Chapter 3

Como Criar seu Próprio Assistente de Pesquisa Científica com LangGraph

Larissa Souza do Nascimento e Ricardo Moura Sekeff Budaruiche

Abstract

This chapter discusses the development of a multi-agent system based on the LangGraph framework to automate scientific searches. The proposed architecture combines specialized agents, including arXiv querying, general web search via Tavily AI, and semantic storage using ChromaDB, coordinated by a supervising agent capable of delegating tasks and interpreting responses. Throughout development, the project faced limitations related to the reliability of AI services, the control of hallucinations, dependency on private API keys, and difficulties in explaining agent behavior. Despite these challenges, the results indicate promising potential for enhancing literature reviews and promoting faster scientific information retrieval. However, further work is needed to improve source filtering, integrate with institutional repositories, and define robust evaluation metrics. The initiative aims to spark new discussions on the responsible and effective deployment of intelligent agents in scientific workflows.

Resumo

Este capítulo apresenta o desenvolvimento de um sistema multiagente baseado no framework LangGraph, voltado à automatização de buscas científicas. A solução propõe a coordenação de agentes especializados, como consultas ao arXiv, buscas gerais na web via Tavily AI e armazenamento semântico com ChromaDB, supervisionados por um agente central que decide e direciona as ações. Durante o desenvolvimento, foram enfrentadas limitações quanto à estabilidade das ferramentas de IA utilizadas, controle de alucinações, necessidade de chaves privadas, além de desafios relacionados à explicabilidade do comportamento dos agentes. Embora os resultados iniciais demonstrem o potencial da abordagem para otimizar revisões bibliográficas e acelerar o acesso à informação científica, ainda são necessárias melhorias quanto à curadoria de fontes, integração com bancos de dados institucionais e definição de métricas confiáveis de desempenho. Espera-se que a prática apresentada inspire novas aplicações com maior robustez, confiabilidade e alinhamento ético no uso de agentes inteligentes na ciência.

3.1. Introdução

O advento da Inteligência Artificial (IA) tem provocado transformações profundas na maneira como o conhecimento científico é produzido, acessado e validado. Nos últimos anos, o volume de dados científicos disponíveis cresceu exponencialmente, exigindo novas abordagens computacionais para que pesquisadores possam lidar com a complexidade e a velocidade de geração dessas informações (FIORILLO; MEHTA, 2024). Nesse contexto, destaca-se o uso de agentes inteligentes, capazes de automatizar tarefas repetitivas, filtrar informações relevantes e auxiliar no processo decisório.

Agentes inteligentes são sistemas computacionais que percebem o ambiente por meio de sensores e reagem por meio de atuadores, conforme ilustrado na Figura 3.1. Sua aplicação no contexto da pesquisa científica é particularmente relevante, pois permite a execução de tarefas como a recuperação automatizada de literatura, a integração de dados provenientes de diferentes fontes e a análise preditiva baseada em grandes volumes de informação (MANZOOR et al., 2012). Tais capacidades ampliam o potencial analítico dos pesquisadores, ao mesmo tempo que reduzem o tempo despendido com atividades operacionais.

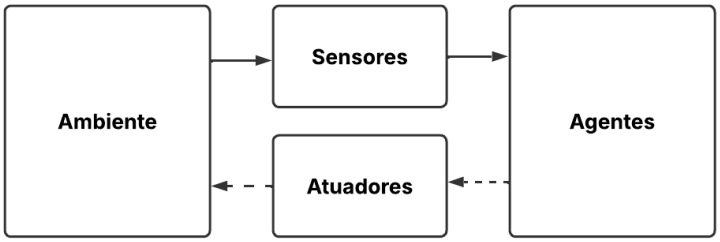

Figure 3.1: Fluxograma de agentes inteligentes

Diversos *frameworks* têm sido desenvolvidos para facilitar a construção desses agentes, destacando-se, entre eles, o *AutoGen* e o *CrewAI*. Essas ferramentas viabilizam a criação de agentes baseados em *Large Language Models* (LLMs) para tarefas específicas. Embora apresentem contribuições valiosas, frequentemente carecem de controle estrutural refinado e de flexibilidade nos fluxos de decisão, fatores cruciais quando se busca precisão e rastreabilidade em contextos científicos.

Diante dessa lacuna, o *LangGraph* surge como uma proposta inovadora. Desenvolvido pela *LangChain* (CHASE, 2024), o *LangGraph* adota uma arquitetura baseada em grafos direcionados, oferecendo suporte a ciclos de *feedback*, persistência de estado e controlabilidade em tempo real. Essas propriedades tornam o *framework* especialmente adequado para aplicações científicas, nas quais o refinamento iterativo das informações e a validação cruzada entre múltiplas fontes constituem práticas essenciais.

Este capítulo tem como objetivo apresentar uma abordagem prática e fundamentada para a construção de um assistente de pesquisa científica utilizando o *LangGraph*.

Por meio de uma combinação de fundamentação teórica, exemplos práticos e reflexões críticas, serão discutidos os seguintes aspectos:

1. A definição e a estrutura dos agentes inteligentes aplicados à ciência;
2. O funcionamento da arquitetura do *LangGraph* e suas vantagens comparativas;
3. O ambiente de desenvolvimento e a implementação prática de agentes com foco em pesquisa;
4. Estratégias de otimização e mitigação de falhas, como alucinações;
5. Aspectos éticos e responsáveis no uso da IA para fins científicos.

Ao longo do capítulo, será demonstrado como o *LangGraph* pode ser utilizado para criar agentes capazes de interagir com repositórios científicos, como o *arXiv*, integrando dados, refinando respostas e auxiliando na geração de conhecimento. Ressalta-se que o objetivo não é substituir o pesquisador humano, mas potencializar sua capacidade analítica, permitindo que ele se concentre em tarefas de maior valor intelectual.

3.2. Fundamentos Teóricos

A ascensão da IA tem transformado significativamente os métodos de investigação científica, promovendo avanços substanciais na automação, extração e interpretação de dados em larga escala. Dentre as soluções mais promissoras, destacam-se os agentes inteligentes baseados LLMs, cuja arquitetura permite a construção de assistentes autônomos voltados ao apoio direto à atividade de pesquisa. Este capítulo apresenta os conceitos teóricos que fundamentam o desenvolvimento desses agentes, com ênfase nas capacidades oferecidas pelo *framework LangGraph*, em comparação a alternativas contemporâneas como o *AutoGen* e o *CrewAI*.

