RAG Multiagente para Recuperação de Conhecimento Organizacional

Guilherme L. Moretti[1], **Kalmax dos S. Sousa**[1], **Matheus dos S. Mendes**[1], **Marcos A. de Oliveira**[1]

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

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 contextualized 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.*

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.

A adoção de sistemas multiagentes (SMA) tem ganhado destaque em aplicações que requerem processamento distribuído e adaptativo, como robótica colaborativa, logística inteligente e gestão de dados corporativos [Wooldridge 2009]. 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 é uma área promissora. Portanto, o objetivo deste artigo é desenvolver e avaliar um SMA com arquitetura baseada em RAG para a recuperação de conhecimento organizacional. Especificamente, buscamos: (i) projetar uma arquitetura de agentes autônomos e especializados; (ii) implementar um protótipo funcional orquestrado por um grafo de estados; e (iii) avaliar o seu desempenho através de um experimento controlado.

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. O sucesso destes sistemas valida a hipótese de que a especialização de agentes e a definição de um fluxo de trabalho estruturado 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 nossa 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 incluindo um agente validador explícito como um ponto de controle de qualidade no *pipeline*. Esta abordagem traduz os princípios do RAG Avançado num sistema multiagente prático e avaliável.

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)
Proposta deste artigo	**Multiagente**	**Processo (RAG)**	**Grafo de estados**	**Sim (agente validador)**

3. Metodologia

Esta seção descreve a concepção conceitual do sistema proposto, focando na arquitetura multiagente e na definição funcional de cada componente.

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 (processos, produtos e humanos);
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;

Para atender a tais requisitos, foi adotada uma estratégia híbrida que combina armazenamento vetorial, recuperação semântica, validação de relevância e geração de linguagem natural, cada qual delegado a um agente especializado.

3.2. Arquitetura Multiagente

O sistema é modelado como um conjunto de agentes computacionais autônomos e especializados, e não como um *pipeline* sequencial rígido. Cada agente encapsula uma lógica de processamento específica e opera sobre um estado compartilhado, focando em sua própria responsabilidade. A colaboração é governada por um grafo de estados, onde a transição entre agentes pode ser condicional. Por exemplo, o agente `Validator` pode decidir encerrar o processo prematuramente se a informação recuperada for de baixa qualidade. Esta capacidade de alterar o fluxo de execução com base em resultados intermediários caracteriza um comportamento colaborativo e dinâmico, alinhado aos princípios de SMA, e diferencia a abordagem de um simples *pipeline*.

3.3. Definição dos Agentes

O sistema foi modelado como um grafo direcionado acíclico, no qual cada nó representa um agente autônomo. Os papéis são definidos da seguinte forma:

- ***Similarity Router*:** Determina o domínio mais relevante para a consulta.
- ***Librarian*:** Recupera os trechos de texto mais pertinentes do domínio selecionado.
- ***Validator*:** Verifica se as evidências recuperadas são de fato relevantes para responder à pergunta, atuando como um filtro de qualidade.
- ***Extractor*:** Isola as evidências citáveis dos trechos validados, por exemplo, através de *few-shot prompting*.
- ***Editor*:** Redige a resposta final, integrando a pergunta do usuário e as evidências extraídas em um texto coeso e em linguagem natural.

As interações entre os agentes seguem o fluxo orquestrado ilustrado na Figura 1. O processo inicia com a consulta do usuário, que é passada como estado inicial para o grafo. O `Similarity Router` é o ponto de entrada, que determina o caminho a ser seguido. A informação flui sequencialmente até o `Validator`, onde uma aresta condicional determina o desfecho do processo.

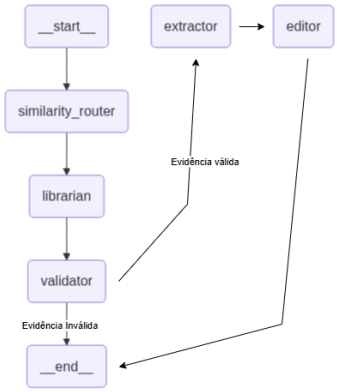

Figura 1. Arquitetura e fluxo de interação entre agentes

4. Desenvolvimento do Sistema

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

4.1. Escolha do *Framework*

A pilha tecnológica foi selecionada com base em critérios de modularidade, comunidade ativa e compatibilidade com LLMs.

