Revista Eletrônica de Iniciação Científica em Computação, 2026, 24:1
ISSN: 3085-8461 • doi: 10.5753/reic.2026.7184

Artigo de Pesquisa/Research Paper

Implementação de um Workflow Multiagentes para Recuperação de Informação Organizacional

Implementation of a Multi-Agent Workflow for Organizational Information Retrieval

Guilherme L. Moretti [Universidade Federal do Ceará - Campus de Quixadá | *Guilherme.moretti@alu.ufc.br*]
Kalmax dos S. Sousa [Universidade Federal do Ceará - Campus de Quixadá |*kalmaxs@alu.ufc.br*]
Matheus dos S. Mendes [Universidade Federal do Ceará - Campus de Quixadá |*matheusmendes@alu.ufc.br*]
Marcos de Oliveira [Universidade Federal do Ceará - Campus de Quixadá |*marcos.oliveira@ufc.br*]

Universidade Federal do Ceará (UFC) Campus de Quixadá Av. José de Freitas Queiroz, 5003 – Cedro Novo – CEP 63902-580 – Quixadá – CE – Brazil

Resumo. Este artigo propõe um sistema multiagente baseado em Geração Aumentada por Recuperação (RAG) para apoiar a gestão do conhecimento em ambientes corporativos. A solução combina a recuperação semântica de documentos com a geração de respostas utilizando grandes modelos de linguagem (LLMs), permitindo um acesso rápido e contextualizado às informações organizacionais. Ao empregar agentes especializados em domínios e uma arquitetura de orquestração inteligente, o sistema fornece respostas precisas e adaptadas a diferentes níveis de tomada de decisão. Essa abordagem visa superar os desafios impostos pela fragmentação e pelo volume de documentos institucionais, promovendo escalabilidade, modularidade e eficiência na recuperação interna de conhecimento.

Abstract. This paper proposes a multi-agent system based on Retrieval Augmented Generation (RAG) to support knowledge management in corporate environments. The solution combines semantic document retrieval with response generation using large language models (LLMs), enabling fast and contextua lized access to organizational information. By employing domain specialized agents and an intelligent orchestration architecture, the system provides accurate responses tailored to different decision-making levels. This approach aims to overcome challenges posed by the fragmentation and volume of institutional documents, promoting scalability, modularity, and efficiency in internal knowledge retrieval.

Palavras-chave: RAG, LLM, Multiagente, Recuperação de Conhecimento

Keywords: RAG, LLM, Multi-agent, Knowledge Retrieval

Recebido/Received: 14 December 2025 • **Aceito/Accepted:** 14 May 2026 • **Publicado/Published:** 12 June 2026

1 Introdução

No contexto empresarial contemporâneo, o volume e a heterogeneidade de documentos dificultam a organização e o acesso ágil ao conhecimento interno. A problemática da sobrecarga informacional, destacada por autores como Davenport and Prusak [2002], evidencia que o excesso de dados, muitas vezes descontextualizados, pode levar à paralisia decisória e à dificuldade em discernir o que é verdadeiramente relevante. Sistemas de Gestão do Conhecimento (KMS, do inglês *Knowledge Management Systems*) procuram capitalizar o saber organizacional, mas frequentemente se deparam com limitações de escalabilidade diante de bases documentais extensas e fragmentadas, dificultando o acesso rápido e preciso a informações relevantes para a tomada de decisão.

Sob a perspectiva da hierarquia de dados–informação–conhecimento proposta por Ackoff [1989], sistemas modernos de gestão do conhecimento buscam ir além da simples recuperação de documentos, fornecendo conhecimento estruturado e contextualizado ao usuário. Nesse contexto, Sistemas de Recuperação do Conhecimento (SRC) emergem como uma evolução dos tradicionais Sistemas de Recuperação de Informação, ao focarem na extração, organização e apresentação de informações relevantes para apoiar a tomada de decisão Yao *et al*. [2007].

