A Multi-Agent Approach to Addressing RAG Limitations for Complex Query Processing and Scientific Research Screening

Abstract

In this paper, I present a novel multi-agent system designed to overcome the inherent limitations of traditional Retrieval Augmented Generation (RAG) approaches. By decomposing complex information retrieval and synthesis problems into specialized agent roles, my implementation demonstrates superior capabilities in handling multi-step reasoning, qualitative assessment, and complex query processing across large document collections. The system particularly excels at scientific literature screening and synthesis, enabling researchers to efficiently process and analyze large volumes of academic papers. Through experimental validation and comparative analysis, I demonstrate how this multi-agent architecture provides a more robust framework for knowledge-intensive tasks that have historically challenged conventional RAG systems.

1. Introduction

1.1 Problem Statement

Traditional Retrieval Augmented Generation (RAG) applications have demonstrated significant utility in information retrieval tasks but continue to face fundamental limitations when processing certain types of queries. These limitations include:

- Difficulty with multi-step reasoning where results from previous steps inform subsequent retrievals
- Inadequate handling of general, open-ended queries
- Poor performance with long, complex queries despite these being natural strengths of Large Language Models (LLMs)
- Inability to address implicit queries requiring holistic context understanding
- Limited capabilities for document collection analysis (e.g., categorizing or counting documents by topic)
- Inefficiency in screening and vetting large volumes of scientific literature

These limitations significantly impact researchers who must process ever-increasing volumes of academic papers when conducting literature reviews. Current tools lack the capabilities to effectively screen, categorize, and synthesize information across large document collections based on complex criteria.

1.2 Research Objectives

My research addresses these limitations through the following objectives:

1. Design and implement a multi-agent system that decomposes complex RAG problems into specialized roles
2. Create an effective document summarization pipeline that preserves key information while enabling semantic comparison
3. Develop robust query analysis capabilities that match research questions to relevant documents through semantic similarity
4. Enable synthesis of information across multiple documents to address complex queries
5. Establish a framework for scientific literature screening that assists researchers in efficiently identifying relevant papers

1.3 Approach Overview

My solution decomposes the problem using specialized agents with distinct responsibilities:

1. **Document Summary Agent**: Creates comprehensive summaries of documents that capture key concepts, methodologies, and findings
2. **Query Analysis Agent**: Analyzes user queries and identifies relevant documents through semantic matching
3. **Document Analysis Agent**: Synthesizes information from multiple documents to generate comprehensive answers
4. **Scientific Document Summary Agent**: Creates structured scientific paper summaries with formatted metadata
5. **Report Agent**: Formats structured summaries into well-organized reports

This approach enables handling complex queries that traditional RAG systems struggle with by distributing cognitive load across specialized agents and establishing clear information pathways between them.

2. Related Work and Inspiration

My implementation draws inspiration from several key papers and research directions in the field:

- **ReAct**: Reasoning and Acting in Language Models (Yao et al., 2023)
- **Chain-of-Thought Prompting**: Step-by-step reasoning in language models (Wei et al., 2022)
- **Multi-Agent Debate**: Collaborative problem-solving through simulated debate (Du et al., 2023)
- **LangChain's Agent Framework**: Modular approach to agent-based LLM applications
- **CrewAI**: Open-source framework for orchestrating role-playing autonomous AI agents
- **Semantic Chunking Approaches**: Advanced document segmentation techniques that respect semantic boundaries

These works provided the theoretical foundation for my agent-based architecture and informed the design decisions throughout implementation.

3. Methodology

3.1 System Architecture

I implemented a FASTAPI backend that coordinates two primary workflows:

1. **Document Analysis Crew**: Answers complex queries across multiple documents
    - Document Summary Agent → Query Analysis Agent → Document Analysis Agent
2. **Document Summary Crew**: Creates detailed scientific paper summaries
    - Scientific Document Summary Agent → Report Agent

The system uses background tasks for asynchronous processing and provides a robust event logging mechanism to track progress. All components operate via the CrewAI framework, which facilitates agent communication and task execution.

