

UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

JOSUÉ COSTA DOS SANTOS ALVES

**RECOMENDAÇÃO CONVERSACIONAL DE PRODUTOS EM COMÉRCIO
ELETRÔNICO: UMA ABORDAGEM BASEADA EM AGENTE LLM E BUSCA
VETORIAL SOBRE REPRESENTAÇÃO TOKENIZADA**

Orientador: Prof. Dr. Carlos Maurício Serodio Figueiredo

Manaus – Junho – 2026

JOSUÉ COSTA DOS SANTOS ALVES

**RECOMENDAÇÃO CONVERSACIONAL DE PRODUTOS EM COMÉRCIO
ELETRÔNICO: UMA ABORDAGEM BASEADA EM AGENTE LLM E BUSCA
VETORIAL SOBRE REPRESENTAÇÃO TOKENIZADA**

Trabalho de Conclusão de Curso apresentado
à banca avaliadora do Curso de Engenharia
de Computação, da Escola Superior de
Tecnologia, da Universidade do Estado do
Amazonas, como pré-requisito para obtenção
do título de Engenheiro de Computação.

Orientador: Prof. Dr. Carlos Maurício Serodio Figueiredo

Manaus – Junho – 2026

Ficha catalográfica

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).
Sistema Integrado de Bibliotecas da Universidade do Estado do Amazonas.

A474r	Alves, Josué Costa dos Santos Recomendação conversacional de produtos em comércio eletrônico : uma abordagem baseada em agente LLM e busca vetorial sobre representação tokenizada / Josué Costa dos Santos Alves. Manaus : [s.n], 2026. 74 f.: color.; 21.0 cm. TCC - Graduação em Engenharia de Computação- Universidade do Estado do Amazonas, Manaus, 2026. Inclui Bibliografia. Inclui Apêndice. Orientador: Figueiredo, Carlos Maurício Serodio. 1. Agentes LLM. 2. Sistemas de recomendação conversacionais. 3. Busca vetorial. 4. Comércio eletrônico. I. Figueiredo, Carlos Maurício Serodio (Orient.) II. Universidade do Estado do Amazonas. III. Título CDU(1997)62:004
-------	--



FOLHA DE APROVAÇÃO

RECOMENDAÇÃO CONVERSACIONAL DE PRODUTOS EM COMÉRCIO ELETRÔNICO: UMA ABORDAGEM BASEADA EM AGENTE LLM E BUSCA VETORIAL SOBRE REPRESENTAÇÃO TOKENIZADA

Josué Costa dos Santos Alves

Trabalho de Conclusão de Curso defendido e aprovado pela banca avaliadora constituída pelos professores:

Carlos Maurício Serodio Figueiredo
Presidente

José Maria Drumond de Vasconcelos Dias Filho
Avaliador

Elloá Barreto Guedes da Costa
Avaliador

Manaus, 10 de junho de 2026.

Resumo

Sistemas tradicionais de busca em comércio eletrônico baseiam-se em filtros estruturados que exigem do usuário a tradução de suas necessidades em atributos categorizados, distanciando-se da forma natural como consumidores expressam seus desejos de consumo em linguagem livre. *Large Language Models* têm potencial para reduzir essa distância, mas seu uso isolado em cenários de recomendação enfrenta limitações relevantes, especialmente quanto ao acesso a informações específicas do domínio e atualizadas, e à tendência à alucinação. Este trabalho investiga como agentes baseados em LLM, dotados de ferramentas com acesso a informações externas, podem ser projetados para realizar recomendação conversacional de produtos a partir de solicitações expressas em linguagem natural. Foi desenvolvido um *chatbot* baseado em agente LLM que opera sobre uma representação tokenizada de produtos, realiza busca por similaridade vetorial em índice mantido na plataforma Databricks, e conduz elicitación de preferências para tratamento de solicitações incompletas. O escopo experimental foi delimitado a quatro categorias de produtos eletrônicos em catálogo de comércio eletrônico brasileiro, com solicitações conduzidas em língua portuguesa. A avaliação experimental, conduzida sobre 15 casos de teste representativos, indicou, dentro do escopo avaliado, correspondência adequada do comportamento do agente às expectativas em todas as dimensões consideradas, sem ocorrência de respostas que não correspondem à realidade dos dados. Documentou-se ainda o fenômeno de *over-elicitation* e a estratégia adotada para sua mitigação. Os resultados sugerem que a arquitetura proposta constitui abordagem promissora para sistemas de recomendação conversacionais em comércio eletrônico.

Palavras-Chave: Agentes LLM; Sistemas de recomendação conversacionais; Busca vetorial; Comércio eletrônico.

Abstract

Traditional search systems in e-commerce rely on structured filters that require users to translate their needs into categorized attributes, distancing themselves from the natural way consumers express their purchasing desires in free language. Large Language Models have the potential to reduce this gap, but their isolated use in recommendation scenarios faces relevant limitations, particularly regarding access to domain-specific and up-to-date information, and the tendency toward hallucination. This work investigates how LLM-based agents, equipped with external tools, can be designed to perform conversational product recommendation from requests expressed in natural language. A chatbot based on an LLM agent was developed, operating on a tokenized representation of products, performing vector similarity search over an index hosted on the Databricks platform, and conducting preference elicitation to handle underspecified requests. The experimental scope was restricted to four categories of consumer electronics, with user queries conducted in Brazilian Portuguese. The experimental evaluation, conducted on 15 representative test cases, indicated, within the evaluated scope, adequate correspondence between the agent’s behavior and the expected outcomes across all considered dimensions, with no occurrence of factually unsupported responses. The over-elicitation phenomenon and the strategy adopted for its mitigation were also documented. The results suggest that the proposed architecture constitutes a promising approach for conversational recommender systems in e-commerce.

Keywords: LLM agents; Conversational recommender systems; Vector search; E-commerce.

Sumário

Lista de Tabelas	viii
Lista de Figuras	x
Lista de Abreviaturas	xi
Lista de Códigos	xii
1 Introdução	1
1.1 Justificativa	3
1.2 Objetivos	6
1.2.1 Objetivo Geral	6
1.2.2 Objetivos Específicos	6
2 Referencial teórico	8
2.1 Processamento de Linguagem Natural	8
2.2 Transformers	9
2.3 Large Language Models	11
2.4 Agentes LLM	12
2.5 Sistemas de Recomendação	14
2.6 Representação de Itens em Sistemas de Recomendação	15
2.7 LangChain	17
2.8 Trabalhos Relacionados	18

3	Metodologia	22
3.1	Análise Exploratória dos Dados	23
3.1.1	Validação de Extração	26
3.2	Tokenização	27
3.3	Busca Vetorial	30
3.4	Agente LLM	31
3.4.1	Ferramentas do Agente	32
3.4.2	Prompt do Sistema	35
3.4.3	Problema de <i>over-elicitation</i>	37
3.4.4	Memória Conversacional	38
3.5	Interface Conversacional	38
4	Resultados e Discussão	41
4.1	Resultados Quantitativos	42
4.2	Resultados Qualitativos	43
4.3	Desempenho Operacional	45
4.3.1	Latência	45
4.3.2	Consumo de Tokens e Custo Monetário	46
4.4	<i>Over-Elicitation</i>	47
4.5	Resumo e limitações	50
5	Considerações finais	52
A	Códigos utilizados	56
B	Relação completa dos casos de teste	60
B.1	Tabela resumida dos casos de teste	60
B.2	Detalhamento dos casos: entradas, consultas e respostas	62
B.2.1	Casos específicos	62
B.2.2	Casos informais	65

B.2.3	Casos parcialmente vagos	68
B.2.4	Casos vagos	69
B.2.5	Casos fora do domínio	70

Lista de Tabelas

3.1	Quantidade de produtos por categoria	24
3.2	Faixas de preço utilizadas na tokenização	25
3.3	Resultados da extração por categoria	26
3.4	Descrição das colunas para a categoria de televisores	28
3.5	Descrição das colunas para a categoria de <i>smartphones</i>	28
3.6	Descrição das colunas para a categoria de tablets	28
3.7	Descrição das colunas para a categoria de notebooks	29
4.1	Resultados da avaliação por dimensão	42
4.2	Comportamento esperado e observado por categoria	43
4.3	Latência média por categoria	45
4.4	Estimativa de custo mensal por cenário operacional	47
4.5	Comparação entre as versões antes e após o ajuste	49
B.1	Relação completa dos casos de teste utilizados na avaliação experimental.	61

Lista de Figuras

1.1	Exemplos de filtros por categoria em diferentes plataformas	4
3.1	Arquitetura do sistema	23
3.2	Exemplo de produto na base original	24
3.3	Exemplo de extração de características	25
3.4	Exemplo de características extraídas	25
3.5	Exemplo de produto tokenizado	30
3.6	Interface inicial	39
3.7	Exemplo de apresentação de recomendação	40
4.1	Ocorrência de <i>over-elicitation</i>	48
4.2	Correção de <i>over-elicitation</i>	49

Lista de Abreviaturas

Abreviatura	Descrição
NLP	<i>Natural Language Processing</i>
LLM	<i>Large Language Models</i>
CRS	<i>Conversational Recommender Systems</i>
API	<i>Application Programming Interface</i>
NLU	<i>Natural Language Understanding</i>
NLG	<i>Natural Language Generation</i>
RNN	<i>Recurrent Neural Networks</i>
LSTM	<i>Long Short-Term Memory</i>
BERT	<i>Bidirectional Encoder Representations from Transformers</i>
GPT	<i>Generative Pre-trained Transformer</i>
PLM	<i>Pre-trained Language Models</i>
RLP	<i>Recommendation as Language Processing</i>

Lista de Códigos

3.1	Ferramentas passadas ao agente	33
3.2	Definição da função utilizada em <code>BuscarProdutos</code>	34
3.3	Definição função utilizada em <code>PedirClarificacao</code>	35
3.4	Prompt de sistema utilizado pelo agente	36
A.1	Extração de atributos para a categoria <i>smartphone</i>	56
A.2	Extração de atributos para a categoria televisão.	57
A.3	Extração de atributos para a categoria <i>notebook</i>	58
A.4	Extração de atributos para a categoria <i>tablet</i>	59

Capítulo 1

Introdução

O Processamento de Linguagem Natural (*Natural Language Processing* – NLP) é uma área da Computação dedicada a permitir que máquinas compreendam e gerem linguagem humana, seja em formato de texto ou voz (CHOWDHURY, 2005). Entre os avanços mais significativos dessa área nos últimos anos destacam-se os *Large Language Models* (LLMs), modelos estatísticos de larga escala baseados em redes neurais profundas, viabilizados pela arquitetura *Transformer*, pelo aumento da capacidade computacional e pela disponibilidade de grandes volumes de dados para treinamento (NAVEED et al., 2024; ZHAO et al., 2023). Desde o lançamento do ChatGPT pela OpenAI em novembro de 2022, as LLMs deixaram de ser objeto restrito ao meio acadêmico e passaram a integrar produtos amplamente utilizados, como assistentes virtuais, sistemas de recomendação e *chatbots* (BENGESI et al., 2024).

Apesar do desempenho expressivo em tarefas como tradução, síntese textual e resposta a perguntas, as LLMs apresentam limitações estruturais bem documentadas (CHENG et al., 2024). Entre elas, destacam-se a defasagem de informações decorrente do alto custo de retreinamento e a incapacidade de acessar diretamente, sem mediação de ferramentas, fontes de dados externas ao seu escopo (NAVEED et al., 2024). Tais limitações tornam-se particularmente críticas em aplicações comerciais relacionadas a produtos, nas quais a informação relevante (preços, categoria, especificações técnicas) é dinâmica e específica de cada plataforma. Discutir os mecanismos pelos quais essa especificidade e essa dinamicidade são incorporadas a um sistema de recomendação é um dos aspectos do presente trabalho. Como resposta a essas lacunas, surgiram os

agentes LLM: sistemas autônomos que combinam um modelo de linguagem com capacidades de planejamento, memória e uso de ferramentas com acesso a informações externas (GUO et al., 2024; CHENG et al., 2024), ampliando consideravelmente o escopo de tarefas executáveis e permitindo respostas contextualizadas e atualizadas.

Um domínio em que essas capacidades se mostram especialmente promissoras é o de sistemas de recomendação conversacionais aplicados ao comércio eletrônico (JANNACH et al., 2021; LIU et al., 2023). Plataformas tradicionais de busca de produtos exigem que o usuário traduza suas necessidades em filtros estruturados, como categorias, faixas de preço, marcas e especificações técnicas, o que impõe uma interação rígida e pouco aderente à forma como as pessoas naturalmente expressam seus desejos de consumo. Um cliente raramente pensa em termos de `categoria: smartphone; RAM: 8GB; armazenamento: 128GB; R$ 2.500 ≤ preço ≤ R$ 3.000`; mais frequentemente, ele formula algo como “*quero um smartphone com 8GB de RAM, 128GB de armazenamento e que custe entre dois mil e quinhentos e três mil reais*”. A lacuna entre essas duas formas de expressão constitui um problema relevante de interação humano-computador, e a combinação entre LLMs e o acesso a catálogos de produtos surge como uma alternativa para reduzi-la.

Diante deste contexto, o presente trabalho investiga como agentes baseados em LLM podem traduzir requisitos expressos em linguagem natural em consultas estruturadas a catálogos de produtos, retornando recomendações alinhadas às preferências declaradas pelo usuário. Para tanto, propõe-se o desenvolvimento de um *chatbot* baseado em agente LLM capaz de interpretar solicitações conversacionais de recomendação, extrair os atributos relevantes (categoria, especificações técnicas, faixa de preço, entre outros), consultar o catálogo de produtos e apresentar ao usuário sugestões pertinentes em linguagem natural. As principais contribuições deste trabalho são: (i) o desenvolvimento de uma estratégia de representação tokenizada de produtos, concebida para viabilizar busca semântica sobre catálogos cujos atributos não se encontram disponíveis em formato estruturado convencional; (ii) o projeto e a implementação de um agente LLM voltado à recomendação conversacional de produtos, utilizando o *framework* LangChain, com mecanismo de eliciação de preferências para tratamento de solicitações incompletas; e

(iii) a avaliação experimental do agente quanto à precisão na interpretação dos requisitos, à adequação das recomendações geradas e ao seu comportamento em cenários conversacionais variados.

1.1 Justificativa

O comércio eletrônico consolidou-se como um dos setores de maior crescimento da economia digital nas últimas décadas (VEIGA et al., 2024), e a forma como consumidores buscam e descobrem produtos em plataformas online tornou-se um fator crítico tanto para a satisfação do usuário quanto para o desempenho comercial das empresas (LIU et al., 2022). No entanto, a interação predominante nesses sistemas ainda se baseia em mecanismos de busca por palavras-chave e filtros estruturados, como ilustrado na Figura 1.1, nos quais o usuário precisa decompor mentalmente sua necessidade em atributos categorizados, como categoria, marca, faixa de preço e especificações técnicas, para então traduzi-los na interface oferecida pela plataforma. Esse processo impõe um esforço cognitivo considerável e frequentemente resulta em frustração, especialmente quando o usuário não domina o vocabulário técnico do domínio ou quando suas preferências envolvem múltiplos critérios simultâneos.

Figura 1.1: Exemplos de filtros por categoria em diferentes plataformas

The figure displays three distinct filter interfaces for different product categories on various e-commerce platforms:

- Smartphone - Amazon:** Features a vertical list of brand filters (Marcas) including Apple, Samsung, XIAOMI, Motorola, POCO, OPPO, realme, Meta Quest, Redmi, and REDMAGIC. Below this is a star rating filter (Avaliação dos Clientes) set to 4 stars and above. The Price filter (Preço) offers ranges from 'Até R\$250' to 'Mais de R\$2000'. The Condition filter (Condição) has options for 'Novo' and 'Usado'.
- Televisão - Mercado Livre:** Shows a 'Preço' section with three price ranges: 'Até R\$ 100', 'R\$ 100 a R\$ 200', and 'Mais de R\$ 200'. It includes a 'Tipo de promoção' section with 'Oferta do dia' and 'Oferta relâmpago'. A 'Marca' section lists brands like Genérica, LG, Samsung, Philco, Pci Eletro Parts, Neshop, TCL, Philips, and Suporteço, with a 'Mostrar mais' link.
- Notebook - Bemol:** Contains four filter sections: 'Armazenamento' (128GB, 256GB, 512GB, 1TB, 64GB), 'Marca' (APPLE, LENOVO, ACER, ASUS, POSITIVO, HP, SAMSUNG, MSI), 'Média de avaliação' (4 stars (0) and 5 stars (5)), and 'Memória RAM' (4GB, 8GB, 12GB, 16GB, 32GB, 24GB).

Smartphone - Amazon

Notebook - Bemol

Em contraste, a comunicação humana é naturalmente expressa em linguagem livre, contextual e muitas vezes implícita. Um consumidor tende a formular sua necessidade de forma integrada, como “*procuro um notebook leve, bom para trabalho remoto, com boa duração de bateria e até cinco mil reais*”, em vez de preencher campos isolados de um formulário. A distância entre essa forma natural de expressar desejos de consumo e a interface estruturada das plataformas tradicionais constitui uma barreira de usabilidade reconhecida na literatura de interação humano-computador e de sistemas de recomendação (JANNACH et al., 2021).

Essa barreira de usabilidade tem desdobramentos especialmente relevantes em populações

com letramento digital limitado. No Brasil, segmentos significativos de consumidores apresentam dificuldade na navegação por interfaces que demandam decomposição da necessidade em filtros categorizados, vocabulário técnico de produtos e configuração manual de critérios. Para esses usuários, interfaces conversacionais em linguagem natural representam não apenas alternativa de conveniência, mas potencial caminho de ampliação efetiva do acesso ao comércio eletrônico, ao operar no registro linguístico com que essas pessoas naturalmente se comunicam. Embora a avaliação experimental conduzida neste trabalho não tenha incluído estudo com esse público específico, o alinhamento da abordagem proposta com essa demanda de inclusão constitui dimensão de relevância prática que merece ser explicitada.