A adoção de Sistemas Multi-Agentes (SMA) tem ganhado destaque em aplicações que requerem processamento distribuído e adaptativo, como robótica colaborativa, logística inteligente e gestão de dados corporativos [Maldonado *et al*., 2024]. Esses sistemas permitem que agentes autônomos especializados atuem de forma coordenada, resolvendo problemas complexos com maior eficiência do que abordagens centralizadas. No domínio da gestão do conhecimento, a integração de técnicas de Geração de Recuperação Aumentada (RAG, do inglês *Retrieval-Augmented Generation*) com Grandes Modelos de Linguagem (LLM, do inglês *Large Language Models*) amplia a capacidade de sistemas tradicionais. A abordagem RAG, que combina recuperação semântica de documentos com a geração contextualizada de respostas via LLMs [Lewis *et al*., 2020], oferece suporte decisório mais ágil e baseado em evidências, tornando as respostas mais precisas e relevantes.

Este trabalho apresenta o desenvolvimento de um SMA com arquitetura baseada em RAG, projetado para a gestão da informação corporativa através da análise automatizada de documentos. O sistema interpreta conteúdos diversos e fornece respostas ágeis, atendendo tanto a usuários operacionais, com demandas pontuais, quanto aos níveis gerencial e estratégico, que necessitam de dados agregados e contextualizados.

Dentre as tecnologias que podem suportar o desenvolvimento de um sistema de recuperação de conhecimento, podemos ressaltar o processamento de linguagem natural, agentes inteligentes [Yao *et al*., 2007] e RAG [Lewis *et al*., 2020]. A combinação das capacidades de agentes inteligentes com RAG para realização de tarefas complexas de recuperação da informação que exigem entendimento das necessidades do usuário, em especial a pacacidade do sistema de criar e executar um plano composto de tarefas menores que juntas atingem um objetivo principal de recuperação da informação [Silva *et al*., 2025].

Nesse cenário, SMAs apresentam-se como uma alternativa promissora para mitigar as limitações de arquiteturas RAG tradicionais. Ao permitir a decomposição de tarefas complexas em subtarefas executadas por agentes especializados e coordenados dinamicamente, SMA oferecem maior flexibilidade, modularidade e adaptação ao contexto da consulta, características essenciais para a recuperação de conhecimento organizacional.

2 Trabalhos Relacionados

2.1 A Evolução para o RAG Avançado

O paradigma RAG, introduzido por [Lewis *et al*., 2020], estabeleceu um método eficaz para fundamentar as respostas de LLMs em fontes de conhecimento externas. Contudo, a arquitetura RAG inicial, ou "ingénua", que consiste num fluxo sequencial simples de recuperação-geração, enfrenta desafios significativos, como a recuperação de documentos de baixa relevância ou a incapacidade de sintetizar informações contraditórias [Li *et al*., 2022]. Estas limitações impulsionaram a pesquisa em direção ao "RAG Avançado", que se caracteriza pela adição de múltiplos estágios ao *pipeline*. Estratégias comuns incluem a reescrita de consultas para otimizar a recuperação, o re-ranqueamento dos documentos recuperados para priorizar os mais relevantes, e a validação da resposta final com base nas fontes. Esta tendência de segmentar o processo RAG em etapas distintas e mais inteligentes cria uma base natural para a aplicação de paradigmas de decomposição de tarefas.

2.2 A Decomposição de Tarefas com Sistemas Multiagentes

Paralelamente, o paradigma de SMA demonstrou grande sucesso na resolução de problemas complexos ao decompor uma tarefa em subtarefas executadas por agentes especializados que colaboram entre si [Wooldridge, 2009]. Com o advento dos LLMs, este paradigma foi revitalizado. Um exemplo proeminente é o ChatDev [Qian *et al*., 2024], um *framework* onde múltiplos agentes (ex: "programador", "testador", "documentador") interagem para completar um ciclo de desenvolvimento de *software*. De forma semelhante, o *framework* MetaGPT [Hong *et al*., 2024] incorpora a ideia de Procedimentos Operacionais Padrão (SOPs) para coordenar agentes que assumem diferentes papéis numa empresa de *software* virtual. Um exemplo recente de utilização de agentes para realização do processo de RAG é o MA-RAG, proposto por [Nguyen *et al*., 2025]. Os autores implementam o processo de um RAG por meio de um sistema multi-agente baseado em LLMs que realiza ciclos de recuperação a partir de um plano que é gerado com base na consulta original do usuário, aprimorando o plano com base nas informações recuperadas após cada ciclo de recuperação. Testes realizados em *datasets* de *Question Answering* como o HotpotQA [Yang *et al*., 2018] provaram que o sistema tem a capacidade de responder consultas complexas com alta precisão.