The architecture is designed to be frontend agnostic, allowing integration with various user interfaces. My current implementation includes a NextJS frontend that provides an intuitive interface for document upload, query submission, and result visualization. This separation of concerns allows for flexible deployment options while maintaining a consistent backend processing pipeline.

3.2 Document Processing Pipeline

My document processing pipeline follows these key steps:

1. **Document Loading**: Support for multiple formats (PDF, TXT, DOCX) using LangChain community loaders
2. **Text Cleaning**: Normalization and removal of unwanted characters and formatting
3. **Semantic Chunking**: Division of documents respecting semantic boundaries rather than arbitrary character counts
4. **Summarization**: Generation of comprehensive summaries capturing key information
5. **Embedding Generation**: Creation of vector representations for summaries
6. **Storage**: Persistence of summaries and embeddings for retrieval

I implemented semantic chunking using the SemanticChunker from LangChain experimental, which produces more coherent text segments than traditional character-based approaches.

3.3 Query Analysis and Document Matching

For query analysis and document matching, I implemented:

1. **Query Embedding**: Generation of vector representations for user queries
2. **Cosine Similarity Calculation**: Comparison of query and document summary embeddings

3. **Adaptive Threshold Mechanism**: Adjustment of similarity thresholds based on query complexity

Through extensive experimentation, I determined that a base threshold of 0.76 provided optimal performance in identifying relevant papers while excluding irrelevant ones. This threshold is dynamically adjusted upward for longer queries (>20 words) to accommodate their increased complexity.

3.4 Document Synthesis

The document synthesis process integrates information across documents through:

1. **Relevant Document Identification**: Selection of documents exceeding the similarity threshold
2. **Multi-Document Integration**: Synthesis of information across selected documents
3. **Comprehensive Response Generation**: Creation of detailed, structured responses addressing the original query

3.5 Scientific Paper Analysis

For scientific paper analysis, I implemented:

1. **Structured Summary Generation**: Creation of comprehensive research paper summaries
2. **Metadata Extraction**: Identification of key paper components (authors, methodology, findings, etc.)
3. **Report Generation**: Formatting of findings into well-structured reports

4. Implementation Details

4.1 Agent Implementation

Each agent is implemented with specific roles, goals, and backstories:

- **Document Summary Agent**: Expert in comprehensive content analysis and summary generation
- **Query Analysis Agent**: Specialist in understanding and categorizing complex queries
- **Document Analysis Agent**: Expert in integrating and synthesizing insights across documents
- **Scientific Document Summary Agent**: Specialist in structured research paper analysis
- **Report Agent**: Expert in transforming structured data into readable reports

4.2 Tool Integration

Custom tools were developed to extend agent capabilities:

- **DocumentSynthesisTool**: For analyzing and synthesizing document content
- **QueryDocumentAnalysisTool**: For matching queries to relevant documents
- **DocSummaryTool**: For generating document summaries
- **SciDocSummaryTool**: For structured scientific paper analysis

4.3 Embedding and Similarity Implementation

For embedding generation and similarity calculation, I utilized:

- **Azure OpenAI text-embedding-ada-002**: For high-quality vector representations
- **Scikit-learn cosine similarity**: For efficient vector comparison
- **NumPy**: For vector normalization and mathematical operations

My choice of the closed-source text-embedding-ada-002 model was deliberate and resulted in significantly higher accuracy compared to older open-source embedding models. The quality of these embeddings proved crucial for correctly identifying relevant papers based on semantic similarity.

4.4 Technical Approach Comparisons

In developing this multi-agent RAG system, I evaluated several technical approaches before arriving at the final implementation. My initial exploration was heavily influenced by the "5 Levels of Summarization" technique framework from Greg Kamradt, which provided valuable insights into handling documents of varying lengths through increasingly sophisticated methods.