As LLMs apresentam-se como uma tecnologia promissora para reduzir essa distância, dada sua capacidade de compreender linguagem natural com fluência próxima à humana. Contudo, seu uso isolado em cenários de recomendação de produtos enfrenta limitações que se manifestam de forma especialmente crítica nesse domínio (CHENG et al., 2024). A defasagem temporal do conhecimento do modelo, restrito ao período de seu treinamento, é incompatível com catálogos cujos preços, estoques e especificações mudam continuamente. A incapacidade de acessar diretamente fontes externas de dados impede a consulta aos próprios catálogos das plataformas, fonte primária da informação relevante. A tendência à produção de respostas plausíveis mas incorretas, fenômeno conhecido como alucinação (JI et al., 2023), pode levar à sugestão de produtos inexistentes ou à atribuição de características fabricadas pelo próprio modelo a produtos reais, comportamento esse particularmente prejudicial em contexto comercial, no qual a veracidade de dados é de suma importância.

Os agentes LLM surgem como uma resposta arquitetural a essas limitações. Ao combinar a capacidade linguística das LLMs com o uso de ferramentas com acesso a dados externos, memória e raciocínio iterativo (GUO et al., 2024; CHENG et al., 2024), um agente pode interpretar a solicitação do usuário em linguagem natural, extrair os atributos relevantes, consultar uma fonte de dados confiável e fundamentar suas recomendações em informações reais e atualizadas. Essa abordagem mitiga simultaneamente os três problemas mencionados, transferindo, da memória paramétrica do modelo para fontes externas verificáveis, a responsabilidade pelo

fornecimento dos dados verdadeiros que sustentam as recomendações.

Apesar do potencial desse paradigma, sua aplicação ao domínio de recomendação conversacional de produtos ainda está em desenvolvimento, e diversos aspectos práticos permanecem pouco explorados na literatura, tais como a estratégia de representação dos produtos que viabilize busca semântica eficaz sobre catálogos heterogêneos, a forma mais eficaz de extrair atributos estruturados a partir de descrições livres, o impacto da escolha do *prompt* e do tipo de agente sobre a qualidade das recomendações, e o comportamento desses sistemas diante de solicitações ambíguas ou incompletas. Investigar tais aspectos contribui não apenas para o avanço da pesquisa em agentes LLM, mas também para a viabilidade prática de uma nova geração de interfaces conversacionais no comércio eletrônico, mais alinhadas à forma natural como os usuários expressam suas necessidades de consumo.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um *chatbot* baseado em agente LLM para a recomendação conversacional de produtos no contexto de comércio eletrônico.

1.2.2 Objetivos Específicos

- Identificar, por meio de revisão bibliográfica, os principais *frameworks*, arquiteturas de agentes LLM e técnicas de busca semântica aplicáveis ao domínio de recomendação conversacional de produtos.
- Desenvolver uma estratégia de representação tokenizada dos produtos, adequada à viabilização de busca semântica sobre o catálogo, e estabelecer as decisões de projeto envolvidas em sua construção.
- Projetar e implementar um agente LLM, capaz de extrair atributos estruturados (categoria, especificações técnicas, faixa de preço, entre outros) a partir de solicitações em

linguagem natural, mapeá-los em consultas semânticas ao catálogo de produtos, conduzir elicitación ativa de preferências quando as solicitações apresentarem informação insuficiente, e manter o contexto conversacional ao longo da interação.

- Avaliar experimentalmente o comportamento do agente desenvolvido, considerando critérios como a precisão na extração dos atributos solicitados, a aderência das recomendações geradas aos requisitos do usuário, a ocorrência de respostas não fundamentadas e a robustez do agente diante de solicitações ambíguas ou incompletas.

Capítulo 2

Referencial teórico

2.1 Processamento de Linguagem Natural

Processamento de Linguagem Natural é o subcampo da Inteligência Artificial dedicado a permitir que computadores compreendam, interpretem e gerem linguagem humana, em formas como texto e voz (CHOWDHURY, 2005; PATWARDHAN; MARRONE; SANSONE, 2023). Trata-se de uma área de longa trajetória na ciência da computação, cujo marco inicial é frequentemente associado à publicação de *Syntactic Structures*, de Noam Chomsky, na década de 1950, que inaugurou as primeiras tentativas de tratamento computacional da linguagem por meio de regras gramaticais formais (JOHRI et al., 2020).

A evolução do campo pode ser caracterizada em três grandes momentos. O primeiro, baseado em abordagens simbólicas, dependia da codificação manual de regras linguísticas e enfrentava limitações severas em escalabilidade e generalização. O segundo momento, caracterizado pela ascensão dos modelos estatísticos, transferiu o foco para a aprendizagem de padrões com base em grandes conjuntos de textos, dando origem aos *Statistical Language Models*. O terceiro, e atual, é dominado por técnicas de aprendizado profundo, com destaque para arquiteturas baseadas em redes neurais que culminaram nos *Large Language Models* (ZHAO et al., 2023). Essa progressão permitiu avanços significativos na capacidade dos computadores de compreender e gerar linguagem natural, expandindo as aplicações de NLP em áreas como tradução automática, síntese de texto e reconhecimento de fala (CHOWDHERY et al., 2022).

Do ponto de vista funcional, NLP é tradicionalmente dividido em duas grandes vertentes (ZHOU et al., 2020): a Compreensão de Linguagem Natural (*Natural Language Understanding* – NLU), responsável por interpretar entradas em linguagem humana, extraindo intenções, entidades, sentimentos ou significados, e a Geração de Linguagem Natural (*Natural Language Generation* – NLG), encarregada de produzir respostas em linguagem natural a partir de representações internas. Em sistemas conversacionais, ambas as vertentes operam de forma integrada: o sistema compreende o que o usuário expressa (NLU) e formula respostas adequadas em linguagem natural (NLG).

Essa dupla capacidade é particularmente relevante para o presente trabalho. Em um cenário de recomendação conversacional, o sistema precisa, por um lado, interpretar solicitações expressas livremente pelo usuário, para extrair atributos como categoria de produto, especificações técnicas e faixa de preço, e por outro, gerar respostas em linguagem natural que apresentem as recomendações de forma clara e contextualizada. Os avanços recentes em NLP, especialmente aqueles viabilizados pelas arquiteturas baseadas em *Transformers* e pelos *Large Language Models*, criaram as condições técnicas para que essas duas tarefas possam ser executadas com qualidade próxima à humana. Tais avanços são detalhados nas subseções seguintes.

2.2 Transformers

Até meados da década de 2010, os modelos predominantes em NLP baseavam-se em arquiteturas recorrentes, como as Redes Neurais Recorrentes (*Recurrent Neural Networks* – RNNs) e suas variantes mais sofisticadas, como as LSTMs (*Long Short-Term Memory*). Embora capazes de lidar com sequências, esses modelos enfrentavam duas limitações estruturais relevantes: a dificuldade em capturar dependências entre elementos distantes em uma sequência, devido à propagação sequencial da informação, e a impossibilidade de paralelização do processamento, uma vez que cada passo dependia do passo anterior. Essas limitações restringiam tanto a qualidade dos modelos quanto a viabilidade de treiná-los em larga escala.

Em 2017, foi proposta a arquitetura *Transformer* (VASWANI et al., 2017), que substituiu o processamento recorrente por um mecanismo baseado em atenção (*attention*), permitindo que

o modelo considerasse todas as posições de uma sequência simultaneamente. Essa mudança removeu o gargalo da sequencialidade, viabilizou o processamento paralelo em hardware especializado e melhorou significativamente a capacidade de modelar dependências de longo alcance em textos (BENGESI et al., 2024). Tais ganhos consolidaram os *Transformers* como a arquitetura dominante em NLP e como base para os avanços subsequentes em modelos generativos (BROWN et al., 2020; CHOWDHERY et al., 2022).

Em sua formulação original, um *Transformer* é composto por dois blocos principais: um encoder, que mapeia uma sequência de entrada em uma representação contínua, e um decoder, que gera a sequência de saída a partir dessa representação, um elemento por vez (VASWANI et al., 2017). A geração ocorre de forma auto-regressiva, isto é, cada novo elemento gerado é incorporado como contexto para a geração do elemento seguinte. Variações arquiteturais posteriores adotaram apenas o encoder (como o *Bidirectional Encoder Representations from Transformers* - BERT) ou apenas o decoder (como a família *Generative Pre-trained Transformer* - GPT), conforme o tipo de tarefa-alvo.

Três componentes são fundamentais para o funcionamento dos *Transformers* (BENGESI et al., 2024). O mecanismo de atenção (*Self-Attention*) permite que, ao processar cada elemento da sequência, o modelo pondere a relevância de todos os outros elementos, capturando relações contextuais de forma direta. A atenção em múltiplas cabeças (*Multi-Head Attention*) estende esse mecanismo executando várias operações de atenção em paralelo, permitindo que o modelo capture diferentes tipos de relação simultaneamente. Por fim, como o processamento paralelo elimina a noção implícita de ordem que existia nos modelos recorrentes, foi introduzida a codificação posicional (*Positional Encoding*), que insere informação sobre a posição de cada elemento na sequência, preservando a estrutura linear do texto (VASWANI et al., 2017).

A combinação entre eficiência computacional, capacidade de paralelização e qualidade na modelagem de sequências longas tornou os *Transformers* a base sobre a qual foram construídos os *Large Language Models* contemporâneos, conceito desenvolvido na subseção seguinte.

2.3 Large Language Models

Os *Large Language Models* (LLMs) são modelos de linguagem estatísticos de larga escala, baseados em redes neurais profundas, normalmente em arquiteturas *Transformer* e treinados sobre vastas quantidades de dados textuais com o objetivo de aprender representações genéricas da linguagem (MINAEE et al., 2024; ZHAO et al., 2023). O termo “*large*” refere-se ao número de parâmetros desses modelos, que pode variar de dezenas de bilhões a centenas de bilhões, escala que se mostrou determinante para o surgimento de capacidades qualitativamente superiores às observadas em modelos menores.

LLMs são geralmente derivadas de uma família anterior de modelos, os *Pre-trained Language Models* (PLMs), por meio do escalonamento massivo do número de parâmetros e do volume de dados de treinamento (ZHAO et al., 2023). O fluxo de desenvolvimento típico envolve uma fase de pré-treinamento sobre grandes bases textuais não rotuladas, na qual o modelo aprende padrões gerais da linguagem, e uma fase opcional de ajuste fino (*fine-tuning*) sobre dados específicos de uma tarefa ou domínio. A escala adotada nas LLMs, contudo, fez emergir capacidades que vão além das observadas em PLMs convencionais, tanto na qualidade da geração e compreensão textual quanto na presença de habilidades emergentes (MINAEE et al., 2024).

Habilidades emergentes são capacidades que surgem em LLMs a partir de determinada escala de parâmetros e dados, sem terem sido explicitamente programadas no processo de treinamento (WEI et al., 2022). Três delas são particularmente relevantes para o presente trabalho. A aprendizagem em contexto (*in-context learning*) permite que o modelo execute uma tarefa nova com base apenas em exemplos fornecidos no prompt, sem qualquer atualização de seus parâmetros, o que viabiliza adaptações rápidas a domínios específicos sem retreinamento. O acompanhamento de instruções (*instruction following*) permite que o modelo execute tarefas descritas em linguagem natural, sem necessidade de exemplos prévios, permitindo que um agente possa interpretar diretamente solicitações livres do usuário. O raciocínio em múltiplas etapas (*chain-of-thought reasoning*) permite que o modelo decomponha tarefas complexas em uma sequência de passos intermediários, facilitando a resolução de problemas que envolvem

decisões encadeadas, como por exemplo, identificar que uma solicitação de recomendação requer extrair atributos do texto, consultar uma fonte externa e formatar a resposta final.

A combinação dessas habilidades, aliada à fluência da geração textual, fez das LLMs uma das tecnologias mais transformadoras do campo da inteligência artificial nos últimos anos. O lançamento do ChatGPT pela OpenAI, em novembro de 2022, marcou um ponto de inflexão na adoção dessas tecnologias pelo público geral (ZHAO et al., 2023), impulsionando a aplicação de LLMs em domínios como assistentes virtuais, sistemas de busca, *chatbots* e sistemas de recomendação (BENGESI et al., 2024).

Apesar dos avanços, as LLMs apresentam limitações reconhecidas, sendo a alucinação uma das mais críticas. O termo designa a tendência do modelo de produzir respostas plausíveis em forma, mas factualmente incorretas, sendo afirmações que não correspondem a nenhuma informação real, seja em sua base de treinamento, seja em fontes externas (JI et al., 2023). Esse fenômeno decorre da natureza probabilística do processo de geração: a LLM produz a sequência de palavras estatisticamente mais provável dado o contexto, sem qualquer mecanismo intrínseco de verificação factual. Em domínios sensíveis à precisão, como medicina, direito e comércio eletrônico, a alucinação representa um obstáculo significativo ao uso isolado dessas tecnologias. Em um cenário de recomendação de produtos, por exemplo, uma LLM operando isoladamente pode sugerir itens inexistentes, atribuir especificações técnicas incorretas ou inventar preços fora da realidade. Tais limitações motivaram o desenvolvimento de arquiteturas que combinam LLMs com fontes externas de dados verificáveis, dentre as quais se destacam os agentes LLM, abordados na subseção seguinte.

2.4 Agentes LLM

Conforme discutido na subseção anterior, as LLMs apresentam limitações que restringem sua aplicação direta em domínios sensíveis a veracidade de dados. A defasagem temporal de seu conhecimento, a incapacidade de acessar fontes externas ao seu escopo e a tendência à alucinação tornam-se obstáculos relevantes em cenários nos quais a resposta precisa refletir informações atualizadas e verificáveis. Como resposta arquitetural a essas limitações, surgiu o paradigma

dos agentes baseados em LLM, nos quais o modelo de linguagem deixa de operar isoladamente e passa a integrar um sistema mais amplo, dotado de capacidade de decisão, memória e acesso a ferramentas de informações externas (GUO et al., 2024).

Um agente LLM pode ser definido como um sistema autônomo que combina um modelo de linguagem com mecanismos adicionais que ampliam suas capacidades operacionais (CHENG et al., 2024). De forma sintética, um agente é caracterizado por quatro componentes principais. O núcleo de raciocínio é desempenhado pela própria LLM, responsável por interpretar entradas, planejar ações e formular respostas. As ferramentas são funções externas que o agente pode invocar para acessar informações ou executar operações que extrapolam suas capacidades intrínsecas, podendo ser consultas a APIs, buscas em bases de dados, execução de cálculos ou qualquer outro processo encapsulado como interface utilizável. A memória mantém o histórico das interações e das ações executadas, permitindo que o agente preserve contexto ao longo do tempo. O mecanismo de orquestração coordena o ciclo de funcionamento, determinando quando o agente deve raciocinar, quando deve agir e quando deve responder ao usuário.

De acordo com (YAO et al., 2023), o funcionamento de um agente LLM tipicamente segue um ciclo iterativo de raciocínio e ação, chamado de paradigma ReAct (*Reasoning and Acting*). Nesse ciclo, o agente, a cada turno, observa a entrada disponível, sendo a solicitação inicial do usuário ou o retorno de uma ferramenta acionada em passo anterior, então raciocina sobre essa entrada com base em seu *prompt* de sistema, decide entre executar uma ação ou produzir uma resposta final ao usuário, e, em caso de ação, invoca a ferramenta adequada com os parâmetros pertinentes. O retorno da ferramenta torna-se nova entrada no ciclo, que se repete até que o agente determine que pode formular a resposta final.

Esse ciclo confere ao agente capacidades qualitativamente distintas daquelas de uma LLM operando isoladamente. A possibilidade de invocar ferramentas externas resolve a limitação de acesso a informações dinâmicas, possibilitando o acesso a dados que não estavam no treinamento do modelo e agora podem ser recuperados em tempo de execução por meio de APIs ou bancos de dados. A possibilidade de invocar múltiplas ferramentas em sequência permite a decomposição de tarefas complexas em passos mais simples, cada qual tratado por um compo-

nente especializado. A possibilidade de iterar com base nos retornos das ferramentas viabiliza estratégias de refinamento progressivo, nas quais o agente ajusta sua abordagem em função das informações obtidas em passos anteriores.

A literatura sobre agentes LLM contempla diversas arquiteturas, que diferem quanto ao número de agentes envolvidos, à complexidade do ciclo de raciocínio e ao grau de autonomia conferido. Sistemas com um único agente, como o adotado neste trabalho, são adequados a domínios bem definidos com escopo restrito de ações. Sistemas com múltiplos agentes coordenados, por sua vez, são empregados em tarefas mais complexas que se beneficiam de especialização e divisão de responsabilidades (GUO et al., 2024). No plano da implementação, frameworks como LangChain, LlamaIndex e AutoGen oferecem abstrações que simplificam a construção de agentes, encapsulando o ciclo ReAct e fornecendo interfaces padronizadas para a definição de ferramentas, prompts e mecanismos de memória.

2.5 Sistemas de Recomendação

Sistemas de recomendação são aplicações de software que auxiliam usuários a encontrar itens de interesse em situações de sobrecarga de informação (JANNACH et al., 2021). Tais sistemas tornaram-se um dos casos de sucesso mais visíveis da inteligência artificial aplicada, especialmente no contexto do comércio eletrônico, onde contribuem tanto para a satisfação do usuário quanto para o desempenho comercial das plataformas. Tradicionalmente, a literatura classifica os sistemas de recomendação em três grandes paradigmas: filtragem colaborativa, que baseia recomendações em padrões de comportamento de usuários semelhantes; filtragem baseada em conteúdo, que utiliza atributos dos próprios itens; e abordagens híbridas, que combinam as duas estratégias.

Segundo (JANNACH et al., 2021), a maioria dos sistemas de recomendação convencionais opera segundo o paradigma de interação única (*one-shot interaction paradigm*), no qual as preferências do usuário são estimadas a partir de comportamento passado observado e a saída do sistema consiste em uma lista ordenada de sugestões apresentada de forma unidirecional. Embora eficaz em diversos cenários, esse modelo apresenta limitações conhecidas, como

a dificuldade em incorporar preferências contextuais não registradas no histórico, pouca explicabilidade das recomendações, e baixa capacidade de adaptação dinâmica diante de mudanças momentâneas no interesse do usuário.

Em resposta a essas limitações, surgiu o paradigma dos Sistemas de Recomendação Conversacionais (*Conversational Recommender Systems* – CRS), onde (JANNACH et al., 2021) define como sistemas que suportam um conjunto mais rico de interações, permitindo que o usuário expresse preferências de forma incremental, faça perguntas sobre as recomendações e ofereça feedback ao longo do diálogo. O recente crescimento do interesse em CRS deve-se principalmente aos avanços em processamento de linguagem natural, ao surgimento de assistentes virtuais controlados por voz e à popularização da tecnologia de *chatbots* (GAO et al., 2021).