O sucesso destes sistemas valida a hipótese de que a especialização de agentes e a orquestração dinâmica do fluxo de trabalho podem levar a resultados mais robustos e coerentes do que uma abordagem monolítica com um único agente.

2.3 Posicionamento e Contribuição

Como a Tabela 1 demonstra, a arquitetura diferencia-se por aplicar a especialização de agentes para refinar o fluxo de trabalho de RAG, utilizando um grafo de estados para uma orquestração flexível e iterativa, contando com agentes para planejamento, roteamento, validação e controle do fluxo de trabalho. Esta abordagem traduz os princípios do RAG Avançado num sistema multiagente robusto e adaptável a cenários diversos.

3 Metodologia

A metodologia adotada neste estudo fundamenta-se na integração entre princípios de sistemas multiagentes e arquiteturas de RAG avançadas, explorando a especialização de agentes e a orquestração dinâmica do fluxo de recuperação como mecanismos centrais para lidar com consultas complexas e multi-hop, ilustrada pela figura 1.

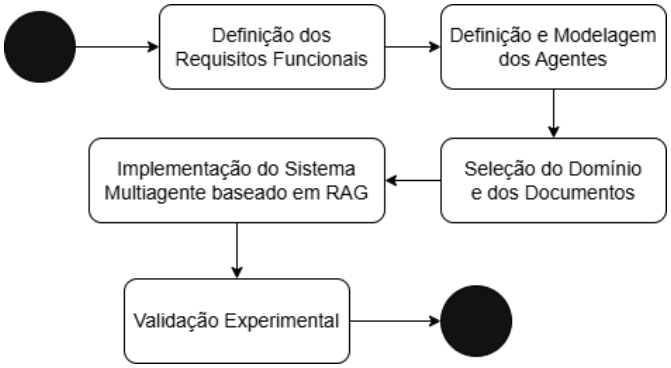

Figura 1. Fluxo metodológico. A figura descreve o fluxo adotado para o desenvolvimento e validação do sistema multiagente baseado em RAG.

3.1 Concepção do Sistema

O sistema foi projetado para fornecer respostas fundamentadas a partir de documentos organizacionais, partindo dos seguintes requisitos funcionais:

1. Receber documentos PDF de múltiplos domínios;
2. Executar ingestão e indexação automática com *embeddings*;
3. Rotear consultas para o domínio mais relevante;
4. Gerar respostas justificadas por evidências extraídas dos documentos com base nas consultas;

Para atender a tais requisitos, foi adotada uma estratégia híbrida que combina armazenamento vetorial, planejamento, recuperação semântica, validação de relevância e geração de linguagem natural, cada qual delegado a um agente especializado. O ponto principal da arquitetura proposta é a capacidade

Tabela 1. Análise comparativa da abordagem proposta

Trabalho	Arquitetura	Foco da Decomposição	Orquestração	Validação Interna Explícita
[Lewis *et al*., 2020]	Pipeline monolítico	N/A (Não se aplica)	Sequencial simples	Não
[Qian *et al*., 2024]	Multiagente	Tarefa (Eng. de Software)	Protocolo de chat (cascata)	Sim (agente testador)
[Hong *et al*., 2024]	Multiagente	Papéis (Empresa de Software)	SOPs e Mensagens	Sim (agente QA)
[Nguyen *et al*., 2025]	Multiagente	Processo (RAG)	Grafo de estados	Não
Proposta deste artigo	**Multiagente**	**Processo (RAG)**	**Grafo de estados**	**Sim (agente validador)**

de planejamento e iteração dinâmica, desenvolvido com o intuito de anteder as necessidades de consultas complexas que necessitam de múltiplas etapas para serem respondidas.