4.4.1 Document Chunking Strategies

Initially, I experimented with manual chunking approaches using KNN (K-Nearest Neighbors) clustering of document embeddings. This method involved:

- Segmenting documents into overlapping chunks
- Generating embeddings for each chunk
- Using KNN to identify semantically similar chunks
- Creating representative vectors for each cluster

While effective, I found that LangChain's `SemanticChunker` module provided better results with significantly less implementation complexity. The `SemanticChunker` automatically identifies semantic boundaries in text, resulting in more coherent chunks that preserve contextual meaning. This approach proved especially valuable for scientific papers where maintaining the integrity of conceptual units is critical.

4.4.2 Vector Database Selection

For vector storage and retrieval, I carefully evaluated four leading options:

1. **FAISS (Facebook AI Similarity Search)**: An open-source library optimized for efficient similarity search and clustering of dense vectors
2. **Pinecone**: A managed vector database service with advanced filtering and metadata capabilities
3. **Weaviate**: An open-source vector search engine with schema-based organization
4. **Azure AI Search:** A cloud search service with built-in AI capabilities that offers vector search functionality alongside full-text search, cognitive search, and knowledge mining features with deep enrichment features on vector indexes

After extensive testing, I selected FAISS for the following reasons:

- **Performance**: FAISS is opensource, easy to set up and by default runs in an in-memory configuration
- **Flexibility**: The ability to toggle between in-memory operation and persistence gave the system adaptability for different deployment scenarios
- **Local Processing**: Unlike cloud-based alternatives, FAISS can operate entirely locally, reducing latency and external dependencies
- **Resource Efficiency**: FAISS performs well even without GPU acceleration, making it suitable for deployment in various environments

My selection of FAISS primarily addressed the immediate research needs for a flexible, open-source solution that could run locally during development and testing phases while maintaining the option to transition to Azure AI Search for production deployments.

4.4.3 Similarity Calculation Implementation

For calculating semantic similarity between queries and document summaries, I compared several approaches:

1. **Direct LLM Evaluation**: Using LLMs to directly compare and rate relevance
2. **Cosine Similarity**: Vector-based similarity using dot product normalized by vector magnitudes
3. **Euclidean Distance**: Spatial distance between embedding vectors
4. **Hybrid Approaches**: Combining vector similarity with LLM verification

The coded implementation of cosine similarity proved most effective, offering:

- **Computational Efficiency**: Direct vector operations are significantly faster than LLM API calls and are free when testing as they are performed locally
- **Consistency**: Results were more reproducible and less subject to LLM output variations
- **Fine-grained Control**: The ability to precisely tune thresholds based on experimental results

My systematic experimentation led to the selection of 0.76 as the optimal base similarity threshold, with dynamic adjustments for longer queries.

4.4.4 Embedding Model Selection

For generating vector representations, I evaluated both open-source and closed-source embedding models:

1. **Open-Source Options**: Models like BERT, MPNet, and other transformer-based encoders
2. **Closed-Source Options**: OpenAI's text-embedding-ada-002 and similar commercial models

Despite the appeal of open-source alternatives, I found that Azure OpenAI's text-embedding-ada-002 model consistently produced higher quality embeddings that more accurately captured semantic relationships between documents and queries. This quality difference directly translated to more precise document matching, which was critical for the research screening use case.

These technical choices reflect a balanced approach that prioritizes performance, accuracy, and implementation feasibility while maintaining the flexibility to adapt as the underlying technologies evolve.

5. Experimental Results

5.1 Threshold Determination

I conducted extensive experimentation to determine the optimal similarity threshold for document matching. Through testing numerous papers against diverse queries, I established that a base threshold of 0.76 provided the best balance between:

- Including all semantically relevant papers
- Excluding papers that were tangentially related but not directly relevant
- Accommodating different query formulations for the same information need

For queries exceeding 20 words, I found that increasing the threshold by 0.03 improved precision without significantly impacting recall.