Um CRS típico envolve componentes para elicitación de preferências (extração das necessidades do usuário a partir do diálogo), gerenciamento de estado conversacional, geração de recomendações, e geração de respostas em linguagem natural (JANNACH et al., 2021). Adicionalmente, todo CRS pressupõe acesso a uma base de informações sobre os itens recomendáveis, que pode conter avaliações, metadados, atributos descritivos ou tags. Além disso, a base serve não apenas para geração das recomendações, mas também à formulação de perguntas ao usuário e elaboração de explicações sobre as sugestões oferecidas.

O surgimento das LLMs introduziu uma nova fronteira nesse campo. Trabalhos recentes investigam como esses modelos podem ser incorporados a CRS, seja como camada conversacional sobre recomendadores tradicionais (GAO et al., 2023), seja como componentes complementares em arquiteturas híbridas (LIU et al., 2023), seja como núcleo de agentes autônomos capazes de interagir com fontes externas de dados sobre produtos. É nessa última linha que se insere o presente trabalho, ao propor um agente LLM que opera como CRS por meio da integração direta com um catálogo de produtos.

2.6 Representação de Itens em Sistemas de Recomendação

Para que sistemas de recomendação operem sobre um conjunto de itens, sejam produtos em um catálogo de comércio eletrônico, sejam filmes, livros, ou qualquer outro elemento recomendá-

vel, esses itens precisam ser convertidos em alguma representação computacional manipulável pelos algoritmos. Essa representação define quais aspectos dos itens podem ser comparados, agrupados ou relacionados a preferências dos usuários, e constitui uma decisão de projeto com impacto direto sobre a qualidade das recomendações (ZHAO et al., 2024).

A literatura sobre o tema contempla três grandes paradigmas de representação de itens. O primeiro, baseado em atributos estruturados, codifica cada item como um vetor cujas dimensões correspondem a características predefinidas, como preço, categoria, marca, especificações técnicas, entre outras. Em sistemas convencionais, dados estruturados são tipicamente formulados como codificações *one-hot* sobre as quais se aplica uma camada de *embedding* simples para obtenção de representações densas (LIN et al., 2024). Essa abordagem é direta e interpretável, mas depende da existência de atributos bem-definidos e exige tratamento explícito para valores ausentes, escalas heterogêneas e atributos categóricos. O segundo paradigma, baseado em *embeddings* aprendidos, representa cada item como um vetor denso em um espaço latente, obtido por meio de técnicas de aprendizado a partir do comportamento dos usuários ou do conteúdo dos próprios itens. Essa abordagem captura relações semânticas mais finas, mas produz representações de menor interpretabilidade.

O terceiro paradigma, mais recente, baseia-se em representações textuais estruturadas, nas quais os itens são descritos como sequências de texto compostas por marcadores ou descritores padronizados. Esse paradigma ganhou força com o trabalho pioneiro de (GENG et al., 2022), que propôs tratar a tarefa de recomendação como um problema de processamento de linguagem, paradigma denominado *Recommendation as Language Processing* (RLP). Na formulação proposta pelos autores, todos os dados relevantes ao sistema de recomendação, como atributos dos itens, metadados, históricos de interação e avaliações, são convertidos em sequências de linguagem natural, viabilizando o uso de modelos de linguagem como núcleo do processo de recomendação. Essa formulação inaugurou uma linha de pesquisa que tem se mostrado especialmente promissora no contexto da disseminação dos LLMs e dos modelos de *embeddings* derivados de arquiteturas *Transformer*.

Trabalhos posteriores aprofundaram essa abordagem em diferentes direções. *Surveys* recen-

tes sobre o emprego de LLMs em sistemas de recomendação identificam, entre os principais usos, o papel das LLMs como codificadores de características (*feature encoders*), capazes de gerar *embeddings* semanticamente ricos a partir de descrições textuais dos itens (LIN et al., 2024). Tais *embeddings* mostram-se particularmente eficazes em capturar relações sutis entre itens, especialmente quando comparados a abordagens baseadas em codificações *one-hot* sobre atributos isolados. Estudos recentes evidenciam, ainda, que representações textuais estruturadas favorecem a generalização para itens fora do conjunto de treinamento e para domínios não vistos, justamente por preservarem informação semântica em vez de operarem exclusivamente sobre identificadores discretos (LIN et al., 2024).

A escolha entre os três paradigmas de representação depende de fatores como a natureza dos itens, a homogeneidade dos atributos disponíveis, a existência de dados comportamentais para treinamento e os requisitos de interpretabilidade e atualização. Representações textuais estruturadas mostram-se particularmente adequadas em cenários nos quais os itens possuem atributos heterogêneos ou parcialmente ausentes, e nos quais a flexibilidade semântica é mais valorizada do que a fidelidade numérica exata. Tais condições são frequentes em catálogos comerciais de produtos diversificados, contexto no qual se desenvolve o presente trabalho. A representação tokenizada adotada nesta investigação, situa-se nesse terceiro paradigma, sendo concebida especificamente para operar em conjunto com mecanismos de busca por similaridade vetorial.

2.7 LangChain

LangChain é um *framework* desenvolvido para facilitar a construção de aplicações que utilizam LLMs como componente central, oferecendo abstrações e ferramentas que simplificam tarefas como a integração com fontes de dados externas, o gerenciamento de contexto conversacional e a coordenação entre múltiplos componentes. O *framework* organiza-se em torno de algumas abstrações principais. *Prompts* representam as entradas fornecidas ao modelo e podem ser construídos dinamicamente a partir de modelos parametrizáveis (*prompt templates*), permitindo a inclusão estruturada de variáveis fornecidas pelo usuário ou pela aplicação. *Models* designam

os modelos de linguagem em si, com diferentes interfaces conforme o caso de uso (modelos de completar texto, modelos de chat, modelos de *embeddings*). *Chains* permitem a composição de múltiplos componentes em fluxos sequenciais, nos quais a saída de um componente serve de entrada para o seguinte. *Memory* refere-se aos mecanismos de retenção de contexto entre turnos conversacionais, oferecendo diferentes estratégias conforme o cenário de aplicação (LANGCHAIN, 2026a).

Particularmente relevante para este trabalho é o conceito de agente no LangChain, que designa um componente capaz de selecionar autonomamente, dentre um conjunto de ferramentas (*tools*) disponíveis, qual delas utilizar para responder a uma dada solicitação do usuário. Cada ferramenta consiste em uma função encapsulada com uma descrição em linguagem natural, e o agente decide seu uso com base no raciocínio sobre essa descrição e sobre a entrada recebida. Esse mecanismo é o que permite, na presente proposta, que o agente acione mecanismos externos de consulta, como um índice de busca vetorial sobre o catálogo de produtos, quando, e apenas quando, a solicitação do usuário demandar tal consulta.

Em sua versão atual, o LangChain implementa o paradigma de uso de ferramentas discutido anteriormente, incorporando recursos nativos de *tool calling* oferecidos por modelos modernos por meio de construtores específicos para agentes que delegam ao próprio modelo a decisão sobre invocação de ferramentas. Essa evolução afasta-se de abordagens anteriores baseadas em *parsing* textual da saída do modelo e tornou-se o padrão recomendado pela documentação oficial. Mecanismos de memória conversacional, por sua vez, são integrados aos agentes por meio de componentes que encapsulam o histórico de mensagens, permitindo recuperação automática de contexto a cada nova invocação. Os detalhes de implementação que utilizam esses recursos são descritos a seguir.

2.8 Trabalhos Relacionados

A intersecção entre LLMs e sistemas de recomendação tornou-se um tópico ativo de pesquisa, com diversos trabalhos explorando como as capacidades linguísticas das LLMs podem ser combinadas com mecanismos de recomendação tradicionais ou com fontes externas de dados estru-

turados. A seguir, apresentam-se quatro trabalhos cujas propostas, arquiteturas ou domínios de aplicação dialogam diretamente com a presente investigação.

O trabalho *Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System* de (GAO et al., 2023), é uma das referências pioneiras na utilização de LLMs como camada conversacional para sistemas de recomendação. Os autores propõem um paradigma no qual perfis de usuários e históricos de interação são convertidos em *prompts* fornecidos a uma LLM, que passa a operar como interface conversacional do recomendador. Na arquitetura proposta, a LLM primeiro sumariza as preferências do usuário a partir do perfil e do histórico, em seguida um sistema de recomendação convencional gera um conjunto amplo de itens candidatos, e finalmente a LLM refina esse conjunto, produzindo a lista final de recomendações entregue ao usuário. Os experimentos demonstraram melhorias em tarefas de recomendação *top-k* e em predição de avaliações em cenário *zero-shot*, bem como ganhos em interatividade e explicabilidade. O Chat-REC compartilha com o presente trabalho a premissa central de que LLMs podem reduzir a rigidez da interação tradicional em sistemas de recomendação; contudo, sua arquitetura depende de um sistema de recomendação convencional pré-existente para gerar candidatos, enquanto a proposta deste trabalho recorre diretamente ao catálogo de produtos por meio de busca semântica acionada pelo agente LLM.

O trabalho *A Hybrid Multi-Agent Conversational Recommender System with LLM and Search Engine in E-commerce* de NIE et al., (2024), desenvolvido no contexto da plataforma chinesa JD.com, propõe um agente LLM híbrido voltado especificamente para o cenário de pré-venda em comércio eletrônico. O agente central do sistema é composto por cinco módulos principais: uma LLM proprietária ajustada (ChatSR, com 13 bilhões de parâmetros); um módulo de reconhecimento de parâmetros de busca, responsável por extrair preço, filtros e critérios de ordenação a partir da entrada do usuário; um módulo de perfil do usuário; um módulo de gerenciamento de histórico que preserva preferências como orçamento, necessidades e restrições; e um conjunto de ferramentas que se comunicam com o *front-end* e com o agente de busca. Este trabalho é particularmente relevante para a presente investigação por compartilhar o mesmo domínio de aplicação e por adotar uma estratégia de extração de parâmetros estruturados a

partir de linguagem natural muito semelhante à proposta aqui apresentada. As principais diferenças residem na escala (o trabalho da JD.com utiliza uma LLM proprietária ajustada com bilhões de parâmetros, enquanto a presente proposta emprega um modelo de uso geral via API), na complexidade arquitetural (multiagente versus agente único) e na representação dos itens (atributos estruturados convencionais versus representação tokenizada com busca semântica vetorial adotada neste trabalho).

O trabalho *Conversational Recommender System and Large Language Model Are Made for Each Other in E-commerce Pre-sales Dialogue* de (LIU et al., 2023), investiga formas de colaboração entre LLMs e sistemas de recomendação conversacional tradicionais (CRS) no contexto de diálogos de pré-venda em comércio eletrônico. Os autores partem da observação de que CRS aprendem representações de usuários e fornecem recomendações precisas com base no contexto do diálogo, mas dependem de conhecimento externo, enquanto LLMs geram respostas conversacionais fluentes mas carecem de conhecimento específico do domínio para recomendações precisas, dessa forma caracterizando os dois paradigmas como complementares. Os autores propõem dois métodos de colaboração (“CRS auxiliando LLM” e “LLM auxiliando CRS”) e validam experimentalmente a eficácia da combinação em quatro tarefas de pré-venda. Esse trabalho fundamenta teoricamente a suposição, também adotada na presente abordagem, de que o paradigma puramente baseado em LLM é insuficiente para tarefas de recomendação que dependem de informações concretas sobre produtos, sendo necessário acoplar mecanismos externos, sejam CRS tradicionais, sejam mecanismos externos de consulta a catálogos

Por fim, o trabalho *PaperQA: Retrieval-Augmented Generative Agent for Scientific Research*, de (LÁLA et al., 2023), embora pertencente a um domínio distinto (literatura científica), oferece uma referência metodológica relevante por implementar um agente LLM com integração a APIs externas utilizando o *framework* LangChain, abordagem semelhante à adotada na presente investigação. O agente desenvolvido pelos autores faz uso de três ferramentas principais: uma de busca, responsável por consultar APIs de bases de dados acadêmicas (Arxiv, PubMed, OpenAccess); uma de coleta de evidências, que filtra trechos relevantes; e uma de resposta, que sintetiza as informações coletadas em uma resposta final. Os resultados demonstraram que o

PaperQA superou modelos como Claude-2 e GPT-4 isolados, e alcançou desempenho próximo ao de pesquisadores humanos em tarefas de resposta a perguntas baseadas em literatura científica. A relevância deste trabalho para a presente proposta reside, sobretudo, na demonstração empírica de que a abordagem de agente LLM combinado com ferramentas de acesso a fontes externas pode produzir resultados superiores ao uso isolado da LLM.

Em síntese, os trabalhos analisados convergem para a constatação de que a combinação entre LLMs e fontes externas de dados produz resultados superiores ao uso isolado de cada componente. O presente trabalho se insere nessa linha de pesquisa, mas com um recorte específico: ao invés de acoplar a LLM a um CRS pré-existente (como em Chat-REC) ou a uma LLM proprietária ajustada (como no trabalho da plataforma JD.com), propõe-se uma arquitetura que não depende de sistema de recomendação prévio nem de ajuste fino de modelos proprietários, baseada em agente LLM construído sobre o *framework* LangChain, com integração direta a um mecanismo de busca semântica sobre o catálogo de produtos. Tal escolha torna a abordagem aplicável a cenários nos quais tais recursos não estão disponíveis, embora a replicação direta dos experimentos requeira adaptação à fonte de dados e à infraestrutura específicas do caso de uso.

Capítulo 3

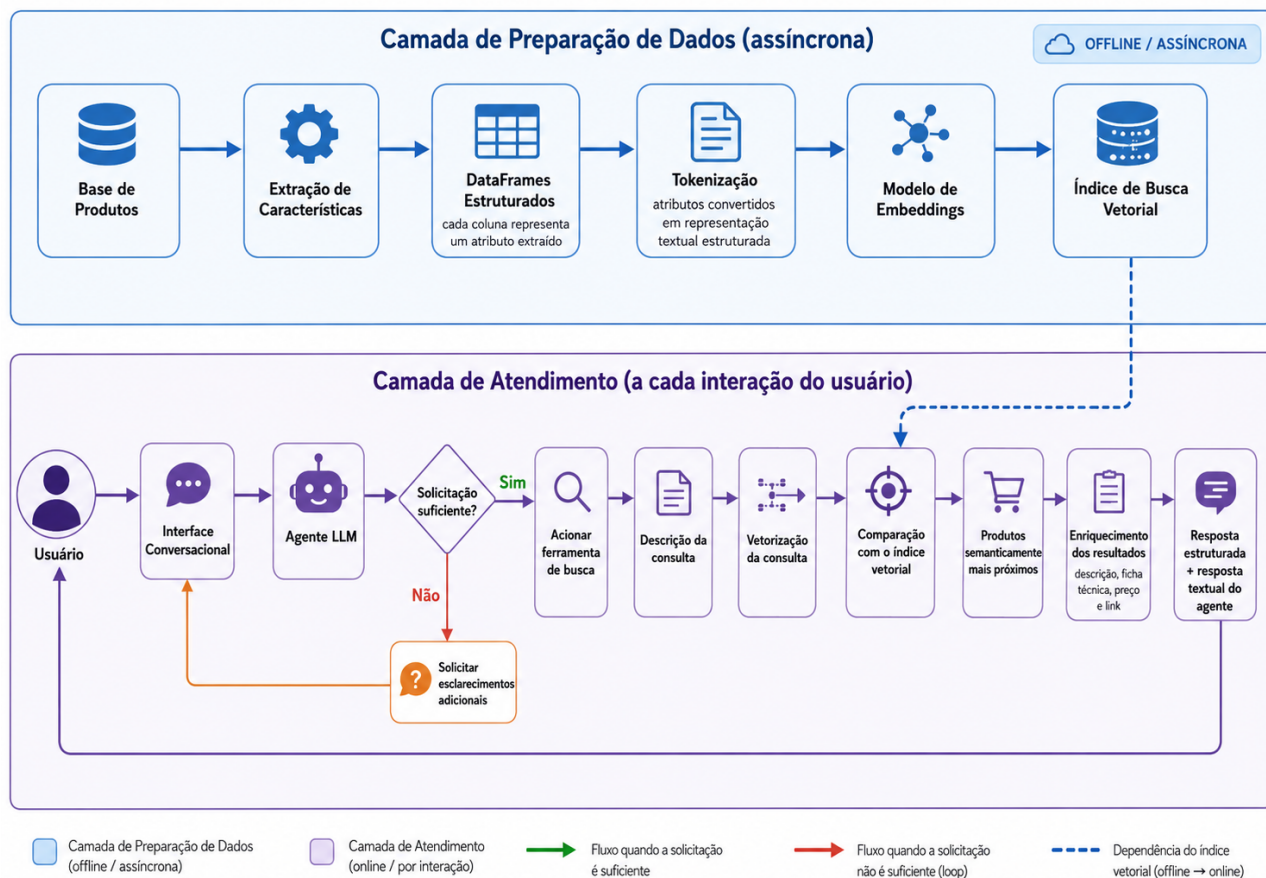
Metodologia

O sistema desenvolvido organiza-se em duas camadas operacionais distintas: uma camada de preparação de dados, executada de forma assíncrona, e uma camada de atendimento, acionada a cada interação do usuário.

Na camada de preparação, a base de produtos é submetida a uma extração de características, estas são estruturadas em *dataframes* onde cada coluna representa uma informação extraída, para em seguida passar por um processo de tokenização, no qual os atributos de cada produto são convertidos em uma representação textual estruturada. Essa representação é então vetorizada por um modelo de *embeddings* e armazenada em um índice de busca vetorial, que constitui a estrutura sobre a qual as consultas semânticas são realizadas.

Na camada de atendimento, a solicitação do usuário, expressa em linguagem natural, é recebida pela interface conversacional e encaminhada ao agente LLM. O agente avalia a solicitação e decide entre duas ações possíveis: solicitar esclarecimentos adicionais, quando a solicitação é considerada insuficiente, ou acionar a ferramenta de busca, quando há informação suficiente para uma consulta útil. Ao acionar a busca, o agente fornece uma descrição que é convertida em vetor e comparada ao índice, retornando os produtos semanticamente mais próximos. Esses produtos são então enriquecidos com informações complementares de descrição, ficha técnica, preço e link de compra, e então são apresentados ao usuário de forma estruturada, acompanhados de uma resposta textual elaborada pelo agente.