3.2.1. Agentes Inteligentes: Conceito e Aplicações

Agentes inteligentes são definidos como entidades computacionais autônomas, capazes de perceber o ambiente, processar informações, tomar decisões e executar ações com o objetivo de atingir metas previamente estabelecidas. No contexto científico, esses agentes são utilizados para automatizar tarefas operacionais, como a busca em repositórios acadêmicos, a análise de documentos e a sumarização de dados, permitindo que os pesquisadores concentrem seus esforços na interpretação crítica e na formulação de hipóteses.

O *LangGraph* pode ser aplicado em uma ampla gama de cenários, como: desenvolvimento de *chatbots*, agentes autônomos, sistemas multiagentes, ferramentas de automação de fluxos de trabalho, ambientes personalizados de aprendizagem e sistemas de recomendação baseados em dados contextuais.

Além da autonomia, os agentes inteligentes distinguem-se por características como adaptabilidade, capacidade de interação com múltiplas fontes e habilidades de aprendizado contínuo. Quando organizados em sistemas multiagentes, esses agentes colaboram entre si, formando redes articuladas que permitem a resolução de problemas mais complexos de maneira cooperativa.

3.2.2. Estrutura de Agentes Baseados em LLMs

Com o avanço dos LLMs, emergiu uma nova geração de agentes capazes de realizar inferências complexas, interpretar linguagem natural com elevado grau de precisão e interagir de forma contextualizada com dados estruturados e não estruturados. *Frameworks* como o *LangGraph*, o *AutoGen* e o *CrewAI* têm popularizado a orquestração desses agentes, ao oferecerem interfaces modulares para integração com *Application Programming Interface* (APIs), bancos de dados e ferramentas externas.

A arquitetura geral desses agentes compreende os seguintes componentes:

- **Interface de entrada**: responsável por receber a demanda do usuário (pergunta ou tarefa);
- **Módulo de processamento**: operado por LLMs, encarregado de interpretar, executar e formular a resposta à solicitação;
- **Camada de integração**: responsável pela conexão com fontes externas de informação, como o repositório *arXiv*;
- **Sistema de retorno**: encarregado de organizar e apresentar a resposta final ao usuário de forma clara e estruturada.

Essa arquitetura viabiliza a automação de tarefas complexas e introduz uma nova abordagem à exploração científica mediada por IA, promovendo eficiência, escalabilidade e aprofundamento analítico no processo de produção do conhecimento.

3.2.3. Arquitetura do *LangGraph*

O *LangGraph* é um *framework* projetado para a construção de fluxos de agentes inteligentes utilizando grafos direcionados, oferecendo maior controle, modularidade e transparência na tomada de decisões executadas por esses agentes. Três propriedades fundamentais distinguem sua arquitetura:

- **Ciclos (feedback loops)**: permitem a retroalimentação das saídas para agentes anteriores, promovendo o refinamento contínuo das respostas por meio de iterações sucessivas;
- **Controlabilidade**: refere-se à capacidade de manipular dinamicamente os nós do grafo de execução, viabilizando decisões condicionais e ajustes em tempo de execução;
- **Persistência**: possibilita a manutenção do estado entre interações, preservando o raciocínio e o histórico de execução ao longo de uma sessão.

A Figura 3.2 ilustra o funcionamento conceitual do *LangGraph*, evidenciando o fluxo de dados entre os nós, as decisões condicionais que modulam o percurso de execução e a presença de ciclos que viabilizam iterações sucessivas. Este modelo torna o

LangGraph especialmente apropriado para aplicações científicas baseadas em agentes, nas quais a robustez do processo decisório depende da capacidade de revisão, persistência e controle granular sobre cada etapa do fluxo.

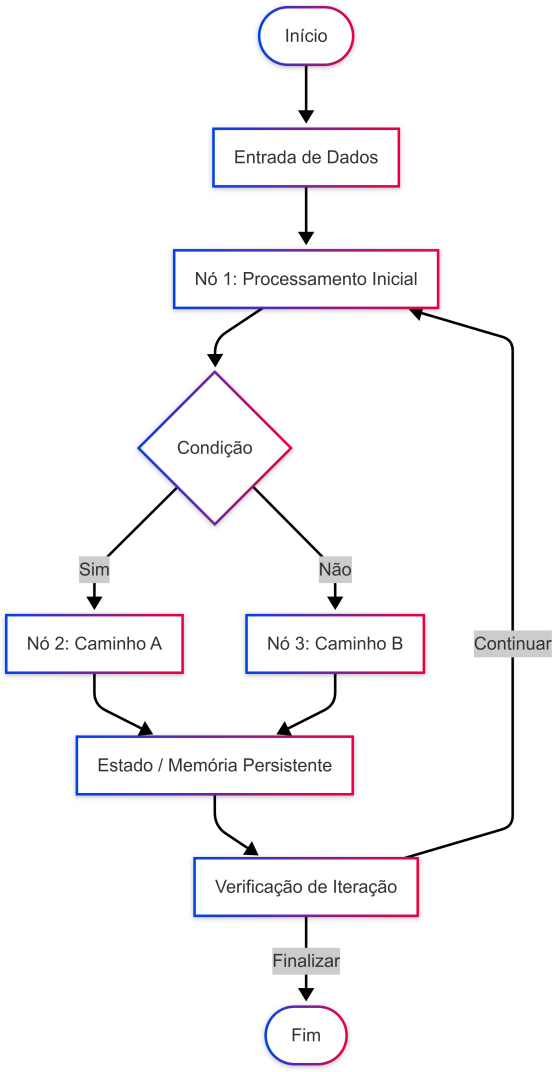

Figure 3.2: Funcionamento do LangGraph

Essas características tornam o *LangGraph* especialmente adequado para aplicações científicas, nas quais o refinamento progressivo, a validação iterativa e a integração com múltiplas fontes de informação são requisitos essenciais para garantir a robustez e a confiabilidade dos resultados.

3.2.4. Vantagens

O *LangGraph* permite a construção de aplicações com múltiplos atores e com gerenciamento de estado, utilizando LLMs de forma acessível e estruturada. O *framework* estende as funcionalidades do *LangChain* ao introduzir a capacidade de criar e gerenciar grafos cíclicos, o que se revela fundamental para a implementação de tempos de execução sofisticados em agentes inteligentes.

