

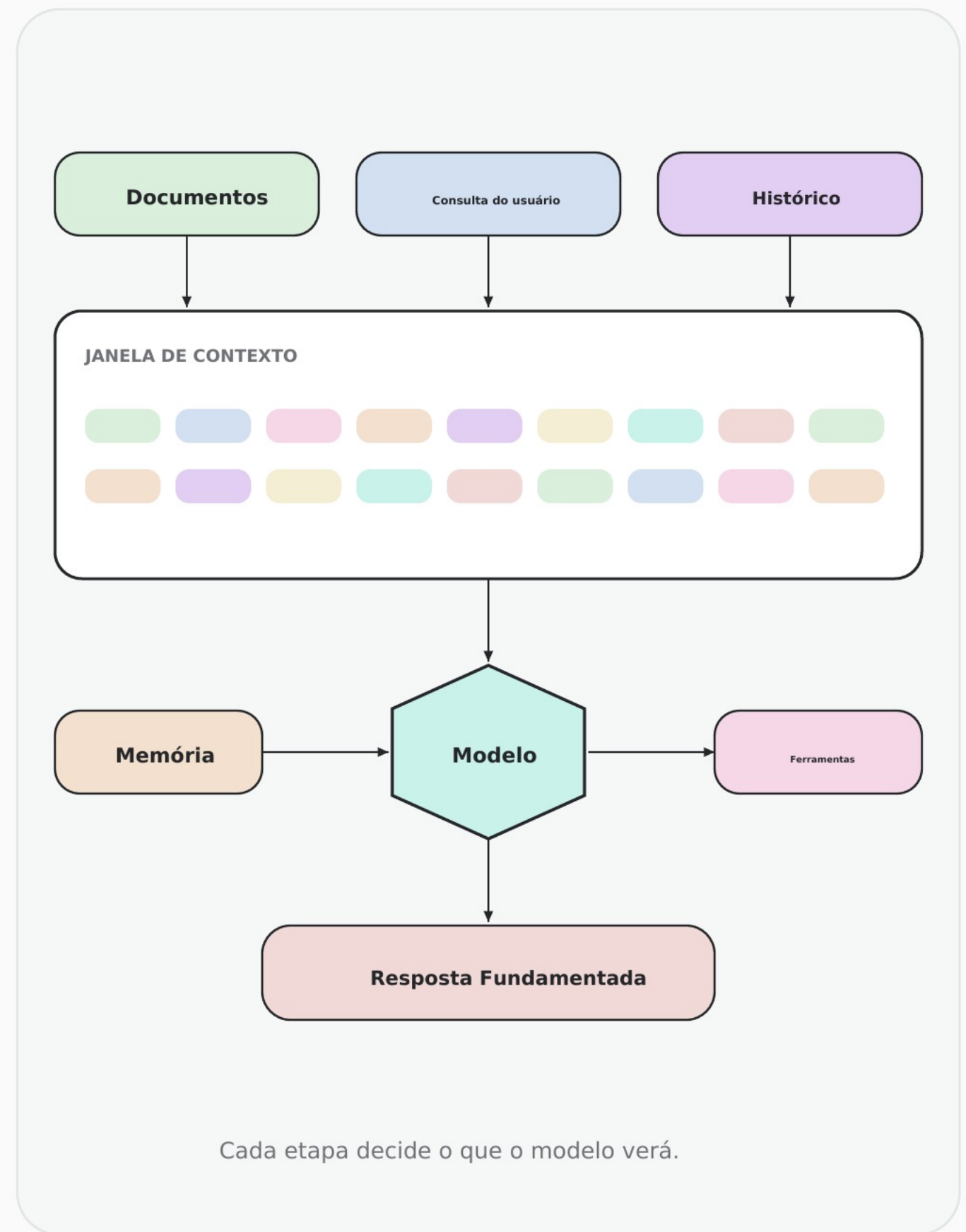
UM GUIA DE CAMPO

Engenharia de Contexto

Projetar os sistemas que controlam o que a informação chega ao modelo e como ela mantém a coerência.

Agentes • Aumento de Consultas • Recuperação • Segmentação

Engenharia de Prompts • Memória • Ferramentas

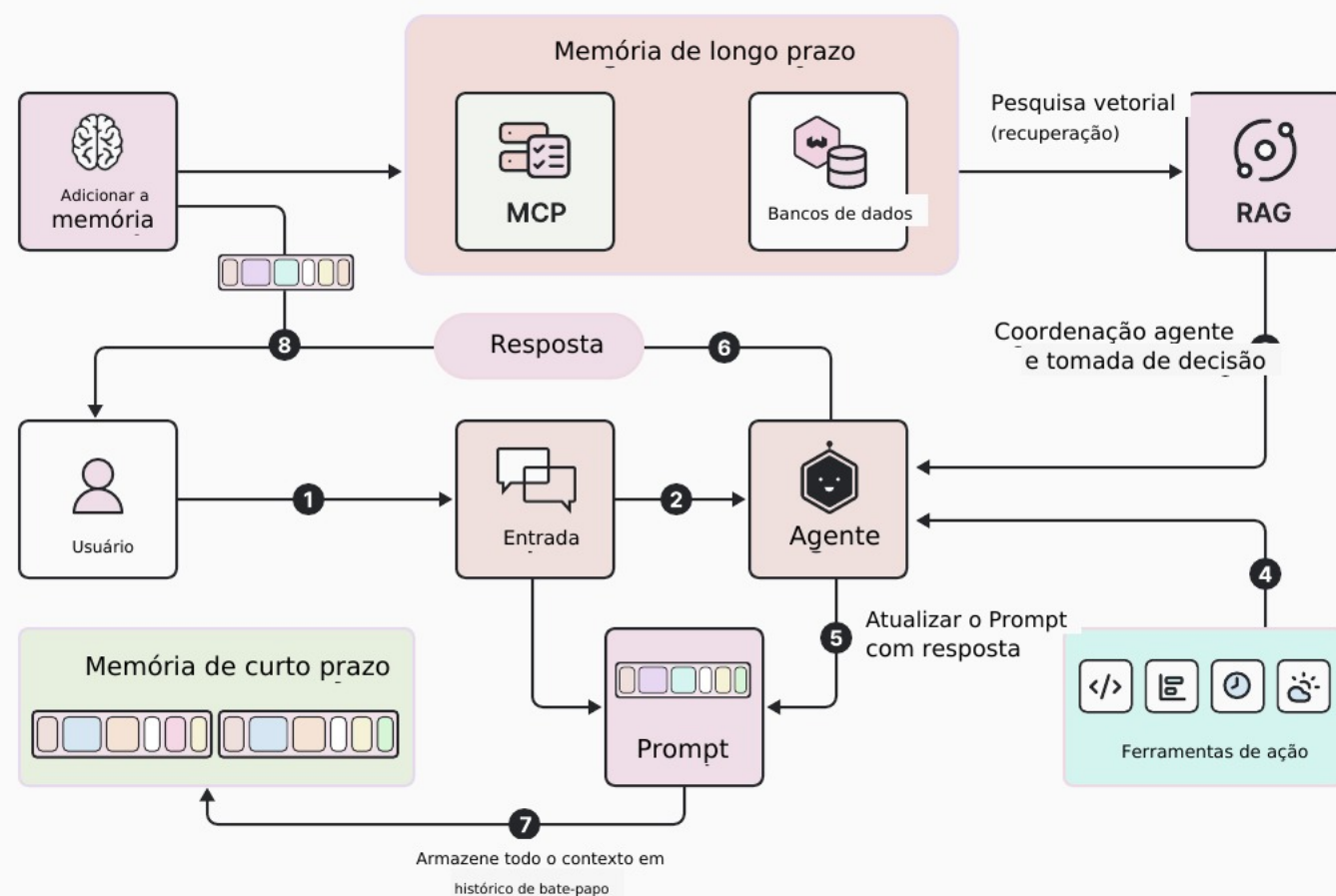


Sumário

Introdução	O que é engenharia de contexto?	01
Agentes	O que são agentes?	04
	O desafio da janela de contexto	05
	Estratégias e tarefas para agentes	07
	Onde os agentes se enquadram na engenharia de contexto	08
Aumento de Consultas	Reescrita de Consultas	10
	Expansão de Consultas	11
	Decomposição de Consultas	12
	Agentes de Consulta	13
Recuperação	Guia de Estratégias de Segmentação	15
	Estratégias Simples de Segmentação	17
	Estratégias Avançadas de Segmentação	18
	Pré-Segmentação vs. Pós-Segmentação	20
	Resumo	23
Técnicas de Engenharia de Prompts	Técnicas Clássicas de Engenharia de Prompts	25
	Técnicas Avançadas de Engenharia de Prompts	26
	Engenharia de Prompts para Uso de Ferramentas	26
	Uso de Frameworks de Prompts	27
Memória	A Arquitetura da Memória do Agente	29
	Princípios-chave para gerenciamento eficaz de memória	31
Ferramentas	A evolução: das instruções às ações	35
	O desafio da orquestração	36
	A próxima fronteira do uso de ferramentas	39
Resumo	O futuro da engenharia de IA	40

Introdução

Todo desenvolvedor que constrói com modelos de linguagem de grande porte (LLMs) eventualmente atinge o mesmo paredão. Você começa com um modelo poderoso que pode escrever, resumir e raciocinar com capacidade impressionante. Mas quando você tenta aplicá-lo a um problema do mundo real, as rachaduras começam a aparecer. Não pode responder perguntas sobre seus documentos privados. Não tem conhecimento dos acontecimentos que aconteceram ontem. Ele inventa coisas com confiança quando não sabe uma resposta.



O problema não é a inteligência do modelo. O problema é que ele está fundamentalmente desconectado. É um cérebro poderoso, mas isolado, sem acesso aos seus dados específicos, à Internet ao vivo ou até mesmo a uma memória da sua última conversa. Este isolamento é um resultado direto de seu limite arquitetônico central: a janela de contexto. A janela de contexto é a memória de trabalho ativa do modelo - o finito espaço onde contém as instruções e informações para a tarefa atual. Cada palavra, número e parte da pontuação consome espaço nesta janela. Assim como um quadro branco, uma vez cheio, as informações mais antigas são apagadas para abrir espaço para novas instruções e importantes detalhes podem ser perdidos.

Você não pode corrigir essa limitação fundamental apenas escrevendo prompts melhores. Você tem que construir seu sistema em torno do modelo.

Isso é Engenharia de Contexto.

Engenharia de Contexto é a disciplina de projetando a arquitetura que alimenta um LLM a informação certa no momento certo. Não é sobre mudar o próprio modelo, mas sobre construindo as pontes que o conectam ao mundo exterior, recuperando dados externos, conectá-lo a ferramentas ativas e dar-lhe uma memória para fundamentar suas respostas em fatos, não apenas seus dados de treinamento.

Este e-book é o modelo para esse sistema. Nós cobrimos os principais componentes necessários para transformar um modelo brilhante, mas isolado, em um modelo confiável, aplicativo pronto para produção.

Dominar esses componentes é o diferencial entre uma demonstração razoável e uma verdadeiramente sistema inteligente. Vamos trabalhar.

Agentes

Um cérebro que toma decisões orquestra como e quando usar informação.

Consulta Aumento

A arte de traduzir confuso, solicitações ambíguas de usuários em intenção precisa e legível por máquina.

Recuperação

A ponte que liga o LLM ao mundo real, recuperando dados externos, bases de conhecimento.

Engenharia de Prompts Técnicas

A habilidade de dar informações claras e eficazes instruções para orientar o modelo raciocínio.

Memória

O sistema que dá ao seu aplicação um senso de história e a capacidade de aprender com as interações.

Ferramentas

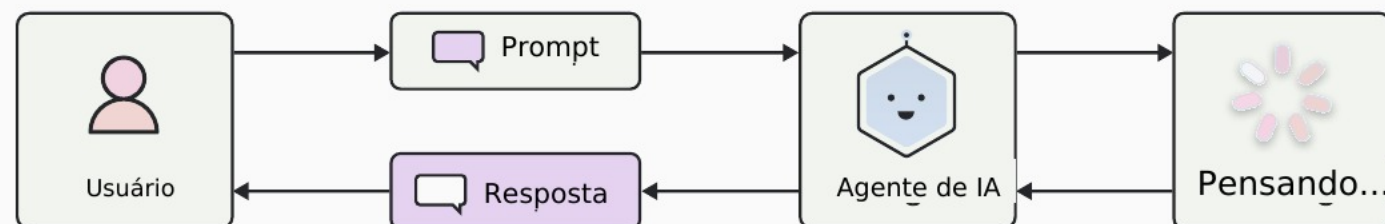
As mãos que permitem o seu aplicativo para tomar medidas directas e interagir com fontes de dados ativas.

Agentes

Assim que você começar a construir sistemas reais com grandes modelos de linguagem, você se deparará com o limites de pipelines estáticos. Uma receita fixa de "recuperar e depois gerar" funciona bem para tarefas simples. Configurações de geração aumentada de recuperação (RAG), mas desmorona quando a tarefa exige julgamento, adaptação ou raciocínio em várias etapas.

É aqui que entram os agentes. No contexto da engenharia de contexto, os agentes gerenciam como (e quão bem) a informação flui através de um sistema. Em vez de seguir cegamente um roteiro, os agentes podem avaliar o que sabem, decidir o que ainda precisam, selecionar as ferramentas certas, e ajustar sua estratégia quando as coisas dão errado.

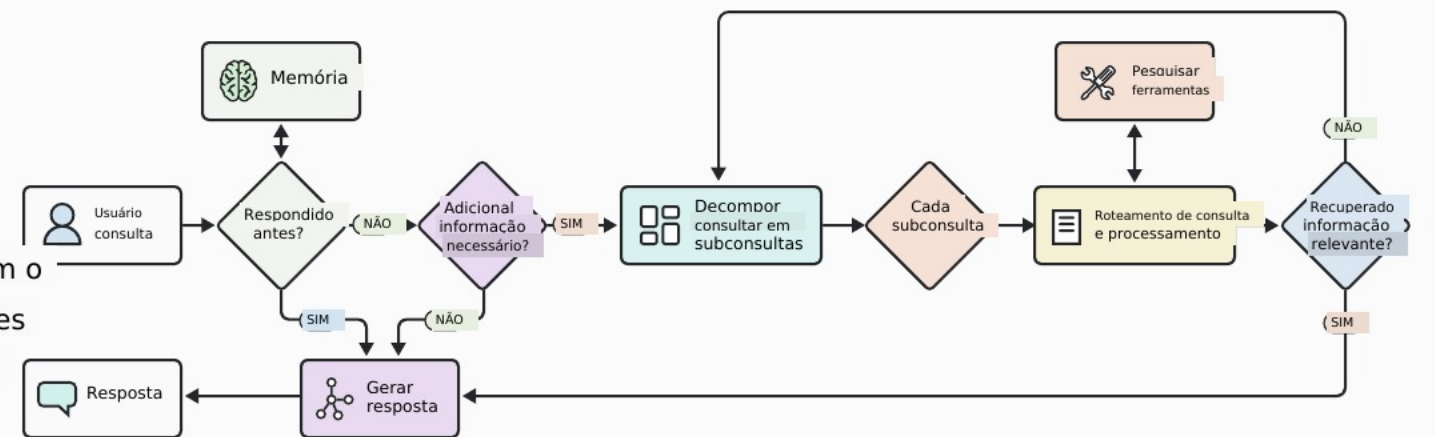
Os agentes são tanto os arquitetos de seus contextos quanto os usuários desses contextos. No entanto, necessitam de boas práticas e sistemas que os orientem, porque a gestão contextualizar bem é difícil, e errar rapidamente sabota todo o resto que o agente pode fazer.



O que são agentes?

O termo "agente" é usado de forma ampla, então vamos defini-lo no contexto da construção com grandes modelos de linguagem (LLMs). Um agente de IA é um sistema que pode:

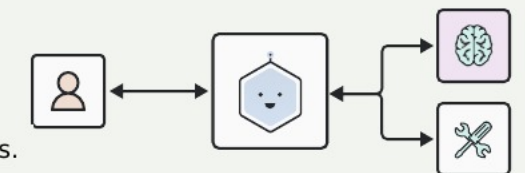
- 1 Tome decisões dinâmicas sobre informações fluxo. Em vez de seguir um padrão pré-determinado caminho, os agentes decidem o que fazer a seguir com base sobre o que aprenderam.
- 2 Mantenha o estado em múltiplas interações. Ao contrário dos sistemas simples de perguntas e respostas, os agentes lembre-se do que eles fizeram e use isso história para informar decisões futuras.