Figura 3.1: Arquitetura do sistema



Elaborado pelo autor.

Todo o desenvolvimento foi realizado no ambiente do Databricks, exceto pela interface gráfica, desenvolvida utilizando Python e Streamlit no ambiente VSCode.

3.1 Análise Exploratória dos Dados

A fase inicial do desenvolvimento compreendeu o levantamento da base de informações de produtos e a realização da análise exploratória dos dados. Todo o levantamento e análise foi realizado utilizando os Databricks Notebooks, ferramenta similar ao Jupyter Notebook, além de PySpark e SQL. Embora a pesquisa tenha se limitado a produtos atualmente comercializados, o elevado volume de dados exigiu o refinamento do escopo analítico, e a base foi delimitada a quatro categorias de produtos eletrônicos: televisores, *smartphones*, *tablets* e *notebooks*, totalizando um total 273 produtos, como visto na Tabela 3.1.

Tabela 3.1: Quantidade de produtos por categoria

Categoria	Quantidade de produtos
Televisores	70
<i>Smartphones</i>	143
<i>Tablets</i>	30
<i>Notebooks</i>	30

Cada produto presente na base original dispunha de um identificador único e de um conjunto de informações, incluindo uma descrição comercial. A partir dessa descrição, foram extraídas as características dos produtos, posteriormente organizadas em *dataframes* específicos para cada categoria, com seus respectivos atributos estruturados em colunas.

Figura 3.2: Exemplo de produto na base original

Δ_C^B id	Δ_C^B title
377081	Smartphone Samsung S25 FE 128 GB Azul Tela 6.7" Câmera 50MP 8GB RAM

Elaborado pelo autor.

Para realizar a extração, foi utilizada uma função SQL denominada `ai_extract`, oferecida pelo Databricks, devido a sua integração nativa e capacidade de extração baseada em LLM. Para orientar a extração, foi definida, para cada uma das quatro categorias de produtos contempladas, uma estrutura com os atributos e quais os valores esperados. A Figura 3.3 apresenta, a título de exemplo, a estrutura definida para a categoria de *smartphones*, contendo os atributos considerados relevantes para a recomendação nesse domínio. As estruturas correspondentes às demais categorias seguem padrão análogo e são apresentadas integralmente no Apêndice A.

Figura 3.3: Exemplo de extração de características

```

WHEN 'smartphone' THEN ai_extract(
  title,
  '{
    "marca": {
      "type": "string",
      "enum": ["Samsung", "Apple", "Motorola", "Xiaomi", "Redmi", "Poco", "Oppo", "Honor", "TCL", "Realme", "Infinix", "Jovi", "outro"],
      "description": "Trate Redmi e Poco como marcas separadas (sub-marcas Xiaomi).",
    },
    "armazenamento_gb": {
      "type": "integer",
      "description": "Armazenamento em GB. Converta 1TB para 1024.",
    },
    "ram_gb": {"type": "integer"},
    "cor_base": {
      "type": "string",
      "enum": ["preto", "branco", "azul", "verde", "vermelho", "rosa", "roxo", "lilas", "amarelo", "laranja", "cinza", "grafite", "prata", "dourado", "marrom", "bege",
      "marfim", "titânio", "outro"],
      "description": "Cor base, ignorando qualificadores como escuro, claro, nevoa.",
    },
    "tela_polegadas": {"type": "number"},
    "camera_principal_mp": {"type": "integer"}
  },
  )

```

Elaborado pelo autor.

Além das informações extraídas, foi adicionada uma coluna auxiliar contendo o preço do produto, proveniente de uma base de preços, com o objetivo de criar uma nova coluna referente à faixa de preço. As faixas foram definidas em intervalos apresentados na Tabela 3.2. Em seguida, a coluna de preço foi removida, permanecendo apenas a coluna com a respectiva faixa de preço. A decisão de considerar a faixa de preço, em vez do preço exato, será esclarecida na próxima seção. A Figura 3.4 mostra um exemplo das características extraídas para um produto da categoria de *smartphones*.

Tabela 3.2: Faixas de preço utilizadas na tokenização

Faixa de preço	Marcador
Preço menor que R\$ 500	ATE_500
R\$ 500 a R\$ 999	500_A_999
R\$ 1.000 a R\$ 1.999	1000_A_1999
R\$ 2.000 a R\$ 2.999	2000_A_2999
R\$ 3.000 a R\$ 4.999	3000_A_4999
R\$ 5.000 a R\$ 6.999	5000_A_6999
R\$ 7.000 a R\$ 8.999	7000_A_8999
Preço igual ou superior a R\$ 9.000	ACIMA_9000

Figura 3.4: Exemplo de características extraídas

$\mathbb{A}_C^B$ id	$\mathbb{A}_C^B$ faixa_preco	$\mathbb{A}_C^B$ categoria	$\mathbb{A}_C^B$ marca	$\mathbb{A}_C^B$ armazenamento_gb	$\mathbb{A}_C^B$ ram_gb	$\mathbb{A}_C^B$ cor_base	$\mathbb{A}_C^B$ tela_polegadas	$\mathbb{A}_C^B$ camera_principal_mp
377081	3000_A_4999	smartphone	Samsung	128	8	azul	6.7	50

Elaborado pelo autor.

3.1.1 Validação de Extração

Considerando que a extração automatizada de atributos via `ai_extract` constitui o insumo primário da representação tokenizada e, por consequência, do índice de busca semântica, foi conduzida uma validação amostral de sua qualidade. A motivação dessa validação decorre da observação de que erros de extração não detectados nessa etapa propagam-se silenciosamente para o índice, comprometendo resultados da etapa posterior.

A validação foi conduzida sobre uma amostra de 30 produtos sorteados aleatoriamente do catálogo, estratificada por categoria com 8 *smartphones*, 7 *tablets*, 8 *notebooks* e 7 televisores. Considerando que cada categoria contempla um conjunto distinto de atributos, a amostra resultou em 189 pares (produto, atributo) submetidos a conferência manual contra o conteúdo original das respectivas descrições comerciais. Cada par foi classificado em uma de quatro categorias: correto, quando o valor extraído correspondia ao valor declarado na descrição; incorreto, quando o valor extraído contradizia a descrição; não recuperado, quando o atributo constava na descrição mas não foi recuperado pela extração; e ausente legitimamente, quando o atributo não constava na descrição original, situação em que a função corretamente preencheu o campo com o marcador `DESCONHECIDO`.

A Tabela 3.3 apresenta a distribuição dos resultados por categoria.

Tabela 3.3: Resultados da extração por categoria

Categoria	Pares avaliados	Corretos	Ausentes legítimos	Erros	Acurácia
Smartphone	48	44	4	0	100%
Tablet	42	39	3	0	100%
Notebook	64	45	19	0	100%
TV	35	28	7	0	100%
Total	189	156	33	0	100%

A inspeção identificou, na amostra avaliada, ausência de erros das categorias incorreto e não recuperado, visto que todos os atributos efetivamente declarados nas descrições originais foram corretamente extraídos pela função, e todos os campos preenchidos com `DESCONHECIDO` correspondiam a atributos legitimamente ausentes da descrição original. Esse resultado, contudo, deve ser interpretado com cautela, pois a avaliação foi conduzida sobre uma amostra

parcial dos dados, o que não permite estabelecer, com rigor estatístico, a acurácia da função sobre o catálogo completo. Ainda assim, os achados constituem uma indicação consistente de adequação da extração, embora não representem uma demonstração definitiva de acurácia.

3.2 Tokenização

Diante da necessidade de viabilizar busca semântica sobre a base de produtos, optou-se por elaborar uma representação textual estruturada, onde as propriedades de cada produto são representadas como uma sequência de *tokens* semânticos padronizados. Essa abordagem foi selecionada por permitir uma representação textual padronizada dos atributos de produtos, fazendo com que as características relevantes sejam priorizadas. Essa forma mostrou-se ideal para indexação vetorial, visto que a similaridade é calculada a partir das propriedades dos produtos, e não descrições gerais.

A partir dessa ideia, cada produto presente na base foi representado como uma sequência de *tokens*, onde cada *token* traduz uma característica do produto. Foi definido um conjunto consistente de marcadores, para que os produtos com mesmas propriedades sejam sempre codificados de forma idêntica. Isso foi feito para que a similaridade vetorial realmente reflita a similaridade entre os produtos. Entre as informações transformadas em *tokens* estão atributos comuns a todos os produtos, como categoria, subcategoria, marca e faixa de preço, além de atributos específicos de cada categoria. É válido mencionar que a representação tokenizada foi concebida em português brasileiro, refletindo o vocabulário do catálogo e as convenções terminológicas brasileiras (por exemplo, “armazenamento” em vez de “*storage*”, “celular” como sinônimo coloquial de “*smartphone*”). A escolha do idioma para a representação tem implicações técnicas relevantes, dado que o modelo de embeddings utilizado foi treinado predominantemente em língua inglesa. As Tabelas 3.4, 3.5, 3.6 e 3.7, descrevem as características dos produtos e seus respectivos marcadores, para cada categoria.

Tabela 3.4: Descrição das colunas para a categoria de televisores

Característica	Marcador	Descrição
categoria	CAT	A categoria do produto
subcategoria	SUBCAT	A subcategoria do produto
marca	MARCA	A marca do produto
tamanho_polegadas	TELA	O tamanho da tela em polegadas
resolucao	RESOLUCAO	A resolução da tela
tecnologia_tela	TECNOLOGIA_TELA	A tecnologia utilizada na tela
sistema_operacional	SISTEMA	O sistema operacional quando Smart TV
faixa_preco	FAIXA_PRECO	A faixa de preço do produto

Tabela 3.5: Descrição das colunas para a categoria de *smartphones*

Característica	Marcador	Descrição
categoria	CAT	A categoria do produto
subcategoria	SUBCAT	A subcategoria do produto
marca	MARCA	A marca do produto
armazenamento_gb	ARMAZENAMENTO	A quantidade de memória, em GB, de armazenamento interno
ram_gb	RAM	A quantidade de memória RAM, em GB
cor_base	COR	A cor do produto
tela_polegadas	TELA	O tamanho da tela em polegadas
camera_principal_mp	CAMERA	Os megapixels da câmera traseira
faixa_preco	FAIXA_PRECO	A faixa de preço do produto

Tabela 3.6: Descrição das colunas para a categoria de tablets

Característica	Marcador	Descrição
categoria	CAT	A categoria do produto
subcategoria	SUBCAT	A subcategoria do produto
marca	MARCA	A marca do produto
armazenamento_gb	ARMAZENAMENTO	A quantidade de memória, em GB, de armazenamento interno
ram_gb	RAM	A quantidade de memória RAM, em GB
cor_base	COR	A cor do produto
tela_polegadas	TELA	O tamanho da tela em polegadas
sistema_operacional	SISTEMA	O sistema operacional
faixa_preco	FAIXA_PRECO	A faixa de preço do produto

Tabela 3.7: Descrição das colunas para a categoria de notebooks

Característica	Marcador	Descrição
categoria	CAT	A categoria do produto
subcategoria	SUBCAT	A subcategoria do produto
marca	MARCA	A marca do produto
processador_marca	PROCESSADOR_MARCA	A marca do processador
processador_familia	PROCESSADOR_FAMILIA	A família do processador
armazenamento_gb	ARMAZENAMENTO	A quantidade de memória, em GB, de armazenamento interno
ram_gb	RAM	A quantidade de memória RAM, em GB
tela_polegadas	TELA	O tamanho da tela em polegadas
gpu_marca	GPU_MARCA	O marca da placa de vídeo
gpu_modelo	GPU_MODELO	O modelo da placa de vídeo
faixa_preco	FAIXA_PRECO	A faixa de preço do produto

Durante o desenvolvimento, foram realizadas duas alterações no processo de tokenização em relação à abordagem original. A primeira consistiu em substituir o preço exato do produto por uma faixa de preço, visando aprimorar a generalização semântica na busca vetorial, uma vez que solicitações de usuários tendem a expressar preço de forma aproximada e por faixas, não por valores exatos. Vale registrar, contudo, que essa decisão implica um descolamento entre a representação utilizada na busca e o valor de preço atual de cada produto. O preço exato é mantido na tabela enriquecida e é recuperado durante a apresentação dos resultados ao usuário, de forma que o valor visualizado pelo consumidor reflete a informação corrente do catálogo. A representação por faixas, portanto, opera como filtro semântico durante a recuperação, sem comprometer a exibição do preço real. A segunda envolveu o tratamento de valores nulos com o marcador específico `DESCONHECIDO`, preservando a estrutura da representação e evitando que a ausência de informações fosse interpretada de forma equivocada pelo modelo de *embedding*. Ao final da representação também foi incluído o marcador `SEP`, apenas para indicar o fim da tokenização.

Por fim, foi criada uma tabela final unindo as quatro categorias, composta pelas colunas de identificador do produto e sua representação tokenizada. A Figura 3.5 apresenta um exemplo de tokenização para um produto da categoria de *smartphone*.

Figura 3.5: Exemplo de produto tokenizado

Δ_C^B id	Δ_C^B tokens
377081	[CAT_ELETRONICOS][SUBCAT_SMARTPHONE][MARCA_SAMSUNG][ARMAZENAMENTO_128GB][RAM_8GB][COR_AZUL] [TELA_6_7_POLEGADAS][CAMERA_50MP][FAIXA_PRECO_3000_A_4999] [SEP]

Elaborado pelo autor.

3.3 Busca Vetorial

Com a representação dos produtos em *tokens* concluída e armazenada, a etapa seguinte consistiu em viabilizar a busca semântica sobre essas informações, utilizando busca vetorial. Para isso, foram necessários dois pilares fundamentais: o modelo de *embeddings*, responsável por converter as representações textuais dos produtos em vetores, e o índice vetorial, encarregado de armazenar esses vetores e possibilitar consultas por similaridade. Para atender aos requisitos citados, optou-se pela utilização do Databricks Vector Search (DATABRICKS, 2026a), que fornece integração direta com a tabela de dados tokenizados e permite configurar de forma simplificada o modelo de *embedding* a ser utilizado.

A configuração do índice vetorial utilizou o modo Delta Sync Index, que mantém o índice sincronizado com as alterações realizadas na tabela de origem. A sincronização do índice foi configurada em modo `triggered`, no qual a atualização do índice é disparada manualmente, e não em modo `continuous`, no qual o índice se atualiza automaticamente a cada alteração na tabela de origem. Essa escolha justifica-se em razão de dois fatores. Em primeiro lugar, o catálogo de produtos sobre o qual o trabalho foi conduzido não sofre alterações em frequência elevada que justifique a sincronização contínua, cujo custo computacional é maior. Em segundo lugar, o controle explícito sobre o ciclo de atualização permite garantir consistência entre a representação tokenizada armazenada na tabela base e o estado do índice durante os experimentos. Reconhece-se, contudo, que cenários de operação real demandariam a adoção do modo `continuous` ou de mecanismos de invalidação seletiva.

O modelo de *embedding* selecionado foi o `text-embedding-ada-002` (DATABRICKS, 2026c), escolha baseada principalmente pelo fato do modelo ser disponibilizado nativamente

pelo Databricks, o que facilita sua integração com a infraestrutura do projeto. Apesar disso, é importante considerar que se trata de um modelo lançado em 2022 e pertencente a uma geração anterior de modelos de *embedding*. Além disso, por não ser um modelo especificamente treinado ou ajustado para o português brasileiro, seu uso pode apresentar limitações na captura de relações semânticas mais sutis no contexto da representação adotada. Nesse sentido, a utilização de modelos multilíngues mais recentes ou de modelos treinados especificamente para o português brasileiro constitui uma possibilidade de melhoria futura.

Para realizar a pesquisa de similaridade, foi utilizada a função `similarity_search` nativa do Databricks. Essa função calcula a similaridade entre a consulta e os itens indexados a partir da relação apresentada na Equação 3.1.

$$\text{sim}(q, x) = \frac{1}{(1 + \text{dist}(q, x)^2)} \quad (3.1)$$

Nessa formulação, q representa o vetor da consulta, x representa o vetor da entrada armazenada no índice, e $\text{dist}(q, x)$ corresponde à distância euclidiana entre esses dois vetores, conforme descrito na Equação 3.2.

$$\text{dist}(q, x) = \sqrt{(q_1 - x_1)^2 + (q_2 - x_2)^2 + \dots + (q_d - x_d)^2} \quad (3.2)$$

A validação inicial do índice foi conduzida por meio de consultas qualitativas representativas das quatro categorias do escopo. Para cada consulta, foram inspecionados os cinco primeiros resultados quanto à aderência à solicitação. A validação indicou que o mecanismo retornava produtos consistentes com os critérios das consulta.

Por fim, foi gerado um *endpoint* de acesso aos dados do índice, para futuramente ser utilizado pelo agente LLM, descrito na próxima sessão.

3.4 Agente LLM

A etapa posterior engloba o desenvolvimento do agente, que teve como função interpretar solicitações do usuário, decidir entre invocar a busca ou solicitar esclarecimentos, e formular a

resposta final a partir dos resultados obtidos. A construção utilizou o *framework* LangChain (LANGCHAIN, 2026a), escolhido pela consolidação da ferramenta, facilidade na criação de agentes baseados em uso de ferramentas, flexibilidade na configuração do *prompt* e do fluxo de raciocínio e sua integração com o Databricks.

O modelo de linguagem adotado como núcleo do agente foi o GPT-4o-mini (DATA-BRICKS, 2026b), acessado via API da OpenAI através do Databricks. A escolha desse modelo fundamentou-se pelo desempenho consolidado em tarefas de raciocínio sobre instruções e de uso de ferramentas, a disponibilidade direta no momento de desenvolvimento, com integração simplificada à infraestrutura do projeto, e o custo de invocação compatível com os volumes de execução previstos para o trabalho. Sabe-se que o modelo selecionado, na data de conclusão do trabalho, já não corresponde à geração mais recente disponibilizada pela OpenAI, e a substituição por modelos mais recentes constitui uma das direções para trabalhos futuros.

A temperatura do modelo foi configurada como igual a zero, com o objetivo de aproximar o comportamento do agente do regime determinístico.