Seus principais conceitos incluem: (i) estrutura de grafo, (ii) gerenciamento de estado e (iii) mecanismos de coordenação. A partir desses fundamentos, o *LangGraph* oferece vantagens significativas para desenvolvedores que trabalham com aplicações complexas baseadas em LLMs, entre as quais destacam-se:

- **Desenvolvimento simplificado**: abstrações de alto nível facilitam a implementação de fluxos complexos;
- **Flexibilidade**: permite modificar os caminhos de execução em tempo real, adaptando-se a diferentes contextos e necessidades;
- **Escalabilidade**: suporta a construção de agentes com múltiplas etapas, integrados a diversas fontes de dados;
- **Tolerância a falhas**: possibilita retentativas e controle de estados intermediários, promovendo resiliência no processo computacional.

3.2.5. Comparação com *Frameworks* Semelhantes

Frameworks como ***LangChain, AutoGen*** e ***CrewAI*** oferecem abordagens eficazes para a construção de agentes baseados em LLMs. No entanto, diferem significativamente do *LangGraph* no que se refere ao grau de controle sobre o fluxo de execução. Enquanto os primeiros operam majoritariamente com sequências lineares ou estruturas do tipo árvore de decisão, o *LangGraph* permite a definição de grafos direcionados, com possibilidade de atualização dinâmica dos caminhos de execução, o que favorece a inclusão de ciclos de validação, refinamento iterativo e lógica condicional.

Um estudo recente (BARBARROXA; GOMES; VALE, 2024) analisou o desempenho de *frameworks* como *AutoGen*, *CrewAI* e TaskWeaver em tarefas de geração automatizada de código, evidenciando suas potencialidades e limitações. Embora o *LangGraph* não tenha sido incluído nessa análise, sua arquitetura modular representa uma evolução relevante em termos de flexibilidade, rastreabilidade e capacidade de adaptação em contextos científicos exigentes.

3.2.6. Aplicações Científicas e Relevância Prática

O uso de agentes baseados em IA tem se consolidado em diversas áreas da investigação científica, destacando-se em tarefas como a revisão automatizada de manuscritos, extração de conhecimento a partir de bases científicas (MELONI et al., 2023) e apoio a decisões clínicas e editoriais. A capacidade desses agentes de interagir com múltiplas fontes e processar grandes volumes de dados em linguagem natural os torna cada vez mais essenciais em um cenário caracterizado pela crescente complexidade e quantidade de informação disponível.

De acordo com a pesquisa de (LUO et al., 2019), arquiteturas baseadas em grafos promovem maior precisão na coordenação entre múltiplos agentes, o que acelera a obtenção de resultados relevantes. Nesse contexto, o *LangGraph* destaca-se como uma solução promissora para o desenvolvimento de sistemas inteligentes de apoio à pesquisa científica, permitindo que pesquisadores configurem agentes especializados, ajustáveis às particularidades e requisitos de seus projetos.

3.3. Arquitetura do Sistema

Um agente inteligente é um sistema que utiliza LLM para decidir o fluxo de controle de uma aplicação. À medida que esses sistemas evoluem, eles tendem a se tornar mais complexos, o que dificulta sua manutenção, escalabilidade e capacidade de resposta. Em determinados contextos, como na implementação de um agente voltado à busca bibliográfica e análise científica, a utilização de apenas um agente pode não ser suficiente.

Entre os desafios enfrentados em sistemas com agente único, destacam-se: a presença de múltiplas ferramentas que confundem a tomada de decisão sobre qual utilizar em cada momento; a complexidade crescente do contexto, que compromete a efetividade de um único raciocinador; e a necessidade de competências especializadas distintas (por exemplo, planejamento, recuperação de documentos, interpretação estatística, entre outras). Para lidar com essas limitações, uma abordagem eficiente é a decomposição da aplicação em múltiplos agentes menores, com funções bem definidas, compondo assim um sistema multiagente (DUAN; WANG, 2024).

Os principais benefícios da adoção de arquiteturas multiagentes incluem:

- **Modularidade**: a separação em agentes independentes facilita o desenvolvimento incremental, além de simplificar o teste e a manutenção dos componentes do sistema.
- **Especialização**: permite a criação de agentes especialistas focados em domínios ou funções específicas, o que contribui para a eficiência e precisão do sistema como um todo.
- **Controle explícito**: possibilita a orquestração detalhada da comunicação entre os agentes, reduzindo a dependência de mecanismos implícitos de chamada de funções e promovendo maior previsibilidade no comportamento do sistema.

Dessa forma, sistemas baseados em múltiplos agentes inteligentes representam uma evolução arquitetural relevante, especialmente em contextos acadêmicos e científicos, nos quais a complexidade das tarefas exige abordagens mais escaláveis, colaborativas e ajustáveis a fluxos de trabalho diversos.

3.3.1. Arquitetura Sistema Multiagente

O *LangGraph* oferece diferentes arquiteturas para a construção de sistemas multiagentes, permitindo que os desenvolvedores escolham a forma mais adequada de orquestrar a comunicação entre agentes. Entre as principais abordagens, destacam-se:

- ***Network***: todos os agentes podem se comunicar entre si. Qualquer agente pode decidir qual outro agente acionar em seguida.
- ***Supervisor***: todos os agentes se comunicam com um agente supervisor central, responsável por decidir qual agente será ativado em cada etapa.
- ***Hierárquico***: extensão da arquitetura de supervisor, onde há supervisores de supervisores, permitindo fluxos de controle mais complexos.

Neste trabalho, foi adotada uma variante da arquitetura do tipo ***Supervisor***, denominada *Supervisor com chamada de ferramentas (tool-calling)*. Nesse modelo, os agentes especializados são representados como ferramentas que ficam à disposição de um agente supervisor. Este utiliza um LLM com capacidade de raciocínio baseado em chamadas de ferramentas para decidir qual agente acionar, bem como quais argumentos serão repassados.

O agente supervisor segue uma lógica de execução baseada em laço (*while-loop*), realizando chamadas sucessivas até que uma condição de parada seja alcançada. A Figura 3.3 ilustra essa arquitetura, na qual o supervisor recebe uma entrada do usuário e, com base no conteúdo da solicitação, redireciona a tarefa ao agente mais apropriado.