5.2 Performance Analysis

The multi-agent approach demonstrated superior performance compared to traditional RAG in several key areas:

- **Multi-step Reasoning**: Successfully handled queries requiring sequential information processing
- **General Queries**: Effectively matched broad, conceptual queries to relevant documents
- **Complex Queries**: Processed lengthy, multi-faceted queries with high accuracy

- **Document Collection Analysis**: Correctly identified document subsets matching specific criteria

5.3 Comparative Analysis

Comparing my implementation to traditional RAG approaches revealed:

- **Higher Recall**: Retrieved more relevant documents for complex queries
- **Improved Synthesis**: Generated more comprehensive and coherent responses
- **Better Context Preservation**: Maintained important relationships between concepts
- **Enhanced Document Classification**: More accurately categorized papers by relevance

6. Discussion

6.1 Strengths of the Multi-Agent Approach

The multi-agent architecture successfully addresses the inherent limitations of traditional RAG systems:

1. **Specialized Cognitive Tasks**: Each agent focuses on a specific aspect of the problem, allowing for deeper expertise
2. **Sequential Processing**: Enables multi-step reasoning by passing intermediate results between agents
3. **Information Integration**: Combines insights across documents in ways that simple retrieval cannot
4. **Threshold Adaptability**: Adjusts matching criteria based on query complexity
5. **Comprehensive Document Analysis**: Captures and utilizes document-level context that chunk-based approaches miss

6.2 Limitations and Challenges

Despite its strengths, my implementation faces several limitations:

1. **Computational Overhead**: Multiple agent interactions increase processing time and resource requirements
2. **Embedding Quality Dependence**: System performance is highly dependent on embedding model quality
3. **Threshold Sensitivity**: Results can vary significantly based on similarity threshold settings
4. **Token Usage**: Comprehensive document summarization requires substantial token processing
5. **Error Propagation**: Mistakes in early processing stages can impact downstream results

6.3 Comparative Strategies

Throughout development, I evaluated several alternative approaches:

1. **LLM-based vs. Coded Similarity**: Using direct vector operations proved more efficient and controllable than LLM-generated similarity assessments
2. **Semantic vs. Token-based Chunking**: Semantic chunking preserved context better but required more computation
3. **Traditional Libraries vs. LangChain Components**: Scikit-learn/pandas/numpy offered more control but LangChain provided superior integration

7. Future Work

Several promising directions for future enhancement include:

7.1 Technical Improvements

1. **Advanced Embedding Management**: Implementing local vector database persistence for faster repeat queries
2. **Enhanced Query Preprocessing**: Adding query expansion techniques to address vocabulary mismatch
3. **Improved Document Processing**: Incorporating OCR and better handling of complex document layouts
4. **Evaluation Mechanisms**: Developing automated quality assessment for system outputs
5. **Scalability Enhancements**: Implementing batch processing for larger document collections

7.2 Functional Extensions

1. **Interactive Query Refinement**: Adding capabilities for users to iteratively refine their queries
2. **Human-in-the-Loop Interactivity**: Implementing feedback mechanisms that allow users to guide the system during document analysis, correct misinterpretations, and provide domain expertise at critical decision points in the processing pipeline
3. **Conversational Agents with Long-Term Memory**: Developing specialized conversational agents with SQL database persistence for long-term memory, enabling continuity across research sessions and the ability to reference previous findings and interactions
4. **Message Queue Implementation**: Replacing the in-memory array for job IDs with a robust message queue system and containerized computation units, allowing the backend to continuously receive requests while processing occurs asynchronously
5. **Collaborative Research Tools**: Enabling multiple researchers to share and build upon analyses
6. **Learning from Feedback**: Incorporating user feedback to improve matching accuracy
7. **Cross-lingual Capabilities**: Extending the system to process documents in multiple languages
8. **Domain-specific Customization**: Adapting the system for specific research domains

7.3 Training and Fine-tuning

1. **Custom Embedding Models**: Training domain-specific embedding models for improved matching
2. **Agent Specialization**: Fine-tuning agents for specific scientific disciplines
3. **Threshold Optimization**: Developing domain-adaptive threshold determination

8. Conclusion

My multi-agent approach effectively addresses the fundamental limitations of traditional RAG systems by decomposing complex information retrieval and synthesis tasks into specialized roles. The implementation demonstrates superior capabilities in handling multi-step reasoning, qualitative assessment, and complex query processing across document collections.