3.4.1 Ferramentas do Agente

Inicialmente foi criada apenas uma ferramenta para o agente, porém, após alguns testes, foi decidido criar mais uma. A primeira denominada de **BuscarProdutos**, consiste em executar a busca semântica no índice vetorial criado anteriormente e retornar os produtos correspondentes à solicitação. A segunda, denominada de **PedirClarificacao**, é acionada somente quando a solicitação do usuário é considerada insuficiente para realizar uma busca efetiva, indicando ao agente a necessidade de interagir com o usuário para coletar informações adicionais. As ferramentas ocasionaram desafios durante o desenvolvimento, especialmente a segunda, gerando o problema de *over-elicitation*, cujo tratamento será detalhado em breve.

```
def construir_ferramentas() -> list:
    """Retorna a lista de ferramentas para o agente."""
    return [
        Tool(
            name="BuscarProdutos",
            func=buscar_e_apresentar,
            description=(
                "FERRAMENTA PRINCIPAL -- use sempre que possível. "
                "Busca produtos no catálogo a partir de uma descrição. "
                "Acione esta ferramenta sempre que o usuário mencionar uma categoria de "
                "produto E qualquer atributo (marca, especificação técnica, preço ou "
                "característica). A entrada deve ser uma descrição direta com todos os "
                "atributos mencionados. Exemplo: 'smartphone Samsung 8GB RAM'."
            )
        ),
        Tool(
            name="PedirClarificacao",
            func=pedir_clarificacao,
            description=(
                "FERRAMENTA DE ÚLTIMO RECURSO -- use apenas quando a busca for impossível. "
                "Acione SOMENTE se faltar a categoria do produto, ou se a solicitação tiver "
                "APENAS qualificadores subjetivos sem nenhum atributo objetivo. "
                "NÃO use se o usuário já forneceu categoria + qualquer atributo. "
                "NÃO use duas vezes em sequência na mesma conversa."
            )
        )
    ]
```

Código 3.1: Ferramentas passadas ao agente

BuscarProdutos

A ferramenta **BuscarProdutos** internamente executa um processo de três etapas. Primeiramente, recebe a descrição textual fornecida pelo agente e a submete ao índice vetorial, recuperando um conjunto de produtos candidatos. Em seguida, enriquece os candidatos com informações adicionais da tabela auxiliar com informações adicionais de produtos SQL parametrizada. Por fim, processa o conteúdo extraído, para exibir de forma estruturada o título, a descrição comercial, eventual conteúdo multimídia e a ficha técnica de cada produto, retornando ao agente uma versão textual enxuta e mantendo a versão rica em memória de sessão para apresentação posterior na interface. É importante frisar que as informações adicionais de produtos são utilizadas somente para a apresentação ao usuário.

A separação entre a versão enxuta retornada ao agente e a versão rica destinada à apresentação visual atende a duas preocupações distintas, visto que a versão enxuta limita o consumo de *tokens* do modelo de linguagem, reduzindo custo e latência, e contém apenas as informações relevantes para o raciocínio do agente sobre as recomendações, já a versão rica preserva todo o

conteúdo necessário para uma apresentação informativa ao usuário.

```
def buscar_e_apresentar(query: str, num_results: int = 5) -> str:
    """
    Busca produtos no catálogo, enriquece com dados completos,
    armazena no estado da sessão para renderização e retorna versão
    enxuta para o agente raciocinar.
    """
    resultados = _index.similarity_search(
        query_text=query,
        columns=["id", "tokens"],
        num_results=num_results
    )
    data_array = resultados.get("result", {}).get("data_array", [])

    if not data_array:
        return "Nenhum produto encontrado para essa busca."

    ids = [linha[0] for linha in data_array]
    enriquecidos = enriquecer_resultados(ids)

    produtos_para_exibir = []
    resumo_para_agente = []

    for i, linha in enumerate(data_array, 1):
        id, tokens, score = linha[0], linha[1], linha[2]
        info = enriquecidos.get(id, {})
        parseado = parsear_summary(info.get("summary", ""))

        produtos_para_exibir.append({
            "id": id,
            "score": score,
            "image_link": info.get("image_link"),
            "price": info.get("price"),
            "link": info.get("link"),
            "parseado": parseado
        })

        ficha = parseado.get("ficha_tecnica", {})
        atributos_chave = [
            "Marca",
            "Modelo",
            "Memória RAM",
            "Armazenamento interno",
            "Tamanho da Tela"
        ]
        ficha_resumida = {
            k: ficha[k] for k in atributos_chave if k in ficha
        }

        preco = info.get("price", "preço não disponível")

        resumo_para_agente.append(
            f"Produto {i} (ID: {id}, similaridade: {score:.2f}): "
            f"{parseado.get('titulo', 'sem título')} | "
            f"Preço: {preco} | {ficha_resumida}"
        )

    return "\n".join(resumo_para_agente)
```

Código 3.2: Definição da função utilizada em BuscarProdutos

PedirClarificacao

A ferramenta `PedirClarificacao` é utilizada para tratar solicitações vagas como “gostaria de um bom notebook para estudar”, onde os atributos para uma busca específica são insuficientes para retornar um resultado de qualidade aceitável. Em tais casos, em vez de produzir recomendações pouco aderentes ou inferir critérios arbitrários, o sistema deve solicitar esclarecimentos ao usuário, conduzindo o diálogo de forma a obter as informações necessárias. Para tratar esses tipos de solicitações, foi implementada uma estratégia de elicitación ativa de preferências, na qual o agente, ao identificar uma solicitação incompleta, aciona a ferramenta `PedirClarificacao` e formula, em seguida, perguntas específicas ao usuário sobre os atributos que auxiliam uma busca mais eficaz.

A decisão entre acionar a busca diretamente ou solicitar clarificação é tomada pelo agente com base em critérios estabelecidos no *prompt* de sistema. Foram definidos como solicitações completas aquelas que contêm uma categoria de produto e, no mínimo, um atributo discriminante objetivo como marca, especificação técnica, faixa de preço ou característica funcional. Foram definidas como solicitações vagas aquelas que carecem de categoria, contêm apenas qualificadores subjetivos como “um bom celular” ou mencionam apenas finalidade sem critérios técnicos como “um notebook para estudar”.

```
def pedir_clarificacao(motivo: str) -> str:
    """
    Sinaliza que a solicitação precisa de clarificação antes da busca.
    """
    return (
        f"CLARIFICACAO_NECESSARIA: {motivo}. "
        "Formule perguntas específicas ao usuário para obter os critérios faltantes."
    )
```

Código 3.3: Definição função utilizada em `PedirClarificacao`

3.4.2 Prompt do Sistema

A elaboração do *prompt* ocorreu por meio de um processo iterativo, conduzido em conjunto com a fase de avaliação experimental, e revelou-se um dos pontos de maior impacto durante o desenvolvimento. A versão final do *prompt*, apresentado no Código 3.4, foi construída com

os objetivos de fornecer clareza nos critérios de seleção das ferramentas, prevenção de comportamentos indesejados (invenção de produtos não encontrados na busca) e robustez diante de solicitações ambíguas.

```
PROMPT_SISTEMA = """Você é um assistente especializado em recomendação de produtos do catálogo fornecido.

Seu papel é ajudar usuários a encontrar produtos adequados às suas necessidades.

IMPORTANTE: Você deve responder apenas sobre recomendações e produtos. DE MANEIRA ALGUMA FALE SOBRE OUTROS
↪ ASSUNTOS. Caso o usuário deseje abordar outros assuntos fora recomendação de produtos, seja cordial e
↪ gentilmente diga que é incapaz de debater outros assuntos além de recomendações.

REGRA FUNDAMENTAL: Sua tendência deve ser BUSCAR, não perguntar. Use a ferramenta PedirClarificacao apenas em
↪ último caso, quando for genuinamente impossível fazer uma busca útil.

CRITÉRIO DE DECISÃO -- siga RIGOROSAMENTE esta lógica:

ACIONE BuscarProdutos imediatamente se a solicitação contém:
- Uma CATEGORIA de produto (smartphone, notebook, TV, fone, etc.) E
- PELO MENOS UM dos seguintes:
  • Uma marca específica (Samsung, Motorola, LG, etc.)
  • Uma especificação técnica (8GB RAM, 256GB, 4K, 55 polegadas, etc.)
  • Uma faixa de preço (até X reais, abaixo de Y, premium, intermediário)
  • Uma característica funcional objetiva (com 5G, dual chip, gamer, etc.)

Exemplos que DEVEM acionar BuscarProdutos diretamente:
- "smartphone de 8GB de RAM" -> tem categoria + especificação
- "notebook Dell" -> tem categoria + marca
- "TV até 3 mil reais" -> tem categoria + preço
- "smartphone Samsung 8GB" -> tem todos os três, MUITO mais que o necessário

ACIONE PedirClarificacao APENAS quando:
- Falta a categoria do produto ("quero algo bom", "me recomenda alguma coisa")
- A solicitação tem APENAS qualificadores subjetivos sem nenhum atributo objetivo
  ("um celular bom", "um notebook legal")
- Menciona apenas finalidade pura sem qualquer critério ("para estudar", "para trabalhar")

REGRA DE NÃO-INSISTÊNCIA:
Se você JÁ pediu clarificação uma vez e o usuário respondeu fornecendo qualquer
informação adicional, ACIONE BuscarProdutos com o que tem. Não peça clarificação
duas vezes seguidas. É melhor buscar com informação parcial e refinar depois do
que frustrar o usuário com perguntas em série.

APRESENTAÇÃO DOS RESULTADOS:
- Traduza tokens para linguagem natural (ex.: RAM_8GB -> "8GB de RAM")
- NUNCA invente produtos que não estejam nos resultados da ferramenta
- Após apresentar os produtos, pode oferecer refinar a busca se o usuário desejar

NÃO acione ferramenta para cumprimentos, agradecimentos ou perguntas sobre o sistema."""
```

Código 3.4: Prompt de sistema utilizado pelo agente

Durante o desenvolvimento do *prompt*, o uso de exemplos no critério de uso das ferramentas mostrou-se fundamental para que o agente funcionasse como o desejado. Em lugar de descrever apenas em termos abstratos quando cada ferramenta deve ser acionada, o *prompt* apresenta exemplos de solicitações que se enquadram em cada categoria, com a justificativa correspondente. Essa decisão aumentou de forma considerável as decisões do agente no uso das

ferramentas.

Além disso, a instrução contra a invenção de produtos também atuou de maneira significativa para que não houvesse alucinação por parte do modelo, já que o *prompt* determina explicitamente que o agente jamais sugira produtos que não estejam entre os resultados retornados pela ferramenta de busca.

3.4.3 Problema de *over-elicitation*

A versão inicial do mecanismo de elicitação apresentou um comportamento indesejado, identificado durante a fase de avaliação experimental, pois o agente tendia a solicitar clarificações de forma excessiva, mesmo após o usuário ter fornecido informações suficientes para uma busca útil. Esse comportamento manifestava-se de duas formas principais.

A primeira forma era o acionamento da clarificação em solicitações já adequadas. Solicitações como “smartphone de 8GB de RAM”, que satisfaziam o critério formal de completude (categoria + especificação técnica), eram interpretadas pelo agente como insuficientes, motivando perguntas adicionais sobre preço, marca e outras especificações.

A segunda forma, mais grave, era o acionamento sucessivo da clarificação em turnos consecutivos da conversa. Após uma primeira rodada de perguntas e respostas, o agente, ao receber a resposta do usuário, em vez de proceder à busca com as informações então disponíveis, formulava nova rodada de perguntas, configurando um ciclo de elicitação potencialmente infinito. A mitigação do problema foi conduzida por meio de três ajustes complementares no *prompt* e nas definições das ferramentas.

O primeiro ajuste consistiu na reformulação do critério de completude com ênfase explícita na disposição de buscar. O *prompt* passou a estabelecer, como diretriz fundamental, que a tendência do agente deve ser proceder à busca, recorrendo à clarificação apenas em último caso. A transição de uma postura baseada em “verificar se há informação suficiente” para outra orientada por “buscar a menos que seja impossível” alterou substancialmente o comportamento decisório do agente.

O segundo ajuste consistiu no refinamento das descrições das ferramentas, com explicitação

textual de que `BuscarProdutos` é a ferramenta principal, a ser utilizada sempre que possível, enquanto `PedirClarificacao` é uma ferramenta de último recurso, restrita aos casos em que a busca seria efetivamente improdutiva.

O terceiro ajuste consistiu na inclusão de uma regra de não-insistência, que estabelece explicitamente que o agente não deve acionar a clarificação duas vezes em turnos consecutivos da conversa. Após uma rodada de perguntas, qualquer resposta do usuário deve ser interpretada como informação suficiente para proceder à busca, ainda que parcial. Essa regra atua como salvaguarda contra o ciclo de elicitación descrito anteriormente.

3.4.4 Memória Conversacional

Para que o agente tivesse a capacidade de manter o contexto durante diferentes turnos de interação, foi adicionado ao mesmo o mecanismo de memória conversacional, através do componente `RunnableWithMessageHistory` do `LangChain` (LANGCHAIN, 2026b).

A solução adotada armazena, para cada sessão de uso, o histórico completo das mensagens trocadas entre usuário e agente, incluindo as invocações de ferramentas e seus retornos. Esse histórico é incorporado ao contexto enviado ao modelo de linguagem a cada nova interação, permitindo que o agente recupere informações fornecidas em turnos anteriores. Assim, quando um usuário responde a uma pergunta de clarificação, o agente combina a nova informação com aquela já presente na solicitação original para compor a consulta à ferramenta de busca.

3.5 Interface Conversacional

A interface entre o usuário e o sistema foi desenvolvida com o *framework* Streamlit (STREAMLIT, 2026), escolhido pela sua oferta nativa de componentes específicos para aplicações conversacionais, a integração direta com o ecossistema Python utilizado no restante do projeto.

A interface organiza-se em três áreas funcionais como mostra a Figura 3.6. Uma área lateral contém informações sobre o sistema, controles auxiliares e indicadores estatísticos da sessão em curso. Uma área central exibe o histórico da conversa, alternando mensagens do usuário e

respostas do agente. Uma área inferior, sempre acessível, contém o campo de entrada de novas mensagens.

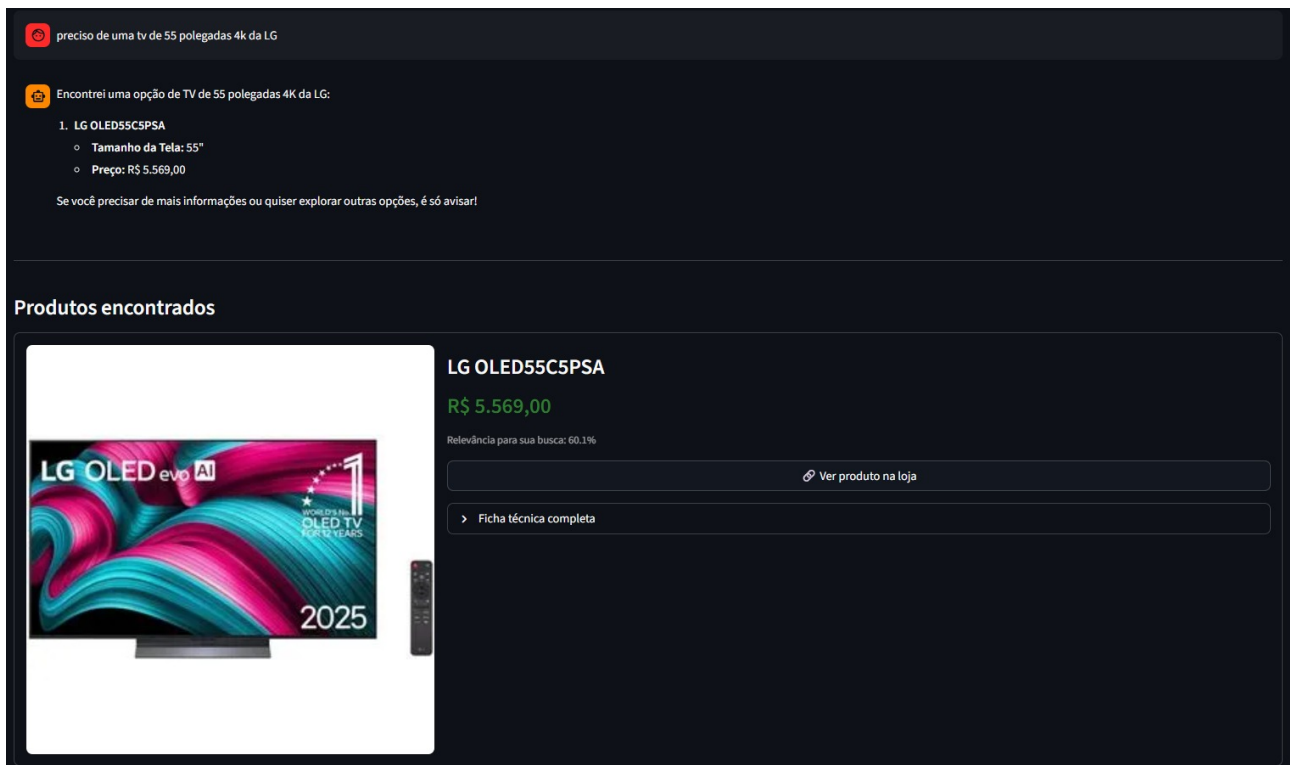
Figura 3.6: Interface inicial



Elaborado pelo autor.

Optou-se por uma apresentação em formato de *cards* individuais, cada *card* organiza-se em duas colunas. À esquerda, a imagem principal do produto. À direita, na seguinte ordem hierárquica: o título do produto, o preço em destaque visual, um botão de acesso direto ao produto na plataforma comercial e três blocos expansíveis contendo, respectivamente, a descrição comercial completa quando disponível, eventual conteúdo multimídia e a ficha técnica detalhada. A Figura 3.7 apresenta um exemplo dessa renderização.

Figura 3.7: Exemplo de apresentação de recomendação



Elaborado pelo autor.

A integração entre a interface e o agente segue um padrão de troca de mensagens. A cada nova mensagem do usuário, a interface aciona o agente passando como argumentos o texto da mensagem e um identificador de sessão, utilizado pelo mecanismo de memória conversacional para recuperar o contexto. A resposta do agente é exibida na área central da conversa.

Os produtos eventualmente retornados pela ferramenta de busca não trafegam pela resposta textual do agente, mas são depositados no estado de sessão da interface durante a execução da ferramenta. Após o término da invocação do agente, a interface inspeciona o estado de sessão e, havendo produtos armazenados, os renderiza no formato de *cards* descrito anteriormente. Esse desacoplamento entre resposta textual e apresentação visual permite que o agente comente sobre os produtos em linguagem natural, enquanto a interface exibe seus detalhes de forma estruturada.