- Lenda
- Raciocínio
 - Uso de ferramentas
 - Memória
- 4 Modifique sua abordagem com base nos resultados. Quando uma estratégia não está funcionando, eles podem tentar abordagens diferentes.
 - 3 Use ferramentas de forma adaptativa. Eles podem selecionar entre as ferramentas disponíveis e combiná-los de maneiras que não foram explicitamente programados.

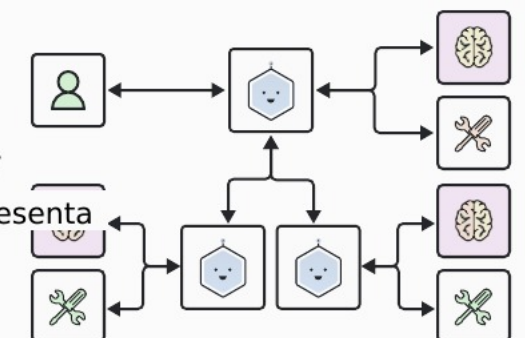
Arquitetura de agente único

Tentar realizar todas as tarefas sozinho, o que funciona bem para fluxos de trabalho moderadamente complexos.



Arquitetura Multiagente

Distribuir o trabalho entre agentes especializados por tarefa. Permite fluxos de trabalho complexos, mas apresenta desafios de coordenação.



Higiene do Contexto

Esta é uma das partes mais críticas do gerenciamento de sistemas de agentes. Os agentes não precisam apenas memória e ferramentas; eles também precisam monitorar e gerenciar a qualidade de seus próprios contexto. Isso significa evitar sobrecarga, detectar informações irrelevantes ou conflitantes, podendo ou compactando conforme necessário e mantendo sua memória no contexto limpa o suficiente para raciocinar de forma eficaz.

O desafio da janela de contexto

LLMs têm capacidade de informação limitada porque a janela de contexto só pode reter tantas informações de uma só vez. Esta restrição fundamental molda o que agentes e sistemas agênicos são atualmente capazes.

Cada vez que um agente processa informações, ele precisa tomar decisões sobre:

- ✓ Quais informações devem permanecer O que deve ser armazenado externamente ativo na janela de contexto e recuperado quando necessário
- ✓ O que pode ser resumido ou compactado para economizar espaço
- ✓ Quanto espaço reservar para raciocínio e planejamento

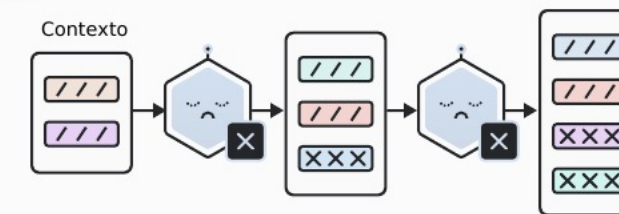


É tentador presumir que janelas de contexto maiores resolvem isso problema, mas este simplesmente não é o caso. Contextos mais longos (centenas de milhares ou até ~1 milhão de tokens) na verdade introduz novos modos de falha.

O desempenho muitas vezes começa a degradar muito antes de o modelo atingir capacidade máxima de token, onde os agentes ficarão confusos, taxas mais altas de alucinação, ou simplesmente parar de atuar no nível eles normalmente são capazes. Esta não é apenas uma limitação técnica, é uma principal desafio de design de qualquer aplicativo de IA.

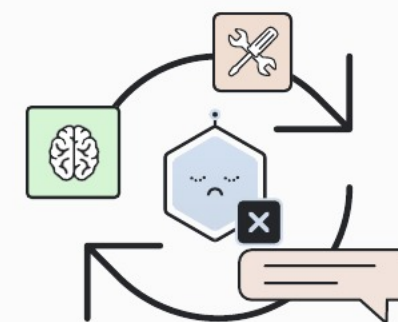
Aqui estão alguns tipos comuns de erros que começam a acontecer ou aumentam

conforme o tamanho da janela de contexto aumenta:



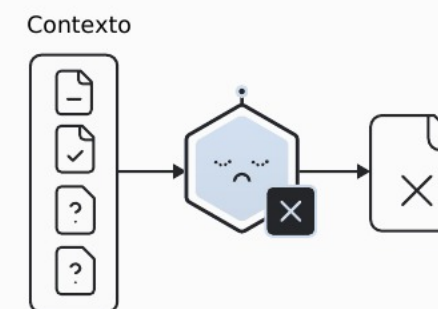
Envenenamento de Contexto

Informações incorretas ou alucinadas entra no contexto. Porque os agentes reutilizar e construir sobre esse contexto, esses erros persistem e se agravam.



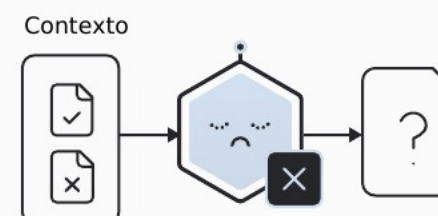
Distração de Contexto

O agente fica sobrecarregado com muitas coisas informações passadas - histórico, resultados de ferramentas, resumos - e depende demais da repetição comportamento passado em vez de raciocínio novo.



Confusão de Contexto

Ferramentas ou documentos irrelevantes lotam o contexto, distraindo o modelo e causando usar a ferramenta ou instruções erradas.



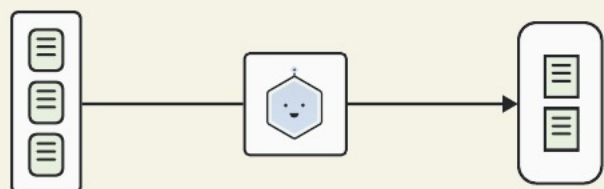
Conflito de Contexto

Informações contraditórias dentro do contexto engana o agente, deixando-o preso entre suposições conflitantes.

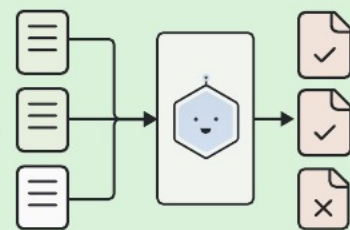
Estratégias e Tarefas para Agentes

Os agentes são capazes de orquestrar o contexto de forma eficaz sistemas devido à sua capacidade de raciocinar e tomar decisões de forma dinâmica. Aqui estão alguns das tarefas mais comuns para as quais os agentes são criados e empregar para gerenciar contextos.

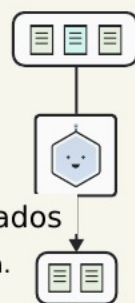
Resumo do Contexto:
Comprimindo periodicamente história acumulada em resumos para reduzir fardo enquanto preserva conhecimento chave.



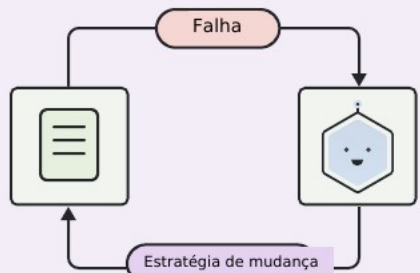
Validação de qualidade:
Verificando se informações recuperadas são consistente e útil.



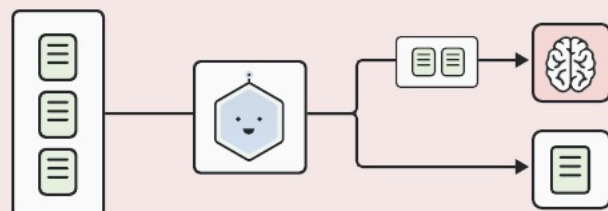
Poda de contexto:
Removendo ativamente irrelevante ou desatualizado contexto, seja com modelos de poda especializados ou uma ferramenta LLM dedicada.



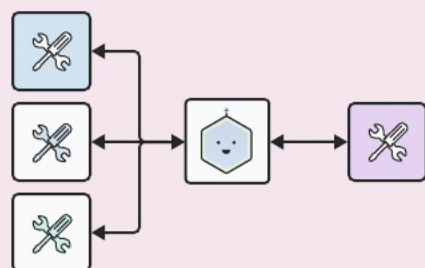
Estratégias de recuperação adaptativa:
Reformulando consultas, trocando bases de conhecimento ou alterando o segmentação estratégias quando as tentativas iniciais falham.



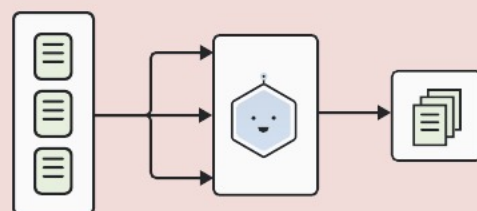
Descarregamento de Contexto: Armazenando detalhes externamente e recuperando-os apenas quando necessário, em vez de manter tudo em contexto ativo.



Seleção dinâmica de ferramentas: em vez de despejar todas as ferramentas possíveis no prompt, agentes de filtre e carregue apenas aqueles relevantes para a tarefa. Isso reduz a confusão e melhora a precisão.



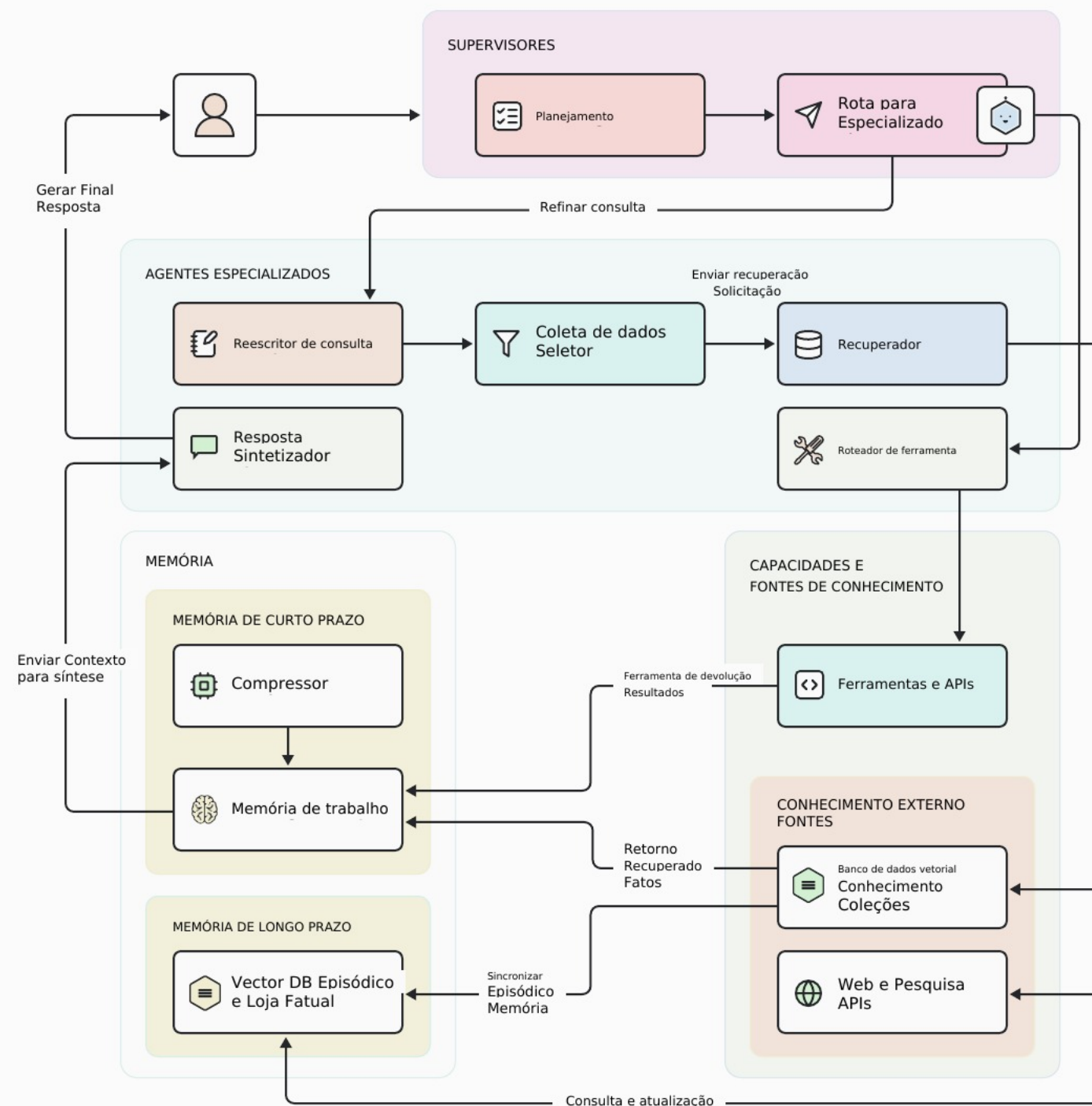
Síntese de múltiplas fontes:
Combinando informações de múltiplas fontes, resolvendo conflitos e produzindo respostas coerentes.



Onde os agentes se enquadram na engenharia de contexto

Os agentes atuam como coordenadores em seu sistema de engenharia de contexto. Eles não substituem o técnicas abordadas em outras seções, em vez disso, elas as orquestram de forma inteligente. Um agente pode aplicar a reescrita de consulta quando as pesquisas iniciais não forem bem-sucedidas, escolha diferentes estratégias de agrupamento com base no tipo de conteúdo encontrado ou decidir quando o histórico de conversas deve ser compactado para abrir espaço para novas informações. Eles fornecer a camada de orquestração necessária para tomar decisões dinâmicas e apropriadas ao contexto sobre gerenciamento de informações.

Diferentes tipos de agentes e funções dentro de um sistema de engenharia de contexto:



Aumento de Consultas

Uma das etapas mais importantes da engenharia de contexto é como você prepara e apresenta a consulta do usuário. Sem saber exatamente o que o usuário está perguntando, o LLM não pode fornecer uma resposta precisa.

Embora pareça simples, na verdade é bastante complexo. Há duas questões principais a serem pensadas sobre:

1 Os usuários muitas vezes não interagem com chatbots ou entradas da maneira ideal.

Muitos criadores de produtos frequentemente desenvolverão e testar chatbots com consultas que fornecem o pedido e todas as informações adicionais que o LLM precisaria entender o pergunta de forma sucinta, perfeitamente forma pontuada e clara. Infelizmente, no mundo real, interações do usuário com chatbots pode ser confuso, confuso e incompleto. Em para construir sistemas robustos, é importante implementar soluções que atendam a todos tipos de interações, não apenas as ideais.

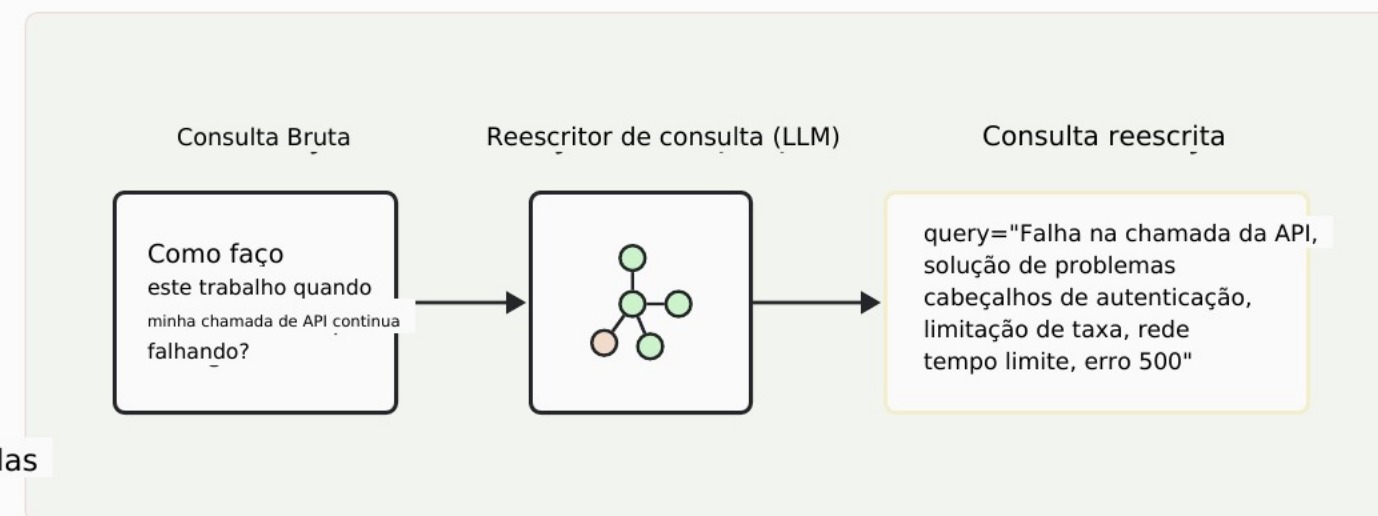
2 Diferentes partes do pipeline precisam lidar com a consulta de maneiras diferentes.