Para implementar a solução, utilizou-se o conjunto LangChain[1] + LangGraph[2], que simplifica o encadeamento de prompts, a composição de fluxos multiagente e o monitoramento do estado global por meio do GraphState. O armazenamento vetorial ficou a cargo do ChromaDB[3], cuja persistência nativa elimina dependências de bancos externos. Para gerar *embeddings*, adotaram-se os modelos da MistralAI[4], que oferecem alto

[1] `https://www.langchain.com`
[2] `https://www.langchain.com/langgraph`
[3] `https://www.trychroma.com`
[4] `https://www.mistral.ai`

desempenho semântico sem custo com baixas quantidades de uso. Por fim, a inferência do modelo de linguagem roda na infraestrutura da Groq[5], garantindo baixa latência e escalabilidade econômica.

Essa combinação reduz a complexidade operacional e favorece a experimentação rápida de novos agentes ou domínios.

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, cria ou atualiza a coleção correspondente no ChromaDB e expõe o método `get_retriever(k)` para parametrizar o número de documentos retornados. Em seguida, o grafo de execução é montado por meio de `StateGraph(GraphState)`, definindo o *similarity router* como ponto de entrada, conectando sequencialmente os demais nós e estabelecendo arestas condicionais, em especial aquela que, após a etapa de validação, decide se o fluxo é encerrado ou se prossegue para o *extractor* e o *editor*.

Quando o usuário submete uma `query`, o estado inicial percorre automaticamente esse grafo: primeiro pelo *router*, depois pelo *librarian* e pelo *validator*, caso não haja resposta final após a validação, o fluxo avança para o *extractor* e o *editor*, retornando por fim uma resposta redigida com referências explícitas às evidências recuperadas. Dessa forma, o processo completo: da ingestão de documentos à geração de respostas permanece modular, rastreável e fiel à sequência ilustrada no diagrama.

5. Resultados Preliminares

Para uma validação inicial da arquitetura proposta, o sistema foi avaliado em um cenário controlado com um conjunto de documentos institucionais genéricos, abrangendo os domínios de Recursos Humanos (RH), Processos e Produtos. O objetivo dos testes foi observar a capacidade do sistema de rotear corretamente as consultas, recuperar evidências pertinentes e gerar respostas coesas.

Nos experimentos realizados, o sistema demonstrou um comportamento promissor. O agente *similarity_router* foi capaz de direcionar as perguntas para os respectivos domínios com boa precisão. Por exemplo, consultas sobre políticas de férias foram corretamente encaminhadas para a base de conhecimento de RH, enquanto dúvidas sobre especificações técnicas foram direcionadas ao domínio de Produtos.

O fluxo de recuperação e validação também se mostrou eficaz. O agente *librarian* recuperou trechos relevantes e o *validator* conseguiu confirmar a pertinência do contexto antes da geração da resposta. Como resultado, o agente editor produziu respostas finais que integravam as evidências extraídas dos documentos, respondendo corretamente às perguntas formuladas durante os testes. Embora a amostragem de testes tenha sido limitada, os resultados indicam que a arquitetura multiagente orquestrada por um grafo é funcional e atinge os objetivos de modularidade e precisão propostos.

[5] `https://www.groq.com`

6. Conclusão

Este artigo apresentou uma arquitetura inovadora 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 desenvolvimento de um protótipo funcional, orquestrado por um grafo de agentes especializados, demonstrou a viabilidade da solução. Os resultados preliminares, embora não exaustivos, indicam que o sistema é capaz de rotear consultas para domínios de conhecimento específicos e gerar respostas precisas e contextualizadas, fundamentadas em evidências documentais. A modularidade inerente à arquitetura de agentes se mostrou uma opção promissora para sanar a natureza estática de sistemas RAG tradicionais.

Como trabalhos futuros, pretende-se realizar uma avaliação de desempenho quantitativa, medindo métricas como tempo de resposta e a precisão em um volume maior e mais heterogêneo de documentos. Além disso, planeja-se expandir a complexidade do grafo, adicionando novos agentes com capacidades analíticas, como a sumarização de múltiplos documentos e a identificação de tendências, a fim de atender demandas estratégicas de forma ainda mais robusta.

Referências

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.

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.

Li, H., Su, Y., Cai, D., Wang, Y., and Liu, L. (2022). A survey on retrieval-augmented text generation.

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.

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

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.