3.2 Definição dos Agentes

O sistema foi modelado como um grafo direcionado cíclico, no qual cada nó representa um agente autônomo. O sistema realiza a decomposição de consultas *multi-hop*[1] [Yang *et al*., 2018], que necessitam de múltiplos passos ou fontes de informação para serem respondidas, em consultas *single-hop*[2] independentes, permitindo uma recuperação de informações mais precisa e dinâmica de acordo com as necessidades do usuário, além de prevenir alucinações [Huang *et al*., 2025]. As consultas resultantes do processo de decomposição são armazenadas e executadas de forma sequencial por meio de um ciclo iterativo que pode parar caso o sistema já tenha informações suficientes para responder a consulta original, reduzindo o custo de processamento total.

Ao todo, o sistema conta com sete agentes distintos. Ao contrário de um *pipeline* fixo, o sistema realiza a orquestração sob demanda de agentes de acordo com a necessidade da tarefa realizada. Essa característica pode ser evidenciada nos agentes `Planner`, `Validator` e `StepDefiner`, que podem decidir dinamicamente o fluxo de trabalho. A definição dos agentes é descrita a seguir:

Planner. O agente `Planner` realiza a tarefa de analise e decomposição da consulta. Caso a consulta possa ser realizada com apenas uma busca (*single-hop*) onde o agente retorna a mesma consulta otimizada para recuperação eficiente. Caso a consulta precise de mais de uma busca o agente analisa os pontos principais da consulta do usuário e cria uma lista de consultas independentes com objetivos específicos para se responder a consulta original.

Router. Define o domínio de uma sub-consulta com base nos domínios disponíveis. Os domínios estão contidos no *prompt* do agente juntamente com suas respectivas descrições, tendo como objetivo orientar a decisão do agente. O agente é orientado a não escolher domínios onde houveram falhas na recuperação por meio de uma lista de domínios proibidos.

Librarian. Realiza a recuperação dos trechos de texto (*chunks*) de maior relevância com base na consulta atual.

Extractor. Processa os trechos recuperados pelo `Librarian`, identificando e resumindo as informações necessárias para responder a consulta atual.

Validator. Ao final de cada processo de recuperação realizado pelos agentes `Librarian` e `Extractor`, o `Validator` analisa as evidências recuperadas respondem à sub-consulta atual, podendo aprovar ou rejeitar. No caso de rejeição, o agente adiciona o domínio atual à lista de domínios proibidos.

StepDefiner. O agente `StepDefiner` tem como objetivo analisar as evidências coletadas e decidir se o próximo passo de acordo com a situação. Ao adicionar as evidências coletadas até o momento, a consulta original e a próxima subconsulta, o agente consegue realizar uma decisão inteligente e contextualizada, podendo encerrar o ciclo de iteração se possível, evitando gastos desnecessários de processamento.

Editor. Compila as todas as evidências recuperadas após o processo de recuperação, respondendo a consulta original do usuário com base nas informações recuperadas de forma coesa e legível.

As interações entre os agentes seguem o fluxo orquestrado ilustrado na Figura 2.

O processo inicia com a consulta do usuário, que é passada como estado inicial para o grafo. O agente `Planner` analisa a consulta do usuário e cria uma lista de sub-consultas independentes que serão utilizadas durante o processo de recuperação. O agente `Router` determina o domínio da consulta, guiando o processo de recuperação realizado pelos agentes `Librarian` e `Extractor`. As evidencias recuperadas são analisadas pelo agente `Validator`, que determina se as mesmas são relevantes ou não, podendo direcionar o fluxo para o `Router` realizar outra busca em um domínio diferente. O agente `StepDefiner` atua analisando as evidencias recuperadas, a consulta do usuário e a próxima consulta da lista, podendo alterar a próxima consulta com base nas evidências ou encerrar o processo caso as evidencias já sejam suficientes para responder a consulta do usuário. Por ultimo, o agente `Editor` edita uma resposta final com base nas evidências recuperadas e na pergunta original do usuário, encerrando o processo.

[1] Conexão de múltiplas peças de evidência espalhadas por diferentes fontes para construir uma resposta.

[2] Consultas diretas, onde a resposta pode ser encontrada em um único documento ou fonte de dados sem a necessidade de relacionar informações externas.