Uma pergunta que um LLM poderia entender bem, pode não ser o melhor formato para pesquisar através de um banco de dados vetorial com. Ou uma consulta termo que funciona melhor para um banco de dados vetorial pode estar incompleto para um LLM responder. Portanto, precisamos de uma maneira de aumentar o consulta que se adapta a diferentes ferramentas e etapas dentro do gasoduto.

Lembre-se de que o aumento de consulta aborda o problema "entra lixo, sai lixo" em o início do seu pipeline. Nenhuma quantidade de algoritmos de recuperação sofisticados, avançados modelos de reclassificação ou engenharia rápida e inteligente podem compensar totalmente intenção do usuário.

Reescrita de Consultas

A reescrita da consulta transforma a consulta original do usuário em uma versão mais eficaz para recuperação de informações. Em vez de apenas recuperar e ler, os aplicativos agora fazem um abordagem reescrever-recuperar-ler. Esta técnica reestrutura perguntas estranhamente escritas para que eles podem ser melhor compreendidos pelo sistema, remove contexto irrelevante, introduz palavras-chave comuns que melhoram a correspondência com o contexto correto e podem dividir palavras complexas perguntas em subquestões mais simples.



As aplicações RAG são sensíveis ao fraseado e às especificidades palavras-chave da consulta, portanto esta técnica funciona da seguinte forma:



Reestruturação
Perguntas pouco claras: Remoção:

Transforma vago ou usuário mal formado entrada em preciso, denso em informações termos.



Contexto

Elimina irrelevantes informações que poderia confundir o processo de recuperação.

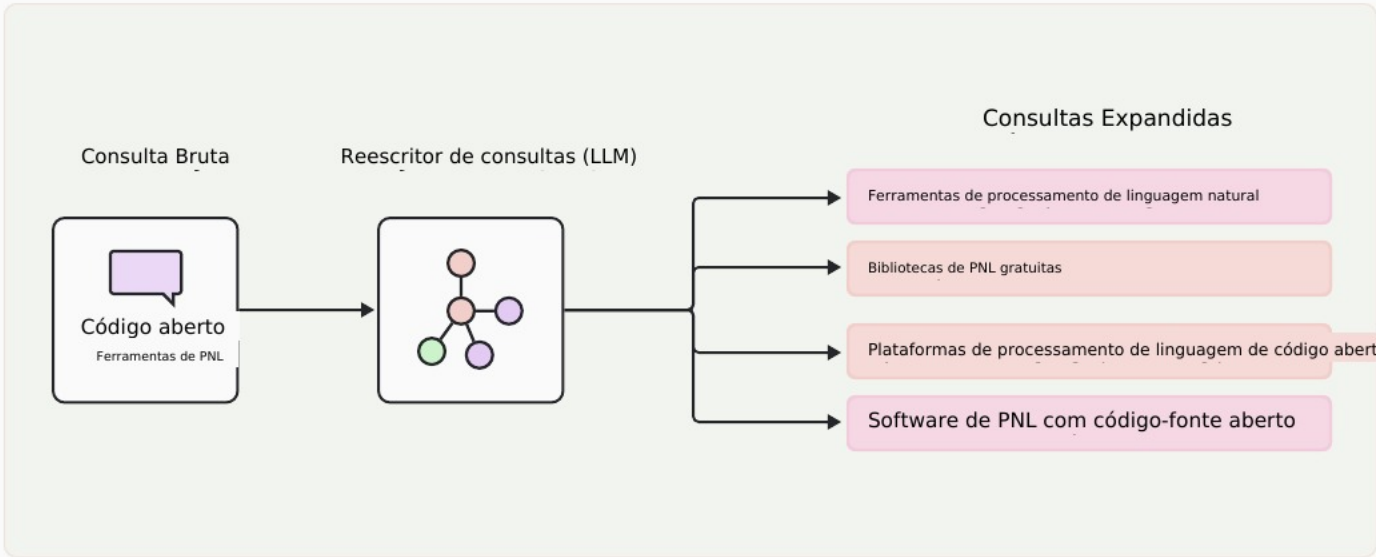


Palavra-chave
Aprimoramento:

Apresenta comum terminologia que aumenta o probabilidade de correspondência documentos relevantes.

Expansão de Consultas

A expansão de consultas aprimora a recuperação, gerando diversas consultas relacionadas a partir de um único A decomposição da consulta divide questões complexas e multifacetadas em questões mais simples e entrada do usuário. Essa abordagem melhora os resultados quando as consultas do usuário são vagas, mal formadas subconsultas focadas que podem ser processadas de forma independente. Esta técnica é especialmente bom para questões que exigem informações de múltiplas fontes ou envolvem vários ou quando você precisar de uma cobertura mais ampla, como sistemas de recuperação baseados em palavras-chave. conceitos relacionados.



No entanto, a expansão da consulta traz desafios que necessitam de uma gestão cuidadosa:

Desvio de consulta:

Consultas expandidas pode divergir do intenção original do usuário, levando a irrelevante ou excessivamente irrelevantes resultados fora do tópico.

Expansão excessiva: computacional

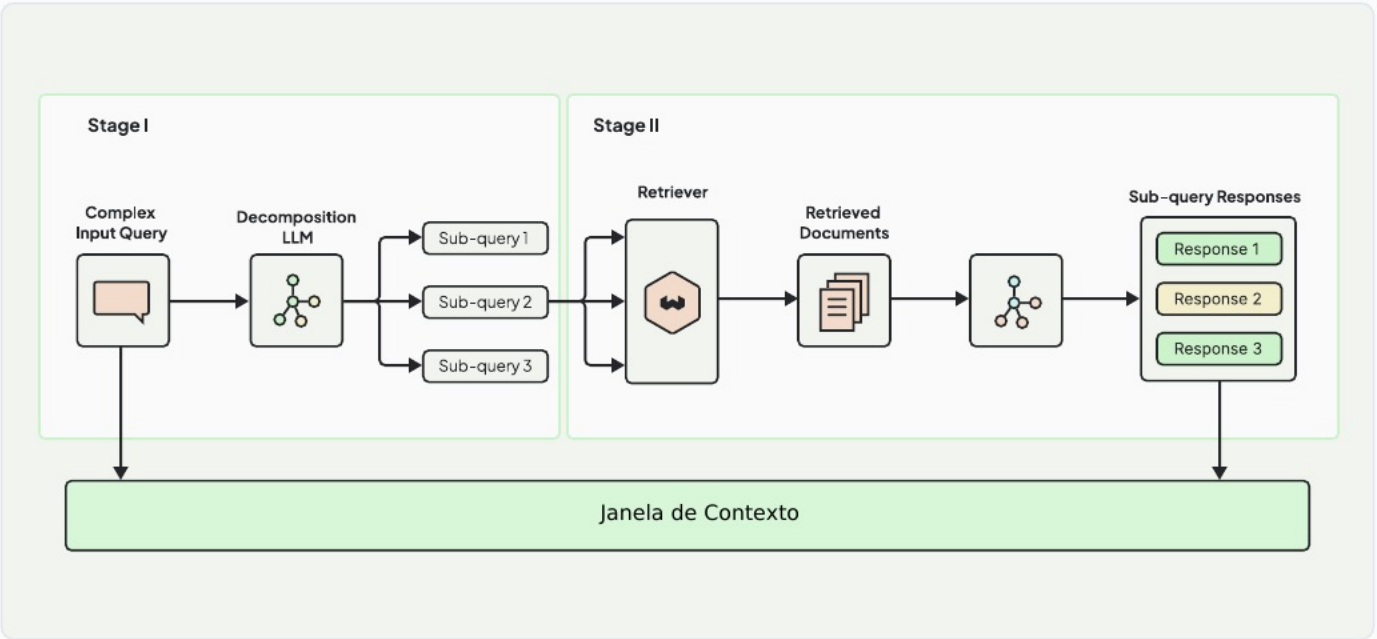
Adicionando muitos termos podem reduzir a precisão e a recuperação de consultas aumentam documentos.

Despesas gerais:

Processando vários latência do sistema e uso de recursos.

Decomposição de Consultas

A decomposição da consulta divide questões complexas e multifacetadas em questões mais simples e entrada do usuário. Essa abordagem melhora os resultados quando as consultas do usuário são vagas, mal formadas subconsultas focadas que podem ser processadas de forma independente. Esta técnica é especialmente bom para questões que exigem informações de múltiplas fontes ou envolvem vários ou quando você precisar de uma cobertura mais ampla, como sistemas de recuperação baseados em palavras-chave. conceitos relacionados.



O processo normalmente envolve duas etapas principais:

Fase de decomposição:

Um LLM analisa o original consulta complexa e a divide em subconsultas menores e focadas. Cada subconsulta tem como alvo um aspecto específico da pergunta original.

Fase de Processamento:

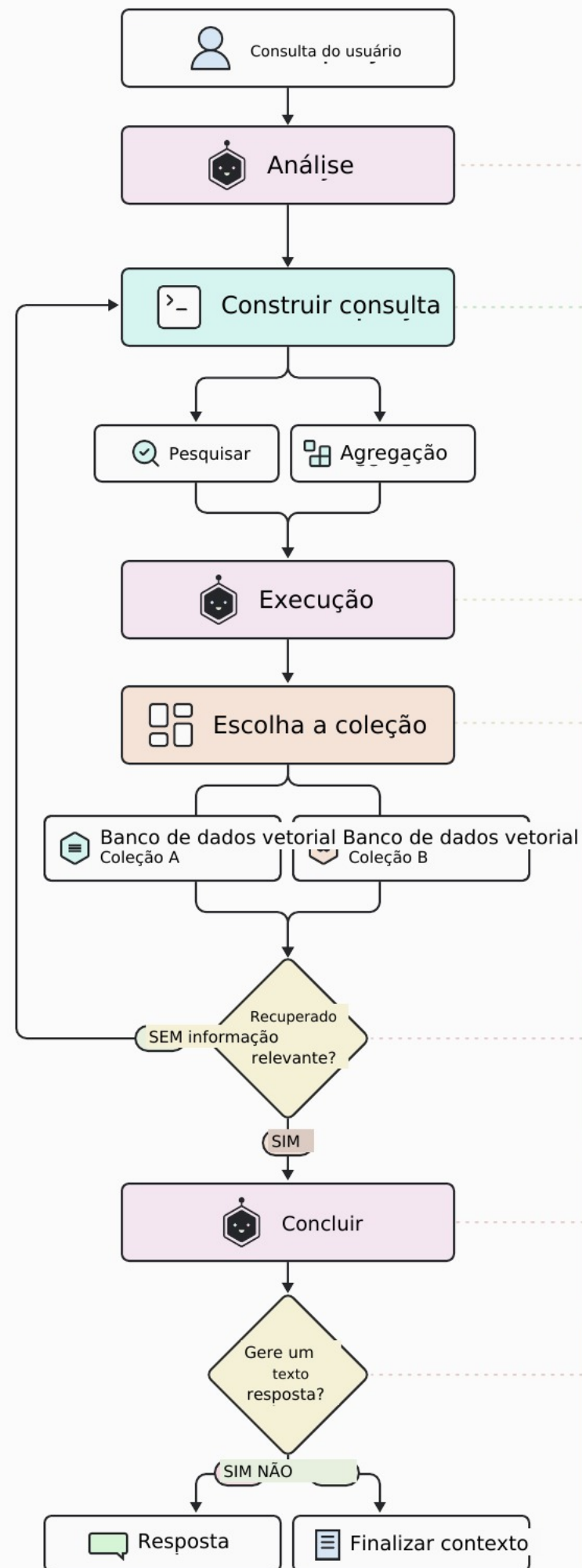
Cada subconsulta é processada de forma independente através da recuperação pipeline, permitindo dados mais precisos correspondência com documentos relevantes.

Após a recuperação, o sistema de engenharia de contexto deve agregar e sintetizar os resultados de todas as subconsultas para gerar uma resposta coerente e abrangente ao original consulta complexa.

Agentes de Consulta

Agentes de consulta são os mais forma avançada de consulta aumento, usando agentes de IA para lidar de forma inteligente com todo o pipeline de processamento de consultas, combinando as técnicas acima.

Um agente de consulta pega a informação de um usuário prompt/pergunta em natural idioma e decide o melhor maneira de estruturar a consulta com base no seu conhecimento do banco de dados e estrutura de dados, e pode decidir iterativamente consultar e ajustar com base no resultados retornados.



Análise: Use modelos generativos (por exemplo, modelos de linguagem grande) para analisar o tarefa e as consultas necessárias. Determine as consultas exatas a serem executadas.

Construção de consulta dinâmica: em vez de usar padrões de consulta predeterminados, o agente constrói consultas sob demanda com base na compreensão do usuário intenção e o esquema de dados. Isso significa que ele pode adicionar filtros e ajustar a pesquisa termos automaticamente para encontrar os resultados mais relevantes no banco de dados, bem como escolhendo executar pesquisas, agregações ou até mesmo ambas ao mesmo tempo para você.

Execução de consultas: Formula e envia consultas para o agente escolhido coleção ou coleções.

Roteamento multicoleta: O agente entende a estrutura de todos os seus coleções, para que possa decidir de forma inteligente quais coleções de dados consultar com base na pergunta do usuário.

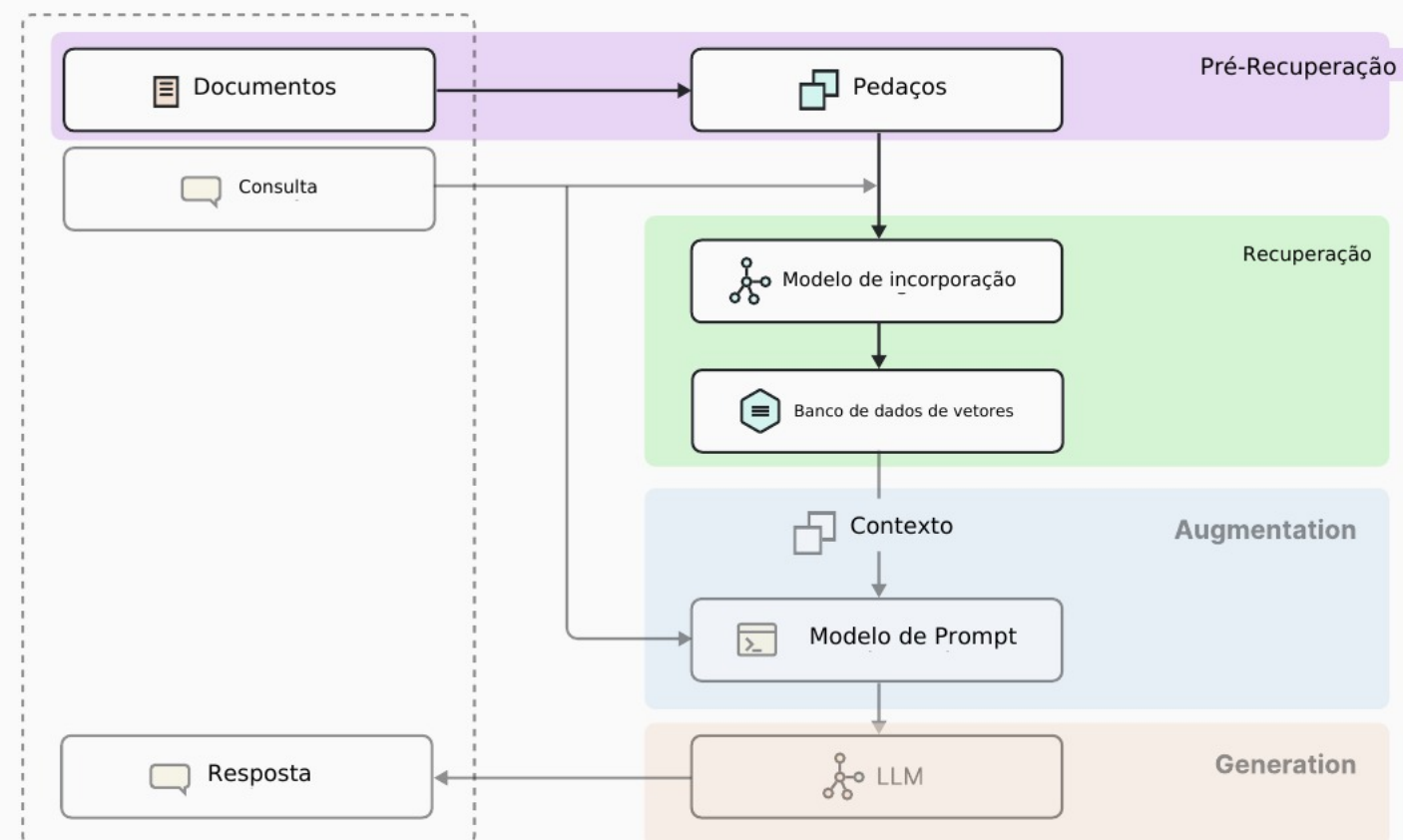
Avaliação: O agente pode avaliar as informações recuperadas dentro do contexto da consulta original do usuário. Se houver informações faltantes, o agente pode tentar uma fonte de conhecimento diferente ou uma nova consulta.

