- Latência de inferência
- Recursos suportados
- Ética e viés

Redução do modelo

- Distillation: Cria modelo significativamente menor, mas com desempenho e capacidades semelhantes (“aluno” e “professor”)
- Pruning: Elimina pesos ou neurônios pouco importantes preservando a precisão
- Quantization: Reduzir a precisão dos pesos e ativações do LLM.
- Knowledge Distillation: Treinar um modelo menor para imitar as saídas da LLM
- Low-rank Factorization: Decompor as matrizes de pesos do LLM em matrizes de baixa ordem.
- Compact Embeddings: Diminui a dimensionalidade dos embeddings de entrada e saída do LLM
- Parameter Sharing: Utilização da mesma matriz de pesos em múltiplas camadas de uma rede neural.

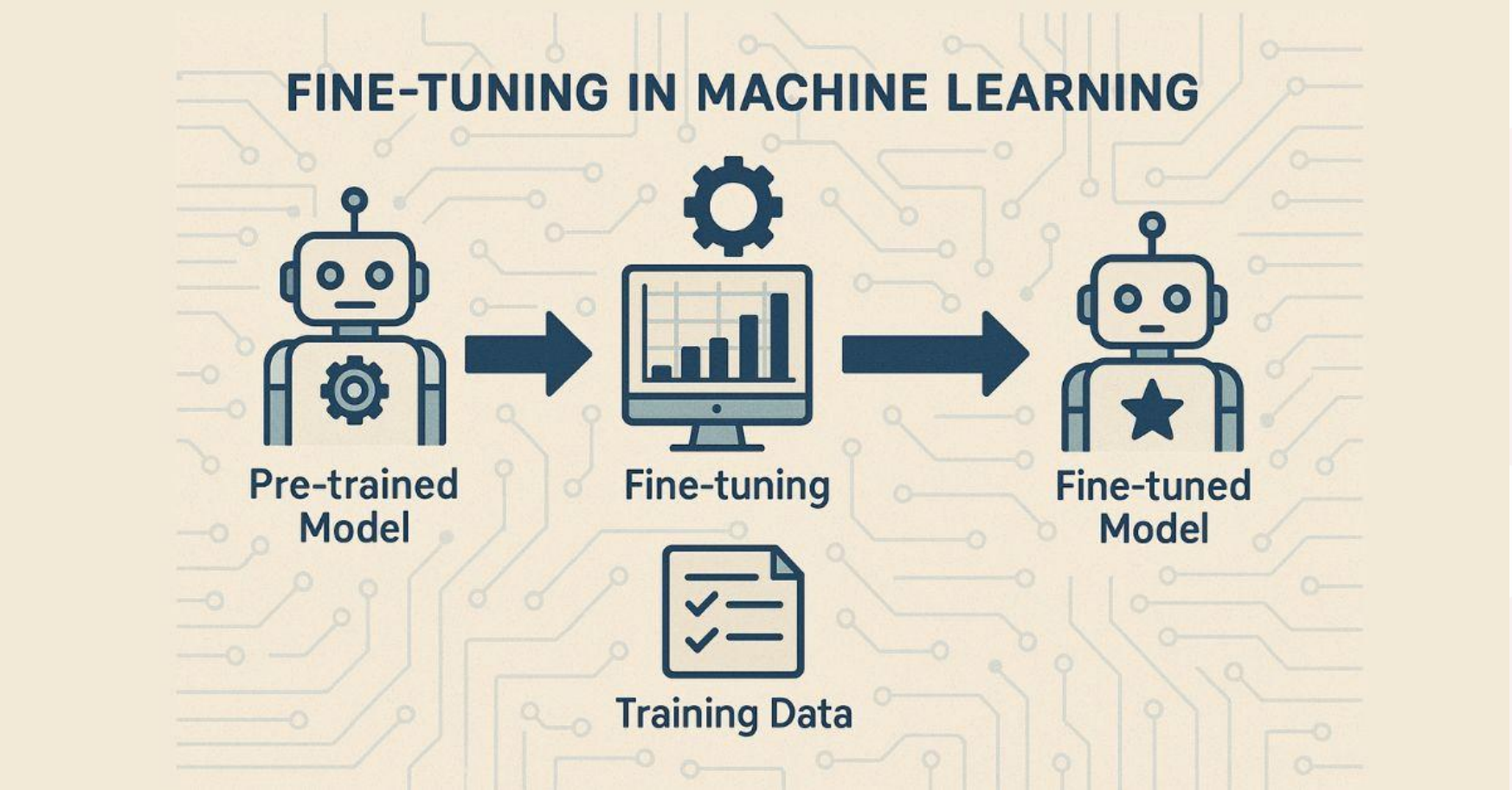


Aprimorar o modelo



Fonte: https://www.fiddler.ai/blog/four-ways-that-enterprises-deploy-llms?WT.mc_id=academic-105485-koreyst