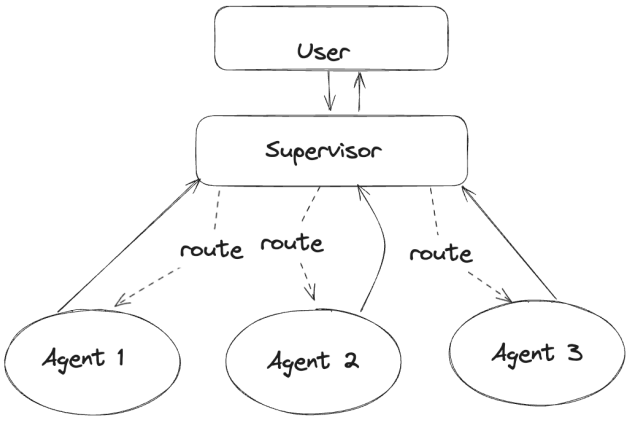

Figure 3.3: Arquitetura Agente Supervisor. Fonte: `https://github.com/langchain-ai/langgraph/blob/main/docs/docs/tutorials/multi_agent/agent_supervisor.ipynb`

No sistema desenvolvido, o agente supervisor coordena a atuação de três agentes especializados, descritos a seguir:

1. ***Tavily Agent***: realiza buscas na *web* por meio da API *Tavily*, adequado para consultas gerais e abertas;
2. ***arXiv Agent***: realiza buscas específicas em artigos científicos no repositório *arXiv*, além de realizar a ingestão dos dados no *ChromaDB*;
3. ***Scheduler Agent***: executa buscas científicas agendadas no *arXiv*, com controle de periodicidade, duração e número de resultados.

O fluxo de funcionamento é representado na Figura 3.4. Quando o usuário envia uma solicitação, o agente supervisor avalia seu conteúdo:

- Se a solicitação indicar a necessidade de uma pesquisa científica, o supervisor encaminha a demanda ao ***arXiv Agent***;

- Se a pergunta for mais ampla, como por exemplo - “qual a previsão do tempo?” -, o supervisor encaminha ao ***Tavily Agent***;
- Se o usuário desejar agendar uma pesquisa periódica — por exemplo, “pesquise artigos sobre transformers na IA durante 3 meses e me envie 10 artigos toda terça-feira” —, a tarefa é repassada ao ***Scheduler Agent***.

Caso a entrada do usuário não se enquadre em nenhum dos cenários esperados (por exemplo: “Oi, tudo bem?”), o supervisor simplesmente não aciona nenhum agente ou ferramenta, apenas responde de acordo a base de conhecimento pré-treinado do LLM.

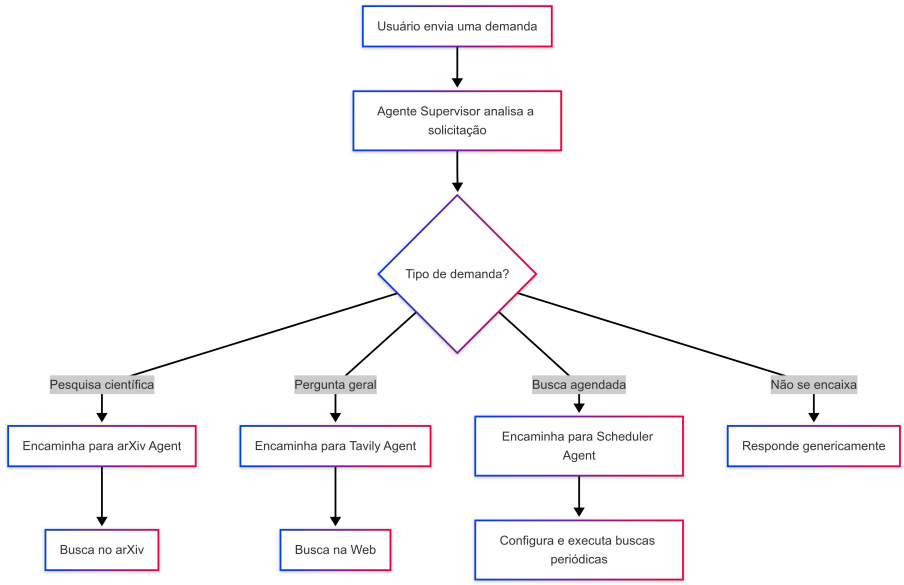

Figure 3.4: Fluxograma do funcionamento do sistema multiagente

Na Figura 3.5, é possível observar uma tela do sistema em operação, evidenciando a interação entre o usuário, o agente supervisor e os subagentes responsáveis por processar a solicitação.

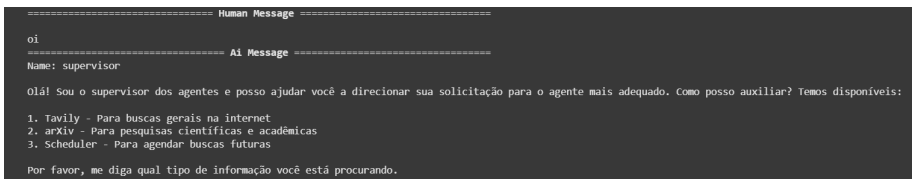

Figure 3.5: Funcionamento do agente

3.3.2. Tecnologias Utilizadas

Para o desenvolvimento do sistema multiagente voltado à realização de pesquisas científicas e buscas na *web*, foram empregadas tecnologias modernas e adequadas ao contexto de aplicações baseadas em modelos de linguagem. A seguir, apresentam-se as principais ferramentas utilizadas:

- ***Python***: utilizada como linguagem de programação principal devido à sua ampla adoção na comunidade de inteligência artificial, simplicidade sintática, vasta gama de bibliotecas voltadas ao desenvolvimento de agentes e suporte nativo a paradigmas como orientação a objetos e programação funcional. Além disso, o *Python* apresenta excelente integração com *frameworks* de IA, como *TensorFlow*, *PyTorch*, *LangChain* e *LangGraph*, facilitando tanto a prototipagem quanto o escalonamento de soluções complexas (KHANDARE et al., 2023).

- ***LangGraph***: *framework* de orquestração baseado em grafos direcionados, utilizado para a construção do sistema multiagente. Permite a criação de fluxos dinâmicos e cíclicos entre agentes, com controle refinado sobre os estados e transições, sendo ideal para aplicações que requerem tomada de decisão condicionada e persistência de estado ao longo de múltiplas interações.