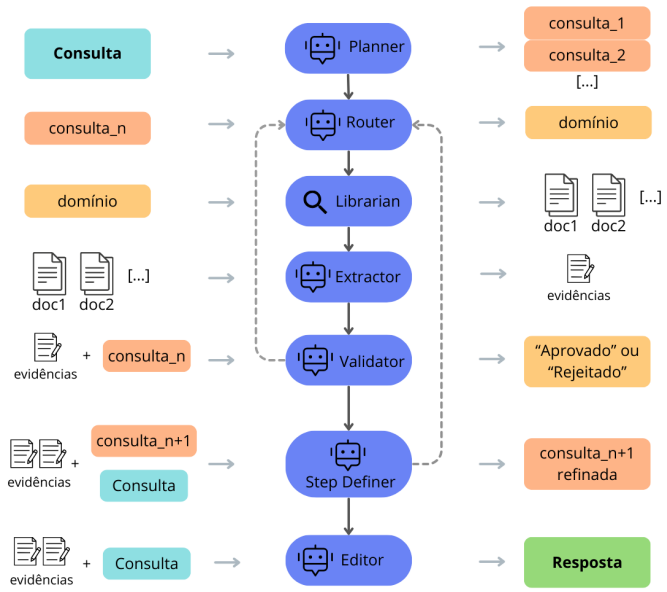

Figura 2. Visão geral da arquitetura. A imagem é composta pelos agentes, o fluxo de interação e o fluxo de entrada e saída de dados. Os fluxos de iteração são identificados pelas linhas tracejadas.

4 Desenvolvimento do Sistema

Esta seção detalha os aspectos técnicos da implementação do protótipo, incluindo a escolha do *framework*, o desenvolvimento da solução e o desenvolvimento em detalhes de cada agente.

4.1 Escolha do *Framework*

Para implementar a solução, utilizou-se o conjunto LangChain[3] + LangGraph[4]. Diferente de *frameworks* como o CrewAI[5], que foca em processos lineares, ou o AutoGen[Wu *et al.*, 2024], orientado a conversações abertas, a escolha pelo LangGraph justifica-se pela necessidade de um controle rigoroso sobre o fluxo de iteração. Além disso, a gestão explícita do *GraphState* possibilita que múltiplos agentes compartilhem uma memória de curto prazo estruturada.

O armazenamento vetorial ficou a cargo do ChromaDB[6], cuja persistência nativa elimina dependências de bancos externos. Para gerar *embeddings*, adotaram-se os modelos da MistralAI[7], que oferecem alto desempenho semântico sem custo com baixas quantidades de uso. Por fim, a inferência do modelo de linguagem roda na infraestrutura da Groq[8], garantindo baixa latência e escalabilidade econômica.

4.2 Desenvolvimento da Solução

Durante o desenvolvimento, a gerência dos *vector stores* foi centralizada na classe `VectorStoreManager`, a qual carrega em lote os PDFs por meio de `PyPDFDirectoryLoader`, divide o texto em blocos de 1000 caracteres com sobreposição de 150 para preservar o contexto semântico entre os fragmentos. Os vetores são armazenados em coleções isoladas para cada domínio usando o ChromaDB. A classe `VectorStoreManager` expõe o método `get_retriever(k)` para parametrizar o número de documentos retornados.

A lógica de execução multi-agente foi montada como um grafo de estados por meio da classe `StateGraph`. O estado compartilhado (`GraphState`) atua como a memória de curto prazo do sistema, armazenando a consulta do usuário, a lista de consultas geradas pelo *Planner*, a consulta atual, o domínio definido para a consulta atual, a lista de domínios proibidos, as evidências coletadas e outras informações necessárias para a lógica de execução. O fluxo inicia no agente *Planner*, conectando sequencialmente os nós de roteamento, recuperação e extração. O sistema estabelece arestas condicionais para os agentes *Validator* e *StepDefiner*, que podem redirecionar o fluxo de trabalho. O comportamento especializado dos agentes é atingindo por meio de *chain-of-thought prompting* e *few-shot prompting* [Wei *et al.*, 2022].