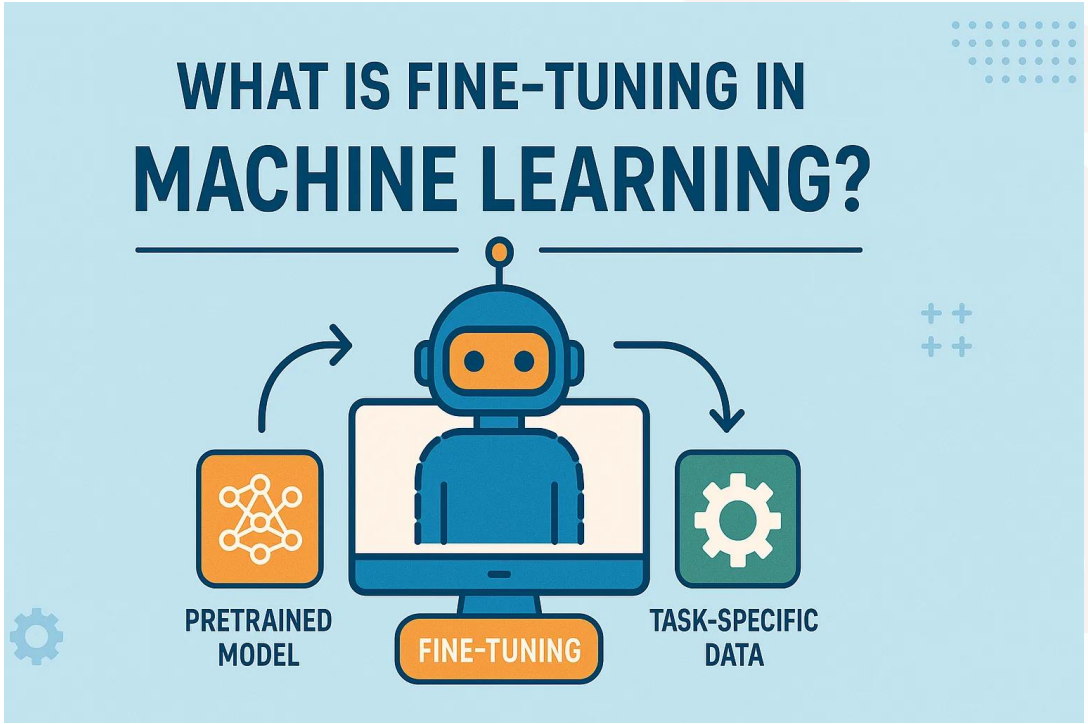
Fine-tuning



Fonte: https://www.mygreatlearning.com/blog/what-is-fine-tuning/

Fine-tuning

É o processo de **transformar** um **modelo pré treinado** em um grande e genérico dataset, adaptando-o para executar uma **tarefa nova** e mais **específica**.



Fonte: https://www.mygreatlearning.com/blog/what-is-fine-tuning/

Fine-tuning

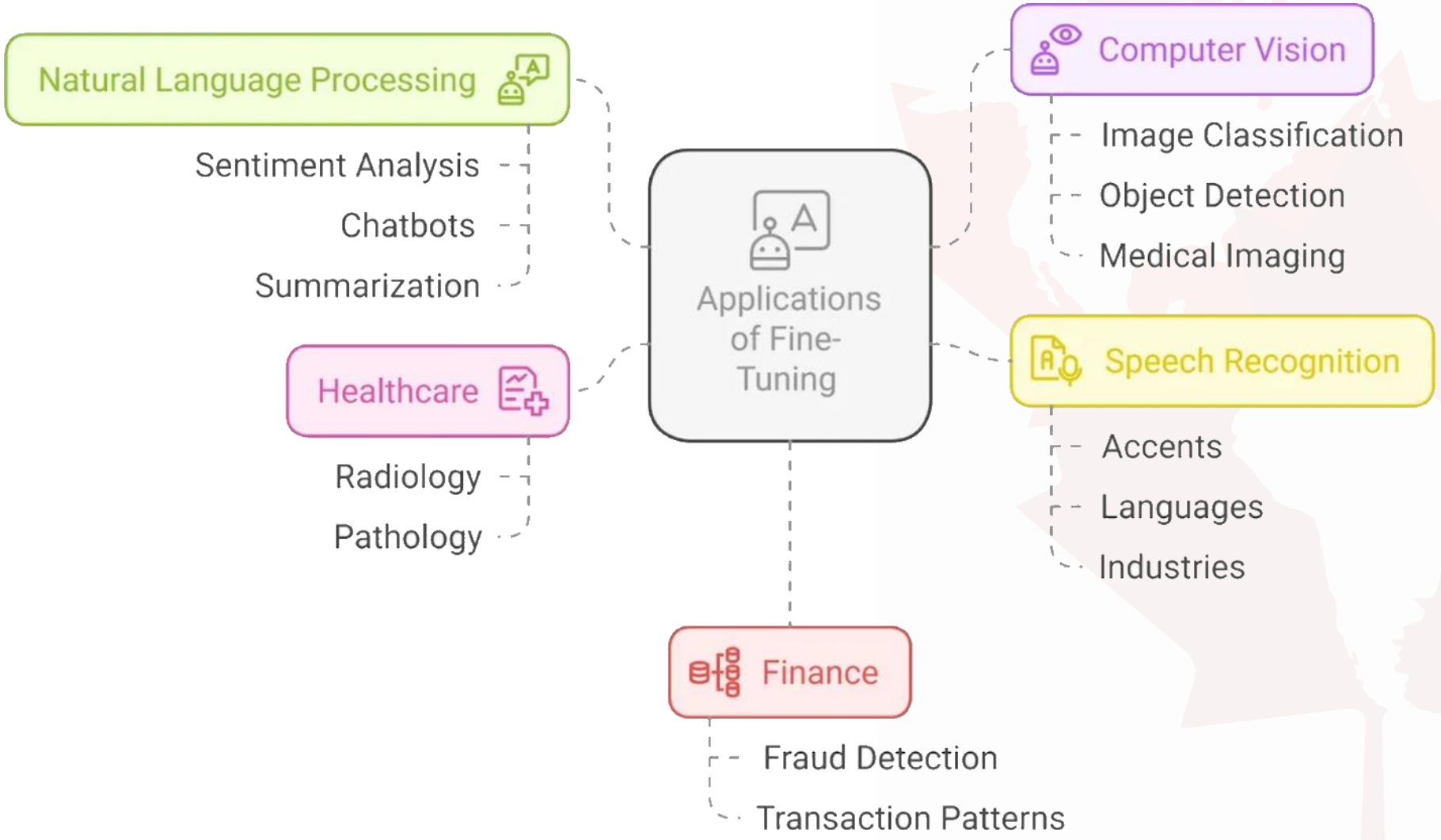
Passos para realizar o ajuste fino:

1. Selecionar um modelo pré treinado
2. Prepara um novo dataset
3. “Congelar” os pesos das camadas base
4. Adicionar ou modificar as camadas de saída
5. Treinar o modelo
6. Avaliar os resultados

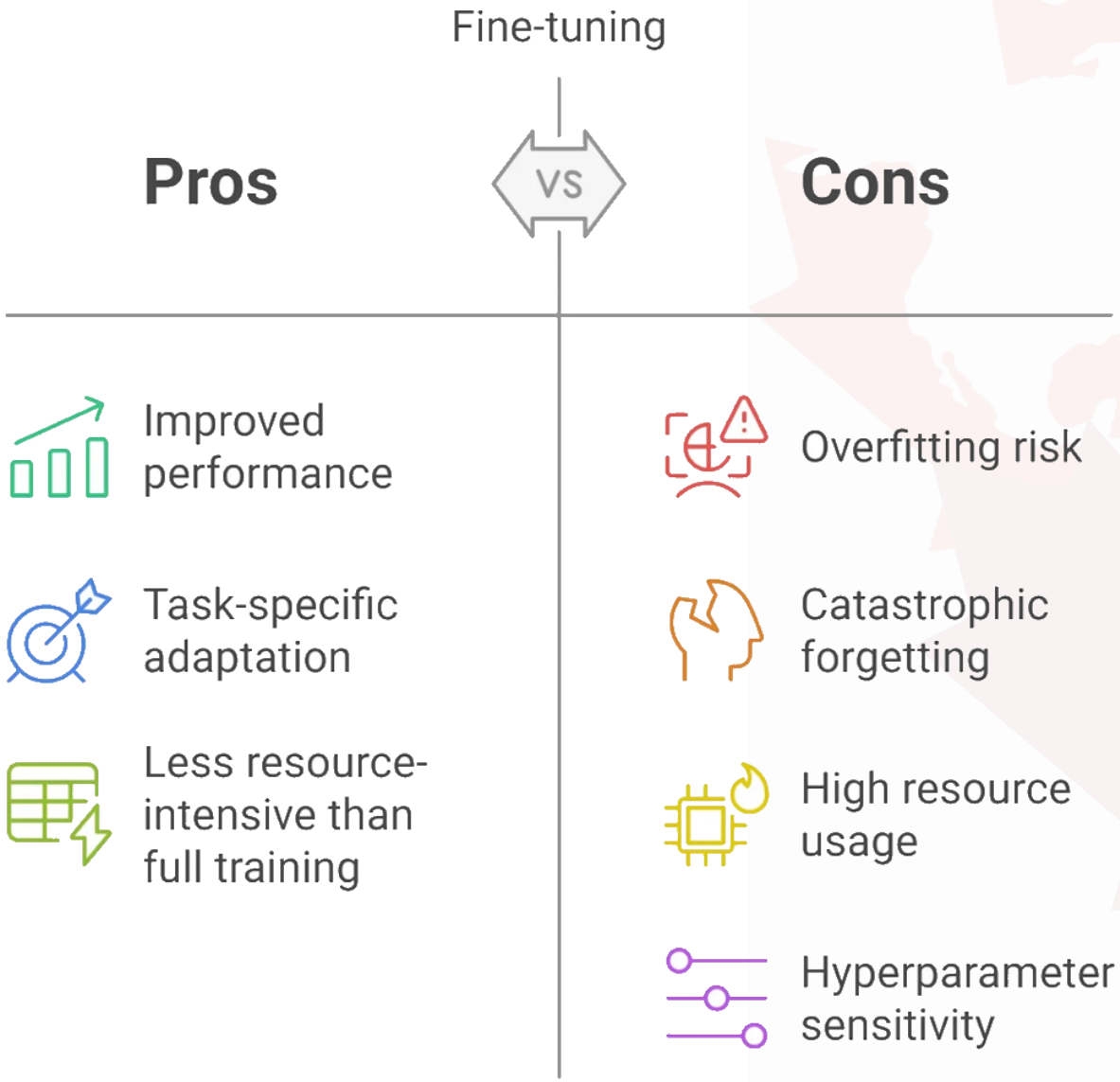
Fine-tuning x Transfer learing

Feature	Fine-tuning	Transfer Learning
Camadas treinadas	Tipicamente as camadas finais	Algumas ou todas as camadas
Dados necessários	Pouco ou moderado	Moderado
Tempo de treinamento	Baixo	Moderado
Flexibilidade	Menos flexível	Mais adaptável