- ***TavilySearch***: ferramenta de busca na *web* integrada via API, utilizada para fornecer respostas baseadas em fontes abertas e atualizadas. Sua principal função no sistema é atender às demandas informacionais gerais que extrapolam o escopo de bases científicas especializadas.

- ***ChromaDB***: banco de dados vetorial responsável pelo armazenamento de *embeddings* extraídos de documentos e artigos científicos. Distingue-se por sua eficiência, interface intuitiva, modelo de código aberto e desempenho otimizado para conjuntos de dados de pequena e média escala. Essas características tornam o *ChromaDB* particularmente apropriado para aplicações acadêmicas e sistemas multiagentes em fase de prototipagem. Além disso, apresenta elevada performance em tarefas de recuperação semântica, como demonstrado em estudos recentes (MATHUR; CHHABRA, 2024).

- **arXiv API**: fonte de dados especializada, utilizada para acesso estruturado a artigos científicos em formato XML. Permite consultas precisas por metadados (autores, palavras-chave, datas e resumos), sendo essencial para a operação do agente responsável por pesquisas bibliográficas.

- ***Anthropic Claude 3.5***: LLM de última geração utilizado para geração e interpretação de respostas dentro do sistema. Sua alta capacidade de raciocínio, compreensão de contexto extenso e desempenho superior em tarefas de múltiplos turnos o tornam adequado para funções de supervisão e coordenação de subagentes.

3.4. Ambiente de Desenvolvimento

Para que os leitores tenham uma experiência prática fluida, é fundamental estabelecer um *ambiente de desenvolvimento* padronizado. Nesta seção, são apresentados os requisitos de *software* e *hardware*, o passo a passo de instalação e configuração, bem como um exemplo inicial de uso do *LangGraph*. É necessário dispor de um editor de código, neste exemplo será utilizado o ***Visual Studio Code***, embora o leitor possa optar por outro de sua preferência e ter o **Python 3** previamente instalado na máquina.

3.4.1. Preparação e Configuração Inicial

1. **Clonar ou baixar o repositório:**

```
git clone https://github.com/larissaNa/LangGraphAgent.git
```

2. **Acessar a pasta do projeto:** No terminal, navegue até o diretório onde o repositório foi clonado:

```
cd <caminho-para-sua-pasta>
```

3. **Criar e ativar ambiente virtual Python**

```
python -m venv venv
source venv/bin/activate
venv scripts activate
```

4. **Instalar dependências:**

```
pip install -r requirements.txt
```

Para que o sistema multiagente funcione corretamente, é necessário obter e configurar as chaves de acesso (API keys) das ferramentas utilizadas.

- ***Anthropic API*** (ANTHROPIC, 2024): A *Anthropic* é uma empresa norte-americana especializada em IA, focada no desenvolvimento de LLMs responsáveis e confiáveis. Seu modelo *Claude-3.5* é utilizado neste sistema como supervisor do fluxo de agentes. A chave de API pode ser obtida no site oficial da empresa: `https://www.anthropic.com/`.

- ***Tavily API*** (TAVILY, 2024): O *Tavily* é uma ferramenta de busca baseada em IA voltada à realização de pesquisas rápidas e eficazes em diversas fontes da *web*. Permite a entrega de informações sintetizadas com alto grau de relevância. Sua API pode ser acessada via: `https://tavily.com/`.

3.4.2. Configuração do Arquivo `.env`

Após obter suas chaves, crie um arquivo `.env` na raiz do projeto e insira as informações da seguinte forma:

```
TAVILY_API_KEY="sua_chave"
ANTHROPIC_API_KEY="sua_chave"
```

Esse arquivo será utilizado pelo sistema para autenticar as chamadas às APIs externas de forma segura.

3.4.3. Execução do Sistema Multiagente

Com o ambiente devidamente configurado e as chaves de API fornecidas, o sistema pode ser executado com o seguinte comando no terminal:

```
python main.py
```

A partir desse ponto, o sistema estará operacional, permitindo ao usuário realizar pesquisas científicas automatizadas com suporte dos agentes especializados.

3.5. Implementação Prática

Nesta seção, será apresentado um guia passo a passo para desenvolver, testar e validar um agente de pesquisa científica utilizando o *framework LangGraph*. A solução integra buscas gerais na *web* com *Tavily*, consultas científicas ao *arXiv* e a capacidade de agendamento de pesquisas periódicas. Os trechos de código a seguir foram extraídos e adaptados do repositório apresentado na seção anterior.

3.5.1. Definindo Variáveis de Ambiente

A primeira etapa do sistema consiste em garantir que as chaves das APIs estejam acessíveis por variáveis de ambiente. O código abaixo solicita essas chaves via terminal caso ainda não estejam definidas:

Listing 3.1: Definindo as APIs Keys

```
{
    def _set_env(var: str):
        if not os.environ.get(var):
            os.environ[var] = getpass.getpass(f"{var}: ")

    _set_env("TAVILY_API_KEY")
    _set_env("ANTHROPIC_API_KEY")
}
```

Esse método permite maior segurança e flexibilidade, evitando o hardcode de credenciais sensíveis.

3.5.2. Inicialização do Modelo de Linguagem e do Banco Vetorial

O sistema utiliza o modelo `Claude 3.5 Sonnet` da *Anthropic* como núcleo dos agentes, além de um banco vetorial local com o `ChromaDB` para armazenamento e consulta semântica dos artigos do arXiv:

Listing 3.2: Configurando LLM e chromaDB

```
{
    # --- Modelo base ---
    llm = init_chat_model("anthropic:claude-3-5-sonnet-latest")

    # 3. Configura ChromaDB
    embeddings = HuggingFaceEmbeddings(model_name="sentence-
        transformers/all-MiniLM-L6-v2")
    vectorstore = Chroma(
        collection_name="artigos_arxiv",
        embedding_function=embeddings,
        persist_directory="./chroma_db"
)
}
```

3.5.3. Criação dos Agentes Especializados

A seguir, são definidos agentes especializados para diferentes tarefas, cada um com suas ferramentas e instruções específicas. Cada agente segue o padrão *ReAct*, ou seja, combina raciocínio com ações sobre ferramentas:

Listing 3.3: Configurando os agentes

```
{

agent = create_react_agent(model=llm, tools=tools)

    # Agente Tavily (busca geral)
tavily_agent = create_react_agent(
    model=llm,
    tools=[tavily_tool],
    prompt= 'You perform web searches',
    name="tavily_agent"
)

# Agente arXiv (pesquisa cient fica)
arxiv_agent = create_react_agent(
    model=llm, tools=[arxiv_search_and_ingest], name="arxiv_agent",
    prompt="You search, ingest into ChromaDB and report arXiv papers."
)

sched_agent  = create_react_agent(
    llm,
    tools=[schedule_research, cancel_research],
    prompt="You schedule or cancel periodic arXiv searches.",
    name="scheduler_agent"
)
)
}
```

3.5.4. Supervisor e Orquestração Multiagente

O supervisor é o componente central do sistema. Ele analisa a solicitação do usuário e decide qual agente deverá ser acionado.