(Opcional) Geração de resposta: Receba os resultados do banco de dados e use um modelo generativo para gerar a resposta final ao prompt/consulta do usuário.

Consciência contextual: O contexto também pode incluir conversas anteriores histórico e qualquer outra informação relevante. O agente pode manter contexto de conversa para perguntas de acompanhamento.

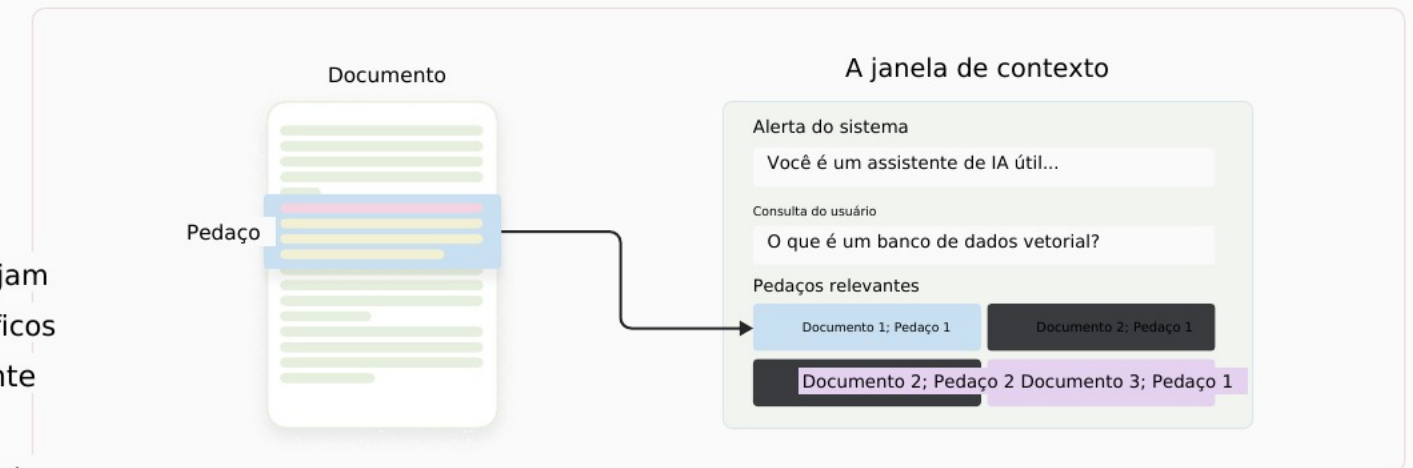
Recuperação

Um modelo de linguagem grande é tão bom quanto as informações que ele pode acessar. Embora os LLMs sejam treinados em conjuntos de dados massivos, eles não têm conhecimento de seus documentos privados específicos e qualquer informação criada após a conclusão do treinamento. Para construir verdadeiramente inteligente aplicativos, você precisa alimentá-los com as informações externas certas no momento certo. Isto processo é chamado de Recuperação. As etapas de pré-recuperação e recuperação constituem as primeiras partes do muitas arquiteturas de IA que dependem de engenharia de contexto, como Retrieval Augmented Generation (RAG).



O desafio é simples no conceito, mas complicado na prática: um conjunto de dados brutos de documentos é quase sempre grande demais para caber na janela de contexto limitada de um LLM (as entradas fornecidas a um modelo de IA). Não podemos simplesmente entregar ao modelo um conjunto completo de manuais de usuário ou documentos de pesquisa. Em vez disso, devemos encontrar a peça perfeita desses documentos, o parágrafo único ou seção que contém a resposta à consulta de um usuário.

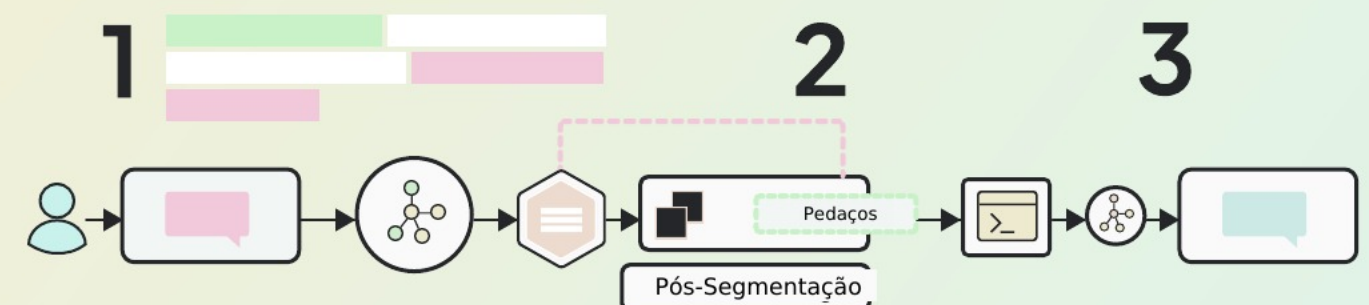
Para tornar nossas vastas bases de conhecimento pesquisáveis e encontrar a peça perfeita, devemos primeiro dividir nossos documentos em partes menores e gerenciáveis. Este processo fundamental, conhecido como segmentação, é a chave para uma recuperação bem-sucedida.



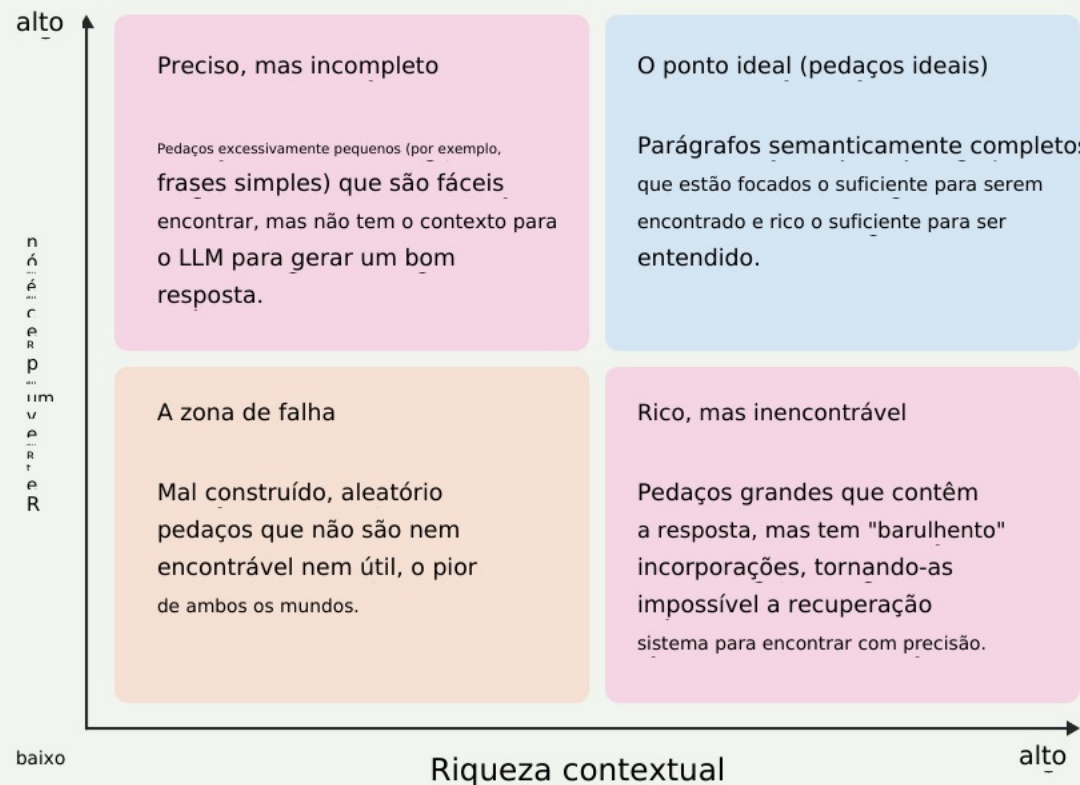
Aprenda como as estratégias de segmentação podem ajudar a melhorar o desempenho do seu RAG e explore diferentes segmentação métodos antes de comprometer seu pipeline com uma única abordagem.

Um guia para técnicas de segmentação

Segmentação é a decisão mais importante que você tomará para o seu sistema de recuperação desempenho. É o processo de dividir documentos grandes em documentos menores, peças gerenciáveis. Faça certo e seu sistema será capaz de identificar fatos relevantes com precisão cirúrgica. Se errar, até mesmo o LLM mais avançado irá falhar.



Matriz de Estratégias de Segmentação



Ao projetar sua estratégia de segmentação, você deve equilibrar duas prioridades concorrentes:

- **Precisão de recuperação:** os pedacos precisam ser pequenos e focados em uma única ideia. Isto cria uma incorporação distinta e precisa, tornando mais fácil para um sistema de pesquisa vetorial encontrar uma correspondência exata para a consulta de um usuário. Grandes pedacos que misturam vários tópicos criam incorporações ruidosas "médias" que são difíceis de recuperar com precisão.
- **Riqueza contextual:** os pedacos devem ser grandes e independentes o suficiente para serem entendido. Depois que um pedaco é recuperado, ele é passado para o LLM. Se o pedaco for apenas um sentença isolada sem contexto, mesmo um modelo poderoso terá dificuldade para gerar uma resposta significativa.

O objetivo é encontrar o "ponto ideal de segmentação", criando pedacos que sejam pequenos o suficiente para recuperação precisa, mas completa o suficiente para dar ao LLM o contexto completo de que necessita.

A sua escolha de estratégia dependerá da natureza dos seus documentos e das necessidades de sua aplicação.

Técnicas Simples de Segmentação

Segmentação de tamanho fixo: o método mais simples. O texto é dividido em pedacos de um tamanho predeterminado (por exemplo, 512 tokens). É rápido e fácil, mas pode cortar frases de maneira desajeitada pela metade. Usar uma sobreposição (por exemplo, 50 tokens) entre pedacos ajuda a mitigar isso.

A fotossíntese é um dos processos mais vitais da natureza.

pedaco 1

pedaco 2

pedaco 3

sobreposição

sobreposição

Segmentação recursivo: uma abordagem mais inteligente que divide o texto usando uma lista priorizada de separadores (como parágrafos, depois frases e depois palavras). Respeita o documento estrutura natural e é uma escolha padrão sólida para texto não estruturado.

O emaranhamento quântico é um conceito-chave na ciência quântica. física. Ocorre quando as partículas ficam ligadas, de modo que o estado de um afeta instantaneamente o estado de outro, não importa a distância entre eles

Esta conexão desafia nossa compreensão do espaço e tempo. Quando você mede uma partícula emaranhada, o estado do outro muda instantaneamente.

1 Defina uma hierarquia de separadores (por exemplo, parágrafos → frases → palavras)

2 Divida o texto usando o separador de nível mais alto

3 Se os pedacos forem muito grandes, divida novamente usando o próximo separador

4 Repita até que todos os pedacos caibam no tamanho desejado tamanho preservando o significado

Segmentação baseado em documento: este método usa a estrutura inerente do documento. Para por exemplo, ele divide um arquivo Markdown por seus títulos (#, ##), um arquivo HTML por suas tags (<p>, <div>) ou código-fonte por suas funções.

Título Este é um título. --## Subtítulo Este é um subtítulo. Podemos continuar com mais conteúdo aqui. --## Este é um segundo subtítulo. Aqui está um conteúdo diferente.

1 Identifique os limites lógicos do documento (por exemplo, capítulos, seções, títulos)

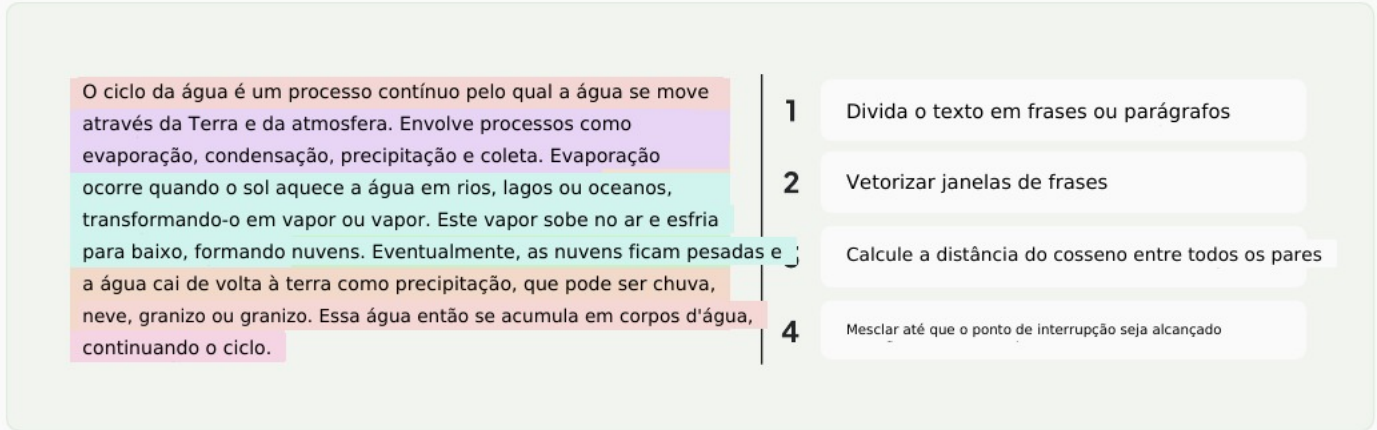
2 Agrupe o conteúdo sob cada limite em unidades coesivas

Vetorize cada unidade como um pedaco independente

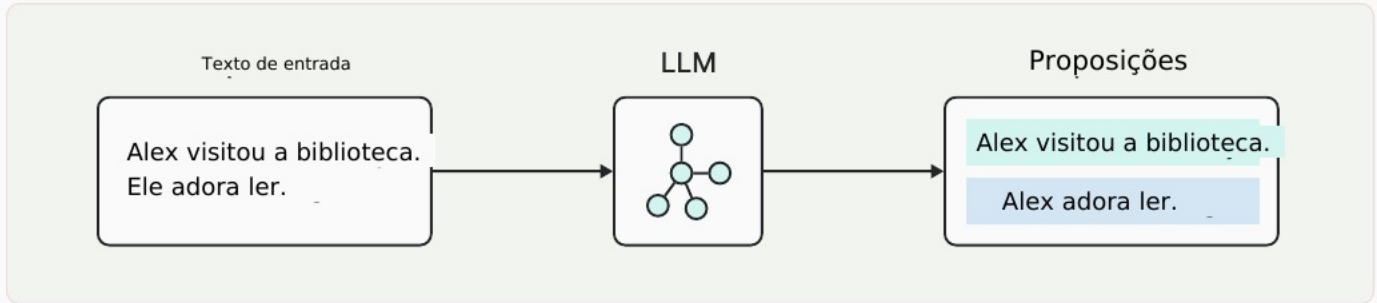
4 Armazene pedacos com metadados vinculando-os ao documento e seção de origem

Técnicas Avançadas de Segmentação

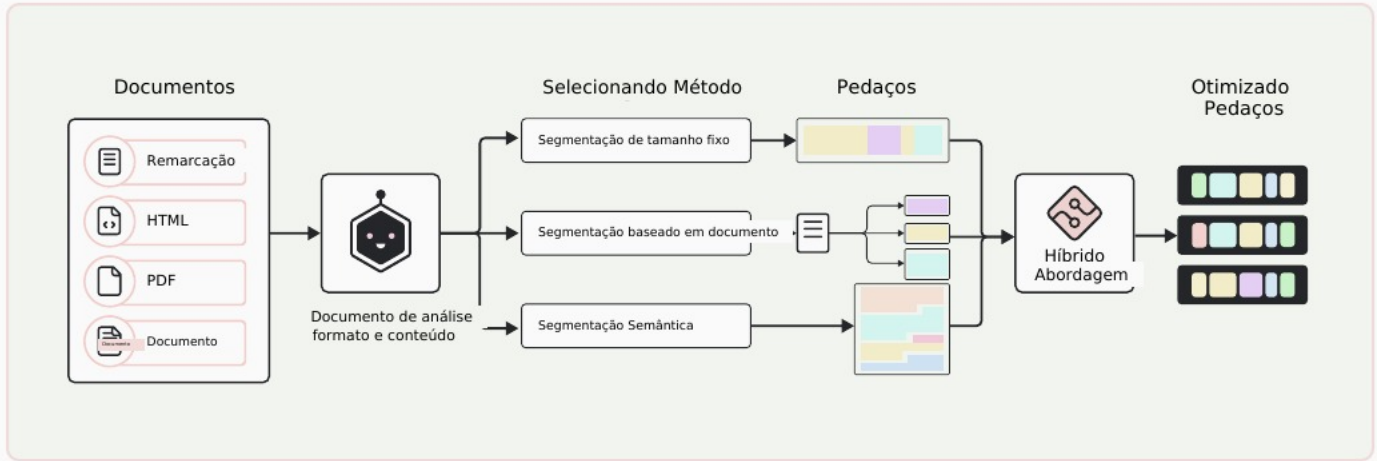
Segmentação semântico: em vez de usar separadores, esta técnica divide o texto com base em significado. Ele agrupa sentenças semanticamente relacionadas e cria um novo pedaço somente quando o tópico muda, resultando em pedaços altamente coerentes e independentes.