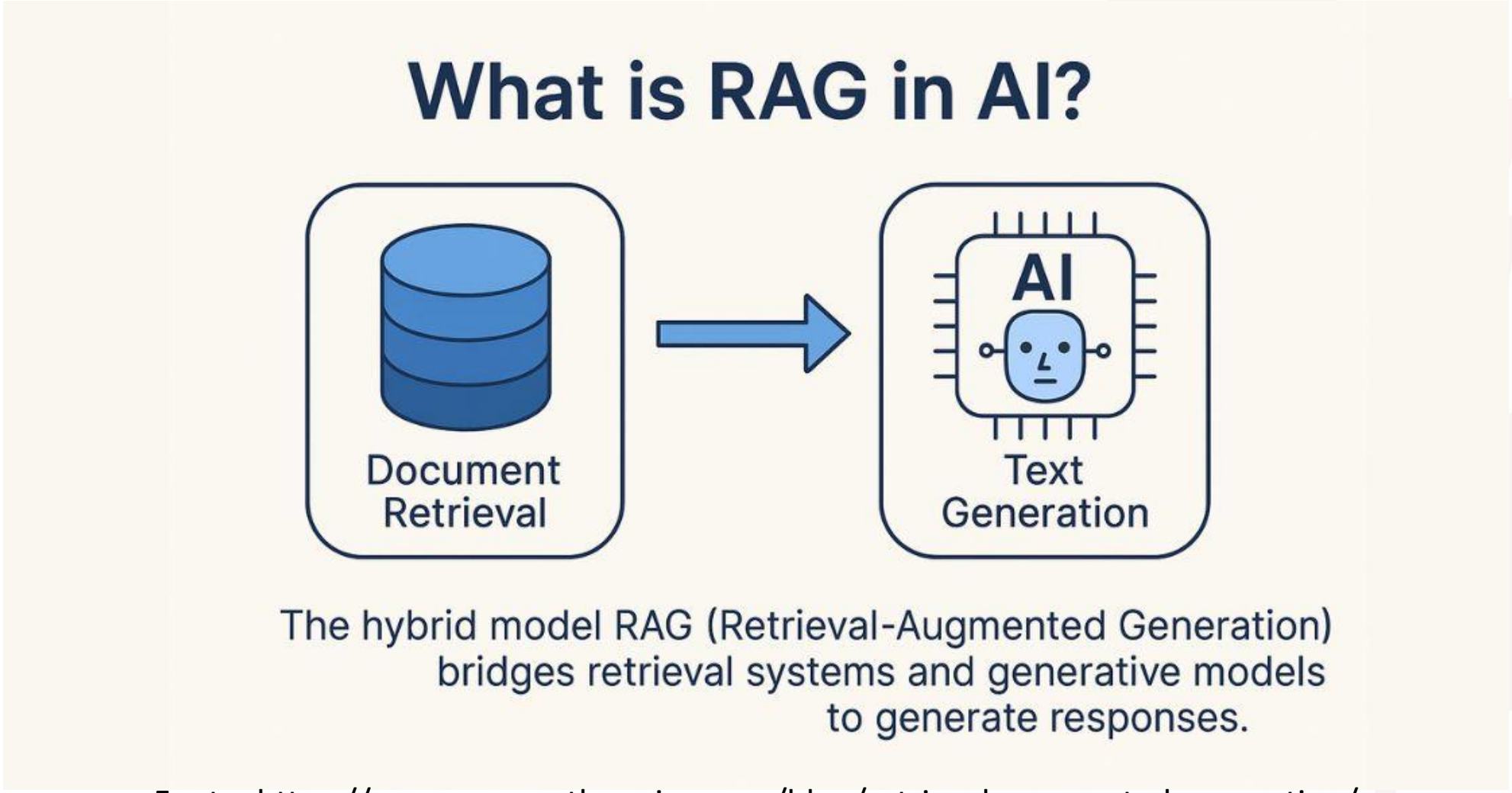
Aplicações do Fine-Tuning



Desafios


Fine-tuning
Pros
VS
Cons
Improved performance
Task-specific adaptation
Less resource-intensive than full training
Overfitting risk
Catastrophic forgetting
High resource usage
Hyperparameter sensitivity

Retrieval Augmented Generation (RAG)