Listing 3.4: Configurando e compilando o supervisor

```
{

supervisor_graph = create_supervisor(
    model=llm,
    agents=[tavily_agent, arxiv_agent, sched_agent],
    prompt=(
        "Voc     um supervisor que delega ao agente mais adequado
           entre:\n"
        "- Tavily (buscas gerais)\n"
        "- arXiv (buscas imediatas)\n"
        "- Scheduler (buscas agendadas)\n"
        "Ap s obter a resposta, sempre gere UMA mensagem final clara
           ao usu rio."
    ),
    add_handoff_messages=True, #permite rastrear a comunica  o entre
       supervisor e agentes, tornando a execu  o mais interpret vel
       .
    add_handoff_back_messages=True,
    output_mode="full_history"
)

compiled_supervisor = supervisor_graph.compile()
}
```

3.5.5. Compilando o Grafo de Estado com LangGraph

Após definir os componentes, o sistema é estruturado como um grafo de estado que inicia no supervisor. A estrutura abaixo permite escalar o sistema com mais nós e ramificações no futuro, mantendo a modularidade e o controle sobre os estados e transições.

Listing 3.5: Monta StateGraph e compila

```
{

class StateSchema(TypedDict):
    messages: Annotated[list[BaseMessage], add_messages]

graph = StateGraph(StateSchema)
graph.add_node("supervisor", compiled_supervisor)
graph.add_edge(START, "supervisor")
compiled = graph.compile(checkpointer=MemorySaver())
}
```

3.5.6. Execução Interativa do Sistema

Por fim, é definido um loop de execução interativa, permitindo ao usuário conversar diretamente com o sistema no terminal:

Listing 3.6: Loop interativo

```
{

if __name__ == "__main__":
    while True:
        ui = input("User: ")
        if ui.lower() in ["exit","quit","q"]:
            print("Encerrando...")
            break
        run(ui)
}
```

Esse loop inicia a inferência do grafo de agentes com base na entrada do usuário, realizando as delegações e execuções necessárias.

A Figura 3.6 ilustra a arquitetura interna do sistema multiagente implementado com o *framework LangGraph*. Diferentemente do diagrama inicial, que apresenta uma visão geral do sistema proposto, este diagrama detalha o fluxo de interação entre o usuário, o supervisor e os agentes especializados *(Tavily, arXiv e Scheduler)*, bem como sua integração com o banco vetorial *ChromaDB*. Essa estrutura possibilita a delegação inteligente de tarefas de busca e agendamento, promovendo uma execução dinâmica e modular baseada em agentes autônomos coordenados por um nó supervisor.

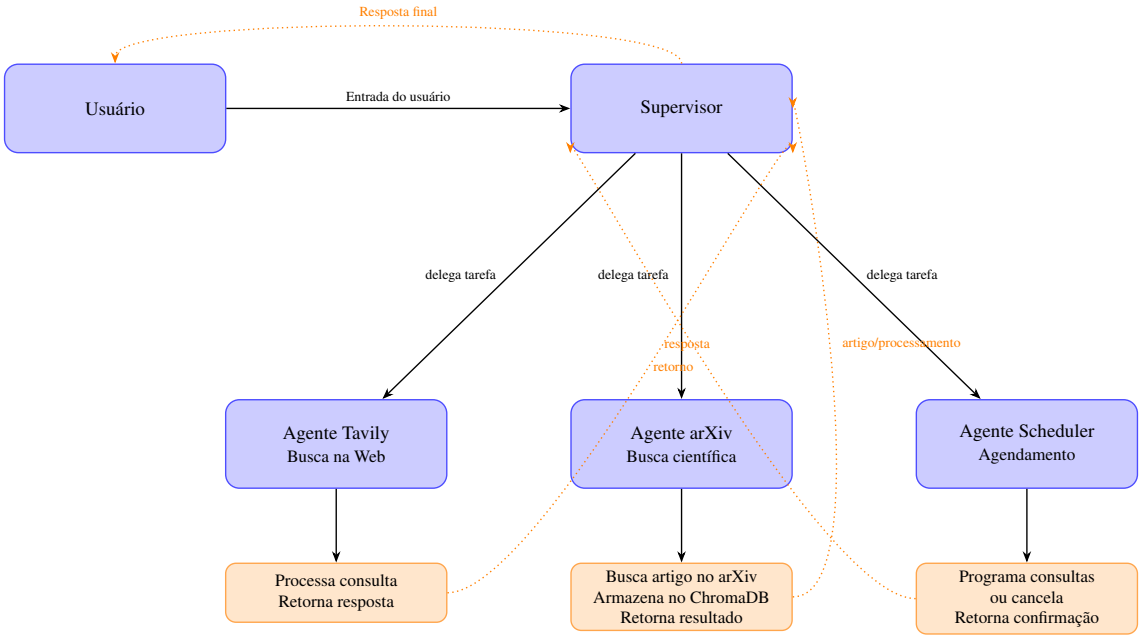

Figure 3.6: Arquitetura do sistema multiagente com supervisor e subagentes especializados.

3.6. Otimização e Melhoria

A construção de agentes baseados em modelos de linguagem de última geração requer não apenas boas práticas de desenvolvimento, mas também estratégias que assegurem confiabilidade, interpretabilidade e desempenho contínuo. Esta seção discute técnicas

aplicáveis à redução de alucinações, métodos para elevar a robustez das respostas geradas e caminhos para evolução futura do sistema proposto.