Capítulo 4

Resultados e Discussão

A avaliação experimental do sistema desenvolvido foi conduzida sobre um conjunto de 15 casos de teste, elaborados para abranger diferentes tipos de solicitações que o agente está sujeito a receber em uso real. Os casos foram organizados em cinco categorias, com três a quatro instâncias cada: solicitações específicas, contendo categoria de produto e ao menos um atributo discriminante; solicitações parcialmente vagas, contendo categoria mas sem atributos suficientes para uma busca útil; solicitações vagas, sem categoria identificável; solicitações com linguagem informal, em que a categoria e os atributos são expressos de forma coloquial ou abreviada; e solicitações fora do domínio, que não dizem respeito a recomendação de produtos. A relação completa dos casos é apresentada no Apêndice B.

Cada caso foi executado uma única vez, em sessão isolada, onde o histórico conversacional do agente é reiniciado após cada execução, para garantir que o comportamento observado em um caso não influenciasse os subsequentes. Para os casos parcialmente vagos e vagos, em que o comportamento esperado envolvia clarificação seguida de busca, a execução incluiu um segundo turno, no qual o usuário fornecia critérios adicionais em resposta às perguntas do agente. Os demais casos foram executados em turno único.

A avaliação dos resultados de cada execução considerou quatro dimensões. A adequação da ferramenta acionada verifica se o agente acionou a ferramenta correta para o tipo de solicitação, sendo `BuscarProdutos` para solicitações específicas e informais com atributos suficientes, `PedirClarificacao` para solicitações parcialmente vagas e vagas, e nenhuma ferramenta para

solicitações fora do domínio. A preservação de atributos verifica, nos casos em que houve busca, se o agente preservou na consulta os atributos mencionados pelo usuário. A adequação da elicitación, aplicável aos casos em que houve clarificação, verifica se as perguntas formuladas pelo agente foram pertinentes ao domínio e ao tipo de informação faltante. A ocorrência de alucinação, aplicável a todos os casos, verifica se a resposta do agente mencionou produtos, atributos ou preços inexistentes ou não retornados pela ferramenta.

4.1 Resultados Quantitativos

Os resultados agregados da avaliação são apresentados na Tabela 4.1, organizada pelas quatro dimensões consideradas.

Tabela 4.1: Resultados da avaliação por dimensão

Dimensão	Casos com sucesso	Casos totais	Resultado
Adequação da ferramenta acionada	15	15	100%
Preservação de atributos (casos com busca)	7	7	100%
Adequação da elicitación (casos com clarificação)	6	6	100%
Ocorrência de alucinação	0	15	0%

A análise dos dados indica, dentro do escopo avaliado, comportamento do agente consistente com as expectativas estabelecidas pelo critério operacional. Nos 15 casos avaliados, o agente acionou em cada um a ferramenta esperada conforme o critério operacional definido para o tipo de solicitação correspondente: invocou diretamente **BuscarProdutos** nas solicitações em que havia categoria e atributos discriminantes suficientes; invocou **PedirClarificacao** nas solicitações vagas ou parcialmente vagas; e absteve-se de acionar qualquer ferramenta nas solicitações fora do domínio, fornecendo respostas conversacionais ou recusando educadamente o atendimento, conforme o caso.

Nos sete casos em que o agente decidiu pela busca, observou-se que os atributos mencionados pelo usuário foram preservados na consulta enviada à ferramenta, até mesmo incluindo atributos expressos em linguagem informal, como “cel” para *smartphone*, “4k” como abreviação de 4.000 reais, e “8gb” sem padronização para *gigabytes* de memória RAM. Em todos os sete

casos com busca, os atributos foram corretamente identificados e incorporados à consulta. O comportamento adequado do agente sobre essas entradas indica que a fluência linguística do modelo de linguagem subjacente é suficiente para o português brasileiro, dentro do escopo dos casos avaliados, apesar da assimetria de treinamento do modelo em favor do inglês.

Nos casos em que o agente optou pela clarificação, as perguntas formuladas foram consideradas pertinentes ao domínio do produto e ao tipo de informação faltante. Em solicitações parcialmente vagas, onde a categoria estava presente mas faltavam critérios discriminantes, as perguntas concentraram-se em obter especificações técnicas, faixa de preço ou marca preferida. Em solicitações totalmente vagas, as perguntas começaram pela própria categoria de produto desejada, prosseguindo apenas após essa definição.

A ausência de alucinação foi observada nos 15 casos avaliados. Em nenhuma resposta, o agente mencionou produtos não retornados pela ferramenta de busca, atributos não constantes nos resultados ou preços inexistentes. Cabe ressaltar que esse achado, baseado em conjunto restrito de casos, deve ser interpretado como indicação preliminar do comportamento factual do sistema, não como demonstração definitiva de imunidade à alucinação.

A análise cruzada por categoria de caso, apresentada na Tabela 4.2, permite identificar o comportamento do sistema em cada tipo de solicitação.

Tabela 4.2: Comportamento esperado e observado por categoria

Categoria	Casos	Comportamento esperado	Comportamento observado
Específico	4	BuscarProdutos	4 de 4 buscaram
Informal	2	BuscarProdutos	2 de 2 buscaram
Parcialmente vago	4	PedirClarificacao	4 de 4 clarificaram
Vago	2	PedirClarificacao	2 de 2 clarificaram
Fora do domínio	3	Nenhuma ferramenta	3 de 3 não acionaram

4.2 Resultados Qualitativos

Alguns casos merecem uma discussão qualitativa por evidenciarem aspectos relevantes para a compreensão do funcionamento do agente., os mesmos serão apresentados a seguir.

A solicitação E12 “quero um fone de até 300 reais” caracterizou-se por requisitar uma catego-

ria de produto não contemplada no escopo do catálogo avaliado, que se restringe a televisores, *smartphones*, *tablets* e *notebooks*. O agente acionou corretamente a ferramenta **BuscarProdutos**, já que a solicitação contém categoria e atributo discriminante de preço, mas o índice vetorial, na ausência de fones de ouvido no catálogo, retornou produtos semanticamente próximos mas não pertinentes à solicitação.

O comportamento subsequente do agente é particularmente relevante. Em vez de apresentar os produtos recuperados como se fossem aderentes, ou de inventar produtos para satisfazer a solicitação, o agente reconheceu explicitamente a não-aderência dos resultados, respondendo ao usuário: “Parece que não consegui encontrar opções de fones de ouvido dentro da faixa de até 300 reais. Os resultados que obtive estão relacionados a smartphones e outros produtos.”.

Esse comportamento ilustra, em um caso concreto, o argumento teórico discutido anteriormente sobre como ferramentas com acesso a dados externos podem auxiliar o agente a não alucinar. Diante da limitação real do catálogo, o agente preferiu informar o usuário a fabricar uma resposta plausível, o que em uma LLM operando isoladamente sobre solicitação semelhante poderia resultar na sugestão de produtos inexistentes com características inventadas.

Além disso, As três solicitações fora do domínio (“olá, tudo bem?”, “como funciona a teoria da relatividade?”, “você é uma inteligência artificial?”) sugerem um aspecto adicional do comportamento do agente, que é a delimitação do escopo conversacional. Em todos os três casos, o agente absteve-se de acionar qualquer ferramenta, conforme esperado, e produziu respostas distintas conforme a natureza da solicitação.

À saudação inicial E13, o agente respondeu cordialmente e ofereceu auxílio com recomendações de produtos, mantendo o tom conversacional sem perder o foco do domínio. Às solicitações de cunho informacional fora do escopo E14 e E15, o agente recusou educadamente o atendimento, indicando seu propósito específico: “Desculpe, mas não posso debater sobre esse assunto. Posso ajudar você a encontrar produtos.”.

Esse comportamento destaca que o prompt de sistema cumpre não apenas a função de orientar a seleção entre ferramentas, mas também a de delimitar o escopo conversacional do agente, restringindo-o ao domínio de recomendação de produtos.

4.3 Desempenho Operacional

A avaliação do desempenho operacional do sistema foi conduzida em paralelo à avaliação comportamental apresentada nas seções anteriores, com o objetivo de caracterizar quantitativamente o consumo de recursos pelo sistema sob carga representativa.

Cada um dos 15 casos de teste foi executado três vezes, totalizando 45 execuções, para captura simultânea da latência ponta a ponta e do consumo de *tokens* junto à API da OpenAI. A repetição de cada caso permite avaliar a consistência do comportamento do sistema entre execuções, dimensão particularmente relevante na operação com modelos de linguagem, em que aspectos como condições de rede e processamento em lote no lado do provedor podem introduzir variabilidade mesmo com configuração determinística do modelo (temperatura igual a zero). Cada execução foi conduzida em sessão isolada, sem contaminação de histórico conversacional entre casos. Os preços de referência para cálculo do custo monetário corresponderam às tarifas vigentes da OpenAI no período da avaliação.

4.3.1 Latência

A latência total média observada foi de 4,10 segundos por interação, com variação significativa em função do tipo de solicitação. A Tabela 4.3 apresenta a decomposição da latência por categoria de caso.

Tabela 4.3: Latência média por categoria

Categoria	N execuções	Latência média (s)	Mínima (s)	Máxima (s)
Fora do domínio	9	1,31	1,11	1,52
Parcialmente vago	12	2,84	2,38	3,28
Vago	6	3,08	2,41	3,43
Específico	12	6,65	2,71	16,15
Informal	6	6,72	5,72	7,25
Total	45	4,10	1,11	16,15

O padrão observado é coerente com a complexidade computacional de cada tipo de solicitação. Solicitações fora do domínio, em que o agente responde diretamente sem acionar ferramentas, apresentaram a menor latência (média de 1,31 segundos). Solicitações vagas ou

parcialmente vagas, em que o agente aciona a ferramenta `PedirClarificacao` e formula perguntas ao usuário, apresentaram latência intermediária (em torno de 3 segundos). Solicitações específicas e informais, que demandam execução completa do *pipeline* de recuperação, incluindo busca vetorial, enriquecimento dos resultados e síntese da resposta, apresentaram a maior latência (cerca de 6,7 segundos).

Vale destacar que, mesmo a categoria de maior latência, mantém-se em faixa compatível com a expectativa de tempo de resposta em sistemas conversacionais para comércio eletrônico, em que valores entre 3 e 10 segundos são considerados aceitáveis dada a complexidade da operação executada. A latência mais elevada observada em uma execução isolada do caso E3 (16,15 segundos) constituiu episódio anômalo, possivelmente atribuível a condições temporárias do provedor do modelo de linguagem ou da rede, e não compromete a interpretação dos resultados agregados.

4.3.2 Consumo de Tokens e Custo Monetário

O consumo médio por interação foi de 1.715 *tokens*, decompostos em 1.600 *tokens* de entrada (compreendendo o *prompt* de sistema, o histórico conversacional, a entrada do usuário e, quando aplicável, o retorno das ferramentas) e 115 *tokens* de saída (a resposta formulada pelo agente, incluindo eventuais decisões de invocação de ferramenta).

Aplicando-se as tarifas vigentes da OpenAI para o modelo utilizado, sendo USD 0,15 por milhão de *tokens* de entrada e USD 0,60 por milhão de *tokens* de saída, o custo médio por interação foi de aproximadamente USD 0,00031, equivalente a cerca de R\$ 0,0015 considerando a cotação do dólar no período. O custo agregado das 45 execuções do experimento foi de USD 0,014, valor que evidencia a viabilidade econômica de avaliações iterativas durante o desenvolvimento. A Tabela 4.4 apresenta uma projeção do custo mensal estimado em diferentes cenários de uso.

Tabela 4.4: Estimativa de custo mensal por cenário operacional

Cenário operacional	Interações/dia	Custo mensal (USD)	Custo mensal (BRL)
Operação modesta	100	0,93	~4,65
Operação moderada	1.000	9,27	~46,35
Operação intensiva	10.000	92,70	~463,50
Operação em larga escala	100.000	927,00	~4.635,00

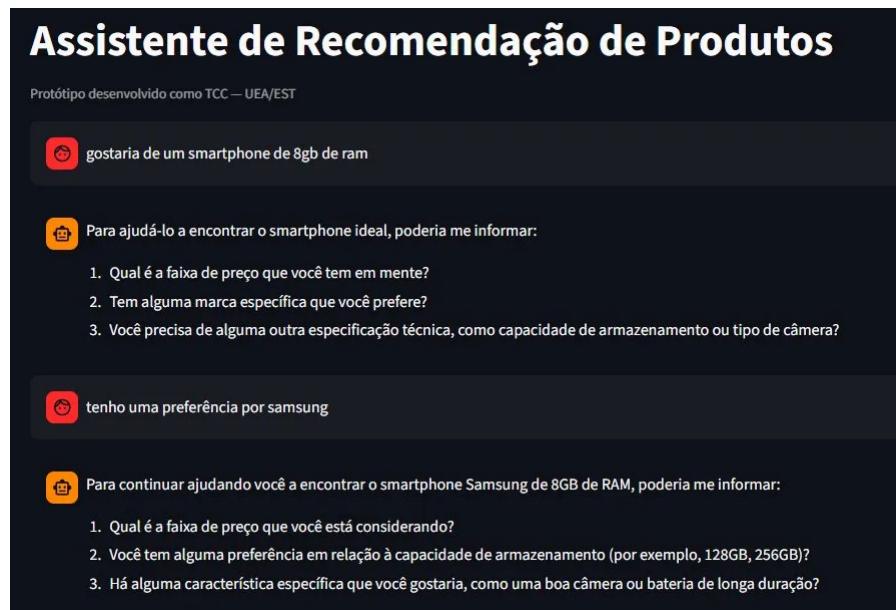
4.4 *Over-Elicitation*

Como citado anteriormente, durante o desenvolvimento surgiu o problema de *over-elicitation*, onde o agente solicitava clarificações em ciclos sucessivos, mesmo após o usuário ter fornecido informações suficientes para uma busca útil.

Para registrar empiricamente o fenômeno e a eficácia da estratégia de mitigação adotada, conduziu-se um estudo de caso comparativo entre o comportamento do agente antes e após os ajustes no prompt de sistema. A sequência de interação utilizada para o comparativo foi a mesma em ambas as versões, sendo a entrada inicial “gostaria de um smartphone de 8gb de ram” seguida, no segundo turno, da resposta “tenho uma preferência por Samsung”. Essa sequência foi escolhida por representar um cenário comum em uso real, no qual o usuário fornece informação progressivamente ao longo do diálogo.

Na versão anterior do prompt de sistema, o agente acionou a ferramenta `PedirClarificacao` no primeiro turno, formulando três perguntas ao usuário: faixa de preço, marca e demais especificações técnicas. Esse comportamento, embora possa ser considerado conservador, é defensável diante de uma entrada com apenas categoria e um atributo discriminante.

A questão crítica manifestou-se no segundo turno. Após o usuário responder com a marca preferida, o agente acionou novamente a ferramenta `PedirClarificacao`, formulando uma nova rodada de perguntas sobre faixa de preço, capacidade de armazenamento e outras características adicionais. O resultado foi um ciclo de elicitação no qual o usuário, mesmo fornecendo informação progressiva a cada turno, não recebia recomendação alguma, apenas mais perguntas. A Figura 4.1 apresenta a captura de tela dessa interação.

Figura 4.1: Ocorrência de *over-elicitation*

Elaborado pelo autor.

Esse comportamento demonstra uma característica importante de agentes baseados em LLM, que é a interpretação de critérios formulados em linguagem natural está sujeita à variabilidade do próprio modelo de linguagem, que pode adotar critérios mais rigorosos do que aqueles pretendidos pelo desenvolvedor. No caso observado, o critério “solicitação completa quando há categoria mais ao menos um atributo discriminante” foi interpretado pelo modelo de forma mais conservadora do que o pretendido, resultando em pedidos sucessivos de informação adicional.

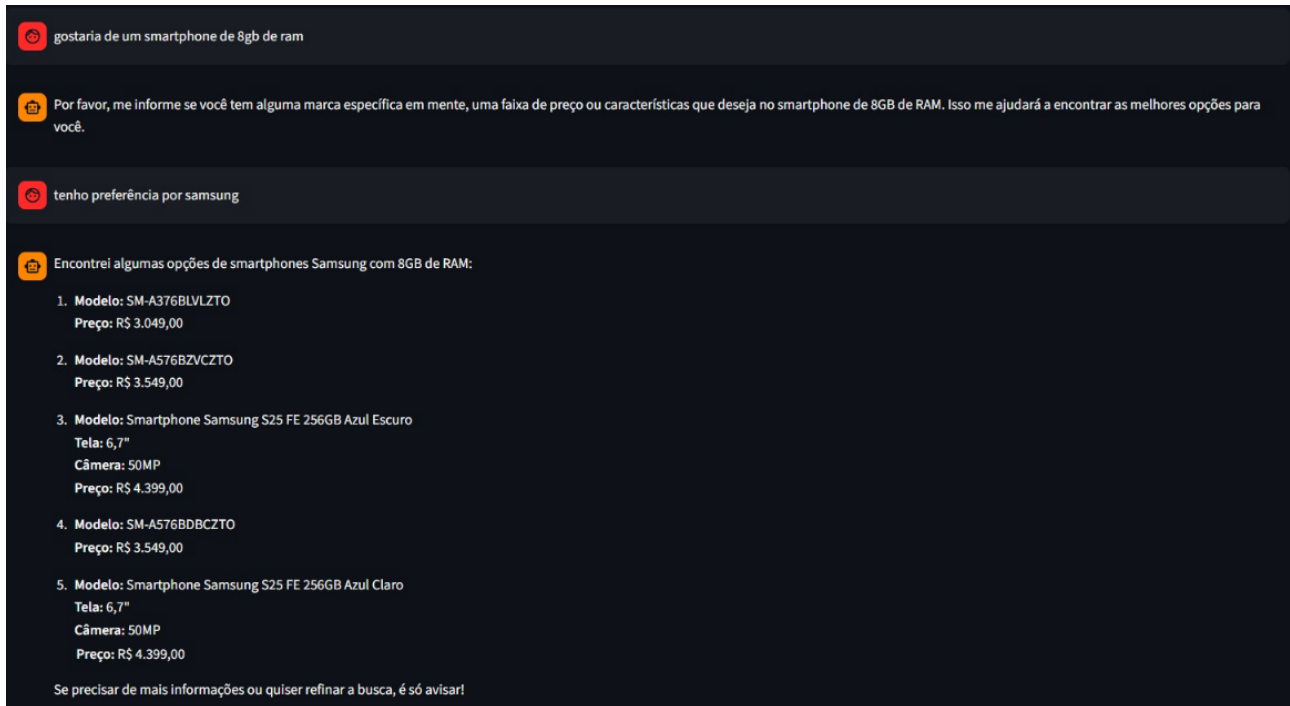
Após os ajustes descritos na metodologia, a mesma sequência de interação foi executada novamente.