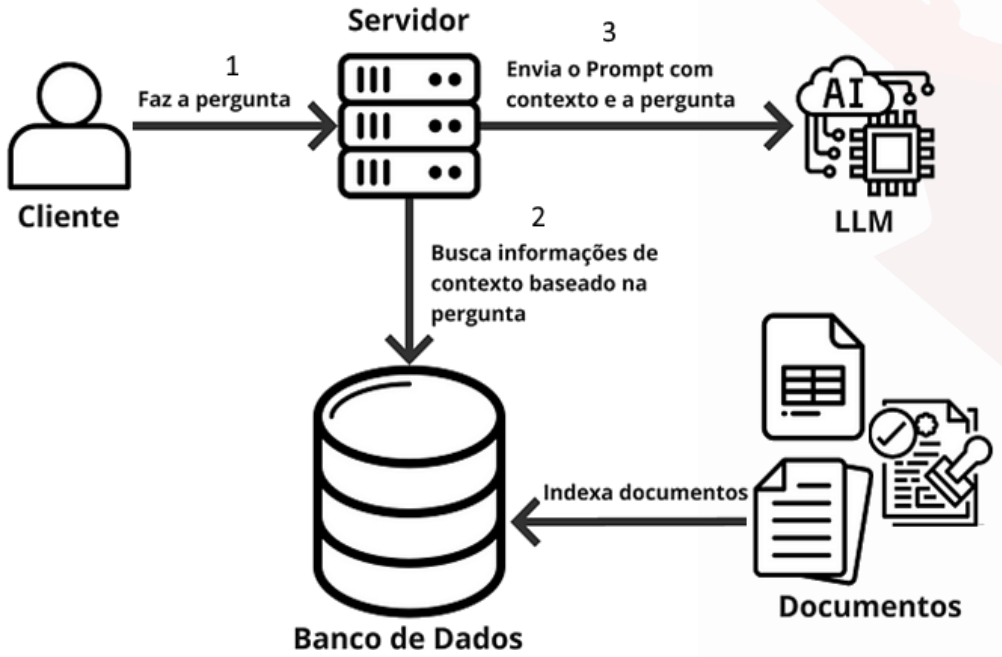


Fonte: https://www.mygreatlearning.com/blog/retrieval-augmented-generation/

Retrieval Augmented Generation (RAG)

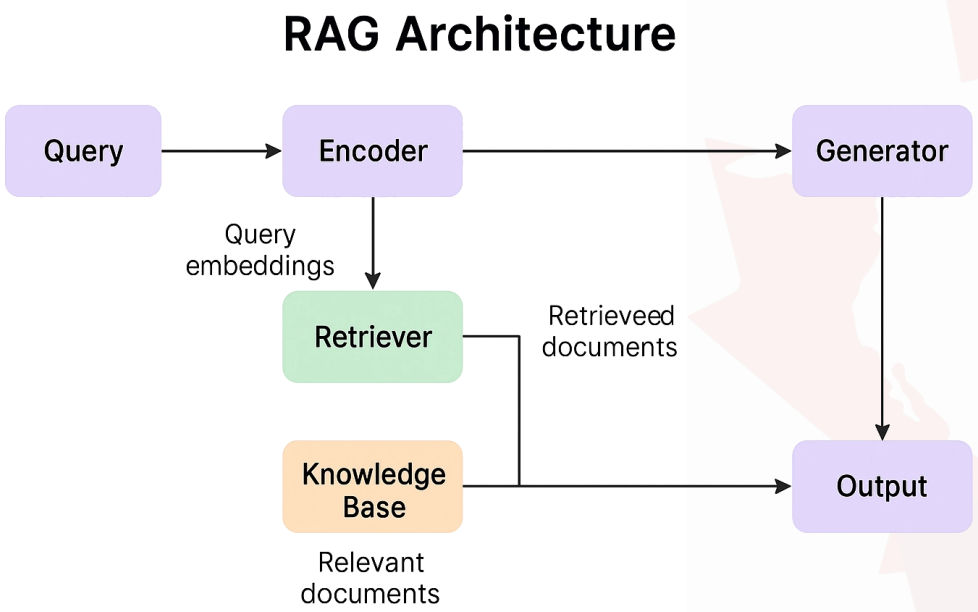
RAG é uma técnica que complementa o prompt com dados externos na forma de blocos de documentos, considerando os limites de tamanho do prompt.

Adiciona um mecanismo de busca, preenchendo a lacuna entre o conhecimento estático do modelo e dados dinâmicos do mundo real.



Fonte: https://medium.com/fretebras-tech/langchain-construindo-um-chatpdf-com-rag-retrieval-augmented-generation-openai-e-streamlit-61125e2122de

Componentes do RAG



Componente	Descrição
Encoder	Converte o texto da pergunta em um vetor
Retriever	Encontra os embeddings dos documentos que são similares ao embedding da pergunta
Generator	Sintetiza a saída considerando tanto o texto da pergunta quanto os trechos recuperados
Knowledge Base	Banco de dados estático ou dinâmico contendo os dados a pesquisar

Passos para utilização do RAG

Passos para a utilização do RAG

1. Indexação
    1. Dividir os textos em “chunks”
    2. Criar os embeddings
    3. Armazenar em banco de dados vetorial
2. Usuário submete uma questão
    1. Converter a consulta em vetores (usar a mesma técnica de vetorização)
3. Recuperação
    1. Efetuar a pesquisa a partir da similaridade dos embeddings
    2. Selecionar os top “k” resultados
4. Augmentation (fusão contextual)
    1. Combinar os fragmentos de texto no contexto do prompt
5. Generation
    1. Submeter o prompt à LLM
    2. Uma resposta coerente e factual é retornada

```
prompt = """Você é um
assistente de IA que responde as
dúvidas dos usuários com bases
nos documentos a baixo.
Os documentos abaixo
apresentam as fontes
atualizadas e devem ser
consideradas como verdade.
Cite a fonte quando fornecer a
informação.
Documentos:
{documents}
"""
```

Bancos vetoriais

- Qdrant: Opção de código aberto para armazenamento e busca eficiente de embeddings vetoriais.
- Chroma: Outra opção de código aberto entre os bancos de dados vetoriais especializados.
- Pinecone: Serviço baseado em nuvem, otimizado para busca de similaridade
- pgvector: Extensão para o PostgreSQL. Permite armazenar embeddings como strings e realizar operações vetoriais
- Elasticsearch: Pode ser usado para armazenar embeddings e também oferece busca baseada em palavras-chave (como BM25)
- OpenSearch: Similar ao Elasticsearch. Banco de dados de código aberto para RAG
- Redis: Pode ser utilizado como um banco de dados vetorial de código aberto.
- Databricks Vector Search: Plataforma de código fechado que oferece recursos para indexação e busca de vetores para aplicações RAG.