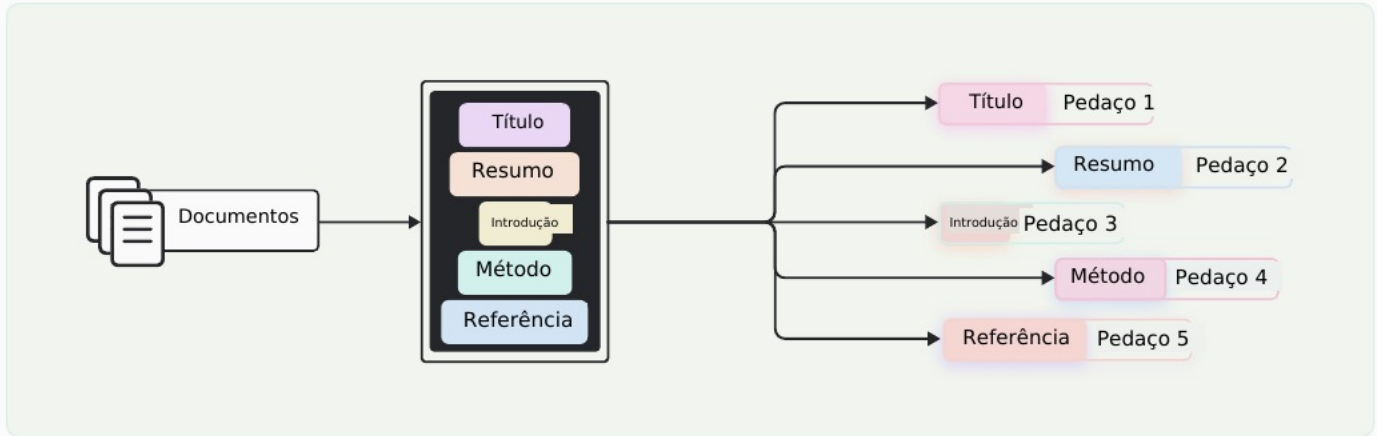
Segmentação baseado em LLM: usa um modelo de linguagem grande para processar um documento de forma inteligente e gerar pedaços semanticamente coerentes. Em vez de confiar em regras fixas, o LLM pode identificar proposições lógicas ou resumir seções para criar peças que preservem o significado.



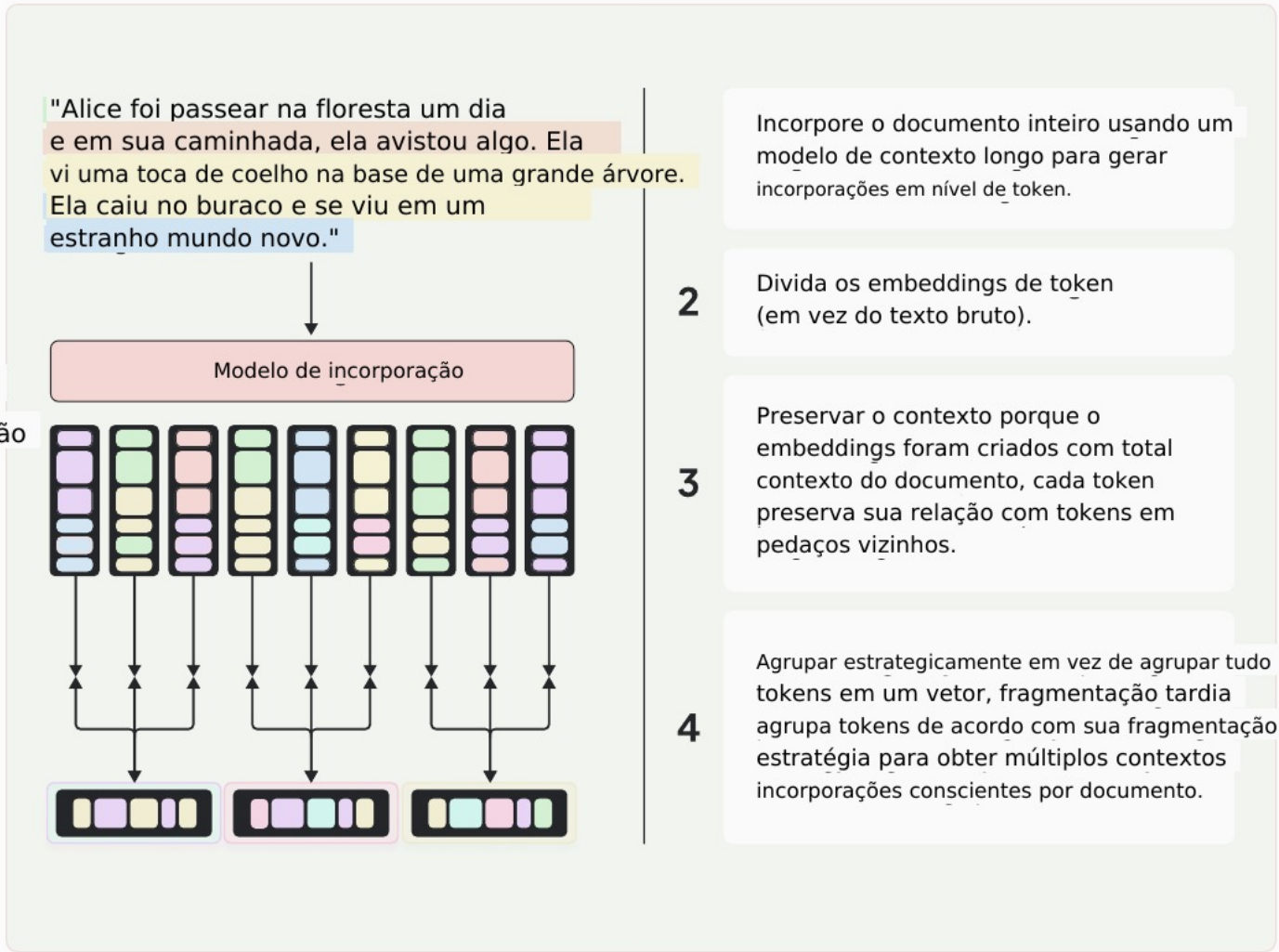
Agentic Segmentação: Isso leva o conceito um passo além do Segmentação baseado em LLM. Uma IA o agente analisa dinamicamente a estrutura e o conteúdo de um documento para selecionar a melhor fragmentação estratégia (ou combinação de estratégias) para se candidatar a esse documento específico.



Segmentação hierárquico: cria múltiplas camadas de pedaços em diferentes níveis de detalhe (por exemplo, resumos de nível superior, seções de nível médio e parágrafos granulares). Isto permite uma recuperação sistema para começar com uma visão geral ampla e, em seguida, detalhar os detalhes conforme necessário.



Late Segmentação: Um padrão arquitetônico que inverte o processo padrão. Ele incorpora todo o documento primeiro para criar embeddings em nível de token com contexto completo. Só então é o documento dividido em partes, com a incorporação de cada parte derivada desses pré-tokens computados e ricos em contexto.

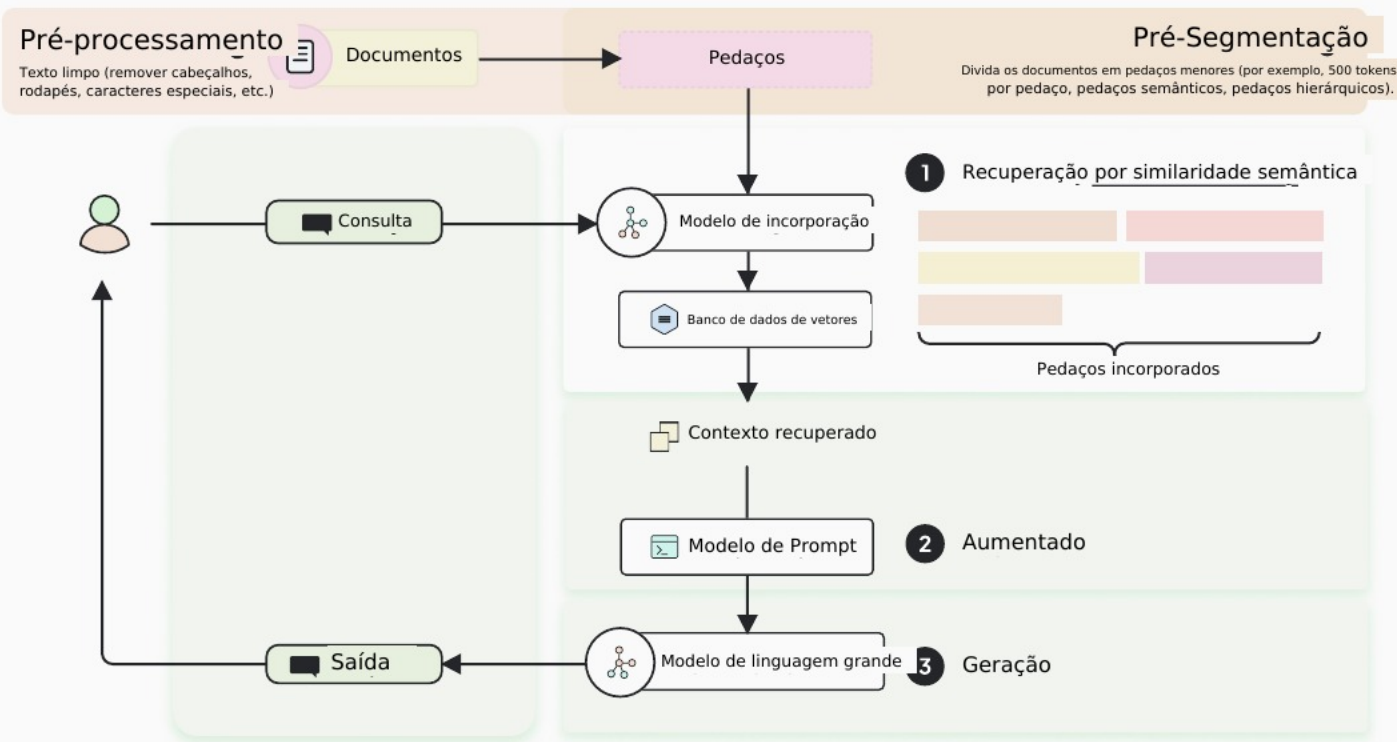


Pré-Segmentação vs. Pós-Segmentação

Além de como você fragmenta, uma escolha importante de design do sistema é quando você fragmenta. Esta decisão leva a dois padrões arquitetônicos primários.

Pré-Segmentação

O método mais comum, onde todo o processamento de dados acontece antecipadamente e offline, antes que qualquer dúvida do usuário chegue.



Fluxo de trabalho: Limpar dados -> Documentos de pedaços -> Incorporar e armazenar pedaços

PRÓ

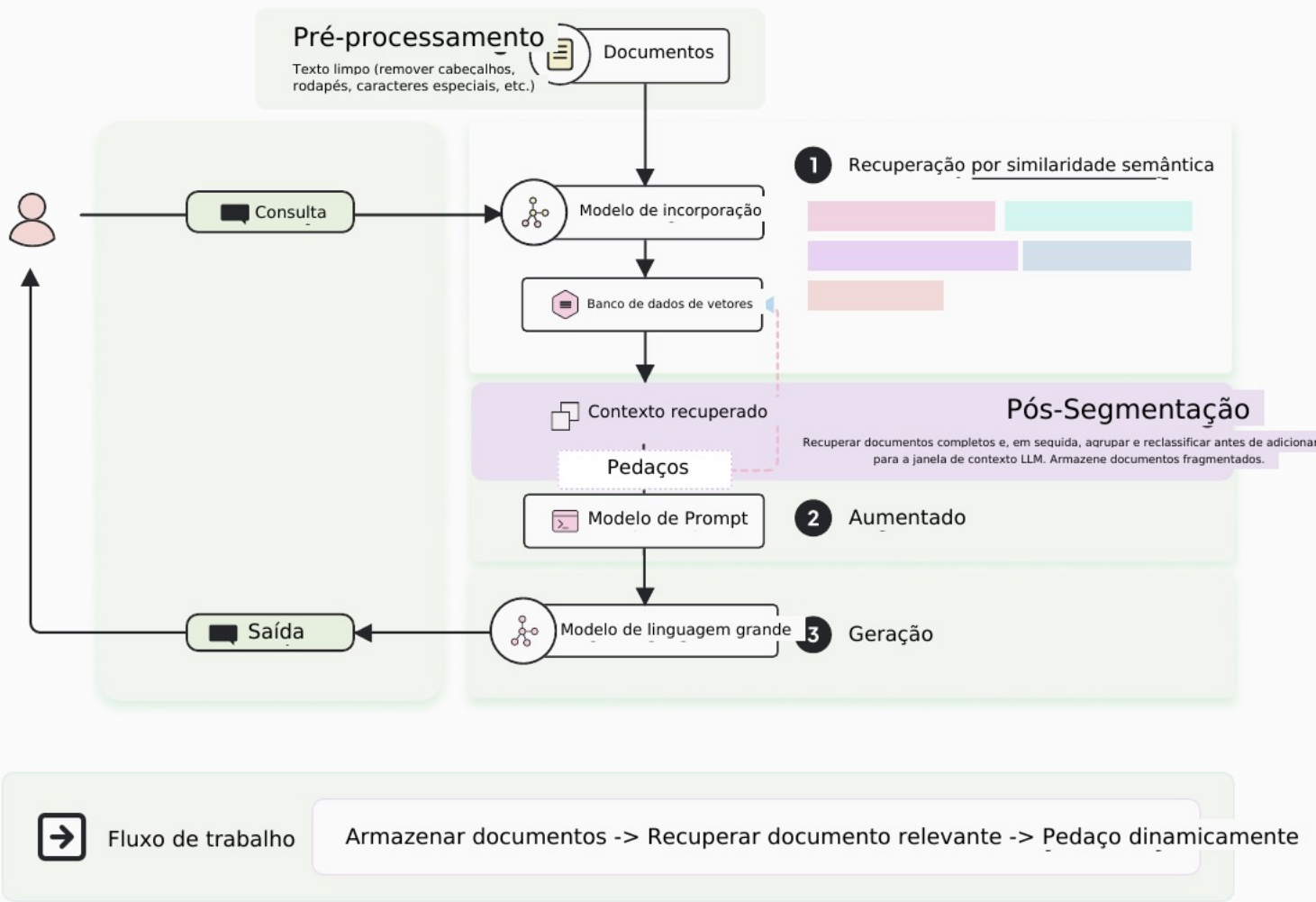
A recuperação é extremamente rápida na consulta tempo porque todo o trabalho já decidiu alterar o tamanho do seu pedaço ou feito. O sistema só precisa de método, você deve reprocessar seu realizar uma rápida pesquisa de similaridade.

CON

A estratégia de segmentação é fixa. Se você conjunto de dados inteiro.

Pós-Segmentação

Uma alternativa avançada e em tempo real onde a fragmentação ocorre após um documento ter sido recuperado, em resposta direta à consulta de um usuário.