3.6.1. Redução de Alucinações por LLMs

Embora os LLMs sejam altamente eficazes na geração de texto natural, eles estão sujeitos a um fenômeno conhecido como alucinação, em que informações incorretas ou inventadas são apresentadas como verdadeiras. Para mitigar esse problema no agente proposto, é possível aplicar duas estratégias complementares:

- ***Re-ranking* de Resultados**: Após a obtenção de múltiplas respostas a partir de fontes externas (como buscas na *web* ou no *arXiv*), um mecanismo de reclassificação pode ser utilizado para ordenar os resultados com base em critérios como similaridade semântica ao *prompt* original, confiabilidade da fonte e presença de citações verificáveis.

- **Checagem Cruzada (*Cross-checking*)**: A verificação cruzada entre diferentes agentes ou ferramentas, como o *Tavily* (busca *web*) e o agente do arXiv (pesquisa científica), permite confirmar a consistência das informações recuperadas. Essa abordagem fortalece a confiança do sistema ao validar evidências em múltiplas fontes independentes (ALMEIDA DA SILVA et al., 2024).

Além disso, a utilização de *retrieval-augmented generation* (RAG) com base vetorial, como *ChromaDB*, também contribui para a fundamentação das respostas, ancorando-as em documentos reais e previamente armazenados.

3.6.2. Aumento da Confiabilidade do Agente

Para garantir maior confiabilidade ao longo do uso do agente, propõem-se medidas como:

- **Auditoria de Histórico de Conversas**: Armazenar os diálogos completos permite reanalisar decisões, treinar classificadores de erros e melhorar respostas futuras.

- ***Logs* Explicáveis por Estado**: Como o agente é construído com *LangGraph*, que segue uma estrutura baseada em grafos de estado, cada transição pode ser registrada de forma transparente, oferecendo interpretabilidade das ações realizadas pelo sistema.

- **Avaliação Contínua por Métricas Customizadas**: Métricas específicas podem ser desenvolvidas para avaliar a precisão das respostas científicas, considerando fatores como uso de fontes primárias, atualidade da informação e cobertura temática.

A combinação dessas práticas reforça a confiança dos usuários no sistema, promovendo maior adoção e segurança em aplicações reais, especialmente no domínio acadêmico.

3.6.3. Perspectivas de Evolução do Projeto

Após a realização do minicurso, diversas direções podem ser exploradas para ampliar o escopo e a sofisticação do agente:

- **Integração com bases de dados institucionais**: Permitir buscas internas em repositórios acadêmicos locais (como o SUAP ou repositórios do IFPI) ampliaria a utilidade do agente para estudantes e pesquisadores da própria instituição.

- **Personalização por Perfil de Usuário**: Implementar perfis adaptativos (aluno, professor, pesquisador) permitiria ajustar o comportamento do agente de acordo com as necessidades específicas de cada grupo.

- **Avaliação Colaborativa por Comunidade**: A criação de mecanismos para que os usuários avaliem as respostas, sugiram melhorias ou reportem inconsistências contribuiria para uma melhoria contínua baseada em dados reais de uso.

- **Multimodalidade e Explicações Visuais**: Com o avanço de modelos capazes de interpretar e gerar imagens, gráficos e vídeos, o agente poderia incorporar respostas multimodais para enriquecer a experiência de busca científica.

A incorporação dessas melhorias visa não apenas ampliar a funcionalidade do sistema, mas também consolidá-lo como uma ferramenta de apoio confiável e adaptável em contextos educacionais e de pesquisa.

3.7. Considerações Éticas

O desenvolvimento e a aplicação de agentes inteligentes para apoio à pesquisa científica exigem atenção rigorosa às implicações éticas, especialmente no que se refere à confiabilidade das informações geradas, à transparência das decisões tomadas e ao impacto social de seu uso. Esta seção discute aspectos centrais relacionados ao viés algorítmico, à explicabilidade dos modelos e à responsabilidade no uso da IA em contextos científicos.

3.7.1. Viés Algorítmico

Os modelos de LLMs, como os utilizados neste projeto, são treinados com vastos volumes de dados disponíveis na internet, muitos dos quais refletem desigualdades sociais, estereótipos e preconceitos históricos. Isso pode resultar em comportamentos enviesados, mesmo em tarefas técnicas ou aparentemente neutras. No contexto da ciência, esses vieses podem influenciar, por exemplo, o tipo de artigo recuperado, as fontes priorizadas ou as interpretações sugeridas pelos agentes. Como discutido por (BENDER et al., 2021), o viés não é apenas um problema técnico, mas também epistemológico, uma vez que afeta a forma como o conhecimento é produzido e difundido.

Portanto, é fundamental incorporar mecanismos de mitigação de viés, como *re-ranking* supervisionado, validação cruzada com múltiplas fontes e auditoria contínua dos resultados gerados pelos agentes.

3.7.2. Transparência e Explicabilidade

Outro aspecto crítico diz respeito à transparência dos sistemas baseados em IA. A opacidade de modelos como o *Claude* ou o *GPT-4.5* dificulta a compreensão do processo pelo qual uma determinada resposta é produzida. Isso compromete a capacidade do usuário de avaliar a confiabilidade das informações e dificulta a rastreabilidade de erros ou desvios.

Para mitigar essa limitação, é necessário empregar abordagens de RAG, que permitem rastrear a origem dos conteúdos apresentados. Além disso, o uso de *logs* detalhados do agente supervisor oferece maior visibilidade sobre o fluxo de decisões internas, ampliando a explicabilidade do sistema (LEWIS et al., 2021). A transparência é não apenas um requisito técnico, mas também ético, especialmente quando os agentes são aplicados em contextos científicos e educacionais.

3.7.3. Responsabilidade no Uso da IA em Ciência

A introdução de agentes autônomos para apoiar a produção de conhecimento científico impõe desafios significativos em termos de responsabilidade. Embora esses sistemas ampliem a capacidade de busca e análise, eles não substituem o papel crítico do pesquisador humano na interpretação dos dados e na formulação de hipóteses.

É necessário estabelecer limites claros de uso, bem como orientações éticas para a adoção responsável dessas tecnologias. Além disso, reforça-se a importância da formação dos usuários quanto às limitações dos sistemas de IA, para evitar uma confiança cega nas respostas geradas (PEROV; PEROVA, 2024).A responsabilização deve ser compartilhada entre desenvolvedores, usuários e instituições que promovem o uso de IA em ambientes acadêmicos.

Por fim, propõe-se que iniciativas como esta incluam diretrizes explícitas de ética em IA, como parte dos materiais educacionais do projeto, promovendo uma cultura de uso crítico, consciente e transparente da inteligência artificial.