Exemplo Simples de RAG

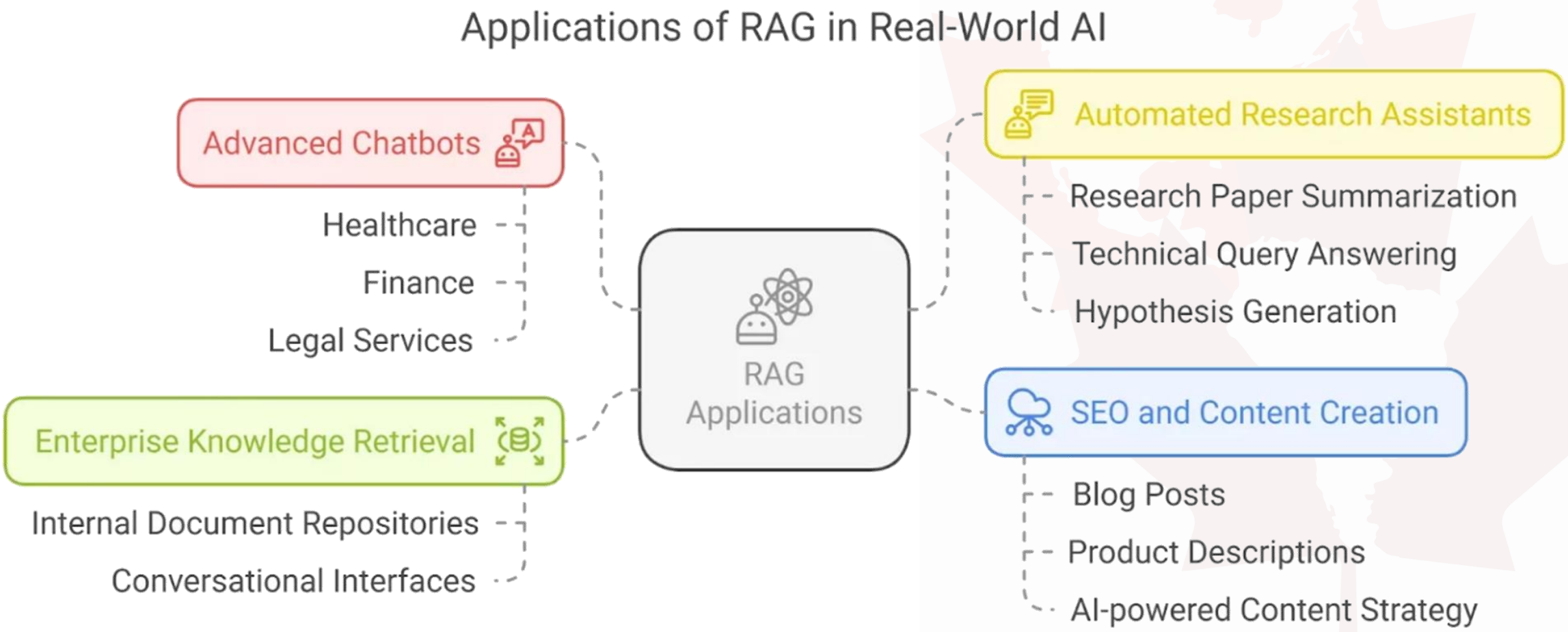
```python
from langchain import OpenAI, DocumentLoader, TextSplitter, VectorStore

# Carregar documentos
document_loader = DocumentLoader("caminho/para/seus/documentos")
documents = document_loader.load()

# Dividir os documentos em trechos menores
text_splitter = TextSplitter(chunk_size=500)
chunks = text_splitter.split(documents)

# Criar uma Vector Store com embeddings da OpenAI
vector_store = VectorStore.from_documents(chunks, embedding_model="openai-embedding")

# Função de recuperação de trechos relevantes
def retrieve_relevant_chunks(query):
    return vector_store.similarity_search(query, top_k=5)

# Função de geração de texto utilizando os trechos recuperados
def generate_response(query):
    relevant_chunks = retrieve_relevant_chunks(query)
    context = " ".join([chunk.text for chunk in relevant_chunks])
    prompt = f"Contexto: {context}\n\nPergunta: {query}\n\nResposta:"
    response = OpenAI().generate(prompt)
    return response

# Exemplo de uso
query = "Qual é a capital do Brasil?"
response = generate_response(query)
print(response)
```

Fonte: https://hub.asimov.academy/tutorial/o-que-e-a-tecnica-de-retriever-augmented-generation-rag/

Benefícios

- **Acurácia Factual**: O RAG baseia suas respostas em dados externos, reduzindo alucinações da IA.
- **Respostas Atualizadas**: Pode acessar conhecimento em tempo real, diferentemente dos LLMs tradicionais limitados a dados até o corte do pré-treinamento.
- **Adaptabilidade ao Domínio**: Fácil adaptação a setores específicos modificando a base de conhecimento subjacente.