5 Experimentos

Nesta seção são apresentados os experimentos realizados, incluindo a descrição dos documentos que foram utilizados, as perguntas realizadas ao sistema, detalhes de implementação, análise dos resultados obtidos e discussão dos resultados.

5.1 Documentos

Para validar a arquitetura proposta, o sistema foi avaliado com um conjunto de documentos institucionais públicos, que compõem a base de conhecimento do sistema. Esses documentos compreendem a parceria entre a Universidade Federal do Ceará (UFC) com a Empresa Brasileira de Serviços Hospitalares (EBSERH) para a gestão dos hospitais universitários. Os documentos escolhidos são listados na Tabela 2.

Esses três documentos foram escolhidos pois compreendem à domínios diferentes, com tipologias textuais distintas e informações relacionadas, permitindo que o sistema navegue por diferentes domínios e recupere informações de textos estruturados em diferentes tipologias textuais. Todos os documentos estão disponíveis online[9][10][11].

5.2 Perguntas

Para testar a capacidade do sistema em responder perguntas simples e complexas, foram definidas seis perguntas de teste para avaliar o sistema em consultas *single-hop* e *multi-hop*. As perguntas estão descritas na Tabela 3.

5.3 Detalhes da Implementação

Para a execução desses experimentos foram escolhidos os modelos: `gpt-oss-120b` para todos os agentes e `mistral-embed` para criação dos *embeddings*.

5.4 Resultados

Nos experimentos realizados, foi visto que o agente `Planner` identificou em 100% a complexidade das perguntas, criando múltiplas consultas para as perguntas de complexidade *multi-hop* e uma consulta para consultas *single-hop*. Ao todo, dez

[3] `https://www.langchain.com`
[4] `https://www.langchain.com/langgraph`
[5] `https://docs.crewai.com/`
[6] `https://www.trychroma.com`
[7] `https://www.mistral.ai`
[8] `https://www.groq.com`

[9] `https://www.gov.br/ebserh/pt-br/hospitais-universitarios/contratos-de-gestao/regiao-nordeste/huwc-ufc/contrato-de-gestao-especial`
[10] `https://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12550.htm`
[11] `https://pdi.ufc.br/wp-content/uploads/2025/01/pdi-2023-2027-3a-revisao-16.12.2024.pdf`

Tabela 2. Descrição dos Documentos

Domínio	Documento	Descrição
Estratégico	Plano de Desenvolvimento Institucional (PDI UFC 2023-2027)	Documento de planejamento que define a missão, visão, metas de longo prazo e indicadores de desempenho da Universidade Federal do Ceará. Serve como base para identificar os objetivos institucionais.
Operacional	Regulamento de Pessoal da EBSERH	Normativa interna que estabelece as regras de contratação, direitos, deveres e regime de trabalho (CLT) dos funcionários que atuam no Complexo Hospitalar da UFC.
Legislativo	Lei Federal nº 12.550/2011	Legislação federal que autoriza a criação da EBSERH. Define a natureza jurídica da empresa pública e estabelece as bases legais para a gestão dos hospitais universitários federais.

Tabela 3. Consultas Utilizadas para Avaliação do Sistema

ID	Pergunta	Complexidade
Q1	Qual é a finalidade principal da EBSERH?	Single-hop
Q2	Quais são as três funções que a EBSERH deve desempenhar no Hospital Universitário da UFC?	Single-hop
Q3	Quais são as responsabilidades da UFC durante o período de transição até a EBSERH assumir a gestão plena do hospital?	Single-hop
Q4	Quais foram as três fases utilizadas na elaboração do PDI 2023–2027 e em que anos ocorreram suas etapas principais?	Multi-hop
Q5	Como o objetivo do Contrato de Gestão Especial entre a UFC e a EBSERH se relaciona com as competências legais atribuídas à EBSERH?	Multi-hop
Q6	Quais são os órgãos que compõem a estrutura administrativa da EBSERH e qual é o regime de pessoal adotado pela empresa?	Multi-hop

consultas foram criadas pelo `Planner`, com cada consulta *single-hop* (**Q1**, **Q2** e **Q3**) contabilizando por uma consulta cada, duas consultas geradas para **Q4**, duas consultas para **Q6** e três consultas para **Q5**.