3.8. Conclusão

Este capítulo apresentou, de forma prática e fundamentada, o desenvolvimento de um sistema multiagente de apoio à pesquisa científica, utilizando a biblioteca ***LangGraph***. A proposta buscou demonstrar como agentes inteligentes, quando bem arquitetados e integrados a ferramentas específicas como o ***Tavily AI*** e o ***arXiv***, podem auxiliar pesquisadores em tarefas como levantamento bibliográfico, busca em bases científicas e agendamento automatizado de pesquisas periódicas.

Ao longo da implementação, foram abordadas desde questões de configuração do ambiente de desenvolvimento até a estruturação do grafo supervisor, que permite a coordenação entre agentes especializados. A abordagem modular e interpretável do LangGraph mostrou-se eficaz para a construção de fluxos interativos mais transparentes, além de facilitar a extensibilidade do sistema.

Entre os principais aprendizados, destaca-se a importância de pensar a inteligência artificial não como substituta da atividade científica, mas como ferramenta complementar que amplifica a capacidade humana de análise, filtragem e interpretação de informações. A combinação entre grandes modelos de linguagem, bancos vetoriais e mecanismos de

agendamento configura um cenário promissor para o uso de IA em atividades de apoio à ciência, especialmente em tempos de sobrecarga informacional.

O impacto esperado da aplicação prática do *LangGraph* vai além da automação. Espera-se que essa tecnologia contribua para a democratização do acesso à informação científica, otimize processos de revisão de literatura e reduza barreiras técnicas para pesquisadores iniciantes ou com menor familiaridade com ferramentas computacionais.

Como caminhos futuros, vislumbra-se a integração do sistema com outras bases científicas e plataformas de gestão científica, o aperfeiçoamento dos mecanismos de avaliação da relevância dos artigos encontrados e a adoção de técnicas de *fine-tuning* para adequar os agentes aos domínios específicos de diferentes áreas do conhecimento. Além disso, propõe-se o avanço em estratégias de mitigação de vieses e a incorporação de métricas de confiabilidade nos fluxos de decisão dos agentes.

Por fim, acredita-se que a formação de pesquisadores sobre o uso ético, crítico e transparente da IA será determinante para o sucesso de iniciativas como esta, que buscam alinhar inovação tecnológica e integridade científica.

References

ALMEIDA DA SILVA, Wildemarkes de et al. Mitigation of Hallucinations in Language Models in Education: A New Approach of Comparative and Cross-Verification. In: 2024 IEEE International Conference on Advanced Learning Technologies (ICALT). [S.l.: s.n.], 2024. P. 207–209. DOI: 10.1109/ICALT61570.2024.00066.

ANTHROPIC. **Anthropic key**. [S.l.: s.n.], 2024. Acessado em 20 de março de 2025. Available from: <https://www.anthropic.com/>.

BARBARROXA, Rafael; GOMES, Luis; VALE, Zita. Benchmarking Large Language Models for Multi-agent Systems: A Comparative Analysis of AutoGen, CrewAI, and TaskWeaver. In_____. **Advances in Practical Applications of Agents, Multi-Agent Systems, and Digital Twins: The PAAMS Collection**. Cham: Springer Nature Switzerland, 2024. P. 39–48. ISBN 978-3-031-70415-4.

BENDER, Emily M. et al. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In: PROCEEDINGS of the 2021 ACM Conference on Fairness, Accountability, and Transparency. Virtual Event, Canada: Association for Computing Machinery, 2021. (FAccT '21), p. 610–623. ISBN 9781450383097. DOI: 10.1145/3442188.3445922. Available from: <https://doi.org/10.1145/3442188.3445922>.

CHASE, Harrison. **LangGraph Documentation**. [S.l.: s.n.], 2024. Acessado em 14 de março de 2025. Available from: <https://langchain-ai.github.io/langgraph/>.

DUAN, Zhihua; WANG, Jialin. **Exploration of LLM Multi-Agent Application Implementation Based on LangGraph+CrewAI**. [S.l.: s.n.], 2024. arXiv: 2411.18241 [cs.MA]. Available from: <https://arxiv.org/abs/2411.18241>.

FIORILLO, Luca; MEHTA, Vini. Accelerating editorial processes in scientific journals: Leveraging AI for rapid manuscript review. **Oral Oncology Reports**, v. 10, p. 100511, 2024. ISSN 2772-9060. DOI: https://doi.org/10.1016/j.oor.2024.100511. Available from: <https://www.sciencedirect.com/science/article/pii/S2772906024003571>.

KHANDARE, Anand et al. Analysis of Python Libraries for Artificial Intelligence. In_____. **Intelligent Computing and Networking**. Singapore: Springer Nature Singapore, 2023. P. 157–177. ISBN 978-981-99-0071-8.

LEWIS, Patrick et al. **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**. [S.l.: s.n.], 2021. arXiv: 2005.11401 [cs.CL]. Available from: <https://arxiv.org/abs/2005.11401>.

LUO, Tianze et al. Multi-Agent Collaborative Exploration through Graph-based Deep Reinforcement Learning. In: 2019 IEEE International Conference on Agents (ICA). [S.l.: s.n.], 2019. P. 2–7. DOI: 10.1109/AGENTS.2019.8929168.

MANZOOR, Umar et al. Automating Academic Tasks (AAT) - An Agent Based Approach. In: 2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems. [S.l.: s.n.], 2012. P. 1770–1775. DOI: 10.1109/HPCC.2012.266.

MATHUR, Srushti; CHHABRA, Aayush. Vector Search Algorithms: A Brief Survey. In: 2024 4th International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS). [S.l.: s.n.], 2024. P. 365–371. DOI: 10.1109/ICUIS64676.2024.10866377.

MELONI, Antonello et al. Integrating Conversational Agents and Knowledge Graphs Within the Scholarly Domain. **IEEE Access**, v. 11, p. 22468–22489, 2023. DOI: 10.1109/ACCESS.2023.3253388.

PEROV, Vadim; PEROVA, Nina. AI Hallucinations: Is “Artificial Evil” Possible? In: 2024 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT). [S.l.: s.n.], 2024. P. 114–117. DOI: 10.1109/USBEREIT61901.2024.10584048.

TAVILY. **Tavily key**. [S.l.: s.n.], 2024. Acessado em 20 de março de 2025. Available from: <https://www.tavily.com/>.