Aplicações do RAG



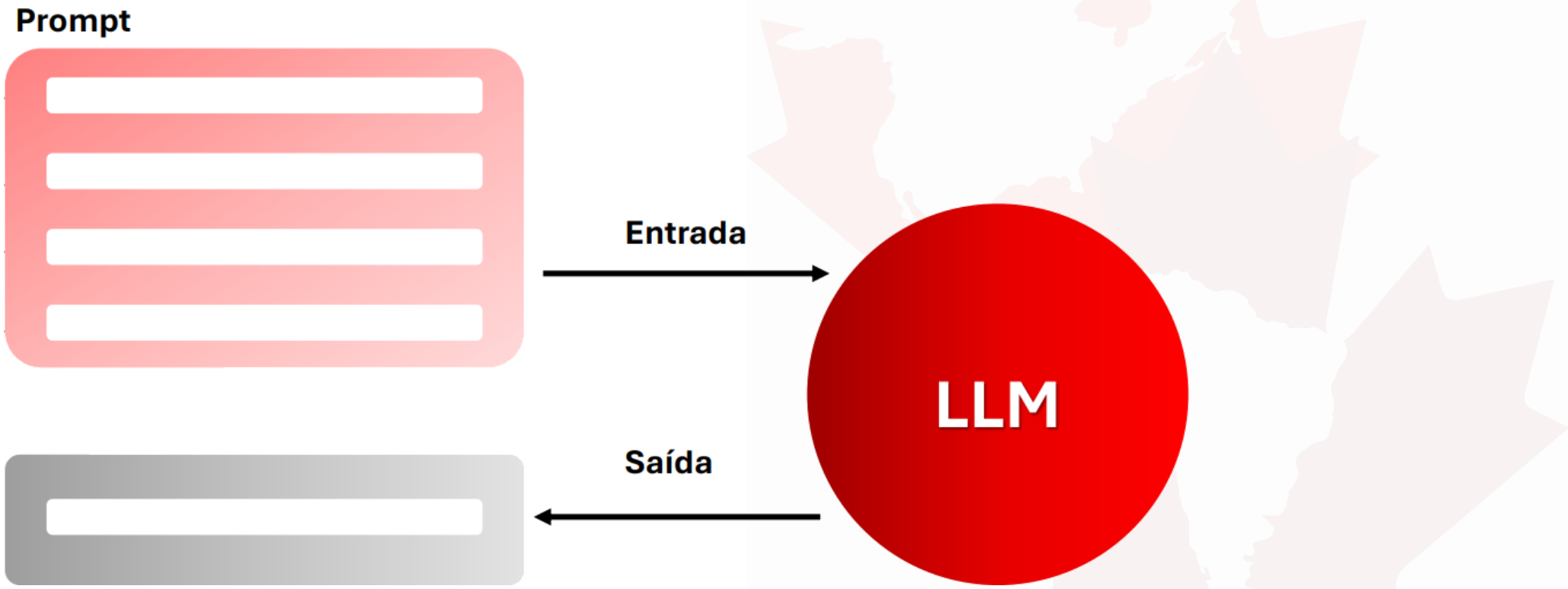
Desafios do RAG

- **Precisão na Recuperação**: Se documentos irrelevantes forem recuperados, o gerador pode produzir respostas fora do tema ou incorretas.
- **Complexidade Computacional**: A adição de uma etapa de recuperação aumenta o tempo de inferência e o uso de recursos.
- **Manutenção da Base de Conhecimento**: A precisão das respostas depende fortemente da qualidade e da atualização da base de conhecimento.

RAG ou Fine-Tuning?

Característica	RAG (Retrieval-Augmented Generation)	Fine-tuning (Ajuste Fino)
Modificação do LLM	**Não modifica o LLM subjacente**; suplementa-o com informações recuperadas.	**Ajusta os pesos e parâmetros** do LLM.
Fonte de Conhecimento	Recupera informações de fontes externas (bancos de dados, documentos, web).	Os dados de treinamento adicionais são **embutidos na arquitetura do modelo**.
Natureza dos Dados	Mais adequado para **informações que são atualizadas regularmente** e dinâmicas.	Adequado para **padrões que não mudam com o tempo** e dados estáticos. Informações podem ficar desatualizadas e exigir retreinamento.
Custo e Recursos	Tende a ser **mais econômico**, pois utiliza dados existentes e reduz a necessidade de retreinamento caro.	Tradicionalmente **mais caro e intensivo em computação**, exigindo muitos dados e hardware de ponta.
Habilidade Técnica	Requer **habilidades de codificação e arquitetura** para construir sistemas de pipeline. Comparativamente mais acessível para obter feedback e solucionar problemas.	Requer **experiência com NLP, aprendizado profundo, configuração de modelos e pré-processamento de dados**. Geralmente mais técnico e demorado.
Transparência	Pode **citar suas fontes**, facilitando a rastreabilidade do output.	Menos direto para rastrear a origem de informações específicas na saída do modelo, pois o conhecimento é internalizado.
Alucinações	Ajuda a **reduzir alucinações** ao fundamentar as respostas em dados recuperados verificáveis.	Pode alucinar se o contexto ou os dados de treinamento forem mal interpretados.
Contexto de Dados	Permite que o LLM acesse informações **domínio-específicas e atualizadas**.	Ajuda o modelo a **interpretar o conhecimento que já tem** de forma mais refinada (e.g., nuances, terminologias).