The system's particular strength in scientific literature screening and synthesis provides researchers with a powerful tool to efficiently process and analyze large volumes of academic papers. By combining the strengths of embedding-based semantic matching with the reasoning capabilities of LLMs, this architecture establishes a more robust framework for knowledge-intensive tasks.

I am currently training an LLM to recognize what constitutes a "good" paper based on how well it adheres to queries and standard research practices. This model leverages the rich datasets of summaries and metadata generated by the system to develop sophisticated quality assessment capabilities. By learning from these patterns, the system will be able to automatically identify and prioritize high-quality research that meets both query relevance and academic rigor standards.

Future work will focus on enhancing scalability, improving embedding management, and extending the system's capabilities to support more interactive and collaborative research workflows. As LLM and embedding technologies continue to evolve, this multi-agent framework provides a flexible foundation that can incorporate new advances while maintaining its fundamental architectural advantages.

9. Acknowledgments

I would like to acknowledge the contributions of the open-source communities behind LangChain and CrewAI, whose frameworks provided essential building blocks for this implementation. Additionally, I appreciate the Azure OpenAI team for their high-quality embedding models that proved crucial for accurate document matching.

References

1. Lee, J., Chen, A., Dai, Z., Dua, D., Sachan, D. S., Boratko, M., Luan, Y., Arnold, S. M. R., Perot, V., Dalmia, S., Hu, H., Lin, X., Pasupat, P., Amini, A., Cole, J. R., Riedel, S., Naim, I., Chang, M.-W., & Guu, K. (2024). Can Long-Context Language Models Subsume Retrieval, RAG, SQL, and More? arXiv:2406.13121 [cs]. https://arxiv.org/abs/2406.13121

2. Phan, H., Acharya, A., Meyur, R., Chaturvedi, S., Sharma, S., Parker, M., Nally, D., Jannesari, A., Pazdernik, K., Halappanavar, M., Munikoti, S., & Horawalavithana, S. (2024). Examining Long-Context Large Language Models for Environmental Review Document Comprehension. arXiv:2407.07321 [cs]. https://arxiv.org/abs/2407.07321

3. Li, Z., Li, C., Zhang, M., Mei, Q., & Bendersky, M. (2024). Retrieval Augmented Generation or Long-Context LLMs? A Comprehensive Study and Hybrid Approach. arXiv:2407.16833 [cs]. https://arxiv.org/abs/2407.16833

4. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. In International Conference on Learning Representations (ICLR 2023). https://arxiv.org/abs/2210.03629

5. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Advances in Neural Information Processing Systems 35 (NeurIPS 2022). https://arxiv.org/abs/2201.11903

6. Du, Y., Qian, K., Coenen, A., Zhou, Z., Reif, E., & Wang, A. (2023). Improving Factuality and Reasoning in Language Models through Multiagent Debate. arXiv:2305.14325 [cs]. https://arxiv.org/abs/2305.14325

7. McInnes, L., & Healy, J. (2017). Accelerated Hierarchical Density Clustering. arXiv:1705.07321 [cs]. https://arxiv.org/abs/1705.07321

8. Bu, L., & Oosterlee, C. W. (2020). On high-order schemes for tempered fractional partial differential equations. arXiv:2009.07321 [math]. https://arxiv.org/abs/2009.07321

9. Sun, Y., & Xia, Y. (2024). A Unified Framework for Large-Scale Classification: Error Rate Control and Optimality. arXiv:2504.07321 [stat]. https://arxiv.org/abs/2504.07321

10. Sagar, S., Javed, M., & Doermann, D. S. (2024). Leaf-Based Plant Disease Detection and Explainable AI. arXiv:2404.16833 [cs]. https://arxiv.org/abs/2404.16833

11. de Queiroz, R. J. G. B. (2024). From the Notebooks to the Investigations and Beyond. arXiv:2504.18949 [cs]. https://arxiv.org/abs/2504.18949

12. Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., et al. (2024). The Llama 3 Herd of Models. arXiv:2407.21783 [cs]. https://arxiv.org/abs/2407.21783