Fluxo de trabalho: Armazenar documentos -> Recuperar documento relevante -> Pedaço dinamicamente

PRÓ

É altamente flexível. Você pode criar estratégias dinâmicas de segmentação que são processos que acontecem em tempo real, tornando específico para o contexto do usuário consulta, potencialmente levando a mais resultados relevantes.

CON

Adiciona latência. A fragmentação a primeira resposta mais lenta para o final usuário. Também requer soluções mais complexas infraestrutura para gerenciar.

Uma estratégia pós-segmentação pode ser construída diretamente em uma estrutura RAG de agência, de modo que os documentos são fragmentados sob demanda à medida que são recuperados.

Guia para Escolher sua Estratégia de Segmentação - Resumo

Segmentação Estratégia	Como funciona	Complexidade	melhor para	Exemplos
	Divisões de tamanho fixo por token ou contagem de caracteres.		Notas de reunião pequenas ou simples, documentos, ou quando acelerar postagens curtas no blog, é o que mais importa.	e-mails, perguntas frequentes simples.
Rekursivo	Divide o texto por dividindo repetidamente até que caiba no tamanho de pedaço desejado, muitas vezes preservando alguma estrutura.		Documentos onde artigos de pesquisa, alguns guias de produtos de estrutura, deveria ser mantido, mas a velocidade ainda é importante.	relatórios curtos.
Documento-Baseado	Divide apenas em documento limites ou por elementos estruturais como cabeçalhos.		Coleções de suporte ao cliente curto e independente documentos ou arquivos altamente estruturados.	Artigos de notícias, ingressos, Markdown arquivos.
Semântica	Divide o texto naturalmente limites de significado (tópicos, ideias).		Artigos técnicos, acadêmicos, científicos, ou narrativa documentos onde whitepapers. os tópicos mudam sem separadores claros.	livros didáticos, romances,
Baseado em LLM	Usa uma linguagem modelo para decidir limites do pedaço com base no contexto e significado.		Texto complexo onde relatórios longos, jurídicos opiniões conscientes do significado, médicos segmentação melhora os registros. tarefas posteriores	como perguntas e respostas.
Agente	Permite que um agente de IA decidir como dividir baseado no significado e estrutura.		Arquivos regulatórios complexos e diferenciados, documentos que têm várias seções exigir personalizado estratégias.	contratos, políticas corporativas.
Tarde Segmentação	Incorpora o todo documento primeiro, depois deriva pedaço incorporações dele.		Use casos em que estudos de caso, pedaços precisam conhecimento dos manuais completos, formato longo contexto do documento.	relatórios de análise.
Hierárquico	Divide o texto em vários níveis (seções → parágrafos → sentenças).		Funcionário grande e estruturado documentos onde manuais, resumo e governo detalhes são necessários.	regulamentos, programas documentação.

A eficácia do seu sistema Retrieval Augmentation não é determinada por um único bala "mágica", mas por uma série de escolhas deliberadas de engenharia. A qualidade do O contexto que você fornece a um LLM é resultado direto de duas decisões principais:

A estratégia de fragmentação

O "Como"

O método que você escolhe para quebrar anotar seus documentos.

O padrão arquitetônico

O "Quando"

O ponto em que você atua o fragmento.

Dominar esses dois elementos é fundamental para a engenharia de contexto. Um bem desenhado sistema de recuperação é a diferença entre um LLM que adivinha e aquele que fornece respostas baseadas em fatos, confiáveis e contextualmente relevantes.

Técnicas de Engenharia de Prompts

Engenharia de prompts é a prática de projetar, refinar e otimizar entradas (prompts) fornecido a modelos de linguagem de grande porte (LLMs) para obter o resultado desejado. A qualidade e a eficácia dos LLMs é fortemente influenciada pelas solicitações que recebem e pela forma como você formula um prompt pode afetar diretamente a precisão, a utilidade e a clareza do resposta.

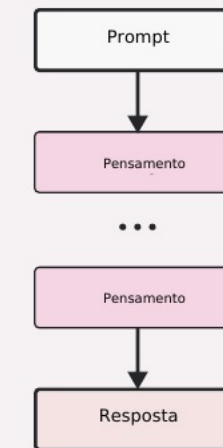
Trata-se essencialmente de interagir com a IA de forma eficiente: dar-lhe instruções, exemplos ou perguntas que orientam o modelo em direção ao resultado que você precisa.

Nesta seção, abordaremos técnicas de estímulo que são essenciais para melhorar Aplicativos de geração aumentada de recuperação (RAG) e desempenho geral do LLM.



Nota importante: a engenharia de prompts se concentra em como você formula instruções para o LLM. Engenharia de contexto, por outro lado, trata-se de estruturar a informação e o conhecimento que você fornecer ao modelo, como documentos recuperados, usuário histórico, ou dados específicos do domínio, para maximizar o modelo compreensão e relevância. Muitas das técnicas abaixo (CoT, Few-shot, ToT, ReAct) são mais eficazes quando combinados com contexto bem projetado.

Técnicas Clássicas de Engenharia de Prompts

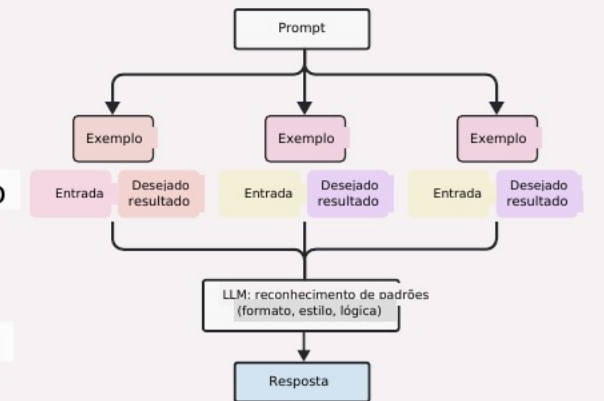


Cadeia de Pensamento

Esta técnica envolve perguntar ao modelo "pensar passo a passo" e decompor raciocínio complexo em intermediário passos. Isto é especialmente útil quando os documentos recuperados são densos ou contêm informações conflitantes que exigem análise cuidadosa. Ao verbalizar seu processo de raciocínio, o LLM pode chegar em conclusões mais precisas e lógicas.

Prompting com Poucos Exemplos

Esta abordagem fornece ao LLM alguns exemplos na janela de contexto que demonstrar o tipo de saída ou respostas "de ouro" que você deseja. Mostrando exemplos ajudam o modelo a entender o formato, estilo ou raciocínio desejado. Esta abordagem, melhorando a precisão da resposta e relevância, especialmente para especialistas ou domínios técnicos.



Combinar exemplos de CoT e Few-shot é uma maneira poderosa de orientar tanto o modelo processo de raciocínio e seu formato de saída para máxima eficiência.



Dica profissional nº 1:

Faça o raciocínio do modelo em Cadeia de Achei muito específico para o seu caso de uso. Por exemplo, você pode pedir ao modelo para:

- E valorizar o meio ambiente
- Repita qualquer informação relevante
- Explique a importância desta informação para a solicitação atual



Dica profissional nº 2:

Maximize a eficiência e reduza contagem de tokens, pedindo ao modelo para raciocinar em forma de "rascunho", sem usar mais de 5 palavras por frase.

Isso garante que o modelo o processo de pensamento é visível enquanto reduzindo a contagem de tokens de saída.

Estratégias Avançadas de Engenharia de Prompts

Com base em técnicas clássicas, estratégias avançadas orientam os LLMs de maneiras mais sofisticadas:

Esta orientação muito precisa, que deve ser incluída como parte da sua ferramenta geral de descrição, ajuda o LLM a compreender os limites e funcionalidades exatos de cada ferramenta disponível, minimizando erros e melhorando a confiabilidade geral do sistema.

Árvore de Pensamentos (ToT):

ToT baseia-se no CoT instruindo o modelo para explorar e avaliar raciocínios múltiplos caminhos em paralelo, como uma árvore de decisão. O modelo pode gerar várias soluções para um problema e escolher as melhores ferramentas ou fontes de dados e ajustar seu raciocínio resultado. Isto é especialmente útil em RAG quando há muitas peças potenciais de evidências, e o modelo precisa pesar diferentes respostas possíveis com base em vários documentos recuperados.

Técnica ReAct:

Esta estrutura combina CoT com agentes, permitindo que o modelo "raciocine" (pense) e "Agir" dinamicamente. O modelo gera ambos traços de raciocínio e ações em um intercalado forma, permitindo-lhe interagir com iterativamente. O ReAct pode melhorar os pipelines RAG ao permitindo que LLMs interajam com os recuperados documentos em tempo real, atualizando o raciocínio e ações baseadas em conhecimento externo para dar respostas mais precisas e relevantes.

Dica profissional: como escrever uma descrição de ferramenta eficaz

A decisão do LLM de usar sua ferramenta depende inteiramente de sua descrição. Faça valer a pena:

- Use um verbo ativo: comece com uma ação clara. `get_current_weather` é melhor que `weather_data`.
- Seja específico sobre as entradas: indique claramente quais argumentos a ferramenta espera e seu formato (por exemplo, cidade (string), data (string, AAAA-MM-DD)).
- Descreva o resultado: diga ao modelo o que esperar return (por exemplo, retorna um objeto JSON com "alto", "baixo", e "condições").
- Mencione Limitações: Se a ferramenta funcionar apenas para um determinado região ou período de tempo, diga-o (por exemplo, Observação: funciona apenas para cidades nos EUA.).

Engenharia de Prompts para Uso de Ferramentas

Quando seu LLM interage com ferramentas externas, uma solicitação clara garante a ferramenta correta seleção e uso.

Definindo Parâmetros e Condições de Execução

Às vezes, os LLMs podem fazer seleções incorretas de ferramentas ou usar ferramentas de maneiras abaixo do ideal. Para evitar isso, solicitação deve definir claramente:

Quando usar uma ferramenta:
Especifique cenários ou condições que acionam entradas, parâmetros e Torre." -> Usar uma ferramenta específica.

Como usar uma ferramenta:
Fornecer o esperado saídas desejadas.

Exemplos: inclua exemplos de poucas cenas mostrando a seleção correta da ferramenta e uso para várias consultas. Por exemplo:

Consulta do usuário:
"Como está o tempo em Paris?" -> Use Weather_API com cidade = "Paris"

Consulta do usuário:
"Encontre-me um restaurante perto da Eiffel" -> Usar Restaurant_Search_Tool com localização = "Torre Eiffel"

Uso de Frameworks de Prompts

Se você estiver construindo um projeto que exija muita orientação ou queira sistematicamente melhorar seus resultados de LLM, você pode considerar o uso de estruturas como: DSPy, Llama Operações de alerta, Synalinks.

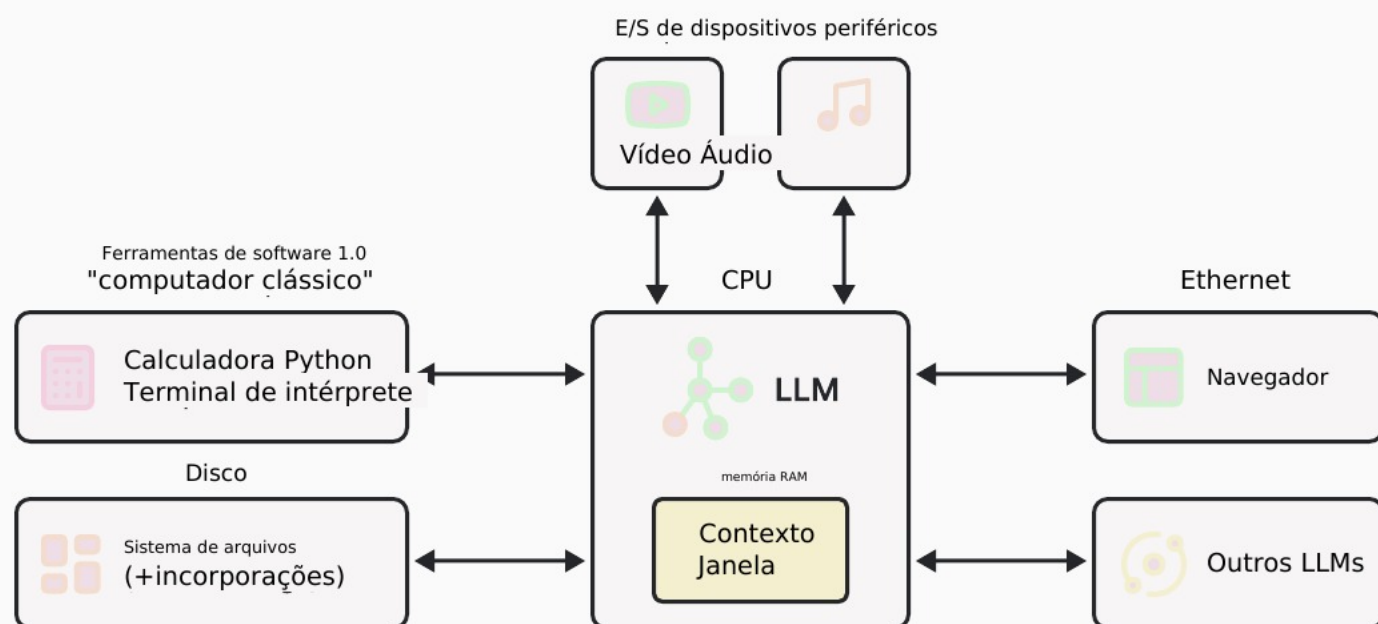
Dito isto, você não precisa necessariamente usar uma estrutura. Seguindo a sugestão de diretrizes descritas (instruções claras, cadeia de pensamento, aprendizado rápido e estratégias avançadas) podem alcançar resultados altamente eficazes sem estruturas adicionais.

Pense nessas estruturas como auxiliares opcionais para projetos complexos, não como um requisito para a engenharia de prompts do dia a dia.

Memória

Quando você está construindo agentes, a memória não é apenas um recurso bônus - é exatamente o que dá vida a eles. Sem ele, um LLM é apenas um processador de texto poderoso, mas sem estado que responde a uma consulta por vez, sem nenhum senso de histórico. A memória transforma estes modelos em algo que parece mais dinâmico e, ousamos dizer, mais 'humano', isso é capaz de se manter no contexto, aprender com o passado e se adaptar rapidamente.

Andrej Karpathy nos deu a analogia perfeita ao comparar o contexto de um LLM janela para a RAM de um computador e o próprio modelo para a CPU. Nessa visão, o contexto A janela é a consciência ativa do agente, onde estão mantidos todos os seus "pensamentos de trabalho". Mas, assim como um laptop com muitas abas do navegador abertas, essa RAM pode ficar cheia rapidamente. Cada mensagem, cada saída de ferramenta, cada informação consome tokens preciosos.



Fonte: Andrej Karpathy: O software está mudando (de novo)

É aqui que a engenharia de contexto se torna uma arte. O objetivo não é inserir mais dados o prompt, mas projetar sistemas que aproveitem ao máximo a janela de contexto ativa - mantendo as informações essenciais ao seu alcance e, ao mesmo tempo, descarregando todo o resto em um armazenamento mais inteligente e persistente.

Context Offloading é a prática de armazenar informações fora da janela de contexto ativa do LLM, frequentemente em ferramentas externas ou bancos de dados vetoriais. Isso libera o espaço limitado de tokens para que apenas as mais informações relevantes permaneçam no contexto.

A Arquitetura da Memória do Agente

A memória em um agente de IA tem tudo a ver com a retenção de informações para navegar pelas tarefas em constante mudança, lembre-se do que funcionou (ou não) e pense no futuro. Para construir agentes robustos, precisamos pensar em camadas, muitas vezes combinando diferentes tipos de memória para obter os melhores resultados.

Memória de curto prazo

Janela de Contexto

Usuário: "Qual é o tempo?"

AI: "Está ensolarado, 24°C"

Usuário: "Devo trazer uma jaqueta?"

AI: "Não precisa, está quente!"

A memória de curto prazo é a memória do agente espaço de trabalho imediato. É o "agora" colocado na janela de contexto para abastecer decisões e raciocínios imediatos. Isto é alimentado por aprendizagem em contexto, onde você reúne conversas, ações recentes, ou dados diretamente no prompt.

Porque é limitado pelo modelo limite de token, o principal desafio é eficiência. E o truque é manter isso simplificado para reduzir custos e latência sem perder nenhum detalhe que possa ser selecionado na base de conhecimento do domínio, permitindo que o importante para as próximas etapas de processamento.

Memória de longo prazo



A memória de longo prazo ultrapassa o imediato janela de contexto, armazenando informações externamente para recuperação rápida quando necessário. Isto é o que permite que um agente construa um persistente compreensão de seu mundo e de seus usuários ao longo tempo. Geralmente é alimentado por Retrieval-Geração Aumentada (RAG), onde o agente consulta uma base de conhecimento externa (como um banco de dados de vetores) para obter informações relevantes.