Engenharia de Prompts

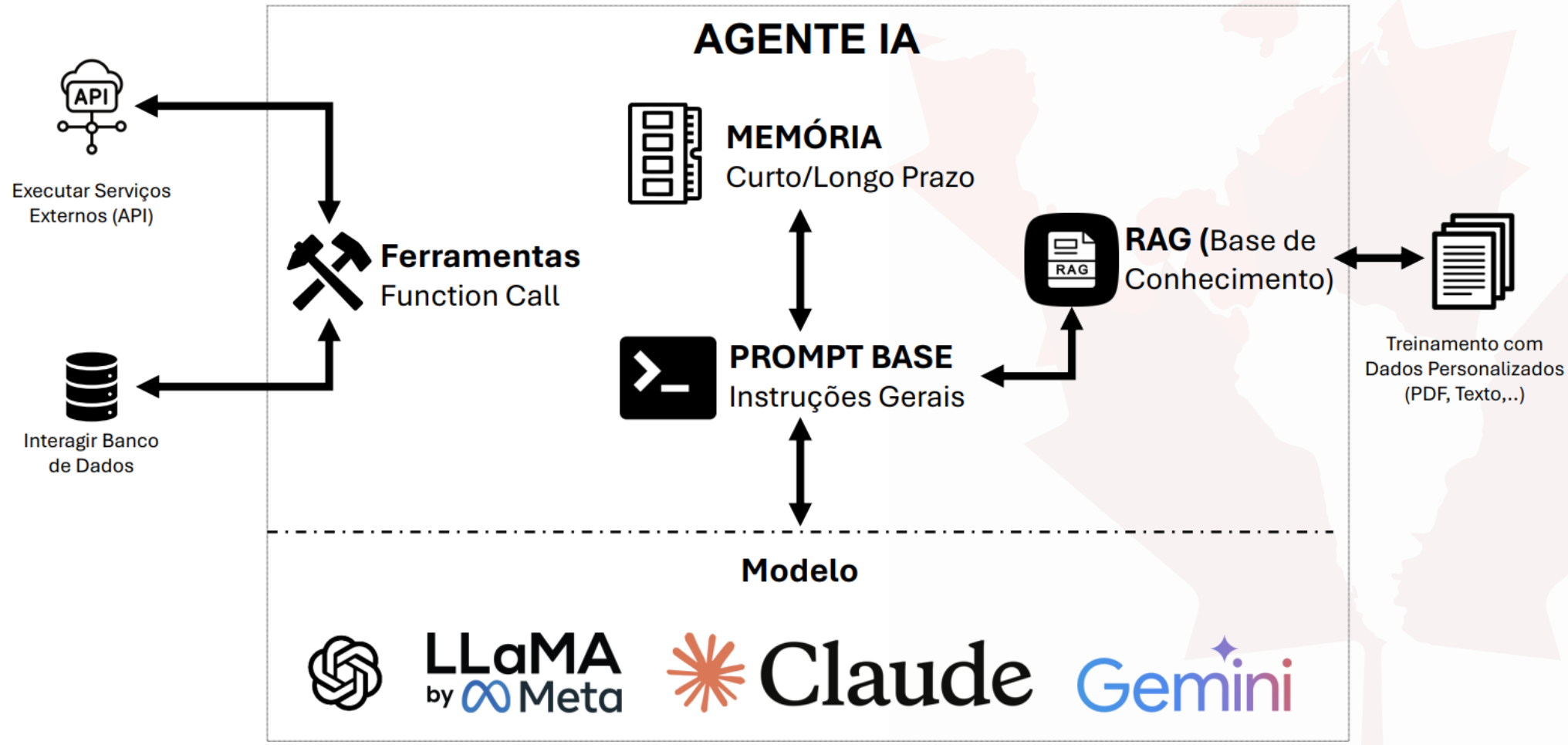


Engenharia de prompts

- Contexto
- Papel
- Expectativa
- Delimitadores
- Instruções
- Formato da resposta
- Placeholders
- Exemplos

In-context Learning	
Característica da Tarefa	**Tipo de Prompt**
Tarefa simples	Zero-shot
Pouca variação de saída	One-shot
Tarefa complexa com padrão desejado	Few-shot
Raciocínio, lógica ou contexto encadeado	Chain-of-Tought

Agentes


AGENTE IA
API
Executar Serviços Externos (API)
Interagir Banco de Dados
Ferramentas
Function Call
MEMÓRIA
Curto/Longo Prazo
PROMPT BASE
Instruções Gerais
RAG
RAG (Base de Conhecimento)
Treinamento com Dados Personalizados (PDF, Texto,..)
Modelo
LLaMA by Meta
Claude
Gemini


PRACTICE
MAKES
PERFECT

Obrigado!

I2a2.academy

+55 16 99213-2650

celso@i2a2.academy

/in/celso-augusto-morato-azevedo