No primeiro turno, o agente novamente acionou `PedirClarificacao` diante da entrada “gostaria de um smartphone de 8gb de ram”. Esse comportamento, mantido após o ajuste, indica que o agente continuou considerando útil pedir esclarecimentos adicionais em uma primeira rodada, mesmo com a nova orientação de favorecer a busca direta.

No segundo turno, contudo, o comportamento mudou substancialmente. Ao receber a informação adicional sobre a marca preferida, o agente acionou a ferramenta `BuscarProdutos`, dessa forma combinando a informação do primeiro turno (categoria *smartphone* e RAM de

8GB) com a do segundo turno (marca Samsung) e retornou recomendações ao usuário. A Figura 4.2 apresenta a captura de tela da interação após o ajuste.

Figura 4.2: Correção de *over-elicitation*



Elaborado pelo autor.

A Tabela 4.5 sintetiza o comparativo entre as duas versões.

Tabela 4.5: Comparação entre as versões antes e após o ajuste

Aspecto	Versão antes do ajuste	Versão após o ajuste
Ferramenta acionada no turno 1	PedirClarificacao	PedirClarificacao
Ferramenta acionada no turno 2	PedirClarificacao (novamente)	BuscarProdutos
Total de rodadas de clarificação	Duas ou mais (ciclo)	Uma
Resultado final	Usuário não recebe recomendação	Usuário recebe recomendação

O ganho efetivo do ajuste materializou-se no segundo turno: a versão anterior insistia em clarificar, enquanto a versão ajustada procedeu à busca com a informação acumulada nos dois turnos. Esse comportamento decorre principalmente da regra de não-insistência incorporada ao prompt, que estabelece explicitamente que o agente não deve acionar clarificação em turnos consecutivos.

Trata-se, contudo, de uma observação sobre uma única sequência reexecutada; uma avaliação

quantitativa sistemática da redução de *over-elicitation*, com múltiplas execuções e variações da entrada, constitui caminho de investigação futura.

4.5 Resumo e limitações

Os resultados apresentados neste capítulo evidenciam que o sistema desenvolvido cumpriu, no conjunto de casos avaliado, os comportamentos esperados em todas as dimensões consideradas. O agente demonstrou capacidade de selecionar adequadamente entre as ferramentas disponíveis conforme o tipo de solicitação, preservar os atributos mencionados pelo usuário ao formular consultas à busca, formular perguntas pertinentes de clarificação quando necessário e abster-se de fabricar informações não fundamentadas em dados reais do catálogo. O estudo de caso de *over-elicitation*, por sua vez, evidenciou empiricamente um fenômeno discutido na literatura e validou a eficácia da estratégia de mitigação adotada.

Esses achados confirmam que a arquitetura baseada em agentes LLM com uso de ferramentas com acesso a informações externas oferece vantagens substantivas sobre o uso isolado de modelos de linguagem em tarefas que demandam informação confiável atualizada e específica. A combinação entre as capacidades linguísticas do modelo de linguagem e o acesso a fontes verificáveis por meio de ferramentas mostrou-se, no escopo avaliado, eficaz tanto em interpretar solicitações de usuários quanto em fundamentar as respostas em dados reais. Apesar dos resultados satisfatórios obtidos, a avaliação conduzida apresenta limitações que devem ser mencionadas.

A primeira limitação refere-se ao tamanho do conjunto de teste. Os 15 casos avaliados, embora distribuídos em cinco categorias representativas dos tipos de solicitação esperados, constituem amostra reduzida. Resultados quantitativos baseados em conjunto dessa dimensão devem ser interpretados como indicativos, não como conclusivos. Avaliações futuras envolvendo conjuntos maiores e mais diversos de casos permitiriam estimativas mais robustas das taxas observadas.

A segunda limitação refere-se ao caráter subjetivo da avaliação de algumas dimensões. A classificação da aderência dos produtos retornados, da adequação das perguntas de clarifica-

ção e da ocorrência de alucinação envolveu, ainda que com critérios operacionais previamente estabelecidos, juízos do próprio autor do trabalho. Combinada à execução única por caso e ao número restrito de instâncias avaliadas (15 casos), essa estratégia de avaliação não permite estabelecer com rigor estatístico as taxas observadas, devendo os resultados quantitativos ser interpretados como indicação preliminar e qualitativa do comportamento do sistema, não como demonstração definitiva de seu desempenho. Estratégias para mitigar essa subjetividade, como avaliação por múltiplos juízes independentes ou por usuários reais em estudo de usabilidade, não foram aplicadas neste trabalho e constituem oportunidade para investigações futuras.

A terceira limitação refere-se ao escopo restrito do catálogo, já que a avaliação foi conduzida sobre um catálogo limitado a quatro categorias de produtos eletrônicos (televisores, *smartphones*, *tablets* e *notebooks*). Isso restringe a generalização dos resultados para catálogos mais amplos ou heterogêneos.

Apesar dessas limitações, os resultados obtidos permitem concluir que a abordagem investigada mostra-se promissora para a construção de sistemas de recomendação conversacionais em comércio eletrônico.

Capítulo 5

Considerações finais

A motivação original deste trabalho partiu da constatação de uma lacuna prática nos sistemas de recomendação convencionais em comércio eletrônico: a distância entre a forma como consumidores naturalmente expressam suas necessidades de consumo e a interação rígida imposta por mecanismos de busca por filtros estruturados. Diante dessa lacuna, foi investigada uma abordagem alternativa fundamentada em agentes baseados em LLM, capazes de interpretar solicitações em linguagem natural, dialogar com o usuário para refinar critérios quando necessário, e retornar recomendações fundamentadas em consultas reais ao catálogo de produtos.

A investigação foi conduzida em ambiente corporativo real, sobre uma base de produtos de uma empresa do setor varejista, com escopo delimitado a quatro categorias de produtos eletrônicos. A abordagem adotada articulou três elementos principais: uma representação tokenizada dos produtos concebida especificamente para o trabalho; um mecanismo de busca por similaridade vetorial implementado sobre essa representação; e um agente conversacional capaz de selecionar entre busca direta e elicitación de preferências conforme a completude da solicitação. A integração desses componentes foi realizada por meio do *framework* LangChain, sobre infraestrutura Databricks, com interface conversacional desenvolvida em Streamlit.

A avaliação experimental, embora conduzida sobre um conjunto restrito de 15 casos de teste, ofereceu evidências consistentes em favor da abordagem investigada. O sistema demonstrou capacidade de selecionar adequadamente a ferramenta para cada tipo de solicitação, preservar os atributos mencionados pelo usuário ao formular consultas, conduzir clarificações pertinentes

quando necessário, e abster-se de fabricar informações não fundamentadas em dados reais do catálogo. O estudo de caso sobre a mitigação do fenômeno de *over-elicitation*, conduzido em paralelo à avaliação principal, evidenciou empiricamente um desafio prático discutido na literatura e demonstrou a eficácia da estratégia adotada.

Retomando os objetivos formulados previamente, é possível afirmar que foram, no conjunto, cumpridos. O objetivo geral (desenvolver um *chatbot* baseado em agente LLM capaz de interpretar solicitações de recomendação em linguagem natural e retornar recomendações fundamentadas) foi alcançado, conforme demonstrado anteriormente. Os objetivos específicos foram igualmente atendidos: a revisão bibliográfica fundamentou as decisões de projeto; a representação tokenizada e a integração com o índice vetorial foram projetadas e implementadas; as ferramentas necessárias ao agente foram definidas e integradas; o comportamento do sistema foi avaliado experimentalmente nas diferentes categorias de solicitação previstas. Reconhece-se, contudo, que o escopo da avaliação foi reduzido em relação ao planejamento original, sem comprometer, todavia, a validade das observações registradas.

Como contribuições do trabalho, destacam-se três frentes. Em primeiro lugar, a estratégia de representação tokenizada de produtos, concebida para viabilizar busca semântica em catálogo cujos atributos não se encontravam disponíveis em formato estruturado convencional, e que dialoga com a linha emergente de representações textuais estruturadas em sistemas de recomendação. Em segundo lugar, a arquitetura de agente conversacional desenvolvida, com seu mecanismo de tratamento de solicitações incompletas, oferece um modelo replicável para cenários similares em comércio eletrônico. Em terceiro lugar, a documentação do fenômeno de *over-elicitation* e da estratégia de mitigação adotada constitui contribuição metodológica relevante, evidenciando empiricamente um problema discutido na literatura e oferecendo um exemplo concreto de como abordá-lo por meio de engenharia de *prompts*.

As limitações do trabalho foram abordadas mas merecem ser brevemente retomadas. O tamanho reduzido do conjunto de teste limita a generalização dos resultados quantitativos. A ausência de comparação experimental sistemática entre diferentes configurações do agente, impede a quantificação precisa dos ganhos atribuíveis especificamente à arquitetura proposta.

A avaliação conduzida pelo próprio autor, embora baseada em critérios operacionais explícitos, introduz subjetividade que poderia ser mitigada por estratégias de avaliação por múltiplos juízes ou por usuários reais.

Tais limitações configuram, ao mesmo tempo, direções naturais para trabalhos futuros, abordados a seguir.

Uma avaliação experimental ampliada, com conjuntos maiores de casos de teste e comparação sistemática entre configurações alternativas do sistema, permitiria caracterizar com maior precisão os ganhos da abordagem. A validação da acurácia da extração, conduzida sobre amostra significativamente maior do que a empregada neste trabalho, permitiria detectar tipos de erro não capturados pela inspeção restrita realizada e fortalecer a confiança na etapa que fundamenta o *pipeline*. Estudos de usabilidade com usuários reais, em condições de uso efetivo, ofereceriam evidências complementares sobre a aderência do sistema às necessidades de quem o utiliza.

Investigações sobre o impacto de diferentes modelos de embeddings, particularmente modelos treinados especificamente em português ou em catálogos de produtos, poderiam aprimorar a qualidade da busca semântica, especialmente diante da observação de que o modelo utilizado foi treinado predominantemente em *corpora* em língua inglesa. De forma análoga, é pertinente a avaliação de modelos de linguagem mais recentes como núcleo do agente, conforme novas gerações se tornem disponíveis através dos provedores utilizados, comparando dimensões como qualidade das decisões, latência e custo. A replicação da arquitetura em outros domínios e catálogos, abrangendo categorias de produtos não contempladas no escopo deste trabalho, contribuiria para avaliar a generalidade da abordagem.

Outra direção relevante de evolução refere-se ao tratamento da dimensão temporal do catálogo. A presente implementação adota sincronização manual do índice vetorial e representação por faixas para o atributo de preço, decisões adequadas ao escopo experimental, mas que limitam a reflexão imediata de mudanças no catálogo. A adoção de sincronização contínua entre a tabela base e o índice, combinada a estratégias de representação numérica de preço no espaço vetorial, constitui caminho natural para a aplicação do sistema em cenários de operação contí-

nua, nos quais a frequência de alterações de preço é elevada e a aderência da recomendação ao estado corrente do catálogo torna-se requisito de maior relevância operacional.

Em síntese, este trabalho oferece evidências preliminares de que a combinação entre agentes baseados em LLM, representações tokenizadas de itens e busca por similaridade vetorial constitui uma abordagem promissora para a construção de sistemas de recomendação conversacionais em comércio eletrônico. Os resultados obtidos, embora circunscritos ao escopo avaliado, indicam que tal arquitetura oferece aos usuários uma forma de interação mais próxima de seus padrões naturais de comunicação, sem comprometer a precisão factual das recomendações apresentadas. Espera-se que o trabalho contribua para o avanço da pesquisa na intersecção entre processamento de linguagem natural, sistemas de recomendação conversacionais e arquiteturas de agentes para aplicação em domínios corporativos.

Apêndice A

Códigos utilizados

```
WHEN 'smartphone' THEN ai_extract(  
  title,  
  '{  
    "marca": {  
      "type": "string",  
      "enum": ["Samsung", "Apple", "Motorola", "Xiaomi", "Redmi",  
        "Poco", "Oppo", "Honor", "TCL", "Realme",  
        "Infinix", "Jovi", "outro"],  
      "description": "Trate Redmi e Poco como marcas separadas (sub-marcas Xiaomi)."  
    },  
    "armazenamento_gb": {  
      "type": "integer",  
      "description": "Armazenamento em GB. Converta 1TB para 1024."  
    },  
    "ram_gb": {"type": "integer"},  
    "cor_base": {  
      "type": "string",  
      "enum": ["preto", "branco", "azul", "verde", "vermelho",  
        "rosa", "roxo", "lilas", "amarelo", "laranja",  
        "cinza", "grafite", "prata", "dourado", "marrom",  
        "bege", "marfim", "titânio", "outro"],  
      "description": "Cor base, ignorando qualificadores como escuro, claro, nevea."  
    },  
    "tela_polegadas": {"type": "number"},  
    "camera_principal_mp": {"type": "integer"}  
  }'  
)
```

Código A.1: Extração de atributos para a categoria *smartphone*.

```

WHEN 'tv' THEN ai_extract(
  title,
  '{
    "marca": {
      "type": "string",
      "enum": ["Samsung", "LG", "TCL", "Philips", "AOC",
        "SEMP", "Philco", "Hisense", "Sony", "outro"]
    },
    "tamanho_polegadas": {
      "type": "integer",
      "description": "Tamanho da tela em polegadas. Valores tipicos: 32, 43, 50, 55, 65, 75, 85."
    },
    "resolucao": {
      "type": "string",
      "enum": ["HD", "FHD", "4K", "8K"],
      "description": "HD = 720p, FHD = Full HD, 4K = UHD, 8K. Mapeie UHD para 4K."
    },
    "tecnologia_tela": {
      "type": "string",
      "enum": ["LED", "QLED", "OLED", "NEO_QLED", "QNED",
        "QD_MINI_LED", "MINI_LED"],
      "description": "Tecnologia da tela. NEO QLED -> NEO_QLED. QD-Mini LED -> QD_MINI_LED."
    },
    "sistema_operacional": {
      "type": "string",
      "enum": ["Google_TV", "Roku", "WebOS", "Tizen", "outro"],
      "description": "Plataforma smart. Inferir Tizen para Samsung sem outro SO mencionado."
    }
  }'
)

```

Código A.2: Extração de atributos para a categoria televisão.

```

WHEN 'notebook' THEN ai_extract(
  title,
  '{
    "marca": {
      "type": "string",
      "enum": ["Samsung", "Lenovo", "Asus", "Acer", "MSI",
        "Dell", "HP", "Apple", "Positivo", "LG",
        "Vaio", "outro"]
    },
    "processador_marca": {
      "type": "string",
      "enum": ["Intel", "AMD", "Qualcomm", "Apple"],
      "description": "Snapdragon -> Qualcomm. Ryzen -> AMD. Core/Celeron -> Intel."
    },
    "processador_familia": {
      "type": "string",
      "description": "Familia do chip. Ex: Core i3, Core i5, Core i7, Core Ultra 7, Ryzen 5, Ryzen 7, Celeron,
        ↳ Snapdragon 7C, Apple M3."
    },
    "ram_gb": {"type": "integer"},
    "armazenamento_gb": {
      "type": "integer",
      "description": "Capacidade em GB. Converta TB para GB (1 TB = 1024 GB)."
    },
    "tela_polegadas": {"type": "number"},
    "gpu_marca": {
      "type": "string",
      "enum": ["NVIDIA", "AMD", "Intel", "integrada"]
    },
    "gpu_modelo": {
      "type": "string",
      "description": "Modelo da GPU. Ex: RTX 3050, RTX 4050, RTX 5060."
    }
  }'
)

```

Código A.3: Extração de atributos para a categoria *notebook*.

```

WHEN 'tablet' THEN ai_extract(
  title,
  '{
    "marca": {
      "type": "string",
      "enum": ["Samsung", "Apple", "Lenovo", "Microsoft",
        "Huawei", "Amazon", "Xiaomi", "Positivo",
        "Vaio", "Multilaser", "Cwell", "reMarkable", "outro"]
    },
    "cor": {
      "type": "string",
      "enum": ["preto", "branco", "cinza", "grafite", "prata",
        "azul", "verde", "rosa", "dourado", "starlight", "outro"]
    },
    "armazenamento_gb": {"type": "integer"},
    "ram_gb": {"type": "integer"},
    "tela_polegadas": {"type": "number"},
    "processador": {
      "type": "string",
      "description": "Chip ou processador. Ex: A17 Pro, A16, M2, Octa-Core, Snapdragon X."
    },
    "sistema_operacional": {
      "type": "string",
      "enum": ["iPadOS", "Android", "Windows", "FireOS", "outro"],
      "description": "Inferir: iPad -> iPadOS, Surface -> Windows, Amazon Fire -> FireOS, demais -> Android."
    }
  }'
)

```

Código A.4: Extração de atributos para a categoria *tablet*.

Apêndice B

Relação completa dos casos de teste

Este apêndice apresenta a relação completa dos 15 casos de teste utilizados na avaliação experimental do sistema. O material está organizado em duas seções. A Seção B.1 apresenta uma tabela resumida com a classificação de cada caso e os respectivos resultados de avaliação. A Seção B.2 apresenta, caso a caso, a entrada do usuário, a consulta enviada à ferramenta de busca (quando aplicável) e a resposta produzida pelo agente.

B.1 Tabela resumida dos casos de teste

A Tabela B.1 apresenta a relação dos casos avaliados, agrupados por categoria, com indicação da ferramenta efetivamente acionada pelo agente e do resultado da avaliação em cada uma das dimensões consideradas. Casos para os quais determinada dimensão não se aplica são identificados como N/A.

Tabela B.1: Relação completa dos casos de teste utilizados na avaliação experimental.