Esta memória pode armazenar diferentes tipos de informações, como por exemplo: memória episódica para armazenar eventos específicos ou interações passadas, ou memória semântica que contém conhecimento geral e fatos. Isso também pode ser informação de documentos da empresa, manuais de produtos ou um agente para responder perguntas com precisão factual.

Configuração de memória híbrida

Na realidade, a maioria dos sistemas modernos utiliza uma abordagem híbrida, combinando memória de curto prazo por velocidade com memória de longo prazo para profundidade. Algumas arquiteturas avançadas até introduzem camadas adicionais:

- **Memória de Trabalho:** Uma área de retenção temporária de informações relacionadas a um assunto específico, tarefa de várias etapas. Por exemplo, se um agente estiver reservando uma viagem, sua memória de trabalho poderá guardar o destino, as datas e o orçamento até que a tarefa seja concluída, sem desorganizar a loja de longo prazo.
- **Memória processual:** ajuda um agente a aprender e dominar rotinas. Ao observar fluxos de trabalho bem-sucedidos, o agente pode internalizar uma sequência de etapas para um processo recorrente tarefa, tornando-a mais rápida e confiável ao longo do tempo.



Princípios-chave para uma memória eficaz

Gestão

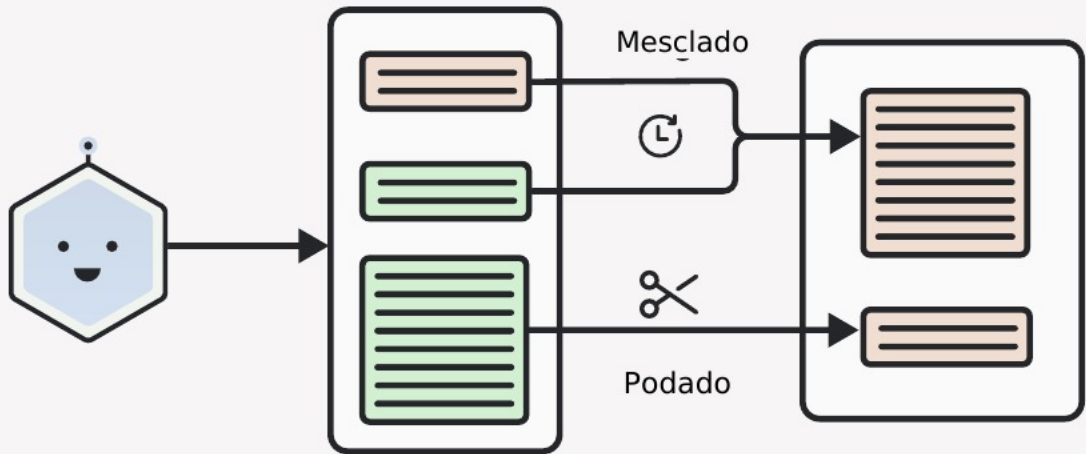
O gerenciamento eficaz de memória pode fazer ou quebrar um agente LLM. Práticas de memória ruins levam à propagação de erros, onde informações ruins são recuperadas e amplificam os erros em tarefas futuras.

Aqui estão alguns dos princípios iniciais para acertar as coisas:

Podar e refinar suas memórias

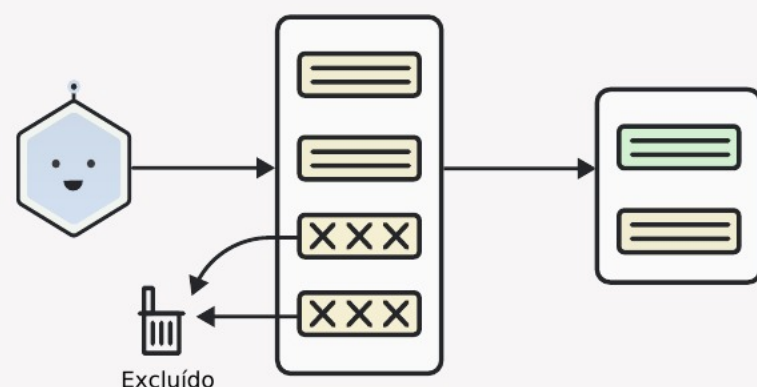
A memória não é um sistema de gravação única. Precisa de manutenção regular. Periodicamente verifique seu armazenamento de longo prazo para remover entradas duplicadas, mesclar relacionadas informações ou descartar fatos desatualizados. Uma métrica simples para isso poderia ser a recência e frequência de recuperação. Se uma memória for antiga e raramente acessada, ela pode ser um candidato à exclusão, especialmente em ambientes em evolução onde informações antigas podem se tornar um passivo.

Por exemplo, um agente de suporte ao cliente pode eliminar automaticamente registros de conversas com mais de 90 dias e marcados como resolvidos, fechados, ou não está mais ativo na memória. Poderia apenas reter os resumos (para tendências detecção e análise) em vez de transcrições completas palavra por palavra.



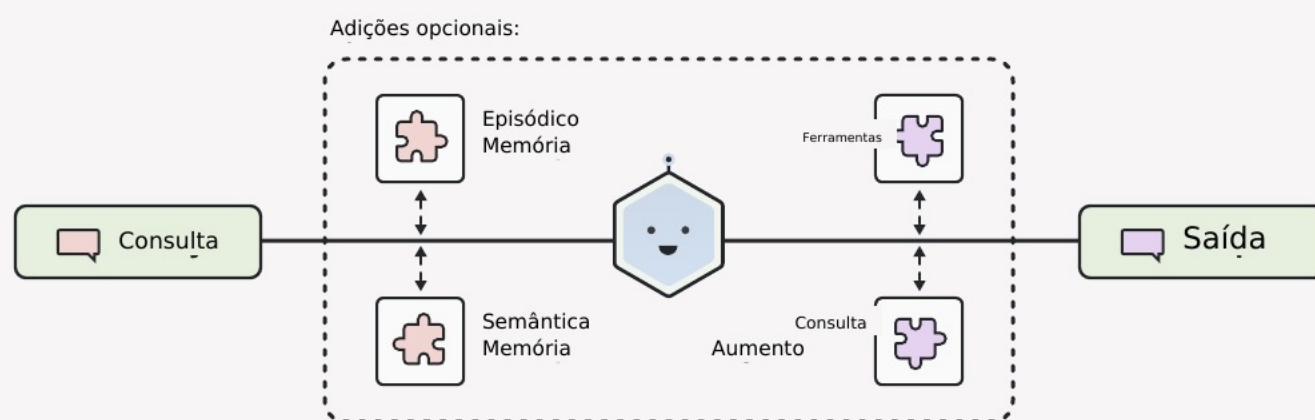
Seja seletivo sobre o que você armazena

Nem toda interação merece um lugar permanente no armazenamento de longo prazo. É preciso implementar algum tipo de critério de filtragem para avaliar a qualidade e relevância antes de salvá-lo. Uma informação incorreta recuperada muitas vezes pode levar a poluição de contexto, onde o agente comete repetidamente os mesmos erros. Um maneira de evitar isso é fazer com que o LLM "refletir" sobre uma interação e atribuir um pontuação de importância antes de memorizá-la.



Adapte a arquitetura à tarefa

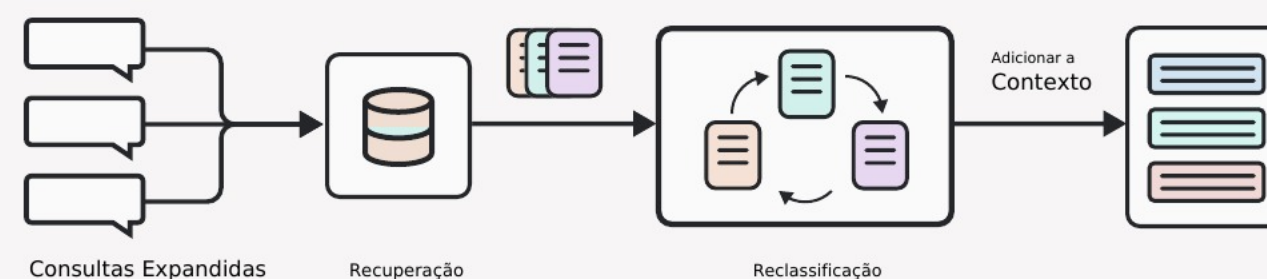
Não existe uma solução de memória que sirva para todos. Um bot de atendimento ao cliente precisa de um forte memória episódica para relembrar o histórico do usuário, enquanto um agente que analisa relatórios financeiros precisam de uma memória semântica robusta repleta de informações específicas de domínio conhecimento. Sempre comece com a abordagem mais simples que funciona (como uma abordagem básica buffer de conversação com as últimas 'n' consultas/respostas) e gradualmente adicionar mais mecanismos avançados conforme o caso de uso exigir.



Domine a arte da recuperação

A memória eficaz tem menos a ver com quanto você pode armazenar e mais com quão bem você pode recuperar a informação certa no momento certo. Uma simples cega a pesquisa muitas vezes não é suficiente, então técnicas avançadas como reclassificação (usando um LLM para reordenar os resultados recuperados por relevância) e recuperação iterativa (refinamento/expandir uma consulta de pesquisa em várias etapas) pode ser usado para melhorar a qualidade de informações recuperadas.

Ferramentas como o Agente de Consulta e o Agente de Personalização oferecem esses recursos pronto para uso, permitindo pesquisas em diversas coleções e reclassificação com base em preferências do usuário e histórico de interação.



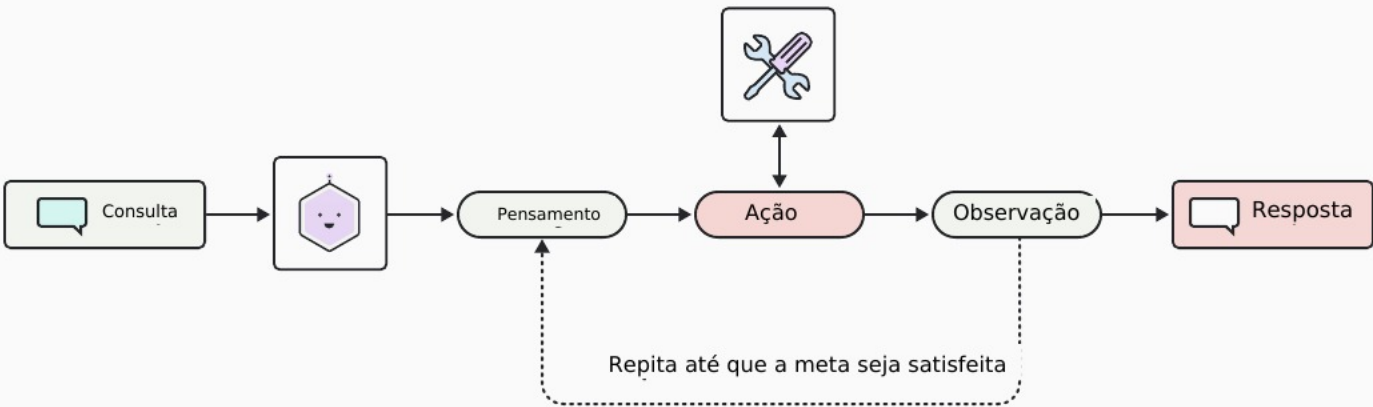
Em última análise, a memória é o que eleva os agentes LLM de simples respondedores a inteligentes sistemas sensíveis ao contexto. A memória eficaz não é simplesmente um armazenamento passivo... É um armazenamento ativo, processo gerenciado! O objetivo é construir agentes que não apenas armazenem memória, mas pode gerenciá-lo - sabendo o que lembrar, o que esquecer e como usar o passado para razão sobre o futuro.

Ferramentas

Se a memória dá ao agente um senso de identidade, então são as ferramentas que lhe conferem superpoderes. Por si só, os LLMs são conversadores e manipuladores de texto brilhantes, mas vivem dentro de uma bolha. Eles não podem verificar o clima atual, reservar um voo ou consultar em tempo real preços das ações. Eles estão, por definição, desconectados do mundo vivo e vibrante dos dados e ação.

É aqui que entram as ferramentas. Uma "ferramenta" é qualquer coisa que conecte um agente LLM ao mundo exterior, permitindo-lhe tomar "ações" diretas no mundo real e recuperar informações necessário para cumprir uma tarefa. A integração de ferramentas eleva um agente de apenas um consultor experiente para algo que pode realmente fazer as coisas.

A engenharia de contexto para ferramentas não envolve apenas fornecer ao agente uma lista de APIs e instruções. É sobre como criar um fluxo de trabalho coeso onde o agente possa entender quais ferramentas são disponíveis, decidir corretamente qual usar para uma tarefa específica e interpretar os resultados para seguir em frente.



A evolução: das instruções às ações

A jornada para o uso de ferramentas modernas tem sido uma evolução rápida. Inicialmente, os desenvolvedores tentaram obter ação dos LLMs com a boa e velha engenharia de prompts, enganando o modelo gerando texto que parecia um comando. Foi inteligente, mas sujeito a erros.

O verdadeiro avanço foi a chamada de função, também conhecida como chamada de ferramenta. Esta capacidade, agora nativa da maioria dos modelos, permite que um LLM produza JSON estruturado que pode conter o nome de uma função a ser chamada e os argumentos a serem usados.

Com isso, existem várias possibilidades:

 Uma ferramenta simples

Um bot de agente de viagens pode usar uma ferramenta `search_flights`, e quando um usuário pergunta: "Encontre um voo para mim Tóquio na próxima terça-feira", o LLM não adivinha a resposta. Isso gera uma chamada para a função que você forneceu, que por sua vez consulta uma API de companhia aérea real.

 Uma cadeia de ferramentas

Para uma solicitação complexa como "Planejar um viagem de fim de semana para São Francisco para mim", o agente pode precisar encadear várias ferramentas juntas: `find_flights`, `pesquisar_hotéis` e `get_local_events`. Isto requer o agente para raciocinar, planejar e executar um fluxo de trabalho de várias etapas.

O trabalho da engenharia de contexto aqui está na forma como você apresenta essas ferramentas. Um bem escrito A descrição da ferramenta é como um mini-prompt que orienta o modelo, deixando claro o que a ferramenta faz, quais entradas ela precisa e o que ela retorna.

O desafio da orquestração

Fornecer uma ferramenta a um agente é fácil (principalmente). Fazer com que ele use essa ferramenta de maneira confiável, segura e efetivamente é onde o verdadeiro trabalho começa. A tarefa central da engenharia de contexto é a orquestração, ou seja, gerenciar o fluxo de informações e a tomada de decisões como o agente razões sobre qual ferramenta usar.

Isso envolve algumas etapas principais que acontecem na janela de contexto. Vamos decompô-los principais etapas de orquestração usando um aplicativo de conhecimento de domínio de cuidados com a pele desenvolvido por uma estrutura de orquestração de agentes, como nosso exemplo em execução.

Descoberta de ferramentas: O agente precisa saber quais ferramentas tem à sua disposição. Isto é geralmente feito fornecendo uma lista de ferramentas disponíveis e suas descrições no sistema alerta. A qualidade dessas descrições é muito crítica. Eles são os únicos do agente guia para entender o que cada ferramenta faz, permitindo que o modelo entenda quando usar uma ferramenta e, mais importante, quando evitá-la.

```
[15:43:27] - AGENT INFO - User demo@example.com authenticated
INFO: ('127.0.0.1', 60060) - "WebSocket /ws/chat" [accepted]
INFO: connection open
[15:43:36] - AGENT INFO - Sending response to frontend: status
[15:43:36] - AGENT INFO - Step 1: Initializing tree for user ed06db1b-719a-5635-8a87-cf3e6339484c with chat_id db6b7a70-ea8d-4407-a9f9-9ffe8bab8df4
[15:43:36] - AGENT INFO - Step 2: Configuring tree settings
[15:43:36] - AGENT INFO - Initialised tree with the following decision nodes:
- base: {}
[15:43:38] - AGENT INFO - Step 3: Fetching current product and stack
[15:43:38] - AGENT INFO - Step 4: Setting up metadata
[15:43:38] - AGENT INFO - Adding new agent metadata
[15:43:38] - AGENT INFO - Step 5: Configuring tools
[15:43:38] - AGENT INFO - Adding product agent tool
[15:43:38] - AGENT INFO - Adding similar products agent tool
[15:43:38] - AGENT INFO - Adding stack agent tool
[15:43:38] - AGENT INFO - Configured tools: ['forced_text_response', 'text_response', 'cited_summarize', 'product_agent', 'similar_products_agent', 'stack_agent']
[15:43:38] - AGENT INFO - Step 6: Starting conversation loop
INFO: Style: Skincare professional and expert recommender who is friendly, empathetic, and patient
INFO: Agent description:
  1. Identity & Context
  You are a warm, professional virtual skincare expert focused on Korean-beauty (K-beauty) routines.
  The app is powered by an agentic decision-tree framework with chain-of-thought self-healing and a
  query agent for context-aware product and ingredient searches.
  2. Mission
  Empower every user to create safe, enjoyable, and effective K-beauty "stacks" (ordered routines) matched to
  their skin type, concerns, sensitivities, and goals.
```

Neste aplicativo, configuramos um conjunto de ferramentas especializadas (Etapas 5) com descrições precisas na inicialização cada nova árvore de bate-papo.