O agente `Router` realizou oito decisões de roteamento, contabilizando apenas um erro, provendo uma taxa de acerto de 85,5%. No caso em que houve um erro no roteamento, a lista de domínios proibidos evitou que o agente escolhesse novamente o mesmo domínio e permitiu um roteamento correto logo em seguida.

Em duas instâncias, o `StepDefiner` encerrou o processo prematuramente em **Q4** e **Q6**, julgando com suficiente as evidências recuperadas no primeiro ciclo de recuperação. Foi avaliado que esses dois casos não comprometeram a qualidade geral da resposta, com o agente evitando a realização de um ciclo desnecessário de processamento.

É importante ressaltar que a arquitetura proposta esta associada com um gasto significativo de *tokens*, como é possivel ver na Tabela 4. Cada invocação de agente está atrelada a um número de *tokens* de entrada e resulta em um número de *tokens* de saída, totalizando um custo significativo de *tokens* em comparação com outros sistemas de RAG simplificados. Esse comportamento pode ser evidenciado nas consultas *multi-hop*, que necessitam da execução de múltiplos ciclos de recuperação, analise e raciocínio.

Tabela 4. Consumo de Tokens e Tempo de Execução por Questão

ID	Input Tokens	Output Tokens	Total Tokens
Q1	2.907	615	3.522
Q2	3.361	1.301	4.662
Q3	3.847	1.406	5.253
Q4	3.442	1.295	4.737
Q5	11.012	3.773	14.785
Q6	2.798	1.596	4.394
Média	**4.561**	**1.664**	**6.225**

Tabela 5. Tempo de execução por questão

ID	Tempo (s)
Q1	10.6
Q2	7.2
Q3	7.5
Q4	6.3
Q5	80.2
Q6	6.1
Média	19.65

A característica multi-agente, juntamente com os ciclos de iteração, são responsáveis por significativos custos de tempo, como é possível ver na Tabela 5. As consultas **Q4** e

Q6 obtiveram resultado similar as consultas simples devido a característica do sistema de encerrar o ciclo a partir da analise das informações obtidas. A consulta **Q5** apresentou maior tempo de execução, isso é devido a erros no roteamento, que levaram a necessidade de um novo ciclo de recuperação.

Os *logs* de execução para cada pergunta realizada estão disponíveis em um documento online[12]. Neles, é possível identificar a sequência de execução dos agentes, as consultas geradas e executadas, os domínios selecionados, as respostas obtidas e o uso geral de *tokens* para cada pergunta.

5.5 Discussão

Os resultados apresentados evidenciam comportamento promissor. Os agentes realizaram suas tarefas individuais de forma especializada e eficiente, em especial os agentes `Planner` e `StepDefiner`, que possuem papeis chaves na arquitetura. Esses dois agentes determinaram o fluxo de trabalho a ser realizado de forma eficiente, evitando a realização de ciclos de trabalho desnecessários, reduzindo custos de processamento e latência. O roteamento das consultas apresentou comportamento adequado, mas que precisa de mais atenção. Um roteamento errado acarreta um ciclo de trabalho potencialmente perdido, levando a maiores custos e latência.

Ao comparar esta abordagem com o paradigma RAG ingênuo [Lewis *et al*., 2020], observa-se que a decomposição de consultas complexas mitigou a incapacidade do sistema de sintetizar informações fragmentadas. Enquanto o trabalho de [Nguyen *et al*., 2025] utiliza cadeias de raciocínio colaborativas, a arquitetura proposta diferencia-se pela inclusão de um agente `Validador` e uma lista de domínios proibidos. Essa configuração permitiu uma orquestração dinâmica com correção de rotas de recuperação em tempo real, atingindo a precisão observada nos experimentos, mesmo em casos onde o roteamento inicial falharia.

A característica multi-agente orientada a LLMs apresenta pontos negativos no que diz respeito a latência e uso de *tokens* quando comparado a arquiteturas de RAG tradicionais, algo que deve ser levado em consideração ao aplicar essa arquitetura em cenários reais. Esses pontos estão intrinsecamente relacionados aos tamanhos dos modelos usados nos agentes. Estudos realizados em arquitetura similares indicam que a utilização de modelos menores em alguns agentes tem impacto mínimo na qualidade geral da arquitetura, podendo ser um técnica viável para redução da latência e uso de *tokens* [Nguyen *et al*., 2025].