ID	Categoria	Entrada do usuário	Ferramenta acionada	Atrib. preserv.	Elicit. adequada	Houve alucinação
E1	específico	<i>estou procurando um smartphone samsung com 8gb de ram e 128gb de armazenamento até 3000 reais</i>	BuscarProdutos	Sim	N/A	Não
E12	específico	<i>quero um fone de até 300 reais</i>	BuscarProdutos	Sim	N/A	Não
E2	específico	<i>preciso de uma tv de 55 polegadas 4k da LG</i>	BuscarProdutos	Sim	N/A	Não
E3	específico	<i>quero um notebook com 8gb de ram até 4000 reais</i>	BuscarProdutos	Sim	N/A	Não
E11	informal	<i>celular samsung de até 4k</i>	BuscarProdutos	Sim	N/A	Não
E9	informal	<i>quero um cel samsung com 8gb</i>	BuscarProdutos	Sim	N/A	Não
E10	parcialmente vago	<i>tv que pega netflix</i>	PedirClarificacao	Sim	Sim	Não
E4	parcialmente vago	<i>quero um tablet bom</i>	PedirClarificacao	Sim	Sim	Não
E5	parcialmente vago	<i>preciso de um notebook para trabalhar</i>	PedirClarificacao	Sim	Sim	Não
E6	parcialmente vago	<i>to precisando trocar de celular</i>	PedirClarificacao	Sim	Sim	Não
E7	vago	<i>me recomenda uma coisa boa</i>	PedirClarificacao	Sim	Sim	Não

(continua na próxima página)

(continuação da Tabela B.1)

ID	Categoria	Entrada do usuário	Ferramenta acionada	Atrib. preserv.	Elicit. adequada	Houve alucinação
E8	vago	<i>quero comprar algo legal</i>	PedirClarificacao	Sim	Sim	Não
E13	fora do domínio	<i>olá, tudo bem?</i>	Nenhuma	N/A	N/A	Não
E14	fora do domínio	<i>como funciona a teoria da relatividade?</i>	Nenhuma	N/A	N/A	Não
E15	fora do domínio	<i>você é uma inteligência artificial?</i>	Nenhuma	N/A	N/A	Não

B.2 Detalhamento dos casos: entradas, consultas e respostas

Esta seção apresenta, para cada caso de teste, a entrada original do usuário, a consulta enviada à ferramenta de busca (quando aplicável) e a resposta produzida pelo agente. Os casos estão agrupados por categoria, seguindo a mesma ordem da Tabela B.1.

B.2.1 Casos específicos

Caso E1

Entrada do usuário: *estou procurando um smartphone samsung com 8gb de ram e 128gb de armazenamento até 3000 reais*

Comportamento esperado: busca direta

Ferramenta acionada pelo agente: BuscarProdutos

Consulta enviada à busca: *smartphone Samsung 8GB de RAM 128GB até 3000 reais*

Resposta do agente:

Encontrei algumas opções de smartphones Samsung com 8GB de RAM e que estão próximos do seu orçamento de até 3000 reais:

1. **Modelo:** SM-A576BZAAZTO

Preço: R\$ 3.049,00

Memória RAM: 8GB

2. **Modelo:** Galaxy S25 FE

Preço: R\$ 3.099,00

Memória RAM: 8GB + 8GB Plus

Armazenamento interno: 128GB

Tamanho da Tela: 6.7"

3. **Modelo:** SM-A576BDBAZTO

Preço: R\$ 3.049,00

Memória RAM: 8GB

4. **Modelo:** SM-A576BZVAZTO

Preço: R\$ 3.049,00

Memória RAM: 8GB

Essas opções estão um pouco acima do seu limite de preço, mas são as melhores disponíveis. Se precisar de mais informações ou quiser refinar a busca, é só avisar!

Caso E12

Entrada do usuário: *quero um fone de até 300 reais*

Comportamento esperado: busca direta

Ferramenta acionada pelo agente: BuscarProdutos

Consulta enviada à busca: *fone de ouvido até 300 reais*

Resposta do agente:

Parece que não consegui encontrar opções de fones de ouvido dentro da faixa de até 300 reais. Os resultados que obtive estão relacionados a smartphones e outros produtos.

Caso E2

Entrada do usuário: *preciso de uma tv de 55 polegadas 4k da LG*

Comportamento esperado: busca direta

Ferramenta acionada pelo agente: BuscarProdutos

Consulta enviada à busca: *TV 55 polegadas 4K LG*

Resposta do agente:

Encontrei uma opção de TV de 55 polegadas 4K da LG:

1. *LG OLED55C5PSA*

- *Tamanho da Tela:* 55"

- *Preço:* R\$ 5.569,00

Se você precisar de mais informações ou quiser explorar outras opções, é só avisar!

Caso E3

Entrada do usuário: *quero um notebook com 8gb de ram até 4000 reais*

Comportamento esperado: busca direta

Ferramenta acionada pelo agente: BuscarProdutos

Consulta enviada à busca: *notebook 8GB de RAM até 4000 reais*

Resposta do agente:

Encontrei algumas opções de notebooks com 8GB de RAM até 4000 reais:

1. Notebook Samsung Galaxy Book4

- **Processador:** Intel Core i3
- **Armazenamento:** 256GB SSD
- **Tela:** 15.6" FHD LED
- **Sistema Operacional:** Windows 11 Home
- **Preço:** R\$ 3.999,00

2. Notebook Samsung Galaxy Book4

- **Processador:** Intel Core i5-1355U
- **Armazenamento:** 512GB SSD
- **Tela:** 15.6"
- **Sistema Operacional:** Windows 11 Home
- **Preço:** R\$ 5.199,00 (acima do seu orçamento)

3. Notebook Samsung Galaxy Book Go

- **Processador:** Snapdragon 7C
- **Memória RAM:** 4GB (não atende ao critério de 8GB)
- **Armazenamento:** 128GB UFS
- **Tela:** 14"
- **Preço:** R\$ 2.399,00 (não atende ao critério de RAM)

As opções que atendem ao seu critério de 8GB de RAM e estão dentro do orçamento são limitadas. O modelo da Samsung Galaxy Book4 com Intel Core i3 é a melhor opção. Se precisar de mais informações ou quiser refinar a busca, é só avisar!

B.2.2 Casos informais

Caso E11

Entrada do usuário: *celular samsung de até 4k*

Comportamento esperado: busca direta com interpretação de linguagem informal

Ferramenta acionada pelo agente: BuscarProdutos

Consulta enviada à busca: *smartphone Samsung até 4000 reais*

Resposta do agente:

Aqui estão algumas opções de smartphones Samsung na faixa de até 4 mil reais:

1. *Samsung SM-A576BZAAZTO*

- *Memória RAM:* 8GB

- *Preço:* R\$ 3.049,00

2. *Samsung Galaxy S25 FE*

- *Memória RAM:* 8GB

- *Armazenamento interno:* 128GB

- *Tamanho da Tela:* 6.7"

- *Preço:* R\$ 3.099,00

3. *Samsung SM-A576BZACZTO*

- *Memória RAM:* 8GB

- *Preço:* R\$ 3.549,00

4. *Samsung S25 FE*

- *Memória RAM:* 8GB + 8GB Plus

- *Preço:* R\$ 3.099,00

5. *Samsung Galaxy S25 FE*

- *Memória RAM:* 8GB

- *Armazenamento interno:* 128GB

- *Tamanho da Tela:* 6.7"

- *Preço:* R\$ 3.099,00

Essas opções estão dentro do seu orçamento. Se precisar de mais informações ou quiser refinar a busca, é só avisar!

Caso E9

Entrada do usuário: *quero um cel samsung com 8gb*

Comportamento esperado: busca direta com interpretação de linguagem informal

Ferramenta acionada pelo agente: BuscarProdutos

Consulta enviada à busca: *smartphone Samsung 8GB de RAM*

Resposta do agente:

Aqui estão algumas opções de smartphones Samsung com 8GB de RAM:

1. Samsung SM-A376BLVLZTO

- **Memória RAM:** 8GB

- **Preço:** R\$ 3.049,00

2. Samsung SM-A576BZVCZTO

- **Memória RAM:** 8GB

- **Preço:** R\$ 3.549,00

3. Samsung Galaxy S25 FE

- **Armazenamento:** 256GB

- **Tamanho da Tela:** 6,7"

- **Câmera:** 50MP

- **Memória RAM:** 8GB

- **Preço:** R\$ 4.399,00

4. Samsung SM-A576BDBCZTO

- **Memória RAM:** 8GB

- **Preço:** R\$ 3.549,00

5. Samsung Galaxy S25 FE

- **Armazenamento:** 256GB

- **Tamanho da Tela:** 6,7"

- **Câmera:** 50MP

- **Memória RAM:** 8GB

- **Preço:** R\$ 4.399,00

Se precisar de mais informações ou quiser refinar a busca, é só avisar!

B.2.3 Casos parcialmente vagos

Caso E10

Entrada do usuário: *tv que pega netflix*

Comportamento esperado: clarificação com pergunta sobre categoria de produto desejada

Ferramenta acionada pelo agente: PedirClarificacao

Resposta do agente:

Para ajudar melhor, você poderia me informar qual é a faixa de preço que você tem em mente ou se prefere alguma marca específica para a TV?

Caso E4

Entrada do usuário: *quero um tablet bom*

Comportamento esperado: clarificação com perguntas sobre uso, orçamento e marca

Ferramenta acionada pelo agente: PedirClarificacao

Resposta do agente:

Para ajudar melhor, você poderia me informar qual categoria de tablet você está procurando? Além disso, tem alguma marca ou especificação técnica em mente?

Caso E5

Entrada do usuário: *preciso de um notebook para trabalhar*

Comportamento esperado: clarificação com perguntas sobre uso, orçamento e marca

Ferramenta acionada pelo agente: PedirClarificacao

Resposta do agente:

Qual tipo de notebook você está procurando? Alguma marca específica, especificação técnica (como memória RAM ou armazenamento) ou faixa de preço em mente? Isso vai me ajudar a encontrar as melhores opções para você.

Caso E6

Entrada do usuário: *to precisando trocar de celular*

Comportamento esperado: clarificação com perguntas sobre uso, orçamento e marca

Ferramenta acionada pelo agente: PedirClarificacao

Resposta do agente:

Qual categoria de celular você está procurando? Alguma marca ou especificação em mente?

B.2.4 Casos vagos

Caso E7

Entrada do usuário: *me recomenda uma coisa boa*

Comportamento esperado: clarificação com pergunta sobre categoria de produto desejada

Ferramenta acionada pelo agente: PedirClarificacao

Resposta do agente:

Para poder ajudar melhor, você poderia me informar qual tipo de produto você está procurando? Por exemplo, um smartphone, notebook, tablet, ou outro?

Além disso, se tiver alguma preferência de marca ou especificação, isso também ajudaria bastante!

Caso E8

Entrada do usuário: *quero comprar algo legal*

Comportamento esperado: clarificação com pergunta aberta

Ferramenta acionada pelo agente: PedirClarificacao

Resposta do agente:

Para ajudar melhor, você poderia me informar qual tipo de produto você está procurando? Por exemplo, um smartphone, notebook, tablet, TV, ou outro? Além disso, se tiver alguma preferência de marca ou especificação, isso também ajudaria bastante!

B.2.5 Casos fora do domínio

Caso E13

Entrada do usuário: *olá, tudo bem?*

Comportamento esperado: resposta conversacional sem acionar ferramenta

Ferramenta acionada pelo agente: Nenhuma

Resposta do agente:

Olá! Estou aqui para ajudar com recomendações de produtos. Como posso assisti-lo hoje?

Caso E14

Entrada do usuário: *como funciona a teoria da relatividade?*

Comportamento esperado: resposta sem acionar ferramenta

Ferramenta acionada pelo agente: Nenhuma

Resposta do agente:

Desculpe, mas não posso debater sobre esse assunto. Posso ajudar você a encontrar produtos. Se precisar de recomendações, é só me informar!

Caso E15

Entrada do usuário: *you are an artificial intelligence?*

Comportamento esperado: resposta sobre a natureza do assistente, sem acionar ferramenta

Ferramenta acionada pelo agente: Nenhuma

Resposta do agente:

Desculpe, mas não posso debater sobre esse assunto. Posso ajudar você a encontrar produtos. Se precisar de recomendações, é só me informar!

Referências Bibliográficas

BENGESI, S. et al. Advancements in generative ai: A comprehensive review of gans, gpt, autoencoders, diffusion model, and transformers. *IEEE Access*, v. 12, p. 69812–69837, 2024.

BROWN, T. B. et al. *Language Models are Few-Shot Learners*. 2020. Disponível em: <<https://arxiv.org/abs/2005.14165>>.

CHENG, Y. et al. *Exploring Large Language Model based Intelligent Agents: Definitions, Methods, and Prospects*. 2024. Disponível em: <<https://arxiv.org/abs/2401.03428>>.

CHOWDHURY, A. et al. *PaLM: Scaling Language Modeling with Pathways*. 2022. Disponível em: <<https://arxiv.org/abs/2204.02311>>.

CHOWDHURY, G. G. Natural language processing. *Annual Review of Information Science and Technology*, Annual Review of Information Science and Technology, v. 37, n. 1, p. 51–89, 2005.

DATABRICKS. *Databricks AI Search*. 2026. Acesso em: 20 junho 2026. Disponível em: <<https://docs.databricks.com/aws/en/ai-search/ai-search>>.

DATABRICKS. *Query a chat model*. 2026. Acesso em: 20 junho 2026. Disponível em: <<https://docs.databricks.com/aws/en/machine-learning/model-serving/query-chat-models>>.

DATABRICKS. *Query an embedding model*. 2026. Acesso em: 20 junho 2026. Disponível em: <<https://docs.databricks.com/aws/en/machine-learning/model-serving/query-embedding-models>>.

GAO, C. et al. Advances and challenges in conversational recommender systems: A survey. *AI Open*, Elsevier BV, v. 2, p. 100–126, 2021. ISSN 2666-6510. Disponível em: <<http://dx.doi.org/10.1016/j.aiopen.2021.06.002>>.

GAO, Y. et al. *Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System*. 2023. Disponível em: <<https://arxiv.org/abs/2303.14524>>.

GENG, S. et al. *Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt Predict Paradigm (P5)*. 2022. Disponível em: <<https://arxiv.org/abs/2203.13366>>.

GUO, T. et al. *Large Language Model based Multi-Agents: A Survey of Progress and Challenges*. 2024.

- JANNACH, D. et al. A survey on conversational recommender systems. *ACM Computing Surveys*, Association for Computing Machinery (ACM), v. 54, n. 5, p. 1–36, maio 2021. ISSN 1557-7341. Disponível em: <<http://dx.doi.org/10.1145/3453154>>.
- JI, Z. et al. Survey of hallucination in natural language generation. *ACM Computing Surveys*, Association for Computing Machinery (ACM), v. 55, n. 12, p. 1–38, mar. 2023. ISSN 1557-7341. Disponível em: <<http://dx.doi.org/10.1145/3571730>>.
- JOHRI, P. et al. *Natural Language Processing: History, Evolution, Application and Future Work*. 2020. EasyChair Preprint no. 3239. Disponível em: <<https://easychair.org/publications/preprint/4qQd>>.
- LÁLA, J. et al. *PaperQA: Retrieval-Augmented Generative Agent for Scientific Research*. 2023. Disponível em: <<https://arxiv.org/abs/2312.07559>>.
- LANGCHAIN. *LangChain Overview*. 2026. Acesso em: 16 maio 2026. Disponível em: <<https://docs.langchain.com/oss/python/langchain/overview>>.
- LANGCHAIN. *LangChain Reference: RunnableWithMessageHistory*. 2026. Acesso em: 20 junho 2026. Disponível em: <<https://reference.langchain.com/python/langchain-core/runnables/history/RunnableWithMessageHistory>>.
- LIN, J. et al. *How Can Recommender Systems Benefit from Large Language Models: A Survey*. 2024. Disponível em: <<https://arxiv.org/abs/2306.05817>>.
- LIU, Y. et al. *Conversational Recommender System and Large Language Model Are Made for Each Other in E-commerce Pre-sales Dialogue*. 2023. Disponível em: <<https://arxiv.org/abs/2310.14626>>.
- LIU, Z. et al. Knowledge distillation based contextual relevance matching for E-commerce product search. In: LI, Y.; LAZARIDOU, A. (Ed.). *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Abu Dhabi, UAE: Association for Computational Linguistics, 2022. p. 63–76. Disponível em: <<https://aclanthology.org/2022.emnlp-industry.5/>>.
- MINAEE, S. et al. *Large Language Models: A Survey*. 2024. Disponível em: <<https://arxiv.org/abs/2402.06196>>.
- NAVEED, H. et al. *A Comprehensive Overview of Large Language Models*. 2024.
- PATWARDHAN, N.; MARRONE, S.; SANSONE, C. Transformers in the real world: A survey on nlp applications. *Information*, v. 14, n. 4, 2023. ISSN 2078-2489. Disponível em: <<https://www.mdpi.com/2078-2489/14/4/242>>.
- STREAMLIT. *Basic concepts of Streamlit*. 2026. Acesso em: 20 junho 2026. Disponível em: <<https://docs.streamlit.io/get-started/fundamentals/main-concepts>>.
- VASWANI, A. et al. *Attention Is All You Need*. 2017. Disponível em: <<https://arxiv.org/abs/1706.03762>>.

VEIGA, C. P. d. et al. E-commerce in brazil: An in-depth analysis of digital growth and strategic approaches for online retail. *Journal of Theoretical and Applied Electronic Commerce Research*, v. 19, n. 2, p. 1559–1579, 2024. ISSN 0718-1876. Disponível em: <<https://www.mdpi.com/0718-1876/19/2/76>>.

WEI, J. et al. *Emergent Abilities of Large Language Models*. 2022.

YAO, S. et al. *ReAct: Synergizing Reasoning and Acting in Language Models*. 2023. Disponível em: <<https://arxiv.org/abs/2210.03629>>.

ZHAO, W. X. et al. *A Survey of Large Language Models*. 2023.

ZHAO, Z. et al. Recommender systems in the era of large language models (llms). *IEEE Transactions on Knowledge and Data Engineering*, Institute of Electrical and Electronics Engineers (IEEE), v. 36, n. 11, p. 6889–6907, nov. 2024. ISSN 2326-3865. Disponível em: <<http://dx.doi.org/10.1109/TKDE.2024.3392335>>.

ZHOU, M. et al. Progress in neural nlp: Modeling, learning, and reasoning. *Engineering*, v. 6, n. 3, p. 275–290, 2020. ISSN 2095-8099. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2095809919304928>>.