Seleção e planejamento de ferramentas (pensamento): Ao se deparar com uma solicitação do usuário, o agente deve raciocinar sobre se uma ferramenta é necessária. Se sim, qual? Para tarefas complexas, pode até precisar encadear várias ferramentas, formando um plano (por exemplo, "Primeiro, pesquise a web para o clima; em seguida, use a ferramenta de e-mail para enviar um resumo").

```

User prompt
what's the best cleanser for my skintype?

[15:43:48] - AGENT INFO - Unknown result type: tree_update
[15:43:48] - AGENT INFO - Sending response to frontend: status
Assistant response
I'm using your skin profile to find the best cleanser for your dry skin, redness, and wrinkles, while also helping with hydration and glow.

[15:43:48] - AGENT INFO - Sending response to frontend: text
[09/23/25 15:43:48] INFO Time taken for Decision Node (decision_node): 4.59 seconds objects.py:127
Current Decision
Node: root
Decision: product_agent
Reasoning: The user is asking for a product recommendation (specifically, a cleanser).
According to the instruction, I should use the 'product_agent' tool for product recommendations.
I should also use the user's skin profile when making recommendations.
The user's skin type is 22.0 (dry), they have redness and wrinkles as concerns, and they want to hydrate and glow up.
Therefore, I should search for a cleanser that is suitable for dry skin, addresses redness and wrinkles, and helps with hydration and glow.
```

Aqui, o agente de decisão analisou corretamente a solicitação recebida e selecionou a ferramenta product_agent.

Formulação do Argumento (Ação): Uma vez selecionada uma ferramenta, o agente deve descobrir quais argumentos passar para ele. Se a ferramenta for get_weather(city, date), o agente precisa extraia "São Francisco" e "amanhã" da consulta do usuário e formatá-los corretamente. Também pode ser uma solicitação estruturada ou uma chamada de API com as informações necessárias para usar a ferramenta.

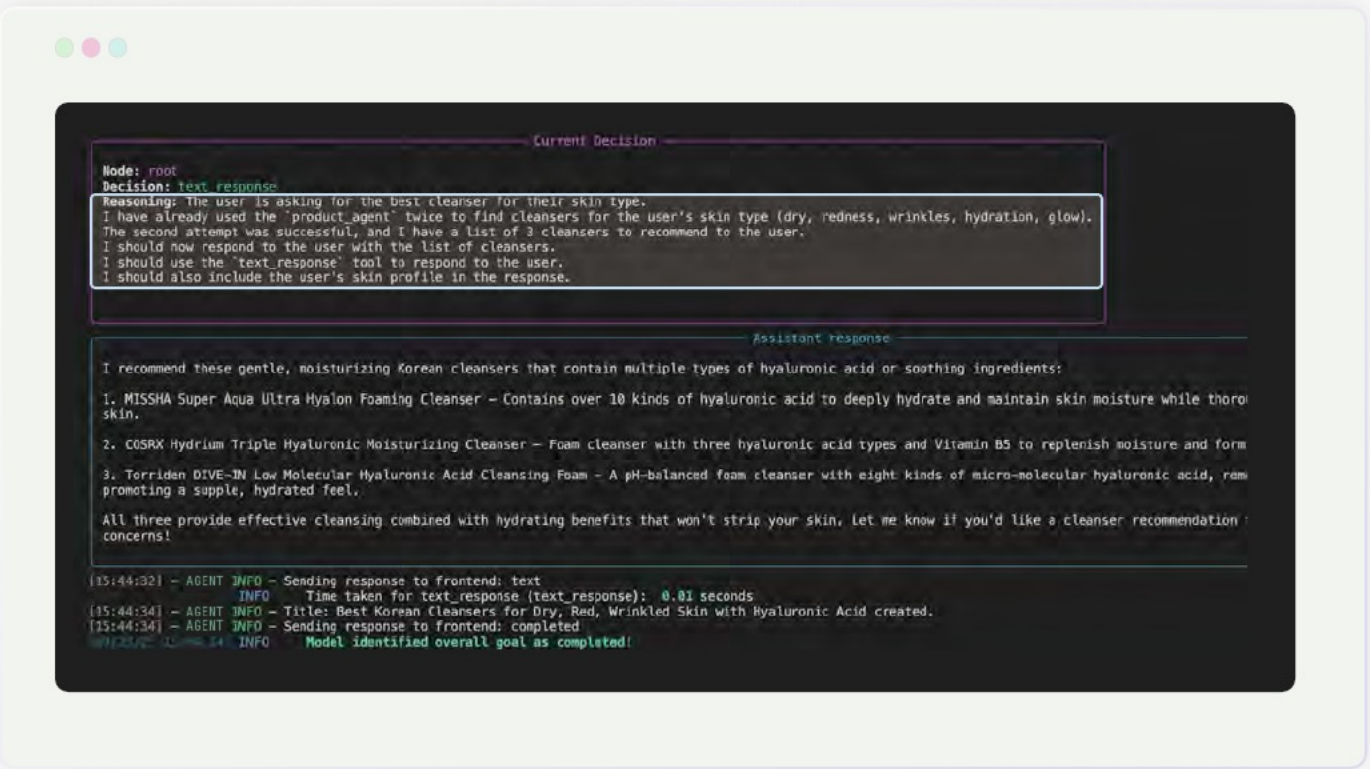
```
Current Decision
Node: root
Decision: product_agent
Reasoning: The user is asking for a product recommendation (specifically, a cleanser).
According to the instruction, I should use the 'product_agent' tool for product recommendations.
I should also use the user's skin profile when making recommendations.
The user's skin type is 22.0 (dry), they have redness and wrinkles as concerns, and they want to hydrate and glow up.
I previously attempted to use the 'product_agent' tool, but it resulted in an error.
I will try again, but this time I will simplify the search query to avoid the error.
I will search for a cleanser that is suitable for dry skin and helps with hydration.

Original Query
cleanser for dry skin and hydration

QueryResultWithCollection{
  queries=['cleanser gentle hydrating moisturizing dry skin'],
  filters=[[TextPropertyFilter(property_name='category', operator=<<ComparisonOperator.EQUALS: '='>, value='Face Washes'))]],
  filter_operators='AND',
  collection='Products_v2'
}
```

Nesse caso, o product_agent exigia uma consulta de texto para pesquisar a coleção de produtos. Aviso como o agente também se corrigiu (autocura) após gerar um argumento mal formado que inicialmente causou um erro (outra peça chave da orquestração).

Reflexão (Observação): Após a execução da ferramenta, a saída (a "observação") é realimentado na janela de contexto. O agente então reflete sobre esse resultado para decidir seu próximo passo. A ferramenta foi bem-sucedida? Produziu as informações necessárias para responder a consulta do usuário? Ou retornou um erro que requer uma abordagem diferente?



Como você pode ver, a orquestração acontece através deste poderoso ciclo de feedback, muitas vezes chamado o ciclo Pensamento-Ação-Observação. O agente observa o resultado de sua ação e usa essas novas informações para alimentar seu próximo "pensamento", decidindo se a tarefa é completa, se necessita utilizar outra ferramenta, ou se deve solicitar esclarecimentos ao usuário.

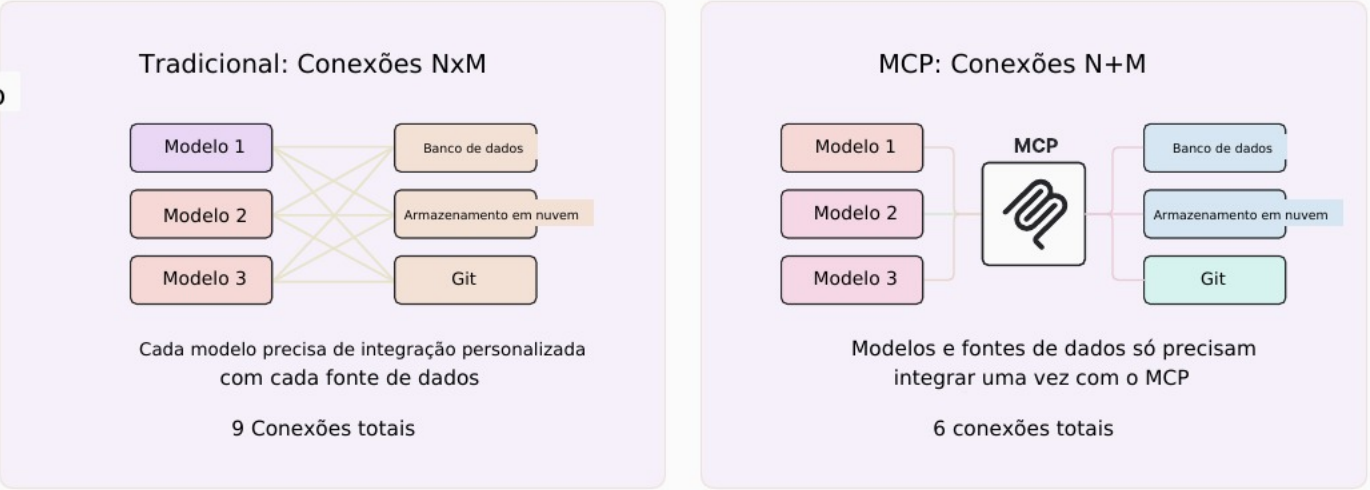
Este ciclo de Pensamento-Ação-Observação forma o ciclo de raciocínio fundamental na moderna estruturas.

A próxima fronteira do uso de ferramentas

A evolução do uso de ferramentas caminha cada vez mais para a padronização. Enquanto a chamada de função/ferramenta funciona bem, cria um ecossistema fragmentado onde cada IA o aplicativo precisa de integrações personalizadas com todos os sistemas externos. O Contexto do Modelo O Protocolo (MCP), introduzido pela Anthropic no final de 2024, aborda isso fornecendo um padrão universal para conectar aplicativos de IA a fontes e ferramentas de dados externas. Eles o chamam de "USB-C para IA" - um protocolo único que qualquer aplicativo de IA compatível com MCP pode use para conectar-se a qualquer servidor MCP.

Assim, em vez de criar integrações personalizadas para cada ferramenta, os desenvolvedores podem simplesmente criar servidores MCP individuais que expõem seus sistemas por meio dessa interface padronizada. Qualquer aplicativo de IA que suporte MCP pode se conectar facilmente a esses servidores usando o Protocolo baseado em JSON-RPC para comunicação cliente-servidor. Isso transforma o $M \times N$ problema de integração (onde M aplicativos precisam de código personalizado para N ferramentas) em um problema $M + N$ muito mais simples.

Integração Tradicional vs Abordagem MCP



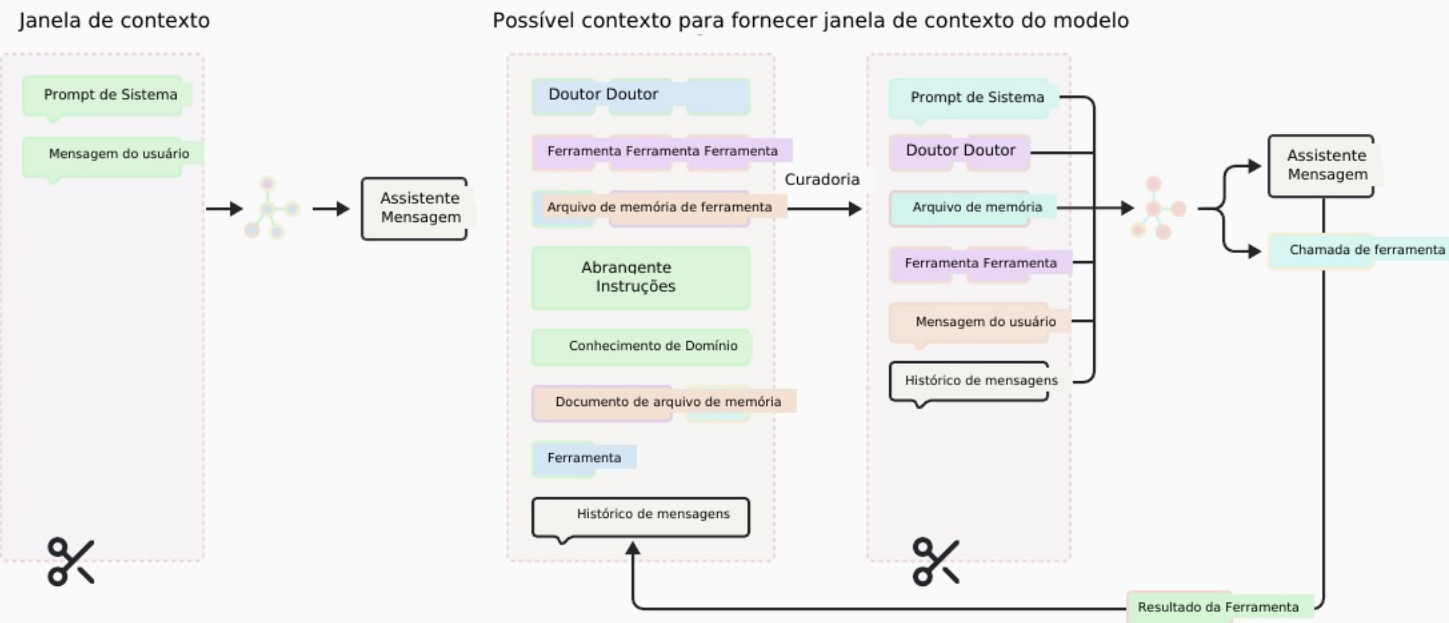
Visual inspirado em: <https://humanloop.com/blog/mcp>

Esta mudança em direção a arquiteturas padronizadas e combináveis, onde as estruturas permitem desenvolvedores para construir agentes a partir de componentes modulares e interoperáveis, representa o futuro das ferramentas de IA. Ele muda a função do engenheiro de escrever integrações personalizadas para orquestrar sistemas adaptativos que possam se conectar facilmente a qualquer ambiente externo padronizado. sistema.

Resumo

A engenharia de contexto envolve mais do que apenas gerar grandes modelos de linguagem, construir sistemas de recuperação ou projetar arquiteturas de IA. Trata-se de construir interligados, sistemas dinâmicos que funcionam de forma confiável em uma variedade de usos e usuários. Todos os componentes descritos neste e-book continuarão a evoluir à medida que novas técnicas, modelos, e descobertas são feitas, mas a diferença entre sistemas verdadeiramente funcionais e a IA aplicativos que falham será o quão bem eles projetam o contexto em toda a sua arquitetura. Nós não estamos mais pensando apenas em sugerir um modelo, estamos analisando como podemos arquitetar sistemas de contexto inteiros.

Engenharia de contexto de engenharia de prompts simples



Visual inspirado na engenharia de contexto eficaz para agentes de IA, Antrópico

A engenharia de contexto é composta pelos componentes descritos neste e-book:

- Agentes para atuar como o cérebro tomador de decisões do sistema.
- Aumento de consulta para traduzir solicitações humanas confusas em intenções acionáveis.
- Recuperação para conectar o modelo a fatos e bases de conhecimento.
- Memória para dar ao seu sistema uma noção de história e o poder de aprender.
- Ferramentas para permitir que seu aplicativo interaja com dados e APIs em tempo real.

Estamos deixando de ser instigadores que falam com um modelo e, em vez disso, nos tornando arquitetos que constroem o mundo em que o modelo vive. Nós - os construtores, os engenheiros e os criadores - saibam a verdade: os melhores sistemas de IA não nascem de modelos maiores, mas de uma melhor engenharia.

Mal podemos esperar para ver o que você construirá



Pronto para construir o próximo geração de aplicações de IA?

Projete o contexto, não apenas o prompt - é aí que de onde realmente vêm aplicativos de IA confiáveis.