6 Conclusão

Este artigo apresentou uma arquitetura baseada em um sistema multiagente (SMA) e Geração de Recuperação Aumentada (RAG) para a recuperação de conhecimento organizacional. A abordagem proposta visou superar desafios como a fragmentação de documentos e a sobrecarga informacional em ambientes corporativos, promovendo escalabilidade e eficiência.

O protótipo funcional, orquestrado por um grafo de agentes especializados, demonstrou a viabilidade da solução especialmente em relação à mitigação de alucinações e erros de roteamento. Os resultados dos experimentos realizados indicam que o sistema é capaz de responder perguntas simples e complexas, por meio da análise e decomposição da pergunta do usuário. O sistema também foi capaz de rotear consultas para domínios de conhecimento específicos e gerar respostas precisas e contextualizadas, fundamentadas em evidências documentais. A implementação de agentes inteligentes permitiu ao sistema decidir dinamicamente o fluxo de trabalho de forma autônoma com base nas informações recuperadas.

[12] `https://github.com/matheuskid/multi-agent-rag-km/blob/main/Logs.pdf`

Declarações complementares

Contribuições dos autores

Todos os autores contribuíram igualmente na concepção desse estudo. Todos os autores leram e aprovaram o manuscrito final.

Conflitos de interesse

Os autores declaram que não têm nenhum conflito de interesses.

Disponibilidade de dados e materiais

Os conjuntos de dados e softwares utilizados durante o estudo atual estão disponíveis em `https://github.com/matheuskid/multi-agent-rag-km`.

Referências

Ackoff, R. L. (1989). From data to wisdom. *Journal of applied systems analysis*, 16(1):3–9. Disponível em: `https://scholar.google.com/scholar?cluster=6242114617204124477`.

Davenport, T. H. and Prusak, L. (2002). *Conhecimento empresarial: como as organizações gerenciam o seu capital intelectual*. Campus, Rio de Janeiro, 6 edition.

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., and Schmidhuber, J. (2024). MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. Disponível em: `https://arxiv.org/abs/2308.00352`.

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., *et al*. (2025). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55. DOI: 10.1145/3703155.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., *et al*. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474. Disponível em: `https://scholar.google.com/scholar?cluster=10679876450978666441`.

Li, H., Su, Y., Cai, D., Wang, Y., and Liu, L. (2022). A Survey on Retrieval-Augmented Text Generation. Disponível em: `https://arxiv.org/abs/2202.01110`.

Maldonado, D., Cruz, E., Torres, J. A., Cruz, P. J., and Benitez, S. d. P. G. (2024). Multi-agent systems: A survey about its components, framework and workflow. *IEEE Access*, 12:80950–80975. DOI: 10.1109/ACCESS.2024.3409051.

Nguyen, T., Chin, P., and Tai, Y.-W. (2025). Ma-rag: Multi-agent retrieval-augmented generation via collabo-

rative chain-of-thought reasoning. *arXiv preprint arXiv:2505.20096*. DOI: 10.48550/arXiv.2505.20096.

Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., Xu, J., Li, D., Liu, Z., and Sun, M. (2024). ChatDev: Communicative agents for software development. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics. DOI: 10.18653/v1/2024.acl-long.810.

Silva, E., Santos, F., Thompson, P., and Reis, J. (2025). Llm-powered conversational multi-agent cognitive system for collaborative task solving. In *Anais do XIX Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações*, pages 59–70, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/wesaac.2025.37528.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., *et al*. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837. Disponível em: `https://scholar.google.com/scholar?cluster=4478103128423899805`.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., *et al*. (2024). Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*. Disponível em: `https://openreview.net/forum?id=BAakY1hNKS`.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. (2018). HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380. DOI: 10.18653/v1/D18-1259.

Yao, Y., Zeng, Y., Zhong, N., and Huang, X. (2007). Knowledge Retrieval (KR). In *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, pages 729–735. IEEE. DOI: 10.1109/WI.2007.113.